IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Logical Transients Diagnosis Reference

Version 6 Release 1

SC33-6324-00

IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Logical Transients Diagnosis Reference

Version 6 Release 1

SC33-6324-00

Note!

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

First Edition (April 1995)

This edition applies to Version 6 Release 1 of IBM VSE/Advanced Functions, which is part of VSE Central Functions, Program Number 5686-066, and to all subsequent releases and modifications until otherwise indicated in new editions.

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.

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, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Programming Interface Information

This publication is intended to help the customer to do diagnosis of VSE/ESA. This publication documents information that is Diagnosis, Modification, or Tuning Information provided by VSE/ESA.

Warning: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Trademarks and Service Marks

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

CICS
ESA/390
IBM
S/370
System/370
VSE/ESA
VTAM

Preface

This publication describes the program logic of the logical transient phases and some $IJBSxxx phases of VSE/Central Functions. The publication supplements the program listings. It is intended primarily for use by persons responsible for providing service for VSE/Central Functions or for changing its program design.

The publication provides the information by logical groups of phases as listed below:

Attention routines
Checkpoint/restart routines
Miscellaneous transient services
Termination routines
Miscellaneous $IJBSxxx support

The publication includes a discussion of available diagnostic aids (in Appendix A) and a message-to-phase cross-reference (in Appendix B).

Related Publications: For a description of the program logic of related system components, you need to refer to additional publications as listed below.

VSE/Central Functions, Diagnosis Reference:

Supervisor, SC33-6323
Error Recovery and Recording Transients, SC33-6326
Initial Program Load and Job Control, SC33-6325
Librarian, SC33-6330
Serviceability Aids, SC33-6327

To efficiently use this publication, you should be familiar with the operational concepts of VSE/Central Functions as described in

VSE/ESA Guide to System Functions, SC33-6611

VSE/ESA System Control Statements, SC33-6613

VSE/ESA System Macros Reference, SC33-6616

VSE/ESA Operation, SC33-6606

and with the functional characteristics of the hardware as described in one of the following publications, whichever applies:

IBM ESA/390 Principles of Operation, GA22-7201

IBM 4300 Processors Principles of Operation, GA22-7070

Titles and abstracts of other related publications are listed in *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001.

Chapter 1. Introduction

This chapter of the manual summarizes the purpose of the logical transients and of some $IJBSxxx phases used by VSE/Central Functions.

Logical Transients

The logical transients, also known as B-transients because their names start with the characters $$B, are loaded into a supervisor area whenever they need to be executed to provide a specific computing service.

The logical-transient phases covered in this manual are described in logical groups as follows:

- Attention routines

  The logical transients making up these routines are called by the attention task of the supervisor whenever an attention routine (AR) command is to be processed. These routines are described in Chapter 2.

- Miscellaneous logical transient service routines

  These are logical transients called by the system to perform specific services such as loading a print control buffer. These service routines are described in Chapter 3.

- Checkpoint/restart routines

  The logical transients making up these routines are called by the system whenever a checkpoint record is to be built for a long-running program; they are also required if, as a result of a failure, the particular program is to be set up for continued operation at one of the recorded checkpoints. These logical transients are described in Chapter 4.

$IJBSxxx Phases

Just like the logical transients, so are the $IJBSxxx phases available to provide specific computing services; they receive control whenever those services are required by other programs.

The $IJBSxxx phases reside in the shared virtual area (SVA) and are executed from there. This publication describes them in logical groups as follows:

- Terminator routines

  The $IJBSxxx phases making up these routines handle abnormal end-of-task conditions. Phase $IJBSEOT (End-of-task processing) is described in chapter 6. For phase $IJBSDMP (System dump processing) refer to manual *Diagnosis Reference: Serviceability Aids*.

- Miscellaneous service routines

  The $IJBSxxx phases making up these routines are called to perform services such as label handling and hard-copy file access. They are described in Chapter 7.

Available Diagnostic Aids

Appendix A, “Diagnostic Aids” on page 171 of this publication discusses the diagnostic aids available for isolating a programming error in a logical transient or a $IJBSxxx phase. The appendix tells how to use this publication for that purpose.

Chapter 2. Attention Routines

Introduction

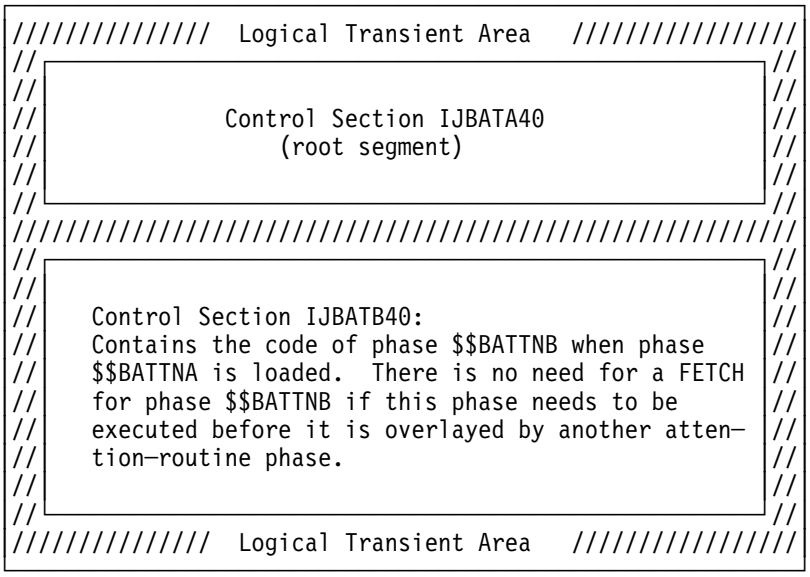
The logical transients making up the attention routines are loaded for execution by the attention task of the supervisor; these transients, whose phase names start with the characters $$BATT, process attention routine (AR) commands. This attention task is activated when the operator at the console does one of the following:

- Presses the Request key at the console and subsequently enters an attention-routine (AR) command.
- Enters an AR command without having pressed the Request key before, which is possible only at a system with a display-type operator console.

If the attention task finds that an AR command is to be processed, the task causes phase $$BATTNA, one of the logical transients that make up the attention routines, to be loaded into the logical transient area (LTA). Phase $$BATTNA includes module $$BATTNH.

The code of phase $$BATTNA functions as a root phase; it acquires supervisor state for the processing of the command. The phase stores the command in a buffer unchanged. Phase $$BATTNH, in turn, selects the appropriate command-processing phase.

The selected command-processing phase, when loaded, overlays phase $$BATTNH.



.

Figure 1. Logical-Transient Area for the Attention Routines

External Input: AR commands as entered by the operator at the system's console. For a detailed description of these commands, see the *System Control Statements* manual.

External Output: Depends on the operator command that is to be processed; messages as required.

Design Information

Figure 2 on page 5 shows the flow of control for initiating the attention routines.

Figure 3 on page 6 shows the flow of control for invoking the logical transients that process a specific AR command. Unless mentioned otherwise in this diagram, the root phase $$BATTNA receives control again after the particular command has been successfully processed by the phase(s) called by phase $$BATTNH.

The logical transients making up the attention routines are described in this section in alphabetical sequence by their phase names.

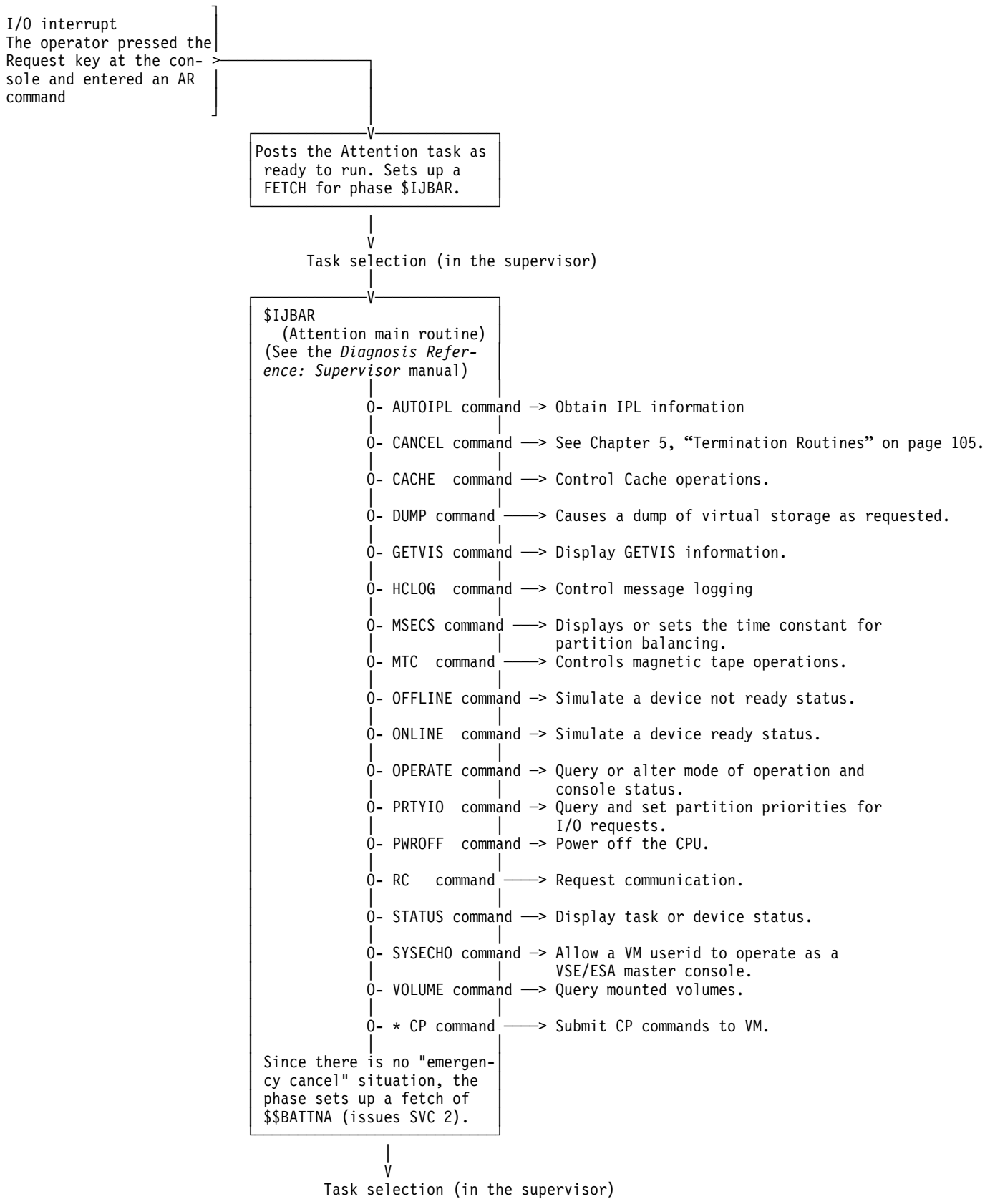


Figure 2. Initiation of the Attention Routines

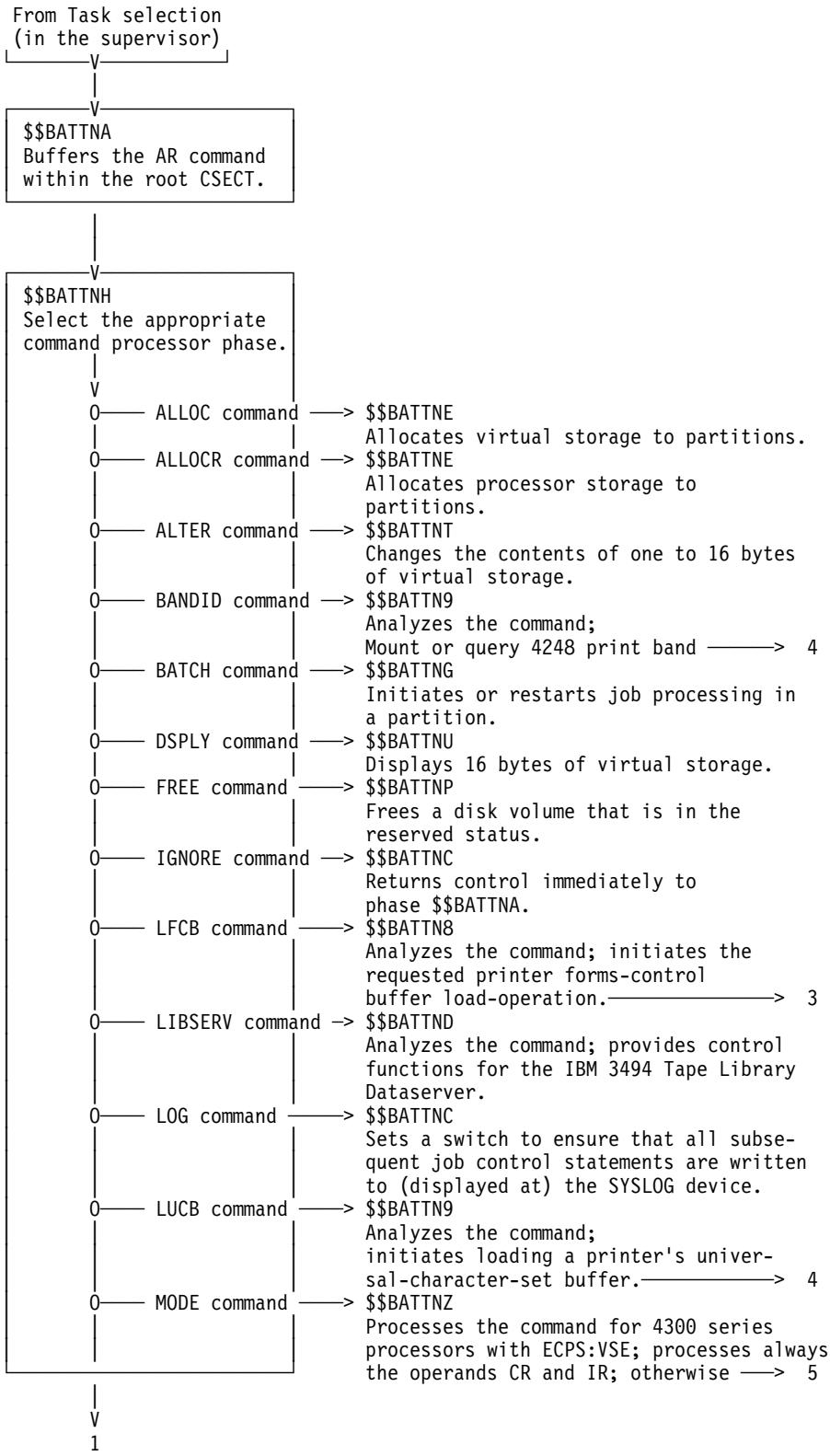


Figure 3 (Part 1 of 5). Attention-Routine Control Flow

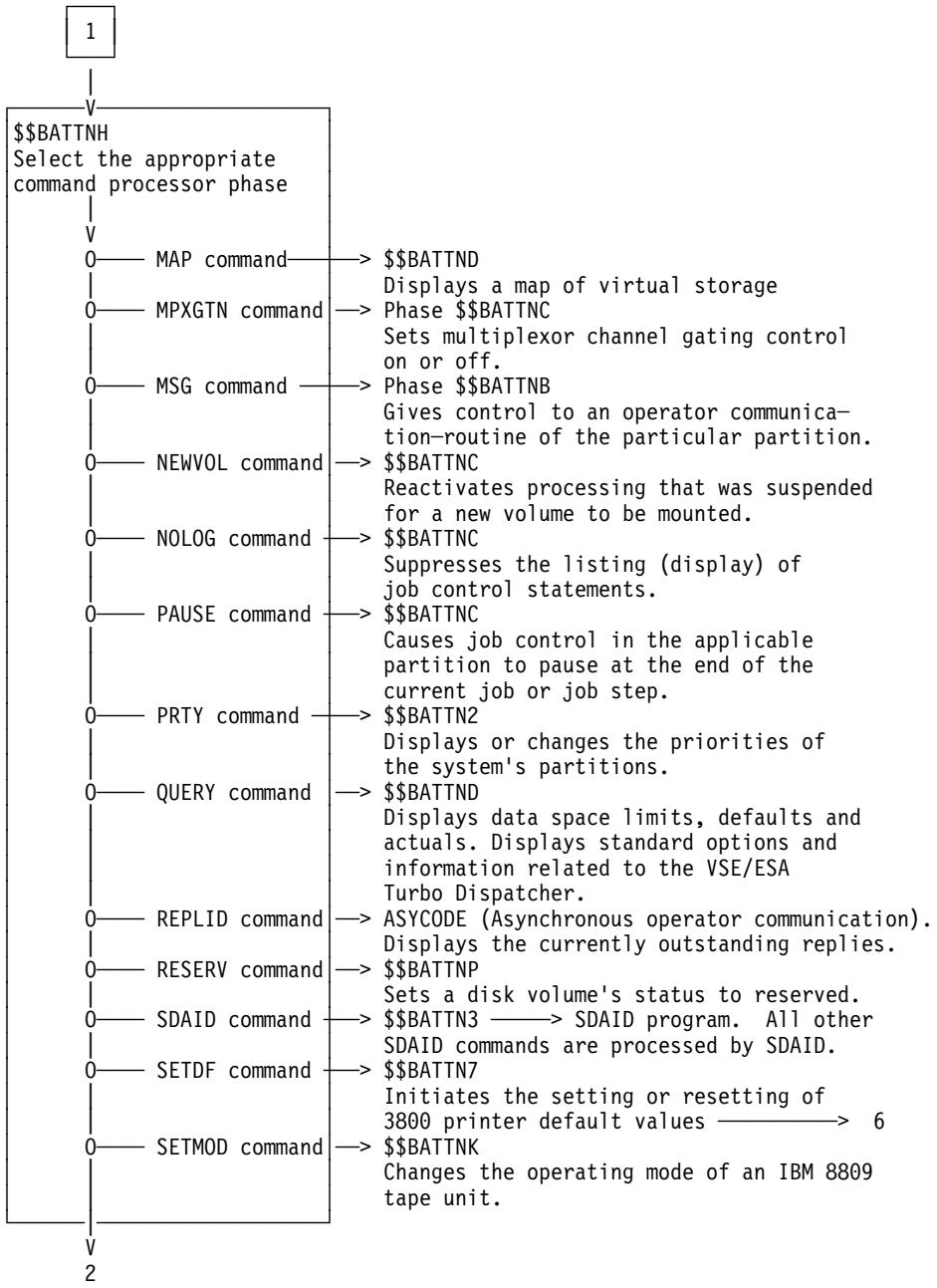
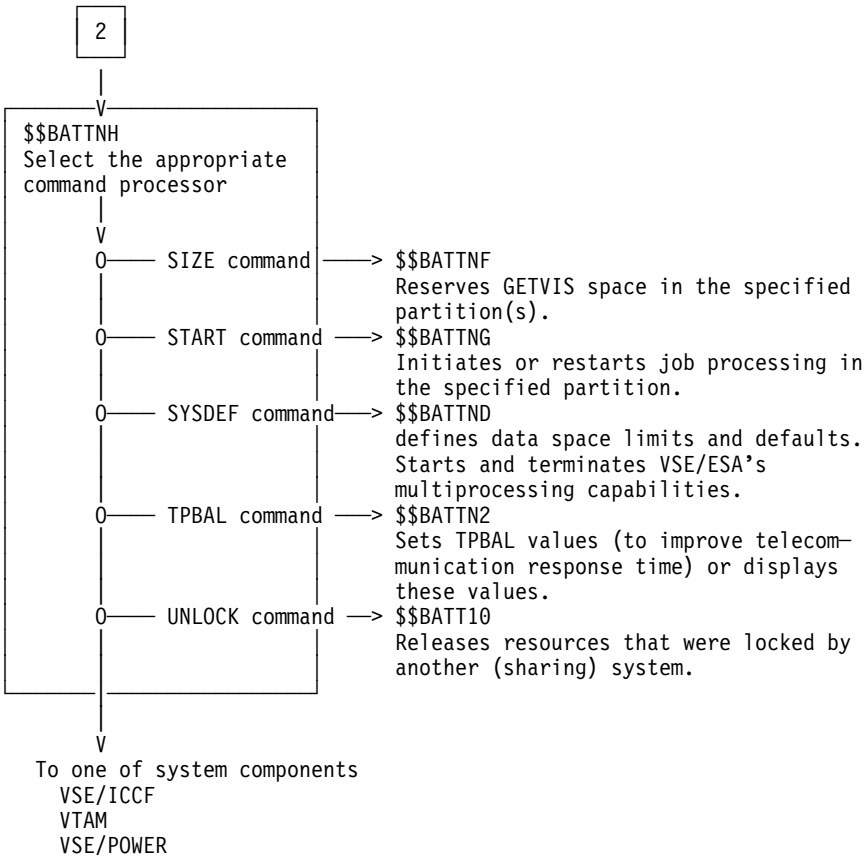


Figure 3 (Part 2 of 5). Attention-Routine Control Flow



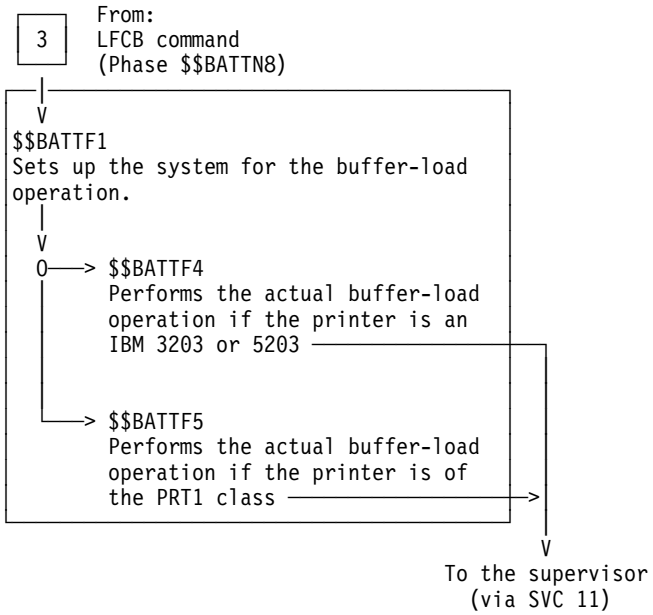
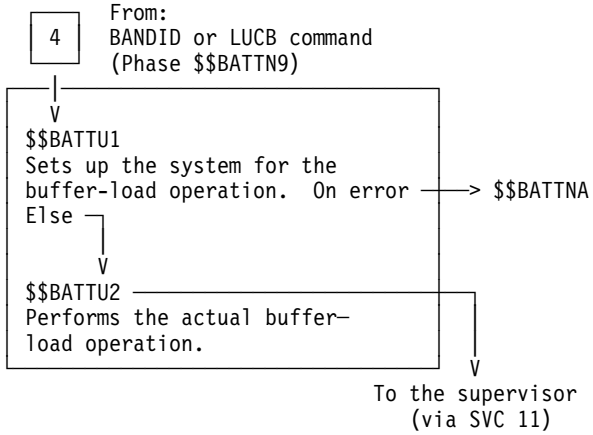


Figure 3 (Part 3 of 5). Attention-Routine Control Flow



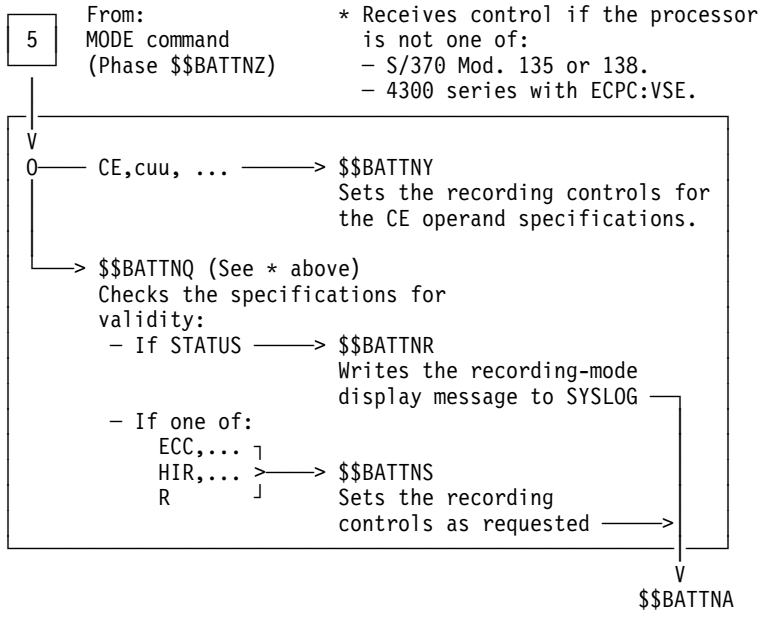


Figure 3 (Part 4 of 5). Attention-Routine Control Flow

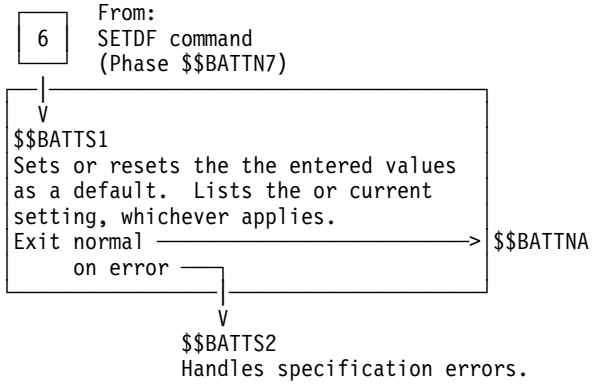


Figure 3 (Part 5 of 5). Attention-Routine Control Flow

Phase $$BATTF1 – LFCB Command Processing, Phase 2

Entry Point: PFRCOMBS.

Function: Prepares the system for an FCB load operation in accordance with user specifications.

Called By: Phase $$BATTN8.

Phases Called

- $$BATTF4 for a 3203 or 5203 printer.
- $$BATTF5 for a PRT1 printer.

Subroutines Used: None.

Data Areas Used

- FINFAREA/INFAREA, information records.
- FPRMVALT, parameter-value table.
- FPRMADRT, parameter-address table.

Messages Caused

```
1B12D   opcode OPERAND n 'erroneous-operand/delimiter' explanation
```

In the message, explanation can be one of the following:

Invalid
Duplicated
Not known to system
Device without FCB
Device down

Messages Issued: None – Message 1B12D is written to SYSLOG by phase $$BATTNA (entry point ERRRT).

Input: Data in areas FINFAREA, FPRMADRT, and FPRMVALT. For their layout and contents, see this chapter's "Data Area Information" section.

Output: Either of the following:

- An error message (see "Messages Originated" above)
- Information record at the label INFAREA (for its layout and content, see this chapter's "Data Area Information" section).

Exit Normal

- Phase $$BATTF4 for a non-PRT1 printer with FCB
- Phase $$BATTF5 for a PRT1 printer

Exit Error: Phase $$BATTNA.

Register Use

12 Base register for the phase.

Sequence of Operation

PFRCOMBS: Returns control to phase $$BATTNA if phase $$BATTN8 found a syntax error.

Looks up the PUB table for an entry that contains the specified device address (in the form cuu); checks whether this device is a PRT1 printer or a non-PRT1 printer with an FCB; ensures that the device is ready.

Sets the controls for loading either of phases $$BATTF4 (non-PRT1) and $$BATTF5 (PRT1); moves the content of FINFAREA to INFAREA.

PFESYNTX: Builds message 1B12D to indicate a syntax error found by phase $$BATTN8 and take the error exit.

PFESMT: Builds message 1B12D to indicate an error condition encountered by this phase and take the error exit.

Phase $$BATTF4 – FCB Load Execution for 3203 and 5203

Entry Point: IJBATTF4+8.

Function: Executes the LFCB command for a non-PRT1 printer using physical I/O (SYSUSE is not used).

The phase, when called, overlays the root phase ($$BATTNA).

Called By: Phase $$BATTF1.

Phases Called: None.

Subroutines Used:

LISTIO: To initiate the required printer operations.
LOGIO: To write a message to SYSLOG.
LOGIOW: To write a message to SYSLOG and read a response.

Data Areas Used

- BUFFER, the FCB-image load area.
- COMREG, partition communication region (in the supervisor).
- INFAREA (see "Input" below)
- PUBTAB, PUB table (in the supervisor).

Messages Caused

```
1B13A   X'cuu' [NEEDS FORMS=xxxx][,SET LPI=n] STOP
        PRINTER IF NECESSARY AND PRESS END
1B15I   PHASE phasename NOT FOUND
1B16I   FCB LOAD INVALID FOR SPECIFIED PRINTER
1B20A   INVALID RESPONSE
```

Messages Issued: None – A message caused by this phase is written to SYSLOG by phase $$BATTNA (entry point ERRRT).

Input

- INFAREA, the last 26 bytes of the B-transient area (for the layout and content of this area, see this chapter's "Data Area Information" section).
- FCB-image phase to be loaded into an FCB.

Output

- Messages as required (see above).
- A loaded FCB.
- A verification message on the printer if this message was requested by the user.

Exit Normal: To the supervisor via SVC 11.

Exit Error: To the supervisor via SVC 11.

Register Use

0	Pointer to the buffer-load address.
1	Used in macros such as EXCP and FETCH.
2-3	Work registers.
4	Link register for I/O subroutine call.
5	Address of partition COMREG.
7	Pointer to (and base for) INFAREA.
8	Printer-PUB pointer-register.
10	Work register.
15	Base register for this phase.

Sequence of Operation

IJBATTF4+8: Set up addressability.

LOADPHAS: Retrieve the required FCB-image phase from the applicable sublibrary.

PUBLOOP: Wait for the involved printer to become free; force a skip to the top of a page (for synchronization purposes).

RETRY: Load the retrieved FCB image into the print buffer.

DOSKIP: Cause a skip to channel 1 operation on the involved printer.

Phase $$BATTF5 – FCB Load Execution for PRT1 Printers

Entry Point: IJBATTF5+8.

Function: Executes the LFCB command for a PRT1 printer using physical I/O (SYSUSE is not used).

The phase, when called, overlays the root phase ($$BATTNA).

Called By: Phase $$BATTF1.

Phases Called: FCB-image phase as required.

Subroutines Used:

LISTIO: To initiate the required printer operations.
LOGIO: To write a message to SYSLOG.
LOGIOW: To write a message to SYSLOG and read a response.

Data Areas Used

- BUFFER, an FCB-image area (holds image that is to be loaded).
- BUFF2, an FCB-image area (holds image of buffer prior to being loaded).

- COMREG, partition communication region (in the supervisor).
- INFAREA (see "Input" below).
- PUBTAB, PUB table (in the supervisor).

Messages Caused

```
1B14A   X'cuu' NEEDS FORMS=xxxx STOP PRINTER AND PRESS END
1B15I   PHASE phasename NOT FOUND
1B16I   FCB LOAD INVALID FOR SPECIFIED PRINTER
1B17I   LPI=n AND PRT1 FCB LOAD DISAGREE
1B20A   INVALID RESPONSE
```

Messages Issued: None – A message caused by this phase is written to SYSLOG by phase $$BATTNA (entry point ERRRT).

Input

- The operands of the LFCB command in the buffer of phase $$BATTNA.
- INFAREA, the last 26 bytes of the B-transient area. (for the layout and content of this area, see this chapter's "Data Area Information" section).

Output

- Message 1B14A, if necessary (see "Messages Originated" above).
- A loaded FCB.
- A verification message on the printer if this message was requested by the user.

Exit Normal: To the supervisor via SVC 11.

Exit Error: To the supervisor via SVC 11.

Register Use

0	Pointer to the buffer-load address.
1	Used in macros such as EXCP and FETCH; base register for CCB access.
2-3	Work registers.
4	Link register for I/O subroutine call.
5	Address of partition COMREG.
7	Address of area INFAREA.
8	Printer PUB pointer register.
10	Work register.
15	Base register for this phase.

Sequence of Operation

IJBATTF5+8: Set up addressability.

LOADPHAS: Retrieve the required FCB-image phase from the applicable sublibrary.

TESTLPI: Check the retrieved FCB-image to see if it is syntactically correct.

PUBLOOP: Wait for the involved printer to become free; read the current buffer contents; build and load a synchronization FCB image and perform a skip to the top edge of forms.

SKIPLOAD: Load the retrieved FCB image into the print buffer.

DOSKIP: Cause a skip to channel 1 operation on the involved printer.

Phase $$BATTNA – Attention-Routine Root

Entry Point: ENTRYP.

Function: The phase functions as root phase for the processing of attention-routine commands; it acquires supervisor state for this purpose. The phase defines the storage areas and constants common to the other phases. The phase contains various subroutines used by other phases; the basic functions of these subroutines are:

ATTDOIT: A multi-purpose routine performing the action that is requested by a parameter passed to the routine in register 11. Actions that may be: Go to DTCHAT, or to MSGOUT or load a new phase.

DTCHAT: To deactivate the attention routine.

DTINUM: To read a record from SYSLOG. If only blanks are red, return to caller.

EXCPRG: To execute a channel program.

ERRRTN: To write an error message to SYSLOG and to read from SYSLOG to accept the next command.

MSGOUT: To write, to SYSLOG, the line of data that is contained in the SYSLOG-write buffer

SCANR: To scan a command and return a pointer to and the length of each operand, one at a time. Returns control to the caller with information in registers as listed below:

5 Pointer to operand that is to be processed.
6 Length of operand field yet to be scanned.
7 Length of the operand that is to be processed.

Called By: Task selection in the supervisor.

Phases Called: $$BATTNH.

Subroutines Used:

GETCMND: To move the command from the AR buffer to the buffer in the root phase. The subroutine is located in phase $$BATTNB.

Data Areas Used

- ATNCOM, the attention-routine communication area (in this phase).
- COMREG, the partition communication region (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1I0nt   INVALID COMMAND
```

Messages Issued: Messages caused by attention-routine phases as indicated in the various phase descriptions.

Input: Attention-routine (AR) commands from SYSLOG or from the area ASYBUFF0 if the ASYNOC function is active.

Output: Error messages as necessary (primarily originated by the AR-command processing phases).

Exit Normal: Return via SVC 11.

Exit Error: Return via SVC 11.

Register Use

9 Subroutine link register.
10 Contains the return address labeled CONTROL.
12 Contains the error-return address labeled NVSERR.
13 Address of partition COMREG.
14 Base register for the phase.
15 Pointer to area ATNCOM.

Sequence of Operation

INITIAL: Initialize the channel program and the CCB for accessing the SYSLOG device.

NVSERR: Print message 1I0nI if the phase called to process a command returns control via this entry point.

Phase $$BATTNB – Command Processing (MSG)

Entry Point: MSG.

Function: Processes the MSG command by selecting the correct operator communications option table.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

SCANR (in phase $$BATTNA): To scan the currently processed MSG command.
EXCPRG: To execute a channel program.
MSGOUT (in phase $$BATTNA): To write, to SYSLOG, any message caused by this phase.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- COMREG, the partition communication region (in the supervisor).
- PIB, the program information block (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1I31I   INSUFFICIENT SVA STORAGE
1IXXI   INPUT DATA TOO LONG.
1P60I   NO ROUTINE LINKAGE
1P70I   PROCESSING ROUTINE ACTIVE
```

Messages Issued: None – Messages caused by the phase are issued by subroutine MSGOUT.

Input: MSG command.

Output: None.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

9 Link register to subroutines.
13 Address of partition COMREG.
14 Base register for the phase.
15 Address of area ATNCOM.

Sequence of Operation.

MSG: Get the operand of the command; pick up the partition-identification key of the specified partition.

STEXCD: Establish operator communication linkage as required.

Phase $$BATTNC – Command Processing

Entry Point: IJBATC40.

Function

- Processes the commands PAUSE, LOG, and NOLOG; sets switches in the job-control switch bytes accordingly.
- Processes the NEWVOL command by reactivating job control in the specified partition, if this partition is waiting for a volume to be mounted.
- Sets or resets the multiplexor gating switch.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1I0nI   INVALID COMMAND
```

Messages Issued: None – Message 1I0nI is written to SYSLOG by phase $$BATTNA (entry point NVSERR).

Input: One of the following AR commands:

```
IGNORE                      NEWVOL
LOG / NOLOG                 PAUSE
MPXGTN (system generated)
```

Output: None.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

9 Link register to subroutines.
13 Address of partition COMREG.
14 Base register for the phase.
15 Address of area ATNCOM.

Sequence of Operation

LOG: For a LOG or NOLOG command, sets the LOG/NOLOG bit in the job control switch byte of the affected partition's COMREG to either:

On = logging is required, or
Off = logging is not required.

whichever applies.

MPXGTN: For an MPXGTN command, sets or resets the byte multiplexer gating switch in field IJBFLG02 of SYSCOM.

NEWVOL: For a NEWVOL command, gets the key of the partition specified in the command; activates this partition by setting the appropriate flag in the PIB.

PAUSE: For a PAUSE command, sets on (in the specified partition's COMREG) the control bit for a pause or for a pause at EOJ.

Phase $$BATTND – Command Processing (LIBSERV, MAP, QUERY, SYSDEF)

Entry Point: IJBATD40.

Function: Processes attention routine commands LIBSERV, MAP, QUERY and SYSDEF.

- LIBSERV
  Provides control functions for IBM 3494 Tape Library Dataserver.
- MAP
  Provides, on SYSLOG, a listing of the following:
  - The amount of either the virtual or real storage (incl. PFIX counters) currently allocated to each partition.
  - Each partition's address.
  - The name of the job that is being executed in a partition if the partition is active.
  - For 'MAP partition-id', among others, the (virtual and real) storage allocated to the partition, the names of job and phase running, the space-id where the partition belongs to.
- QUERY
  Provides, on SYSLOG, a listing of information about data spaces (DSPACE operand), standard options (STDOPT operand), or information related to the VSE/ESA Turbo Dispatcher (TD operand).
- SYSDEF
  Sets the system defaults for data spaces (DSPACE operand) and allows to activate and terminate multiprocessing in VSE/ESA (TD operand).

Functions as an overlay phase of root-phase $$BATTNA.

Called By: phase $$BATTNH.

Phases Called:

- $IJBSLIB for LIBSERV
- $IJBMAP for MAP
- $IJBSDSP for SYSDEF and QUERY

Subroutines Used:

ATTDOIT (in phase $$BATTNA):
To perform the action requested by the parameter in register 11.
SCANR2/3 (in phase $$BATTNA):
To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- JCARAREA, the interface area to the SVA routines $IJBMAP/$IJBSDSP.

Messages Caused

- 1I0nI INVALID COMMAND
- 1S40t SYSTEM ERROR, GETVIS RET.CODE = rc
- 1Y04I INVALID STATEMENT. NO NATIVE ESA MODE.

Messages Issued: None.

Input/Output

INPUT	OUTPUT
LIBSERV cmd	Provides control functions for IBM 3494 Tape Library Dataserver.
MAP cmd	A map of the system's storage displayed on the console
SYSDEF cmd	System wide defined limits and defaults for data spaces, start and terminate VSE/ESA's multiprocessing capabilities.
QUERY cmd	Display of data space information, standard options and information related to VSE/ESA's Turbo Dispatcher.

Exit Normal: To phase $$BATTNA

Exit Error: To phase $$BATTNA

Register Use

1-4	Work registers.
7	Address of SYSCOM.
9-11	Link registers.
13	Address of the BG partition's COMREG.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

IJBATD40: Set up interface area for IJBMAP/IJBSDSP/IJBSLIB
call IJBMAP/IJBSDSP/IJBSLIB
return to $$BATTNA

OUTROUT: display line or message prepared by IJBMAP/IJBSDSP/IJBSLIB
return to IJBMAP/IJBSDSP/IJBSLIB

SCNROUT: scan additional operands in LIBSERV, MAP, QUERY or SYSDEF command
return to IJBMAP/IJBSDSP/IJBSLIB

Phase $$BATTNE – Command Processing (ALLOC, ALLOCR)

Entry Point: IJBATE40.

Function: Processes the attention-routine commands ALLOC and ALLOCR.

Functions as overlay phase of phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used

ATTDOIT (in phase $$BATTNA): To perform the action requested by the parameter in register 11.
SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the communication region (in the supervisor).
- SIDSTR, the space control block address table (in the supervisor).

Messages Caused

```
1P03I   ALLOCATION COMPLETED, WARNING, RC=xx
1P05D   SYNTAX ERROR IN SIZE STATEMENT - error-field
1I39t   INVALID SPACE ID OR PARTITION ID
```

Messages Issued: None.

Input: The ALLOC or ALLOCR command as submitted by the user.

Output: Specified partition sizes inserted in the system's partition table.

Exit Normal: To phase $$BATTNA.

Exit Error: Same as for "Exit Normal" avove.

Register Use

9-11	Link registers for subroutines.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

ALLOC: Checks the operands of the storage allocation commands for validity.

GETK: Gets the storage out of the command and converts it to binary.

ISALL: Issues the ALLOCATE macro and checks its return code.

Phase $$BATTNF – Command Processing (SIZE)

Entry Point: IJBATF40.

Function: Processes the SIZE command.

For each of the specified partitions, the phase

- Builds, at SLPL, the SETLIMIT-macro parameter list.
- Issues the SETLIMIT macro.
- Checks the macro return code and, if that code is not zero, issues an appropriate error message.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used

ATTDOIT (in phase $$BATTNA): To perform the action requested by the parameter in register 11.
SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SLPL, the SETLIMIT-macro parameter list.
- SYSCOM, the communication region (in the supervisor).

Messages Caused

```
1P03I   ALLOCATION COMPLETED, WARNING, RC=04
1P04D   INVALID SIZE VALUE FOR PARTITION nn, RC=xx
```

Messages Issued: None – A message caused by the phase is written to SYSLOG by phase $$BATTNA (entry point ERRRTN).

Input: The SIZE command as submitted by the user.

Output: The new values for program size(s) set in table SLPL (in the supervisor).

Exit Normal: To phase $$BATTNA.

Exit Error: Same as "Exit Normal" above.

Register Use

1-4	Work registers.
5-7	Pointer registers.
8-11	Link registers.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

SIZE: Checks the SIZE operands for validity.

SETUP: Issues the SETLIMIT macro and checks its return code.

CHKPART: Checks the partition name and converts it into SLPL.

Phase $$BATTNG – Command Processing (BATCH, START)

Entry Point: IJBATG40.

Function: Activates the partition indicated in the START or BATCH command, whichever triggered the operation.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

INITOPR: To check whether the specified partition (BG by default) is available and inactive. If so, the subroutine starts the partition by issuing a TREADY macro.
SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).
- COMREG, the partition communication region (in the supervisor).

Messages Caused

```
1I0nI   INVALID COMMAND
1P1nD   AREA NOT AVAILABLE OR PARTITION ACTIVE
```

Messages Issued: None – Messages are written to SYSLOG by phase $$BATTNA:

Message 1P1nD – Via entry point ERRRTN.
Message 1I0nI – Via entry point NVSERR.

Input: START or BATCH command.

Output: A partition-active indication for the supervisor.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

9	Subroutine link register.
13	Address of partition COMREG.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

START: Calls subroutines INITOPR.

Sets a bit in byte JCSW4 of the involved partition to indicate that a BATCH or START command was issued for this partition.

Phase $$BATTNH – Command Scan

Entry Point: CHKSTT.

Function: Loads the first (or only) command processing phase for the currently processed attention routine command.

Functions as an overlay phase of root-phase $$BATTNA.

Loaded together with phase $$BATTNA.

Phases Called: The appropriate command-processing phase if an attention-routine (AR) command is being processed. For the command-to-phase relationship, see Figure 3 on page 6.

If the command to be processed is not an AR command, another system component (VSE/ICCF, VTAM, or VSE/POWER) receives control (see "Sequence of Operation" below).

Subroutines Used:

ATTDOIT (in phase $$BATTNA): To perform the action requested by the parameter in register 11.
EXCPRG: To execute a channel program.
SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1I0nI   INVALID COMMAND
```

Messages Issued: None – Messages are written to SYSLOG by phase $$BATTNA:

Message 1I0nI – Via entry point NVSERR.

Input: Attention routine command as contained in the buffer of the root phase.

Output: None.

Exit Normal: To the selected command processing phase or to another system component.

Exit Error: To phase $$BATTNA.

Register Use

9	Link register for subroutine calls.
13	Address of partition COMREG.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

CHKSTT: Scans the command that is to be processed. For an attention routine (AR) command: determines the phase that is to be called to process the command and, if applicable, the correct displacement within that phase; uses a branch table for that purpose. For a command other than attention routine, see the label HKDOSE below.

LOADBAS: Sets up the load request for the required command-processing phase and uses the load-SVC instruction (at LOADIT in the root phase, $$BATTNA) to initiate the load operation.

HKDOSE: For a command other than attention routine, transfers control to other system components for command analysis, provided these components are active:

- VSE/ICCF – At label ETSSHOOK

  VSE/ICCF returns control to label ETSSHK1 if the command was an ICCF command; this initiates detaching the attention routine task. VSE/ICCF returns control to label HKETSSE if the command was not an ICCF command.

- VTAM – At label VTAMHOOK

  VTAM returns control to label VTAMHK1 if the command was a VTAM command; this initiates detaching the attention routine task. VTAM returns control to label HKVTAME if the command was not a VTAM command.

- VSE/POWER – At label POWRHOOK

  VSE/POWER returns control to label POWRHK1 if the command was a POWER command; this initiates detaching the attention routine task. VSE/POWER returns control to label HKPOWRE if the command is not a POWER command.

HKEND: Issues an 'INVALID STATEMENT' message if the command to be processed applies neither to the attention routine nor to any of the above listed components; initiates detaching the attention routine task.

Phase $$BATTNK – Command Processing (SETMOD)

Entry Point: IJBAT040.

Function: Processes the SETMOD command and sets the mode-control byte for the IBM 8809 tape unit according to the value specified in the command. The new mode setting is valid until the next SETMOD command submitted by the user or until next IPL, whichever occurs earlier.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

SCANR (in phase $$BATTNA): To scan the currently processed command.
MSGOUT: To write, to SYSLOG, any message caused by this phase.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1P40D   DEVICE cuu NOT DEFINED
1P41D   INVALID DEVICE TYPE IN SETMOD COMMAND
1P42D   INVALID TAPE MODE ss HAS BEEN SET FOR DEVICE cuu
1P43I   TAPE MODE ss HAS BEEN SET FOR DEVICE cuu
```

Messages Issued: None.

Input: SETMOD command as submitted by the user.

Output: New setting of the mode-control byte (byte 5 in the PUB entry for the particular IBM 8809).

Exit Normal: To phase $$BATTNA.

Exit Error: Same as "Exit Normal" above.

Register Use

9-10 Link registers.
13 Base register for access to COMREG.
14 Base register for this phase.
15 Address of area ATNCOM.

Sequence of Operation

IJBAT040: Inserts the specified mode into byte 5 of the PUB entry for the IBM 8809 whose address is specified in the command.

Note: The supervisor uses this mode indication for generating a set-tape-mode CCW when an EXCP for that IBM 8809 occurs.

Valid mode specifications are given in the table below:

Specification is Mnemonic	Numeric	Operation Mode
HL	90	High speed and long gap
HS	30	High speed and short gap
LL	50	Low speed and long gap
* LS	60	Low speed and short gap

* The default.

Phase $$BATTNP – Command Processing (FREE, RESERV)

Entry Point: IJBATP40.

Function: Processes the FREE and RESERV commands by

- Resetting the reserve status of the specified disk device in response to the FREE command.
- Setting the specified disk device into the reserved status in response to the RESERV command.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

ATNSCN2 (an alias name of SCANR2 in phase $$BATTNA): To scan the currently processed command.
ATNDOIT (an alias name of ATTDOIT in phase $$BATTNA): To perform the action requested by the parameter in register 11.

Data Areas Used: ATNCOM, the attention-routine communication area (in phase $$BATTNA).

Messages Caused

1I33I cuu CANNOT BE RESERVED
1I34I cuu CANNOT BE FREED

Messages Issued: None – A message caused by the phase is written to SYSLOG by phase $$BATTNA (entry point ERRRTNN).

Input: Either a FREE or a RESERVE command specifying a disk address in the form cuu.

Output: Messages as required.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

9	Link register for subroutine calls.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

IJBATP40: Determines the type of command, FREE or RESERVE, and passes control to the appropriate processing routine.

FREE: Checks for correct syntax of the specification; sets the status of the device to "free," and prints a completion message.

RESERV: Checks for correct syntax of the specification; sets the status of the device to "reserved," and prints a completion message.

Phase $$BATTNQ – MODE Command Analysis (/370-145/148 and up, 30xx)

Entry Point: IJBRASB1+12.

Function: Checks for validity the operands of a MODE command if the calling phase found that the command applies to a System/370 CPU model 145/148 or 155/158 or to a 3031 or 3033 processor complex operated under control of VSE. Sets indicator bits and value bytes in phase $$BATTNA's I/O buffer accordingly.

Functions as overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNZ.

Phases Called: Phases $$BATTNR and $$BATTNS.

Subroutines Used:

SCANR2 (in phase $$BATTNA): To scan the currently processed command.
SCANR3 (in phase $$BATTNA): To scan the specified operands, one at a time.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).
- COMREG, the partition communication area (in the supervisor).
- RASLINK, the RAS-linkage area (in the supervisor).
- RASTAB, the RAS-table (in the supervisor).

Messages Caused: None.

Messages Issued: None.

Input: Indicator and command specific values in the I/O buffer of phase $$BATTNA (see also "Sequence of Operation" below).

Output: Contents of registers 4, 5, and 8 as indicated under "Register Use" below.

Exit Normal

- To phase $$BATTNR for a MODE STATUS specification.
- To phase $$BATTNS otherwise.

Exit Error: To phase $$BATTNA.

Register Use

4	The new error-count threshold or time value.
5	Pointer to the RAS table; pointer to operand that is currently processed.
6	Length of MODE command operand field that remains to be processed.
7	Length of the operand that is currently processed.
8	Pointer to the RAS linkage area.
9	Link register for subroutine calls.
13	Base register for access to COMREG.
14	Base register for this phase.
15	Base register for the root phase, $$BATTNA.

Sequence of Operation

IJBRASB1+2: Checks byte CONFIG in COMREG to verify that a System/370-architecture processor of Model 145/148 or larger is used; ensures that the command is of correct length (not longer than 27 bytes); converts to uppercase any lowercase letters in the parameter field; initializes the parameter indicator byte to zero.

Checks the values specified for E=number and T=number, which must be within ranges as follows:

Type of Processor	Valid Range for E=number	Valid Range for T=number*
S/370 Mod. 145/148	16 – 9999	8 – 9999
Other processors	8 – 9999	8 – 9999

* The value specified for T=number is converted to clock units (3433 units/hour) before it is moved to $$BATTNA's I/O buffer.

A valid parameter causes an indicator bit to be set in $$BATTNA's I/O buffer as follows:

Bit 0 = Hardware instruction retry (HIR)
Bit 1 = Error correction code (ECC)
Bit 2 = Recording mode (,R)
Bit 3 = Quiet mode (,Q)
Bit 4 = Threshold mode (,TH)
Bit 5 = Main (processor) storage (,M)
Bit 6 = Control storage (,C)
Bit 7 = Error count threshold (,E=number) or time threshold (,T=number)

Phase $$BATTNR – MODE Command Status Report Processing

Entry Point: IJBRASB2+12.

Function: Builds and then writes, to SYSLOG, recording mode status messages except when the processor is one of the following:

A System/370 Model 135 or 138
A 4331 or 4341

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNQ.

Phases Called: None.

Subroutines Used

EXCPRG in phase $$BATTNA: To print status report on SYSLOG.

Data Areas Used

- ATNCOM, the attention routine communication area (in phase $$BATTNA).
- COMREG, the partition communication area (in the supervisor).
- RASLINK, the RAS-linkage area (in the supervisor).
- RASTAB, the RAS-table (in the supervisor).

Messages Caused: Status report message.

Messages Issued: None.

Input: MODE command as submitted by the user.

Output: Recording-mode status display on SYSLOG

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

5	Pointer to RAS table.
8	Pointer to RAS linkage area.
9-11	Link registers for subroutine calls.
13	Address of partition COMREG.
14	Base register for this phase.
15	Address of area BATTNA.

Sequence of Operation

IJBRASB2+12: Builds and writes to SYSLOG the recording-mode status messages, which have a format as follows:

- For a System/370 Mod. 145/148

  HIR {R|Q},aaaa/eeee,bbbb/tttt
  ECC {R|Q},{M|C},aaaa/eeee,bbbb,tttt

Builds and writes two such messages, one for the user-available processor storage and one for the control storage.

- For a processor other than 145/148 and serviced by this phase

  HIR {R|Q},aaaa/eeee,bbbb/tttt
  ECC {R|Q},aaaa/eeee,bbbb,tttt
  BUF DLT=xxx

In these messages:

```
aaaa  =  Current error-count
bbbb  =  Elapsed time
   C  =  Control storage
 ECC  =  Error correction code
eeee  =  Error count threshold
 HIR  =  Hardware-instruction retry
   M  =  Main (processor) storage
   Q  =  ECC or HIR is in quiet (not recording) mode
   R  =  ECC or HIR is in recording mode
tttt  =  time threshold
 xxx  =  Number of deleted inoperable buffer pages
```

To build an HIR message line, the phase

- Checks the following bits of byte MCMODE in the RAS table:

  Bit 0 = Recording mode – HIR,R
  Bit 1 = Quiet mode – HIR,Q

- Gets the error count and time information recorded in the RAS table at label HIR.

To build an ECC message line, the phase (1) uses the information recorded in the RAS table at ECCMAIN and (2) the following bits of byte MCMODE:

Bit 2 = Recording mode – ECC,R,... was specified.
Bit 3 = Quiet mode – ECC,Q,... was specified

If the processor is an S/370 Mod. 145/148, the phase inserts an M into the display line to signify main (processor) storage (see the format descriptions, above). Moreover, the phase uses the following MCMODE bits to build the ECC status line for control storage:

Bit 4 = Recording mode – ECC,R,C,... was specified.
Bit 5 = Quiet mode – ECC,Q,C,... was specified.
Bit 6 = Threshold mode – ECC,TH,C,... was specified.

If the processor is not an S/370 Mod. 145/148, the phase writes, to SYSLOG, a separate line indicating the number of deleted buffer pages.

Notes:

1. The HIR and ECC control bits (0, 2, and 4 of byte MCMODE) are initialized to 1's to indicate "recording" mode.
2. The elapsed time and the time threshold are converted from clock units to hours before they are printed. If the elapsed time is less than one hour, or if the time of day clock is not operational with the clock validity indicator on, the phase inserts for the elapsed time a value of zero.
3. In the RAS table, fields HIR and ECC are always initialized to values as follows:

RAS Field	System/370 Mod.145/148 eeee	tttt	Other Processors eeee	tttt
HIR	8	8	8	8
ECC	16	8	8	8

Phase $$BATTNS – MODE Command Validity Checking

Entry Point: IJBRASB3+12.

Function: Checks whether the operands specified in the currently processed MODE command are a valid combination and, if so, sets the requested controls accordingly.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNQ.

Phases Called: None.

Subroutines Used

ATTDOIT (in phase $$BATTNA): To print a message.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- I/O buffer in phase $$BATTNA, accessed via DSECT BATTNA.
- RAS linkage area (in the supervisor), accessed via DSECT RASLADR.
- RAS table (in the supervisor), accessed via DSECT RASTAB. Fields accessed are:
  - CPUID, which indicates the type of processor being used.
  - ECCMAIN, which contains the accumulated ECC count for main (processor) storage.
  - ECMACNT, which contains the current ECC count for main (processor) storage.
  - HIR, which contains the accumulated HIR count.
  - HIRACNT, which contains the current HIR count.
  - MCMODE, a byte that indicates the current system mode of operation.
- COMREG, the partition communication region (in the supervisor).

Messages Caused

```
1P30I   COMMAND IGNORED IN VM/370 ENVIRONMENT.
```

Messages Issued: None.

Input: Updated operand-specifications indicator byte MODEBITS in $$BATTNA's I/O buffer.

Output: Indications in the RAS table for the following, if specified in the MODE command:

- Mode change
- Error-count threshold
- Time threshold

For more details about the RAS table, refer to *VSE/Central Functions, Error Recovery and Recording Transients Diagnosis Reference*.

Exit Normal: To phase $$BATTNA at label CONTROL.

Exit Error: To Error-message subroutine pointed to by register 12.

Register Use – General Purpose

3 Requested hardware threshold (TH) value.
4 Requested error-count threshold.
5 Pointer to the RAS table.
8 Pointer to the RAS linkage area.
12 Pointer to $$BATTNA's error message subroutine, NVSERR.
13 Base register for access to COMREG.
14 Contains the end address of the root phase, $$BATTNA.
15 Address of area BATTNA, an alias name of ATNCOM.

Register Use – Control

14 Used in LCL instructions.

Sequence of Operation

IJBRASB3+12: Checks the user specifications as set by phase $$BATTNQ into phase $$BATTNA's I/O buffer for a valid combination. If these specifications are valid, the phase sets the bits of byte MCMODE of the RAS table as shown below:

Bit 0 = HIR,R
HIR in recording mode
Bit 1 = HIR,Q
HIR and ECC in quiet mode
Bit 2 = ECC[,M],R
ECC (for S/370 Mod.145/148 only main (processor) storage) in in recording mode
Bit 3 = ECC[,M],Q
ECC (for S/370 Mod. 145/148 only main (processor) storage) in recording mode
Bit 4 = ECC,C,R
ECC for control storage (S/370 Mod. 145/148 only) in recording mode
Bit 5 = ECC,C,Q
ECC for control storage (S/370 Mod. 145/148 only) in quiet mode
Bit 6 = ECC,C,TH
ECC for control storage (S/370 Mod. 145/148 only) in threshold mode

Following is a list of valid combinations of specifications:

- For any of the processors serviced by this phase:
  - R
  - HIR,{R|Q}
  - HIR[,R],E=eeee[,T=tttt]
  - HIR[,R],T=tttt
  - ECC[,M],{R|Q}
  - ECC[,M],E=eeee[,T=tttt]
  - ECC[,M],T=tttt
- For System/370 Mod 145/148 processors
  - ECC,C,{R|Q}
  - ECC,C,TH
  - ECC,C,E=eeee[,T=tttt]
  - ECC,C,T=tttt

Valid operand specifications result in actions as given below; also indicated is the method of implementation.

R ECC,R HIR,R

R causes error counts in the RAS table to be set to zero: for ECC if specified with ECC, for HIR if specified with HIR, for both if specified alone. It places ECC into recording mode if specified with ECC, it places HIR into recording mode if specified with HIR, it places both into recording mode if specified alone. The counts are reset in fields as follows:

For ECC: ECMACNT in the RAS table.
For HIR: HIRACNT in the RAS table.

Method: Recording mode is set for HIR by reloading control register 14, it is set for ECC by issuing the Diagnose instruction.

ECC,Q HIR,Q

Q places ECC into quiet mode if specified with ECC, it places both HIR and ECC into quiet mode if specified with HIR.

Method: Reloading the control register 14 for setting HIR recording to quiet mode; issuing the Diagnose instruction for setting ECC recording to quiet mode.

C Is valid only for a processor of a System/370 Mod. 145/148. Indicates that the requested action is to apply to the processor's control storage.

M Of significance only if the processor is a System/370 Mod. 145/148. Indicates that the requested action is to apply to main (processor) storage.

E=eeee T=tttt

Modifies the E (error count) or T (time) value either in the ECC field, if specified with ECC, or in the HIR field, if specified with HIR.

Method: Move instructions in the code.

TH Valid only if the processor is a System/370 Mod. 145/148. Places ECC for the control storage into threshold mode.

Method: Issuing the Diagnose instruction.

Note: When HIR is in quiet mode, the phase rejects a command that requests ECC to be placed into recording mode or, on a System/370 Mod. 145/148, into threshold mode.

Phase $$BATTNT – Command Processing (ALTER)

Entry Point: IJBATT40+8.

Function: Processes the ALTER command, which allows the user to change the contents of an area of 16 bytes beginning at the address specified in the command.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used

SCANR2 (in phase $$BATTNA): To get an operand.
RDSTMT (in phase $$BATTNA): To read a statement.
PUTMSG (in phase $$BATTNA): To issue a message.
EXCHPRG (in phase $$BATTNA): To issue a message.

Data Areas Used

- COMREG, the partition communication region (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).
- TRT, a translate table for converting hex numbers to binary.
- IJBXTRTB, a common character translation table.
- TDSERV (FUNC=TDIDSECT), Turbo Dispatcher Information DSECTs

Messages Caused

```
1I37I   UPDATE ON PREFIX PAGE NOT POSSIBLE IN MP ENVIRONMENT
1I38I   SPACE OR PARTITION NOT ACTIVE
1I39t   INVALID SPACE ID OR PARTITION ID
1I41t   INVALID ADDRESS
1I42D   ADDRESS WITHIN SUPERVISOR OR SVA.
1I45D   INVALID ENTRY
1I47I   nn BYTES ONLY CAN BE ALTERED.
1S40t   SYSTEM ERROR, GETVIS  RET.CODE = 0C
1S41I   SYSTEM ERROR, PROGR. CHECK IN LTA.
```

Messages Issued: None.

Input

- The one- to six-byte hexadecimal address specified by the user in the ALTER command.
- At least two but no more than 32 hexadecimal characters that are to replace the data at the specified address.
- The operator's response for the error message.

Output

- Display of the original 1 to 16 bytes of storage at the specified address: both in hexadecimal an EBCDIC presentation.
- The data that replaces the data currently stored at the specified address.
- A message, written to SYSLOG, if an error occurs.

Exit Normal: To phase $$BATTNA.

Exit Error: To program-check exit routine specified in the STXIT macro.

Register Use

1	Pointer to SYSLOG CCB or work register.
2-4	Work registers.
5	Pointer to data or buffer.
6	Work register.
7	Link register.
8	Address of error routine.
9-10	Link registers.
13	Address of partition COMREG.
14	Base register for the root phase, $$BATTNA.
15	Base register for this phase.

Sequence of Operation

INIT: Routine to scan operand.

TRYLOGID: Check a specified LOGID.

TRYSPID: Check a specified space id.

LOADR3: Scan specified address.

ALTDSPL: Display 1 to 16 bytes of storage at the specified address.

READ: Calculate number of bytes to be altered and alter it.

ERR2: Routine to issue error message.

Phase $$BATTNU – Command Processing (DSPLY)

Entry Point: IJBATU41+8.

Function: Processes the DSPLY command to display: displays, on SYSLOG, the contents of 16 bytes of storage, beginning at the specified hexadecimal address.

Functions as overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: None.

Subroutines Used

SCANR2 (in phase $$BATTNA): To get an operand.
EXCHPRG (in phase $$BATTNA): To issue a message.

Data Areas Used

- SYSCOM, the system communication region (in the supervisor).
- Translate tables DSTABLE and DSTABLE1.
- IJBXTRTB, a common character translation table.

Messages Caused

```
1I38I   SPACE OR PARTITION NOT ACTIVE
1I39t   INVALID SPACE ID OR PARTITION ID
1I41t   INVALID ADDRESS.
1I48I   nn BYTES ONLY CAN BE DISPLAYED.
1S41I   SYSTEM ERROR, PROG. CHECK IN LTA.
```

Messages Issued: None.

Input: The one- to six-byte hexadecimal address specified in the command.

Output

- The contents (in hexadecimal) of the 16 bytes of virtual storage at the specified address.
- A message to SYSLOG if an error condition occurs.

Exit Normal: To phase $$BATTNA.

Exit Error: To program-check exit (routine specified in the STXIT macro).

Register Use

1	Pointer to SYSLOG CCB; work register.
2-4	Work registers.
5	Pointer to I/O buffer.
6	Work register.
9	Pointer to message; link register for subroutines.
10	Work register.
11	Work register.
13	Address of partition COMREG.
14	Base register for phase $$BATTNA.
15	Base register for this phase.

Sequence of Operation

INITIAL: Scan the operand.

DSTRYLOG: Check a specified LOGID.

DSTRYSPC: Check a specified space id.

DSLOADR3: Scan specified address and convert it.

DSPLYRTN: Display 16 bytes of main storage on SYSLOG.

Phase $$BATTNY – MODE CE Command Processing

Entry Point: IJBRASB4+10.

Function: Processes a MODE CE,... command: checks the operand specification for validity and sets controls accordingly.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNZ.

Phases Called: None.

Subroutines Used

SCANR (at entry SCANR3) in phase $$BATTNA: To scan the specified operands, one at a time.

EXCPRG (in $$BATTNA): To issue a message on SYSLOG.

Data Areas Used

- ATNCOM, the attention-routine communication area (in phase $$BATTNA).
- SYSCOM, the system communication region (in the supervisor).
- COMREG, the partition communication region (in the supervisor).
- PUB2 table (in the supervisor).
- RFTABLE, the recorder-file-table (in the supervisor).

Messages Caused

1P31I COMMAND IGNORED FOR DEVICE.

Messages Issued: None.

Input: MODE CE,... command.

Output: Fields PUB2FLG and PUB2BBM in PUB2 table (see "Sequence of Operation" below).

Exit Normal: To phase $$BATTNA at label CONTROL.

Exit Error: To phase $$BATTNA at label NVSERR.

Register Use

5-7	Parameter registers for subroutine SCANR.
12	Pointer to the error-message subroutine in phase $$BATTNA.
13	Address of partition COMREG.
14	Base register for this phase.
15	Address of area BATTNA, an alias name of ATNCOM.

Sequence of Operation

IJBRASB4+10: Checks the user's specifications in the MODE CE,... command for validity. Given below is a list of valid specifications; this list shows, in addition, how these options are posted in the PUB2 table:

CE,cuu
Sets the recording mode to "Normal." Causes:

- PUB2FLG, the flag byte in the PUB2 table, to be set to zero.
- PUB2BBM, the bit/byte mask in the PUB2 table, to be set to zero.

CE,cuu,D,byte,bit
Sets the recording mode to "Diagnostic." Causes:

- PUB2FLG to be set to X'40'
- PUB2BBM to be posted with a byte/bit mask (a displacement value from 0 to 31 for the byte part and a value from 0 to 7 for the bit part)

CE,cuu,I,byte,bit
Sets the recording mode to "Intensive." Causes:

- PUB2FLG to be set to X'80'
- PUB2BBM to be posted with a byte/bit mask (see above)

Phase $$BATTNZ – MODE Processing (43xx, S/370-135/138)

Entry Point: IJBRASB5+10.

Function: Processes MODE commands as follows:

- For any serviced processor:
  - MODE CR
  - MODE IR
- For a System/370 Mod. 135/138:
  - MODE STATUS
  - MODE ECC [,RI,Q]
  - MODE STATUS

The phase:

- Writes the recording-mode display message to SYSLOG.
- Sets the recording-mode mask in control register 14 in accordance with the specification given in the command.
- Processes the tape-mode specifications IR and CR, if this is specified in the MODE command.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: See "Exit Normal" below.

Subroutines Used

EXCPRG (in $$BATTNA): To issue a message on SYSLOG.

Data Areas Used

- RFTABLE, RMS recorder file table (in the supervisor).
- COMREG, the partition communication region (in the supervisor).
- RASLINK, the RAS-linkage area (in the supervisor).
- RASTAB, the RAS-table (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).
- ATNCOM, the attention routine communication area.

Messages Caused

```
1P32I   COMMAND IGNORED FOR CPU.
```

Messages Issued: None.

Input: The MODE command as submitted by the user.

Output

- Flags MCMODE and RFFLAGS3 in the RMS table.

Exit Normal

- To phase $$BATTNA at end of processing.
- To phase $$BATTNY to process a MODE CE command.

- To phase $$BATTNQ to process MODE commands if the processor does not have a service diskette for error logging.

Exit Error: To phase $$BATTNA.

Register Use

5 Pointer to parameter list.
9 Link register for subroutines.
14 Base register for this phase.
15 Base register of the root phase, $$BATTNA

Sequence of Operation

LOAD00: Determine the system's mode of operation.

STATUS00: Alter the mode of operation to recording or quiet.

PARAM00: Pass control to $$BATTNY to process other MODE commands.

Phase $$BATTN2 – Command Processing (PRTY, TPBAL)

Entry Point: IJAATP30.

Function: Processes the PRTY and TPBAL commands: displays (on SYSLOG) or alters the currently set processing priority for the system's partitions in response to a PRTY command; displays (on SYSLOG) or alters the current setting of the TP balancing controls in response to a TPBAL command.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH

Phases Called: $IJBPRTY

Subroutines Used:

SCANR (in phase $$BATTNA): To scan the currently processed command.

Data Areas Used

- ATNCOM, attention-routine communication area (in phase $$BATTNA).
- COMREG, the partition communication region (in the supervisor).
- PIB, the program information block (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).

Messages Caused

```
1I0nI   INVALID COMMAND
```

Messages Issued: None – Message 1I0nI is written to SYSLOG by phase $$BATTNA (entry point NVSERR).

Input: One of the following:

- PRTY command without operand (requests a display of dispatching priorities).
- PRTY command with operands (requests the dispatching priority or partition balancing controls to be altered).
- TPBAL command without operand (requests a display of the setting of the TP balancing controls).
- TPBAL command with an operand (requests the TP balancing controls to be altered).

Output: Display, on SYSLOG, of the current control settings for dispatching priority or TP balancing, respectively.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

0-4	Work registers.
5-8	Pointer registers.
9	Link register.
13	Address of the partition COMREG.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

PRTY: Entry point for a PRTY command; passes control to phase $IJBPRTY and prints PRTY string or error message on return.

TPBAL: Examines the specified TP balancing value and set the controls for TP balancing in SYSCOM accordingly.

TPBDISPL: Gets the current TP balancing setting and writes this setting to SYSLOG.

Phase $$BATTN3 – SDAID Program Initiation

Entry Point: $$BATTN3: to process an SDAID command.
IOSMOD: to perform a SYSLOG I/O operation.

Function

1. If an SDAID command is to be processed, call and transfer control to the SDAID program initialization manager.
2. Perform a SYSLOG I/O operation whenever there is a need during SDAID initialization or termination.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: Phases IJSDMSG and IJSDROT.

Subroutines Used

ENDSD: To free the GETVIS space used by phases IJSDMSG, the SDAID message manager, and IJSDROT, the SDAID initialization manager.
IORTN (at entry point IOSMOD): To perform SYSLOG read and write operations during SDAID initialization.
SDLOAD: To load phases IJSDMSG and IJSDROT into the SVA.

Data Areas Used: GDTCB, Global data table (in the SDAID program).

Messages Caused

```
4C07I   function FOR phasename FAILED
```

Messages Issued: None. All messages are written by routine IORTN.

Input

- For function 1: the SDAID command entered at the console.
- For function 2: an indicator for the requested function (read or write) and the address of the I/O area.

Output: Failure message, if necessary (see "Messages Caused" above.)

Exit Normal: For function 1: to phase $$BATTNA.
For function 2: to calling phase (or routine).

Exit Error: Same as for "Exit Normal" above.

Register Use: Is compilation-dependent.

Sequence of Operation

ATN0100: Establishes addressability of control blocks and inserts the address of I/O routine IORTN in table GDTCB; gets supervisor state and key zero.

ATN2000: Loads, in the given sequence, phases IJSDMSG and IJSDROT, if they are not yet in storage.

ATN3000: For an ENDSD command: frees the storage that has been used by phases IJSDMSG and IJSDROT; gives up supervisor state and key zero.

Phase $$BATTN7 – SETDF-Processing Initiation

Entry Point: PHENTRY.

Function: Processes a SETDF command, which requests default operation control values to be set or reset or both. Scans the operand field of the command and loads the update phase or error phase, whichever is required. This phase is used for 3800 printer only.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: Either of the following:

- $$BATTS1 to complete processing the SETDF command.
- $$BATTS2 to build an error message and make it available in the message output area.

Subroutines Used

HEXCUU: To convert the specified cuu to binary.
SCANR: To scan the currently processed command.

Data Areas Used

- SETDFWRK, work area for the phase.
- KEYTABLE, keyword table for the phase.
- VALIDTY, keyword-specification validity table.

Messages Caused: See output below.

Messages Issued: None.

Input: Operand field of the SETDF command.

Output: Error codes to trigger message output as follows:

ERR2 = No parameter specified – message P102I.
ERR3 = Invalid key word or specification – message P103I.
ERR4 = Erroneous delimiter or operand for a keyword – message P104I.
ERR7 = Invalid or no unit specification – message P107I.
ERR9 = Duplicated keyword – message P109I.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

0	Load-point register.
1	Pointer register; work register.
2-4	Work registers.
5	Pointer to keyword or specification.
7	Address of phase-entry point; pointer to entry in error table; length-register for keyword or operand.
8	Scan-indexing register; pointer to entry in validity table; pointer to entry in PUB table (in the supervisor).
9	Link register for the scan routines.
10	Error-code register.
11	Pointer to keyword entry in the keyword table; work register for the MODCTB macro.
12	Base register for the phase.
13	Address of partition COMREG.
14	Base register for the phase.
15	Address of area ATNCOM; return-code register (for macros EXTRACT and MODCTB).

Sequence of Operation

$$BATTN7: When receiving control, the phase expects register 5 to contain the address of the command code and register 7 the length of this code minus 1. The phase:

- Checks for correct unit specification.
- Processes any keywords that are specified, one after the other.
- Checks the specified values for validity: if a value is OK, moves this value into table KEYTABLE; if a value is not OK, sets an appropriate error code. The valid values are the characters A through Z, 0 through 9, $, # and @.
- Passes control to the appropriate phase:

  $$BATTS1 = If no error was encountered.
  $$BATTS2 = If an error was encountered.

Phase $$BATTN8 – LFCB Command Processing, Phase 1

Entry Point: PFOPRFLD.

Function: Initiates the necessary processing in response to an LFCB command.

Functions as an overlay phase of root phase $$BATTNA.

Called By: $$BATTNH.

Phases Called: $$BATTF1.

Subroutines Used

PFGETOPF: To scan an operand and check it for correct length.

Data Areas Used

- FINFAREA, information record.
- FPRMADRT, parameter-address table.
- FPRMVALT, parameter-value table.

Messages Caused: None.

Messages Issued: None.

Input: The operands of the currently processed LFCB command in the command buffer of phase $$BATTNA.

Output: Command-processing information in information records at FINFAREA, FPRMADRT, and FPRMVALT; error information in the byte FPRMECOD.

Exit Normal: To Phase $$BATTF1.

Exit Error: To Phase $$BATTF1.

Register Use

9	Subroutine link register.
10	Contains the return address labeled CONTROL.
12	Contains the error-return address labeled NVSERR.
13	Address of partition COMREG.
14	Base register for the phase.
15	Pointer to area ATNCOM.

Sequence of Operation

PFOPRFLD: Scans the operand field operand after operand; checks the operands for correct syntax and saves information about the specified operands in area FINFAREA.

PFEOPR: Sets the appropriate error indications, if any, into FPRMECOD; loads and passes control to phase $$BATTF1.

Phase $$BATTN9 – BANDID/LUCB Command Processing, Phase 1

Entry Point: PUOPRFLD.

Function: Initiates the necessary processing in response to a BANDID or LUCB command.

Functions as an overlay phase of root phase $$BATTNA.

Called By: Phase $$BATTNH.

Phases Called: $$BATTU1.

Subroutines Used

PUGETOPR: To scan an operand and check it for correct length.

Data Areas Used

- UINFAREA, information record.
- UPRMADRT, parameter-address table.
- UPRMVALT, parameter-value table.

Messages Caused: None.

Messages Issued: None.

Input: The operands of the currently processed BANDID or LUCB command in the command buffer of phase $$BATTNA.

Output: Command-processing information in information records at UINFAREA, UPRMADRT, and UPRMVALT; error information in byte UPRMECOD.

Exit Normal: To Phase $$BATTU1.

Exit Error: To Phase $$BATTNA.

Register Use

9	Subroutine link register.
10	Contains the return address labeled CONTROL.
12	Contains the error-return address labeled NVSERR.
13	Address of partition COMREG.
14	Base register for the phase.
15	Pointer to area ATNCOM.

Sequence of Operation

PUOPRFLD: Scans the operand field, operand after operand; checks the operands for correct syntax and saves information about the specified operands in area UINFAREA.

PUORFCP: Loads and passes control to phase $$BATTU1.

PUEOPR: Sets error indications in UPRMECOD.

Phase $$BATTS1 – SETDF Execution

Entry Point: ENTRYPT1.

Function: Updates the PUB2 entry for the 3800 printer; lists the default values that are set in this entry.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTN7.

Phases Called: None.

Subroutines Used

CONTROL: To read a statement and transfer control to the proper processing routine for processing that statement.
EXCPRG: To initiate an I/O for a device and wait for channel end.

Data Areas Used

- BUFFER, an Input/Output buffer area of 72 bytes.
- COMREG, the partition communication region (in the supervisor).
- KEYTABLE, a keyword table for processing a SETDF command.
- PUBTAB, the physical unit block table (in the supervisor).
- PUB2 table, an I/O control block (in the supervisor).
- SETDFWRK, a work area for processing the SETDF command.

Messages Caused: See "Output" below.

Messages Issued: None.

Input

- SETDF command specifications as stored in area KEYVALUE by phase $$BATTN7.
- Flagbyte KEYFLAG, which indicates whether or not a specific control value is to be updated.

Output: Error codes to trigger message output as follows:

ERR5 = Unknown device address – message P105I.
ERR6 = Device not a 3800 – message P106I.
ERR8 = SETDF failed for the specified device – message P108I.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTS2.

Register Use

12 Base register for the phase.
13 Address of partition COMREG.
14 Base register for SETDF root phase.
15 Base register for ATTN routine.

Sequence of Operation

$$BATTS1: Finds the correct PUB and checks for a 3800 printer device-type code.

Issues the EXTRACT and MODCTB macros to update the associated PUB2 entry.

Lists the PUB2 default values if this is requested.

Passes control to $$BATTS2 if an error is detected by the phase.

Phase $$BATTS2 – SETDF Error Handling

Entry Point: IJBAS240+10.

Function: Builds an error message based on the error code that is passed to the phase when it receives control; makes this message available in an output buffer.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTN7 or $$BATTS1.

Phases Called: None.

Subroutines Used

EXCPRG (in $$BATTNA): To initiate an I/O operation and wait for channel end.

Data Areas Used

- ERRTAB, an error-code table with entries containing: message code, message number, length of message, and message text.

Messages Caused

```
P101I   INVALID MESSAGE CODE
```

Note: The message should never occur.

Messages Issued: Depending on error code.

Input: Error code passed by the calling phase in register 10.

Output: Error message.

Exit Normal: To phase $$BATTNA.

Exit Error: None.

Register Use

10 Parameter (error code) register.
12 Base register for the phase.
13 Address of partition COMREG.
14 Base register for SETDF root phase.
15 Base register for ATTN routine.

Sequence of Operation

FINDMATCH: Find a match between error code (register 10) and ERRTAB. If no match, issue message P101I.

PROCMSG: Process appropriate message referred by error code.

Phase $$BATTU1 – BANDID/LUCB Command Processing, Phase 2

Entry Point: PURCOMBS.

Function: Prepares the system for a UCB-load operation in accordance with user specifications.

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTN9.

Phases Called: Phase $$BATTU2.

Subroutines Used: None.

Data Areas Used

- UINFAREA/INFAREA, information records.
- UPRMADRT, parameter-address table.
- UPRMVALT, parameter-value table.

Messages Caused

```
1B12D   opcode OPERAND n 'erroneous-operand/delimiter' explanation

        In the message, explanation can be one of the following:

        Invalid
        Duplicated
        Not known to system
        Device without FCB
        Device down
```

Messages Issued: None.

Input: Data in areas FINFAREA, FPRMADRT, and FPRMVALT.

Output: Either of the following:

- An error message (see "Messages Caused" above).
- Information record at the label INFAREA (for its layout and content, see this chapter's "Data Area Information" section).

Exit Normal: Phase $$BATTU2.

Exit Error: Phase $$BATTNA.

Register Use

9	Subroutine link register.
10	Contains the return address labeled CONTROL.
12	Contains the error-return address labeled NVSERR.
13	Address of partition COMREG.
14	Base register for the phase.
15	Pointer to area ATNCOM.

Sequence of Operation

PURCOMBS: Returns control to phase $$BATTNA if phase $$BATTN9 found a syntax error.

Looks up the PUB table for an entry that contains the specified device address (in the form cuu); checks whether this device is a printer with a UCB; ensures that the device is ready.

Sets the controls for loading phase $$BATTU2 and loads this phase, unless an error condition had occurred; moves the content of UINFAREA to INFAREA.

PUESYNTX: Builds message 1B12D to indicate a syntax error found by phase $$BATTN8; takes the error exit.

PUESMT: Builds message 1B12D to indicate an error condition encountered by this phase; takes the error exit.

Phase $$BATTU2 – BANDID/LUCB Load Execution

Entry Point: IJBAU240+8.

Function: Executes the LUCB command for a printer with a UCB; uses physical I/O (SYSUSE is not used).

Functions as an overlay phase of root-phase $$BATTNA.

Called By: Phase $$BATTU1.

Phases Called: None.

Subroutines Used:

LISTIO: To initiate the required printer operations.
LOGIO: To write a message to SYSLOG.
LOGIOW: To write a message to SYSLOG and read a response.

Data Areas Used

- BUFFERU2, UCB-image area.
- COMREG, the partition communication region (in the supervisor).
- INFAREA (see "Input" below).
- PUBTAB, the physical unit-block table (in the supervisor).

Messages Caused

```
1B15I   PHASE phasename NOT FOUND
1B16I   UCB LOAD INVALID FOR SPECIFIED PRINTER
1B18A   X'cuu' NEEDS TRAIN=number. STOP PRINTER AND PRESS END
1B20A   INVALID RESPONSE
```

Messages Issued: None.

Input

- UCB-image phase to be loaded into a UCB.
- INFAREA, the last 26 bytes of the B-transient area (for the layout and content of this area, see the this chapter's "Data Area Information" section).

Output

- Message 1B18A, if necessary (see "Messages Caused" above).
- A loaded UCB
- A verification message on the printer if this message was requested by the user.

Exit Normal: To the supervisor via SVC 11.

Exit Error: To the supervisor via SVC 11.

Register Use

0	Pointer to the buffer-load address.
1	Used in macros such as EXCP and FETCH; base register for access to the CCB.
2-3	Work registers.
4	Link register for I/O subroutine call.
5	Base register for access to COMREG.
7	Pointer to (and base for) INFAREA.
8	Printer PUB pointer register.
15	Base register for this phase.

Sequence of Operation

IJBAU240+8: Establish addressability.

READDIR: Read the directory entry of the UCB-image to be loaded; check for correct length.

LOADPHAS: Retrieve the required UCB-image phase from the applicable sublibrary.

PUBLOOP: Wait for the involved printer to become free; load the retrieved UCB image into the UCB.

Phase $$BATT10 – Command Processing (UNLOCK)

Entry Point: IJBAT10.

Function: Processes the UNLOCK command.

Functions as an overlay phase of root-phase $$BATTNA.

Called by: Phase $$BATTNH.

Phases Called: None.

Subroutines Used:

ATTDOIT (in phase $$BATTNA): To perform the action requested by the parameter in register 11.
SCANR (in phase $$BATTNA): To scan the currently processed UNLOCK command.

Data Areas Used

- ATNCOM, attention-routine communication area (in phase $$BATTNA).
- COMREG, the partition communication region (in the supervisor).

Messages Caused

```
1I0nI   INVALID COMMAND
1P51D   UNLOCK COMMAND FOR OWN SYSTEM, NOT ALLOWED
1P52D   RELEASING ALL SYSTEM systemname LOCKS. REPLY 'YES' OR 'NO'
1P54I   UNLOCK COMMAND ABORTED
1P55D   INVALID SYSTEM-ID SPECIFIED
1P56D   SYSTEM ERROR, macroname RET.CODE = nn
```

Messages Issued: None – Messages are written to SYSLOG by phase $$BATTNA:

Message 1I0nI – Via entry point NVSERR.
All others – Via entry point ERRRTN.

Input: The UNLOCK command.

Output: None.

Exit Normal: To phase $$BATTNA.

Exit Error: To phase $$BATTNA.

Register Use

9	Link register for subroutine calls.
13	Address of partition COMREG.
14	Base register for the phase.
15	Address of area ATNCOM.

Sequence of Operation

START: Checks the operand of the UNLOCK command.

GETSYS: Extracts the ID of the system being used and displays a verifying message on the console.

DISPL: Reads the operator's reply.

UNLOCK1: Unlocks the resources locked by the specified system.

Organization Information

This section lists the attention-routine phases alphabetically by their names. The list gives a one-sentence statement about each phase's function; it shows for each phase from which other phase(s) it may receive control (the Called-By column) and which other phase(s) a phase may call or exit to (the Calls/Exits To column).

Phase	Called By	Function	Calls/Exits To
$$BATTF1	$$BATTN8	Processes the LFCB command (sets up the system for the buffer-load operation).	$$BATTF4, $$BATTF5
$$BATTF4	$$BATTF1	Processes the LFCB command (performs the actual buffer-load operation for an IBM 3203 or 5203).	supervisor (via SVC 11)
$$BATTF5	$$BATTF1	Processes the LFCB command (performs the actual buffer-load operation for a printer of the PRT1 class).	supervisor (via SVC 11)
$$BATTNA	$$ABERRZ (in superv.)	Initializes command processing; stores the command into a buffer.	$$BATTNH
$$BATTNB	$$BATTNH	Processes the MSG command (gives control to an operator-communication routine of the indicated partition).	returns control
$$BATTNC	$$BATTNH	Processes the following commands: MPXGTN, IGNORE, LOG, NEWVOL, NOLOG, and PAUSE (suspends or restarts processing in a partition).	returns control
$$BATTND	$$BATTNH	Processes the following commands: LIBSERV, MAP, QUERY and SYSDEF.	returns control
$$BATTNE	$$BATTNH	Processes the ALTER and ONLINE commands (allocates virtual storage to partitions and generates an I/O interrupt for the named device, respectively).	returns control
$$BATTNF	$$BATTNH	Processes the SIZE command (reserves GETVIS space in the specified partition(s)).	returns control
$$BATTNG	$$BATTNH	Processes the BATCH and START commands (initiates or restarts job processing in a foreground partition).	returns control

Phase	Called By	Function	Calls/Exits To
$$BATTNH		Selects the appropriate command processing phase.	$$BATTNB, $$BATTNC, $$BATTND, $$BATTNE, $$BATTNF, $$BATTNG, $$BATTNK, $$BATTNP, $$BATTNT, $$BATTNU, $$BATTNZ, $$BATTN0, $$BATTN2, $$BATTN3, $$BATTN7, $$BATTN8, $$BATTN9, $$BATT10
$$BATTNI	$$BATTNH	Processes the MAP command (displays a map of virtual storage —— in ECPS:VSE mode).	returns control
$$BATTNK	$$BATTNH	Processes the SETMOD command (changes the operating mode of an IBM 8809).	returns control
$$BATTNP	$$BATTNH	Processes the FREE and RESERV commands (frees a disk volume that is reserved or reserves a disk volume that is free).	returns control
$$BATTNQ	$$BATTNZ	Processes the MODE command without the CE operand (analyzes the specified operands).	$$BATTNR, $$BATTNS
$$BATTNR	$$BATTNQ	Processes a MODE STATUS command (displays the recording-mode setting on the SYSLOG device).	returns control (to $$BATTNA)
$$BATTNS	$$BATTNQ	Processes a MODE STATUS command (completes analysis of operands and sets recording controls according to user specifications).	returns control (to $$BATTNA)
$$BATTNT	$$BATTNH	Processes the ALTER command (changes 16 bytes of virtual storage).	returns control
$$BATTNU	$$BATTNH	Processes the DSPLY command (displays 16 bytes of virtual storage).	returns control
$$BATTNY	$$BATTNZ	Processes the MODE command with the CE operand (sets recording controls as requested).	returns control (to $$BATTNA)
$$BATTNZ	$$BATTNH	Processes the MODE command (analyzes the command; initiates processing the command).	$$BATTNQ $$BATTNY

Phase	Called By	Function	Calls/Exits To
$$BATTN2	$$BATTNH	Processes the PRTY and TPBAL commands (displays or changes the processing priorities of the system's partitions).	returns control
$$BATTN3	$$BATTNH	Processes the SDAID command (causes the SDAID program to be loaded into the system's GETVIS area).	SDAID program
$$BATTN7	$$BATTNH	Processes the SETDF command (initiates the setting or resetting of 3800 printer default values).	$$BATTS1, $$BATTS2
$$BATTN8	$$BATTNH	Processes the LFCB command (analyzes the command; initiates oaring a printer's forms-control buffer).	$$BATTF1
$$BATTN9	$$BATTNH	Processes the BANDID/LUCB command (analyzes the command; initiates loading a printer's universal character-set buffer).	$$BATTU1
$$BATTS1	$$BATTN7	Processes the SETDF command (sets or resets a default as requested; lists the new or current setting).	returns control (to $$BATTNA)
$$BATTS2	$$BATTN7	Processes the SETDF command (handles error conditions).	returns control (to $$BATTNA)
$$BATTU1	$$BATTN9	Processes the BANDID/LUCB command (sets up the system for the buffer-load operation)	$$BATTU2
$$BATTU2	$$BATTU1	Processes the BANDID/LUCB command (performs the buffer-load operation).	supervisor (via SVC 11)
$$BATT10	$$BATTNH	Processes the UNLOCK command (releases resources that were locked by another (sharing) system).	return control

Data Area Information

This section provides a summary of the key data areas used by the attention routines; these areas are given in alphabetic order of their names. Displacements of fields within an area are given in decimal in the first (or only) displacement column; if meaningful, they are also given in hexadecimal in a second displacement column.

ATNCOM – Attention-Routine Communication Area

The area ATNCOM at the label BATTNA in phase $$BATTNA is used for phase to phase communication during the processing of an attention-routine command. ATNCOM is the name of the macro used to generate a DSECT in each of the attention-routine phases that access this area. The layout and contents of the area are:

Displacement		
0-7	0-7	Current phase name.
8-17	8-11	Reserved.
18-25	12-19	Command verb (or currently processed command).
26-151	1A-97	Input/Output buffer.
152-159	98-9F	Save area for saving pointers during scan operations.
		BRANCH TABLE
160-163	A0-A3	Address of the Detach AR entry.
164-167	A4-A7	Address of the main line (CONTROL) entry.
168-173	A8-AD	Execute instruction used to clear the buffer (CLRBUF).
174-177	AE-B1	Reserved.
178-181	B2-B5	Address of the EXCP (execute channel program) routine.
182-185	B6-B9	Address of the first-operand-scan (SCANR2) routine.
186-189	BA-BD	Address of the other-operand-scan (SCANR3) routine.
190-193	BE-C1	Address of the error-message output (ERRRTN) routine.
		CONSTANTS (WITHIN THE BRANCH TABLE)
194	C2	Operand number (OPNUMB).
195	C3	Scan-stop character (SCNSTP).
		END OF CONSTANTS
196-199	C4-C7	Address of multi-purpose (ATTDOIT) routine.
		END OF BRANCH TABLE
200-207	C8-CF	CCW for a SYSLOG-read operation.
208-215	D0-D7	CCW for a SYSLOG-write operation.
216-231	D8-E7	CCB for a SYSLOG-read operation.
232-247	E8-F7	CCB for a SYSLOG-write operation.
248-251	F8-FB	Address of the dump delimiter.

FINFAREA/INFAREA – FCB-Load Information Record

Area FINFAREA is built by phases $$BATTN8 and $$BATTF1; it is used for passing control information to the appropriate load-execution phase, $$BATTF4 or $$BATTF5. The load execution-phase picks up this information from INFAREA to which the contents of FINFAREA is moved before $$BATTF1 calls the load-execution phase. The layout of the area is as follows:

Displ.

0-7	Specified phase name.
8-11	Form number, if specified, or blank.
12	Reserved.
13	Number of lines per inch, if specified, or blank.
14-15	Hex value of specified unit address.
16-19	Address of PUB.
20-22	Specified unit address as printable characters.
23	Reserved.
24	Information byte with bits indicating the following if they are set to 1:

Bits

0	Reserved.
1	FOLD was specified.
2	NOCHK was specified.
3	NULMSG was specified.
4	FORMS=xxxx was specified.
5	LPI=n was specified.
6-7	Reserved.

25	19 X'00'; used to clear table FPRMADRT.

FPRMADRT and UPRMADRT – FCB/UCB-Load Address Table

FPRMADRT, the FCB-load address table, is used by phase $$BATTN8; table UPRMADRT, the UCB-load address table, is used by phase $$BATTN9. The layout and content of these tables, which are the same in either phase, are shown below:

Displ.

0-3	Address of keyword.
4-5	Length of keyword.
6-7	Sequence number of operand.
8	Reserved.
9	Delimiter character.
10-13	Address of keyword value.
14-15	Length of keyword value.
16-17	Sequence number of operand.
18	Reserved.
19	Delimiter character.

FPRMVALT and UPRMVALT – FCB/UCB-Load Specification-Value Table

FPRMVALT, the FCB load specification-value table, is used by phase $$BATTN8; table UPRMVALT, the UCB load specification-value table, is used by phase $$BATTN9. The layout and content of these tables, which are the same in either phase, are shown below:

Displ.

0-9 Specified keyword.

10-19 Specified keyword value.

FPRMECOD and UPRMECOD – FCB/UCB-Load Error-Code Byte

The FCB-load error-code byte FPRMECOD is used by phase $$BATTN8, the UCB-load error-code byte UPRMECOD by phase $$BATTN9. The meaning of the bits of the bytes are as follows:

01 = Specified device is down.
03 = Specified device has no FCB (for an FCB load operation) or no UCB (for a UCB load operation).
07 = No PUB entry was found for the specified channel and unit addresses.
10 = Keyword is specified more than once.
30 = Keyword value has wrong length or the value's delimiter is erroneous.
70 = Length of positional operand or of keyword is incorrect or the operand's delimiter or the keyword is erroneous.
F0 = Operand is invalid.

SLPL – SETLIMIT-Macro Parameter List

The SETLIMIT parameter list is built by phase $$BATTNF for giving a new size value to the corresponding partition.

Displ.

0-1 Specifies the partition ID 'BG' or 'Fn'.

2-3 Specifies the K-bytes amount of virtual storage.

UINFAREA/INFAREA – UCB-Load Information Record

Area UINFAREA is built by phases $$BATTN9 and $$BATTU1 to pass control information to the load-execution phase, $$BATTU2. The load execution-phase picks up this information from INFAREA. The layout of the area is:

Displ.

0-7	Specified phase name.
8-13	Train number, if specified, or blank.
14-15	Hex value of specified unit address.
16-19	Address of PUB.
20-22	Specified unit address as printable characters.
23	Reserved.
24	Information byte with bits indicating the following if they are set to 1:

0	Reserved.
1	FOLD was specified.
2	NOCHK was specified.
3	NULMSG was specified.
4	TRAIN=xxxx was specified.
5	LPI=n was specified.
6-7	Reserved.

25	19 X'00'; used to clear table UPRMADRT.

Chapter 3. Checkpoint/Restart Routines

Introduction

For program that runs for a long time, it may be desirable to take checkpoint information at regular intervals during execution of that program. This information, a one-to-one mapping of the contents of the partition in which the program runs, reflects the program's I/O status and the contents of the general registers at the time of the checkpoint. Thus, a "checkpointed" program can be restarted at one of the checkpoints should the processing for that program end before normal end-of-job (for example, because of a power failure).

The routines that control taking a checkpoint and, to some extent restarting a checkpointed program, are contained in logical transients. The checkpoint and restart routines are loaded for execution into the logical transient area (LTA), which is a part of the supervisor.

Checkpoint/restart does not support data spaces; that is, data spaces that a program may access are not recorded during checkpoint requests.

Checkpoint/restart does not support the 31-bit environment; the macro is cancelled when issued from a partition that crosses the 16 MB line.

Taking a Checkpoint

To cause a checkpoint to be taken, the user codes, in his program, a CHKPT macro. Up to 9999 checkpoints may be written for a program, and each checkpoint is assigned an identification number. The user can choose to have the checkpoints for a program to be written either onto tape or on disk.

If checkpoint data is to be written into a separate tape file with standard labels (defined by the user with a DTFPH macro), the labels must either be checked by an OPEN macro or bypassed by an MTC command before the first checkpoint is written.

If checkpoint data is to be written onto disk, the user must define a file (by a DTFPH macro with MOUNTED=SINGLE, TYPEFLE=OUTPUT specified) and open this file before the application program requests the first checkpoint to be written.

A checkpoint cannot be taken if the checkpointed program uses a gating mechanism such as the system's track hold facility.

External Input: To take a checkpoint, the checkpoint routines require user definitions as operands of the CHKPT macro. The format of this macro is:

```
CHKPT     SYSnnn,{rest-addr|(r1)}
          [,end-address|,(r2)]
          [,tpointer|,(r3)][,dpointer|,(r4)]
          [,filename|,(r5)]
```

For an explanation of the operands, see *VSE/ESA System Macros Reference*.

External Output: The checkpoint data, which contains the following in the given sequence:

- A header record (a header and save record if on tape)
- 3800 printer-information records
- Extent-information records
- PFIX-information records
- Program-dump records
- A trailer record, if on tape

The checkpoint routines do not save any of the following:

- The SVA and the system GETVIS area (if a checkpointed program uses phases in the SVA or occupies space in the system GETVIS area, the user must make sure that the same locations are occupied by the same data on restart)
- The floating-point registers
- Linkages to the checkpointed program set up by the STXIT or the SETPFA macro
- Any XECBs defined by the checkpointed program
- Any timer values set by the SETIME macro
- The program mask in the checkpointed program's PSW

If any of this information is required for a restart, the user must define it or supply it for this restart.

Restarting a Program

After having encountered a // RSTRT statement, the job control program initiates program-restart processing. Job control:

- Reads the applicable checkpoint data's header record.
- Reads the 3800-printer information record (if present) and the extent-information records.
- Verifies the partition's boundaries.
- Clears the partition.
- Restores the partition's communication region (COMREG) to its status when the checkpoint was taken.

The phase $$BRSTRT restores PFIX information and the application program area; it calls phase $$BRSTR2 if DASD verification or tape repositioning is necessary.

External Input: A restart parameter list built by the job control phase $JOBCTLK based on the // RSTRT statement submitted by the user.

External Output: Other than system messages, the restart routines generate no external output.

Design Information

This section provides an overview of the control flow of both the checkpoint and the restart logical transients. It includes descriptions of the logical-transient phases that make up the checkpoint-restart routines.

The expansion of a CHKPT macro in an application program includes a parameter list whose layout is shown in this chapter's "Data Area Information" section. This expansion issues an SVC 2 instruction with registers 0 and 1 containing pointers as follows:

Reg. 0 = CHKPT macro parameter list (for the layout of this parameter list, a field of 22, 28 or 32 bytes in length, refer to this chapter's "Data Area Information" section).

Reg. 1 = Pointer to the name of logical transient to be called ($$BCHKPD or $$BCHKPT).

Control Flow for Taking a Checkpoint

Figure 4 on page 62 and Figure 5 on page 63 give an overview of the control flow for taking a checkpoint on disk and tape, respectively. The various transients involved in taking a checkpoint depend on control information and data passed by the calling phase. Such information and data is passed primarily in the checkpoint common work area, the last 356 bytes of the LTA. The area has the label COMDSCT if the checkpoint records are written to tape; its label is COMDSCTD if the checkpoint records are written to disk.

All messages caused by the routines for taking checkpoint records are written by one phase, $$BRMSG1.

Control Flow for Restarting a Checkpointed Program

Figure 6 on page 64 gives an overview of the control flow for restarting a checkpointed program.

The logical transients needed for the restart of a checkpointed program function as an extension of the job control phase $JOBCTLB, which invokes the restart logical transients.

The logical transient $$BRSTRT, which receives control from $JOBCTLB, expects a pointer to a parameter list in register 0; it uses the control information contained in the parameter list created by the expansion of the CHKPT macro in the checkpointed program's area.

All messages caused by the routines for restarting a checkpointed program are written by one phase, $$BRMSG2.

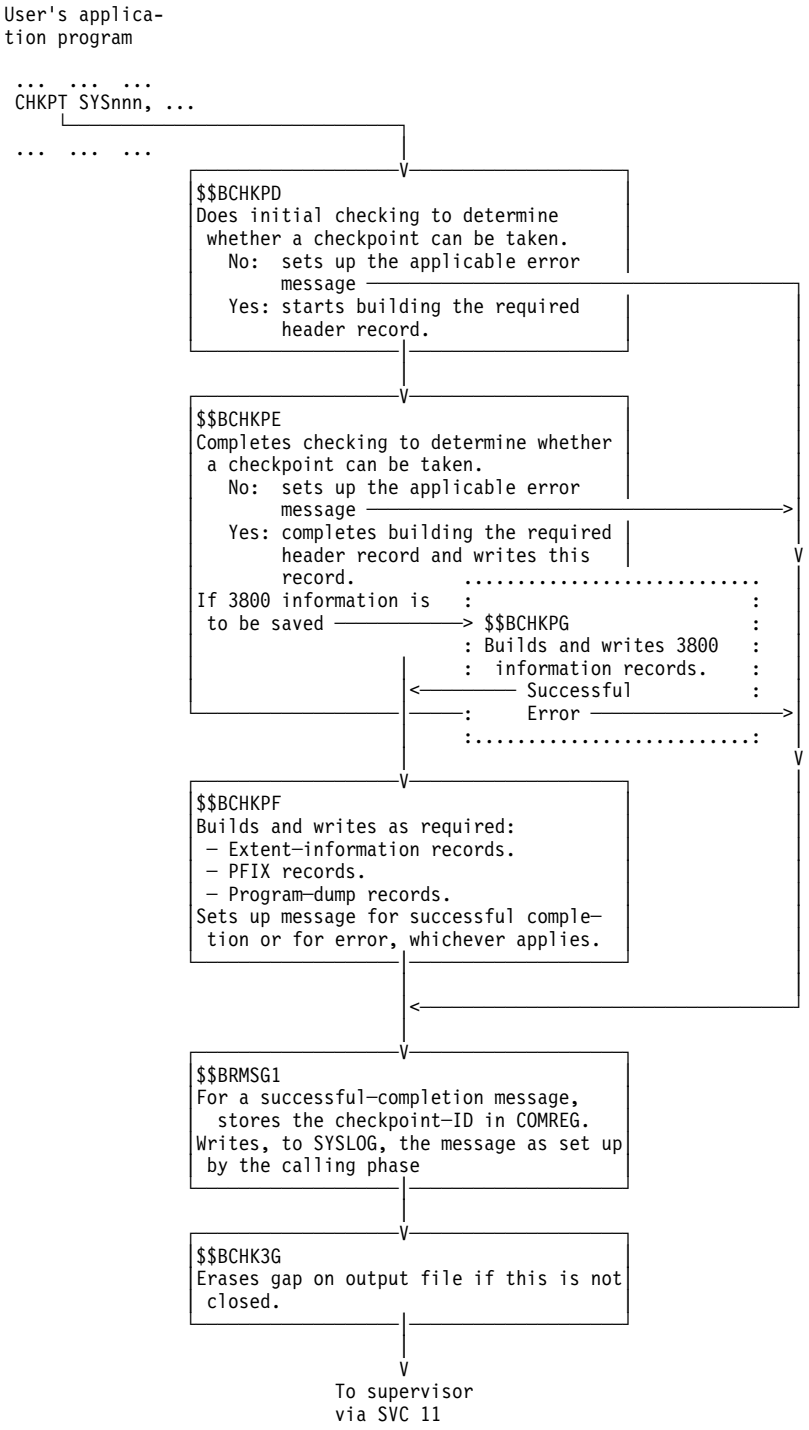


Figure 4. Control Flow for Taking a Checkpoint on Disk

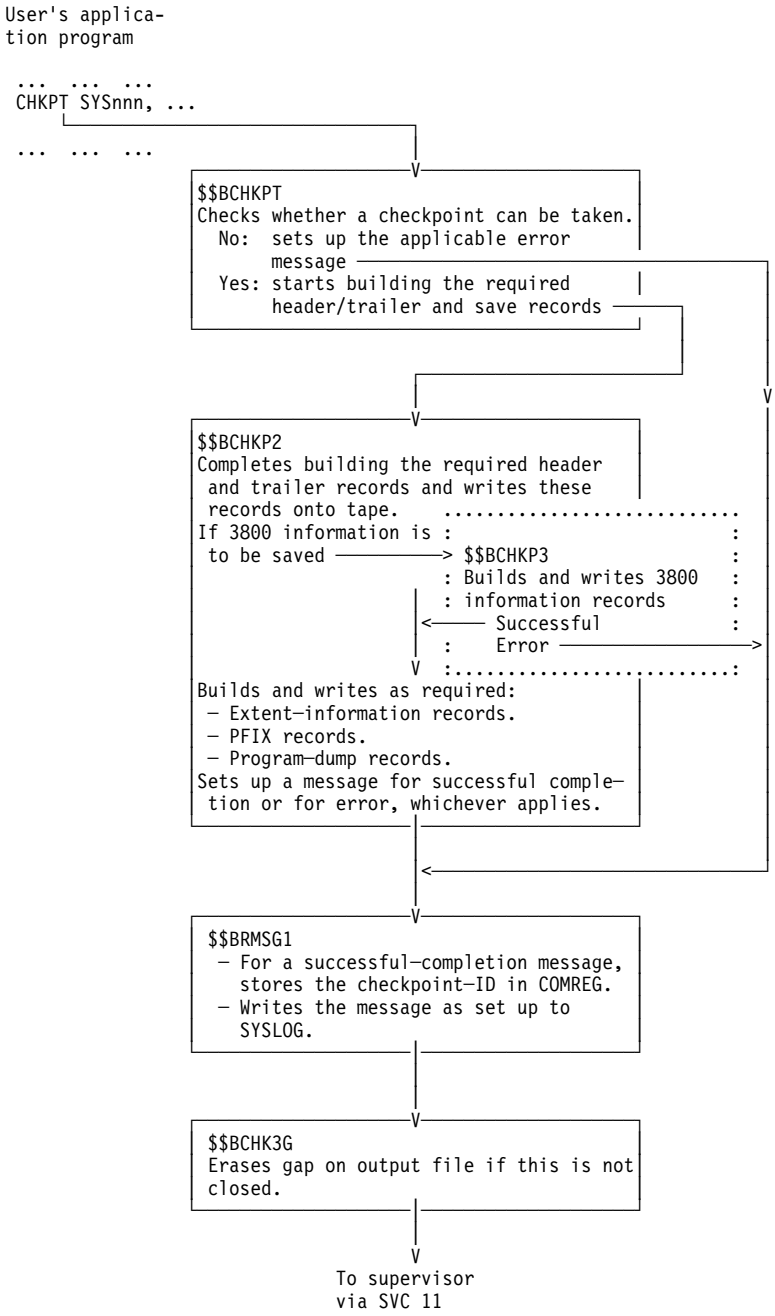
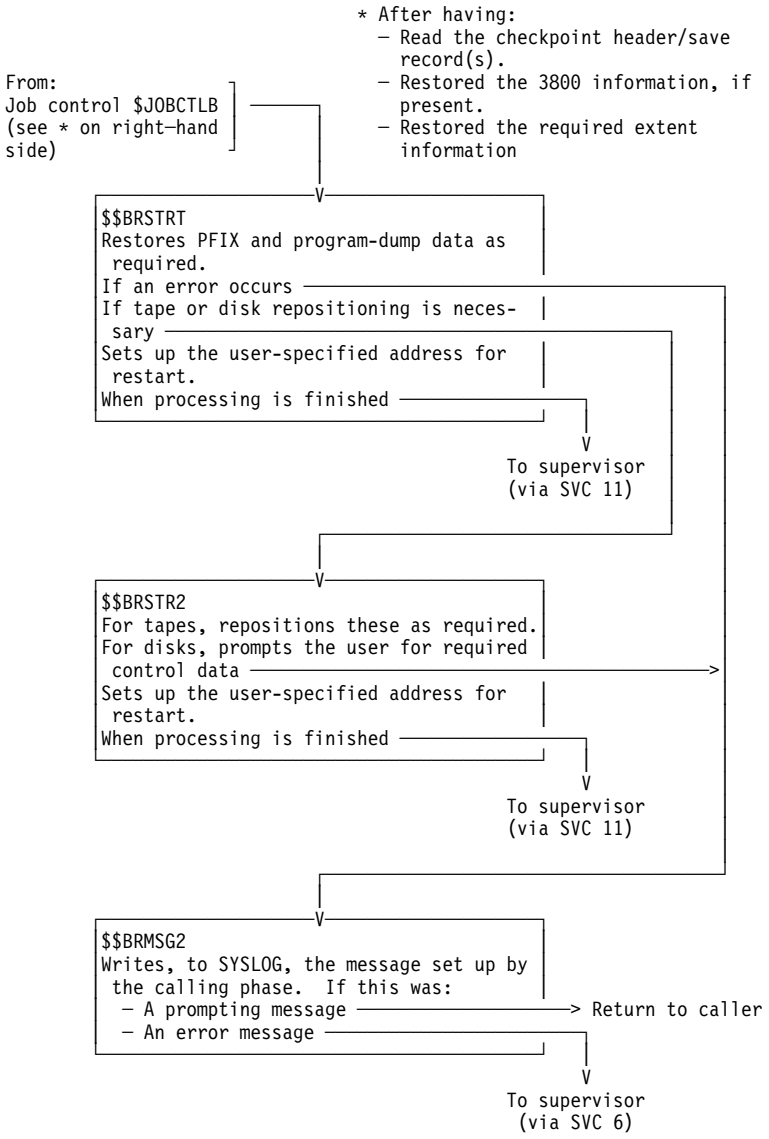


Figure 5. Control Flow for Taking a Checkpoint on Tape



Note: An SVC 6 requests the job to be canceled.

Figure 6. Control Flow for Restarting a Checkpointed Program

Phase $$BCHKPD – Disk Checkpoint Phase 1

Entry Point: IJBCKD40.

Function: Checks whether a checkpoint can be taken and, if so, starts building the checkpoint header records.

Called by: CHKPT macro expansion via SVC 2.

Phases Called: None.

Subroutines Used: None.

Data Areas Used

- The CHKPT macro parameter list.
- COMREG, the partition communication region (in the supervisor).
- COMDSCTD, the disk-checkpoint common work area.

Messages Caused

```
0C04I   INVALID END ADDRESS SPECIFIED. CHECKPOINT IGNORED
0C05I   CHKPT DTFPH IS NOT OPEN. FILE=filename. CHECKPOINT IGNORED
0C06I   CHKPT DTFPH MOUNTED=ALL. FILE=filename. CHECKPOINT IGNORED
0C07I   CHKPT DTFPH NOT OUTPUT. FILE=filename. CHECKPOINT IGNORED
0C08I   CHKPT UNIT NOT A VALID DISK. SYSxxx=cuu. CHECKPOINT IGNORED
0C10I   SUBTASK ISSUED CHKPT. CHECKPOINT IGNORED
0C11I   SUBTASKS ATTACHED. CHECKPOINT IGNORED
0C12I   TRACKS HELD. CHECKPOINT IGNORED
0C14I   CHKPT DEVICE NOT ASSIGNED SYSxxx. CHECKPOINT IGNORED
0C17I   INTERNAL CHKPT ERROR IN $$BCHKxx. macroname FAILED. RC=X'nn'
0C19I   CHKPT DEVICE ERROR SYSxxx=cuu. CHECKPOINT IGNORED
0C20I   PFIX OUTSIDE ALLOCR AREA. CHECKPOINT IGNORED
```

Messages Issued: None.

Input: The CHKPT macro parameter list.

Output: Partially completed checkpoint header record in the disk-checkpoint common work area.

Exit Normal: To phase $$BCHKPE.

Exit Error: To phase $$BRMSG1.

Register Use

1	Address of the checkpoint-file DTF block; work register.
2	Paramter register for $$BRMSG1: macro-return code.
2-8	Work registers.
9	Address of the CHKPT-macro parameter list; address of COMDSCTD, the common disk-checkpoint work area.
10	Base register for access to COMREG.
11	Base register for access to SYSCOM.
12	Base register for the phase.
15	Macro return-code register.

Sequence of Operation

CHKPD000: Checks to ensure that:

- A main task has issued the checkpoint request.
- No subtasks are attached to this main task.
- No page is PFIXed outside the ALLOCR area.
- No tracks (or FBA blocks) are held.
- The file into which the checkpoint is to be written has been properly defined (DTFPH MOUNTED=SINGLE TYPEFLE=OUTPUT) and is open.
- The specified checkpoint device is a supported disk.

CHKPD000: Looks for a user defined program-area end address in the CHKPT-macro generated parameter list (pointed to by register 0); uses the partition's logical-end address as contained in field PPEND of COMREG if the user did not specify a program-area-end address.

If the GETVIS flag in COMREG is on, sets the program-area end address to the upper boundary of the checkpointed program's partition.

CHKPD290: Calculates the required number of:

- Program-dump records of 2K bytes in length (except for a checkpoint file on a 2311 or a 3340, in which case the program-dump records have a length of 1K bytes).
- Extent-information records of 256 bytes in length (an eight-byte header plus up to fifteen 16-byte entries, one per extent).
- 3800-printer information records of 76 bytes in length.

Stores these values, including the partition's temporary boundaries and its save area, in the checkpoint header, which is being built (at label OUTAREA) in the common work area of the disk checkpoint phases.

CHKPD660+8: Retrieves and stores, in the checkpoint header record, the SYSCAT volume's serial number if that volume exists.

Phase $$BCHKPE – Disk Checkpoint Phase 2

Entry Point: IJBCKE50.

Function: Completes checking for checkpoint prerequisites and completes building the checkpoint header record; writes this record into the checkpoint file.

Called by: Phase $$BCHKPD.

Phases Called: None.

Subroutines Used:

IOCKD: To write to a CKD disk.
IOFBA To write to an FBA disk.
The subroutines issue a write request on EXCP level and expect the referenced channel program to be updated and to meet the existing I/O requirements.

Data Areas Used: COMDSCTD, the checkpoint common work area.

Messages Caused

```
0C03I   I/O REQUEST PENDING ON TP DEVICE. CHECKPOINT IGNORED
0C09I   INSUFF. SPACE ALLOCATION. FILE=filename. CHECKPOINT IGNORED
0C17I   INTERNAL CHKPT ERROR IN $$BCHKxx. macroname FAILED. RC=X'nn'
```

Messages Issued: None.

Input: A pointer to COMDSCTD, the common disk-checkpoint work area.

Output

- The checkpoint-header record.
- The channel program for writing into the checkpoint file.

Exit Normal

- To phase $$BCHKPG if 3800-printer information records are required.
- To phase $$BCHKPF if 3800-printer information records are not required.

Exit Error: To phase $$BRMSG1.

Register Use

1	Pointer to phase name (for SVC 2); work register.
2	Macro return-code (for $$BRMSG1) or zero.
3-5	Work registers.
6	Address of the user's DTFPH block (for $$BRMSG1); work register.
7	Work register.
8	Base register for access to the user's DTFPH block.
9	Address of area COMDSCTD, the common disk-checkpoint work area.
10	Address of the partition COMREG.
11	Address of SYSCOM.
12	Base register for the phase.
13	Main-line place holder; subroutine link register.
14	Subroutine link register; work register.
15	Macro return-code register.

Sequence of Operation

CHKPE000: Computes the number of required PFIX information records.

Checks whether sufficient space is available in the checkpoint disk file; sets the begin-write address to the beginning of that file if the available free space is insufficient to accommodate the currently processed checkpoint request.

CHKPE240: Ensures that no I/O request is pending for a telecommunication device that belongs to the checkpointed partition.

CHKPE360: Completes building the checkpoint header record: stores in this record the information needed from COMREG – the partition allocations; sets the checkpoint-identifier into COMREG.

CHKPE440: Writes the header record into the checkpoint file.

Phase $$BCHKPF – Disk Checkpoint Phase 3

Entry Point: IJBCKF40.

Function: Builds checkpoint records that have to follow the 3800-printer information record(s), if present, or the header record; writes these records into the checkpoint file on disk.

Called by

- Phase $$BCHKPE if no 3800-printer information records are required.
- Phase $$BCHKPG if 3800-printer information records are required.

Phases Called: None.

Subroutines Used

IOCKD: To write to a CKD disk.
IOFBA: To write to an FBA disk.

The subroutines issue a write request on EXCP level and expect the referenced channel program to have been updated before they receive control.

Data Areas Used

- The system-extent area
- The PFIXCHPT-macro work area
- COMDSCTD, the checkpoint common work area

Messages Caused

```
0C00I   CHKPT NO. number WAS TAKEN ON SYSxxx=cuu
```

Messages Issued: None.

Input: A pointer to the checkpoint common work area.

Output: Checkpoint records as follows:

- Extent-information records (a header record followed by extent entries – see the chapter's "Data Areas" section), if they are required (see "Sequence of Operation" below).
- PFIX information records.
- Program-dump records. A program-dump record has a length of 1K bytes if the checkpoint file is on a 2311 or a 3340; it has a length of 2K bytes otherwise.

Exit Normal: To phase $$BRMSG1 to write, to SYSLOG, a successful-completion message.

Exit Error: To phase $$BRMSG1 to write, to SYSLOG, an appropriate error message.

Register Use

0	Macro parameter register.
1	Macro parameter register; address of the DTFPH for the checkpoint file; pointer to the phase name (for issuing an SVC 2).
2	PFIXCHPT macro return-code; work register.
3	Work register.
4	Pointer to logical unit of the checkpoint device (for call of $$BRMSG1); work register.
5	Message code (for call of $$BRMSG1); work register.
6-8	Work registers.
9	Address of area COMDSCTD, the common disk-checkpoint work area.
10	Address of the partition COMREG.
11	Work register.
12	Base register for the phase.
13	Subroutine link register.
14	Subroutine link register; work register.
15	Subroutine link register.

Sequence of Operation

CHKPF000: If DASD file protection is active, builds and writes extent-information records for the SYSCAT volume and for extents assigned to programmer logical units used by the checkpointed program. Retrieves extent information from the system-extent area.

CHKPF180: Builds and writes PFIX information records: retrieves the required information by appropriate PFIXCHPT macros; this macro returns the requested information in a supplied work area.

Writes program-dump records into the checkpoint file.

Phase $$BCHKPG – Disk Checkpoint Phase 4

Entry Point: $$BCHKPG.

Function: Builds 3800-printer information records and writes them into the checkpoint file.

Called by: Phase $$BCHKPE.

Phases Called: None.

Subroutines Used

IOCKD: To write to a CKD disk.
IOFBA: To write to an FBA disk.

The subroutines issue a write request on EXCP level and expect the referenced channel program to be updated before they receive control.

Data Areas Used

- COMDSCTD, the checkpoint common work area.
- The LUB table (in the supervisor).
- QWORK, the 3800-printer information-record build and output area.

Messages Caused

```
0C16I   QSETPRT FAILED. RC=X'oonnxxrr' SYSxxx=cuu. CHECKPOINT IGNORED
```

Messages Issued: None.

Input: A pointer to the checkpoint common work area.

Output: The 3800-printer information record.

Exit Normal: To phase $$BCHKPF.

Exit Error: To phase $$BRMSG1.

Register Use

1-3	Work registers.
4	Logical-unit number (for insertion in a message text).
5	Message-code (for $$BRMSG1); work register.
6-7	Work registers.
8	Address of the DTFPH block of the checkpoint file.
9	Address of area COMDSCTD, the common disk-checkpoint work area.
10	Base register for access to COMREG.
11	Main-line place holder.
12	Base register for the phase.
13-14	Link registers for subroutine call.
15	Macro return-code.

Sequence of Operation

CHKPG005: Finds the logical units to which a 3800 printer is assigned and sets up the required work and I/O areas; retrieves the 3800-printer information using the QSETPRT macro and builds the required information records, one for each assigned 3800 printer.

Writes each 3800-printer information record into the checkpoint file.

Phase $$BCHKPT – Tape Checkpoint Phase 1

Entry Point: IJBCKT40.

Function: Checks whether a checkpoint can be taken and, if so, starts building the checkpoint header/save records.

Called by: CHKPT macro expansion via SVC 2.

Phases Called: $$BCHKP2.

Subroutines Used: None.

Data Areas Used

- The CHKPT-macro parameter list.
- COMREG, the checkpointed partition's communication region (in the supervisor).
- COMDSCT, the common tape checkpoint work area.

Messages Caused

```
0C02I   CHKPT LOGICAL UNIT NOT TAPE SYSxxx=cuu. CHECKPOINT IGNORED
0C03I   I/O REQUEST PENDING ON TP DEVICE. CHECKPOINT IGNORED
0C04I   INVALID END ADDRESS SPECIFIED. CHECKPOINT IGNORED
0C10I   SUBTASK ISSUED CHKPT. CHECKPOINT IGNORED
0C11I   SUBTASKS ATTACHED. CHECKPOINT IGNORED
0C12I   TRACKS HELD. CHECKPOINT IGNORED
0C14I   CHKPT DEVICE NOT ASSIGNED SYSxxx. CHECKPOINT IGNORED
0C15I   CHKPT LOGICAL UNIT INVALID SYSxxx. CHECKPOINT IGNORED
0C17I   INTERNAL CHKPT ERROR IN $$BCHKxx. macroname FAILED. RC=X'nn'
0C20I   PFIX OUTSIDE ALLOCR AREA. CHECKPOINT INGORED.
```

Messages Issued: None.

Input: The CHKPT-macro parameter list.

Output: Partially completed checkpoint header/save records in the common tape checkpoint work area.

Exit Normal: To phase $$BCHKP2.

Exit Error: To phase $$BRMSG1.

Register Use:

0 Pointer to CHKPT macro parameter list.
1 Return code from RUNMODE and GETFLD macros; pointer to phase name (for issuing an SVC 2).
2 Macro return-code (for insertion into a message text).
3 Parameter register for $$BRMSG1: macro- or phase-ID; EXTRACT-macro parameter register; work register.
4 Logical unit number (for insertion into a message text); EXTRACT macro parameter register; work register.
5 Parameter register for $$BRMSG1: message-code; work register.
6 EXTRACT macro parameter register; work register.
7 Main-line place holder; work register.
8 Address of the PIB; work register.
9 Address of the CHKPT-macro parameter list; address of COMDSCT, the common tape-checkpoint work area.
10 Address of the partition COMREG.
11 Address of the area SYSCOM.
12 Base register for the phase.
15 Macro return-code.

Sequence of Operation

CHKPT000: Checks to ensure that:

- A main task has issued the checkpoint request.
- No subtasks are attached to this main task.
- No page is PFIXed outside the ALLOCR area.
- No I/O is pending for a telecommunication device.
- No tracks (of FBA blocks) are held.
- The checkpoint device is a tape.

CHKPT090: Looks for a user defined program-area-end address in the CHKPT-macro generated parameter list (pointed to by register 0); uses the partition's logical-end address as contained in field PPEND of COMREG if the user did not specify a program-area-end address.

If the GETVIS flag in COMREG is on, sets the program-area-end address to the upper boundary of the checkpointed program's partition.

CHKPT150+8: Calculates the required number of:

- Program-dump records, which are 16K bytes long.
- PFIX information records, which are up to 256 bytes long.
- Extent-information records, which are up to 256 bytes long.
- 3800-printer information records of 256 bytes in length.

Stores these values in the checkpoint header that is being built in the common work area for tape checkpoint phases.

CHKPT500: Retrieves and stores, in the checkpoint save record, required COMREG information: the updated checkpoint-identifier and the partition allocations.

Phase $$BCHKP2 – Tape Checkpoint Phase 2

Entry Point: IJBCK240.

Function: Completes building the tape-checkpoint header/save and trailer records and all other checkpoint records; writes these records onto tape mounted on the specified tape drive.

Called by: Phase $$BCHKPT.

Phases Called: $$BCHKP3 to build and write 3800-information records as required.

Subroutines Used

IORT: To write data onto tape as follows:

- Header and trailer records in the tape mode set by the user.
- All other checkpoint records with the convert feature on if the involved tape unit has this feature.

The density defined for the checkpoint tape drive remains unchanged.

If an end-of-reel condition occurs, the subroutine causes the tape to be backspaced to its original position and the header record to be scratched.

Data Areas Used

- COMDSCT, the common tape checkpoint work area.
- COMREG, the checkpointed partition's communication region (in the supervisor).
- The extent-information record-build and output area.
- The PFIXCHKPT macro work area.
- The PFIX-information record-build and output area.
- The system-extent area as set up by the supervisor.

Messages Caused

```
0C00I   CHKPT NO. nnnn WAS TAKEN ON SYSxxx=cuu
0C13I   INSUFF. SPACE FOR CHKPT ON SYSxxx=cuu. CHECKPOINT IGNORED
0C17I   INTERNAL CHKPT ERROR IN $$BCHKxx. macroname FAILED RC=X'nn'
```

Messages Issued: None.

Input

- A pointer to the common tape checkpoint work area.
- The system-extent area.

Output: All records that make up the complete checkpoint data, except the 3800-information record; that record, if required, is built and written onto tape by phase $$BCHKP3.

Exit Normal: To phase $$BRMSG1.

Exit Error: To phase $$BRMSG1.

Register Use

0	Macro parameter register.
1-2	Work registers.
3	Phase- or macro-ID (for $$BRMSG1); work register.
4	Logical unit number (for $$BRMSG1); work register.
5	Pointer to CCW on subroutine call.
6-8	Work registers.
9	Address of COMDSCTD, the common disk checkpoint work area; pointer to CCB and channel-program area.
10	Address of the partition COMREG.
11	Work register.
12	Base register for the phase.
13	Subroutine link register.
14-15	Work registers.

Sequence of Operation

CHKP2000: Completes building the header/save and trailer records: it computes the number of checkpoint records that have to be built to follow the header/save records; writes the header/save records onto the checkpoint tape.

CHKP2300: Builds and writes onto tape extent information records for:

- The SYSCAT device (if present).
- Programmer logical units, if DASD file protection has been activated (retrieves the required information from the system extent area, whose address is contained in the LUBTAB at offset X'04').

CHKP2500: Required PFIX-information records (retrieves the information needed to build these records by appropriate PFIXCHPT macros). These macros provide PFIX information in the work area supplied by the phase.

CHKP2600: Program-dump records as required and the tape-checkpoint trailer record, which actually is a copy of the tape-checkpoint header record.

Phase $$BCHKP3 – Tape Checkpoint Phase 3

Entry Point: IJBCK24O.

Function: Builds the required 3800-printer information records and writes them onto the specified tape as part of the currently generated checkpoint data.

Called by: Phase $$BCHKP2.

Phases Called: None.

Subroutines Used

IORT: To write data onto tape.

For more details about the function of this subroutine, see "Subroutines Used" in the description of phase $$BCHKP2.

Data Areas Used

- COMREG, the checkpointed partition's communication region (in the supervisor).
- The LUB table (in the supervisor).
- COMDSCT, the common tape-checkpoint work area.
- QWORK, the 3800-printer information-record-build and output area.

Messages Caused

```
0C13I   INSUFF. SPACE FOR CHKPT ON SYSxxx=cuu. CHECKPOINT IGNORED
0C16I   QSETPRT FAILED RC=X'oonnxxrr' SYSxxx=cuu. CHECKPOINT IGNORED
```

Messages Issued: None.

Input: None.

Output: 3800-printer information records written onto tape.

Exit Normal: To the calling phase, $$BCHKP2.

Exit Error: To phase $$BRMSG1.

Register Use

1	Pointer to phase name (for SVC 2).
2	Macro return-code (for $$BRMSG1).
3	LUB-table scan register.
4	Logical unit number (for $$BRMSG1; work register.
5	Message code (for $$BRMSG1); work register.
6-8	Work registers.
9	Address of COMDSCT, the common tape checkpoint work area.
10	Address of the partition's COMREG.
11	Work register.
12	Base register for the phase.
13	Link register for subroutine call.
15	Macro return-code register.

Sequence of Operation

CHKP3005: Finds the logical unit(s) to which a 3800 printer is assigned; sets up the required work and I/O areas; retrieves the 3800-printer information using the QSETPRT macro.

Writes the 3800-printer information record(s) onto the checkpoint tape, one for each assigned 3800 printer.

CHKP3070: If the QSETPRT macro request fails or an end-of-reel condition occurs while writing a 3800-printer information record, the phase repositions the checkpoint tape to its position at the time of the checkpoint request.

$$BCHK3G – Tape/Disk Checkpoint, Last Phase

Entry Point: CHK3G.

Function: Issues an erase-gap command for any of the checkpointed program's tape output files that are referenced in the tape-repositioning table for logical files but have not been closed.

Called by: Phase $$BRMSG1.

Phases Called: None.

Subroutines Used: None.

Data Areas Used

- The CHKPT macro parameter list
- The partition's save area

Messages Caused

```
0C18I   ERRORS DETECTED IN REPOSITIONING TABLE
```

Messages Issued: None.

Input: See "Data Areas Used" above.

Output: None.

Exit Normal: To the supervisor via SVC 11.

Exit Error: To phase $$BRMSG1.

Register Use

1	Pointer to CCB for tape I/O; pointer to phase name for an SVC 2 call.
2	Error counter.
5	Message code (for $$BRMSG1).
7-8	Work registers.
9	Pointer to the partition's save area.
11	Pointer to CHKPT macro parameter list.
14	Base register for the phase.
15	Macro return-code register.

Sequence of Operation

CHK3G005: Scans the tape repositioning table if the user has specified one; for each tape-output device defined in this table, the routine issues an erase-gap command if this device is still open.

If an entry in the user's logical IOCS table(s) refers to a file defined by a DTFxx macro other than DTFMT or refers to a device which is not a tape drive, the routine ignores the entry and causes message 0C18I to be issued.

Phase $$BRMSG1 – Checkpoint Message-Writer

Entry Point: IJBMSG50.

Function: Builds and writes to SYSLOG checkpoint messages for the calling phase.

Called by: One of the following phases

```
$$BCHKPD     $$BCHKP2
$$BCHKPE     $$BCHKP3
$$BCHKPF     $$BCHKPG
$$BCHKPT     $$BCHK3G
```

Phases Called: None.

Subroutines Used: None.

Data Areas Used

- COMREG, the partition communication region (in the supervisor).
- MSGTEXT, the message-text build and branch table.

Messages Caused: None.

Messages Issued: Those caused by the calling phases as listed under "Called By" above.

Input: Contents of registers 2 to 6, which are used for building the message.

Output: Depends on the type of message that is to be written:

- For an error message

  Register 0, and also that register's field in the partition's save area, set to zero.

- For a successful-completion message

  The updated checkpoint number in the partition's COMREG and (in unpacked decimal) in the register-0 field of the partition's save area.

- For an internal-error message

  A storage dump and a checkpoint-cancel request.

Exit Normal: To phase $$BCHK3G.

Exit Error: To the supervisor (via SVC 11).

Register Use

1	Pointer to CCB for writing to SYSLOG.
2	Parameter register: return code (for insertion into a message).
3	Parameter register: phase/macro identifier (for insertion into a message) or zero.
4	Parameter register: logical unit number (for insertion into a message) or zero; work register.
5	Parameter register: the message code; work register.
6	Pointer to the checkpoint file DTFPH.
7-8	Work registers.
9	Pointer to the partition save area.
11	Pointer to the CHKPT macro parameter list.
14	Base register for the phase.
15	Macro return-code register.

Sequence of Operation

MSG1000: Uses the message number passed to the phase for getting

- The address of the corresponding message text.
- The values needed for conversion and completion of this text.

Writes the completed message to SYSLOG, using physical IOCS.

Phase $$BRMSG2 – Restart Message-Writer

Entry Point: IJARMG20.

Function: Builds messages and writes them to SYSLOG. For action-type messages, the phase checks the operator's response for validity.

Called by: Phase $$BRSTRT or $$BRSTR2.

Phases Called: None.

Subroutines Used: None.

Data Areas Used

- MSGTEXT, the message-text build table.
- The DTFPH block for the checkpoint file.

Messages Caused

```
0R30D   INVALID RESPONSE, TRY AGAIN
```

Messages Issued: Same as caused plus those caused by the calling phases (see "Phases Called" above).

Input

- The identifier of the message that is to be issued.
- Message response from SYSLOG if the message-originating phase expects a response.

Output: Message to SYSLOG or, if necessary, the operator's response to a message written to SYSLOG.

Exit Normal

- To the supervisor (via SVC 6) if the message to be issued indicates that the program cannot be restarted.
- To the calling phase ($$BRSTR2) if the operator's response is valid.

Exit Error: To the supervisor (via SVC 6).

Register Use

1	Pointer to tape CCB (for SVC 0) and to phase name (for SVC 2).
3	Work register.
4	Pointer to macro name (for insertion into a message; work register.
5	Paramter register: message-code; work register.
6	Base register for access to the user's DTFPH block.
7	Work register.
14	Base register for the phase.
15	Macro return-code.

Sequence of Operation: Saves the contents of registers 2 through 13 when it receives control; restores their contents again when it returns control to the calling phase.

MSG2005: Uses the message identifier passed to the phase for finding the correct message text and for converting values as required to complete this text. Writes the completed message to SYSLOG using physical IOCS. If the message requires a response, this routine checks the response for validity. For an invalid response, the routine prompts the operator to enter a valid response.

Phase $$BRSTRT – Restore Checkpointed Partition

Entry Point: IJARST40.

Function: Restores PFIX information saved when the currently processed checkpoint was taken; restores the program area in the partition.

Called by: Phase $JOBJCCN (of job control).

Phases Called: None.

Subroutines Used

CKDIO00: To retrieve program-dump records from a checkpoint file on a CKD disk.
FBAIO00: To retrieve program-dump records from a checkpoint file on an FBA disk.
TAPIO00: To retrieve program-dump records from a checkpoint file on tape.

Data Areas Used

- CHKPT-macro parameter list.
- DTFPH block if a checkpoint file on disk is used.
- PFIX information record.
- Program-dump record.
- Restart information record.

Messages Caused

```
0R09I   INTERNAL RSTRT ERROR IN phasename SETLIMIT FAILED RC=X'nn'
        SIZE=nnnnK
0R13I   INTERNAL RSTRT ERROR IN phasename EXPECTED RECORD NOT FOUND
```

Messages Issued: None.

Input: The checkpoint parameter-list as passed by the job control phase $$JOBCTLB.

Output: Restored job information: the PFIX table and the program area

Exit Normal

- To phase $$BRSTR2 if the checkpoint parameter list indicates that the user specified a tape repositioning table or a DASD verification table or both.
- To the supervisor (via SVC 11) if the user did not specify any of these tables.

Exit Error: To phase $$BRMSG2.

Register Use

0	Pointer to the Checkpoint parameter list as passed by $JOBCTLB.
1-4	Work registers
5	Parameter register for $$BRMSG2: message-code; work register.
6-7	Work registers.
8	Subroutine-link register.

9 Subroutine-link register; pointer to common restart work area.
10 Address of COMREG.
11 Base register for access to the CHKPT macro parameter list.
12 Pointer to the partition save area.
13 Work register.
14 Base register for the phase.
15 Macro return-code.

Sequence of Operation

RSTRT005: Saves the information passed from $JOBCTLB as a parameter list; frees the system GETVIS area that was used for passing this list.

RSTRT100: Corrects the partition's logical end address if the checkpointed partition was running in real mode and GETVIS is indicated. Reads and, as applicable, restores PFIX information using the PFIXREST macro.

RSTRT200: Reads the program-dump records into the partition:

If the checkpoint data is on tape, the phase positions the tape for reading immediately behind the checkpoint-trailer record.

If the checkpoint data is on disk, the phase updates the checkpoint DTFPH by setting the seek address to the next higher track address (for a CKD disk) or block address (for an FBA disk).

RSTRT500: Retrieves the CHKPT-macro generated parameter list from the restored program to determine the proper exit. Retrieves the program-restart address from that parameter list if neither a tape-repositioning nor a DASD-verification table was specified for the checkpoint request; stores this address in the user's PSW as is contained in the restored partition's save area.

Phase $$BRSTR2 – Disk and Tape Verification

Entry Point: IJARS240.

Function: Verifies DASDs as specified in the user-supplied DASD verification table; repositions tapes as specified in the user-supplied tape reposition table.

Called by: Phase $$BRSTRT.

Phases Called: Phase $$BRMSG2.

Subroutines Used

RSTR1000: To check for the type of the device that is assigned to the defined logical unit.
RSTR2000: To repositions a tape for forward reading.
RSTR2500: To repositions a tape for backward reading.
TAPFSR: To execute tape I/O operations for tape positioning.

Data Areas Used

- CHKPT macro parameter list.
- DASD verification table.
- Tape repositioning table.

Messages Caused

```
0R15I   INTERNAL RSTRT ERROR IN phasename. macroname FAILED. RC=X'nn'
0R20A   RIC TAPE REPO: SER volserno SEQ seqno SYSxxx=cuu
0R21D   IC TAPE REPO: TAPE MARK IN DATA SYSxxx=cuu
0R22D   IC TAPE REPO: DEVICE NOT A TAPE SYSxxx=cuu
0R23D   IC TAPE REPO: DTFTYPE X'nn' INVALID filename
0R24D   IC TAPE/DASD: UNIT NOT ASSIGNED SYSxxx
0R25A   RIC DASD VERI: SER volserno ASSIGNED SYSxxx=cuu
0R26A   RIC DASD VERI: VOL. SER. NO. INVALID SYSxxx=cuu
0R27D   RIC DASD VERI: DEVICE NOT A DISK SYSxxx=cuu
0R28A   RIC DASD VERI: DEVICE IS NOT READY SYSxxx=cuu
0R29D   RIC DASD VERI: LOG. UNIT INVALID SYSxxx=cuu
```

Messages Issued: None.

Input: See "Data Areas Used" above.

Output: Messages as required (see "Messages Caused" above).

Exit Normal: To the supervisor (via SVC 11).

Exit Error: To phase $$BRMSG2 for a condition that causes message 0R15I.

Register Use

1 Pointer to tape CCB (for SVC 0) and to phase name (for SVC 2).
2 Work register.
3 Pointer to logical tape-repositioning table.
4 Pointer to macro name (for $$BRMSG2); work register.
5 Message code (for $$BRMSG2); work register.
6 Pointer to user DTF block.
7 Work register.
8 Main-line routine place holder.
9 Address of the common restore work area.
10 Address of area COMREG.
11 Address of the CHKPT-macro parameter list.
12 Address of the partition-save area.
13 Address for return to the main-line routine.
14 Base register for the phase.
15 Macro return-code register.

Sequence of Operation

RSTR0140: Handles logical tape-file repositioning:

Checks the entries in the logical reposition table and issues a message if one of the following conditions exists:

- The DTF type is incorrect (not DTFMT).
- The device is not assigned.
- The assigned device is not a tape.

Ignores the entry if the DTF is not for an output file or if there has been no OPEN for the file.

Picks up the number of blocks recorded in the DTF. If a VOL1 label exists, writes the volume serial number and the volume sequence number to SYSLOG for verification that the correct tape reel has been mounted; the operator can mount a new tape, cancel the job, or continue. Bypasses standard labels and tape marks without decrementing the block count.

Ignores a backward-read indication in the DTF if the tape has standard labels.

Expects, for backward reading, the tape to be positioned past the tapemark following the last record and ahead of any non-standard trailer labels. Moves the tape backwards beyond the tapemark (and any noise record) and positions the tape the indicated number of blocks from the end of the file (see also "Note" below).

RSTR0400: Handles physical repositioning:

Checks for the presence of physical reposition entries. For a physical reposition entry, the routine checks whether the defined logical unit is assigned to a tape unit. Issues a message if that logical unit is not assigned to a tape unit; spaces forward the specified number of files (tapemarks) and the indicated number of records (see "Note" below).

Note: If, during tape positioning, the routine encounters a checkpoint record, it skips this record without incrementing the record count. If the routine finds an unexpected tapemark, it issues an error message.

RSTR0600: Handles DASD verification, either after tape repositioning is completed or at once if there is no need for such repositioning. Ensures, for each table entry, that the logical unit specified in the table is assigned to a disk device; reads the VOL1 label and writes the volume's serial number to SYSLOG for verification. The operator can cancel the job, mount a new pack and retry, or ignore the message and continue.

RSTR0800: Stores the restart address in the user's PSW as is contained in the restored partition's save area and exits to the supervisor.

Organization Information

This section provides a list of the checkpoint/restart phases sorted alphabetically by their names. The list gives a summary of each phase's function; it shows by which phase each phase is called and which phase(s) are called by a particular phase.

Phase	Called by	Function	Calls/Exits to
$$BCHKPD	CHKPT macro (SVC 2)	Does initial checking to determine whether a checkpoint can be taken. Starts building the checkpoint the checkpoint header record if, at this point, taking a checkpoint is OK.	$$BCHKPE for an OK condition. $$BRMSG1 otherwise.
$$BCHKPE	$$BCHKPD	Completes checking to determine whether a checkpoint can be taken. Completes building the record if taking a checkpoint is OK. Checks whether 3800 information is to be saved.	$$BRMSG1 for writing an error message. $$BCHKPG for saving 3800 information. $$BCHKPF for checkpointing without 3800 information.
$$BCHKPF	$$BCHKPE	Builds and writes: - Extent records - PFIX records - Program dump records Sets up successful-completion or error message.	$$BRMSG1
$$BCHKPG	$$BCHKPE	Builds and writes 3800-information records.	$$BRMSG1 for writing an error message. $$BCHKPE on successful completion.
$$BCHKPT	CHKPT macro (SVC 2)	Checks whether a checkpoint can be taken. Starts building the checkpoint header record if taking a checkpoint is OK.	$$BCHKP2 for an OK condition. $$BRMSG1 otherwise.
$$BCHKP2	$$BCHKPT	Completes building the required header/trailer records and writes them onto tape. Checks whether 3800 information is to be saved. Builds and writes: - Extent-information records - PFIX records - Probram-dump records Sets up completion message	$$BRMSG1 for writing an error or successful completion message. $$BCHKP3 for building and writing 3800 information.

Phase	Called by	Function	Calls/Exits to
$$BCHKP3	$$BCHKP2	Builds and writes 3800-information records.	$$BRMSG1 for writing an error message. $$BCHKP2 on successful completion.
$$BCHK3G	$$BRMSG1	Erases gap on the output file if this is open.	Supervisor (via SVC 11) $$BRMSG1 in case of an error situation.
$$BRMSG1	$$BCHKPD $$BCHKPE $$BCHKPF $$BCHKPG $$BCHKPT $$BCHKP2 $$BCHKP3 $$BCHK3G	Writes to SYSLOG a message as set up by the calling phase.	Supervisor (via SVC 11) for an error message. $$BCHK3G for a successful-completion message.
$$BRMSG2	$$BRSTRT $$BRSTR2	Writes to SYSLOG a message as set up by the calling phase (either prompting for tape repositioning or DASD verification or an error message).	$$BRSTR2 for a prompting message. Supervisor (via SVC 6) for an error message.
$$BRSTRT	$JOBCTLB (job control)	Restores PFIX and program dump information. Sets up the user-specified restart address, unless tape repositioning or DASD verification is necessary.	$$BRSTR2 for tape repositioning or DASD verification. $$BRMSG2 for writing error messages. Supervisor (via SVC 11) otherwise.
$$BRSTR2	$$BRSTRT	Repositions tapes. Verifies DASD information. Sets up the user-specified restart address.	$$BRMSG2 to prompt for DASD information. Supervisor (via SVC 11) otherwise.

Data Area Information

This section provides a summary of the key data areas used by the checkpoint/restart routines; these areas are given in alphabetic order of their names. Displacements of fields within an area are given in decimal in the first (or only) displacement column; if meaningful, displacements are given also in hexadecimal in a second displacement column, with or without source-code labels.

@PARLIST – CHKPT-Macro Parameter-List

Built by CHKPT macro code-expansion. When the checkpoint transient ($$BCHKPT or $$BCHKPD) receives control, register 0 contains the address of this list. Its layout is:

Displ.

0-3 A four-byte address, or X'FF' followed by a three-byte register number; specifies a restart address.

4-7 A four-byte address, or X'FF' followed by a three-byte register number; specifies the checkpoint end-address. This field is all zeros if the user did not specify an end address.

8-11 A four-byte address, or X'FF' followed by a three-byte register number; specifies the address of the tape repositioning table. This field is all zeros if the user did not specify the tpointer operand.

12 Set to X'01'.

13 The system unit number for SYSxxx as specified in the macro.

The checkpoint is to be written onto tape, and the macro does not include the dpointer operand:

14-21 Contains C'$$BCHKPT'.

The checkpoint is to be written onto tape, and the macro includes the dpointer operand:

14-15 Set to all zeros.

16-19 A four-byte address, or X'FF' followed by a three-byte register number; specifies the address of DASD verification information.

20-27 Contains C'$$BCHKPT'.

The checkpoint is to be written onto disk (the macro includes the filename operand), and the user omitted the dpointer operand:

14-19 Set to all zeros.

20-23 The four-byte address of the user-specified DTFPH for the checkpoint file on disk (or X'FF' followed by the 3-byte number of the register that contains this address).

24-31 C'$$BCHKPD'.

The checkpoint is to be written onto disk (the macro includes the filename operand), and the user specified a dpointer value:

14-15 Set to X'0000'.

16-19 A four-byte address, or X'FF' followed by a three-byte register number; specifies the address of DASD verification information.

20-23 The four-byte address of the user-specified DTFPH for the checkpoint file on disk (or X'FF' followed by the 3-byte number of the register that contains this address).

24-31 Contains C'$$BCHKPD'.

Common Checkpoint Work Area

The area is located at the high end of each phase using it, and its address is passed from one phase to the next by a pointer in register 9. The area is defined to begin at label:

COMDSCTD for taking a checkpoint on disk.
COMDSCT for taking a checkpoint on tape.

For either type of checkpoint, the area has an overall structure as shown below. Each of the structure items given below is discussed separately following this overview:

1. Common constants, which is a different set for either checkpoint device-class.
2. Tape-checkpoint header/trailer record (not present if a checkpoint is to be written onto disk).
3. Checkpoint-record-build and output area.

Common Constants for Writing Checkpoint Data onto Disk Displacement

```
 0-3     0-3    SAVDTFPT
                Address of the DTFPH block for the checkpoint file.

 4-7     4-7    SAVECKPT
                Save address specified in the CHKPT macro.

 8-11    8-B    SAVPGMLE
                First programmer logical unit in LUB extension.

12-15    C-F    SAVSYSLE
                Pointer to SYSCAT in LUB extension.

16-19   10-13   SAVLADR
                The partition's lower boundary.

20-23   14-17   SAVHADR
                The partition's upper boundary.

24-25   18-19   SAVRECTR
                Track address for the current program-dump record.

26-27   1A-1B   SAVDMPLN
                Length of one program-dump record.

28-29   1C-1D   SAVRECNO
                Number of 126-byte records per track.

30-31   1E-1F   SAVPCOMA
                Address of the checkpointed partition's COMREG.
```

32-33	20-21	SAVSCOMA Address of SYSCOM.
34-35	22-23	SAVPGMLB Pointer to the first programmer LUB in the LUB table.
36-37	24-25	SAVSYSLB Pointer to the SYSCAT LUB in the LUB table.
38-39	26-27	SAVNICL Pointer to the checkpointed partition's NICL.
49-41	28-29	SAVDMPTR Number of tracks per program-dump record.
42-43	2A-2B	SAV3800 Number of 3800-printer information records.
44-45	2C-2D	SAVXTNT Number of extent-information records.
46-47	2E-2F	SAVPFIX Number of PFIX-information records.
48-49	30-31	SYSCAT Logical unit for SYSCAT (H'13').
50-57	32-39	SAVESEEK Save field for the current Seek command (initially set to all zeros).
58-65	3A-41	$$BRMSG1 Name of the message-writer phase.
66	42	CHARE Charcter E (to identify phase $$BCHKPE by its last character).
67	43	CHARG Charcter G (to identify phase $$BCHKPE by its last character).
68	44	SCTR Value for a set-sector operation (initialized to X'FF').
69	45	SCTRR Current sector value (initialized to X'FF').

Common Constants for Writing Checkpoint Data onto Tape Displacement

0-3	0-3	SAVPGMLE First programmer logical unit in LUB extension.
4-7	4-7	SAVSYSLE Pointer to SYSCAT in LUB extension.
8-11	8-B	SAVLADR The partition's lower boundary.
12-15	C-F	SAVHADR The partition's upper boundary.
16-17	10-11	SAVPCOMA Address of the checkpointed partition's COMREG.
18-19	12-13	SAVSCOMA Address of SYSCOM.
20-21	14-15	SAVPGMLB Pointer to the first programmer LUB in the LUB table.
22-23	16-17	SAVSYSLB Pointer to the SYSCAT LUB in the LUB table.
24-25	18-19	SAVNICL Pointer to the checkpointed partition's NICL.
26-27	1A-1B	SAVERECN Countfield for records written to tape.
28-29	1C-1D	SAVCHKID ID of currently processed checkpoint data.
30-31	1E-1F	SAV3800 Number of 3800-printer information records.
32-33	20-21	SAVXTNT Number of extent-information records.
34-35	22-23	SAVPFIX Number of PFIX-information records.
36-37	24-25	SAVLOGUN Logical unit for the checkpoint device.
38	26	SAVEMODE Set-Mode command for the checkpoint unit (initially set to X'00').
39	27	INDAPRT Indicator for communication with phase $$BCHKP3: X'00' = 3800-printer information records yet to be written X'01' = Writing of 3800-printer information records is complete

Checkpoint-Record-Build and Output Area

OUTAREA at COMDSCTD + 1420 (X'58C') – Output area if checkpoints are to be written onto disk.

OUTAR at COMDSCT + 1420 (X'58C) – Output area if checkpoints are to be written onto tape.

The phase in control uses this area to build checkpoint records as listed below, one at a time. The beginning of the particular record-build area is always identical with the address of OUTAREA or OUTAR, respectively. The records are:

- **Disk-checkpoint header** record if the checkpoint data is to be written onto disk – For its layout and content, see "Disk-Checkpoint Header-Record" later in this section.
- **Tape-checkpoint save** record if the checkpoint data is to be written onto tape – For its layout and content, see bytes 20-183 (X'14'-X'B7') under "Disk-Checkpoint Header-Record" later in this section.
- **PFIX-information** record (256 bytes) – For its layout and contents, see "PFIX-Information Record" later in this section.
- **Extent-information** record (256 bytes) – For its layout and contents, see "Extent-Information Record" later in this section.
- **3800-printer information** record (76 bytes) – For its layout and contents, see "3800-Printer Information-Record" later in this section.

DASD-Verification Table

It is the user's responsibility to set up this area in the program that is to be checkpointed. When properly coded, this table is aligned to a halfword boundary. Phase $$BRSTRT picks up the address of this table from the checkpoint-parameter list that the job control phase $JOBCTLB passes to phase $$BRSTRT. The layout of the table is:

Displacement

0-1 Number of entries contained in the table (in binary notation).

2-5 The first entry of the table:

2-3 = Symbolic unit as copied from the CCB bytes 6 and 7.
4-5 = Reserved.

6-9 The second entry of the table – It has the same format as the first entry.

And so on.

Disk-Checkpoint Header-Record

The record is built by phases $$BCHKPD and $$BCHKPE. The area is part of the common checkpoint work area and begins at the symbolic address OUTAREA. Phase $$BCHKPE writes the record into the checkpoint file. The layout of the record is:

Displacement

0-11	0-B	@HEADID Checkpoint header identifier — The characters /// CHKPT //.
12-13	C-D	@HEADUMP Number of program-dump records (in binary notation).
14-15	E-F	Not used.
16-17	10-11	@HEADNUM Checkpoint-identification number (in binary notation).
18-19	12-13	@HEADCTR Used only if the disk device is of type FBA. Contents of the bytes: 18 = Number of blocks needed for information records. 19 = Number of blocks needed for program-dump records.
20-23	14-17	@HEADADD Highest storage address to be checkpointed.
24-87	18-57	@HEADREG Contents of general registers as stored in the partition save area.
88-107	58-6B	@HEADCR1 Partition communication region, bytes 12 through 31.
108-117	6C-75	@HEADCR2 Partition communication region, bytes 36 through 45.
118-119	76-77	@HEADCR3 Partition communication region, bytes 56 and 57.
120-121	78-79	@HEADCR4 Partition communication region, bytes 92 and 93.
122	7A	@HEADCR5 Partition communication region, byte 78.
123	7B	@HEADCR6 Supervisor control-bit settings as shown below.

Byte in Supervisor	Bit	Stored Here as Bit	Meaning if Set On
COMREG 98	0	1	One or more 3800 printer DTFs for extended buffering are open.
COMREG 134	7	7	The GETVIS function has been started.
TCB 6	1	6	Writing to the SYSRES file is allowed.

```
124-127  7C-7F   @HEADCR7
                 Address of GETVIS control information.

  128     80     @HEADPIB
                 Partition PIB, byte 12.

  129     81     @HEADFLAG
                 Checkpoint-system-configuration flag whose bits, if set to 1,
                 indicate the following:
                      X'40' = ECPS:VSE mode.
                      X'20' = The checkpoint device is a tape unit.
                      X'10' = The checkpoint device is an FBA disk.
                      X'08' = DASD file protection is active.
                      X'04' = PFIX information is to be handled.
                      X'02' = GETVIS information is to be handled.
                      X'01' = The program to be restarted was running in real mode.

130-131  82-83   @HEADLAB
                 Length of the partition's label save area
                 (as contained in the partition save area).

132-151  84-97   @HEATBDY
                 Temporary partition boundaries.

152-171  98-AB   @HEAPBDY
                 Permanent partition boundaries.

172-173  AC-AD   @HEAPFIX
                 Number of PFIX-information records.

174-175  AE-AF   @HEAXTNT
                 Number of extent-information records.

176-177  B0-A1   @HEA3800
                 Number of 3800-printer information records.

178-183  A2-B7   @HEASCAT
                 Volume serial number of SYSCAT.
```

Extent-Information Record

Built by phase $$BCHKP2 if the checkpoint is to be written onto tape and by phases $$BCHKPF if the checkpoint is to be written onto disk. Either phase builds the record in the output area of the common work area, using the DSECT labels from XTNRECID to XTNSAV5. An extent-information record consists of an eight-byte header followed by up to fifteen 16-byte extent entries.

The phase that builds this record also writes it onto tape or disk, respectively.

The layout of the record is:

Displacement

0-2 Contains C'XTN' (= identifies this as an extent-information record).

3-7 Reserved.

8-256 Extent entries, 16 bytes per entry. The layout of an extent entry, which is accessed via DSECT XTNDSCT, is as follows:

Displacement

0 Extent-record control flag:

X'80' = Last entry of the record.
X'01' = The entry applies to the same device as the preceding entry.

1 PUB device type code.

2-3 Logical unit class (X'00' = system, X'01' = programmer) in byte 2; index into the checkpointed partition's LUBTAB in byte 3.

4 Saved extent-flag field.

5 Reserved.

6-7 Saved extent-usage count.

8-11 Upper extent limit.

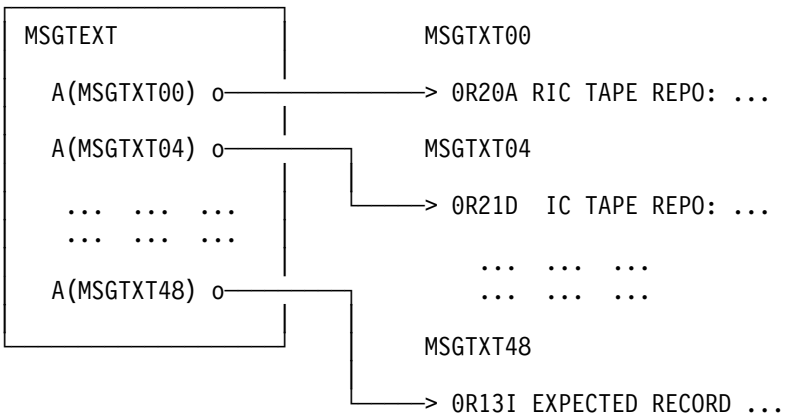
12-15 Lower extent limit.

Header/Trailer Record for Writing Checkpoint Data onto Tape

@HEADID – The label of the first field of the 20-byte header/trailer record. Marks the beginning of the area for building this record if the checkpoint is to be written onto tape. For a detailed description of this record, refer to "Tape-Checkpoint Header/Trailer Record" later in this section.

MSGTEXT – Message-Text-Build Table

Phases $$BRMSG1 and $$BRMSG2 use a similar technique for building messages based on the message code that is passed to them by one of the checkpoint or restart phases, respectively. The table's layout is (using the one in phase $$BRMSG2 as an example):



The instruction

```
L  R5,MSGTEXT(R5)
```

loads the address of the required message text into register 5. If the message code passed to the message writer phase in register 5 is 04, for example, the instruction loads the address of the text at label MSGTXT04 into register 5.

PFIX-Information Record

Built by phase $$BCHKP2 if the checkpoint is to be written onto tape and by phase $$BCHKPF if the checkpoint is to be written onto disk. Either phase builds the record in the output area of the common work area, using the DSECT labels @FIXID and @FIXAR.

The phase that builds this record also writes it onto tape or disk, respectively.

The layout of the record is:

Displacement

0-3 Contains C'PFIX' (identifies this as a PFIX-information record).

4-255 Data as retrieved via the PFIXCHPT macro. The format of this data depends on the supervisor being used:

For a MODE=E or MODE=VM supervisor:

Bytes	
0-3	Address of PFIXed page
4-5	PFIX count

For a MODE=370 supervisor:

Bytes	
0-3	Address of PFIXed page
4-7	Address of page frame
8-9	PFIX count

Restart Information Block

The block contains information about the partition to be restarted. The block is built by phase $JOBCTLB in the system GETVIS area, and its address is passed to phase $$BRSTRT in register 0.

The layout of the block is:

Displacement		Labels
0-3	0-3	STARTADD Begin address of partition to be restarted.
4-7	4-7	HIGHADD Highest address to be restored.
8	8	HEADFLG Partition information byte whose bits, if set to 1, indicate the following: X'40' = ECPS:VSE mode. X'20' = The checkpoint device is a tape unit. X'10' = The checkpoint device is an FBA disk. X'08' = DASD file protection is active. X'04' = PFIX information is to be handled. X'02' = GETVIS information is to be handled. X'01' = The program to be restarted was running in real mode.
9	9	Reserved.
10-11	A-B	HEAPFIX Number of PFIX records to be handled.
12-13	C-D	SYSINDJ Logical unit number of the checkpoint device.
14-15	E-F	DEVIND Depending on the type of device used for checkpointing: For tape: 7- or 9-track indicator. For an FBA disk: Byte 14 - Block count for information records. Byte 15 - Block count for program-dump records.
16-17	10-11	DMPRECL Length of a program-dump record.
18-19	12-13	DMPRECN Number of program-dump records.
20-23	14-17	PENDLOG Logical end of the checkpointed partition.

```
24-27   18-1B  JCBCKRGT
               Address of the GETVIS-area control information.

28-31   1C-1F  Reserved.

The remaining area if the checkpoint-device is a disk:

32-113  20-71  DTF
               The DTFPH block for the checkpoint file.

The remaining area if the checkpoint device is a tape unit:

32-39   20-27  STMODCCW
               Status-modifier CCW.

40-47   28-2F  TAPCCW
               I/O CCW

48-51   30-33  STMODAD
               Address of the set-mode CCW.

52-55   34-37  CCWAD
               Address of the I/O CCW.

56-63   38-3F  TAPCCB
               Tape I/O CCB.
```

Tape-Checkpoint Header/Trailer-Record

Built by phases $$BCHKPT and $$BCHKP2 at label @HEADID, an alias of OUTAR, in the common tape-checkpoint work area. Phase $$BCHKP2 writes the record onto tape at the beginning of the checkpoint data and, as a trailer record, also at the end of that data. When phase $$BCHKP2 receives control, register 9 contains a pointer to the common work area.

The layout of the record is:

Displ. Label

```
 0-11     @HEADID
          Checkpoint header identifier — The characters /// CHKPT //.

12-13     @HEADUMP
          Number of program-dump records (in binary notation).

14-15     @HEADREC
          Number (in binary notation) of checkpoint records (3800-printer
          information, extent information, PFIX information,
          program dump, header, trailer) that follow the header record.

          MTMOD uses this field to skip checkpoint data if this is
          interspersed with user data; it is used by the restart routines
          for skipping checkpoint data while scanning for a requested
          checkpoint.

16-19     @HEADNUM
          Checkpoint-identification number (in character notation).
```

Tape-Checkpoint Save-Record

Built by phases $$BCHKPT and $$BCHKP2. Phase $$BCHKP2 writes this record onto tape immediately behind the leading header/trailor record. The record is being built at label OUTAR in the common checkpoint work area. When phase $$BCHKP2 receives control, register 9 contains a pointer to that common work area.

The record's layout is identical with the layout of the disk-checkpoint header record, bytes 20 through 183 and accessed using the same set of labels. For details refer to "Disk Checkpoint Header-Record" above.

Tape-Repositioning Table (Logical)

The user is responsible for building the table within the program that is to be checkpointed. The address of this table is picked up by phase $$BRSTRT from the checkpoint-parameter list which the job control phase $JOBCTLB passes to phase $$BRSTRT. The layout of the table is as follows:

Displ.

0-1 Number of entries contained in the table (in binary notation).

2-5 First entry – The address of a DTFMT table.

6-9 Second entry – The same as the first entry

And so on.

Tape-Repositioning Table (Physical)

The user is responsible for building the table within the program that is to be checkpointed. The address of this table is picked up by phase $$BRSTRT from the checkpoint-parameter list which the job control phase $JOBCTLB passes to phase $$BRSTRT.
The layout of the table is as follows:

Displ.

0-1 Number of entries contained in the table (in binary notation).

2-7 The first entry of the table:

- 2-3 = Symbolic unit as copied from the CCB bytes 6 and 7.
- 4-5 = Number of files (tapemarks), in binary notation, that are to be skipped to get to the checkpoint position.
- 6-7 = Number of blocks, in binary notation, that are to be skipped following the last tapemark that is to be skipped.

8-13 Second entry – It is the same as the first entry.

And so on.

3800-Printer Information-Record

A 3800-printer information-record is built at by by phase $$BCHKP3 if the checkpoint is to be written onto tape and by phase $$BCHKPG if the checkpoint is to be written onto disk. Either phase builds a 3800 printer information record at label QWORK.

The record is written onto tape by phase $$BCHKP2 and onto disk by phase $$BCHKPG.

The layout of the record is as follows:

Displ.	
0-3	Contains C'PRT' (= identifies this as a 3800 printer information record).
4-5	Logical unit number.
6-7	Reserved.
8-76	Information as retrieved via the QSETPRT macro.

Chapter 4. Miscellaneous Logical Transient Service Routines

The service routines available as logical transients are:

- An automatic CLOSE for 3800 printer files
- System-residence file write-access
- Loading a forms control buffer from within a program

Phase $$BPCLOS - Automatic CLOSE for 3800 Files

Entry Point: $$BPCLOS+8

Function: Performs CLOSE processing for any 3800 printer file opened in extended-buffering mode and left unclosed by the user program; this ensures that all buffered data is printed.

Called By

- The terminator of the supervisor at the end of a main task
- Phase $$BCLOSE and $$BCLOS2 at the end of CLOSE processing initiated by $$BPCLOS.

Phases Called: $$BCLOSE to close automatically the 3800 printer file.

Subroutines Used

$$BEOJ4: End of task processing module.
$$BCLOSE: Close processing module.
IJDPR3+20: Extended buffering trunk routine.
IJDPR3+36: Extended buffering lock cleanup routine.

Data Areas Used: The chain of DTFXWAs (DTF extension work areas). This chain, which is anchored in the anchor-table extension (ATX), is created when the 3800 OPEN routine IJDPR3 loads the non-executable phase IJDANCHX. Phase IJDANCHX, the ATX, is pointed to by an entry in the anchor table for phase IJDANCHX.

Messages Issued: P201I AUTOMATIC CLOSING OF PRINTER FILES WAS UNSUCCESSFUL.

Input: The anchor table extension (ATX).

Output: A close parameter list for phase $$BCLOSE.

In COMREG bit OPN3800 is set off, when close completed.

Exit Normal: To the calling phase.

Exit Error: Same as for "Exit Normal" above.

Register Use: Compiler dependent.

Sequence of Operation

$$BPCLOS: This phase uses the chain of DTFXWAs (DTF extension work areas) to identify 3800 printer files opened in extended buffering mode and left unclosed by the user program.

The phase selects the first DTFXWA on the chain, builds a CLOSE parameter list using the DTF pointer in the DTFXWA, and initiates CLOSE processing for the file by transferring control to $$BCLOSE. Before invoking $$BCLOSE, $$BPCLOS marks the DTFXWA as having been selected already for CLOSE processing. If this processing fails and the DTFXWA is left on the chain, that DTFXWA is marked as already processed.

After completing their processing, the CLOSE routines fetch $$BPCLOS; this phase initiates CLOSE processing for the next unprocessed file on the DTFXWA chain. This operation is repeated until all unclosed 3800 files of the program are closed.

If there are still DTFXWAs on the chain after all files have been processed, the files associated with these DTFXWAs did not close successfully.

Phase $$BSYSWR -- System-Residence File Write-Access

This transient gives the calling program access to the system residence file also for write operations. The phase is included in the system primarily for compatibility reasons.

Loading a Forms Control Buffer from Within a Program

Loading the forms control buffer (FCB) of a printer can be requested from within a program by means of an LFCB macro. An FCB load request of this kind is handled by phase $$BATTF0 and either phase $$BATTF2 (if the printer is one of the PRT1 class) or phase $$BATTF3 (if the printer is not one of that class).

Based on the user specifications, the LFCB macro expansion builds an information area and passes its address to phase $$BATTF0 in register 0.

External Input: The LFCB macro as coded in the particular program; for details of its format, refer to *VSE/ESA System Macros Reference*.

External Output: The particular printer's FCB reloaded as requested.

Return codes in register 15 (in hexadecimal) as follows:

00 = The FCB load operation has been completed successfully.
04 = The assigned printer is of class PRT1, and the LPI operand specified in the macro does not agree with the FCB image.
08 = No LUB is available for the specified logical unit.
0C = The specified logical unit either is not assigned or is currently unassigned.
10 = The specified logical unit is assigned to a device without an FCB.
14 = The printer assigned to the specified logical unit is down.
18 = The specified FCB image was not found.
1C = The specified FCB image is invalid for the printer assigned to the specified logical unit.

Phase $$BATTF0 -- Macro FCB-Load Initiation

Entry Point: BATTF0+8

Function: Analyzes the user's specifications as passed to the phase by the LFCB macro expansion; sets up control values in a communication area for use by the LFCB macro execution phase.

Called By: The LFCB macro expansion via the supervisor or by IPW$$LW for FCB load of power printer.

Phases Called

- $$BATTF2 if the printer involved is one of the class PRT1
- $$BATTF3 if the printer involved is not one of the class PRT1

Subroutines Used: None.

Data Areas Used

- LFCB macro information area (MINFAREA)
- LFCB macro communication area (INFAREA)

Messages Issued: None.

Input: LFCB Macro information area -- for details, see "Data Area Information" following the LFCB macro phase descriptions.

Output: LFCB macro communication area -- for details, see "Data Area Information" following the LFCB macro phase descriptions.

One of the following return codes in register 15 (given in hexadecimal):

08 = No LUB is available for the specified logical unit.
0C = The specified logical unit either is not assigned or is currently unassigned.
10 = The specified logical unit is assigned to a device without an FCB.
14 = The printer assigned to the specified logical unit is down.

Exit Normal: To phase $$BATTF2 or $$BATTF3.

Exit Error: To the program that issued the macro (via task selection).

Register Use

12 Interface register to power
13 Address of MINFAREA
14 Address of INFAREA
15 Base register for the phase.

Sequence of Operation

$$BATTF0: The phase processes the macro-operand information as contained in MINFAREA in the sequence as indicated below and at labels given as applicable:

- FORMS=formnumber
- LPI=value -- at PFLPI
- NULMSG -- at PFNUL
- logical unit specification -- at PFUNIT

Unless it encounters a specification error, the phase inserts the processing results in INFAREA.

Phase $$BATTF2 -- LFCB Macro Execution for PRT1 Printers

Entry Point: IJBATTF2+8

Function: Executes the buffer-image load operation in accordance with the specifications in the LFCB macro.

Called By: Phase $$BATTF0

Phases Called: None.

Subroutines Used

LISTIO: To initiate a printer operation under control of the channel program whose address is supplied to the subroutine.
LOGIO (an entry of subroutine LISTIO): To initiate a write to SYSLOG operation.

Data Areas Used: LFCB macro communication area (INFAREA).

Messages Issued:

```
1B14A   cuu PRINTER NEEDS FORMS=xxxx
1B19I   cuu LFCB WITH PHASE (PHASENAME) EXECUTED
1B20A   INVALID RESPONSE
An optional verification message may be written.
```

Input: LFCB macro communication area (INFAREA).

Output: Buffer image loaded into the printer's forms control buffer.

Exit Normal: To the program that issued the macro (via task selection).

Exit Error: Same as "Exit Normal" above.

Register Use

0	Address for buffer load.
1	Excp, fetch register.
2-3	Work register.
4	Link register for I/O routine.
7	Information area base register.
8	Printer PUB address.
9	Address of register 15 in user save area.
10	Work register.
12	Address of LFCB expansion.
15	Base register for the phase.

Sequence of Operation

IJBATTF2: Converts the printer's unit address to printable characters; checks whether the specified buffer-image is available online and whether it is of correct length. Finds the address of the verification message and, if an LPI value was specified, verifies this value.

TSTBUFF: Checks the buffer-image for correct coding. Unless the printer is spooled by VSE/POWER, loads an auxiliary FCB and causes a skip to channel 1 to allow the operator to align the forms on the printer; passes a spool record to VSE/POWER if the printer is being spooled.

LOAD: Loads the required buffer-image; handles the necessary operator communication if the macro specified also a forms change.

DOSKIP: Causes a skip to channel 1; writes the verification message to the printer, if this is requested; writes the successful-completion message to SYSLOG.

Phase $$BATTF3 -- LCFB Macro Execution for 3203 and 5203 Printers

Entry Point: IJBATTF3+8

Function: Executes the buffer-image load operation in accordance with the specifications in the LFCB macro.

Called By: Phase $$BATTF0

Phases Called: None.

Subroutines Used

LISTIO: To initiate a printer operation under control of the channel program whose address is supplied to the subroutine.
LOGIO (an entry of subroutine LISTIO): To initiate a write to SYSLOG operation operation.

Data Areas Used: LFCB macro communication area (INFAREA).

Messages Issued:

```
1B13A   cuu (PRINTER NEEDS FORMS=xxxx)(,SET LPI=x)
1B19I   cuu LFCB WITH PHASE (PHASENAME) EXECUTED
1B20A   INVALID RESPONSE
An optional verification message may be written.
```

Input: LFCB macro communication area (INFAREA).

Output: Buffer image loaded into the printer's forms control buffer.

Exit Normal: To the program that issued the macro (via task selection).

Exit Error: Same as "Exit Normal" above.

Register Use

0	Address for buffer load.
1	Excp, fetch register.
2-3	Work register.
4	Link register for I/O routine.
7	Information area base register.
8	Printer PUB address.
9	Address of register 15 in user save area.

10 Work register.
12 Address of LFCB expansion.
15 Base register for the phase.

Sequence of Operation

IJBATTF3: Converts the printer's unit address to printable characters; prepares the message for forms mounting and line-space setting; checks whether the specified buffer image is available online and whether it is of correct length; finds the address of the verification message and inserts the length of the buffer-image phase in the library-read CCW.

LOADPHAS: Loads the required buffer image from the library into storage and from storage into the printer's buffer; causes a skip to channel 1 in preparation of forms alignment by the operator; handles the necessary operator communication for mounting of new forms, if required, and for alignment of forms.

DOSKIP: Unless the printer is spooled by VSE/POWER, causes a skip to channel 1; writes the verification message to the printer, if this is requested; writes the successful-completion message to SYSLOG.

Data Area Information

This section provides a summary of the key data areas used by the routines for loading a printer's forms control buffer from within a program. Displacements within the area are given in decimal notation in the first (or only) displacement column and, if appropriate, also in hexadecimal notation in a second displacement column.

MINFAREA -- Macro Information Area: The area is created by the LFCB macro expansion and passed to phase $$BATTF0 by a pointer to it in register 0. The layout of the area is:

Displ.

0-7 Specified phase name

8-10 xxx of SYSxxx specified in the macro

11 FORMS specification indicator: X'F1' = FORMS=xxxx was specified

12-15 Form number specified in the macro

16 LPI value specified in the macro or X'F0'

17 NULMSG indicator:

X'F1' = NULMSG was specified
X'F2' = Suppress information message (1B19I) on SYSLOG
X'F3' = Action for both X'F1' and X'F2'

INFAREA -- LFCB Macro Communication Area: The area is built by phase $$BATTF0; the phase passes the address of the area in register nn to either of phases $$BATTF2 and $$BATTF3. The layout of the area is:

Displ.

0-7	Specified phase name
8-11	Form number, if specified, or blank
12	Reserved
13	Number of lines per inch, if specified, or blank
14-15	Logical unit class and number
16-19	Address of PUB
20-23	Address of the user-register 15 save-area
24	Information byte. Its bits, if set on indicate the following:
	2 Suppress information message (1B19I) on SYSLOG
	3 NULMSG was specified
	4 FORMS=xxxx was specified
	5 LPI=n was specified
25	Reserved

Chapter 5. Termination Routines

Introduction

The system's termination routines consist of the phases $IJBSEOT and $IJBSDMP. Both these phases reside in the shared virtual area (SVA). $IJBSEOT is described in this chapter. For a description of $IJBSDMP refer to manual *Diagnosis Reference: Serviceability Aids*.

Phase $IJBSDMP is a reentrant resource. It processes ABEND dumps and macro requests for all partitions in parallel.

Phase $IJBSEOT receives control after program (or task) execution. It calls the phase $IJBSDMP when a dump of virtual storage is to be provided.

Phase $IJBSDMP receives control if:

- A program issues a dump request macro (DUMP, JDUMP, PDUMP, SDUMP, or SDUMPX).
- A system routine issues an IDUMP macro.
- A program (or task) ends before reaching normal end of job (task).

The main functions of the termination routines, which you might think of as a supervisor extension, are:

- If a subtask ends abnormally, to detach that task from the supervisor's task selection mechanism; otherwise to cancel the current job step.
- To allow I/O operations requested by the program (or task) to be completed.
- If DASD file protection is active, to dequeue disk extents owned by the program (or task).
- To write an ABEND message.
- To generate a storage dump as requested.

External Input

For phase $IJBSEOT

- Supervisor control block status at task-termination time.

External Output

For phase $IJBSEOT

- Task control data reset for initiation of another task.
- Messages as required.

Design Information

Figure 7 shows the conditions that cause the system's termination routines to receive control from the supervisor.

This chapter describes phase $IJBSEOT.

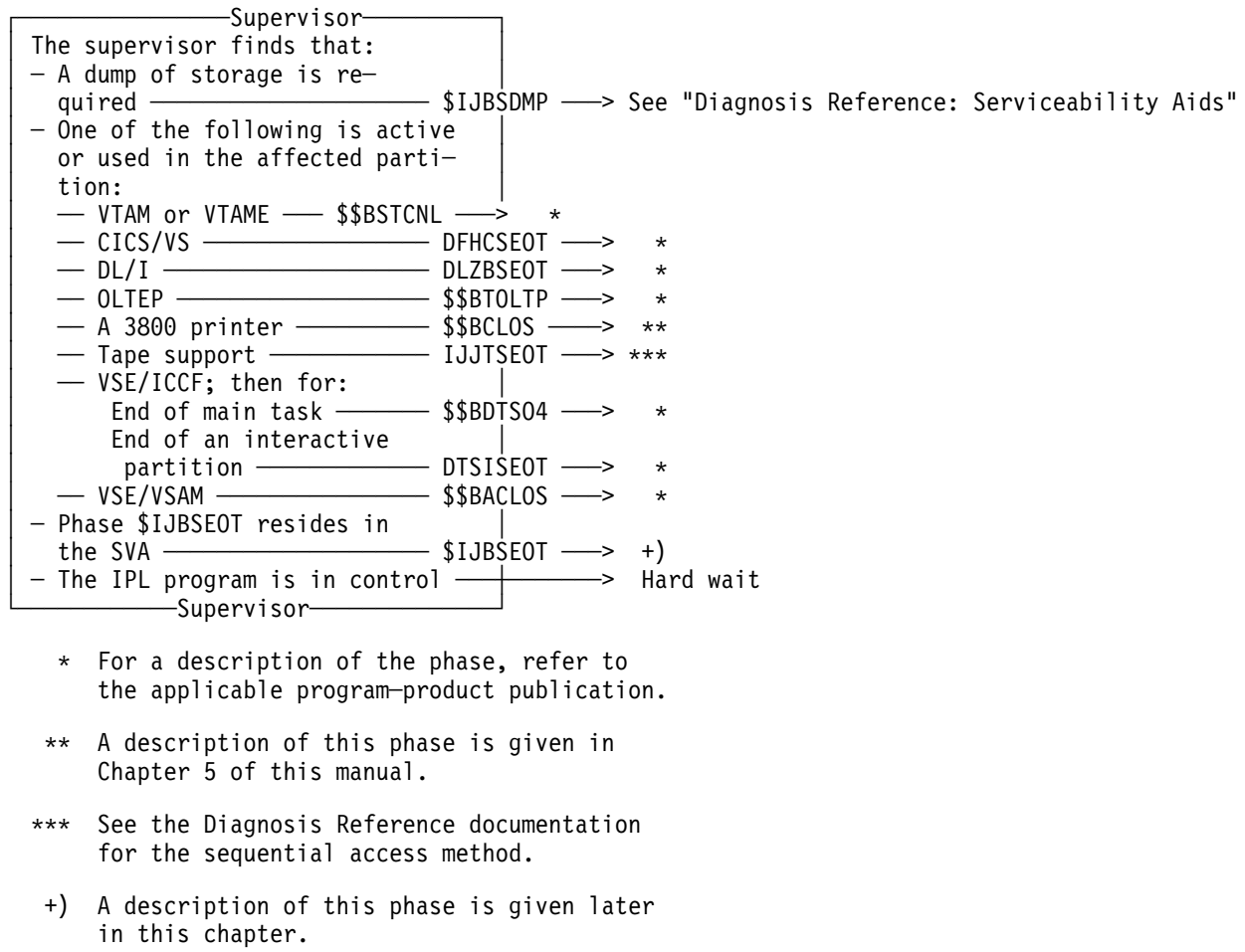


Figure 7. Control flow of termination routines

Phase $IJBSEOT – End-of-Task Routine

Entry Point: IJBEOTSK.

Function: Resets information contained in control blocks and tables within the supervisor.

Called By

- The supervisor's terminator routine.

Phases Called

- RPSLOAD to perform RPS reset functions.
- $JOBCTLA for return of control to job control.

Subroutines Used

EOT3000: To write messages issued by the phase to the SYSLOG device.

Supervisor-call routines as listed below:

```
SVC   0: To handle the requested I/O operation.
SVC   3: To wait for the completion of I/O operations started by the
         ending task.
SVC   4: To load the requested phase into virtual storage.
SVC   7: To wait for the completion of a request.
SVC  11: To return control to the supervisor from the LTA or the SVA.
SVC  22: To seize or release the system.
SVC  25: To halt I/O for a telecommunication device and dequeue the
         associated CCB.
SVC  27: To halt I/O for a telecommunication device without dequeuing the
         associated CCB.
SVC  33: To get the address of the affected partition's MAPCOMR.
SVC  36: To free tracks held by the ending task.
SVC  39: To detach the indicated subtask and return control to the
         supervisor.
SVC  44: To write a record to the system recorder file.
SVC  54: To release page frames fixed by a PFIX request.
SVC  59: To reinitialize, for the specified pages, the associated entries
         of the page table and the page-frame table.
SVC  62: To release partition GETVIS space.
SVC  92: To reset the affected partition's entries in the
         cross-partition event control-blocks (XECBs), if any.
SVC  98: To extract the boundaries of the specified partition.
SVC 103: To handle an I/O operation for SYSFIL on an FBA device.
SVC 105: To reset subsystem information relating to the ending task.
SVC 107: To retrieve information from or modify information contained in
         a supervisor control-block.
SVC 110: To release resources locked for the ending task.
SVC 112: To update stored device-assignment information.
SVC 113: To release cross-partition communication control-blocks (XPCC),
         if any.
SVC 114: To deallocate VIO work areas.
```

Data Areas Used: Areas in the supervisor as listed below:

- CHANQ, the channel queue table.
- MAPCOMR, the partition communication region.
- PHLST, the phase-load list.
- PIBTAB, the partition information block table.
- PUBTAB, the physical unit block table.
- PUB2TAB, the physical unit block extension table.
- RASLINK, the RAS linkage area.
- RFTABLE, the recorder-file table.
- SYSCOM, the system communication region.

Messages Caused

```
0P91I   TERMINATOR ROUTINE CANCELED, CANCEL CODE = nn
```

Messages Issued: The one caused by this phase.

Input: None.

Output

- Task control data reset for initiation of another task.
- Message as required (see "Messages Issued" above).

Exit Normal: To the supervisor via:

- SVC 11 for ending the attention task or a main task.
- SVC 39 for ending a subtask.

Exit Error: None.

Register Use

0-6	Work registers.
8	Pointer to PUBTAB.
9	Address of SYSCOM (in the supervisor).
10	Address of the affected partition's MAPCOMR (in the supervisor).
11	Base register for this phase.
13-15	Work registers.

Sequence of Operation

IJBEOTSK: Establishes addressability for the phase.

EOT0000: Gets the addresses of MAPCOMR and SYSCOM; gets the cancel code. If the terminator routine itself has been canceled, this routine writes message 0P91I and passes control to EOT3000.

EOT0100: Gets the partition boundaries, using the service of SVC 98.

EOT0200: Scans tables PUBTAB and PUBOWNER for telecommunication devices active and owned by the ending task; issues a halt I/O request for each of those devices (SVC 25 or 27); dequeues the halted I/O operation if the ending task is a main task.

Reactivates channel-queue entries deactivated by BTAM or OLTEP. To do so, the routine enqueues these entries to the free list of the channel queue.

EOT0400: Issues an SVC 3 to eventually dequeue all I/O requests not yet completed for the ending task. Cancels the task-timer interval, if one was specified. Issues SVC 110 to release all resources owned by the ending task.

Returns control to the supervisor (via SVC 11) if the ending task is a system task.

EOT0500: If a MICR DTF table exists, clears the entries related to the ending task.

Issues SVCs to request supervisor services as required; these SVCs are:

- SVC 36 to free tracks held by the ending task.
- SVC 92 to reset XECBs.
- SVC 105 to reset task-related subsystem information.
- SVC 107 to reset the DASD file-protect indicator.
- SVC 113 to reset XPCCs.
- SVC 114 to deallocate VIO work areas.

Issues SVC 39 to return control to the supervisor if the ending task is a subtask.

EOT0800: Receives control if the ending task is a main task. Resets the EREP-ownership flag if this flag is related to that task.

Waits until RAS has completed if this is active; resets temporary assignments for SYSUSE.

EOT1200: Posts the PUB2 extensions of the devices owned by the ending main task as being closed. For a tape unit, records tape error statistics, if any have been accumulated, in the system's recorder file (IJSYSRC) by issuing SVC 44 and, subsequently, clears that unit's PUB2 entry.

EOT1440: Performs the following reset functions:

- Clears the phase-load list.
- Resets the BTAM indicator in MAPCOMR.
- Resets RPS controls (by calling phase RPSLOAD).
- Resets the GETVIS OPEN indicator in MAPCOMR.
- Resets the appendage indicator in the partition's PIB.

EOT1700: For an ending task running in real mode in a 370 mode environment, issues SVCs to request supervisor services as shown:

- SVC 62 to clear the partition's GETVIS area.
- SVC 58 to disconnect (invalidate) page frames.

EOT1800: If the ending main task is the first one to run after IPL, issues SVC 112 to assign system logical units to that task's partition.

EOT2000: Loads the job control root phase $JOBCTLA by issuing SVC 4. Sets up the PSW for the partition and the partition's save area; issues SVC 11 to return control to the supervisor.

Program Organization Information

Program organization information for phase $IJBSEOT, which has a one-to-one relation to module IJBEOTSK, is not meaningful.

Data Area Information

There is no meaningful data area information for $IJBSEOT.

Chapter 6. Miscellaneous $IJBSxxx Service Routines

This chapter describes services other than task termination which are available through phases that reside in the shared virtual area. The services described in this chapter are:

Hard-copy-file support – Phase $IJBSHCF
Symbolic label-access support – Phase $IJBSLA
Partition-initiation support – Phase $IJBSTRT.

$IJBSHCF – Hard-Copy-File Support

Phase $IJBSHCF contains the routines which write data into and retrieve data from the hard-copy-file (HCF); this phase is reentrant and resides in the SVA. The services of the phase are available via the HCF access macros listed under "External Input" below; for a detailed description of these macros, refer to *Diagnosis Reference: Supervisor*.

Phase $IJBSHCF is made available for system use when job control (phase $JOBCTLC) processes the first // JOB statement. At this time, job control issues a LOAD macro specifying TXT=NO in order to verify that the phase $IBHSHCF is available in the SVA. To make the phase accessible, job control stores the phase's entry point in the field AHCFIOMD of table CRTTAB (in the supervisor).

The phase expects to receive control via a macro call with the entry address of the phase contained in register 15 and the return address in register 14.

External Input

Reg. 0 =
Pointer to the request parameter-list

Reg. 1 =
Pointer to the HCFCB to be used

Reg. 2 =
Service-request code issued by system code via one of the following macros:

POINTHCF: To open the HCF and set the controls for later access to the HCF.
WRITEHCF: To write an HCF record into the HCF.
READHCF: To read an HCF record from the HCF.
MODHCF: To change the processing direction for reading from the HCF.
SKIPHCF: To skip a specified number of HCF records while reading or to skip to the end or the beginning of the HCF, whichever is indicated.
CLOSEHCF To end the HCF access started by a particular POINTHCF macro.

For a list of the request codes, refer to Figure 8 on page 113 in the section "Design Information".

Reg. 13 =
Save area address if the request macro is POINTHCF.

Note: Field HCFCBWRT in CRTTAB contains the address of the systems HCFCB initialized when the HCF was opened for write.

External Output

- An HCF record written to the HCF if the requested service was an HCF-write.
- An HCF record in an available I/O area if the requested service was an HCF read.
- The applicable return code in register 15 (for a complete list of these codes, refer to the description of these macros in *Diagnosis Reference: Supervisor*).

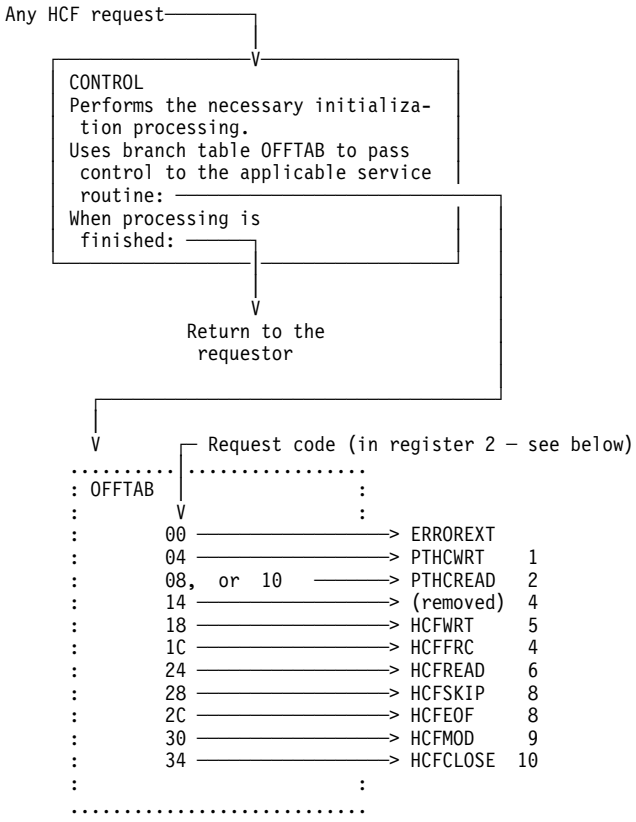
The routines of the HCF support neither cause nor issue any messages.

Design Information

The HCF support consists of a mainline routine, CONTROL, that initializes the processing of a service request by calling one of the processing routines; the routine provides also for the necessary exit processing.

Figure 8 on page 113 shows a control-flow overview within phase $IJBSHCF. This overview is followed by a summary on each of the involved routines.

The routines of the phase are described in alphabetic order of their names.



Request Code	Type of Request
00	An error condition exists.
04	POINTHCF WRITE
08	POINTHCF READ,PRINTLOG
0C	(removed)
10	POINTHCF READ,REDISPLAY
14	(removed)
18	WRITEHCF ...
1C	FORCEWRITE.., with GETVIS
24	READHCF ...
28	SKIPHCF ...,COUNT
2C	SKIPHCF ...,EOF
30	MODHCF ...
34	CLOSEHCF ...

Figure 8 (Part 1 of 6). HCF-Support Control Flow

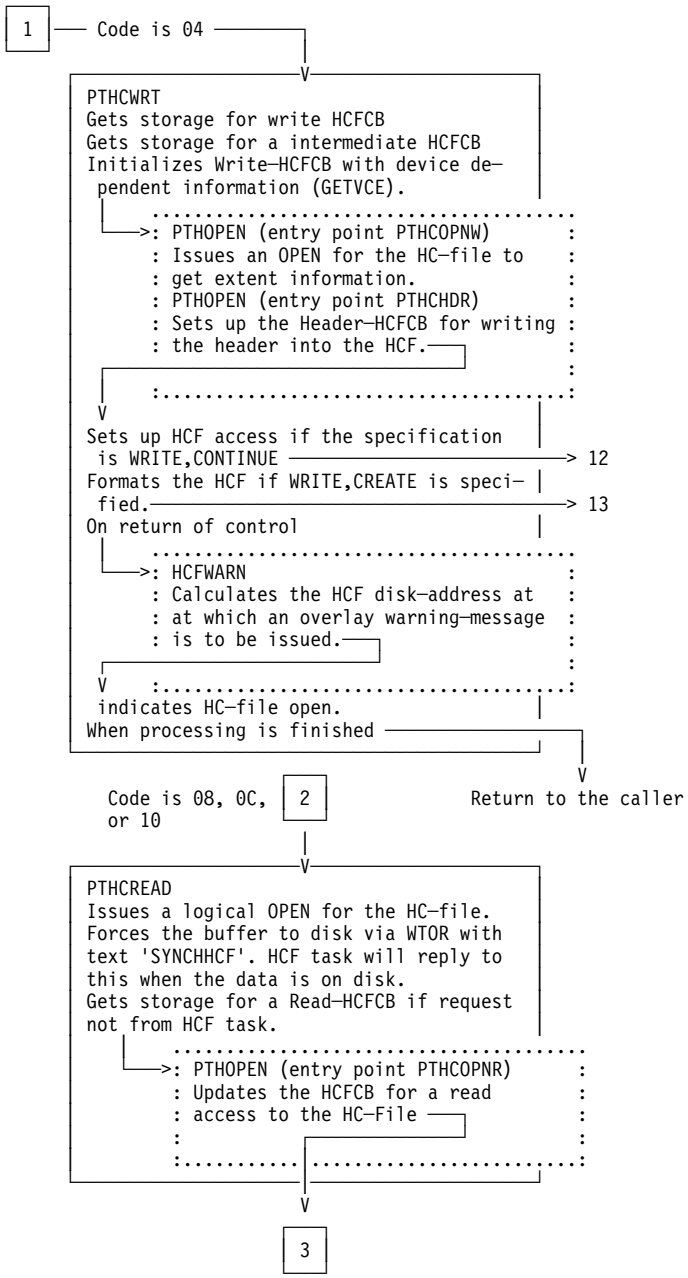


Figure 8 (Part 2 of 6). HCF-Support Control Flow

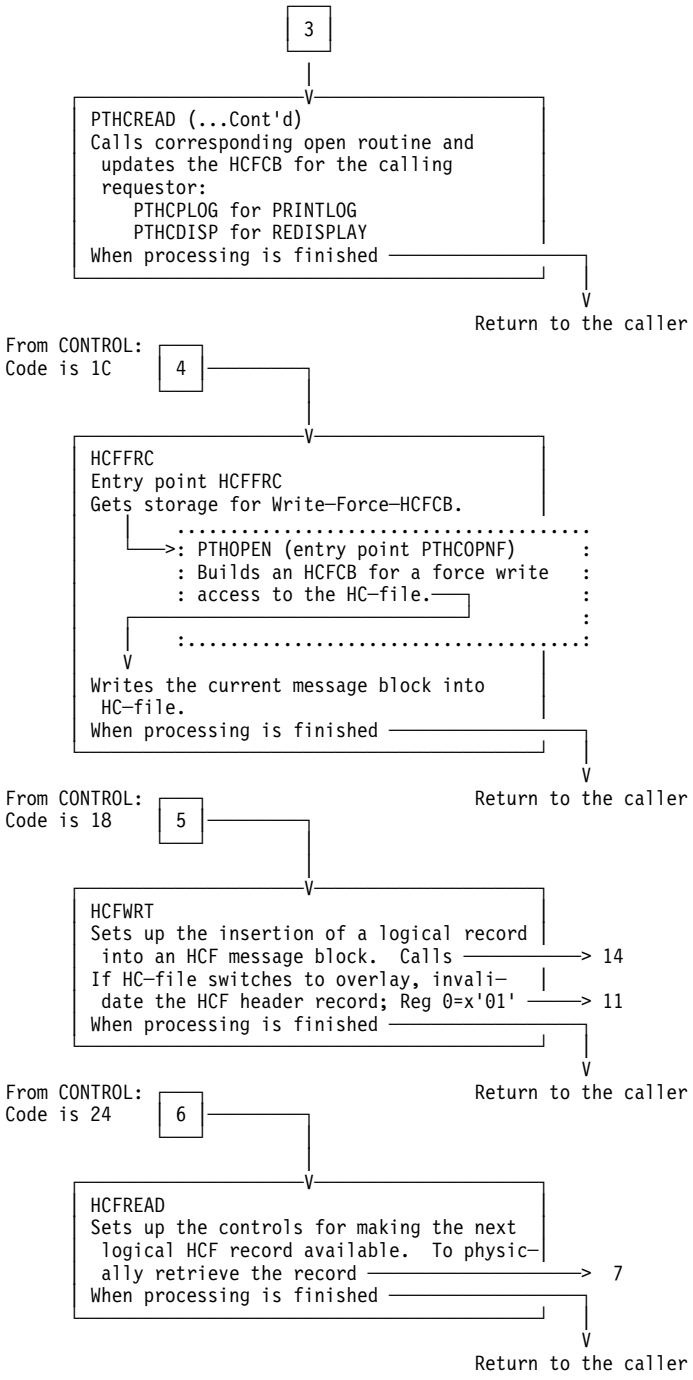


Figure 8 (Part 3 of 6). HCF-Support Control Flow

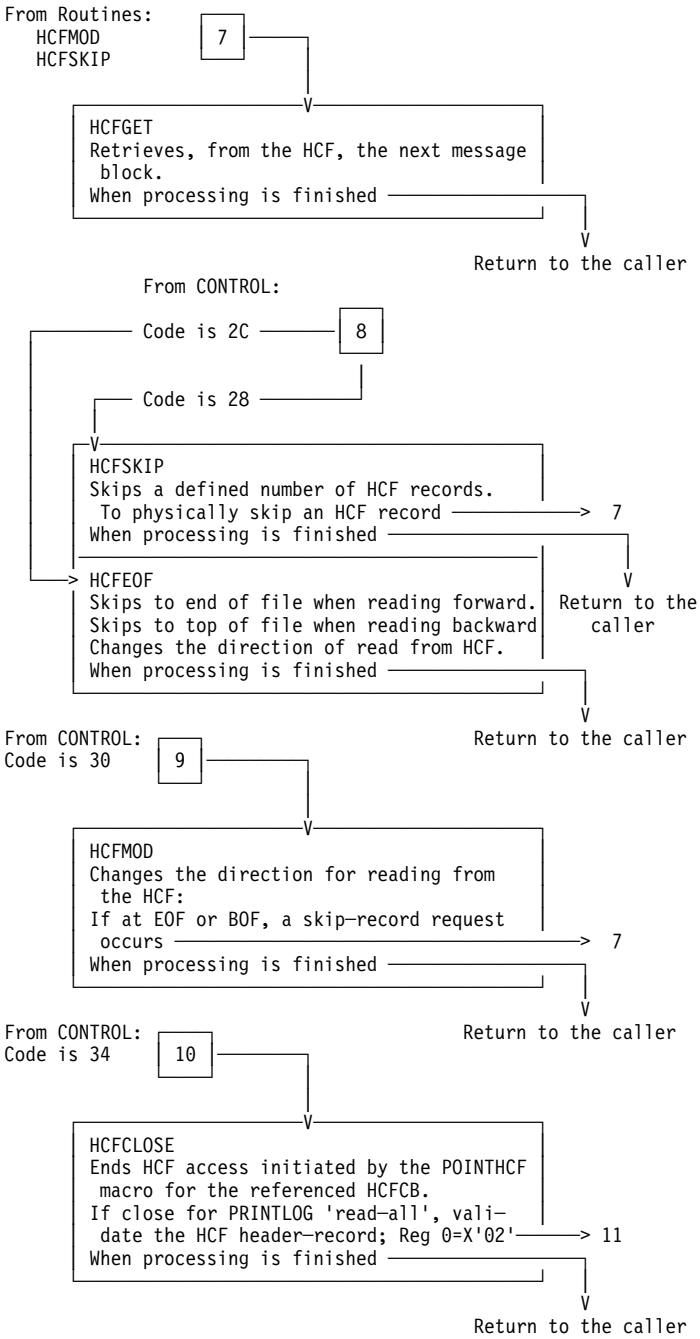


Figure 8 (Part 4 of 6). HCF-Support Control Flow

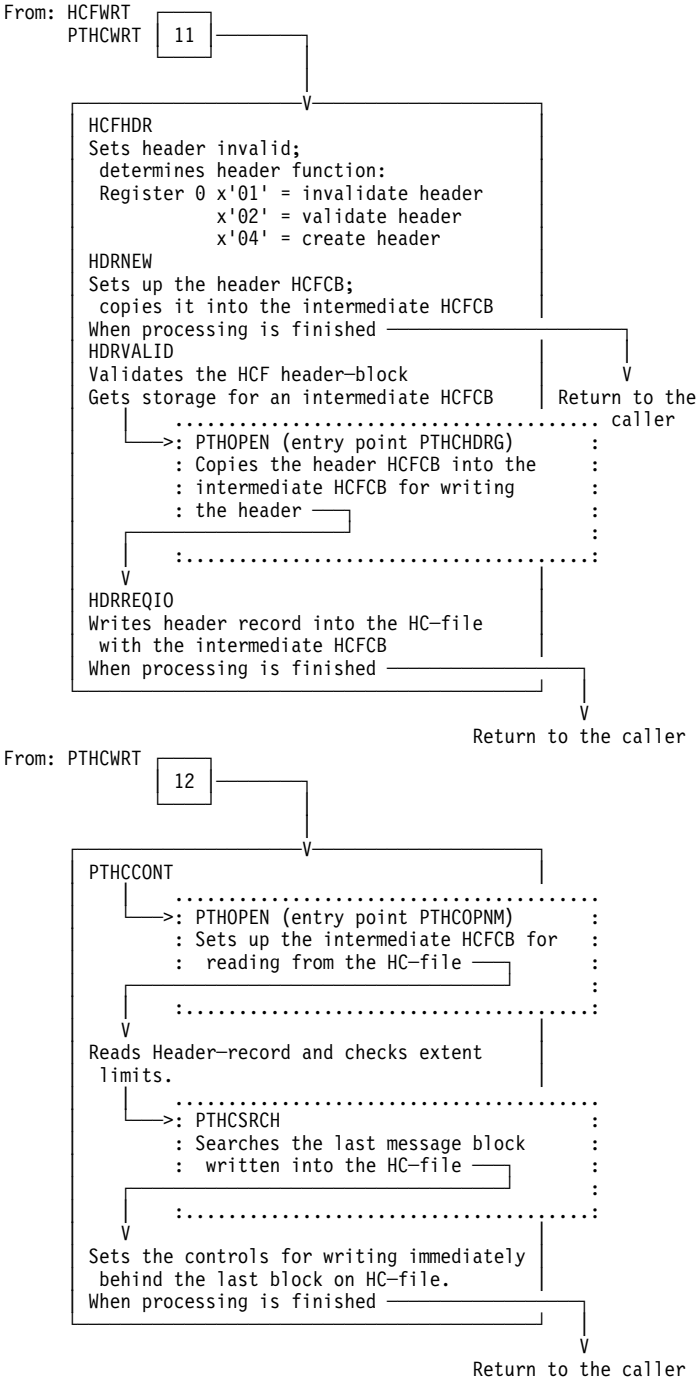


Figure 8 (Part 5 of 6). HCF-Support Control Flow

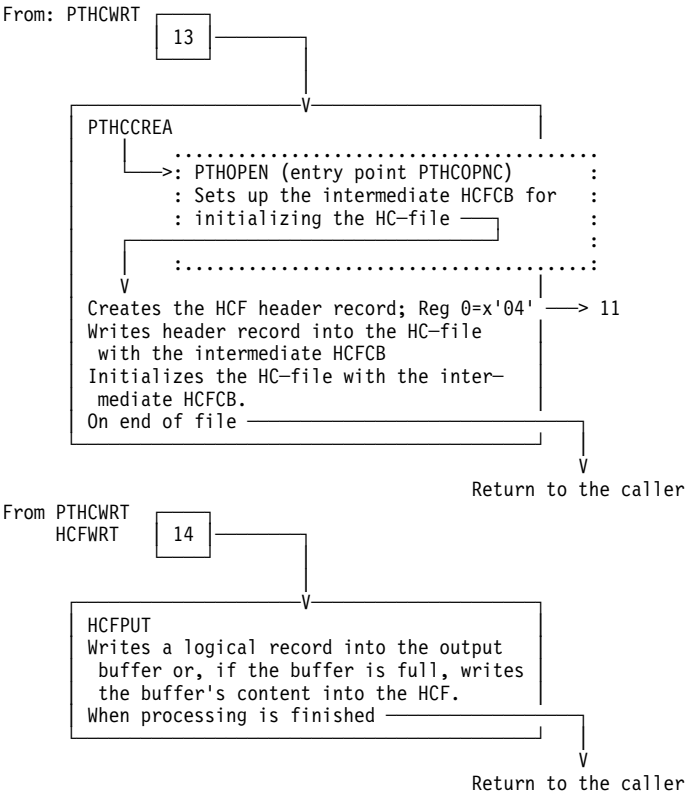


Figure 8 (Part 6 of 6). HCF-Support Control Flow

Routine CONTROL – Initialization and Exit Processing

Entry Points

- CNTEPRD to do all read functions.
- CNTEPPNT to do all point (logical open) functions.
- CNTEPWRT to do all write functions.

Function: Performs the necessary support-initialization processing; Selects the applicable HCF access routine and transfers control to that routine.

Called By: the different HCF access macros.

Other Routines Called: See part 1 of Figure 8 on page 113.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).

Input: See "External Input" earlier in this section on the HCF support.

Output: See "External Output" earlier in this section on the HCF support.

Exit Normal: To the caller with register 15 set to zero.

Exit Error: Return via the error handler, to the caller; return code in register 15.

Register Use: Register 13 points to a register save area; the routine passes the contents of all other registers unchanged to the called processing routine.

Routine PTHCWRT – Open the HCF for Write Access

Function: Issues an OPEN for the HCF; builds an HCFCB that matches the device characteristics of the HCF device; builds the HCF-header block and makes the HCFCB available.

Formats the HCF if the requestor specifies WRITE,CREATE ...; sets up the controls for writing behind the message block written last if WRITE,CONTINUE ... was specified by the requestor.

Called By: Routine CONTROL (request code 04).

Other Routines Called

- PTHCOPNW: To initiate HCF open processing.
- PTHCCONT: To set up continuation of HCF write access following the record written last into the HCF.
- PTHCCREA: To initialize (format) the HCF.
- HCFWARN: To provide the HCF-full warning address.

Subroutines Used: None.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).

Input: Contents of registers 0, 1, 2, 10, and 13 (see "Register Use" below).

Output: Either of the following:

- The HCFCB initialized for the processing of WRITEHCF requests and a return code of zero in register 15.
- One of the return codes listed below, contained in register 15:
  04 = Inconsistent state: wrong owner of write of HCFCB;
  08 = No record found or incorrect length.
  0C = Unrecoverable I/O error.
  10 = Incorrect HCF referenced.
  18 = Unauthorized POINTHCF request.
  1C = The HCF is not accessible.
  20 = The HCF is too small.
  24 = Execution of GETVIS failed.
  28 = The HCF does not match the extent specifications.

Exit Normal: To the requestor with register 15 set to zero.

Exit Error: To the caller with return code in register 15.

Register Use:

0	Function code (X'00' for CREATE; X'04' for CONTINUE) on input; work register otherwise.
1	Pointer to POINTHCF macro generated parameter list.
2	Request (branch) code on input; work register.
3-7	Work registers.
8	Subroutine-call link register.
9	Pointer (and base register for access) to a write-HCFCB operation.
10	Base register for access to CRTTAB.

11 Pointer to the (temporary) HCFCB if in the GETVIS area.
12 Work register.
13 Pointer to the requestor's save area.
14 (Sub)routine-call link register.
15 Base register for the phase; return-code register on end of processing.

Sequence of Operation:

1. Gets storage for the write-HCFCB; obtains GETVIS space to accommodate the (temporary) HCFCB.
2. Retrieves (via GETVCE macro) the characteristics of the HCF device; initializes fields HCFBLKLN (HCF block-length)and HCFDEVTP (device type) in CRTTAB.
3. Performs the necessary open processing and initializes the HCF header; calls routine PTHOPEN (via entry point PTHCOPNW and via entry point PTHCDR) for that purpose.
4. For a POINTHCF with CREATE specified, the routine calls routine PTHCCREA to initialize the HCF.
5. For a POINTHCF with CONTINUE specified, the routine calls routine PTHCCONT to set up continuation with the referenced HCFCB.
6. Calls routine HCFWARN to provide the HCF-full warning address.
7. Uses subroutine PWRTFRVS to free previously obtained GETVIS space.
8. Sets the HCF open indicator CRTFLGHC

Subroutine PTHCCREA – Format the HCF Extent

Function: Initializes the HC-file extent and creates the header record.

Called By: Routine PTHCWRT.

Other Routines Called: PTHOPEN (entry point PTHCOPNC) to initialize the required HCFCB.

Subroutines Used

- HCFUPD: To update the HCF access address.
- HCFHDR: To create the HC-header record.
- REQIO: To write a block into the hard-copy file.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).

Input: Contents of the following registers: 1, 9, 10, 11, 13, and 14 (see "Register Use" below).

Output

- The initialized HC-file.

Exit Normal: Return to the caller.

Exit Error: Return to the caller.

Register Use

1 Pointer to the parameter list supplied by macro POINTHCF; work register.
2-7 Work registers.
9 Pointer (and base register for access) to the currently used write-HCFCB.
10 Pointer (and base register for access) to CRTTAB.

11 Pointer (and base register for access) to the HCFCB copy in the GETVIS area.
13 Pointer to the caller's register-save area.
14 Return register; (sub)routine-call link register.
15 Base register for the routine.

Sequence of Operation

PTHCCREA: Builds the header record, writes it into the HC-file; initializes the write HCFCB for further processing; initializes the HC-file with a special record.

Subroutine PTHCCONT – Provide for Write Continuation

Function: Searches for the block written last into the HCF; sets the controls for writing the next block immediately behind the located one.

Called By: Routine PTHCWRT.

Other Routines Called: PTHOPEN (entry point PTHCOPNM) to initialize the required HCFCB.

Subroutines Used

- HCFUPD: To update the HCF access address.
- PTHCSRCH: To search for the last written message block on the HC- file.
- REQIO: To retrieve a block from the hard-copy file.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).

Input: Contents of the following registers: 1, 9, 10, 11, 13, and 14 (see "Register Use" below).

Output: Either of the following:

- The disk address at which to continue writing message blocks and a return code of zero in register 15.
- In register 15 a return code of X'28' if the requestor supplied invalid extent specifications.

Exit Normal: Return to the caller.

Exit Error: Return via the error handler, to the caller.

Register Use

1 Pointer to the parameter list supplied by macro POINTHCF; work register.
2-7 Work registers.
9 Pointer (and base register for access) to the currently used write-HCFCB.
10 Pointer (and base register for access) to CRTTAB.
11 Pointer (and base register for access) to the HCFCB copy in the GETVIS area.
13 Pointer to the caller's register-save area.
14 Return register; (sub)routine-call link register.
15 Base register for the routine; return code register on end of processing.

Sequence of Operation

PTHCCONT: Calls the routines (subroutines) listed below in order to read the header record; to verify the HCF extent specifications supplied by the requestor; to locate the wrap-position (last written block before reipl); to set up the write HCFCB for continue writing just behind the wrap position:

PTHCOPNM: Builds an intermediate HCFCB for read access.

HCFUPD: To set the disk-read address to the next message block.

PTHCSRCH: To locate the last written message block (wrap position) by help of the cycle byte change.

REQIO: To read in HC-blocks for cycle byte change checking.

Subroutine PTHCSRCH – Locate Block with Changed Cycle Byte: Provides the disk address of the last message block written into the HC-file by use of the cycle-byte change.

Input: Contents of registers 1, 8, and 9 (see "Register Use" below.

Output: Temporary HCFCB with updated disk addresses.

Register Use

1	Pointer (and base register for access) to a temporary read HCFCB in the GETVIS area.
2-6	Work registers.
7	Subroutine-call link register.
8	Return register.
9	Pointer (and base register for access) to the write-HCFCB being used.
10	Base register for access to CRTTAB.
15	Base register for the subroutine.

Routine PTHCREAD – Open the HCF for Read Access

Function: Issues a logical OPEN for the HCF; builds an HCFCB that matches the device characteristics of the HCF device.

Called By: Routine CONTROL (request code X'08', X'0C', or X'10').

Other Routines Called

- PTHOPEN via entry point PTHCOPNR to build an HCFCB for read access

Subroutines Used: One of the following to obtain required extent information:

- PTHCDISP: If the POINTHCF request was originated by REDISPLAY.
- PTHCPLOG: If the POINTHCF request was originated by PRINTLOG.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- The GETVIS'ed or pre-allocated (HCF task) HCFCB for read access.

Input: Contents of registers 1, 2, 10, and 15 (see "Register Use" below).

Output: Either of the following

- The HCFCB initialized for the processing any of the following requests: READHCF, MODHCF, and SKIPHCF; a return code of zero in register 15.

- One of the return codes listed below, contained in register 15:

```
08 = No record found or incorrect length.
0C = Unrecoverable I/O error.
14 = The referenced HCF has not been opened.
18 = Unauthorized POINTHCF request.
1C = The HCF is not accessible.
20 = The HCF is too small.
24 = GETVIS failed.
28 = The HCF does not match the extent specifications.
```

Exit Normal: Return to the caller.

Exit Error: Via the error handler, to the caller.

Register Use

0	Work register.
1	Pointer to the parameter list passed by the POINTHCF macro.
2	Contains the request (branch) code.
3-5	Work registers.
7	Work register.
9	Pointer to the HCFCB in the GETVIS area.
10	Base register for access to CRTTAB.
12	Work register.
13	Pointer to the requestor's save area.
14	(Sub)routine-call link register.
15	Base register for the phase.

Sequence of Operation

PTHCREAD: Picks up the length of the I/O buffer and obtains GETVIS area needed to accommodate the referenced read-HCFCB. Initializes the HCFCB for read access via PTHOPEN (entry point PTHCOPNR).

Examines the request code (in register 2) and calls the appropriate routine for providing required extent information:

Register 2:

08	initialize HCFCB for PRINTLOG (PTHCPLOG).
0C	initialize HCFCB for LISTLOG (PTHCLILO).
10	initialize HCFCB for REDISPLAY (PTHCDISP).

Subroutine PTHCDISP – Set HCFCB for Redisplay: Sets the controls, in the currently used HCFCB, for a redisplay of message lines recorded in the HCF.

Input: Contents of registers 9, 13, and 14 (see "Register Use" below).

Output: HCFCB updated for subsequent reads for a redisplay of message lines.

Register Use

0	Work register.
1	Pointer to POINTHCF macro generated parameter list; work register.
2-4	Work registers.
9	Pointer (and base register for access) to the temporary HCFCB in the GETVIS area.
10	Base register for access to CRTTAB.

13 Pointer to the requestor's save area.
14 (Sub)routine-call link register.
15 Base register for the subroutine; return-code register at end of processing.

Subroutine PTHCPLOG – Provide Extent Information, PRINTLOG: Provides HCF extent information if a PTHCREAD request has been issued by the PRINTLOG program.

Input: Contents of registers 9, 10, 13, and 14 (see "Register Use" below).

Output: Updated extent information in the HCFCB.

Register Use

0 Work register.
1 Pointer to POINTHCF macro generated parameter list; work register.
2-4 Work registers.
9 Pointer (and base register for access) to the temporary HCFCB in the GETVIS area.
10 Base register for access to CRTTAB.
13 Pointer to the requestor's save area.
14 (Sub)routine-call link register.
15 Base register for the subroutine.

Routine HCFFRC – Force a Write to the HCF

Function: Writes a message block into the HCF. The buffer from which the message is written depends on the entry point used by the requesting routine. If the entry point is:

HCFFRC – From a HCFCB which is GETVIS'ed by this routine.

Called By: Routine CONTROL via entry point

HCFFRC (request code X'1C').

Subroutines Used: REQIO: To execute the channel program for HCF access.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).
- The write HCFCB referenced by the request macro.

Input: Contents of registers 1, 9, 10, and 13 (see "Register Use" below).

Output: Either of the following:

- The write buffer's contents written into the HCF and a return code of 0 in register 15.
- One of the return codes listed below, contained in register 15:
  04 = Execution of FREEVIS failed. FREEVIS return code in byte 0.
  08 = No record found or incorrect length.
  1C = An irrecoverable I/O error occurred.
  10 = The HCF device is not ready.
  24 = Execution of GETVIS failed. GETVIS return code in byte 0.

Exit Normal: Return to the caller.

Exit Error: Via error handler, to the caller.

Register Use

0 Work register.
1 Pointer to the HCFCB to be used.
2-3 Work registers.
7 Work register.
9 Pointer to the GETVIS area allocated by the supervisor.
10 Pointer to CRTTAB.
13 Pointer to the caller's register-save area.
14 (Sub)routine-call link register.
15 Base register, return-code register at the end of processing.

Sequence of Operation

HCFFRC: Obtains GETVIS space for a HCFCB sufficient to meet the requirements of the HCF device; initializes a HCFCB into this area by calling PTHOPEN (via entry point PTHCOPNF).

Writes the message block into the HCF by calling subroutine REQIO; issues a FREEVIS request if the routine received control via HCFFRC.

Routine HCFWRT – Set Up Logical-Record Write

Function: Inserts an HCF record in the currently used output block. If the record does not fit, the routine writes this block into the hard-copy file and inserts the currently processed HCF record into a new output block.

Called By: Routine CONTROL (request code X'18').

Other Routines Called

- HCFHDR: To update the HCF header record.
- HCFPUT: To insert a logical record into a message block or write a message block into the HCF, whichever is required.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- COMREG, the requesting partition's communication region (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).
- The currently used write-HCFCB.

Input: Contents of registers 0, 10, and 13 (see "Register Use" below).

Output: Either of the following

- Updated message block and a return code of zero in register 15.
- In register 15, one of the return codes listed below:
  - 04 = Inconsistent state: opcode in the HCFCB not write; requestor does not own the write HCFCB or requestor is not HCF task.
  - 08 = No record found or incorrect length.
  - 0C = Irrecoverable I/O error.
  - 10 = The HCF device is not ready.
  - 14 = Message lines previously written to the HCF are being overwritten.
  - 18 = An HCF-full warning message is to be issued.

Exit Normal: Return to the caller.

Exit Error: Via the error handler, to the caller.

Register Use

0	Pointer to the logical record that is to be inserted.
10	Pointer to CRTTAB.
13	Pointer to the caller's register-save area.
14	Return register; (sub)routine-call link register.
15	Base register for the routine.

Sequence of Operation

HCFWRT: The HCF task only is allowed to write to HC-File. If HC-file enters overlay mode, the routine HCFHDR is called to invalidate the header record.

Routine HCFREAD – Make Next Record Available

Function: Sets up the controls for making the next logical HCF record available.

Called by: Routine CONTROL (request code X'24').

Other Routines Called: HCFGET to read the next logical record into the available record buffer.

Data Areas Used: The read HCFCB to be used for the operation.

Input: The contents of registers 0, 1, 10, 13, and 14 (see "Register Use" below).

Output: Either of the following:

- The next record in the area defined by the requestor and a return code of zero in register 15.
- One of the return codes listed below, contained in register 15:
  04 = Inconsistent state: invalid opcode in the HCFCB.
  16 = The beginning of the HCF has been reached – There is no new record available.
  24 = The end of the HCF has been reached – There is no new record available.
  28 = The next record has been overwritten – There is no new record available.

Exit: Return to the caller.

Register Use

0	Pointer to the requestor's I/O area.
1	Pointer (and base register for access) to the applicable HCFCB.
3	Work register.
4	Pointer to the record-length field.
5	Pointer to the record in the message block.
6	Sequence number of the record in the message block.
7	(Sub)routine-call link register; work register.
10	Pointer to CRTTAB.
13	Pointer to the caller's register-save area.
14	Return register; (sub)routine-call link register.
15	Base register for the routine; return code register.

Sequence of Operation

HCFREAD: Calls routine HCFGET to read the next HCF record into the area pointed to by register 0, except if end-of-HCF or beginning-of-HCF is indicated.

Sets the applicable return code (see "Output" above).

Subroutine HCFPUT – Write Logical Record

Function: Writes a logical record into the output buffer; writes that buffer's contents into the HCF if necessary.

Called By: Either of routines HCFWRT and PTHCWRT.

Subroutines Used

- REQIO: To write a block into the HCF.
- HCFUPD: To update the HCF access addresses.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- The currently used HCFCB.

Input: Contents of registers 0, 1, 10, 13, and 14 (see "Register Use" below).

Output

- A logical HCF record either in the message block currently being built or, as part of a message block in the HCF.
- A return code in field HCRETCOD:
  14 = An HCF wrap-around has occurred.
  18 = The HCF-full warning message is to be issued.

Exit: Return to the caller.

Register Use

0	Pointer to the record to be written.
1	Pointer to the HCFCB being used.
3-4	Work registers.
5	Length-1 of the record without trailing blanks
10	Pointer to CRTTAB.
13	Pointer to the caller's register-save area.
14	Return register; (sub)routine-call link register.
15	Base register for the routine.

Sequence of Operation

HCFPUT: Builds a logical record; suppresses trailing blanks, if any.

If this logical record does not fit into the currently used buffer, the routine executes subroutine REQIO to write the buffer's contents into the HCF.

For a block written into the HCF: Updates the disk I/O address (by using subroutine HCFUPD) and gets a new message-block build buffer; sets an indicator if the limit for an HCF-full warning has been reached; sets the wrap-around indicator if this has occurred.

HCFPUT30: Inserts a logical record into a message block; updates the logical-record count of the current-record address.

Subroutine HCFGET – Retrieve HCF Records

Function: Retrieves, from the HCF, the next message block.

Called By: Any of the routines HCFMOD, HCFREAD, and HCFSKIP.

Subroutines Used

- HCFUPD: To update the disk addresses for HCF access.
- REQIO: To read a message block from the HCF.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- The write HCFCB referenced by the requesting macro.

Input: Contents of registers 0, 1, 10, and 14 (see "Register Use" below).

Output: Either of the following:

- For a successful completion
  - The next HCF record stored at the address contained in register 0.
  - The updated HCF access addresses.
  - A return code of zero in field HCRETCOD.
- A return code of 04 = (incorrect record length detected) in field HCRETCOD.

Exit: Return to the caller.

Register Use

0	Pointer to the area that is to contain the next logical record.
1	Pointer to the HCFCB to be used.
3	Work register.
4	Pointer to the record-length field.
5	Pointer to the record in the currently processed message block.
6	Sequence number (within the message block) of the currently processed logical record.
7	(Sub)routine-link register; work register.
10	Pointer to CRTTAB.
14	Return Address; (sub)routine-link register.

Sequence of Operation

HCFGETRD: Retrieves from the HCF, by executing subroutine REQIO, the message block next in sequence:

- When the routine receives control for the first time for an HCF read operation.
- Whenever the last record of a message block was made available already on a previous service request.

Uses the subroutine HCFUPD to adjust the HCF-access disk-addresses; makes the next logical record available after having it rebuilt for display; adjusts the record place-holder to the next logical record.

For a SKIP NEXT request, the routine updates the logical record count.

If end-of-file or beginning-of-file is reached, the routine sets the applicable return code.

Routine HCFSKIP – Skip HCF Records

Function: Skips HCF records. The kind of record-skip operation performed depends on the entry point used by the calling routine. If this is:

HCFSKIP – The number of records as indicated by register 0.
HCFEOF – To the end of the HCF if the direction of read is forward; to the beginning of the HCF if the direction of read is backward. This involves a change of the read direction.

Called By: Routine CONTROL via entry point

HCFSKIP for a request code of X'28'.
HCFEOF for a request code of X'2C'.

Other Routines Called: HCFGET to physically skip one HCF record.

Data Areas Used: The currently used HCFCB.

Input: Contents of registers 0, 1, 10, 13, and 14 (see "Register Use" below).

Output: Either of the following:

- The HCFCB updated and a return code of zero in register 15.
- One of the return codes listed below, contained in register 15:
  04 = Inconsistent state: invalid opcode in the HCFCB or negativ skip count requested.
  14 = For a skip-records request, beginning of file reached before the requested number of records had been skipped; Register 0 contains the number of records the service was unable to skip.
  18 = For a skip-records request, end of file reached before the requested number of records had been skipped; Register 0 contains the number of records the service was unable to skip.

Exit: Return to the caller.

Register Use

0	Number of records to be skipped (if the routine is called to service a skip-records request).
1	Pointer (and base register for access) to the applicable HCFCB.
3	Work register.
4	Pointer to the record-length field.
5	Pointer to the record in the message block.
6	Sequence number of the record in the message block.
7	Routine-link register; work register.
10	Pointer to CRTTAB.
13	Pointer to the caller's register-save area.
14	Return register.
15	Base register for the phase.

Sequence of Operation

HCFSKIP: Sets the skip function-indicator and the skip count; successively executes routine HCFGET to cause a record to be skipped for each execution of that routine.

Sets the applicable return code (see "Output" above).

HCFEOF: Sets the HCF access address to:

- The file's end address if the direction of read is forward.
- The file's start address if the direction of read is backward.

Switches the direction for reading from the HCF; sets the first-time indication for routine HCFGET.

Routine HCFMOD – Change Direction of Read from the HCF

Function: Changes the direction of read from HCF for the subsequent read requests.

Called By: Routine CONTROL (request code X'30').

Other Routines Called: HCFGET to skip an HCF record.

Data Areas Used: The currently used read HCFCB.

Input: Contents of registers 0, 1, 10, 13, and 14 (See "Register Use" below).

Output: Either of the following:

- The direction-of-read indication in the referenced HCFCB set to the requested direction and a return code of zero in register 15.
- In register 15 the return code 04 (= Inconsistent state: invalid opcode in HCFCB).

Exit: Return to the caller.

Register Use

0 Read direction code (see "Sequence of Operation" below).
1 Pointer to the HCFCB being used.
4 Work register.
7 Routine-link register.
10 Pointer to CRTTAB.
13 Pointer to the caller's register-save area.
14 Return register.
15 Base register for the phase.

Sequence of Operation

HCFMOD: Changes the direction-of-read indication as requested by the code in register 0:

00 = Set the direction of read to forward.
04 = Set the direction of read to backward.
08 = Reverse the currently active read direction.

If the request occurs at EOF or BOF of the HCF, the routine resets this EOF (or BOF) indication and executes routine HCFGET in order to skip one record.

Routine HCFCLOSE – End HCF Access

Function: Ends the requestor's HCF access by deactivating the referenced HCFCB and freeing the area used by this HCFCB.

Called By: Routine CONTROL (request code X'34').

Other Routines Called: HCFHDR to update the HCF header record, in case of PRINTLOG request.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).
- SYSCOM, the system communication region (in the supervisor).
- The corresponding HCFCB referenced by the request macro.

Input: Contents of registers 1, 10, 13, and 14 (see "Register Use" below).

Output: Either of the following:

- The area of the HCFCB cleared and freed.
- A return code in register 15:
  04 = Request inconsistent with the referenced HCFCB.

Exit: To the caller.

Register Use

0 Work register.
1 Pointer to the HCFCB to be used.
2-4 Work registers.
5 Contains the function code (0 = close for read HCFCB; 4 = close for write HCFCB.
6 Work register.
10 Pointer to CRTTAB.
13 Pointer to the caller's register-save area.
14 Return register; routine-link register.
15 Base register for the phase.

Sequence of Operation

HCFCLOSE: Ensures that the request is consistent with the operation code in the requestor's HCFCB.

If the HCFCB is owned by the HCF task, the routine:

1. Issues an SVC 106 instruction to set the storage-protection key for the write-HCFCB to zero.

If the HCFCB contains the operation code RDALL (a request from the PRINTLOG program), the routine:

1. Sets the address in CWRPDADR to the end address.
2. Executes routine HCFHDR to update the wrap-around address in the HCF header-record.
3. Resets the PRINTLOG-noselect-active and the overlay indicators.

Clears the HCFCB and frees the virtual storage occupied by this HCFCB;

Subroutine HCFHDR – Update the HCF Header-Block

Function: Updates the HCF header-block and writes this to disk.

Called By: One of the following routines:

HCFCLOSE
HCFWRT
PTHCWRT

Subroutines Used

- REQIO: To write a block of data into the HCF.
- PTHCHDRG: To initialize an header-HCFCB.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).

Input: Contents of registers 0, 1, 10, and 14 (see "Register Use" below).

Output: Either of the following:

- An updated HCF header-block and a return code of zero in register 15.
- One of the following return codes in register 15:
  04 = Inconsistent input or FREEVIS failed. FREEVIS return code in byte 0.
  08 = No record found or incorrect length.
  0C = Irrecoverable I/O error.
  10 = The HCF device is not ready.
  18 = Execution of GETVIS failed. GETVIS return code in byte 0.

Exit Normal: Return to the caller.

Exit Error: To the caller with return code in register 15.

Register Use

0	Requested header function.
1	Pointer to the HCFCB being used.
2-3	Work registers.
5-8	Work registers.
10	Pointer to CRTTAB.
13	Pointer to the caller's register-save area.
14	Return Address; (sub)routine-link register; work register.

15 Base register for the routine.

Sequence of Operation

HCFHDR: Ensures that the operation code in the HCFCB is consistent with the service request being processed. Sets byte HDRFLAG of the HCF header block as shown below and builds the fields of the block as required.

Moves the updated header to the output buffer and calls subroutine REQIO to write the header into the hard-copy file.

Subroutine HCFWARN – Provide HCF-Full Address:

Function: Calculates the HCF disk-address at which the overlay warning-message is to be issued.

Called By: Routine PTHCWRT or HCFHDR.

Data Areas Used

- CRTTAB, the CRT table (in the supervisor).

Input: Contents of registers 4, 9, 13, and 14 (see "Register Use" below).

Output: The calculated disk address in field WARNDSKA.

Exit: To the calling routine.

Register Use

0-3 Work registers.
4 Pointer to the calculated disk address; work register.
5 Work register.
9 Pointer to the HCFCB being used.
10 Pointer to CRTTAB.
13 Pointer to the caller's register-save area.
14 Return register.

Subroutine PTHOPEN – Build the Required HCFCB

Entry Points

- PTHCHDR to set up an HCFCB for writing the HCF header record.
- PTHCHDRG to set up an HCFCB for writing the HCF header record built in a GETVIS area.
- PTHCOPNC to set up an HCFCB for HCF initialization.
- PTHCOPNF to set up an HCFCB for a force write into the HCF.
- PTHCOPNM to set up an HCFCB for a search for a cycle byte with changed setting.

- PTHCOPNR to set up an HCFCB for HCF read.
- PTHCOPNW to set up an HCFCB for a write into the HCF.

Function: Builds an HCFCB and the associated channel program; opens the HCF itself if an HCFCB for write access is to be built.

Called By: One of the following routines or subroutines

PTHCCONT
PTHCCREA
PTHCREAD
PTHCWRT
HCFFRC
HCFHDR

Other Routines Called: None.

Subroutines Used: HCFCBMVE: To move data to an area in GETVIS storage.

Data Areas Used: The areas indicated under "Register Use" below.

Input: Contents of registers 9, 10, and 14 (See "Register Use" below).

Output: Either of the following:

- An HCFCB initialized according to the requestor's specifications and a return code of zero in register 15.
- In Register 15, one of the following return codes:
  1C = The HCF is not accessible.
  20 = The HCF extent is too small.

Exit Normal: Return to the caller.

Exit Error: Return via the error handler, to the caller; return code in register 15.

Register Use

0-5	Work registers.
7	Target address for a data move operation.
8	Source address for a data move operation.
0	Pointer (and base register for access) to the temporary HCFCB.
10	Base register for access to CRTTAB.
11	Pointer to the (temporary) HCFCB if in the GETVIS area.
12	Pointer to the requestor's save area.
14	(Sub)routine-call link register.
15	Base register for the routine; return code register on end of processing.

Subroutine HCFCBMVE – Move Data to GETVIS Area: Moves data frames from the HCF-support program area in the SVA to storage obtained via GETVIS, relocating address constants and pointers as required.

Input: Contents of registers 7, 8, and 12 (see "Register Use" below).

Output: Data moved to the system GETVIS area as requested with pointers and constants relocated as required.

Register Use

0	Work register.
2-6	Work registers.
7	Target address for a data move operation.
8	Source address for a data move operation.
12	Contains read/write option and length of I/O buffer.
15	Base register for the subroutine.

Subroutine HCFUPD – Update disk addresses: Updates disk I/O addresses for access to the HCF; inverts the setting of the cycle byte in the logical HCF record if the end of the HCF has been reached.

Input: Contents of registers 1, 10, 13, and 14 (see "Register Use" below).

Output: Updated disk addresses in the HCFCB.

Register Use

1	Pointer (and base register for access) to the applicable HCFCB.
4-5	Work registers.
10	Pointer (and base register for access) to CRTTAB.
13	Pointer to the caller's register save area.
14	Return register.
15	Base register for the subroutine.

Subroutine REQIO – Request Physical I/O: Schedules physical I/O for HCF read or write operations via:

- SVC 0 if requestor is no system task.
- SVC 15 if requestor is a system task (HCF).

Input: Contents of registers 1, 10, and 14 (See "Register Use" below).

Output: One of the below listed return codes in register 15:

```
00 = Requested I/O operation has been completed successfully.
08 = No record found or incorrect length.
0C = Irrecoverable I/O error.
10 = The HCF device is not ready.
```

Register Use:

0	Work register.
1	Pointer to POINTHCF macro generated parameter list.
2-3	Work registers.
8	Work register.
10	Base register for access to CRTTAB.
13	Pointer to the requestor's save area.
14	Return register.
15	Base register for the phase; return-code register on end of processing.

Organization Information

Not applicable to a one-phase/one-module programming support.

Data Area Information

For an understanding of the program logic of the HCF support, an overview of the structure of the hard-copy file can be helpful. Figure Figure 9 shows the file's layout schematically.

Following this file-structure overview, this section provides a summary of the key data areas used by the HCF access support; these summaries are given in alphabetic order of area names. Each summary includes a listing of the area's fields; it provides the displacements in decimal notation; if meaningful, it provides the displacement of fields also in hexadecimal (in the second displacement column) and includes the applicable source-code labels.

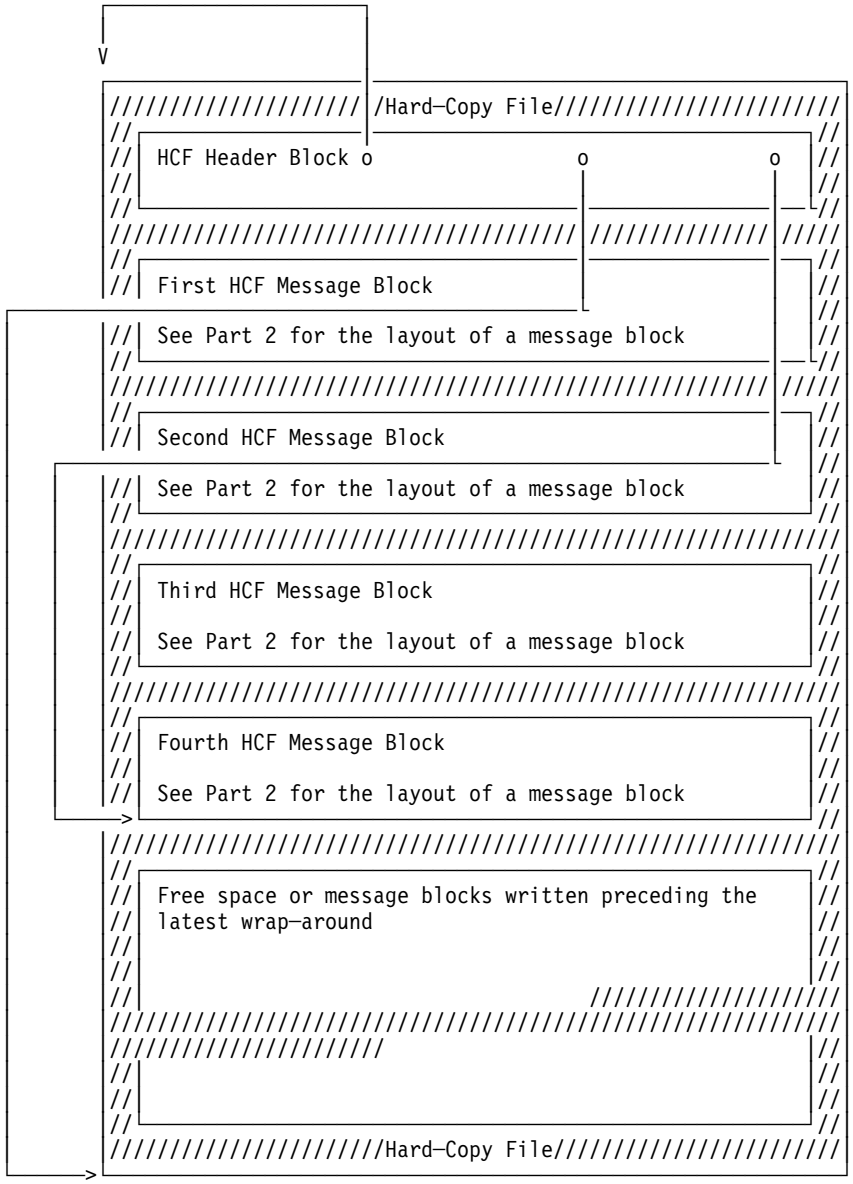


Figure 9 (Part 1 of 2). Schematic Layout of the Hard-Copy File

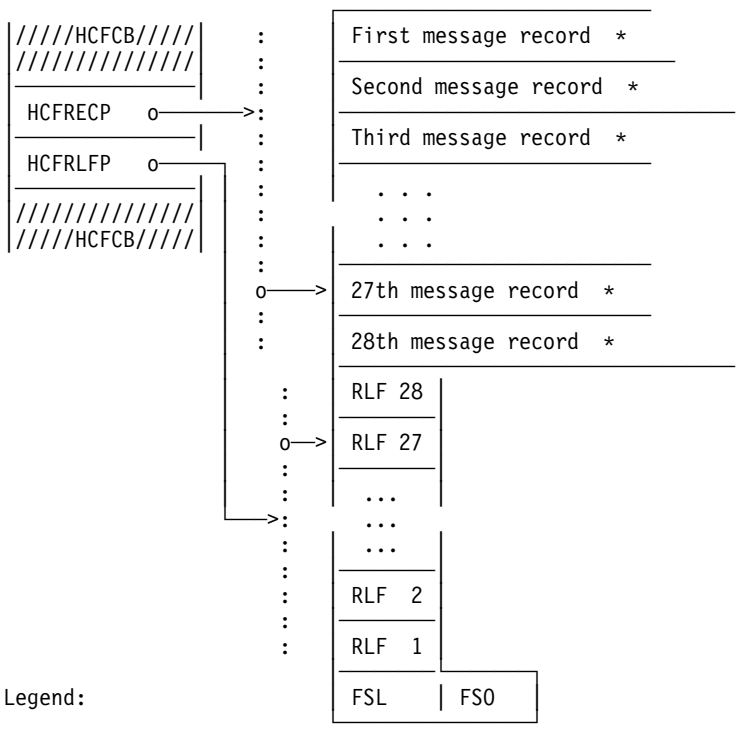


Figure 9 (Part 2 of 2). Schematic Layout of the Hard-Copy File

HCF Header Block: The block is built by subroutine PTHCCREA and accessed by other routines of the HCF support via DSECT HCHEADER. The layout of the block is:

Displ.

0 HDRFLAG
Flagbyte:

```
X'00' = Not set
X'01' = Wrap-around address is valid.
X'02' = Wrap-around address is invalid.
```

1 Reserved.

2-3 HDRLNREQ
Length of last requestor's screen record.

4-7 HDRBGEXT
Pointer to the beginning of the extent as follows:

```
For a CKD disk: CCHH.
For an FBA disk:  The block number relative to the beginning
                  of the disk volume.
```

8-11 HDRNDEXT
Pointer to the end of the extent as follows:

```
For a CKD disk: CCHH.
For an FBA disk: block number relative to the beginning
                 of the extent (of the HCF).
```

12-19 HDRWRAPA

Pointer to the wrap-around address as follows:

```
For a CKD disk  = CCHHRrll
For an FBA disk = rrnnnnll

where:    ll = Logical record number
        nnnn = Block number relative to the beginning
               of the extent
           r = Reserved
```

HCFCB – The HCF Control Block: The control block is built by routine PTHCREAD for read access and by routine PTHCWRT for write access; it is accessed via CSECT HCFCB by the other routines of the HCF support whenever an HCF access request is to be serviced. The layout of the area is:

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE			HC FILE CONTROL BLOCK
0	(0) BITSTRING	1	HCFOPCOD	OPERATION (REQUEST) CODE
	1...		WRITE	"X'80'" WRITE HCFCB
	.1..		WRTHDR	"X'40'" WRITE HEADER HCFCB
	..1.		WRTFRC	"X'20'" WRITE FORCE HCFCB
	.... 1...		READ	"X'08'" READ HCFCB FOR ALL READ ACCESSES
	.... .1..		RDALL	"X'04'" READ HCFCB FOR PRINTLOG WITHOUT SELECTION
	....		HC#GTVIS	"HCFOPCOD" LENGTH OF GETVISED AREA
1	(1) BITSTRING	1	CSTATE0	PRINTLOG OPTION BYTE
	.... 1...		OPNEW	"X'08'" OPTION=NEW
	11		OPNEWP	"X'03'"
	.... .1..		OPSELECT	"X'04'" OPTION=SELECT
	.... ..1.		OPSELCTP	"X'02'"
2	(2) BITSTRING	1	CSTATE1	STATUS INFORMATION BYTE 1
	1...		REQSKIP	"X'80'" SKIP ONE OR MORE BLOCKS
	.1..		BACKWARD	"X'40'" READ DIRECTION IS BACKWARD
	.... .1..		UPDNEW	"X'04'" UPDATE HEADER NEW
	.... ..1.		UPDVAL	"X'02'" VALIDATE HEADER FLAG
	1		UPDINV	"X'01'" INVALIDATE HEADER FLAG
3	(3) BITSTRING	1	CSTATE2	STATUS INFORMATION BYTE 2
	1...		PNTHCF	"X'80'" POINTHCF BEING PROCESSED
	..1.		EOFHCF	"X'20'" EOF REACHED
	...1		BOFHCF	"X'10'" BOF REACHED
	.... 1...		BNDCROS	"X'08'" PHYSICAL BOUNDARY CROSSED
	.... .1..		IOERROR	"X'04'" IOERROR ON HCF
	.... ..1.		NOWRAP	"X'02'" NO WRAP_AROUND_PROCESSING (FORMATTING)
	1		GETFIRST	"X'01'" GET FIRST RECORD OF BLOCK
4	(4) BITSTRING	1	CSTATE3	GATING-STATUS INFORMATION
	1111 1111		GATED	"X'FF'" HCFCB IS IN USE
	....		UNGATED	"X'00'" HCFCB IS NOT IN USE
5	(5) BITSTRING	1	CSTATE4	CYCLE BYTE
6	(6) SIGNED	2	(0)	HW BOUNDARY
6	(6) SIGNED	2	CSTATE5	INDICATOR THAT THE HCFCB IS UPDATED FOR THE CORRESPONDING REQUESTOR

```
  6    (6) BITSTRING        ACTIVE         "X'FFFF'" HCFCB IS READY FOR
                                           PROCESSING
           .... ....        INACTIVE       "X'0000'" HCFCB IS NOT READY FOR
                                           PROCESSING
  8    (8) ADDRESS       4  HCRETCOD       RETURN CODE AREA FOR THE REQUESTOR
 12    (C) ADDRESS       4  HCSAVPTR       POINTER TO FIRST LEVEL S A
 16   (10) ADDRESS       4  HC@GTVIS       ADDRESS OF GETVISED AREA
 20   (14) ADDRESS       4  HCFIOBUF       POINTER TO I/O BUFFER AND END ADDRESS
                                           OF HCFCB
 20   (14) SIGNED           HCFLIOBF       "4096" LENGTH OF I/O BUFFER
 20   (14) SIGNED           HCFLHDR        "512" LENGTH OF HEADER HCFCB
 24   (18) ADDRESS       4  HCFACIDF       POINTER TO CONTROL INTERVALL
                                           DESCRIPTION (CIDF)
           .... ....        HCFSL          "0" OFFSET OF FSL IN CIDF
           .... ..1.        HCFSO          "2" OFFSET OF FSO IN CIDF
 28   (1C) ADDRESS       4  HCFRECP        POINTER TO LOGICAL RECORD IN I/O
                                           BUFFER
 32   (20) ADDRESS       4  HCFRLFP        POINTER TO LENGTH OF CORRESPONDING
                                           LOGICAL RECORD
 36   (24) SIGNED        4  HCB1STSA(17)   FIRST LEVEL SAVE AREA
104   (68) SIGNED        4  HCB2NDSA(6)    SECOND LEVEL SAVE AREA
128   (80) SIGNED        4  HCB3RDSA(6)    THIRD LEVEL SAVE AREA
152   (98) SIGNED        4  HCB4THSA(6)    FOURTH LEVEL SAVE AREA
           1..1 1...        HCFCBREQ       "HCB4THSA" REQUESTOR ADDR
176   (B0) DBL WORD      8  (0)
176   (B0) BITSTRING     2  HCBIOCCB       RESIDUAL COUNT
178   (B2) BITSTRING     2                 COMMUNICATIONS BYTES
180   (B4) BITSTRING     2                 CSW STATUS BYTES
182   (B6) ADDRESS       1                 LOGICAL UNIT CLASS
183   (B7) ADDRESS       1                 LOGICAL UNIT
184   (B8) BITSTRING     1
185   (B9) ADDRESS       3                 CCW ADDRESS
188   (BC) BITSTRING     1                 FLAG BYTE
189   (BD) ADDRESS       3                 CSW CCW ADDRESS
192   (C0) DBL WORD      8  HCBCCWS(0)     CCW CHAIN FOR HCF ACCESS
192   (C0) BITSTRING    32                 MAY BE VARIABLE LENGTH
           ..1. ....        LENCCWS        "*-HCBCCWS"
224   (E0) CHARACTER    16  HCDEFEXT(0)    DEFINE EXTENT DATA
224   (E0) ADDRESS       1
225   (E1) BITSTRING    15
           .1.. ....        DEFREAD        "X'40'"
           111. .1..        HCEXTBEG       "HCDEFEXT+4" PHYSICAL START OF
                                           EXTENT
           111. 1...        HCBEGLOG       "HCDEFEXT+8" START OF LOGICAL EXTENT
           111. 11..        HCEXTEND       "HCDEFEXT+12" END OF LOGICAL EXTENT
240   (F0) CHARACTER    16  ECKDDEFX(0)    DEFINE EXTENT DATA FOR ECKD
240   (F0) BITSTRING     1  ECKDMASK       MASK BYTE
241   (F1) BITSTRING     1  ECKDGLOB       GLOBAL ATTRIBUTE
242   (F2) BITSTRING     2  ECKDBLK        BLOCK SIZE
244   (F4) BITSTRING     2  ECKDCACH       CACHE FAST WRITE ID
246   (F6) BITSTRING     2  ECKDXATT       EXTENTED GLOBAL ATTRIBUTES
248   (F8) BITSTRING     4  ECKDBEGX       BEGIN OF EXTENT
252   (FC) BITSTRING     4  ECKDENDX       END OF EXTENT
256  (100) CHARACTER    16  ECKDLOCT(0)    LOCATE RECORD DATA FOR ECKD
256  (100) BITSTRING     1  ECKDOPER       OPERATION BYTE
257  (101) BITSTRING     1  ECKDAUX        AUXILIARY BYTE
258  (102) BITSTRING     1                 RESERVED
259  (103) BITSTRING     1  ECKDCNT        COUNT
```

```
  260  (104) BITSTRING     4  ECKDSEEK       SEEK ADDRESS
  264  (108) BITSTRING     5  ECKDSRCH       SEARCH ARGUMENT
  269  (10D) BITSTRING     1  ECKDSECT       SECTOR NUMBER
  270  (10E) BITSTRING     2  ECKDLEN        TRANSFER LENGTH FACTOR
  272  (110) BITSTRING    10  STRDSKA(0)     PHYSICAL BEGIN OF ACTIVE HCF EXTENT
  272  (110) BITSTRING    10  CBEGDSKA(0)    BEGIN OF HCF DATA RECORDS
  272  (110) BITSTRING     2
  274  (112) BITSTRING     8  STRDADR(0)
  274  (112) BITSTRING     8  CBEGDADR(0)
  274  (112) BITSTRING     2
  276  (114) BITSTRING     6  STRFBA#
  282  (11A) BITSTRING     2                  RESERVED FOR ALIGNMENT
  284  (11C) BITSTRING    10  ENDDSKA(0)     PHYSICAL END OF ACTIVE HCF EXTENT
  284  (11C) BITSTRING    10  CLSTDSKA(0)    END OF HCF FILE
  284  (11C) BITSTRING     2
  286  (11E) BITSTRING     8  ENDDADR(0)     END OF ACTUAL HCF EXTENT
  286  (11E) BITSTRING     8  CLSTDADR(0)    END OF HCF FILE
  286  (11E)                  CLSTREC#       "CLSTDSKA+6" LAST RECORD NUMBER
  286  (11E) BITSTRING     2
  288  (120) BITSTRING     6  ENDFBA#
  294  (126) BITSTRING     2  HCBCUU          CUU ADDRESS FOR DUMPS
  296  (128) BITSTRING    10  CRTDSKA(0)     CURRENT DISK ADDRESS TO BE USED NEXT
  296  (128) BITSTRING    10  CWITDSKA(0)    DISK ADDR OF LAST WRITTEN REC
  296  (128) BITSTRING     2
  298  (12A) BITSTRING     8  CRTDADR(0)     CURRENT DISK ADDRESS
  298  (12A) BITSTRING     8  CWITDADR(0)    DISK ADDR OF LAST WRITTEN REC
  298  (12A)                  FBADADR        "CRTDSKA+3" FBA DISK ADDRESS FOR JOB
                                             STMT
  298  (12A)                  HCFBABL#       "CRTDSKA+4" RELATIVE BLOCK NR (FBA)
  298  (12A)                  HCFCKDCC       "CRTDSKA+2" ADDR OF CC FIELD )
  298  (12A)                  HCFCKDHH       "CRTDSKA+4" ADDR OF HH FIELD )
                                             -(CKD)
  298  (12A)                  HCFCKDR        "CRTDSKA+6" ADDR OF R FIELD )
  298  (12A)                  HCFLREC#       "CRTDSKA+8" ADDR OF LOGICAL RECORD
                                             FIELD
            ....  ....       CC             "0" OFFSET OF CC VALUE ) ---
            ....  ..1.       HH             "2" OFFSET OF HH VALUE ) RELATED
            ....  .1..       R              "4" OFFSET OF R VALUE ) TO
            ....  ..1.       FBABL          "2" OFFSET OF FBA BLOCK) ----
  298  (12A) BITSTRING     2
  300  (12C) BITSTRING     6  CRTFBA#
            ....  .11.       LDSKADDR       "6" LNG OF DISK ADDR WITHOUT 'BB'
  306  (132) BITSTRING     6                  RESERVED
  312  (138) SIGNED        4  HCFSECB        ECB FOR SYNCHHCF
  316  (13C) CHARACTER     4  HCFSREP        REPLY AREA FOR SYNCHHCF
  320  (140) BITSTRING     8                  RESERVED
================================================================================
--------------------------------------------------------------------------------
 FIELDS NEEDED FOR HCF SUPPORT WITH ROUTER QUEUE
================================================================================

  328  (148) DBL WORD      8  HCFRED(0)      PARAMETERS OF CUR.RED CMD
  328  (148) BITSTRING    64  HCFRDF         ' PARSED PARAMETERS
  392  (188) BITSTRING     1  HCFCREA        INDICATES HOW HCFCB WAS
                                             CREATED 'G' VIA GETVIS
  393  (189) BITSTRING     1  HCFMED         MEDIUM FROM WHICH LAST RED
                                             MESSAGE WAS READ
                                             = 'H' FROM HARD COPY FILE
```

```
                                                = 'R' FROM ROUTER QUEUE
394  (18A) BITSTRING     1  HCFPROC         COMMAND IN PROCESS
395  (18B) BITSTRING     1  HCFLINES        LINES STILL TO FIND
396  (18C) SIGNED        4  HCFRRIPT        POINTER TO LAST REDISPLAY R.
                                            ITEM RECEIVED FROM CONSOLE
400  (190) SIGNED        4  HCFRRIID        MESSAGE IDENTIFER OF LAST R.
                                            ITEM
404  (194) SIGNED        4  HCFMSQN         MIN. SEQUENCE NUMBER
408  (198) SIGNED        4  HCFLSQN         LAST READ SEQUENCE NUMBER
412  (19C) SIGNED        4  HCFRECNR        RECORD NUMBER OF LAST HC REC.
                                            READ BY LAST REDISPLAY CMD
416  (1A0) BITSTRING     1  HCFFLAGS        RECORD NUMBER OF LAST HC REC.
           1...  ....       HCFREDMD       "X'80'" CONSOLE IS IN REDISPLAY MODE
           .1..  ....       HCFDSKST       "X'40'" CURRENT DISK ADDRESS IS SET
           ..1.  ....       HCFMSGAD       "X'20'" ANY MSG ADDED FOR CUR. CMD
417  (1A1) BITSTRING     1  (7)             RESERVED
424  (1A8) BITSTRING    11  HCFSAVTI        TIME OF LAST REDISPLAYED ITEM
435  (1B3) BITSTRING    10  HCFSAVDA        DATE OF LAST REDISPLAYED ITEM
445  (1BD) BITSTRING   117  HCFREDSV        SAVE AREA FOR PREVIOUS DATA
562  (232) BITSTRING     1  (6)             SAVE AREA FOR DISK ADDRESS
568  (238) SIGNED        4  CANKEEP         REC. TO POS. TO ON CANCEL
572  (23C) SIGNED        4  HCFKEEP         REC. TO BE POSITIONED TO
576  (240) DBL WORD      8  (0)             FOR ALIGNMENT
576  (240) CHARACTER     8  HCFCNT(0)       COUNT FIELD WRITE CKD
576  (240) CHARACTER     5  HCFSKARG(0)     SEEK ARG. IN COUNT FILED
576  (240) SIGNED        2  HCFRDTIB        LOGICAL POSITION FOR ORMSG
578  (242) SIGNED        2                  RESERVED
580  (244) SIGNED        4                  RESERVED
580  (244)                  HCFCBEND        "*"
580  (244)                  LNHCFCB         "*-HCFOPCOD"
```

HCFREC – HCF Message-Line Record: Following is the layout of the DSCECT used to build the record in the HCF message block output area:

```
 Offsets       Type    Length   Name          Description
     0    (0) STRUCTURE
     0    (0) BITSTRING     1  HCMSGLVL      0 MESSAGE LEVEL OUT OF MDB
              1...  ....       HCMLR         "X'80'" ITEM WAS WTOR
              .1..  ....       HCMLIA        "X'40'" ITEM WAS IMMEDIATE ACTION
                                             MSG
              ..1.  ....       HCMLCE        "X'20'" ITEM WAS CRITICAL EVENTUAL
                                             ACTION M
              ...1  ....       HCMLE         "X'10'" ITEM WAS EVENTUAL ACTION MSG
              ....  1...       HCMLI         "X'08'" ITEM WAS INFORMATIONAL MSG
              ....  .1..       HCMLBC        "X'04'" ITEM WAS BROADCAST MESSAGE
              ....  ..1.       HCMLRSG       "X'02'" RESERVED
              ....  ...1       HCMLRSH       "X'01'" RESERVED
     1    (1) BITSTRING     1  HCTYPE        1 HARD COPY RECORD FLAGS
              1...  ....       HCTYPCMD      "X'80'" COMMAND OR COMMAND RESPONSE
              .1..  ....       HCTYPSUP      "X'40'" ITEM WAS SUPPRESSED
              ..1.  ....       HCTYPROU      "X'20'" ITEM WAS ROUTED TO/FROM
                                             NETVIEW
              ...1  ....       HCTYPSPE      "X'10'" LOGGING STARTED MESSAGE
              ....  1...       HCTYPCNT      "X'08'" CONTINUATION LINE
              ....  .1..       HCTYPIPL      "X'04'" ITEM WAS GENERATED DURING
                                             IPL
              ....  ..1.       HCTYPJOB      "X'02'" ITEM IS JOBCARD
              ....  ...1       HCTYPCYC      "X'01'" CYCLE BIT
     2    (2) BITSTRING     2  HCLOGID       2 PARTITION SYSLOG ID
     4    (4) BITSTRING     5  HCTIME        4 PACKED TIME STAMP HHMMSSTH
     9    (9) BITSTRING     4  HCDATE        9 PACKED DATE STAMP YYYYDDD
    13    (D) BITSTRING     1  HCNRLNS       13 NUMBER OF LINES OF CURRENT MSG
    14    (E) BITSTRING     2  HCROUT        14 ROUTING CODE
    16   (10) BITSTRING     8  HCCRCON       16 CONSOLE NAME ON WHICH CMD/REPLY
                                             WAS
                                             ENTERED OR WHERE MSG WAS ROUTED TO
    24   (18) SIGNED        4                23 RESERVED
    28   (1C) SIGNED        4  HCBRECNR      27 NUMBER OF THIS RECORD, USED TO
                                             SKIP
                                             READING RECORDS FOR FAST REDISPLAY
    32   (20) SIGNED        4                31 RESERVED
    36   (24) BITSTRING    79  HCMSG         35 CONSOLE MSG AREA
              ..1.  11..       HCMSGBEG      "HCMSG+8" 43 BEGIN OF ACTUAL MESSAGE
              ..1.  ..11       HCMINLN       "HCMSG-HCMSGLVL-1" MINIMUM LENGTH -
                                             1 OF RECORD
              .1..  1111       HCMSGLNG      "*-HCMSG" HARD COPY FILE MESSAGE
                                             LENGTH
              .111  ..11       HCRECLN       "*-HCMSGLVL" HARD COPY FILE RECORD
                                             LENGTH
```

$IJBSLA – Symbolic Label-Access Support

The symbolic label-access support, provided by phase $IJBSLA, functions as an extension of the job control program. The phase receives control when label information is to be added to or deleted from the system's label information area or when information contained in this area is to be retrieved or modified. System routines that use this support (referred to as SLA support in this chapter) request its services via the macros LABEL and LPL; for a description of these macros, refer to *Diagnosis Reference: IPL and Job Control*.

A system routine that requests an SLA support service must operate with storage protection key 0.

$IJBSLA can be called in 31-bit and 24-bit addressing mode. $IJBSLA saves the caller's addressing mode, and restores it again before it returns to the caller.

External Input: The external input for the SLA support consists of:

- A function request passed to phase $IJBSLA by the expansion code of macro LABEL
- A label parameter-list passed to phase $IJBSLA by the expansion code of macro LPL

External Output: Either of the following:

- The label information area updated as requested and a return code of zero in register 15.
- A return code other than zero in register 15 if execution of the requested service failed (Exception: Function request SRCLBL. For details see the function description of SRCLBL). These return codes are given in proper context in the subsequent "Design Information" section.

None of the SLA support's routines issue an error message.

Design Information

The SLA support consists of a main-line routine, IJBSLA, and a number of function-request processing routines. These processing routines receive control from the main-line routine in accordance with the requestor's function specification; they return control to the main-line routine when the requested processing is finished.

If the label area was defined on the data space of a virtual disk (VDISK .....,USAGE=DLA), then IJBSLA transfers control immediately to IJBSLAD each time it is called. IJBSLAD is also part of $IJBSLA and performs the identical functions as routine IJBSLA but by moving the label information to and from a data space.

Figure 10 on page 144 shows a control-flow overview within phase $IJBSLA for the CKD/FBA part (IJBSLA). Following this control-flow overview is a description of the routines in alphabetical sequence of their names.

Being familiar with the structure of the system label-information area may help you to understand the overall design of the SLA support. Figure 11 on page 164 in the subsequent "Data Area Information" section shows you an overview of this structure; it shows that the routines handle label information in chains of subareas called label-area (LA) segments. There is one such chain per label group (such as system standard or partition-permanent for a partition).

The routines of this support add a currently free LA segment to a chain whenever space in the currently last LA segment in the chain runs out of space. The support releases an LA segment from a chain if, by deletion for example, the information contained in a segment is no longer required.

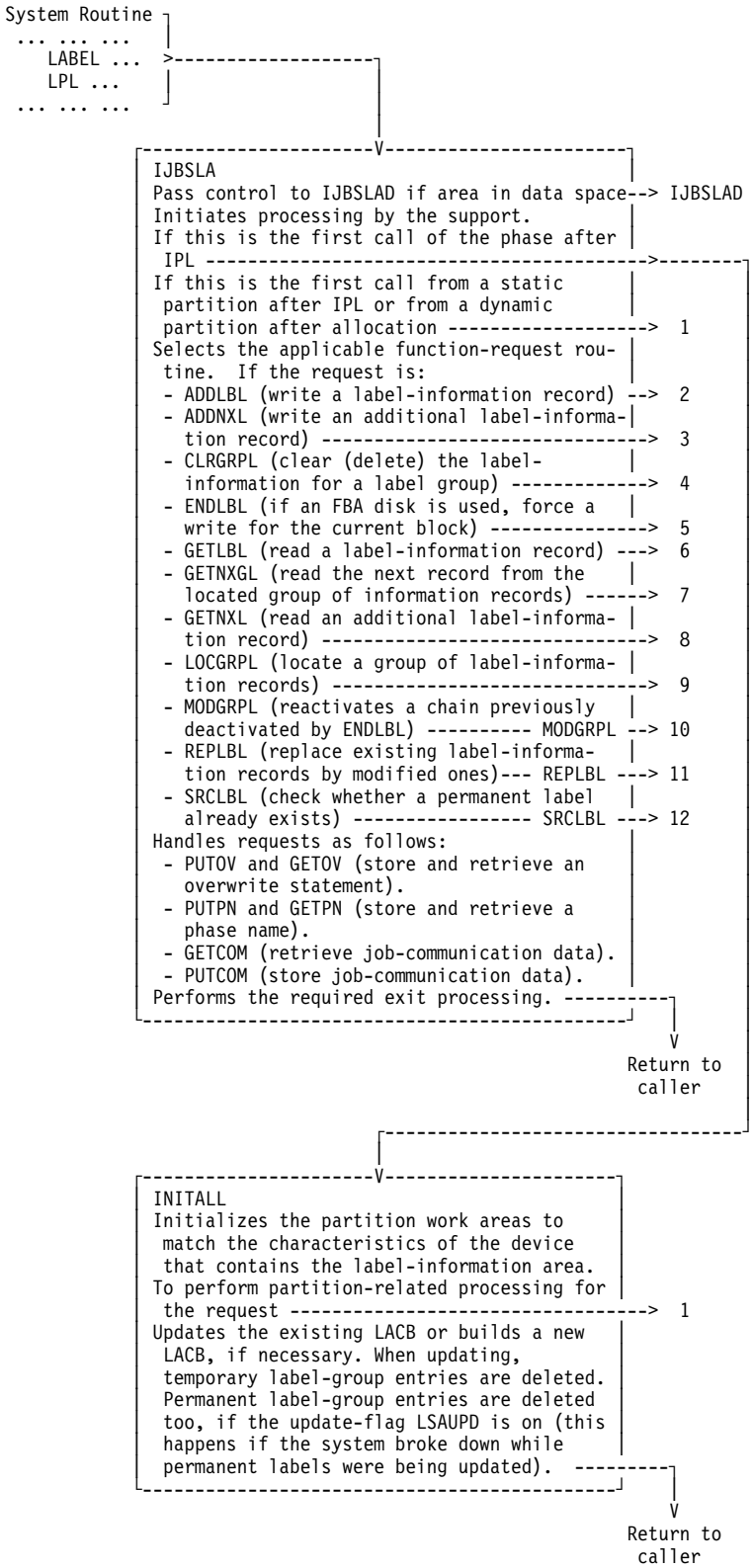


Figure 10 (Part 1 of 5). Symbolic Label Access -- Control-Flow Overview

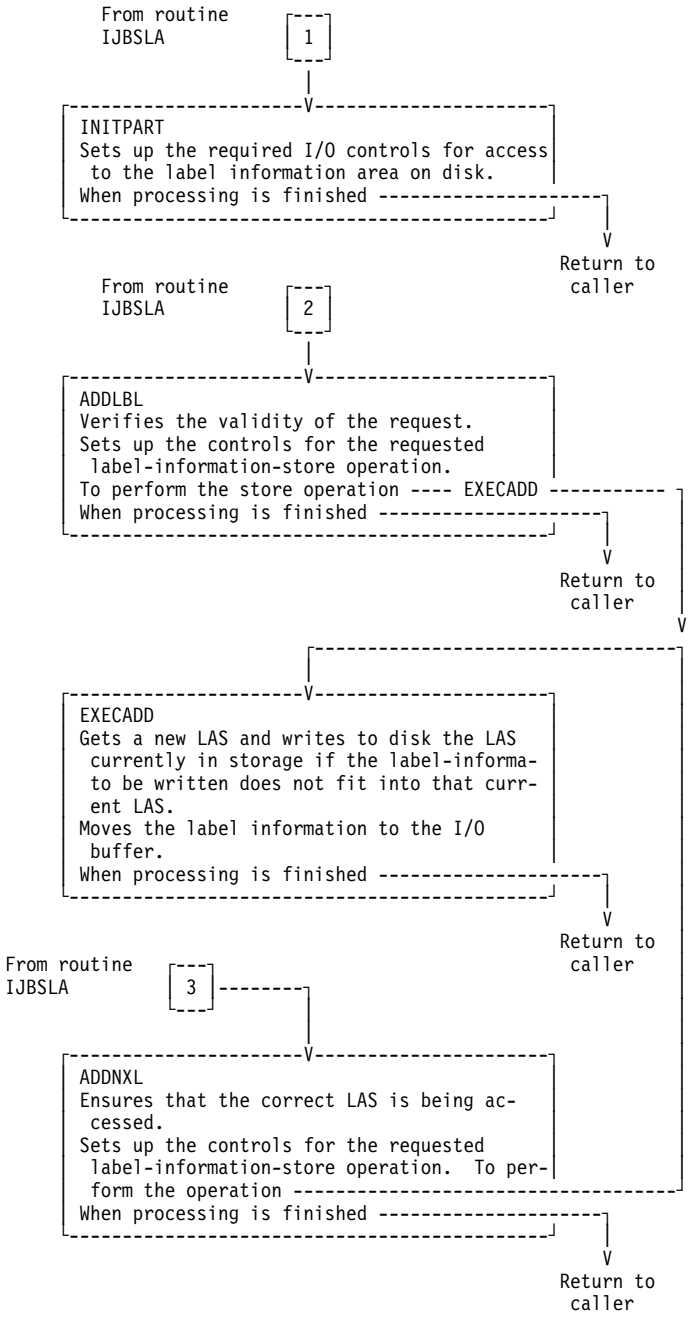


Figure 10 (Part 2 of 5). Symbolic Label Access -- Control-Flow Overview

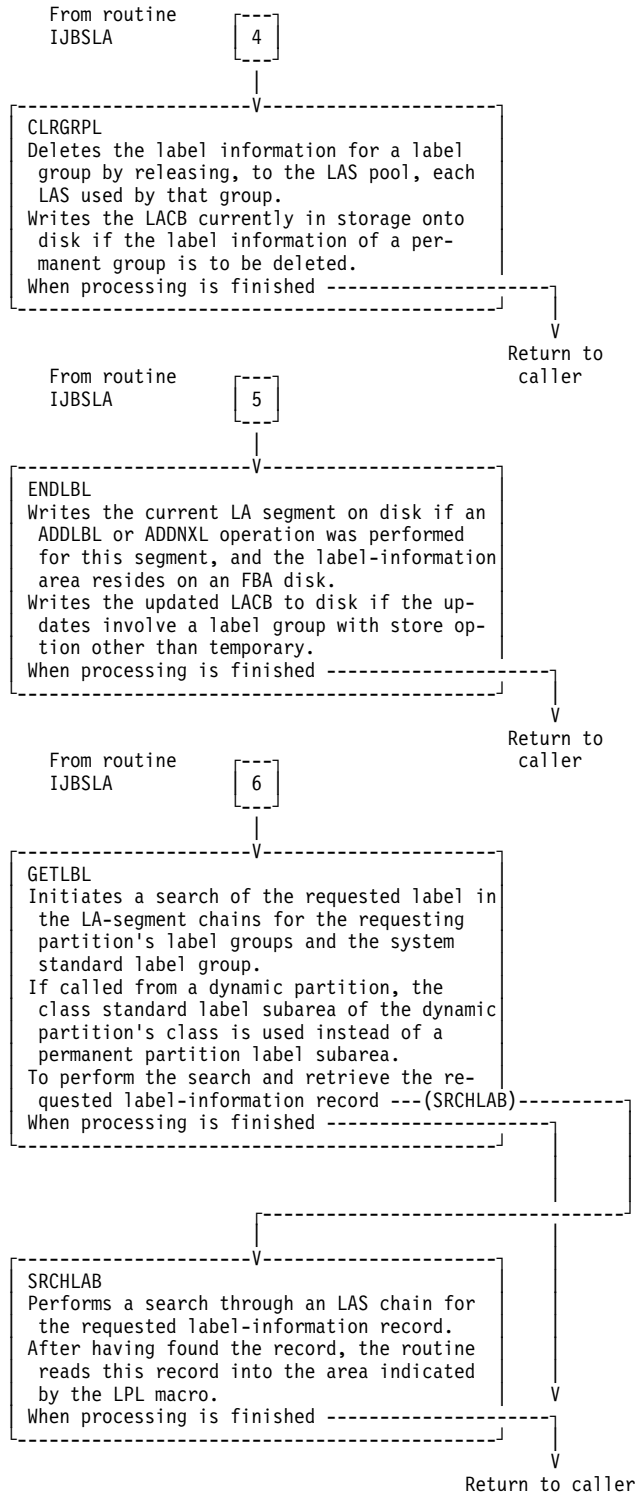


Figure 10 (Part 3 of 5). Symbolic Label Access -- Control-Flow Overview

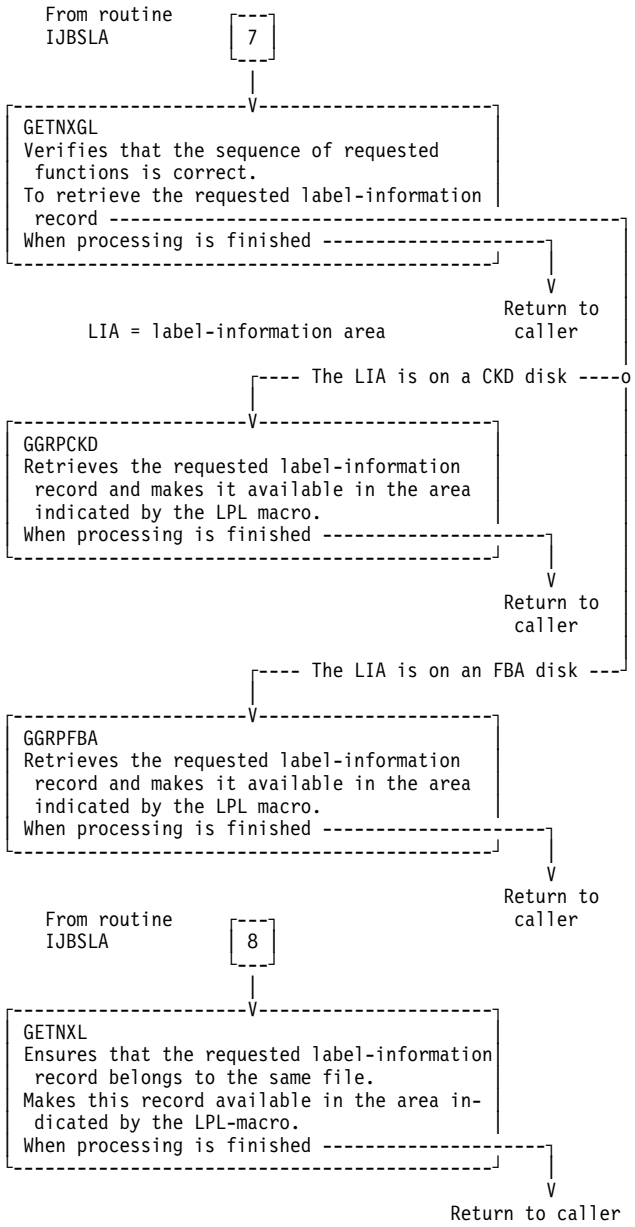


Figure 10 (Part 4 of 5). Symbolic Label Access -- Control-Flow Overview

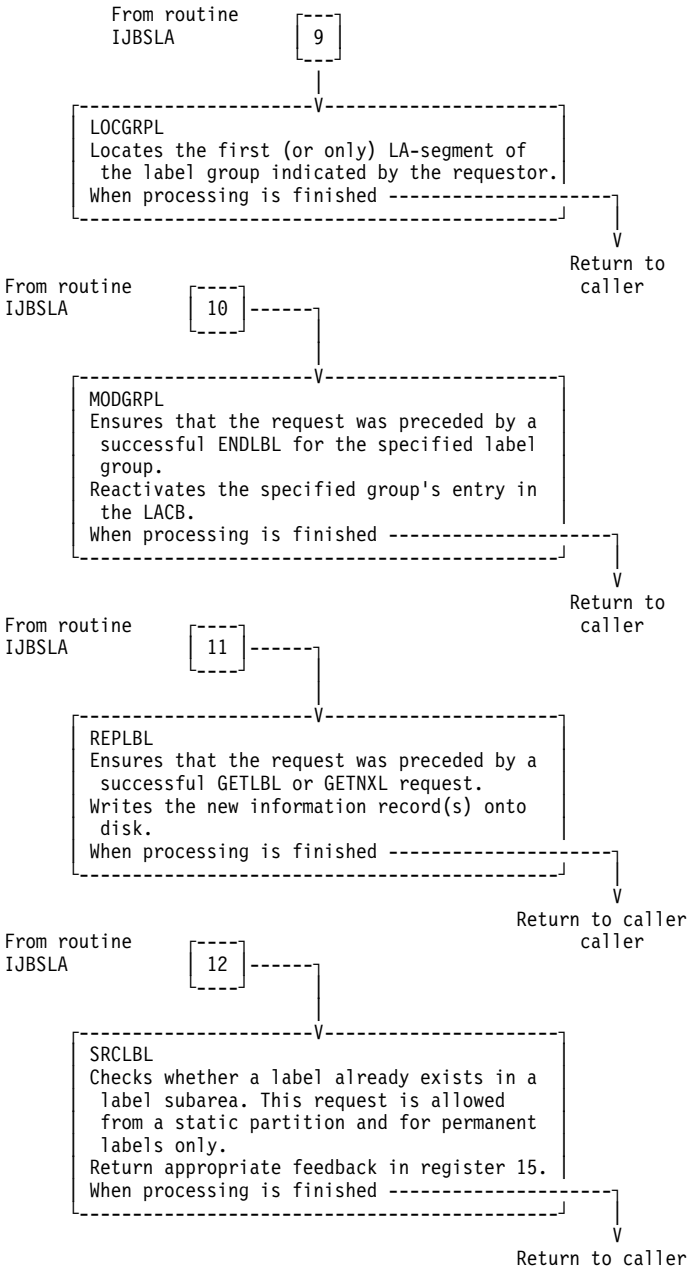


Figure 10 (Part 5 of 5). Symbolic Label Access -- Control-Flow Overview

Routine ADDLBL – Add a Label-Information Record(s)

Function: Moves a label-information record from the area indicated by the LPL macro to the applicable LA-segment.

An ADDLBL request must be preceded by a CLRGRPL or MODGRPL request for the same label group – A CLRGRPL request to clear the affected subarea, a MODGRPL request to reactivate it, whichever applies.

Called By: Routine IJBSLA.

Other Routines Called: Routine EXECADD.

Subroutines Used: FIRSTADD – To have an LA segment allocated.

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.
- The LA-segment buffer in storage.

Input: Contents of the LPL with values in fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLAREA	AREA=address
LPLFLEN	AREALEN=value
LPLLBLEN	LABLEN=value
LPLNAM	FILENAME=name
LPLGRP	GROUP={PARTITION|SYSTEM|partition-id|class-id}
LPLSTORE	STORE={TEMP|PERM|FREE|CLASS}

Output: Either of the following:

- The label-information record passed to the routine (by the AREA=address specification) is written into the applicable LA segment; a return code of 0 in register 15.
- One of the following return codes in register 15:
  X'0C' = The request does not follow a CLRGRPL or another ADDLBL request for the same label group.
  X'14' = The contents of the LPL are invalid.
  X'18' = No space available in the label-information area.
  X'0E' = A system error - A GETVCE macro failed.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation:

ADDLBL Checks the specified length. Ensures that the same label group is being referenced if a CLRGRPL, a MODGRPL, an ADDLBL, or an ADDNXL request was interrupted by a GETLBL, a GETNXL, or a REPLBL request.

Gets an LA segment if this is the first call of ADDLBL after a CLRGRPL request; otherwise, the routine restores the control status of the preceding ADDLBL request for the referenced label group.

Indicates, in bit LSAUPD in flag-byte LSASW of the applicable label-group entry, that label-information records for the particular label group are being written into an LA segment; initiates inclusion of the presented label-information record into the obtained segment by calling routine EXECADD.

Routine ADDNXL – Add the Next Label-Information Record

Function: Inserts an additional label-information record into the label-information area. This record must refer to the same file as the record presented with the preceding ADDLBL or ADDNXL request from the same partition.

Called By: Routine IJBSLA.

Other Routines Called: Routine EXECADD

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.
- The LA-segment buffer in storage.

Input: Contents of the LPL with values in fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLAREA	AREA=address
LPLFLEN	AREALEN=value
LPLNAM	FILENAME=name

Any other operand specification is ignored.

Output: Either of the following:

- The label-information record passed to the routine (by the AREA=address specification), written into the applicable LA segment; a return code of 0 in register 15.
- One of the following return codes in register 15:
  X'0C' = The label-information record does not refer to the same file as the record presented with the preceding ADDLBL or ADDNXL from the same partition.
  X'14' = The contents of the LPL are invalid.
  X'18' = No space available in the label information area.
  X'0E' = A system error - A GETVCE macro failed.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

ADDNXL: Ensures that the current ADDNXL request was preceded by an ADDLBL or another ADDNXL request or by a GETNXGL request.

Initiates inclusion of the presented label-information record into the applicable LA segment by calling routine EXECADD.

Routine CLRGRPL – Clear (Delete) a Label Group

Function: Clears (deletes) all label information from the LA-segment chain of a label group.

If used for temporary labels (LPLSTORE=TEMP) and no temporary labels exist for the calling partition, a free header in the LACB is searched for for further ADDLBL requests.

Called By: Routine IJBSLA.

Subroutines Used

DETENTRY: To determine the affected group entry in the LACB.
WRFBALAS: To write an LA segment into the label-information area on an FBA disk.
WRLACB: To write the LACB in storage into the label-information area on disk.

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.
- The LA-segment buffer in storage.

Input: Contents of the LPL with values in fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLGRP	GROUP={PARTITION|SYSTEM|partition-id|class-id}
LPLSTORE	STORE={TEMP|PERM|CLASS}

Operand specifications related to LPL fields not used for the requested service are ignored.

Output: One of the following return codes in register 15:

X'00' = Request successfully completed (all labels of the specified label group have been deleted).
X'14' = The contents of the LPL are invalid (no labels have been deleted).
X'18' = No free header found in LACB.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

CLRGRPL: Writes the LA segment currently in the buffer into the label-information area on disk if:

- This area resides on an FBA disk, and
- An ADDLBL or ADDNXL request caused a label-information record to be stored into that segment.

The routine calls subroutine WRFBALAS for this purpose.

Calls subroutine DETENTRY to determine the applicable label-group entry.

If called for temporary labels (LPLSTORE=TEMP) and no temporary label exists for the calling partition, a free label group entry (header) in the LACB is searched for following ADDLBL requests.

Updates the LACB in storage by:

- Resetting bit LSAUPD in byte LSASW in the label-group entry to indicate end-of-write.
- Saving the last-segment number in field LSADDLAS of that entry and the next available record position in the entry's field LSADDPOS.
- Clears the applicable label-group entry (header) in the LACB to X'00' thus indicating that no LA segment is occupied by this label-group.

- Indicating the cleared segments as available again in the LA-segment pool and in the group entry for the cleared group.

Calls subroutine WRLACB if the cleared label group was a permanent one. This causes the updated LACB to be written to disk.

Sets to zero those entries in table HISTAB that refer to the label group just cleared.

Routine ENDLBL – End the Current Label Access Function

Function: Indicates to the symbolic label-access support that the current label-access function is complete.

Called By: Routine IJBSLA.

Subroutines Used

WRFBALAS: To write an LA segment into the label-information area on an FBA disk.
WRLACB: To write the currently used LACB into the label-information area on disk.

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.
- The LAS buffer in storage.

Input: Contents of LPL specified by the following LPL-macro operand

```
STORE=DEL|CLDEL
```

if the preceding request was an information-record delete request (can be used only by job control).

Output: A return code of 0 in register 15. If the label-information area resides on an FBA disk, the LA segment used last is written to disk.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

ENDLBL: If the function request is preceded by an ADDLBL or ADDNXL request and the label-information area resides on an FBA disk, the routine calls subroutine WRFBALAS to have the currently used LA segment written into the label-information area.

Writes the LACB onto disk by calling subroutine WRLACB after having done the following:

- Updated fields LSADDLAS and LSADDPOS of the affected label group's LACB entry and the group is a permanent one.
- Having set the corresponding bits (in the LA-segment pool and the affected LACB entry) to indicate that the cleared segments are as available again.

If the ENDLBL function is preceded by a function request other than delete, the routine resets (to zero) all entries made in HISTAB with regard to that preceding function request.

Routine EXECADD – Execute a Label-Add Operation

Function: Moves a label-information record to the LA segment in storage if the label-information area is on an FBA disk; writes a label-information record to disk for a label-information area on a CKD disk.

Called By

Routine ADDLBL
Routine ADDNXL

Subroutines Used

FBAOUT: To move a label-information record to the output area.
CHAINREC: To write an LA-segment-chaining record onto a CKD disk.
SRCHLAS: To obtain a free LA segment.
WRFBALAS: To write an LAS into the label-information area onto an FBA disk.
WRLABREC: To write a label-information record onto a CKD disk.

Data Areas Used: LACB, the label area control block.

Input: Label-information record that is to be moved or written.

Output: Either of the following:

- The label-information record moved to the output area or written onto disk, whichever applies; a return code of 0 in register 15.
- A return code other than 0 contained in register 15 if a subroutine function fails (e.g. X'18' if no free LA-segment is available).

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

EXECADD: If the label-information area is on a CKD disk, checks whether the label-information record fits into the currently used track. If the record does not fit, calls subroutine SRCHLAS and then CHAINREC to make another LA segment, a track on disk, available to the routine. Calls subroutine WRLABREC to have the label-information record written into the label information area.

If the label-information area is on an FBA disk, checks whether the label-information record fits into the LA segment that is being built in storage. If the record does not fit, calls subroutine SRCHLAS and then WRFBALAS to make another segment available and to write the currently used LAS onto disk. Calls subroutine FBAOUT to have the label-information record moved to the current or new segment, whichever applies.

Routine GETLBL – Get a Label-Information Record

Function: Retrieves the requested label-information record from the label-information area.

Called By: Routine IJBSLA.

Other Routines Called: Routine SRCHLAB.

Subroutines Used

HISTORY: To maintain a record of the sequence of function requests.
WRFBALAS: To write an LA segment into the label-information area if this is on an FBA disk.

Data Areas Used

- HISTAB, a record of request sequence per partition.
- LACB, the label area control block.
- LPL, the label parameter list.

Input: Values in LPL fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLAREA	AREA=address
LPLFLEN	AREALEN=value
LPLNAM	FILENAME=name

Output: Either of the following:

- The requested label-information record at the location specified by the requestor in the LPL macro; a return code of 0 in register 15.
- One of the following return codes in register 15:
  X'04' The requested label-information record does not exist.
  X'08' The length of the retrieved information record exceeds the length of the user-supplied buffer; however, as much of the label-information record as fits has been moved into the buffer.
  X'14' The contents of the LPL are invalid.
  X'1C' No GETVIS storage available.
  X'0B' A system error - A CDLOAD macro failed.

For a return code of 0 or 8, the two-byte field LPLLBLEN in the LPL contains the length (in binary) of the label-information record.

If the specified label is found, LPLGRP and LPLSTORE are updated according to the label subarea where the label is found.

For a return code other than 0 and 8, the contents of the buffer remain unchanged.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

GETLBL: If the label-information area is on an FBA disk, calls subroutine WRFBALAS to write the LA segment currently in storage into the label-information area (ASSGNs may occur between sets of DLBL and EXTENT statements and interrupt the processing of label information).

Searches for the requested record in the various label groups in the sequence given below:

1. The partition's free-usage label group.
2. The partition's temporary label group.
3. The partition's permanent label group (if called from a static partition) or the class standard label group (if called from a dynamic partition).
4. The system label group (unless a write operation for the group is pending).

To accomplish the search, the routine looks up the group's entry in the LACB. If field STARTLAS of that block is set to zero, that group is empty and the routine continues to look up the next group's entry; if the field is not set to zero, the routine calls subroutine SRCHLAB to search for the label-information record in the corresponding LA segment(s) and to make this record available in the buffer provided by the requestor.

Gets control if the requested label-information record has been located or, after having searched through all label groups, the record has not been found.

Writes a request-sequence record into HISTAB by calling subroutine HISTORY if the requested record has been found.

Routine GETNXL – Get the Next Label-Information Record

Function: Gets the label-information record for the next extent of a sequential file if this file consists of two or more extents (for a multi-extent sequential file, job control writes a separate information record for each of the extents) or an additional label information record for a a VSAM file.

A GETNXL request must be preceded by a GETLBL or another GETNXL request issued by the same task.

Subroutines Used

GNXTFBA: To handle a GETNXL request if the label-information area resides on an FBA disk.
GNXTHIS: To look up table HISTAB.
RDLABREC: To read a label-information record.
UPDHIS: To update an entry in table HISTAB.

Data Areas Used

- HISTAB, a record of request sequence per partition.
- LACB, the label area control block.
- LPL, the label parameter list.

Input: The same as for a GETLBL request.

Output: Either of the following:

- The next label-information record stored at the location specified by the requestor in the LPL macro; a return code of 0 in register 15.
- One of the following return codes in register 15:
  X'04' = No additional information record exists for the file.
  X'08' = The length of the retrieved information record exceeds the length of the user-supplied buffer; however, as much of the label-information record as fits has been moved into the buffer.
  X'0C' = The request does not follow a GETLBL or another GETNXL request.
  X'14' = The contents of the LPL are invalid.

For a return code of 0 or 8, the two-byte field LPLLBLEN in the LPL contains the length (in binary) of the label-information record.

For a return code other than 0 and 8, the contents of the buffer remain unchanged.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation:

GETNXL: Ensures that the read-in buffer is of sufficient length; ensures that the sequence of requests is correct by having subroutine GNXTHIS look up table HISTAB.

If the label-information area resides on an FBA disk, has subroutine GNXTFBA get the next label-information record; if the area is on a CKD disk, calculates the disk address of the next label-information record and has subroutine RDLABREC retrieve that record.

Records the request in table HISTAB by calling subroutine UPDHIS.

Routine GETNXGL – Get the Next Record After a Locate Request

Function: Reads the next label-information record of a label group, treating that group of records as if it were a sequential file.

A GETNXGL request must be preceded by a LOCGRPL or another GETNXGL request.

Called By: Routine IJBSLA.

Subroutines Used

GGRPFBA: To execute a GETNXGL if the label-information area resides on an FBA disk.
GGRPCKD: To execute a GETNXGL if the label-information area resides on a CKD disk.
WRFBALAS: To write an LA segment into the label-information area if this resides on an FBA disk.

Data Areas Used

- HISTAB, a record of request sequence per partition.
- LPL, the label parameter list.

Input: Values in LPL fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLAREA	AREA=address
LPLFLEN	AREALEN=value

All other LPL fields are the same as they were set by the LPL specification for the preceding LOCGRPL request.

Output: Either of the following:

- The requested label-information record at the location specified by the requestor in the LPL macro; a return code of 0 in register 15.
- One of the following return codes in register 15:
  X'04' No further label-information records of the particular label group are stored.
  X'08' The length of the available buffer is smaller than the length of the label-information record.
  X'0C' The GETNXGL request is not preceded by a LOCGRPL request or another GETNXGL request.
  X'14' The contents of the LPL are invalid.
  X'20' Update for the label group is in progress; it cannot be accessed.

  For a return code other than 0, no label is stored and no further label can be retrieved from the specified label group.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

GETNXGL: Ensures that the request is preceded by a LOCGRPL or another GETNXGL request.

If the label-information area resides on an FBA disk and an ADDLBL or ADDNXL request preceded the LOCGRPL request, the routine has subroutine WRFBALAS write the LA segment currently in storage into the label-information area.

To retrieve the next information record of the involved label group, the routine uses subroutine GGRPFBA if the label-information area resides on an FBA disk; the routine uses subroutine GGRPCKD if that area resides on a CKD disk.

Routine IJBSLA – Initialization and Exit Processing

Function: Together with the routines INITALL and INITPART, this routine initializes the requested label-information access operation. Based on the nature of the request, the routine calls another service routine.

Stores or retrieves job-communication values and overwrite statements.

Called By: System routine requesting the services of the SLA support.

Other Routines Called: See Figure 11 on page 164.

Subroutines Used: None.

Data Areas Used: LPL, the label parameter list.

Input: Contents of the LPL.

Output

- For a label-information access request – None.
- For a store-data request – The applicable data stored or retrieved, whichever applies.
- One of the following return codes in register 15:
  X'0D' A system error - An ASSIGN macro failed.

Exit: To calling system routine.

Register Use: Compiler-dependent.

Routine LOCGRPL – Locate the Specified Label Group

Function: Determines the label group whose contents are to be retrieved by subsequent GETNXGL requests, one information record after the other.

Called By: Routine IJBSLA.

Other Routines Called: None.

Subroutines Used: None.

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.

Input: Contents of the LPL with values in fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLGRP	GROUP={partition–number|SYSTEM|class-id}
LPLSTORE	STORE={TEMP|PERM|DEL|CLASS|CLDEL}

Note: If LOCGRPL is used for partition labels, LPLGRP must contain the partition number, even it 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/4, ie. x'0D' for the first allocated dynamic partition, x'0E' for the second one and so forth.

Output: Either of the following:

- A place holder for the label-group entry corresponding to the LA segments whose information records are to be retrieved; a return code of zero in register 15.
- One of the following return codes in register 15:
  X'04' The specified label group contains no information records.
  X'14' The contents of the LPL are invalid.
  X'20' Update for the label group is in progress; it cannot be accessed.

A return code other than zero implies that a subsequent GETNXGL request fails.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

LOCGRPL: Find the label group's entry (header) in the LACB.

Save the number of the first LA segment for following GETNXGL requests.

If LPLSTORE=DEL|CLDEL prepare for following ADDLBL functions.

Routine MODGRPL – Reactivate a Label Group's Entry

Function: Reactivates a label-group entry in the LACB for subsequent access of the label-information in the associated LA segments.

Called By: Routine IJBSLA.

Other Routines Called: None.

Subroutines Used: DETENTRY – To determine the affected group entry in the LACB.

Data Areas Used

- LACB, the label area control block.
- LPL, the label parameter list.

Input: Contents of the LPL with values in fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLGRP	GROUP={PARTITION|SYSTEM|partition-id|class-id}
*LPLSTORE	STORE=PERM|CLASS

* Not needed for GROUP=SYSTEM.

Output: Either of the following:

- Reactivation of the affected label group; a return code of zero in register 15.
- One of the following return codes in register 15:
  X'04' The specified label group contains no information records.
  X'0C' The MODGRPL request is not preceded by an ENDLBL request for the
  X'14' The contents of the LPL are invalid.

Register Use: Compiler-dependent.

Sequence of Operation

MODGRPL: Simulates an ENDLBL request for the same label group.

Determines the label-group entry to be reactivated by calling subroutine DETENTRY. Reactivates the affected label group.

Routine REPLBL – Replace an Existing Label-Information Record

Function: Replaces a label-information record previously retrieved by a GETLBL or GETNXL request from the same task.

Called By: Routine IJBSLA.

Other Routines Called: None.

Subroutines Used

FBAREP: To move the replacement information record into the output area.
READLAS: To read an LA segment into the input area.
WRFBALAS: To write an LA segment into the label-information area if this resides on an FBA disk.

Data Areas Used

- HISTAB, a record of request sequence per partition.
- LPL, the label parameter list.

Input: Values in LPL fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLAREA	AREA=address
LPLFLEN	AREALEN=value
LPLNAM	FILENAME=name

Output: Either of the following:

- The replacement information record moved to the LA-segment build (output) area or written to disk, whichever applies; a return code of zero in register 15.

- One of the following return codes in register 15:
  X'0C' The replacement record made available either is inconsistent with the original information record, or an attempt is made to change fields whose contents must remain unchanged.
  X'14' The contents of the LPL are invalid.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

REPLBL: Picks up the applicable group entry (same TIK and a GETLBL or GETNXL request) in HISTAB.

If the label-information area resides on an FBA disk, the routine reads the applicable LA segment from disk into the output area for a subsequent write and moves the replacement record into the output area; writes the LA segment into the label-information area again. To accomplish this, the routine calls subroutines READLAS, FBAREP, and WRFBALAS in this sequence.

If the label-information area resides on a CKD disk, the routine updates the channel program for an information-record overwrite operation; uses DOIO, the SLA support's EXCP routine to perform the actual I/O operation.

Updates the table HISTAB with the cylinder and track (block number in case of an FBA disk) values used for the last disk write operation.

Routine SRCHLAB – Search for a Label-Information Record

Function: To search for the required label-information record in a label group' LA segments and, if the record is found, to make it available in the buffer provided by the requestor.

Called By: Routines GETLBL, SRCLBL.

Other Routines Called: None.

Subroutines Used

CHECKLAS: To scan the LA segment in the input area for the requested label-information record.
FBAIN: To move a label-information record from the input area to the user-supplied buffer.
RDLABREC: To read a label-information record.
READLAS: To read an LA segment into the input area.

Data Areas Used: See the description of GETLBL and SRCLBL, the calling routines.

Input: See the description of GETLBL and SRCLBL, the calling routines.

Output: See the description of GETLBL and SRCLBL, the calling routines.

Exit: Return to the caller.

Register Use: Compiler-dependent.

Sequence of Operation

SRCHLAB: Stores the number of the LA segment whose contents are to be scanned.

If the label-information area resides on an FBA disk, the routine calls subroutines READLAS and CHECKLAS to perform the search in the given LA segment and, if necessary, also in any chained segments. Uses subroutine FBAIN to make the requested record available to the requestor, if the search was successful.

If the label-information area resides on a CKD disk, the routine prepares the channel program for the search and calls subroutine RDLABREC.

Routine SRCLBL – Search for Same Label in Label Subarea

Function: To search for a label record in a permanent label subarea. This function is supported only for permanent labels (system, partition, class). It must be used either from BG for system, partition or class labels or from a (static) foreground partition for its own labels.

Called By: Routine IJBSLA.

Other Routines Called: Routine SRCHLAB

Subroutines Used

WRFBALAS: To write an LA segment into the label-information area if this resides on an FBA disk.

Data Areas Used

- LPL, the label parameter list.
- LACB, the label area control block.

Input: Values in LPL fields as follows:

Field Name	Applicable Operand of LPL Macro (or Default)
LPLNAM	FILENAME=name
LPLGRP	GROUP={PARTITION|SYSTEM|partition-id|class-id}
LPLSTORE	STORE=PERM|CLASS

Output: One of the following return codes in register 15:

X'00' The specified file name (label name) has not been found.
X'14' The contents of the LPL are invalid.
X'28' The specified file name (label name) does already exist in the specified label subarea.

Exit: Return to the caller.

Register Use: Compiler dependent.

SLA-Support Subroutines

CHAINREC – Write LA-Segment Chaining Record: A subroutine used if the label-information area resides on a CKD disk.

Prepares the channel program for writing, into the label-information area, a chaining record as the last one of a track. Executes the subroutine WRSYSRES to set the write-to-SYSRES switch. Performs the actual I/O by using DOIO, the SLA support's EXCP routine.

CHECKLAS – Scan Input Area for a Label-Information Record: The subroutine scans the LA segment stored in the input area for the requested label-information record. If it finds the record, the subroutine sets on switch LABFND.

DETENTRY – Determine Affected Entry in LACB: The subroutine DETENTRY picks up the values specified in the requesting LPL macro and determines the affected entry in the LACB.

If an entry for temporary labels is to be determined, and no entry exists for temporary labels of the calling partition, the number of the first free entry is returned.

FBAIN – Move Label-Information Record from Input Area: A subroutine used if the label-information area resides on an FBA disk.

Moves the previously located label-information record (or record set) from the input area to the user-supplied buffer. If the record (set) does not fit into this buffer, the subroutine moves only as much of the record (set) as fits. Stores the length of the moved record (set) in the LPL.

If the requested record crosses the boundaries of two chained LA segments, this subroutine uses subroutine DOIO to read the chained LA segment into storage and to complete the move operation for the information record.

FBAOUT – Move Label-Information Record to Output Area: A subroutine used if the label-information area resides on an FBA disk.

Moves the label-information record (or record set) to be written into the output area. If the record (set) does not fit into the currently built LA segment, the subroutine calls subroutine SRCHLAS to get another LA segment and inserts the proper chaining address in the current LA segment. Writes the current LA segment into the label-information area by calling subroutine WRFBALAS.

FBAREP – Move Replacement Information Record to Output Area: A subroutine used if the label-information area resides on an FBA disk.

Moves the replacement information record (or record set) into the correct record slot of the output area; calls subroutine WRFBALAS to write the segment back into the label information area.

If the replacement record (set) does not fit into the LA segment, the subroutine enters a loop for reading the group's next LA segment into storage, inserting the record (set) which did not fit into the preceding one, and shifting the remaining records in the segment. This loop, which includes subroutines READLAS and WRFBALAS, is active until all subsequent information records have been shifted.

FIRSTADD – Allocate an LA (label area) Segment: Uses subroutine DETENTRY to find the affected label group. If that label group is a permanent one, subroutine FIRSTADD causes the currently used LACB to be written into the label-information area by calling subroutine WRLACB.

Calls subroutine SRCHLAS to obtain a free LA segment.

GNXTFBA – Handle a GETNXL Request: A subroutine used if the label information area resides on an FBA disk.

A loop that scans the LA segment currently in storage for the requested information record and calls subroutine READLAS to read in the next LA segment if there is no find. When it finds a record for the same sequential file, the subroutine has this record moved to the requestor's buffer by subroutine FBAIN.

GNXTHIS – Look Up Table HISTAB: Obtains the TIK of the requesting task. Scans the table HISTAB for an entry referring to a GETLBL or GETNXL request from the same task. Verifies the extent sequence number if this is requested.

HISTORY – Maintain a Record of Sequence of Requests: Gets the TIK of the requesting task and scans the table HISTAB for an entry referring to the same file. Uses subroutine UPDHIS to either update an entry if there was a find or write a new entry if there was no find. Extends the table (in GETVIS storage) should there be a need.

RDLABREC – Read a Label Record: Initiates reading a record from a label-information area on a CKD disk; calls the DOIO, the SLA support's EXCP routine, to perform the actual read operation. Uses the loop RDCHREC to check whether the retrieved record is the required one and to have the next record retrieved if not.

On a find of the required record, this subroutine uses subroutine DETLEN to ensure that the record fits into the requestor's buffer and to read that record into this buffer.

READLAS – Read LA Segment into Input Area: Updates the channel program for a subsequent disk-read operation that reads the required LA segment into the input area. Uses DOIO, the SLA support's EXCP routine.

SRCHLAS – Obtain a Free LA Segment: Scans the LA-segment pool of the LACB for the first free LA segment. Makes this segment available to the caller by setting the applicable LA-segment control bits in the segment pool and the group entry off and on, respectively, and by storing the LA-segment chain start value in field STARTLAS of the LACB entry for the label group.

UPDHIS – Update an Entry in Table HISTAB: Stores, in the applicable entry of table HISTAB, access-history data referring to a sequential multi-extent file. The data stored includes the file name, the label group, the TIK, the applicable LA segment, and the length of the information record (or record set).

WRFBALAS – Write an LA Segment onto an FBA Disk: A subroutine used if the label-information area resides on an FBA disk.

Prepares the channel program for writing a filled up LA segment into the label-information area. Executes the subroutine WRSYSRES to set the write-to-SYSRES switch. Performs the actual I/O by using DOIO, the SLA support's EXCP routine.

WRLABREC – Write a Label-Information Record: A subroutine used if the label-information area resides on a CKD disk.

Prepares the channel program for writing a label information record (or record set) into the label-information area. Executes the subroutine WRSYSRES to set the write-to-SYSRES switch. Performs the actual I/O by using DOIO, the SLA support's EXCP routine.

WRLACB – Write an LACB to Disk: Moves the LACB from the key-0 protected area to an I/O area in the requesting partition. Prepares the channel program for writing an LACB into the label-information area. Executes the subroutine WRSYSRES to set the write-to-SYSRES switch. Performs the actual I/O by using DOIO, the SLA support's EXCP routine.

WRSYSRES – Allows Write on SYSRES: Sets the PIB ASSIGN flag bit in the Program Information Block on, to allow write operation on SYSRES.

Program Organization Information

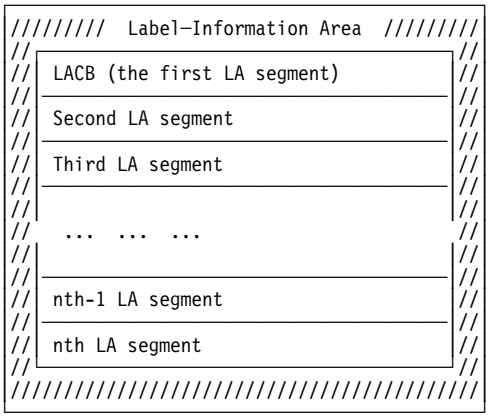
Not applicable to a one-phase/one-module programming support.

Data Area Information

Figure 11 on page 164 shows the structure of the label-information area; being familiar with this structure helps you to understand the program logic of the SLA support.

This structural overview is followed by a description of the key areas in alphabetic sequence of their names. Each of these descriptions shows the displacement of the area's fields in decimal notation. If

reasonable, the descriptions give this displacement also in hexadecimal in a second displacement column and provide the source code labels of the fields.




```
Legend:  LACB = Label-area control block
           LA = Label area - LA segment:
                - A full track if the label-information area is
                  defined on a CKD disk
                - An area of 2K bytes (four CIs) if the label-
                  information area is defined on an FBA disk
                For more details on the layout of an LA segment, re-
                fer to Part 3.
            n = Any number equal to or less than 248
```

Figure 11 (Part 1 of 4). Structure of the Label-Information Area

Note: The references within parentheses refer to additional explanations following the illustration.

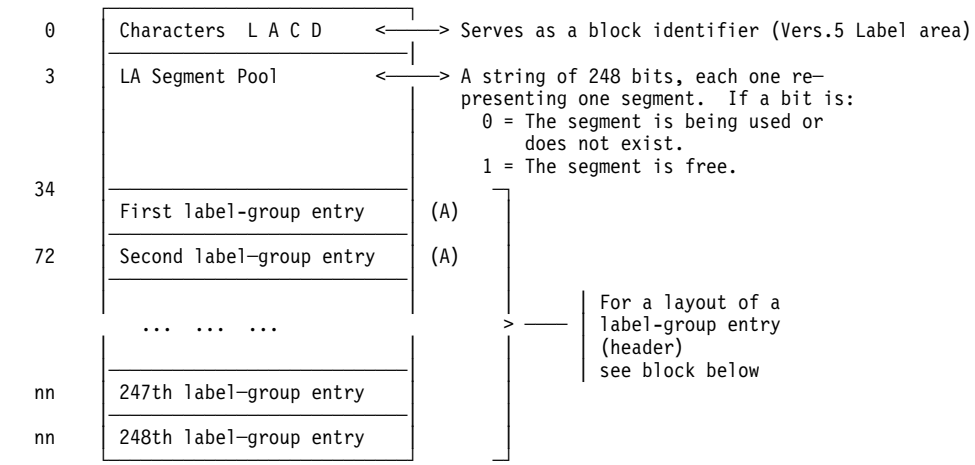


Figure 11 (Part 2 of 4). Structure of the Label-Information Area

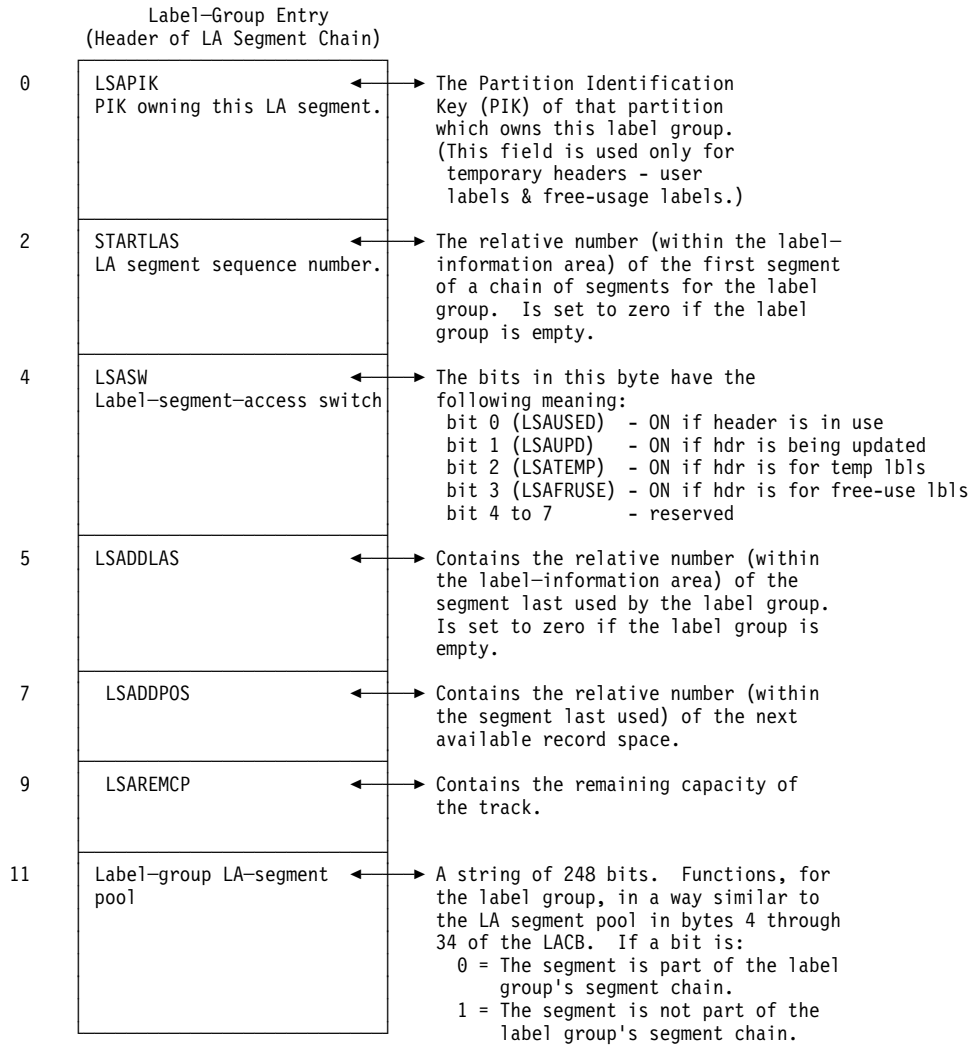
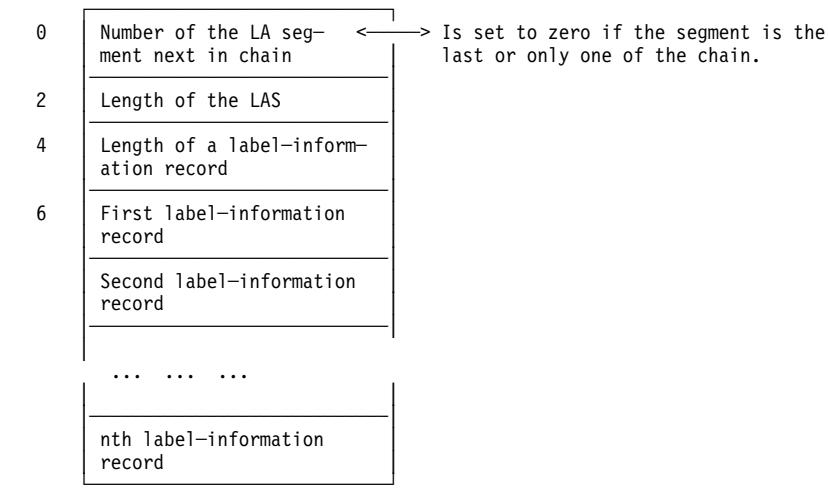
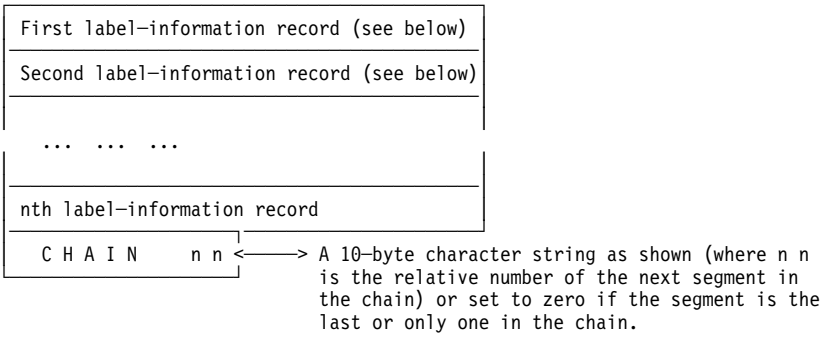


Figure 11 (Part 3 of 4). Structure of the Label-Information Area

Format of an LA segment on an FBA disk:



Format of an LAS on a CKD disk:



Format of a Label-Information Record on a CKD Disk

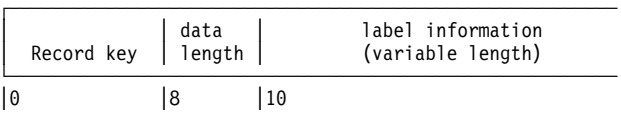


Figure 11 (Part 4 of 4). Structure of the Label-Information Area

Ref. Explanation

(A) When initial program load of the system is complete, the LACB contains entries (headers) for chains of segments as follows:

Entry 1: Entry for the LAS chain for system-standard labels; the chain contains label information submitted with option STDLABEL active.

Entry 2: Entry for the BG partition's permanent-labels LAS chain; the chain contains label information submitted from and for that partition with the option PARSTD active for the partition.

Entry 3: Entry for the F1 partition's permanent-labels LAS chain; the chain contains label information submitted from and for that partition with the option PARSTD active for the partition.

Entry 4 - 13: Entries for the F2, F3, ..., FB partition's permanent labels chain.

Entry 14 - 15: Reserved entries.

Entry 16 - 18: Entry for the class 'C', 'D', 'E' standard labels LAS chains; these chains contain label information submitted with option CLASSTD active.

Entry 19: Reserved entry.

Entry 20 - 39: Entry for the class 'G' - 'Z' standard labels LAS chains.

Entry 40 - 239: Temporary headers (dynamically assigned and freed) for

- user labels LAS chain
- system free-usage labels LAS chain

Entry 240: Entry used for deleting labels.

$IJBSTRT – Partition-Start Phase

This phase is used during system start-up by automated system initialization (ASI); the phase includes module INITOPR, which tests the conditions for partition activation.

Description of Phase $IJBSTRT

Entry Point: IJBSTRT.

Function: This phase starts a (static) partition when it is called, via macro STARTP, from another partition. Macro STARTP, which is described in *VSE/Central Functions Supervisor Diagnosis Reference*, can be issued only during a start-up by ASI and if the partition has not been started previously.

Called By: A system phase that has to start (activate) a partition. The phase uses macro STARTP for this purpose.

Phases Called: INITOPR

Subroutines Used: None.

Data Areas Used: None.

Messages Caused or Issued: None.

Input: The partition number (0 for BG, 1 for F1, and so on) in register 1 and the address of an 18-fullword save area in register 2.

Output: A return code in register 15 as follows:

0 = Successful completion of the partition-start request.
4 = Area not available (partition is already active, for example).
6 = Request not issued during system start-up by ASI.
8 = No GETVIS space could be obtained for the work data of $IJBSTRT.

Exit: Return to the caller.

Register Use

7 Base register.
11 Address of COMREG.
12 Address of SYSCOM.

Sequence of Operation

IJBSTRT: Sets a storage-key restore indicator if the requestor's storage key is other than 0. Executes module INITOPR to ensure that the pre-requisites for starting the affected partition are met. If so, issues a TREADY macro to activate the partition.

Restores the requestor's storage key, if necessary.

Description of Module INITOPR

Entry Point: INITOPR

Function: Ensure that the prerequisites for the requested partition start are met.

The subroutine sets the applicable return code for any of the following conditions:

The ASI bit is off.
The partition to be started is not allocated.
The partition to be started is active.

Called By: Phase $IJBSTRT.

Phases Called: None.

Data Areas Used

- COMREG, the communication region of the partition to be activated (in the supervisor).
- PIB, the partition information block (in the supervisor).

Messages Caused or Issued: None.

Exit: Return to the caller.

Register Use

7	Base register.
11	Address of COMREG.
12	Address of SYSCOM.

Appendix A. Diagnostic Aids

This appendix briefly discusses how you can use this publication as a help in programming service activities such as isolating an error; it gives you hints for locating certain areas or phases in a dump and includes an example for tracing a programmed event within a specific range of storage.

For a message-to-phase (message-to-module) cross-reference, see
Appendix B, “Message-to-Phase (or Module) Cross-Reference” on page 175.

For a command-to-phase cross-reference, see Appendix C, “Command-to-Phase Cross-Reference” on page 179. For a cancel-code-to-message cross-reference, refer to *Diagnosis Reference: Supervisor*.

Using the Publication as a Diagnostic Aid

Assume you have a problem of which you know it is related to software and that this problem occurred when the system invoked one of the services whose supporting code is described in this manual (the attention routines, for example).

Possibly you have not been done problem solving for components of VSE/Central Functions for quite some time, and you want to refresh your knowledge. For instance, you might want to recall in your mind how the phases of the attention routine tie into the overall logic of VSE/Central Functions, which system component passes control to these phases, which system component receives control from those phases, and what the external input to and output from those phases can be. The "Introduction" section of the applicable chapter in this manual provides this kind of information.

As a next step, you might want to relate the function that did not perform properly with one or a limited group of phases (or modules). The logic-flow overview at the beginning of the applicable chapter's "Design Information" section relates functions to phases (or modules). This logic-flow overview has been drawn top-down with off-page connect conventions as follows:

```
Page exit-point:  ----->  n

Page entry-point: | n |----->
```

The logic-flow overview is followed by a description of each of the involved phases (or modules).

If you are fairly familiar with the externals of a function and also know the phase or module that was in control when the error occurred, you might want to do a quick backward tracing of the control flow. The "Organization Information" section of the applicable chapter contains, if meaningful, a table that allows you to do this kind of tracing on a phase or module level.

When inspecting code at a microfiche viewer, you certainly would dislike exchanging microfiche cards (or push them back and forth) in order to look at the layout and use of specific data areas and control blocks. You can use the information provided in the "Data Area Information" section of this manual to avoid such cumbersome exchange of microfiche cards in most instances.

Appendix B provides a message-to-phase (message-to-module) cross-reference;

Locating a Phase in the Logical Transient Area

Proceed as follows:

1. Locate the system communication region (SYSCOM)

   You find the address of this supervisor area in bytes X'80' through X'83' of virtual storage.

2. Look up the address of the LTA

   You find this address at displacement X'1C' from the beginning of the SYSCOM area.

3. Locate the LTA itself

   If an IBM supplied phase was last loaded into this area, the first eight bytes of this area contain the name of that phase.

Locating a Phase in the Shared Virtual Area

1. Locate the system communication region (SYSCOM).

   You find the address of this supervisor area in bytes X'80' through X'83' of virtual storage.

2. Look up the address of the Shared Virtual Area.

   You find this address at displacement X'F4' from the beginning of the SYSCOM area.

3. Locate the System Directory List in the Shared Virtual Area.

   Bytes X'08' through X'11' from the beginning of the Shared Virtual Area point to the start of the System Directory List. The two halfwords at offset X'1C' and X'1E' will show the number and the length of the directory entries, while offset X'18' will point to the first free directory entry, which means the end of the list.

4. Locate the System Directory List entry in the Shared Virtual Area.

   The first directory entry will start at offset X'04' in the System Directory List with the name of the corresponding phase. Now scan the directory entries until you will find the phase name, you are looking for.

5. Look up the entry point of phase in the Shared Virtual Area.

   At displacement X'3C' in your current directory entry you will find the entry point of your phase.

Tracing a Programmed Event in a Specific Area

Figure 12 shows how to use an SDAID procedure for a storage alteration trace.

```
¤ = The operator presses END/ENTER

     // JOB TEST ¤
(1)  // EXEC PROC=SDSTOR,AREA=ALL,ADDRESS='4568:4569',PRINTER=00E ¤

       The SDAID initialization routine will respond with messages 4C05I and 4C44I:

     4C05I PROCESSING OF 'SDAID'  COMMAND SUCCESSFUL
     4C05I PROCESSING OF 'OUTDEV' COMMAND SUCCESSFUL
     4C05I PROCESSING OF 'TRACE'  COMMAND SUCCESSFUL
     4C05I PROCESSING OF 'READY'  COMMAND SUCCESSFUL
     4C44I ENTER 'STRTSD' ATTENTION COMMAND TO ACTIVATE SDAID

(2)  strtsd ¤

     4C05I PROCESSING OF 'STRTSD' COMMAND SUCCESSFUL
       ...  ...  ...
       SDAID traces the specified events; the program prints the event records
       on printer 00E.
       ...  ...  ...

(3)  endsd ¤

     4C05I PROCESSING OF 'ENDSD'  COMMAND SUCCESSFUL
```

Figure 12. Example of an SDAID Session -- Tracing a Storage Alteration Event

(1) procedure statement

The procedure parameters describe the trace areas and the SDAID output device:

- **PRINTER=00E** (or **TAPE=280**) describes the SDAID output device,
- **AREA=ALL** means that all tasks in the system area are to be traced,
- **ADDRESS='4568:4569'** defines a two-byte field which is altered.

(2) strtsd ¤

Indicates to SDAID to activate the trace now.

(3) endsd ¤

Issue an ENDSD command immediately after SDAID has supplied the required trace information. The command causes all system resources held by SDAID during the trace operation to be released.

Appendix B. Message-to-Phase (or Module) Cross-Reference

Message-ID	Phase / Module of Phase in Parentheses
0C00I	$$BCHKPF, $$BCHKP2
0C02I	$$BCHKPT
0C03I	$$BCHKPE, $$BCHKPT
0C04I	$$BCHKPD, $$BCHKPT
0C05I	$$BCHKPD
0C06I	$$BCHKPD
0C07I	$$BCHKPD
0C08I	$$BCHKPD
0C09I	$$BCHKPE
0C10I	$$BCHKPD, $$BCHKPT
0C11I	$$BCHKPD, $$BCHKPT
0C12I	$$BCHKPD, $$BCHKPT
0C13I	$$BCHKP2, $$BCHKP3
0C14I	$$BCHKPD, $$BCHKPT
0C15I	$$BCHKPT
0C16I	$$BCHKPG, $$BCHKP3
0C17I	$$BCHKPD, $$BCHKPE, $$BCHKPT, $$BCHKP2
0C18I	$$BCHK3G
0C19I	$$BCHKPD
0C20I	$$BCHKPD, $$BCHKPT
0P91I	$IJBSEOT
0R09I	$$BRSTRT
0R13I	$$BRSTRT
0R15I	$$BRSTR2
0R20A	$$BRSTR2
0R21D	$$BRSTR2
0R22D	$$BRSTR2
0R23D	$$BRSTR2
0R24D	$$BRSTR2
0R25A	$$BRSTR2
0R26A	$$BRSTR2
0R27D	$$BRSTR2
0R28A	$$BRSTR2
0R29D	$$BRSTR2
0R30D	$$BRMSG2
0S30I	IJBXLBIO ($IJBSDMP)
1A5nD	$$BATTNL
1B12D	$$BATTF1, $$BATTU1
1B13A	$$BATTF3, $$BATTF4
1B14A	$$BATTF2, $$BATTF5
1B15I	$$BATTF4, $$BATTF5, $$BATTU2

Message-ID	Phase / Module of Phase in Parentheses
1B16I	$$BATTF4, $$BATTF5, $$BATTU2
1B17I	$$BATTF5
1B18A	$$BATTU2
1B19I	$$BATTF2, $$BATTF3
1B20A	$$BATTF2, $$BATTF3, $$BATTF4, $$BATTF5, $$BATTU2
1I0nI	$$BATTNA
1I31I	$$BATTNB
1I33I	$$BATTNP
1I34I	$$BATTNP
1I37I	$$BATTNT
1I38I	IJBXDUMP, $$BATTNT, $$BATTNU
1I39t	$$BATTNE, $$BATTNT, $$BATTNU
1I41I	$$BATTNU, IJBXDPAR ($IJBSDMP)
1I42D	$$BATTNT
1I43I	IJBXDUMP
1I44I	IJBXDUMP ($IJBSDMP)
1I45D	$$BATTNT
1I46D	$$BATTNV
1I47D	$$BATTNT
1I48D	$$BATTNU
1I49E	IJBXLBIO ($IJBSDMP)
1I51I	IJBXLBIO ($IJBSDMP), IJBXDPIO ($IJBSDMP)
1I52I	IJBXLBIO ($IJBSDMP), IJBXDPIO ($IJBSDMP)
1I53D	IJBXLBIO ($IJBSDMP)
1IXXI	$$BATTNB
1P01D	$$BATTNE, $$BATTNE
1P04D	$$BATTNF
1P05D	$$BATTNF
1P10D	$$BATTNG
1P30I	$$BATTNS
1P31I	$$BATTNY
1P32I	$$BATTNZ
1P40D	$$BATTNK
1P41D	$$BATTNK
1P42D	$$BATTNK
1P43I	$$BATTNK
1P51D	$$BATT10
1P52D	$$BATT10
1P54I	$$BATT10
1P55D	$$BATT10
1P56D	$$BATT10
1P60I	$$BATTNB
1P70I	$$BATTNB, $$BATTNH
1S40t	$$BATTND, $$BATTNT
1S41E	$$BATTNT

Message-ID	Phase / Module of Phase in Parentheses
1Y04I	$$BATTND
4C07I	$$BATTN3
P101I	$$BATTS2
P102I	$$BATTN7
P103I	$$BATTN7
P104I	$$BATTN7
P105I	$$BATTS1
P106I	$$BATTS1
P107I	$$BATTN7
P108I	$$BATTS1
P109I	$$BATTN7

Appendix C. Command-to-Phase Cross-Reference

Figure 13. Command-to-phase Cross-Reference

AR Command	Phase(s)
ALLOC	$$BATTNE
ALLOCR	$$BATTNE
ALTER	$$BATTNT
AUTOIPL	$IJBAR
BANDID	$$BATTN9 $$BATTU1 $$BATTU2
BATCH	$$BATTNG
CACHE	$IJBAR
CANCEL	$IJBAR
DSPLY	$$BATTNU
DUMP	$IJBAR
FREE	$$BATTNP
GETVIS	$IJBAR
HCLOG	$IJBAR
IGNORE	$$BATTNC
LFCB	$$BATTF1 $$BATTF5 $$BATTN8
LIBSERV	$$BATTND
LOG	$$BATTNC
LUCB	$$BATTN9 $$BATTU1 $$BATTU2
MAP	$$BATTND
MODE	$$BATTNQ $$BATTNR $$BATTNS $$BATTNY $$BATTNZ
MPXGTN	$$BATTNC
MSECS	$IJBAR
MSG	$$BATTNB
MTC	$IJBAR
NEWVOL	$$BATTNC
NOLOG	$$BATTNC
OFFLINE	$IJBAR
ONLINE	$IJBAR
OPERATE	$IJBAR
PAUSE	$$BATTNC
PRTY	$$BATTN2
PRTYIO	$IJBAR
PWROFF	$IJBAR
QUERY	$$BATTND
RC	$IJBAR
RESERV	$$BATTNP
SDAID	$$BATTN3
SETDF	$$BATTN7 $$BATTS1 $$BATTS2
SETMOD	$$BATTNK
SIZE	$$BATTNF
START	$$BATTNG
STATUS	$IJBAR
SYSDEF	$$BATTND
SYSECHO	$IJBAR
TPBAL	$$BATTN2
UNLOCK	$$BATT10
VOLUME	$IJBAR
* CP	$IJBAR

Index

Special Characters

Numerics

A

B

C

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

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.

SC33-6324-00