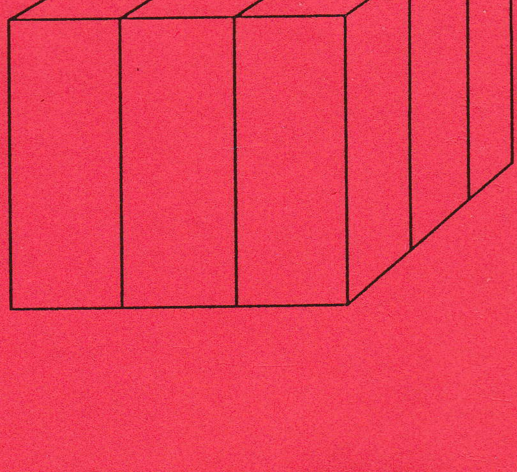
IBM

VSE/Advanced Functions Diagnosis Reference

Supervisor



VSE/Advanced Functions Diagnosis Reference

Supervisor

Program Number 5666-301

Order Number LY33-9107-0
File No. S370/4300-36

First Edition (March 1985)

This edition applies to Version 2 Release 1 of IBM VSE/Advanced Functions, Program Number 5666-301, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30XX and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

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 program product in this document is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

IBM Corporation
Dept. 6R1
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

IBM Deutschland GmbH
Dept. 3248
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This manual is intended primarily for use by IBM personnel responsible for program service. It is one of three publications that describe the design and the internal control flow of the VSE/Advanced Functions Supervisor. The manual supplements the program listings by providing text and charts as follows:

- Chapter 1: Introduction

  Provides general information about the VSE supervisor, its basic functions, storage organization in 370 mode and ECPS:VSE mode, and storage allocations.

- Chapter 2: Design Information

  Contains a detailed description of the various supervisor functions and components. These descriptions provide information necessary to become familiar with the internal logic of the supervisor.

- Chapter 3: Organization Information

  The overview charts of this chapter show the sequence of significant program steps as well as interfaces and linkages between different routines.

- Chapter 4: Data Area Information

  Layout of commonly used supervisor data areas and control blocks.

- Chapter 5: Diagnostic Aids

  In this chapter information is provided which may be especially helpful in diagnosing program errors.

- Appendixes:
  A: Describes the supervisor generation macros.
  B: Contains descriptions of internal VSE macros.
  C: Contains a list of VSE device type codes.
  D: Contains a quick reference list of supervisor calls (SVCs).
  E: Contains samples of track hold processing.

Related Publications

The other two publications describing supervisor functions are:

- VSE/Advanced Functions Diagnosis Reference: Error Recovery and Recording Transients, LY33-9108
- VSE/Advanced Functions Diagnosis Reference: Logical Transients and $IJBSxxx Phases, LY33-9109

For overall system logic, the following manuals are to be used in addition:

- VSE/Advanced Functions Diagnosis Reference: Initial Program Load and Job Control, LY33-9110
- VSE/Advanced Functions Diagnosis Reference: Librarian, LY33-9111
- VSE/Advanced Functions Diagnosis Reference: Linkage Editor, LY33-9112

For efficient use of Diagnosis Reference publications, the reader should be familiar with the information contained in:

- IBM System/370 Principles of Operation, GA22-7000
- IBM 4300 Processors Principles of Operation, GA22-7070
- VSE/Advanced Functions System Management Guide, SC33-6191
- OS/VS-DOS/VSE-VM/370 Assembler Language, GC33-4010

Procedures for isolating problems and analyzing storage dumps are contained in:

- VSE/Advanced Functions, Diagnosis: Service Aids, SC33-6195

Titles and abstracts of other related publications are listed in the:

- IBM System/370, 30xx and 4300 Processors Bibliography, GC20-0001.

CONTENTS

FIGURES

CHAPTER 1: INTRODUCTION

The SUPERVISOR is that part of the VSE system which controls the execution of programs and which provides common services for them. The supervisor consists of:

- The **Supervisor nucleus**
  part of which may be pageable.
- The **Supervisor transients**
  some of which are executed from the Shared Virtual Area (SVA). (For a listing of companion manuals refer to preface.)
- Several **SVA resident phases**

The supervisor nucleus and the SVA resident phases are loaded at IPL time, whereas transients, unless they execute from the SVA, are loaded as needed from the core image library. Subsequent transients will overlay any previous one in the corresponding transient area, (see below) thus making maximum use of processor storage allocated to the supervisor.

The following labels define the supervisor storage locations which are preserved for the different type of transient routines:

LTA	(Logical Transient Area) ($$B..... Phases)
PTA	(Physical Transient Area) ($$A..... Phases)
RTA	(Ras Transient Area) ($$R..... Phases)
CRTTRNS	(CRT Transient Area) ($$BOCRT. Phases)

The major functions performed by the supervisor are:

- Interrupt processing
- Task dispatching
- Physical input/output control (PIOCS)
- Channel program translation
- Page management
- Storage management
- Resource management
- Job accounting
- Program retrieval (FETCH or LOAD)
- Error recovery and recording
- Operator communication
- Common Supervisor Services (SVCs)

In an installation either an IBM provided supervisor may be used or, a supervisor which meets the installation specific requirements must be generated by means of the supervisor generation macros (refer to Appendix A).
Figure 1 on page 3 illustrates the storage organization of a 9-partition system for MODE=370, Figure 2 on page 6 for ECPS:VSE Mode and VM Mode.

For a detailed physical organization of the supervisor refer to Figure 5 on page 10.

The following supervisor routines (functions) are located in the SVA (Virtual Library) :

- SVC 58: INVPART (IJBSINP)
- SVC 83/84: ALLOCATE/SETLIMIT (IJBSSM)
- SVC 103: Part of I/O for SYSFIL on FBA (IJBFBA)
- SVC 112: MSAT (IJBSSAT)
- SVC 113: XPCC (IJBSXPC)
- SVC 114: VIO services OPEN, EXTND and CLOSE (IJBSVIO)

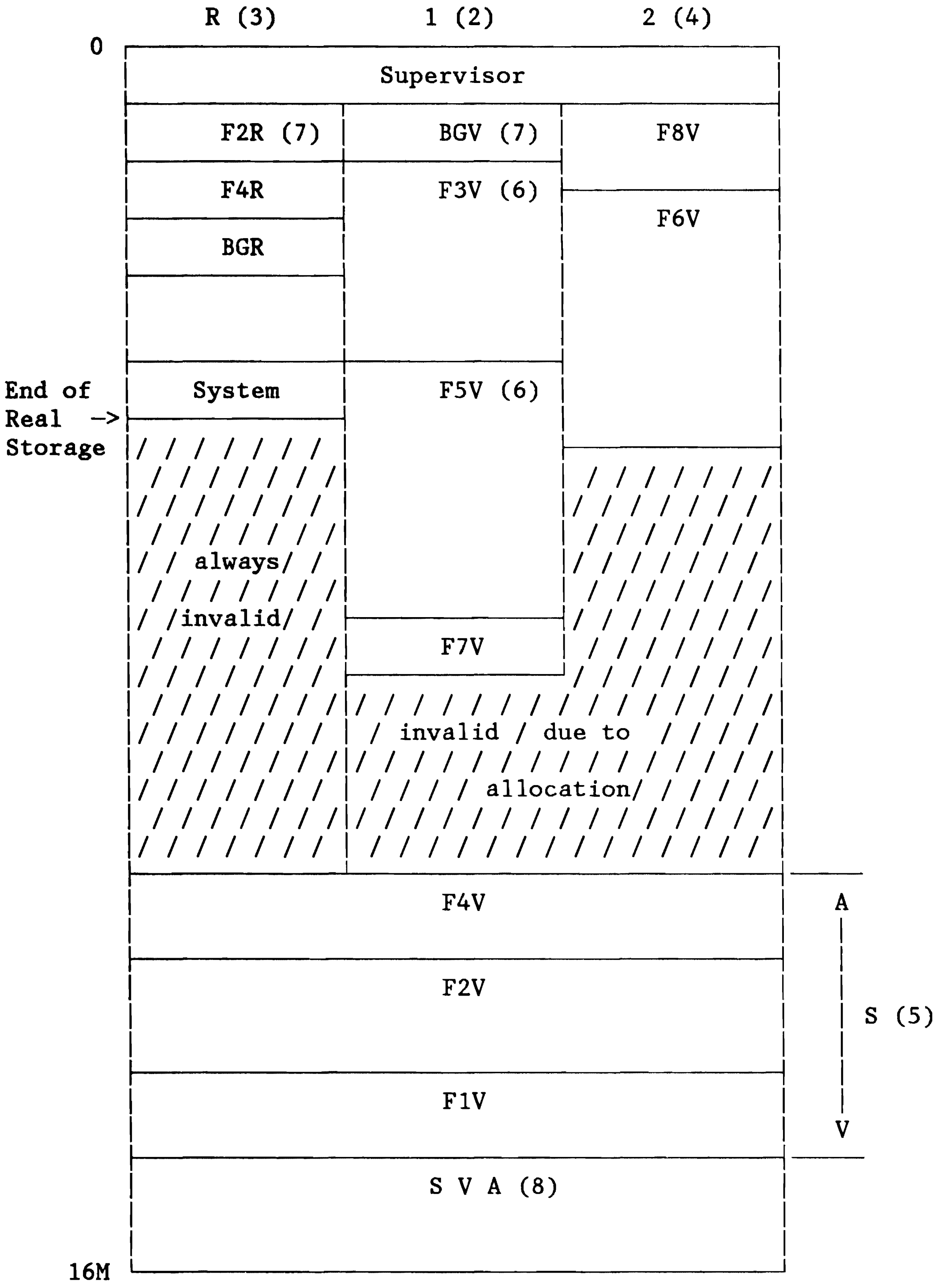


Figure 1. Example of Storage Layout for MODE=370 (1)

1. The organization of a 9-partition system with 2 virtual spaces is shown. Each partition may or may not have a contiguous area of processor storage allocated for program execution in REAL mode.

   An active virtual partition comprises at least 128K bytes in the virtual address area. The virtual partition size is always an integer multiple of 64K bytes.

   The virtual background partition (BGV) is always active.

   The address area of an inactive virtual partition may be reduced to zero.

2. Space 1 is the primary address space, which is used by default to allocate private virtual partitions. It is initialized by IPL with an initial BG size of 1M bytes.

   The BG size can be explicitly changed by a later ALLOC command.

   The example assumes that three other partitions (F3, F5 and F7) are allocated later in the same address space. These new allocations have no implicit effect on the size of the BG partition.

3. Space R is used to allocate real partitions and for real execution. Real allocations are restricted to an area below end of real storage (EOR). No addressability exists between real and private virtual partitions. The system area allocated in space R is used to PFIX SVA pages.

   The segment table for space R does not yet exist after IPL and is allocated at the first EXEC REAL request.

   Allocations for real partitions must be an integer multiple of 4K bytes.

4. Space 2 is a secondary address space, which is created as a result of the first ALLOC request explicitly referring to it. The example assumes that such an ALLOC request was issued for partitions F8 and F6.

   Private partitions are allocated in contiguous areas above the supervisor.

   A free space exists in each address space between private and shared partitions and can be used to increase the total size of the private area, provided that the VSIZE is not exceeded.

5. Shared partitions are allocated in a contiguous area below the SVA. The portion of free space common to all address spaces can be used to increase the total size of the shared area, provided that the VSIZE is not exceeded.

6. There is no fixed ordering of partitions. New partitions are allocated in the specified order within the free space. Reallocations are subject to a set of restrictions, which are reflected by the ALLOCATE return codes.

7. For the layout of a virtual and real partition see **Figure 3 on** page 8.

8. For the layout of the Shared Virtual Area (SVA) see **Figure 4 on** page 9.

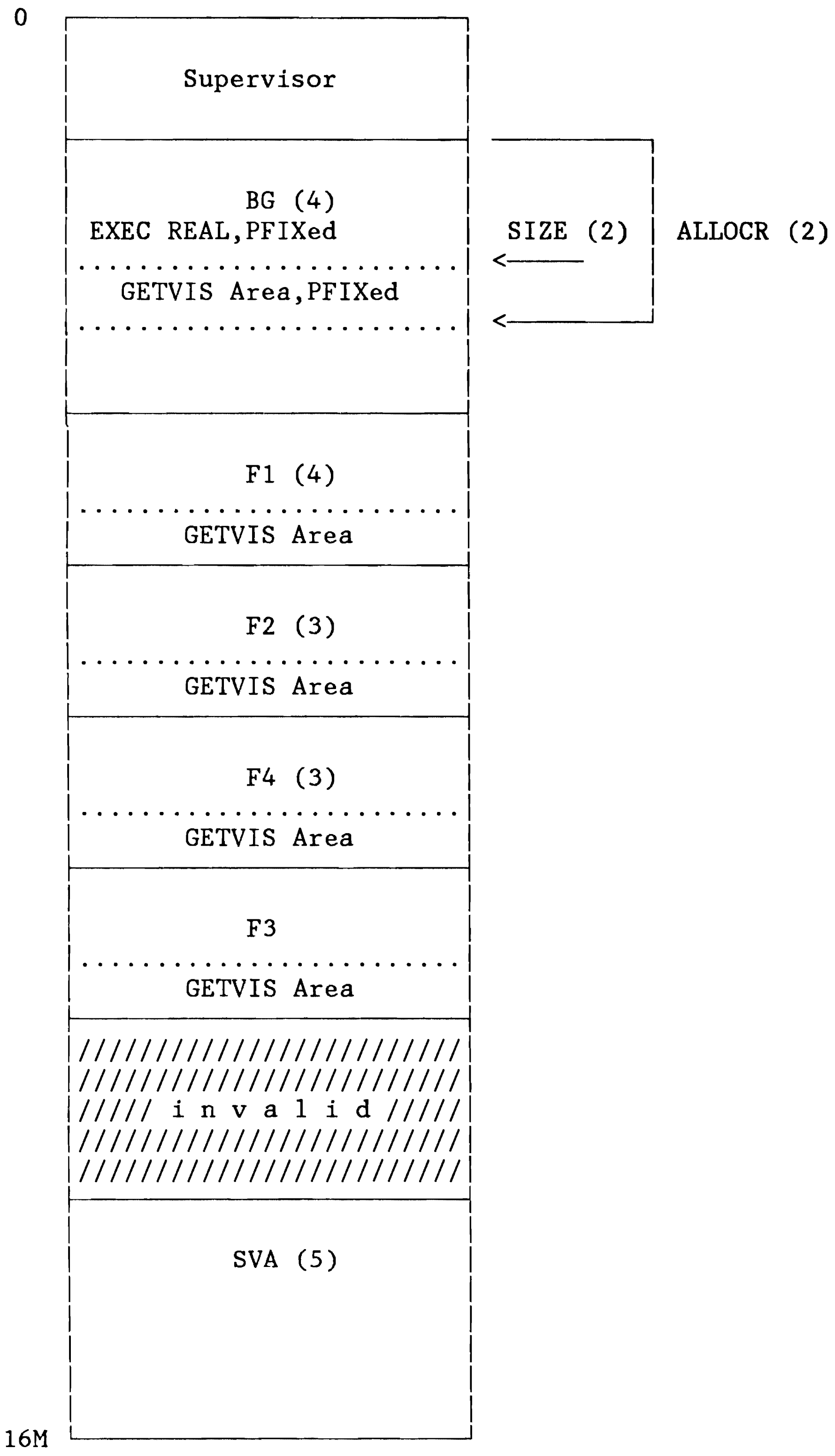


Figure 2. Example of Storage Layout (ECPS:VSE Mode and VM Mode)

1. The layout of a 5-partition system is shown. An active partition comprises at least 128K in the virtual address space and the number of bytes is always an integer multiple of 2K bytes (ECPS:VSE Mode) or 4K bytes (VM Mode).

   The virtual background partition (BGV) is always active.

   The address area of an inactive virtual partition may be reduced to zero.

   After IPL the system consists of the supervisor area and the SVA and BG, which is initialized with a size of 1M bytes. The BG size can be explicitly changed by a later ALLOC command.

2. By use of the ALLOCR command, (as shown for the BG partition), an upper limit for the area to be PFIXed is set.

   Allocation for real partitions must be an integer multiple of 2K bytes (ECPS:VSE Mode) or 4K bytes (VM Mode).

   ECPS:VSE Mode:

   To execute a program in REAL mode, Job Control PFIXes the storage area defined by ALLOCR. If the user wants a real GETVIS area he must specify a SIZE parameter in EXEC statement accordingly.

   VM Mode:

   In this case only the PFIX counters are maintained, EXEC REAL has no effect on the running mode.

3. By use of the ALLOC command, the user may define the partitions according to his needs. These new allocations have no implicit effect on the size of the BG partition.

   The partitions are allocated in contiguous areas above the supervisor.

   There is no fixed ordering of partitions. New partitions are allocated in the specified order within the free space. Reallocations are subject to a set of restrictions, which are reflected by the ALLOCATE return codes.

4. For the layout of a virtual and real partition see Figure 3 on page 8.

5. For the layout of the shared virtual area (SVA) see Figure 4 on page 9.

```
┌─────────────────────────┐
│                         │
│ Partition Save Area     │
│   Program Name          │
│    (8 bytes)            │
│   Program Status Word   │
│    (8 bytes)            │
│   General Registers 9-8 │
│    (64 bytes)           │
│   Job Start Time (1)    │
│    (8 bytes)            │
│   Floating-point        │
│    Registers (32 bytes) │
│                         │
├─────────────────────────┤
│                         │
│  User's Program         │
│                         │
│                         │
│                         │
│.........................│
│  GETVIS Area (2)        │
└─────────────────────────┘
```

Figure 3. Partition Layout (All Modes)

1. Job start time, for the time stamp, is stored in the last 6 bytes of this area (bytes 82-87) when specified.

2. A virtual partition always has a GETVIS area (minimum and default is 48K). In real mode the minimum/default value is OK. If the user wants to have a GETVIS area he must specify it implicitly by using the SIZE parameter in the EXEC statement.

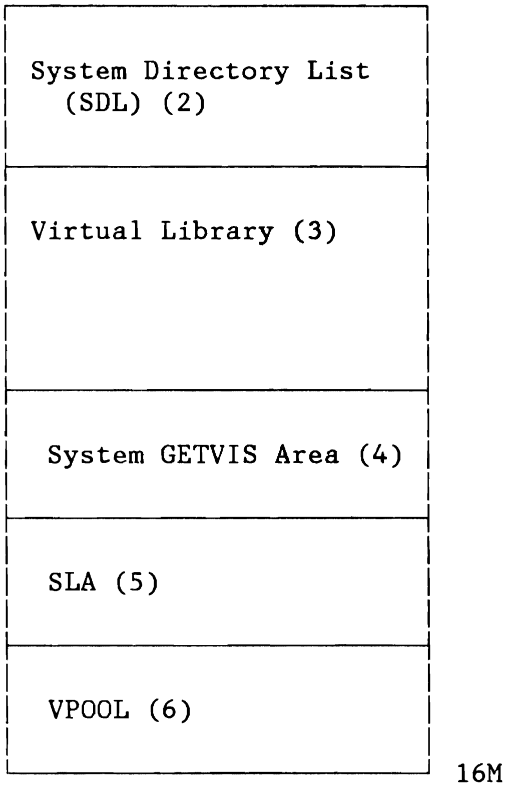


Figure 4. Shared Virtual Area (SVA) Layout (All Modes) (1)

1. Shared Virtual Area (SVA): An area where heavily used reentrant programs are loaded; they can be shared between partitions, and also parts of the system (e.g. End-of-Job processing routines).

2. System Directory List (SDL): In-core directory of highly used programs (phases). For further information refer to "Shared Virtual Area (SVA)" on page 266.

3. Virtual Library (Phase Area): Contains highly used programs (phases) which can be shared between partitions and the system. For further information refer to "Shared Virtual Area (SVA)" on page 266.

4. The GETVIS area for the system can only be used by requestors with a storage protection key of zero.

5. This area is allocated during IPL and used by Label Processing (SLA).

6. This area is allocated during IPL depending on the VPOOL parameter and is used as a buffer pool for VIO.

SUPERVISOR STORAGE ALLOCATION

Figure 5 shows the supervisor generation macros, describes the code they generate and indicates the physical organization of the code in storage.

Generation Macro	Calls	Generated Code	Base Regs used
SUPVR		None, this macro only sets globals	–
FOPT		None, this macro only sets globals	–
IOTAB	SGEND	DSECTS, EQUATES	–
	SGLOWC	HW/SW interface (PSWs, logout areas, etc.)	–
		Various constants and tables must be below 4K. CRTGEN, PIB tables, exit tables, I/O tables, foreground communication regions, etc., having Y-type address pointers in low storage, must be below 32K	–
	SMICR	External interrupt handler	R14
		C-transient, B-transient, and A-transient area	– –
	SGEFCH	Temporary library control blocks and TFIX table for pageable FETCH routines	R9
	ASYCODE	Asynchronous operator communication routines	R9
	ASYTAB	Asynchronous operator communication tables	R9
	SGATAB	Tables having A-type address pointers in low storage (CRTSAV, SDAGDT, ISTAVT, DTSVECTB, SCYVECTB).	–
	DISP	Task selection	R6

Note: For tables/buffers added at IPL see Figure 6 on page 13.

Figure 5 (Part 1 of 3). Supervisor Storage Allocation

Generation Macro	Calls	Generated Code	Base Regs used
IOTAB	SGNUC	Interrupt handler, job accounting in-line routine	R13
	SGPCK	Program check handler	R13
		(DTSMCIC) ICCF Monitor Call intercept routine.	R14
	SGAFCH	Fetch data section (CCWs, control blocks)	R11
	SGDFCH	Fetch overall logic and directory search	R9
	SGCCWT	CCW translation for 370 mode	R8,R9
	SGCCWF	CCW analysis and fixing routine for ECPS:VSE mode	R8,R9
	SGSVC	Various SVC routines	R13
	SGSVCX	Various SVC routines	R13
	MCRAS	Machine/Channel Check Handler, RTA	R15
	SGSTAR	System track algorithm routines	R9
	SGIOS	SVC0 (EXCP) and SVC15 (SYSIO) routines, (SGSCHED) Channel scheduler routine, (IOINTER) I/O interrupt handler, (SGMIH) Missing interrupt handler, (SGDSK) Disk error recovery routine, (SGSERI) Service task interface and data	R13 R13 R9 R13 R13,R14 R12
	SGCFCH	Fetch SVC routines	R13
	SGERP	Interface to ERP transients	R13
	SGAP	Asynchronous processing SVC routines	R13
	SGTINF	Tasking Interface routines	R12,R13
	DTSSVCIC	ICCF SVC intercept routine	R14
	DTSSVCIN	ICCF SVC routine	R14

Note: For tables/buffers added at IPL see Figure 6 on page 13.

Figure 5 (Part 2 of 3). Supervisor Storage Allocation

Generation Macro	Calls	Generated Code	Base Regs used
IOTAB	SGRM	Resource management SVC routines	R13
		Tasking control blocks, (SGPDATA) Data for page manager	R12 R8
	SGLOCK	LOCK, UNLOCK routines	R13
	SGAM	CDLOAD, GETVIS, and FREEVIS routines, (SGAMSUBR) Subroutines of SGAM	R14 R14
	SGNPGR	Allocate Programmer Logical Units (LUBS)	R13
	SGBFCH	Input buffer, program fetch and I/O processing	R9
	SGSER	Automatic Volume Recognition and related SVC routines	R13
		SGSLDUP, SLD update routine, DASD sharing only	R14
	SGACF	Security and Audit support.	R13
	SGXECB	Cross partition common SVC routines.	R13
	SGACCT	GETJA SVC routine. Change/Display Priority SVC routines	 R13
	SGINF	Logical SV/PP common SVC routines	R12
	SGIUCV	IUCV-VCNA connection	R13
	SGPREAL	Get/free processor storage for 370	R9
	SGPMR	Page manager (SGPSVC) VIOPOINT service, (SGPLLEV) Load leveler, (SGPFIX) Fixing routines, (SGPOPT) Page in SVCs	R9 R9 R15 R9 R9
		IPL initialization routines, CCW translation copy buffers	R7,R9

Note: For tables/buffers added at IPL see Figure 6 on page 13.

Figure 5 (Part 3 of 3). Supervisor Storage Allocation

The following table shows the supervisor tables and buffers which are allocated at IPL time.

Copy buffers
Channel queue
CCW chains for DASD file protection
CCW chains for TAPE set mode
PUB2 areas
PUBX area
Pubscan tables
AVR table
Reentry rate table
Page frame table
Page table
Segment table
Page Table Assignment String
Console buffer
Hardcopy buffer
SYSREC buffer
PAGEIN table
Extended logout areas: IOEL, MCEL
Phase load lists
VIO/VPOOL area
Device control blocks
External interrupt buffer for IUCV
Path ID table for IUCV

Figure 6. Supervisor Areas, Allocated at IPL Time

FETCH-PROTECTION FOR THE SUPERVISOR

To prevent unauthorized or unintended use of supervisor information, most areas of the supervisor which are exclusively used by the supervisor are fetch protected.

An overview of areas that are not fetch protected is given below:

- Low storage area including tables known to be used by problem programs
- User TCBs (depending on the TP access method used)
- The pageable part of the supervisor if previously paged-out (370 mode only)
- The tables/areas dynamically allocated at IPL time (see Figure 6 on page 13).

System Communication Region (SYSCOM)

BG Communication Region (BGCOMREG)

CCWs for SYSLOG prefix

PIB Table

PIB Table Extension

ACCTUSER, save area for Job Accounting

FICL

NICL

LUB

PUB Table

PUBOWNER Table

Foreground Communication Regions

DIB Tables

Code and areas for MICR

A-Transient Area

B-Transient Area

C-Transient Area

Code and data for ASYNOC

SYSCODE constant + Patch Area

SYSPARM fields

Job Accounting Partition Tables

Job Accounting label save area

Fetch Table

Second Level Directory(SLD)

Track Hold Table

Console Buffer Table

Data needed for CRT support

Table for Access Control Facility

Communication Table for ICCF

Communication Table for VTAM

Recorder File Table

Figure 7. Supervisor Areas/Tables, which are not Fetch-Protected

```
User TCBs if TP=VTAM.
Pageable supervisor code,
          having been paged-out in /370 mode.
```

Figure 8. Supervisor Areas Conditionally not Fetch Protected

CHAPTER 2: DESIGN INFORMATION

This chapter presents the design information by functions.

- Interrupt Processors
  The different interrupt types and supervisor routines to handle these interrupts.

- Dispatcher, Task Selection
  A description of the dispatching of system and user tasks.

- Physical Input/Output Control System (PIOCS)
  A description of device scheduling and I/O interrupt processing.

- Lock Management
  A description of the lock/unlock mechanism.

- CCW Translation and Retranslation
  A description of CCW-translation, retranslation and CCW fixing.

- Page Management
  Virtual storage concept; page handling.

- Storage Management
  Short description of storage management routine.

- Program Retrieval
  FETCH/LOAD operations including SVA usage.

- Machine- and Channel Check Recovery and Recording
  Types of machine checks, channel checks and resulting actions.

- Job Accounting
  Short description of job accounting routines.

INTERRUPT PROCESSORS

The supervisor is designed to operate in an IBM System/370 CPU in the Extended Control (EC) mode, which is determined by the Program Status Word (PSW) bit 12 being set ON, or in one of the two IBM 4300 processors modes (either in ECPS:VSE mode or in 370 mode).

Processing may be interrupted by any of the following conditions:

- Input/Output Interruption
- Program Interruption
- Machine-Check Interruption
- Supervisor-Call Interruption
- External Interruption

An interruption condition consists in storing the current PSW as an old PSW, storing further detail information identifying the cause of the interruption, and fetching a new PSW (Refer to IBM System/370 Principles of Operation). Processing resumes with the appropriate First Level Interrupt Handler as specified by the new PSW.

The first level interrupt handler saves all the information which is necessary to resume the interrupted processing at a later point in time. After initialization control is passed to the second level interrupt handler.
For a more detailed description of the first level interrupt handler refer to Overview Chart, Figure 124 on page 303.

The second level interrupt handler which will be described in detail below performs the actual interrupt processing and after completion returns to the task selection routine.

I/O INTERRUPT

Refer to Physical Input/Output Control System (PIOCS) later in this chapter.

PROGRAM CHECK INTERRUPT

The program check handler inspects the program interruption code and passes control to the appropriate processing routine. It mainly differentiates between programming exceptions and translation exceptions. If the system is running in 370 mode, the program check handler is entered in real mode which means that the DAT-bit in the PSW is off.

Handling of a Normal Program Check

If a normal program check is to be handled, the DAT bit in the current PSW is first turned on if running in 370 mode.
If the program check occurs in the supervisor code the system enters a hard wait, unless one of the following conditions is fulfilled:

- The supervisor failed due to incorrect input parameters passed by the user program. A list of addresses in the Supervisor is scanned to see if the program check occurred at any of these addresses. If so, the user program is canceled.

- A check is made to see if ACF/VTAM is active and executing an SVC 49 (X'31') or 53 (X'35') or one of its appendage routines. If so, that partition is canceled.

- A check is made to see if ICCF (SVC 82 - X'52') or an ICCF intercept routine is active. If so the ICCF partition is canceled.

If the system goes into a Hard Wait, SYSCOM bytes 4 through 7 and low-storage bytes 0 through 3 contain the appropriate hard wait code (see Chapter 5, Figure 315 on page 624).

If the program check occurs in a page handling overlap (PHO) appendage routine or in an I/O appendage routine, the interrupt status and general registers are saved in a separate save area (label SVPCSAVE) and the users program is canceled.

If the program check occurs in the problem Program, it will be canceled, unless a program check exit routine was specified. In this case the Program Check Handler passes control to a special routine at label PCROUT which saves the interrupt status information and general registers in the save area specified by the user's STXIT (PC) macro for the following purposes:

- To restore for continuation.

- To enable the user's PC routine to analyze the status.

- To facilitate analysis of a dump, should a dump be requested (the dump then contains all interrupt information).

To enter the user's PC exit routine, the PSW saved in SVEPSW is modified to point to the users PC exit routine and a special bit in the TCB is turned on, indicating that the PC exit routine is active. A program check encountered at a time while this bit is still on, causes the task to be canceled.

The user's PC exit routine must end with an SVC 17 (X'11' - EXIT PC) to resume processing at the point where it was disrupted. In this case the interrupt status information and general registers are restored to the program save area and the PC routine active bit in the TCB is reset.

Handling of Page Fault Interrupts

Page faults are a special type of program checks and are handled by an extension to the program check handler, the page fault first level interrupt handler (PFFLIH). By means of the RID (Routine identifier, label RID in the supervisor) it is determined what action is to be taken. Figure 9 on page 24 shows the various RIDs along with the actions taken if one of the appropriate routines causes a page fault.

The page fault handler also sets the TIBFLAG. This flag tells the dispatcher how to dispatch the task after the page fault has been handled. The TIBFLAG indicates that control is to be passed to SVRETURN if a supervisor service is to be reactivated.

If no page-fault appendage is provided for the interrupted task, a page fault request is queued for handling by the page management routines and the interrupted task is set not dispatchable (PMRBND).

If an appendage is present for the task, control is passed to the appendage, and the task causing a page fault remains dispatchable unless the page fault occurred during a supervisor service for the task. (Refer to VSE/Advanced Functions Macro User's Guide for a more detailed explanation of Page Fault appendages.)

If, for a task owning an appendage, a page-fault-handling request has been queued previously, the pending request is not queued.

Handling of Pseudo Page Faults (MODE=VM and MODE=370)

(MODE=VM and MODE=370 when running under VM) Pseudo-page-faults are a special type of program checks.

Pseudo-page-faults are page faults detected while processing on a virtual machine that is operating in EC mode with I/O interrupts enabled and for which the SET PAGEX ON command has been issued. When these conditions are satisfied, VM/370 causes a pseudo-page-fault exception by storing the virtual machine address that caused the page fault, reflecting a program interrupt to the virtual machine, and removing the virtual machine from page and execution wait. When VM/370 has satisfied the page request for the virtual machine, it reflects a pseudo-page completion.

For both pseudo-page-fault exception and pseudo-page-fault completion, the VSE virtual machine is removed from the wait state by VM/370 and given control by a program check interrupt.

A pseudo-page-fault is first tested to see if it is a completion. If this is the case and the page brought in is the same as the previous completion with no faults in between, control is returned to the problem program.

If the pseudo-page-fault was a completion, interrupt status is not saved. For completions the waiting task is found in the page wait queue and posted dispatchable. The previous completion address is set equal to the current one for the duplicate test and control is returned to the dispatcher.

If the pseudo-page-fault is an exception it is tested to see if it occurred in the dispatcher. If this is the case control is returned to the dispatcher with disabled PSW. If the fault is not in the dispatcher the page fault address is saved in the TIB, the TIB is enqueued in the page wait queue and the task is set to the WAIT state.

NAME	ID	MEANING	ACTION
SYSTEMID	00	System error condition, for example, page fault in the I/O interrupt handler	Hard wait. (see Note)
REENTRID	04	Page fault or GETREAL request in a reenterable routine.	Save PSW and registers in user task's system save area.
USERID	08	Page fault from a user task or from a system task.	Hard wait X'FFB' if this is a system task and the TCB shows that the task does not expect page faults; else registers and interrupt status are saved in the users save area. If the task operated in disabled mode, the task is canceled with cancel code X'15'; otherwise the page request is enqueued.
APPENDID	0C	Page fault in I/O appendage routine	Task is canceled with cancel code X'36'.
RESVCID	10	Page fault in SVC 7 (X'07') in SVC 13 (X'0D')	Set RETRYSVC bit in TIB save interrupt status and registers in user save area; enqueue page request.
DISPID	14	Page fault in a routine which does not require any information to be saved, e.g. page fault in the dispatcher.	Enqueue page request.
PFARID	18	Page fault in a page fault appendage routine.	Save interrupt status and registers in an internal save area and cancel user task with cancel code X'0E'.

Figure 9 (Part 1 of 2). Routine Identifiers (RID) as Used by the Page Fault Handler (PFFLIH)

NAME	ID	MEANING	ACTION
ETSSID SUBSYSID	1C	Page fault in subsystem	Save interrupt status and registers in an internal save area.
MICRID	20	Page fault in MICR or subsystem appendage.	Save interrupt status and registers in an internal save area and cancel user task with cancel code X'0E'
	40 . . . . FF	Page fault in a gated Supervisor service.	Close gate to routine (routine cannot be used until gate is opened). Save PSW and registers in the user task's system save area; set TIBFLAG to return to SVRETURN. Enqueue page request. (Any task accessing a gated resource is put in a wait state and is marked resource bound. It is released from the wait state when the resource is ungated after the page request has been completed.)

Note: Refer to "Hard Wait Codes" on page 624.

Figure 9 (Part 2 of 2). Routine Identifiers (RID) as Used by the Page Fault Handler (PFFLIH)

EXTERNAL INTERRUPT

The external interruption provides a means by which the CPU responds to various signals originating either from within or from outside the system. The sources that may present a request for an external interrupt are:

- Clock comparator
- CPU timer
- External interrupt key
- External signal
- VM/IUCV (VMLE only)

Refer to "External Interrupt Routines" on Figure 133 on page 320.

MACHINE CHECK INTERRUPT

The resident machine check handler (MCH) analyzes the machine check interruption code and tests the problem state bit (Old PSW bit 15). The action taken depends on the conditions detected. For a more detailed description refer to "Machine Check and Channel Check Handling" on page 291 later in this chapter and to Figure 221 on page 460.

SUPERVISOR CALL INTERRUPT (SVC)

The different processing routines, refer to Charts Figure 134 on page 323, are entered by the First Level Interrupt Handler (FLIH). Some SVCs are optional and cause a CANCEL (ERR21) if the supervisor was generated without the appropriate option. A short functional description can be found in Appendix D.

After completion of the requested service (SVC), control is generally passed to the task selection routine. The only exception is SVC 107 (X'6B' - FASTSVC) which may return directly to the issuing program. The following pages will describe any SVC in detail.

Supervisor Call Services

SVC 0 (X'00' - EXCP)

Executes a channel program (EXCP). The address of the user's control block (CCB/IORB) is contained in general register 1. A more detailed description is given in "Physical Input/Output Control System (PIOCS)" on page 111.

SVC 1 (X'01' - FETCH)

Fetches a phase. A FETCH loads a phase from the IJSYSRS.SYSLIB sublibrary (SYSLIB) or one of the concatenated private sublibraries (PSUBLIB) and branches to the entry address in that phase.

The directory entry may be found in the SYSTEM DIRECTORY LIST (SDL), in local entries, in one of the PSUBLIB directories (if there are concatenated sublibraries assigned), or in the SYSLIB directory. A phase residing in the SVA is not loaded into the user partition.

The load and entry addresses are obtained from the directory entry for the phase being fetched. The storage address of the phase name or the address of the parameter list must be supplied in general register 1 before this SVC is issued. For a relocatable phase the ENTRY and LOAD address is relocated. The entry address contained in the associated directory entry can be overridden by a user-supplied entry address in general register 0.

If the access control option is active, the SVC routine checks whether the issuer is authorized to fetch the phase.

For a more detailed description, refer to "Program Retrieval" on page 255.

SVC 2 (X'02')

Fetches a logical transient ($$B- or B-Transient).
Loads a B-transient from the SVA, the IJSYSRS.SYSLIB sublibrary (SYSLIB) or one of the concatenated private sublibraries (PSUBLIB) into the Logical Transient Area (LTA) and enters the logical transient at its load address plus 8 bytes. The directory entry for the phase may be found in the SDL, in the SYSLIB directory, or in one of the PSUBLIB directories (if there are concatenated sublibraries). For a more detailed description, refer to "Program Retrieval" on page 255.

If the VSE/Advanced Functions "Fast B- and C-Transient Fetch" is supported, the logical transients can be loaded into the LTA without activating the Fetch (FCH) system tasks.

The SVC 2 (X'02) routine moves the phase from the SVA directly into the LTA if all of the following conditions are met:

- The directory entry of the phase has been found in the SDL.
- The directory entry of the phase has already been activated.
- The phase resides in the SVA and it is self-relocatable.

The storage address of the logical transient phase name must be supplied in general register 1 before this SVC is issued. To ensure system integrity, it is required that the logical transient phase name starts with $$B.

The logical transient is loaded at the origin of the LTA and this address is put into general register 15, which may then be used by the transient as a base register.

Only one task can use the LTA at a time. If the area is already occupied by another task, the task requesting the LTA will be set "LTABND" until the LTA is released by the occupying task (SVC 11 - X'0B', see also Figure 242 on page 522).

The SVC 2 (X'02') routine supports the following routines:

- $$BACLOS
- $$BDUMP
- $$BEOJ3A
- $$BEOJ4
- $$BJDUMP
- $$BPDUMP

If an SVC 2 (X'02') has been issued for one of these phases, the Terminator is entered and the SVA resident routine gets control.

The terminator and end-of-task (EOT) processing is described in chart Figure 127 on page 309.

If Access Control option is active, B-Transient routines must reside in protected libraries.

SVC 3 (X'03')

Provides an interface between the supervisor and $IJBSEOT. An SVC 3 (X'03') waits for termination of I/O requests that belong to the partition or task that is being canceled or has reached the End-Of-Job step.

SVC 4 (X'04' - LOAD)

A phase from the IJSYSRS.SYSLIB sublibrary (SYSLIB) or one of the concatenated private sublibraries (PSUBLIB) is loaded and control is returned to the requesting task.

The directory entry may be found in the SDL, in local entries, in one of the active PSUBLIB directories or in the SYSLIB directory. A phase residing in the SVA is not loaded into the user partition.

The storage address of the phase name or the address of a parameter list must be supplied in general register 1 before this SVC is issued. The user may override the link edited load address by supplying a load address in general register 0.

Upon return to the user, general register 1 contains the actual (relocated) entry point address of the phase. General register 0 points to the active directory entry in storage if a local directory list was supplied (parameter LIST=) and the phase was found in this local list. If the phase was not found in the list, register 0 contains 0. (In this case, the load must be performed from the sublibraries or the SDL/SVA).

If the access control option is active, the SVC routine checks whether the issuer is allowed to load this phase.

For a more detailed description, refer to "Program Retrieval" on page 255.

SVC 5 (X'05' - MVCOM)

When issued by a user through a MVCOM macro, it modifies the partition communication region in the supervisor as specified by the parameters of the MVCOM macro.

When issued by the ERP system task, another physical transient phase, the name of which is contained in the Error Block (ERBLOC) is requested to be loaded into the physical transient area (PTA), and to be entered at its load address plus 10 bytes.

SVC 6 (X'06' - CANCEL)

Cancels a program, task, or partition. This is usually achieved by the requesting program, task, or subtask issuing a CANCEL or CANCEL ALL macro.

If a subtask issues CANCEL, only that subtask is terminated. If a maintask issues CANCEL, or a subtask issues CANCEL ALL, then the entire partition is canceled. The maintask is always the last one to be terminated.

- CANCEL macro issued by a maintask without subtasks:
  the issuing task is terminated normally:

  - Cancel code 35 (X'23') is posted to the issuer's TIB.
  - Message '(issuer*) CANCELED DUE TO PROGRAM REQUEST'.

- CANCEL macro issued by a subtask:
  the issuing subtask is terminated normally, and the action of the SVC 6 (X'06') routine for this subtask is the same as described above for a maintask without subtasks:

  - Cancel code 35 (X'23') is posted to the issuer's TIB.
  - Message '(issuer*) CANCELED DUE TO PROGRAM REQUEST'.

- CANCEL macro issued by a maintask with subtasks attached:
  the maintask is terminated normally; attached subtasks are terminated abnormally.

  - Cancel code 29 (X'1D') is posted to each subtask TIB.
  - Cancel code 23 (X'17') is posted to the maintask TIB.
  - Message '(subtask*) CANCELED DUE TO MAINTASK TERMINATION'.
  - Message '(issuer*) CANCELED DUE TO PROGRAM REQUEST'.
  - A dump is forced at the start of the termination of the maintask if the DUMP option is active (DUMP=YES).

- CANCEL ALL macro issued by a subtask:
  the issuing subtask is terminated normally; other subtasks and the maintask are terminated abnormally.

  - Cancel code 35 (X'23') is posted to the issuing subtask TIB.
  - Cancel code 28 (X'1C') is posted to each of the other subtasks PIBs and to the maintask TIB.
  - Message '(issuer*) CANCELED DUE TO PROGRAM REQUEST'.
  - Message '(main or subtask*) CANCELED DUE TO CANCEL ALL MACRO'.
  - A dump is forced at termination of the subtask, if the DUMP option is active (DUMP=YES).

Note: * This is the program name as contained in the task or subtask save area (see Figure 237 on page 517).

If linkages to a user's AB routines have been established through the STXIT (AB) macro, these routines are entered for all tasks that are terminated abnormally by the task that issues the CANCEL or CANCEL ALL macro.

A task, however, that issues an SVC 6 (X'06') never enters its AB routine assuming the CANCEL (ALL) was not issued from within the LTA or a functionally related routine residing in the SVA.

An AB routine normally terminates through a DETACH, EOJ, or CANCEL macro, but an abnormal condition encountered in an AB routine also terminates that AB routine.

SVC 7 (X'07' - WAIT)

Provides the supervisor service for the WAIT macro and waits for the completion of a special event. General Register 1 points to the control block that is to be inspected for the occurrence of a special event. (CCB/IORB/TECB/ECB)

The different events that a user can wait on are:

- I/O Interrupt,
- Timer Interrupt,
- Program POST request.

If the event bit (byte 2, bit 0 in the control block) has not been turned on, the task status of the issuing task is set to "I/O-BOUND" and its PSW is set up to reissue the SVC 7 (X'07'), whereas, in case the event has already occurred, control is directly passed back to the issuing task to the instruction following the SVC X'07' instruction.

SVC 8 (X'08')

Provides the supervisor support to temporarily return from a logical transient to the problem program. This SVC may be issued only from the logical transient area (LTA) and does not free this area. The entry address to the problem program must be specified in general register 14.

The task selection routine loads the problem program registers. General registers 0 and 1 are passed unchanged to the problem program. To return to the logical transient, the problem program must issue an SVC 9 (X'09').

SVC 9 (X'09' - LBRET)

Provides the supervisor support to pass back control from the problem program to the logical transient routine. An SVC 9 (X'09') can only be issued by the problem program. The task selection routine loads the logical transient registers. General registers 0 and 1 are passed unchanged to the logical transient routine.

SVC 10 (X'0A' - SETIME)

Sets a timer interval.

SVC 11 (X'0B')

Returns from a B-transient to the problem program releasing the B-transient area (LTA). SVC 11 (X'0B') is invalid if issued by a phase or program other than an active B-transient. The logical transient area is released for use by other tasks.

SVC 11 (X'0B') is also used to return from the SVA resident Terminator or EOT routine to the supervisor. The terminator routine in the supervisor releases the SVA resident terminator routine for use by other tasks.

The terminator and end-of-task (EOT) processing is described in chart Figure 127 on page 309.

SVC 12 (X'0C')

Register 1 must contain a non-zero value. The function of this SVC depends on byte 0 of register 1.

- If byte 0 of general register 1 is X'FF', this SVC provides the supervisor support to set or reset flags in a specified byte of the partition's communication region. The user has provided a displacement in byte 2 and a mask in byte 3 of general register 1. The mask is ANDed with the byte at the specified displacement in the partition communication region. If byte 2 and 3 are X'00FF', SVC 12 (X'0C') supplies the Partition protection key in the PSW.

- If the byte contains a value other than X'FF', the SVC performs an AND operation on the linkage editor control byte at displacement 57 of the partition's communication region using as a mask the byte pointed to by bytes 1 through 3 of register 1.

SVC 13 (X'0D')

Register 1 must contain a non-zero value. The function of this SVC depends on byte 0 of general register 1.

- If this byte is not X'FF', this SVC supplies the supervisor support to set flags in the linkage control byte (displacement 57 in the partition communications region). The user has provided the address of a mask (1 byte) in general register 1. This mask is ORed with the linkage control byte.

- If byte 0 of general register 1 is X'FF', this SVC supplies the supervisor support to set flags in a specified byte of the partition communications region. The user has provided a displacement in byte 2 and a mask in byte 3 of general register 1. The mask is ORed with the byte at the specified displacement in the partition communication region. If byte 2 and 3 are zero, the SVC 13 (X'0D') supplies the PSW key zero.

SVC 14 (X'0E' - EOJ)

This is the normal End Of Job (EOJ) service. Cancel code 16 (X'10') is posted to the task information block (TIB) for the program issuing the SVC 14 (X'0E'). Refer to Figure 128 on page 313 for the general cancel routine. The next time the terminated program is selected by the task selection routine, a branch is made to the Terminator routines.

If any EOJ clean-up routine returns with SVC 14 from SVA, the LTA is freed (if owned by clean-up routine) and clean-up (EOJ-) processing is continued.

SVC 15 (X'0F' - SYSIO)

Provides the supervisor service for the SYSIO macro and Executes a channel program prior to normal (SVC 0) I/O requests, already enqueued for the same device but not yet started (head queueing). The address of the I/O control block (CCB/IORB) is contained in register 1. SVC 15 (X'0F') is treated as an EXCP-REAL request and can be used by system tasks only.

Processing of the SVC 15 (X'0F') is similar to SVC 0 processing (refer to channel and device scheduling later in this chapter) except that the SVC 15 (X'0F') does provide the ability to make use of reserved channel queue entries.

An internal I/O scheduling priority is assigned to each System task to determine the proper place within the I/O request chain queued to the requested I/O device. This scheduling priority differs from the dispatching priority and is determined by the head queue request priority table (HQTPRI). The system task ID is used to index a 1-byte field in this table which contains the I/O scheduling priority. From its nature, an SVC 15 (X'0F') always supersedes an SVC 0.

SVC 16 (X'10' - STXIT PC)

Provides support for the STXIT PC macro. The address of the user routine which is to be entered in case of a program check is contained in general register 0 and the address of the save area which is to be provided by the user is contained in general register 1. The save area address (R1) will be validated (ERR25) before both addresses and the caller's PSW key are saved in the PC entry of the task control block (TCB).

The format of the PC routine entry is shown in Figure 246 on page 526 and the format of the user save area in Figure 239 on page 519.

SVC 17 (X'11' - EXIT PC)

Provides supervisor support for the EXIT PC macro. Returns from the user's PC routine to the next sequential instruction in the program that was interrupted due to a program check. This is accomplished by copying the contents of the user-supplied save area to the problem program save area. Refer also to "Program Check Interrupt" on page 21.

For the format of the PC routine entry, refer to Figure 246 on page 526, and for the format of the user save area to Figure 239 on page 519.

SVC 18 (X'12' - STXIT IT)

Provides support for the STXIT IT macro. The address of the user routine which is to be entered in case of an elapsed time interval, is contained in general register 0 and the address of the save area which is to be provided by the user is contained in general register 1. The save area address (R1) will be validated (ERR25) before both addresses and the caller's PSW key are saved in the IT entry of the task control block (TCB).

The format of the IT routine entry is shown in Figure 246 on page 526 and the format of the user save area in Figure 239 on page 519.

The entries in the field named TIBITREQ are either active or inactive. An active entry contains the significant part of the clock comparator (= end of interval) in the first 6 bytes of this field, followed by a 2-byte displacement to the TIB address table entry of the next task in chain. The lowest value occupies the first position of the table, the highest value the position before the inactive entries.

All bits of an inactive entry are set to one. The last entry is always inactive and all entries are set inactive right after IPL.

The clock comparator is always set to the value contained in the first entry of the chain (refer to Figure 10).

Significant part of clock comparator value	see Note
0 5	6 7

Note: Displacement to TIB address of next task in chain.

Figure 10. Interval Timer Request Entry in TCB

SVC 19 (X'13' - EXIT IT)

Supervisor support for the EXIT IT macro. Returns from the user's IT routine to the next sequential instruction in the program that was interrupted due to the clock comparator interrupt. This is accomplished by copying the contents of the user-supplied save area to the problem program save area.

For the format of the IT routine entry, refer to Figure 246 on page 526, and for the format of the save area to Figure 239 on page 519.

SVC 20 (X'14' - STXIT OC)

Provides support for the STXIT OC macro. The address of the user routine which is to be entered for operator-communication, is contained in general register 0 and the address of the save area which is to be provided by the user is contained in general register 1. The save area address (R1) will be validated (ERR25) before both addresses and the caller's PSW key are saved in the OC entry of the partition control block (PCB). Only the maintask can process the interruption.

The format of the OC routine entry is shown in Figure 246 on page 526 and the format of the user save area in Figure 239 on page 519.

SVC 21 (X'15' - EXIT OC)

Supervisor support for the EXIT OC macro. Returns from the user's OC routine to the next sequential instruction in the program that was interrupted by the attention routine MSG command. This is accomplished by copying the contents of the user-supplied save area to the problem program save area.

For the format of the OC routine entry, refer to Figure 246 on page 526, and for the format of the save area to Figure 239 on page 519.

SVC 22 (X'16')

Indicates to the system, that the issuing task does not allow another task to issue a SVC 22 (X'16' - SEIZE) until this same task has issued another SVC 22 (X'16' - RELEASE). This SVC is intended for system components only. The PSW protection key must be zero, otherwise the issuing program is canceled (ERR21).
Any SEIZE request from a task while another task has already got the SEIZE state will result in setting the requesting task "SEIZEBND (X'73')".

If byte 3 of general register 0 is zero, the system mask is set to disable I/O and external interrupts; if the byte is not zero, the system mask is set to enable I/O and external interrupts.

If general register 0 is negative, the user protection key is set in the user's PSW.

SVC 23 (X'17')

Retrieves the load address for a specified phase from the directory entry for the phase. The program issuing an SVC 23 (X'17') is canceled if the PSW protection key is not zero (only job control and B-transient programs can issue an SVC 23 (X'17')).

Register 1 contains the storage address of the phase name and register 0 contains the address where the load address is to be stored. If the phase is relocatable, the load address returned is the relocated load address. The high order byte of the storage area is not changed.

The fetch routine scans the SDL, the IJSYSRS.SYSLIB sublibrary (SYSLIB) and the active chain of private sublibraries (PSUBLIB) for the directory entry of the phase.

For more details, refer to "Program Retrieval" on page 255.

SVC 24 (X'18' - SETIME)

Provides the supervisor service for the SETIME macro. The address of the user's Timer Event Control Block (TECB) is contained in general register 0 and the time interval that the user wants to be set is contained in general register 1. The TECB address is saved in the IT entry in the TCB. For the format of the TECB-IT entry refer to Figure 11.

TECB address	0
0 3	4 7

Figure 11. Format of IT Entry when Used with TECB

This service resets the event bit (byte 2 bit 0) in the TECB and then adds the specified time interval to the present value of the TOD clock to get the absolute time, when the clock comparator interrupt is to occur. This value is stored into the TIBITREQ entry. The task's TIB is inserted in ascending order of the calculated timer value into the ITREQ chain. If the new value is smaller than all the values already contained in the chain, that is, the new value is the first entry in the chain, then the clock comparator is set to this value.

The event bit is set when the clock comparator interrupt occurs. So, if the issuing task wants to wait for this event, it has to issue a WAIT or WAITM (SVC 7 or SVC 29 (X'1D')) macro.

SVC 25 (X'19' - HALTIO)

Provides the supervisor service for the HALTIO macro. Halts an I/O operation on a specified device. The address of an I/O control block (CCB / IORB) is contained in general register 1.
If the SVC 25 (X'19') is used by a program other than OLTEP, an HDV instruction is issued to the device provided that it is a teleprocessing device, it is not a BTAM controlled 270x device, and an I/O interrupt is pending for this device. If the specified device is not a teleprocessing device or if no interrupt is pending, this routine directly returns to the issuing program.
In case the SVC 25 (X'19') is for a BTAM-controlled 270x device the SVC 25 (X'19') causes the system translated channel program (polling loop) to be modified such that the polling loop is discontinued.

If OLTEP is the issuing program, a HDV instruction is issued to the device provided that an I/O interrupt is pending for the device.
In case the I/O operation has not yet been initiated or was already completed, due to device sharing, SVC 25 (X'19') just forces the channel queue entry to be dequeued from the channel queue.

SVC 26 (X'1A')

Validate address limits. The program issuing an SVC 26 (X'1A') is canceled if the PSW protection key is not zero.
The lower address must be specified in general register 1, and the upper address must be specified in general register 2.

If a CRT routine issues an SVC 26 (X'1A'), control is always returned to the CRT routine, even in case of an error. For any other routine, if either address is outside the requester's partition, the task is canceled (ERR25 on Figure 128 on page 313).

SVC 27 (X'1B')

Provides exactly the same service as SVC 25 (X'19') does, except that SVC 27 (X'1B') will ensure that due to a HDV instruction a CHANQ entry will not be dequeued from the channel queue.

SVC 28 (X'1C' - EXIT MR)

Provides return from a user's stacker select routine back to the MICR external interrupt routine. This SVC is optional and causes a cancel (ERR21) if issued at any point other than in a stacker select routine for MICR devices.

SVC 29 (X'1D' - WAITM)

Provides supervisor support for the WAITM macro. On entry, general register 1 contains the address of an ECB list. The ECBs are all checked for the traffic bit (byte 2 bit 0). When an ECB is found with the traffic bit already posted, the SVC 29 (X'1D') routine returns with the address of the posted ECB in general register 1. If none of the specified ECBs has the traffic bit yet posted, the task is set "I/O-BOUND" and it's PSW is modified, so that the SVC will be reissued when the task is selected again (RESVC).

SVC 30 (X'1E')

Reserved.

SVC 31 (X'1F')

Reserved.

SVC 32 (X'20')

Reserved.

SVC 33 (X'21' - COMRG)

The COMRG macro (SVC 33 - X'21') provides the supervisor service to immediately enter the task selection.

SVC 34 (X'22' - GETIME)

Provides the supervisor service for the GETIME macro; updates the date field in the communication region of the issuing partition. Upon return, general register 1 contains the time of day in timer units (1/300 sec.)

SVC 35 (X'23')

Protects a track (or block if an FBA device) from use by more than one task at a time. A task requesting a held track must wait until the track is free. If more than sixteen holds on a track are attempted, the requesting task is canceled.

Exits are to execute the I/O operation, or to repeat the SVC (label RESVC) if the track is already held. At RESVC, the program old PSW is set to execute the SVC 35 (X'23') again, and a branch is taken to task selection. See Figure 310 on page 609 for the format of the track-hold table.

SVC 36 (X'24' - FREE)

FREEs a track (or block if an FBA device) that is held by the task issuing the FREE macro. An attempt to free a track not owned by the requester causes the issuing task to be canceled.

SVC 37 (X'25' - STXIT AB)

SVC 37 (X'25') provides supervisor support for the STXIT AB macro.

- OPTION=DUMP
  The address of the user routine which is to be entered in case of an abnormal end is contained in general register 0 and the address of the save area which is to be provided by the user is contained in general register 1.

- OPTION=NODUMP or OPTION=EARLY
  A parameter list will be built. The pointer to this list is placed in registers 0 and 1.

Notes:

1. The save area is a 72-byte area in which the interrupt status information and general registers 0-15 are stored.

2. Contents of the parameter list (EARLY and NODUMP):

X'20' or X'40'	Exit routine address	Save area address	
0	1	4	7

NODUMP	results in setting X'40' in TCBFLAGS,
EARLY	results in setting X'20' in TCBFLAGS.

3. After the invocation of an AB exit routine, register 0 contains the abnormal termination code and register 1 the pointer to the AB save area.

The save area address (R1) will be validated (ERR25) before both addresses and the caller's PSW key are saved in the AB entry of the task control block (TCB).

The SVC 37 returns to the program that issued the STXIT macro.

The format of the AB routine entry is shown in Figure 246 on page 528 and the format of the user save area in Figure 239 on page 521.

SUBSYSTEM SUPPORT VIA OPTION=EARLY
OPTION=EARLY indicates that the AB exit routine has to be invoked for any type of termination (normal and abnormal) and, for a maintask, before propagating the termination to its subtasks.

Restrictions:

- An EARLY exit can only be set once during the whole lifetime of a task.
- Reset is not allowed.
- OPTION=EARLY is supported only for subsystems, depending on their identification via the SUBSID macro.

If the user violates these rules, the setting will be ignored and a return code will be put in register 15:

0 (X'00')	Exit successfully set.
4 (X'04')	Exit already set.
8 (X'08')	Reset not allowed.
12 (X'0C')	No subsystem request.

Normally the TCBABPTR contains the address of the AB exit routine, but if EARLY exit is set and a second exit is used (example: OPTION=DUMP or NODUMP), the second AB pointer will be put into TCBABSEC. However, if an EARLY exit has been established, only this exit will be invoked for any kind of termination.

The maintask and subtasks may have the same or different AB routines. When a subtask is attached after an STXIT AB macro has been issued by the maintask, the subtask gets the AB routine address (from TCBABPTR respectively in case of OPTION=EARLY from TCBABSEC) specified by the maintask only if the ATTACH macro for that subtask has the ABSAVE parameter specified. The subtask can override this by issuing its own STXIT AB macro.

SVC 38 (X'26' - ATTACH)

A subtask is to be established via the ATTACH macro. If no further subtask is available, if the maximum number of subtasks attachable to a partition is exhausted or if no SVA space for task control blocks is available, supervisor will supply an ECB pointer, indicated by the high order bit set to 1 which is stored into the user's R1 before control is returned to the user. The user may now issue a WAIT for this ECB.

If a set of task information and task control blocks is available (otherwise SVA space will be allocated), a subtask is attached by initializing the two control blocks and by inserting the task ID and status byte into the Task Identifier String (TIDSTR) and Task Selection String (TSS), respectively. The newly attached subtask gets a processing priority just above that of the main task.

The issuing task's save area is copied to the subtask's save area. The subtask is set to 'ready-to-run'. Bit 0 of the issuing task's register 1 is set to 0 to indicate a successful attach. Control is then returned to task selection.

SVC 39 (X'27' - DETACH)

Performs normal termination of a subtask. DETACH may be issued by the subtask being terminated, by the maintask, or by the task which attached this subtask. If DETACH is issued by a problem program, the cancel code 16 (X'10' - normal End-Of-Job) is set in the subtask's TIB and the Terminator is entered. At the end of the termination process, DETACH is issued by the EOJ SVA-resident routine setting the subtask inactive and posting its ECB for termination (for layout of ECB see Figure 245 on page 525). The subtask status byte and identifier are removed from the Task Identifier String (TIDSTR) and Task Selection String (TSS). The task is freed; any waiting attach requests and associated supervisor ECBs are posted.

SVC 40 (X'28' - POST)

Used for inter-task communication. POST may be issued by either a maintask or a subtask. It is issued so that a task is aware of the termination of an event. Normal completion of the specified event is posted in the ECB (byte 2, bit 0 = 1). If the SAVE=parameter is present, only the task, owning the save area and waiting for this ECB, is taken out of the wait state; otherwise, all tasks waiting for this ECB are removed from the wait state.

Only tasks running in the issuing partition are affected.

SVC 41 (X'29' - DEQ)

Informs the system that a resource (shared data area) is now available for use by another task. A task may issue the DEQ macro only to a resource that it currently owns. If it attempts to issue the DEQ macro to some other resource, the task is canceled.

If any other tasks are waiting for the resource, the highest priority task ready to run is removed from the wait state and gains control. If no other task is waiting for the resource, control returns to the task that issued the DEQ macro.

If a task terminates without dequeueing all of its enqueued resources, either in its normal coding or in its abnormal termination exit routine, any task subsequently attempting to enqueue the resources is canceled. See Figure 257 on page 536 for the Resource Control Block (RCB).

SVC 42 (X'2A' - ENQ)

ENQ prevents tasks from simultaneous manipulation of a resource (shared data area). This is accomplished by setting all bits of byte 0 of the specified Resource Control Block (RCB) to 1. Then the Event Control Block (ECB) address is placed in bytes 4 through 7 of the RCB. A task attempting to enqueue a resource that is already enqueued by another task is placed in a queue and put in a waiting condition. The old PSW is set to re-execute the SVC 42 (X'2A') and task selection is performed.

A task is canceled if it attempts to nest ENQ(s) of a resource or if it attempts to ENQ a resource that is still owned by a terminated task.

When a task is finished with a resource, it should inform the system by issuing the DEQ macro. Tasks subsequently requesting the resource are canceled, if the tasks owning that resource have been terminated in between. See Figure 257 on page 536 for the Resource Control Block (RCB).

SVC 43 (X'2B')

Reserved.

SVC 44 (X'2C')

Provides the supervisor service to write a SD record (statistical data) onto the recorder file.

General register 1 contains the address of the SD record that the user wants to be recorded onto the recorder file. The specified area is first validated (ERR25) and subsequently all virtual pages containing the statistical data are TFIXed, if they are not already PFIXed.

If the error entry in the ERBLOC area is not available, all pages just TFIXed are TFREEd and the requesting task is set "ERQBND (X'62')" until the ERP task completes its current processing. If the ERROR ENTRY is available, the information passed by the user is saved into the error entry (refer to ALTERNATE ERROR ENTRY description) and the phase name of the first physical transient to be fetched is set ($$ABERA6). The transient ERP is activated to asynchronously do the recording.

SVC 45 (X'2D')

Reserved.

SVC 46 (X'2E')

This SVC can be used by OLTEP only and allows OLTEP to operate in supervisor state. General register 1 contains the address of an OLTEP appendage routine that is immediately to be entered via a BALR interface and which will return via the link register.

SVC 47 (X'2F' - WAITF)

Provides identification to the supervisor for MICR type device multiple waits (WAITF). The same routine is entered as for SVC 29 (X'1D'). However, when no Event Control Block (ECB) is posted, the task's PSW is not modified to reissue the SVC, as is done when SVC 29 (X'1D') is issued.

SVC 48 (X'30')

Fetches a CRT-transient phase. The program issuing an SVC 48 (X'30') is canceled (ERR21) if the PSW protection key is not zero. The first SVC 48 (X'30') is issued during IPL to load the CRT root phase $$BOCRTA into the C-transient area (CRTTRNS).

Each SVC X'30' tries to move the specified phase from the SVA into the CRT transient area. If unsuccessful, the fetch routine retrieves the phase from the SDL, the IJSYSRS.SYSLIB sublibrary (SYSLIB) or the active chain of Private Sublibraries (PSUBLIB) and loads it into the CRT transient area.

The PSW address of the issuing task (CRT) is set to the address of the C-transient area (CRTTRNS) + 10.

SVC 49 (X'31')

Provides I/O services for ACF/VTAM only. Any other user will be canceled (ERR21) when using SVC X'31'. This SVC makes all the provisions necessary to HALT an ongoing teleprocessing I/O operation, or, to START a teleprocessing I/o operation. In case the device is to be halted, this service routine directly returns to task selection. In case a new I/O operation is to be started, this routine passes control to the supervisor Start I/O routine. Refer to "Physical Input/Output Control System (PIOCS)" on page 111.

SVC 50 (X'32')

This SVC forces the issuing task to be canceled ERR09 and is intended to be used by LIOCS (LOGICAL INPUT/OUTPUT CONTROL SYSTEM) for error diagnostics.

SVC 51 (X'33' - HIPROG)

Provides the ability to determine the length of a phase without loading it. On entry to the SVC 51 (X'33') routine, register 1 must point to the storage address of the phase name. This 8-byte area must be followed by an area large enough to hold further directory entry information. The length of the area must be specified in byte 3 of the area itself as the number of halfwords. The rest of the area must be set to X'00'. The FETCH/LOAD service returns the selected part of the directory entry. The area is not altered if no directory entry for the particular phase has been found.

FETCH scans the SDL, the IJSYSRS.SYSLIB sublibrary (SYSLIB) and the active chain of private sublibraries (PSUBLIB) for the directory entry of the phase.

If job control provides a parameter list and sets the flag option to X'01' or X'02', the HIPROG value for this partition will be calculated.

In case of X'01', only the length of the specified phase is considered.

In case of X'02', the HIPROG value is the address of the uppermost byte of the phase with the highest ending address for the corresponding partition. All phases starting with the same four characters as the phase name given on the EXEC statement are considered.

The calculated value is stored in bytes 40 through 43 of the partition communication region. If the phase searched for is in the SVA, the partition start address plus page size will be used.

SVC 52 (X'34' - TTIMER)

Provides the TTIMER macro support.
The remaining time interval (in 1/100 seconds) to elapse before the clock comparator interrupt occurs is returned as an unsigned 32-bit binary number in general register 0.
All zeros are returned if no timer interval was set by the task issuing the TTIMER macro.

If the task issuing the TTIMER macro has an entry in the IT request (ITREQ) chain, the value returned is the difference between the values of the ITREQ entry and the TOD clock.

If TTIMER CANCEL is specified, and the task owns an entry in the ITREQ chain, then that entry is deleted.

SVC 53 (X'35')

Used by ACF/VTAM (and ACF/VTAME) to perform a number of functions, such as enqueuing or dequeueing ACF/VTAM resources or posting ACF/VTAM ECBs. After an SVC 53 (X'35') is issued, the supervisor passes control to ACF/VTAM to perform the required function. Control is passed back to the calling module via the supervisor.

SVC 54 (X'36')

This supervisor call is only valid in 370 mode.
In ECPS:VSE mode the same support is provided by PFREE (SVC 68 - X'44'). SVC 54 (X'36') provides the supervisor support for the FREEREAL function to release page frames to the page pool. These page frames may be released from the partition's REAL address area, or the SDAID area. The task issuing this SVC is canceled (ERR21), if it does not run with a protection key of zero and the phase name is other than SDAID, $$BATTN or $$BVSEPT.

A zero value in general register 2 indicates that the request is issued by SDAID. In this case, the lower and upper limit of the area to be released are obtained from the internal page manager address fields. Control is passed immediately to task selection if no SDAID area exists.

The page frames are freed, one after the other, by updating the corresponding Page Frame Table Entries (PFTEs). The partition PFIX counter in the Storage Management Control Block (SMCB) part of the Partition Control Block (PCB) is decremented by one for each page that is freed.

The released page frames are enqueued at the beginning of the invalid page frame queue.

The SVC 54 (X'36') posts as "READY TO RUN" all tasks waiting for page frames, if more than the minimum number of page frames is available in the Page Selection Queue (PSQ).

SVC 55 (X'37')

This supervisor call is only valid in 370 mode. In ECPS:VSE mode the same function is provided by PFIX (SVC 67 - X'43'). SVC 55 (X'37') provides supervisor support for the GETREAL function to request pages from the page pool for the SDAID area. Control is passed immediately to task selection if such a request is already in progress or if the SDAID area already exists. If the requester does not have protection key zero, and is neither identified as SDAID nor as $$BATTN, the issuing task is canceled (ERR21).

The number of requested page frames is passed in register 0. This value is replaced by the number of page frames that are available for GETREAL if this number is less than the requested number. After handling the request, the number of page frames taken from the main page pool and the address of the SDAID area are passed to the user in registers 0 and 1. Register 0 contains zero and register 1 remains unchanged if no page frames are available.

The SVC 55 (X'37') routine passes the begin and end addresses of the requested SDAID area as parameters to the GETREAL routine. Refer to "GETREAL Request" on page 230.
The begin and end address of the SDAID area are saved in the internal page manager address field.

SVC 56 (X'38' - CPCLOSE)

Issued by VSE/POWER to close printer or punch files written by VSE/POWER and spooled by VM/370 or VM/System Product at the End-of-Job. VSE/POWER passes a parameter list to the SVC. This list contains the HEX address of the device to be closed in low order half of the first word and the device address in EBCDIC in the second word and the job name in the third and fourth words.

SVC 57 (X'39' - GETPRTY)

Allows to display and/or change the partition priorities.

Display request: Can be issued by any task. The following parameters are passed to this routine:

R1: BIT 0=1 BIT 8-31= address of the area to which the information about the current priority setting is to be moved.
R0: Length of the specified area. The length in bytes must be three times the number of partitions.

The priority list (see below) is moved to the area specified in general register 1.

Change request: Issued by the Attention Task following an attention PRTY command given by the system operator. Only one parameter is passed to this routine:

R1: BIT 0=0 BIT 8-31= address of the area containing the priority information (see below) that the system operator wants to be established.

The SVC 57 (X'39') routine scans the priority list and updates the PPRTYOWN field (see Figure 24 on page 91) accordingly.

The format of the priority list as it has to be made available to the SVC 57 (X'39') routine is as follows:

```
|<——low—————————PRIORITY—————————high——>|
|                                       |
┌─────────┬───┬─────────┬───┬──•••─┬─────────┐
|Partition|   |Partition|   |      |Partition|
|   ID    | | |   ID    | | |      |   ID    |
|         | | |         | | |      |         |
└─────────┴─┼─┴─────────┴─┼─┴──•••─┴─────────┘
 0        1 3|  4       5 6|                n
             |             |
             └——>Separator └——>Separator
  Partition ID = BG and F1 through Fn-1 where
             n = number of user partitions
  Separator    = comma(,) or equal sign(=)
```

Figure 12. Format of Priority List

SVC 58 (X'3A' - INVPART)

Issued by job control to initialize a partition. The task issuing this SVC is canceled if it does not run with a protection key of 0. Refer also to the description of the INVPART macro in Appendix C.

The following parameters are passed to this routine:

R2:

zero	Next program in the partition is to run in REAL mode.
not zero	Next program in the partition is to run in virtual mode.

R3: Address of first page to be invalidated.
R4: Address of last page to be invalidated.

If the partition is running in virtual mode and its first page has to be invalidated, the first page is cleared except for the first 200 bytes (save area).

If the next program is to run in REAL mode, the translation mode bit in PIBFLAG0 in PIB is reset (370 and ECPS:VSE Mode only), and the entry in SYSCOM indicating the number of the active partitions running in virtual mode is decreased by one. All copy blocks used by the fast CCW translation routines (REPLICA, CCB, CCW, etc.) are released, and the pages which were TFIXed by these blocks are freed.

The pages are invalidated corresponding to the address range defined by R3 and R4, that is: from the address given in R3 to the address defined in R4. If, however, R3 contains the address of the first page of the partition, it is adjusted to the address of the second page of the partition, thus preventing invalidation of the partition save area. The invalidation is done using a subroutine of the SVC X'3B' routine.

370 mode only:

If the partition is to run in REAL mode, the following actions are taken, in addition to the general actions described above:

- All page table entries belonging to the partition running in REAL mode are initialized.

- The page frame table entries (PFTEs) that correspond to the partition running in REAL mode are initialized and removed from the page selection queue (PSQ). The partition PFIX counter in the storage management control block (SMCB) and the counter for the number of page frames in the PSQ is updated.

- The storage protection key of the page frames of the partition running in REAL mode is set equal to the PIK of the partition.

- The first 200 bytes of the partition running in virtual mode are moved to the first 200 bytes of the partition running in REAL mode. The appropriate entry in the PIB and the SMCB are updated to point to the save area of the partition running in REAL mode instead of pointing to the save area of the partition running in virtual mode.

- The first page of the partition running in virtual mode is invalidated.

Page frames required for the partition which is to run in REAL mode are reserved by the GETREAL routine (refer to "GETREAL Request" on page 230). The number of page frames to be reserved depends on the specification for SIZE in the EXEC statement (analyzed by job control), or on the size of the partition running in REAL mode.

If, during execution of the GETREAL, a page frame in the partition running in REAL mode is found to be failing, the job that is to be initialized is canceled (ERR2D). The save area is not moved to the partition running in REAL mode, the save area pointer in the TCB remains unchanged, and virtual mode is posted in the PIB.

ECPS:VSE mode only:

- In addition to the general actions described above, the number of active partitions running in REAL mode is increased by one.
  - If the partition is to run in REAL mode, the area from the beginning of the partition to the address stored in PPEND of the partition's COMREG is PFIXed, using the same routine as described for SVC X'43'.

SVC 59 (X'3B' - INVPAGE)

In 370 mode, SVC 59 (X'3B') initializes the page table and page frame table entries belonging to specific pages.

For a supervisor generated with MODE=VM, the specified area is cleared, using the 'RELEASE PAGE' diagnose instruction of VM/370.

In ECPS:VSE mode, SVC 59 (X'3B') invalidates the pages of the specified area. Refer also to the description of the INVPAGE macro in Appendix B. The task issuing an SVC 59 (X'3B') is canceled after IPL has been successfully completed.

The following parameters are passed to this routine:

R3: Address located in the first page of the area to be invalidated.
R4: Address located in the last page of the area to be invalidated.
R5:
    >0 storage key for disconnected pages.
    <0 area to be deactivated.

370 mode only:

If the area to be invalidated belongs to an active virtual partition the corresponding page table entries (PTE) are set to B'kkkkp00000001000' where kkkk corresponds to storage key and where p indicates whether the page is fetch protected (p=1). If the area to be invalidated belongs to a non-active part of an virtual partition or to a real partition, the corresponding PTE's are set to B'kkkkp00000101000' where kkkk corresponds to storage key and where p indicates whether the page is fetch protected (p=1). Each PTE within the area defined by R3 and R4 is initialized in that way. If the page referred to by an entry is in processor storage, the page frame table entry of the corresponding page frame is initialized as follows:

- The page frame is marked as unused (the PNRINV bit in S370FLG is set), and the PFIX counter is set to zero.
- The page frame is removed from the page selection queue and enqueued to the top of the invalid page frame queue.
- If a page is found to be TFIXed, the system enters the hard wait state (debug mode only).
- The page frame is cleared.

ECPS:VSE mode only:

Each page within the area defined by R3 and R4 is invalidated as follows:

- If the page is disconnected, the reference, change and page-data-set bits are reset.
- If the page is connected, the same action is taken as for disconnected; in addition, the hold bit of the connected page frame is reset.
- If the page is addressable, the corresponding page frame table entry is removed from the page selection queue, and the page is disconnected by resetting the reference, change and page data set bits.

All copy blocks used by the fast CCW translation routines (REPLICA, CCW, CCB, etc.) are released, and the pages which were TFIXed by these blocks are freed.

SVC 60 (X'3C' - GETDADR)

Calculates from the real address the virtual address of a location within the data area of an I/O request.

For a supervisor with MODE=VM specified the virtual address is not calculated.

Before this SVC is issued, general register 8 must contain the address of the CCW. General register 0 must contain the displacement of the desired address from the start of the I/O area. Using the data address or the address of the indirect addressing list (IDAL) specified in the CCW, the supervisor calculates the virtual address and returns it in general register 15.

If the real address is invalid (in an unused page frame or beyond the end of processor storage), all zeros are returned.

SVC 61 (X'3D' - GETVIS)

Provides the supervisor support for the GETVIS macro. It reserves part of the GETVIS area which may either be part of a partition or part of the shared virtual area (SVA).

On successful completion of the operation, X'00' is returned in general register 15, the start address of the reserved area returned is in general register 1. The length of the area, which must be specified by the user, is contained in general register 0.

For further information see "Storage Management" on page 245 described later in this chapter.

SVC 62 (X'3E' - FREEVIS)

Provides the supervisor support for the FREEVIS macro. It releases a block of virtual storage. The start or subpool name address of the area to be released is contained in register 1. The length of the area to be released is in register 0.

For further information see "Storage Management" on page 245 described later in this chapter.

If the return code in register 15 is not zero, no action was taken by FREEVIS.

SVC 63 (X'3F' - USE)

Allows supervisor controlled access to a system resource as requested by the internal macro USE. SVC 63 (X'3F') requests are converted by the LOCK manager to LOCK requests (refer to "Lock Management" on page 137).

SVC 64 (X'40' - RELEASE)

Provides the supervisor service for the RELEASE macro. A resource previously locked by means of the USE macro (SVC 63 - X'3F') is now to be RELEASEd. SVC 64 (X'40') requests are converted by the LOCK manager to UNLOCK requests (refer to "Lock Management" on page 137).

SVC 65 (X'41' - CDLOAD)

Loads a phase dynamically into the partition GETVIS area when called by the macro CDLOAD.

In case the phase is found in the SVA and the requesting program is not running in REAL mode, then the load will not be performed and just the SVA load address is returned to the caller.

For further information see "Storage Management" on page 245 described later in this chapter.

SVC 66 (X'42' - RUNMODE)

Provides the supervisor service for the RUNMODE macro which returns the mode in which a partition is running. It returns 0 in register 1 if the program is running in virtual mode and returns 4 in register 1 if the program is running in REAL mode.

SVC 67 (X'43' - PFIX)

Fixes as many pages as requested by the PFIX macro. For a supervisor with MODE=VM specified a PFIX request will be ignored. A PFIX request is also ignored if it is issued by a program running in REAL mode, and return code 0 is passed in register 15.

When the SVC 67 (X'43') routine is entered, register 1 must point to the list of pages that are to be fixed. Each entry in the list consists of 8 bytes. The first four bytes contain the beginning address of the area that is to be fixed and the last four bytes contain the length of this area.

370 mode only:

The PFIX requests are gated, that is, a task is set to "PFXBND (X'92')" if it issues a PFIX request for a partition for which another PFIX request is still being processed.

Before a page can be fixed it must be determined whether this can be done immediately or not. If the page occupies a page frame in the area allocated to the partition for REAL mode execution, the page can be fixed at once. If the page occupies a page frame outside the area allocated to the partition for REAL mode execution, a page frame must be selected in that area.

The page frame table entry address of this reserved page frame is stored in the partition control block (label PFTERSVD) by the PFIX routine, or by the PFREE routine (see "SVC 68 (X'44' - PFREE)"), or by the TFREE routine, if a page has to be freed before the page frame can be reserved.

If no page frame is available but some of them are only TFIXed, all page frames in the partition are set to 'not temporarily fixable' and the task is put into the wait state. Processing of the request continues as soon as a page has been freed by either a TFREE or by a PFREE request issued by another task of the same partition.

370 and ECPS:VSE mode:

A page is fixed by increasing the page PFIX counter by 1. If the page was neither PFIXed nor TFIXed, the corresponding page frame table entry is removed from the page selection queue, NPSQE is decreased by 1, and the partition PFIX counter is increased by 1. If a page to be fixed has an invalid address, all pages that have already been fixed for the request are freed, return code 12 is placed in register 15, and control is returned to task selection. Control is also returned to task selection and the already fixed pages are freed if insufficient page frames are available to fix the requested number of pages. Return code 4 will be passed in register 15 if the REAL partition is too small to ever satisfy the request. Return code 8 is passed if the partition is large enough to satisfy the request but has insufficient page frames available to satisfy the request at this time.

SVC 68 (X'44' - PFREE)

Frees as many pages as requested by the PFREE macro. For a supervisor with MODE=VM specified the PFREE request is ignored. A PFREE request may come from a user task or from the RSTRT command processor.

When the SVC 68 (X'44') routine is entered, register 1 points to the list of pages to be freed. Each entry in the list is 8 bytes long. The first 4 bytes contain the address of the area to be freed and the last 4 bytes contain the 370 length of that area. The pages are freed sequentially until the list of requests is exhausted. If a page is not addressable or not PFIXed, the PFREE request for this page is ignored.

When a page is freed, the PFIX counter is decreased by 1. If the counter is zero, the page frame is released and enqueued at the end of the page selection queue; any task waiting for a freed page is then posted, and the next page to be freed is selected if this page is temporarily fixable in the released page frame (NFRP bit in S370FLG of the page frame table entry is OFF).

370 mode only:

If the PFREEd page frame is reserved for another PFIX request, the address of the page frame table entry is inserted into the PFTERSVD of partition control block (PCB) and the task that issued this PFIX is posted ready to run.

SVC 69 (X'45' - REALAD)

Returns the real address for the virtual address specified in the REALAD macro. (With a supervisor with MODE=VM or in ECPS:VSE mode, the two addresses are the same.) On entry to the SVC 69 (X'45') routine, the virtual address must be contained in register 1. The real address is returned in register 0.

No address is returned if:

- The virtual address in register 1 is invalid.

- The address is within a page which is not PFIXed (does not apply to MODE=VM).

SVC 70 (X'46' - VIRTAD)

Returns the virtual address for the real address specified in the VIRTAD macro, if it is 370 mode. (With a supervisor with MODE=VM or in ECPS:VSE mode, the two addresses are the same.) On entry to the routine, the real address must be contained in register 1, and register 0 must contain zero. The virtual address is returned in register 0.

No address is returned (register 0 contains zero) if:

- The real address is contained in a page frame that is not used (does not apply to MODE=VM).

- The real address is invalid.

- The virtual address is within a page which is not fixed (does not apply to MODE=VM).

SVC 71 (X'47' - SETPFA)

Provides support for the SETPFA macro. SVC 71 (X'47') is ignored, if MODE=VM is specified. SVC 71 (X'47') establishes linkage between the supervisor and the user-written page fault appendage routine in the page fault appendage TIB located in the PCB (PHOTIB). Only one task of a partition may have an active PHO-appendage.

SVC 72 (X'48' - GETCBUF)

Only valid in 370 mode. Gets or releases a copy block used for channel program translation. The program issuing a SVC 72 (X'48') is canceled if it is not the ERP system task.

If a request for copy blocks is made, the chain of free copy blocks is searched. If the chain is not empty, a copy block is dequeued from the chain, and the address of the copy block is passed to the ERP system task. Otherwise, a value of zero is returned. A copy block is released by enqueueing it to the chain of free blocks. Any tasks waiting for copy blocks are then posted.

SVC 73 (X'49' - SETAPP)

Authorizes linkage to a channel end appendage routine for authorized programs.

SVC 74 (X'4A' - PFIXCHPT/PFIXREST)

Builds a parameter list during checkpointing or when a restart occurs, fixes pages in accordance with the parameter list, and stores the correct values in the PFIX counter located in the page frame table entry and the partition PFIX counter located in the SMCB. SVC 74 (X'4A') is ignored, if the supervisor was generated with MODE=VM specified.

The PFIX counter indicates how often a page is PFIXed, the partition PFIX counter keeps tracks of how many pages of a partition have been PFIXed. If checkpointing is requested, a parameter list is built with an entry for each PFIXed page of an affected program. The format of the parameter list is shown below.

ECPS:VSE mode:

```
┌──────────────┬────────┐
│ Address of   │ PFIX   │
│ PFIXed Page  │ Count* │
└──────────────┴────────┘
0              4        5
```

370 mode:

```
┌──────────────┬──────────────┬────────┐
│ Address of   │ Address of   │ PFIX   │
│ PFIXed Page  │ Page Frame   │ Count* │
└──────────────┴──────────────┴────────┘
0              4              8        9
```

* indicates how often the page is fixed

Figure 13. Restart-PFIX Parameter List Entry

Only tasks running with protection key 0 may issue an SVC 74 (X'4A'). Register 0 contains the length of the parameter list, and register 1 points to the parameter list. On return, register 2 contains zero, if no additional parameter list is needed, and four, if an additional parameter list is needed. A non-zero byte is placed right after the last generated entry of a parameter list.

If the restart function is requested, register 0 contains zero. Register 1 points to the parameter list built by the checkpoint function. The pages identified by the parameter list are PFIXed by calling the PFIX routine (see "SVC 67 (X'43' - PFIX)" on page 54) for each page. After the PFIXing of a page, the PFIX counter is set as indicated in the parameter list, and the partition PFIX counter is increased by 1.

370 mode only:

Since a page has to be fixed in the same page frame in which it was fixed at checkpoint time, the address of the page frame table entry belonging to the page frame in which the page should be fixed is saved in the PTFERSVD of the PCB before control is transferred to the PFIX routine.

SVC 75 (X'4B' - SECTVAL)

Calculates the sector value for a position on a track of a DASD device supporting rotational position sensing (RPS). If the supervisor was generated without RPS specified in the FOPT macro, the issuing program is canceled (ERR21). The routine calculates the position for either fixed or variable length records.

On return to the caller R0 contains the calculated sector value.

SVC 76 (X'4C')

Initiates the recording of an RMSR record on the Recorder File (SYSREC). If the system runs under VM/370, not all information in the record may be valid. VM/370 gains control to perform the recording function. When not running under VM/370, the effect of this SVC is the same as for SVC 15 (X'0F' - SYSIO).

The address of the user's CCB must be supplied in general register 1 before this SVC is issued. The data address must be supplied in general register 0. If the recorder file is on a CKD device, register 1 must have the high-order bit on to indicate that VM/370 must intercept this SVC. After having intercepted, VM/370 zeros out this register so that, on return, the issuing program can check whether VM/370 handled the I/O request.

If the recorder file is on an FBA device, the interrupt is not intercepted by VM/370.

SVC 77 (X'4D' - TRANSCSW)

Only valid in a supervisor specified with MODE=370 and for system tasks. Used by routines which print the CCW address of a failing I/O operation, such as the ERP message writer. The virtual address of a copied CCW is calculated.

On entry to the SVC 77 (X'4D') routine, register 0 contains the address of the copied CCW, and register 1 the address of the copied CCB.

The retranslated CCW address is returned in register 0 if the address passed in this register by the user points to a copied CCW related to the I/O operation being handled. If the user passed an invalid address, register 0 contains the value zero on return. The contents of register 1 are not changed.

SVC 78 (X'4E' - CHAP)

Provides support for the CHAP macro. The priority of the issuing subtask is made the lowest priority of all subtasks in that partition by modifying the TIDSTR and TSS fields. The TID of a subtask is inserted immediately before the identifier of the main task. The identifiers of the subtasks with lower priority, are moved one byte higher. The use of this SVC by the main task is ignored.

SVC 79 (X'4F')

Reserved.

SVC 80 (X'50' - SETT)

Provides support for the SETT macro, which sets the task timer. Register 1 contains the task time interval, specified in milliseconds. The highest allowable value is 21474836 milliseconds.

The time interval is converted to the appropriate units and inserted in TTTAB, the task timer table (see Figure 258 on page 537). The task timer bit (OWNTIMER in TCBFLAGS) in the task control block (TCB), is turned on. The time interval specified is decremented only when this task is executing. When the time interval has elapsed the task timer bit is turned off and the routine specified via the STXIT TT is entered. If no TT routine has been specified, program execution continues.

The SETT macro can be issued only by the main task of the partition owning the task timer.

SVC 81 (X'51' - TESTT)

Provides support for the TESTT macro. The TESTT macro is used to test how much of a task time interval set by an associated SETT macro has been left.

The time remaining in the interval, expressed in hundredths of milliseconds in binary, is returned in register 0.

If register 0 is not zero at entry to SVC 81 (X'51'), the remaining time of the interval is canceled, and the task timer bit (OWNTIMER in TCBFLAGS) in the TCB is turned off. The TESTT macro can be issued only by the main task of the partition owning the task timer (refer to Figure 258 on page 537).

SVC 82 (X'52')

This function supports the setting and resetting of monitor calls class 4 for VSE/ICCF, or transfers control immediately to that program for having services executed by VSE/ICCF.

SVC 83 (X'53' - ALLOCATE)

Allocates or reallocates real or virtual partitions when called by the macro ALLOCATE. Register 1 contains the address of the allocation parameter list; register 0 the mode value, where a contents of zero means virtual mode, a contents of non-zero REAL mode. Register 15 contains a return code after completion. For a description of the ALLOCATE macro refer to Appendix B. The allocation or reallocation process involves:

- Calling the page manager invalidation routine to flag the contents of the pages as invalid.

- Setting the storage key (ECPS:VSE mode only)

- Updating the partition or SVA limits in the appropriate entry of the Storage Management Control Block (SMCB), see Figure 101 on page 246 and Figure 102 on page 247).

For further information see "Storage Management" on page 245 described later in this chapter.

SVC 84 (X'54' - SETLIMIT)

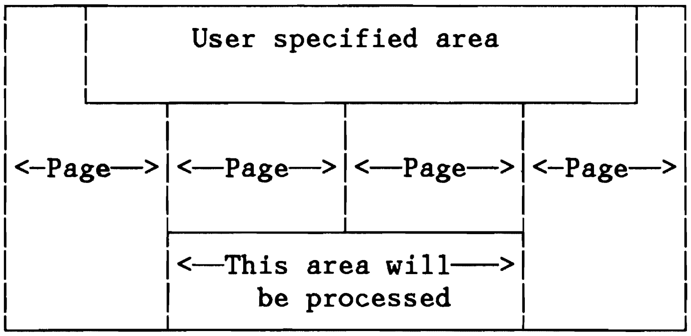
Changes partition sizes when called by the macro SETLIMIT. For a description of the SETLIMIT macro, including a description of information that is passed to and returned by the supervisor, refer to Appendix B.

For further information see "Storage Management" on page 245 described later in this chapter.

SVC 85 (X'55' - RELPAG)

Provides the support for the RELPAG macro. Part of the code is common for both, SVC 85 (X'55') and SVC 86 (X'56') (FCEPGOUT macro).

When the SVC 85 (X'55') routine receives control, register 1 points to an 8 byte parameter list containing both the begin and the length of the requested area. The begin and end addresses of the specified area hardly ever coincide with the begin and end addresses, respectively, of a page. Therefore, the area specification has to be changed to proper page addresses, as shown below, before it can be processed.



The common actions are:

- If the page is outside the address range of the requested program's partition, the return code is set to 4 (X'04').
- If the page is fixed or has an entry in the page fault queues the return code is set to 8 (X'08').
- If a negative length is detected, the return code is set to 2 (X'02').

If the list of areas that are to be handled is not completely in the requesting program's partition, the request is ignored and return code 16 (X'10') is set.

The parameter list is processed until the end of list is reached, or a page has to be handled for which none of the above three conditions are true.

The latter condition causes a return to the caller with offset 4 to allow specific actions. These actions for a RELPAG request are:

- Reset reference, change and page data set bits.
- If the page is not disconnected, disconnect the page, enqueue the associated PFTE at the top of the page selection queue and clear the page frame.

If it is a supervisor with MODE=VM specified, RELPAG just clears the page.

SVC 86 (X'56' - FCEPGOUT)

Provides support for the FCEPGOUT macro. The processing of the parameter list, pointed to by register 1, is done by the routine HANDLLST/HANDLENT (refer to "SVC 85 (X'55' - RELPAG)" on page 61). SVC 86 (X'56') is ignored, if the supervisor was generated with MODE=VM specified.

The following action is executed for the FCEPGOUT request, if the page is addressable:

- Reset the reference and the hold bit in the PFTE and enqueue the PFTE at the beginning of the PSQ.

If the page is not addressable, no specific action is taken.

SVC 87 (X'57' - PAGEIN)

Provides support for the PAGEIN macro by initiating the PGN system task.

When the SVC routine receives control, register 1 points to a list of area specifications. Each entry in that list is eight bytes long and contains:

Bytes	Description
0 – 3	Address of the area to be paged–in.
4 – 7	370 length.

Register 0 points to an ECB if an ECB address was given in the PAGEIN macro, else it contains zero.

The SVC 87 (X'57') routine ignores a user PAGEIN request if one of the following conditions exists:

- The PAGEIN macro was issued by a program running in REAL mode.
- The list of areas that are to be paged in is not completely contained in the requesting program's partition.
- The table PAGETAB (see Figure 14 on page 64) is full.
- The ECB address, if specified, is outside the requesting program's partition.

For each PAGEIN request, the SVC routine builds an 8-byte entry in a table called PAGETAB (see Figure 14 on page 64).

The entries of the table are stacked and processed (by the PGN system task) FIFO. The maximum number of table entries is specified during IPL in the PAGEIN parameter of the SYS command.

For a valid PAGEIN request, the SVC 87 (X'57') routine passes control to the PGN system task either directly or via task selection.

TID	PAGE Address	FLAG	ECB Address
0	1 3	4	5 7
• • •			• • •

Bytes	Description
0	Identifier of task that issued the PAGEIN macro.
1 - 3	Pointer to the areas to be paged-in.
4	Flag Byte: X'80' PAGEIN request completed, second scan needed. 40 Reserved. 20 At least one page is outside partition boundary. 10 At least one entry with a negative length was found. 08 Reserved. 04 Paging activity too high, termination was requested by LOAD LEVELER. 02 Task is terminating, entry has to be deleted. 01 Second scan in progress.
5 - 7	Pointer to ECB (if used) or zero.

Figure 14. Page-in Table (PAGETAB)

If the address of an ECB was specified in the PAGEIN macro, information is returned in byte 2 of that ECB as shown below:

Bits	Meaning	Set by:	
		SVC Routine	PGN Task
0	PAGEIN request completed (see Note below).	Y	Y
1	Page-in table (PAGETAB) is full.	Y	N
2	One or more of the requested pages are outside the address range of the requesting program's partition.	N	Y
3	At least one negative length has been detected in the processed area specifications.	N	Y
4	List of areas that are to be paged-in is not completely contained in the requesting program's partition.	Y	N
5	Paging activity too high. PAGEIN request terminated by LOAD LEVELER.	N	Y
6	Reserved.		
7	Reserved.		

Figure 15. Return Information in Byte 2 of PAGEIN ECB

Note: Bit 0 is set by the PGN system task if that task receives control to process the pertinent PAGEIN request, otherwise the bit is set by the SVC routine. If the supervisor was generated with MODE=VM specified, an SVC 87 (X'57') causes only the ECB to be posted, provided an ECB was specified in the PAGEIN macro and the specified ECB address is valid.

SVC 88 (X'58' - TPIN)

Provides the support for the TPIN macro. SVC 88 (X'58') is ignored and control is returned to the user if the supervisor was generated with MODE=VM specified. This SVC should always be used in combination with SVC 89 (X'59' - TPOUT). Both SVCs are intended for exclusive use by teleprocessing access methods such as ACF/VTAM and by data base/data communication interface programs such as CICS/VS. The SVCs are required for the supervisor to perform TP-Balancing.

SVC 88 (X'58') indicates to the supervisor that an immediate demand is to be made on system resources by the subsystem issuing the SVC 88 (X'58'). The SVC requests the supervisor to deactivate one or more partitions of lowest processing priority.

The demand is ignored in each of the following cases:

- The user has not requested TP Balancing via the TPBAL command.
- None of the partitions specified in the TPBAL command can be deactivated (no tasks running in virtual mode other than TPIN-issuing task).
- There are no page faults in the system.

Forced deactivation is obtained by indicating to the page manager that no page fault should be handled for the partitions. At the same time, reactivation and deactivation is prohibited by setting the TP-in-progress bit and the batch-deactivated bit in the SUPFLAG byte.

SVC 89 (X'59' - TPOUT)

Provides the support for the TPOUT macro. SVC 89 (X'59') is ignored and control is returned to the user if the supervisor was generated with MODE=VM specified. This SVC is the necessary counterpart of SVC 88 (X'58' - TPIN), it resets the TP-in-progress bit in the SUPFLAG byte and permits normal reactivation and deactivation of partitions. All partitions deactivated by the previous TPIN request are reactivated immediately.

SVC 90 (X'5A' - PUTACCT)

Provides support for the PUTACCT macro. If VSE/POWER provides account support, that program's account appendage is entered. Otherwise, the user ECB is posted (byte 2, bit 0).

SVC 91 (X'5B')

Provides interface between job control and VSE/POWER. On entry of the SVC 91 (X'5B') routine, the VSE/POWER account appendage is given control.

SVC 92 (X'5C' - XECBTAB)

Provides supervisor support for the XECBTAB macro.
SVC 92 (X'5C') adds, removes, checks, or resets one entry in the cross-partition event control block (XECB) table or deletes all entries belonging to the issuing task. Figure 16 shows the format of an entry in the XECB table.

Bytes	Description
0 - 7	XECB name
8	ACCESS Control byte
	X'80' Table entry in use
	40 Task that issued XPOST was canceled
	20 Reserved
	10 Reserved
	XWAIT access indicator
	08 ACCESS=XPOST specified
	04 ACCESS=XWAIT specified
	XPOST access indicator
	02 ACCESS=XWAIT specified
	01 ACCESS=XPOST specified
9 - 11	XECB address
12 - 13	TID of owner
14 - 15	TID of first task that posted XWAIT or XPOST for XECB
16 - 19	Forward chain pointer
20 - 23	Backward chain pointer

Figure 16. Format of XECB Table Entry

If bits 4 and 5 of byte 8 are set to 10, bytes 14 and 15 contain the TID of the first task that issued an XWAIT for this XECB. If bits 6 and 7 of byte 8 are set to 10, bytes 14 and 15 contain the TID of the first task that issued an XPOST for this XECB. Label XECBTAB identifies the first byte of the table.

Whenever SVC 92 (X'5C') is invoked, register 1 must point to a user parameter list describing the XECB. The length of the parameter list is 12 bytes for TYPE=DEFINE and 10 bytes for the other type options.

XECB name	Flags	XECB address
0	8	9 11

Bytes	Description
0 - 7	XECB name
8	Flag byte X'80' Reserved 40 Reserved 20 Reserved 10 Reserved XWAIT access indicator 08 ACCESS=XPOST specified 04 ACCESS=XWAIT specified XPOST access indicator 02 ACCESS=XWAIT specified 01 ACCESS=XPOST specified
9 - 11	XECB address

Figure 17. Parameter List for TYPE=DEFINE

XECB name	Flags	
0	8	9

Bytes	Description
0 - 7	XECB name
8	Flag byte X'80' TYPE=CHECK 40 TYPE=DELETE 20 Reserved 10 TYPE=RESET 08 DELETALL Request 04 Reserved 02 Reserved 01 Reserved
9	Reserved

Figure 18. Parameter List for TYPE=DELETE, DELETALL, RESET or CHECK

Actions for TYPE=DEFINE

The XECB table is scanned for a possible duplicate entry. If none is found, an empty slot is identified and the user parameters are moved to it.

On successful completion of the operation, the XECB address is returned in general register 1, and the address of the table entry being handled is returned in general register 14.

One of the following codes is returned in register 15 after an XECBTAB TYPE=DEFINE has been issued:

X'00'	DEFINE function completed successfully.
X'04'	XECB already defined in the XECB table.
X'08'	No GETVIS storage available.

Actions for TYPE=RESET

The table is scanned for the specified XECB name. If the name is found, a check is made to see whether the issuing task owns the XECB. If it does, the waiting task (if any) is posted ready-to-run, the XECB table entry is cleared and chained to the free-chain. On successful completion of the operation, registers 1 and 14 are set to zero.

One of the following codes is returned in register 15 after an XECBTAB TYPE=RESET has been issued:

X'00'	RESET function completed successfully.
X'04'	XECB not found in the XECB table.
X'08'	Issuing task does not own the XECB.

Actions for TYPE=DELETE

The table is scanned for the XECB name specified. If the name is found, a check is made to see if the issuing task owns the XECB. If it does, the waiting task (if any) is posted ready-to-run, the XECB table entry is cleared and the use count is decreased by one. On successful completion of the operation, registers 1 and 14 are set to zero.

One of the following codes is returned in register 15 after an XECBTAB TYPE=DELETE has been issued:

X'00'	DELETE function completed successfully.
X'04'	XECB not found in the XECB table.
X'08'	Issuing task does not own the XECB.

Actions for TYPE=CHECK

The table is scanned for the XECB name specified. Depending on the result of the scan, one of the following codes is returned in general register 15:

X'00'	XECB name found in the table. General register 1 contains the address of the XECB; general register 14 contains the address of the table entry.
X'04'	XECB name not found in table. General registers 1 and 14 are set to zero.

Actions for TYPE=DELETALL

The XECB table is scanned for all entries which are owned by the issuing task or which communicate with the issuing task.

Any XECB table entry which is owned by the issuing task is deleted from the XECB table. The waiting task (if any) is posted ready-to-run.

An XECB table entry which contains the issuing task's ID in its communication field has the ID field (bytes 12 and 13) cleared to zero. When the owning task is a waiting task (entry defined with ACCESS=XWAIT), it is posted ready-to-run, and an abnormal termination flag is set in the XECB table entry and in the XECB (byte 2, bit 1).

On return from SVC X'5C' (with TYPE=DELETALL) the contents of the general registers 14 and 15 are not changed.

A common subroutine, FNDLOOP, is used to scan the XECB table for a specified XECB name.

SVC 93 (X'5D' - XPOST)

Provides supervisor support for the XPOST macro. The routine posts a specified XECB and marks the task (if any) waiting for this XECB as ready.

Every time SVC 93 (X'5D') is invoked, register 1 must point to a field in the issuing partition that contains the XECB name;

register 14 must contain the table entry address. The XECB table (see Figure 16 on page 67) is scanned for the specified name by the subroutine FNDLOOP. If the entry is found, a check is made to see if the task is authorized to XPOST this XECB, as indicated in the table entry. Subroutine XECCHK is used to make this check.

If the task is found to be authorized, the traffic bit in the XECB is set and the entry is examined to see if a task is waiting on this XECB. If there is a task, it is posted ready-to-run again.

Return codes in register 15:

0 (X'00')	Successful completion.
4 (X'04')	XECB name not found in XECB table.
13 (X'0D') and 14 (X'0E')	Task not authorized to issue XPOST. The return code is made up of 12 (X'0C') plus, in the rightmost two bits, the XPOST access code that was contained in the table entry (see SVC 92 (X'5C') "SVC 92 (X'5C' - XECBTAB)" on page 67 for an explanation of the access code).

SVC 94 (X'5E' - XWAIT)

Provides supervisor support for the XWAIT macro. SVC X'5E' checks to see if the specified XECB has been posted. If not, the issuing task is marked waiting and its TID is entered into the table entry.

Whenever SVC 94 (X'5E') is invoked, register 1 must point to a field that contains the XECB name. Register 14 must contain the address of the table entry.

If register 14 does not point to the correct table entry, the correct entry is found through the XECB name pointed to by register 1. For this purpose, subroutine FNDLOOP is used. Authorization of the task is tested by means of subroutine XECCHK.

When the entry is found, the traffic bit in the XECB is tested. If it is off, the task status is set to "WAITBND (X'82')", and the waiting task's TID is entered into the table entry. (The task's continuation address is lowered by two bytes for re-SVC when the task gets selected.)

If the XECB was already posted, an immediate exit is taken to task selection and register 15 contains a return code of X'00', while registers 1 and 14 are set to zero. If the abnormal termination flag is posted in the XECB table entry, that is, the communicating task has broken communication without XPOSTing the waiting task, the communicating task gets control with a return code of X'08'.

Return codes in register 15:

0 (X'00') Successful completion, the XECB has been posted

4 (X'04') XECB name not found in table

8 (X'08') Communication with the other task using this XECB was broken. The other task issued an SVC 92 (X'5C') with TYPE=DELETALL.

13 (X'0D') Task not authorized to issue XWAIT. The return code is made up of 12 (X'0C') plus, in the rightmost two bits, the XPOST access code that is contained in the table entry (see the description of SVC 92 (X'5C') for an explanation of the access code).

SVC 95 (X'5F' - EXIT AB)

This supervisor call is invoked by an EXIT AB macro; it provides for a return from the user's abnormal termination routine to the supervisor to reset the cancel condition and ABEND indication in the PIB table after the error condition has been cleared up by the Exit routine. Control is then returned to the user program and processing continues with the instruction following the EXIT AB macro.

Before the abnormal termination routine is entered, the linkages to the OC, IT, AB, and PC routines are invalidated and the logical transient area (LTA) is released. If the routine ends with SVC 95 (X'5F') the linkage to those routines is reestablished.

If an IT interrupt specified by the SETIME macro occurs during the processing of the user's abnormal termination routine, the interrupt is delayed until processing ends with an EXIT AB macro.

> **Note:** If an interrupt is delayed, the TTIMER macro returns a value of zero in register 0. Therefore, a contents of zero indicates that either no time interval has been set or a time interval has elapsed during abnormal termination processing and handling of the timer interrupt is suspended until processing is completed.

SVC 96 (X'60' - EXIT TT)

Provides supervisor support for the EXIT TT macro. It provides a return from the user's task timer Exit routine to the program that was interrupted by the expiration of the task timer interval. The user-supplied save area is restored to the problem program save area.

This SVC may be issued only by the main task of the partition which owns the task timer support.

SVC 97 (X'61' - STXIT TT)

Provides supervisor support for the STXIT TT macro. It establishes linkage from the supervisor to a task timer Exit routine in a problem program. It stores the address of the Exit routine, the caller's PSW key and the address of the save area in the task timer table. The save area is a 72-byte area in which the interrupt information is stored. See Figure 258 on page 537 for the task timer table and Figure 239 on page 519 for the save area format. The issuing program is canceled if the supervisor is generated without the support or if the SVC is issued from a task other than the main task of the partition which owns the task timer.

SVC 98 (X'62' - EXTRACT/MODCTB)

Provides supervisor support for the macros EXTRACT and MODCTB. For a description of the macros refer to
Appendix B.
The EXTRACT macro can retrieve and supply the following information:

- Partition boundaries from the Storage Management Control Block (SMCB)
- Unit information from the PUB table
- Control registers
- PUB2 table entries
- Device information as retrieved by a 'SENSE-ID' command.
- CPU ID and SYSLOG ID

The MODCTB macro is used to change PUB2 table entries. The SVC 98 (X'62') routine expects that register 1 points to a parameter list. Register 15 contains a return code after completion (refer to Appendix B).

Bytes	Description
0	Identification Field EXTRACT: X'01' PUB2 specified 03 BDY specified 04 CR specified 05 PUB specified 06 CPUID specified 07 MAP specified 08 DEVICE specified MODCTB: F0 PUB2 specified
1	Flag byte X'00' SEL specified 01 SEP specified 02 BDY and MODE=S or DEVICE and PU specified 03 BDY and MODE=P specified
2 – 3	Reserved
4 – 5	Length of communication area as specified by the user.
6 – 7	Displacement in PUB2 or PUB table entry; retrieval of information begins at this point.
8 – 11	Address of communication area as specified by the user.
12 – 15	Address of two bytes identifying the device. They either identify a logical unit in the CCB format (if 'SEL' was specified) or contain the physical device address (if 'SEP' was specified) or the PUB index (if 'PU' was specified).
16 – 19	Address of PIK.

Figure 19. Format of the SVC 98 (X'62) Parameter List

SVC 99 (X'63' - GETVCE)

Provides the supervisor service for the GETVCE macro. It passes DASD specific information back to the user. The following table describes the layout of the parameter list the address of which is contained in general register 1.

Bytes	Description
0 - 3	Address of output area
4 - 7	Address of device identifier
8	Length of output area - 1
9	Device identification code: X'03' DEVTYPE code is given 1-byte field containing PUB device-type code 02 CUU code is given 2-byte field containing channel+device address 01 LNO code is given 2-byte field in CCB logical unit number format 00 VOLID is given a 6-byte field containing the volume serial number
10	Device type code or zero
11	Macro version identifier
12 - 13	Physical unit number (cuu) or zero
14 - 15	Data length
16	Key length
17	Record number
18	Reserved (must be zero)
19	Processing flags
20 - 21	Remaining track balance or zero if not known

Figure 20. Format of the SVC 99 (X'63) Parameter List

For the output format and the return codes refer to the GETVCE macro (Appendix B).

SVC 100 (X'64' - PFIX/PFREE)

Only valid in ECPS:VSE mode. Supports the PFIX/PFREE macro to fix or to free a page in the system GETVIS area. The caller must have a storage protection key of 0 and run in the supervisor state.

If register 0 contains 0, PFIX is requested; if it contains 4, PFREE is requested. Register 1 points to the list of pages to be handled. The same routines are used as for the PFIX and PFREE requests using SVC 67 (X'43') or SVC 68 (X'44').

SVC 101 (X'65' - MODVCE)

The DASD device specified by the logical unit or device address in the MODVCE macro is interrogated for status, volume and device characteristics. The volume characteristics table (VCT) is updated, if necessary. It should be used whenever a program changes the VOLID of a DASD or when the information given back by the GETVCE macro may not reflect the current status. Refer to the description of the MODVCE macro in Appendix B and to "Automatic Volume Recognition (AVR)" on page 122.

SVC 102 (X'66' - GETJA)

Provides supervisor support for the macro GETJA. For the format of the GETJA macro, refer to Appendix B.

The supervisor service differentiates between a GETJA issued by by job control and a GETJA issued by any other task. The GETJA macro if issued by job control requires general register 0 to contain the Function code, which has been defined as described below:

Function code:

0	UPDATE	Update all account information, maintained by the supervisor
1	CLRTIME	Reset JOB related information to zero.
2	RESET	Reset JOB-STEP related information.

The service, if requested by any user other than job control, always forces the account information, maintained by the supervisor to be updated regardless of the contents of register 0.

The time counters (ACCTCPUT, ACCTOVHT and ACCTBNDT) in the job accounting interface partition tables (ACCTxx) are replaced with the most current CPU time, OVERHEAD time and ALLBOUND time.
The OVERHEAD time is distributed in proportion to the CPU time accumulated for each of the active partitions.
The ALLBOUND time is distributed in equal parts among the active partitions.

SVC 103 (X'67')

Performs input/output operations for SYSFIL on FBA devices. Register 1 contains the address of the CCB/IORB. The code consists of:

- A resident part, within the supervisor
- A pageable part, residing in the SVA (Module $IJBFBA).

The resident part contains the following supervisor functions:

- Check device type
- Validate I/O area address
- Gate SVC 103 (X'67') against simultaneous use of a disk information block (DIB).
- Establish an interface to the SVC 0 routine, the I/O interrupt handler, the dispatcher, and their appropriate supervisor subroutines.

The pageable part contains the following data management functions:

- Check CCB/IORB contents
- Initialize data buffers (CIDF, RDF)
- Perform blocking and deblocking between user's I/O area and data buffer
- Supply CCB/IORB return information

SVC 104 (X'68' - EXTENT)

Serves as DASD file protect interface for adding, returning or deleting extent information. For more details, refer also to the description of the EXTENT macro in Appendix B.

SVC 105 (X'69' - SUBSID)

Provides supervisor support for an execution time subsystem identification. The support consists of three functions, defined by a value passed in register 0. For more details, refer to the description of the SUBSID macro in Appendix B.

SVC 106 (X'6A')

The area specified by registers 1 and 2 is invalidated and the storage protection key of this area is set to the value provided in register 0. The registers contain the following operands:

R0 The storage protection key to be used by the SSK instruction. The storage protection key has to be outside the range of the storage keys for the partitions of the system and must be unequal zero.
R1 Begin address (in first page) of the area to be invalidated. The high order byte must be zero.
R2 End address (in last page) of the area to be invalidated.

This function can only be used by OCCF or a task running in an ICCF partition and having a storage protection key of zero.

SVC 107 (X'6B' - Fast SVC)

The "FASTSVC = SVC 107 (X'6B')" has been established to provide retrieval and modifications of fields which are only known to the supervisor. This SVC covers a lot of services and whenever possible, runs over a "Fast path" (mainly bypasses GENERAL ENTRY [GENENT] and GENERAL EXIT Routines) and returns control to the caller without redispatching. Special services can therefore be invoked within other SVC routines and within supervisor appendages.

The FASTSVC functions can be included in reentrant code without special provisions, since all parameters are passed in registers. If any input parameter is invalid, the request owner is canceled.

These functions are provided by various macros which set up different Function codes (see below).

Macro	**Function**
GETFLD	Get a task-related field
MODFLD	Modify a task-related field
TREADY	Post or cancel a task
TSTOP	Deactivate the current task or partition
RLOCK	Obtain access to a specified resource or wait for it
SRCHFLD	Retrieve PUB index
DEVUSE	Force a device to be set "in use"
SENTER	Enter a Sub-system
SLEAVE	Leave a Sub-system
DEVREL	Release a device that was "in use"
VIOPOINT	Point to VIO control block (VIORB)
VALID	Validate a specified area
GETJA	Update Job Accounting information

Note: For a detailed description see Appendix B.

Any of the FASTSVC services belongs to one of the following three classes:

Class	**Usage**
A	Unrestricted usage. The function can be invoked in any variation from any type of code. The fast path is always taken.
B	Task related usage. A request other than from problem program code can refer in the TASK parameter only to the current task. Requests from appendages related to asynchronous events like I/O interrupts are not allowed, since no current task is defined in this case. The fast path is taken only for requests referring to the current task.
C	Problem program usage. The function can only be invoked from problem program code. The fast path is never taken.

Note: System task code is considered as problem program code.

The class is given for each function (see SVC X'6B'). If an invalid request for a class B or C function is issued, the request owner (i.e. the current task or the owner of the asynchronous event) is canceled.

Note: "SVC 107 (X'6B') Function Codes" on page 613.

SVC 108 (X'6C' - SECHECK)

The SVC X'6C' routine checks whether a user is allowed to access a specified resource.

A pointer to the resource name is given through an access control authorization parameter list (DTSAPL). The address of this DTSAPL must be supplied in register 1. The access control authorization checking routine, loaded into the SVA during IPL, is entered to check whether access is allowed or not.

Return codes:
Register 15 will pass back one of the following return codes.

0 (X'00')	Access allowed.
4 (X'04')	Access control facility not supported.
8 (X'08')	Access control violation.
12 (X'0C')	Resource name not in access control resource table (DTSECTAB), which is only possible for sublibrary and member.

Access Control Authorization Checking Routine

The access control authorization checking routine runs as an SVC appendage to the SVC 108 (X'6C') routine. The routine is re-entrant and loaded into the SVA during IPL after a pointer to that routine has been initialized in the supervisor security vector table (SCYVECTB), addressed by means of SYSCOM.

This routine performs the actual access control check for the resource referenced in the DTSAPL against the access control resource table, which is initialized by the user and contains the various resources to be protected (such as libraries, sublibraries, members and files).

Based on the DTSAPL, access control authorization checking takes place either for a library, a sublibrary, a member or a file:

- A library request originates from a VSE system component such as the librarian or JCL. This component has to determine whether a system library is protected or not. The component issues an OPEN for a DTF with a DTSAPL addressed by the DTF. The $$B-transient OPEN phase passes control to the access control

OPEN appendage routine which builds the DTSAPL for the library to be validated before issuing the SVC 108 (X'6C').

- A sublibrary request originating from a VSE system component such as the librarian. This component has to determine whether a sublibrary is protected or not.

- A member request originating from a VSE system component such as the librarian. This component has to determine whether a member of a sublibrary is protected or not.

- A member request as the result of a FETCH/LOAD/CDLOAD request which is intercepted by the supervisor for access control authorization checking. A direct branch is made into the access control SVC processing routine. This routine prepares the DTSAPL for the phase to be loaded and then comes to this SVC appendage routine to perform the actual validation.

- A file request originates from any OPEN request for a file on a DASD volume or on magnetic tape. As stated above, the $$B-transient OPEN phase transfers control to the OPEN appendage routine which completes the DTSAPL for processing by the SVC 108 (X'6C') routine.

The access control authorization checking routine uses main input parameters to do the access control validation:

DTSJPL Addressed by the COMREG and initialized by job control from the user profile record (macro with DSECT available).

DTSAPL Contains the resource or object to be validated against the access control resource table (DTSECTAB) for the user defined by the DTSJPL (macro with DSECT available).

DTSECTAB This is the access control resource table which contains all resources to be protected plus the corresponding access classes and logging options.

The checking routine locates the object defined by the DTSAPL in the DTSECTAB; it checks the access classes and access rights stored in the DTSJPL against those given in the matching table entry. If the user has any of the allowed access classes assigned via the profile record and at least the requested access right, (transferred to the DTSJPL by job control), then the user has passed the access control authorization check. An access class of zero means that no secured resources can be accessed.

If the request was to check if a user has the authorization to catalog a $$B-transient, then the DTSJPL field JPSA is checked for this special authority.

If the user is found to be unauthorized, the request is canceled after logging the request to the log data set (if VSE Access Control PP is installed).

The logging options in the DTSECTAB are set per class. Every access to this resource is logged if specified; violations are always logged. Determination depends on the access classes of the user and those in the table entry and the corresponding logging classes. The data for the required logging record is moved directly into the logging queue by means of the VSE-Q-manager. The queue entries are written to the log data set by means of the logger system task.

Return codes are set accordingly in register 15 to be handled by the access control check SVC:

0 (X'00') Normal return with no logging:
The user has authority for the resource.

4 (X'04') Post logging task and continue normal operation:
The user has authority for the resource but the access request must be logged to the log data set.

8 (X'08') Lost logging task and cancel:
The user is not allowed to access the resource. He is canceled with cancel code 11 (X'0B') and the request is logged to the log data set.

12 (X'0C') Cancel due to inactive logger:
The user is not allowed to access the resource. He is canceled with cancel code 11 (X'0B') but the request can not be logged to the log data set because VSE Access Control PP is not installed.

16 (X'10') 'Log-queue-full' wait condition:
The caller is set to RESVC.

20 (X'14') Cancel due to authorization routine processing error:
Cancel code is 10 (X'0A').

Detailed information about the return code is contained in the DTSAPL fields APJCL, APERR, APOAT, APUAR and APSPR. For the description of the content of these fields see macro DTSAPL.

SVC 109 (X'6D' - PAGESTAT)

Returns the status of an area as requested by the PAGESTAT macro. When the SVC 109 (X'6D') routine is entered, register 0 contains the begin address and register 1 the end address of the area. On return, byte 0 of register 15 contains the status of the actual first page of the area. Bytes 1 through 3 of register 15 contain the address of the first page of the area with a different status.

For the format of PAGESTAT macro and status settings refer to Appendix B: Macro Descriptions.

If the status in register 15 indicates 'address is invalid', invalid address means that a reference to this address forces the task to be canceled due to invalid address, i.e.:

- Address beyond virtual storage or
- in ECPS:VSE mode: page belongs to a partition in REAL mode and page is not addressable
- in 370 mode: HABIT (bit 10) and IBIT (bit 12) are on in corresponding page table entry.

Invocation of this SVC with beginaddr on a higher page than endaddr results in 'canceled due to invalid address'.

SVC 110 (X'6E' - LOCK/UNLOCK)

Lock Manager (LOCK and UNLOCK)

Locks a resource against simultaneous use by other tasks.
Unlocks a given resource that was previously locked.
The SVC is invoked by the LOCK and UNLOCK macros.

For more information refer to "Lock Management" on page 137.

SVC 111 (X'6F')

Reserved.

SVC 112 (X'70' - MSAT)

Manipulates assignment and device ownership information. For details of the external specification, refer to the description of the MSAT macro in Appendix B.
Input is a parameter list in Reg.1, containing an identification of the required subfunction. The subfunctions can be group together as follows:

1. Retrieve assignment information for one or more logical units in a given partition (ID=INQ,CKU,RTL,RTP).

2. Modify the assignment information for one or more logical units in a given partition (ID=PER,DEL,ALP,ALT,NXT,RSU,RSA,NPM,NTM,DRL).

3. Modify the status of a unit record device in a given partition relative to spooling by VSE/POWER (ID=PST,PSP).

4. Modify the status of a device relative to physical addressing access (i.e. without a logical unit) by a system function or by an authorized component.

The following data areas are accessed (see also the overview Figure 270 on page 550):

LUB	To retrieve/modify the current assignment of a logical unit in a given partition.
LUB Extension	To indicate what type of assignments of a logical unit in a given partition are stored (in addition to the current) and to store the permanent assignment or set up a pointer to a chain of SAT (Stored Assignment Table) entries.
SAT Entry	To store up to 5 permanent or permanent alternate or temporary alternate assignments of a logical unit in a given partition, together with control information.
PUB	To control the status of a device (up or down) and to retrieve the device type code.
PUB Extension	To maintain ownership and usage counters of a non-sharable device (e.g. tape and TP device).
Device Usage Field	To maintain ownership and usage counters of a partition-sharable device for the system or for a given partition.
PUB Ownership Entry	To maintain system/partition ownership indicators of a device (for compatibility only).

SVC 113 (X'71' - XPCC)

Provides cross-partition communication control (XPCC), as requested by macros XPCC, XPCCB, and MAPXPCCB. For descriptions of the macros refer to Appendix B.

The cross-partition communication control facility enables the various VSE subsystems to communicate with each other or with their user applications. The support provides supervisor service for the following functions:

- Identify:
  The VSE subsystem or user application identifies itself to the XPCC.

- Termination control:
  Removes information about the application from XPCC. Application may no longer use XPCC services (except with a new IDENT). Or, depending on the parameter specified, only the existing connections can be used, no new connection can be built.

- Connect:
  Establishes a connection between two subsystems or a subsystem and an application. The connection is completed and data transfer can start when both sides have issued their CONNECT. The connection is related to the corresponding applications.

- Disconnect:
  Breaks one specific connection or all connections established for an application.

- Transmit data without reply:
  With SEND the other side of a connection is posted, enabling it to receive data. With RECEIVE, the data is moved into the receiver's input buffers.

- Transmit data with reply:
  Same as SEND/RECEIVE, but with REPLY, the receiver can immediately transfer data into a predefined buffer in the sender's partition.

- Clear:
  Allows the sender of a message to withdraw that message before the receiver has issued a RECEIVE for picking up the message.

- Purge:
  Allows a message to be purged, from the receiving side, indicating to the sender that it is not able to receive this message.

Two control blocks are used by XPCC to control data transmission between partitions. The anchor to the identification control blocks (IDCB) is in the XPCC code. All IDCBs are in one chain. For the layout of the IDCB refer to Figure 265 on page 543.

For each CONNECT request a CRCB is built which contains all relevant information from both sides of a connection. Each CRCB is a member of two different CRCB chains. The CRCB chain is pointed to by a field in the IDCB.

The CRCB consists of 3 parts.

- Part 1 contains information, common to both partners.

- Part 2 contains information, describing the partner which issued connect first.

- Part 3 contains the same information as Part 2, but for the other partner.

For the layout of the CRCB refer to Figure 266 on page 544.

SVC 114 (X'72' - VIO)

Supports the allocation, extension and deallocation of VIO files. For details of the external specifications, refer to the description of the VIO macro in the Appendix B. VIO data areas are described in "Chapter 4: Data Area Information" on page 477. The selected function is identified by a function code supplied in register 15.

The following functions are available:

- Allocate VIO file
- Extend VIO file
- Deallocate VIO file

Allocate (Function code = 0 - VIO OPEN)

Input is the address of a parameter list in register 1 and, optionally, a size specification in register 0. If register 0 contains zero, the size specification is taken from the parameter list (see Figure 261 on page 540). The requestor is canceled if the address in register 1 is invalid (ERR25) or any specified parameter is invalid (ERR21).

Space allocation is based on a byte string. Each byte corresponds to a VIO segment and contains X'00' for a free segment and X'FF' for an allocated segment. A number of not necessarily contiguous VIO segments sufficient for the requested size is allocated. Furthermore, a VIOTAB entry (Figure 262 on page 540) and one or more File Segment Tables are allocated in the system GETVIS area. For a successful allocation, the corresponding Block Tables entries (Figure 264 on page 542) and several fields in the VIOTAB are initialized. The VIOTAB address is returned in register 1 and the return code in register 15 is set to 0.
For an unsuccessful allocation (not enough VIO or system GETVIS space), intermediate allocations are freed up and the return code in register 15 is set to 8.

Extend (Function code = 1 - VIO EXTND)

Input is a VIOTAB address in register 1 and a size increment in register 0. The requesting task is cancelled if register 1 does not point to a VIOTAB (ERR25) or if it is not the owner of the VIO file described by the VIOTAB (ERR21).

A number of additional VIO segments sufficient for the requested size increment is allocated. If necessary, additional file segment tables are allocated in the system GETVIS area. For a successful allocation, the corresponding block tables entries are initialized, field VIORBASZ in the VIOTAB is adjusted to the new total size of the VIO file and the return code in register 15 is set to 0.
For an unsuccessful allocation (not enough VIO or system GETVIS space), intermediate allocations are freed up and the return code in register 15 is set to 8.

Deallocate (Function code = 2 - VIO CLOSE)

Input is an option indicator in register 0 and, if the indicator is zero, a VIOTAB address in register 1. The requesting task is cancelled if the indicator is non-zero, except for EOT or JC, (ERR21) or if the VIOTAB address is invalid (ERR25) or if it is not the owner of the VIO file described by the VIOTAB (ERR21).
If the option indicator is non-zero, all VIO files owned by the requesting task and having the life-time of a job-step (indicator = X'08') or of a job (indicator = X'10') are deallocated. If the indicator is zero, only the specified VIO file is deallocated. All allocated VIO segments and all associated system GETVIS space is freed.
The return code in register 15 is always set to 0.

SVC 115 (X'73' - PWROFF)

The SVC 115 (X'73') allows authorized subsystems to power-off a 4361 CPU via the DIAGNOSE X'80' interface through a supervisor service.

This function is currently authorized to SSX.

SVC 116 (X'74' - NPGR)

Allocates or reallocates the programmer LUBs of the specified partition(s) when called by the macro NPGR.

Register 1 contains the address of the NPGR parameter list. Register 15 contains a return code after completion. For a description of the NPGR macro refer to Appendix B.
When called by JCL via the NPGR macro, the SVC 116 (X'74') routine takes the specified NPGR values, performs some checks (see return codes) and transfers these values into the corresponding PIB(s).
When starting a partition the first time after IPL, the PIB values are taken and the corresponding LUB Table is allocated for that partition within the main LUB Table pool, which was statically reserved at supervisor generation time via the NPGR parameter.

SVC 117 (X'75')

Reserved.

SVC 118 (X'76' - CPCOM)

The SVC 118 (X'76') allows authorized subsystems to submit CP commands via the DIAGNOSE X'08' interface through a supervisor service. The command is passed unchanged to CP and the completion code is returned to the caller. The retrieval of information from CP is not supported.

The function is to be considered as a generalization of the current CPCLOSE macro (SVC 56 - X'38') and is currently authorized to VSE/POWER and FTP.

For the description of the CPCOM macro format see "CPCOM" on page 645.

SVC 119 - 140 (X'77'-X'8C')

Reserved.

SVC 141 (X'8D' - VSIUCV)

Subsystem support for VM/VCNA (VTAM Communication Network Application).

The SVC X'8D' is used by VM/VCNA to establish or end communication with the subsystem support, which in the listings is also referred to as VSE/Advanced Functions IUCV. The support is available in a supervisor that is generated with MODE=VM specified in the SUPVR macro.

The SVC 141 (X'8D') performs the following functions:

SSTE Give VM/VCNA supervisor state (only VM/VCNA is authorized).

OPEN Inform VSE/Advanced Functions that the corresponding application is a potential VSE/Advanced Functions IUCV user.

CONN Establish a connection between the application and another user of VM/System Product IUCV via VSE/Advanced Functions.

CLOS Stop usage of IUCV by a user and delete all connections related to this application.

SEVR Delete a connection between the application and another user of VM/System Product IUCV via VSE/Advanced Functions.

ACPT Accept a connection issued by another user of VM/System Product IUCV via VSE/Advanced Functions and dedicated to the application.

The handling of all IUCV related events, SVCs as well as external interrupts, are managed by means of the Application and Path ID Tables (see Figure 21 on page 88 and Figure 22 on page 88).

IUCV Application ID Table Entry (DSAIDENT)

DEC	HEX	Label	Description
0- 7	0- 7	DSAIDNME	Application ID name
8-11	8- B	DSAIDEXT	Exit address for application
12-15	C- F	DSAIDTIB	TIB pointer of exit owner
16-17	10-11	DSAIDPIK	PIK of application
18-19	12-13		Reserved

Figure 21. Formats of IUCV Application ID Table Entry

IUCV Path ID Table Entry (DSPIDENT)

DEC	HEX	Label	Description
0- 1	0- 1	DSPIDID	Path ID
2	2		Reserved
3	3	DSPIDSW	Path ID table entry switch
		PTHINACT	X'00' Path is inactive
		PTHACTVE	X'04' Path is active
		PTHCCTIS	X'08' CONNECT issued for this path
		PTHCCTRV	X'0C' CONNECT received for this path
4- 7	4- 7	DSPIDAIP	Address of application ID table entry
8-11	8- B	DSPIDDAT	Data passed to Exit routine
12-19	C-13	DSPIDTGT	Target name

Figure 22. Format of IUCV Path ID Table Entry

SVC 142 - 255 (X'8E'-X'FF')

Reserved.

DISPATCHER

COMPARISON SYSTEM TASK / USER TASK

For better understanding of system tasks processing it is important to distinguish between server and service owner.

1. Normally a system task is performing the service which has been requested by a user task.
   In this case

   - the system task is the server and
   - the user task is the requester and owner of the service.

2. A system task may request participation of another system task on the same service.
   In this case, the service owner will remain the original service requester whereas the first system task will be the immediate service requester to the second one.

3. System tasks (e.g. attention task) may perform processing which is not connected to any kind of user task.
   In this case (similar to user task) a system task is server and service owner of its own.

SYSTEM PARTITION

In order to allow system and user task selection by the same mechanism identical control blocks are used with both kinds of tasks. In addition to the user partitions a pseudo partition (system partition) is used, which is the home of all system tasks including attention. Task selection differentiates between two control blocks which are related to partitions. These are

- Partition Control Block (PCB)
- Partition Information Block (PIB)

A PCB represents a partition as the server whereas the PIB is representing the service owner. This means that in case of system task processing the system PCB is involved in a combination with a user PIB whereas in the case of user task processing the PCB and the PIB belong to the same user partition.

TASK SELECTION

Task selection is performed in six main steps:

1. Initialize task selection.

2. Scan for the highest priority partition which has at least one task ready to run.

3. Determine the highest priority ready-to-run task within the above partition.

4. Establish the connections to the control blocks involved in the task processing.

5. Perform all the supervisor services that have to be processed prior to the task's normal processing.

6. Initialize the task's processing and give control to it.

These steps are controlled by different flags, fields and control blocks in the supervisor.

In the following, the steps of task selection are described, including a description of the involved flags, fields, etc.

1. Initialization

At the very beginning task selection identifies itself setting up the Routine Identifier (RID) field by the value DISPID. This indicates that no task processing is active and is used to prevent status saving in case of any following interruption.

2. Determine the Highest Priority Ready-to-Run Partition

The status of a partition (ready to run or not ready to run) is given by its status bit in the partition selection string (PSS), which is located in low core, as shown in Figure 23 on page 91.

```
PSS

 0   1   2          n-1 n
┌───┬───┬───┬─     ─┬───┬───┐
│ s │ s │ s │  •••  │ s │ s │          bit string
└───┴───┴───┴─     ─┴───┴───┘
 SYS P1  P2         Pn-1 Pn

|———> priority order,high to low

Where:
          n = number of partitions
  s equal 0 = no task of the partition is ready to run
  s equal 1 = at least one task of the partition is ready to run
  SYS,P1,P2,...,Pn = partition priorities
```

Figure 23. Partition Selection String (Label PSS in the Supervisor)

Figure 24 shows the layout of PPRTYOWN, the partition priority owner table (pointer to PPRTYOWN located in low core). It is an extension to the partition selection string and consists of PCB pointers stored in descending partition priority order.

```
PPRTYOWN

 0       4       8                    4*n
┌───────┬───────┬───────┬─     ─┬───────┬───────┐
│ pcbptr│ pcbptr│ pcbptr│  •••  │ pcbptr│ pcbptr│   fullword pointers
└───────┴───────┴───────┴─     ─┴───────┴───────┘
 SYS     P1      P2              Pn-1    Pn

|———> priority order,high to low

  pcbptr = pointer to partition control block of priority owner
  SYS,P1,P2,...,Pn = partition priorities
```

Figure 24. Partition Priority Owner Table (PPRTYOWN)

The selection of the ready to run partition with the highest priority can be done by a left-to-right scan of PSS up to the first non-zero bit. Order number of this bit within PSS multiplied by four gives the displacement into PPRTYOWN table.

Having the PCB pointer to the selected partition, step 3 of task selection can be performed.

> **Note:** When no system task or user task is ready to run (all bits of the PSS are zeros) the dispatcher branches to the allbound routine to perform allbound time processing and then enter the allbound wait state.

Example:

with the default settings of a 12-partition system as generated at SYSGEN time, the BG will be selected as shown in Figure 25:

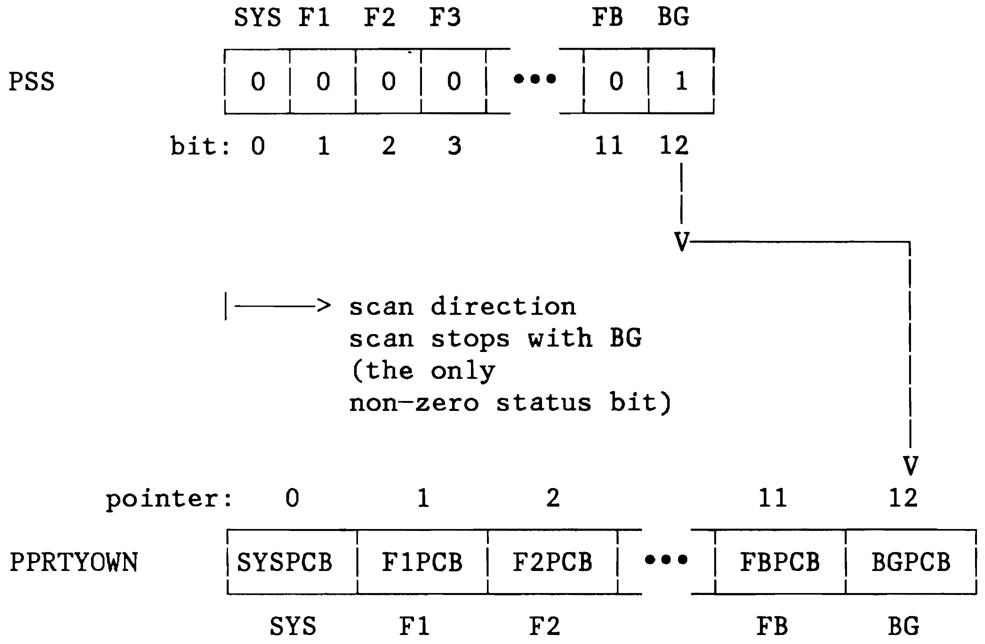


Figure 25. Selection of a Ready-to-Run Partition

As a result of the scan the order number of the first significant bit in the PSS can be calculated. In our example this order number is 12,it provides the displacement to pointer 12 of the PPRTYOWN table.

3. Determine the 'Ready' Partition Task with the Highest Priority

The status of a partition's task is given by its status bit in the Task Selection String (TSS), the layout of which is shown in Figure 26.

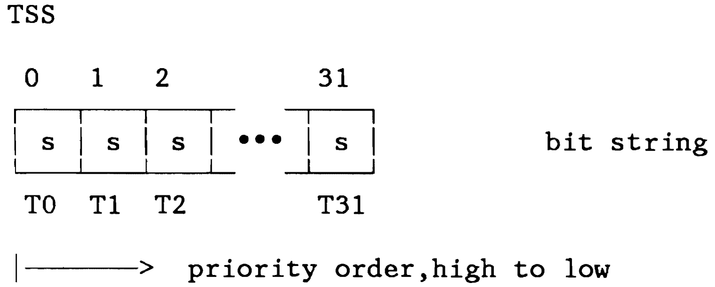


Figure 26. Task Selection String (TSS)

There are entries for up to 31 subtasks and for 1 maintask. A separate TSS exists for each partition which is located in the PCB. A task is ready to run when its status bit (s) is one. Since the TSS is set up in task priority order the status bit of the highest priority task being ready to run can be found by a left to right scan. The corresponding task identifier can be found in the task identifier string (see Figure 27).

```
TIDSTR

┌─────┬─────┬─────┬──      ──┬─────┐
│ TID │ TID │ TID │   •••    │ TID │         1-byte entries
└─────┴─────┴─────┴──      ──┴─────┘
 T0    T1    T2               T31

|───────>  priority order,high to low
```

Figure 27. Task Identifier String (TIDSTR)

TIDSTR describes the priority of tasks within a partition. It is located in the Partition Control Block (PCB).

Since both TSS and TIDSTR are set up in priority order, the status bit and the task identifier of a task can be addressed by mean of the same order number.

The following two samples illustrate the TID setting for the the BG partition and for the system-partition.

Example 1:
Default setting of a BG-partition
immediately after IPL:

```
TIDSTR
┌────┬────┬────┬──     ──┬────┐
│ 21 │ 00 │ 00 │   •••   │ 00 │
└────┴────┴────┴──     ──┴────┘
  |    |    |              |
  |    |    |              |
  |    V    V              V
  |    zeros because no subtasks are attached yet
  |
  V
  TID of BG maintask
```

Example 2:
Setting within the system-partition

TIDSTR

```
|01 |02 |03 |04 |06 |08 |07 |09 |0A |0B |0E |0F |0C |20 |00 | •••

 SNS DSK RAS PMR PGN DIR SUP CRT ASY ERP LOG SVT LCK AR    |
                                                           V
                                                           Reserved

|-------->  System task priority order
```

With a given TID the task's TIB pointer can be found via the TIB address table (TIBATAB), the layout of which is shown in Figure 28.

TIBATAB

```
|<---System Tasks--->|<------------User Tasks------------>|
|                    |                                    |
|                    |                                    |
|0   4   8        128|                    192 |           |

| 0 |TIB|TIB| •••  |TIB|TIB|TIB| •••  |TIB| •••  |TIB| ••• |   |

| -  SNS DSK       |AR  BG  F1       Fn-1      |S1        Sm |
|                  |                           |              |
|                  |                           |              |
|                  |<--------Maintasks-------->|<-Subtasks--> |
|---->4*TID (offset within TIBATAB)
```

Where:
n = number of partitions
m = number of subtasks
TIB = Address of TIB

Figure 28. TIB Address Table (TIBATAB)

Once a TIB pointer is known, all related control blocks and areas can be accessed as shown in Figure 29 on page 95.

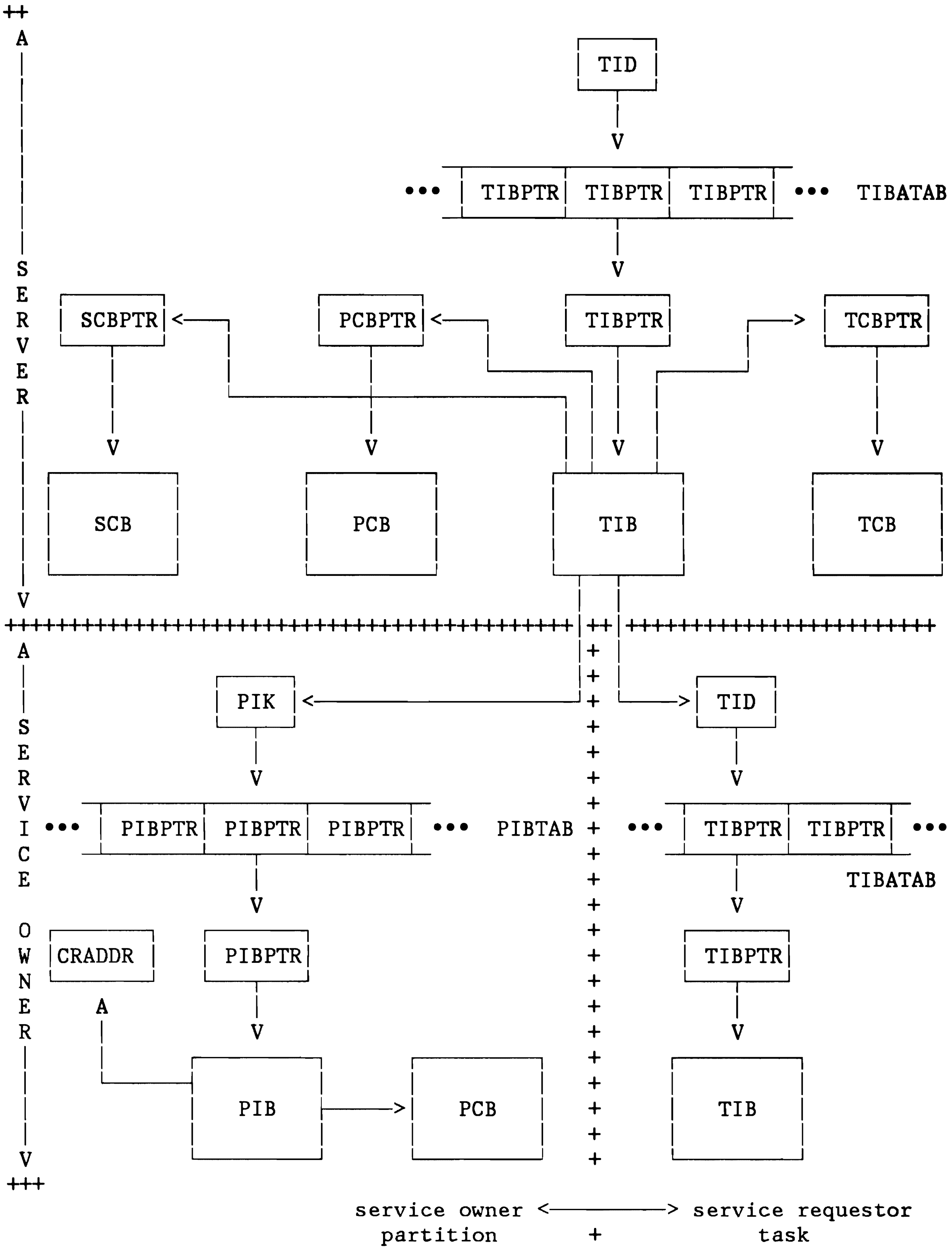


Figure 29. Task Selection Control Block Interrelationship

4. Relating Control Blocks to Tasks

Control block connection is done by setting up the TID, PIK, TIBPTR, TCBPTR, PIBPTR, PCBPTR, and CRADDR fields in such a way that they correspond to a task which has to be made active. Figure 29 on page 95 shows that a control block connection can be done by assuming a given task identifier. In case of task selection, some pointers (e.g. PCB pointer) are already known as a result of the first two steps of task selection.

> **Note:** The service owner and the server may be the same: in case of user task processing, the PIB and PCB belong to the same partition. On the other hand there may be a chain of service requesters and then the last task in the chain will be a member of the service owner partition.

If there is a chain of requesters all tasks in this chain apart from the last one will be system tasks.

Once the control block connections have been established a task is active. But prior to returning to task processing it might be necessary to perform some supervisor services for these tasks. This is done by step 5 of task selection.

5. Processing of Task Selection Exit Routines

Before a user task is activated the task selection routine tests whether control has to be transferred to any task selection exit routine.

Bits 0 to 7 of the TIBFLAG byte are associated with specific routines. They are scanned left to right and, if the bit is set to one, the corresponding routines are entered. After entry to a routine the corresponding bit is reset to zero.

There are the following exit routines:

- SVRETURN (Bit 0: X'80' - CSVRET in TIBFLAG)
  Return to an interrupted (reentrant or gated) supervisor service routine. When partition balancing and/or job accounting support is active and and the new accounting owner is not the old one the current accounting interval is determined and added to the old owners time counter field (system overhead or user CPU time) and a new accounting interval is initialized. The routine identifier is moved from the TCB into the RID field.

  In case of a gated routine the resource (which is given by the RID) is freed and any waiting tasks are posted. The general registers of the interrupted routine are loaded from the task's system save area and control is returned to the routine loading its program status word.

- REENTSVC (Bit 1: X'40' - RETRYSVC in TIBFLAG)
  Reenters the SVC first level interrupt handler routine without issuing an SVC. It is used for performance purposes. It allows a short path when the entry to an SVC routine should be retried.

- DELMOVE (Bit 2: X'20' - TIBDELMV in TIBFLAG)
  Enters the general delayed move routine. Bits 0 to 7 of the TIBDMFLG byte are associated with the delayed move routines. The routine address is determined via a left to right scan of TIBDMFLG.

  One of the following routines will be activated:

  - MOVECCB (Bit 0: X'80' - TIBCMVEX in TIBDMFLG)
    This exit routine has two different functions:
    1. Move a CCB which could not be copied back after completion of channel program translation because the page containing the virtual CCB was not in processor storage. Return to task selection entry (for 370 mode only).
    2. Return to SVC 119 (X'77') processing after the FBA I/O operation has been completed.
  - XPCCEXIT (Bit 1: X'40' - TIBXPCEX in TIBDMFLG)
    If a XPCC request is executed, where the destination is not in the same space than the originator, the control information to be stored into destination XPCCB (such as traffic bits, user data, etc.) will be saved into a supervisor control block (CRCB, see Figure 266 on page 544) and transferred to the destination XPCCB, if the associated path is dispatched.
  - SV103RET (Bit 2: X'20' - TIBSFLEX in TIBDMFLG)
    If I/O is made by the SVC 103 routine, the SV103RET flag will be set in order to return to the SVC 103 routine after I/O processing.
  - TINFMOPD (Bit 3: X'10' - TIBPERST in TIBDMFLG)
    Modifies the PER active indication in the partition control block (PCB) and the save area PSW of the specified partition.

- CNCLEXIT (Bit 3: X'10' - FETCHEOJ in TIBFLAG)
  There is no save area available to be used by the resident part of the terminator routines. This exit is used to activate the terminator and to return control to it after an interruption.

- ICCFEXIT (Bit 4: X'08' - ROLLOUT in TIBFLAG)
  It supports synchronization between an ICCF 'Pseudo Partition' task and the ICCF High Priority Task.

- EXTRETRN (Bit 5: X'04' - CDELEX in TIBFLAG)
  This activates the user timer exit routine or posts the timer ECB after a timer interrupt for this task. Since timer interrupts are asynchronous to user task processing, activation and posting is delayed in order to have the system save area

available. This is necessary because a page fault may occur when accessing the save areas or the timer ECB.

- OCEXIT (Bit 6: X'02' - OCPEND in TIBFLAG)
  Provides delayed activation of a user OC exit routine. This is necessary because an MSG command is asynchronous to the corresponding maintask processing and the save areas involved may be paged-out.

- APSEXIT (Bit 7: X'01' - APSEXFLG in TIBFLAG)
  Gives control to the ACF/VTAM APS SWAP routine. After returning from an APS routine a test is made whether any OC or timer interrupts are unprocessed yet. If so, the corresponding TIBFLAG bit is set. In addition to this CNCLEXIT may be reactivated when the APSEXIT was called during EOJ processing. After processing, the APSEXIT routine returns to the entry of the task selection routine.

6. Initialize Task's Processing and Give Control to it

Before control is given to a task a test is made whether tasks program status word (PSW) is in a disabled state. If so, an interrupt window is opened, allowing for any pending interrupt to occur. The interrupt window is closed immediately. This interrupt window prevents any task from running fully disabled (i.e. over a boundary of supervisor services). When partition balancing and/or job accounting support is active and and the new accounting owner is not the old one the current accounting interval is determined and added to the old owners time counter field (system overhead or user CPU time) and a new accounting interval is initialized. For a maintask which is task timer owner the remaining time slice is set. At the end of task selection the Routine Identifier (RID) field is set to the value USERTID. This indicates that normal tasks processing is active. The task's floating point and general registers are loaded and control is given to the task loading its Program Status Word (PSW).

INTERNAL GATING MECHANISM

The internal gating mechanism controls the usage of internal resources.

Its function is to

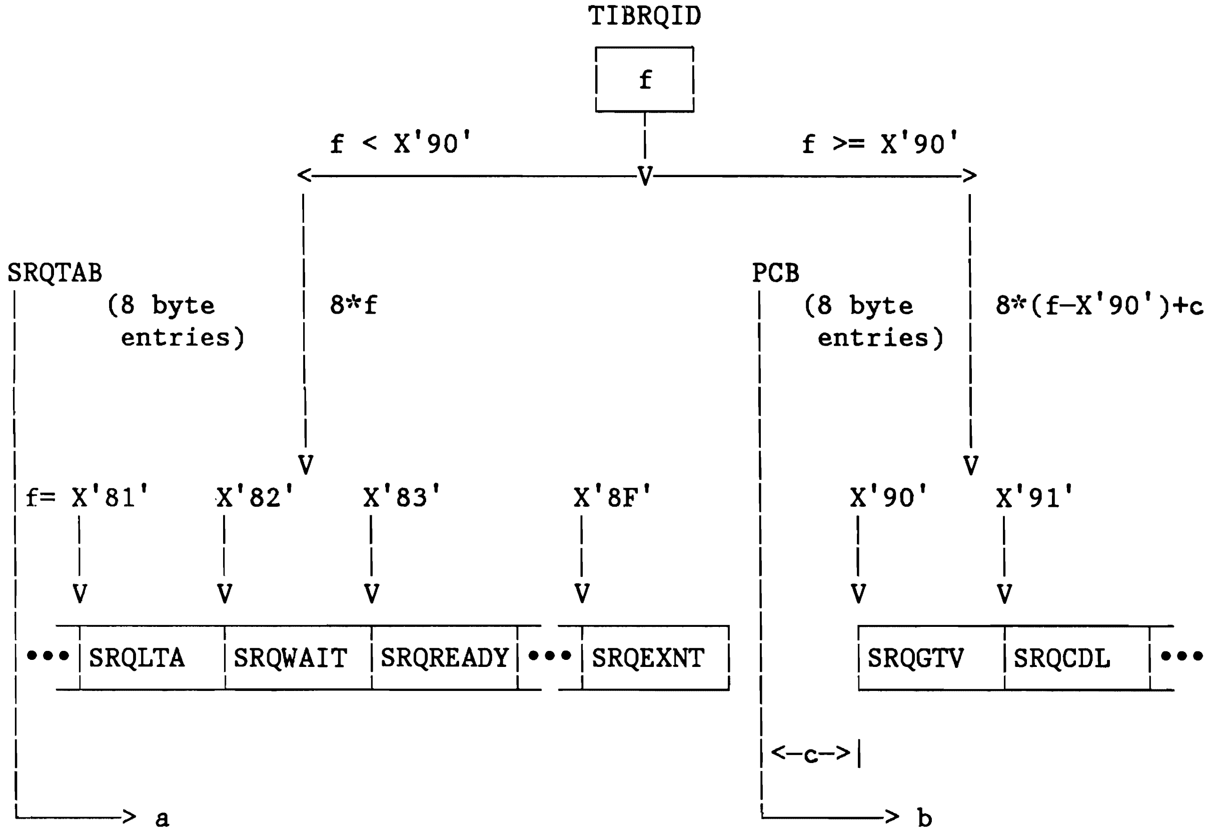
- Post/unpost Tasks and Partitions
- Free/occupy Resources
- Maintain Wait Queues

Flags, fields, tables involved in internal resource handling are:

- Partition and Task Selection String (PSS, TSS)
- Task Status Flags (located in TIB, label TIBRQID in the supervisor - Figure 311 on page 611)
- Resource descriptors (located in SRQTAB and in PCBs) including a header for building wait queues
- Wait Queues (chains of TIBs enqueued on a resource)

The rough status of a task (ready to run or not ready to run) is given by its status bit in the Task Selection String (TSS). A more exact description of a task's status is given by its task status flag and the corresponding resource descriptor.

Addressing Resource Descriptors



```
f  = Value in TIBRQID byte (task status flag)
c  = Displacement of first descriptor (SRQGTV) within PCB
a  = 8 * f               (displacement to an entry in SRQTAB)
b  = 8 * (f-X'90') + c   (displacement to an entry in PCB)
```

Figure 30. Addressing Resource Descriptors

Resource Descriptors

For compatibility and performance reasons there are different gating concepts implemented. The method which has to be used with a given resource is specified via a resource descriptor entry, shown in Figure 31.

```
┌───────────────────┬───┬───┬───┬───┐
│                   │   │   │   │   │
│ A   C   C   C     │ I │ F │ G │ O │
│   │   │   │       │   │   │   │   │
└───┴───┴───┴───────┴───┴───┴───┴───┘
        A           A   A   A   A
        │           │   │   │   └──Owner ID  (RQOWNER)
        │           │   │   └──Resource byte (RBYTE)
        │           │   └──Flag byte         (RQFLAG)
        │           └──Resource ID           (RQID)
        │                (= task status flag)
        └──4 byte queue header               (RQCHAIN)
```

Figure 31. Resource Descriptor Entry

Description of Entries

- ACCC: Queue header
  In combination with specific resources the queue header is used for building wait queues.

  - A = X'FF' (first byte of a queue pointer)
    indicates end of a wait queue. In this case pointer ACCC points to the first byte of the corresponding resource descriptor.
  - A = X'00' (first byte of a queue pointer)
    indicates a (or another) waiter is enqueued on a resource. In this case pointer ACCC points to the first byte of the waiters TIB.

  A = X'FF' in a queue header indicates that there are no waiters enqueued on the resource. A task is enqueued on a resource inserting its TIB to the front of a wait queue.

  Note: The symbolic names of gates, their types and displacements (flag values) are shown in Figure 311 on page 611.

- I: Resource ID:
  For identification purposes, byte 4 of each entry contains the corresponding task status flag value. For example, in the entry SRQREADY, I = X'83'.

- F: Flag byte (Resource Queue ID):
  specifies the gating method to be used.

Flag	Labels	Apprev.	Type Description
X'80'	SYSTQ	S	system queue, priority posting, switchable gate
X'40'	PARTQ	P	partition queue, priority posting, switchable gate
X'20'	WAITCHN	T	TIB chain, selective posting, permanently closed gate
X'10'	IOCHN	I	I/O chain, selective or direct posting, permanently closed gate
X'08'	PGATE	C	no queue, direct posting, permanently closed gate
X'04'	PREADY	O	ready to run state, permanently opened gate
X'01'	NORDY	N	do not ready task for cancel

- G: Resource byte (Gate):
  The most significant element of internal resource handling is a resource byte, known as a gate. The content of the resource byte is used as a switch:

  - G = X'00' : a resource is occupied (NOTFREE)
  - G = X'80' : a resource is free (FREE)

  1. Switchable gates: (P or S)
     The content of a switchable gate may be changed. It may represent a single item resource (routine, system task, etc.) or multiple items of a resource (channel queue, copy buffers, etc.). Services are provided to close/open the gate, dequeue/enqueue waiters.
  2. Permanently opened gates: (O)
     They are used in combination with the ready to run status of tasks. Whenever a task is ready to run its TSS bit is turned on and its status flag points to a permanently opened gate.
  3. Permanently closed gates: (C, I or T)
     They are used in combination with the not ready to run status of tasks when switchable gates cannot be used. They are assigned to fixed owners. Tasks pointing to permanent gates are posted/unposted individually by the resource owners upon completion of a service (I/O, program fetch, etc.).

- O: Owner ID:
  ID of resource owner (Task ID).

Gating Methods

The different gating methods are described in the following, also the range of application of the different kinds of gates and the function of the UNPOST/POST/RPOST routines in connection with the gate types.

Setting a Task Ready-to-Run

Tasks selection bit in TSS is turned on.
Partitions selection bit in PSS is turned on.
Tasks status flag (TIBRQID) is setup to point to a permanently opened gate (either READY or CONDRDY).

Setting a Task Not Ready-to-Run

Tasks selection bit in TSS is turned off.
TSS is tested, when it is a zero string partitions selection bit in PSS is turned off too.
Tasks status flag (TIBRQID) is setup to point to a closed gate.

UNPOST Routine

> **Note:** The UNPOST routine is always called by a task setting itself to wait.

The parameter to the UNPOST routine is a pointer to the corresponding resource descriptor. In some cases an ECB (or any other) address is in the caller's register R1 which will be passed from the UNPOST to the RPOST routine. For this purpose the last three bytes of R1 are stored to the three bytes at label TIBSTATE+1 (located in the TIB).

RPOST Routine

The RPOST routine is called in order to post one or more tasks enqueued on a resource. Parameter to the RPOST routine is a pointer to the corresponding resource descriptor. In some cases an ECB (or any other) address is in the caller's register R1 which will be used to identify a wait condition: the last three bytes of R1 are compared with the content of the three bytes at TIBSTATE+1.

POST Routine

POST routine is called to post a special task, which must be waiting for a permanently closed resource with no central wait queue support. It provides a fast post service e.g. for I/O bound tasks. The parameter is a TIB pointer instead of a pointer to a resource descriptor. Note that calls to POST and RPOST are not interchangeable. It is necessary to call the right one in order to get a correct result.

Processing of Conditionally Ready State (CONDRDY)

In combination with resource types PS and SS tasks are posted one at a time. When there are any other tasks enqueued on the resource the posted one becomes the CONDRDY state, which means that it has been posted in order to take a resource. In order to allow later identification the old resource pointer is saved to tasks TIB. In some situations the task is not able to take the reserved resource and tries to enter any new wait state. When the UNPOST routine detects a task which is conditionally ready and the corresponding resource is not occupied yet it sets up an implicit call to RPOST using the saved resource pointer. Such a way the next waiter from the reserved queue is posted, allowing current task to enter the new wait state.

Description of Routines

1. Using a Permanently Closed Gate with no Wait Queue Implemented (Type P).

   This method is used when the waiting routines are known to the posting routine and can, therefore, be posted directly.

   UNPOST routine:
   When the task has a reserved resource RPOST is called. After this tasks status byte is set up to point to the given gate and the task is set not ready to run.

   POST routine:
   Tasks status byte is changed to READY (X'83') and the task is set ready to run.

   > **Note:** A call to RPOST would not be correct, since there is no possibility implemented to find a waiting routine using the resource descriptor.

2. Processing of a Partition Wait Queue with Switchable Gate (Type PS).

   This mechanism is used in combination with the partition internal gates (located in the PCBs). It is assumed that the waiting and the posting tasks belong to the same partition.

UNPOST routine:
When the task has a reserved resource RPOST is called. After this the gate is closed (if not closed already) and tasks status byte is setup to point to the closed gate. Tasks TIB is inserted to the front of the wait queue. The task is set not ready to run.

RPOST routine:
The gate is opened by the posting routine. The queue is scanned and the oldest waiter (when any) is dequeued. Status byte of the task is set to CONDRDY (respectively READY when it was the only task enqueued on the resource). The dequeued task is set ready to run.

3. Using a Common Wait Queue and a Permanently Closed Gate (Type CP).

   This mechanism is an extension to 1. A wait queue is maintained which queues the TIBs of the waiting routines together. In addition the contents of the waiting routine's and the posting routine's register 1 is used for wait identification.

   UNPOST routine:
   When the task has a reserved resource RPOST is called. After this tasks status byte is setup to point to the given gate. The waiting routine's register 1 is stored to the TIBSTATE field. The task's TIB is inserted at the beginning of the corresponding wait queue. (The header of the wait queue can be addressed via the resource descriptor entry.) The task is set not ready to run.

   RPOST routine:
   A scan of the wait queue is performed. All tasks whose TIBSTATE match the passed contents of the posting routine's register 1 are removed from the queue. Status bytes of the tasks are changed to READY. The tasks are set ready to run.

4. Using a System Wait Queue and a Switchable Gate (Type SS).

   This is an extension to 2. By maintaining a common wait queue, tasks of multiple partitions can be handled.

   UNPOST routine:
   When the task has a reserved resource RPOST is called. After this the gate is closed (if not closed already) and the task's status byte is set up to point to the given gate. The task's TIB is inserted at the beginning of the corresponding wait queue. The task is set not ready to run.

   RPOST routine:
   The gate is opened by the posting routine. The queue is scanned and the partition priorities of all tasks compared. The oldest waiter (when any) from the highest priority partition is dequeued. Status byte of the task is set to CONDRDY

(respectively READY when it was the only task enqueued on the resource). The dequeued task is set ready to run.

5. Gating Via a Permanently Closed Gate With the Additional Possibility to Scan for Waiting Routine (Type FP).

   This is an extension to 1. It allows fast direct posting as well as a scan for tasks waiting for a specific ECB. It is implemented for two resources: RBWAIT and RBENQ.

   UNPOST routine:
   When the task has a reserved resource RPOST is called. After this task's status byte is set up to point to the given gate. The last three bytes of the caller's register 1 (ECB pointer) are saved into the TIBSTATE field. The task is set not ready to run.

   POST routine:
   Direct posting is supported in order to allow fast posting e.g. from I/O bound state. The task's status byte is set to 'ready' with no regard to the contents of TIBSTATE. The dequeued task is set ready to run.

   RPOST routine:
   The task identifier string of the presently active partition (label TIDSTR, located in the PCB) is scanned for a task with the requested status flag. Each task with the given status flag is posted if

   - the contents of the posting routine's register 1 is zero or
   - the contents of the waiting routine's TIBSTATE is zero or
   - the posting routine's register 1 is equal to the contents of the waiting routine's TIBSTATE field.

TASK AND PARTITION KEY DEFINITIONS

Storage Protection Key

Each partition in VSE is assigned a unique storage protection key. A storage protection key is the hexadecimal representation of the value 16*n, where

0 <= n <= number of partitions

Storage protection keys are assigned depending on the number of partitions according to the scheme shown in Figure 32:

		PIK Value in COMREG										
Part. id	Part. name	Number of Partitions										
		12	11	10	9	8	7	6	5	4	3	2
00	SYS	00	00	00	00	00	00	00	00	00	00	00
01	BG	10	10	10	10	10	10	10	10	10	10	10
0C	F1	C0	B0	A0	90	80	70	60	50	40	30	20
0B	F2	B0	A0	90	80	70	60	50	40	30	20	
0A	F3	A0	90	80	70	60	50	40	30	20		
09	F4	90	80	70	60	50	40	30	20			
08	F5	80	70	60	50	40	30	20				
07	F6	70	60	50	40	30	20					
06	F7	60	50	40	30	20						
05	F8	50	40	30	20							
04	F9	40	30	20								
03	FA	30	20									
02	FB	20										

Figure 32. Storage Protection Key

Partition Identification

Normally a partition is identified by its unique storage protection key. Due to its additional use a special storage protection key value is often called a 'Partition Identification Key' (PIK). In some cases a partition is identified by a 'Partition Identifier' (PID) value which is just the value PIK/16.

Note: The PID values are contained in the first digit of a storage protection key in the table of storage protection keys.

Task Identification

Tasks are identified by hexadecimal numbers 1 to X'FF'. The following table shows the task identifier (TID) values and their assignments to particular tasks:

System Task		Main Task		Sub Task	
TID		TID		TID	
00	Unused	20	AR	30	**
01	SNS - CCH/MCAR task to issue	21	BG	31	**
	SENSE command	22	F1 *	32	**
02	DSK - resident disk error	23	F2 *	33	**
	recovery task	24	F3 *	34	**
03	RAS - CCH/MCAR maintask	25	F4 *	35	**
04	PMR - page manager task	26	F5 *	...	
05	Unused	27	F6 *	nn	**
06	PGN - page in task	28	F7 *		
07	SUP - fetch task	29	F8 *		
08	DIR - directory read task	2A	F9 *		
09	CRT - display operator console	2B	FA *		
	support task	2C	FB *		
0A	ASY - asynchronous operator	2D	Unused		
	communication support task	2E	Unused		
0B	ERP - error recovery task	2F	Unused		
0C	LCK - lock service task				
0D	Unused				
0E	LOG - logger task				
0F	SVT - automatic volume recognition				
	task				
10	Unused				
...					
1F	Unused				
20	AR - attention routine task				

Figure 33. Task Identifier (TID) Values

Notes:

* Depending on the number of partitions, all or some of these identifiers may be unused (in descending order of values).

** A pool of subtasks is created and maintained by the supervisor. The size of this pool is given by the maximum number of subtasks active at the same time.

Identification of Current Partition and of Current Service Owner

Before control is given to a task the dispatcher sets up the PIK field (bytes 46-47 of the background communication region - BG-COMREG) by a partition identifier key value.

In case of a system task it is the PIK value of the service owner partition (in special situations it may be the system partition key). In case of a user task it is the PIK value of the task's home partition.

> **Note:** Whenever a task of the BG partition is active, the PIK field is set to the partition identifier key of the BG partition. Since bytes 46-47 of the other communication regions are generated with the corresponding foreground partition identification keys, any active user task may find its own partition identification key via its own COMREG.

Identification of Current Task

Before the dispatcher gives control to a task, it puts the task identifier into the TID field at displacement 90-91 in the system communication regions (SYSCOM). The TID value in the TID field identifies the task which is currently active. This may be any system or user task.

LTID (Logical Transient Owner)

The LTID, a halfword (LIK) at displacement 88 in the system communication region (SYSCOM) contains the same value as the TID when the Logical Transient Area (LTA) is in use and, therefore, identifies the owner of the LTA. When the LTA is free, the LTID is zero. The SVC 2 (X'02') routine sets the LTID, and the SVC 11 (X'0B') routine resets it to zero.

Notes:

1. Do not use this interface anymore.

2. Any logical transient routine may find its own task identifier by using the TID field.

LTK (Logical Transient Key)

The logical transient key, a halfword (LTK) at displacement 110 in each partition communication region (COMREG), has a zero value in the high-order byte and a key value in the low-order byte. In a foreground communication, the key value in the LTK is not significant. The LTK in the background communication region (BG-COMREG) has the same value as the PIK of the partition of the task that owns the LTA, or contains zeros when the LTA is free. When the LTA is occupied by the task, therefore, the BGCOMREG has the same value in its LTK as in its PIK when the owning task is active.

Note: This LTK interface should not be used anymore.

PHYSICAL INPUT/OUTPUT CONTROL SYSTEM (PIOCS)

Physical IOCS is that portion of the resident supervisor that:

- Builds a schedule of I/O operations for all devices on the system (CHANQ Table). Refer also to Figure 139 on page 349 (I/O Request Enqueuer).

- Starts the actual I/O operations on a device (SIO/SIOF Instruction). Refer also to Figure 138 on page 344 (Channel Scheduler).

- Monitors all events associated with I/O operations. Refer also to Figure 136 on page 333 (I/O Interrupt Handler).

- Performs error recovery actions. Refer also to Figure 141 on page 351 (Disk Error Recovery). Refer also to VSE/Advanced Functions Diagnose Reference: Error Recovery and Recording Transients, LY33-9108.

I/O REQUEST ENQUEUER

When a channel program is to be executed for a user, the I/O Request Enqueuer routine first checks to see if a channel queue entry is available.

If the channel queue is full, the issuer is set CHANQ-BOUND until a channel queue entry is available again, which is normally the case after completion of I/O interrupt processing.

> **Note:** The occurrence of this bound condition is an indication that the number of CHANQ entries, either the default value or the value specified at IPL time, is less than the number of concurrent I/O requests. Low performance may be the result. This situation could have been prevented by increasing the defaulted or specified number in the CHANQ parameter of the SYS-command at IPL time.

If an entry is available in the channel queue, the GETPUB routine first validates the users parameters and checks them for correctness (Error Exits: ERR21, ERR25, ERR26, ERR27). In case the users input has been proven to be correct, the I/O request enqueuer does some special work for privileged devices and/or components.

- For all I/O requests directed to a device which is logically assigned IGN (Ignore):

  It ensures that these type of requests are immediately posted I/O complete without having actually been started.

- For Cathode Ray Tube (CRT) I/O requests directed to the operator's console (SYSOCDEV):

  It ensures that all I/O requests, except SNS task and VSE/OCCF requests, are set "I/O-BOUND" (RESVCIO) as long as CRT-FETCH is in progress.

  It ensures that all I/O requests, except SNS task requests, are passed to the OCCF intercept routine (if VSE/OCCF is active) to inspect whether and where this message is to be routed.

- For Unit Record (UR) and Diskette I/O:

  It ensures that VSE/POWER can process none system task I/O requests directed to a device which the user did define as a 'spooled' device.

- For I/O requests directed to the operator console device (SYSOCDEV):

  It ensures that these I/O requests are passed to the Console Buffering Routine (described later in this section) for further processing.

- For DASD and Diskette I/O requests:

  It ensures that the associated channel programs starts with a valid command (ERR33). (Refer also to system files described later in this section.)

  It ensures proper DASD file protection in case the user specified DASDFP=YES (ERR42).

Special processing information is saved in general register 5 until a CHANQ entry has been allocated (after CCW Translation).

If the I/O request needs to be translated (370 mode only) control is passed to the CCW-Translation Routine (described later in this chapter) to get the virtual channel program copied into the copy blocks within the supervisor and to get all virtual addresses translated to their correct real addresses.
The CCW-Fixing Routine is used to get all referenced I/O areas TFIXed, if they are not already PFIXed, thus making sure, that this page will not be paged out by the PAGE MANAGER routine.
After return from the CCW-Translation or CCW-Fixing Routine, all the information which is needed to further process the I/O operation is saved in the CHANQ entry which is then enqueued into the chain of I/O requests that might already be waiting for this device.
The I/O requests are normally queued in a First In First Out (FIFO) sequence except SYSIO requests (SVC 15) which are enqueued due to a preassigned system task scheduling priority which does not match the dispatching priority. (For a sample of SYSIO request enqueuing refer to Appendix E).

If the just enqueued I/O request is not the first one in the device chain, control goes directly back to task selection.
If the request is the first one in the device chain, the CPU time is charged to the partition which issued the I/O request (refer to Job Accounting described later in this chapter) and control is passed to the Device Scheduler Routine.

Special processing support provided by the I/O scheduler or related SVC-routines will be described on the following pages.

Block Protection (SVC 35)

Block protection ensures that a 'block' on a disk device which is being held by one task is not accessed by another task unless the holding task has released the 'block' again.

CKD Devices (BBCCHH): The unit of protection is one track. The track address is retrieved from the users SEEK CCW, which must be the first CCW. The whole track is always protected against access by another task.

FBA Devices: The unit of protection is always the range of FBA blocks as specified in the DEFINE EXTENT CCW which must be the first CCW. The whole range of blocks is protected against access by another task.
If the first CCW is not a SEEK or a DEFINE EXTENT CCW, block protection is simply ignored and normal SVC 0 processing is done.
All requests to protect a track on a CKD device or a range of FBA blocks against simultaneous use will be entered into the Track Hold Table before the I/O Request Enqueuer gains control. The block protection routine forces the issuing task to be set TRK-bound if the given block is already held by another task. It will be reactivated as soon as the requested block becomes available which is normally the case after the holding task has released the track.
Multiple I/O requests for tracks or ranges of FBA blocks which are to be held are chained in a device chain with forward and backward pointers, and the appropriate PUB contains the index to the first Track Hold Table ENTRY. For the format of the Track Hold Table (THTAB) see Figure 310 on page 609.

Console Buffering

The Request Enqueuer provides special buffering support for all I/O requests directed to the hard copy console printer, assuming the I/O requests meet special requirements (see below), thus enabling the issuing task to immediately reuse the I/O area although the I/O operation is still ongoing.

The console buffering routine is bypassed however, if one of the following conditions exists:

- The I/O request is not a single write CCW with a byte count not higher 80 bytes.
- The PCI bit in the user's CCW is on.
- The user requested device end, irrecoverable I/O errors or sense data to be returned to him.
- The user has his own I/O error routine.
- The user issued an EXCP, REAL request.

The Console Buffer Table (CBTAB - see Figure 302 on page 596) consists of 104 bytes fixed length entries. The number of entries is two times number of partitions. CBNEXT is a fullword constant that points to the next buffer entry and is initialized at IPL time with the address of CBTAB, also allocated by IPL. CBNEXT is updated to point to the next entry every time a buffer is used. Whenever its value becomes greater than CBEND, CBNEXT is reinitialized with the value of CBTAB.
When the console buffering routine is entered from the Device Scheduler Routine, provided the next buffer entry (pointed to by CBNEXT) is free, the command code, flag byte, and byte count in the user's CCW are moved to the CBCCW, the user's output data is moved to the CBDATA area, and the user's symbolic unit address to the CBCCB.
The console buffering routine turns on the WAIT bit in the user's CCB/IORB, exchanges the users CCB/IORB pointer (R1) to point to CBCCB and then returns to the Device Scheduler Routine to get a CHANQ entry set up and properly enqueued to the operator console device.

In case the next buffer entry is not free (CBCCB WAIT bit not yet posted) the issuing program is set CBF-BOUND (Console Buffer Table entry bound).

System Files

The SYSFIL support of the supervisor allows to have system files (SYSRDR, SYSIPT, SYSPCH and SYSLST) on CKD and/or on FBA disk devices. The scheduler turns on a special bit in the CHANQ entry to ensure proper program flow within the I/O supervisor. Special processing however, is required for system files residing on FBA devices.

System Files on FBA Devices: SVC 103 (X'67') performs the input/output operations for system files on FBA devices. The code of the SVC 103 (X'67) consists of:

- The resident part, performing supervisor functions.
- The pageable part, loaded into the SVA, performing data management (blocking/deblocking) functions.

For details see description of SVC 103 (X'67').

CHANNEL AND DEVICE SCHEDULER

The I/O Scheduler is functionally subdivided into two very close related routines.

- The Device Scheduler which is only entered from the I/O Request Enqueuer and which drives a single device.

- The Channel Scheduler which is only entered from the I/O Interrupt Handler (described later in this section) and which drives a channel.

The Channel and Device Scheduler both ensure that all requests which have been enqueued by the I/O Request Enqueuer are started in FIFO order as soon as the resource (Channel, Subchannel or Device) is, or becomes available. The Scheduler ensure the accessibility of a device and the availability of a channel. If the channel is gated due to an I/O error, it ensures that only SNS-task requests are started until this condition is reset. In case the device is gated and the Scheduler was not entered from the I/O Request Enqueuer it tries to select another device which is attached to the same interrupting channel. If the Scheduler performs Device Scheduling functions and the device is gated control is directly passed to task selection.
If the channel and the device are available, the Scheduler does some SIO-preprocessing for special devices.

- For SYSIN I/O requests:
  It ensures not reading past /& (ERR30)

- For Tape I/O requests:
  It ensures control to be passed to the tape ERP in case the tape ERP did indicate that the next I/O request needs to be passed to it.

  In case of an I/O error on a previously initiated I/O operation it ensures the recovery channel program, as specified by the ERP System Task and not the channel program as specified in the users CCB to be initiated.

  All other tape requests, not meeting one of the conditions described above are ensured to be started with the assigned (PUB) Mode setting.

- For SYSLOG I/O requests:

  It ensures the appropriate (Message Reply ID (ASYNOC=YES)) as well as the partition prefix to be supplied on every message written to the console after IPL has completed.

- For Cathode Ray Tube (CRT) I/O requests directed to the operators console (SYSOCDEV):

  The I/O Scheduler ensures that all channel programs which are not CRT device compatible are passed to phase $$BOCRTA (C-transient) which will activate the CRT-System task to get these channel programs translated and executed afterwards.

- For DAS-Devices:

  The I/O Scheduler ensures that the user can only access those Records or EXTENTS on a volume, that he is authorized to access (ERR30, ERR32).

Following the SIO-preprocessing the I/O Scheduler actually carries out the requested I/O operation by means of a

```
  SIOF
```

instruction. Depending on the resulting SIO condition code the Scheduler either enters the I/O INTERRUPT PROCESSOR to further process condition codes 01 (CSW STORED) and 11 (DEVICE NOT OPERATIONAL) or it completes its Device Scheduling process by updating the appropriate SIO processing and SIO accounting information and passing control to the task selection routine.

Rescheduling of Selector- and BMPX Channel

The (non-MPX) Channel Scheduler starts or continues its processing by selecting another device attached to the interrupting channel which has not yet been started and which has an I/O request enqueued. The PUBs are scanned in a rotating sequence which covers the devices attached to the interrupting channel only, and the sequence always starts with the PUB following the one that has been started last. This rotating PUB scan ensures that the channel is shared by all devices. As many devices as the channel is capable to handle concurrently will be started. Once the channel is responding with condition code 10 (channel busy) the channel scheduling process is suspended and control is passed to the task selection routine. The channel rescheduling process will be resumed after the first I/O interruption from this channel.

In case all devices that have an I/O request enqueued have successfully been started, an indication will be given to prevent Channel Rescheduling next time an I/O interruption is encountered on this channel.

Rescheduling of MPX Channel

The Byte Multiplexer Channel per definition is considered to be available, which means that it is capable to drive almost all I/O devices concurrently. If, however for any reason the MPX-CHANNEL needs to be restarted, the rescheduling mechanism is exactly the same as described for Selector and BMPX Channels (see above) with the restrictions as described in "Burst or Overrunable Devices on MPX".
If the Channel Control (CHNTAB) Table does not indicate that the MPX must be restarted, the MPX Channel Restart depends on the type of devices attached to this channel.

Non-overrunable device on MPX Channel only

Whenever an interruption from a non-overrunable device occurs and there are more requests queued to the same PUB, the next I/O request for this device and only this one is started. Control is passed to the task selection routine if no more requests are enqueued or if the device is gated.

Burst or overrunable device on MPX Channel

When an interruption occurs on a byte multiplexer channel and burst-mode devices as well as overrunable devices are attached to it, the MPX-Channel Scheduler must provide special programming precautions to prevent 'Device Overruns' or 'Document Rejects (1419 MICR)'.

> Suppose that the START I/O on an overrunable high-speed byte-mode device is followed immediately by a request for I/O on a burst-mode device. Without any software precautions, because

the channel is available (operating in multiplex mode), I/O will be started immediately on the burst-mode device. The channel, now operating in burst mode, will be monopolized by this device. Any interruption from the overrunable byte-mode device may be lost, and the device may overrun, or in case the device is a 1419 (MICR) too many documents may be selected into the Reject Pocket due to late stacker select command.

These special programming support will be activated only if the operator did specify the AR command 'MPXGTN ON' and it will prevent an overrunable device to be running concurrently with a burst mode device by setting an indication in the Channel Control Table. Another burst or overrunable device (identified as such by IPL) will not be started before the already started device has concluded its I/O operation in which case the channel will be restarted (similar to Selector or Block Multiplex Channel Scheduling).

I/O INTERRUPT HANDLER

An I/O interruption occurs when an I/O operation terminates or the operator intervenes on the device, or if a block multiplexer channel becomes available (Channel Available Interrupt). The cuu address stored with the I/O interrupt into low core is used to allocate the PUB entry and to set up the related I/O pointers. It should be noted here, that, in order to prevent system hangs, a PUB must have been defined for any device of the installation, regardless of whether this device is being used or not. All interruptions presented from a device which was not defined will either be ignored, passed to the Channel Check Handler or, this also applies to channel available interrupts (CAI), will force the next device of the interrupting channel to be started. Before an I/O interrupt for a known PUB is actually processed, privileged components (OLTEP, BTAM, VTAM, POWER) are given the ability to inspect the Channel Status Word (CSW) via a BAL-type interface (channel end appendage).

> If, however, the I/O operation was initiated by the BTAM component, and the associated CCB indicates that this is a copied one, the I/O Interrupt Handler must first 'retranslate' the CCW address within the CSW before the BTAM appendage routine is entered. For more detailed information on BTAM processing see "BTAM Considerations (370 Mode Only)" on page 121.

If none of the above conditions exists, the CSW is evaluated and action is taken according to the table in Figure 34 on page 120.

CSW Status Bit On	Status Condition	Action
45 46	Channel Control Check Interface Control Check	Branch to the Channel Check Handler to interrogate the bits attempting recovery.
38 42 43 44 47	Unit Check Program Check Protection Check Channel Data Check Channel Chaining Check	Retrieve the sense information from the device and if user routine available, provide error information otherwise pass control to I/O error processing routine for resident and/or transient error recovery.
32	Attention	For attention from the operator console (SYSOCDEV) to activate the CRT-System task and/or the AR-task to further process this request. Branch to task selection routine. Attention interruptions are ignored if: 1. IPL is in progress. 2. Attention interruption is not from the operator console (SYSOCDEV).
35	Busy	Indicate that the channel is to be restarted. Branch to General Exit routine.
36	Channel End	Post user and/or reschedule the channel.
37 34	Device End Control Unit End	Post user and/or reschedule the channel. On a byte multiplexer channel, attempt to reschedule the device only.

Figure 34. CSW Testing in I/O Interrupt Handler

If the device status indicates that the channel program has been completed, the appropriate information is posted in the CCB/IORB.

If the CCB/IORB indicates that this is a copied CCB/IORB (X'20' in byte 6) control is given to a special routine (CSWTRANS) which,

- Frees all pages fixed for I/O areas.
- Retranslates the CCW address placed in the copied CCB/IORB to the correct virtual address.
- Releases the CCW copy blocks and IDAL blocks.
- Moves changed parts of the CCB to the virtual-mode program and release the CCB copy block. If the virtual CCB is not in real storage, the end of channel information is not copied to the virtual CCB by CSWTRANS. Instead, CSWTRANS posts a bit in the corresponding task information block (TIB) indicating to the dispatcher that the CCB should be moved before the task is dispatched. This is necessary because CSWTRANS may not cause a page fault and, therefore, cannot request that the virtual CCB be brought into real storage.
- Activate tasks waiting for copy blocks or waiting for page frames.

CSWTRANS returns control to the interrupt handler when it has finished processing. The I/O Interrupt Handler will then dequeue the CHANQ entry from the channel queue, assuming this was the final interrupt for a specific request and pass control to the Channel Scheduler to get another or the same device started again.

BTAM Considerations (370 Mode Only)

If a BTAM channel appendage is to be called after a BTAM request, the CCW address left in the CSW at channel end is retranslated by CSWTRBTM and returned to the I/O Interrupt Handler before the appendage is given control.

A BTAM I/O request is translated and copied in the same manner as normal I/O requests unless it comes from a BTAM channel appendage.

An I/O request coming from a BTAM channel appendage via the I/O Interrupt Handler must be translated without incurring any interruption (wait, page fault, etc.). In order to do this, BTAM specifies the maximum number of copy blocks that will be needed in addition to the number used by the original request. This number is contained in the residual count field of the CCB when an I/O request is made. After a BTAM request has been translated, and before it is put in the channel queue, the additional blocks specified are taken from the free queue and saved, so that they will be available when the request for the appendage is made. All of the copy blocks used by BTAM (except the CCB copy block) are first freed when a translation request for the BTAM appendage comes. They cannot be lost, however, because no other task can gain control of the CCW translation routines when an appendage routine is being processed. The special BTAM TCB is used for a BTAM channel appendage I/O request.

Once the channel program for an appendage has been translated, control is returned to the I/O Interrupt Handler. There is no need to enqueue the request in the channel queue, since the original request has not been dequeued. The I/O Interrupt Handler will immediately pass control to the Channel Scheduler to get the new request started.

(For more information see "Channel Program Translation (370 Mode)" on page 157).

Automatic Volume Recognition (AVR)

This facility keeps track of device-specific information of each DASD device in the system. The supervisor keeps a table, the volume characteristics table (VCT), which contains the specific information for each device (see "Layout of the VCT and DCT Tables" on page 124).

SVC 99 (GETVCE macro, see also Appendix B) retrieves data from this table for the user. The Service System Task (SVT) facility interrogates the device when requested, and updates the table. Requests for updating the table are made by the I/O Interrupt Handler and by SVC 101 (MODVCE macro, see also Appendix B), which can be issued by any user, but especially IPL and utility programs when a change to the device is suspected.

$AVRINIT: This phase is used by IPL and by the DVCUP command of Job Control to force a selective update of the Volume Characteristics Table entry (VCT entry) by means of a MODVCE macro (SVC 101).

To ensure that no interrupts are lost and that the information provided by GETVCE is valid, the system task (SVT) facility is needed to process the request. The request flow is shown in Figure 35 on page 123.

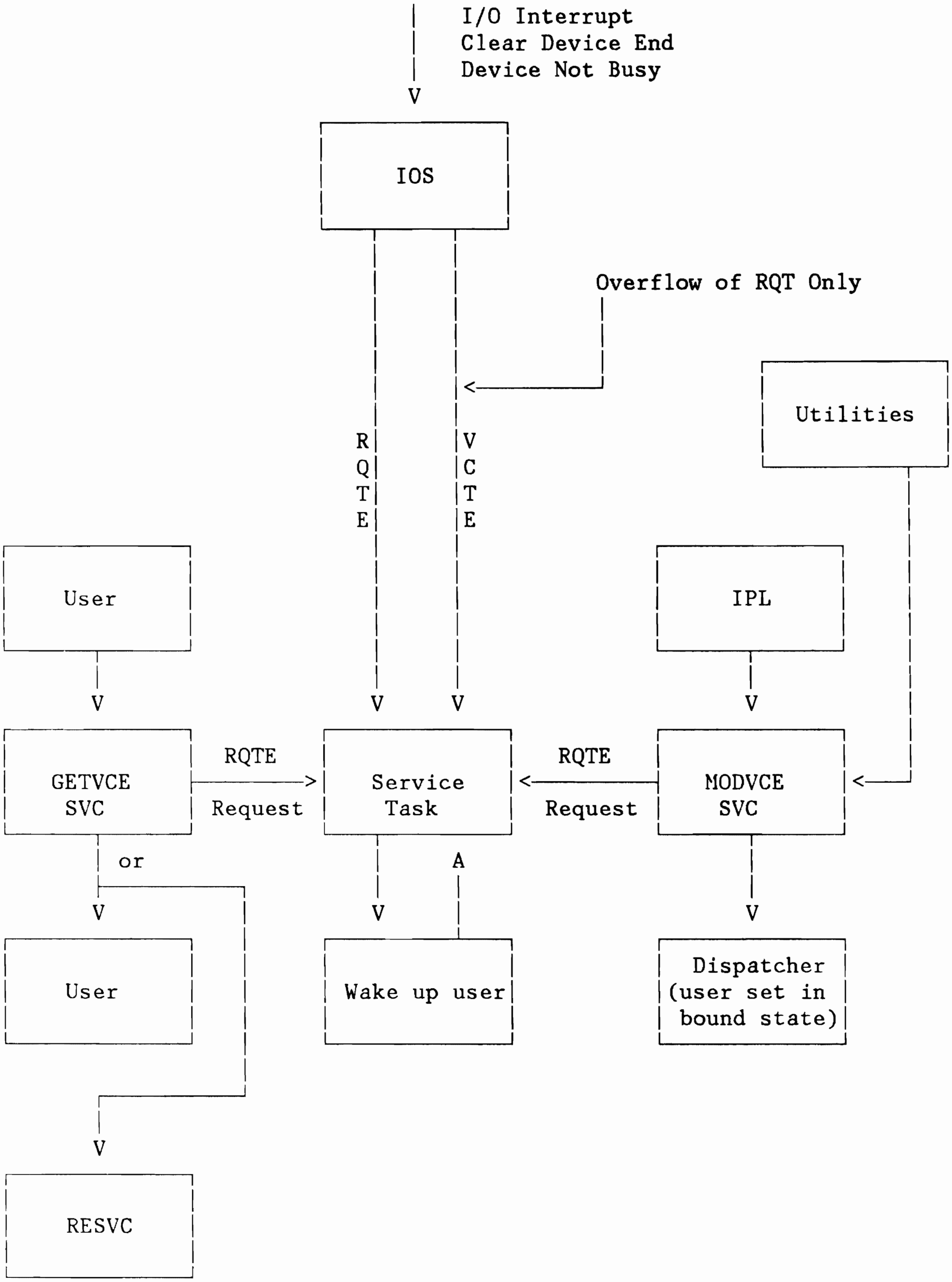


Figure 35. General Flow of Volume Characteristic Table Entry Update

Updating the VCT Table

The update request may come from two sources:

1. From the I/O INTERRUPT PROCESSOR whenever a DAS-Device became 'READY'

2. From SVC 101 (MODVCE)

The first category must issue the request immediately, since it cannot save the status. The second category, however, requires that the requesting task is readied not before the VCT-Table has been updated. Also, more than one task can request an update for a given device at a time, and if an entry is in the process of being updated, any GETVCE request must be queued in order to wake up the requesting task after the update. This results in two request queues:

- Device related only.
- Task and device related.

The first queue is the VCT table itself. Each entry (VCTE) has a work-to-do and work-in-progress flag.

The second queue is the RQT request table, a simple vector of task IDs, PUB indexes, and function flags. The address of the RQT table can be found at label RQTTAB. If the RQT overflows, the SVC can be retried at a later time using RESVC.

LAYOUT OF THE VCT AND DCT TABLES
The volume characteristics table (VCT) entry is defined by the AVRLIST macro (see Appendix B). An entry within this table has the following format:

DEC	HEX	Description
0	0	PUB address.
4	4	Volume ID.
10	A	Flag byte (indicating fields that are invalid)
11	B	Format of device characteristics (FBA, CKD, or CKD with RPS)
12	C	Address of volume table of contents (VTOC)
18	12	Offset to appropriate DCT entry

The Device Characteristics Table (DCT) entry is described by the DCTENTRY macro. Each entry is fixed length and describes the device characteristics of a CKD or FBA DASD device.

Usage of the two macros (AVRLIST and DCTENTRY) is discussed under "GETVCE Macro" in Appendix B. The start address of the VCT table can be found at label AAVRTAB, the start address of the DCT table at label DCTTABLE.

Asynchronous Operator Communication

Asynchronous replies to operator requests on the printer-keyboard or display operator console will be allowed. The console is no longer blocked up by Read requests so that the operator can respond to a request any time he wants to.

HANDLING WRITE-CCWS

The execution of Read-CCWs on SYSLOG is suppressed for all programs (except for the asynchronous operator communication support itself). Only the Write part will be executed when an SVC 0 is issued. For this purpose, the Write part of all CCW chains is copied and the command-chaining bit of the last Write CCW before the first Read CCW is set off.

For channel programs consisting only of Read CCWs, only the prefix is written, which is inserted by the supervisor.

Handling of channel programs containing only Write CCWs is unchanged, except for the longer prefix (reply ID).

Since the user's CCB, too, is copied for chains with read requests and since the address of the copy is loaded into register 1, the user's CCB is never posted when the write request is completed.

For the read part of the SVC, an internal control block, ORE (operator reply element), is initialized with all the relevant information about this read request; this block is set to 'waiting for a reply'. One ORE is generated at supervisor generation time for the attention task and one for every user task. system tasks use the ORE with the TID of the user.

HANDLING READ CCWS

The Attention key is used to enter both commands and replies.

1052 MODE

Whenever an attention interrupt is detected by the I/O Interrupt Handler, the asynchronous operator communication task, a system task, will be activated, which

- Starts a read operation on the console (the only read which is actually executed in the system)

- Determines whether a command or a reply (identified by the numeric reply ID) was entered.

If a command was entered, the attention task is activated. If a reply was entered, the asynchronous operator communication task finds the related ORE, moves the reply to the requester's task.

CRT MODE: The attention interrupt is handled by CRT transient routine $$BOCRTK in the same way as it is done by the I/O Interrupt Handler for the 1052 mode.

Bytes			
Dec	Hex	Label	Description
0 - 15	0 - F	ASYCCB	CCB used to write the cut channel program. It is an updated copy of user's CCB.
16 - 19	10 - 13	ASYUSCCB	Address of user's CCB.
20 - 23	14 - 17	ASYUSCCW	Address of user's CCW chain.
24 - 27	18 - 1B	ASYCUT	Address of READ CCW in the user's channel program or zero.
28 - 29	1C - 1D	ASYOCTID	TID of requestor task
30 - 32	1E - 20	ASYREPLY	Reply ID which is printed.
33	21	ASYCCHQP	Channel queue entry number within the message text.
34	22	ASYFLAG	Flag byte:
		OCCUPIED	X'80' ORE is active
		ASYRSTAT	40 ORE in READ state
		ASYERDEQ	20 Dequeue at SIO ERR39
		ASYEND	10 Just dequeue
		ASYQEDER	08 PUB queued in error with ORE
			04 Reserved
			02 Reserved
		ASYWRAP	01 ASYTASK wrap around SVC
35 - 37	23 - 25	ASYPT	Partition ID and MSG severity code '+ or -'
38 - 39	26 - 27		Reserved.

Figure 36. Layout of ORE (Operator Reply Element)

I/O ERROR PROCESSING

The main function of the I/O error processing routine is to save the error information into the appropriate error entry and to pass the error entry to one of the ERP system tasks.

Error Entries: There is one error entry for each device added at IPL time. This error entry is used for errors related to non-system task requests or to unsolicited interrupts. There is one additional error entry for each system task (except SNS and PGN), which is used for errors related to requests by this system task.

Error Chains: One chain of error entries is maintained for each of the three error recovery system tasks: sense task (SNS), disk error recovery task (DSK), transient error recovery task (ERP) and machine and channel check handler (RAS). Each error chain consists of an error chain header pointing to the first (if any) error entry in the chain. System task error entries are enqueued on top of the chain, device error entries at the bottom. Any error entry can be in only one error chain at a time.

General Procedure: The I/O error processing routine locates the appropriate error entry and removes it, if necessary, from any error chain. After setting the error information, it enqueues the error entry in one of the error chains, depending on the type of error and on the available information. The chain owner is posted, if not already active.

Each chain owner processes its chain in FIFO order. The first entry is dequeued, a recovery action is carried out and the error entry is then passed to another chain, if additional processing is needed; or freed, if the error recovery is completed. Error recovery system tasks always exit to the I/O Interrupt Handler, before resuming operation with the next entry in the chain.

Sense Task (SNS): The main function of the sense task is to read the sense data related to a unit check error and to save them, if needed, in the error entry. The error entry is then passed to the disk error recovery task (disk errors) or to the transient error recovery task (other errors).

Disk Error Recovery Task (DSK): The function of the DSK task is to analyze the sense data related to a unit check error from a disk device and to perform retry operations, if appropriate. The error entry is passed to the transient error recovery task for operator communication and/or error logging, if necessary. The DSK deactivates itself, when its error chain becomes empty.

Transient Error Recovery Task (ERP): Three distinct functions are assigned to the ERP task:

- Recovery operations for all I/O errors on non-disk devices
- Handling of all operator messages related to I/O errors
- Logging of I/O error information on the recorder file

The activity of the ERP task is monitored by resident code. The resident ERP logic dequeues the first error entry from the ERP chain and moves the contents of the error entry into a fixed area (ERQ1), which is accessible to the transient phases. Other system functions (SVC 44 and the Missing Interrupt Handler) also move information to be recorded directly into the ERQ1 area, when it is available. In this case, the ERP task first handles the information already available in the ERQ1 area, before processing the ERP error chain.

Machine and Channel Check Handler (RAS): Functions assigned to the Channel Check RAS task are:

- Logging of I/O error information on the recorder file
- Handling of all operator messages related to I/O errors
- Recovery operations for all I/O errors on non-disk devices

The activity of the RAS task is monitored by resident code. The resident RAS logic dequeues the first error entry from the RAS chain and moves the contents of the error entry into a fixed area (ERPIB), which is accessible to the RAS transients.

ERBLOC Area

The ERBLOC area is used as a common interface between all system components involved in I/O error processing. Byte 0-3 of the SYSCOM contain a pointer to this area. The layout of the ERBLOC area is shown in Figure 297 on page 589.

Error Entries

There is one error entry of each device added at IPL time. The pointer to this entry can be found in the PUB extension (PUBX). The length of the device related error entries varies from 29 to 60 bytes, depending on the number of sense bytes.

There is one additional error entry (in the I/O error block) per system task, except PGN and SNS task. The address to the I/O error block is contained in the appropriate system task TCB. Error entries are chained together and enqueued to the appropriate processing task. There is a separate chain for each, the SNS, DSK ERP and RAS task. A bit combination of outstanding recovery operations is used to address the appropriate chain. The anchor address of any of these chains is contained within the ERBLOC area (see Figure 297 on page 589). For the format of the error entries as processed by SNS, DSK, RAS and the ERP see Figure 298 on page 590 (see also "Error Recording Information" described next).

Error/Recording Information

The error/recording information stored in the error entries of the ERP error chain is passed to the transient ERP (one error entry at a time) via a single area which is the ERBLOC area. The information supplied in an error entry is completed by the ERP transients during processing. Two different layouts are used, for proper I/O error processing and simple recording, respectively (refer to Figure 298 on page 588).

Loading an ERP Transient

ERP is a system task, and when exit is to be taken to a physical transient a special routine initiates the ERP system task save area and then issues an SVC 5 to get the ERP monitor phase loaded. The physical transient phase that was read into the physical transient area is then entered.

To fetch another ERP transient, the active phase issues an SVC 5.

MISSING INTERRUPT HANDLER

The Missing Interrupt Handler (MIH) is a resident supervisor routine that interrogates all entries in the channel queue on an interrupt driven time slice basis. The MIH is entered whenever an ATTENTION interrupt from the system operator console (SYSLOG) is recognized, or whenever the system is going to enter an ENABLED WAIT state.

The MIH will first ensure that a defined time interval has elapsed, otherwise it will immediately return via the linkage register. If the defined interrupt has elapsed, all channel queue entries will be examined to determine whether they have been flagged as long-term entry. If the entry is not a long-term entry, it will be flagged as such if the associated I/O operation has been successfully initiated and if it is a device to be handled (see below). All entries which are already flagged will be further investigated in order to determine why these entries are still in the channel queue, for example, a channel end or device end is outstanding.

For this purpose, any associated I/O interrupt information as well as the current device status, retrieved by means of a TIO instruction, will be used to set up the appropriate message. The result of the TIO determines whether an information-type message or a decision-type message is provided. For both types of messages, the final action performed by the MIH depends on the communication bytes in the CCB and on the task which issued the I/O operation. All missing interrupts that can be uniquely identified as device errors will result in a record being written to the recorder file in a standard format.

Certain TP devices cannot be supported since the supervisor cannot distinguish between an endless polling loop or a subchannel hanging due to a missing interrupt. These conditions are handled by the individual components, usually by timer interrupts. These devices are the 2701, 2702, 2703, the ICA and the 7770. It should also be noted that, as long as SDAID is active, the MIH process will be bypassed.

DISK ERROR RECOVERY

Disk error recovery routines are the only resident device error recovery routines. They are described below. A-transients are only fetched when the error is to be recorded, or when an operator message is required.

For all other devices error recovery and recording is performed by A-transients. These transients are fully described in the VSE/Advanced Functions Diagnose Reference: Error Recovery and Recording Transients, LY33-9108.

The following "Error Correction Table" (see Figure 37 on page 132) and "Action Table" (see Figure 38 on page 135) show the errors that may occur and the action to be taken.

Error Correction Table for Disk Error Recovery

(To be used together with the 'Action Table' shown in Figure 38 on page 135.)

	Sense					
Error	Byte	Bit	X' '	May occur on..	Action	Logged
Logging only (LOGONLY)	2	7	01	FBA	Q	yes
Environmental data (ENVDATA)	2	3	10	All devices except 23xx	D	yes
Permanent error (PERMERR)	1	0	80	All devices except 3340	B	yes
Command reject (COMREJ)	0	0	80	All devices	A	no
Intervention required Equipment check (INTVRQD, EQUIPCHK)	0 0	1 3	50	FBA	C	yes
Intervention required * (INTVRQD)	0	1	40	All devices	C	no
Busout parity check (BUSOUT)	0	2	20	All devices except 23xx	C	yes
Busout parity check (BUSOUT)	0	2	20	23xx devices only	D	yes
Equipment check (EQUIPCHK)	0	3	10	23xx devices only	C	yes
Equipment check Alternate interface disabled (EQUIPCHK, ALTINTDA)	0 1	3 3	10 10	All devices except 23xx	E	yes
Equipment check (EQUIPCHK)	0	3	10	All devices except 23xx	D	yes
Data check (DATACHK)	0	4	08	23xx devices only	F	yes

* The error is logged for the 3340 and 3350, if bit 4 and/or 5 of sense byte 10 is on.

Figure 37 (Part 1 of 3). Error Correction Table for Disk Error Recovery

	Sense					
Error	Byte	Bit	X' '	May occur on..	Action	Logged
Data check	0	4	0A	3340 only	R	yes
Track condition check	0	6				
Operation incomplete	1	7	01			
Error correctable	2	1	40			
(DATACHK, TRCNDCHK, OPINCOMP, CORRECT)						
Data check	0	4	08	3340, 3350	I	yes
Operation incomplete	1	7	01			
Error correctable	2	1	40			
(DATACHK, OPINCOMP, CORRECT)						
Data check	0	4	08	3330, FBA	I	no
Operation incomplete	1	7	01			
Error correctable	2	1	40			
(DATACHK, OPINCOMP, CORRECT)						
Data check	0	4	08	3330, FBA	H	no
Error correctable	2	1	40			
(DATACHK, CORRECT)						
Data check	0	4	08	3340, 3350	H	yes
Error correctable	2	1	40			
(DATACHK, CORRECT)						
Data check	0	4	08	3330, 3350, FBA	G	no
Operation incomplete	1	7	01			
(DATACHK, OPINCOMP)						
Data check	0	4	08	3340	D	yes
(DATACHK)						
Overrun	0	5	04	FBA	G	no
Operation incomplete	1	7	01			
(OVERRUN, OPINCOMP)						
Overrun	0	5	04	All devices except FBA	D	yes
(OVERRUN)						
Overrun	0	5	04	FBA only	D	no
(OVERRUN)						

Figure 37 (Part 2 of 3). Error Correction Table for Disk Error Recovery

	Sense					
Error	Byte	Bit	X' '	May occur on..	Action	Logged
Track condition check Operation incomplete (TRCNDCHK, OPINCOMP)	0 1	6 7	02 01	3340	R	no
Track condition check (TRCNDCHK)	0	6	02	23xx devices and 3340 only	K	no
Seek check (SEEKCHK)	0	7	01	23xx devices and 3340 only	L	yes
Track overrun (TRKORUN)	1	1	40	23xx devices only	J	no
Track overrun (TRKORUN)	1	1	40	All devices except 23xx and FBA	N	no
End of cylinder (ENDOFCYL)	1	2	20	All devices except FBA	J	no
No record found (NORECFND)	1	4	08	All devices except FBA	M	no
File protection Operation incomplete (FILEPROT, OPINCOMP)	1 1	5 7	05	All devices except 23xx and FBA	O	no
File protection (FILEPROT)	1	5	40	All devices	A	no
Missing address marker (MISSMARK)	1	6	02	23xx devices only	D	yes
Operation incomplete (OPINCOMP)	1	7	01	3330, 3350, and FBA	O	no
Check data error Error correctable (CHKDATA, CORRECT)	2 2	0 1	C0	FBA only	P	no
Check data error (CHKDATA)	2	0	80	FBA only	D	yes

Figure 37 (Part 3 of 3). Error Correction Table for Disk Error Recovery

Action Table for Disk Error Recovery

(To be used together with the 'Error Correction Table' shown in Figure 37 on page 132.)

Action	Explanation
A	Write error message and cancel task.
B	Write error message. If operator decision is required, wait for operator reply and take appropriate action.
C	Retry once. If error persists, do action B.
D	Retry up to 10 times. If error persists, do action B.
E	Retry up to 10 times. Then do action B (also if recovery is successful).
F	Retry up to 254 times. If error persists, do action B.
G	Build CCW2 and all necessary restart CCWs to continue interrupted I/O operation.
H	Execute error correction function. If necessary, build restart CCWs and continue interrupted I/O operation.
I	Execute error correction function. Build CCW2 and all necessary restart CCWs to continue interrupted I/O operation
J	Post error condition in CCB. Ignore the error.
K	Do defective/alternate track switching and continue interrupted I/O operation.
L	Recalibrate and retry up to 10 times. If error persists, do action B.
M	If user wants retry on 'no record found', do action D, otherwise do action J.
N	If error occurred on a read command, do action C, otherwise do action J.
O	Build CCW1 and all necessary restart CCWs to continue interrupted I/O operation.

Figure 38 (Part 1 of 2). Action Table for Disk Error Recovery

Action	Explanation
P	Retry once. If error persists, ignore the error.
Q	Ignore the error. No recovery action required.
R	Build CCW1 and all necessary restart CCWs. Then do action K.

Note: CCW1 is built when 'operation incomplete' occurs without 'Data Check'. CCW2 is built when 'operation incomplete' occurs together with 'Data Check'.

Figure 38 (Part 2 of 2). Action Table for Disk Error Recovery

LOCK MANAGEMENT

Locks a resource against simultaneous use by other tasks.
Unlocks a given resource that was previously locked.
The SVC 110 (X'6E') is invoked by the LOCK and UNLOCK macros.

Resources that may be locked/unlocked are:

- Data sets
- Libraries
- Catalogs
- Program routines
- Control blocks, etc.

In a DASD sharing environment the SVC 110 (X'6E') may be used:

- To lock resources against simultaneous use by other tasks of the own system (internal locking), or

- To lock resources against simultaneous use by tracks of another VSE system (cross-system locking).

The SVC 110 (X'6E') routine (the lock manager), including the SVC 63 (X'3F') and SVC 64 (X'40') routines and the associated tables, is contained in the pageable part of the supervisor.

The lock manager is a serially reusable routine. Only one LOCK or UNLOCK request may be executed by the system at a time. If the lock manager is already active, the issuing task will be set to USEBND (X'8B') and afterwards into WAIT state (RESVCX).

Required Control Information

The resource to be locked/unlocked is described by the control block DTL (Define The Lock), the address of which is passed to the SVC 110 (X'6E') routine in register 1. Register 0 is used as a parameter passing register. The contents of register 0 is used to differentiate between LOCK and UNLOCK.

DEC	Description
0 - 2	Zero
3	Option Flag Byte
	X'80' Reserved
	X'40' UNLOCK JC=SYSID
	X'20' UNLOCK ALL
	X'10' UNLOCK ALL,JC=EOJ
	X'08' FAIL=WAITC
	X'04' FAIL=WAIT
	X'02' LOCK (USE) request
	X'01' SVC 110 (X'6E') request

Notes:

1. LOCK - Option flag byte contains: X'03'
2. UNLOCK - Option flag byte contains: X'01'

Figure 39. Contents of Parameter Passing Register 0

LOCK AND UNLOCK (SVC 110 - X'6E')

Locking a Resource

If a requested resource is available, it is assigned to the requesting task by building an entry for this resource in LOCKTAB and chaining an owner element to the LOCKTAB entry.

If the permanent LOCKTAB resp. owner element space (following the lock manager code) is exhausted, SVA space for LOCKTAB resp. owner element entries will be allocated.

If cross-system locking is requested an entry is placed into the external lock file, too. For the relationship between LOCKTAB and owner elements refer to Figure 250 on page 530.

The SVC 110 (X'6E') routine cannot issue an I/O request to the external lock file. When access to the external lock file is requested, the SVC X'6E' routine changes its status to that of a system task.

If a requested resource is locked by another task of the same system and FAIL=WAIT or FAIL=WAITC is specified in the LOCK macro, a deadlock test is performed to avoid a soft wait condition. If the system is deadlock free, the requesting task is set into WAIT state (RESVCX).

A deadlock test is also performed if FAIL=WAIT is specified and the supervisor runs out of LOCKTAB space or of owner element space.

For external locks a deadlock test is performed, if the disk block where an external lock entry should be entered is full and all entries of that block are in use by tasks of the own system.

> **Note:** Deadlocks, where tasks of different systems lock resources in reversed order, will not be detected.

If a task wants to lock a resource which is locked by a task of another system, the LCK system task sets up a time interval (SVC 10 - X'0A') and sets the requesting task to the "RURBND (X'8E')" state (RESVCX). When the time interval elapses, the timer interrupt handler takes all tasks waiting for externally locked resources out of the WAIT state.

Lock Options

LOCKOPT	CONTROL	Description
1	E	No other user is allowed to use the resource concurrently.
	S	Other 'S' users are allowed concurrent access, but no concurrent 'E' user is allowed. (Note 1)
2	E	No other 'E' user gets concurrent access, however, other 'S' users can have access to the resource(Note 2
	S	Other 'S' users can have concurrent access and, in addition, one 'E' user is allowed.
4	E	No other 'E' user from another system is allowed. However, other 'S' users from other systems may use the resource concurrently (LOCKOPT=2 support across systems).
	S	Other 'S' users and in addition one 'E' user from another system is allowed.

Notes:

1. Either one 'E' user or n 'S' users are allowed (n = number of 'S' users).
2. One 'E' user and n 'S' users are allowed.
3.

CONTROL=E	Resource is enqueued in exclusive mode.
CONTROL=S	Resource is enqueued in shared mode.
LOCKOPT=4	Defines a system action, which treats the lock request across systems as a LOCKOPT=2 request.

Figure 40. Lock Option and Control Parameter

incoming LOCK request		Current LOCK status of resource					
		LOCKOPT=1		LOCKOPT=2		LOCKOPT=4	
		CONTROL=		CONTROL=		CONTROL=	
LOCKOPT	CONTROL	E	S	E	S	E	S
1	E	W	W	W	W	W	W
	S	W	G	I	I	I	I
2	E	W	I	W	G	I	I
	S	W	I	G	G	I	I
4	E	W	I	I	I	G/W	G
	S	W	I	I	I	G	G

G = The LOCK request is granted (ret. code = 0).
I = Incoming LOCK request is inconsistent with current LOCK status (ret. code = 12).
W = Access to resource cannot be granted (ret. code = 4 or 16).
G/W = The access is granted, if the resource is already exclusively owned by the own system. The access is denied (ret. code = 4), if the resource is exclusively held by the other system.

Figure 41. System Actions Depending on Control Definition in DTLs

Unlocking a Resource

When a resource is to be unlocked, the appropriate LOCKTAB entry is cleared to zeros or, if there is more than one user of this resource, the unlocking task is removed from the owner chain of the entry.

If a LOCKTAB entry is cleared to zero, or if the locking status of the particular resource is changed to a lower control level (i. e. from exclusive to shared control), all tasks of the own CPU waiting for this resource are activated so that they retry their lock request.

If a resource is locked 'cross-system' and the locking status is changed, the entry on the external lock file is updated; as a result tasks of another CPU will find the resource available when they retry their lock request.

UNLOCK SYSTEM=sys-id (AR-Command)

All resources, which are held by another sharing system, will be freed (unlocked) and the corresponding entries will be removed from the external lock file. 'sys-id' specifies the CPU-id of the other system.

This service can be used only by the Attention task. Any other task issuing this macro, will be canceled with 'illegal SVC'.

RETURN CODES IN REGISTER 15

0 (X'00')	Successful request. All locks held by the other system have been unlocked.
4 (X'04')	The specified sys-id has not been found in the external Lock file (the operator specified probably a wrong system-ID).
8 (X'08')	External Lock file damaged.
12 (X'0C')	Irrecoverable I/O error on the Lock file.

UNLOCK ALL

All resources, which were locked by the task with 'KEEP=NO' will be freed (unlocked).

The SVA space of owner elements and LOCKTAB entries (if no more owner elements chained) is released.

UNLOCK ALL will be automatically called at task detach time and EOJ step.

UNLOCK ALL,JC=EOJ

All resources, which were locked by the issuing task including those with 'KEEP=YES', will be freed (unlocked).

The SVA space of owner elements and LOCKTAB entries (if no more owner elements chained) is released.

At EOJ time (/& or // JOB statement processing) all resources still owned by the partition are freed via UNLOCK ALL,JC=EOJ.

LOCK MANAGER INTERNALS

Entry Points

SVC110	LOCK / UNLOCK
SVC63	USE
SVC64	RELEASE

LOCK / UNLOCK Input Registers

Reg. 0	any parameter flags (stored to LOCKPARM)
Reg. 1	DTL address

Exit

DISP	exit to dispatcher
ERR1E	I/O error on lock file
ERR21	invalid parameter list format
ERR25	invalid parameter list limits
ERR2E	possible deadlock
RESVC or RESVCX	if lock manager in use or resource already locked

Permanent Usings

Reg. 1	DTL address	(DTLADR)
Reg. 2	LOCKTAB entry pointer	(LOCKADR)
Reg. 6	dispatcher	(DISP)
Reg. A	save area pointer	(SVEARA)
Reg. B	base register	
Reg. C	owner element pointer	(LOKOADR)
Reg. D	base register	

Note: Refer to "Lock Management Areas (DTLADR, LOCKADR, LOKOADR, DLFADR)" on page 529.

Lock Manager Flags

Label	Flag	Description	Value
LOCKPARM		Flag - lock/unlock parameters (in register 0)	
	UNLSYS	UNLOCK JC+SYSID is specified	X'40'
	UNLALL	UNLOCK ALL is specified	X'20'
	UNLEOJ	Request from EOJ routine	X'10'
	WAITCFLG	FAIL=WAITC (conditional)	X'08'
	WAITUFLG	FAIL=WAIT (unconditional)	X'04'
	LOCKSVC	LOCK (SVC110) or USE (SVC63)	X'02'
	NEWLOCK	LOCK/UNLOCK (SVC110)	X'01'
UNLCKFLG		Flag - unlock SVC (UNLOCK)	
	BLKMODF	External block modified (write back)	X'10'
	WAKEUPE1	Activate E1 requestors	X'08'
	FREELE	Give up a LOCKTAB entry	X'04'
	FREEOE	Give up an owner element	X'02'
	WAKEUP	Activate waiting tasks	X'01'
DSHRFLG		Flag - for lock system task	
	LCKSYS	System task is active	X'80'
	LCKTIM	Timer request is already set	X'40'
	LCKREQ	Update on ext. file required	X'20'
	LCKRESVD	Disk drive reserved (lock file)	X'10'

Figure 42. Lock Manager Flags

Return Codes

Lock Return Codes

Return Code		Flag	Description
Dec	Hex		
0	0		Request executed successfully
4	4		Resource owned by other task
8	8	ERRINTSP	LOCKTAB space exhausted
12	C	ERRINCON	Resource request inconsistent with present lock status
16	10	ERRDEL01	Deadlock
20	14	ERRDTLFO	DTL format error
24	18	ERRDEL02	Already locked by issuing task (deadlock)
28	1C	ERREXTSP	Space exhausted on external lock file
32	20	ERRNOVOL	Volume not mounted
36	24	ERREXTIO	Irrecoverable error on external lock file

Figure 43. Lock Manager Return Codes (LOCK Macro)

Unlock Return Codes

Return Code		Flag	Description
Dec	Hex		
0	0		Request executed successfully
4	4		Resource is not locked for the issuing task/partition
8	8		DTL format error

Figure 44. Lock Manager Return Codes (UNLOCK Macro)

DEADLOCK DETECTION

Assume that task T1 requests a resource, say RES1, which is already locked.
The owner chain of RES1 is scanned for owners who prevent T1 from locking this resource. If T1 itself is an owner of RES1 then a dead lock is detected.
The RESOURCE-BOUND owners (Task Status Byte, see Figure 311 on page 611) are entered into the dead lock test table (DLTT) and processed the same way as T1, owners that are not RESOURCE-BOUND are ignored.
This test is repeated for all entries of the DLTT (if there are any). Let's assume T2 is the first/next entry in the DLTT waiting for resource RES2. If T1 is an owner of RES2 then a dead lock is detected. The RESOURCE-BOUND owners are entered into the DLTT.
This testing is repeated until there are no more DLTT entries to be checked or until a dead lock is detected.

Deadlock Test via Deadlock Test Table (DLTT)

The DLTT contains as many 2-byte entries as the maximum number of tasks specified for supervisor generation. If deadlock test is performed, the DLTT entries will contain the TIDs of the lock-bound (RURBND - x'8E') tasks. The pointer to the resource (LOCKTAB) on which a lock-bound task is waiting, will be found in the TIBSTATE. If the last bit of TIBSTATE is on, the task will lock the resource exclusively (E1 request).

Notes:

E1 request:	CONTROL=E, LOCKOPT=1
E2 request:	CONTROL=E, LOCKOPT=2

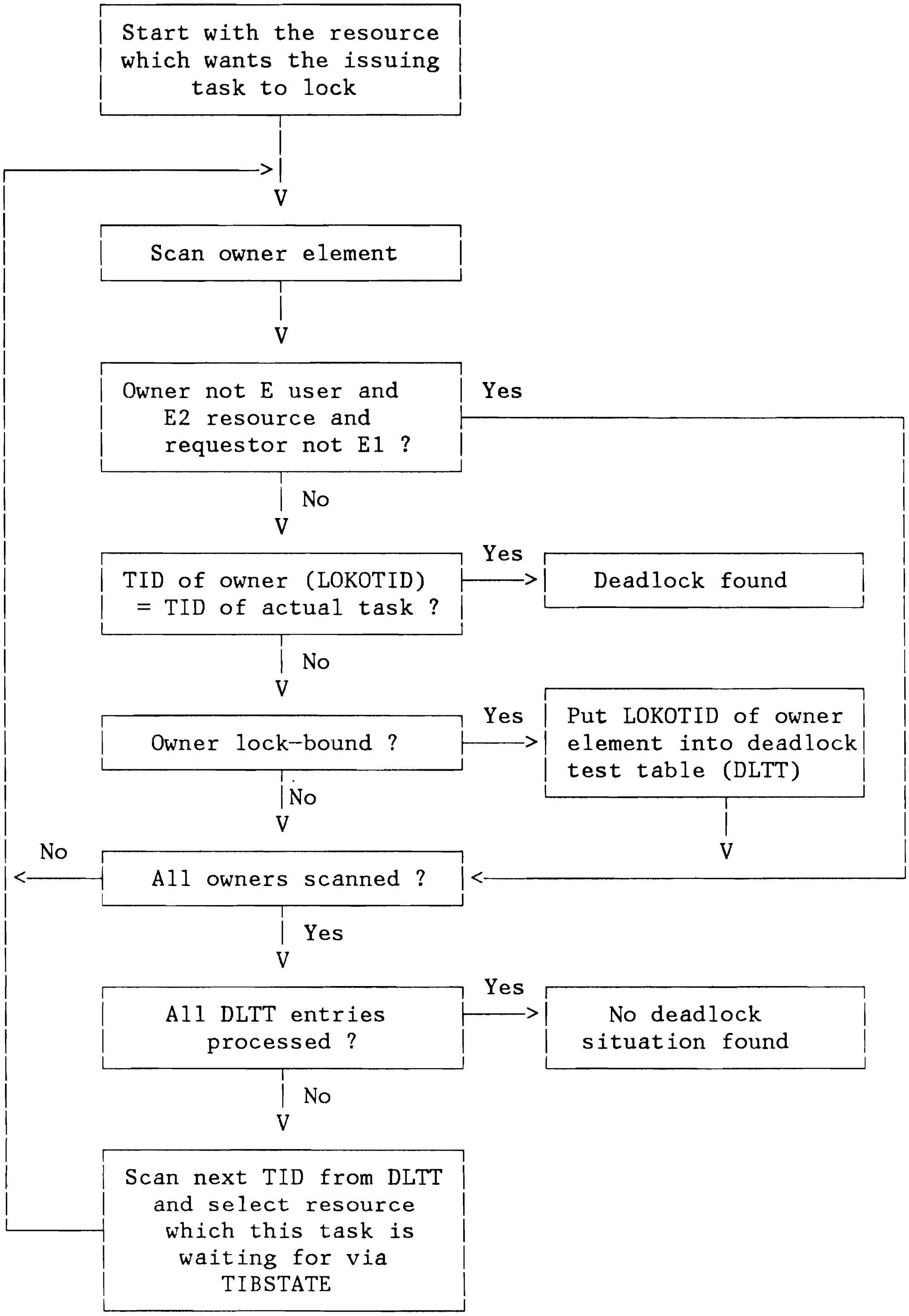

Start with the resource which wants the issuing task to lock
Scan owner element
Owner not E user and E2 resource and requestor not E1 ?
Yes
No
TID of owner (LOKOTID) = TID of actual task ?
Yes
Deadlock found
No
Owner lock-bound ?
Yes
Put LOKOTID of owner element into deadlock test table (DLTT)
No
No
All owners scanned ?
Yes
All DLTT entries processed ?
Yes
No deadlock situation found
No
Scan next TID from DLTT and select resource which this task is waiting for via TIBSTATE

Figure 45. Deadlock Test

Possible Deadlock Situations

1. External space is exhausted:

   - All resources of this block are owned by the issuing task or by a resource-bound task of this CPU (only deadlocks are detected which are caused by actions of one system).

2. Supervisor space is exhausted:

   - Waiting for a free owner element:
     - No owner element of this resource found, whose owning task is not resource bound (owner element has the TID of the requesting task).
   - Waiting for LOCKTAB space:
     - No LOCKTAB entry found, whose owners are all running (no owner is resource bound). Every LOCKTAB entry has just one owner element where its owning task is waiting for.

3. Resource is already locked:

   - Locked by the issuing task itself:
     - Deadlock if E1 request.
     - Deadlock if resource already locked with E1 by the issuing task.
     - Deadlock if resource already locked with E2 by the issuing task.
   - Not locked by the issuing task:
     - Find deadlock situation via deadlock test table (DLTT). (See paragraph: Deadlock Test)

DASD SHARING (LOCK MANAGER)

When DASDSHR=YES is specified in the FOPT macro the Lock Manager ("SVC 110 (X'6E' - LOCK/UNLOCK)" on page 82) contains additional code for maintaining the external lock file. When resources are locked across systems, the resource name and some control information are entered into the external lock file to assign the resource to this CPU.

When an externally locked resource is unlocked, the lock entry is removed from the external lock file, to allow other CPUs to lock the resource.

Within the SVC 110 (X'6E') processing routine it is not possible to issue SVC instructions. Therefore, the external lock file processing is done by a special system task, the Lock-System-Task (LCK). The LCK-Task is activated when the SVC X'6E' processing routine wants to read from or write to the external lock file. For additional information see description of "LOCK and UNLOCK (SVC 110 - X'6E')" on page 139.

External Locking

An external communication area, the external lock file, reflects at any time to all the sharing systems the system-wide locking status.

The external lock file is a system file which is shared among all sharing systems. Any resource to be locked across systems is contained in this external lock file.

The communication between the sharing systems is established during IPL via the DLF (Define Lock File) command. The VSE system which is IPLed first creates the external lock file. The other systems refer to this already created lock file, when they join the sharing environment.

Lock File Format

The external lock file consists of a header block and data blocks. The header block contains a file description of the external lock file and information about the sharing CPUs. The data blocks contain the lock entries (resource name plus control information).

```
┌───/          /────┬────/     /────┬────/     /────┬────/
│ lock file header  │ data block 1  │ data block 2   │
│   min. 52 bytes   │  512 bytes    │  512 bytes     │      ....
│    (20 + N * 8)   │               │                │
└───/          /────┴────/     /────┴────/     /────┴────/
0                  512           1024             1536
```

Notes:

1. N = Number of CPUs
2. default 4 CPUs, max. 31 CPUs

Figure 46. Lock File Format

Header Record Format

The lock file header record starts with a 20 byte file description of the lock file. The fields of this file description are identical with the first 20 bytes of the DLF Table in the supervisor. (See also Dsect DLFADR, Figure 254 on page 533.)

This file description is followed by a list of the CPU IDs of the sharing systems. For any sharing CPU there is an 8-byte field containing two flag bytes and a 6-byte CPU identification.

```
┌─────────────────┬────────────────┬──/ ... /──┬────────────────┬──/
│ Identical with  │     CPU 1      │           │     CPU 4      │
│ first 20 bytes  │2 flag + CPU    │           │2 flag + CPU    │  ...
│ of DLF Dsect    │ bytes   ident. │           │ bytes   ident. │
│ (at DLFADR)     │ (at DLFCPUS)   │           │ (at DLFCPUS)   │
└─────────────────┴────────────────┴──/ ... /──┴────────────────┴──/
0                 20              28           44              52
```

Figure 47. Lock File Header Format

Lock File Data Blocks

The physical block length is 512 bytes for CKD devices. For FBA devices the physical block length equals the physical block length of the FBA device (presently always 512 bytes).

Each block contains a 2-byte identification field, a 2-byte count field and lock entries.

The identification field contains the characters 'LF' (Lock File) The count field contains the number of lock entries stored in this data block. The lock entries contain the 12-byte resource name and one lock byte for any sharing CPU (a minimum of 4 and a maximum of 31 bytes).

```
┌──┬──┬────────────────┬────────────────┬──/-...-/─┬────────────────┐
│LF│cl│lock entry 1    │ lock entry 2   │          │ lock entry E   │
│  │  │                │                │          │                │
└──┴──┴────────────────┴────────────────┴──/-...-/─┴────────────────┘
0   2  4                                                            512

   cl = Count of lock entries
        in this data block
   E  = Maximum possible number
        of lock entries
```

Figure 48. Lock File Data Block Format

```
┌────────────────┬────┬────┬────┬────┬─/ ... /─┬────┐
│resource name   │cpu │cpu │cpu │cpu │         │cpu │
│                │ 1  │ 2  │ 3  │ 4  │         │ N  │
│                │flag│flag│flag│flag│         │flag│
└────────────────┴────┴────┴────┴────┴─/ ... /─┴────┘
0                 12   13   14   15   16        12+(N-1)
```

Figure 49. Lock File Entry Format

Flag	Appr.	Description
x'00'		no locking
x'01'	S1	CONTROL=S LOCKOPT=1
x'11'	E1	CONTROL=E LOCKOPT=1
x'02'	S2	CONTROL=S LOCKOPT=2
x'12'	E2	CONTROL=E LOCKOPT=2
x'04'	S4	CONTROL=S LOCKOPT=4
x'14'	E4	CONTROL=E LOCKOPT=4

Figure 50. CPU N Flag

Lock File Block Capacity

The length of one lock entry depends on the number of sharing CPUs. The maximum number of lock entries which may be stored into one disk block is dependent on the number of sharing CPUs (max. 31) and on the data block length (presently always 512 bytes).

The number of sharing CPUs is restricted to 31.

```
     Example:

       Number of sharing CPU:                                       4
       Length of one lock entry
         (resource name length + no. of CPUs):                     16
       Length of available space in one data block
         (512 - (2 byte ID + 2 byte count)):                      508
       ------------------------------------------------------------
       Number of lock entries per data block
         (length of avail. space  DIV  length of lock entry):     31
```

Figure 51. Maximum Number of Lock Entries in One Data Block (ex. 4 CPUs)

Mapping of Locks into Disk Blocks

Locked resources are stored into the external lock file at random. A hashing algorithm maps the resource name into the disk block number. This is done to spread the lock entries evenly over the external lock file. Within the disk block, lock entries are stored on the next free place.
When a lock entry is deleted, the last lock entry is moved to the free place.

Hashing Algorithm

1. Compress the 12-byte resource name by two EXCLUSIVE OR instructions into a full word.
2. Divide this full word by the number of blocks in the lock file.
3. You will get the relative block number within the external lock file, if you use the remainder of this division and add one block (for the header record block).

Example

```
   Ex.:  Look for resource "LOCKFILE001" and compute disk
         block number.

         Number of blocks in our example: X'25' (=DLFNBLK)


Resource name (12 bytes):
┌────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐
│ D3 │ D6 │ C3 │ D2 │ C6 │ C9 │ D3 │ C5 │ F0 │ F0 │ F1 │ 40 │
└────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┘
  L    O    C    K    F    I    L    E    0    0    1
|<──part 1───>|<──part 2───>|<──part 3───>|


Part 1:                 Part 2:                 Result 1:
┌────┬────┬────┬────┐        ┌────┬────┬────┬────┐     ┌────┬────┬────┬────┐
│ D3 │ D6 │ C3 │ D2 │  XOR   │ C6 │ C9 │ D3 │ C5 │  =  │ 15 │ 1F │ 10 │ 17 │
└────┴────┴────┴────┘        └────┴────┴────┴────┘     └────┴────┴────┴────┘
  L    O    C    K             F    I    L    E


Result 1:               Part 3:                 Result 2:
┌────┬────┬────┬────┐        ┌────┬────┬────┬────┐     ┌────┬────┬────┬────┐
│ 15 │ 1F │ 10 │ 17 │  XOR   │ F0 │ F0 │ F1 │ 40 │  =  │ E5 │ EF │ E1 │ 57 │
└────┴────┴────┴────┘        └────┴────┴────┴────┘     └────┴────┴────┴────┘
                               0    0    1


Result 2:               Number of blocks:       Remainder:
┌────┬────┬────┬────┐        ┌────┬────┬────┬────┐     ┌────┬────┬────┬────┐
│ E5 │ EF │ E1 │ 57 │  MOD   │ 00 │ 00 │ 00 │ 25 │  =  │ 00 │ 00 │ 00 │ 23 │
└────┴────┴────┴────┘        └────┴────┴────┴────┘     └────┴────┴────┴────┘


      1 header record block                   +                    1
                                              ──────────────────────

      Disk block number of lock entry (=DLFHBLK)                  24
```

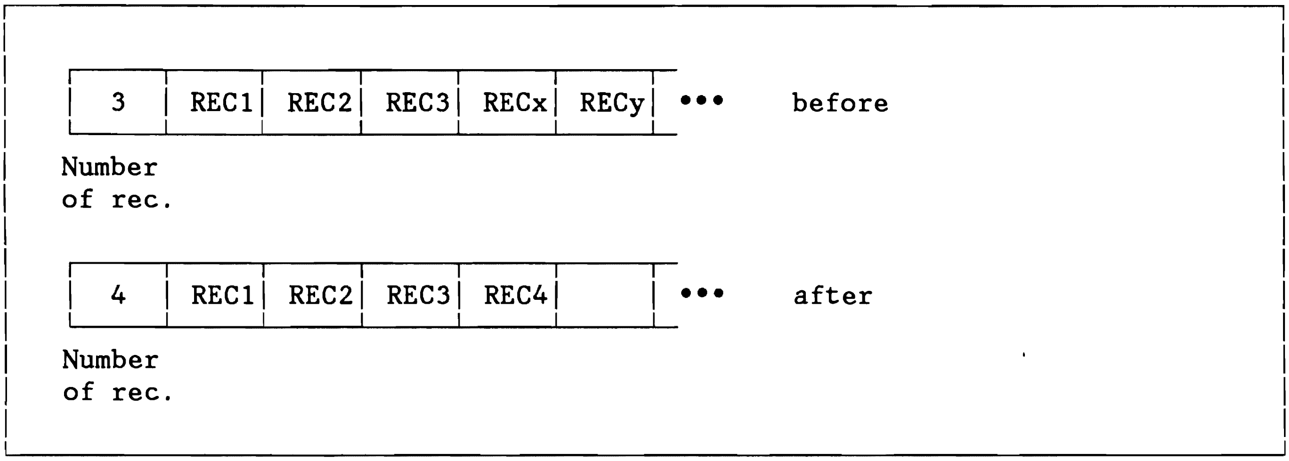
Figure 52. Mapping of Locks into Disk Blocks

Lock File Size

During IPL the lock file size is determined.

Lock Entry - Storing and Retrieval

Record insertion: New lock entries are entered into the first free place of the selected block (selected via hashing). Records within one block are not ordered.



Record retrieval: Scan the whole block to find the required lock entry.

Record deletion: When a lock entry is deleted, the last lock entry is moved to the free place (to keep the block 'dense').

Example: REC2 is deleted, REC4 is moved to the free place.

4	REC1	REC2	REC3	REC4	RECx	••• before

Number
of rec.

3	REC1	REC4	REC3	REC4	RECx	••• after

Number
of rec.

Fetch in a DASD Sharing Environment

For FETCH (Program Retrieval) in a DASD Sharing Environment see "DASD Sharing Environment" on page 285.

CHANNEL PROGRAM TRANSLATION (370 MODE)

The operations of the CHANNEL PROGRAM TRANSLATION routines depend on whether or not the fast CCW translation option (FASTTR in the macro FOPT) is active.

The first part of this section deals with the normal translation of channel programs, and the second part handles the additional functions and control blocks for fast CCW translation.

> **Note:** Whenever in this section (Channel Program Translation, 370 mode) a reference is made to a CCB (Channel Command Block), it also includes the IORB (Input/Output Request Block).

Normal Translation (FASTTR=NO)

The supervisor must do the following before initiating an I/O operation for a virtual-mode program:

- Copy the CCB and the entire channel program into copy blocks in the supervisor.

- Translate the addresses used by the CCB and the channel program into real storage addresses and place these addresses into the copied CCB and channel program.

- Build IDALs (Indirect Data Address Lists) for all I/O areas which cross one or more page boundaries.

- Fix all pages containing I/O areas in real storage for the duration of the I/O operation.

These functions are performed by the routine CCWTRANS. CCWTRANS is called by the channel scheduler every time a virtual-mode I/O request is made. For I/O requests from BTAM channel appendages this routine is entered at its entry point CCWTRBT2 (for further information, refer to "BTAM Considerations (370 Mode Only)" on page 121).

At the completion of an I/O operation, the routine CSWTRANS is called by the I/O interrupt handler. It must do the following:

- Retranslate the address of the last CCW pointed to by the CSW at channel end to its correct virtual address. This address is placed in the copied CCB.

- Free the data areas.

- Release the copy blocks used for the translation except the CCB copy block.

- Transfer the CCB information which has changed to the original CCB. If this is not possible (because the original CCB is not in real storage) indicate to the dispatcher that this must be done before the user task is given control again. In this case, the dispatcher calls a special routine (MOVECCB) to transfer the end of channel information from the copied CCB to the CCB in the user program.

Translation Control and Copy Blocks

The following control and copy blocks are used to copy and translate a CCB and channel program for a virtual-mode I/O request:

- A translation control block (CCWTCB). This is a work and save area, located in the task control block (TCB) and used during translation. The format of the CCWTCB is shown in Figure 235 on page 508 (Part 7).

- A CCB copy block. The user CCB and sense CCW (if any) are copied into this block. The CCB copy block also contains information about the copied and translated channel program.

- CCW copy blocks. Each block contains copy locations for up to 7 contiguous CCWs and queueing information.

- IDAL blocks used for building Indirect Data Address Lists for data areas which cross page boundaries.

- Fix information blocks containing the page frame numbers of pages freed for this request.

The Translation Control Block (CCWTCB)

Because a translation request may be interrupted (by a page fault, wait), it is necessary that the translation routine be partially reenterable so that several requests may be handled simultaneously.

The CCWTCB is located in the work area of the task control block (TCB) of the requesting task. The other blocks are 72-byte blocks located at the end of the supervisor. They are dequeued from the free copy block queue (pointed to by AFCB) as needed, and enqueued again when they are no longer needed by the requesting task.

If the queue of free copy blocks is empty when a request for a copy block is made, one of the following actions will be taken:

- If the request is from a BTAM appendage routine, the system will enter a hard wait (refer to "BTAM Considerations (370 Mode Only)" on page 121).

- If the requesting task is the only one using the CCW translation routines, it will be canceled (not enough copy blocks available to ever satisfy the request).

- If the request is for a CCB copy block or if at least one request has been handled successfully, the requesting task is set copy block bound.

If no other task is complete, and if the request is not for a CCB copy block, the used copy blocks are freed and the task is set translation bound. When another translation has been successfully completed, the request will be started again from the beginning.

CCB Copy Blocks

For each virtual-mode request one copy block is used to contain the copied CCB and its sense CCW, if any. The rest of the block contains control information about the translated program. Figure 53 on page 160 shows the layout of the CCB copy block.

If an Input/Output Request Block (IORB) is used for the request, bytes 0-15 (identical to a CCB) are set into the CCB copy block.

All the CCB copy blocks in use are queued in the queue pointed to by ACCBB. Each CCB copy block is also individually pointed to by a field in the request's TCB. After translation, the address of the copied CCB is placed in the channel queue. Figure 53 on page 160 shows the mutual and external relationships of the CCB copy blocks.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th></th></tr>
<tr><td>0</td><td colspan="2">CCBCNT</td><td>CCB
COM1</td><td>CCB
COM2</td><td>CCB
STA1</td><td>CCB
STA2</td><td>CCB
CLS*</td><td>CCB
LNO</td><td rowspan="2">A
Copied
CCB
|
V</td></tr>
<tr><td>8</td><td colspan="4">CCBCCW
Address of first CCW</td><td>CCBBY3</td><td colspan="3">CCBCSWW</td></tr>
<tr><td>16</td><td colspan="8">CCBSENS
Sense CCW if any</td><td></td></tr>
<tr><td>24</td><td colspan="2">TID
TASKID</td><td>CCB
Flag**</td><td>Unused</td><td colspan="4">CCBVA
Virtual address of CCB</td><td></td></tr>
<tr><td>32</td><td colspan="4">CCBACB
Address of first CCW copy block in channel program with lowest VBA</td><td colspan="4">CCBICB
Address of first IDAL block in channel program</td><td></td></tr>
<tr><td>40</td><td colspan="8">CCBXINF (Fix information)
Real page numbers of TFIXed pages</td><td></td></tr>
<tr><td>64</td><td colspan="4">CCBXPTR
Address of additional fix information block</td><td>***
X'80'</td><td colspan="3">CCBNEXT Address of next CCB copy block</td><td></td></tr>
</table>

Figure 53. CCB Copy Block

- * - Bit 2 is set (X'20') to indicate copied CCB

- ** - Legend CCBFLAG:

Bits Description

0: Indicates that CCW-translation of this request is complete; indicator is set before I/O request is enqueued in channel queue.

1: Indicates that control has been transferred to TFIX routine at least once during CCW translation; if 0, scan through CCBXINF for freeing pages is skipped; indicator is set immediately before control is passed to TFIX routine.

2: Reserved.

3: Indicates that the next CCW translation request from BTAM is from BTAM channel appendage. This indicator is set immediately after the first time a request from BTAM has been completed.

4: Indicates that the channel program is valid for fast CCW translation (CCWs are contiguous, the requestor is not BTAM and it is not a system task request with an I/O area in the SVA).

5: Indicates that this CCB copy block is on the saved CCB queue.

6: Indicates that the pages containing I/O areas for this channel program require fixing.
7: Reserved.

- *** - 'Block in use' indicator

  Note: If the fast CCW translation option is active, bytes 56-67 of the CCB copy block have a different meaning, as shown below:

Bytes	Description
56 - 59:	The address of the REPLICA block associated with this channel program.
60 - 63:	Pointer to the next CCB in the saved CCB queue used by the fast CCW translation routines.
64 - 67:	Pointer to the saved CCB queue.

The fix information normally held in these locations is not required when fast CCW translation is active, as it is kept in the REPLICA for the channel program.

CCW Copy Blocks

Each CCW copy block consists of 7 copy locations and 16 bytes for pointers and inserted TIC commands. The layout of a CCW copy block is shown in Figure 55 on page 163.

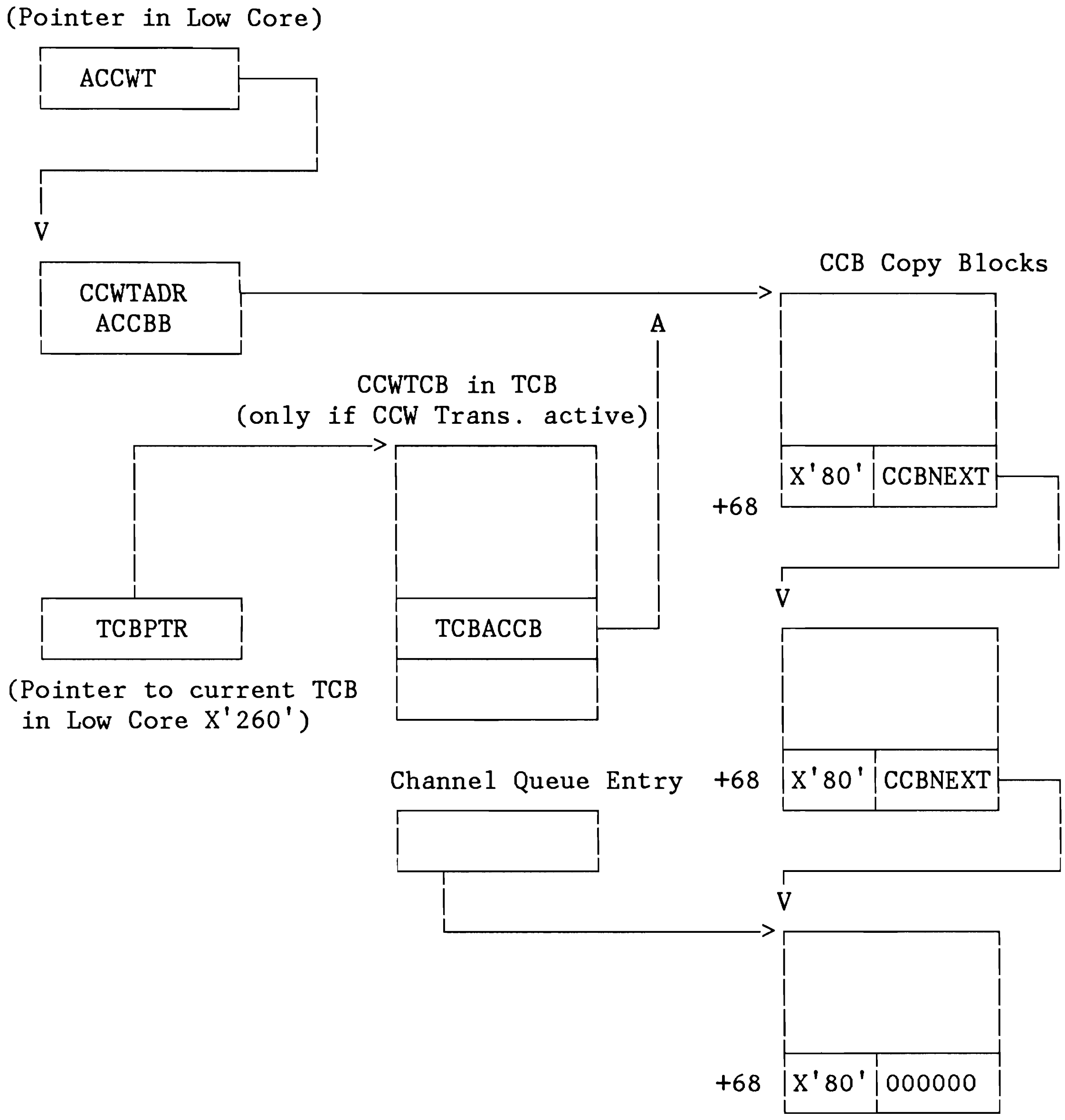


Figure 54. Locating CCB Copy Blocks

	0	1	2	3	4	5	6	7
0	1st Copy location for CCW							
8	2nd Copy location for CCW							
16	3rd Copy location for CCW							
24	4th Copy location for CCW							
32	5th Copy location for CCW							
40	6th Copy location for CCW							
48	7th Copy location for CCW							
56	X'80' *	X'000000'			Virtual address of first CCW in copy block (VBA)			
64	X'88' **	X'000000'			*** X'80'	Addr. of next CCW copy block in chain (ANB)		

Figure 55. CCW Copy Block

Notes:

1. * X'80' indicates the end of the CCW copy locations in the block. It is replaced by a TIC (Transfer in channel command) if the 7th copy location contains a copied CCW with data- or command chaining. Bytes 57-59 will then point to the copy location of the CCW following the CCW in the 7th copy location. Bytes 56-59 will not be changed if the CCW in the 7th copy location is a TIC.
2. ** X'88' indicates the last 8-byte entry in the block. It is replaced by a TIC if the CCW in the 7th copy location is a status modifier CCW. Bytes 65-67 will then point to the copy location of the second CCW following the status modifier CCW.

   The CCW copy blocks for a translation are queued in order of increasing VBAs (see Figure 55) with the lowest one being pointed to by the field CCBACB in the CCB copy block. Figure 56 on page 164 shows the relation of CCW copy blocks to one another.
3. *** X'80' 'Copy block in use' indicator

IDAL Blocks

CCWs whose data areas cross 2K boundaries must have an IDAL (Indirect Data Address List) in the copied channel program.

If a data area crosses a 2K boundary, the CCW is changed to show that an IDAL is used (bit 37 of the copied CCW is set) and the address of the IDAL is placed in the data address of the CCW. The IDAL pointed to contains one entry for the beginning of the data area and one entry for each 2K boundary crossed.

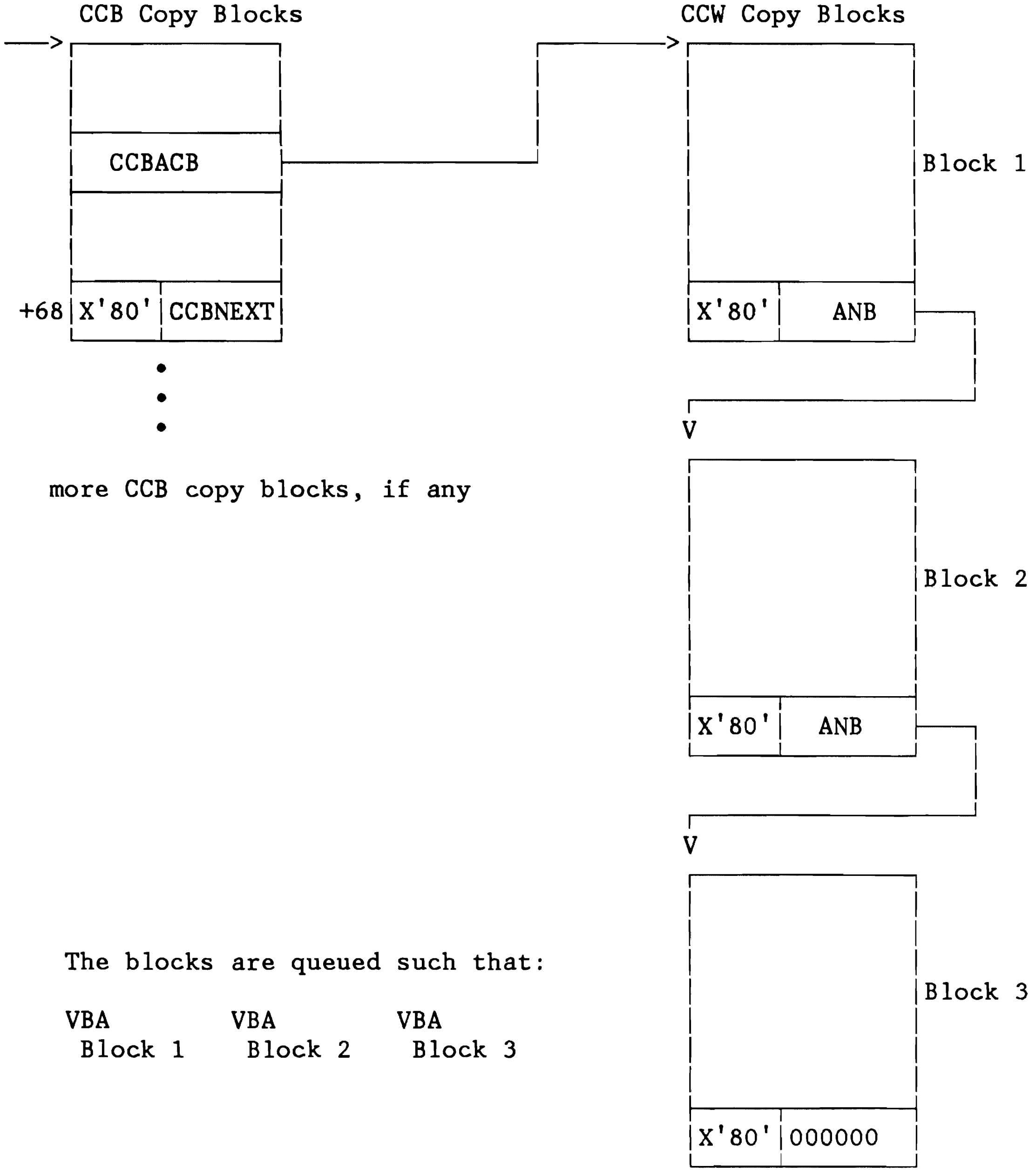


Figure 56. Locating CCW Copy Blocks

An IDAL must be located in consecutive copy block locations, so that if an IDAL cannot fit into the last block in the queue (the count in IDALCNT is less than the number required) a new block must be enqueued. For I/O areas with a length of less than 32K bytes a single copy block is dechained as IDAL block with 17 locations for Indirect Data Address Words (IDAWs). If the area is larger than 32K bytes two consecutive copy blocks are dechained from the free copy block queue. This double block has 33 locations for IDAWs.

After an I/O area has been TFIXed in real storage, the addresses in the IDAL are translated to point to the correct real storage locations (the begin address of the I/O area and the begin address of the page frames for the rest of the I/O area, or for a read-backward command, the end address of the I/O area, and the end address of the page frames).

Each IDAL is pointed to by the CCW which references it. In addition, the IDAL blocks are queued with the first one being pointed to by the field CCBICB in the CCB copy block. Figure 57 on page 166 shows the relation between the IDAL blocks and the other blocks.

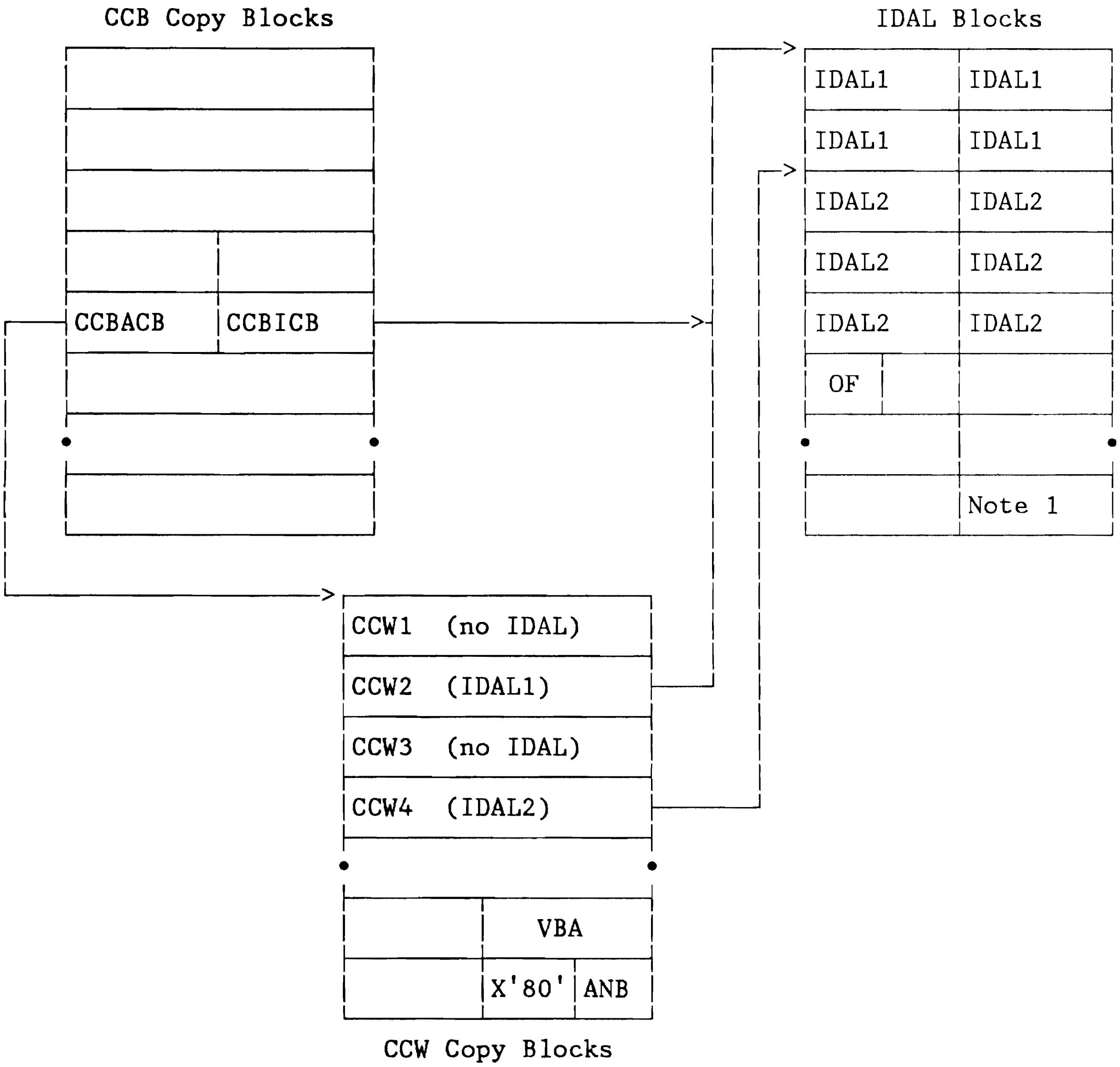
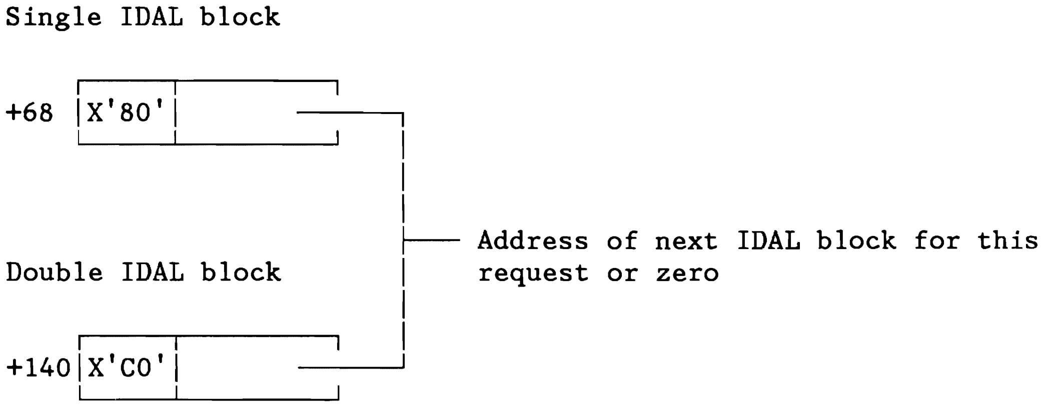


Figure 57. Relation of IDAL Blocks to other Blocks

Notes:

1.



the contents of X'C0' being:
X'80' Block in use
X'40' Double copy block

2. The X'0F' in the first byte of the 11th IDAW indicates the end of the IDAWs for the block. In this case, the IDALCNT field in the CCWTCB would show seven free copy locations.

3. The data area of CCW2 crosses three 2K boundaries (may be up to 8K) and the data area of CCW4 crosses five 2K boundaries (may be up to 12K).

Fix Information Blocks

In order to keep track of which page frames have been TFIXed for a request, the real page frame numbers of the pages fixed are kept in the copied CCB at label CCBXINF. If more than six pages have to be TFIXed for the I/O request, additional copy blocks are used. They are queued with the first one being pointed to by CCBXPTR in the copied CCB.

A page used more than once by a request is only TFIXed once.

Copying and Translating Channel Programs

User channel programs are copied into the copy blocks described in the previous section by the routine CCWTRANS (entered at CCWTRBT2 for BTAM channel appendage I/O request).

By way of initialization, the following is done before the actual copying and translation is begun:

- The CCWTCB for the requesting task is initialized. As part of the initialization procedure, the TCB pointers to the two special command lists for the device are filled in (see Figure 58 on page 169).

- Two copy blocks are dequeued from the free copy block queue for the CCB copy block and the first CCW copy block.

- The CCB is copied and initialized so that the CCW address points to the first location in the first CCW block. The VBA in the first CCW copy block is set to the virtual address of the CCW the virtual CCB is pointing to (which is the virtual address of the first CCW to be executed).

- If a sense CCW was present, it is also copied into the CCB copy block and its data areas are TFIXed in real storage (unless it crosses a 2K boundary, in which case an IDAL is built), and the address is translated.

The channel program is then copied and any necessary IDALs are built. The channel programs translated can be divided into three classes according to the types of commands they contain. They are described in the following order:

1. Channel Programs without TIC or Status Modifier Commands.

2. Channel Programs with TIC Commands.

3. Channel Programs with Status Modifier Commands.

A schematic representation of channel program translation is shown in Figure 59 on page 171.

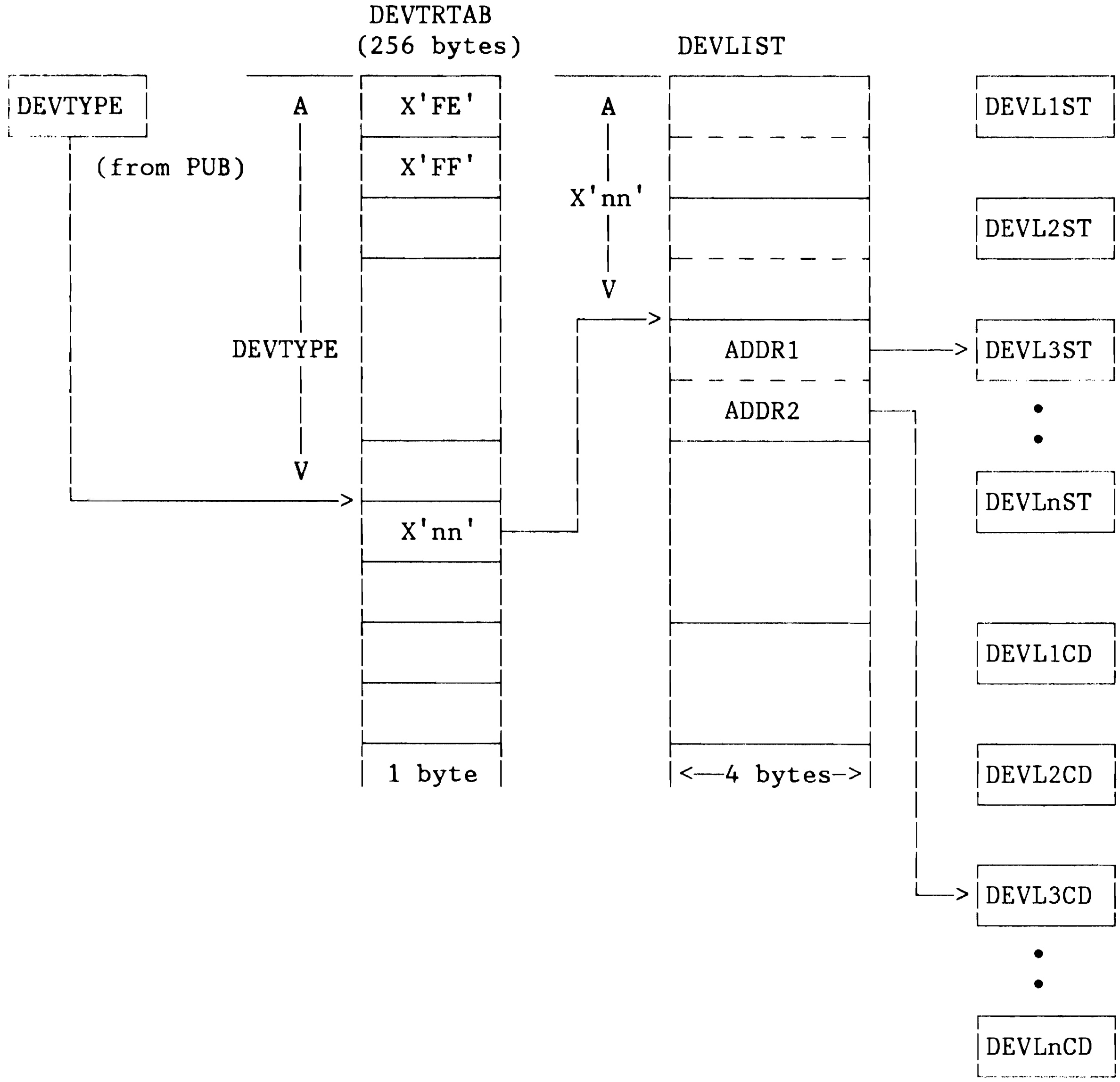


Figure 58. Initializing Special Command List Pointers in CCWTCB

DEVTYPE: Device type code from PUB

DEVTRTAB: Entries:

X'FF' = Unsupported device.

X'FE' = Device does not support status modifier commands or control commands with data area.

X'nn' = Displacement to entry in DEVLIST if device supports status modifier commands and/or control commands with data area.

DEVLIST: List of pointers to the special command lists. The two entries (if any) for the device on which the I/O is requested are moved to the TCB when this is initialized.

DEVLnST: Status modifier command list for device type n.

DEVLnCD: Control command with data area list for device type n (see note below).

Note: DEVLnST and DEVLnCD are bit strings. When a CCW is copied, the command code is used to refer to a bit in these strings. By testing this referred bit it is determined whether a CCW is a status modifier command or a control command with data area, or does not belong to these categories.

Copying Channel Programs without TIC or Status Modifier Commands

The first CCW in a channel program is always copied into the first copy location pointed to by the copied CCB. If command chaining or data chaining is specified in the CCW the following chained CCWs are copied into successive copy locations.

If a program of chained CCWs should contain 8 or more commands, a new CCW copy block must be used. The eighth copy location of the first copy block is then converted into a TIC command pointing to the first location of the next copy block. The VBA of the next copy block is set to the virtual address of the eighth chained CCW.

Figure 60 on page 174 is an example of a copied channel program containing 11 chained CCWs.

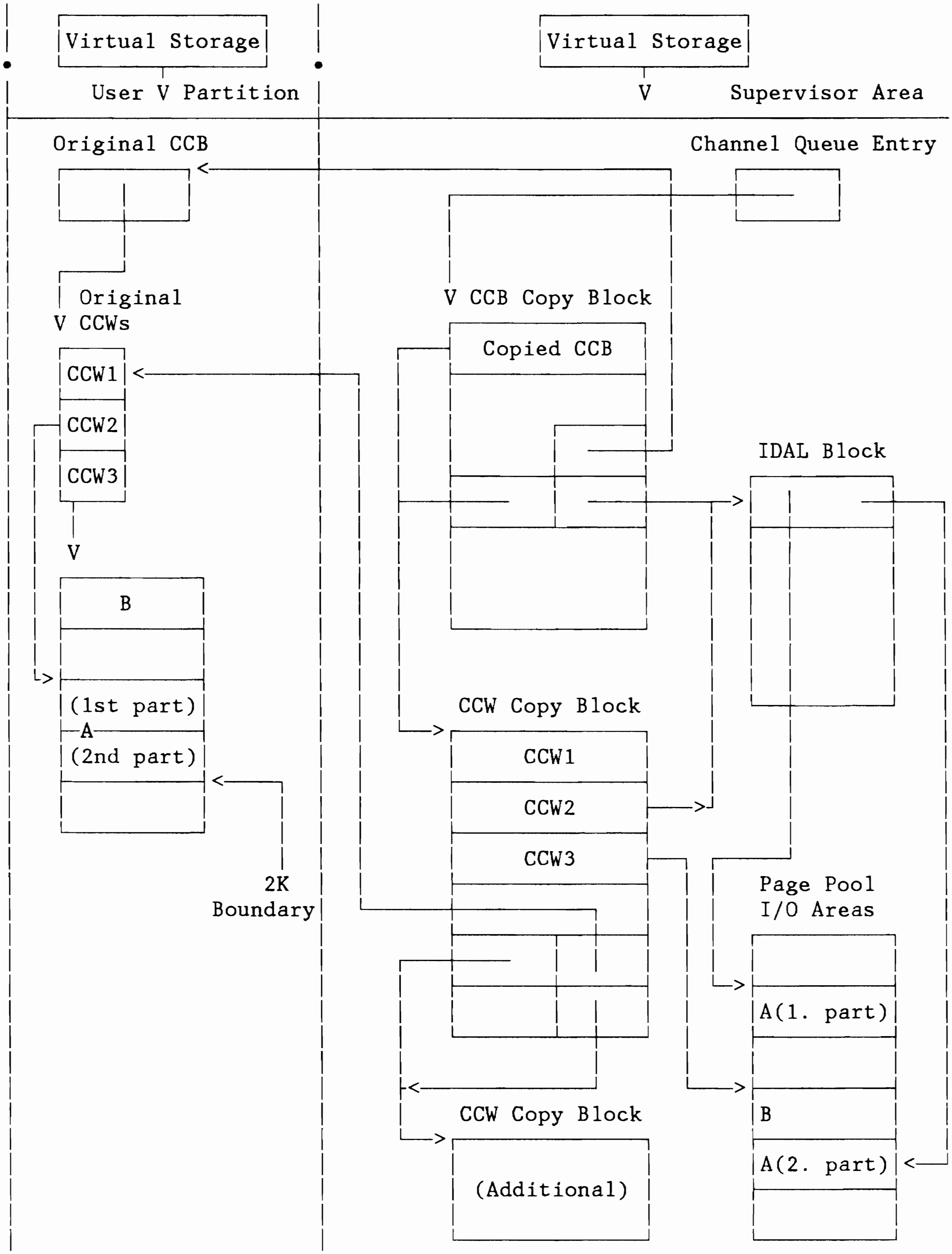


Figure 59. Schematic Representation of Channel Program Translation

Copying Programs Containing TIC Commands but no Status Modifier Commands

A TIC command (transfer in channel) command is, when encountered, copied into the next copy location just as any other chained command is. Although a TIC is 8 bytes long, only the first 4 bytes have any meaning (the command code and transfer address). The second four bytes of the copied TIC are set to zero. These bytes are used as a chain pointer for TICs which follow status modifier commands (refer to the section "Copying Status Modifier Commands"). The command code of a copied TIC is set to X'08' (standard user TIC).

The virtual storage location pointed to by the TIC command must be mapped into a location in the copied channel program. This mapped location is then placed in the copied TIC (unless the copied TIC is the first location of a copy block, in which case the address is placed in the end-of-block TIC (eighth copy location of the previous copy block) and used as the copy location for the CCW pointed to by the TIC. The mapped location is determined in the following way:

- If the CCW pointed to by the TIC command has a copy location in an existing copy block (that is, there is a block such that the virtual CCW address lies between the block's VBA and the block's VBA+56), place the location thus found in the TIC and copy the CCW in the location if it is free. If the location is not free, go to the translation termination routines. Figure 61 on page 175 is an example of a TIC which points to an already existing copy location.

- If there is no existing copy location, a new CCW copy block must be enqueued. The new block is enqueued at either end of the existing queue or between two existing blocks, depending upon where the virtual address in the TIC is in relation to the VBAs of the existing blocks. Figure 53 on page 160 shows how a new CCW copy block is queued to provide a copy location for a CCW pointed to by a TIC. Once enqueued, the VBA of the new copy block must be determined. If at all possible, the new block will be aligned to the one either above or below it (the VBA is 56 greater than the VBA of the lower block or 56 less than the VBA of the upper block). This is only possible if the address pointed to by the TIC lies within one of the ranges (that is, is less than 56 below the VBA of the above block or less than 112 above the VBA of the block chained below). If possible to align to both blocks the alignment is made to the lower block. Considering the example in Figure 62 on page 176 again it is copied in the fourth copy location.

- If it is possible to align the new block to both the upper and lower blocks but not to both at the same time (the difference between the VBAs of the two blocks is less than 112), a short block must be created by moving the end-of-block indicators to the copy location following the last logical copy locations. Figure 63 on page 177 shows how a short block is enqueued.

- If no alignment of the new block with either of its neighbors is possible, the VBA of the new block is made equal to the virtual address pointed to by the TIC and the first copy location in the block is used. Figure 64 on page 178 shows such a copy block being enqueued.

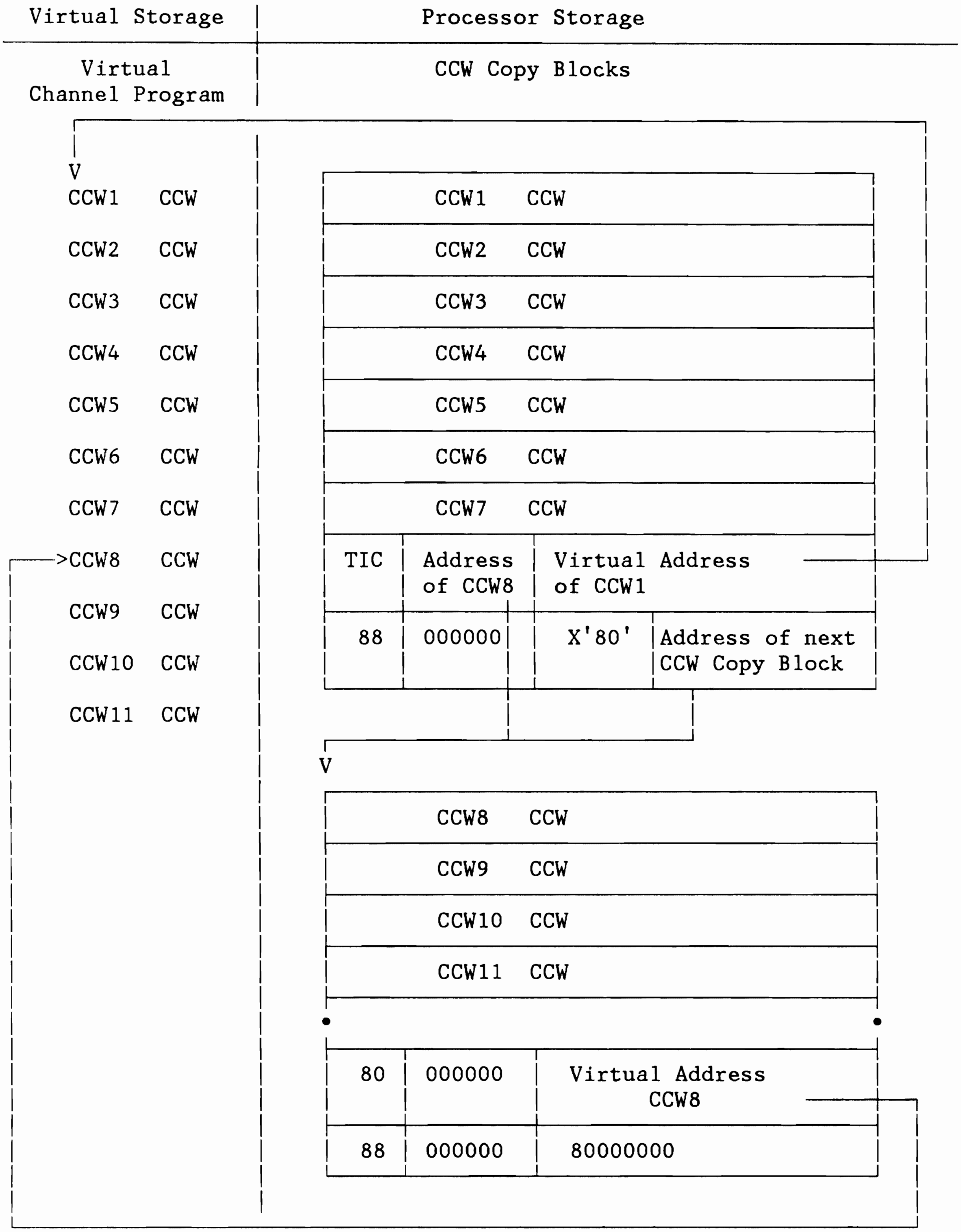


Figure 60. CCW Translation for a Channel Program. Without TIC or Status Modifier Commands.

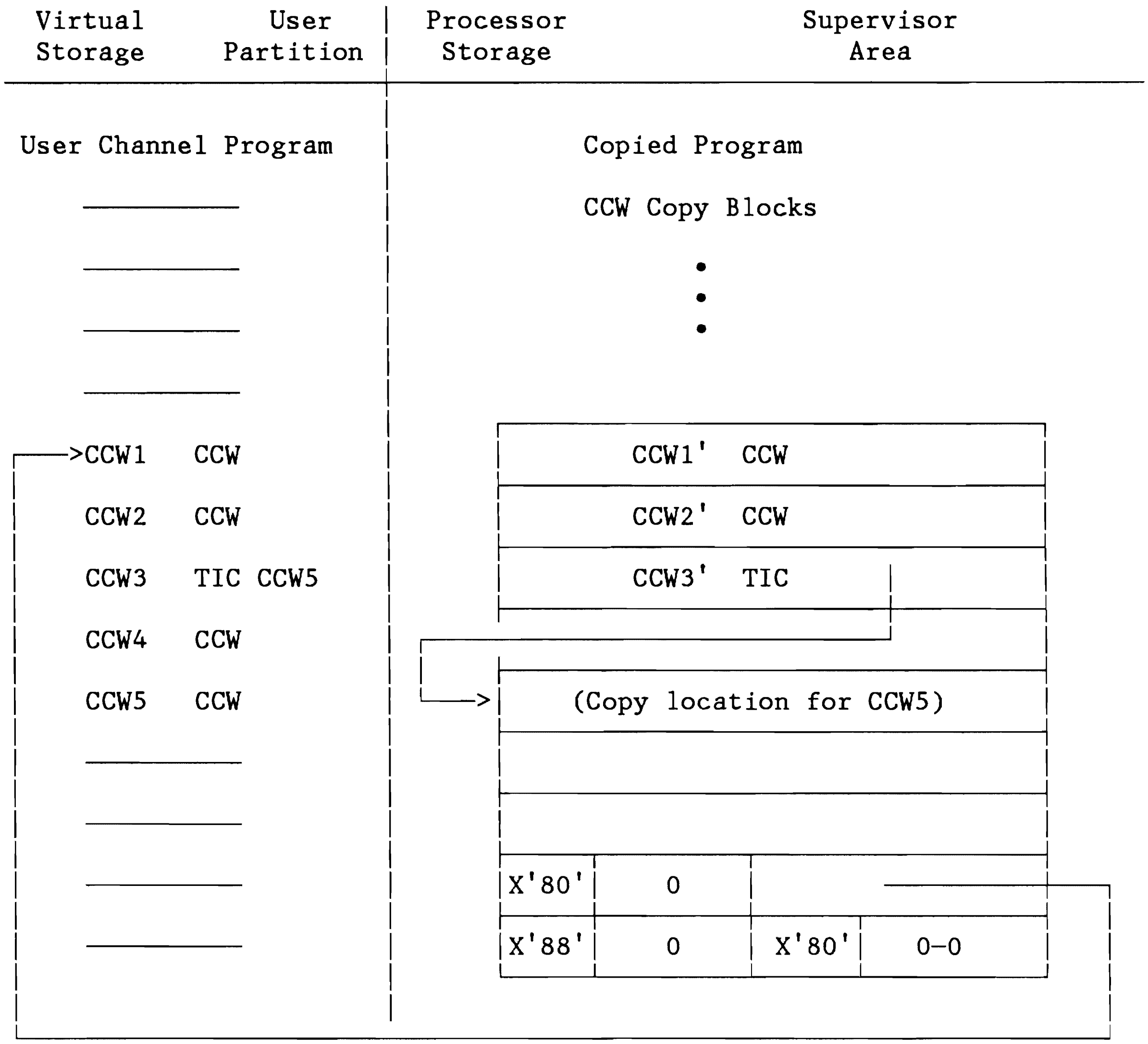


Figure 61. Copy Location for a CCW Pointed to by a TIC. If location is in already used copy block.

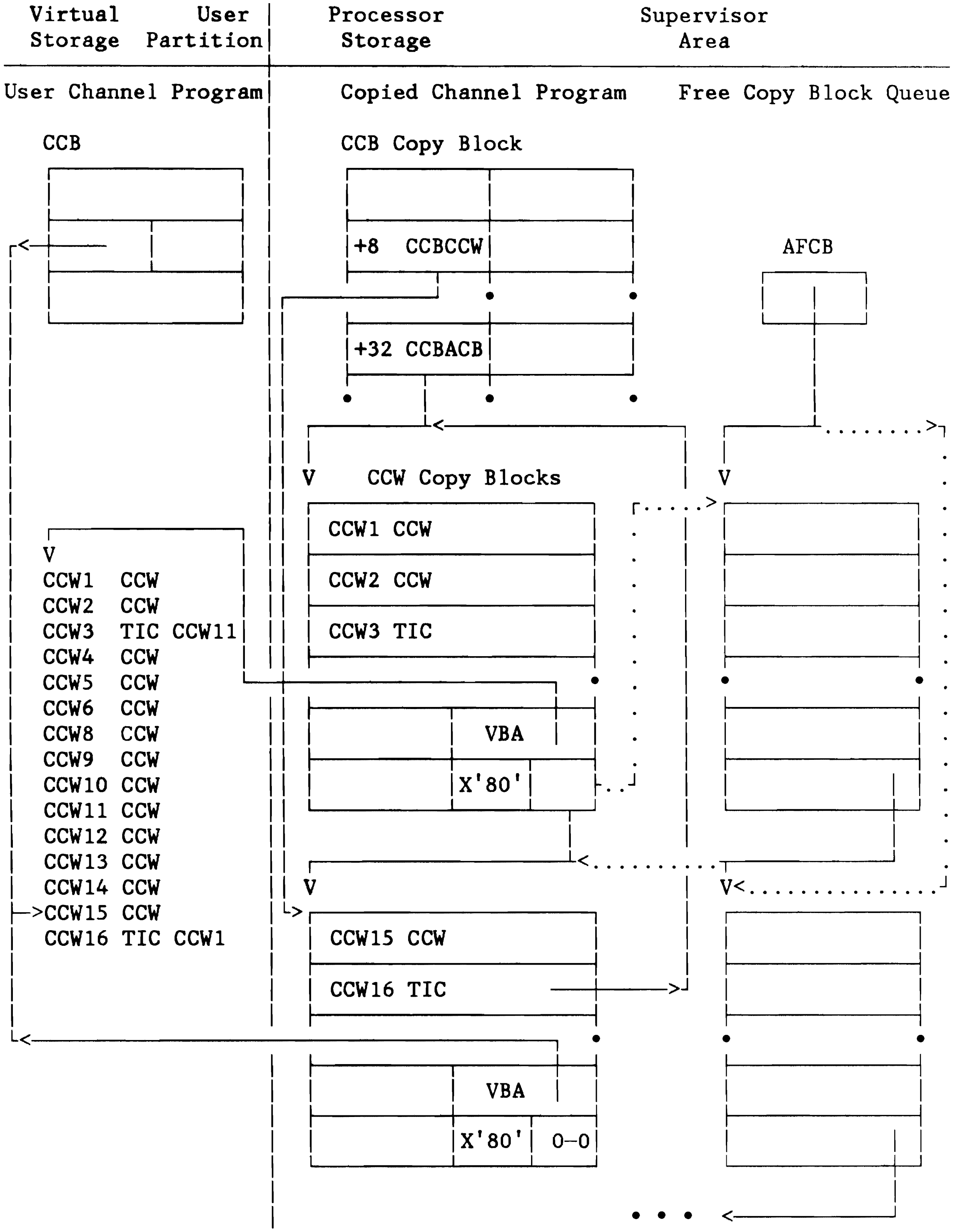


Figure 62. Enqueueing a New Copy Block. To the correct location in the CCW copy block chain to handle a CCW pointed to by a TIC (see Note 1).

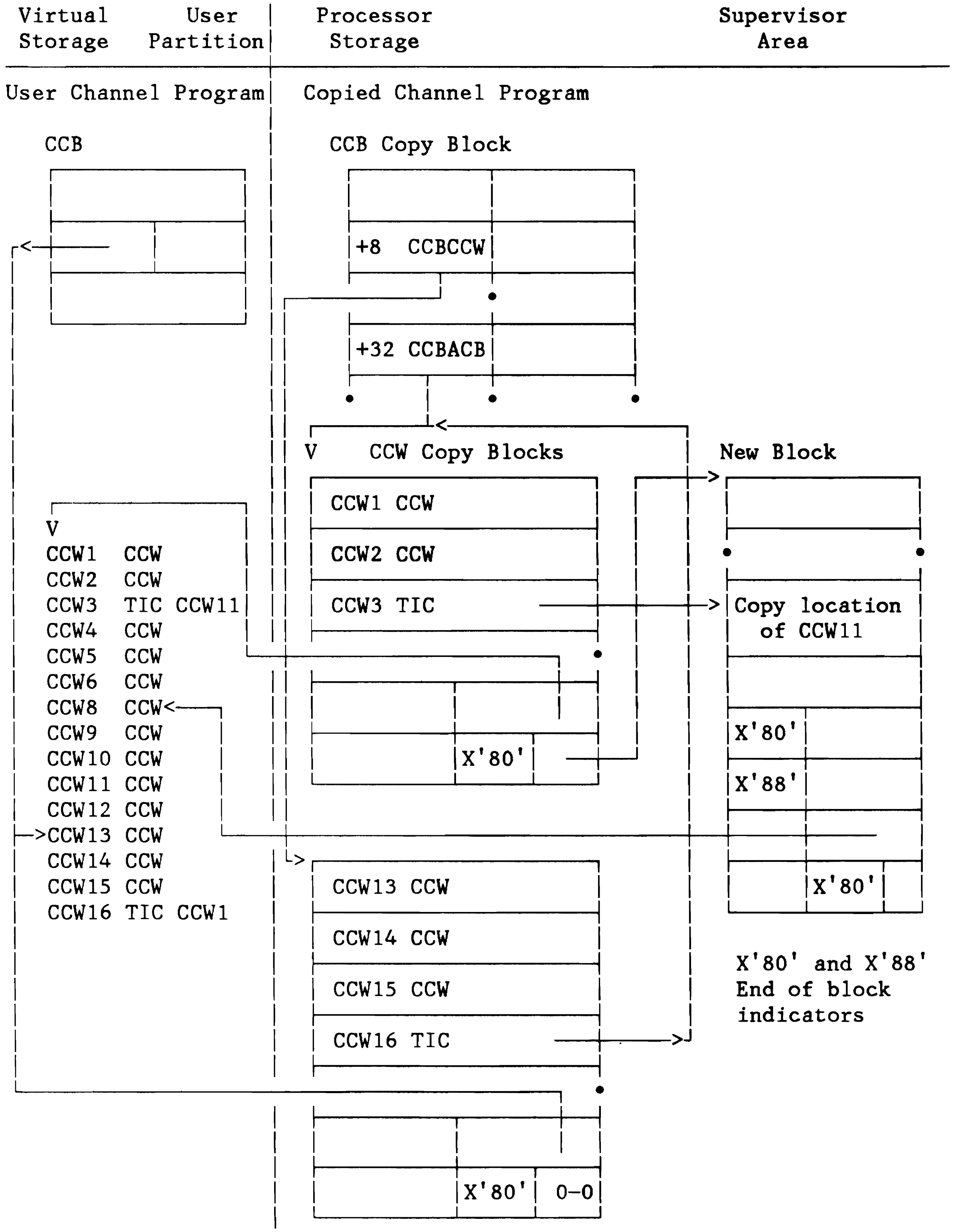


Figure 63. CCW Copy Block Queueing. Requiring the creation of a "short" block to maintain alignment (see Note 2).

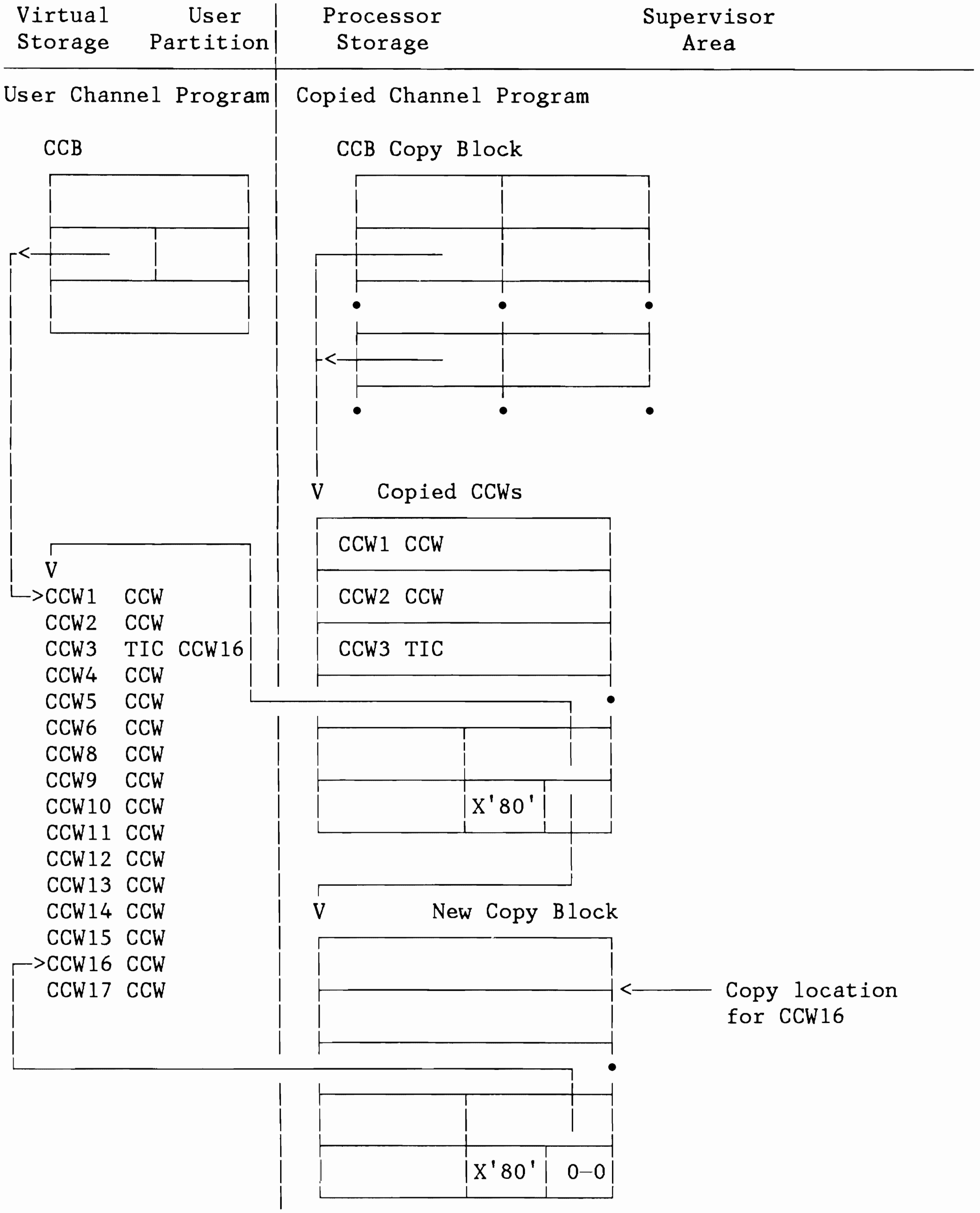


Figure 64. Enqueueing New Copy Block to Existing Block. Because the copy block cannot be aligned. CCW is too far removed from VBA of any existing block (see Note 3).

Notes:

1.

Problem — CCW3 has just been copied. The problem is to find the copy location for CCW11.

Solution — Free copy block is queued between A and B because the address used by the TIC at CCW3 lies between the VBA for A and the VBA for B. The solid line shows the condition before the new block is enqueued and the dotted lines the condition afterwards.

Once enqueued the VBA in the newly enqueued block will point to CCW8 (the block is aligned to the next lower block) and the TIC in CCW3 will point to the fourth copy location in the new block. Copying will then continue with CCW11 being copied into that location.

2.

Problem — CCW3 has just been copied and the copy block for CCW11 has been enqueued. The problem is to align the block.

Solution — Make the new block a 'short' block in that the end of block indicators are moved to the copy position following that for CCW12.

3.

Problem — CCW3 has just been copied and it is necessary to find a copy location for CCW16, the next CCW copied.

Solution — Enqueue a new copy block behind the first one and use the first copy location for CCW16 because it is impossible to align the new block to an existing block.

Copying Status Modifier Commands

Status modifier commands may transfer control to either of the next two following CCWs depending upon the result of the status modifier's operation. If, for example, a SEARCH command is unsuccessful, control is transferred to following CCW. If it is successful, on the other hand, the following CCW is skipped and control is passed to the second following command.

Consider the following chain of commands:

```
      READ
      READ
      SEEK
      SEARCH
      TIC   A
      READ
      READ
  A   WRITE
      WRITE
      SEARCH
      TIC   B
      READ
      READ
  B   READ
      READ
```

If the first SEARCH in this program is successful, no branch is taken as the TIC command is skipped. If the SEARCH is not successful the chained commands beginning at A are executed. The same is true when the second SEARCH is encountered. This can be done any number of times in a program. Since a program is copied as it is executed, the presence of status modifier commands makes it necessary to take several passes through a program in order to cover all the possible branches.

In the first pass through a program, a TIC following a status modifier command is copied but otherwise ignored (unless the status modifier is copied into the last copy location of a copy block). The TICs thus encountered are queued in a line pointed to by LINEPTR in the TCB (the queueing addresses are in the second 4 bytes of the copied TICs). Figure 65 on page 182 shows a program with status modifier commands after the first pass has been made a copying it.

If a status modifier command happens to be copied into the last copy location of the block, an entry in a different queue is made. This contains as entries the last locations of blocks where a status modifier command is copied into the last copy location. The first entry in the queue is pointed to by BENDPTR in the TCB. The queuing addresses are in bytes 1-3 of the queue elements (last location of the CCW copy blocks concerned). Copying continues with the first CCW following the status modifier command being copied into the first location of the next queued copy block, and, if chained, copying continues with the following command. If, as is usually the case, the first command after the status modifier command is a TIC, the branch taken by the TIC command is copied. Figure 53 on page 1604 shows a program with a status modifier command in the last copy position.

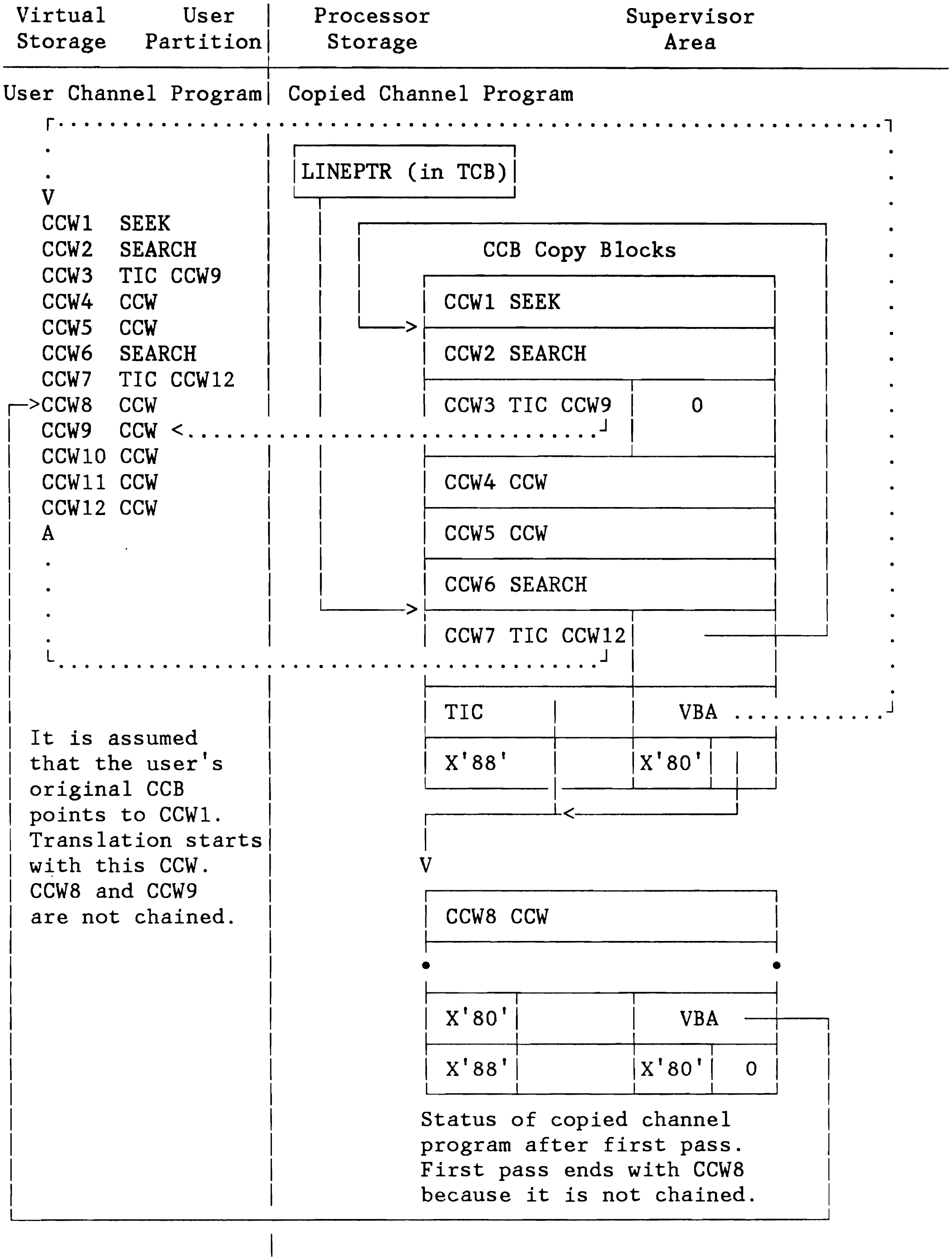
As soon as an end is reached in copying a program (a command without data or command chaining is copied or a copy location for a command is already filled) the program checks to see if there are any members in the queue pointed to by LINEPTR or BENDPTR. The members of these queues are handled one at a time. See Figure 66 on page 183 to Figure 68 on page 185.

> **Note:** LINEPTR and BENDPTR entries can be created while others are being handled. Translation is complete when both LINEPTR and BENDPTR are zero (that is, no more entries in either queue).

Translating Data Addresses and Page Fixing

Parallel to the copying of a channel program, the pages containing the data areas for the various CCWs are TFIXed in real storage and the virtual addresses of the data areas are translated into real addresses.

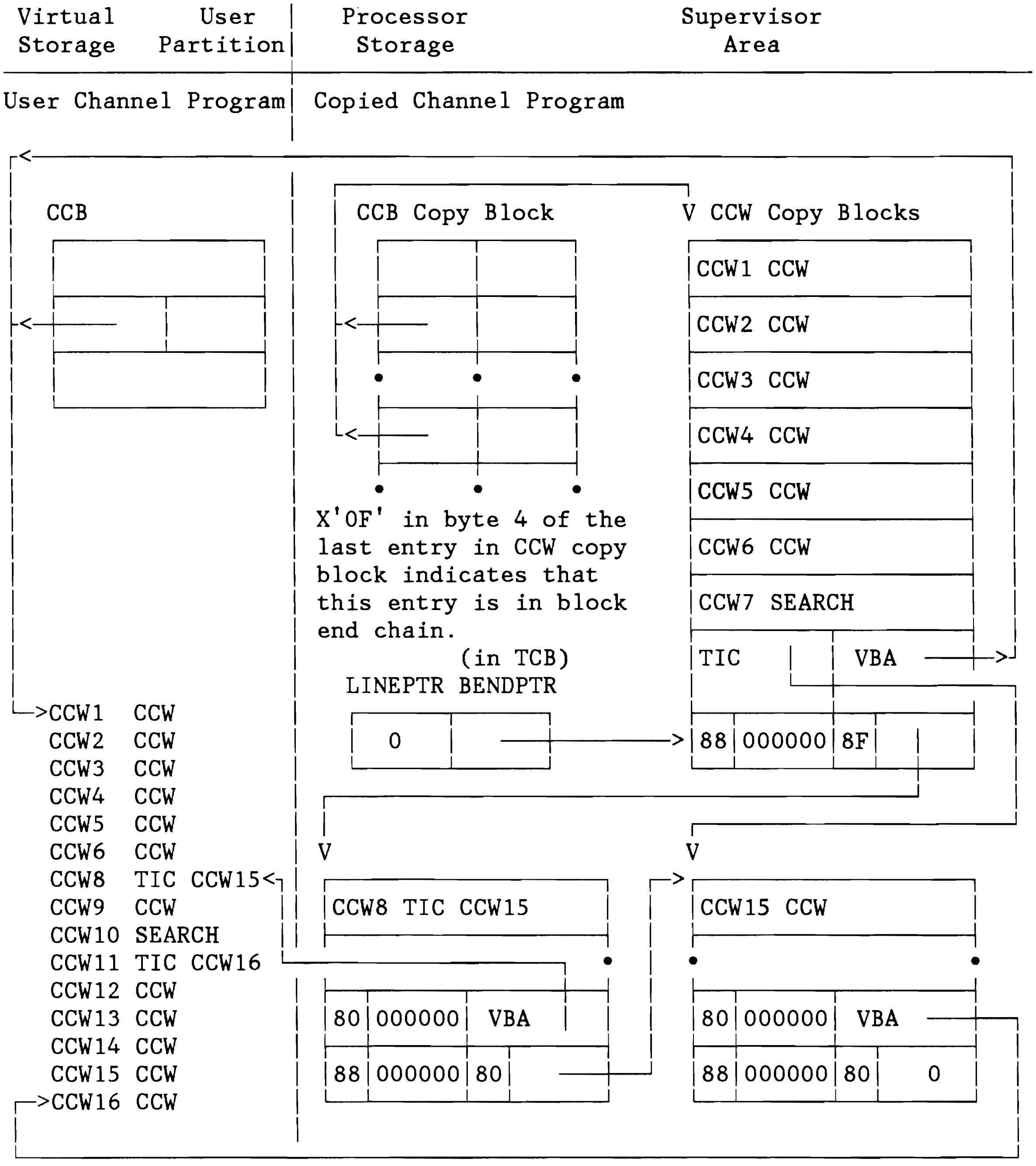
IDALs are first built using the virtual addresses of the beginning of the data area and the 2K boundaries. When the individual pages are TFIXed in real storage these addresses are replaced with the correct real addresses. Figure 69 on page 186 shows an IDAL built for a data area both before and after the pages have been TFIXed. Figure 70 on page 187 shows how the IDAL looks if the command is a read backward command.



It is assumed that the user's original CCB points to CCW1. Translation starts with this CCW. CCW8 and CCW9 are not chained.

Status of copied channel program after first pass. First pass ends with CCW8 because it is not chained.

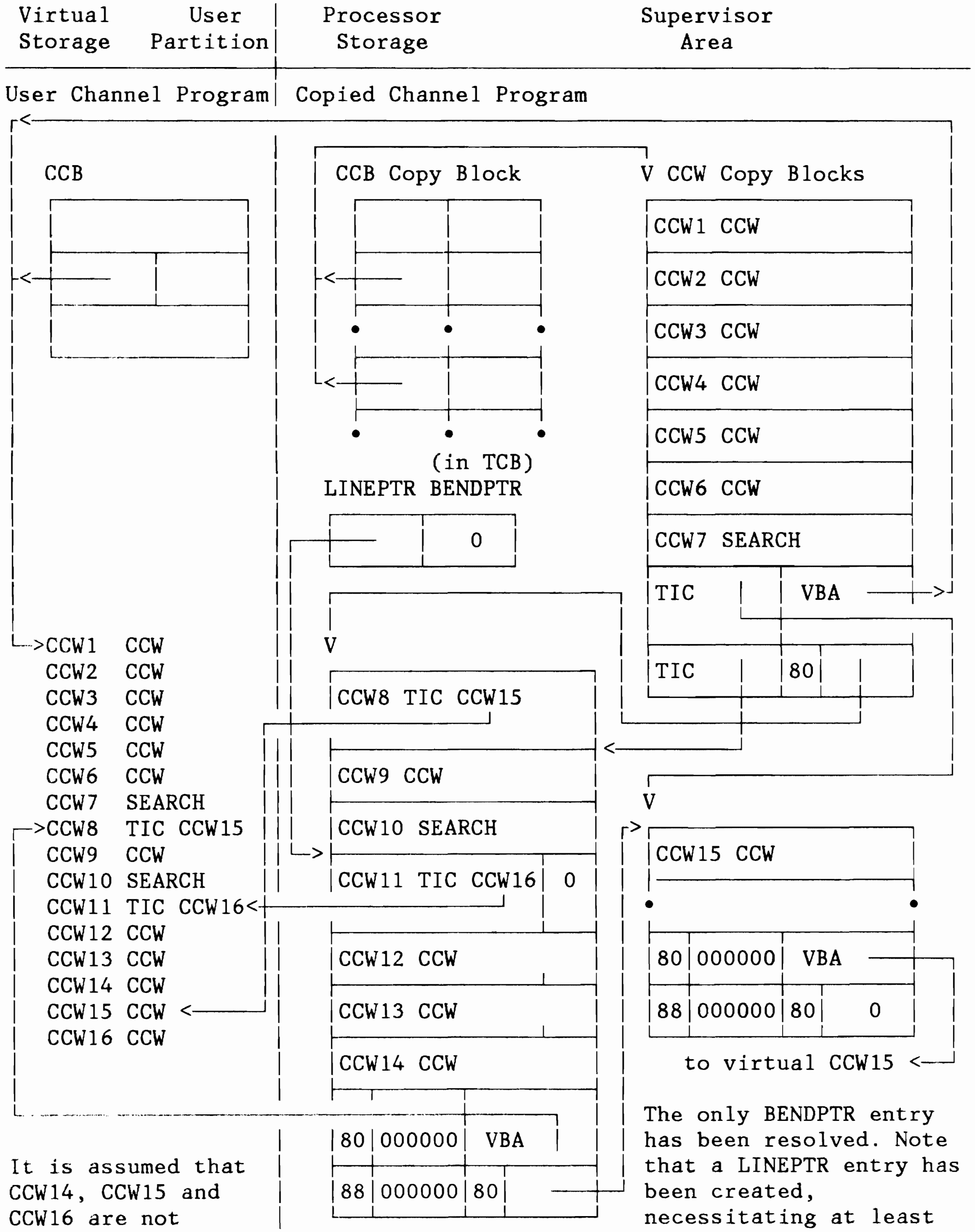
Figure 65. Channel Program. Containing status modifier commands after its first copying path has been made.



It is assumed that CCW14, CCW15 and CCW16 are not chained.

That status modifier command CCW7 is copied into the 7th copy position necessitating an entry into the BENDPTR queue. The first pass ends when CCW15 is copied, because this CCW is not chained.

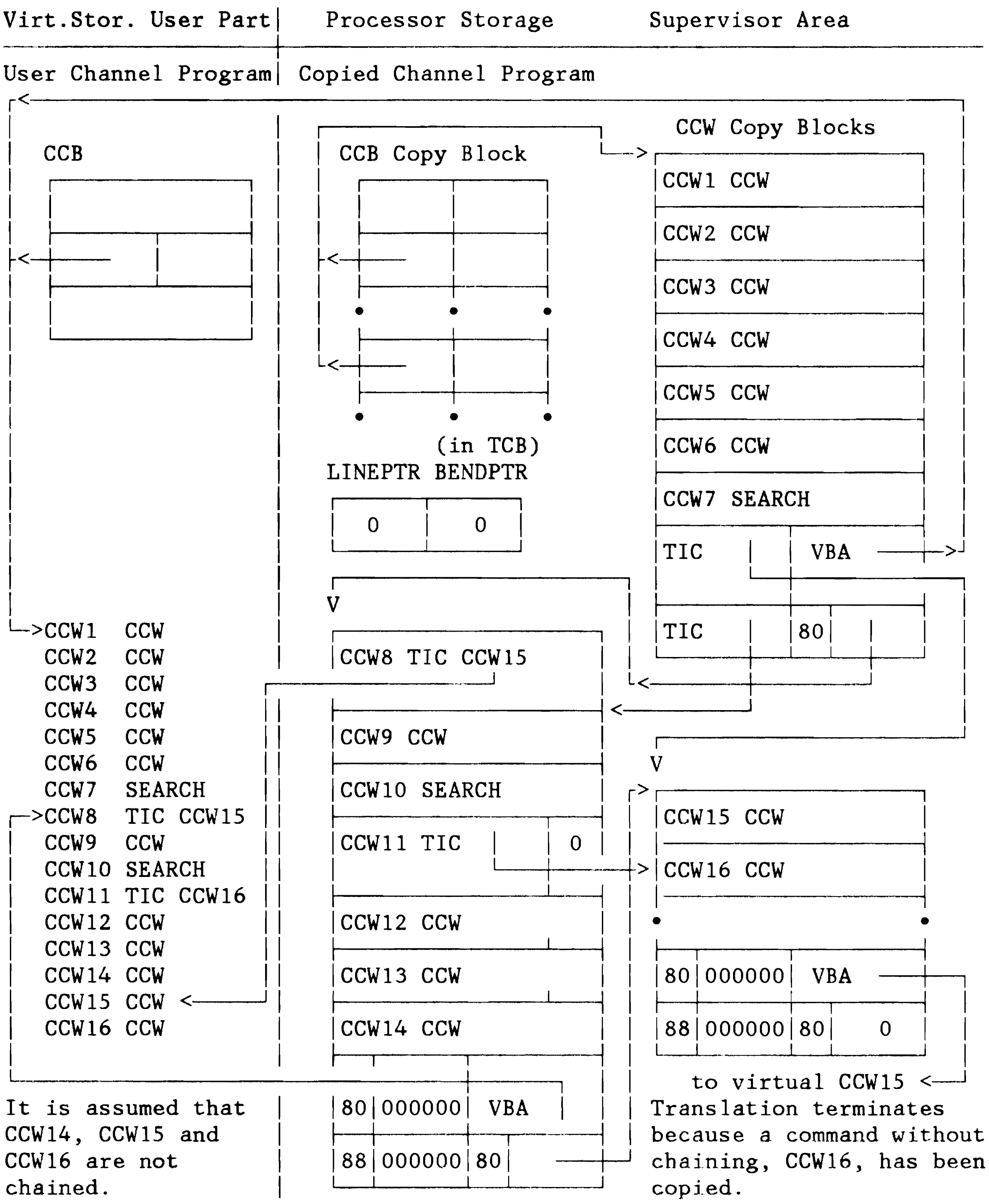
Figure 66. Channel Program. Containing status modifier commands after its first copying path has been completed.



It is assumed that CCW14, CCW15 and CCW16 are not chained.

The only BENDPTR entry has been resolved. Note that a LINEPTR entry has been created, necessitating at least one more pass to complete the copying of the program.

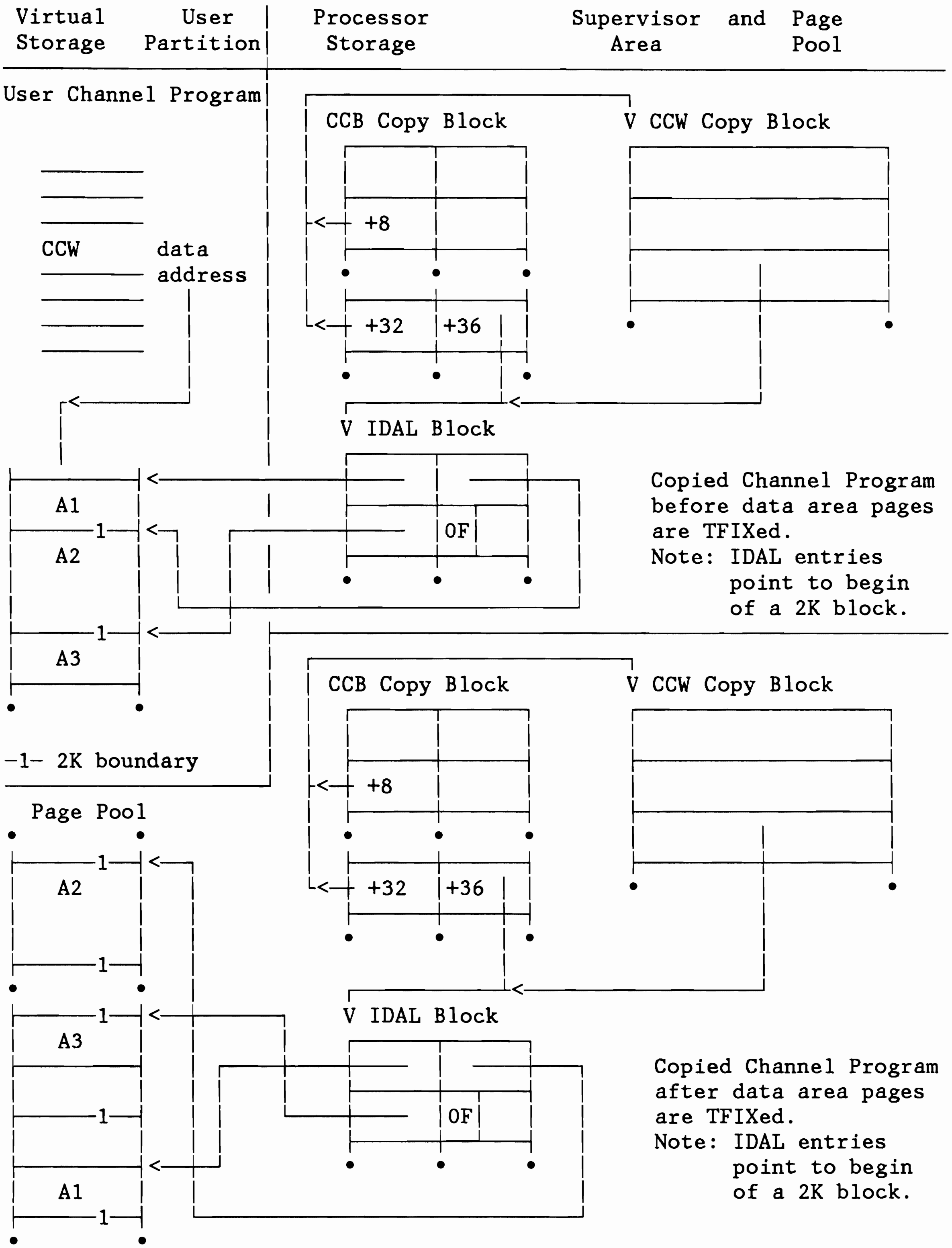
Figure 67. Channel Program. Containing status modifier commands after completion of the second path.



It is assumed that CCW14, CCW15 and CCW16 are not chained.

Translation terminates because a command without chaining, CCW16, has been copied.

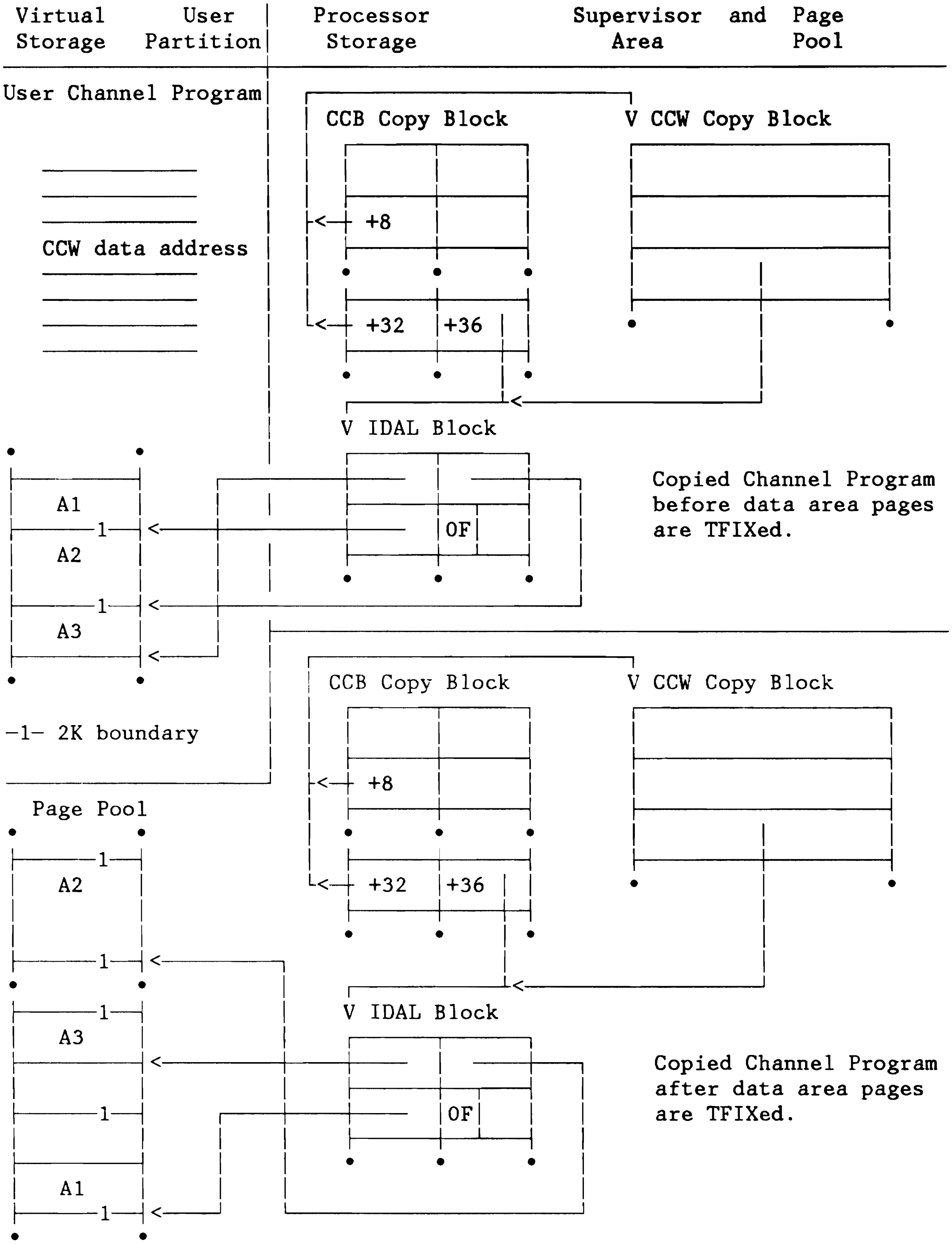
Figure 68. Channel Program. Containing status modifier commands after completion of translation.



Copied Channel Program before data area pages are TFIXed.
Note: IDAL entries point to begin of a 2K block.

Copied Channel Program after data area pages are TFIXed.
Note: IDAL entries point to begin of a 2K block.

Figure 69. Copied CCW. Requiring an IDAL to be Built (normal READ or WRITE command)



Copied Channel Program before data area pages are TFIXed.

Copied Channel Program after data area pages are TFIXed.

Figure 70. Copied CCW. Requiring an IDAL to be Built (READ Backward Command)

Fast CCW Translation (FASTTR=YES)

In order to save time when translating a series of similar channel programs, the translation routines attempt to save and reuse any channel programs that have already been translated, and to keep the pages containing the associated I/O areas fixed in real storage. This is done until the number of copy blocks in the copy block pool becomes insufficient and/or the paging rate becomes too high due to the large number of fixed pages.

In order to carry out these operations, the translation routines require two additional control blocks:

REPLICA	- A copy of a virtual channel program and its virtual CCB.
DIDAL	- A double-word indirect data address list which is used to locate the I/O areas in real storage.

These blocks and their formats are described in more detail later in this section.

Operation

When the fast translation option is active, the translation routine CCWTRANS first checks, after receipt of a channel program that is to be translated, whether there is a REPLICA of this program available. If so, the translation routine tests whether the pages containing I/O areas for the program are still fixed, fixes the pages again if necessary, and returns control to the calling routine.

If there is no REPLICA of the channel program available, normal translation takes place and the DIDAL blocks are built. The translation routine then checks if the channel program is valid for fast translation (the CCWs must be contiguous, no user IDALs are used and the request may not come from BTAM). If so, a REPLICA of the channel program is built and stored for future use.

A typical example of the resulting control block structure is shown in Figure 71 on page 189. After completion of the I/O request, the translation routine CSWTRANS simply retranslates the CSW command address, moves the CCB copy from the active queue to the top of the saved CCB queue, and transfers the necessary parts of the CCB copy to the virtual CCB. If, however, there is an insufficient number of copy blocks in the copy block pool, the routine frees the copy blocks of the least recently used CCB copy.

If the paging rate exceeds a given threshold, the I/O areas of the least recently used CCB copy with fixed pages are freed, but the copy blocks are retained.

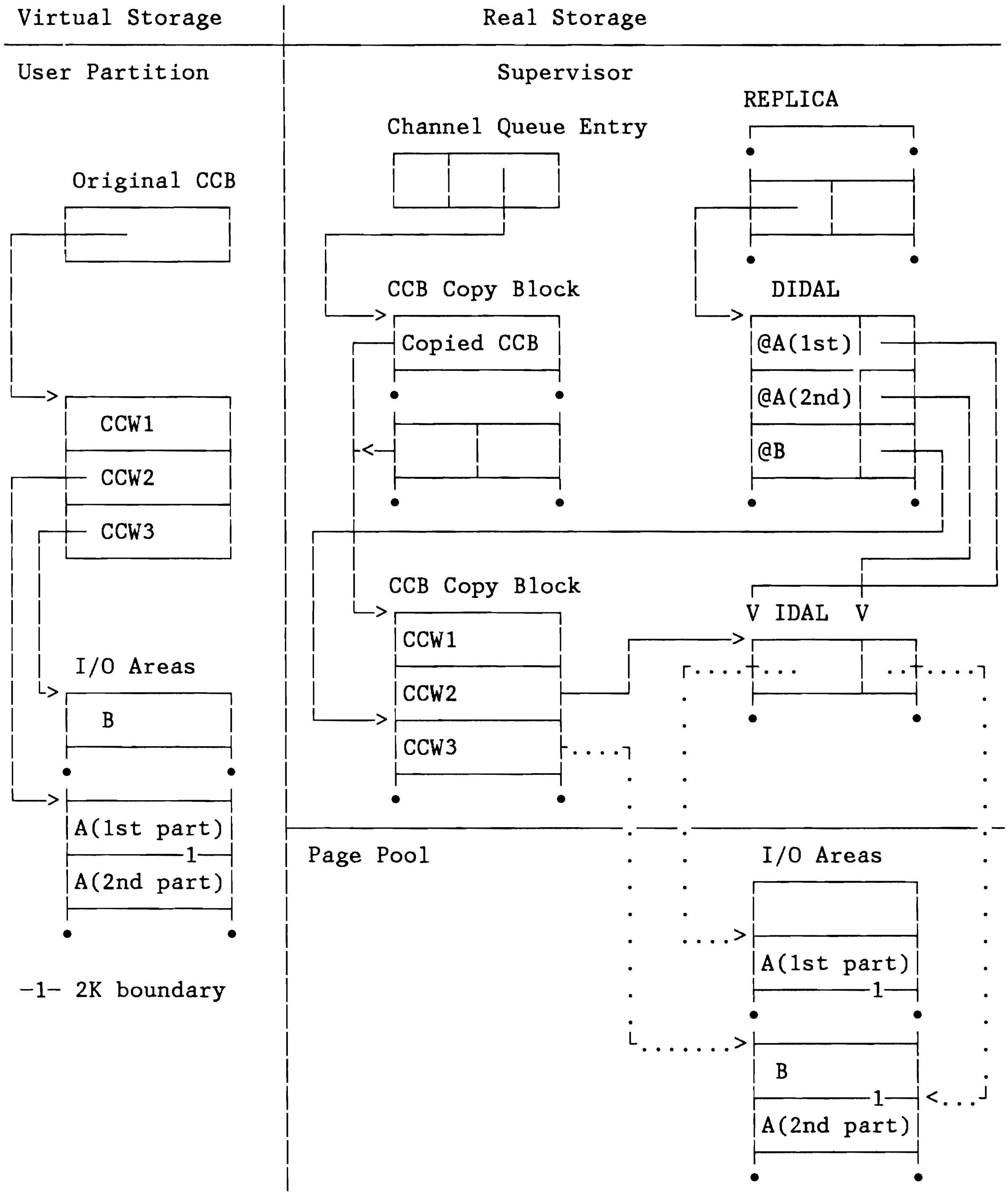


Figure 71. Control Block Structure for Fast CCW Translation

DIDAL Block

<table>
<tr><td colspan="2">ENTRY 1</td></tr>
<tr><td colspan="2">ENTRY 2</td></tr>
<tr><td colspan="2">ENTRY 3</td></tr>
<tr><td colspan="2">ENTRY 4</td></tr>
<tr><td colspan="2">ENTRY 5</td></tr>
<tr><td colspan="2">ENTRY 6</td></tr>
<tr><td colspan="2">ENTRY 7</td></tr>
<tr><td colspan="2">ENTRY 8</td></tr>
<tr><td>Reserved</td><td>CHAIN POINTER **</td></tr>
</table>

DIDAL Entry

1 3	4	5 7
Virtual address	Flag byte*	Pointer to real location ***

Legend:

- * Flag byte

Bits	Description
0:	Indicates that TFIXing is not necessary because the page has already been TFIXed for this request.
1-6:	Reserved.
7:	Indicates that the TFIX request for this entry has been completed.

- ** Pointer to (next) additional DIDAL. Contains X'80000000' in last DIDAL.

- *** Real location (copied CCW or IDA word) that should contain the translated I/O area address.

Figure 72. Doubleword Indirect Data List (DIDAL)

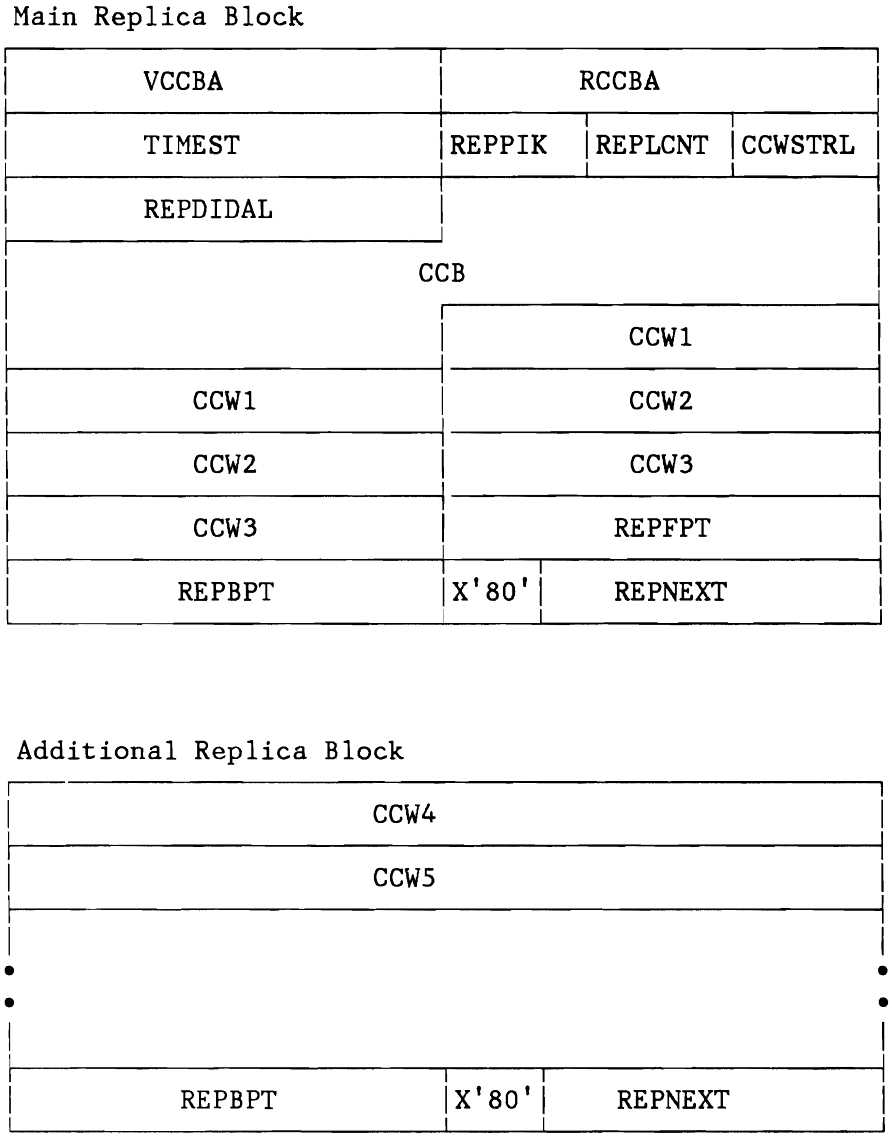


Figure 73. REPLICA Control Block

VCCBA	Virtual CCB address.
RCCBA	Address of copied CCB.
TIMEST	Timestamp.
REPPIK	Partition identification key.
REPLCNT	The number of tasks currently testing this replica for a match with their channel program.
CCWSTRL	Length of CCW string (number of CCWs).
REPDIDAL	Address of DIDAL block.
REPFPT	Forward pointer used for chaining REPLICAs.
REPBPT	Backward pointer, used for chaining REPLICAs.
REPNEXT	Pointer to (next) additional REPLICA block

Additional Control Blocks

The DIDAL block is created by the CCW translation routine in order to save the virtual addresses of the I/O areas and the addresses of the locations which contain the corresponding real addresses (CCW copy of IDAL block). The formats of the DIDAL block and its entries are shown in Figure 74 on page 194. Each DIDAL block occupies one or more copy blocks.

The replica block is created by the CCW translation routines and contains replicas, or copies, of the CCB and CCWs of a channel program. The first, or main, REPLICA block also contains additional header information. If there is insufficient space in the main REPLICA block, additional blocks, without the header, are added. Each main or additional block occupies one copy block. The formats of the main and additional REPLICA blocks are shown in Figure 73 on page 191.

Queue Organization

Similarly to normal channel program translation, the CCB copy blocks (for which I/O is or will be executed) are placed in the channel queue, which can be regarded as an active CCB queue. All pages containing I/O areas for CCBs in this queue are fixed.

After completion of the I/O operations for a given channel program, the CCB copy is placed in a second queue, called the saved CCB queue, and retained until it is needed again or is deleted. The pages associated with CCBs in this queue may or may not be fixed.

Each partition contains a replica queue which holds replicas of channel programs issued by the partition.

CHANNEL PROGRAM FIXING (ECPS:VSE MODE)

Before initiating any I/O operation in ECPS:VSE mode the supervisor must ensure that all the areas necessary for the I/O operation (channel program I/O areas) are present in storage, at least between SIO and the end of data transfer. This is ensured by the CCW fixing function which TFIXes the pages needed (the translation is done by the hardware). The following fixing function routines are called by other supervisor routines:

CCWEXCP Called by the channel scheduler before SIO, if the user has provided a CCB. CCWEXCP builds the fixlist by scanning the user channel program, decides which pages have to be fixed, inserts the appropriate page addresses in the fixlist, and fixes the pages defined by it.

CCWDOIO Called by the channel scheduler before SIO, if the user has provided an IORB. It transforms the user fixlist into an internal fixlist, and fixes the pages defined by it.

CCWFREE Called by the I/O interrupt handler after the I/O operation has been completed, to free resources (work blocks, page frames).

DELREPA Called by INVPAGE and the load leveler, if a minimum number of page frames is not available.

DEFIXALL Called by the load leveler, if a minimum number of page frames is not available.

DEFIXCON Called by TFIX/PFIX, if a minimum number of page frames is not available.

FAST FIXING SUPPORT

This function reduces the scanning and fixing overhead for EXCP requests that use the CCB, or an IORB with an unchained fixlist. Contiguous channel programs are copied and saved, and the storage areas fixed for them remain fixed beyond I/O interrupt time. The data areas for the next EXCP of the same channel program are then known and fixed, and can be used again.

Control and Work Blocks

Figure 74 shows the relationship of the control and work blocks for the CCW fixing function. The following paragraphs describe the control and work blocks in detail.

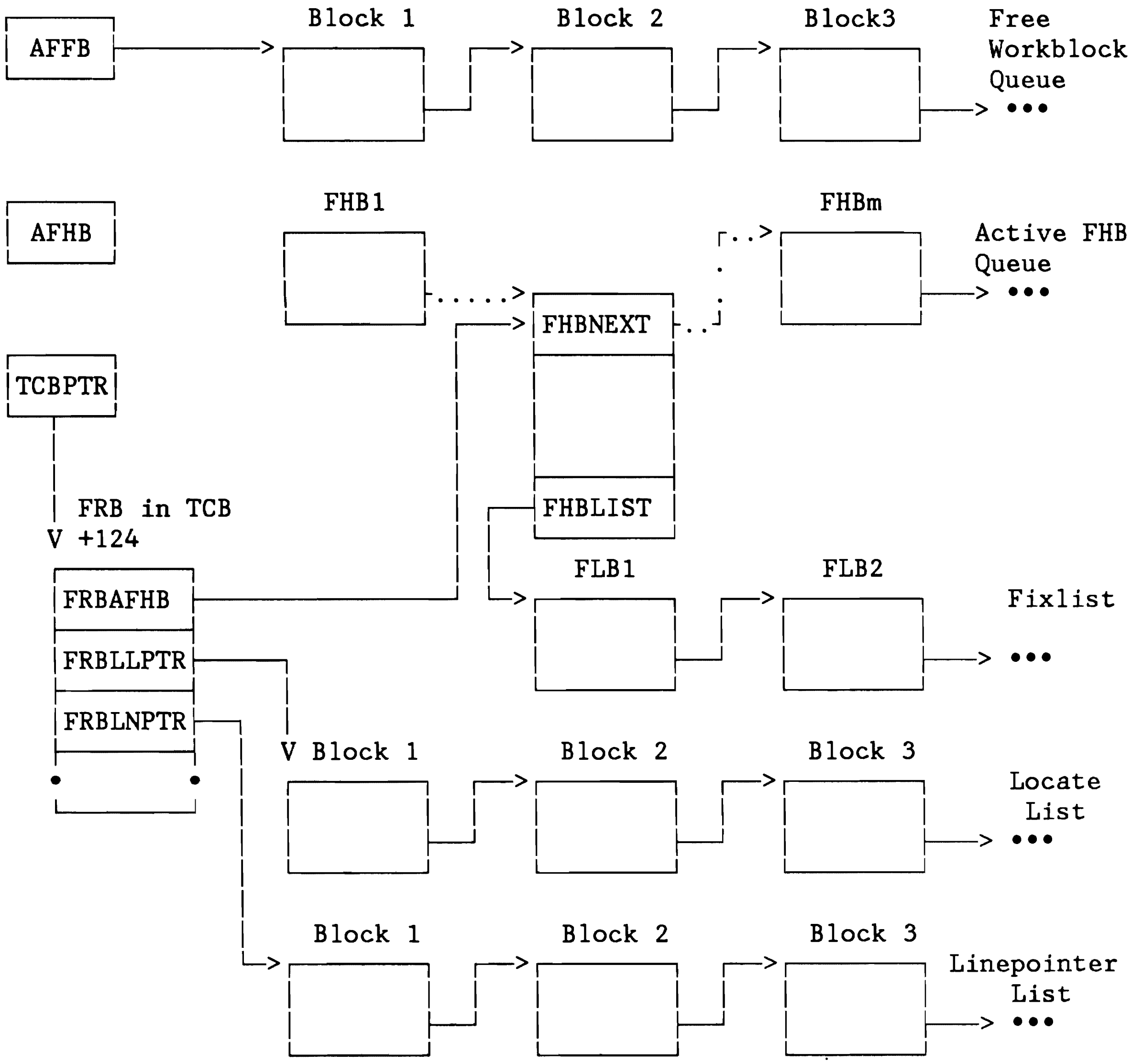


Figure 74. Relationship of Control and Work Blocks for CCW Fixing

Fix Request Block

The Fix Request Block (FRB) serves as a dynamic save area and work area. It is located in the TCB work area. Since a fixing request may be interrupted (for example by a page fault, wait), the fixing routine has to be partially re-enterable to enable the handling of several requests simultaneously.
The layout of the FRB is described in Figure 235 on page 508.

Work Blocks

Work blocks are required to support the following functions:

- To build the internal fixlist (work blocks used as fixlist blocks).

- To build locate list blocks.

- To store information during the channel program scan about parts of the channel program still to be handled (work blocks used as line pointer blocks).

- If fast fixing is supported, to save the replica (work blocks used as replica blocks). The replica consists of the channel program and related information.

Work blocks are 36 bytes long. The minimum number of blocks is determined at supervisor generation time. The total number can be specified at IPL time using the command SYS BUFSIZE=nn. They are allocated by IPL and chained into a free work block queue, pointed to be AFFB. If a work block is needed, it is dequeued from the free work block queue and returned after completion of the request.

Fixlist Blocks

Fixlist blocks (FLB) are required to build the internal fixlist which describes the storage areas that are to be TFIXed for this I/O request. Each storage area is described by a 4-byte entry in a fixlist block. Each entry identifies the first (BA) and the last (EA) page of the area to be TFIXed.

The first fixlist block, called fixlist header block (FHB), contains control information related to the fixlist and the first 6 fixlist entries; the first 3, if fast fixing is supported. Each additional fixlist block contains 8 fixlist entries and can TFIX a minimum of 16K storage.

Figure 75 and Figure 76 on page 197 show the layout of the FHB; Figure 77 on page 198 shows the layout of an FLB.

BA is the page number multiplied by (pagesize/2**8) of the first page to be TFIXed. EA is the page number multiplied by (pagesize/2**8) of the last page to be TFIXed.

0	Flag Byte 1 (General Fixing Function)	Flag Byte 2 (Reserved)	TID of Requester
4	Pointer to next active FHB		
8	BA1	EA1	
12	BA2	EA2	
16	BA3	EA3	
20	BA4	EA4	
24	BA5	EA5	
28	BA6	EA6	
32	Pointer to next fixlist block		

Figure 75. Layout of Fixlist Header Block (FHB) for General Fixing Function

0	Flag Byte 1 (General Fixing Function)	Flag Byte 2 (Fast Fixing Support)	TID of Requester
4	Saved queue forward pointer		
8	Saved queue backward pointer		
12	Pointer to replica or zero		
16	Pointer to next active FHB		
20	BA1	EA1	
24	BA2	EA2	
28	BA3	EA3	
32	Pointer to next fixlist block		

Figure 76. Layout of Fixlist Header Block (FHB) for Fast Fixing Support

The meaning of the flag bytes is as follows:

Flag Byte 1 (General Fixing Function):

Bit 0=1: Fixing function request complete.

Bit 1=1: At least one page is fixed for this task or the fixing request is pending.

Bit 2=1: Fixing of pages required.

Bit 3-7: Reserved.

Flag Byte 2 (Fast Fixing Support):

Bit 0=1: Fast fixing in progress.

Bit 1=1: FHB belongs to saved FHB queue.

Bit 2=1: IORB request.

Bit 3-7: Reserved.

0	BA1	EA1
4	BA2	EA2
8	BA3	EA3
12	BA4	EA4
16	BA5	EA5
20	BA6	EA6
24	BA7	EA7
28	BA8	EA8
32	Pointer to next fixlist block or zero	

Figure 77. Layout of Fixlist Block (FLB)

Locate List Blocks

A locate list is built during the scanning of the channel program of an EXCP request that uses a CCB. Each locate list block consists of four 8-byte entries, and an entry contains a begin (BA) and an end address (EA) which define an area of the channel program whose CCWs have been checked already.

After completion of the scanning procedure the locate list defines those areas of a channel program that have to be TFIXed. The entries describe isolated areas; they are not adjacent or overlapping, and are arranged in ascending sequence. Figure 78 on page 199 shows the layout of a locate list.

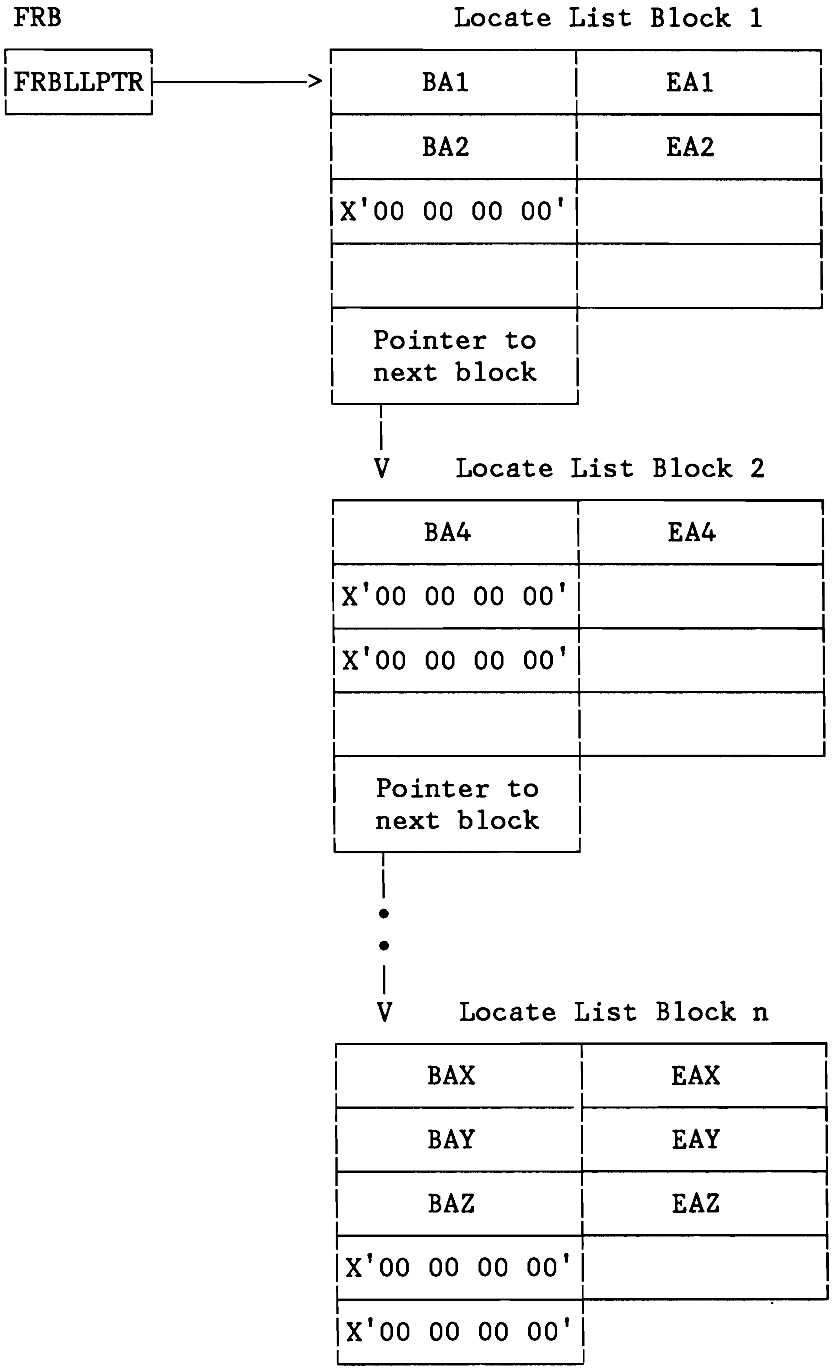


Figure 78. Locate List Example

Line Pointer Blocks

Line pointer blocks are used for storing addresses of channel program areas during the scanning procedure. These areas may not have been checked for fixing yet, because the channel program consists of several lines, deriving from STMs followed by TICs. The line pointer blocks ensure that all lines of the channel program will be checked for fixing. Figure 79 shows a line pointer list.

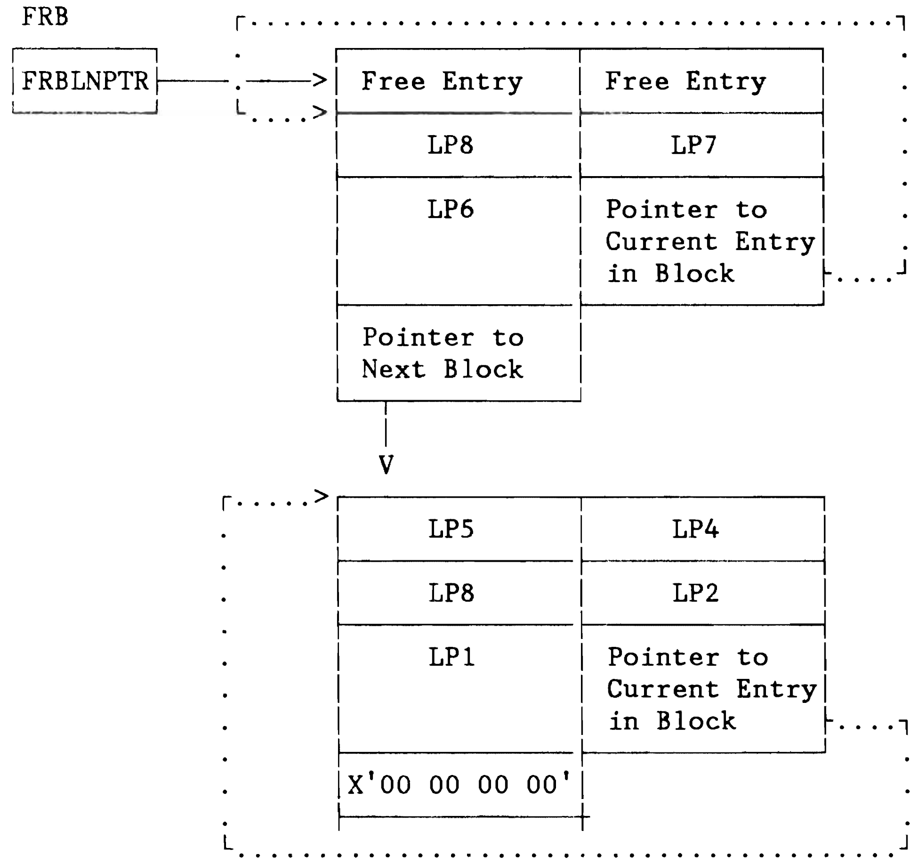


Figure 79. Line Pointer List Example

REPLICA Blocks

If fast fixing is supported, REPLICA blocks are needed to save the channel program and related information. The first and the second REPLICA blocks are called REPLICA Header Blocks (RHB1 and RHB2). The contents of RHB1 and RHB2 for a CCB request is shown in Figure 80 and Figure 81 on page 202. Figure 82 on page 202 shows a CCW REPLICA block.

For an IORB with an unchained fixlist RHB1 and RHB2 and the normal REPLICA block are shown in Figure 83 on page 203, Figure 84 on page 204 and Figure 85 on page 204.

0	(See Note) Flag Byte	Number of Tests on REPLICA	PIK of Requester
4	Forward pointer in partition's REPLICA queue		
8	Backward pointer in partition's REPLICA queue		
12	Address of virtual CCB		
16	Saved CCB		
32	Pointer to RHB2		

Note:

- Bit 0=1 Freeing of REPLICA requested.
- Bits 1-7 Reserved.

Figure 80. Layout of REPLICA Header Block (RHB1) for a CCB

0	Pointer to RHB1
4	Pointer to associated FHB (fixlist)
8	Time stamp
12	Length of saved channel program
16	Reserved
20	Reserved
24	Saved user SENSE
32	Pointer to next REPLICA block

Figure 81. Layout of REPLICA Header Block (RHB2) for a CCB

0	CCW1
8	CCW2
16	CCW3
24	CCW4
32	Pointer to next block or zero

Figure 82. Layout of REPLICA Block for a CCB

0	(See Note) Flag Byte	Number of Tests on REPLICA	PIK of Requester
4	Forward pointer in partition's IORB REPLICA queue		
8	Backward pointer in partition's IORB REPLICA queue		
12	Reserved		
16	First user fixlist entry		
24	Second user fixlist entry		
32	Pointer to RHB2		

Note:

- Bit 0=1 Freeing of REPLICA requested.
- Bits 1-7 Reserved.

Figure 83. Layout of REPLICA Header Block (RHB1) for an IORB

0	Pointer to RHB1
4	Pointer to associated FHB (fixlist)
8	Time stamp
12	Length of saved channel program
16	Third user fixlist entry
24	Fourth user fixlist entry
32	Pointer to next REPLICA block

Figure 84. Layout of REPLICA Header Block (RHB2) for an IORB

0	Fifth user fixlist entry
8	Sixth user fixlist entry
16	Seventh user fixlist entry
24	Eighth user fixlist entry
32	Pointer to next block or zero

Figure 85. Layout of REPLICA Block for an IORB

PAGE MANAGEMENT

Introduction into Page Management

The page management is responsible for the management of the data set containing the virtual address space(s), for the allocation of the processor real storage to parts of the virtual address space being requested and for the related replacement strategy. The unit of logical storage is the PAGE, the data set is called PAGE DATA SET (PDS). The real storage area containing a page is called a PAGE FRAME.

A page management function satisfies those processor requests created by addressing a valid logical area not yet assigned to and located in real storage (PAGE FAULTS). The related page is in disconnected state and its copy - if valid - has to be read from the PDS into a selected page frame. The page has thereafter addressable state. However, if there is no free page frame, at first the page currently located in the selected page frame has to be saved onto the PDS before the frame can be used by the new page. The state of the saved page is changed from addressable into disconnected.

A further function provides the capability to FIX a page in the real storage. This function is required for the I/O subsystem which operates on real storage (page frames) only. Because of performance considerations the fixing can be also desirable for frequently used address ranges.

Another function allows the user to control the paging environment by its own services. These services are implemented for the various subsystems to allow an optimization of the 'page' resources.

As seen, the total page management can be subdivided into the following main parts:

- Page handling support
  - Page fault handling via PGQUI with the page selection algorithm
  - Page out handling via PGQUO
  - SVC services concerning page state (SVC58, SVC59, SVC106, SVC109)
- FIX / FREE support
  - TFIX / TFREE services for the I/O subsystem.
  - SVC services for user PFIX / PFREE and SVA PFIX / PFREE (SVC67, SVC68 and SVC100)
  - SVC services for allocation of real storage (370 Mode only) with SVC54 and SVC55
  - SVC service for CHECKPOINT / RESTART (SVC74)
- Page handling by user
  - PHO capability (Page Fault Overlap) with SVC71
  - SVC services concerning page-in, release page and forced page out (SVC85, SVC86, SVC87)
  - VIO (Virtual I/O) support.

Description of Parallel Page I/O

Parallel page I/O is done by overlapping the page I/O operations for separate page-data-set devices. Therefore parallel page I/O requires a multiple extent page-data-set, at best each extent distributed on a separate device but at least two extents on two devices.

For every page-data-set device, there is one page-in queue per partition (inclusive system partition) and one page-out queue.

The I/O operations are controlled by a system task, the so called PMR-task. The page-data-set devices are serviced in wrap around mode. The PMR-task tries to start an I/O request on each device as long as requests are pending and not yet started. Thereafter the PMR-task waits for completion of at least one I/O.

However, before the page-fault request is enqueued it is check whether the request can be serviced without any I/O. If so, the request is handled under the requesting task without any activation of the PMR-task.

Introduction into VAE Support (370-Mode Only)

The concept of VAE in DOS/VSE provides n virtual address spaces or memories, each up to 16MB. Each address space is separated into a private addressable part and a shared addressable part. The shared part is unique in the system. Programs and data used in any address space must be located in the shared part (e.g. supervisor routines, SVA programs, control blocks in the system GETVIS area). The sum of all private address spaces and the shared address space is restricted to 40 MB.

The total virtual address range can be thought as a contiguous and linear space. This area is represented by list of Page Table Entries (PTE). Each entry is associated to an unique address range of the size of one page (PAGESIZE) - identified by its page number (PNR) - and to a unique block on an external storage medium. These blocks build the Page Data Set (PDS), consisting of a set of data extents on one or more disk devices. The contiguous area is addressed by the Extended Page Address (EPA), that is PNR*PAGESIZE.

The address mechanism, as defined by the /370 architecture (see Principles of Operation, GA22-7000), is done by the Segment Table Entry (STE) and the Page Table Entry (PTE). Each STE addresses a list of a contiguous PTEs which describe a logical address range of 64 K or 1 M bytes. The different memories are represented by different segment tables and are managed by Space Control Blocks (SCB). The shared area is addressable via any valid segment table (that means: it is part of any address space). The private address areas are only addressable via an unique segment table.

In VSE/VAE the page size is 4K and the STE points to page table of 16 entries (page table segment).

Every page table segment is associated to an entry in the Page Table Assignment String (PTAS), indicating whether the page table segment is already in use or not. The entry is abbreviated as PTASE and contains either zero (if unused) or the address space number and the segment number where the related page table is assigned to. The allocation algorithm provides both minimal SEEK time and an uniform distribution over the extents and devices.

The concept of real partitions is separately implemented. There is a real address space with own space control block (SCBR), segment table (STABR) and page tables (RPT-s). In opposition to the virtual address spaces there is no PTAS.

DATA STRUCTURES OF PAGE MANAGEMENT

Segment Table (370 Only)

Each address space (virtual memory) is identified by its segment table. At IPL time the complete segment table is generated for the first address space. This table contains one entry for each 64K-byte segment of virtual storage.

The segment tables for the other address spaces are allocated and initialized whenever the first partition of this address space is activated and the related Space Control Blocks (SCB) are built. The SCB provides a pointer to the associated segment table origin.

The segment table entry is given by the /370 Architecture (see Principles of Operation, GA22-7000), as shown in the following Figure 86:

```
┌───┬─────────────────┬─┬─┬─┐
│PTL│ page table addr │0│C│I│
└───┴─────────────────┴─┴─┴─┘
0   4                29 30 31
```

Figure 86. Segment Table Entry

```
PTL :   (16/max * len) - 1
        len = actual length of page table
        max = maximum size of page table

page table address:
        address of page table segment allocated to entry

C   :   common segment bit

I   :   invalid segment bit ( = 0 - the segment is valid)
                            ( = 1 - the segment is invalid)
```

Page and Page Table Entries (370 Only)

The unit of virtual storage is the page of the size of 4 respectively 2K bytes (for ECPS:VSE mode). It is represented by the associated PTE which describes the state of a page.

A page is addressable, if it is located in a page frame; it is disconnected, if it is not in a page frame and it is connected if it is located in a frame but not addressable. (Only page I/O is running on connected pages).

The PTE is given by the /370 Architecture (see Principles of Operation, GA22-7000) if the invalid bit is off. If the invalid bit is on the PTE is interpreted by the VSE/VAE software as shown in Figure 88 on page 209 .

Bit	Label	Description
0-15	PTE	Page addressable
0-11	PFRA	Page frame number
12	IBIT	Invalid bit = 0
13-15		Architected = 0

Figure 87. Page Table Entry (PTE) for Addressable Page

Bit	Label	Description
0-15	PTE	Page not addressable
0- 4	STKEY	Storage key of page
5- 9		reserved
10	HABIT	Invalid state: HABIT = IBIT = 1
11	COBIT	Connected state: COBIT = IBIT = 1
12	IBIT	Invalid bit = 1
13-14		Architected = 0
15	PDSBIT	Valid copy on PDS = 1 no copy on PDS = 0

Figure 88. Page Table Entry (PTE) for not Addressable Page

The page has addressable state: IBIT = 0.
That means the page is currently in real storage and the frame is given by the PFRA value.

```
PFRA = frame-address * 2**(-12)
```

The page has invalid state: IBIT = HABIT = 1 and COBIT = 0.
That means the page is not in the address range of the memory, e.g. a reference to the real partition if the virtual partition is active.

The page has connected state: IBIT = COBIT = 1 AND HABIT = 0.
That means page I/O is running for this page.

The page has disconnected state: IBIT = 1 and HABIT=COBIT = 0.

A data invariant is given as: HABIT=COBIT=1 not possible.

The invalidation pattern for the PTE (used to set a page into invalid state) is B'STKEY00000101000'.

Page Table Initialization

1. For a VSE system with MODE=VM specified, the page table entries are initialized at IPL time and are never changed during processing.

   - All page table entries belonging to VM-storage:

     ```
     Bits    0 - 11 = The leftmost 12 bits of the address of
                      the corresponding page frame.
            12 - 15 = 0
     ```

   - All remaining page table entries:

     ```
     Bits    0 - 11 = 0
            12      = 1
            13 - 15 = 0
     ```

2. For a MODE=370 system during IPL, page table entries are initialized as follows:

   - All page table entries belonging to the supervisor area (nucleus and transient areas):

     ```
     Bits    0 - 11 = The leftmost 12 bits of the address of
                      the corresponding page frame.
            12 - 15 = 0
     ```

   - All page table entries for allocated REAL partitions:

     ```
     Bits    0 -  3 = Storage key of corresponding partition.
            10      = 1
            11      = 0
            12      = 1
            13 - 15 = 0
     ```

   - Page table entries belonging to VIRTUAL partitions:

     ```
     Bits    0 -  3 = Storage key of corresponding partition.
            10      = 0
            11      = 0
            12      = 1
            13 - 15 = 0
     ```

   - Page table entries belonging to SVA:

     ```
     Bits    0 -  4 = Storage key of SVA.
            10      = 1
            11      = 0
            12      = 1
            13 - 15 = 0
     ```

   - All remaining page table entries:

     ```
     Bits    0 -  9 = 0
            10      = 1
            11      = 0
            12      = 1
            13 - 15 = 0
     ```

Status of a Page Table Entry

- If the page is addressable:

Bits	0 - 11	=	The leftmost 12 bits of the page frame address where the page is located.
	12	=	0
	13 - 14	=	0
	15	=	0 No copy on page data set.
		=	1 Copy on page data set.

- If the page is disconnected:

Bits	0 - 3	=	Storage protection key of the page.
	4	=	Page is fetch protected (SVA).
	5 - 11	=	0
	12	=	1
	13 - 14	=	0
	15	=	0 No copy on page data set.
		=	1 Copy on page data set.

- If the page is connected:

Bits	0 - 11	=	The leftmost 12 bits of the frame address to which the page is connected (valid only if conditional page-out of that page is requested by GETREAL).

Bits	0 - 3	=	Storage protection key of the page.
	4		Page is fetch protected (SVA).
	5 - 10	=	0
	11	=	1
	12	=	1
	13 - 14	=	0
	15	=	0 No copy on page data set.
		=	1 Copy on page data set.

- If the page is invalid (that is the page belongs to the inactive part of the partition or to the page pool):

Bits	0 - 3	=	Storage protection key of the partition, if the PFTE belongs to a partition.
	4 - 9	=	0
	10	=	1
	11	=	0
	12	=	1
	13 - 15	=	0

The storage key is part of the page, in 370-mode part of the frame and must be saved in the PTE whenever the page is disconnected.

Page Frame Table (PFT)

The real storage is subdivided into page frames of the size of 4 respectively 2K bytes (for ECPS:VSE mode). Each frame is uniquely associated to an entry in the PFT describing the status of the frame. This entry is abbreviated as PFTE.
The page selection queue (PSQ) contains all PFTEs of frames occupied by page and usable for page replacement (essentially pages which are not FIXed). The number of PFTEs in PSQ is given by length(PSQ). In E-mode the free page frames are maintained by the hardware and the counter is given by the value FFCC (Free Frame Capacity Count) of the STCAP instruction. The related PFTEs are not in PSQ; their content is undefined. A page is connected to a frame via the LFI instruction. The hardware returns a condition code and, if FFCC > 0, the frame index of the page. The frame index identifies the related PFTE.
In 370-mode the free page frames are managed by the page management itself and the associated PFTEs are queued in the invalid page frame queue (IPFQ). The length of IPFQ is given by len(IPFQ).

The PFT is built at IPL time and contains one 16-byte entry for each real storage block of 2 respectively 4K. Field APFT contains the begin address of the table.

Figure 89 shows the layout of a page frame table entry (PFTE).

0	1	3	4	5	6	8	12 15
PFTE Flag	Page Number (PNR)	370 Mode Flag	Waiting Task ID (WID)	PFIX Counter	TFIX Counter	Forward Pointer	Backward Pointer

Figure 89. Page Frame Table Entry (PFTE)

Byte(s)	Bit	Description
0 - 2		Page frame number (0, 1, 2, 3, ...)
3	4	=0 Page frame belongs to supervisor or IPL partition.
	4	=1 Page frame belongs to initial page pool
4 - 7		zero

Figure 90. Initialization of all PFTEs at IPL Time

Byte(s)	Label	Description
0	PFTEFLG	PFTE flag
	HBIT	X'80' Each task causing a page fault can use the the page before it is disconnected again.
	POEBIT	40 The PFTE is enqueued for page-out.
	POBIT	20 An active entry from the PMR task is enqueued for page-out.
	POABIT	10 I/O for a page-out has been started for this PFTE.
	PCBIT	08 The page which belongs to the page frame has connected state. Either a page-in or an unconditional page-out request is in progress.
	POSYSBIT	04 A page-out request is in a system queue.
		02 Reserved
		01 Reserved
1 – 2	PNR	If a page belongs to the page frame, these bytes contain the page number (='virtual-page-address'/pagesize). If a block of VIO storage belongs to the frame, these bytes contain the block number.
3	S370FLG	370 mode flag
	NFRP	X'80' Frame is used by a PFIXed page. Since the frame is in the PSQ or IPFQ this page must not be TFIXED if the TFIX counter is zero.
	NFVP	40 Page belonging to this frame is requested by PFIX. The frame is not in the PSQ. The PFIX request cannot be satisfied immediately.
	DRAP	20 The address space belonging to the PFTE is failing storage.
	PFTEBLK	10 Only block of VIO-storage connected to frame
	PNRINV	08 Page frame is unused. The PNR-, FIX- and WID fields are invalid. Also the PFTE- and 370 mode flags (except for NFRP and DRAP bits) are invalid.
	PFTEREAL	04 Frame is used by real partition.
		02 Reserved
		01 Reserved
4	PFTEWID	Waiting task id (370 mode only): Contains the PIK of the partition requesting PFIX. The page frame of the page to be PFIXed does not belong to the corresponding real partition.
5	PFIXC	Indicates how often the page is PFIXed.
6 – 7	TFIXC	Indicates how often the page is TFIXed.
8 – 11	PFTEFPTR	Pointer to the next PFTE.
12 – 15	PFTEBPTR	Pointer to the preceding PFTE.

Figure 91. PFT Entry Byte Description

Note: The pointers in bytes 8 through 15 are only valid if the PFTE is in the PSQ, or, for 370 mode, in the IPFQ.

Status of a Page Frame Table Entry (PFTE)

1. If a PFTE is not assigned to a page:

   - ECPS:VSE mode:
     The PFTE is not enqueued to the Page Selection Queue (PSQ).
   - 370 mode:
     If no block of VIO-storage is connected to the frame, the PFTE is enqueued to the Invalid Page Frame Queue (IPFQ), the PNRINV bit is set, and the NFRP bit may be set, and the remaining contents of the PFTE is undefined.
     If only a block of VIO-storage is connected to the frame, the PFTE is enqueued to PSQ and the TFIX and PFIX counters are zero.

2. If a PFTE is assigned to a connected page:

   - ECPS:VSE mode:
     The PFTE is not enqueued to the PSQ and the contents of the PFTE is valid. PFIX and TFIX counter must be zero, and PNR must indicate a connected page.
   - 370 mode:
     The PFTE is neither enqueu3d to the PSQ nor to the IPFQ. The contents of the PFTE is valid. The PC bit is set, PNR indicates a connected page, and the PFIX and TFIX counters are zero.

3. If a PFTE is assigned to an addressable page, the contents of the PFTE is valid:

   - In ECPS:VSE mode:
     If the PFIX and TFIX counters are zero the PFTE is enqueued to the PSQ. If the PFIX or TFIX counter is not zero, the PFTE is not enqueued to the PSQ.
   - In 370 mode:
     If the NFVP bit is set, the PFTE is neither enqueued to the PSQ, nor to the IPFQ. If the NFVP bit is reset, and the PFIX and the TFIX counter are zero, the PFTE is enqueued to the PSQ. If the NFVP bit is reset, and the PFIX or the TFIX counter is not zero, the PFTE is neither enqueued to the PSQ nor to the IPFQ.

Status of Supervisor Page Frames

The PFTEs that belong to the fixed supervisor part and to the IPL partition (ECPS:VSE Mode) are marked as PFIXed and are not enqueued to the Page Selection Queue (PSQ). The PFTEs that belong to the pageable supervisor part are enqueued at the end of the PSQ. The PFTEs that belong to the initial main page pool are enqueued at the end of the Invalid Page Frame Queue (IPFQ) in 370 mode or are returned to the hardware in ECPS:VSE mode.

Page Table Assignment String (PTAS)

Every page table segment describing a contiguous address range of 64 K bytes is associated to an entry in the Page Table Assignment String (PTAS). It indicates whether the related page table segment is already in use or not. The entry is abbreviated as PTASE and contains either zero (if unused) or the address space number and the segment number where the related page table is assigned to.

Bytes	Label	Description
0 – 1	PTASE	Entry length 2 bytes
0	PTASESPN	Space number where the PTAB belongs to
1	PTASESGN	Segment number where the PTAB is assigned to

Figure 92. Page Table Assignment String Entry (PTASE)

Storage Management Control Block (SMCB)

The SMCB - being part of the partition control block (PCB) - contains the necessary control information for the storage allocation. The page management is concerned by:

SMAXPFIX	partition/SVA PFIX limit in pages (size of real partition)
SMPFIX	actual PFIX count

Moreover, the virtual and real partition boundaries are considered by the page management.

NPSQE

NPSQE represents the actual value of page frames available for replacement. That means in ECPS:VSE mode

NPSQE = len (PSQ) + FFCC

and in 370-mode

NPSQE = len (PSQ) + len (IPFQ)

In order to satisfy a page fault under all conditions, the number of available page frames must not be lower than a specific limit MINPSQE, that means there is data invariant:

NPSQE > MINPSQE - 1

Please see sections Page Frame Table and Selection Pool Queues.

Page Data Set Table

Page Management uses the Page Data Set Table (DPDTAB) to calculate the correct address for a given page on the Page Data Set, if a read or write operation is necessary. Bytes 224-227 (X'E0'-X'E3') of the System Communication Region (SYSCOM) contain the address of the DPDTAB. The DPDTAB consists of a header and 15 extent definitions. Label DPDTAB identifies the first byte of the table. The table has the following layout:

Dec	Hex	Label	Description
0-15	0- F	DPDADR	Header
0- 1	0- 1	DPDEXT#	Number of possible extents
2- 3	2- 3	DPDAEXT#	Number of actual extents
4- 7	4- 7	DPDPAG#	Number of supported pages
8-11	8- B	DPDLLCON	Address of load leveling constants
12-13	C- D		Reserved
14-15	E- F	DPDLEN	Length of header

Figure 93. Page Data Set Table Header

Dec	Hex	Label	Description
0-31	0-19	DPDENTR	Extent definition
0- 1	0- 1	DPDUNT	CUU of PDS device
2	2	DPDDEVT	Device type:FBA, CKD, RPS
3	3	DPDDEVC	Device code (DTF)
4- 5	4- 5	DPDREC#	CKD: # records/track
4- 5	4- 5	DPDBLKLG	FBA: block length
6- 7	6- 7	DPDTRCK#	CKD: # tracks/cylinder
6- 7	6- 7	DPDBLKPG	FBA: # blocks/page
8-11	8- B	DPDRTLL	CKD: track# of lower extent limit
8-11	8- B	DPDBLKLL	FBA: block# of lower extent limit
12-15	C- F	DPDTRCKU	CKD: # of used tracks
12-15	C- F	DPDBLKU	FBA: # of used blocks
16-17	10-11	DPDPUB	PUB index
18-23	12-17	DPDVOLID	Volume id of PDS
24-27	18-1B	DPDPGUL	Page # of upper limit
28-31	1C-1F	DPDDEVCB	Addr. of DEVCB for extent

Figure 94. Page Data Set Extent Definition

Device Control Block (DEVCB)

Every PDS device is described by its associated Device Control Block (DEVCB).

Bytes			
Dec	Hex	Label	Description
0	0	DEVCB	Device control block
0- 3	0- 3	DEVCBNXT	Addr. of next DEVCB if any, addr. of first DEVCB in chain for last DEVCB
4	4	DEVSTAT	Status byte
		DEVSTRT	X'80' I/O request started
		DEVEMPTY	X'40' no I/O request enqueued
		DEVPGWO	X'20' request waits for unconditional page out
5	5	DEVCBTYP	Device type: FBA,CKD,RPS
6- 7	6- 7	DEVEXT#	Number of extents on device
8- 11	8- B	DEVACT	Address of PGQE
12- 15	C- F	DEVDPD	Addr. of 1st DPD entry for device
16- 19	10- 13	DEVRELO	Relocation for 1st DPD entry on device
20- 23	14- 17	DEVAPTAS	Addr. of 1st PTASE for device
24- 25	18- 19	DEVPTASA	Highest offset of PTASE already occup.
26- 27	1A- 1B	DEVPTASB	Number of contiguously located PTASEs and still available on device
28- 31	1C- 1F	DEVPCB	Address of related PCB
32- 35	20- 23	APFPSS	Address of PFPSS for dev.
36- 36	24- 27		Reserved
40- 55	28- 37	DEVCCB	CCB for device
56-103	38- 67	DEVCCW	CCW program area
104-107	68- 6B	PFRQBEG	Begin addr. of system page fault queue
108-111	6C- 6F	PFRQEND	End addr. of system page fault queue
112-	70-		Partition queue headers in the sequence BG, Fn, ... , F1 length = NPART*2*4
			NPART = 12 :
208-211	D0- D3	PORQBEG	Begin address of page-out queue
212-215	D4- D7	PORQEND	End address of page-out queue

Figure 95. Device Control Block (DEVCB)

Page I/O Request Element (PGQE)

The PGQE is part of Task Information Block (TIB). The following fields are relevant for page management.

Dec	Hex	Label	Description
0	0	TIBADR	Task information block
0- 3	0- 3	TIBCHAIN	
4- 7	4- 7	TIBSTATE	Bound state information page-in: page fault addr page-out: pageframe addr
8-11	8- B	TIBPFAPP	Addr. of PHO appendage
8-11	8- B	TIBVIOTB	Addr. of VIOTAB entry
12-15	C- F	PGQE	Page I/O request element
12	C	PGQTYP	Request type
		PGSEL X'80'	Page selection required
		PGNCNT X'40'	Page-in, counting done
		PGO X'10'	Page-out request
		PGOWAIT X'18'	Page-out req. with waiting task
		PGOPGIN X'14'	Page-out req. with waiting page-in
		PGOVIO X'12'	Page-out req. from VIO
13-15	D- F	PGINF	Information for page I/O handling
.	.		.... further TIB
.	.		
.	.		

Figure 96. Page I/O Request Element

Relationships between Control Blocks (370-Mode Only)

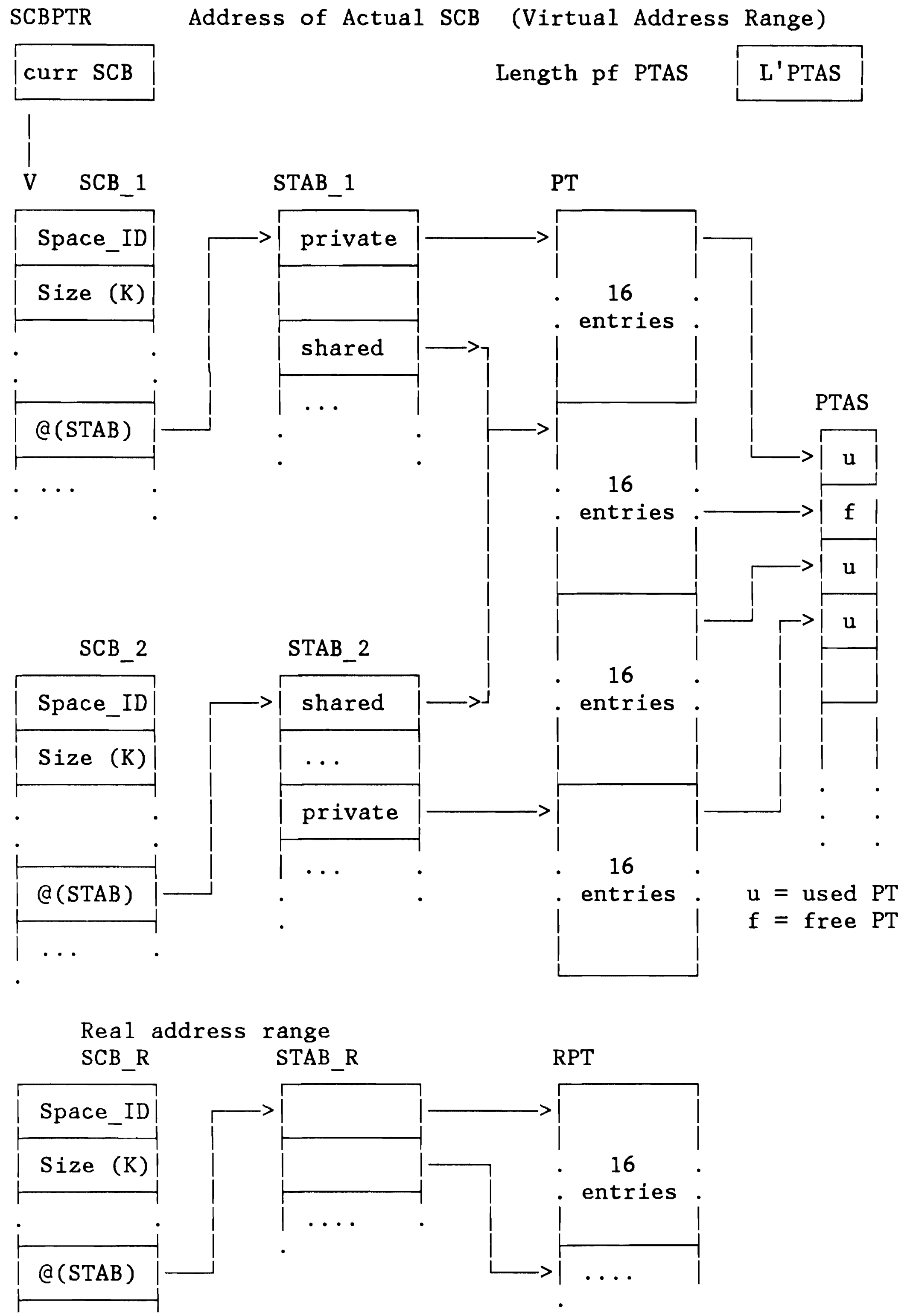


Figure 97. Relations between SCB, Segment Table, Page Table and PTAS

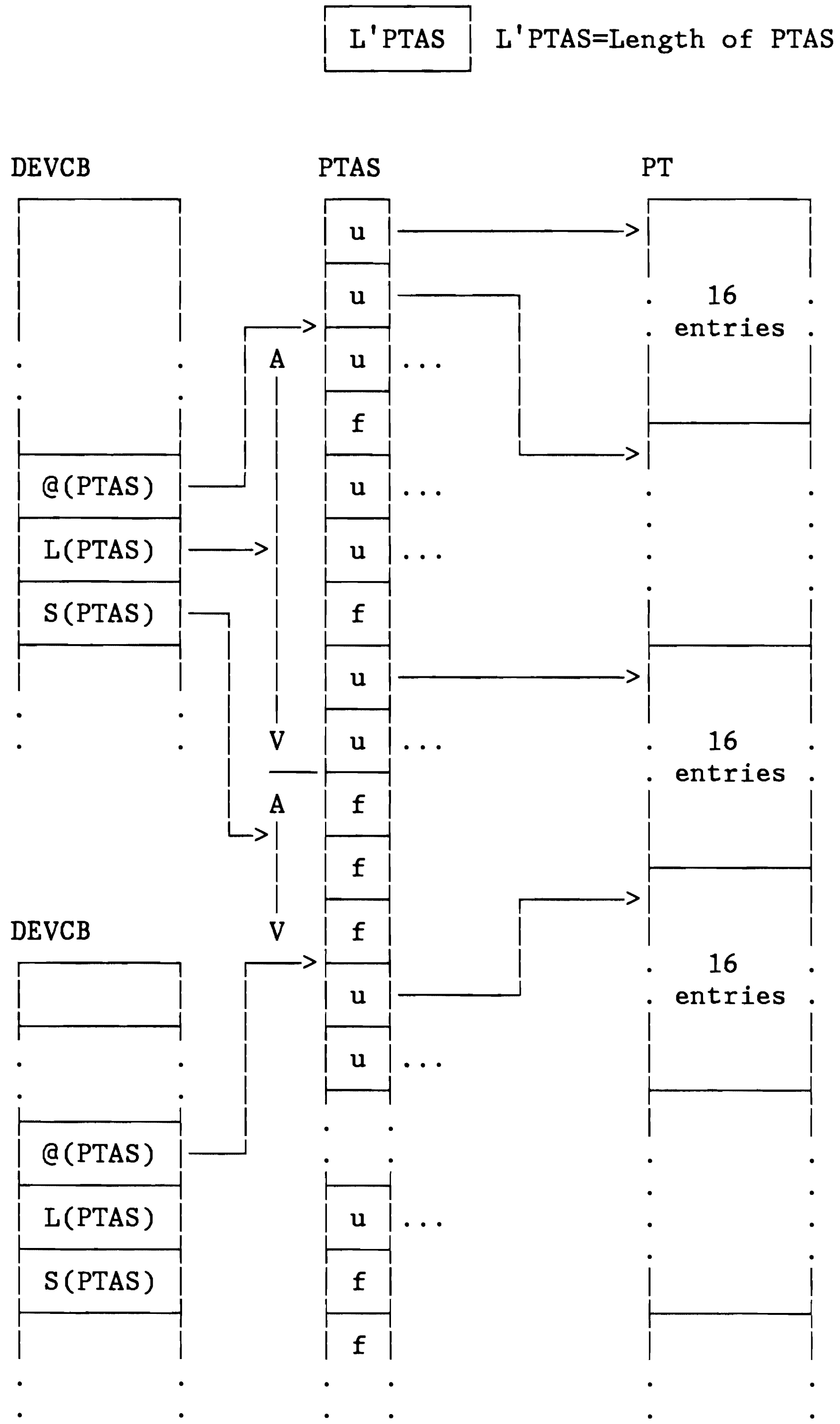


Figure 98. Relations between DEVCB, PTAS and Page Table

Selection Pool

The selection pool consists of all page frames which can be selected by the Page Management routines for paging. The selection pool contains all those pages which do not belong to the fixed part of the supervisor, to active real partitions, or to the alternate address area, and which are not fixed in some way (either by TFIX or PFIX).

Selection Pool Queues

The PFTEs that are not fixed (TFIX and PFIX counter zero) and have a page assigned are queued in the Page Selection Queue (PSQ). The PFTEs that have no page assigned are queued in the Invalid Page Frame Queue (IPFQ), if it is 370 mode. If it is ECPS:VSE mode, the PFTEs are not queued and the corresponding frames are maintained by the hardware.

Each queue has a queue header, which is 16 bytes long. Bytes 8 through 11 point to the first queue entry and bytes 12 through 15 to the last queue entry. How the selection pool page frame entries are queued is explained in the following section and in the section on the page frame selection.

Selection Algorithm

As mentioned above, all PFTEs available for selection are queued in the PSQ. The page selection algorithm ensures that at least MINPSQE page frames are available in PSQ and IPFQ (370 mode), respectively FFCC (ECPS:VSE mode). If the number NPSQE is lower than MINPSQE, page-out requests are performed in order to provide P > MINPSQE available frames. Therefore the PSQ is scanned during each page selection and the tupel (REFERENCE bit, CHANGE bit) of each associated page frame is inspected. This is done until P PFTEs with HOLD = OFF and related frames with (REF=OFF,CHANGE=OFF) are found, respectively Q > P PFTEs if at least one of the first P frames has been detected with CHANGE = ON. The PFTEs whose frames have REF = ON are removed to the bottom of PSQ with REF=OFF. PFTEs of frames with CHANGE = ON are enqueued in the PGQUO.

A 'HOLD' mechanism is implemented to ensure that a page just paged-in is not paged-out before the task caused the page fault is dispatched. The HOLD bit is set ON in the related PFTE and the PFTE is enqueued at the bottom of the PSQ. The reference- and change bits of the frame are set OFF. The HOLD bit is set OFF when during a subsequent page selection the reference bit of the frame has been found as ON. If all PFTEs are found with HOLD = ON the system is in thrashing state and load leveling is required; the HOLD bit is set OFF in all PFTEs.

A special interface is established for a TFIX request from the Fetch routine. It has to be ensured that at least MINPSQE-MINPSQEF pages can be fixed. As long as NPSQE > MINPSQEF, the Fetch request is satisfied. If not all requested pages can be fixed, control is given back to the Fetch routine without freeing the pages already fixed for this request.

The reservation of MINPSQE, or if it is a request from the Fetch routine, of MINPSQEF page frames for page replacement ensures that a page fault can always be handled by the PMR task.

If the PFIXPGE or GETREAL routine is executed, the counter NPSQE does not reflect all the time the actual number of PFT entries in the PSQ. The actual number of entries in the PSQ can be greater than the number indicated in NPSQE. Those additional entries are reserved by the PFIXPGE or GETREAL routine and cannot be used for other requests. In ECPS:VSE mode unused page frames are available if the hardware Free Frame Capacity Counter (FFCC) is greater than zero and they are selected by the hardware.

In 370 mode unused page frames are available if the IPFQ is not empty and they are selected by using the first one in the queue.

> **Note:** In 370 mode the reference (R) bit and the change (C) bit are located in the page frame. In ECPS:VSE mode they are located in the page. Whenever they are mentioned in this paragraph they refer to the page frame or the page belonging to the entry presently handled.

To ensure that pages newly paged-in are not paged-out before the task causing the page fault is dispatched, a hold-queue-mechanism works as follows:

The page is enqueued at the end of the PSQ, the R- and C-bit are reset, the hold bit is set. The hold bit is reset only if the R-bit was found set during the scan. If a certain number of PSQ entries have their hold bit set, the routine DEACTP of the load leveler is called to deactivate a partition and to reset the hold bits of all PSQ entries.

To overlap the page-in and page-out functions and to avoid the necessity of executing a page-out immediately before a page-in, a pre-page-out is implemented. It ensures that a minimum number of page frames is available (i.e. the page belonging to a page frame has its R- and C-bit reset). The pre-page-out is only active if FFCC=0 in ECPS:VSE mode, if IPFQ is empty in 370 mode.

The two functions of the page selection algorithm are:

- To select a page to be replaced.
- To ensure that a pre-page-out is executed if necessary.

To achieve this, the PSQ is scanned and the state of the R- and C-bit is checked.

The PSQ is scanned till a minimum number of entries with the R-bit reset has been found. If all these entries have their C-bit reset, too, the selection is finished, and the first PFTE found in the PSQ with the R-bit and the C-bit reset is used for replacement.

If at least one of the entries found has the C-bit on, the scan is continued for more entries with the R-bit reset. The first PFTE in the PSQ with the R-bit and the C-bit reset is used for replacement. If no such entry is found, the first PFTE with the R-bit reset is used.

For each PFTE found during the scan and with the R-bit reset and the C-bit on, a page-out request is enqueued to the correct Page-out Queue. Each PFTE found with the R-bit on is enqueued at the end of the PSQ with the R-bit and the hold bit reset.

Rearranging of Page Selection Queues

1. The PFTE of a page frame is dequeued from the PSQ:

   - If a TFIX or PFIX is requested for a page assigned to a page frame.
   - If the page assigned to a page frame has to be disconnected next (SELECTPG, INVPAGE, RELPAG).
   - If GETREAL is requested for the page frame (370 mode only).

2. The PFTE of a page frame is enqueued to the PSQ:

   - If a page has been TFREEed and is otherwise not fixed, and the NFVP bit is reset (if TFREE is from Fetch, the PFTE is enqueued at the beginning of the PSQ; if it is not from FETCH, the PFTE is enqueued at the end of the PSQ).
   - If a page has been PFREEed and is otherwise not fixed (the PFTE is enqueued at the end of the PSQ).
   - If a page-in has been completed (the PFTE of the page frame assigned to the page is enqueued at the end of the PSQ).

     370 mode only:

     If the page has been invalidated in the meantime, the PFTE is enqueued at the beginning of the IPQF.

     ECPS:VSE mode only:

     If an addressable page has to be invalidated but page-out is active, the PFTE is enqueued at the beginning of the PSQ.

3. The PFTE of a page frame is moved within the PSQ:

   - If during page selection a page is found with the R-bit on (PFTE is enqueued at the end of the PSQ).
   - If a PAGEIN request is for a page that is already in storage (the PFTE is enqueued at the end of the PSQ).

- If a FCEPGOUT request is for a page that is in storage (the PFTE is enqueued at the beginning of the PSQ).

4. The PFTE of a page frame is enqueued at the beginning of the IPFQ (370 mode only):

    - If no page is assigned to the page frame (after disconnect, INVPAGE RELPAG, FREEREAL).
    - If a page-in has been completed, but the page read-in has been invalidated in the meantime.

5. The PFTE of a page frame is dequeued from the IPFQ (370 mode only):

    - If an unfixed page frame is needed and the IPFQ is not empty during page selection, and in case of TFIX, PFIX and GETREAL to exchange page frames.

PAGE HANDLING ROUTINES

The following conditions result in some form of page movement or reassignment of page frames and may require activity by the page manager (PMR) system task:

- Page Fault
- GETREAL request (370 mode only)
- TFIX request
- PFIX request
- PAGEIN request
- VIO POINT request

However, the PMR system task is not activated for the following requests:

- FREEREAL request
- TFREE request
- PFREE request
- RELPAG/FCEPGOUT request
- INVPAGE request

The requests that require the activity of the PMR system task (always the case for a page fault) are queued in the page-in queues or the page-out queue for the device on which the page to be handled resides.

For each device on which the page-data-set resides, control information is maintained in a Device Control block (DEVCB). The page-data-set devices are serviced in wrap around mode. The PMR system task tries to start an I/O-request on each device as long as requests are pending and not yet started.

One page-in queue exists for each partition and one for the system. 'User-page-faults' (i.e. page faults in the user area) are queued in the corresponding partition page fault queue; 'system-page-faults' (i.e. all other page faults) are queued in the system page fault queue. Each queue consists of a forward chain of TIBs. For page-in requests the TIBs are the normal TIBs of the tasks waiting for completion of the page-fault handling and TIBSTATE (in TIB) contains the address of the page to be handled. For page-out requests pseudo TIBs are used which don't belong to any specific task and TIBSTATE contains the address of the PFTE to be handled. Begin and end of chain are maintained per device in the DEVCB to allow for enqueue at the bottom and dequeue at the top of the queue.

The requests that require writing pages onto the page data set (it may be requested by GETREAL and for the handling of a PGQUI entry) are queued in the page-out queue (PGQUO), and handled on a FIFO (first-in-first-out) basis.

The page-out queue consists of max. fifteen 20-byte entries, and the label PGQUO identifies the first byte of the table. The layout of a page-out queue entry is shown in Figure 99.

0	4
Address of next PGQUO entry	PGQE

Note: The page I/O request element (PGQE) is identical to the first 16 bytes of the pseudo TIB (see Figure 96 on page 219).

Figure 99. Page-Out Queue (PGQUO) Entry

Handling of a Page-In Request

A page-in request is enqueued to the proper page-in queue by the routine ENQUI. The PMR system task handles the page request queues in the priority order of the corresponding partitions. The system queue has the highest priority, the page-out queue has the lowest priority. Within each queue the entries are handled on a FIFO (first-in-first-out) basis.

The page manager (PMR) system task does the following steps when handling a page-in request.

- Select a page frame for the requested page (see Page Frame Selection and Pre-Page-Out) and remove it from the PSQ or the IPFQ.

- If the page frame selected is in use and its contents are not the same as that of the copy on the page data set (PDS), the page is set to the connected state and enqueued for page-out.

If the selected page frame is in use and its contents are the same as that of the copy on the PDS, the page is disconnected

- Read the requested page from the PDS, if a valid copy exists on the PDS. If not, the page is cleared to zero.
- Make the page addressable, that means: reset the reference and change bits, initialize the corresponding PFTE with the hold bit on, and enqueue it at the end of the PSQ.

Handling of a Page-Out Request

A page-out request is enqueued to the page-out queue (PGQUO) or on top of the system page-fault queue by the routine ENQUO with its different entry points. The page manager (PMR) system task performs the following steps when handling an entry:

- Reset the change bit and set the PDS bit of the requested page.
- Indicate page-out as active in the PFTE and write the page onto the PDS.
- Reset the reference bit.
- If posting is required, post the tasks that are waiting for the page frame.
- Reset the PGQUO indication in the PFTE.

Note: The handling of a normal page-out request does not change the status of a page. After the completion of an unconditional page-out request the page or block is disconnected.

If the selected page frame is in use and its contents are the same as that of the copy on the PDS, the page is disconnected

- Read the requested page from the PDS, if a valid copy exists on the PDS. If not, the page is cleared to zero.

- Make the page addressable, that means: reset the reference and change bits, initialize the corresponding PFTE with the hold bit on, and enqueue it at the end of the PSQ.

Handling of a Page-Out Request

A page-out request is enqueued to the page-out queue (PGQUO) or on top of the system page-fault queue by the routine ENQUO with its different entry points. The page manager (PMR) system task performs the following steps when handling an entry:

- Reset the change bit and set the PDS bit of the requested page.

- Indicate page-out as active in the PFTE and write the page onto the PDS.

- Reset the reference bit.

- If posting is required, post the tasks that are waiting for the page frame.

- Reset the PGQUO indication in the PFTE.

Note: The handling of a normal page-out request does not change the status of a page. After the completion of an unconditional page-out request the page or block is disconnected.

Page Fault Handling Overlap

Programs that execute in virtual mode and do their own multi-tasking can use the page fault handling overlap facility. This gives the user the opportunity to control the page-in queue entry for the page fault caused by its own task. This is done by a user-written page fault appendage routine.

Whenever a page fault occurs, page management first checks if a page fault appendage has been initiated for the task.

If the task has an appendage, control is first passed to that appendage, unless the task is using a supervisor service, the LTA, or an ACF/VTAM function. The request is then enqueued in the page fault queue using a special TIB (PHOTIB) located in the PCB, or it gives an indication that a page fault is already pending for that task. The task causing the page fault is not set into the wait state.

On a supervisor with MODE=VM specified, and with the VM/370 command SET PAGEX ON issued, page fault overlap handling appendages are not entered when pseudo-page faults occur. Programs that use page fault appendages to do their own multi-tasking, run as a single task under these circumstances.

If the page fault was caused by a supervisor service or logical transient, or if an ACF/VTAM function is outstanding, no overlap is performed. The page fault is handled like any normal page fault condition and the task is set into the wait state.

When a page fault has been handled for a task having a page fault appendage, the appendage is entered again to see if there are any more page faults to be processed. If so, the page-in request returned from the appendage is enqueued in the correct device queue.

Pseudo-Page Fault

Pseudo page faults are a special type of program check used with VSE/VMLE. There are two different types of pseudo page faults:

- Pseudo page fault exception (whenever VM gets a page fault and must do I/O operations)
- Pseudo page fault completion (whenever the I/O operation of VM is completed)

For both exceptions VM passes control to VSE by means of program check interruption.

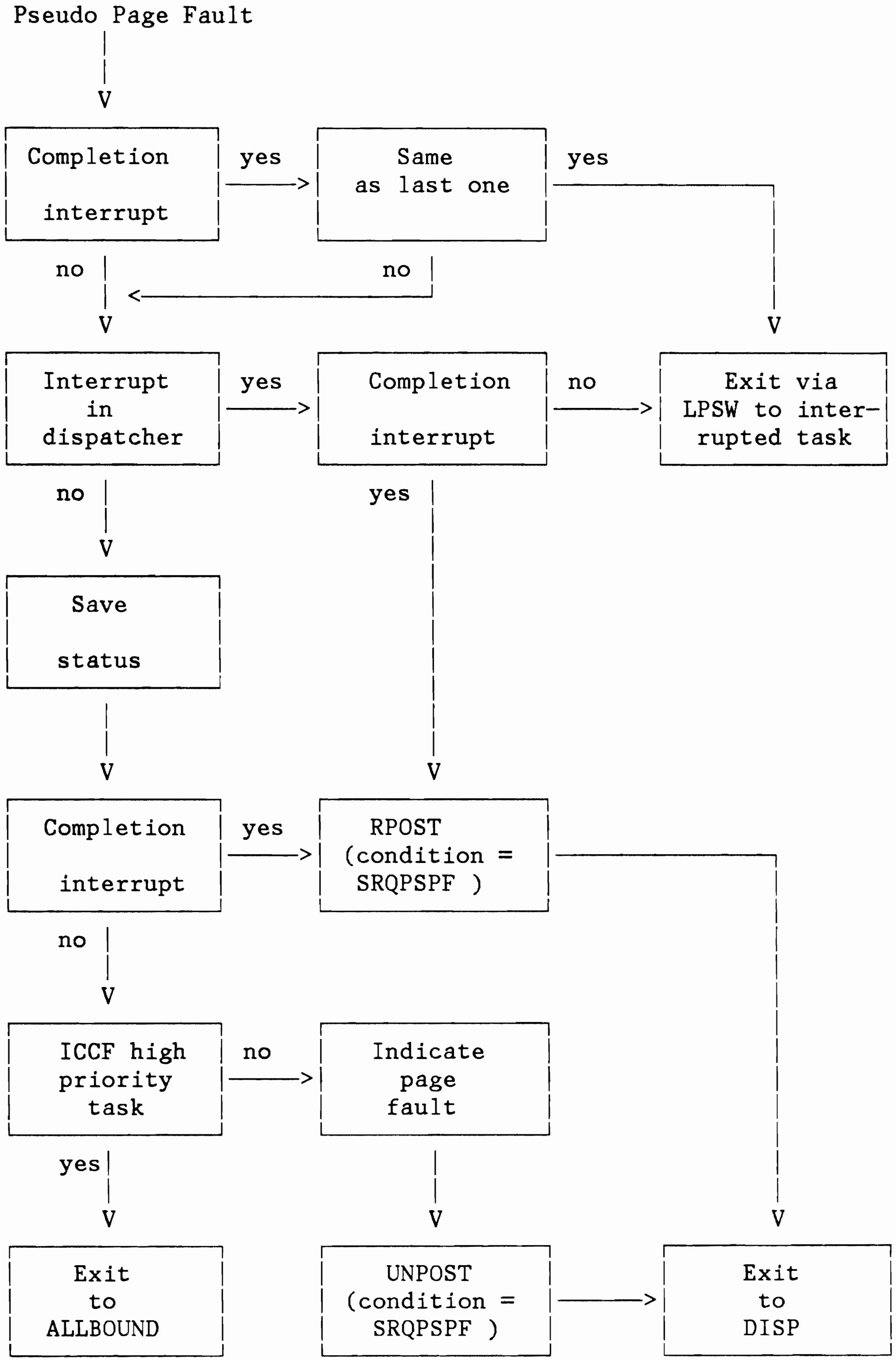


Figure 100. Pseudo Page Fault Handling

GETREAL Request

A GETREAL request is valid only in 370 mode and is issued by SVC X'37' (request for SDAID area), SVC X'3A' (if initialization of a real partition is requested), and Storage Management (GETVIS request for a program running in real mode), to reserve an area of real storage.

On entry, register 2 contains the begin, and register 3 the end address of the area requested. All PFT entries of the page frames in this area are posted as not fixable, and the TFIX counter of each entry is checked for zero (page is not TFIXed). If a page frame is found to be TFIXed, and if fast CCW translation is not active, the requesting task is set to PGFX bound. If fast CCW translation is active, all pages currently held by saved CCB copy blocks are released and the requested area is checked for TFIXed pages again. If such a page is still found in the area, DEFIXCNT is increased by 1 to force fast CCW translation to release the pages of active channel programs at I/O interrupt time, and the requesting task is set to PGFX bound. After posting, DEFIXCNT is decreased by 1 and the area is checked again for TFIXed pages. This is continued until all TFIXed pages are released. If the area requested is free of TFIXed pages, the following steps are executed:

1. If the page frame is unused:
   Remove the PFTE from its appropriate queue, clear the page frame, set the storage key of the corresponding PTE in the page frame with the reference and change bit reset, validate the PTE and insert the leftmost 12 bits of the page frame address into bits 0 - 11 of the PTE. Insert the page address of the new page in the PFTE of the page frame, reset the NFRP bit, and increase the PFIX counter and partition PFIX counter by 1.

2. If the page is connected to a page frame:
   Wait for end of page connected state and take actions depending on the new state of the page frame.

3. If the page frame contains a valid page that is requested by PFIX:
   Get an unfixed page frame and exchange the contents of the two page frames. Take actions according to the new state of the page frame.

4. If the page frame contains a valid page which is not in connected state and which is not requested by PFIX:
   Remove the PFTE from its appropriate queue. If the change bit for the page frame is set, call ENQUOW to write the page onto the PDS. On return, take actions depending on the new state of the page frame.
   If the change bit for the page frame is not set, the page is disconnected using routine PAGDISCA and the actions described in 1. are taken (except for the removing of the PFTE from its queue).

5. If a page frame is found to be unusable because of a hardware error (DRAP bit in PFT entry on):
   No area is allocated when this condition is detected in the first page frame. If the page frame in error is not the first one, the allocated area ends at the start address of the failing page frame.

The following return codes are passed by GETREAL:

0 = The requested area is reserved (PFIXed).
4 = The page frame belongs to failing storage and is not the first page of the real partition.
8 = The page frame belongs to failing storage and is the first page of the real partition.

TFIX Request

A TFIX request is ignored, if it is a VSE system with MODE=VM specified.

The TFIX routine fixes pages temporarily, that is, a page is fixed in a page frame for the duration of an I/O operation. This routine is called by the CCW translation routines, the Fetch routine, the SVC X'2C' routine, and others.

The caller provides in register 1 an address that points to a parameter list consisting of blocks of entries, where each block defines a string of pages to be TFIXed. The blocks are chained and the list is finished by a halfword of zeros in the last block. Register 0 contains the number of entries in the first block.

A TFIX request for p pages which are not already TFIXed or PFIXed n can be satisfied as long as the condition

$$p < NPSQE - MINPSQE + 1$$

is satisfied . Otherwise the requesting task is set into wait. NPSQE is reduced by p pages:

$$NPSQE' = NPSQE - p$$

> **Note:** Here and in the following formulas the new value of the variable xxxx is noted as xxxx', the original value is noted as xxxx.

The length of PSQ is reduced by q page frames:

$$len(PSQ)' = len(PSQ) - q$$

where q = p - r with r = min(FFCC,P) for E-mode, respectively r = min(len(IPFQ),p) for 370-mode. Analogously yields:

$$len(IPFQ)' = len(IPFQ) - r \quad \text{for 370-mode}$$

```
FFCC'     = FFCC     - r    for ECPS:VSE-mode
```

The TFIX counter of all PFTEs is increased in any case:

```
PFTE.TFIXC' = PFTE.TFIXC + 1
```

The following return codes are passed by the TFIX routine:

0 = If the request is issued by the fetch routine and the number of available page frames in the PSQ reaches a minimum, and no page can be TFIXed; or

if the request is not from the Fetch routine and the number of available page frames in the PSQ reaches a minimum, and no page can be TFIXed.

4 = The TFIX counter has reached the maximum value for a page and the page cannot be TFIXed.

8 = All requested pages are TFIXed.

PFIX Request

A PFIX request is ignored, if it is a supervisor with MODE=VM specified.

A PFIX request may be issued by a user task or by the restart (RSTRT) statement processor (Job Control). Actually it is issued by the SVC X'3A' routine if a switch to 'real' is required, by SVC X'43', SVC X'6E', and storage management.

Register 1 points to a parameter list that defines the pages to be PFIXed. If the page is not in storage the request is enqueued to the page queue and the PMR system task is activated.

A PFIX request for p pages can be performed as long as the conditions

- p + SMPFIX < SMAXPFIX + 1
- p < NPSQE - MINPSQE + 1
- PFTE.PFIXC < MAXPFIX for all PFTEs associated to the PFIXed pages
- SMRPBEG - 1 < PFTENR*2**(12) < SMRPEND
  - additionally in 370-mode for all PFTEs

are satisfied. Otherwise the requesting task is set into wait or is posted with a return code indicating no PFIX possible.

The state changes for NPSQE etc are the same as for TFIX. The PFIX counter of all PFTEs is increased:

```
PFTE.PFIXC' = PFTE.PFIXC + 1
```

The pages are PFIXed one after each other and if during this process the free frames are exhausted, all pages which have just been PFIXed are freed again. A special return code, is passed to the requesting task, indicating that the PFIX request cannot be performed under the actual system conditions.

370 mode only:

The page will be fixed immediately, if the page is in real storage and if the following conditions are true:

- The page frame is in the correct real partition. In that case, it is only necessary to increase the PFIX counter by 1 and to remove the page frame from the selection pool if it has not already been removed.

- The page is not TFIXed and the page frame is not in the correct real partition, but a page frame in the real partition is available for PFIXing. The two pages are then exchanged and the page is PFIXed.

PFIX Requests for RSTRT

Handling PFIX requests for the RSTRT routine (Job Control) requires special action because each PFIXed page must be returned to the page frame in which it was located at the time the program was checkpointed. When a page is PFIXed by the RSTRT processor, not only the page address but also the page frame address and the value of the PFIX counter are passed. The address of the reserved page frame is placed in the FIXWTAB entry for the task; the PFIX counter for the page frame is set to one less than its value at the checkpoint, and the page is PFIXed in the reserved page frame.

The following return codes are passed by the PFIX routine:

0 = Function successfully completed.
4 = Maximum number of allowed PFIXed pages for the partition is exceeded by this request only.
8 = Maximum number of allowed PFIXed pages for the partition is exceeded because of previous PFIX requests.
12 = Negative length of area or invalid address.

PAGE-IN Request

A PAGEIN request is ignored, if it is a VSE system with MODE=VM specified.

A valid page-in request is handled by the PGIN system task, which is activated when the SVC X'57' routine has received such a request. The task's dispatching priority is higher than that of the Fetch (SUPVR) task, but lower than that of the page manager (PMR) system task.

The PGIN task runs asynchronously with the requesting user task.

For a page in real storage, the task determines (by looking at the corresponding PFT entry) whether this page is fixed.

- If the page is fixed, the request for the page is ignored.
- If the page is not fixed, its reference bit is set and the associated page frame is enqueued at the end of the Page Selection Queue (PSQ).

For a page not in real storage, the PGIN system task uses the ENQUI routine to have this page enqueued to the page-in queue . The request is then handled like a page-in request that resulted from a normal page fault; however, no exit is taken to a private routine that may be specified in a SETPFA macro in the program which issued the page-in request.

The PGIN system task detects the following error conditions and takes the actions indicated:

- If a page is outside the partition in which the requesting program is executing, the request for that page is ignored.
- If an area specification contains a negative length, the request for that area is ignored.

The task posts an ECB (if one is specified) as shown for SVC X'57' in "Supervisor Call Interrupt (SVC)" on page 28. The ECB's address is obtained from the currently processed PAGETAB entry.

Whenever a task is terminated, the scan routine SCANPGT scans table PAGETAB and deletes all entries that carry the task's TID. If the PGIN system task is processing a page-in request of a task which is being terminated, the PGIN system task stops processing of that page-in request.

TFREE Request

A TFREE request is ignored, if it is a supervisor with MODE=VM specified. A TFREE request is issued by routines such as CCW translation, SVC X'2C' or Fetch, to release TFIXed pages.

Register 1 points to a parameter list that defines the pages to be freed, and register 0 contains the number of entries in the first block (see description of TFIX).

The TFREE request frees p page frames and the TFIX counter of all affected PFTEs is decreased:

PFTE.TFIXC' = PFTE.TFIXC - 1

Only if the conditions

PFTE.TFIXC' = 0
PFTE.PFIXC = 0

are satisfied for q &Al. p + 1 PFTEs, the q related page frames can be used by the page replacement algorithm or - in 370-mode - can be used for PFIX / GETREAL requests. The page frames are inserted in the PSQ; that means:

NPSQE' = NPSQE + q

len(PSQ)' = len(PSQ) + q

Additionally, the tasks waiting for free page frames must be posted if NPSQE' &Ar. MINPSQE.

370 mode only:

Depending on the setting of bits NFRP and NFVP in the PFTE, additional actions may be taken when returning the PFTE to the PSQ:

NFVP=ON:
The freed page is requested by PFIX but the page frame does not belong to the real partition. The task identified by the WID field in the PFTE is posted ready to run.

NFRP=ON:
The freed page frame is requested by PFIX. The address of the PFTE of the freed page frame is inserted in PFTERSVD of PCB (see SVC X'43') and thus reserved for the PFIX request. The task issuing the PFIX request is posted ready to run. All page frames in the partition, except the reserved one, are set to temporarily fixable (NFRP=OFF) before the next request is processed.

FREEREAL and PFREE Requests

The PFREE request frees p page frames and the PFIX counter of all affected PFTEs is decreased:

```
PFTE.PFIXC' = PFTE.PFIXC - 1
```

The conditions for further processing and the processing itself is analogous to that one of TFIX.

For handling of FREEREAL and PFREE requests see SVC X'36' (FREEREAL) and SVC X'44' (PFREE) in "Supervisor Call Interrupt (SVC)" on page 28.

RELPAG and FCEPGOUT Requests

For the handling of RELPAG and FCEPGOUT requests see SVC X'55' (RELPAG) and SVC X'56' (FCEPGOUT), in "Supervisor Call Interrupt (SVC)" on page 28.

VIO POINT Request

The VIO storage is considered as an extension of the page data set. The size of a VIO storage block is equal to the size of a page. To control the VIO storage a number of pages in the address space is reserved for system usage. This area is named V-POOL and is located at the end of the address space.
As a result of a VIO point request, the user gets access to a page out of V-POOL, which contains the requested block of his VIO-file. The next VIO POINT request frees implicitly the block obtained by the previous request (i.e. the user is no more allowed to access it directly).
The system tries to keep as much VIO-blocks as possible in real storage. Therefore, if a block is freed it is not immediately written to page data set but the page representing the block is set in connected state instead. If a page is requested by a VIO POINT request and no free page exists in V-POOL, a available V-POOL page is freed by disconnecting the page and setting the PFTEBLK bit on in corresponding PFTE (370 mode) or by writing the underlaying block on PDS if it was changed and disconnecting the page (ECPS:VSE mode). The page frames occupied by VIO storage blocks are written on the PDS due to paging or if a V-POOL page has to be freed (ECPS:VSE mode).

The VIO storage is managed using the following tables:

- VTAB (V-POOL table) which contains one entry per page in V-POOL

- BLKTAB (block table) which contains one entry per block of VIO storage.

- VIOTAB (vio identification block) one VIOTAB entry exists per open VIO-file.

Handling of VTAB-entries (VTABEs):
Two queues are maintained to handle the VTAB-entries. One the free queue contains all VTABEs which are not connected to a VIO storage block (VTUSCNT<0). The other the available queue contains all VTABEs which are connected to a VIO storage block, but the user is not allowed to access it directly (VTUSCNT=0). VTABEs which are active i.e. the user is allowed to access the page represented by the entry (VTUSCNT>0) are not queued.
To allow enqueue at the bottom and dequeue at the top of the available queue, begin and end of this queue is maintained. For the free queue only begin of queue is maintained.
If due to a free the VTUSCNT reaches zero, the VTABE is enqueued on the bottom of the available queue. If a free VTABE is requested and the free queue is empty the first entry in the available queue is freed.

LOAD LEVELING

In regard to unnecessarily high paging activities in the system - that is thrashing - the page management provides algorithms to measure and to reduce high paging activities. This is done by the deactivation of one or more partitions. Deactivation means, that no paging requests are satisfied for the partition; however, the partition is still in the dispatching queues and may be dispatched.

A second condition for deactivation is the state of the PSQ. Whenever a frame with PFTE.HOLD=ON was found by the selection algorithm before a frame with the reference-change tupel (0,0) or (0,1) had been selected the deactivation is done immediately (without any consideration of the load leveling parameters).

When thereafter the paging activities are dropped under an acceptable level, the deactivated partition(s) can be reactivated.

Load Leveling Parameters

The load leveling algorithm is managed by so called load leveling constants which are determined by size and speed of the processor type.

NPI	Maximum number of page-ins during measurement interval
ACONST	Maximum number of page-ins per second
DCONST	Threshold for Fast CCW Translation
MINTIME	Minimum time interval for reactivation measurement

There are some further variables indicating actual values of the paging environment. They are listed below:

PIDCTR	No. of page-ins for deactivation measurement interval
PIRCTR	No. of page-ins for reactivation measurement interval
TIME1	Begin of reactivation interval
TIME2	Actual time at reactivation measurement
TIMEA	Begin of deactivation interval
TIMEB	Actual time at deactivation measurement
RRCTR	Reentry rate during deactivation interval
RRCTRX	Reentry rate during reactivation interval
EXPAVD	Exponential average of NPI/(TIMEB-TIMEA)
EXPAVE	Exponential average of PIRCTR/(TIME2-TIME1)
EXPAVR	Exponential average of RRCTR/(TIMEB-TIMEA)
EXPAVX	Exponential average of RRCTRX/(TIME2-TIME1)
REACTECB	ECB set up for a timer interval; after posting the reactivation can take place.

As system parameters the following variables are used by the load leveling routines:

IJBAPNO	Number of active virtual partitions
NDEACTP	Number of deactivated partitions

Considerations to the Parameters

Exponential Average of Page-Ins per Second (for Deactivation)

The exponential average is a value which is calculated periodically (every time NPI page-ins have occurred). The old exponential average is used to calculate the new exponential average:

New exp. av. = EXPAVD'=(EXPAVD + (NPI/measurement period))/2

The measurement period is the time between the time when PIDCTR reached NPI (and was reset to zero) and the moment when it reaches this value again.

When NPI page-ins have occurred for the first time after IPL, the old exponential average does not exist. It is, therefore, set equal to NPI/measurement period and then the above formula is applied. Analogously, the exponential average EXPAVR is defined as the reentry rate RRCTR per second during the deactivation measurement interval.

Reentry Rate

The reentry rate is equal to the number of page-ins of pages that were paged-out earlier in the same measurement period. To establish this value, a reentry rate counters RRCTR and RRCTRX are maintained. This counter is set to zero at the start of each measurement period. If the page manager determines that a page which is to be paged-in was paged-out earlier in the same measurement period, it increases the reentry rate counter by one. This procedure makes use of the reentry rate tables RTAB and RTABX, which are bit strings containing a bit for each page in the virtual storage. At the beginning of a measurement period, all bits of RTAB respectively RTABX are set to zero.

When the page manager determines that a page is to be read in from the page data set, the bits in RTAB respectively RTABX corresponding to the page is tested. If this bit is on, reentry is detected, and the reentry rate counter RRCTR respectively RRCTRX is increased by one.

Deactivation Algorithm

After completion of a page-in request the variable PIDCTR is increased by one and tested if it is got equal to the constant NPI. If so, control is passed to the DEACT routines and further condition for deactivation are checked:

```
      if RTAB(page) = ON   ( page previously paged out )
         then RRCTR' = RRCTR + 1
         else RRCTR' = RRCTR

      if RTABX(page) = ON  ( page previously paged out )
         then RRCTRX' = RRCTRX + 1
         else RRCTRX' = RRCTRX
```

Note: Here and in the following formulas the new value of the variable xxxx is noted as xxxx', the original value is noted as xxxx.

```
      if PIDCTR + 1 < NPI
         then PIDCTR' = PIDCTR + 1
         else PIDCTR' = 0
              RTAB'   = 0
              RRCTR'  = 0
              TIMEB'  = actual time
              TIMEA'  = TIMEB'
              EXPAVD' = (EXPAVD + NPI/(TIMEB'-TIMEA))/2
              EXPAVR' = (EXPAVR+RRCTR/(TIMEB'-TIMEA))/2
              if EXPAVD' >= DCONST
                 then free page frames kept by FAST_CCW_X
              if EXPAVD' >= ACONST and 2*EXPAVR' >  EXPAVD'
                 then deactivate
```

If the deactivation conditions are satisfied, the virtual partition with the currently lowest dispatching priority is selected for deactivation. The set of these partitions is given by the formula:

```
      ( part (deactivation)) =

           (part | part = not(POWER or VTAM or ICCF or CICS or OCCF)
                      & part = virtual
                      & part = not(deactivated or TPIN or inactive)
                      & part = not(open ACBs)     )

    if number (part(deactivation)) > 1
          then DEACT_P'  =  min_disp_priority(part(deactivation))
               REACTECB' = 4 sec
               NDEACT'   = NDEACT + 1
               IJBAPNO'  = IJBAPNO - 1
          else DEACT_P'  = not determined
                   REACTECB' = REACTECB
                   NDEACT'   = NDEACT
                   IJBAPNO'  = IJBAPNO
```

Deactivation means that no user page fault will be handled anymore. However, if the deactivated partition owns the LTA or other system resources the deactivation is delayed until the resources are released.

Reactivation Algorithm

Whenever the dispatcher algorithm doesn't find a task ready to run the system enters into ALLBOUND state. During this cycle a load leveling routine checks the criteria for reactivation of partitions - if there are any. There are two different types of reactivation:

- the unconditional and
- the conditional reactivation.

Unconditional reactivation is done if:

- there is no active virtual partition or
- no I/O is queued to any PUBS other than CRT or TP devices

Conditional reactivation is done if:

- exponential average of page-ins not greater than CCONST and
- measurement interval not lower than MINTIME and
- PMR task not active

After completion of a page-in request the variable PIRCTR is increased by one.

```
PIRCTR' = PIRCTR + 1
```

The conditions and actions are :

```
if IJBAPNO = 0      (no active virtual partition)
   then unconditional reactivation
   else if NDEACTP = 0       (no deactivated partition)
       then             (no action)
       else if ( PUB(I/O) pending & not(CRT or TP DEVice) &
                 not(U/R device under POWER) )
            then conditional reactivation
            else unconditional reactivation

TIME2' = actual time
if TIME2'-TIME1 < MINTIME
   then if unconditional reactivation
        then reactivate highest priority partition
        else                  (no reactivation)
   else TIME1' = TIME2'
       EXPAVE' = (EXPAVE + PIRCTR/(TIME2'-TIME1))/2
       EXPAVX' = (EXPAVX + RRCTRX/(TIME2'-TIME1))/2
       PIRCTR' = 0
       RRCTRX' = 0
       RTABX'  = 0
       if conditional activation
          then if 4*EXPAVX' < EXPAVE'
               then reactivate highest priority partition
               else              (no reactivation)
          else reactivate highest priority partition
```

Reactivation means:

```
          if reactivation
             then REACT_P'  = max_disp_priority(deactivated partitions)
                  DEACT_P'  = not determined
                  DEACTP'   = DEACTP - 1
                  IJBAPNO'  = IJBAPNO + 1
                  REACTECB' = 4 sec
             else                    ( no action )
```

After successful reactivation all PDS devices are set to NONEMPTY in order to continue with the possibly already queued page requests for the reactivated partition(s).

The page manager will be activated if it is not yet active and gets control in any case.

Exponential Average of Page-Ins per Second (for Reactivation)

The exponential average of page-ins per second for reactivation is calculated for both conditional and unconditional requests. The calculation is similar to the calculation of the exponential average for deactivation:

New exp. av. = EXPAVE'=(EXPAVE + (PIRCTR/time interval))/2

Note that two other quantities are used. PIRCTR is the page-in counter for reactivation. It is reset to zero after calculation of the new exponential average, and is increased by one each time a page-in occurs. Time interval is the elapsed time between the previous call of the reactivation routines and this call.

The highest priority partition which is deactivated is selected for reactivation. This is done by scanning STATPOWN from left to right (decreasing priorities). When the partition is found, it is reactivated. The byte for the partition in DEACTPSS is posted X'FF' (was X'00'), and the entry in the system communications region indicating the number of active virtual partitions is increased by one.

Fast CCW Translation Restriction

A further measure for reduction of paging activity is to force the fast CCW translation routines (if active) to free all pages containing I/O areas as soon as the channel program has been completed (at I/O interrupt time). This is done, when the exponential average of page-ins per second equals or exceeds the specified constant DCONST, by turning on the free-pages switch in DEFIXCNT.

The fast CCW translation restriction is reset when the following two conditions are fulfilled:

- The exponential average of page-ins per second again drops below the value of DCONST.

- The time specified in MINTIME has elapsed since the last reset.

Teleprocessing Balancing (TP Balancing)

Teleprocessing balancing is a special way of load leveling which is triggered by:

1. The TBAL command (see VSE/Advanced Functions Operating Procedures),

2. The combined use of SVC X'58' (TPIN) and SVC X'59' (TPOUT)

3. The occurrence of page faults.

Teleprocessing balancing is not done for a supervisor with MODE=VM specified.

In a system with both teleprocessing and concurrent batch processing the teleprocessing subsystem may, at certain times, monopolize system resources in order to improve its response time. The performance of batch processing is decreased. TP balancing works via the deactivation string DEATPSS by deactivating one or more of the batch partitions on request. SVC X'58' represents the request for TP Balancing, and is issued by the teleprocessing subsystem. After a certain amount of processing has been completed, SVC X'59' must be issued in order to reset TP balancing.

The TPBAL command allows the operator to turn this special load leveling on or off. If it is off, SVC X'58' and SVC X'59' have no effect. The same is true if there is no page traffic in the system, since a page fault may trigger deactivation. The operator may turn on TP Balancing by specifying the number of partitions in which delayed processing can be tolerated. This number is stored in the TPBAL parameter in the SYSCOM.

Only as many lowest-priority partitions as indicated by the TPBAL parameter are deactivated. The partition that issued the SVC X'58' is always protected from being deactivated.

STORAGE MANAGEMENT

General

The storage management part of the supervisor consists of the following routines:

```
GETVIS   (SVC 61 - X'3D')
FREEVIS  (SVC 62 - X'3E')
CDLOAD   (SVC 65 - X'41')
ALLOCATE (SVC 83 - X'53')
SETLIMIT (SVC 84 - X'54')
```

The first three routines provide a dynamic load facility of system components, as well as work spaces for re-entrant programs. The other two routines allocate and reallocate partitions, distribute real storage, and change partition sizes.

Static Storage Allocation

Static storage allocation is realized by the ALLOC/ALLOCR and SIZE JCL commands which cause permanent partition boundaries. The SIZE parameter of the EXEC statement defines a temporary partition size.

ALLOCATE and SETLIMIT routines are located in the SVA-module IJBSSM. The interface between IJBSSM and the supervisor is established via various communication areas and control blocks, especially the Storage Management Communication Area (SMCOM see Figure 102 on page 247), which is accessible via SYSCOM.IJBSMCOM.

The actual partition boundary (i. e. the boundary between the partition and its GETVIS area) can be found in the corresponding partition communication region at label PPEND (PPEND + 1 = address of partition GETVIS area). All information about permanent partition boundaries can be found in the Storage Management Control Block (SMCB), (see Figure 101 on page 246), which is part of the Partition Control Block (PCB). An address table, pointed to by SYSCOM.IJBASMCB provides addressability to the specific SMCB entries.

SMCB Address Table Format:

Address of SVA Entry	Address of BG Entry	Address of Fn-1 Entry	••••	Address of F1 Entry
0	4	8	12	4xn

n = Number of partitions specified at supervisor generation (NPART)

Note: The pointer to the SMCB Address Table can be found in the SYSCOM at offset X'DC'.

SMCB Entry Format (SMCB)

DEC	HEX	Label	Description
0	0	SMAXPFIX	Partition: PFIX limit in pages System : SVA PFIX limit in pages
2	2	SMPFIX	Partition: PFIX count in pages System : SVA PFIX count in pages
4	4	SMPSAVE	Partition: Save area address System : Reserved
8	8	SMVFLAG	Partition: GETVIS area flags X'80' : SETLIMIT given indicator
8	8	SMVGVIS SMSGVIS	Partition GETVIS area address System GETVIS area address
12	C	SMVPBEG SMSVABEG	Virtual Partition Begin Address SVA Begin Address
16	10	SMVPEND SMSVAEND	Virtual Partition End Address + 1 SVA End Address + 1
20	14	SMRPBEG	Real Begin Address
24	18	SMRPEND	Real End Address + 1
28	1C	<—— Length of SMCB	

Figure 101. Format of Storage Management Control Block (SMCB) and SMCB Address Table

Storage Management Communication Area (SMCOM)

DEC	HEX	Label	Description
0	0	SMALCVSZ	Allocated virtual storage in K.
4	4	SMFSVP	Size of fixed supervisor in pages
6	6	SMPPMIN	Minimum page pool in pages
8	8	SMINSVPX	Minimum system real partition in pages
10	A	SMINPART	Minimum partition size in K
12	C	SMINSIZE	Minimum permanent virtual 'SIZE' in K
14	E		Minimum temporary virtual 'SIZE' in K
16	10		Reserved
18	12		Minimum temporary real 'SIZE' in K
20	14	SMINGTVS	Minimum permanent virtual GETVIS in K
22	16		Minimum temporary virtual GETVIS in K
24	18		Reserved
26	1A		Minimum temporary real GETVIS in K
28	1C	<—— Length of SMCOM	

Figure 102. Format of Storage Management Communication Area (SMCOM)

Dynamic Storage Allocation

Dynamic storage allocation performs the management of the Partition- or System GETVIS area(s). The dynamic allocation is done by means of the Next-Fit algorithm (i.e. Wrap-Around First-Fit).

If the GETVIS area is part of a partition, the length of the specified area must be a multiple of 128 bytes. If it is part of the SVA, it must be a multiple of 16 bytes. If the specified length is not a multiple as required, it is rounded to the next higher multiple of 128 or 16 respectively.

The control information for each Partition GETVIS area is located at the end of the corresponding partition, but with a storage key of zero. Its address can be found at label IJBGVCTL in the corresponding Partition COMREG. The pointer to the System GETVIS control information, which is at the beginning of the System GETVIS area, can be found at label IJBSVIS(ASVIS) in the SYSCOM. The general layout of the GETVIS control information area is shown in Figure 103 on page 248.

GETVIS Area Control Information Layout (MAPGVCTL)

DEC	HEX	Label	Description
0	0	ANCHDIR	Start of 51 CDLOAD Entries
1024	400	BVIRTMEM	Pointer to begin of GETVIS area
1028	404	EVIRTMEM	Pointer to end of GETVIS area
1032	408	BVISTAB	Begin of VISTAB
1036	40C	EVISTAB	End of VISTAB
1040	410	BSUBPIND	Begin of Subpool Index Table
1044	414	ESUBPIND	End of Subpool Index Table
1048	418	BSUBPCHN	Begin of Subpool Page Chain Table
1052	41C	ESUBPCHN	End of Subpool Page Chain Table
1056	420	EGVCTLB	Last byte of control information
1060	424	ENDGVCTL	End of control area
1064	428	GTVSHIGH	Page Chain high water mark
1068	42C	FIRSTPNT	First page within empty pool
1072	430	CURPOINT	Start of chain of last used pages
1076	434	SSEARCH	New start search address (work field)
1080	438	SVWORK1	Save area for register 1
1084	43C	NBRGVPG	Number of pages in GETVIS area
1086	43E	GTVSPGCT	Number of current used pages
1088	440	GTVSMXCT	Maximum number of pages to be used
1090	442	GTVSEXCT	Max. number of pages for excessive requestors
1092	444	MXSUBPLH	Maximum number of subpools available
1094	446	VISTAB	Begin of bit pattern
xxxx	XXX		Begin subpool of Index Table
yyyy	YYY		Begin subpool of Chain Table
zzzz	ZZZ	<—— Length of Anchor Table, depends on length of VISTAB, Subpool Index Table and Subpool Chain Table	

Note: Due to compatibility reasons, the VSAM control information remains at the same location within the GETVIS area, i.e. it has the same offsets relative to PPEND as in former releases. The mapping macro for the VSAM control information is still MAPANCH and contains only this information.

Figure 103. Format of the GETVIS Control Information Area (Anchor Table)

Within each GETVIS area there may be one or more (up to 128) subpools, which are managed separately. Each of the subpools has the following properties:

- A subpool may be created within a partition or the SVA.
- The maximum number of subpools for each partition and the SVA is 128.
- Each subpool consists of a number of pages which are allocated dynamically.
- All GETVIS requests which do not specify a specific subpool are satisfied within a general subpool.
- subpool pages are only contiguous if they are requested contiguous, i.e. when requesting more than one page.
- Empty pages are automatically deallocated from the subpool.
- Each task may own one subpool within a partition for exclusive use.
- Each subpool, except the general and the exclusive subpool, is defined by means of a 8-byte name which consists of a 6-byte user supplied name and a concatenated 2-byte system supplied identifier.
- All subpools, except the exclusive one, may be accessed by each task of the corresponding partition.
- Single SVA subpool pages may be pfixed if requested by the caller.
- SVA subpools may be fetch protected (only for internal GETVIS calls).

Subpool Index Table Entry Format (SUBPINT)

DEC	HEX	Label	Description
0	0	SPITNAME	Subpool name
6	6	SPITNMBR	Subpool number
7	7	SPITFLAG	Subpool flag
		SPFTCHPR	X'01' : Subpool is fetch-protected
		SPPERMID	X'02' : Subpool ID is permanent
8	8	SPITFRST	Ptr to 1st Chain Table entry of subpool
12	C	SPITCURP	Ptr to current Chain Table entry of subpool
16	10	SPITRLVB	Relative current ptr. within current page
17	11	SPITBITO	OR mask for SPITRLVB (impl. current bit ptr.)
18	12	<—— Length of SUBPINT	

Subpool Chain Table Entry Format (SUBPCHN)

DEC	HEX	Label	Description
0	0	SPCHFORW	Subpool page forward pointer
4	4	SPCHBACK	Subpool page backward pointer
8	8		Reserved
12	C	SPCHVSTB	Relative Page VISTAB Pointer
14	E	SPCHFLAG	Subpool Page Flags
		SPPGCONC	X'01' : Next subpool page is contiguous
		SPPGPFIX	X'10' : Page is PFIXed (only SVA)
15	F	SPCHNMBR	Subpool ID (Number)
16	10	<—— Length of SUBPCHN	

Figure 104. Formats of Subpool Index Table and Subpool Chain Table

Due to the subpool function the GETVIS areas are managed on a page basis. For each GETVIS area page there exists one entry in the Subpool Chain Table (see Figure 104). When a GETVIS area is created all pages are chained together and they represent the pool of empty GETVIS pages. The entries are chained in ascending order and the search for empty GETVIS pages always start at the beginning of the pool of empty pages.

All subpools are logged in the Subpool Index Table (see Figure 104). At initialization there exists only the pool of empty pages.

Reservation of the required area is logged in the virtual storage table (VISTAB) of either the SVA GETVIS area or the Partition GETVIS area.

The VISTAB is a bit string. In a partition GETVIS area, each bit in the VISTAB represents 128 bytes. In the SVA GETVIS area, each bit in the VISTAB represents 16 bytes. If a VISTAB bit is 1, the associated 128 or 16 bytes are already allocated; if it is 0, they are free.
Each bit in VISTAB is checked until a string of zeros representing the required length is found. The area associated with this bit string is then allocated by setting each bit to 1.

The area is released by setting the associated bits in the VISTAB to zero. If a whole page is freed it is chained to the top of the pool of empty page(s).

Input for GETVIS service (SVC X'3D'):

R0: Length of requested area
R1:

- Not required or
- Pointer to area to start search (if POOL specified) or
- Pointer to subpool name field (if SPID specified)

R15: Option in low order byte:

X'01':	Page boundary requested (always 2K boundary)
X'02':	POOL specified
X'04':	SVA space requested
X'08':	Subpool specified
X'10':	PFIX requested
X'20':	Exclusive subpool wanted
X'40':	Fetch protection requested (only internal call)
X'80':	Prevent page boundary crossing (only for internal calls &Al. 1 page)

Output for GETVIS service (SVC X'3D'):

R1: Pointer to found area
R15: Return code in low order byte:
See GETVIS Macro description

Input for FREEVIS service (SVC X'3E'):

R0: Length of area to be freed
R1:

- Pointer to area to be freed or
- Pointer to subpool name field (if SPID specified)

R15: Option in low order byte:

X'02':	Subpool specified
X'04':	SVA space to be freed
X'08':	FREEVIS ALL specified EOJ: Invalidate the corresp. partition GETVIS area. EOT: Free the task related exclusive subpool.

Output for FREEVIS service (SVC X'3E'):

R15: Return code in low order byte:
- see FREEVIS Macro description

CDLOAD Support (SVC X'41')

This function loads a phase dynamically into the partition GETVIS area when called by the macro CDLOAD.

Exception: The phase is found in the SVA and the requesting program is not running in real mode.

Before the SVC X'41' routine is invoked, the name of the phase to be loaded (specified by the first operand of the CDLOAD macro) must be pointed to by general register 1.

CDLOAD first checks to see if the GETVIS area control table is already initialized; if so, the Anchor table is searched for an entry for the requested phase. If an entry is found, the return parameters are retrieved from the entry and control is returned to the caller.

If the Anchor table does not exist or does not have an entry for the requested phase, a LOAD is issued with the parameters DE=YES and TXT=NO. The FETCH routine moves only the directory entry for the requested phase into an area specified by CDLOAD (an area at DFWKNAME in the TCB). The CDLOAD routine then checks the directory entry: if the phase is not found, control is passed to ERR22, or the return code is passed. If the phase resides in the SVA, the required parameters are retrieved from the directory entry and passed in registers 0, 1, and 14. In addition, return code X'00' (successful completion) is passed in register 15.

A phase residing in the SVA is not added to the Anchor table. If the requesting task runs in a real partition, a SVA phase is loaded into the corresponding real partition Getvis area.

The phase name is inserted in the first free entry in the Anchor table (see Figure 103 on page 248 and Figure 105 on page 253). SVC X'41' then obtains the length of the phase to be loaded from the directory entry and passes this information to the GETVIS routine.

The GETVIS routine reserves the required storage and returns the load address of the phase to SVC X'41'. SVC X'41' then loads the phase by issuing a LOAD with the parameters TXT=YES and DE=YES. After completion of the load operation, the load point, the entry point, and the length of the phase are stored in the anchor table and in registers 0, 1, and 14, respectively. Successful completion is indicated by passing the return code X'00' in register 15. The layout of the anchor table is shown in Figure 103 on page 248. The layout of an anchor table entry is shown in Figure 105 on page 253.

If the anchor table is full (max. 50 entries) and the phase cannot be stored in the table, return code X'10' is passed in register 15.

Input for CDLOAD service (SVC X'41'):

R1: Pointer to phase name
R15: Option in low order byte:
X'01': Page boundary requested (always 2k boundary)
X'10': Return if phase not found

Output for CDLOAD service (SVC X'41'):

R0: Load address of phase
R1: Entry point of phase
R14: Length of phase
R15: Return code in low order byte:

- See CDLOAD Macro description

Anchor Table Entry Layout (ATENTRY)

DEC	HEX	Label	Description
0	0	ATPHSNME	Phase Name Field
8	8	ATLOADP	Load Point in GETVIS Area
12	C	ATENTP	Entry Point in GETVIS Area
16	10	ATPHSLEN	Length of loaded Phase
20	14	<—— Length of Anchor Table Entry (ATENTRY)	

Figure 105. Format of Anchor Table Entry

PROGRAM RETRIEVAL

External and Internal Interface

The program retrieval provides a set of services either to get the information about an executable program or to load such a program into the storage. The programs are contained in a partitioned dataset, the so called LIBRARY. This library is divided into sublibraries each of these may contain programs (or phases). The services are realized by means of supervisor calls.

The SVCs are:

```
  SVC  X'01'   (FETCH macro)
  SVC  X'02'   (B-Transient load)
  SVC  X'04'   (LOAD and SLOAD macros)
  SVC  X'05'   (A-Transient load)
  SVC  X'17'   retrieves load address; req. can only be JCL or B-trans.
  SVC  X'30'   (C-Transient load)
  SVC  X'33'   (HIPROG macro)
  SVC  X'41'   (CDLOAD macro)
```

The interface to these SVCs is described in "Supervisor Call Interrupt (SVC)" on page 28. Any of the above SVC routines has a common interface to the program retrieval service, the so called FETCH / LOAD service. This interface is described below:

Input:

```
  Register 1 =  address (parameter-list | phasename)

  Register 0 =  null
            | address(loadpoint) for SVC X'02',X'04',X'05',X'30',X'41'
            | address(entrypoint) for SVC X'01'
            | address(area, where loadpoint should be stored, is passed
                    for SVC X'17')

  parameter list = [id,addr(phasename),flag,addr(local-list)]

id   = 00 - for normal LOAD / FETCH
       01 - ICCF load request
       02 - CDLOAD load request
       03 - SLOAD request
       04 - reserved
flag = 80 - return code requested
       40 - SVA load / update
       20 - no SDL search
       10 - reserved
```

```
      08 - directory entry with SDL format
      04 - system search sequence
      02 - directory entry
      01 - bypass program fetch (phase in SVA or TXT=NO)
```

addr(local-list) = address(list) | null

Register 2 = addr (comreg) of pseudo partition if identified as ICCF request.

Output:

If successful, requested directory information and / or phase processing.

```
Register 0  NIL for SVC X'17'
            Address of entrypoint otherwise
Register 1  NIL for SVC X'17'
            null
            address of directory entry in local list
Register 2  NIL for SVC X'17'
            Address of entrypoint (otherwise)

            Return code if requested
```

Note: A load point must be specified for self-relocatable phases.

Structure of the FETCH Environment

The diagram in Figure 106 gives an overview of the flow of control for the execution of a FETCH request.

Part 1 shows the actual control flow, part 2 shows the interrelationship between logic and control blocks.

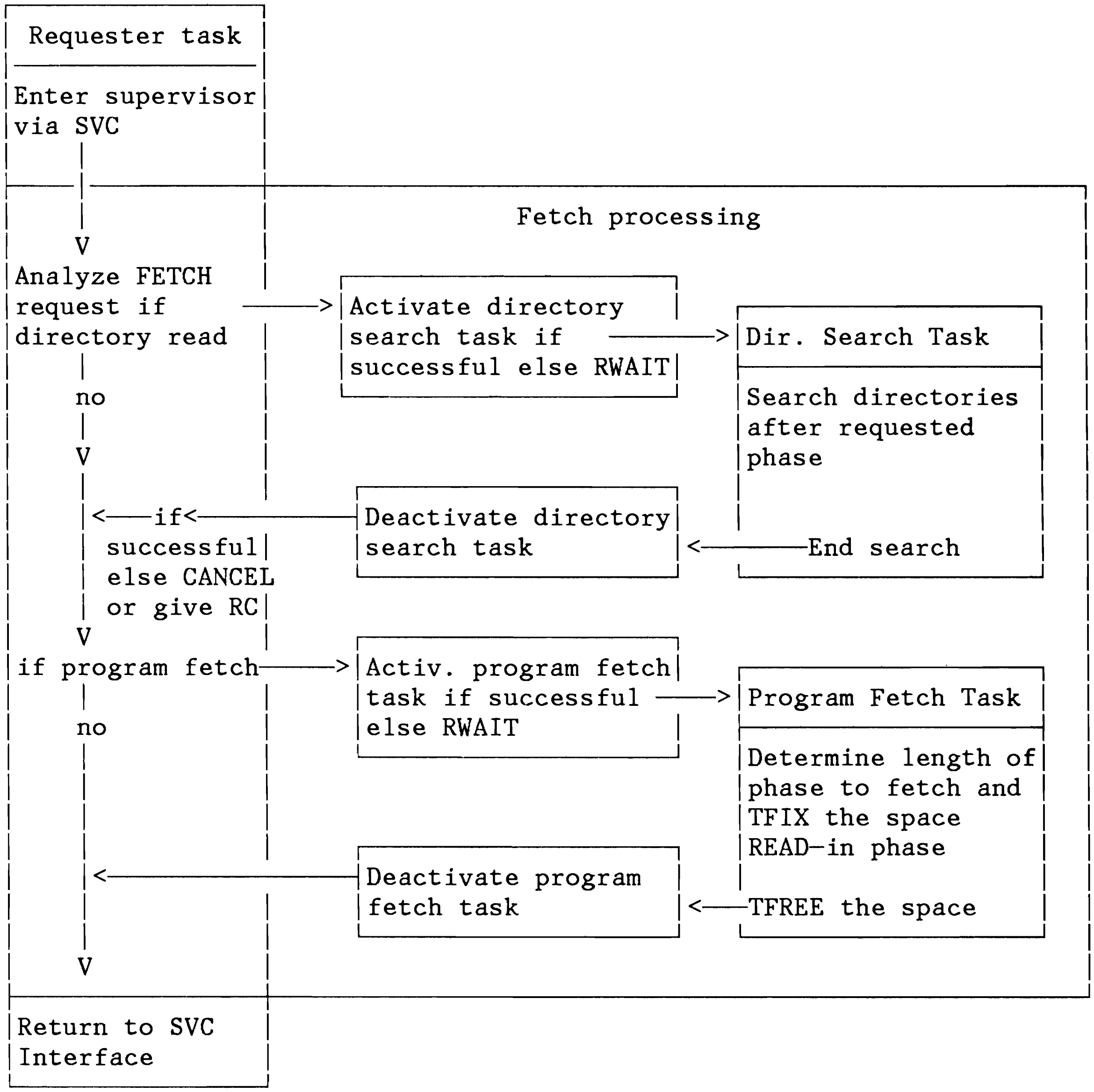

Requester task
Enter supervisor via SVC
Fetch processing
Analyze FETCH request if directory read
no
Activate directory search task if successful else RWAIT
Dir. Search Task
Search directories after requested phase
End search
Deactivate directory search task
if
successful else CANCEL or give RC
if program fetch
no
Activ. program fetch task if successful else RWAIT
Program Fetch Task
Determine length of phase to fetch and TFIX the space READ-in phase
TFREE the space
Deactivate program fetch task
Return to SVC Interface

Figure 106 (Part 1 of 2). Fetch Control Flow

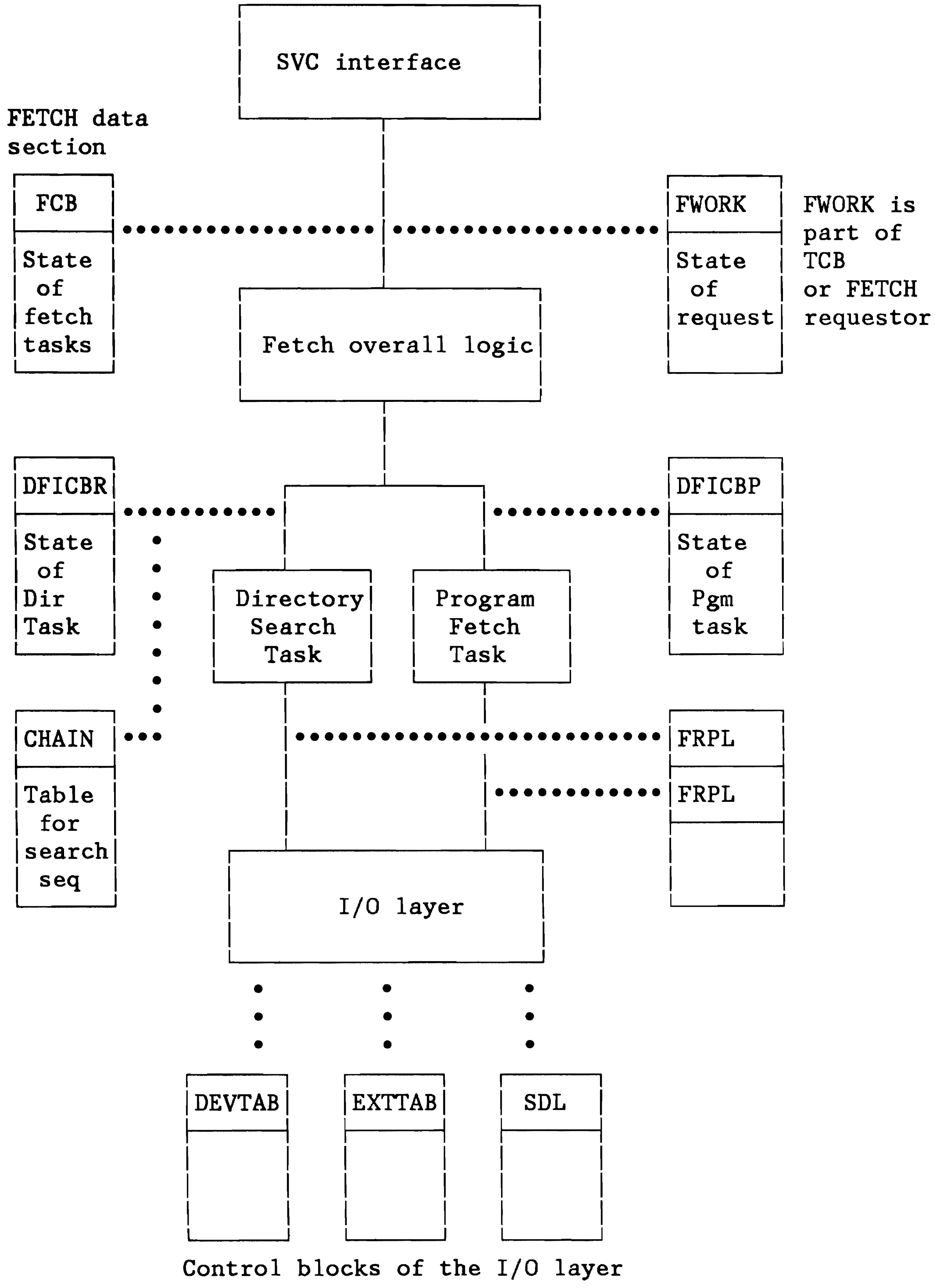


Figure 106 (Part 2 of 2). Fetch Control Flow

Fetch Concept in New Librarian

The new librarian supports a uniform and condense-free library concept. A New Library (NLIB) consists of a non-empty set of sublibraries each may contain members of various types like PHASE, MODULE, PROCEDURE etc. A sublibrary consists of a directory, alphamerically ordered after 'TYPE.MEMBERNAME', and a member space. It may have more than one extent on more than one volume of the same disk device type. For faster search algorithm, the directory can be accessed via an index set (B-tree).

The physical organization of the library is done into so called Library Blocks (LBs) of the size of 1K. The LBs are comparable to the CIs (Control Intervals) in VSAM. A LB contains the data record and VSAM like control information. This is called LBCF and consists of CIDF (Control Interval Definition Field), RDF (Record Definition Field), phase ID and LB chaining field. The next logical LB entity is addressed by the LB chaining field. In such a way the requirement of condense-freeness is satisfied.

As a consequence however, the contiguity of the directory and the space of an individual member cannot be guaranteed. In a frequently updated library respectively sublibrary the degree of fragmentation (directory-, index- and member-space) is increased during its lifetime. The resulting FETCH performance will be essentially decreased. A reorganization of the library is recommended for a proper FETCH performance.

For CKD devices the search on key high or equal is no longer used.

The system library IJSYSRS supports only one extent (on a single volume) and contains at least one sublibrary called SYSLIB. The system library starts on a fixed disk location and contains at least all phases and procedures necessary for IPL.

The library-sublibrary pairs, active in the system, are described by control blocks located in the System GETVIS area. The allocation of these pairs to the VSE partitions is given in the Library Offset Table (LOT).

New Librarian Structure

Library Format

The following figure shows the structure of the NEW LIBRARIAN in such a detail necessary for understanding the FETCH / LOAD processing.

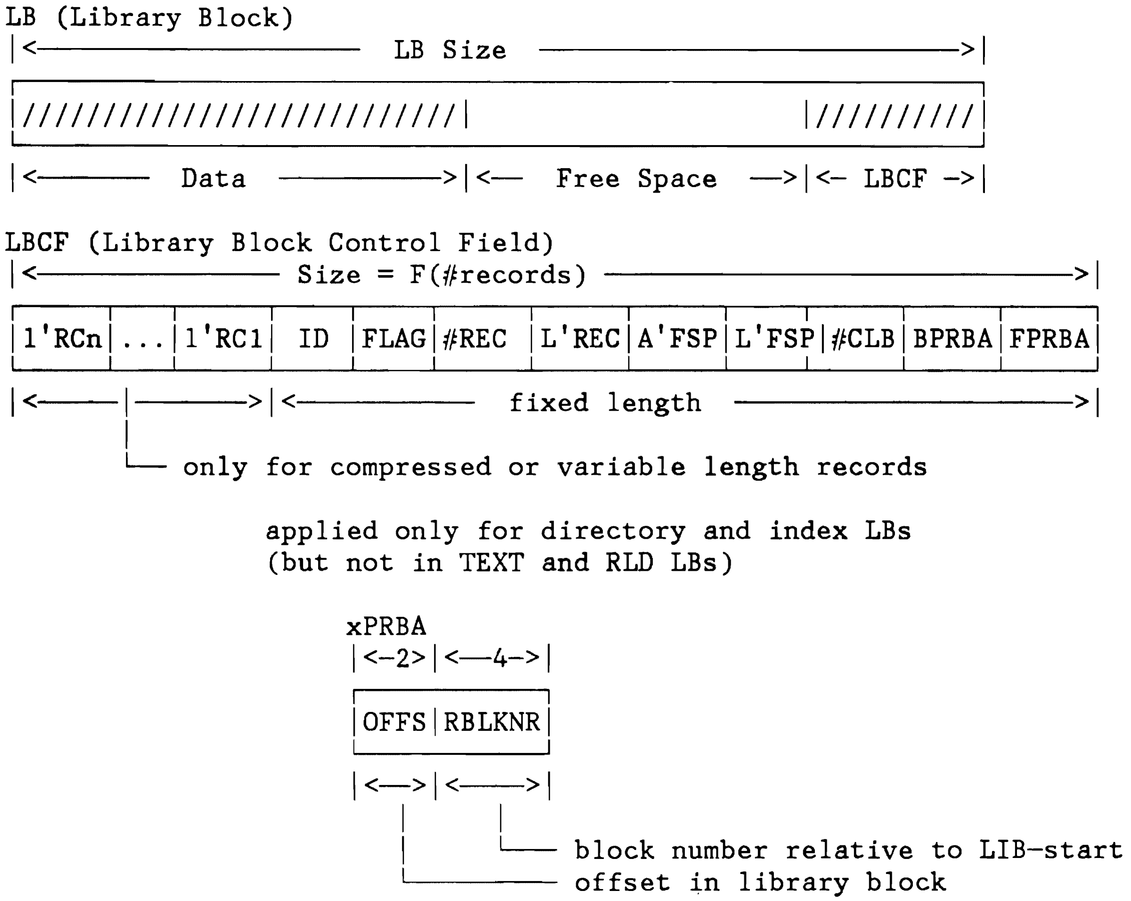

LB (Library Block)
LB Size
Data
Free Space
LBCF
LBCF (Library Block Control Field)
Size = F(#records)
l'RCn
...
l'RC1
ID
FLAG
#REC
L'REC
A'FSP
L'FSP
#CLB
BPRBA
FPRBA
fixed length
only for compressed or variable length records
applied only for directory and index LBs
(but not in TEXT and RLD LBs)
xPRBA
2
4
OFFS
RBLKNR
block number relative to LIB-start
offset in library block

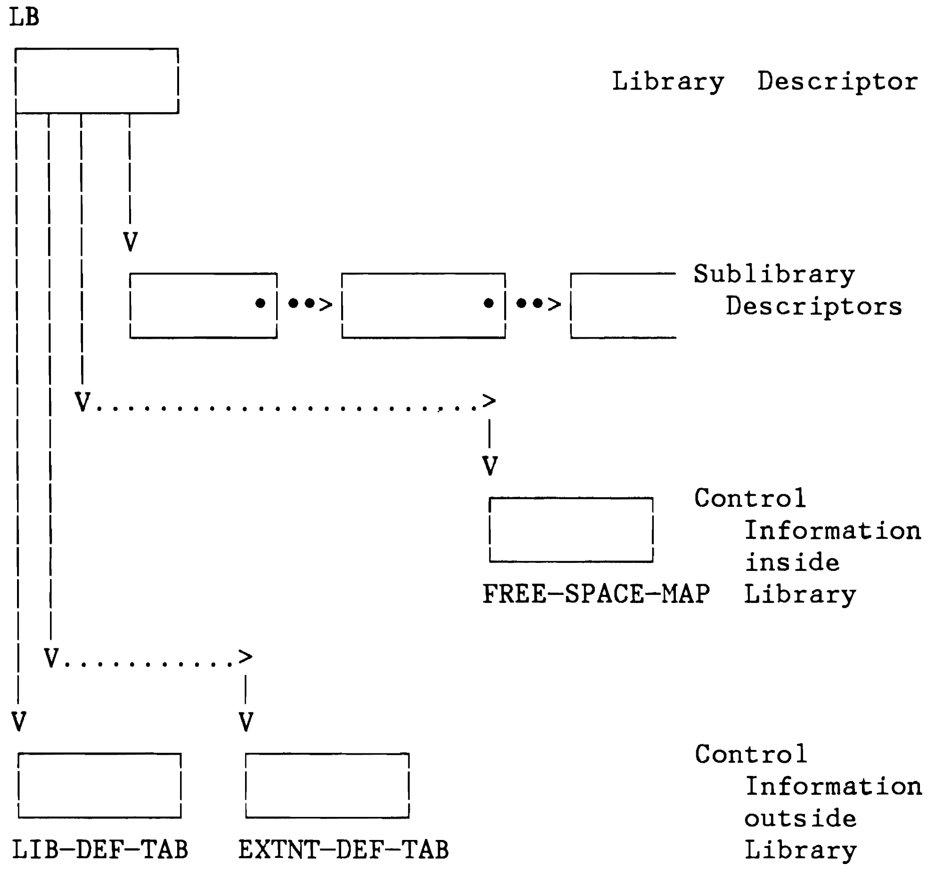
Figure 107. Library Format

The abbreviations are:

F(#records):	function of number of records contained in the LB
l'RCn:	length of record number n (at least one record differs in length from the others)
ID:	phase ID
#REC:	number of records
L'REC:	length of records (if all records of same length)
A'FSP:	begin address of free space
L'FSP:	length of free space
#CLB:	number of contiguous LBs following this LB
BPRBA:	backward pointer RBA (relative byte and block address)
FPRBA:	forward pointer RBA of next logical LB

xPRBA: FPRBA or BPRBA

LIBRARY STRUCTURE



'•' represents the logical LB chain pointers
'.' represents the logical chaining of data entities

Figure 108. Library Structure

Notes:

1. The control information tables are not necessarily located as physical fields in the library. They may be built during "Library Allocation" time by means of label information etc...

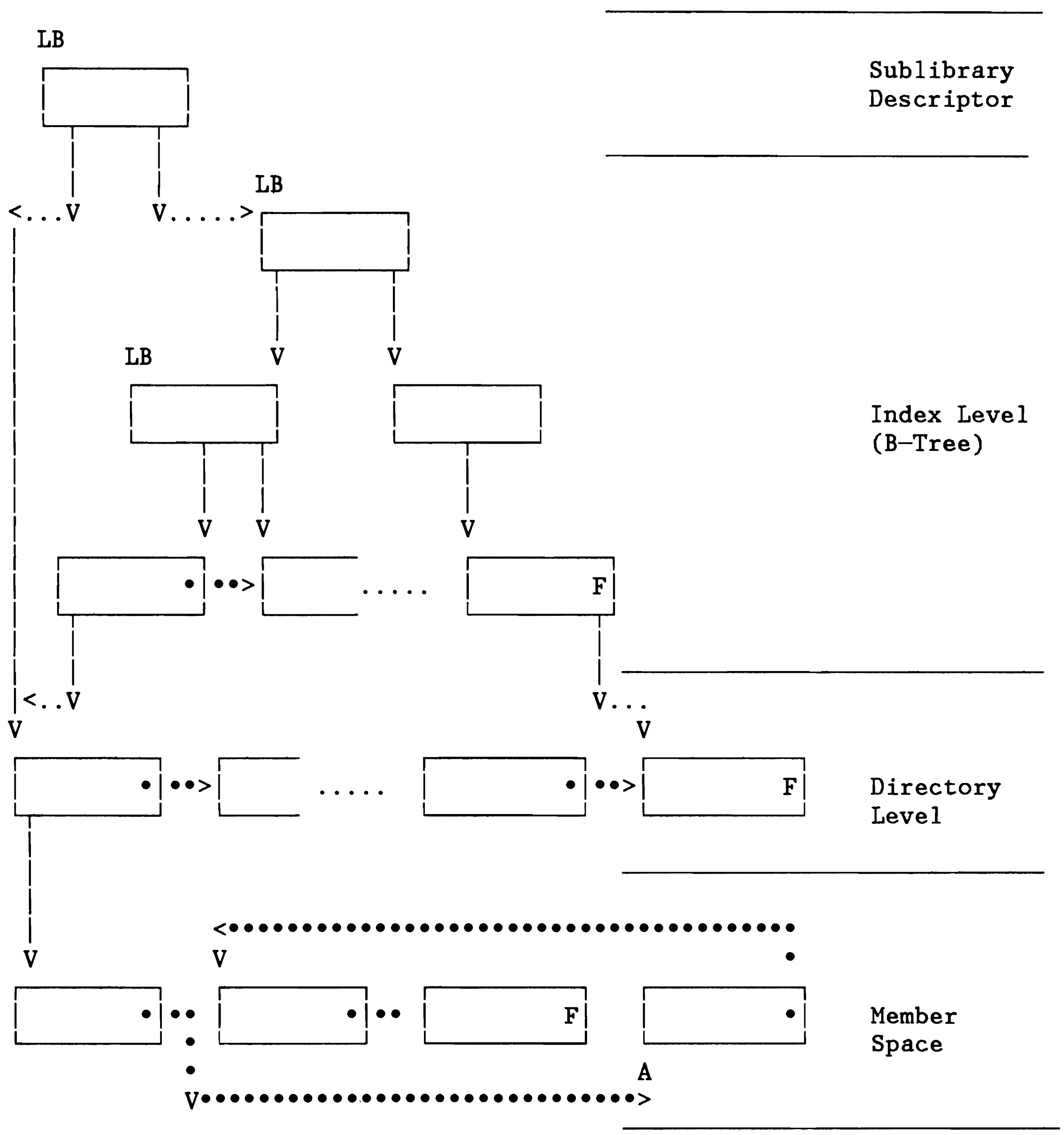
2. All directory LBs are on the same (lowest) index level and are alphamerically sorted after "TYPE.MEMBERNAME". The highest index level of a sublibrary consists of one or more LBs (performance considerations).

3. The data length of TXT, or RLD LBs is L'LB - L'LBCF.

4. The EOB indication for DIR or INDEX LBs is given by: LBCF.L'REC = X'0'

5. End of a logical chain (e.g. member, directory) is given by: FBRBA = X'FFFFFFFFFFFF'.

SUBLIBRARY STRUCTURE

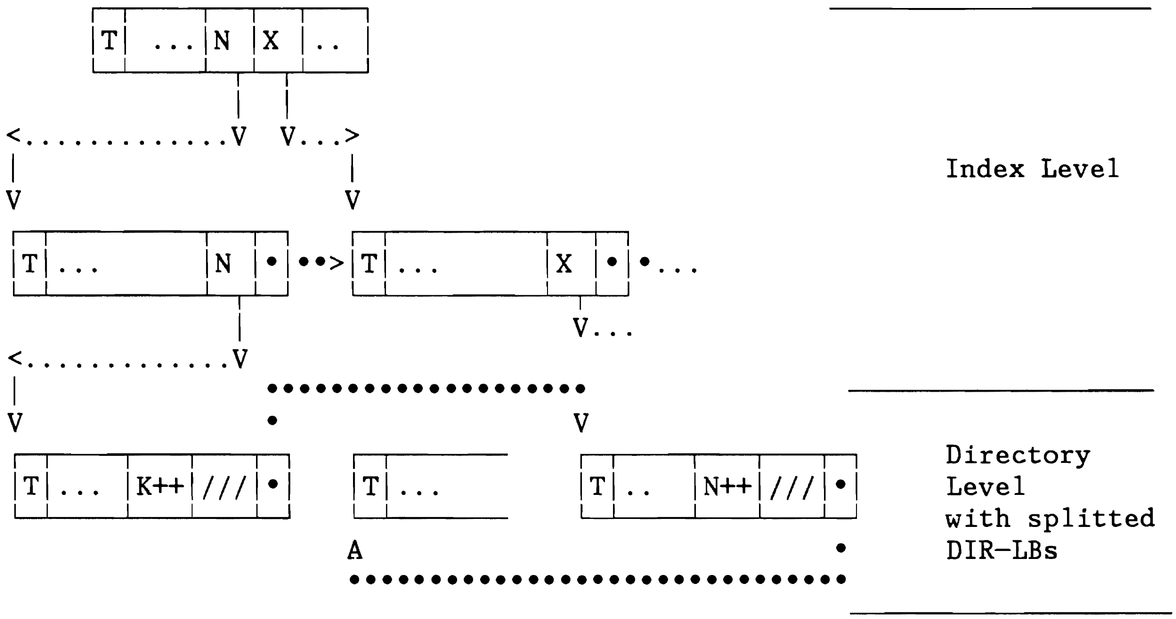


'•' represents the logical LB chain pointers
'.' represents the logical chaining of data entities

Figure 109. Sublibrary Structure

Directory and Index

Each member of a sublibrary is described by a corresponding directory entry. Directory entries on one physical LB are accessible via an index entry in the (next higher) index level. If this index level consists of more than 3 LBs, then a higher index level is provided in order to support a fast search algorithm.
However, at any point in time these relationships might not be valid: a LB-split of a lower level LB can be already successfully performed but is not yet reflected in the higher level index-LB.
In such a case more than one I/O operations must be done for the same index level.
The data part of a directory or index LB may be empty.
As a consequence, the SLD might not be consistent to the directory LBs, therefore the possible LB-split must be considered by the directory search algorithm too.



Note:

T:	TYPE entry
N:	Index entry
X:	Index entry
K++:	Directory entry
N++:	Directory entry

'•' represents the logical LB chain pointers
'.' represents the logical chaining of data entities

Figure 110. Directory and Index

The general format of a directory is as follows:

```
  DIRECTORY      :  header     : descriptor record
                    lb-list    : < dir-LB >
  dir-LB         :  datalist   : data1 | data2 | NIL
                    lengthlist: length-data  | NIL
                    LBCF       : control field

  data1          :  datah1     : <type-entry> v <index-entries>
                    datarest1 : data1 | NIL

  data2          :  datah2     : <type-entry> v <dir-entries>
                    datarest2 : data2 | NIL

  length-data    :  < tail (length-data), head (data i) >

  typ-entry      :  typname    : (PHASE,PROCEDURE,...)
                    typflag    : flag value
                    typdata    : type data

  dir-entry      :  dirname    : name
                    dirflag    : flag value
                    dirdata    : directory data

  Data invariant :  tail(datah1) not= NIL
                    tail(datah2) not= NIL
```

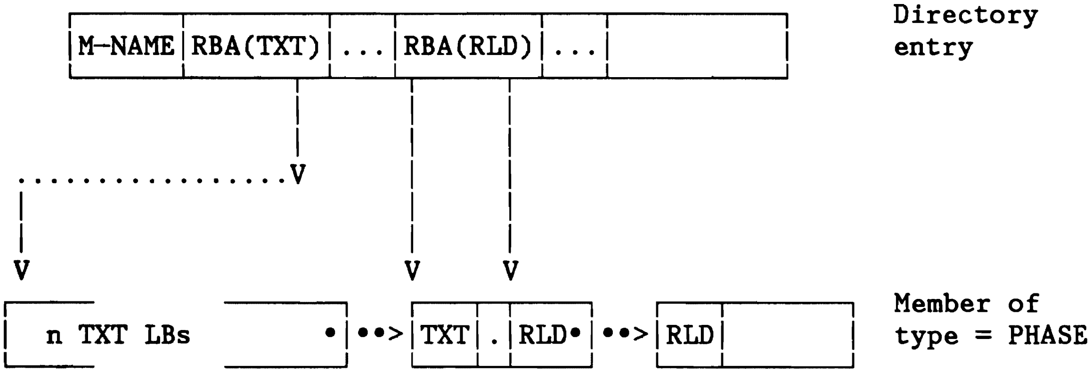
Library Member

A member is the smallest unit of data which is accessed by the FETCH services. A member of the type=PHASE uses the complete data section available on the LB. A member starts always on LB boundary and consists of two different types of information:

TXT: Contains the executable code is cataloged by the Linkage Editor.

RLD: Contains addresses in the TXT to be relocated.

The following diagram shows the relationship between directory entry and member:



'•' represents the logical LB chain pointers
'.' represents the logical chaining of data entities

Figure 111. Library Member

The general format of LB of a PHASE - member is as follows:

```
PHASE      :  PHASELIST : <LB-PHASE>

LB-PHASE   :  DATA      : record
              LBCF      : control field

RECORD     :  TXT       : phasetxt | NIL
              RLD       : < rlditems > | NIL

Data invariant :  RECORD not= NIL
```

Shared Virtual Area (SVA)

The shared virtual area (Figure 4 on page 9) is located in the high end of virtual storage and has a storage protection key of zero. It is built by IPL. The SVA contains:

- A system directory list (SDL) providing a list of either descriptors of programs (phases) located in the SVA or in-storage directory entries of highly used programs (phases) located in the SYSLIB sublibrary of the SYSRES file.

  - The SDL entry is a subset of the directory entry of the library and contains all information required to satisfy the fetch / load services. The SDL has fixed-length entires of 72 bytes. The last entry contains 8X'FF' as phasename. The external directory format is mapped into an internal directory format which is also used as SDL entry format.

- Highly used programs (phases) located in the SVA can be shared between partitions (virtual library). These programs run with the PSW of the requesting task. SVA resident programs must be relocatable and refreshable. If used in connection with VSE/Advanced Functions Fast B- and C-transient Fetch, the SVA resident transients must be self-relocatable. In any case, the programs (phases) must be loaded into the virtual library during IPL or job control time. Any subsequent Fetch request for a B- or C-transient moves the SVA copy of the phase into the LTA/CRT area, instead of loading it from the library on disk.

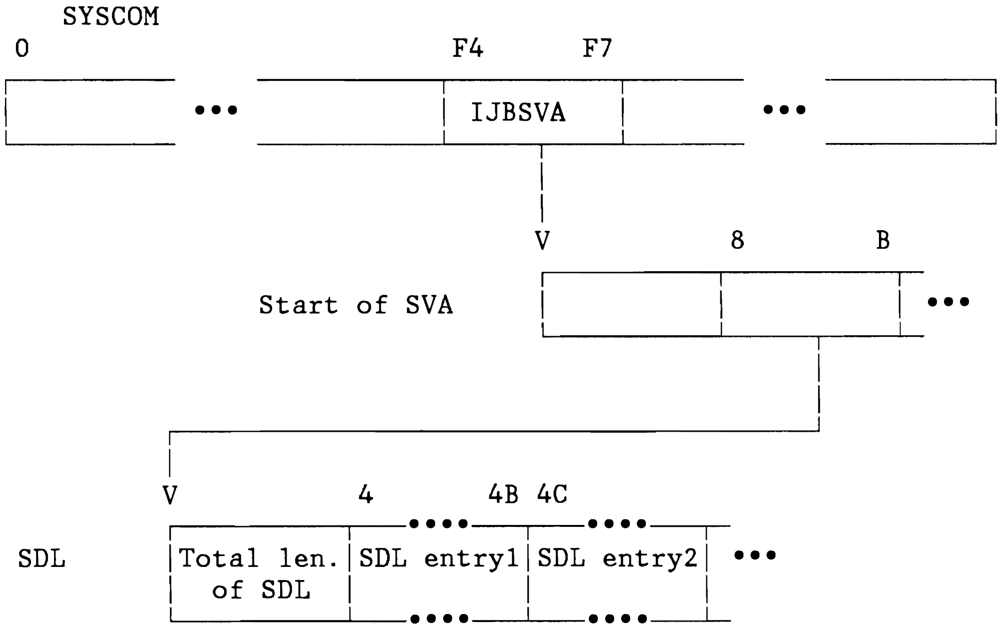


Figure 112. How to Locate SDL Entries

DEC	HEX	Label	Description
0	0	SDLESEG1	Directory Entry (DE) - common segment
0	0	SDLENAM	Member name
8	8		Reserved
9	9	SDLEDEF1	Attributes for DE (flag byte)
		SDLEETYP	X'80' Type of entry = type
		SDLEEHLX	40 Type of entry = high level index
		SDLEEDIR	20 Type of entry = directory
			10 - 01 Reserved
10	A	SDLEPRBA	PRBA of member
16	10	SDLECONT	Number of contiguous LBs
18	12		Reserved
20	14	SDLEPFL	User areal (type = PHASE)
20	14	SDLEFLG	Flags
		SDLEBSR	X'80' Self relocating phase
		SDLEBRL	40 Relocating phase
		SDLEBSE	20 SVA eligible
		SDLEBSV	10 Phase in SVA
		SDLEBPC	08 PCIL flag for incore directory
		SDLEBNF	04 Not found flag (incore directory)
		SDLEBAC	02 Entry active (incore directory)
			01 Reserved
21	15	SDLESWT	Switches
		SDLECLM	X'80' Set SDL: move mode phase
		SDLECLS	40 Set SDL: SVA eligible
			20 - 01 Reserved
22	16		Reserved
24	18	SDLEPLN	Length of phase(TXT) in bytes
28	1C	SDLELPT	Load point at link-edit time
32	20	SDLEENP	Entry point at link-edit time
36	24	SDLESTR	Partition start at link-edit time
40	28	SDLERLD	Number of RLD items
42	2A	SDLERLDA	PRBA of first RLD item if any, otherwise x'FF'
48	30		Reserved
56	38	SDLESVAP	Entry point in SVA if any, otherwise X'00'
60	3C	SDLEIDEN	Library block id
64	40	SDLEALIB	Address of LIB-DEF-TAB
68	44	SDLEASLB	Address of SUBLIB-DEF-TAB
72	48		Total length

Figure 113. SDL Format of a Directory Entry

A program is loaded into the requesting partition only, if it is not in the virtual library.

A phase is loaded into the SVA at the next available double word boundary.

Directory List Support

Directory list support allows the user to create in-storage directories of highly used phases. Once initialized, loads and fetches of such selected phases are made without searching the allocated sublibrary directories on disk. A system directory list, available to all partitions, is provided in the SVA for phases resident in the SVA and for other highly used phases.

Local directory lists may be created by the user at any time. A local directory list exists for the duration of the job step, in which it is created.

It should be noted that an in-storage directory entry in the user's partition does not contain any valid information, except for the phasename, length of directory entry and entry status, until the first FETCH or LOAD request for the phase specifying this entry has been executed. The first FETCH or LOAD request for the phase activates the entry and subsequent requests can use this entry.

If an in-storage directory entry points to a phase which is already deleted, then FETCH reacts as if the 'phase not found' condition had occurred - that means: the phase will not be loaded. Notice, that in previous releases the phase was loaded in this case.

The user macros LOAD, FETCH and GENL generate the new directory entry format if the option DE=VSE is specified. Otherwise a list in old DE-format is generated.
Old versions of LOAD, FETCH, and GENL macros are still supported by the FETCH environment of the NEW LIBRARIAN; a recompilation is not required.
But in regard to NEW LIBRARIAN and security aspects, some directory entry fields are no longer supported or their meaning has been changed.

DEC	HEX	Label	Description
0	0	DIRNAME	Member name
8	8		*** internally used ***
11	B	DIRN	Number of halfword containing user data
12	C	DIRTT	Number of TXT blocks (1024 bytes)
14	E	DIRNN	TXT bytes in last TXT block
16	10	DIRC	Flags
		SELFREL	X'80' selfrelocatable
		RELPHASE	X'40' relocatable
		SVAELIG	X'20' SVA eligible
		SVAPHASE	X'10' phase is SVA-loaded
		PCIL	X'08' not-SYSLIB flag for in-core-DE
		NOTFND	X'04' not found flag
		ACTIVE	X'02' active DE (but possibly not found)
			X'01' reserved
17	11		Reserved
18	12	DIRPPP	Loadpoint at LINKEDT time
21	15	DIREEE	Entrypoint at LINKEDT time
24	18		*** not supported ***
27	1B	DIRAAA	Partition begin at LINKEDT tme
30	1E	DIRVEE	SVA entry point (if SVA-loaded)
34	22	A	*** not supported ***
38	26		Total Length

Figure 114. Layout of the Old LIBRARIAN User DE-Format

DEC	HEX	Label	Description
0	0	DIRNAME	Member name
8	8		X'FFFFFF'
11	B	DIRN	Number of halfword containing User data (X'0E')
12	C	DIRLMBR	Length of phase in bytes
16	10	DIRC	Flags
		SELFREL	X'80' selfrelocatable
		RELPHASE	X'40' relocatable
		SVAELIG	X'20' SVA eligible
		SVAPHASE	X'10' phase is SVA-loaded
		PCIL	X'08' not-SYSLIB flag for in-core-DE
		NOTFND	X'04' not found flag
		ACTIVE	X'02' active DE (but possibly not found)
			X'01' reserved
17	11		reserved
20	14	DIRCOPY	PTR TO DE-COPY
24	18	DIRPPP	Loadpoint at LINKEDT time
28	1C	DIREEE	Entrypoint at LINKEDT time
32	20	DIRAAA	Partition begin at LINKEDT time
36	24	DIRVEE	SVA entry point (if SVA-loaded)
40	28		Total Length

Figure 115. Layout of the New LIBRARIAN User DE-Format

The length DIRN is given in number of halfwords following this field. If the user does not specify the length (field is zero), nothing is moved into the user's directory entry.

Fetch Initialization

Before a FETCH service can be activated, all physical and logical descriptions about the library (-ies) must be available. Especially the control blocks for the SYSLIB (SYSRES) must be initialized before the first FETCH request can be satisfied.

These control blocks are:

- DEVTAB
- EXTTAB (one entry only because SYSLIB consists of one extent)
- LIBRARY DEFINITION TABLE for the IJSYSRS file
- SUBLIBRARY DEFINITION TABLE for SYSLIB sublibrary

Functions and Algorithms

The control blocks and their related functions are as follows :

```
  EXTTAB ==
     init1(SYSCOM,GETVCE(IJSYSRS),SYSLIB-PUB)

  DEVTAB ==
     init2(EXTTAB(SYSLIB),SYSLIB-PUB)
```

The access path to index set is as follows:

```
  start(IJSYSRS)---RBA --->Lib-Descr(IJSYSRS)
                ---ptr --->Slib-Descr(SYSLIB)
                ---PRBA--->Index-Set
```

The relationships between the control blocks are the same as for the NEW LIBRARIAN. A so called system searching chain is established during the FETCH initialization and will be maintained by the LIBRARIAN services.

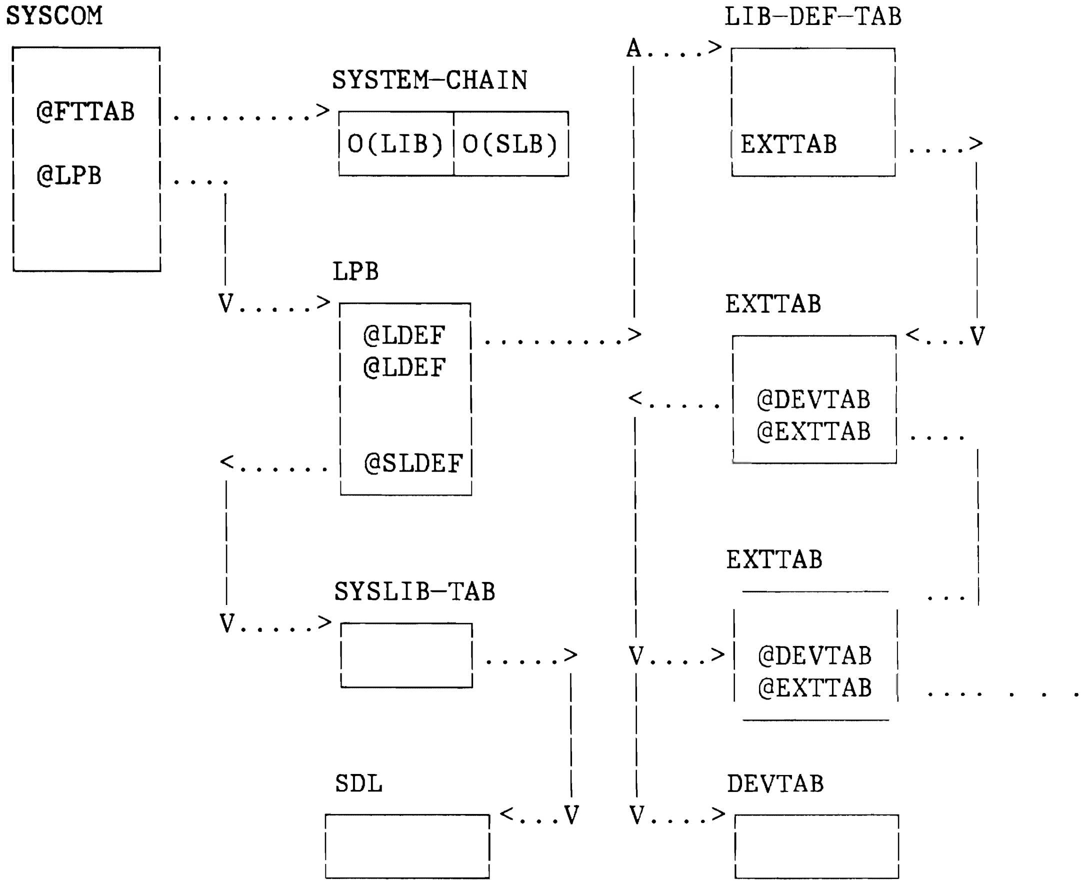


Figure 116. Relationship Between Library Control Blocks

Notes:

1. The SLD of the SYSLIB is built by the librarian at end of IPL time.

2. The meanings of the various control blocks are given below.

FETCH/LOAD Processing

The SVC interface routine at SGCFCH passes control to the fetch overall logic routine in SGDFCH. Before entering the fetch routine, the return address is stored in register 11. Moreover, in register 9, a parameter is stored indicating the SVC interface routine that requested the fetch routine. The meaning is as follows:

0	=	Requested by SVC X'17'.
4	=	Requested by any other routine.
8	=	Requested by SVC X'33'.

Registers 8 through 14 are saved in the requestor's TCB. The user supplied register 1 points either to a parameter list or to an entry of the fetch/load list or a phasename (each time an 8-byte area). For a more detailed description please see the section 'External and Internal Interface'.

Directory Searching Sequence and Directory Entry Processing

The directory search is performed by the directory search task. Prior to accessing the directories, the searching sequence must be determined. The searching sequence depends on the following conditions:

1. Request given by attention task

   a. SDL (system directory list)
   b. SYSLIB directory

2. Request given in test-mode (COMREG byte 59, bit 5=1)

   a. VIO-directory, if any
   b. Temporarily chained sublibrary directories (SDL)
   c. Permanently chained sublibrary directories
   d. SYSLIB directory

3. No test-mode and ($-phase or SYS=YES)

   a. SDL
   b. SYSLIB directory
   c. Temporarily chained sublibrary directories
   d. Permanently chained sublibrary directories
   e. VIO directory, if any

4. No test-mode and non-$-phase and SYS=NO

   a. VIO directory, if any
   b. SDL
   c. Temporarily chained sublibrary directories
   d. Permanently chained sublibrary directories
   e. SYSLIB directory

However, a directory search is not necessary if the user has provided an active directory entry for the requested phase. Such a directory entry has been built as a result of a preceding FETCH/LOAD request. It can be provided in one of the following ways:

- As a directory element to which the phase name parameter is pointing (DE=YES in the FETCH/LOAD macro)
- As a directory entry in a local directory list (LIST parameter in FETCH/LOAD macro)
- As an SDL entry (for special system services)

  Note: It is an essential prerequisite that the FETCH must not be locked against LIBRARIAN services.

If the user has passed a local list, this list is validated and searched for the requested phasename.

If no active directory entry has been provided, the directory search task is activated and the first level chain (FETCH CHAIN) is built.

The so called first level entries are available for SDL and SYSLIB, whereas so called second level entries are reserved for the concatenation chain. In the later case, the addresses of the Library Definition Table (LDT) and the Sublibrary Definition Table (SDT) are calculated by means of the actual entry in the LOT (Library Offset Table).

The directory search operates on a set of control blocks described as follows:

- DEVTAB (Device Definition Table)
  The DEVTAB describes the library device in all its physical aspects, such as device types and device characteristics.
- EXTTAB (Extent Definition Table)
  The EXTTAB describes the location of the library on a device and provides the relation to the RBA addresses. Moreover, it contains the PUB index of the device.
- LPB (Library Pointer Block)
  The LPB is the focal point for any access to chained libraries. It provides the maximum number of entries in the search chain (=maximum number of chained sublibraries) and addresses to the searching chains of library-sublibrary pairs. There is a

temporary S-chain (search chain) which is reset at EOJ time and a permanent S-chain which must explicitly be reset. The existence of such a search chain is considered by the searching algorithm. The LPB is addressable via SYSCOM. The address pointer will be negative (X'80000000'), if the control tables are not yet initialized.

- LOTxxxx (Library Offset Table)
  The LOTs (one for a permanently assigned library chain and one for a temporarily assigned library chain) describes the various S-chains of a specific library type in the various partitions. It can be imagined that the S-chain of a partition is represented by one row of the related LOT and this row is accessed by means of the LOT pointer in the LPB, the PIK, and the maximum number of chained libraries. The fields relevant for FETCH are:

- VIO library
  The VIO library has no separate description. Essentially it is identified by its related VIORB. An address to the VIORB is given by a special LOT row.

  The FETCH searching algorithm works on an internal control table which is built for each FETCH request. All information of the searching chains in the LOTs is mapped into this internal table. Thus, the complete searching mechanism is staged in three levels:

- The FETCH chain table DSRCHNx located in the fixed part of the supervisor reflects the searching chain described above. The entries for SDL and SYSLIB are filled, while the other entries are dummy entries only. By this way any unnecessary page fault is avoided if the phase is found in the SDL or on the SYSLIB ($-phase) The essential information are the address of the DEVTAB and EXTTAB.

- The searching chain of the LOT is accessed whenever an entry in the DSRCHNx is found indicating permanent or temporary chain or VIO-library. If the chain entry is active and the DSRCHNx is not yet initialized, the related LOT is accessed and the first chain entry is taken in order to activate the DSRCHNx entry. otherwise the next chain entry will be taken as long as there are active entries. In the case of end of chain the next DSRCHNx entry will be processed.

- The DEVTAB and EXTTAB entries are required to read on the physical library device.

  A FRPL for directory (DIR) respectively VIO read must be set up for each search of a sublibrary. Moreover, the related addresses of the LDT and SDT for DIR-read respectively of the VIORB for VIO-read must be provided in FCHWORK.

If finally the requested phase is found its directory information is built up in the FCHWORK for further processing.

> **Note:** FCHWORK is part of the requester's TCB. If the directory entry is found in the SDL and the corresponding phase resides in the SVA, no further processing is done. The entry point address, available in the SDL entry, is passed to the user.

If unsuccessful, the user is notified by a 'not found' indication in the directory entry or by a return code in general register 15 (as RET=YES has been specified) or is canceled with the message 'phase(name) not found'.

After a successful search, the user provided directory entry will be activated and updated.

Functions and Algorithms

The directory search is structured into two levels, a logical level determined by the searching chain and a physical level for the I/O operations. On the logical side the related control blocks are FCHWORK and FETCH-CHAIN; on the physical side DEVTAB, EXTTAB and SLD are concerned.
The directory search mechanism is provided by the following control blocks and their related functions:

```
FETCH-CHAIN ==
         bldchain ( PARM-LIST, state )

FETCH-CHAIN (entry) ==
         nxtchain ( LOT-CHAIN )

LB-DIR-ENTRY ==
        scandir ( phasename, FETCH-CHAIN )

SDL-DIR-ENTRY ==
        binsrch ( SDL )

FCHWORK == bldwrk ( .-DIR-ENTRY )
```

> **Note:** The functions BLDCHAIN and NXTCHAIN build together with the FETCH-CHAIN control table an abstract data type.

The initialization of the FETCH-CHAIN control table is represented by the following algorithm:

```
  bldchain (parm-list,state) ==

  select

     when state = ATTENTION-mode
     then FETCH-CHAIN := (SDL,SYSLIB)
     when state = TEST-mode
     then FETCH-CHAIN := (LOT-TEMP or SDL,
                          LOT-PERM,SYSLIB)
     when state = SYS-mode
     then FETCH-CHAIN := (SDL,SYSLIB,LOT-TEMP,
                          LOT-PERM)
     when state = USER-mode
     then FETCH-CHAIN := (SDL,LOT-TEMP,
                          LOT-PERM,SYSLIB)

  endselect;
```

The algorithm for provision of the first / next entry of the LOT-CHAIN is given by the following program:

```
  nxtchain (LOT-CHAIN) ==

  if entry(FETCH-CHAIN) = EMPTY
     then state.LOT-CHAIN := not EOL
     else
  endif
  get-next(LOT-CHAIN)       / may post EOL ...
                  ... if no more valid LOT entry/
  if state.LOT-CHAIN = not EOL
     then entry(FETCH-CHAIN) := entry(LOT-CHAIN)
     save-ptr(LOT-CHAIN)   / save addr of current...
                  ... of actual LOT entry       /
     else
  endif;
```

The searching algorithm on the LBs is defined by the following program:

```
scandir(phasename,entry(FETCH-CHAIN)) ==

state.DIR := not EOF
do while state.DIR = not EOF
   DIRREAD (ENTRY(FETCH-CHAIN))
                    / EOF if no more dir-LBs        /
   do while state.LB = not EOB or state.DIR = not EOF
      get-next( LB )
                    / EOB if LB is empty or processed/
      select
        case phasename = name(LB-entry)
           then DIRENTRY := LB-entry
                state.DIR := FOUND & EOF
        case phasename < name(LB-entry)
           then state.DIR = (not FOUND) & EOF
      endselect
   enddo
enddo;
```

The algorithm for searching the sublibraries is given by the following program:

```
find(phasename,parmlist,state) ==

FETCH-CHAIN := bldchain(parmlist,state)
do while state.DIR = (not FOUND) or FETCH-CHAIN = EOL
   get-next(FETCH-CHAIN)
                 / EOL if FETCH-CHAIN is processed/
   if entry(FETCH-CHAIN) = 2NDLEVEL and FETCH-CHAIN = not EOL
      then entry(FETCH-CHAIN) := nxtchain(LOT-CHAIN)
                 / FETCH-CHAIN = EOL IF LOT-CHAIN /
                 /               IS PROCESSED     /
      else
   endif
   if LOT-CHAIN = not EOL
      then if entry(FETCH-CHAIN) = SDL
        then DIRENTRY := binsrch(phasename,SDL)
        else DIRENTRY := scandir(p-name,entry(FETCH-CHAIN))
                 / state.DIR = (not FOUND) if phase not In dir/
        endif
      else
   endif
enddo;
```

Program Fetch Service

The program fetch task provides services for:

- Load-in of phases (TXT processing)
- Address relocation (RLD processing)

In opposite to the DIRECTORY SEARCH TASK, the library and the sublibrary are known. The addresses of the related control tables are part of the internal directory entry.

Essentially, the related TXT and RLD LB's must be read in and be processed. To do so, CCW-programs have to be generated. If the storage is virtual, the input space must be TFIXed before any read request can be performed. The size of TFIXed area is calculated via the number of contiguous TXT-LBs. But the CCWs are generated in dependence of the actual TFIXED space (not all space might be TFIXed).

For TXT processing the first RLD-LB is read-in with the first TXT-LBs (via chaining of TXT and RLD CCW-programs due to performance reasons).
This can only be done, if RLD and TXT are located on the same DASD device.

The offset of the RLD in a LB is provided in the directory entry (type=phase user information part).

Program Fetch Interface

The program fetch task operates on a library member. Therefore the library device and the (absolute) disk address must be known (or at least derivable).

The related directory entry is provided in the FCHWORK area before activation of the PROGRAM FETCH task. It contains the relative block numbers of begin of TXT or RLD. The disk device address and the disk address must be calculated by means of DEVTAB and EXTTAB information.

The addresses of the related DEVTAB and EXTTAB are saved in FCHWORK too.

The interface to the I/O layer is the FRPL. As for the Directory search the FRPL must be initialized before the first read request for TXT or RLD LBs can be performed.

ALGORITHM FOR TXT PROCESSING

The algorithm for TXT processing is as follows:

```
gettxt (fchwork) ==

BEG-PHASE := function( d-entry(phase),loadpoint)
LEN-PHASE := function( d-entry(phase) )
END-PHASE := BEG-PHASE + LEN-PHASE
RELO-FACT := function( loadpoint,loadpoint(LINK-EDIT))
validate (BEG-PHASE,END-PHASE)
if RELO-FACT = not 0
   then read(RLD-LB)
   / might result in chaining RLD-CCWs to TXT-CCWs/
   else
endif
   BEG-READ := BEG-PHASE
do while LEN-PHASE > 0
  LEN-READ := function (contiguous TXT-LBs)
  do while LEN-READ > 0
    if is-address-space virtual
      / only the contiguous part is TFIXed /
      then TFIX (BEG-READ,LEN-READ)
      else
    endif
  read(TXT-LBs,l'TFIXED space)
  process(RLDs,REL-FACT)
    if is-address-space virtual
      then TFREE(BEG-READ,l'TFIXED space)
      else
    endif
    LEN-READ := function(l'TFIXED space)
    BEG-TXT  := BEG-TXT +  LEN-READ
  enddo
  LEN-PHASE := LEN-PHASE - LEN-READ
enddo;
```

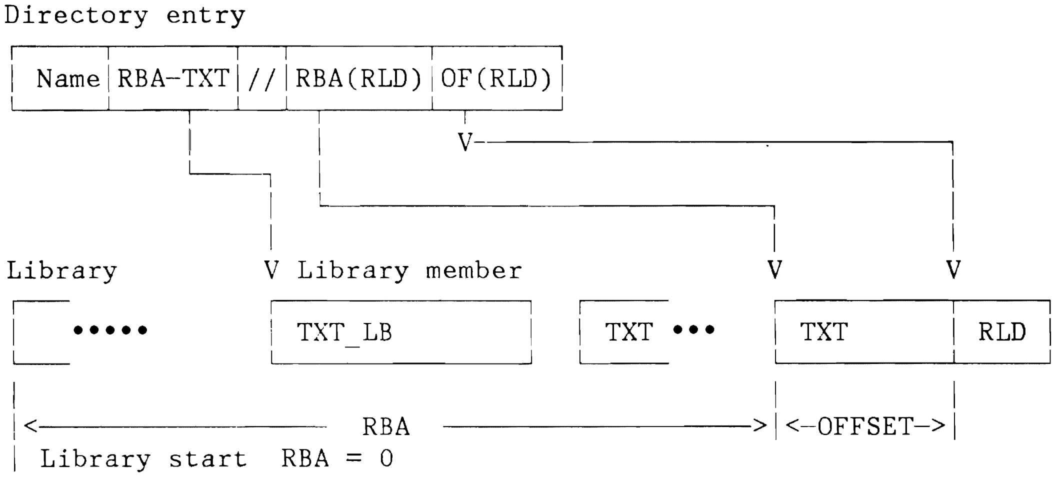
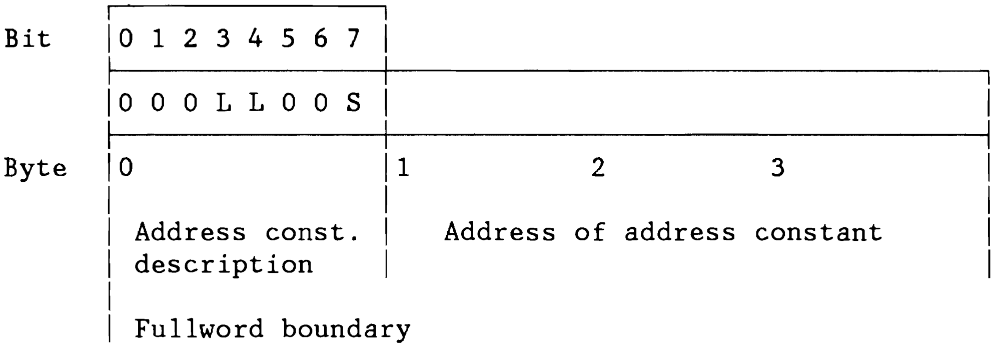


Figure 117. Relationship Between Directory and Phase-Member

RLD Processing

The FETCH / LOAD services relocates the address constants given in the TXT part of the requested phase.
The load point of a phase is either provided in general register 0 or is implicitly determined by the load address at linkage edit time. In the later case the load point is the partition start address (behind the save area) plus the difference between load address and partition start address at linkage edit time (given in the directory entry).
The load point of a self-relocatable phase must always explicitly provided.

The relocation of the address constants is performed for relocatable phases. Such a phase contains additional information of the location of address constants, the so called RLD items.



```
Bit    |0 1 2 3 4 5 6 7 |
       |0 0 0 L L 0 0 S |                                      |
Byte   |0               |1          2          3               |
       |                |                                      |
       | Address const. |  Address of address constant         |
       | description    |                                      |
       |
       | Fullword boundary
```

LL = Object length of address constant in TXT
S = Relocation factor application
     0 = Add
     1 = Subtract

Figure 118. Layout of RLD Items

I/O Processing

The I/O layer handles all I/O operations for the FETCH/LOAD processing. It provides a control block interface, the so called FRPL. This FRPL must be built for each sequence of I/O operations (like TXT-read-in) and identifies the CCW program to be used, the record and the block-length, the number of records to be read-in and the input area. Its format is described in the section Control Blocks.

Directory Read Algorithm

The directory read algorithm is given below. The input parameters are the FRPL, the phasename and the (sub-) library'S EXTTAB, DEVTAB and SLD.

```
dirread (FRPL, phasename, sublib) ==

do while FRPLOPC = 1strd
   get-acc (sublib)
   FRPLOPC: = nxtrd
   IF sublib.SLD = active and not in back level state
      then call SCANSLD (phasename)
      /searches SLD - returns ok or/
      /                       phase not found in SLD/
      else
   endif
   if sublib.SLD = (inact v in back level state
                           v phase not found in SLD)
      then get-LB-addr (index)
      do while index = not processed
         call REQIO   / read index-LB     /
         get-LB-addr (phasename)
      enddo
      else
   endif
enddo
call REQIO         / read directory-LB    /
save (LBCF.FRBA);  / save addr of next-LB /
```

The directory or index LBs are read into the DIRBUF area, part of the pageable supervisor and located on page boundary.

As all other internal FETCH input buffers, the DIRBUF is TFIXED whenever a directory read request must be performed. The related CCWs must be translated (370 mode only).

TXT and RLD Processing

The I/O of TXT and RLD LBs is performed by a generated CCW program for the TXT LBs and -if appropriate- a command-chained RLD CCW program. The TXT CCWs are generated in a special area, the so called GENarea.

Layout of the I/O Buffers

```
┌────────────────────────┬────────────────────────────┬──────────────────┐
│   DIRBUF               │          RLDBUF            │     GENAREA      │
├────────────────────────┼────────────────────────────┼──────────────────┤
│<──────   L' LB ── ───> │<───────   L' LB   ───────> │<──── 512 ────>   │
│                        │                            │                  │
│                        │                            │                  │
│<..Page boundary        │                            │                  │
   ( 370 mode only)      │                            │CCW──>....<──IDAL │
```

Note: CCW generation is done upwards
IDAW generation is done downwards

Figure 119. Layout of I/O Buffers

The algorithm for TXT read-in is given below:

```
txtread(fchwork,FTTAB,EXTTAB)==

do while FRPLOPC = 1strd
   FRPLOPC = nxtrd
   get-LB-addr (phase)
enddo
do while FRPLNRC > 0
   call REQCCW
     / provide space for next CCW  /
   do while FRPLNRC > 0  &  not EOG
             / EOG = END OF GENERATION /
       generate-CCW (FRPL)
       if mode = 370
          then if IDAL = yes
                then call REQIDAL
                  / provide space for idal /
                generate-IDAL(CCW-addr)
                else provide-REAL (CCW-addr)
               endif
           else
        endif
       call REQCCW
       FRPLNRC = FRPLNRC - 1
   enddo
if last TXT LB processing
   then if len (TXT) < len(LBCIF)
           then  if last TXT LB not contiguous
                then call REQLBLK
           /adjust CCWs to read LBCIF(2nd last TXT-LB/
                else call REQFBA
           /do not read LBCIF(2nd last TXT-LB)       /
                     set-CCW-len (FRPLLRC)
                endif
            else
         endif
   else
endif
if RLD = delayed
   then chain (RLD-CCW)
   else
endif
call REQIO/ perform I/O request /
enddo;
```

RLD Read

A RLD read request supplies the information necessary to relocate the address constants of the relocatable phase to be fetched or loaded. The RLD-LB s are read into the RLD-buffer, from where the RLD items are processed by the program fetch task.

The necessary data are:

- Start address for the library is available in the EXTTAB
- Relative start address of the phase is available in the directory entry as a library block number.
- Relative start address of the RLD item

The first RLD block is read-in with the first TXT blocks (by means of chaining of TXT and RLD CCW programs). For any further RLD LB a separate SVC X'0F' must be issued.

DASD SHARING ENVIRONMENT

A DASD sharing environment is built of two or more CPUs which are operating on common DASD devices. In general there is no direct signalling between the CPUs. Any data access control must be done via gating the shared DASD devices. The related hardware facilities are the DEVICE-RESERVE and DEVICE-RELEASE commands. That means gating on device level.

The software however wants to provide a locking facility for the entity 'data set'. Therefore a special data set (the LOCK FILE) is established by the software, which describes the locks of all DASD shared resources in the system. Only the device containing the LOCK FILE is protected by the hardware facilities.

The FETCH is concerned by DASD sharing in regard to PHS-LIBs and corresponding SLDs.

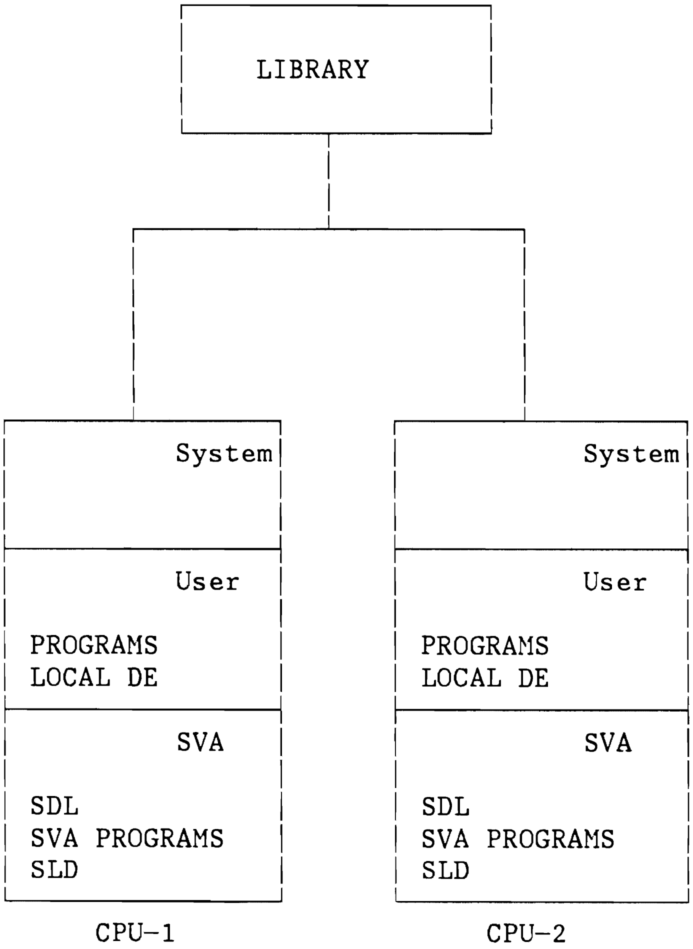


Figure 120. DASD Sharing Environment

The Second Level Directory (SLD), General Remarks

The SLD was introduced in order to have a quick access to a directory entry.

It has one entry for each directory block of a sublibrary. The entry contains the highest phase name, for which a directory entry is contained inside the directory block and the relative block address. So, by searching through the SLD, Fetch can find at once, (that means with only one I/O operation) the directory block which contains a special directory entry.

Highest phase name inside directory block 1	PRBA of directory block 1
• •	• •
highest phase name inside directory block n	PRBA of directory block n

one entry for each directory block

Figure 121. SLD Layout

The SLD of the system sublibrary IJSYSRS.SYSLIB is initialized at IPL time; the SLD of the private sublibraries at LIBDEF time.

Initiation of a SLD Update

Some operations, for example the deleting, cataloguing or renaming of a phase, change the directory and might leave the SLD in a back level state.

A SLD update should be done to avoid performance degradation. The SLD update is done by the librarian after a delete, catalogue etc. command was given.

If shared disks are used and the delete command f. ex. was given by another CPU, then it is Fetch which makes the SLD update.

Fetch identifies a back level SLD by the following criterions:

- More than one library block had been read in to find the directory entry although the SLD was used.

  OR

- The SLD entry does not point at all to a directory block, but f. ex. to a TXT or RLD block (space reclamation took place).

If Fetch identifies a back level SLD, then a SLD update is initiated unless one of the following is true:

- The sublibrary, whose SLD is in a back level state, is part of a temporary (not permanent) search chain
- The SLD update for the sublibrary is already initiated, but not finished yet.

SLD Update / Algorithm

The SLD update processing for a sublibrary consists in the following activities:

- Enqueuing the SLD update request into the SLD queue
- Activating the Service Task
- Doing the SLD update by the UPDSLD routine, which is called by the Service Task
- Dequeuing the SLD update request from the SLD queue
- Deactivating the Service Task, if there are no more SLD queue entries to be processed

Each one of these points will be discussed in more detail.

Notice, that the SLD update itself is done by the Service Task and not by Fetch. That means, that the SLD update runs in parallel with Fetch and does not lead to a lower Fetch performance.

The SLD Queue

The enqueuing of a SLD update request into the SLD queue is done by the ENQSLD routine, the dequeuing by the UPDSLD routine.

The elements of the SLD queue are chained together. The first element is pointed to by SLDACT.

The layout is:

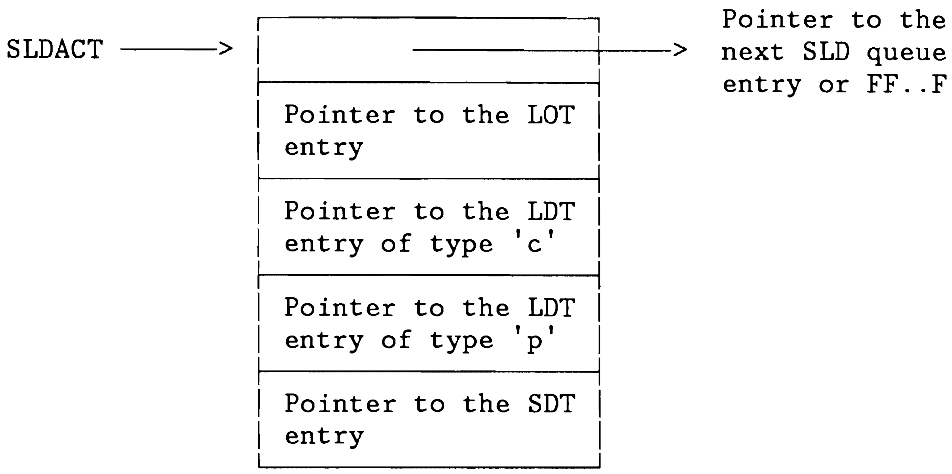

SLDACT
Pointer to the next SLD queue entry or FF..F
Pointer to the LOT entry
Pointer to the LDT entry of type 'c'
Pointer to the LDT entry of type 'p'
Pointer to the SDT entry

Figure 122. Layout of the SLD Queue

A type 'c' LDT entry is a complete LDT entry while a type 'p' LDT entry refers to a library which is already defined by another LDT. It only contains the library name and a pointer to the corresponding 'c' entry. A type 'p' entry exists when a library is accessed at the same time under another name by the same or another partition.

The layout of an SLD queue entry, disregarding the pointer field, is identical to the layout for the Library Information Area (see Librarian Diagnosis Reference Manual). Enqueuing a SLD request into the SLD queue means:

- Dequeuing an element from the SLD free chain, pointed to by SLDFREE. The SLD update request is canceled, if there are no free entries in the SLD free chain (SLDFREE=0).

- Putting the right values in the element

- Enqueuing the element into the SLD queue pointed to by SLDACT.

The Activation / Deactivation of the Service Task

The activation of the Service Task to do the SLD update is controlled by the flag RQTUPSLD.

If set, the UPDSLD routine is called by the Service Task. The flag is reset if there are no more entries in the SLD queue.

The UPDSLD Routine

The UPDSLD routine calls the Librarian services INLMSCON, INLMRESN, INLMSLD and INLMDIS to update the SLD.

See VSE/Advanced Functions Diagnosis Reference: Librarian, LY33-9111 for a description of these services.

Algorithm of the UPDSLD Routine

The following actions must be done for each entry of the SLD queue:

- Call INLMSCON to connect the sublibrary
- Call INLMRESN to get the resource name of the sublibrary
- Lock the sublibrary
- Call INLMSLD to update the SLD
- Unlock the sublibrary
- Call INLMDIS to disconnect the sublibrary
- Dequeue SLD entry

MACHINE CHECK AND CHANNEL CHECK HANDLING

Machine Check Analysis and Recording

MCAR responds to MCIs, attempts recovery, and provides operator messages on SYSLOG. Machine check records are written to the recorder file IJSYSRC by the RMSR transients.

When a machine check occurs, hardware first logs the error in the machine check logout area in low real storage and in the extended logout area pointed to by control register 15, and then retries the failure by CPU retry and ECC (Error Checking and Correction). If the retry is successful, a soft machine check (if enabled) occurs on Models 135 through 158 and 3031. All ECPS:VSE-machines do not generate an interrupt for a machine check from which it could recover. For soft MCIs, the recording is controlled through the error frequency limit (EFL). If a specified error count is reached, the recording mode is changed from recording to quiet. The MODE command gives the operator control of soft MCIs. It permits the operator three options:

- Determine whether the system is in quiet or recording mode.
- Alter the mode of operation.
- Change error threshold values.

If hardware retry is not successful, a hard machine check interrupt is generated. A hard MCI occurs when:

- CPU retry is not successful.
- Interrupted instruction cannot be retried.
- Storage failure is permanent.

In the event of a hard MCI, the affected task is canceled. MCAR assesses the damage and continues system operation when possible. The system enters the hard wait state when a hard MCI:

- Interrupts supervisor coding.
- Occurs while assessing critical information or phases from SYSRES.
- Damages privileged coding through a permanent storage error.

MCAR attempts to notify the operator about:

- Machine check type.
- Wait state, re-IPL.
- Problem program termination.
- Mode operation change.
- Buffer deletion.

The resident machine check handler analyzes the machine check interruption code (MCIC) and the problem state bit (bit 13) of the machine check old PSW. It categorizes errors into three classes:

1. System operation termination condition.
   The MCIC indicates:

   - System damage.
   - CPU-mask (IMWP) in old PSW is invalid.
   - Instruction processing damage (while the CPU is in the supervisor state).
   - One or more old PSW bits, other than in the CPU-mask, are invalid or the general registers are invalid (while the CPU is in the supervisor state).
   - Storage or Protection error while the failing storage address is invalid.
   - Warning bit on.
   - No subclass bits on in first byte of MCIC.
   - General registers invalid.
   - External damage (if not secondary report).

   Action: Post C'A' in location 0 (system termination code) and the emergency exit bit (X'08') is posted in the RAS Linkage Area (see Figure 307 on page 604)

2. Hard machine checks. The system can continue but the damaged task is to be canceled. While the CPU is in the problem state, the MCIC indicates:

   - Instruction processing damage. All the general registers are invalid.
   - A storage or protection error with valid failing storage address occurred.
   - One or more old PSW bits, other than in the CPU-mask, are invalid.

   Action: Activate RAS system task and branch to the cancel routine to cancel the task.

3. Soft machine check (if none of the above conditions is present). Only recording is required for errors from which hardware recovered successfully.

   Action: If interruption occurred while in problem state, activate RAS system task and exit to task selection. If supervisor function is being performed or system task active, activate RAS system task (if not already active) and return to the interrupted code by loading the machine check old PSW.

Nonresident machine check handling is described in VSE/Advanced Functions Diagnosis Reference: Error Recovery and Recording Transients, LY33-9108.

Channel Check Handler(CCH)

The resident CCH gains control from the I/O interrupt handler when either the interface control check or channel control check bit is posted in the CSW. The channel supplies additional channel check information in the 4-byte limited logout area (ECSW) and, under control of CR14 bit 2, in the I/O extended logout area. The ECSW is inspected to determine if enough information is valid to isolate the damage to either a channel or a device or if a system termination condition exists. For each channel check an error entry in the PUB extension is used to save error and recording information. If channel and device information is valid the error entry of the corresponding PUB is used. If a channel damage condition exist, the error entry of the first busy disk-device not queued in error on the indicated channel is used.

For channel checks on disk devices the recovery actions are initiated by the resident CCH. After recovery is done, the error entry is completed and chained to the RAS error chain. The RAS task is posted and control is given to the dispatcher. For an unsuccessful recovery the task in error is canceled.

For channel checks on non-disk devices the error entry is completed, enqueued to the RAS error chain, the RAS task is posted and control is given to the dispatcher. Device dependent recovery actions and recovery dependent cancel actions are performed by the RAS monitor and the R-transients.

When a system termination condition is detected, the emergency exit bit is posted in the RAS linkage area (see Figure 307 on page 604) and the RAS task is entered. The applicable termination code is posted at storage location 0. The following list gives the termination codes for the various types of disastrous channel errors:

B	Irrecoverable channel check on fetch.
C	Irrecoverable channel check on paging channel.
E	ECSW not stored.
G	Channel address invalid.
H	Channel check on log with RASMSG.

Nonresident channel check handling is described in VSE/Advanced Functions Diagnosis Reference: Error Recovery and Recording Transients.

	Record error	Message on SYSLOG	Terminate System	CLRCH	HIO CLRIO	Recovery action
No ECSW stored		X	X			
Channel address invalid		X	X			
Unit address invalid	X	X			X	X
Interface inoperative	X	X		X		X
System reset code on	X	X			X	X
CUA valid	X	X				X

RECOVERY ACTION VERIFICATION						
	Retry channel program	Post error in CCB	Cancel channel user			
User own error recovery		X				
Channel program retryable	succ.					
	unsucc.	X	X			
User accepts I/O error	succ.					
	unsucc.	X				

Figure 123. Channel Check Handling Overview

Recovery Transients and RAS Monitor

The recovery transients (R-transients) perform machine check and channel check recovery and recording.

The RAS monitor is a supervisor resident control program which

- Dequeues error blocks from the RAS error chain.
- Moves error information to the work ERPIB.
- Fetches R-transients into the RTA.
- Schedules I/O requests from the RTA.
- Performs services for the R-transients.
- Provides an exit interface from R-transients.

The RAS monitor table (RASTAB, Figure 308 on page 605), the RAS linkage area (RASLINK, Figure 307 on page 604) and the Error Recovery Procedure Information Block (ERPIB, Figure 309 on page 608) contain the necessary information for the RAS monitor and the R-transients.

JOB ACCOUNTING

The support for job accounting is always generated in the VSE/AF Supervisor and is optionally activated at IPL time by SYS JA=YES.

Job accounting is associated with the following data areas:

- Some fields in the system communication region SYSCOM
- The job accounting common table ACCTCOMN (see Figure 243 on page 523).
- Some fields in the partition communication region COMREG and in the Partition Control Block (PCB).
- For each partition the job accounting partition table ACCTABLE (see Figure 244 on page 523).
- A set of device usage and SIO counters associated with the PUB-extension PUBX and the PCB.
- A 1K user save area.

Job accounting logic consists of three distinct parts:

- The initialization of accounting areas and fields at IPL time.
- The maintenance of accounting information at system run time.
- The interface to the user written accounting routine $JOBACCT.

Initialization

Most of the initialization work is done by the phase $INITSYS, which is executed during IPL after all system option have been specified. When $INITSYS is invoked, the following initialization relevant to job accounting is already done:

- SYSCOM.IJBFLG02.IJBSEC is set on if JA=YES was specified.
- A PUBX is allocated and initialized for every added device.
- The total number of added device is stored in SYSCOM.IJBNDEV.
- The total number of added 'partition sharable' devices is stored in SYSCOM.IJBNSDEV.

DASD devices, unit record devices and the SYSLOG device are considered as partition sharable. Unit record devices are included because they can be used as dummy devices for VSE/POWER in more than one partition. If SYSCOM.IJBFLG02.IJBSEC is on, $INITSYS

- Allocates a 1K user save area and saves its address in ACCTCOMN.ACCTUSEP.
- Calculates the length of ACCTABLE depending on SYSCOM.IJBNDEV and saves it in ACCTCOMN.ACCTABLN.
- Allocates and initializes one ACCTABLE per partition in pageable system GETVIS space.
- Saves the address of each ACCTABLE in COMREG.JAPART.

- Sets COMREG.JCSW1.JASWITCH off as an external indication (mainly for job control and VSE/POWER) that job accounting is active.
- Sets PCB.PCBJAPTR = A(PCB) as an internal indicator that job accounting is active.
- Allocates in fixed system GETVIS space strings of usage and SIO counters for partition sharable devices, one string per partition, saves the address of each string in PCB.PCBCNT and the offset within the string of the SIO counter for each device in PUBX.PBXJAOFF.

Maintenance

At system run time, CPU time and SIO counters are maintained in internal fields, which are not directly accessible to the user.

For CPU time accounting, short time intervals (typically between dispatching and interrupt times) are measured with the CPU Timer in units of 16 microseconds and assigned to a partition, whenever possible, or to the system as overhead times. Time intervals with the CPU in wait state are accumulated in a separate allbound time counter.
The criterion for assigning a time interval to a partition is that the time interval represents a reproducible portion of productive work for that partition. System activities, which do not fall under this categories, are the following:

- Paging
- Channel scheduling
- Hardware error recovery
- First level timer interrupt processing
- Attention routine processing (operator commands)

CPU time intervals assigned to a partition are accumulated in the field PCB.RUNTIME. The corresponding field in the system PCB is used to accumulate overhead time intervals. System wait state intervals are accumulated in the low core field SBNDTIME.
Field PCB.PCBJAPTR points to the PCB to which the current time interval is to be assigned. For partition PCB's, PCBJAPTR may point to the PCB itself (partition time) or the system PCB (overhead time). For the system PCB, PCBJAPTR may point to the PCB of the service owner (partition time) or to the system PCB itself (overhead time). System tasks, whose processing is always counted as overhead time, are flagged by TIB.TIBFLAG2.OVHIND.

Whenever a CPU Timer interrupt occurs or a GETJA request is issued (see below), the contents of the fields PCB.RUNTIME and SBNDTIME are transferred to another set of internal fields in each partition PCB, namely PCPUTIME, POVHTIME and PBNDTIME. PCB.RUNTIME in the partition PCB is simply added to PCPUTIME. The accumulated overhead PCB.RUNTIME in the system PCB is distributed among the fields PCB.POVHTIME of all active partitions in proportion to their PCB.RUNTIME values. SBNDTIME is distributed in equal parts among the fields PCB.PBNDTIME of all active partitions.

SIO counters are maintained for all devices in internal fields associated with each PUBX. For devices, which are not partition sharable, a single counter PUBX.PBXJACNT is sufficient. For partition sharable devices, there is one internal SIO counter per device and partition located at (PCBCNT)+(PUBX.PBXJAOFF).

The SIO counter is updated immediately after a successful SIO and, for spooled dummy devices, after successful invocation of the VSE/POWER SVC 0 appendage. SIOs for system tasks with TIBFLAG2.OVHIND on as well as those associated with the logical unit SYSUSE are not counted.

User Interface

Whenever a job step is completed, job control invokes the user accounting routine $JOBACCT. Accounting data is passed to the user in the accounting partition table ACCTABLE. The transfer of the internal counters into the ACCTABLE is controlled by the macro GETJA, which is invoked by job control at well defined points within job processing, in order to restrict the data in ACCTABLE to single job steps. For details on the function of GETJA refer to the internal macro descriptions in Appendix B.

The GETJA routine is also internally invoked by SVC 112 (X'70' - MSAT macro), to save the SIO counter of a device, which is not partition sharable, into ACCTABLE whenever device ownership is released.

CHAPTER 3: ORGANIZATION INFORMATION

This chapter provides information about the general structure of the supervisor, contains descriptions of the main routines and functions, and shows interfaces and communications between the various routines.

The following parts are described:

SUPERVISOR GENERAL ENTRY

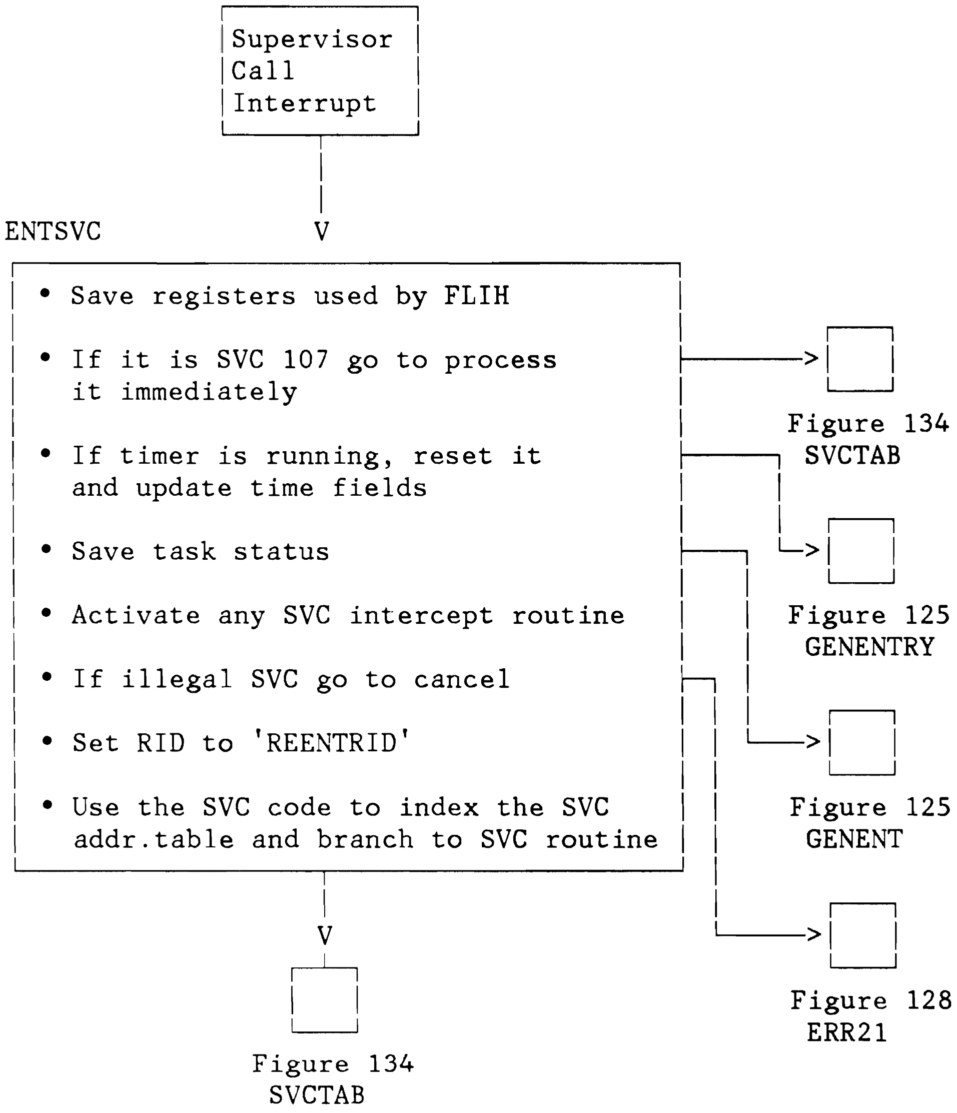

Supervisor
Call
Interrupt
ENTSVC
V
• Save registers used by FLIH
• If it is SVC 107 go to process it immediately
• If timer is running, reset it and update time fields
• Save task status
• Activate any SVC intercept routine
• If illegal SVC go to cancel
• Set RID to 'REENTRID'
• Use the SVC code to index the SVC addr.table and branch to SVC routine
Figure 134
SVCTAB
Figure 125
GENENTRY
Figure 125
GENENT
Figure 128
ERR21
V
Figure 134
SVCTAB

Figure 124 (Part 1 of 5). Supervisor First Level Interrupt Handler

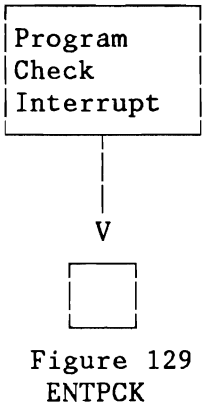


Figure 124 (Part 2 of 5). Supervisor First Level Interrupt Handler

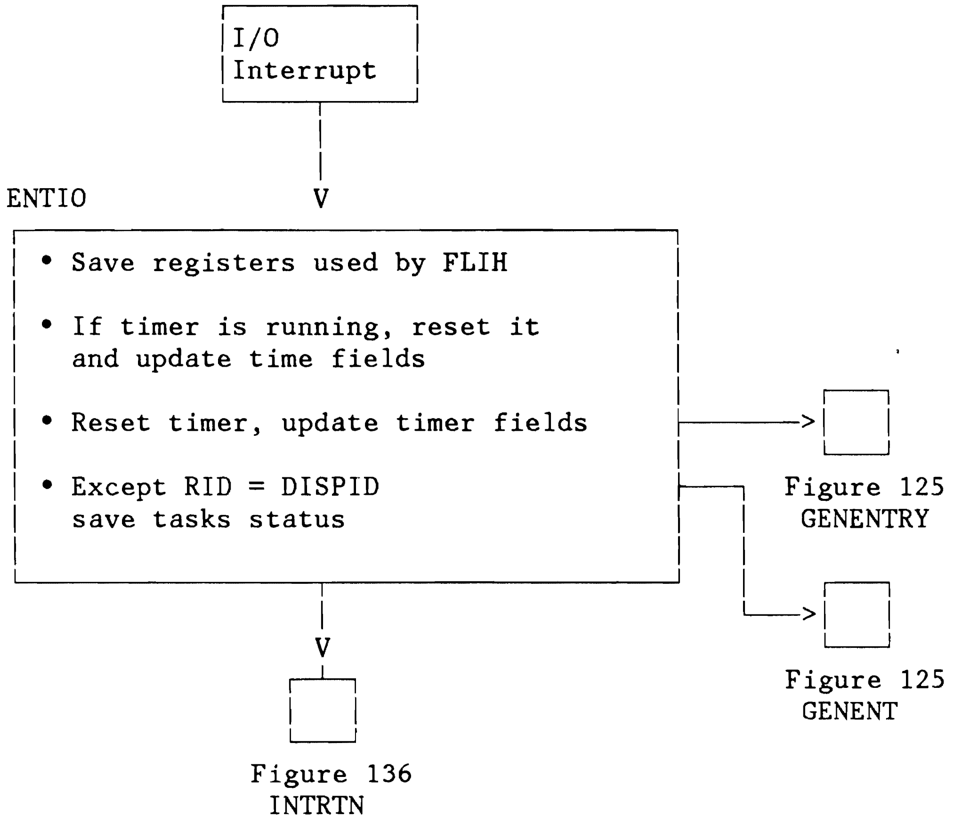

I/O
Interrupt
ENTIO
• Save registers used by FLIH
• If timer is running, reset it and update time fields
• Reset timer, update timer fields
• Except RID = DISPID save tasks status
Figure 125
GENENTRY
Figure 125
GENENT
Figure 136
INTRTN

Figure 124 (Part 3 of 5). Supervisor First Level Interrupt Handler

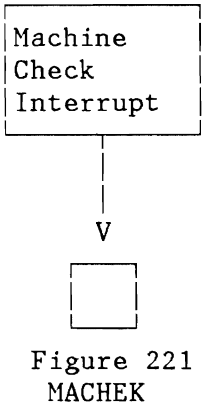


Figure 124 (Part 4 of 5). Supervisor First Level Interrupt Handler

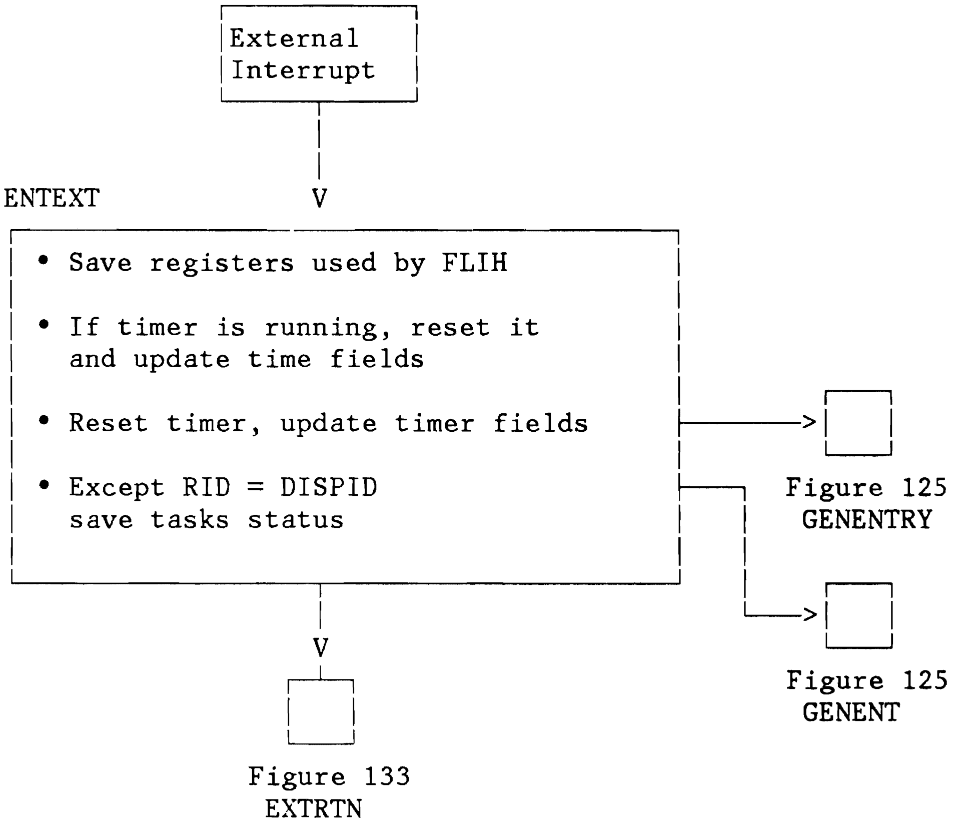

External Interrupt
ENTEXT
• Save registers used by FLIH
• If timer is running, reset it and update time fields
• Reset timer, update timer fields
• Except RID = DISPID save tasks status
Figure 125
GENENTRY
Figure 125
GENENT
Figure 133
EXTRTN

Figure 124 (Part 5 of 5). Supervisor First Level Interrupt Handler

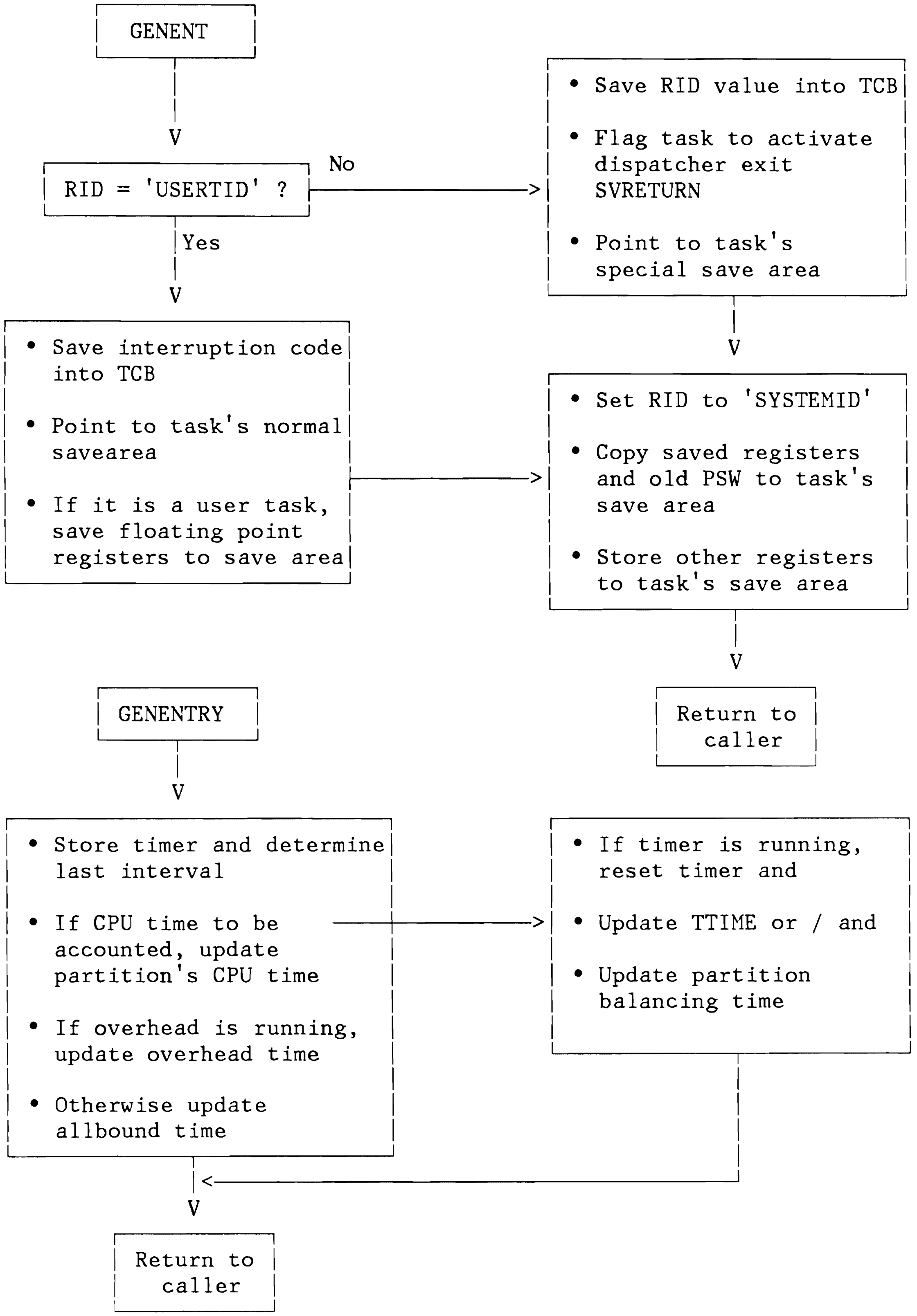

GENENT
RID = 'USERTID' ?
No
Yes
• Save RID value into TCB
• Flag task to activate dispatcher exit SVRETURN
• Point to task's special save area
• Save interruption code into TCB
• Point to task's normal savearea
• If it is a user task, save floating point registers to save area
• Set RID to 'SYSTEMID'
• Copy saved registers and old PSW to task's save area
• Store other registers to task's save area
Return to caller
GENENTRY
• Store timer and determine last interval
• If CPU time to be accounted, update partition's CPU time
• If overhead is running, update overhead time
• Otherwise update allbound time
• If timer is running, reset timer and
• Update TTIME or / and
• Update partition balancing time
Return to caller

Figure 125. Supervisor FLIH, Subroutines

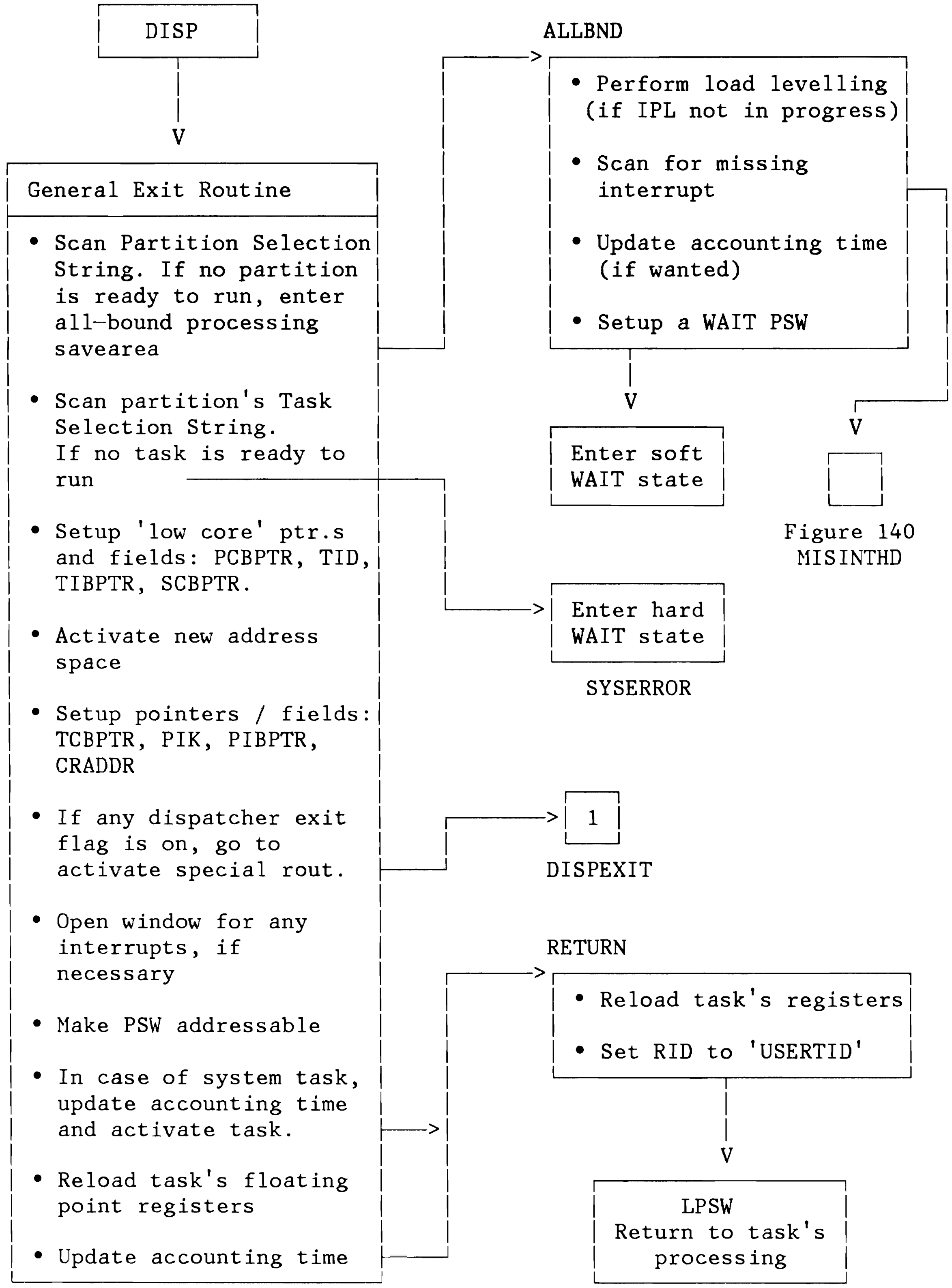
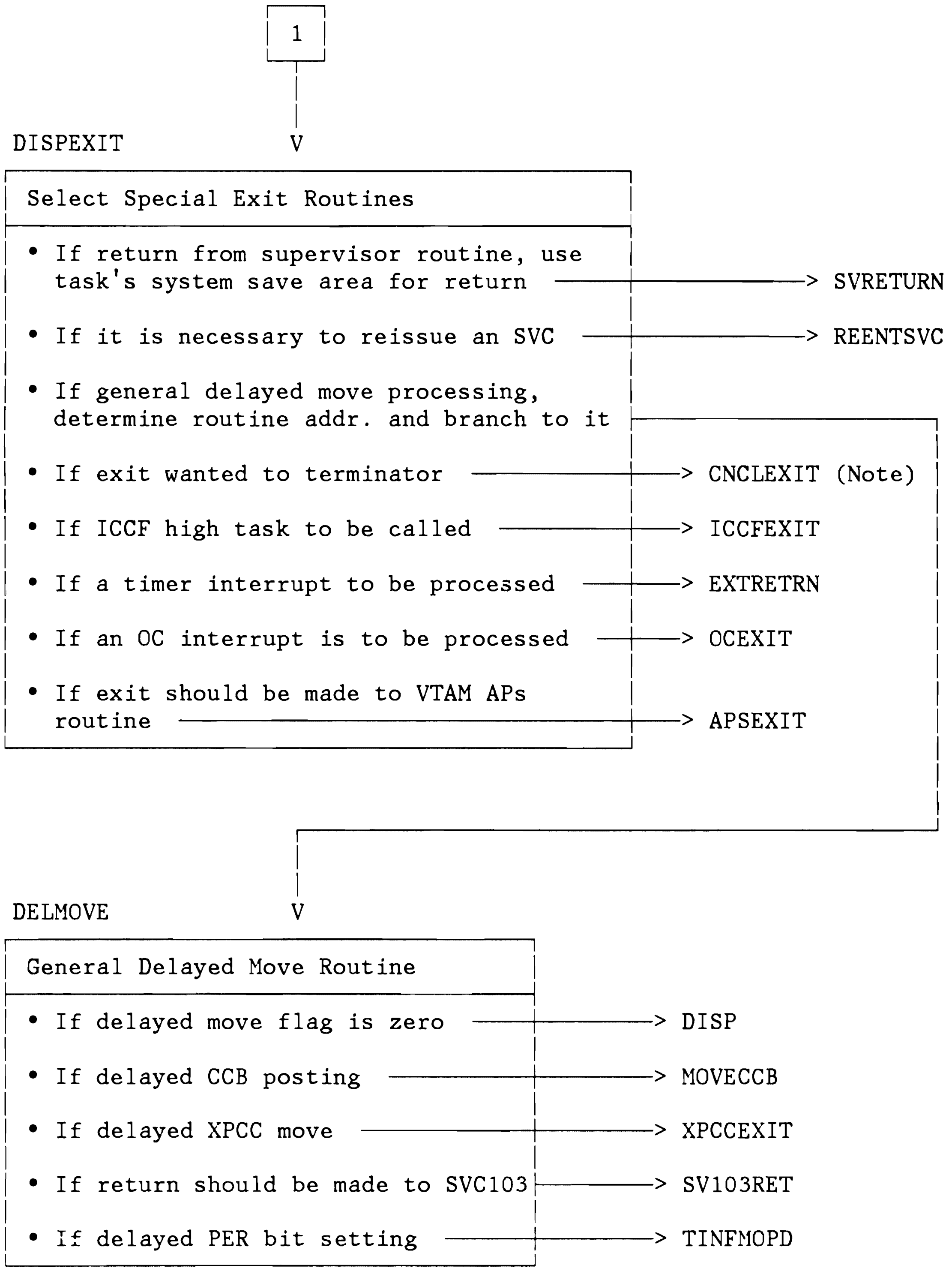

DISP
General Exit Routine
• Scan Partition Selection String. If no partition is ready to run, enter all–bound processing savearea
• Scan partition's Task Selection String. If no task is ready to run
• Setup 'low core' ptr.s and fields: PCBPTR, TID, TIBPTR, SCBPTR.
• Activate new address space
• Setup pointers / fields: TCBPTR, PIK, PIBPTR, CRADDR
• If any dispatcher exit flag is on, go to activate special rout.
• Open window for any interrupts, if necessary
• Make PSW addressable
• In case of system task, update accounting time and activate task.
• Reload task's floating point registers
• Update accounting time
ALLBND
• Perform load levelling (if IPL not in progress)
• Scan for missing interrupt
• Update accounting time (if wanted)
• Setup a WAIT PSW
Enter soft WAIT state
Figure 140
MISINTHD
Enter hard WAIT state
SYSERROR
1
DISPEXIT
RETURN
• Reload task's registers
• Set RID to 'USERTID'
LPSW
Return to task's processing

Figure 126 (Part 1 of 2). Supervisor General Exit, Task Selection


1
DISPEXIT
Select Special Exit Routines
• If return from supervisor routine, use task's system save area for return
SVRETURN
• If it is necessary to reissue an SVC
REENTSVC
• If general delayed move processing, determine routine addr. and branch to it
• If exit wanted to terminator
CNCLEXIT (Note)
• If ICCF high task to be called
ICCFEXIT
• If a timer interrupt to be processed
EXTRETRN
• If an OC interrupt is to be processed
OCEXIT
• If exit should be made to VTAM APs routine
APSEXIT
DELMOVE
General Delayed Move Routine
• If delayed move flag is zero
DISP
• If delayed CCB posting
MOVECCB
• If delayed XPCC move
XPCCEXIT
• If return should be made to SVC103
SV103RET
• If delayed PER bit setting
TINFMOPD

Note: CNCLEXIT - See Figure 127 on page 309.

Figure 126 (Part 2 of 2). Supervisor General Exit, Task Selection

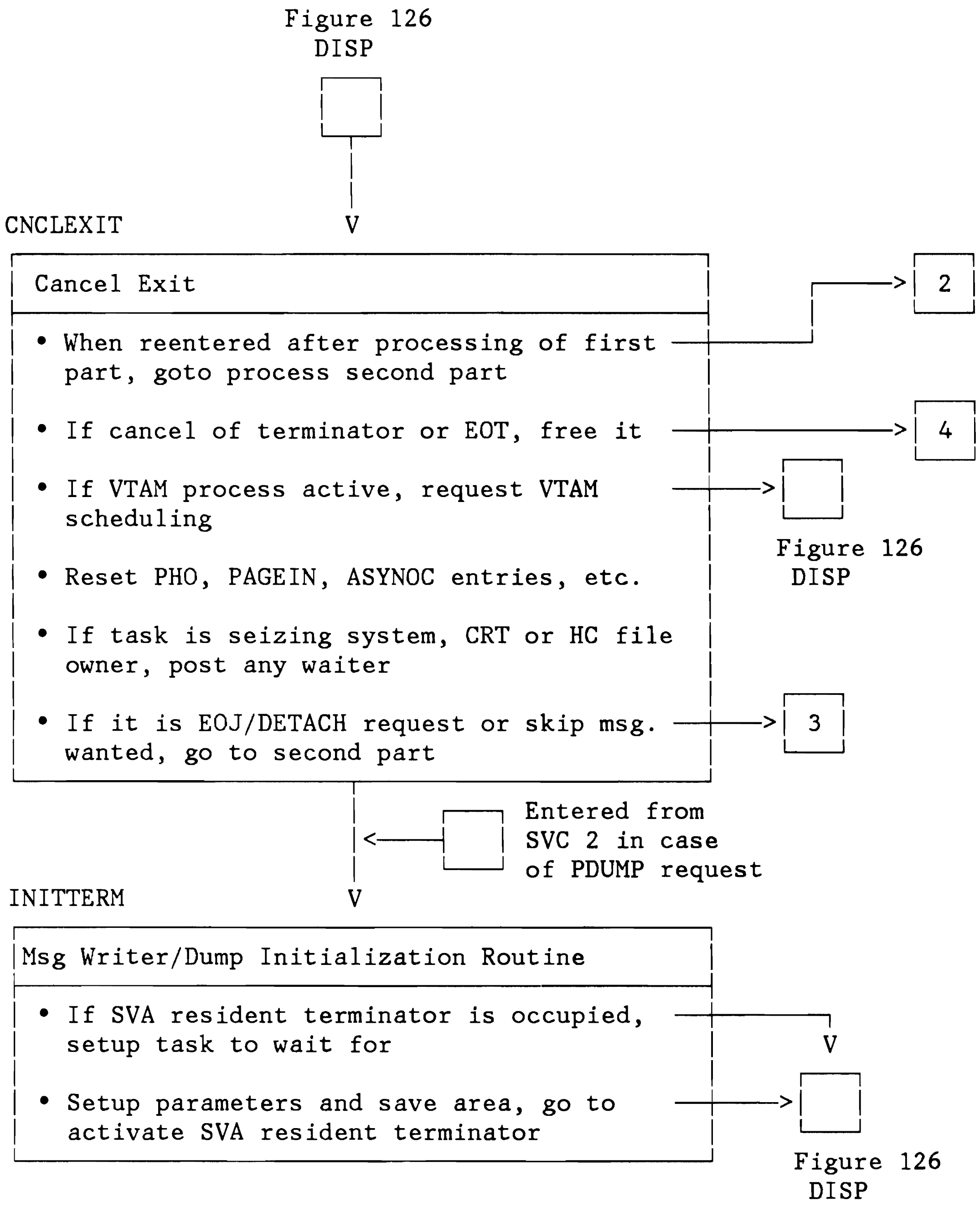
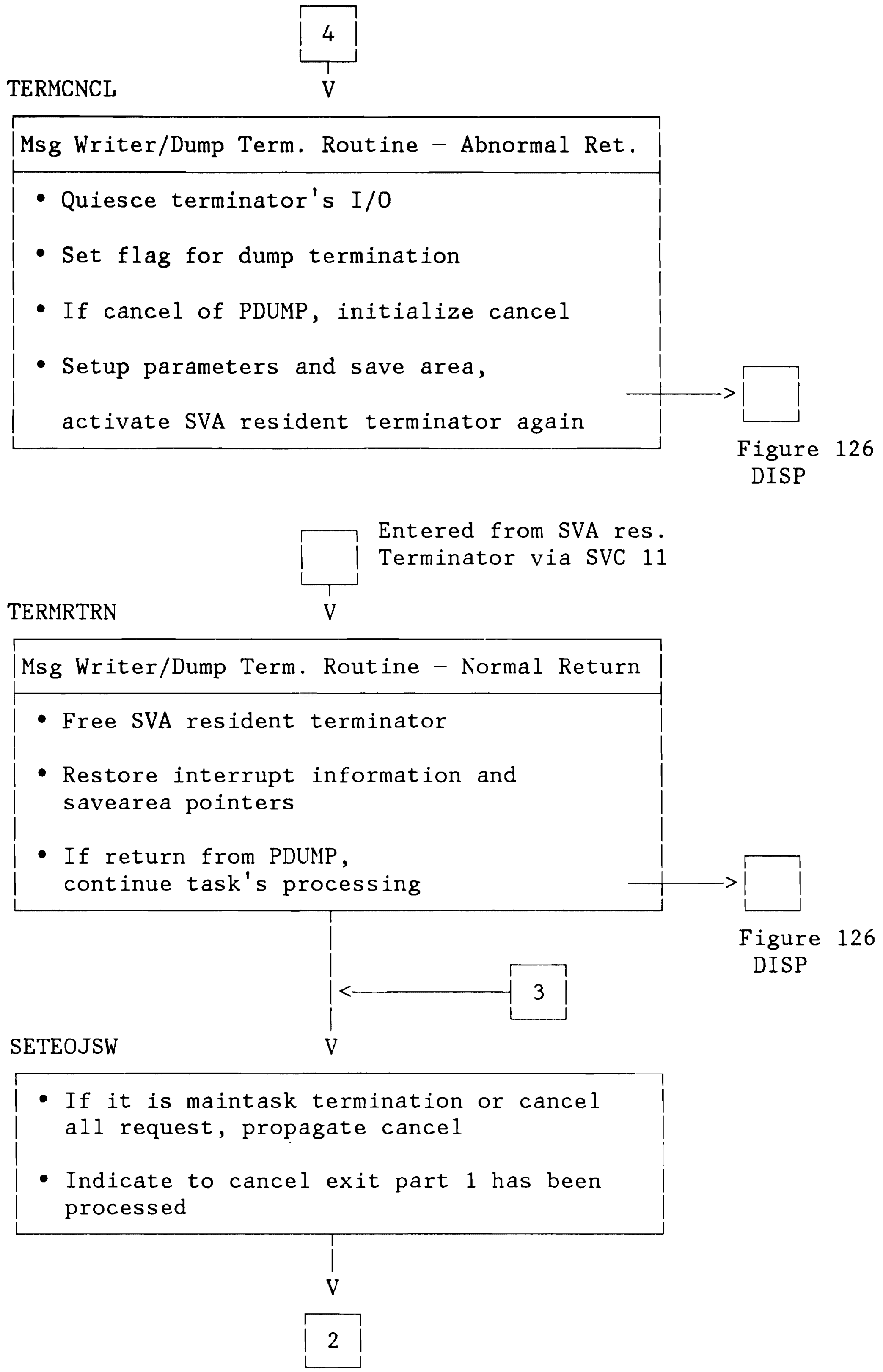


Figure 126
DISP
CNCLEXIT
Cancel Exit
• When reentered after processing of first part, goto process second part
2
• If cancel of terminator or EOT, free it
4
• If VTAM process active, request VTAM scheduling
Figure 126
DISP
• Reset PHO, PAGEIN, ASYNOC entries, etc.
• If task is seizing system, CRT or HC file owner, post any waiter
• If it is EOJ/DETACH request or skip msg. wanted, go to second part
3
Entered from SVC 2 in case of PDUMP request
INITTERM
Msg Writer/Dump Initialization Routine
• If SVA resident terminator is occupied, setup task to wait for
• Setup parameters and save area, go to activate SVA resident terminator
Figure 126
DISP

Figure 127 (Part 1 of 4). Supervisor General Exit, Cancel Exit



Msg Writer/Dump Term. Routine - Abnormal Ret.

- Quiesce terminator's I/O
- Set flag for dump termination
- If cancel of PDUMP, initialize cancel
- Setup parameters and save area, activate SVA resident terminator again

Msg Writer/Dump Term. Routine - Normal Return

- Free SVA resident terminator
- Restore interrupt information and savearea pointers
- If return from PDUMP, continue task's processing

- If it is maintask termination or cancel all request, propagate cancel
- Indicate to cancel exit part 1 has been processed

Figure 127 (Part 2 of 4). Supervisor General Exit, Cancel Exit

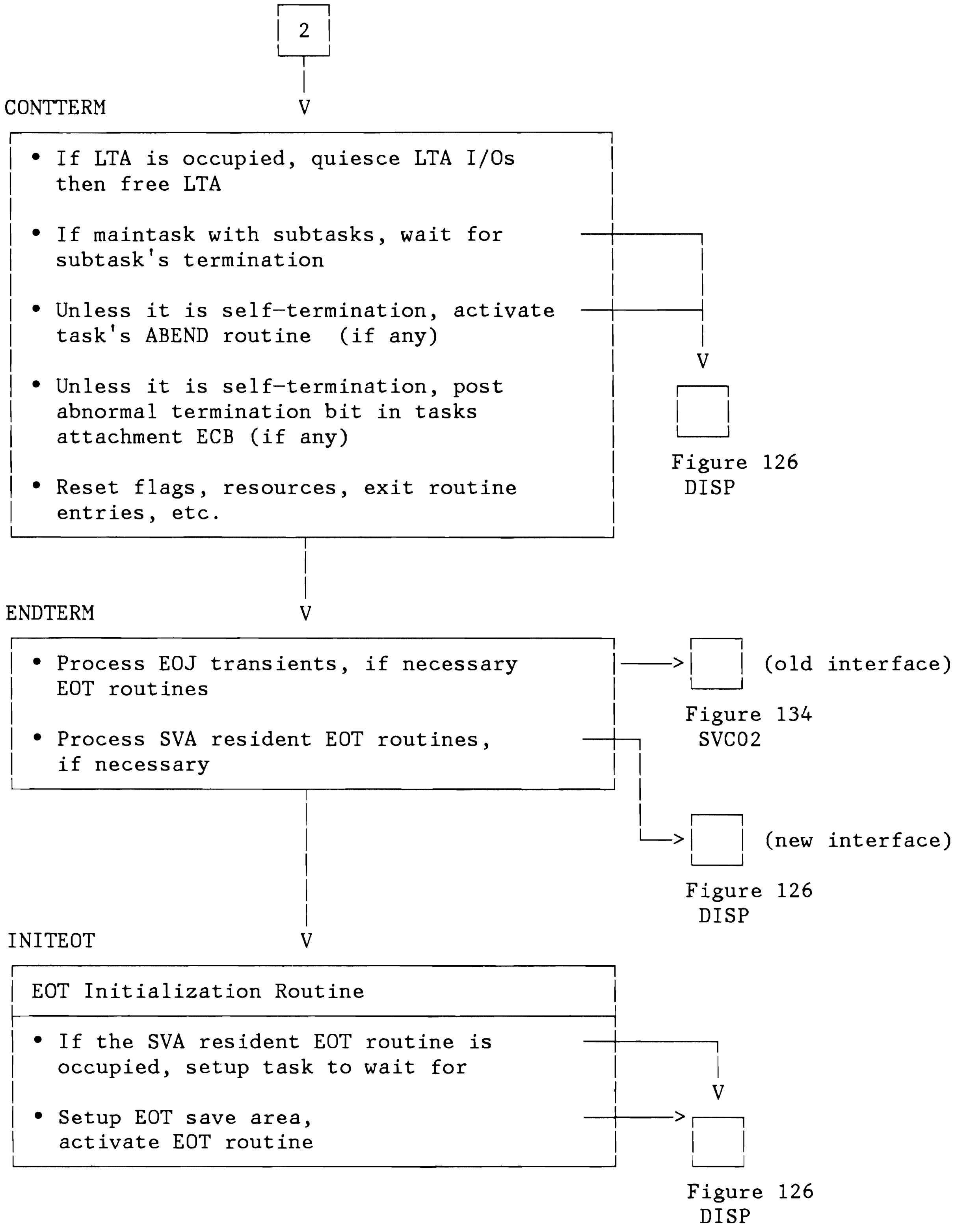
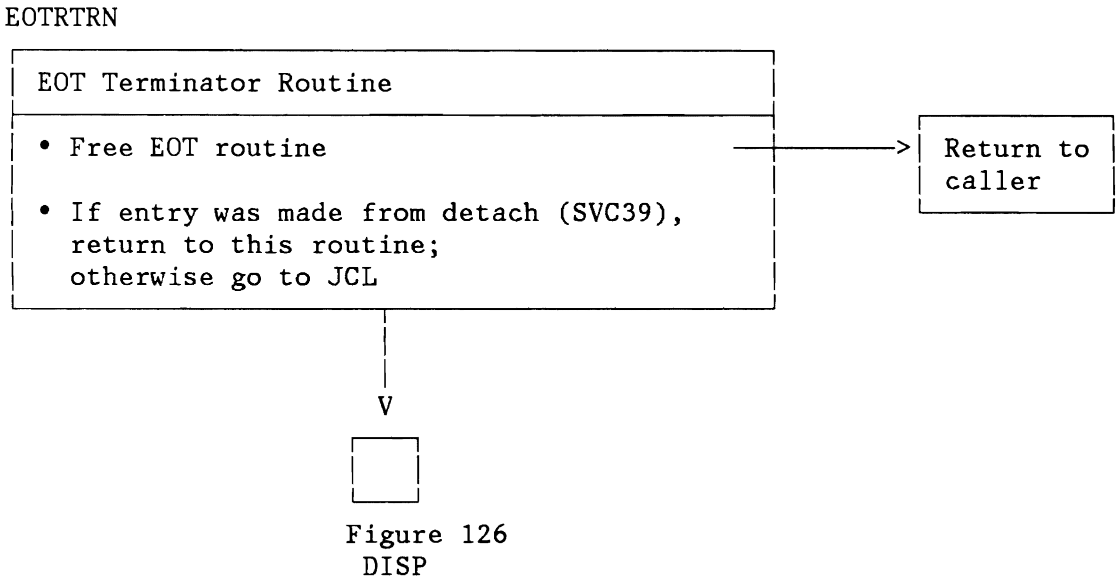
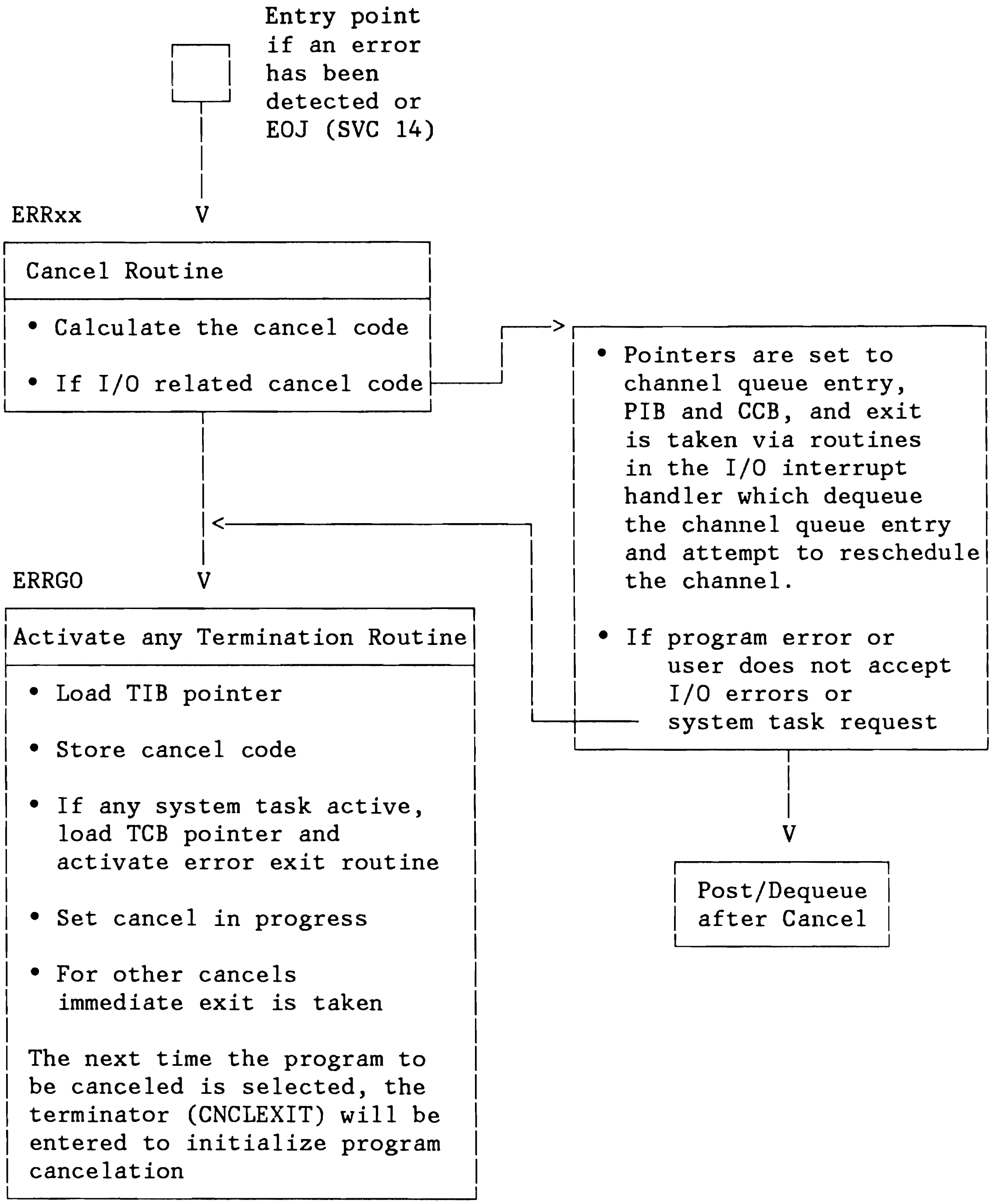

2
CONTTERM
• If LTA is occupied, quiesce LTA I/Os then free LTA
• If maintask with subtasks, wait for subtask's termination
• Unless it is self-termination, activate task's ABEND routine (if any)
• Unless it is self-termination, post abnormal termination bit in tasks attachment ECB (if any)
• Reset flags, resources, exit routine entries, etc.
Figure 126
DISP
ENDTERM
• Process EOJ transients, if necessary EOT routines
• Process SVA resident EOT routines, if necessary
(old interface)
Figure 134
SVC02
(new interface)
Figure 126
DISP
INITEOT
EOT Initialization Routine
• If the SVA resident EOT routine is occupied, setup task to wait for
• Setup EOT save area, activate EOT routine
Figure 126
DISP

Figure 127 (Part 3 of 4). Supervisor General Exit, Cancel Exit



- Free EOT routine
- If entry was made from detach (SVC39), return to this routine; otherwise go to JCL

Figure 127 (Part 4 of 4). Supervisor General Exit, Cancel Exit



Entry point if an error has been detected or EOJ (SVC 14)

ERRxx

Cancel Routine

- Calculate the cancel code
- If I/O related cancel code

- Pointers are set to channel queue entry, PIB and CCB, and exit is taken via routines in the I/O interrupt handler which dequeue the channel queue entry and attempt to reschedule the channel.
- If program error or user does not accept I/O errors or system task request

Post/Dequeue after Cancel

ERRGO

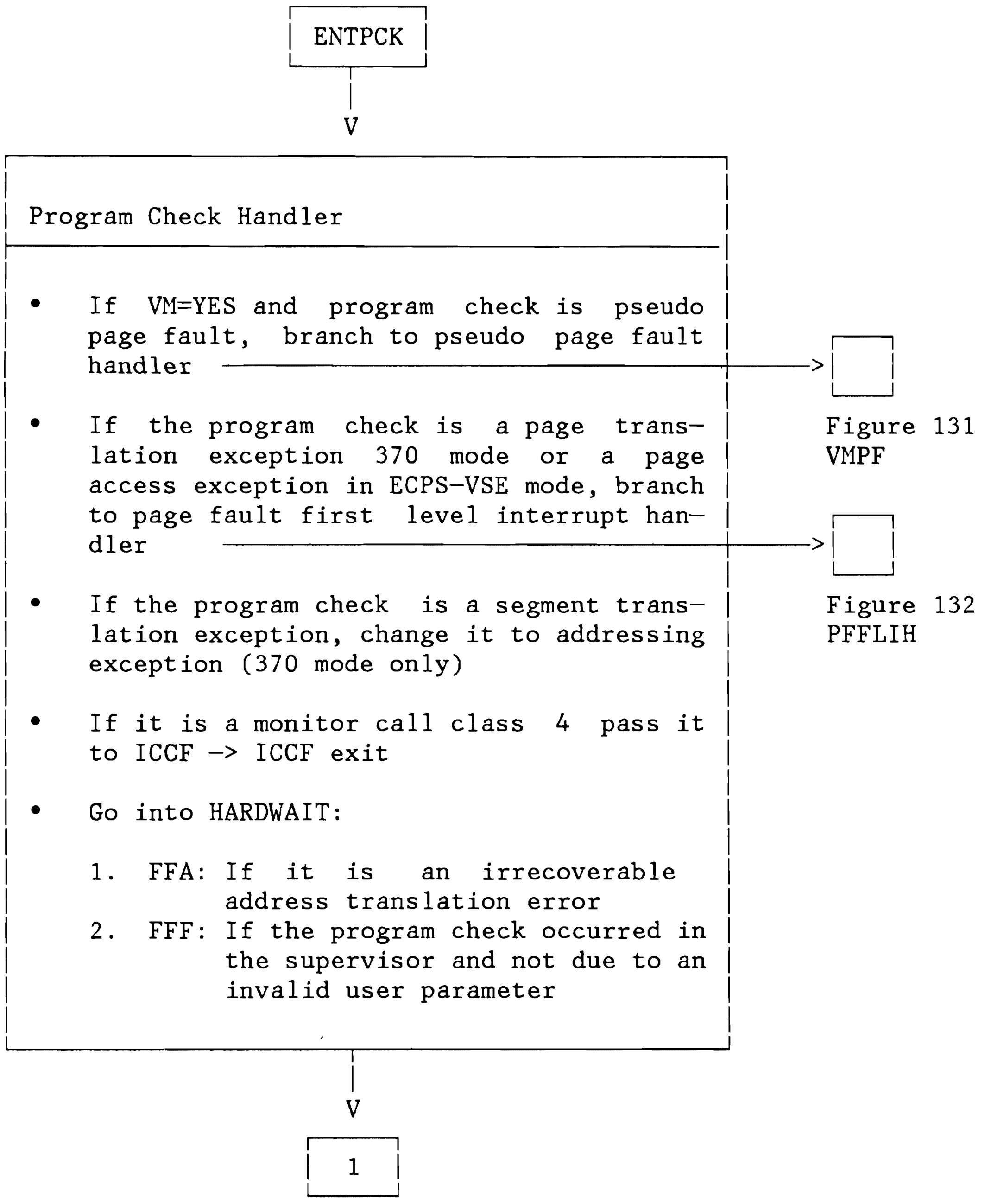
Activate any Termination Routine

- Load TIB pointer
- Store cancel code
- If any system task active, load TCB pointer and activate error exit routine
- Set cancel in progress
- For other cancels immediate exit is taken

The next time the program to be canceled is selected, the terminator (CNCLEXIT) will be entered to initialize program cancelation

Figure 128. Cancel Routine

PROGRAM CHECK HANDLER

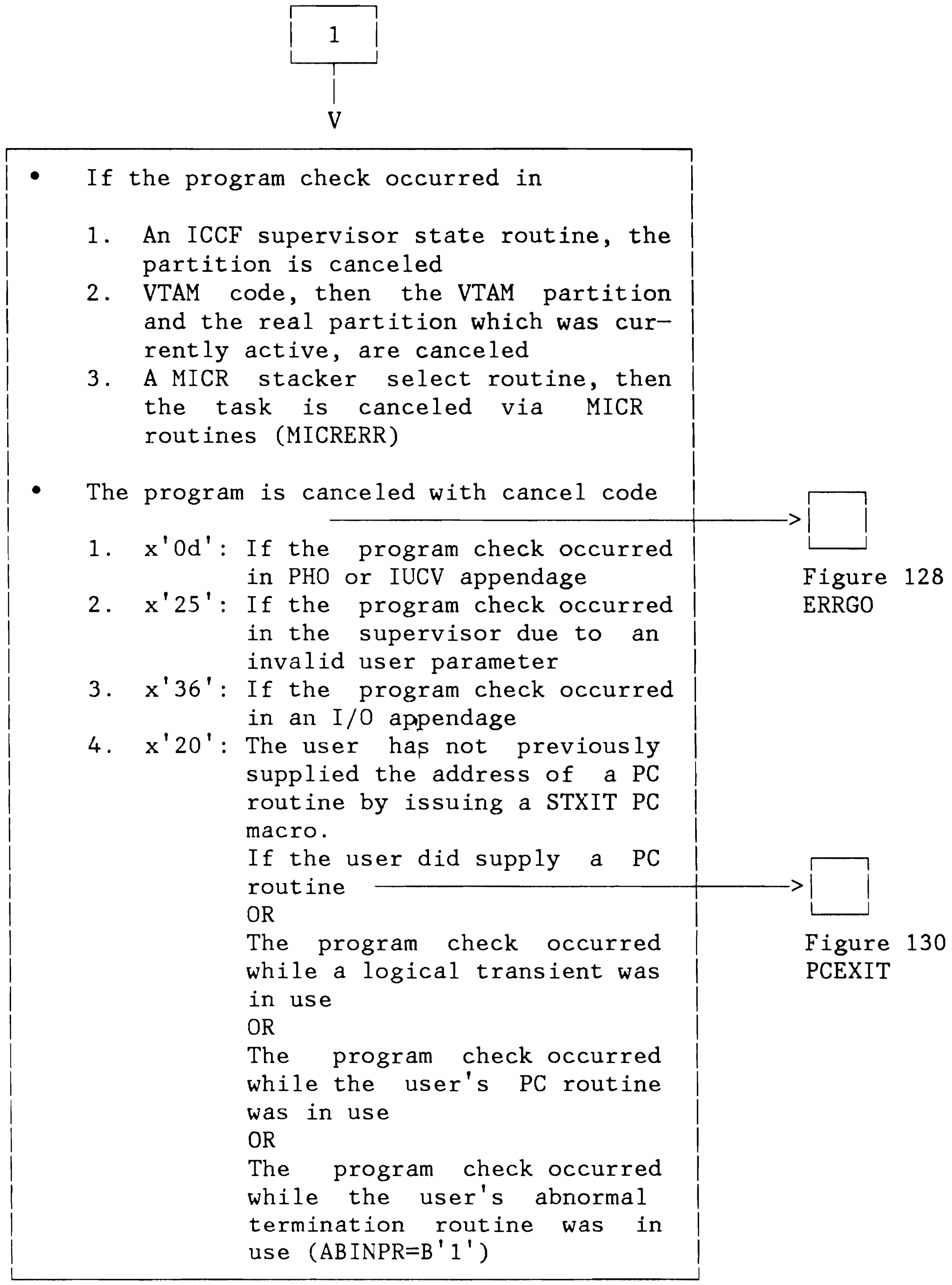


Program Check Handler

- If VM=YES and program check is pseudo page fault, branch to pseudo page fault handler
- If the program check is a page translation exception 370 mode or a page access exception in ECPS-VSE mode, branch to page fault first level interrupt handler
- If the program check is a segment translation exception, change it to addressing exception (370 mode only)
- If it is a monitor call class 4 pass it to ICCF -> ICCF exit
- Go into HARDWAIT:

  1. FFA: If it is an irrecoverable address translation error
  2. FFF: If the program check occurred in the supervisor and not due to an invalid user parameter

Figure 129 (Part 1 of 2). Program Check Handler



1

- If the program check occurred in

  1. An ICCF supervisor state routine, the partition is canceled
  2. VTAM code, then the VTAM partition and the real partition which was currently active, are canceled
  3. A MICR stacker select routine, then the task is canceled via MICR routines (MICRERR)

- The program is canceled with cancel code

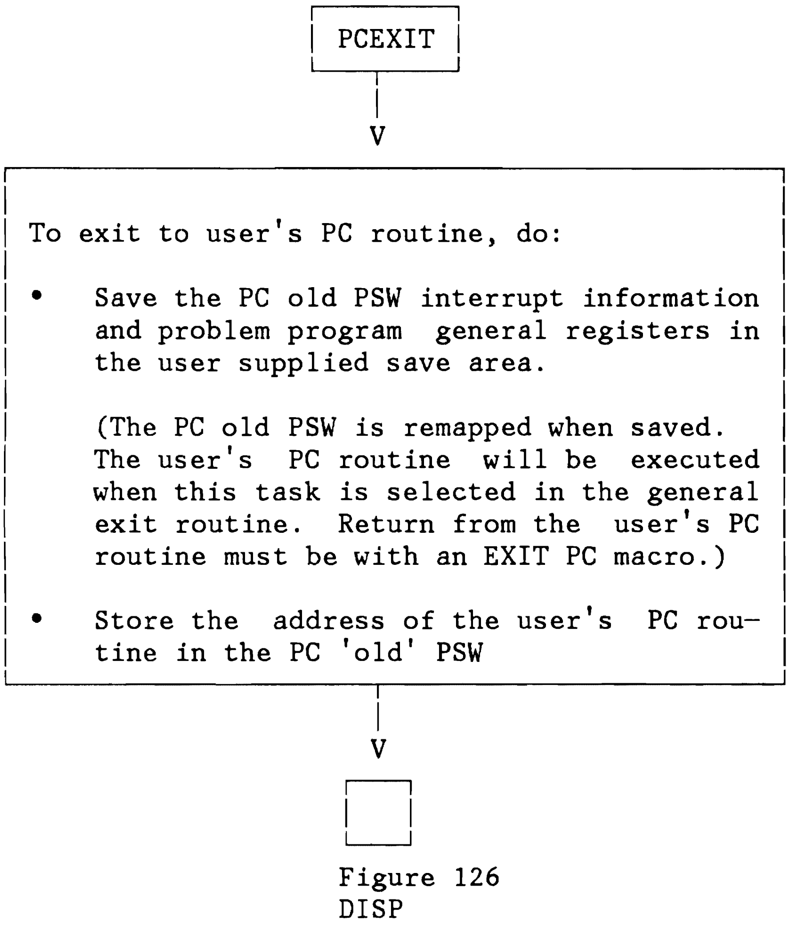
  1. x'0d': If the program check occurred in PHO or IUCV appendage
  2. x'25': If the program check occurred in the supervisor due to an invalid user parameter
  3. x'36': If the program check occurred in an I/O appendage
  4. x'20': The user has not previously supplied the address of a PC routine by issuing a STXIT PC macro.

     If the user did supply a PC routine
     OR
     The program check occurred while a logical transient was in use
     OR
     The program check occurred while the user's PC routine was in use
     OR
     The program check occurred while the user's abnormal termination routine was in use (ABINPR=B'1')

Figure 128
ERRGO

Figure 130
PCEXIT

Figure 129 (Part 2 of 2). Program Check Handler



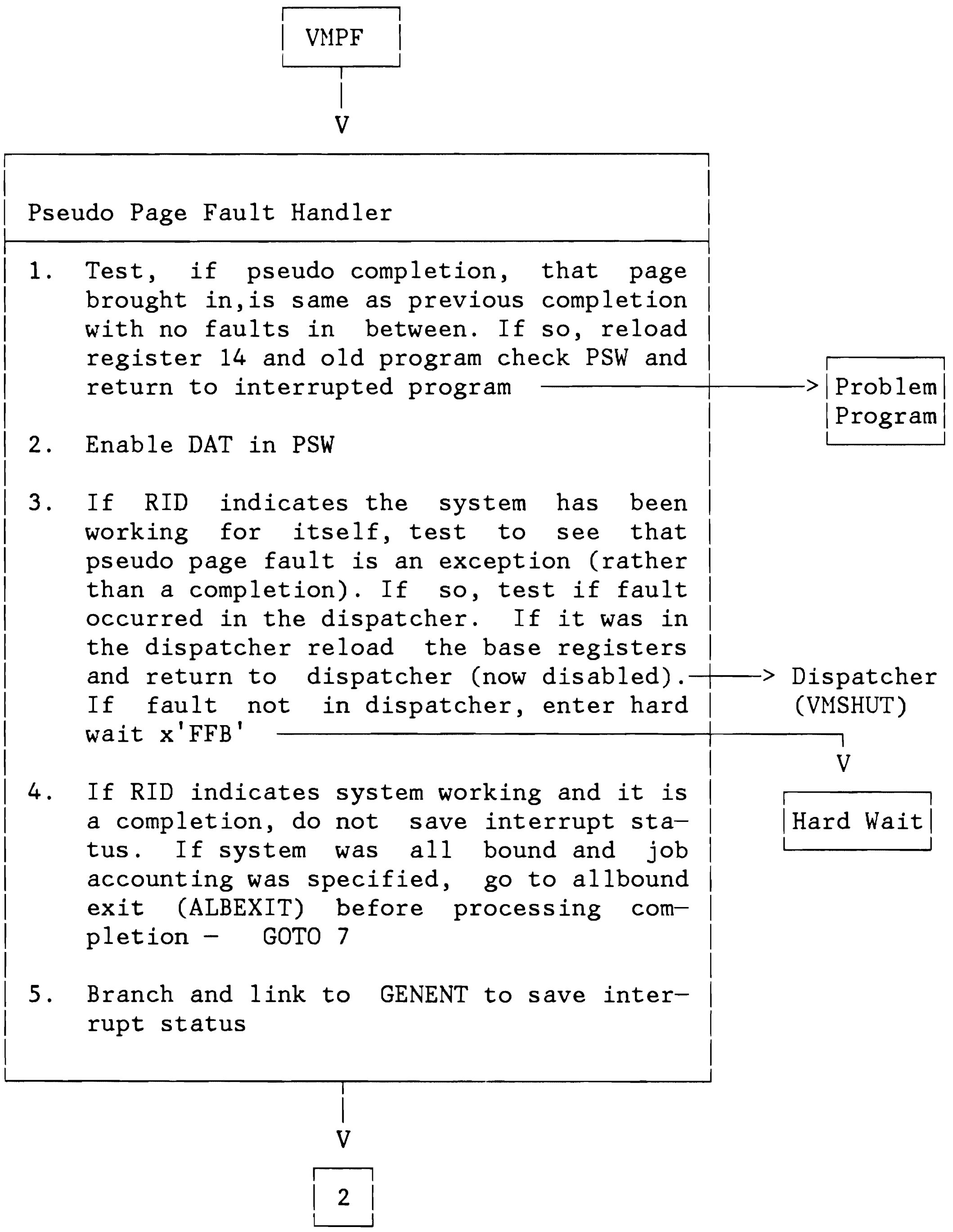
To exit to user's PC routine, do:

- Save the PC old PSW interrupt information and problem program general registers in the user supplied save area.

  (The PC old PSW is remapped when saved. The user's PC routine will be executed when this task is selected in the general exit routine. Return from the user's PC routine must be with an EXIT PC macro.)

- Store the address of the user's PC routine in the PC 'old' PSW

Figure 130. Exit to User's PC Routine



Pseudo Page Fault Handler

1. Test, if pseudo completion, that page brought in,is same as previous completion with no faults in between. If so, reload register 14 and old program check PSW and return to interrupted program

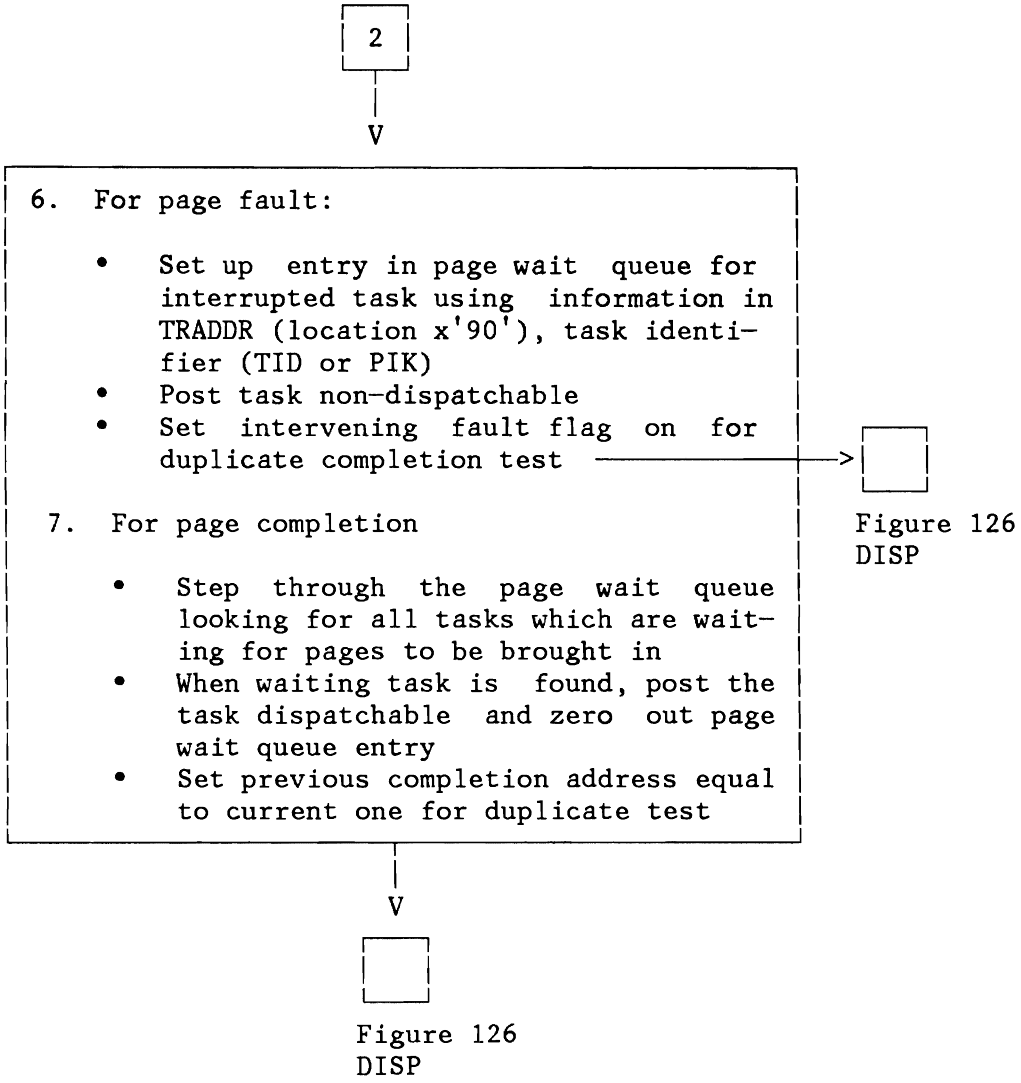
2. Enable DAT in PSW

3. If RID indicates the system has been working for itself, test to see that pseudo page fault is an exception (rather than a completion). If so, test if fault occurred in the dispatcher. If it was in the dispatcher reload the base registers and return to dispatcher (now disabled). If fault not in dispatcher, enter hard wait x'FFB'

4. If RID indicates system working and it is a completion, do not save interrupt status. If system was all bound and job accounting was specified, go to allbound exit (ALBEXIT) before processing completion - GOTO 7

5. Branch and link to GENENT to save interrupt status

Figure 131 (Part 1 of 2). Pseudo Page Fault Handler



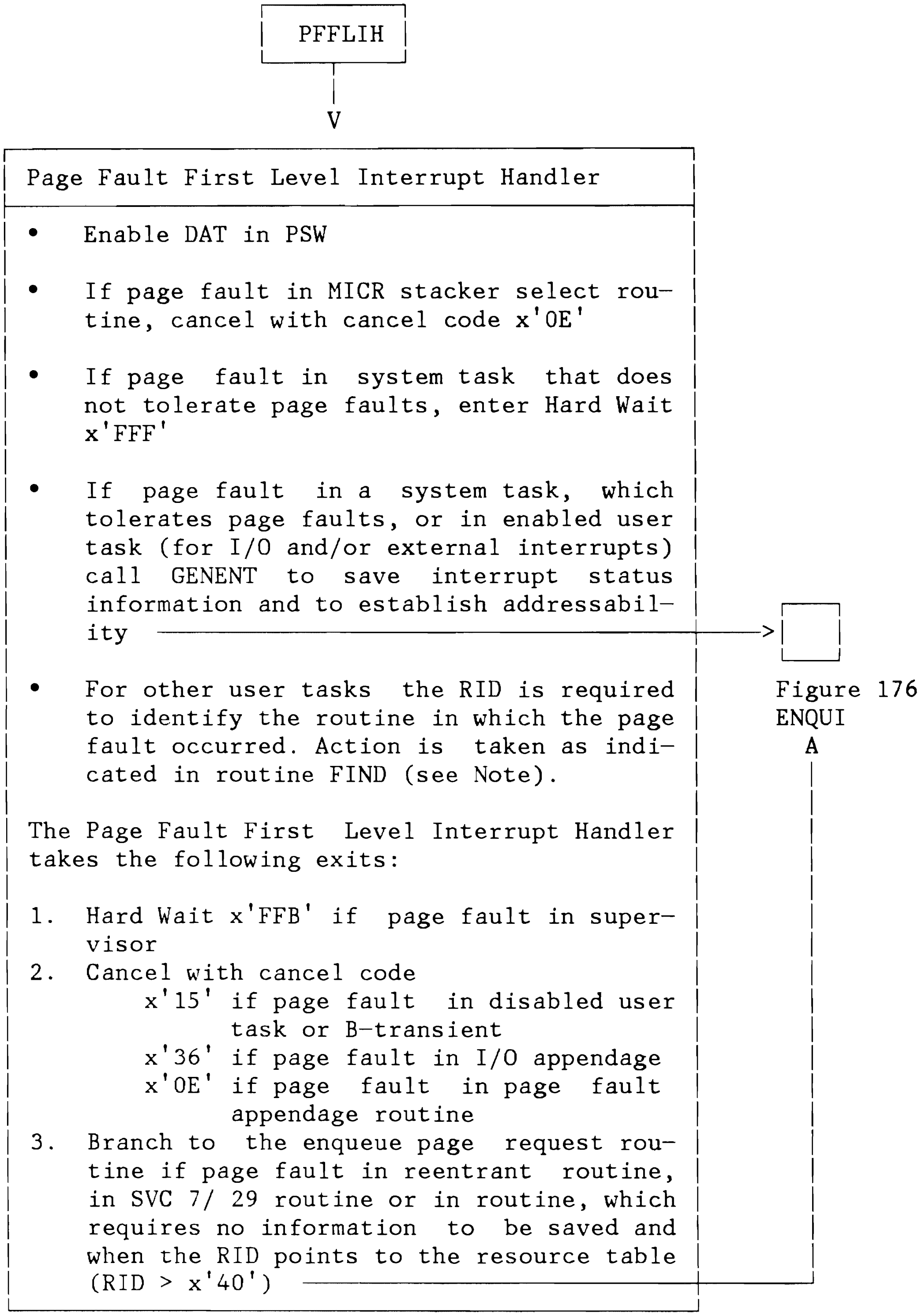
6. For page fault:

- Set up entry in page wait queue for interrupted task using information in TRADDR (location x'90'), task identifier (TID or PIK)
- Post task non-dispatchable
- Set intervening fault flag on for duplicate completion test

7. For page completion

- Step through the page wait queue looking for all tasks which are waiting for pages to be brought in
- When waiting task is found, post the task dispatchable and zero out page wait queue entry
- Set previous completion address equal to current one for duplicate test

Figure 131 (Part 2 of 2). Pseudo Page Fault Handler



Page Fault First Level Interrupt Handler

- Enable DAT in PSW
- If page fault in MICR stacker select routine, cancel with cancel code x'0E'
- If page fault in system task that does not tolerate page faults, enter Hard Wait x'FFF'
- If page fault in a system task, which tolerates page faults, or in enabled user task (for I/O and/or external interrupts) call GENENT to save interrupt status information and to establish addressability
- For other user tasks the RID is required to identify the routine in which the page fault occurred. Action is taken as indicated in routine FIND (see Note).

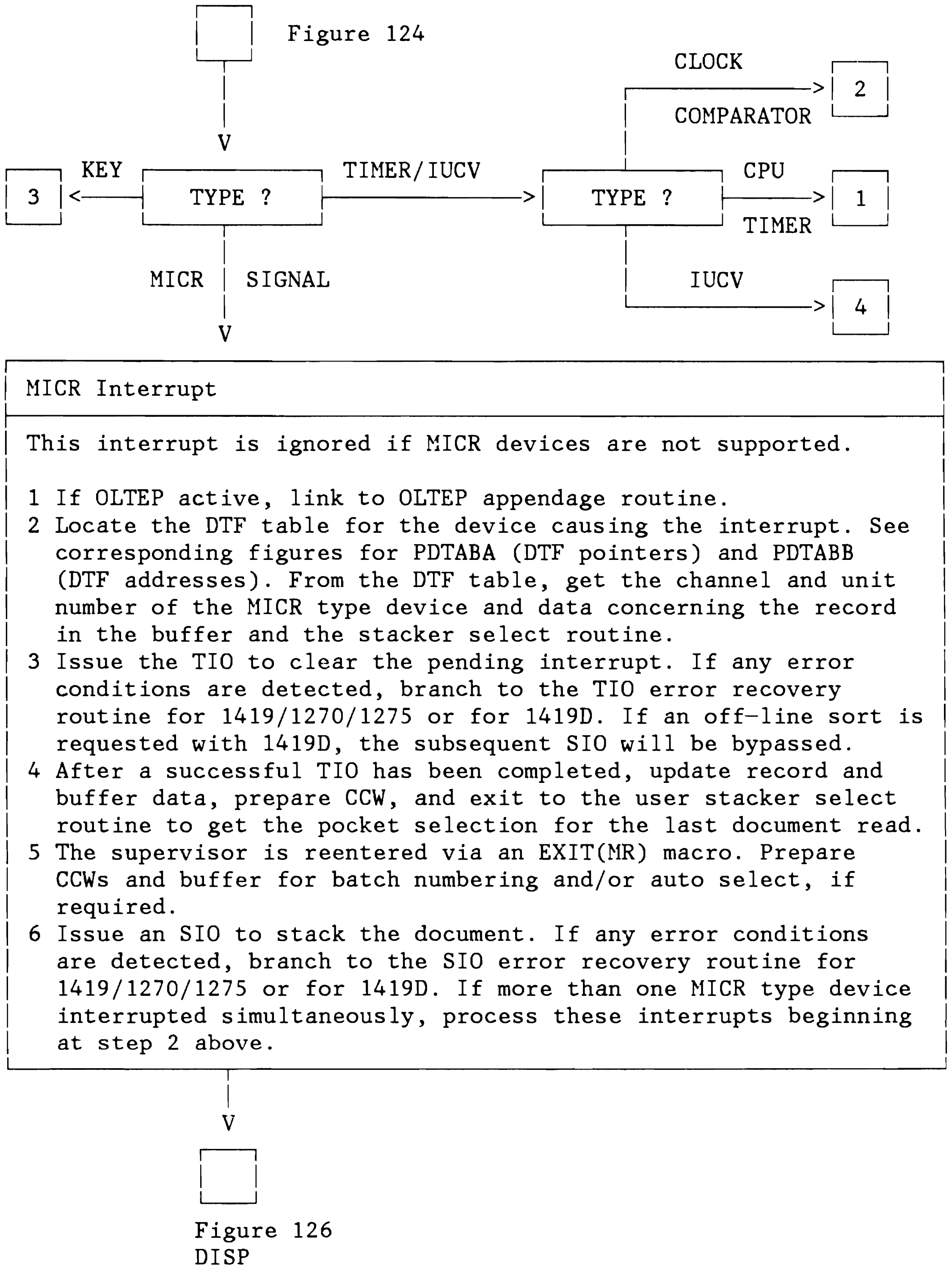
The Page Fault First Level Interrupt Handler takes the following exits:

1. Hard Wait x'FFB' if page fault in supervisor
2. Cancel with cancel code
   x'15' if page fault in disabled user task or B-transient
   x'36' if page fault in I/O appendage
   x'0E' if page fault in page fault appendage routine
3. Branch to the enqueue page request routine if page fault in reentrant routine, in SVC 7/ 29 routine or in routine, which requires no information to be saved and when the RID points to the resource table (RID > x'40')

Note: Routine FIND in Figure 217.

Figure 132. Page Fault First Level Interrupt Handler

EXTERNAL INTERRUPT ROUTINES

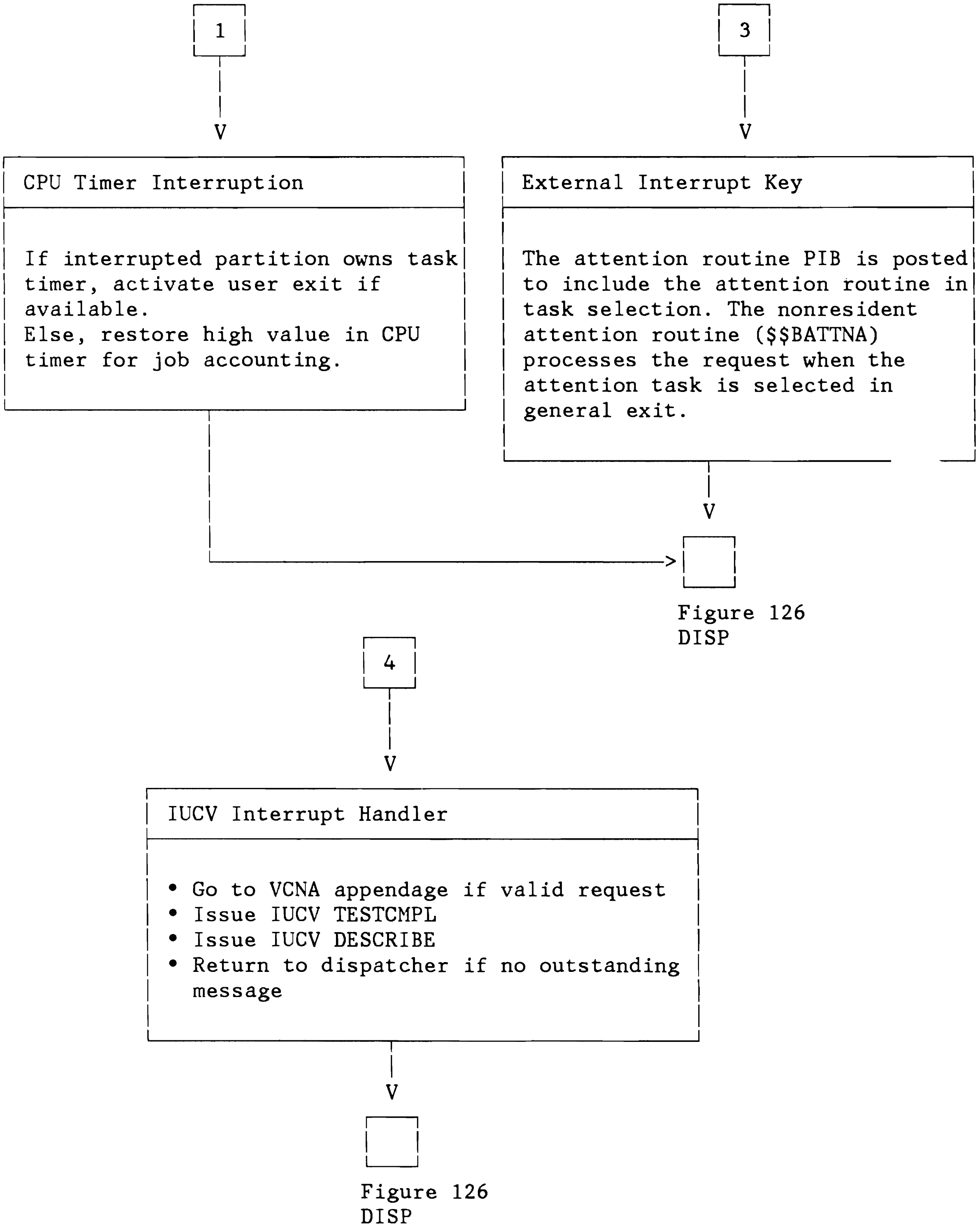


MICR Interrupt

This interrupt is ignored if MICR devices are not supported.

1 If OLTEP active, link to OLTEP appendage routine.
2 Locate the DTF table for the device causing the interrupt. See corresponding figures for PDTABA (DTF pointers) and PDTABB (DTF addresses). From the DTF table, get the channel and unit number of the MICR type device and data concerning the record in the buffer and the stacker select routine.
3 Issue the TIO to clear the pending interrupt. If any error conditions are detected, branch to the TIO error recovery routine for 1419/1270/1275 or for 1419D. If an off-line sort is requested with 1419D, the subsequent SIO will be bypassed.
4 After a successful TIO has been completed, update record and buffer data, prepare CCW, and exit to the user stacker select routine to get the pocket selection for the last document read.
5 The supervisor is reentered via an EXIT(MR) macro. Prepare CCWs and buffer for batch numbering and/or auto select, if required.
6 Issue an SIO to stack the document. If any error conditions are detected, branch to the SIO error recovery routine for 1419/1270/1275 or for 1419D. If more than one MICR type device interrupted simultaneously, process these interrupts beginning at step 2 above.

Figure 133 (Part 1 of 3). External Interrupt Routines



CPU Timer Interruption

If interrupted partition owns task timer, activate user exit if available.
Else, restore high value in CPU timer for job accounting.

External Interrupt Key

The attention routine PIB is posted to include the attention routine in task selection. The nonresident attention routine ($$BATTNA) processes the request when the attention task is selected in general exit.

IUCV Interrupt Handler

- Go to VCNA appendage if valid request
- Issue IUCV TESTCMPL
- Issue IUCV DESCRIBE
- Return to dispatcher if no outstanding message

Figure 133 (Part 2 of 3). External Interrupt Routines

2

V

Clock Comparator Interruption

If no further active interval is in the ITREQ table, the clock comparator and the current ITREQ table entry are set to the highest possible value. Else, the first entry in the ITREQ table is deleted and all other entries are shifted one slot up. The clock comparator is set to the value which appears now in the first position of the ITREQ table.

- With user TECB address - The traffic bit in the user's TECB (timer event control block) is posted and the program is included in task selection. The TECB address is cleared in ITTAB (IT option table). The TECB has the same format and is used as a normal ECB.

- With user IT routine - Before exit to the user's IT routine the following is done:
  1 - Save the interrupt status information and the timer-supported-program registers in the user supplied save area.
  2 - Store the address of the user's IT routine in this save area.
  3 - Branch to general exit. The user's IT routine will be executed when this task is selected for dispatching. Return from the user's IT routine must be with an EXIT IT macro.

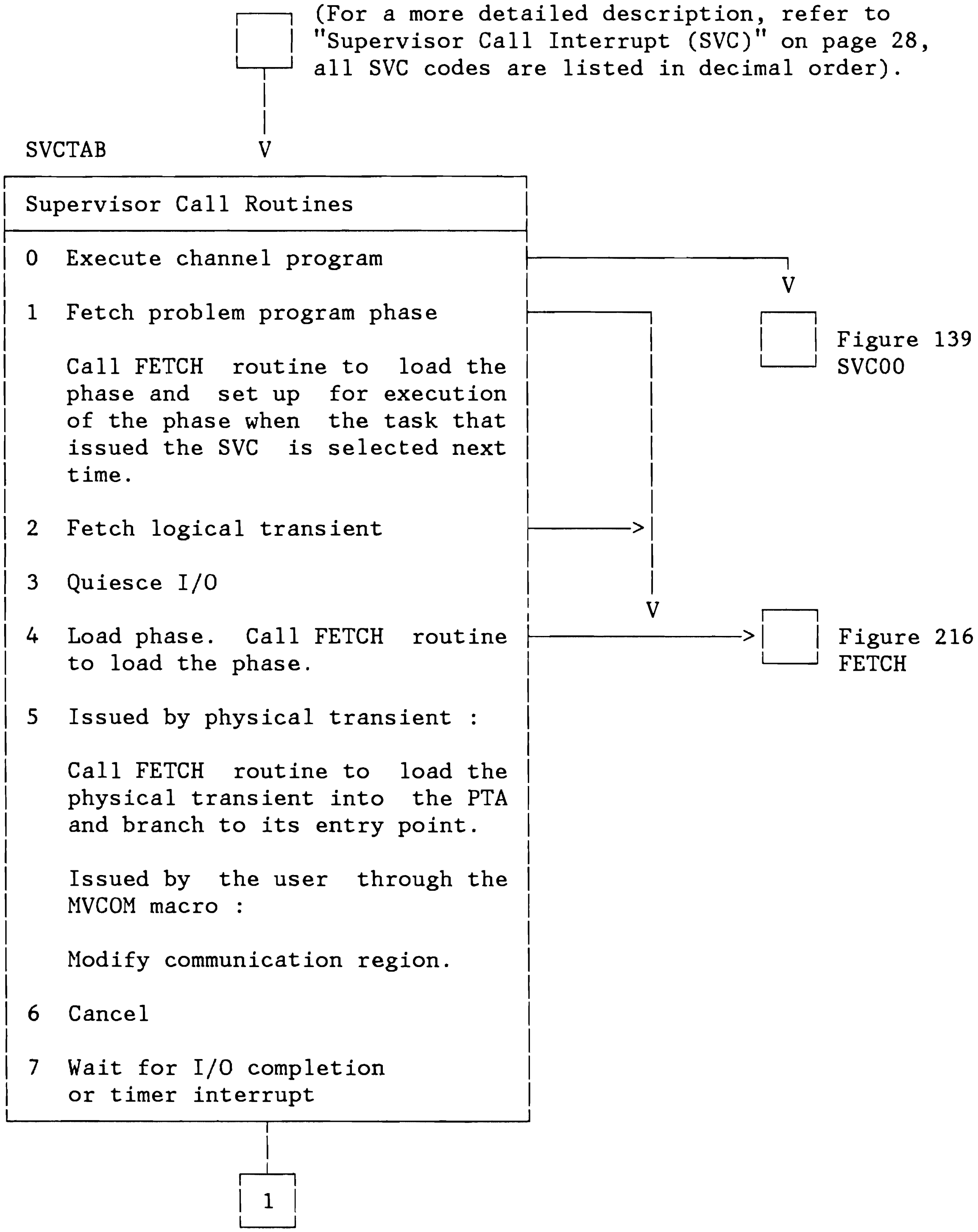
NOTE 1: If a B-transient is operating for the timer supported program when the timer interrupt occurs, interrupt handling is deferred. When on return from LTA the SVC 11 routine finds that a timer interrupt is pending, a branch is taken to the timer interrupt handler to resume processing of the interrupt.

NOTE 2: If a user's AB-routine of a timer-supported program is being processed and a timer interrupt occurs, the interrupt handling is deferred until AB-routine is completed with EXIT AB. The timer interrupt handler gets control and continues to process the timer interrupt.

V

Figure 126
DISP

Figure 133 (Part 3 of 3). External Interrupt Routines



(For a more detailed description, refer to "Supervisor Call Interrupt (SVC)" on page 28, all SVC codes are listed in decimal order).

SVCTAB

Supervisor Call Routines

0 Execute channel program

1 Fetch problem program phase

Call FETCH routine to load the phase and set up for execution of the phase when the task that issued the SVC is selected next time.

2 Fetch logical transient

3 Quiesce I/O

4 Load phase. Call FETCH routine to load the phase.

5 Issued by physical transient :

Call FETCH routine to load the physical transient into the PTA and branch to its entry point.

Issued by the user through the MVCOM macro :

Modify communication region.

6 Cancel

7 Wait for I/O completion or timer interrupt

Figure 134 (Part 1 of 9). Supervisor Call Routines

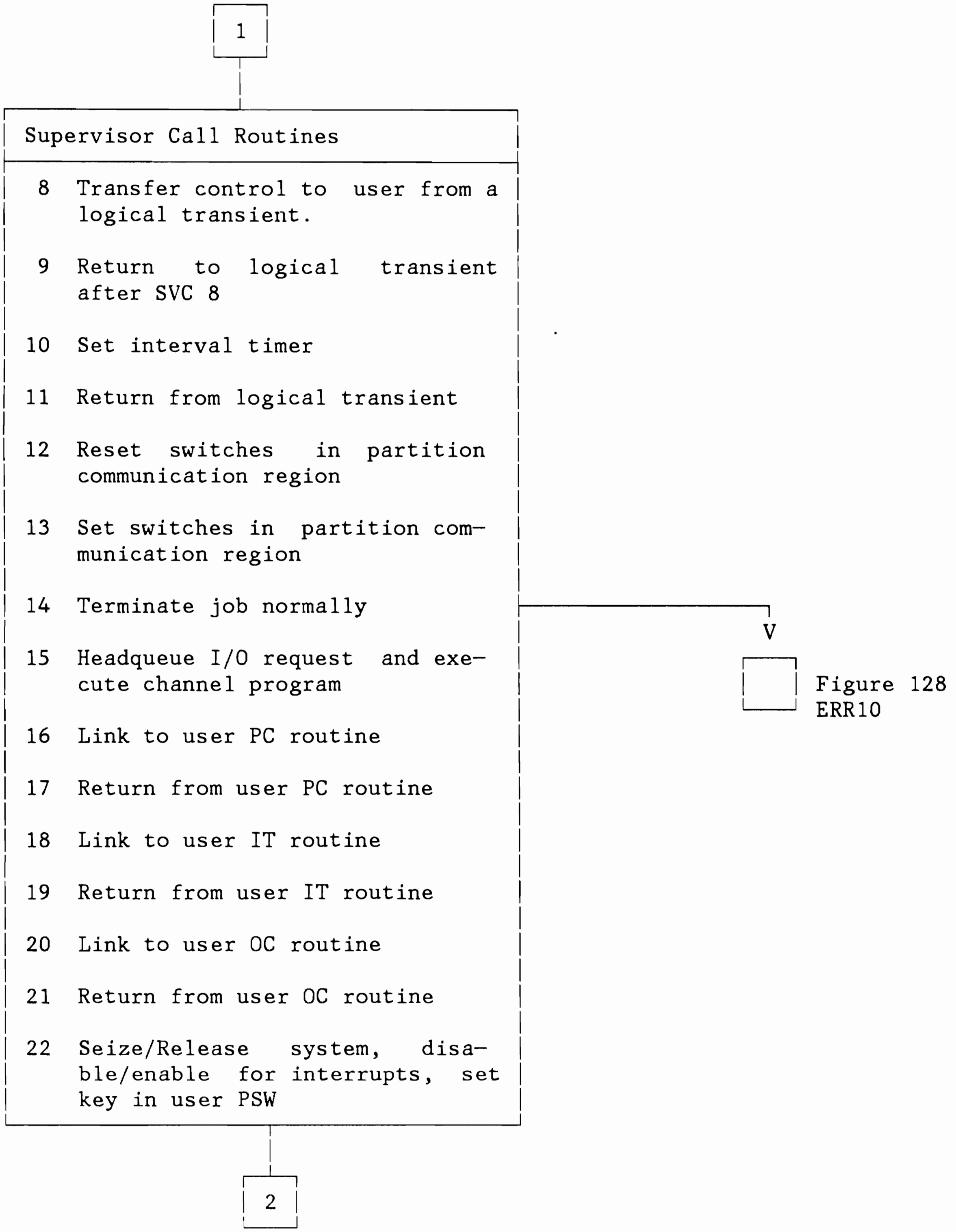
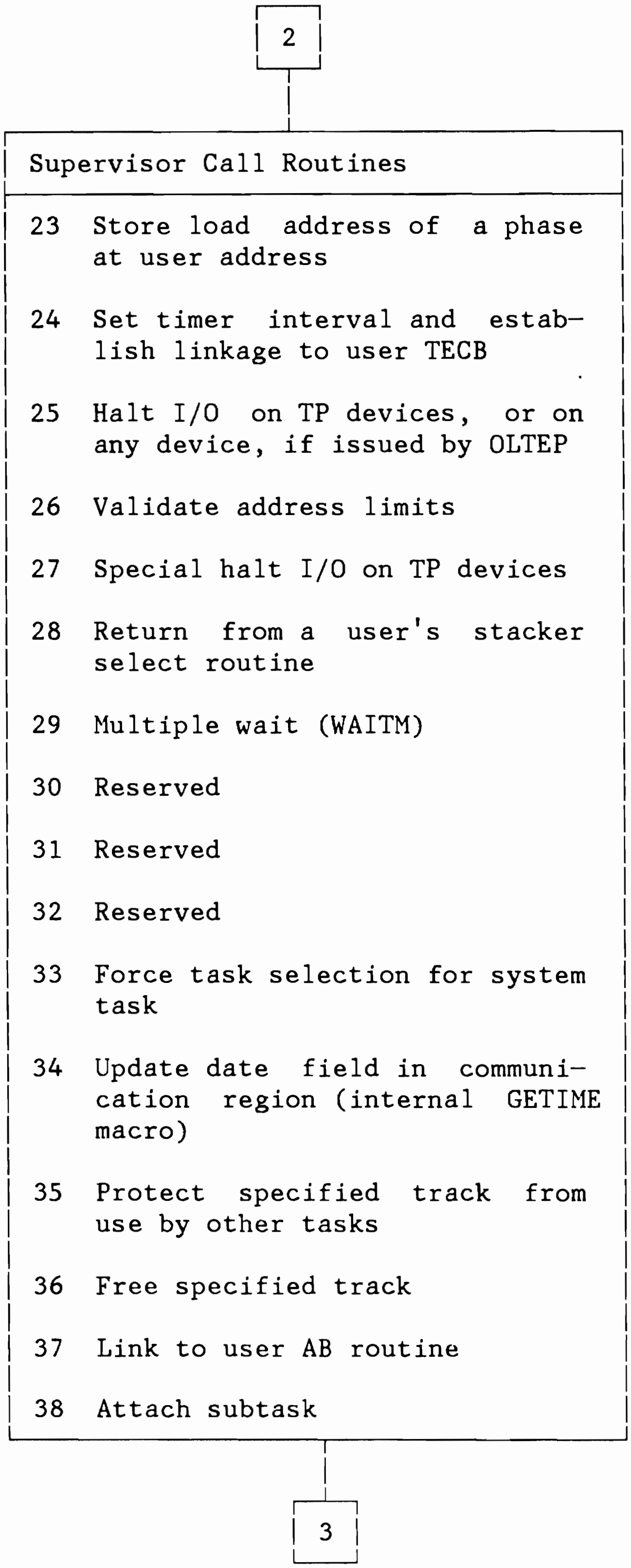

1
Supervisor Call Routines
8 Transfer control to user from a logical transient.
9 Return to logical transient after SVC 8
10 Set interval timer
11 Return from logical transient
12 Reset switches in partition communication region
13 Set switches in partition communication region
14 Terminate job normally
15 Headqueue I/O request and execute channel program
16 Link to user PC routine
17 Return from user PC routine
18 Link to user IT routine
19 Return from user IT routine
20 Link to user OC routine
21 Return from user OC routine
22 Seize/Release system, disable/enable for interrupts, set key in user PSW
V
Figure 128
ERR10
2

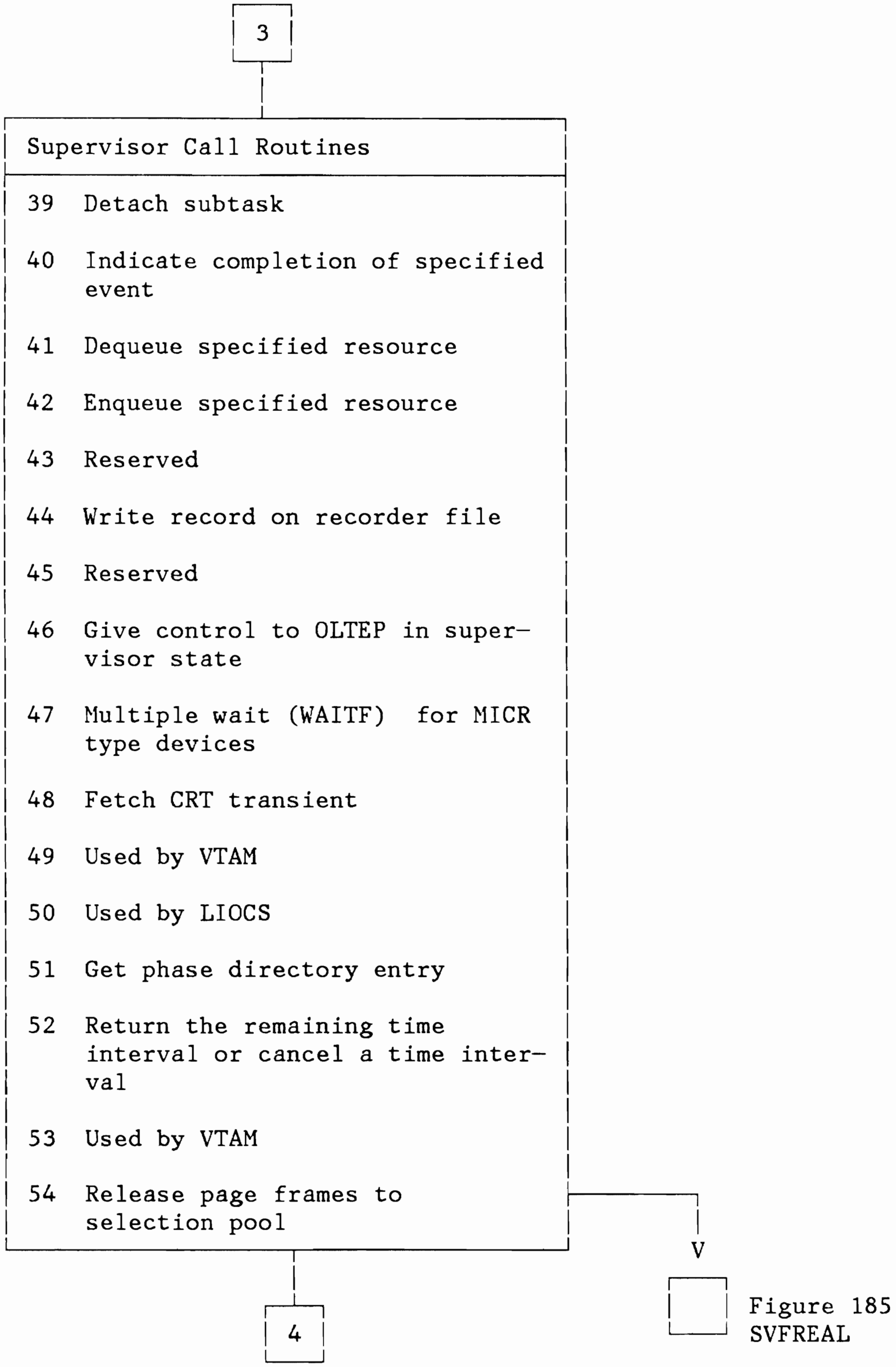
Figure 134 (Part 2 of 9). Supervisor Call Routines



Supervisor Call Routines

23	Store load address of a phase at user address
24	Set timer interval and establish linkage to user TECB
25	Halt I/O on TP devices, or on any device, if issued by OLTEP
26	Validate address limits
27	Special halt I/O on TP devices
28	Return from a user's stacker select routine
29	Multiple wait (WAITM)
30	Reserved
31	Reserved
32	Reserved
33	Force task selection for system task
34	Update date field in communication region (internal GETIME macro)
35	Protect specified track from use by other tasks
36	Free specified track
37	Link to user AB routine
38	Attach subtask

Figure 134 (Part 3 of 9). Supervisor Call Routines



Supervisor Call Routines

39 Detach subtask

40 Indicate completion of specified event

41 Dequeue specified resource

42 Enqueue specified resource

43 Reserved

44 Write record on recorder file

45 Reserved

46 Give control to OLTEP in supervisor state

47 Multiple wait (WAITF) for MICR type devices

48 Fetch CRT transient

49 Used by VTAM

50 Used by LIOCS

51 Get phase directory entry

52 Return the remaining time interval or cancel a time interval

53 Used by VTAM

54 Release page frames to selection pool

Figure 134 (Part 4 of 9). Supervisor Call Routines

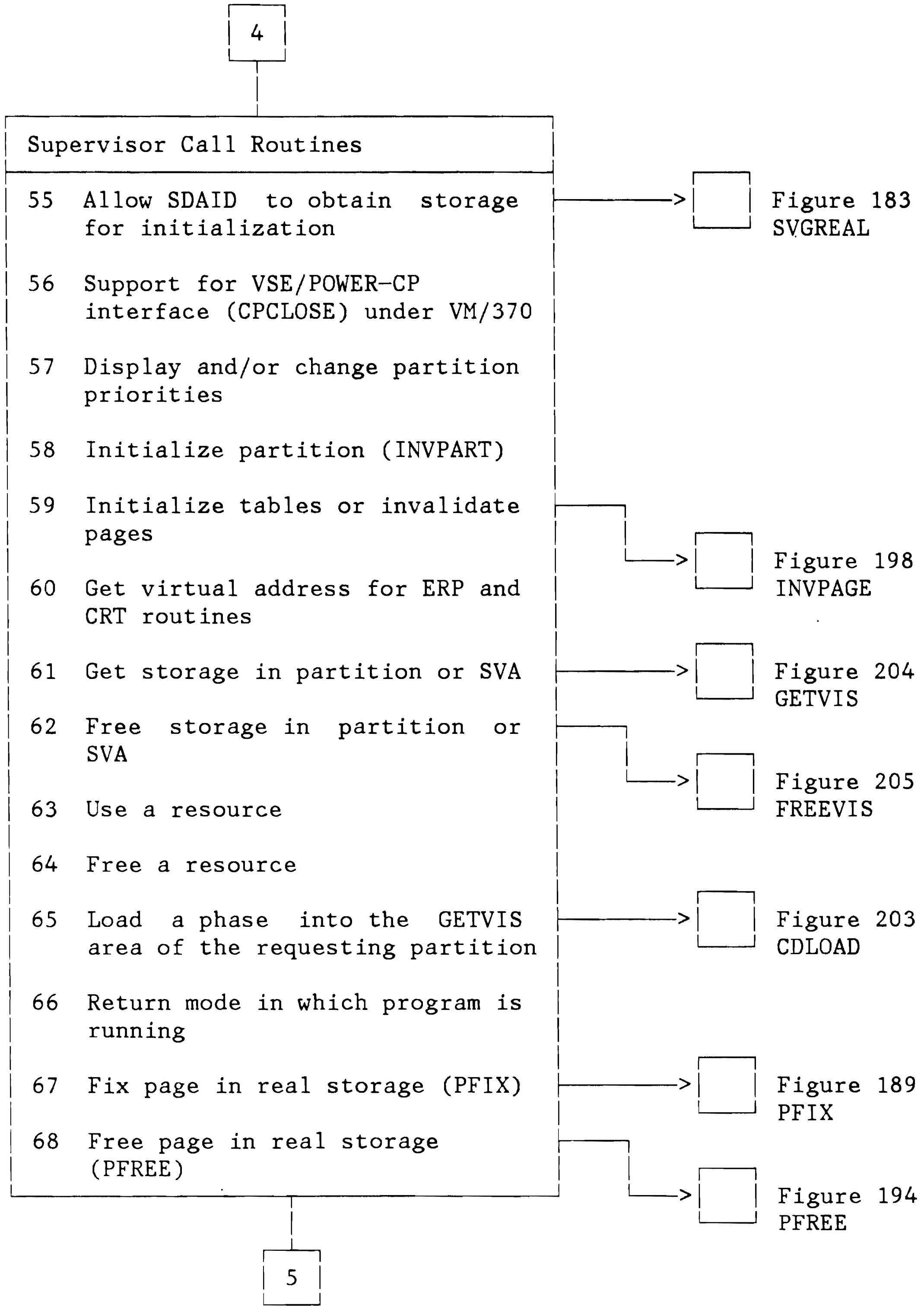
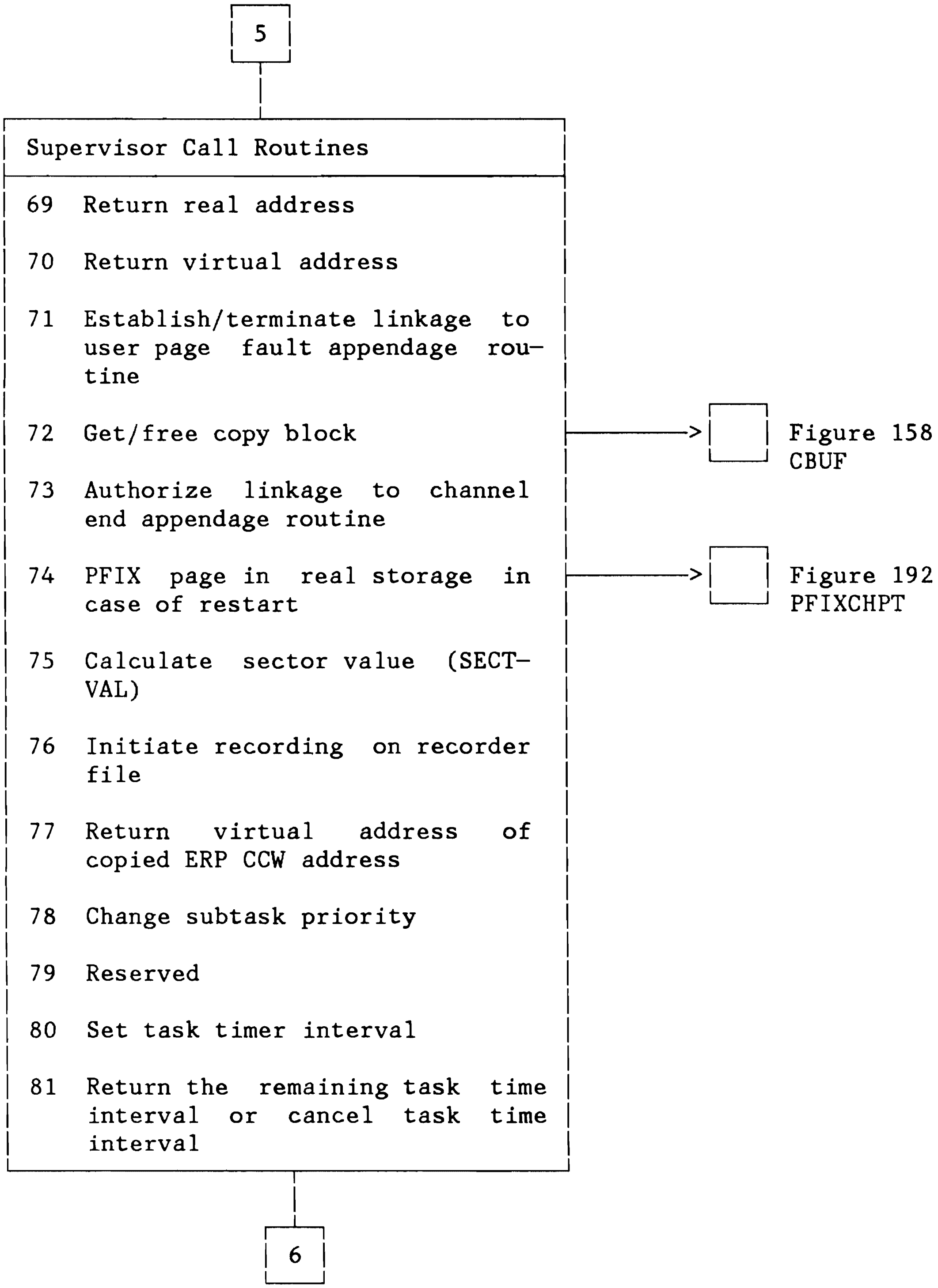

4
Supervisor Call Routines
55 Allow SDAID to obtain storage for initialization
56 Support for VSE/POWER-CP interface (CPCLOSE) under VM/370
57 Display and/or change partition priorities
58 Initialize partition (INVPART)
59 Initialize tables or invalidate pages
60 Get virtual address for ERP and CRT routines
61 Get storage in partition or SVA
62 Free storage in partition or SVA
63 Use a resource
64 Free a resource
65 Load a phase into the GETVIS area of the requesting partition
66 Return mode in which program is running
67 Fix page in real storage (PFIX)
68 Free page in real storage (PFREE)
Figure 183 SVGREAL
Figure 198 INVPAGE
Figure 204 GETVIS
Figure 205 FREEVIS
Figure 203 CDLOAD
Figure 189 PFIX
Figure 194 PFREE
5

Figure 134 (Part 5 of 9). Supervisor Call Routines



Supervisor Call Routines

69 Return real address

70 Return virtual address

71 Establish/terminate linkage to user page fault appendage routine

72 Get/free copy block

73 Authorize linkage to channel end appendage routine

74 PFIX page in real storage in case of restart

75 Calculate sector value (SECTVAL)

76 Initiate recording on recorder file

77 Return virtual address of copied ERP CCW address

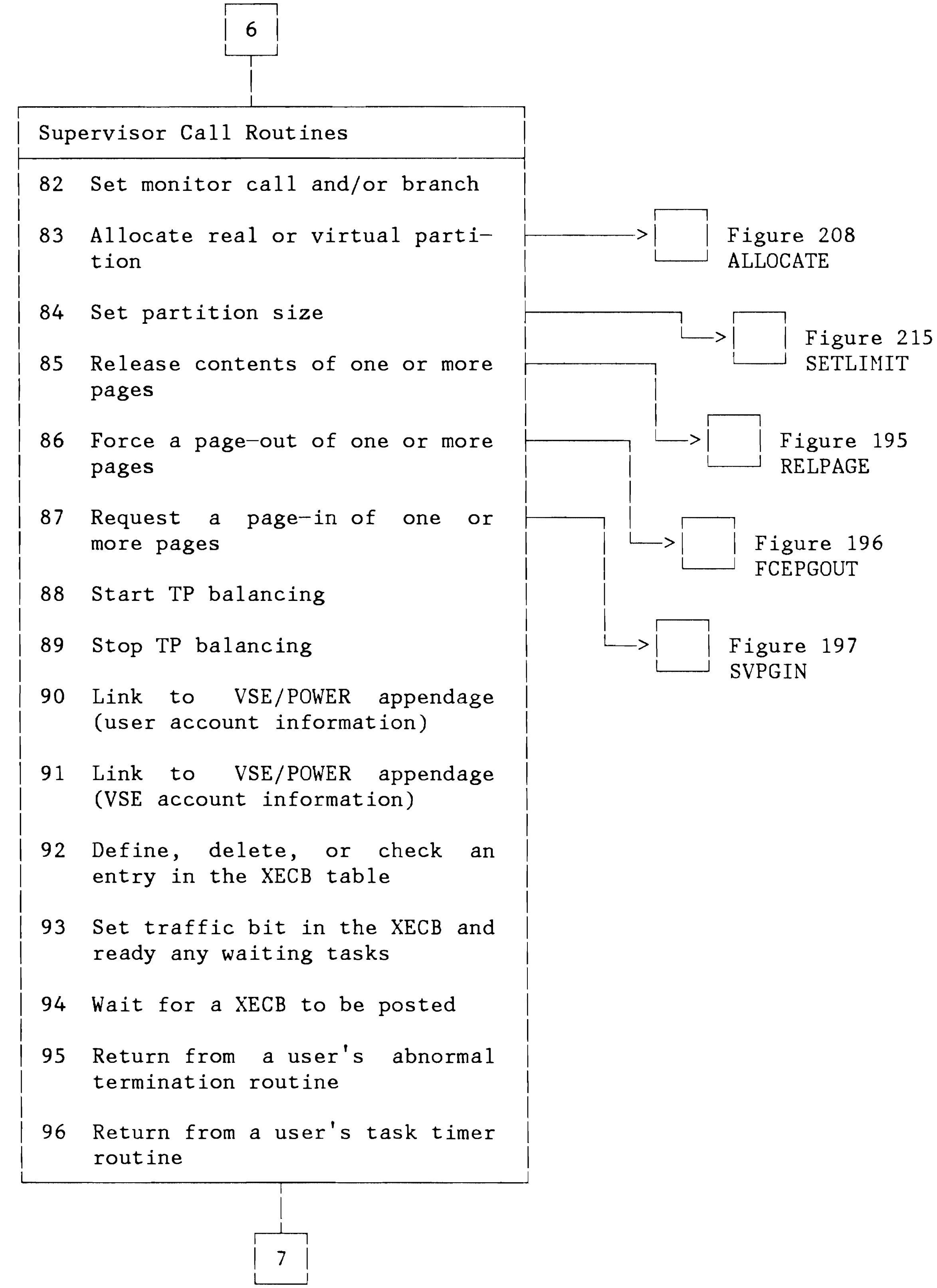
78 Change subtask priority

79 Reserved

80 Set task timer interval

81 Return the remaining task time interval or cancel task time interval

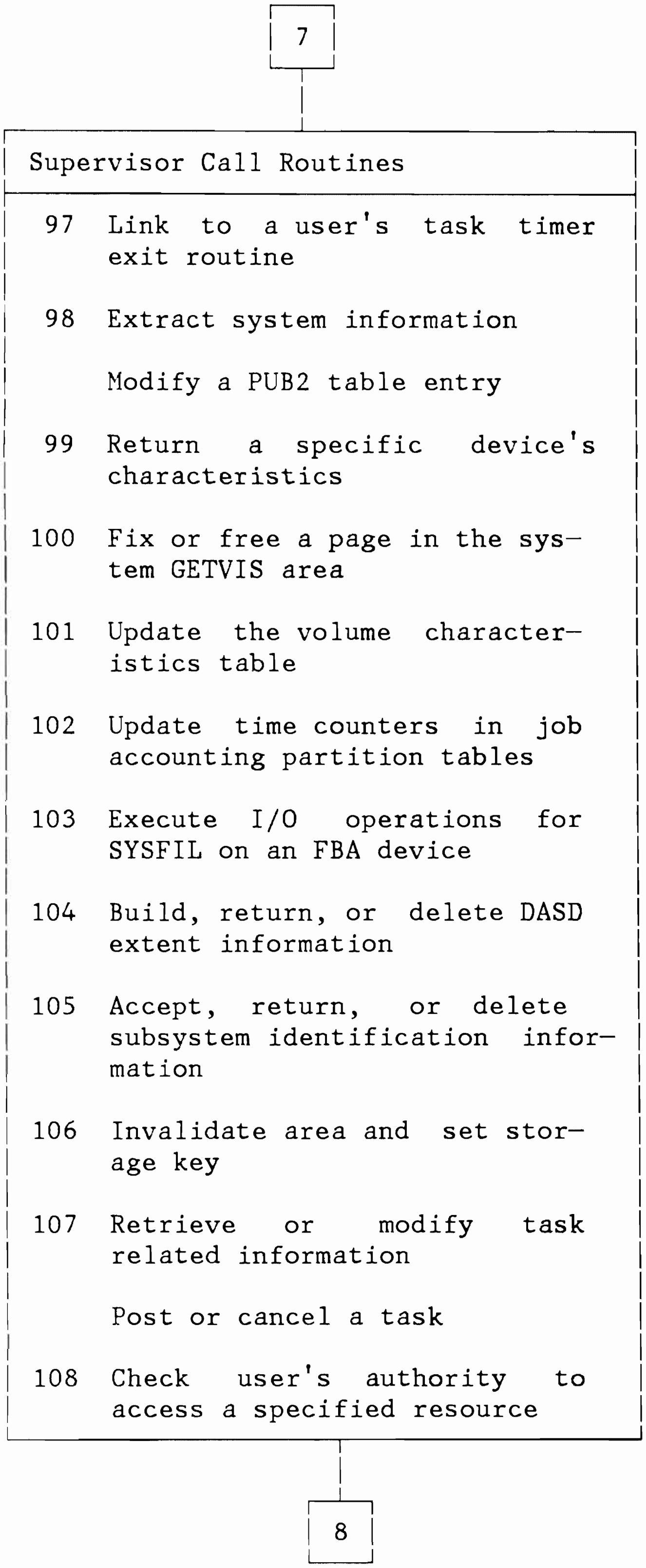
Figure 134 (Part 6 of 9). Supervisor Call Routines



Supervisor Call Routines

82	Set monitor call and/or branch
83	Allocate real or virtual partition
84	Set partition size
85	Release contents of one or more pages
86	Force a page-out of one or more pages
87	Request a page-in of one or more pages
88	Start TP balancing
89	Stop TP balancing
90	Link to VSE/POWER appendage (user account information)
91	Link to VSE/POWER appendage (VSE account information)
92	Define, delete, or check an entry in the XECB table
93	Set traffic bit in the XECB and ready any waiting tasks
94	Wait for a XECB to be posted
95	Return from a user's abnormal termination routine
96	Return from a user's task timer routine

Figure 134 (Part 7 of 9). Supervisor Call Routines



7

Supervisor Call Routines

97 Link to a user's task timer exit routine

98 Extract system information

Modify a PUB2 table entry

99 Return a specific device's characteristics

100 Fix or free a page in the system GETVIS area

101 Update the volume characteristics table

102 Update time counters in job accounting partition tables

103 Execute I/O operations for SYSFIL on an FBA device

104 Build, return, or delete DASD extent information

105 Accept, return, or delete subsystem identification information

106 Invalidate area and set storage key

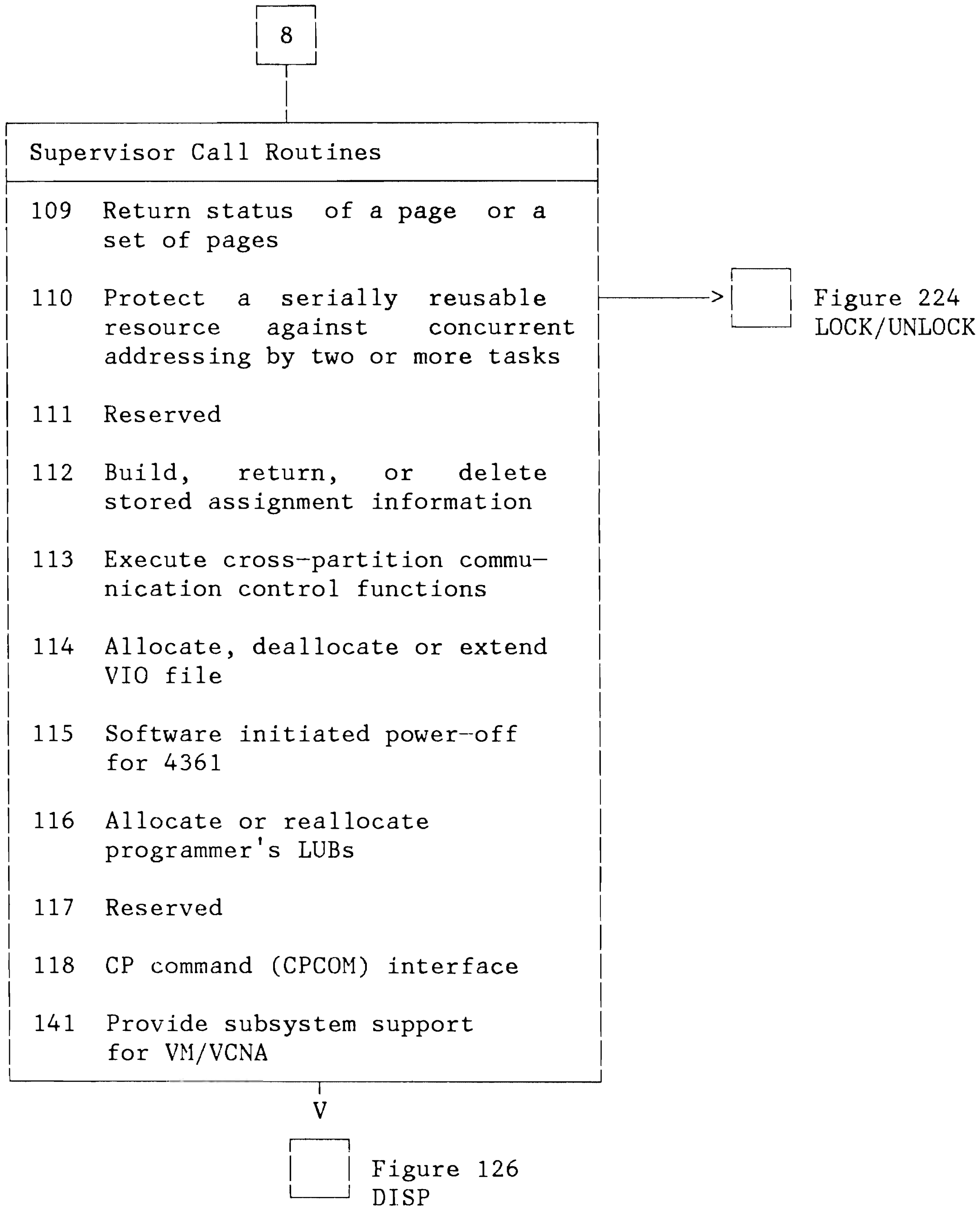
107 Retrieve or modify task related information

Post or cancel a task

108 Check user's authority to access a specified resource

8

Figure 134 (Part 8 of 9). Supervisor Call Routines



Supervisor Call Routines

109 Return status of a page or a set of pages

110 Protect a serially reusable resource against concurrent addressing by two or more tasks

111 Reserved

112 Build, return, or delete stored assignment information

113 Execute cross-partition communication control functions

114 Allocate, deallocate or extend VIO file

115 Software initiated power-off for 4361

116 Allocate or reallocate programmer's LUBs

117 Reserved

118 CP command (CPCOM) interface

141 Provide subsystem support for VM/VCNA

Figure 134 (Part 9 of 9). Supervisor Call Routines

I/O ROUTINES

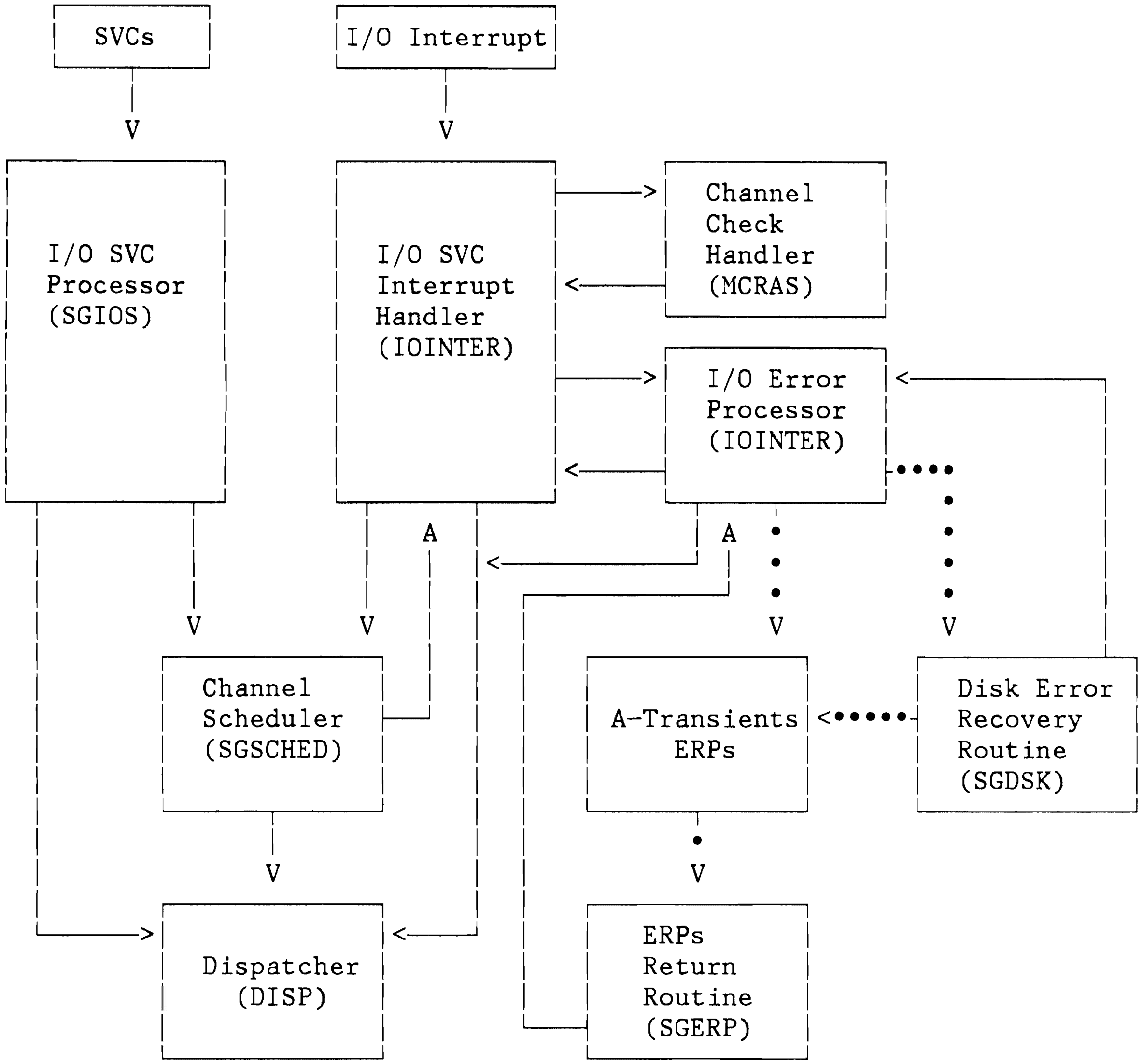

SVCs
I/O Interrupt
I/O SVC Processor (SGIOS)
I/O SVC Interrupt Handler (IOINTER)
Channel Check Handler (MCRAS)
I/O Error Processor (IOINTER)
A
A
Channel Scheduler (SGSCHED)
A-Transients ERPs
Disk Error Recovery Routine (SGDSK)
Dispatcher (DISP)
ERPs Return Routine (SGERP)

Figure 135. General Overview and Control Flow of I/O Routines

Notes:

1. Dotted lines indicate, entered via dispatcher.
2. References:

DISP	Figure 126 on page 307
IOINTER	Figure 136 on page 333 (interrupt handler) and Figure 137 on page 342 (error processor)
MCRAS	"MCH/CCH Routines" on page 460
SGDSK	Figure 141 on page 351
SGERP	Figure 142 on page 352
SGSCHED	Figure 138 on page 344

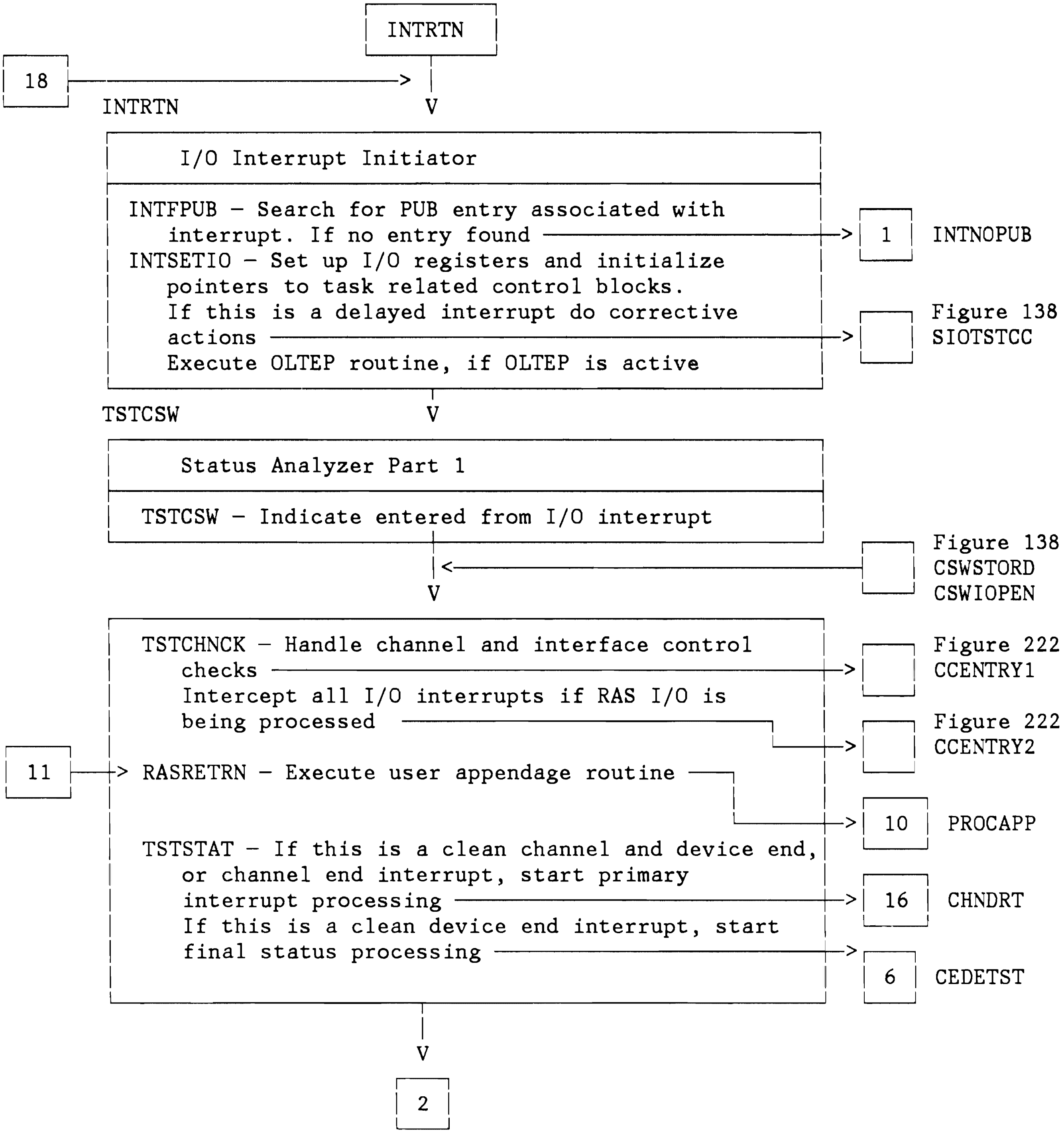
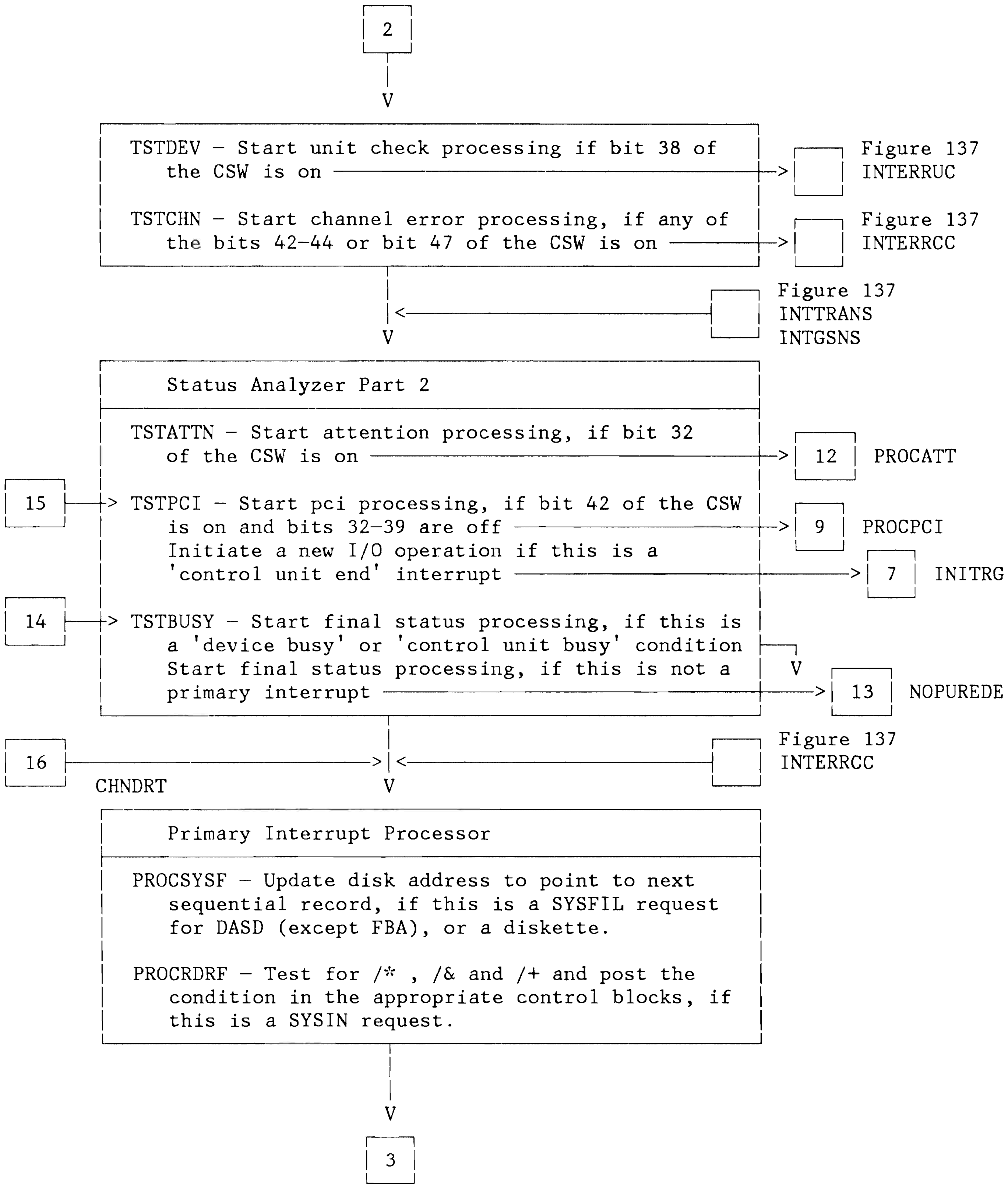

INTRTN
18
INTRTN
I/O Interrupt Initiator
INTFPUB - Search for PUB entry associated with interrupt. If no entry found
1
INTNOPUB
INTSETIO - Set up I/O registers and initialize pointers to task related control blocks.
If this is a delayed interrupt do corrective actions
Figure 138
SIOTSTCC
Execute OLTEP routine, if OLTEP is active
TSTCSW
Status Analyzer Part 1
TSTCSW - Indicate entered from I/O interrupt
Figure 138
CSWSTORD
CSWIOPEN
TSTCHNCK - Handle channel and interface control checks
Figure 222
CCENTRY1
Intercept all I/O interrupts if RAS I/O is being processed
Figure 222
CCENTRY2
11
RASRETRN - Execute user appendage routine
10
PROCAPP
TSTSTAT - If this is a clean channel and device end, or channel end interrupt, start primary interrupt processing
16
CHNDRT
If this is a clean device end interrupt, start final status processing
6
CEDETST
2

Figure 136 (Part 1 of 9). I/O Interrupt Handler



TSTDEV - Start unit check processing if bit 38 of the CSW is on

TSTCHN - Start channel error processing, if any of the bits 42-44 or bit 47 of the CSW is on

Status Analyzer Part 2

TSTATTN - Start attention processing, if bit 32 of the CSW is on

TSTPCI - Start pci processing, if bit 42 of the CSW is on and bits 32-39 are off
Initiate a new I/O operation if this is a 'control unit end' interrupt

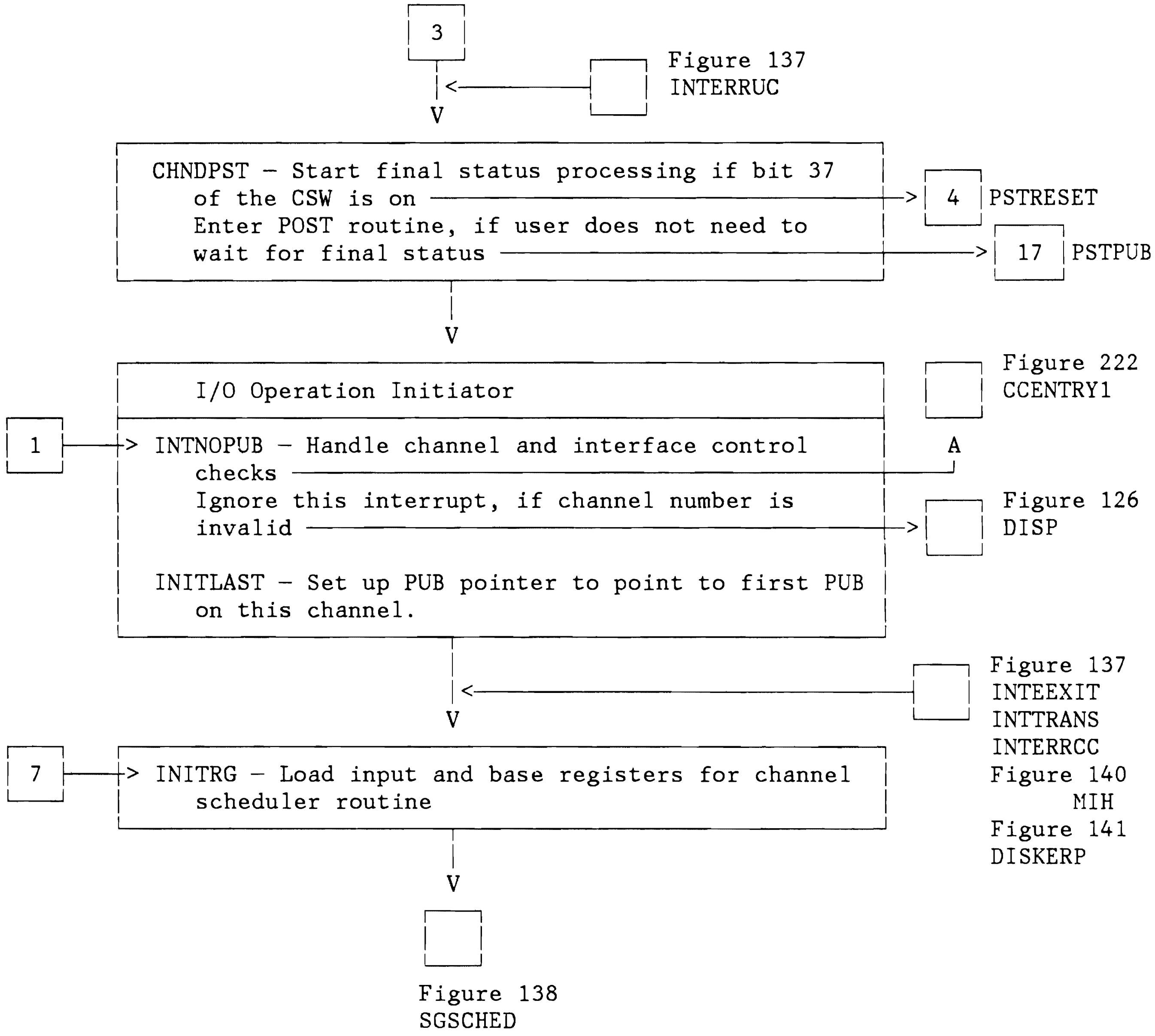
TSTBUSY - Start final status processing, if this is a 'device busy' or 'control unit busy' condition
Start final status processing, if this is not a primary interrupt

Primary Interrupt Processor

PROCSYSF - Update disk address to point to next sequential record, if this is a SYSFIL request for DASD (except FBA), or a diskette.

PROCRDRF - Test for /* , /& and /+ and post the condition in the appropriate control blocks, if this is a SYSIN request.

Figure 136 (Part 2 of 9). I/O Interrupt Handler



CHNDPST - Start final status processing if bit 37 of the CSW is on
Enter POST routine, if user does not need to wait for final status

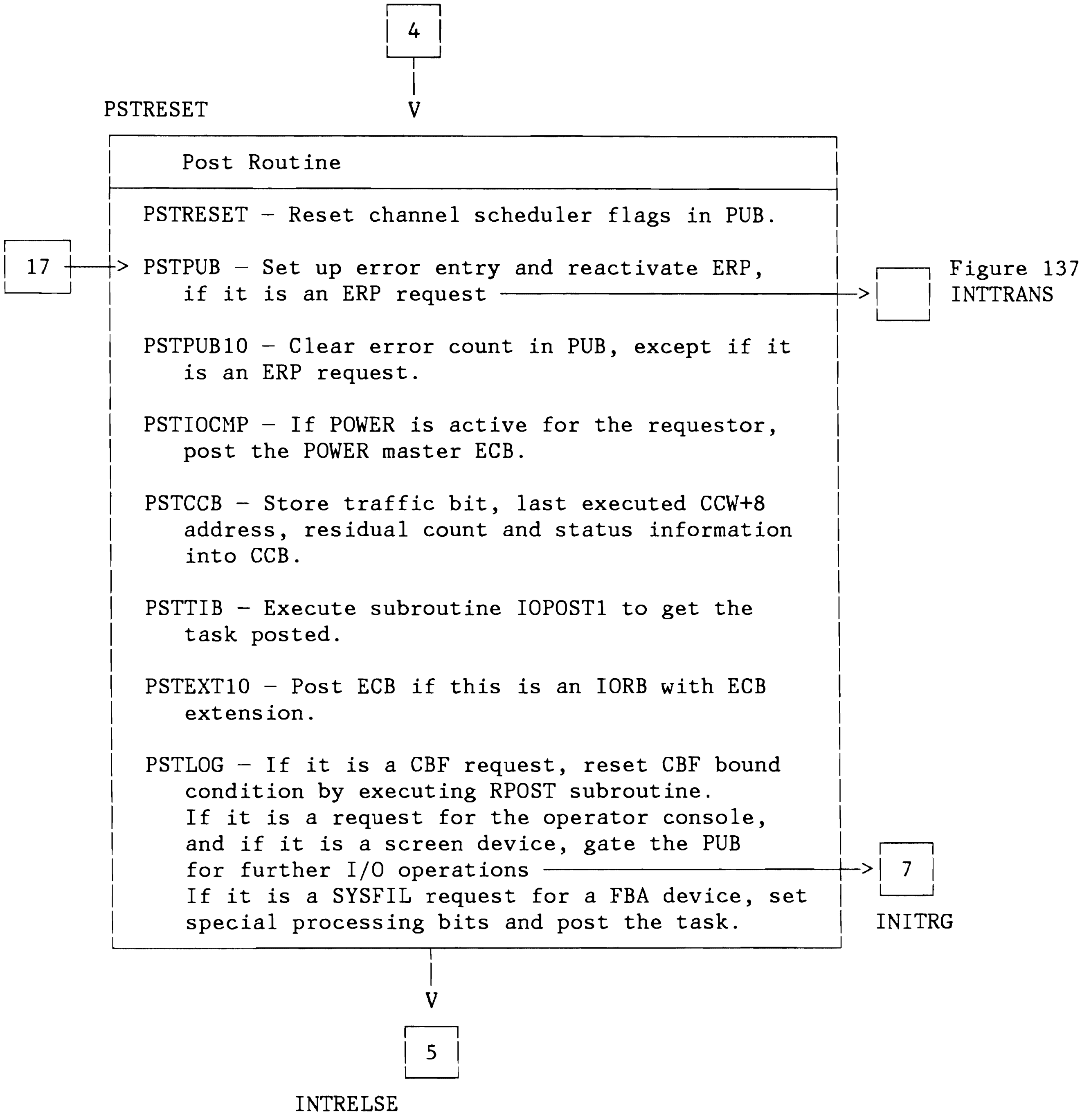
I/O Operation Initiator

INTNOPUB - Handle channel and interface control checks
Ignore this interrupt, if channel number is invalid

INITLAST - Set up PUB pointer to point to first PUB on this channel.

INITRG - Load input and base registers for channel scheduler routine

Figure 136 (Part 3 of 9). I/O Interrupt Handler



Post Routine

PSTRESET - Reset channel scheduler flags in PUB.

PSTPUB - Set up error entry and reactivate ERP, if it is an ERP request

PSTPUB10 - Clear error count in PUB, except if it is an ERP request.

PSTIOCMP - If POWER is active for the requestor, post the POWER master ECB.

PSTCCB - Store traffic bit, last executed CCW+8 address, residual count and status information into CCB.

PSTTIB - Execute subroutine IOPOST1 to get the task posted.

PSTEXT10 - Post ECB if this is an IORB with ECB extension.

PSTLOG - If it is a CBF request, reset CBF bound condition by executing RPOST subroutine. If it is a request for the operator console, and if it is a screen device, gate the PUB for further I/O operations If it is a SYSFIL request for a FBA device, set special processing bits and post the task.

Figure 136 (Part 4 of 9). I/O Interrupt Handler

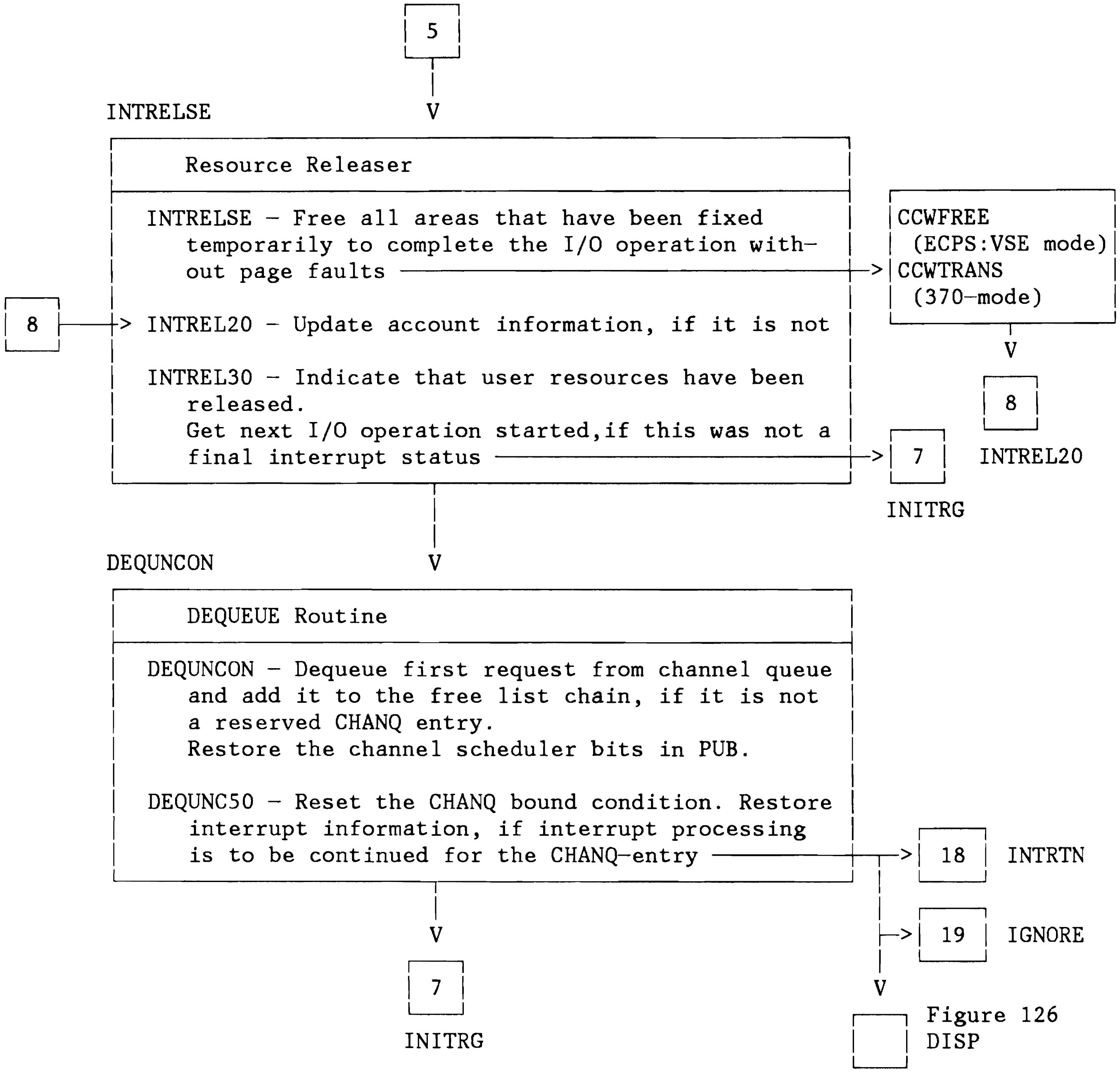
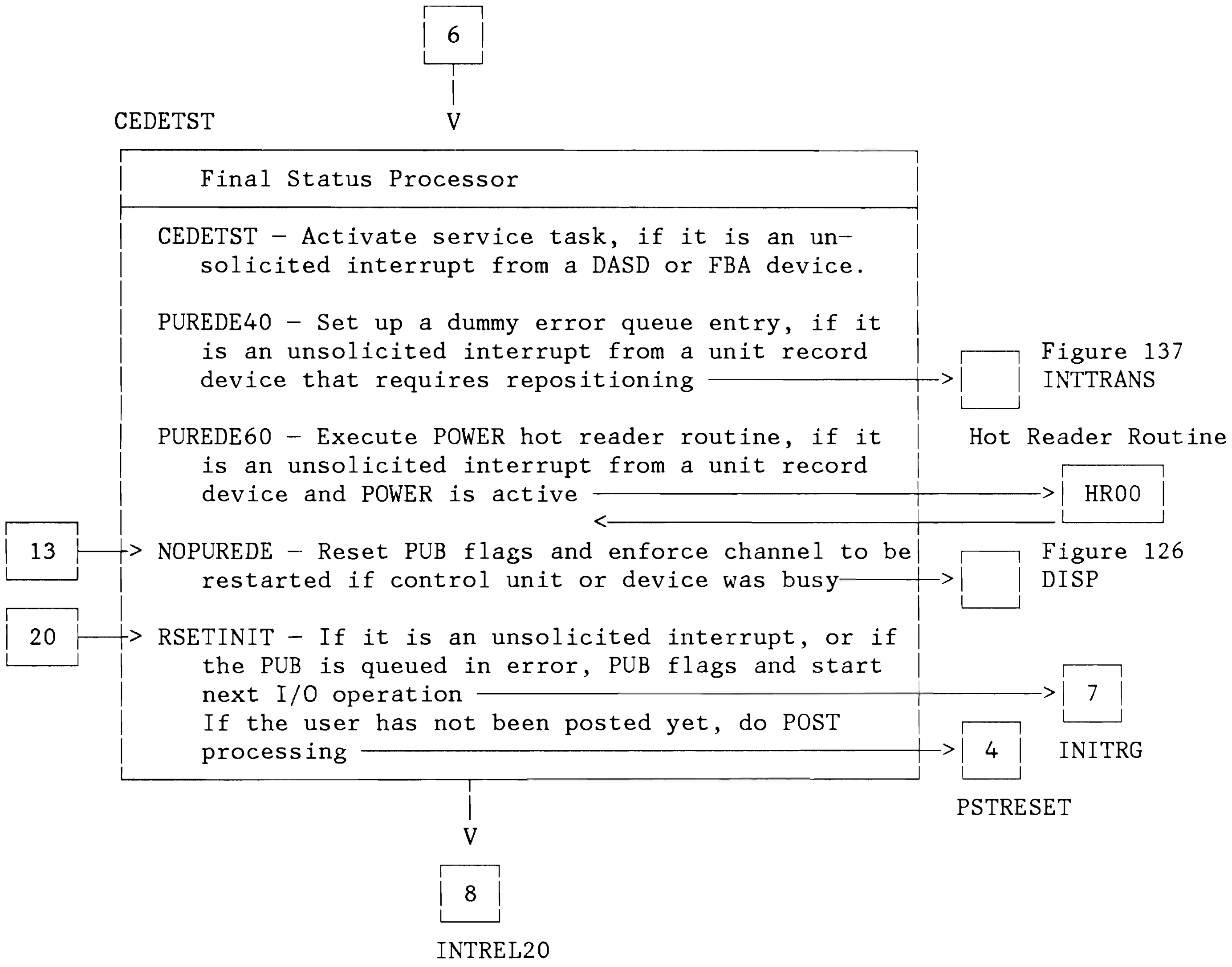

5
INTRELSE
Resource Releaser
INTRELSE - Free all areas that have been fixed temporarily to complete the I/O operation without page faults
CCWFREE (ECPS:VSE mode)
CCWTRANS (370-mode)
8
INTREL20 - Update account information, if it is not
8
INTREL20
INTREL30 - Indicate that user resources have been released.
Get next I/O operation started,if this was not a final interrupt status
7
INITRG
DEQUNCON
DEQUEUE Routine
DEQUNCON - Dequeue first request from channel queue and add it to the free list chain, if it is not a reserved CHANQ entry.
Restore the channel scheduler bits in PUB.
DEQUNC50 - Reset the CHANQ bound condition. Restore interrupt information, if interrupt processing is to be continued for the CHANQ-entry
18
INTRTN
19
IGNORE
7
INITRG
Figure 126
DISP

Figure 136 (Part 5 of 9). I/O Interrupt Handler



Final Status Processor

CEDETST - Activate service task, if it is an unsolicited interrupt from a DASD or FBA device.

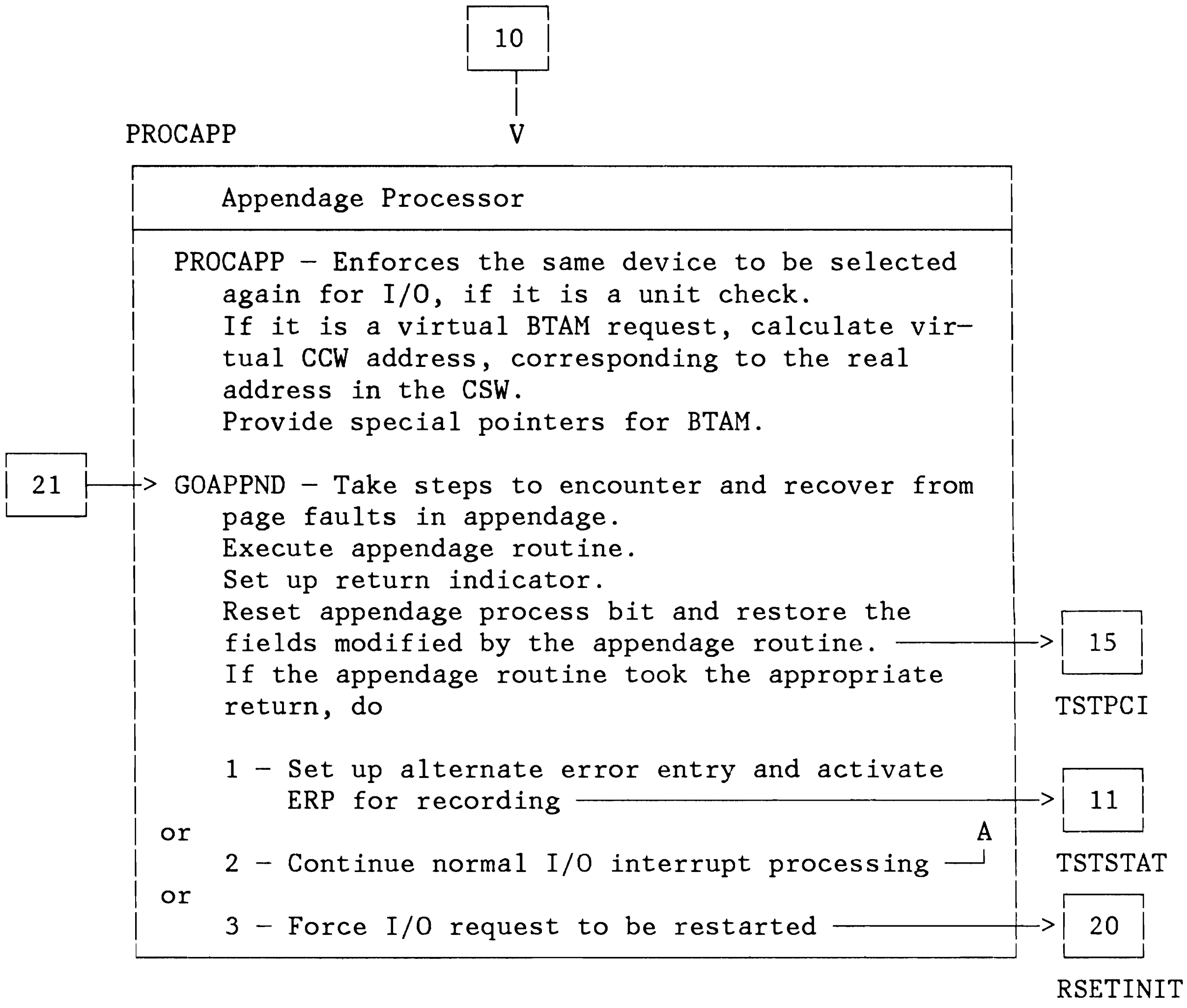
PUREDE40 - Set up a dummy error queue entry, if it is an unsolicited interrupt from a unit record device that requires repositioning

PUREDE60 - Execute POWER hot reader routine, if it is an unsolicited interrupt from a unit record device and POWER is active

NOPUREDE - Reset PUB flags and enforce channel to be restarted if control unit or device was busy

RSETINIT - If it is an unsolicited interrupt, or if the PUB is queued in error, PUB flags and start next I/O operation
If the user has not been posted yet, do POST processing

Figure 136 (Part 6 of 9). I/O Interrupt Handler



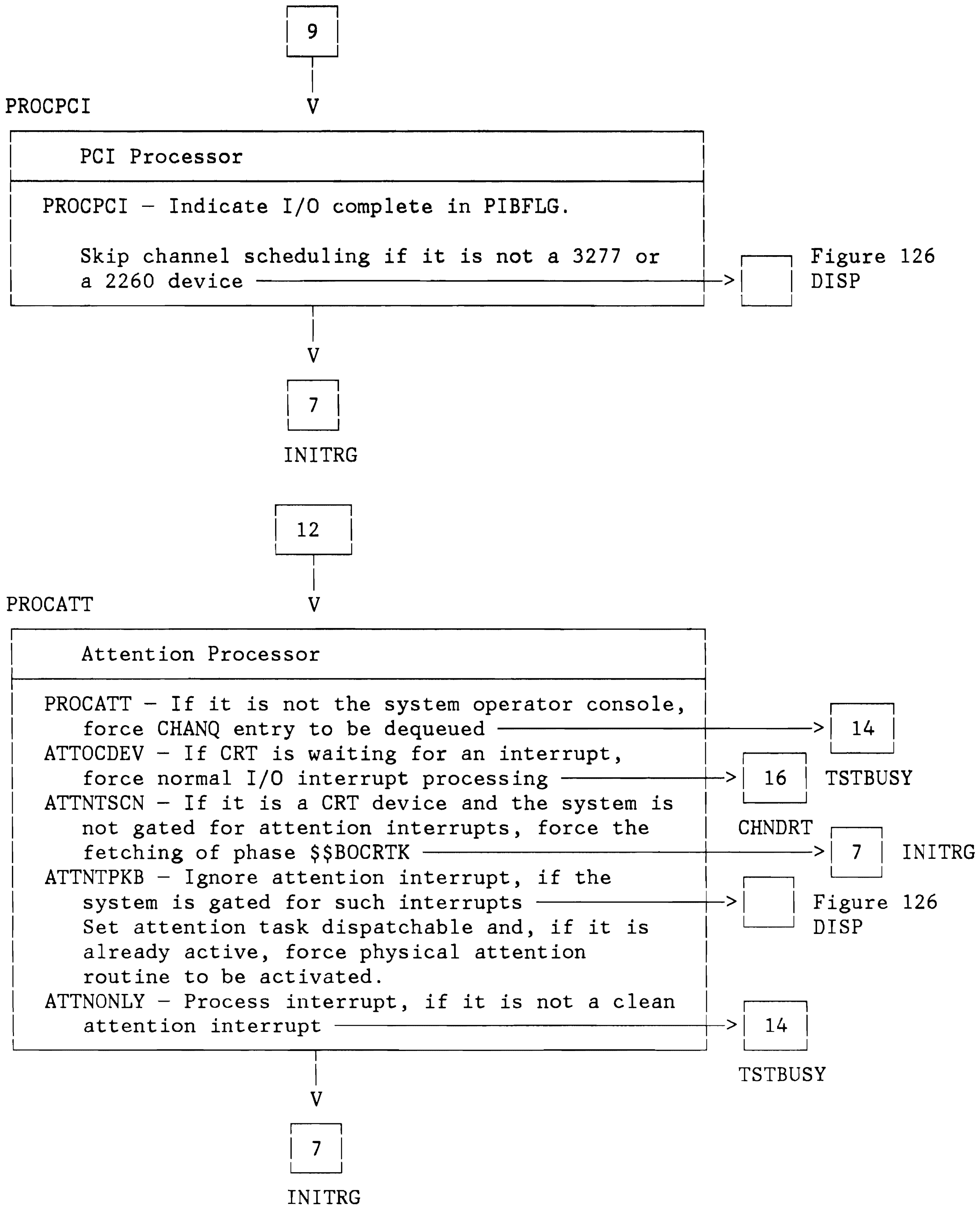
Appendage Processor

PROCAPP - Enforces the same device to be selected again for I/O, if it is a unit check.
If it is a virtual BTAM request, calculate virtual CCW address, corresponding to the real address in the CSW.
Provide special pointers for BTAM.

GOAPPND - Take steps to encounter and recover from page faults in appendage.
Execute appendage routine.
Set up return indicator.
Reset appendage process bit and restore the fields modified by the appendage routine.
If the appendage routine took the appropriate return, do

1 - Set up alternate error entry and activate ERP for recording
or
2 - Continue normal I/O interrupt processing
or
3 - Force I/O request to be restarted

Figure 136 (Part 7 of 9). I/O Interrupt Handler



PCI Processor

PROCPCI - Indicate I/O complete in PIBFLG.

Skip channel scheduling if it is not a 3277 or a 2260 device

Attention Processor

PROCATT - If it is not the system operator console, force CHANQ entry to be dequeued

ATTOCDEV - If CRT is waiting for an interrupt, force normal I/O interrupt processing

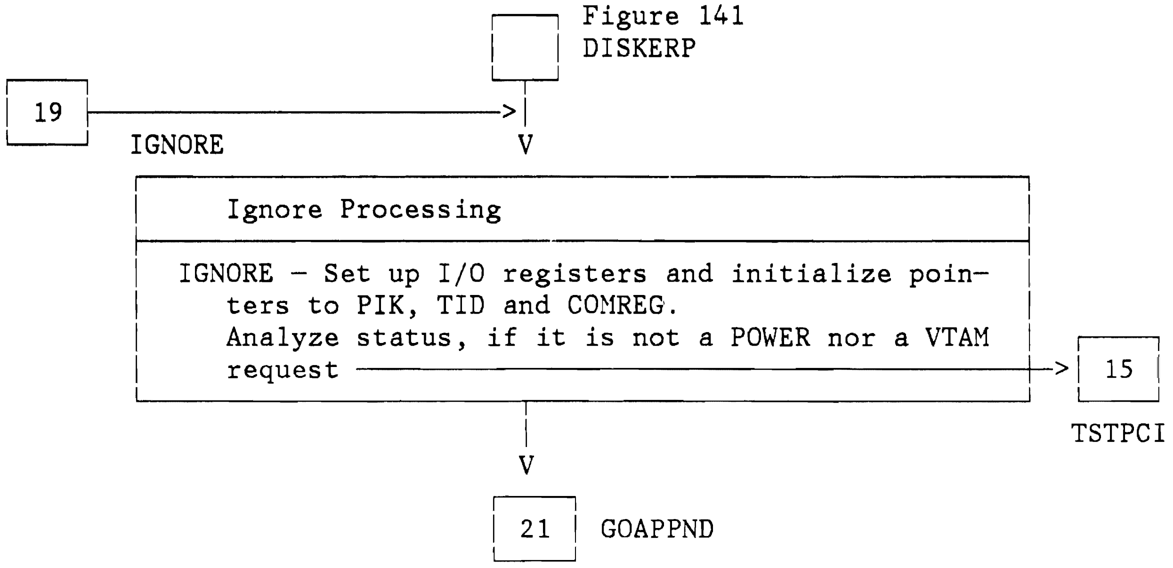
ATTNTSCN - If it is a CRT device and the system is not gated for attention interrupts, force the fetching of phase $$BOCRTK

ATTNTPKB - Ignore attention interrupt, if the system is gated for such interrupts

Set attention task dispatchable and, if it is already active, force physical attention routine to be activated.

ATTNONLY - Process interrupt, if it is not a clean attention interrupt

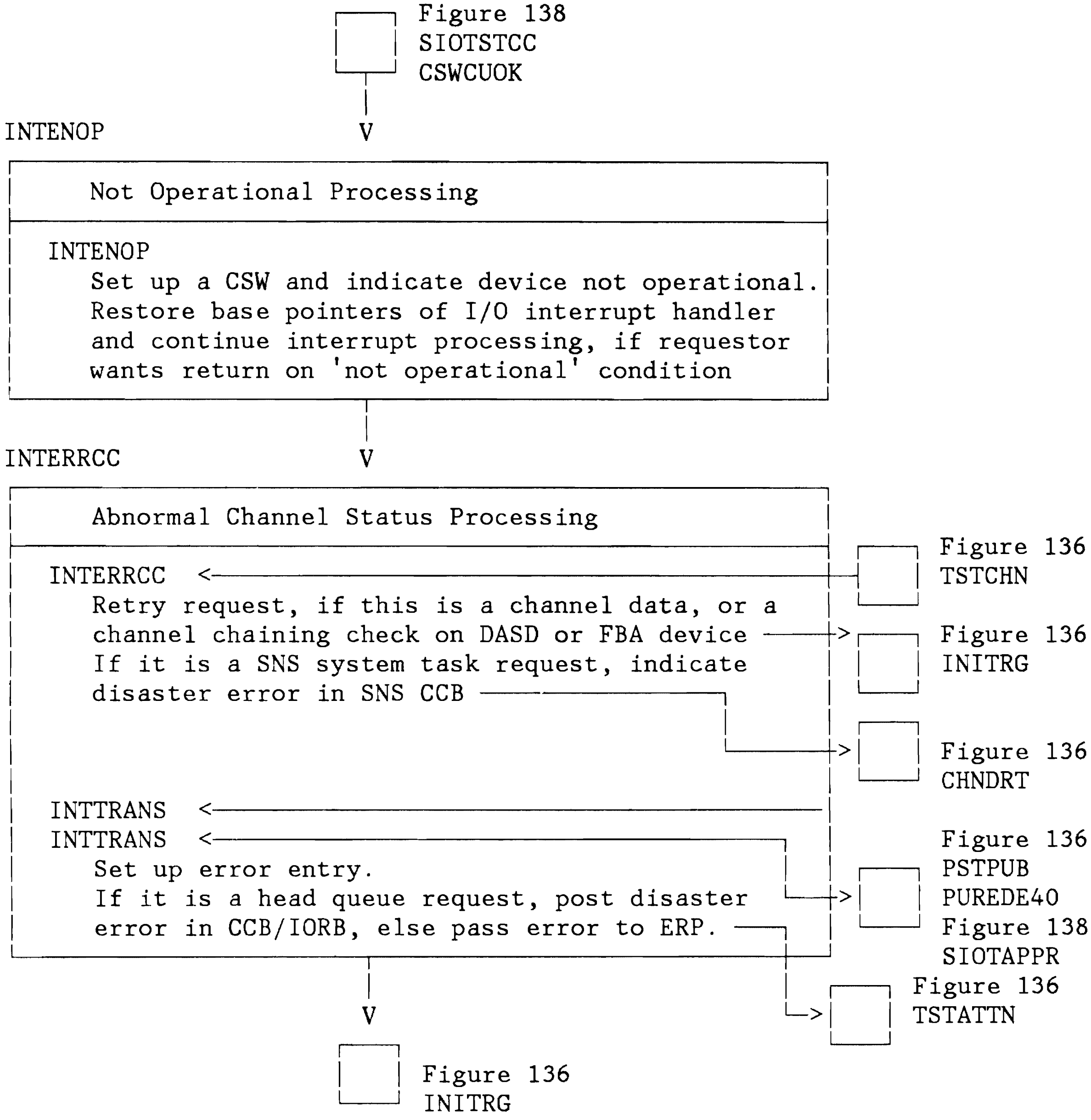
Figure 136 (Part 8 of 9). I/O Interrupt Handler



Ignore Processing

IGNORE - Set up I/O registers and initialize pointers to PIK, TID and COMREG.
Analyze status, if it is not a POWER nor a VTAM request

Figure 136 (Part 9 of 9). I/O Interrupt Handler



Not Operational Processing

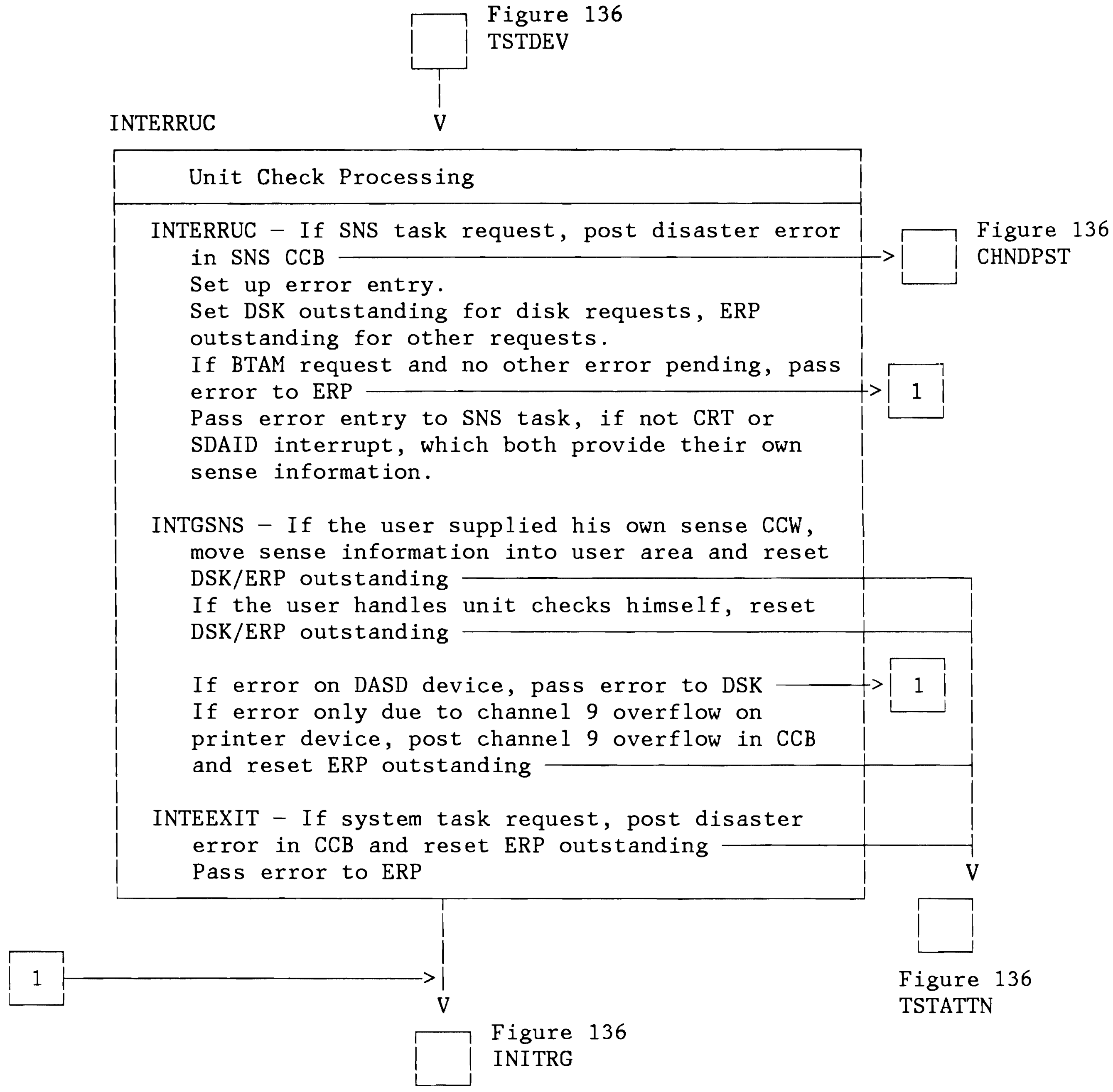
INTENOP
Set up a CSW and indicate device not operational.
Restore base pointers of I/O interrupt handler and continue interrupt processing, if requestor wants return on 'not operational' condition

Abnormal Channel Status Processing

INTERRCC
Retry request, if this is a channel data, or a channel chaining check on DASD or FBA device
If it is a SNS system task request, indicate disaster error in SNS CCB

INTTRANS
INTTRANS
Set up error entry.
If it is a head queue request, post disaster error in CCB/IORB, else pass error to ERP.

Figure 137 (Part 1 of 2). I/O Error Processor



Unit Check Processing

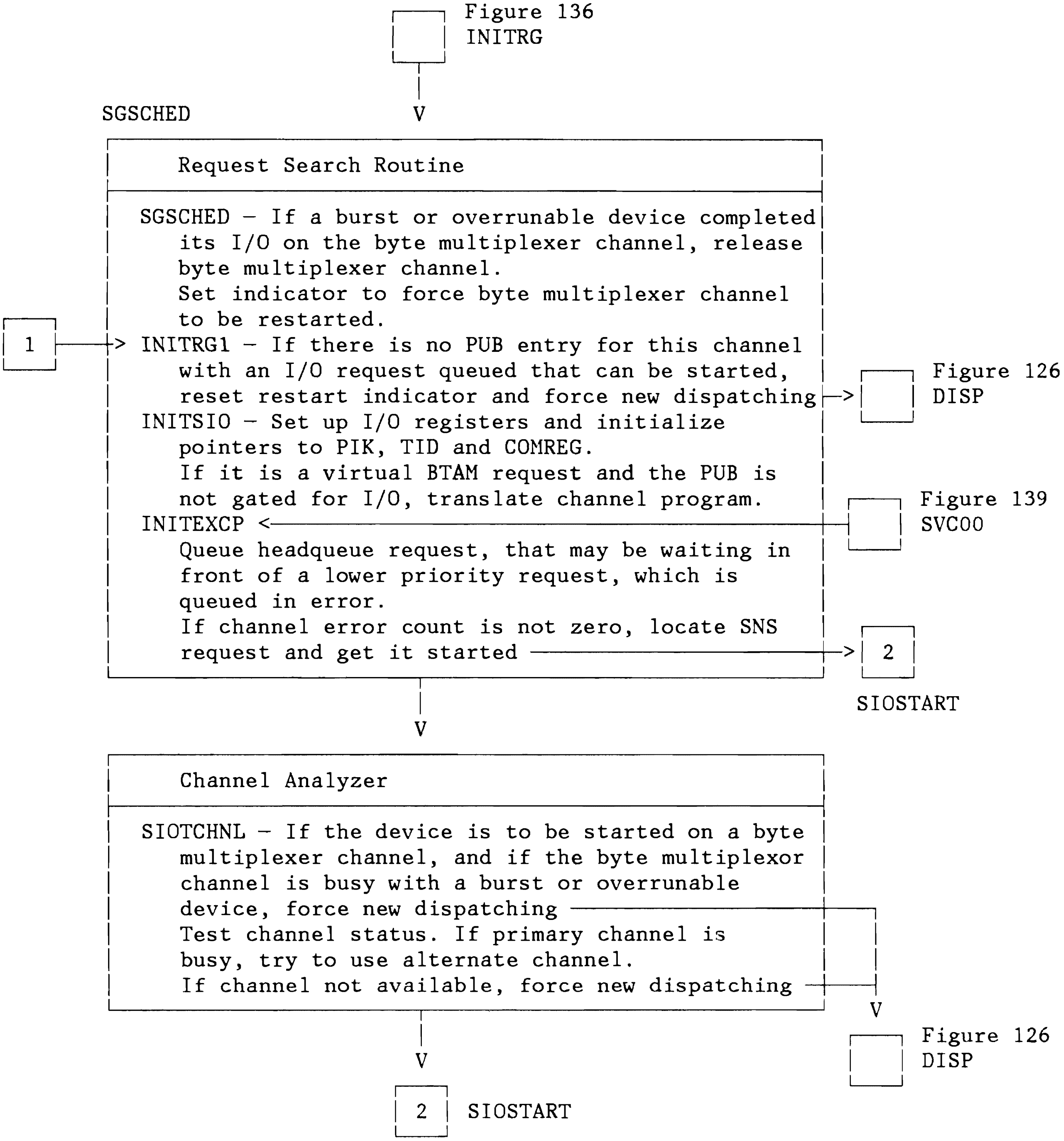
INTERRUC - If SNS task request, post disaster error in SNS CCB
Set up error entry.
Set DSK outstanding for disk requests, ERP outstanding for other requests.
If BTAM request and no other error pending, pass error to ERP
Pass error entry to SNS task, if not CRT or SDAID interrupt, which both provide their own sense information.

INTGSNS - If the user supplied his own sense CCW, move sense information into user area and reset DSK/ERP outstanding
If the user handles unit checks himself, reset DSK/ERP outstanding

If error on DASD device, pass error to DSK
If error only due to channel 9 overflow on printer device, post channel 9 overflow in CCB and reset ERP outstanding

INTEEXIT - If system task request, post disaster error in CCB and reset ERP outstanding
Pass error to ERP

Figure 137 (Part 2 of 2). I/O Error Processor



Request Search Routine

SGSCHED - If a burst or overrunable device completed its I/O on the byte multiplexer channel, release byte multiplexer channel.
Set indicator to force byte multiplexer channel to be restarted.

INITRG1 - If there is no PUB entry for this channel with an I/O request queued that can be started, reset restart indicator and force new dispatching

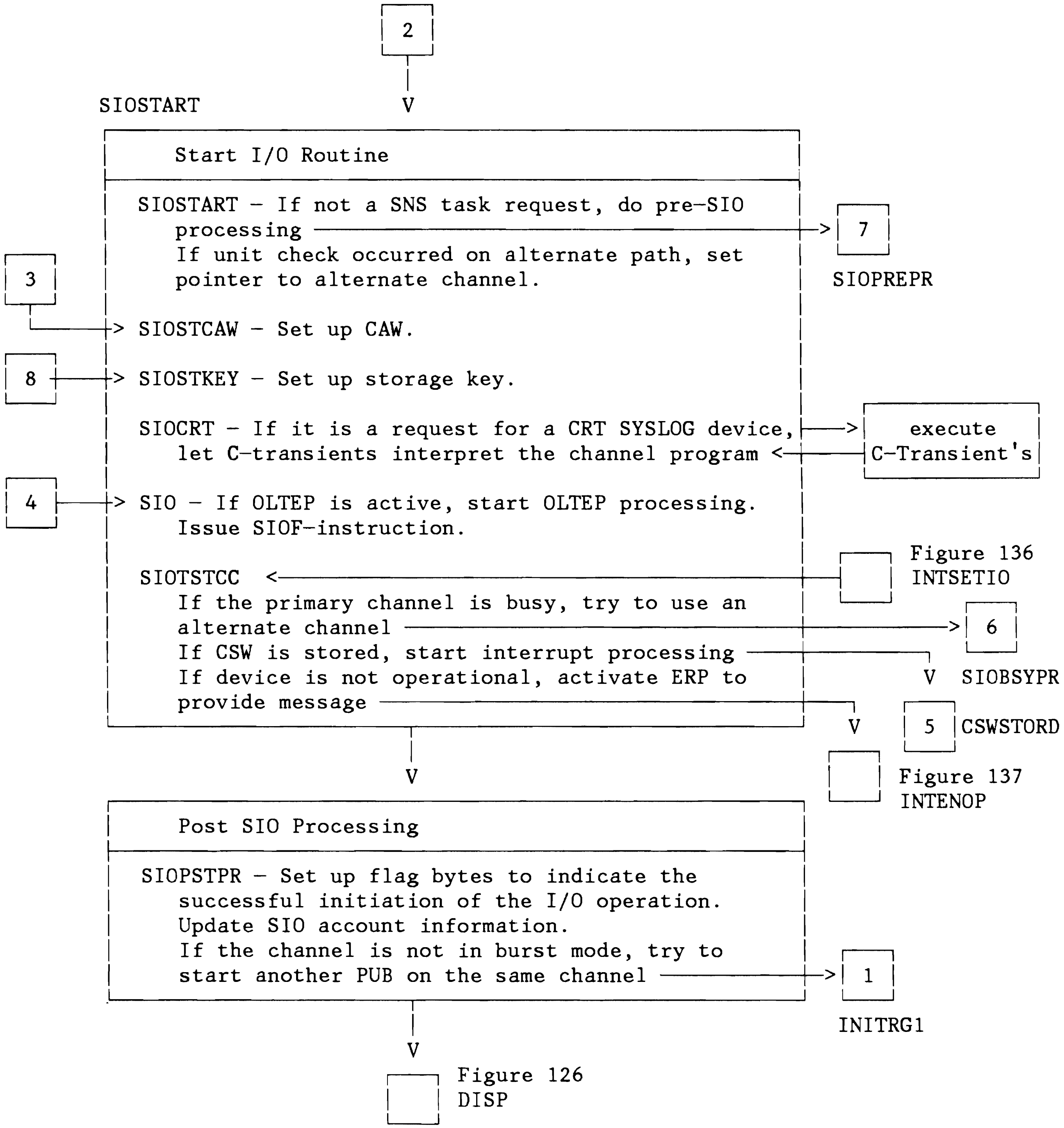
INITSIO - Set up I/O registers and initialize pointers to PIK, TID and COMREG.
If it is a virtual BTAM request and the PUB is not gated for I/O, translate channel program.

INITEXCP
Queue headqueue request, that may be waiting in front of a lower priority request, which is queued in error.
If channel error count is not zero, locate SNS request and get it started

Channel Analyzer

SIOTCHNL - If the device is to be started on a byte multiplexer channel, and if the byte multiplexor channel is busy with a burst or overrunable device, force new dispatching
Test channel status. If primary channel is busy, try to use alternate channel.
If channel not available, force new dispatching

Figure 138 (Part 1 of 5). Channel Scheduler



SIOSTART

Start I/O Routine

SIOSTART - If not a SNS task request, do pre-SIO processing
If unit check occurred on alternate path, set pointer to alternate channel.

SIOSTCAW - Set up CAW.

SIOSTKEY - Set up storage key.

SIOCRT - If it is a request for a CRT SYSLOG device, let C-transients interpret the channel program

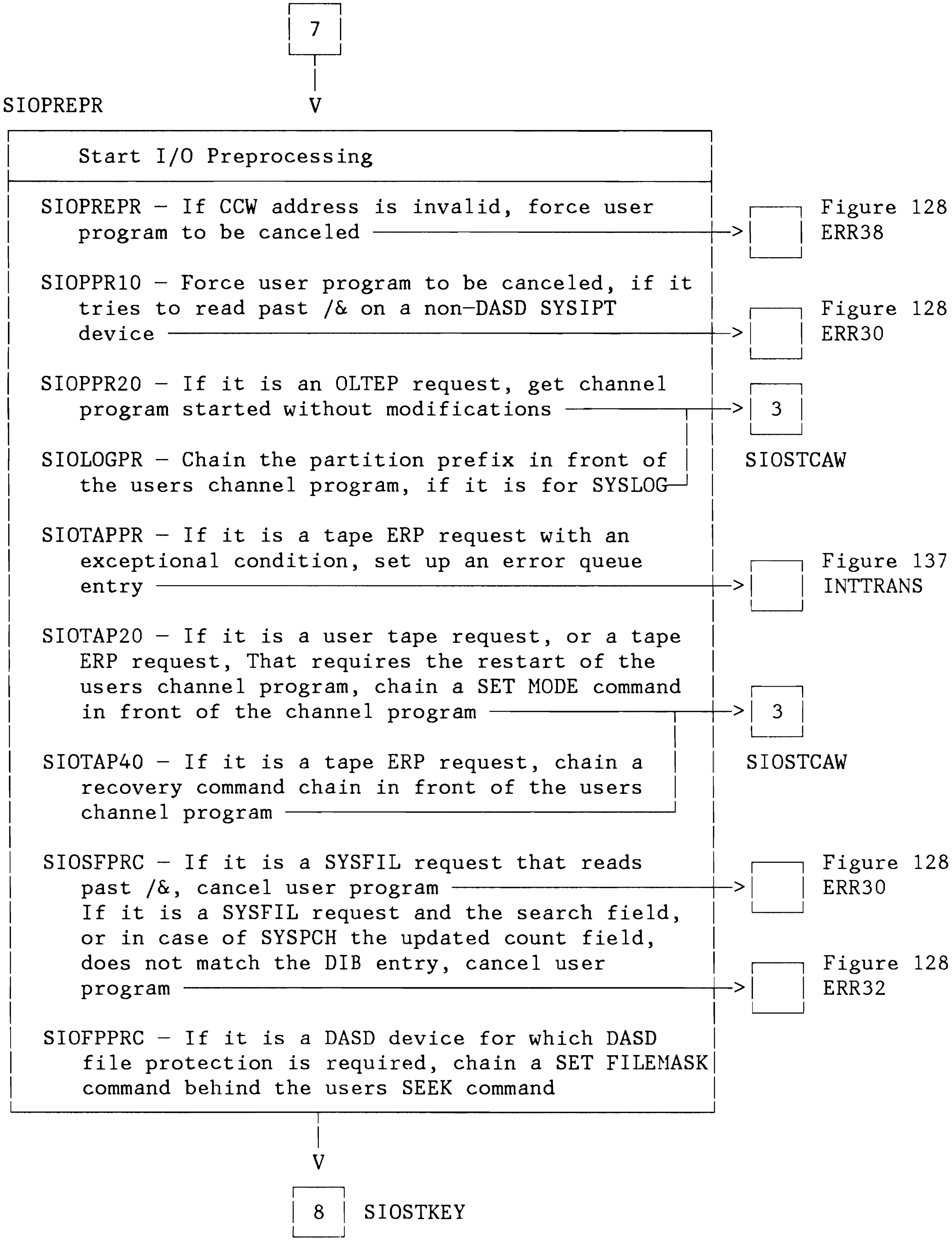
SIO - If OLTEP is active, start OLTEP processing.
Issue SIOF-instruction.

SIOTSTCC
If the primary channel is busy, try to use an alternate channel
If CSW is stored, start interrupt processing
If device is not operational, activate ERP to provide message

Post SIO Processing

SIOPSTPR - Set up flag bytes to indicate the successful initiation of the I/O operation.
Update SIO account information.
If the channel is not in burst mode, try to start another PUB on the same channel

Figure 138 (Part 2 of 5). Channel Scheduler



Start I/O Preprocessing

SIOPREPR - If CCW address is invalid, force user program to be canceled

SIOPPR10 - Force user program to be canceled, if it tries to read past /& on a non-DASD SYSIPT device

SIOPPR20 - If it is an OLTEP request, get channel program started without modifications

SIOLOGPR - Chain the partition prefix in front of the users channel program, if it is for SYSLOG

SIOTAPPR - If it is a tape ERP request with an exceptional condition, set up an error queue entry

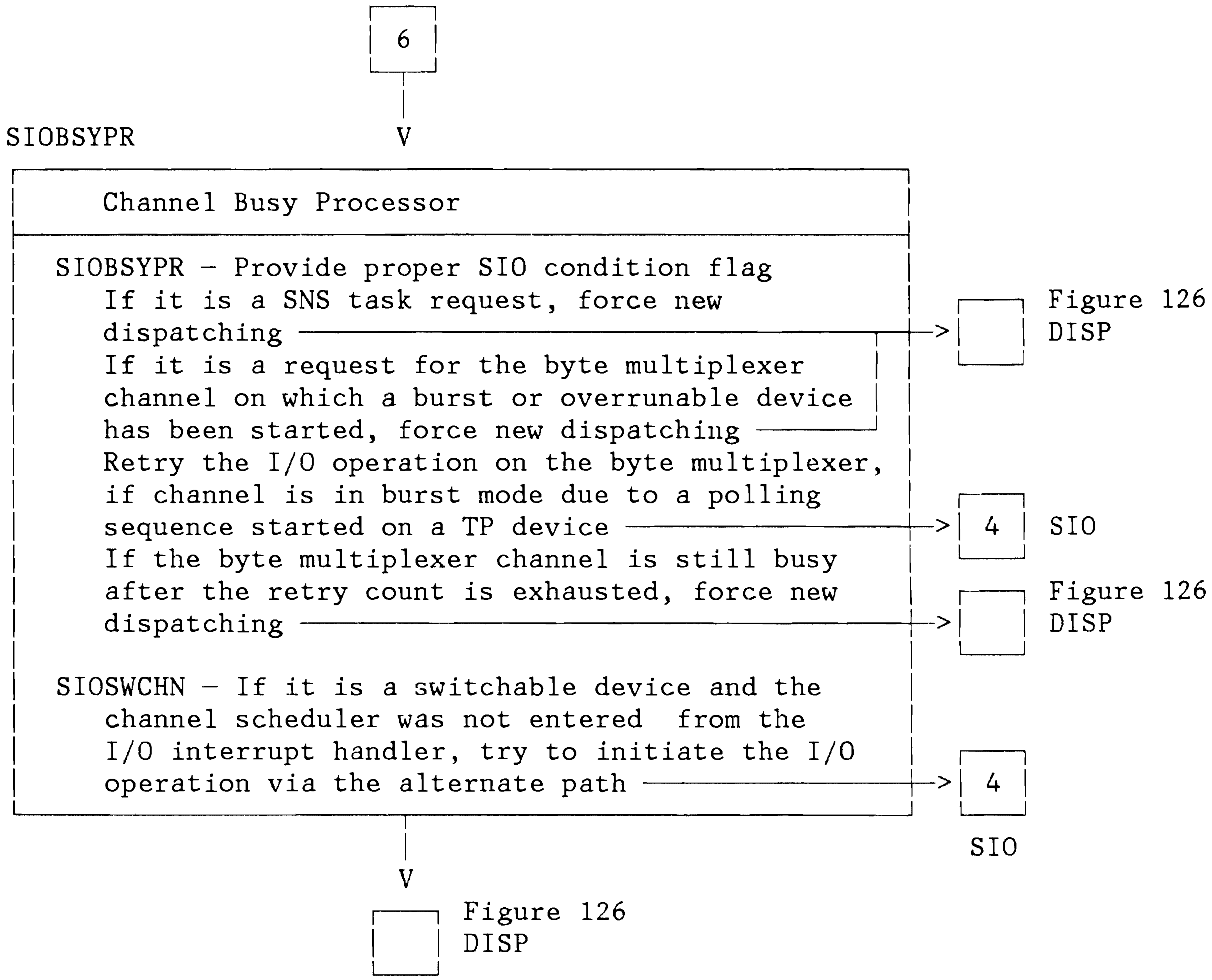
SIOTAP20 - If it is a user tape request, or a tape ERP request, That requires the restart of the users channel program, chain a SET MODE command in front of the channel program

SIOTAP40 - If it is a tape ERP request, chain a recovery command chain in front of the users channel program

SIOSFPRC - If it is a SYSFIL request that reads past /&, cancel user program
If it is a SYSFIL request and the search field, or in case of SYSPCH the updated count field, does not match the DIB entry, cancel user program

SIOFPPRC - If it is a DASD device for which DASD file protection is required, chain a SET FILEMASK command behind the users SEEK command

Figure 138 (Part 3 of 5). Channel Scheduler

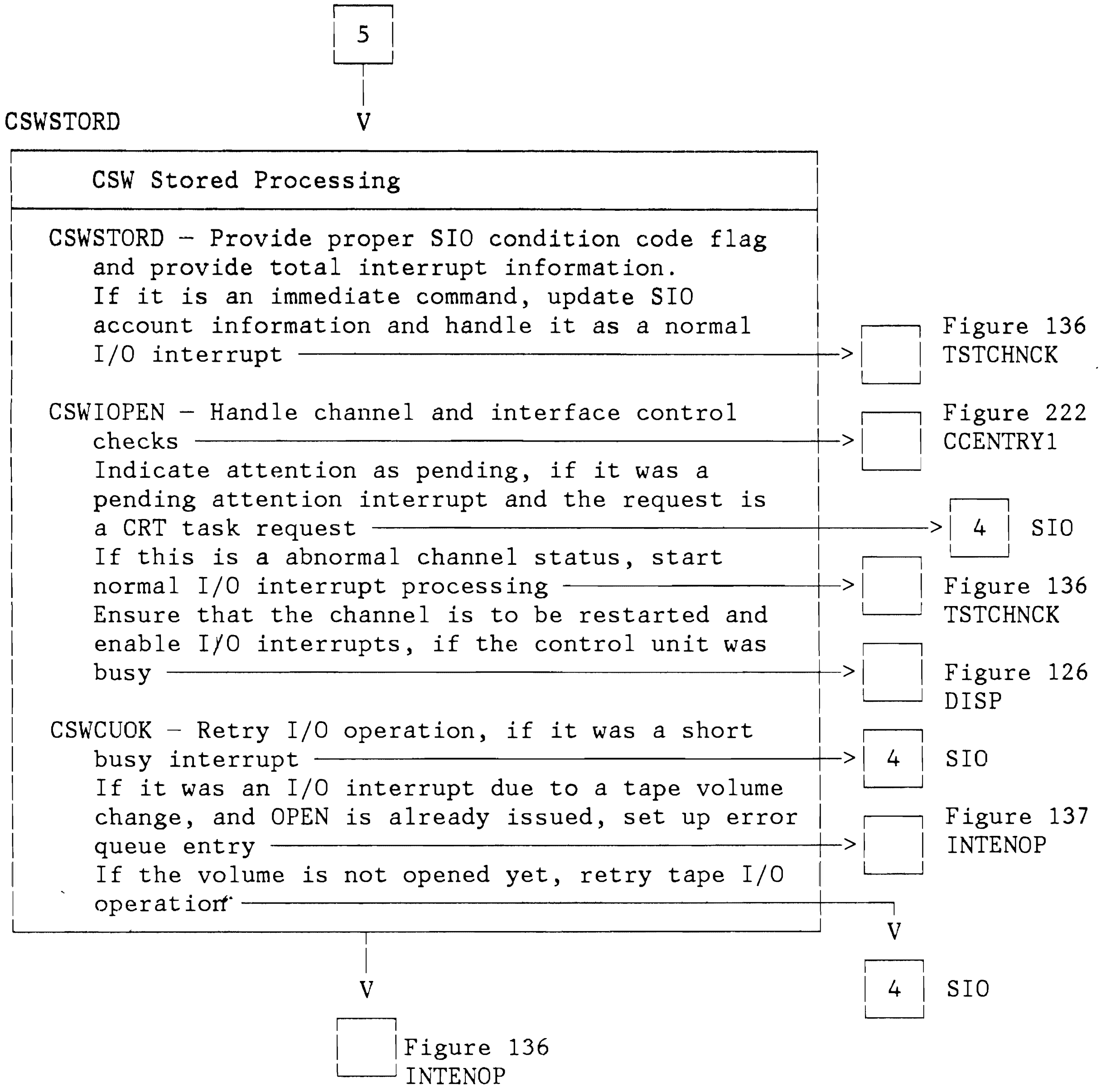


Channel Busy Processor

SIOBSYPR - Provide proper SIO condition flag
If it is a SNS task request, force new dispatching
If it is a request for the byte multiplexer channel on which a burst or overrunable device has been started, force new dispatching
Retry the I/O operation on the byte multiplexer, if channel is in burst mode due to a polling sequence started on a TP device
If the byte multiplexer channel is still busy after the retry count is exhausted, force new dispatching

SIOSWCHN - If it is a switchable device and the channel scheduler was not entered from the I/O interrupt handler, try to initiate the I/O operation via the alternate path

Figure 138 (Part 4 of 5). Channel Scheduler



CSWSTORD - Provide proper SIO condition code flag and provide total interrupt information.
If it is an immediate command, update SIO account information and handle it as a normal I/O interrupt

CSWIOPEN - Handle channel and interface control checks
Indicate attention as pending, if it was a pending attention interrupt and the request is a CRT task request
If this is a abnormal channel status, start normal I/O interrupt processing
Ensure that the channel is to be restarted and enable I/O interrupts, if the control unit was busy

CSWCUOK - Retry I/O operation, if it was a short busy interrupt
If it was an I/O interrupt due to a tape volume change, and OPEN is already issued, set up error queue entry
If the volume is not opened yet, retry tape I/O operation

Figure 138 (Part 5 of 5). Channel Scheduler

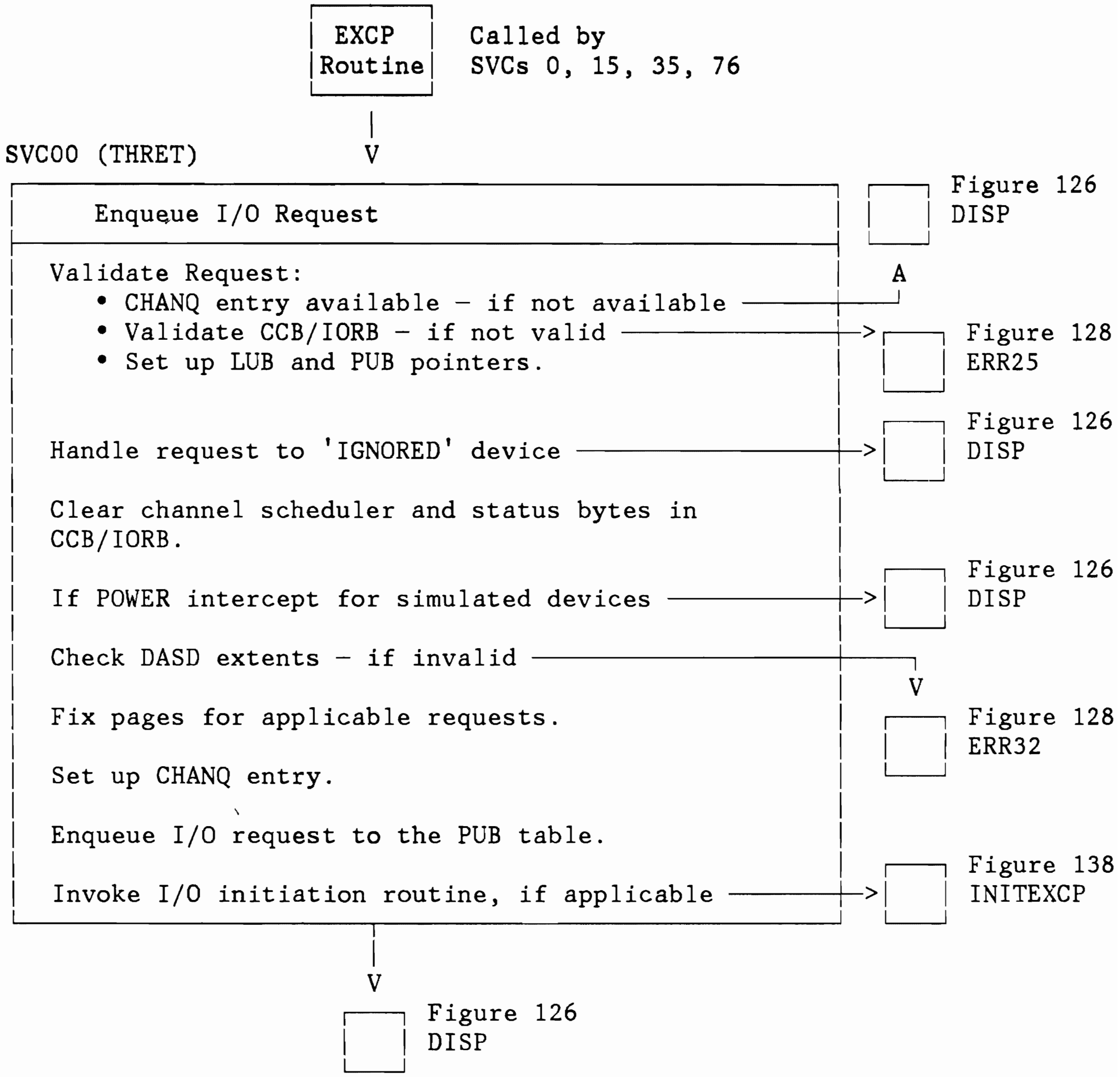
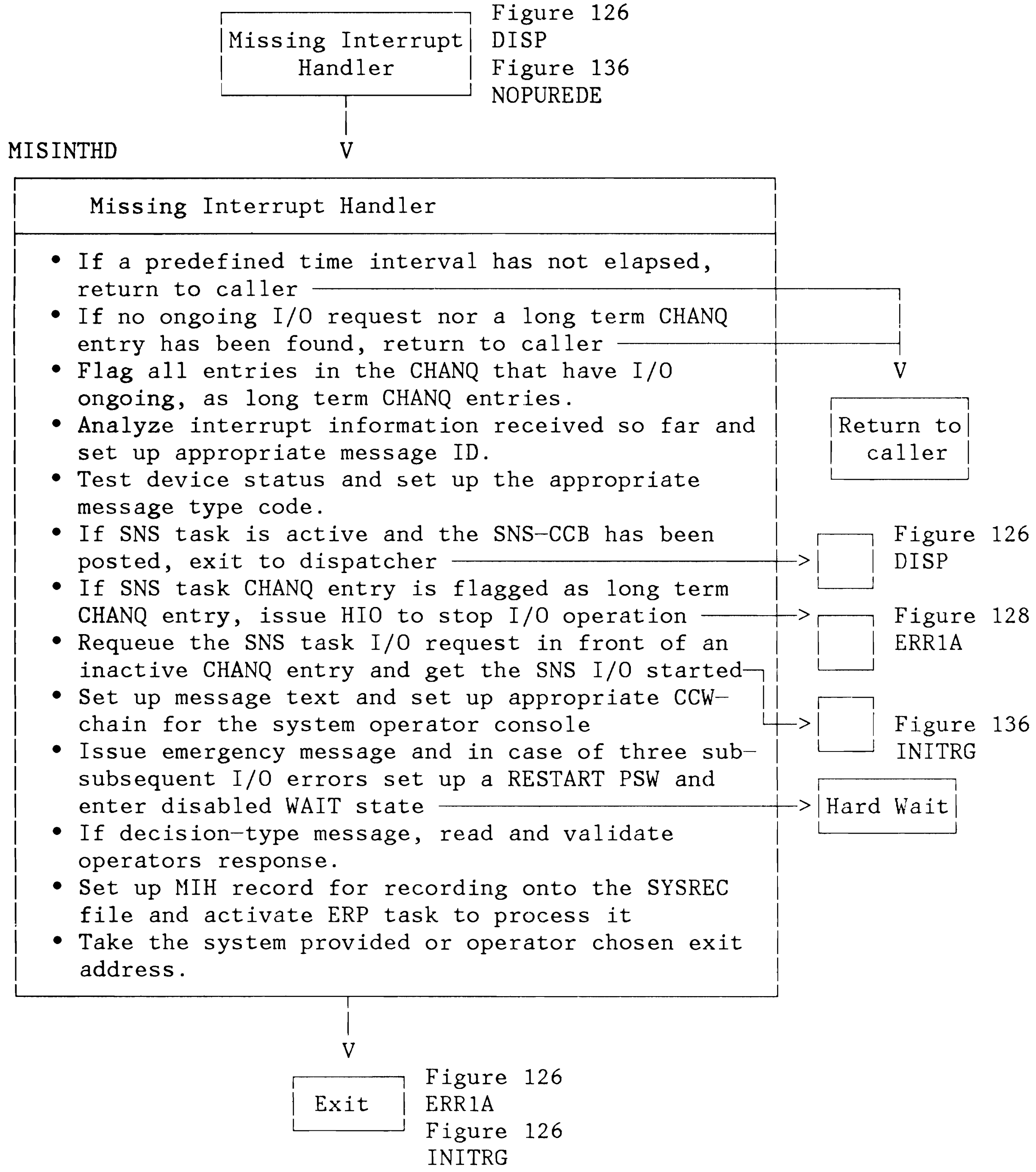

EXCP Routine
Called by SVCs 0, 15, 35, 76
SVC00 (THRET)
Enqueue I/O Request
Validate Request:
• CHANQ entry available - if not available
• Validate CCB/IORB - if not valid
• Set up LUB and PUB pointers.
Handle request to 'IGNORED' device
Clear channel scheduler and status bytes in CCB/IORB.
If POWER intercept for simulated devices
Check DASD extents - if invalid
Fix pages for applicable requests.
Set up CHANQ entry.
Enqueue I/O request to the PUB table.
Invoke I/O initiation routine, if applicable
A
Figure 126 DISP
Figure 128 ERR25
Figure 126 DISP
Figure 126 DISP
Figure 128 ERR32
Figure 138 INITEXCP
Figure 126 DISP

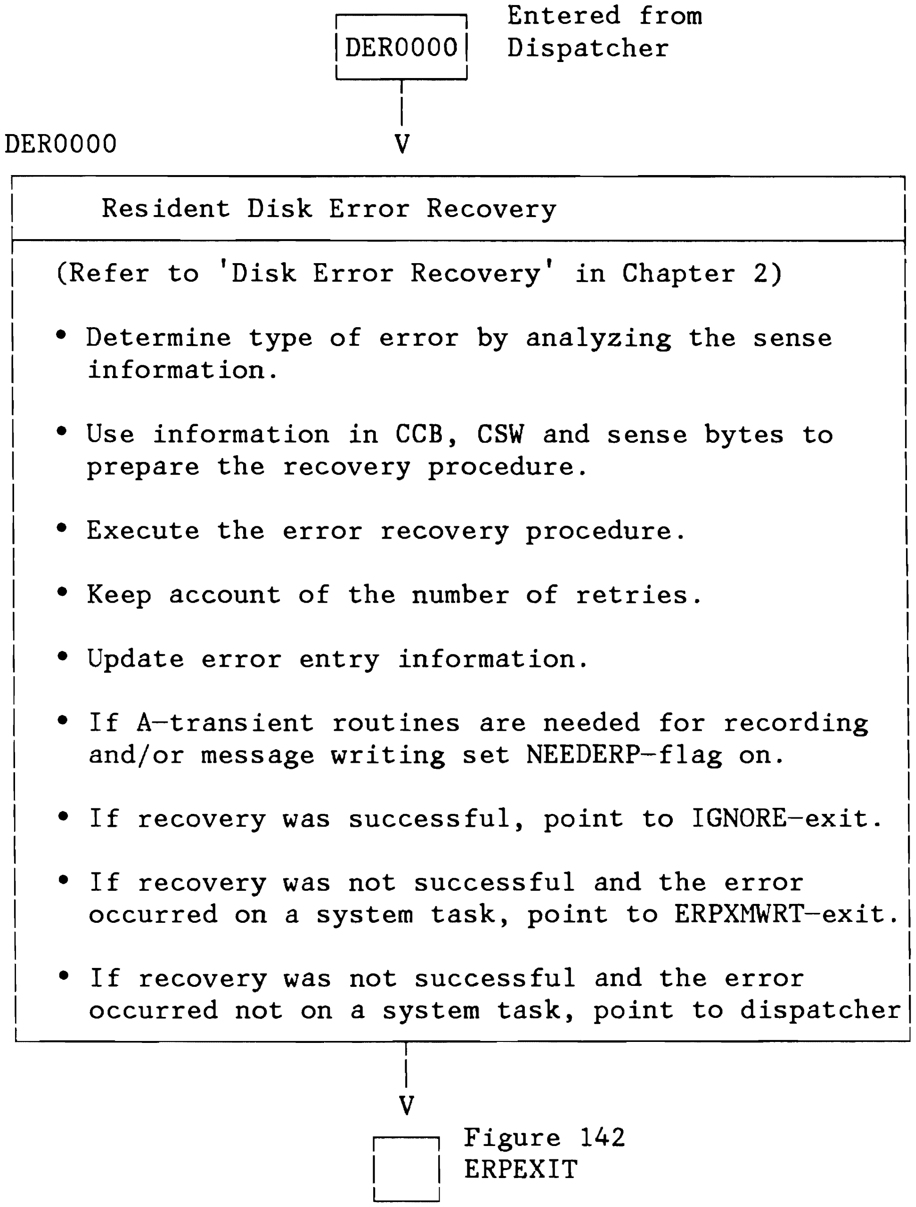
Figure 139. I/O Request Enqueuer, EXCP Routine

Missing Interrupt Handler



- If a predefined time interval has not elapsed, return to caller
- If no ongoing I/O request nor a long term CHANQ entry has been found, return to caller
- Flag all entries in the CHANQ that have I/O ongoing, as long term CHANQ entries.
- Analyze interrupt information received so far and set up appropriate message ID.
- Test device status and set up the appropriate message type code.
- If SNS task is active and the SNS–CCB has been posted, exit to dispatcher
- If SNS task CHANQ entry is flagged as long term CHANQ entry, issue HIO to stop I/O operation
- Requeue the SNS task I/O request in front of an inactive CHANQ entry and get the SNS I/O started
- Set up message text and set up appropriate CCW–chain for the system operator console
- Issue emergency message and in case of three sub–subsequent I/O errors set up a RESTART PSW and enter disabled WAIT state
- If decision–type message, read and validate operators response.
- Set up MIH record for recording onto the SYSREC file and activate ERP task to process it
- Take the system provided or operator chosen exit address.

Figure 140. Missing Interrupt Handler



DER0000

Entered from Dispatcher

DER0000

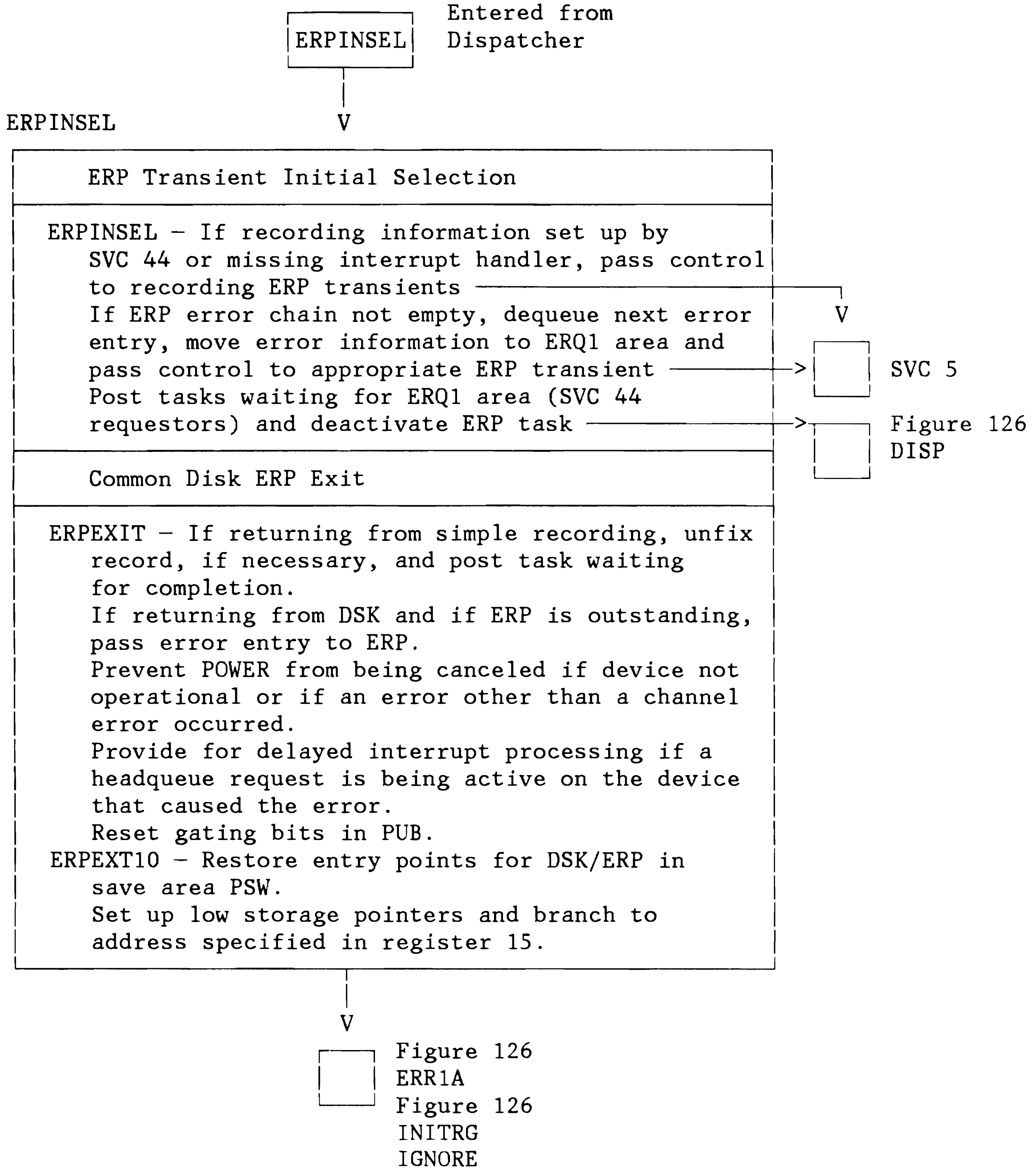
Resident Disk Error Recovery

(Refer to 'Disk Error Recovery' in Chapter 2)

- Determine type of error by analyzing the sense information.
- Use information in CCB, CSW and sense bytes to prepare the recovery procedure.
- Execute the error recovery procedure.
- Keep account of the number of retries.
- Update error entry information.
- If A-transient routines are needed for recording and/or message writing set NEEDERP-flag on.
- If recovery was successful, point to IGNORE-exit.
- If recovery was not successful and the error occurred on a system task, point to ERPXMWRT-exit.
- If recovery was not successful and the error occurred not on a system task, point to dispatcher

Figure 142
ERPEXIT

Figure 141. Disk Error Recovery



ERPINSEL

Entered from Dispatcher

ERPINSEL

ERP Transient Initial Selection

ERPINSEL - If recording information set up by SVC 44 or missing interrupt handler, pass control to recording ERP transients

If ERP error chain not empty, dequeue next error entry, move error information to ERQ1 area and pass control to appropriate ERP transient

Post tasks waiting for ERQ1 area (SVC 44 requestors) and deactivate ERP task

SVC 5

Figure 126 DISP

Common Disk ERP Exit

ERPEXIT - If returning from simple recording, unfix record, if necessary, and post task waiting for completion.

If returning from DSK and if ERP is outstanding, pass error entry to ERP.

Prevent POWER from being canceled if device not operational or if an error other than a channel error occurred.

Provide for delayed interrupt processing if a headqueue request is being active on the device that caused the error.

Reset gating bits in PUB.

ERPEXT10 - Restore entry points for DSK/ERP in save area PSW.

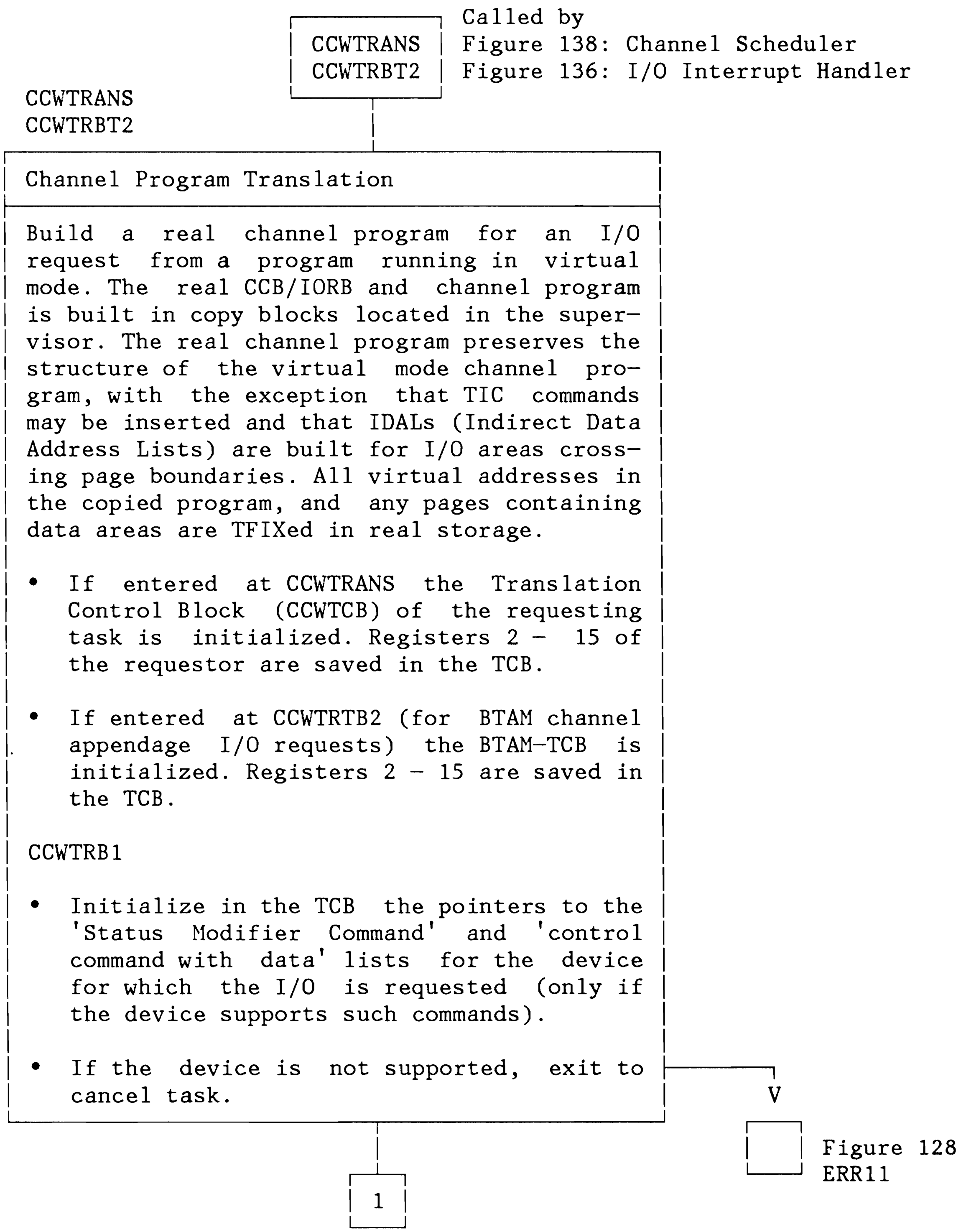
Set up low storage pointers and branch to address specified in register 15.

Figure 126 ERR1A

Figure 126 INITRG IGNORE

Figure 142. ERP Transient Initial Selection

CHANNEL PROGRAM TRANSLATION ROUTINES



CCWTRANS
CCWTRBT2

Called by
Figure 138: Channel Scheduler
Figure 136: I/O Interrupt Handler

CCWTRANS
CCWTRBT2

Channel Program Translation

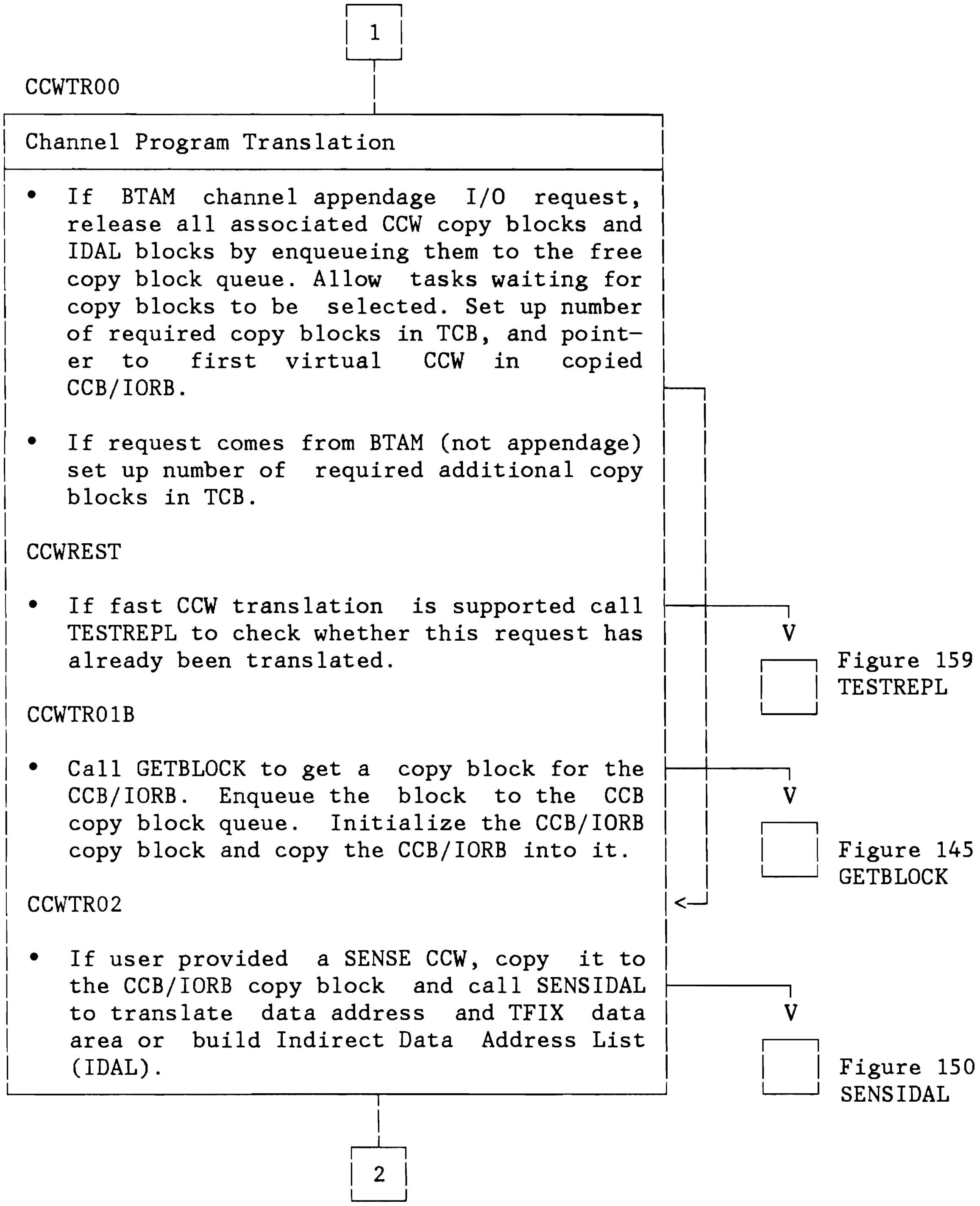
Build a real channel program for an I/O request from a program running in virtual mode. The real CCB/IORB and channel program is built in copy blocks located in the supervisor. The real channel program preserves the structure of the virtual mode channel program, with the exception that TIC commands may be inserted and that IDALs (Indirect Data Address Lists) are built for I/O areas crossing page boundaries. All virtual addresses in the copied program, and any pages containing data areas are TFIXed in real storage.

- If entered at CCWTRANS the Translation Control Block (CCWTCB) of the requesting task is initialized. Registers 2 - 15 of the requestor are saved in the TCB.
- If entered at CCWTRTB2 (for BTAM channel appendage I/O requests) the BTAM-TCB is initialized. Registers 2 - 15 are saved in the TCB.

CCWTRB1

- Initialize in the TCB the pointers to the 'Status Modifier Command' and 'control command with data' lists for the device for which the I/O is requested (only if the device supports such commands).
- If the device is not supported, exit to cancel task.

Figure 143 (Part 1 of 5). Channel Program Translation: General Routine



1

CCWTR00

Channel Program Translation

- If BTAM channel appendage I/O request, release all associated CCW copy blocks and IDAL blocks by enqueueing them to the free copy block queue. Allow tasks waiting for copy blocks to be selected. Set up number of required copy blocks in TCB, and pointer to first virtual CCW in copied CCB/IORB.

- If request comes from BTAM (not appendage) set up number of required additional copy blocks in TCB.

CCWREST

- If fast CCW translation is supported call TESTREPL to check whether this request has already been translated.

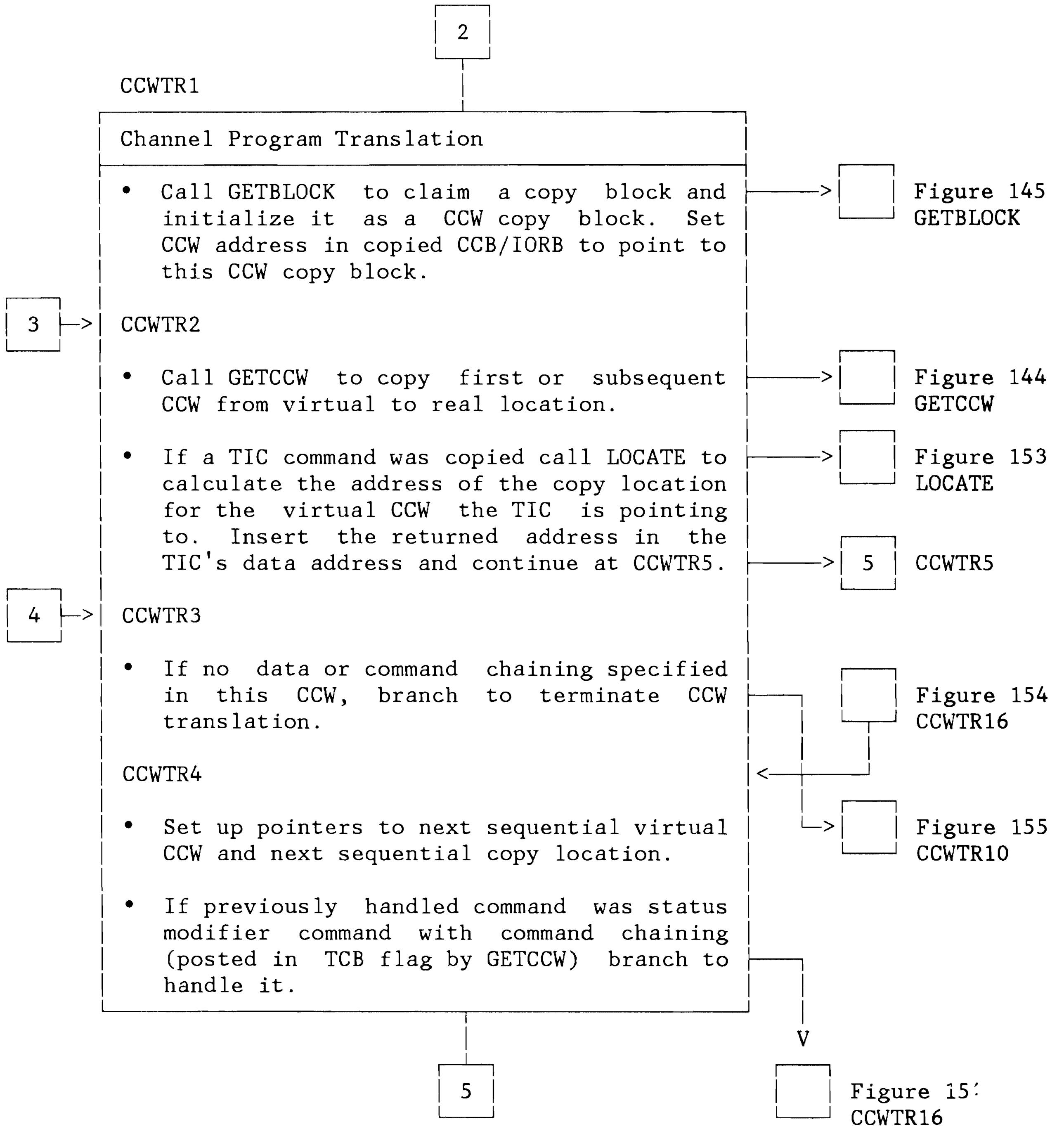
CCWTR01B

- Call GETBLOCK to get a copy block for the CCB/IORB. Enqueue the block to the CCB copy block queue. Initialize the CCB/IORB copy block and copy the CCB/IORB into it.

CCWTR02

- If user provided a SENSE CCW, copy it to the CCB/IORB copy block and call SENSIDAL to translate data address and TFIX data area or build Indirect Data Address List (IDAL).

Figure 143 (Part 2 of 5). Channel Program Translation: General Routine



Channel Program Translation

- Call GETBLOCK to claim a copy block and initialize it as a CCW copy block. Set CCW address in copied CCB/IORB to point to this CCW copy block.

CCWTR2

- Call GETCCW to copy first or subsequent CCW from virtual to real location.
- If a TIC command was copied call LOCATE to calculate the address of the copy location for the virtual CCW the TIC is pointing to. Insert the returned address in the TIC's data address and continue at CCWTR5.

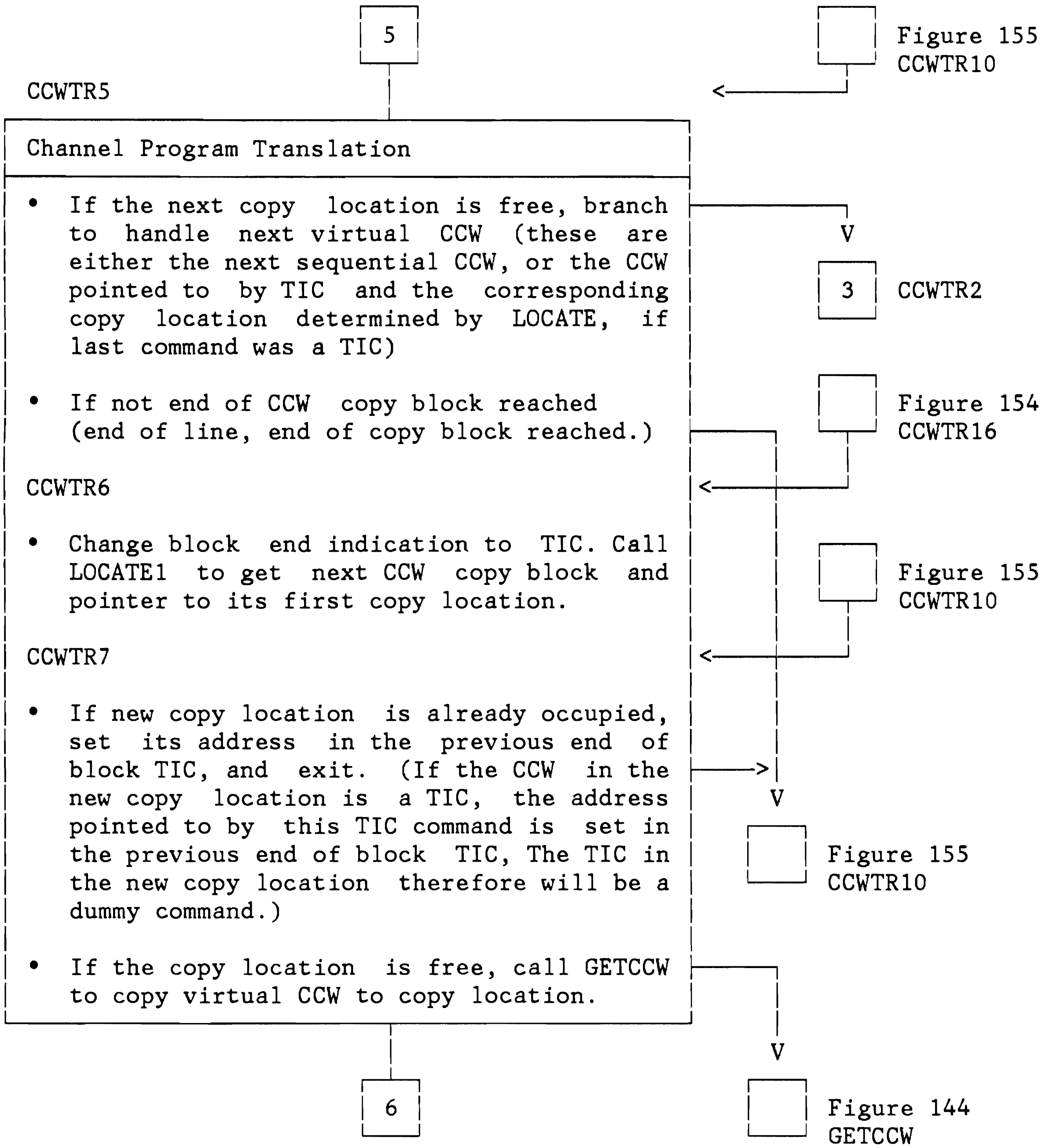
CCWTR3

- If no data or command chaining specified in this CCW, branch to terminate CCW translation.

CCWTR4

- Set up pointers to next sequential virtual CCW and next sequential copy location.
- If previously handled command was status modifier command with command chaining (posted in TCB flag by GETCCW) branch to handle it.

Figure 143 (Part 3 of 5). Channel Program Translation: General Routine



5

Figure 155
CCWTR10

CCWTR5

Channel Program Translation

- If the next copy location is free, branch to handle next virtual CCW (these are either the next sequential CCW, or the CCW pointed to by TIC and the corresponding copy location determined by LOCATE, if last command was a TIC)

3 CCWTR2

- If not end of CCW copy block reached (end of line, end of copy block reached.)

Figure 154
CCWTR16

CCWTR6

- Change block end indication to TIC. Call LOCATE1 to get next CCW copy block and pointer to its first copy location.

Figure 155
CCWTR10

CCWTR7

- If new copy location is already occupied, set its address in the previous end of block TIC, and exit. (If the CCW in the new copy location is a TIC, the address pointed to by this TIC command is set in the previous end of block TIC, The TIC in the new copy location therefore will be a dummy command.)

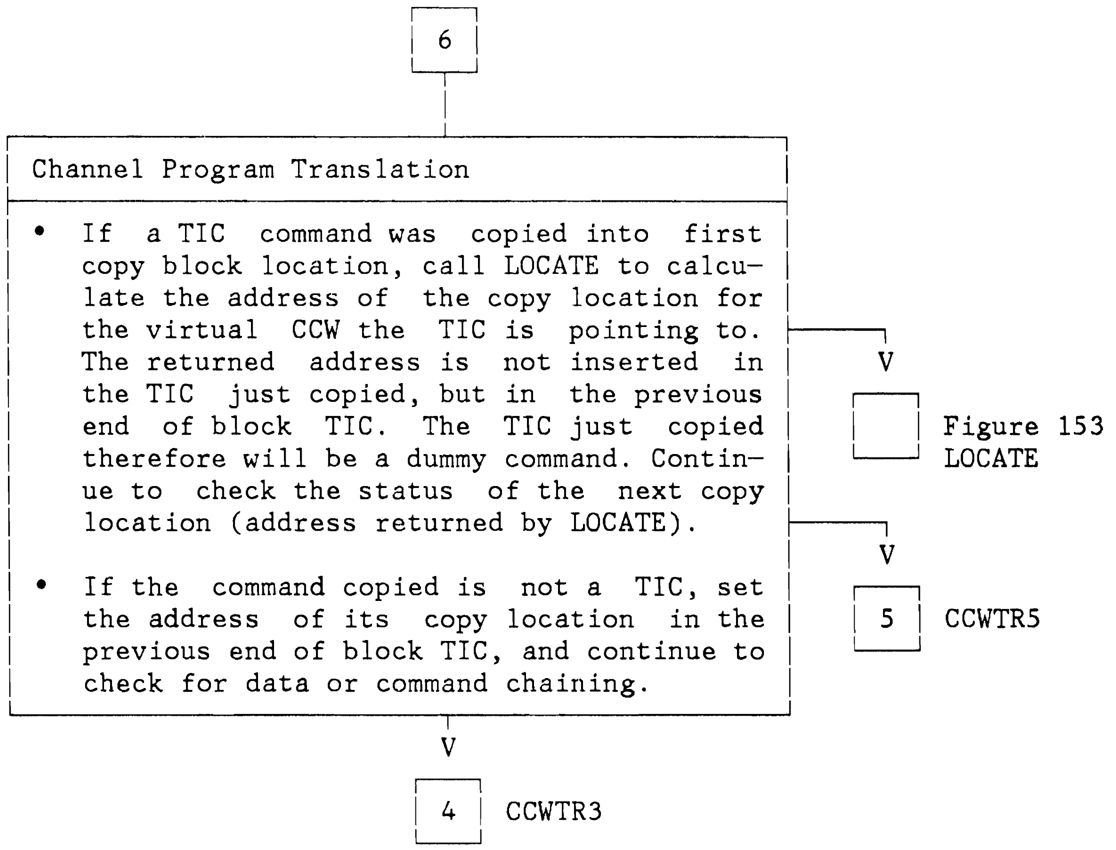
Figure 155
CCWTR10

- If the copy location is free, call GETCCW to copy virtual CCW to copy location.

6

Figure 144
GETCCW

Figure 143 (Part 4 of 5). Channel Program Translation: General Routine

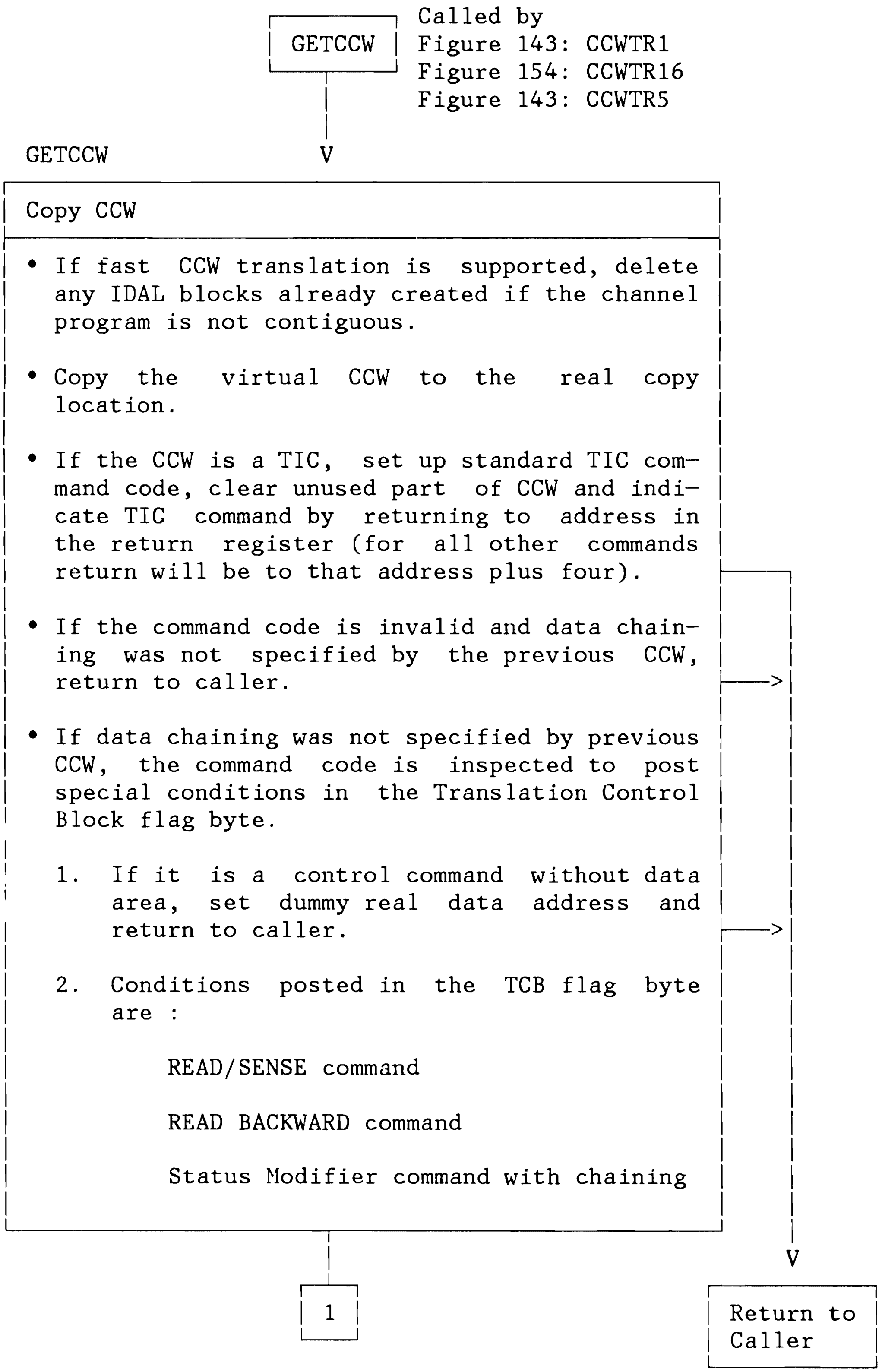


Channel Program Translation

- If a TIC command was copied into first copy block location, call LOCATE to calculate the address of the copy location for the virtual CCW the TIC is pointing to. The returned address is not inserted in the TIC just copied, but in the previous end of block TIC. The TIC just copied therefore will be a dummy command. Continue to check the status of the next copy location (address returned by LOCATE).

- If the command copied is not a TIC, set the address of its copy location in the previous end of block TIC, and continue to check for data or command chaining.

Figure 143 (Part 5 of 5). Channel Program Translation: General Routine



Copy CCW

- If fast CCW translation is supported, delete any IDAL blocks already created if the channel program is not contiguous.
- Copy the virtual CCW to the real copy location.
- If the CCW is a TIC, set up standard TIC command code, clear unused part of CCW and indicate TIC command by returning to address in the return register (for all other commands return will be to that address plus four).
- If the command code is invalid and data chaining was not specified by the previous CCW, return to caller.
- If data chaining was not specified by previous CCW, the command code is inspected to post special conditions in the Translation Control Block flag byte.

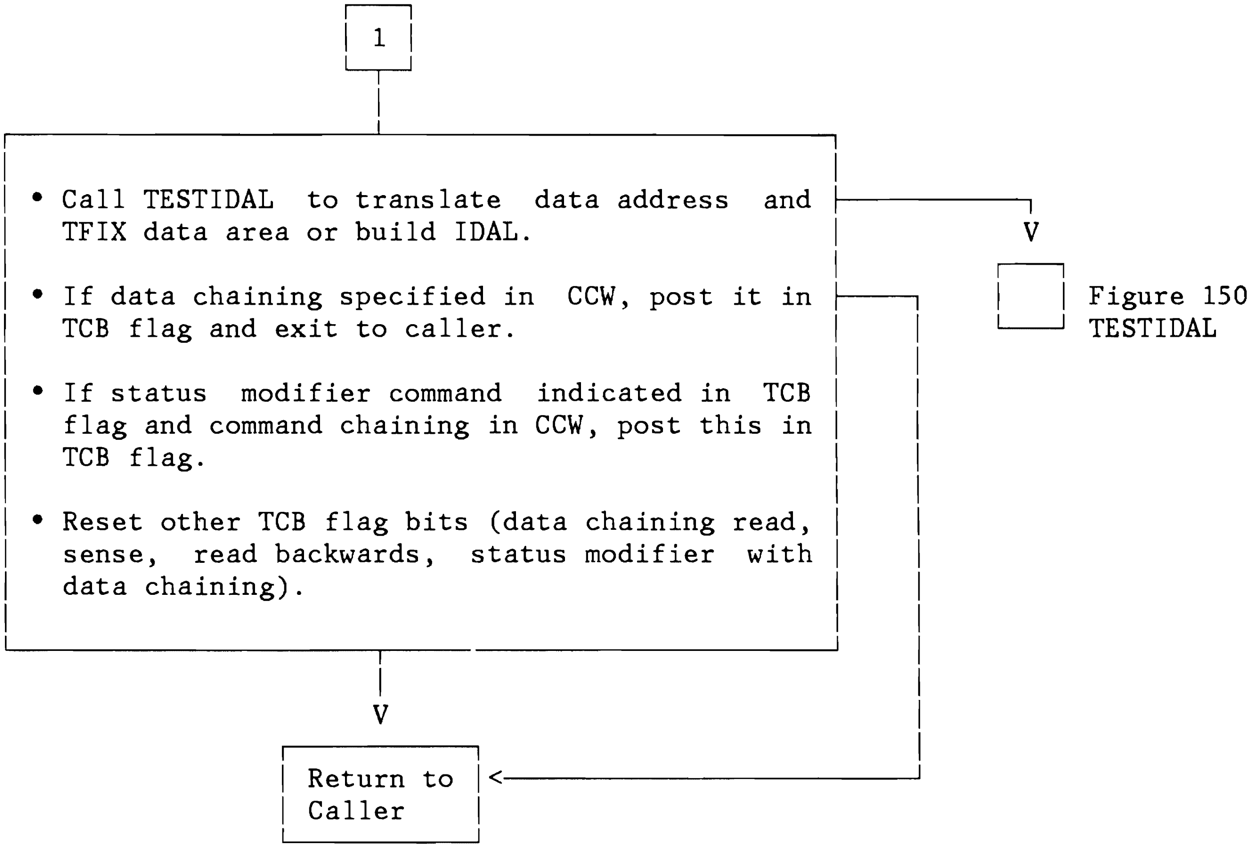
  1. If it is a control command without data area, set dummy real data address and return to caller.
  2. Conditions posted in the TCB flag byte are :

     READ/SENSE command

     READ BACKWARD command

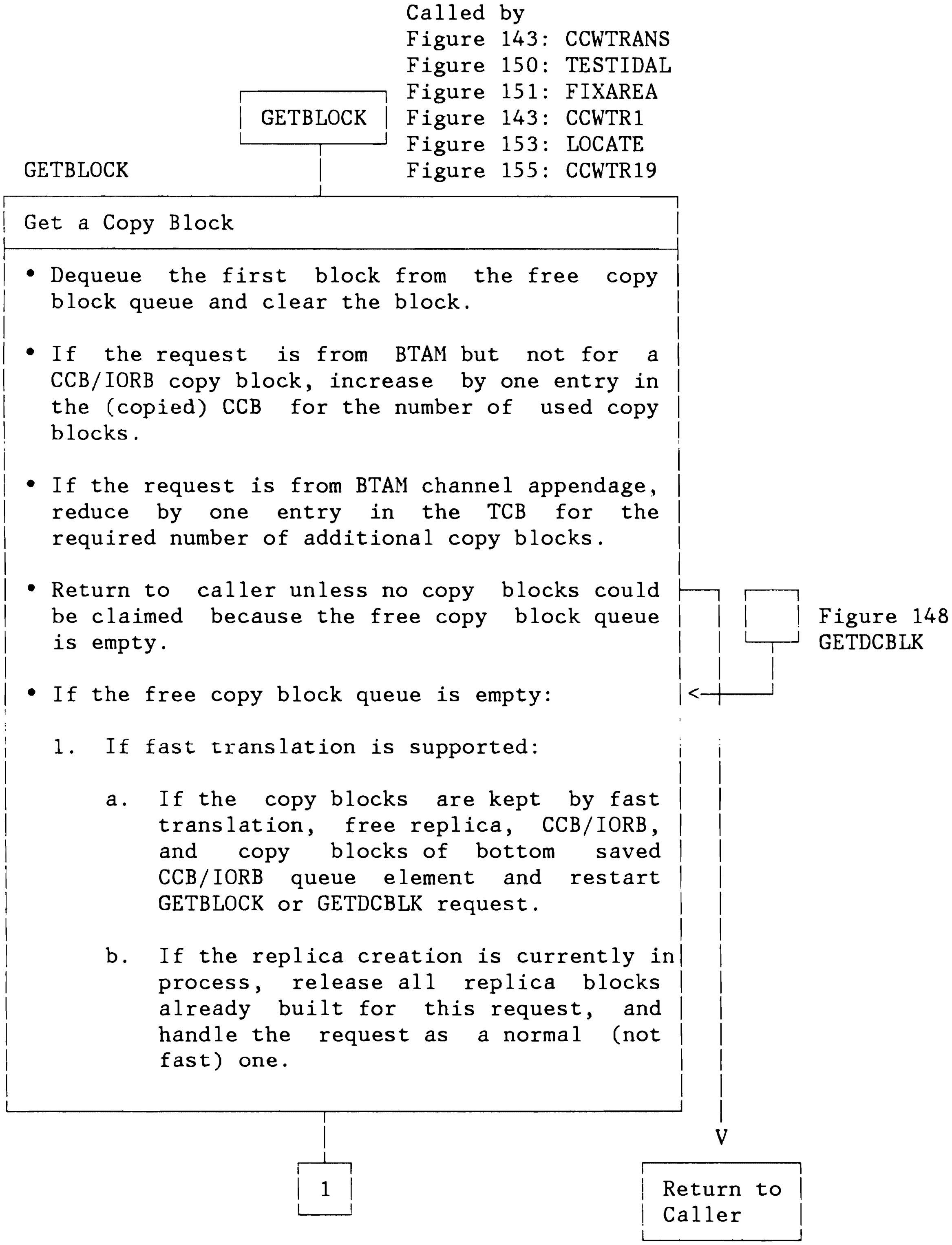
     Status Modifier command with chaining

Figure 144 (Part 1 of 2). Channel Program Translation (370 Mode): Subroutine GETCCW



- Call TESTIDAL to translate data address and TFIX data area or build IDAL.
- If data chaining specified in CCW, post it in TCB flag and exit to caller.
- If status modifier command indicated in TCB flag and command chaining in CCW, post this in TCB flag.
- Reset other TCB flag bits (data chaining read, sense, read backwards, status modifier with data chaining).

Figure 144 (Part 2 of 2). Channel Program Translation (370 Mode): Subroutine GETCCW



Called by
Figure 143: CCWTRANS
Figure 150: TESTIDAL
Figure 151: FIXAREA
Figure 143: CCWTR1
Figure 153: LOCATE
Figure 155: CCWTR19

Get a Copy Block

- Dequeue the first block from the free copy block queue and clear the block.

- If the request is from BTAM but not for a CCB/IORB copy block, increase by one entry in the (copied) CCB for the number of used copy blocks.

- If the request is from BTAM channel appendage, reduce by one entry in the TCB for the required number of additional copy blocks.

- Return to caller unless no copy blocks could be claimed because the free copy block queue is empty.

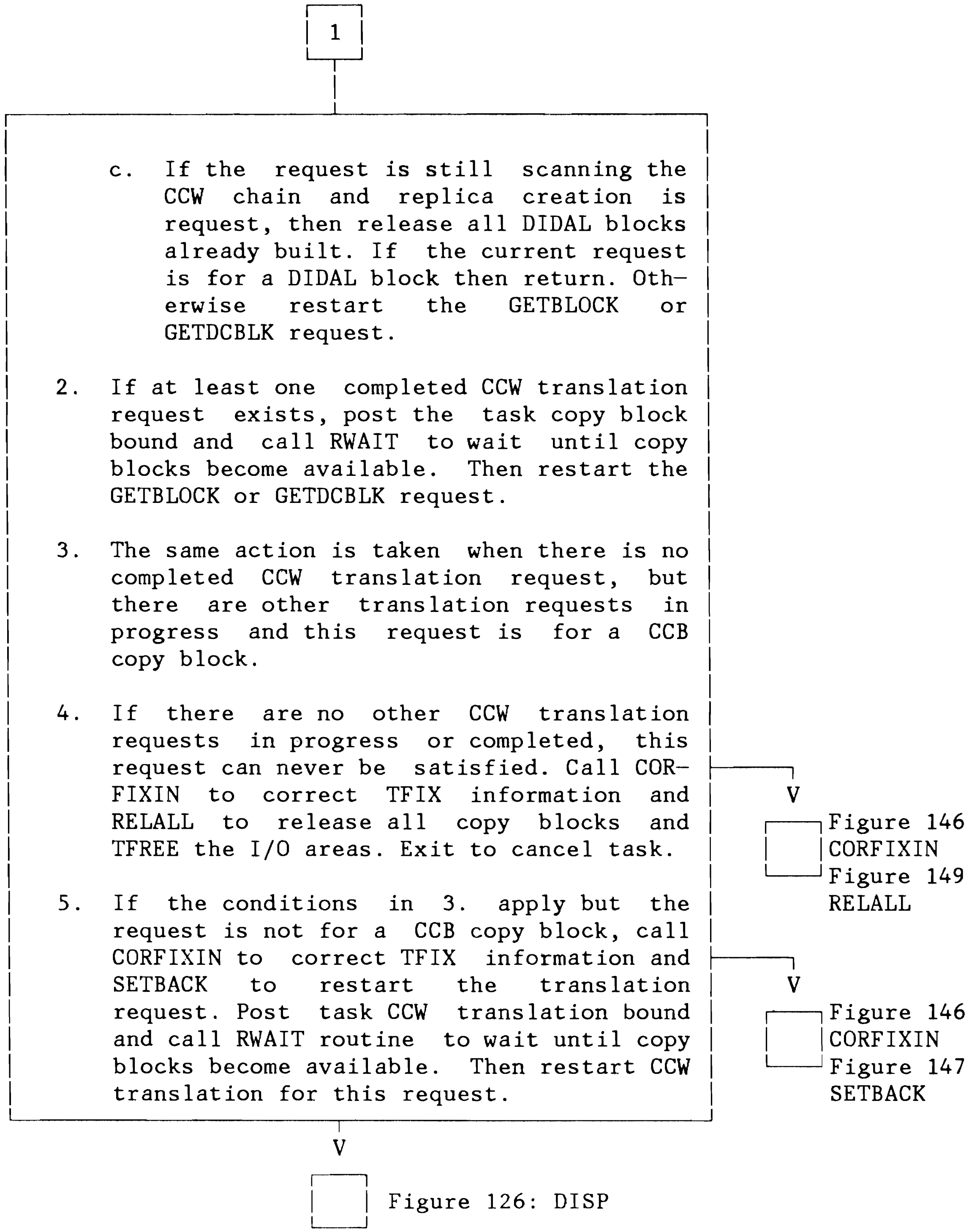
- If the free copy block queue is empty:

  1. If fast translation is supported:

     a. If the copy blocks are kept by fast translation, free replica, CCB/IORB, and copy blocks of bottom saved CCB/IORB queue element and restart GETBLOCK or GETDCBLK request.

     b. If the replica creation is currently in process, release all replica blocks already built for this request, and handle the request as a normal (not fast) one.

Figure 145 (Part 1 of 2). Channel Program Translation (370 Mode): Subroutine GETBLOCK



c. If the request is still scanning the CCW chain and replica creation is request, then release all DIDAL blocks already built. If the current request is for a DIDAL block then return. Otherwise restart the GETBLOCK or GETDCBLK request.

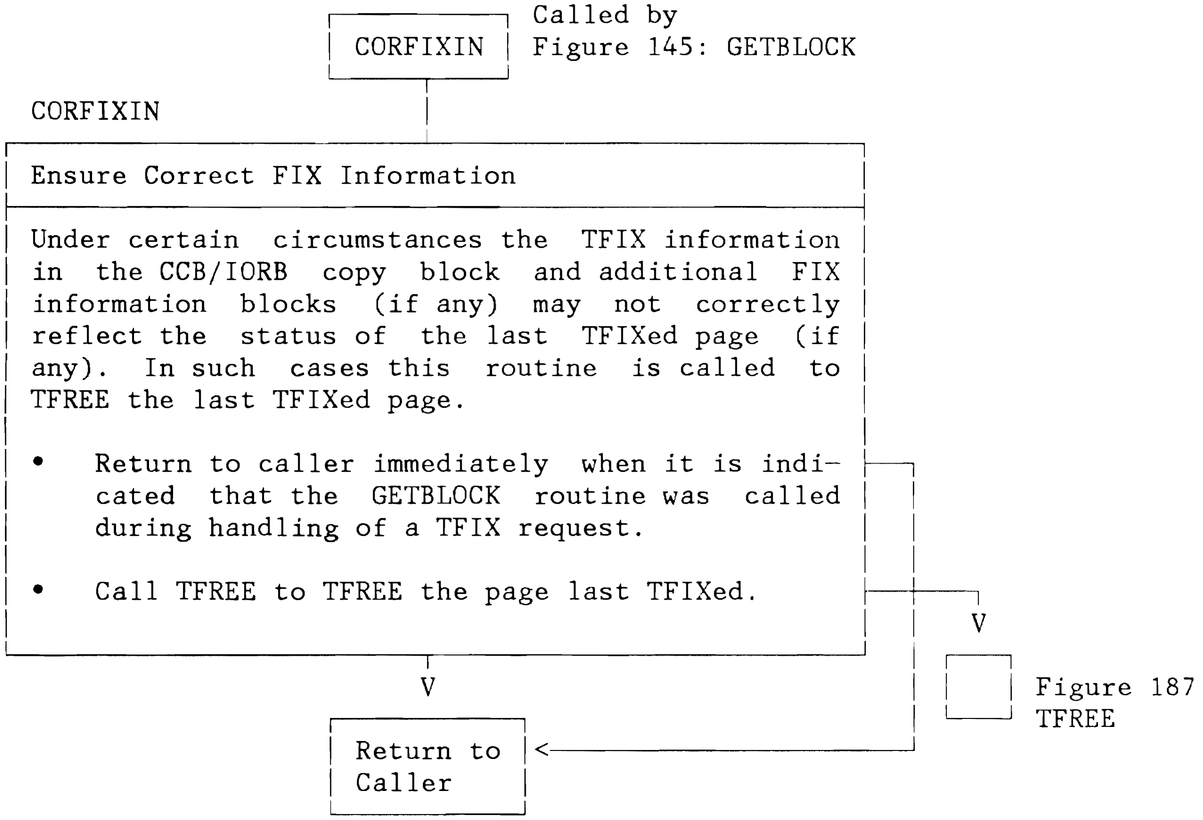
2. If at least one completed CCW translation request exists, post the task copy block bound and call RWAIT to wait until copy blocks become available. Then restart the GETBLOCK or GETDCBLK request.

3. The same action is taken when there is no completed CCW translation request, but there are other translation requests in progress and this request is for a CCB copy block.

4. If there are no other CCW translation requests in progress or completed, this request can never be satisfied. Call CORFIXIN to correct TFIX information and RELALL to release all copy blocks and TFREE the I/O areas. Exit to cancel task.

5. If the conditions in 3. apply but the request is not for a CCB copy block, call CORFIXIN to correct TFIX information and SETBACK to restart the translation request. Post task CCW translation bound and call RWAIT routine to wait until copy blocks become available. Then restart CCW translation for this request.

Figure 145 (Part 2 of 2). Channel Program Translation (370 Mode): Subroutine GETBLOCK



Under certain circumstances the TFIX information in the CCB/IORB copy block and additional FIX information blocks (if any) may not correctly reflect the status of the last TFIXed page (if any). In such cases this routine is called to TFREE the last TFIXed page.

- Return to caller immediately when it is indicated that the GETBLOCK routine was called during handling of a TFIX request.

- Call TFREE to TFREE the page last TFIXed.

Figure 146. Channel Program Translation (370 Mode): Subroutine CORFIXIN

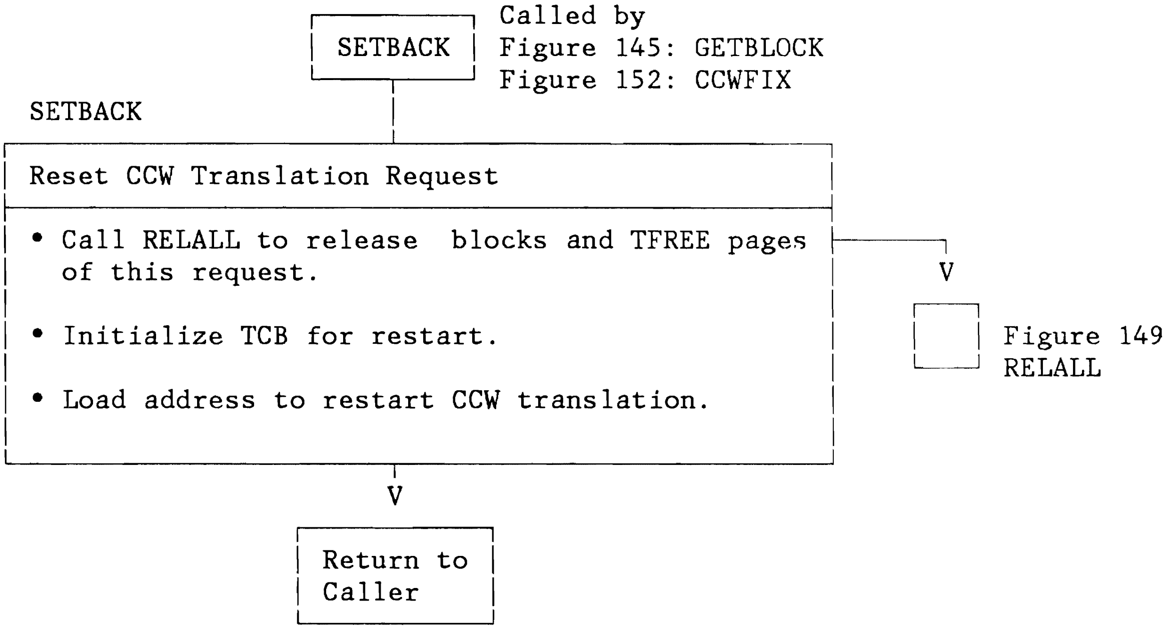
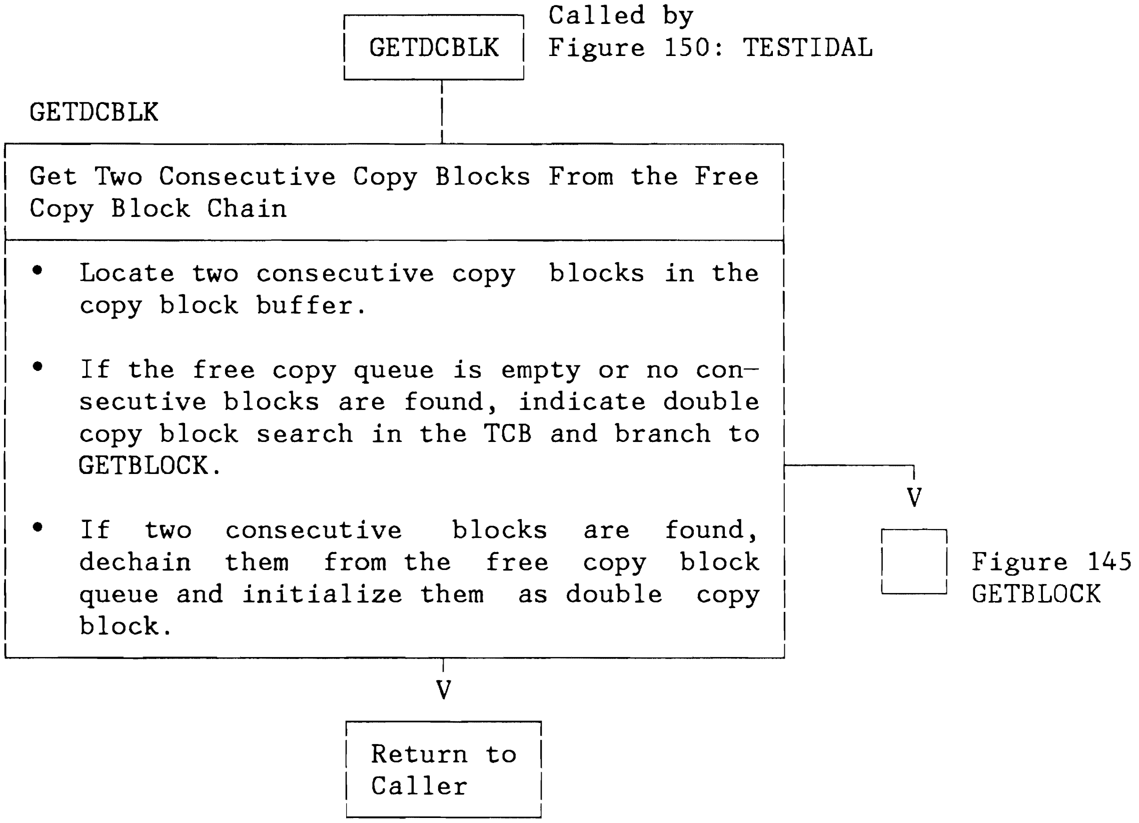

SETBACK
Called by
Figure 145: GETBLOCK
Figure 152: CCWFIX
SETBACK
Reset CCW Translation Request
• Call RELALL to release blocks and TFREE pages of this request.
• Initialize TCB for restart.
• Load address to restart CCW translation.
V
Figure 149
RELALL
V
Return to Caller

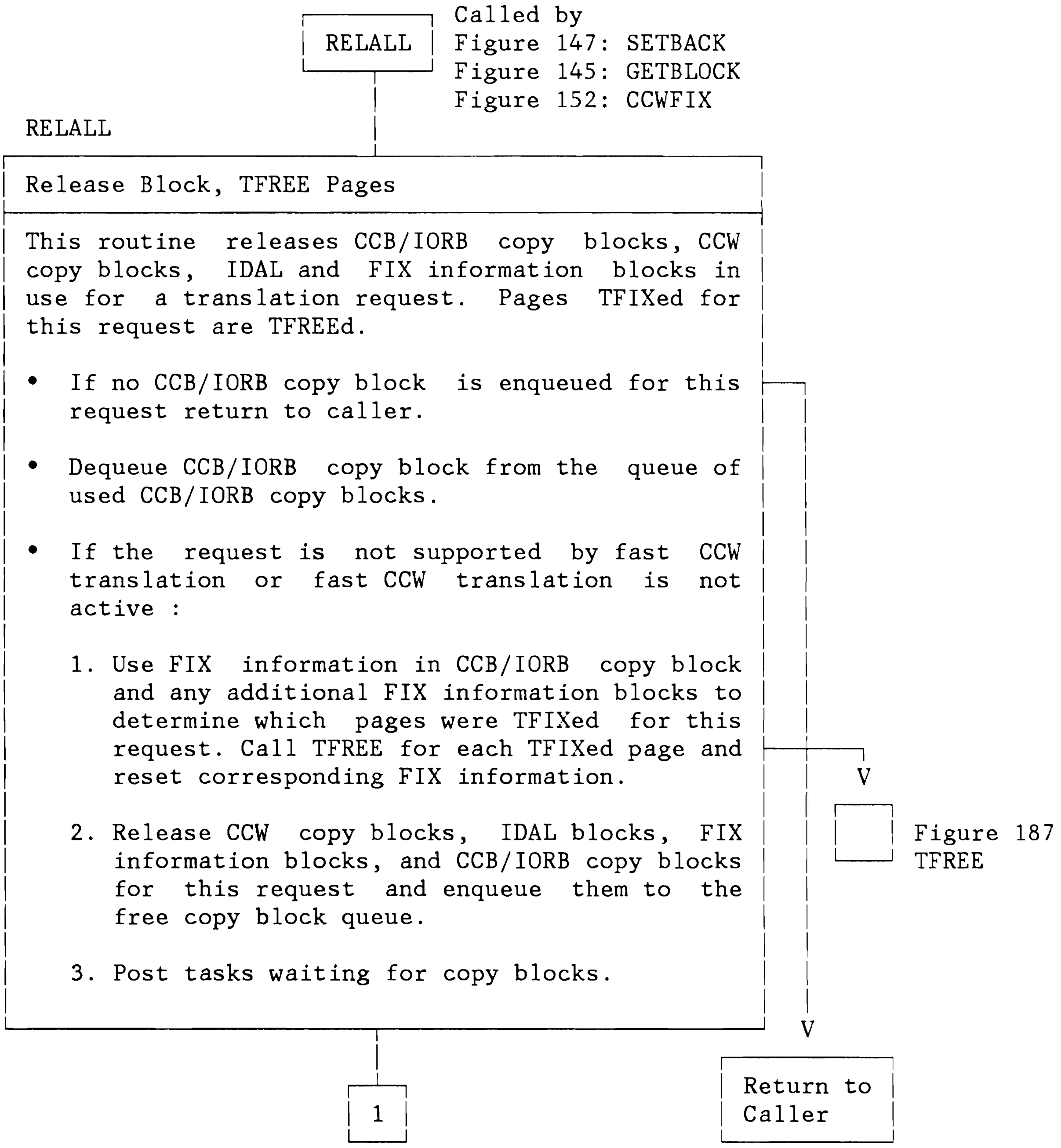
Figure 147. Channel Program Translation (370 Mode): Subroutine SETBACK



Get Two Consecutive Copy Blocks From the Free Copy Block Chain

- Locate two consecutive copy blocks in the copy block buffer.
- If the free copy queue is empty or no consecutive blocks are found, indicate double copy block search in the TCB and branch to GETBLOCK.
- If two consecutive blocks are found, dechain them from the free copy block queue and initialize them as double copy block.

Figure 148. Channel Program Translation (370 Mode): Subroutine GETDCBLK



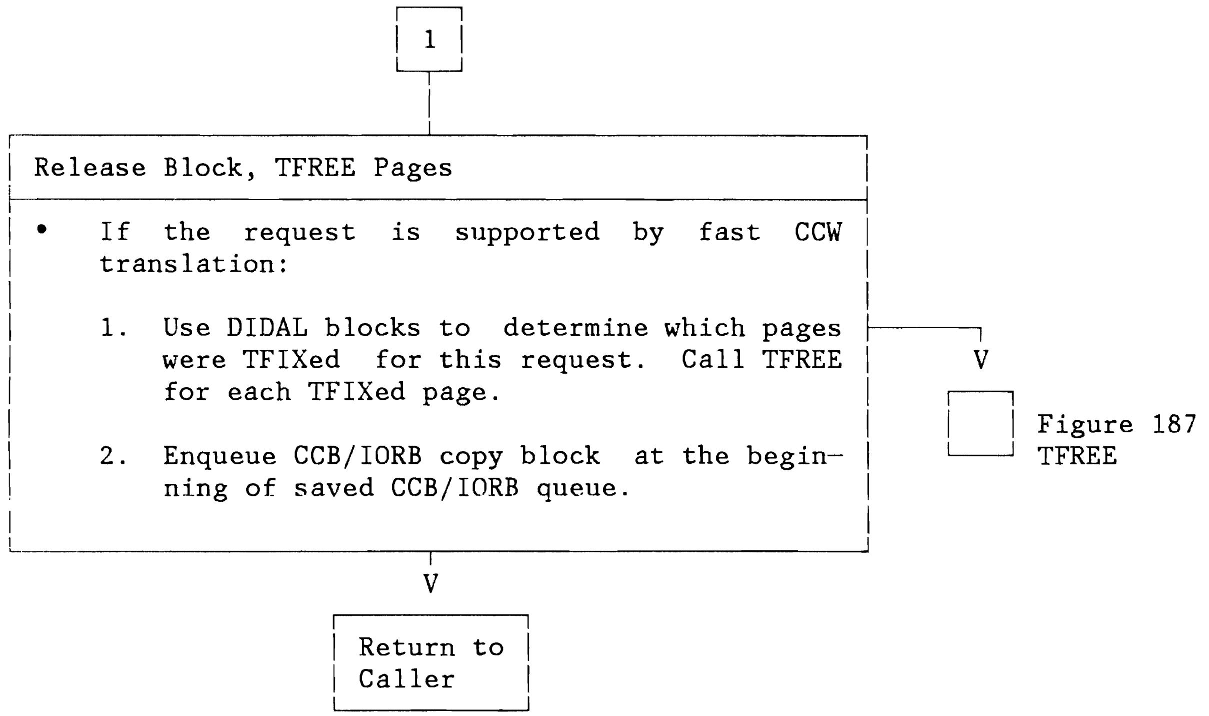
Release Block, TFREE Pages

This routine releases CCB/IORB copy blocks, CCW copy blocks, IDAL and FIX information blocks in use for a translation request. Pages TFIXed for this request are TFREEd.

- If no CCB/IORB copy block is enqueued for this request return to caller.
- Dequeue CCB/IORB copy block from the queue of used CCB/IORB copy blocks.
- If the request is not supported by fast CCW translation or fast CCW translation is not active :

1. Use FIX information in CCB/IORB copy block and any additional FIX information blocks to determine which pages were TFIXed for this request. Call TFREE for each TFIXed page and reset corresponding FIX information.
2. Release CCW copy blocks, IDAL blocks, FIX information blocks, and CCB/IORB copy blocks for this request and enqueue them to the free copy block queue.
3. Post tasks waiting for copy blocks.

Figure 149 (Part 1 of 2). Channel Program Translation (370 Mode): Subroutine RELALL



1

Release Block, TFREE Pages

- If the request is supported by fast CCW translation:

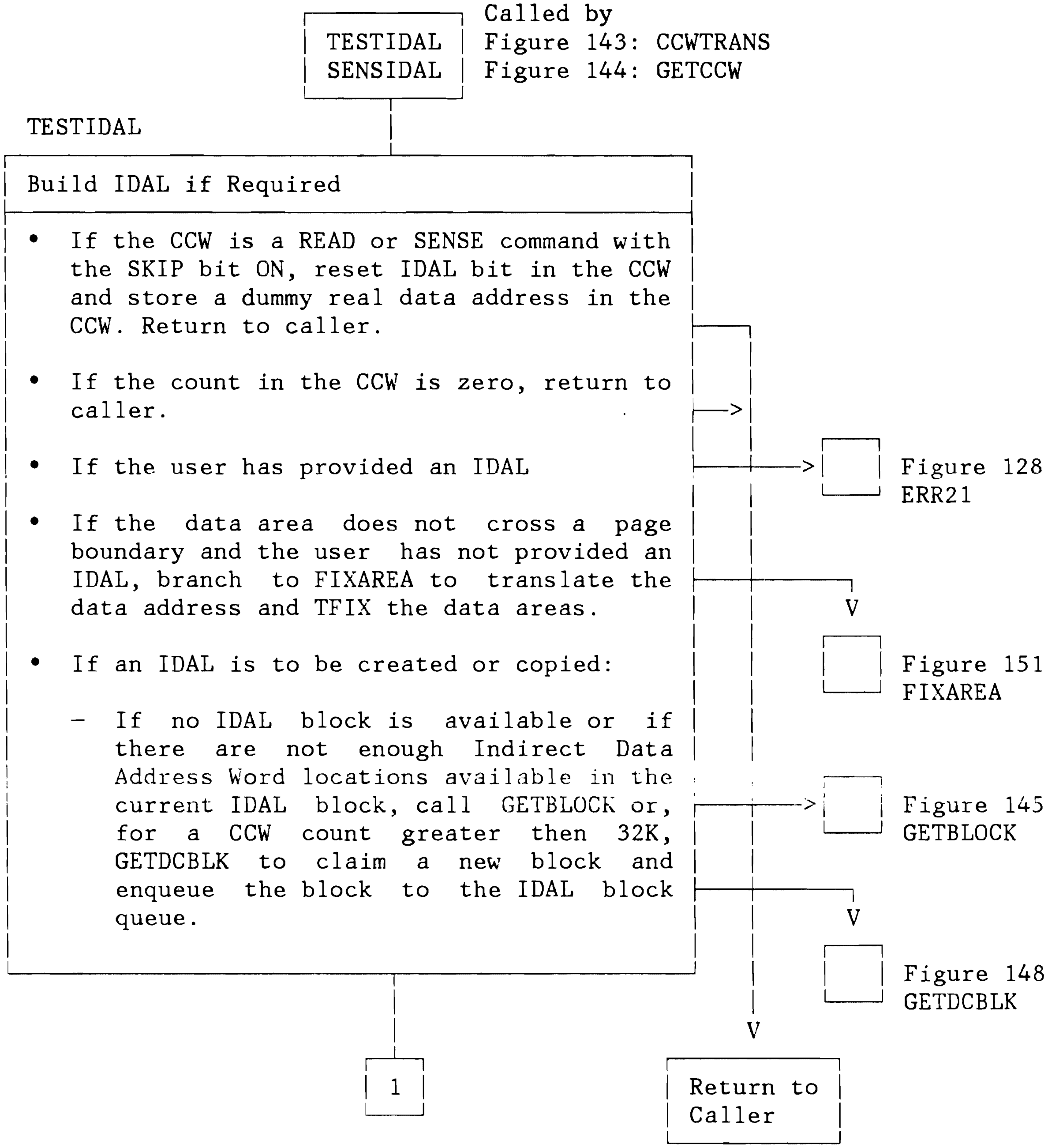
  1. Use DIDAL blocks to determine which pages were TFIXed for this request. Call TFREE for each TFIXed page.

  2. Enqueue CCB/IORB copy block at the beginning of saved CCB/IORB queue.

Figure 187
TFREE

Return to Caller

Figure 149 (Part 2 of 2). Channel Program Translation (370 Mode): Subroutine RELALL



TESTIDAL
SENSIDAL

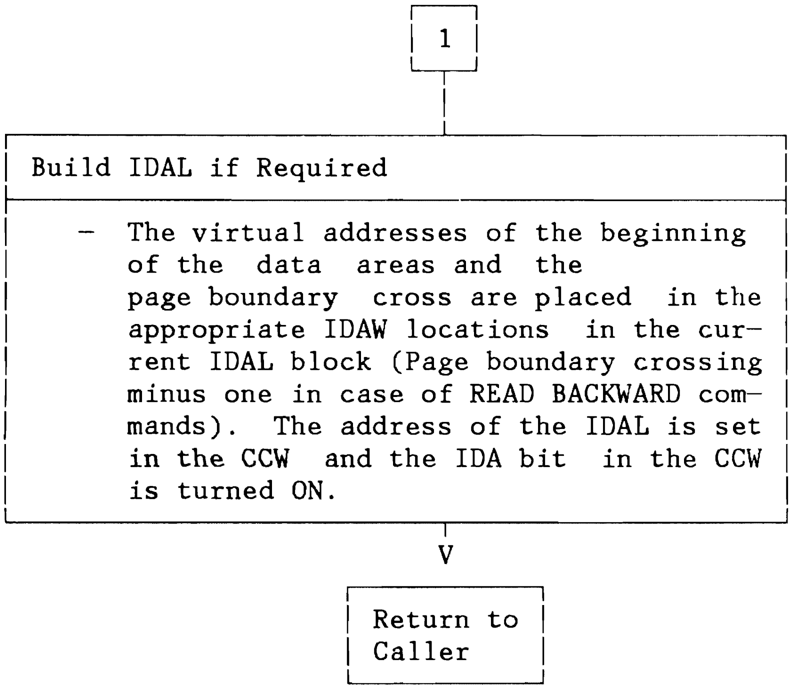
Called by
Figure 143: CCWTRANS
Figure 144: GETCCW

TESTIDAL

Build IDAL if Required

- If the CCW is a READ or SENSE command with the SKIP bit ON, reset IDAL bit in the CCW and store a dummy real data address in the CCW. Return to caller.
- If the count in the CCW is zero, return to caller.
- If the user has provided an IDAL
- If the data area does not cross a page boundary and the user has not provided an IDAL, branch to FIXAREA to translate the data address and TFIX the data areas.
- If an IDAL is to be created or copied:
  - If no IDAL block is available or if there are not enough Indirect Data Address Word locations available in the current IDAL block, call GETBLOCK or, for a CCW count greater then 32K, GETDCBLK to claim a new block and enqueue the block to the IDAL block queue.

Figure 150 (Part 1 of 2). Channel Program Translation (370 Mode): IDAL Building



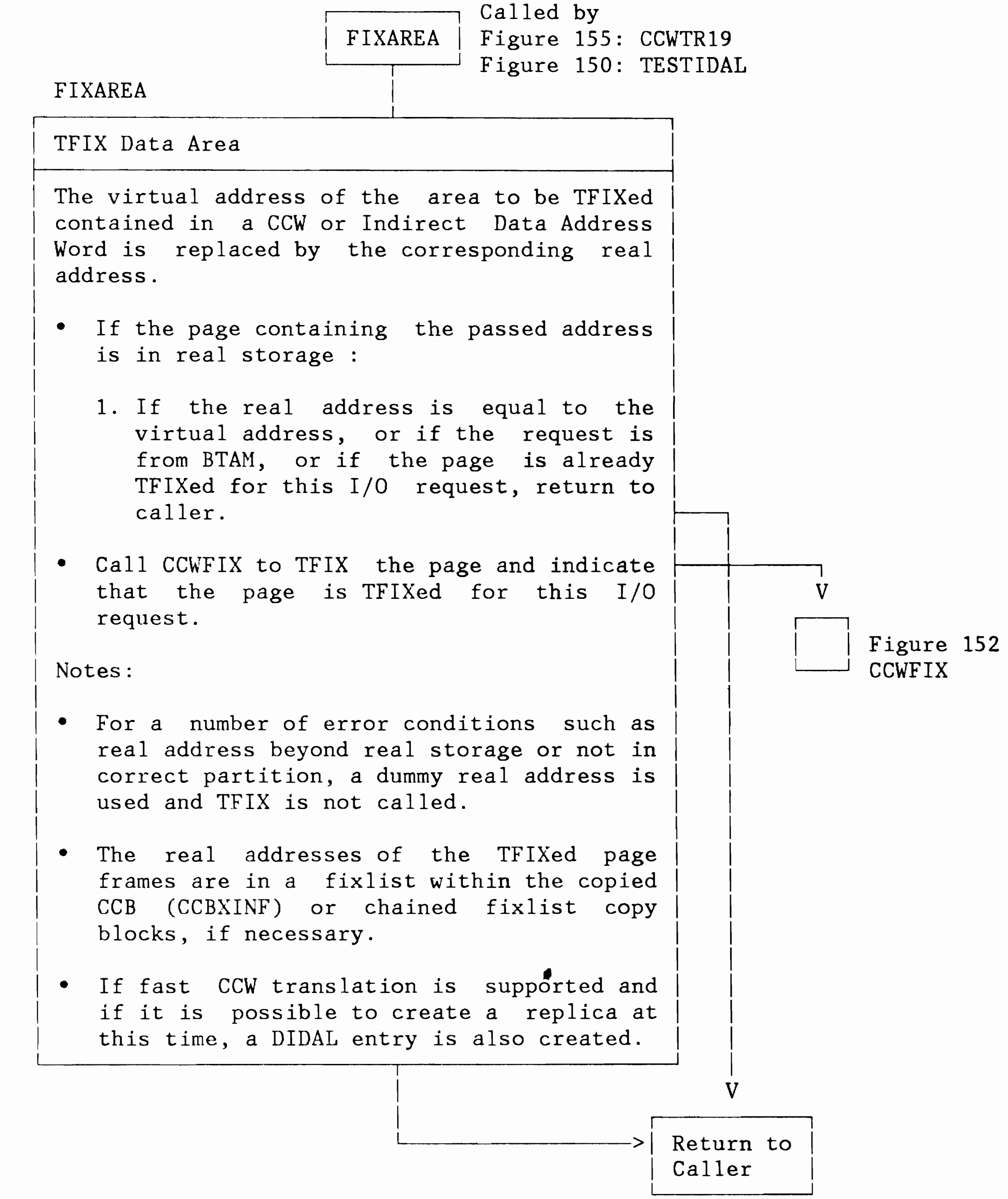
1

Build IDAL if Required

- The virtual addresses of the beginning of the data areas and the page boundary cross are placed in the appropriate IDAW locations in the current IDAL block (Page boundary crossing minus one in case of READ BACKWARD commands). The address of the IDAL is set in the CCW and the IDA bit in the CCW is turned ON.

Return to Caller

Figure 150 (Part 2 of 2). Channel Program Translation (370 Mode): IDAL Building



TFIX Data Area

The virtual address of the area to be TFIXed contained in a CCW or Indirect Data Address Word is replaced by the corresponding real address.

- If the page containing the passed address is in real storage :

  1. If the real address is equal to the virtual address, or if the request is from BTAM, or if the page is already TFIXed for this I/O request, return to caller.

- Call CCWFIX to TFIX the page and indicate that the page is TFIXed for this I/O request.

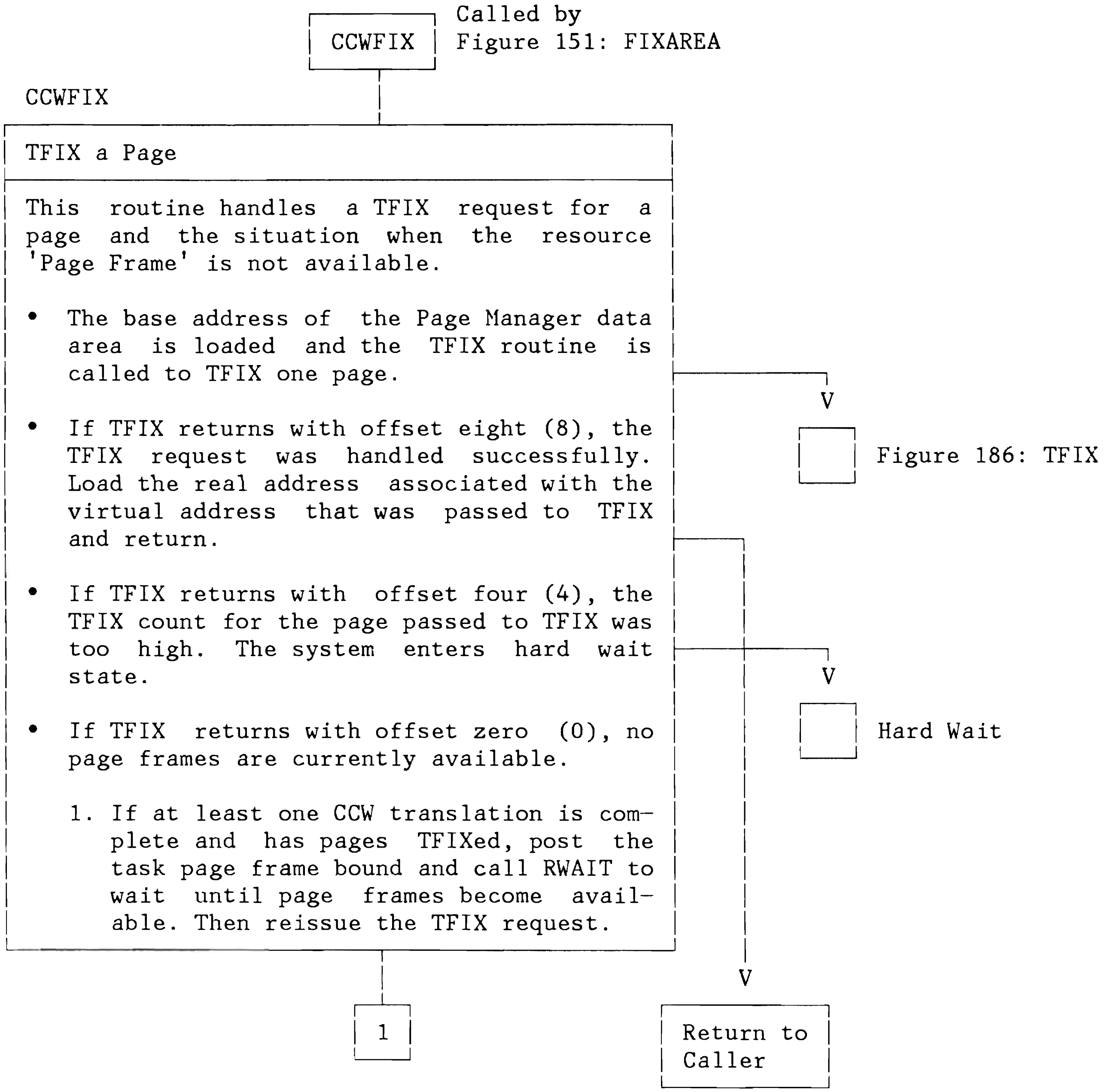
Notes:

- For a number of error conditions such as real address beyond real storage or not in correct partition, a dummy real address is used and TFIX is not called.

- The real addresses of the TFIXed page frames are in a fixlist within the copied CCB (CCBXINF) or chained fixlist copy blocks, if necessary.

- If fast CCW translation is supported and if it is possible to create a replica at this time, a DIDAL entry is also created.

Figure 151. Channel Program Translation (370 Mode): Data Area Fixing



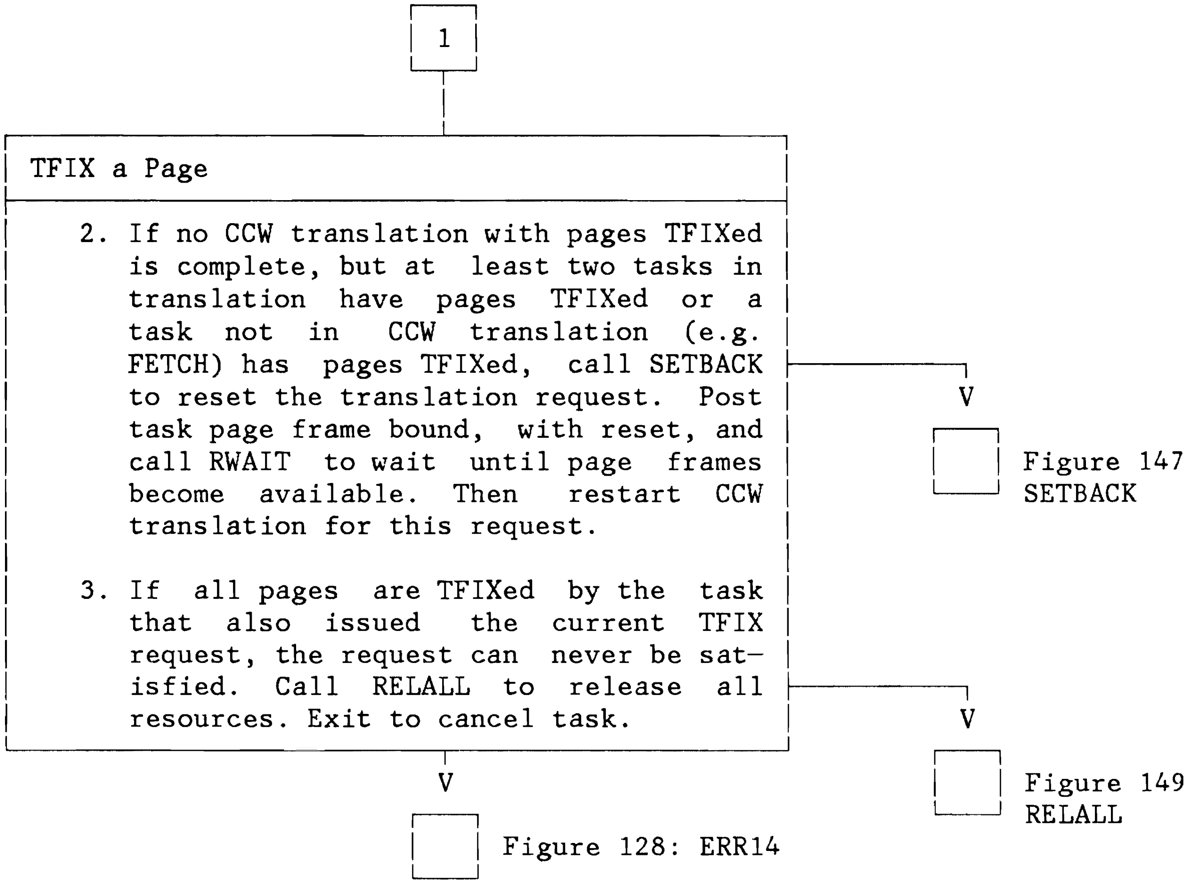
TFIX a Page

This routine handles a TFIX request for a page and the situation when the resource 'Page Frame' is not available.

- The base address of the Page Manager data area is loaded and the TFIX routine is called to TFIX one page.
- If TFIX returns with offset eight (8), the TFIX request was handled successfully. Load the real address associated with the virtual address that was passed to TFIX and return.
- If TFIX returns with offset four (4), the TFIX count for the page passed to TFIX was too high. The system enters hard wait state.
- If TFIX returns with offset zero (0), no page frames are currently available.

  1. If at least one CCW translation is complete and has pages TFIXed, post the task page frame bound and call RWAIT to wait until page frames become available. Then reissue the TFIX request.

Figure 152 (Part 1 of 2). Channel Program Translation (370 Mode): TFIXing a Page

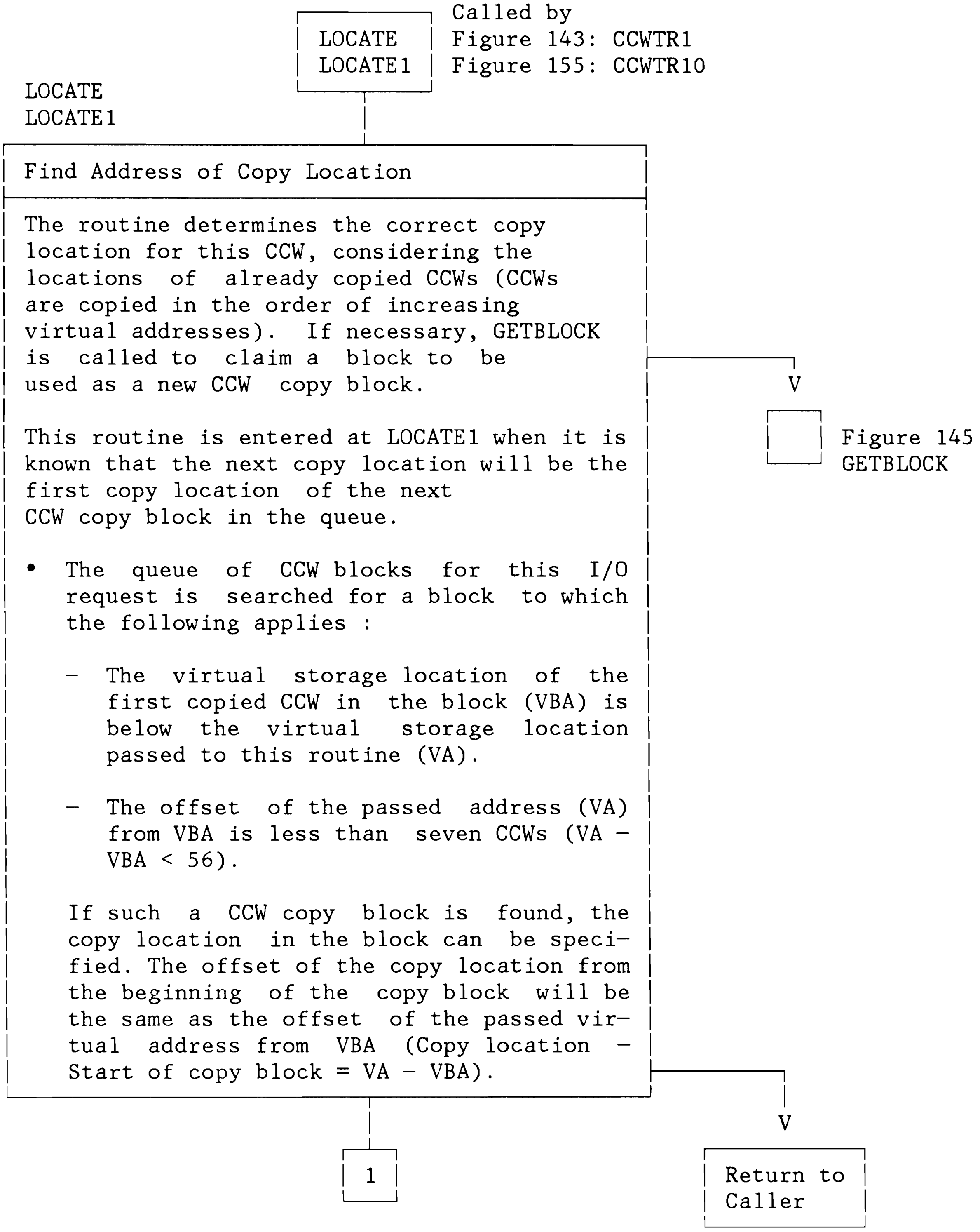


TFIX a Page

2. If no CCW translation with pages TFIXed is complete, but at least two tasks in translation have pages TFIXed or a task not in CCW translation (e.g. FETCH) has pages TFIXed, call SETBACK to reset the translation request. Post task page frame bound, with reset, and call RWAIT to wait until page frames become available. Then restart CCW translation for this request.

3. If all pages are TFIXed by the task that also issued the current TFIX request, the request can never be sat‑isfied. Call RELALL to release all resources. Exit to cancel task.

Figure 152 (Part 2 of 2). Channel Program Translation (370 Mode): TFIXing a Page



LOCATE
LOCATE1

Called by
Figure 143: CCWTR1
Figure 155: CCWTR10

LOCATE
LOCATE1

Find Address of Copy Location

The routine determines the correct copy location for this CCW, considering the locations of already copied CCWs (CCWs are copied in the order of increasing virtual addresses). If necessary, GETBLOCK is called to claim a block to be used as a new CCW copy block.

This routine is entered at LOCATE1 when it is known that the next copy location will be the first copy location of the next CCW copy block in the queue.

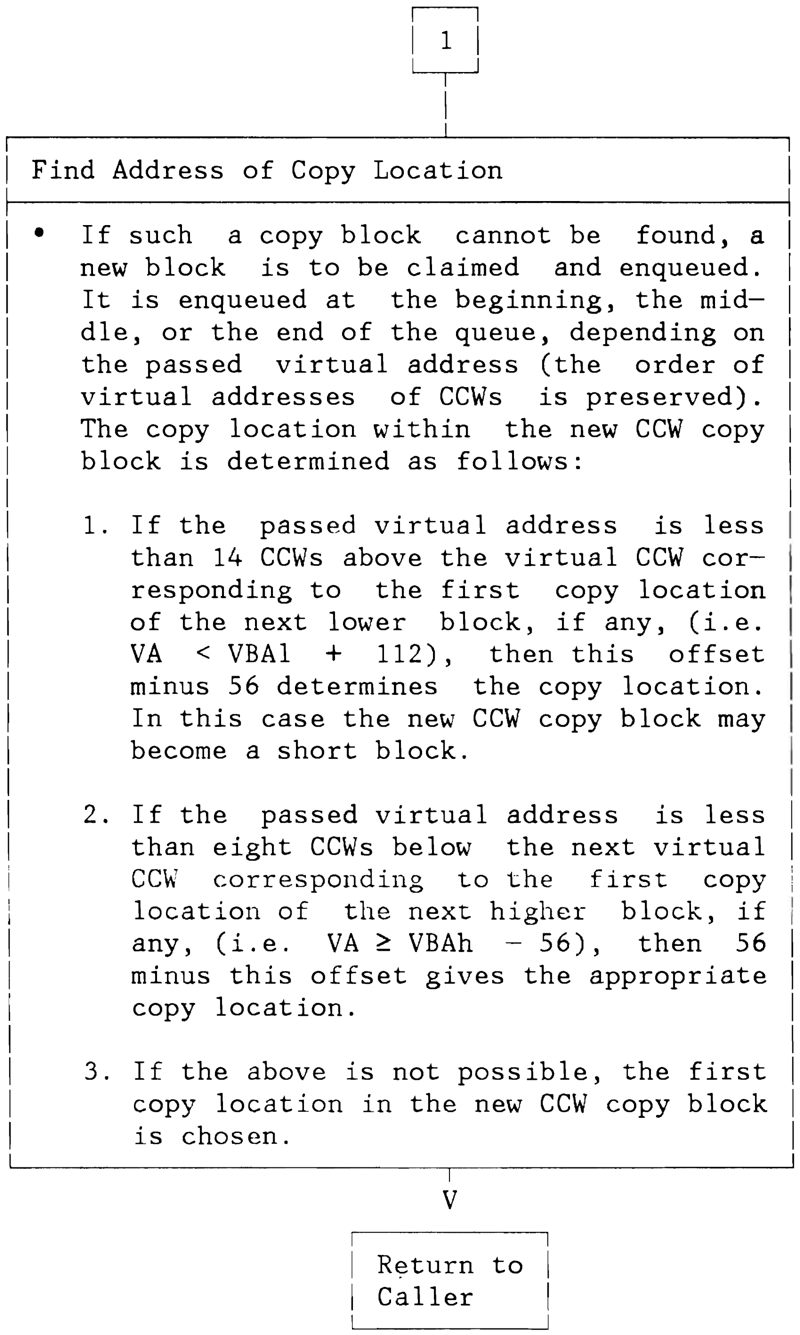
- The queue of CCW blocks for this I/O request is searched for a block to which the following applies :

  - The virtual storage location of the first copied CCW in the block (VBA) is below the virtual storage location passed to this routine (VA).

  - The offset of the passed address (VA) from VBA is less than seven CCWs (VA - VBA < 56).

  If such a CCW copy block is found, the copy location in the block can be specified. The offset of the copy location from the beginning of the copy block will be the same as the offset of the passed virtual address from VBA (Copy location - Start of copy block = VA - VBA).

Figure 153 (Part 1 of 2). Channel Program Translation (370 Mode): Locate Routine



1

Find Address of Copy Location

- If such a copy block cannot be found, a new block is to be claimed and enqueued. It is enqueued at the beginning, the middle, or the end of the queue, depending on the passed virtual address (the order of virtual addresses of CCWs is preserved). The copy location within the new CCW copy block is determined as follows:

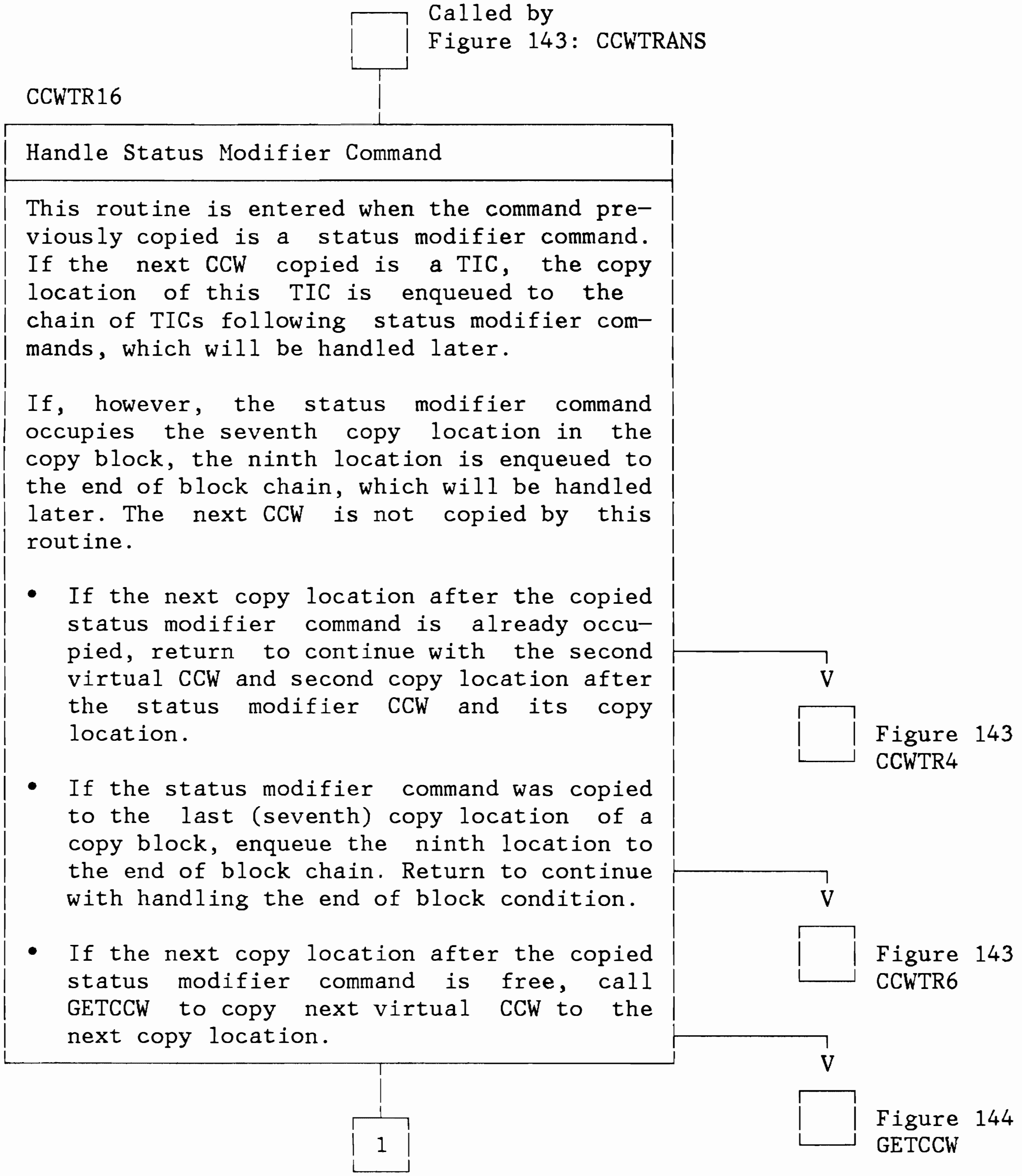
  1. If the passed virtual address is less than 14 CCWs above the virtual CCW corresponding to the first copy location of the next lower block, if any, (i.e. VA < VBAl + 112), then this offset minus 56 determines the copy location. In this case the new CCW copy block may become a short block.

  2. If the passed virtual address is less than eight CCWs below the next virtual CCW corresponding to the first copy location of the next higher block, if any, (i.e. VA ≥ VBAh - 56), then 56 minus this offset gives the appropriate copy location.

  3. If the above is not possible, the first copy location in the new CCW copy block is chosen.

Return to Caller

Figure 153 (Part 2 of 2). Channel Program Translation (370 Mode): Locate Routine



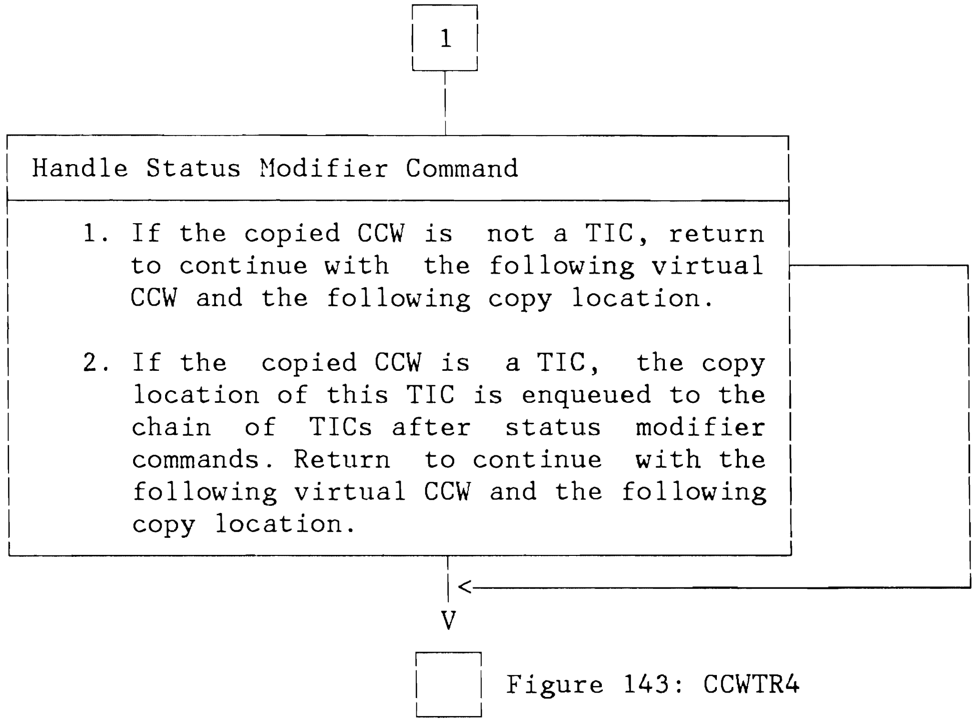
Handle Status Modifier Command

This routine is entered when the command previously copied is a status modifier command. If the next CCW copied is a TIC, the copy location of this TIC is enqueued to the chain of TICs following status modifier commands, which will be handled later.

If, however, the status modifier command occupies the seventh copy location in the copy block, the ninth location is enqueued to the end of block chain, which will be handled later. The next CCW is not copied by this routine.

- If the next copy location after the copied status modifier command is already occupied, return to continue with the second virtual CCW and second copy location after the status modifier CCW and its copy location.
- If the status modifier command was copied to the last (seventh) copy location of a copy block, enqueue the ninth location to the end of block chain. Return to continue with handling the end of block condition.
- If the next copy location after the copied status modifier command is free, call GETCCW to copy next virtual CCW to the next copy location.

Figure 154 (Part 1 of 2). Channel Pr. Trans.(370 Mode): Status Modifier Command Handling

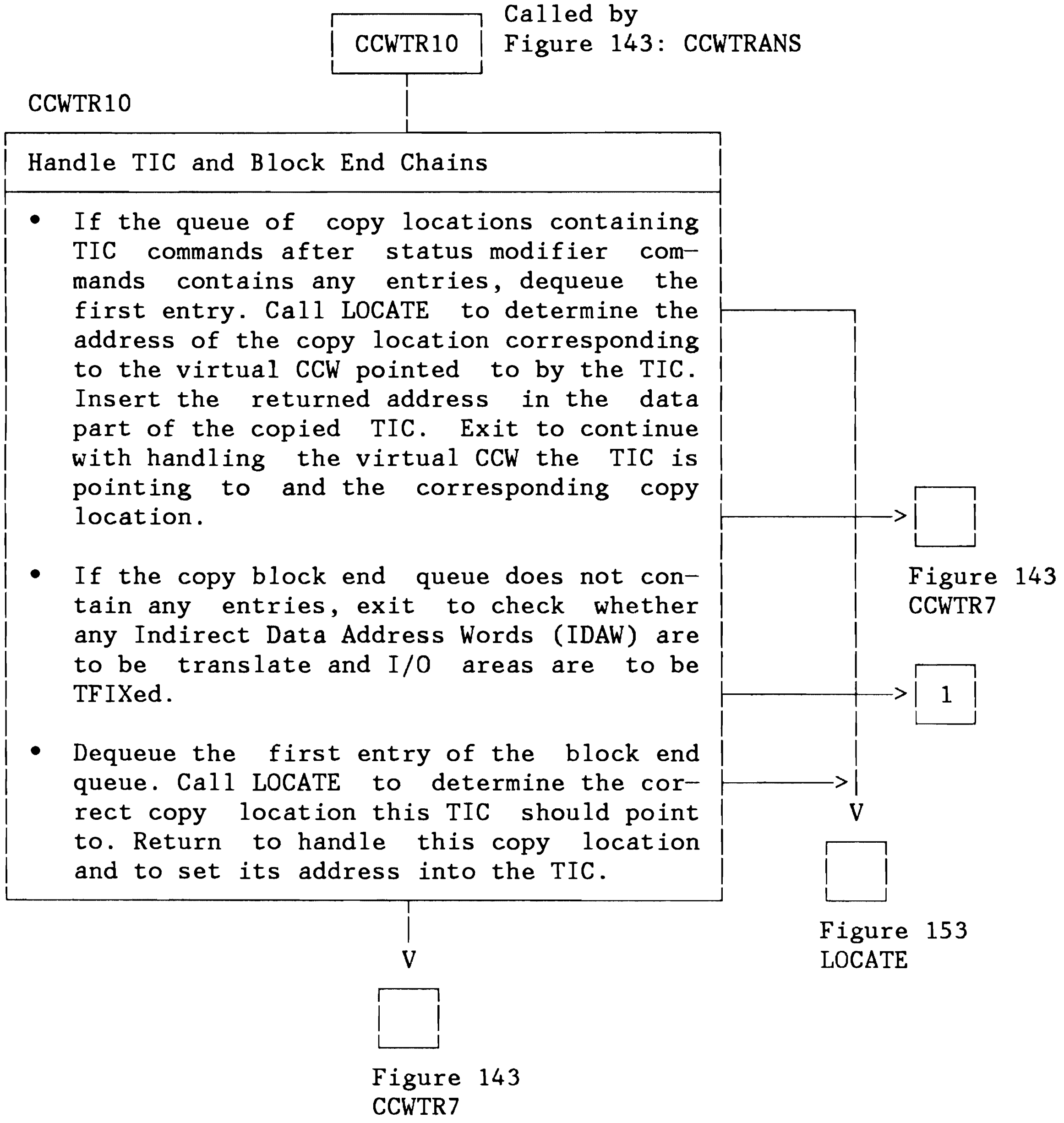


Handle Status Modifier Command

1. If the copied CCW is not a TIC, return to continue with the following virtual CCW and the following copy location.

2. If the copied CCW is a TIC, the copy location of this TIC is enqueued to the chain of TICs after status modifier commands. Return to continue with the following virtual CCW and the following copy location.

Figure 154 (Part 2 of 2). Channel Pr. Trans.(370 Mode): Status Modifier Command Handling



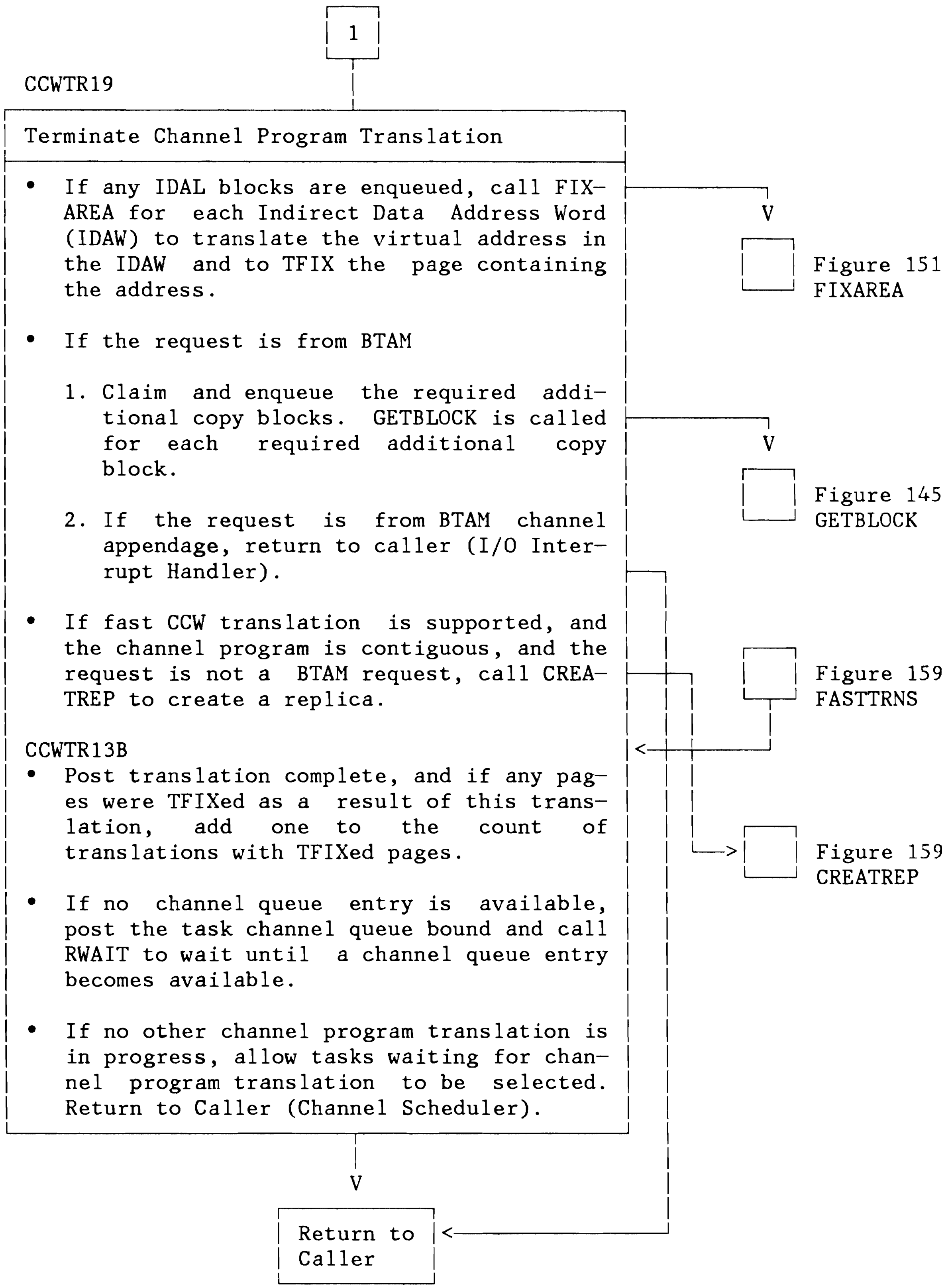
Handle TIC and Block End Chains

- If the queue of copy locations containing TIC commands after status modifier commands contains any entries, dequeue the first entry. Call LOCATE to determine the address of the copy location corresponding to the virtual CCW pointed to by the TIC. Insert the returned address in the data part of the copied TIC. Exit to continue with handling the virtual CCW the TIC is pointing to and the corresponding copy location.

- If the copy block end queue does not contain any entries, exit to check whether any Indirect Data Address Words (IDAW) are to be translate and I/O areas are to be TFIXed.

- Dequeue the first entry of the block end queue. Call LOCATE to determine the correct copy location this TIC should point to. Return to handle this copy location and to set its address into the TIC.

Figure 155 (Part 1 of 2). Channel Program Translation (370 Mode): Handling of TIC



1

CCWTR19

Terminate Channel Program Translation

- If any IDAL blocks are enqueued, call FIXAREA for each Indirect Data Address Word (IDAW) to translate the virtual address in the IDAW and to TFIX the page containing the address.

- If the request is from BTAM

  1. Claim and enqueue the required additional copy blocks. GETBLOCK is called for each required additional copy block.

  2. If the request is from BTAM channel appendage, return to caller (I/O Interrupt Handler).

- If fast CCW translation is supported, and the channel program is contiguous, and the request is not a BTAM request, call CREATREP to create a replica.

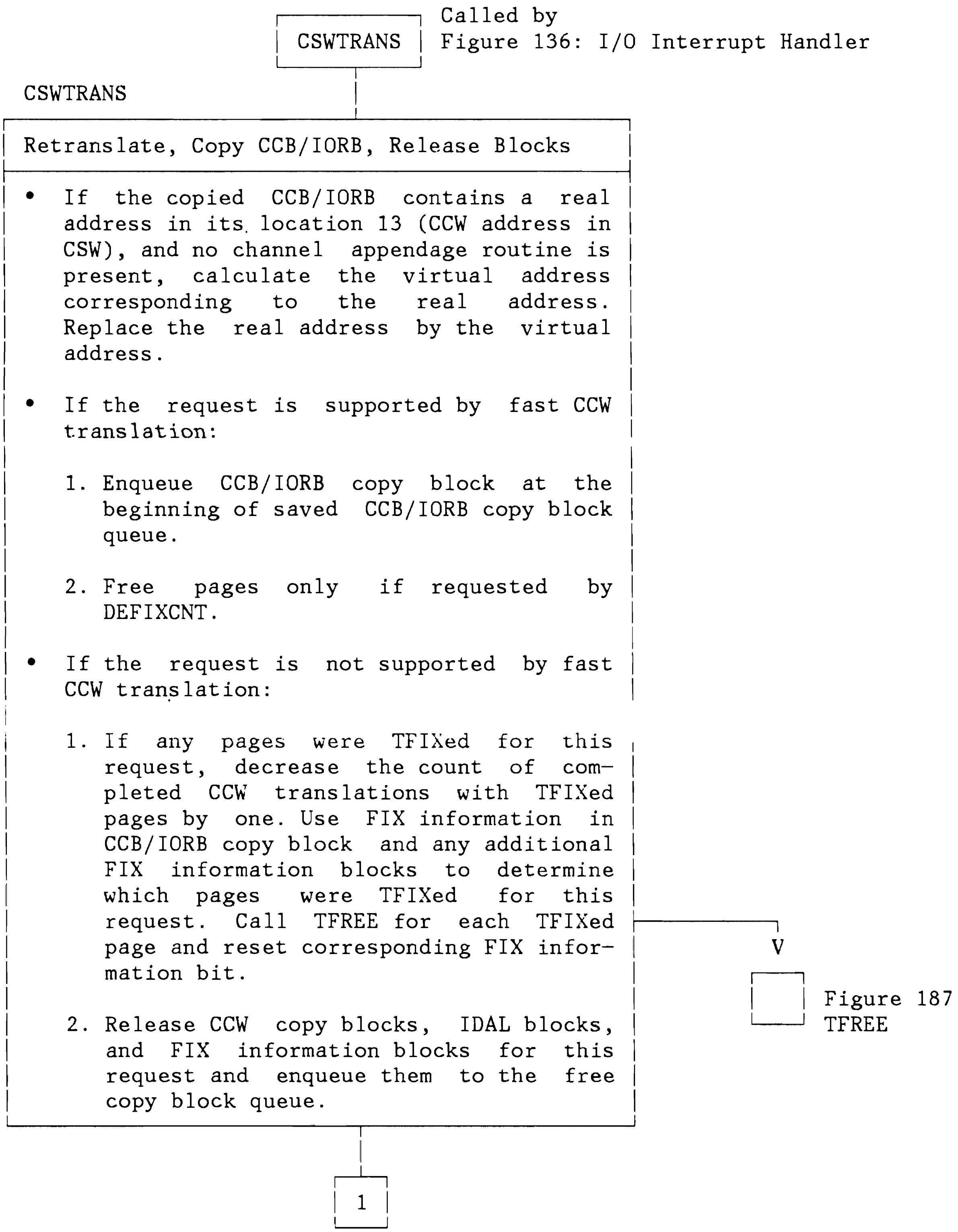
CCWTR13B

- Post translation complete, and if any pages were TFIXed as a result of this translation, add one to the count of translations with TFIXed pages.

- If no channel queue entry is available, post the task channel queue bound and call RWAIT to wait until a channel queue entry becomes available.

- If no other channel program translation is in progress, allow tasks waiting for channel program translation to be selected. Return to Caller (Channel Scheduler).

Figure 155 (Part 2 of 2). Channel Program Translation (370 Mode): Handling of TIC



CSWTRANS

Called by
Figure 136: I/O Interrupt Handler

CSWTRANS

Retranslate, Copy CCB/IORB, Release Blocks

- If the copied CCB/IORB contains a real address in its location 13 (CCW address in CSW), and no channel appendage routine is present, calculate the virtual address corresponding to the real address. Replace the real address by the virtual address.

- If the request is supported by fast CCW translation:

  1. Enqueue CCB/IORB copy block at the beginning of saved CCB/IORB copy block queue.

  2. Free pages only if requested by DEFIXCNT.

- If the request is not supported by fast CCW translation:

  1. If any pages were TFIXed for this request, decrease the count of completed CCW translations with TFIXed pages by one. Use FIX information in CCB/IORB copy block and any additional FIX information blocks to determine which pages were TFIXed for this request. Call TFREE for each TFIXed page and reset corresponding FIX information bit.

  2. Release CCW copy blocks, IDAL blocks, and FIX information blocks for this request and enqueue them to the free copy block queue.

Figure 156 (Part 1 of 2). Channel Program Translation (370 Mode): Retranslation

1

Retranslate, Copy CCB/IORB, Release Blocks

3. Allow any tasks waiting for copy blocks to be selected.

- If virtual CCB/IORB is in real storage, and emulator ECB is not to be posted, move significant parts of the copied CCB/IORB to the virtual CCB/IORB. Release the CCB/IORB copy block and enqueue it to the free copy block queue.

- If the virtual CCB/IORB is not in real storage, post PIB to indicate that copying of CCB/IORB and release of CCB/IORB copy block is to be done at a later stage (when the task is selected the next time, as is also the case, if the emulator ECB is to be posted).

- If fast CCW translation is supported:

  1. If tasks are waiting for copy blocks, free copy blocks kept by fast CCW translation.

  2. If tasks are waiting for page frames, free page frames kept by fast CCW translation.

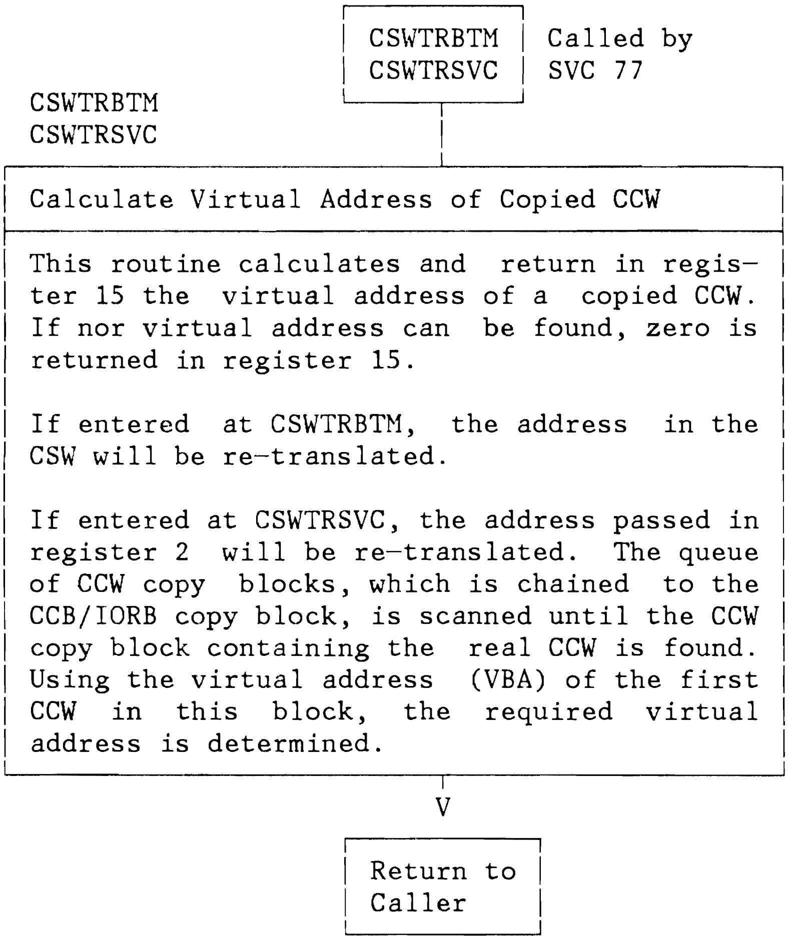
- Allow tasks waiting for channel program translation or page frames to be selected.

- If count of completed translations with TFIXed pages is zero, allow all tasks waiting for page frames to be selected.

V

Return to Caller

Figure 156 (Part 2 of 2). Channel Program Translation (370 Mode): Retranslation



CSWTRBTM
CSWTRSVC

Called by
SVC 77

CSWTRBTM
CSWTRSVC

Calculate Virtual Address of Copied CCW

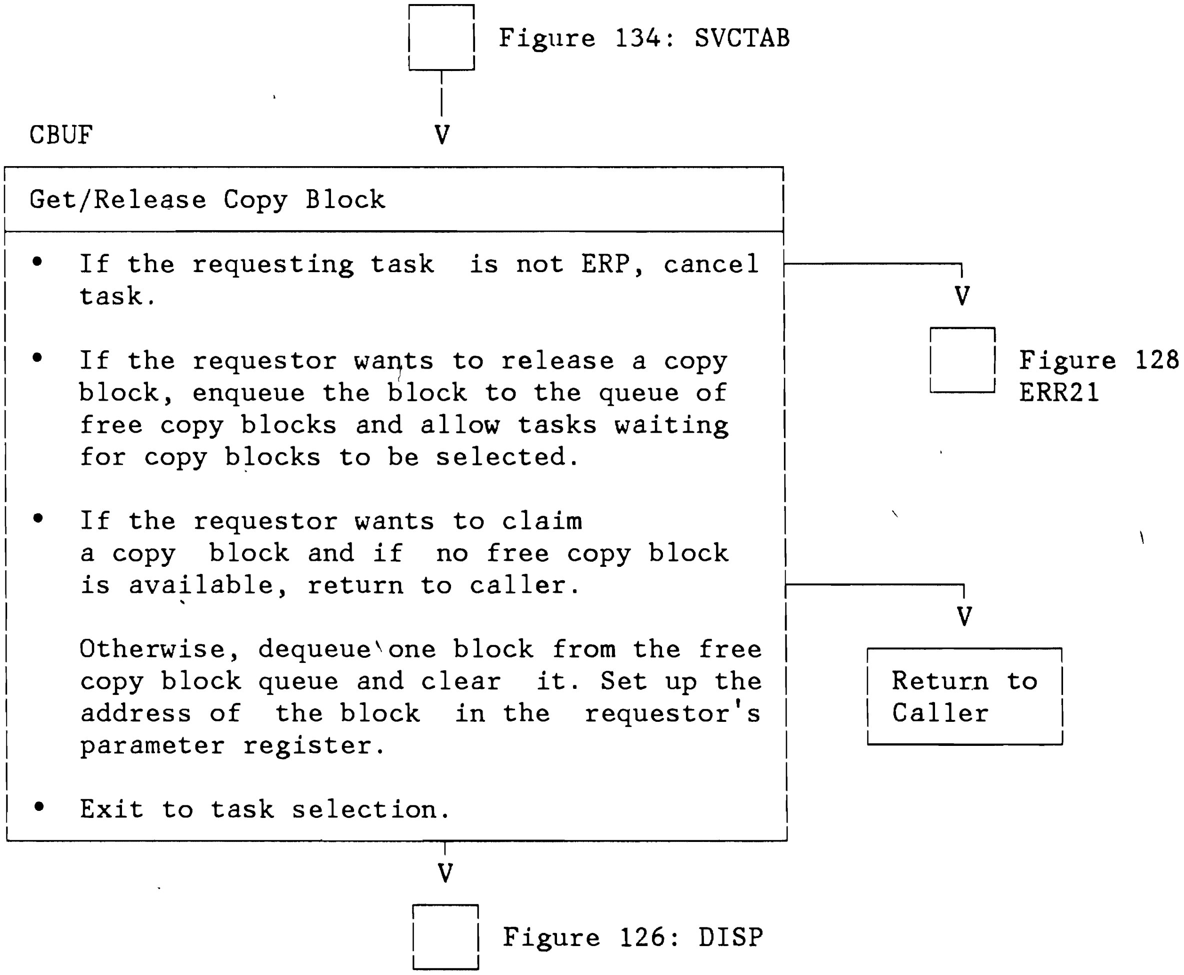
This routine calculates and return in register 15 the virtual address of a copied CCW. If nor virtual address can be found, zero is returned in register 15.

If entered at CSWTRBTM, the address in the CSW will be re-translated.

If entered at CSWTRSVC, the address passed in register 2 will be re-translated. The queue of CCW copy blocks, which is chained to the CCB/IORB copy block, is scanned until the CCW copy block containing the real CCW is found. Using the virtual address (VBA) of the first CCW in this block, the required virtual address is determined.

Return to
Caller

Figure 157. Channel Program Translation (370 Mode): SVC 77

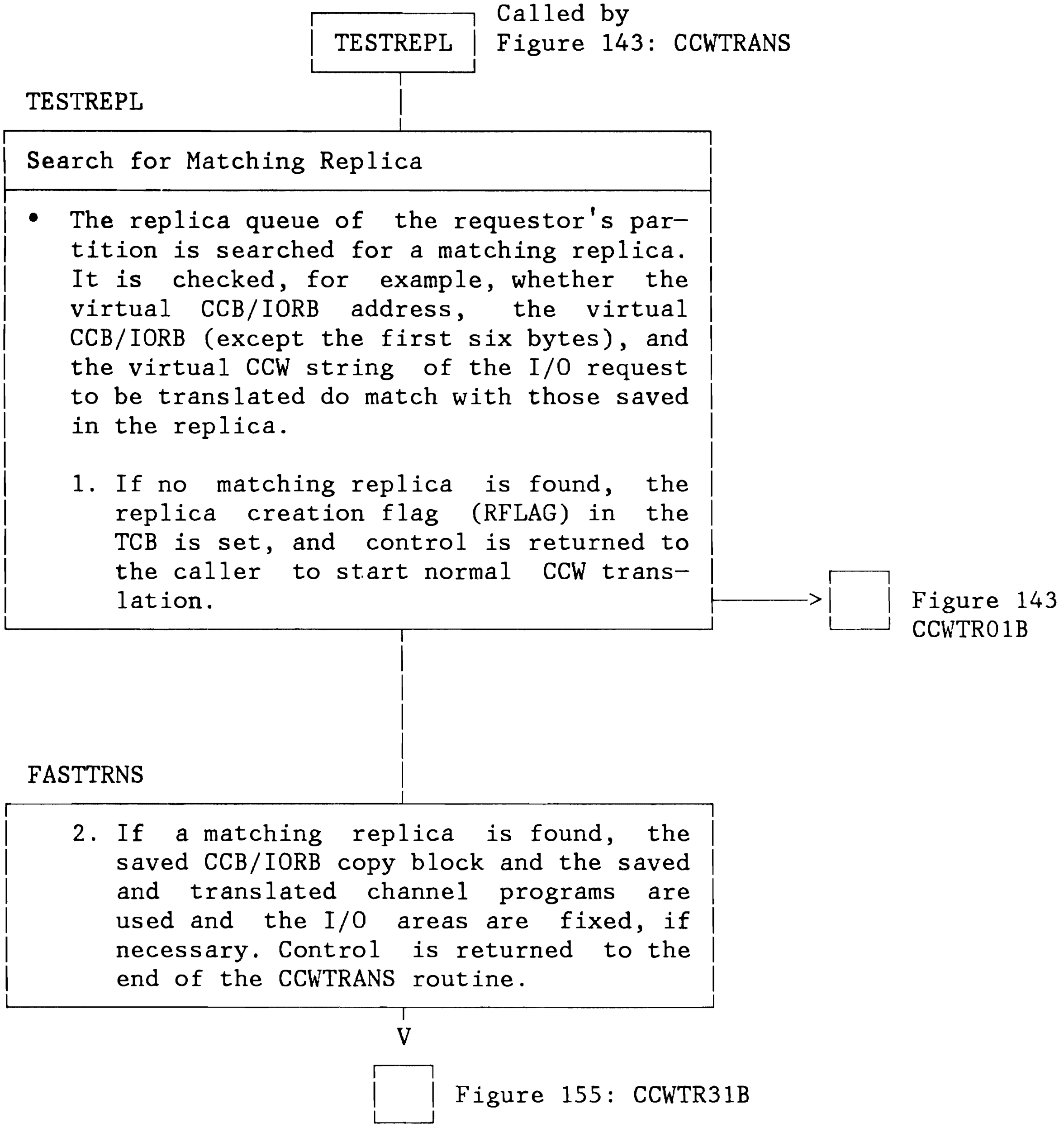


Get/Release Copy Block

- If the requesting task is not ERP, cancel task.
- If the requestor wants to release a copy block, enqueue the block to the queue of free copy blocks and allow tasks waiting for copy blocks to be selected.
- If the requestor wants to claim a copy block and if no free copy block is available, return to caller.

  Otherwise, dequeue one block from the free copy block queue and clear it. Set up the address of the block in the requestor's parameter register.
- Exit to task selection.

Figure 158. Channel Program Translation (370 Mode): SVC 72

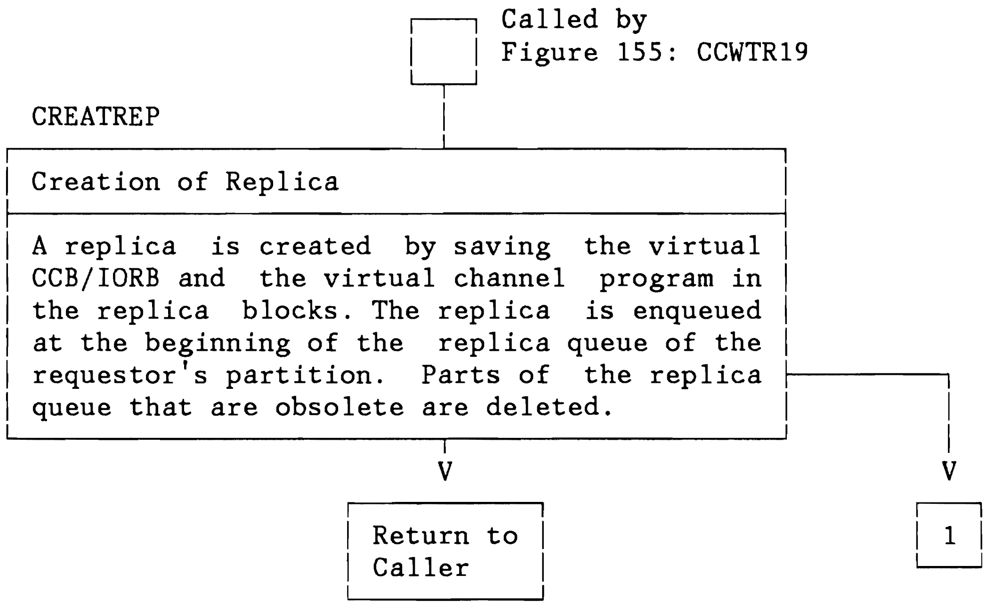


- The replica queue of the requestor's partition is searched for a matching replica. It is checked, for example, whether the virtual CCB/IORB address, the virtual CCB/IORB (except the first six bytes), and the virtual CCW string of the I/O request to be translated do match with those saved in the replica.

  1. If no matching replica is found, the replica creation flag (RFLAG) in the TCB is set, and control is returned to the caller to start normal CCW translation.

  2. If a matching replica is found, the saved CCB/IORB copy block and the saved and translated channel programs are used and the I/O areas are fixed, if necessary. Control is returned to the end of the CCWTRANS routine.

Figure 159 (Part 1 of 3). Channel Program Translation (370 Mode): Fast CCW Translation



Called by
Figure 155: CCWTR19

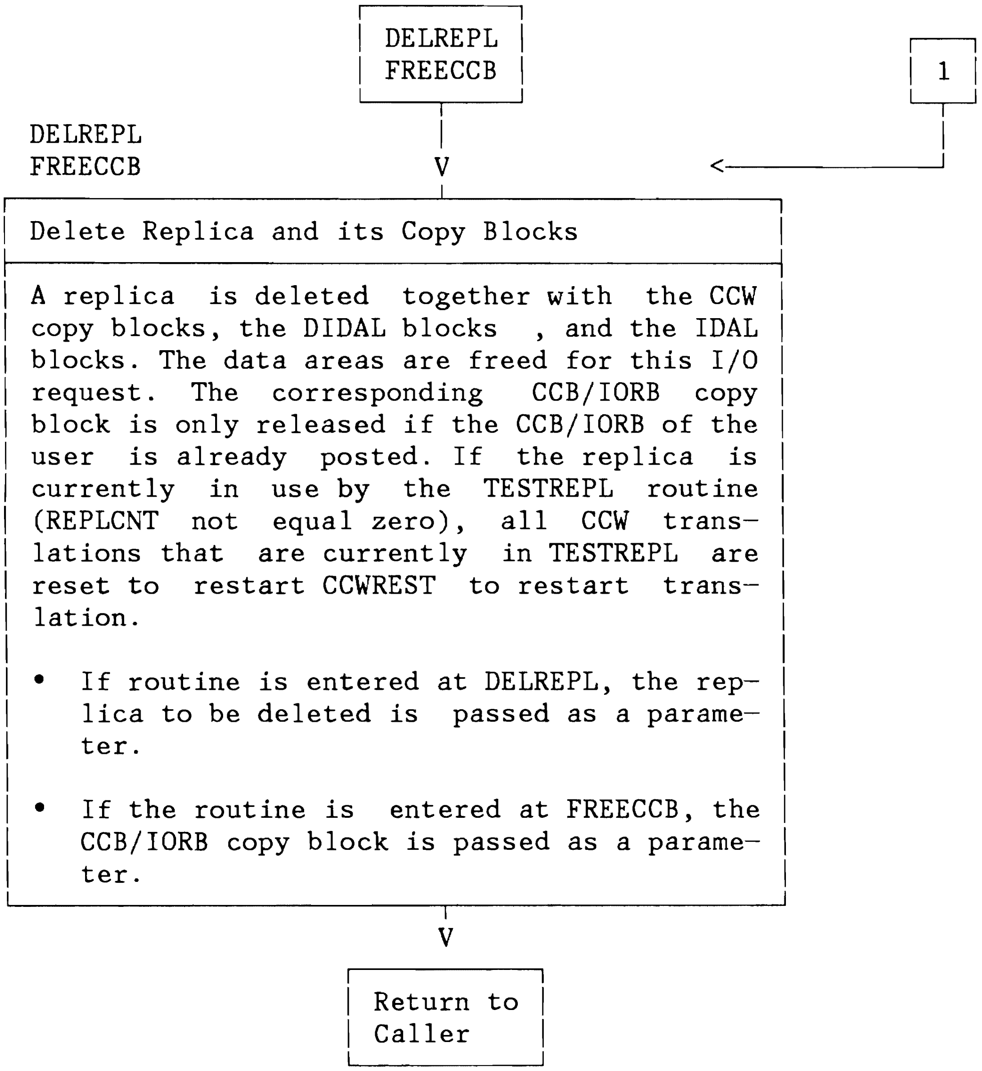
CREATREP

Creation of Replica

A replica is created by saving the virtual CCB/IORB and the virtual channel program in the replica blocks. The replica is enqueued at the beginning of the replica queue of the requestor's partition. Parts of the replica queue that are obsolete are deleted.

Return to
Caller

Figure 159 (Part 2 of 3). Channel Program Translation (370 Mode): Fast CCW Translation



DELREPL
FREECCB

1

DELREPL
FREECCB

Delete Replica and its Copy Blocks

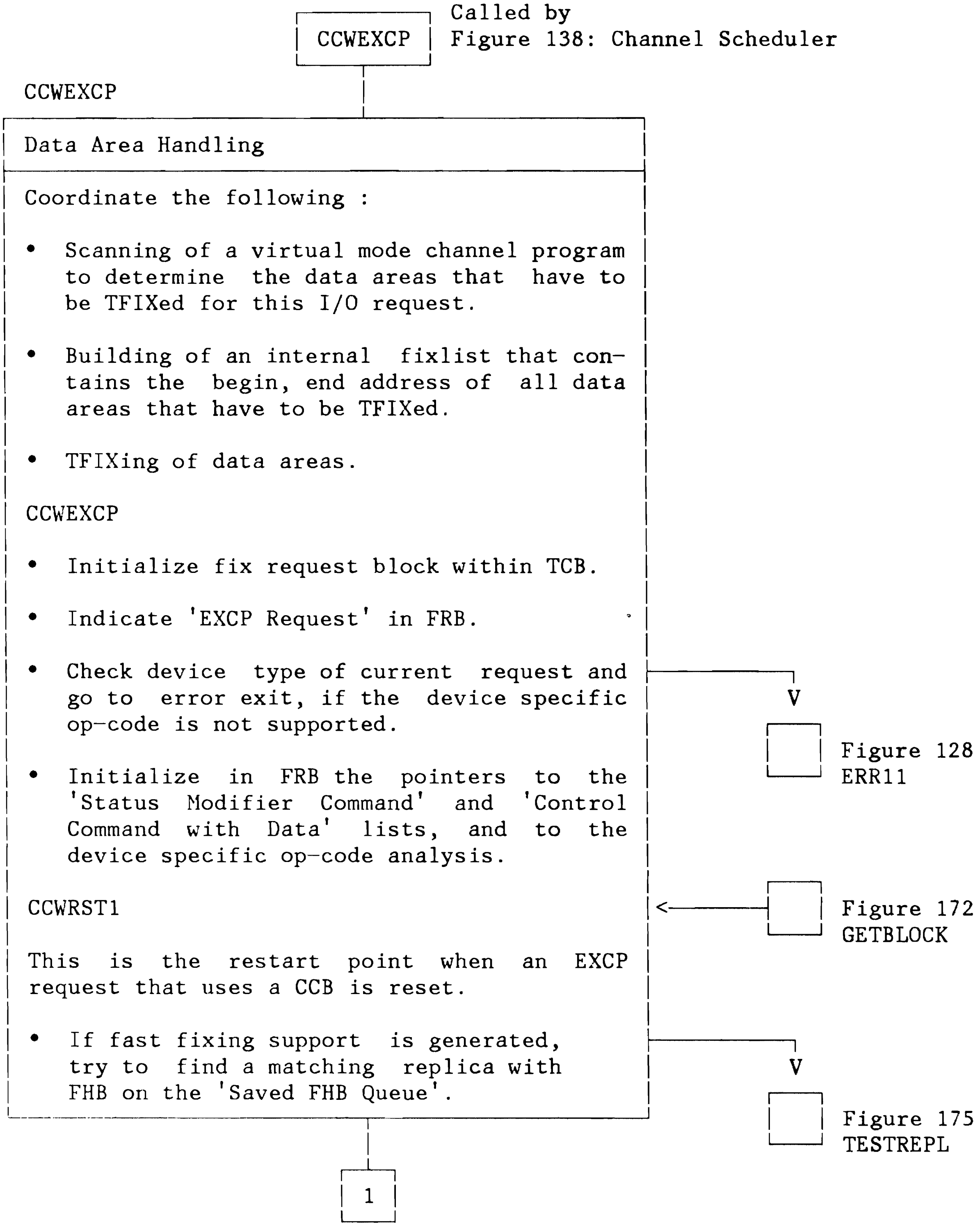
A replica is deleted together with the CCW copy blocks, the DIDAL blocks , and the IDAL blocks. The data areas are freed for this I/O request. The corresponding CCB/IORB copy block is only released if the CCB/IORB of the user is already posted. If the replica is currently in use by the TESTREPL routine (REPLCNT not equal zero), all CCW translations that are currently in TESTREPL are reset to restart CCWREST to restart translation.

- If routine is entered at DELREPL, the replica to be deleted is passed as a parameter.
- If the routine is entered at FREECCB, the CCB/IORB copy block is passed as a parameter.

Return to Caller

Figure 159 (Part 3 of 3). Channel Program Translation (370 Mode): Fast CCW Translation

CHANNEL PROGRAM FIXING ROUTINES



CCWEXCP

Called by
Figure 138: Channel Scheduler

CCWEXCP

Data Area Handling

Coordinate the following :

- Scanning of a virtual mode channel program to determine the data areas that have to be TFIXed for this I/O request.
- Building of an internal fixlist that contains the begin, end address of all data areas that have to be TFIXed.
- TFIXing of data areas.

CCWEXCP

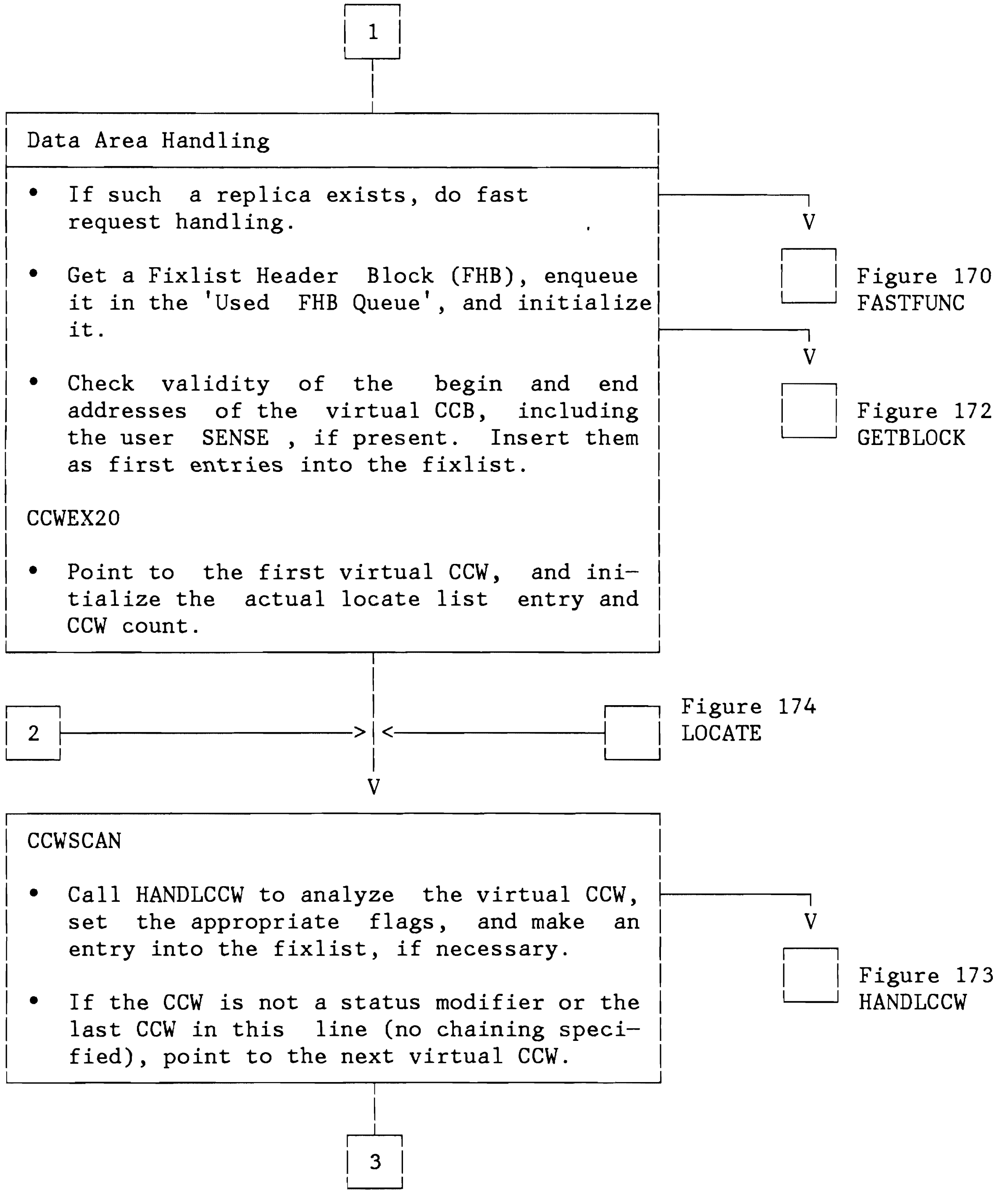
- Initialize fix request block within TCB.
- Indicate 'EXCP Request' in FRB.
- Check device type of current request and go to error exit, if the device specific op-code is not supported.
- Initialize in FRB the pointers to the 'Status Modifier Command' and 'Control Command with Data' lists, and to the device specific op-code analysis.

CCWRST1

This is the restart point when an EXCP request that uses a CCB is reset.

- If fast fixing support is generated, try to find a matching replica with FHB on the 'Saved FHB Queue'.

Figure 160 (Part 1 of 4). Channel Program Fixing (ECPS:VSE Mode): CCWEXCP Routine



Data Area Handling

- If such a replica exists, do fast request handling.
- Get a Fixlist Header Block (FHB), enqueue it in the 'Used FHB Queue', and initialize it.
- Check validity of the begin and end addresses of the virtual CCB, including the user SENSE , if present. Insert them as first entries into the fixlist.

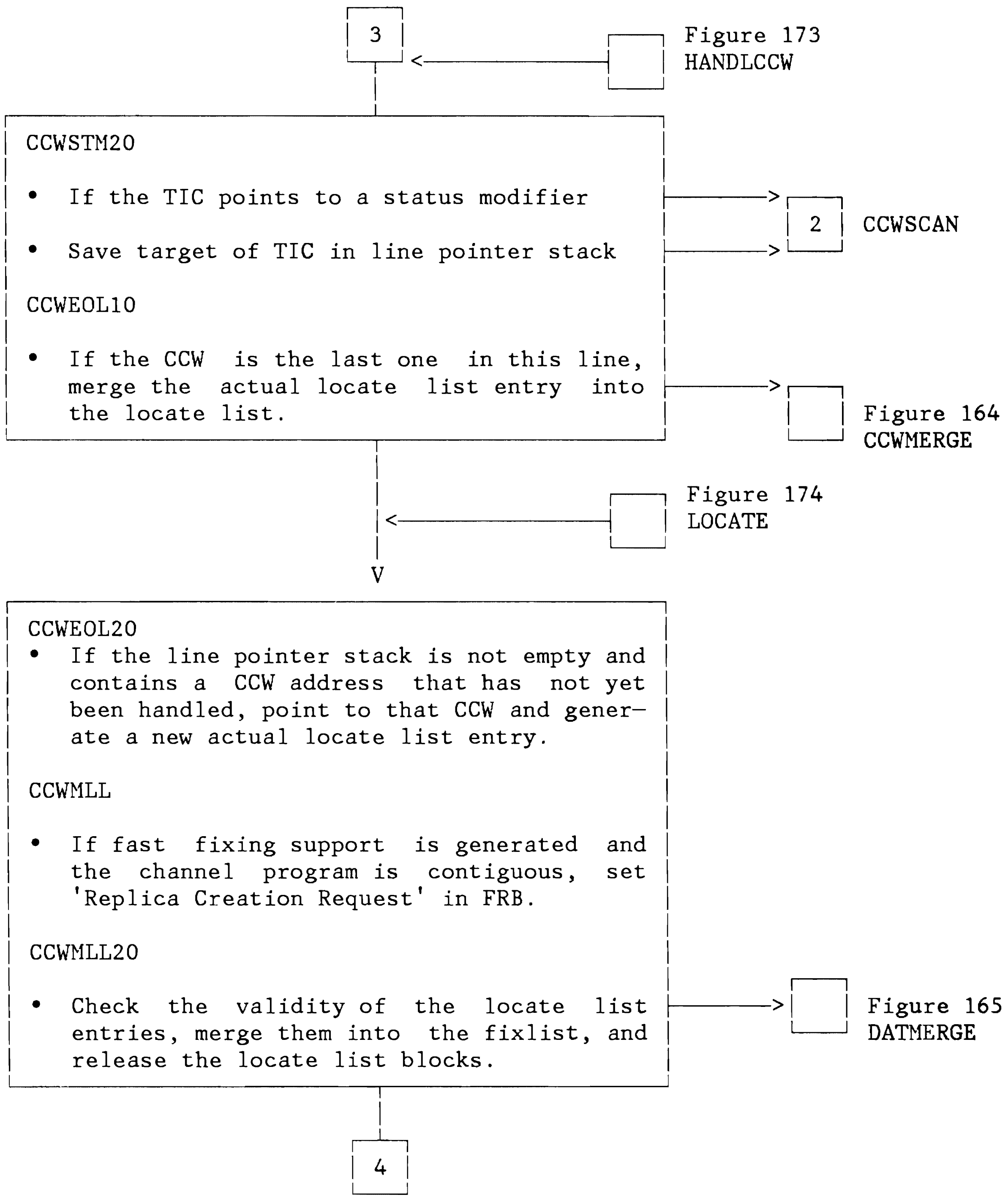
CCWEX20

- Point to the first virtual CCW, and initialize the actual locate list entry and CCW count.

CCWSCAN

- Call HANDLCCW to analyze the virtual CCW, set the appropriate flags, and make an entry into the fixlist, if necessary.
- If the CCW is not a status modifier or the last CCW in this line (no chaining specified), point to the next virtual CCW.

Figure 160 (Part 2 of 4). Channel Program Fixing (ECPS:VSE Mode): CCWEXCP Routine



CCWSTM20

- If the TIC points to a status modifier
- Save target of TIC in line pointer stack

CCWEOL10

- If the CCW is the last one in this line, merge the actual locate list entry into the locate list.

CCWEOL20

- If the line pointer stack is not empty and contains a CCW address that has not yet been handled, point to that CCW and generate a new actual locate list entry.

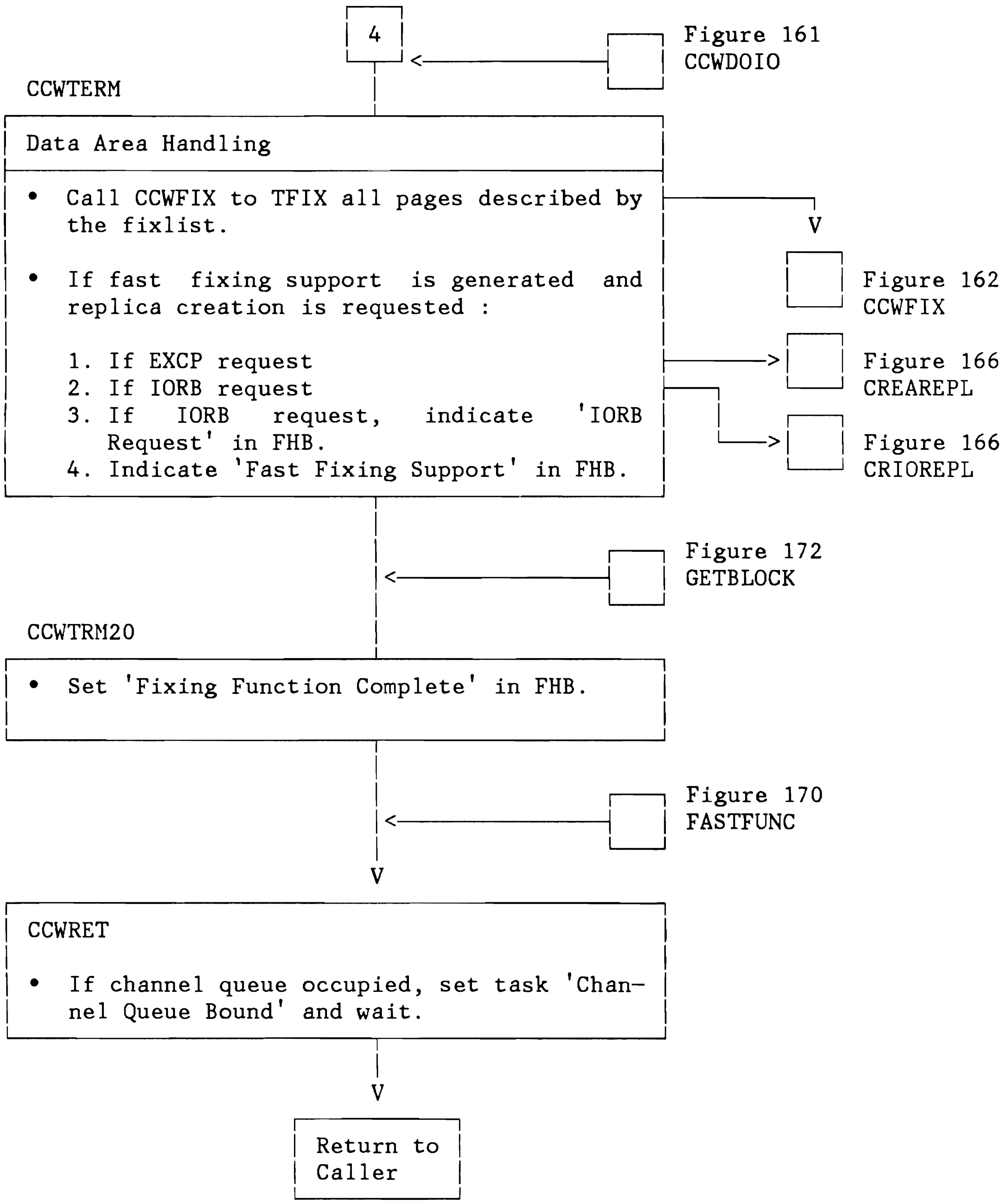
CCWMLL

- If fast fixing support is generated and the channel program is contiguous, set 'Replica Creation Request' in FRB.

CCWMLL20

- Check the validity of the locate list entries, merge them into the fixlist, and release the locate list blocks.

Figure 160 (Part 3 of 4). Channel Program Fixing (ECPS:VSE Mode): CCWEXCP Routine



Data Area Handling

- Call CCWFIX to TFIX all pages described by the fixlist.
- If fast fixing support is generated and replica creation is requested :

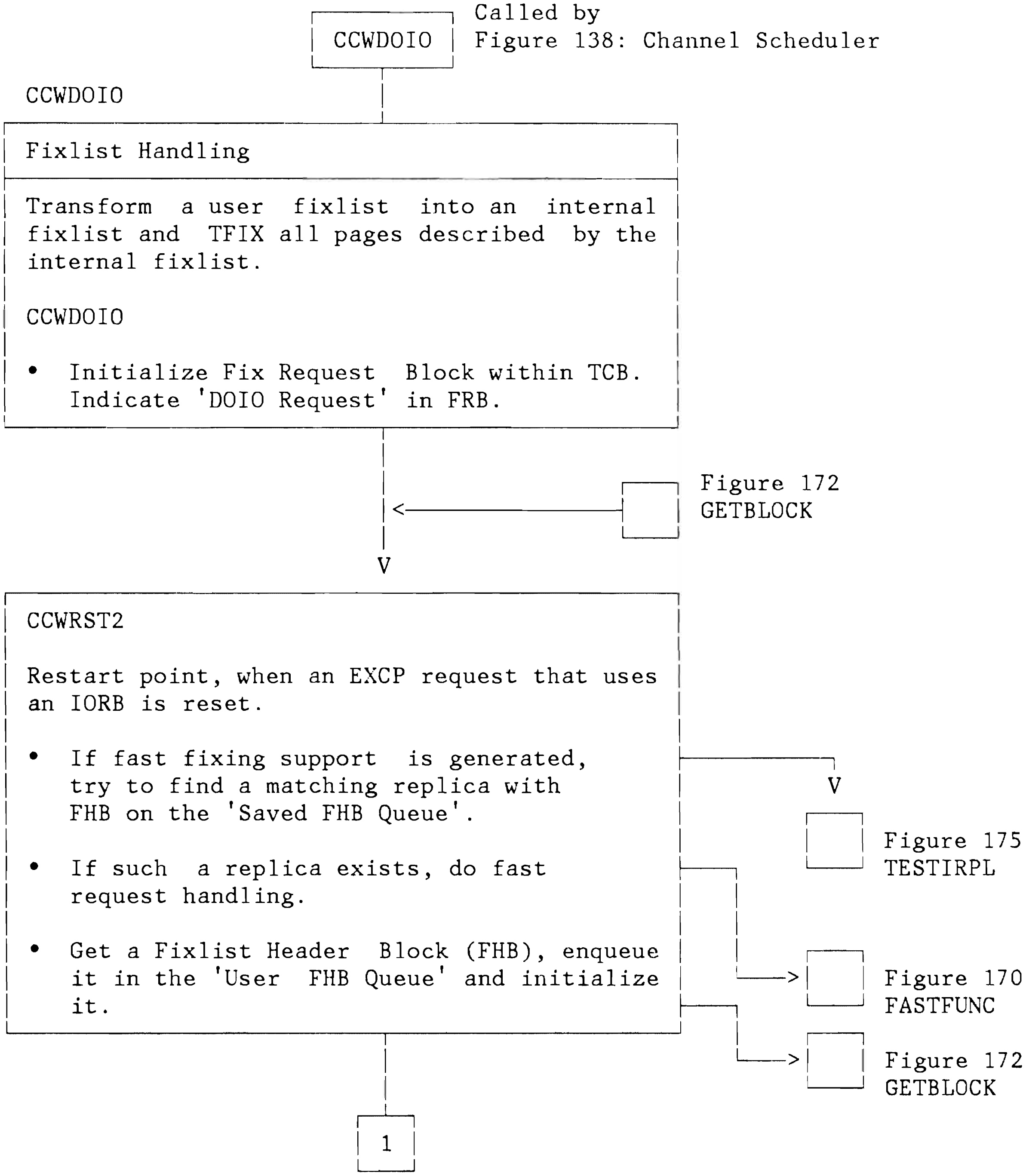
  1. If EXCP request
  2. If IORB request
  3. If IORB request, indicate 'IORB Request' in FHB.
  4. Indicate 'Fast Fixing Support' in FHB.

- Set 'Fixing Function Complete' in FHB.

CCWRET

- If channel queue occupied, set task 'Channel Queue Bound' and wait.

Figure 160 (Part 4 of 4). Channel Program Fixing (ECPS:VSE Mode): CCWEXCP Routine



Fixlist Handling

Transform a user fixlist into an internal fixlist and TFIX all pages described by the internal fixlist.

CCWDOIO

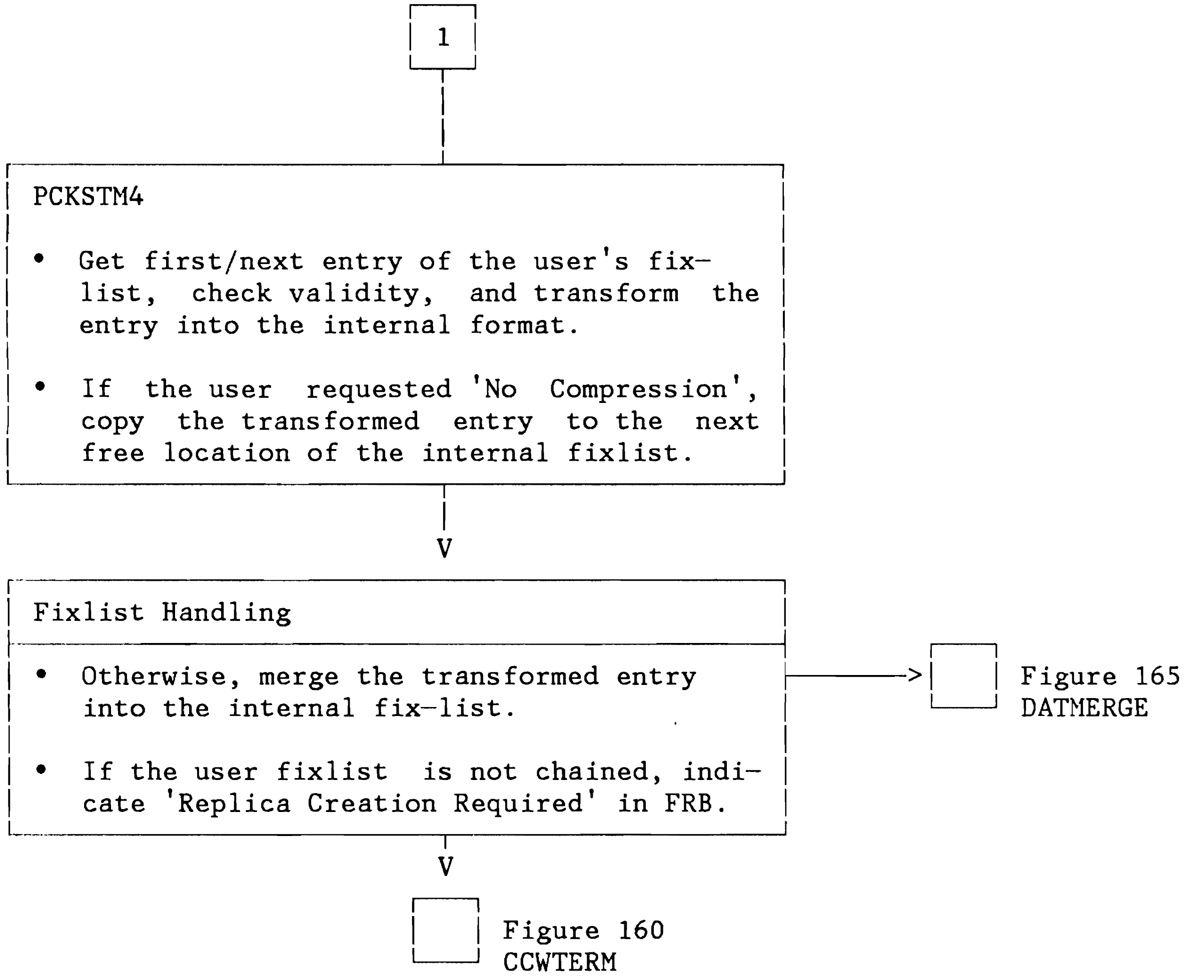
- Initialize Fix Request Block within TCB. Indicate 'DOIO Request' in FRB.

CCWRST2

Restart point, when an EXCP request that uses an IORB is reset.

- If fast fixing support is generated, try to find a matching replica with FHB on the 'Saved FHB Queue'.
- If such a replica exists, do fast request handling.
- Get a Fixlist Header Block (FHB), enqueue it in the 'User FHB Queue' and initialize it.

Figure 161 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWDOIO Routine



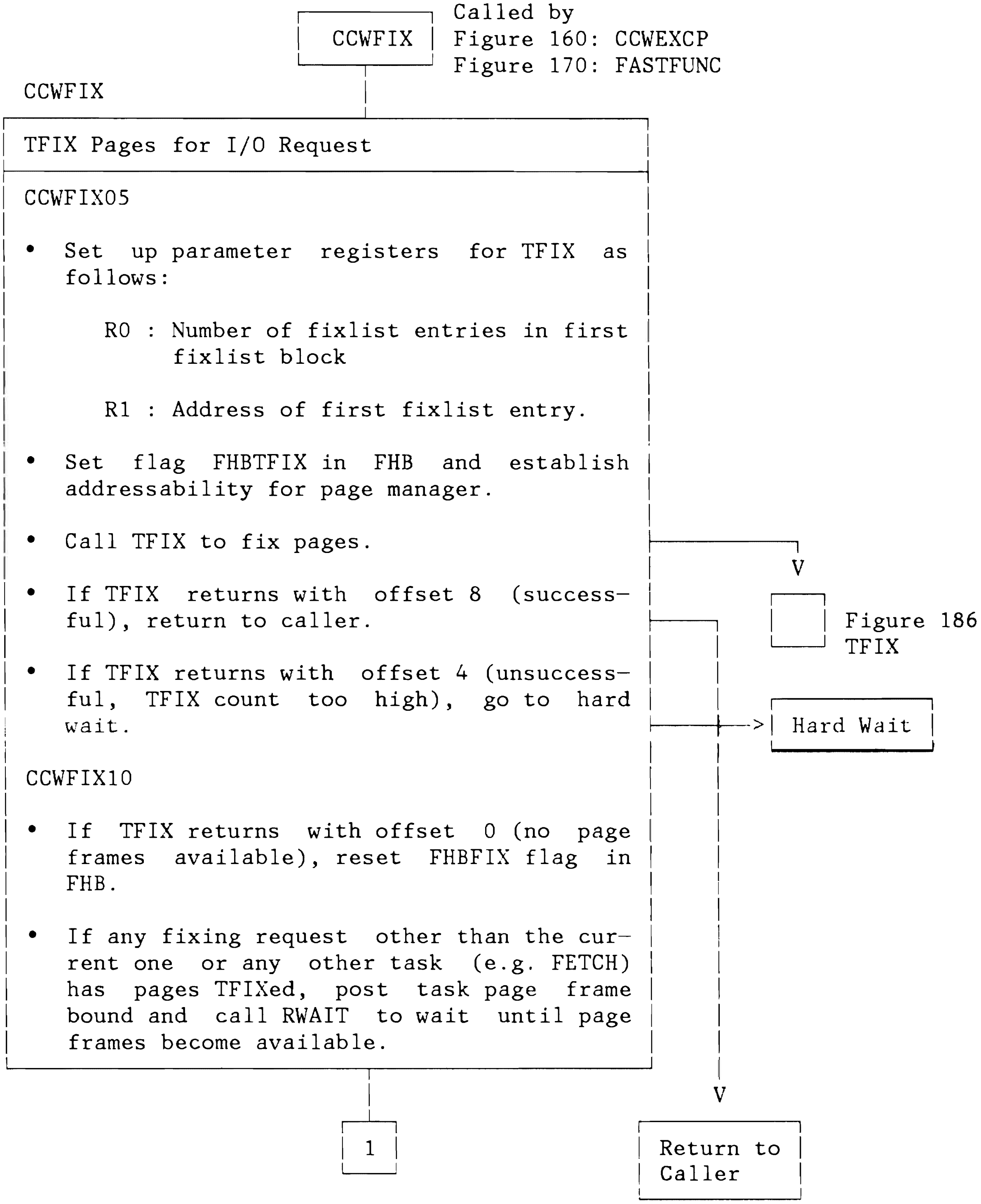
PCKSTM4

- Get first/next entry of the user's fix-list, check validity, and transform the entry into the internal format.
- If the user requested 'No Compression', copy the transformed entry to the next free location of the internal fixlist.

Fixlist Handling

- Otherwise, merge the transformed entry into the internal fix-list.
- If the user fixlist is not chained, indicate 'Replica Creation Required' in FRB.

Figure 161 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWDOIO Routine



CCWFIX

Called by
Figure 160: CCWEXCP
Figure 170: FASTFUNC

CCWFIX

TFIX Pages for I/O Request

CCWFIX05

- Set up parameter registers for TFIX as follows:

  R0 : Number of fixlist entries in first fixlist block

  R1 : Address of first fixlist entry.

- Set flag FHBTFIX in FHB and establish addressability for page manager.

- Call TFIX to fix pages.

- If TFIX returns with offset 8 (successful), return to caller.

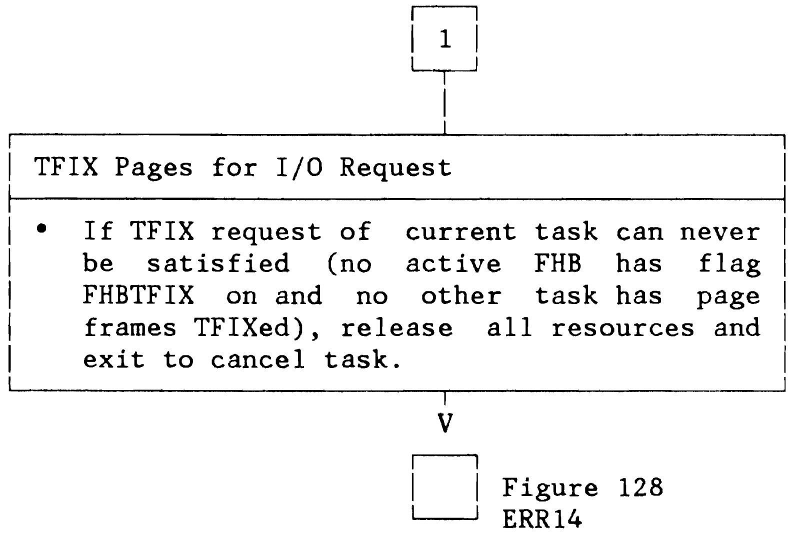
- If TFIX returns with offset 4 (unsuccessful, TFIX count too high), go to hard wait.

CCWFIX10

- If TFIX returns with offset 0 (no page frames available), reset FHBFIX flag in FHB.

- If any fixing request other than the current one or any other task (e.g. FETCH) has pages TFIXed, post task page frame bound and call RWAIT to wait until page frames become available.

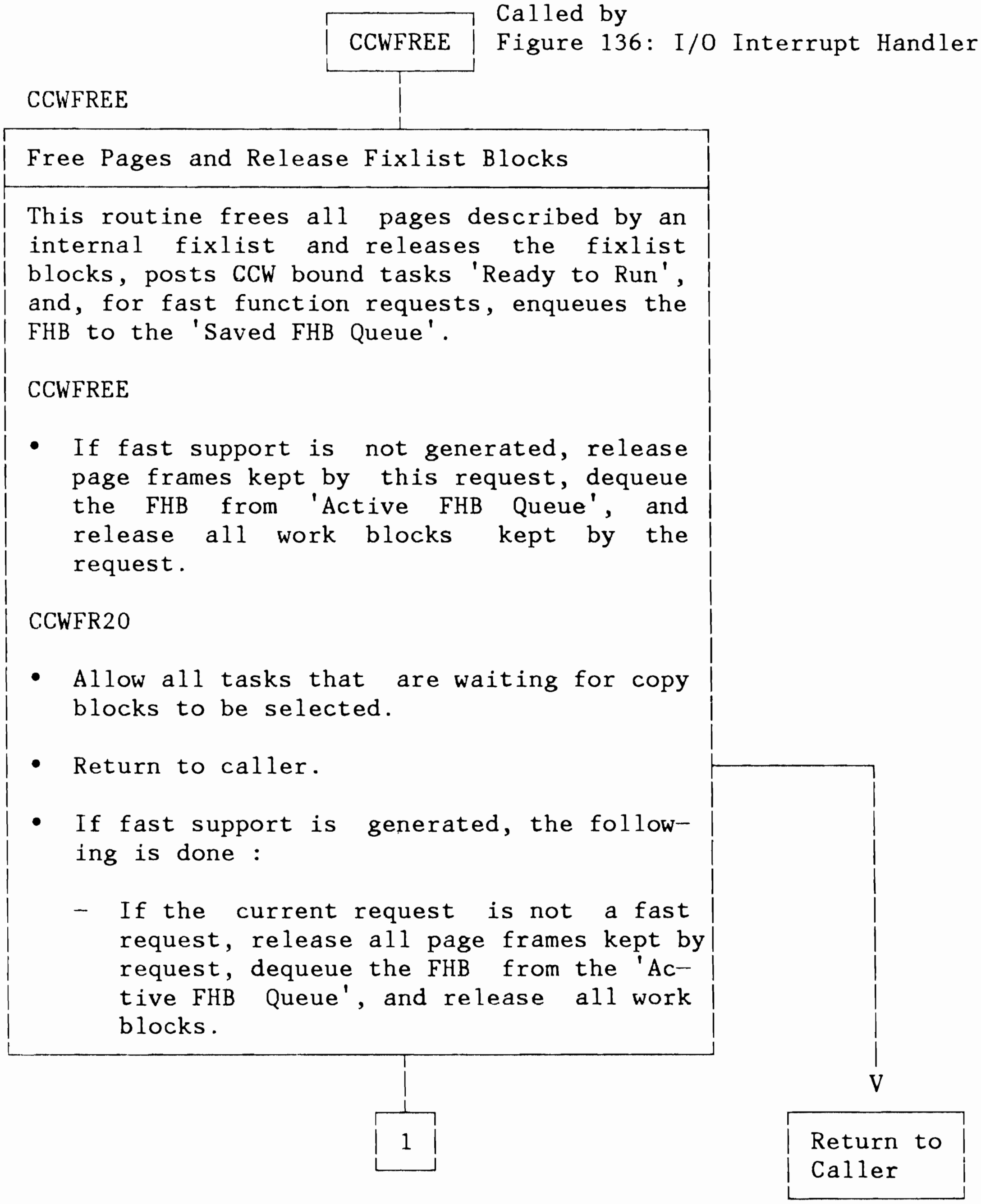
Figure 162 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWFIX Routine



TFIX Pages for I/O Request

- If TFIX request of current task can never be satisfied (no active FHB has flag FHBTFIX on and no other task has page frames TFIXed), release all resources and exit to cancel task.

Figure 162 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWFIX Routine



CCWFREE

Called by
Figure 136: I/O Interrupt Handler

CCWFREE

Free Pages and Release Fixlist Blocks

This routine frees all pages described by an internal fixlist and releases the fixlist blocks, posts CCW bound tasks 'Ready to Run', and, for fast function requests, enqueues the FHB to the 'Saved FHB Queue'.

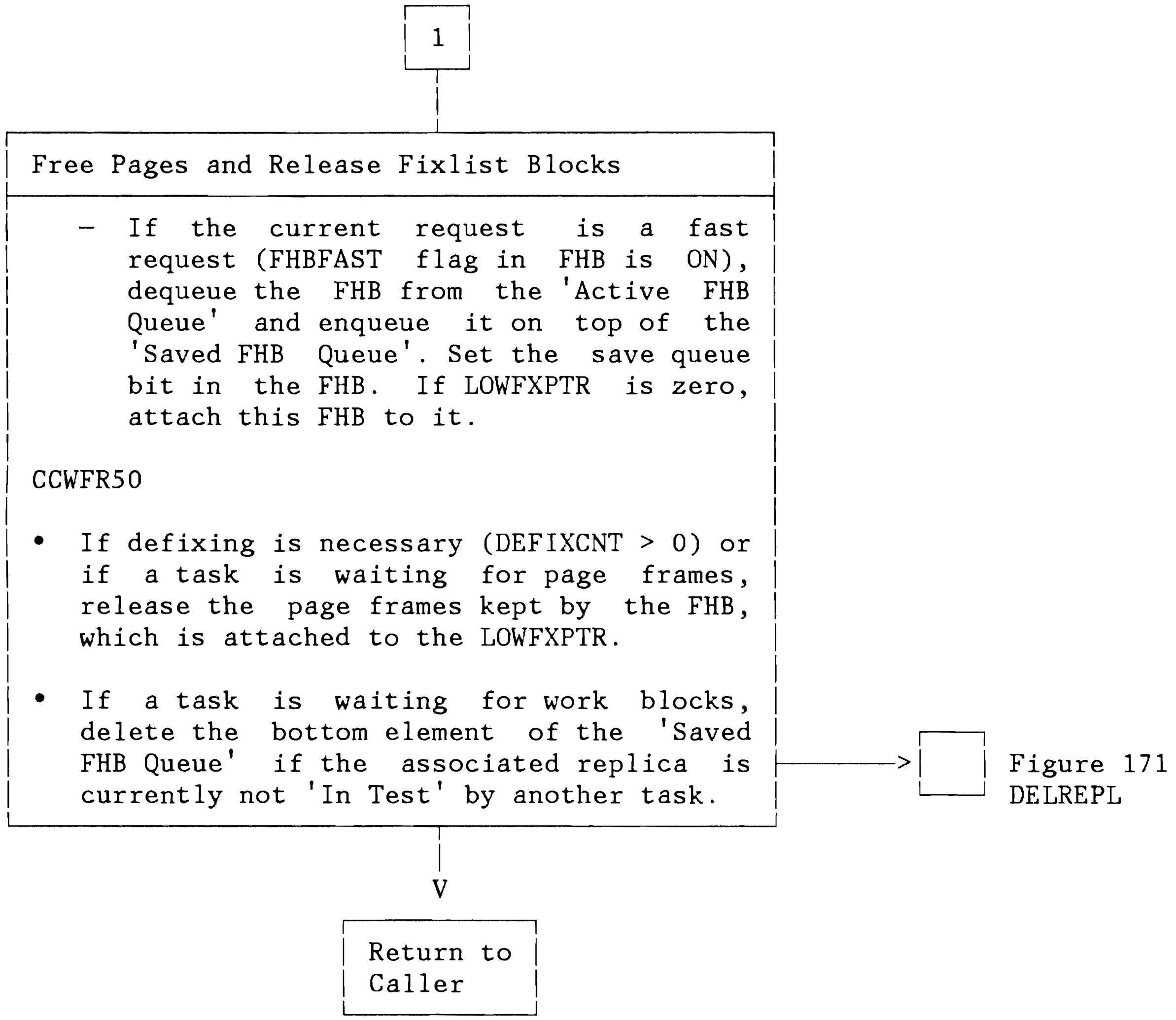
CCWFREE

- If fast support is not generated, release page frames kept by this request, dequeue the FHB from 'Active FHB Queue', and release all work blocks kept by the request.

CCWFR20

- Allow all tasks that are waiting for copy blocks to be selected.
- Return to caller.
- If fast support is generated, the following is done :
  - If the current request is not a fast request, release all page frames kept by request, dequeue the FHB from the 'Active FHB Queue', and release all work blocks.

Figure 163 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWFREE Routine



Free Pages and Release Fixlist Blocks

- If the current request is a fast request (FHBFAST flag in FHB is ON), dequeue the FHB from the 'Active FHB Queue' and enqueue it on top of the 'Saved FHB Queue'. Set the save queue bit in the FHB. If LOWFXPTR is zero, attach this FHB to it.

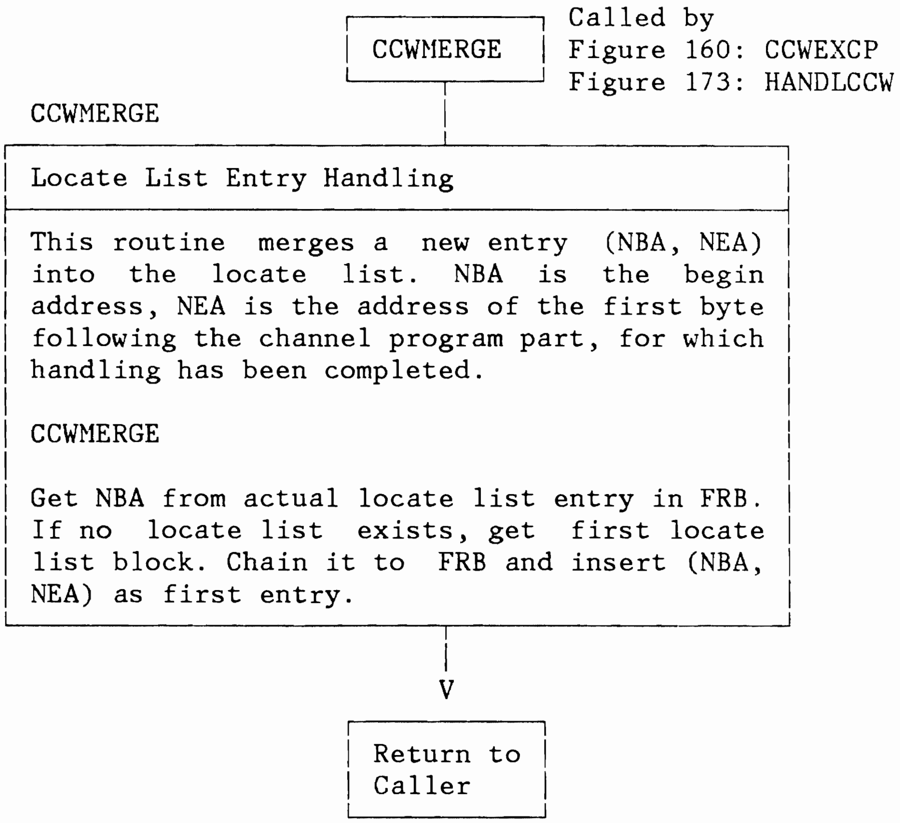
CCWFR50

- If defixing is necessary (DEFIXCNT > 0) or if a task is waiting for page frames, release the page frames kept by the FHB, which is attached to the LOWFXPTR.

- If a task is waiting for work blocks, delete the bottom element of the 'Saved FHB Queue' if the associated replica is currently not 'In Test' by another task.

Return to Caller

Figure 163 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): CCWFREE Routine



CCWMERGE

Called by
Figure 160: CCWEXCP
Figure 173: HANDLCCW

CCWMERGE

Locate List Entry Handling

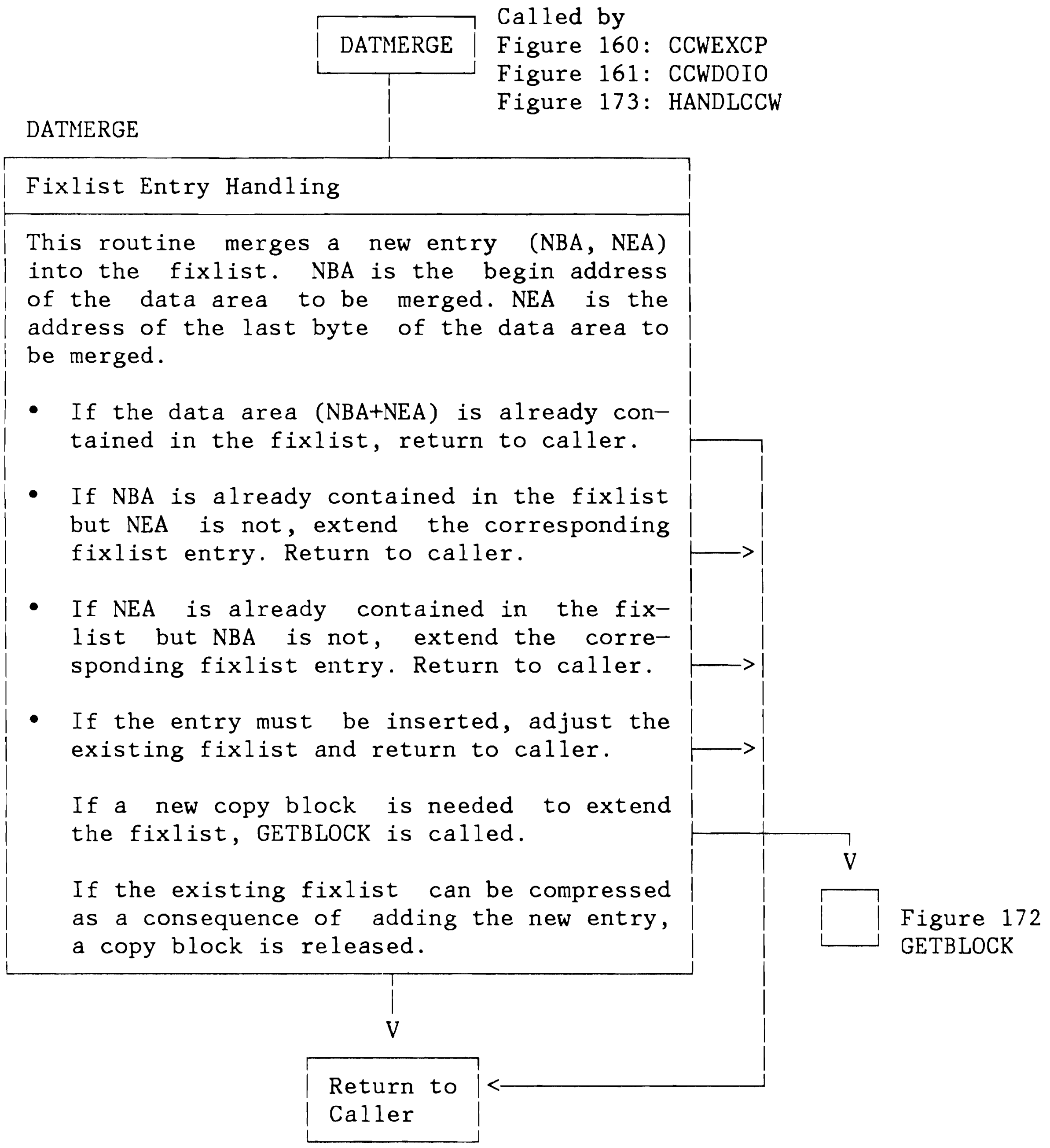
This routine merges a new entry (NBA, NEA) into the locate list. NBA is the begin address, NEA is the address of the first byte following the channel program part, for which handling has been completed.

CCWMERGE

Get NBA from actual locate list entry in FRB. If no locate list exists, get first locate list block. Chain it to FRB and insert (NBA, NEA) as first entry.

Return to Caller

Figure 164. Channel Program Fixing (ECPS:VSE Mode): CCWMERGE Routine



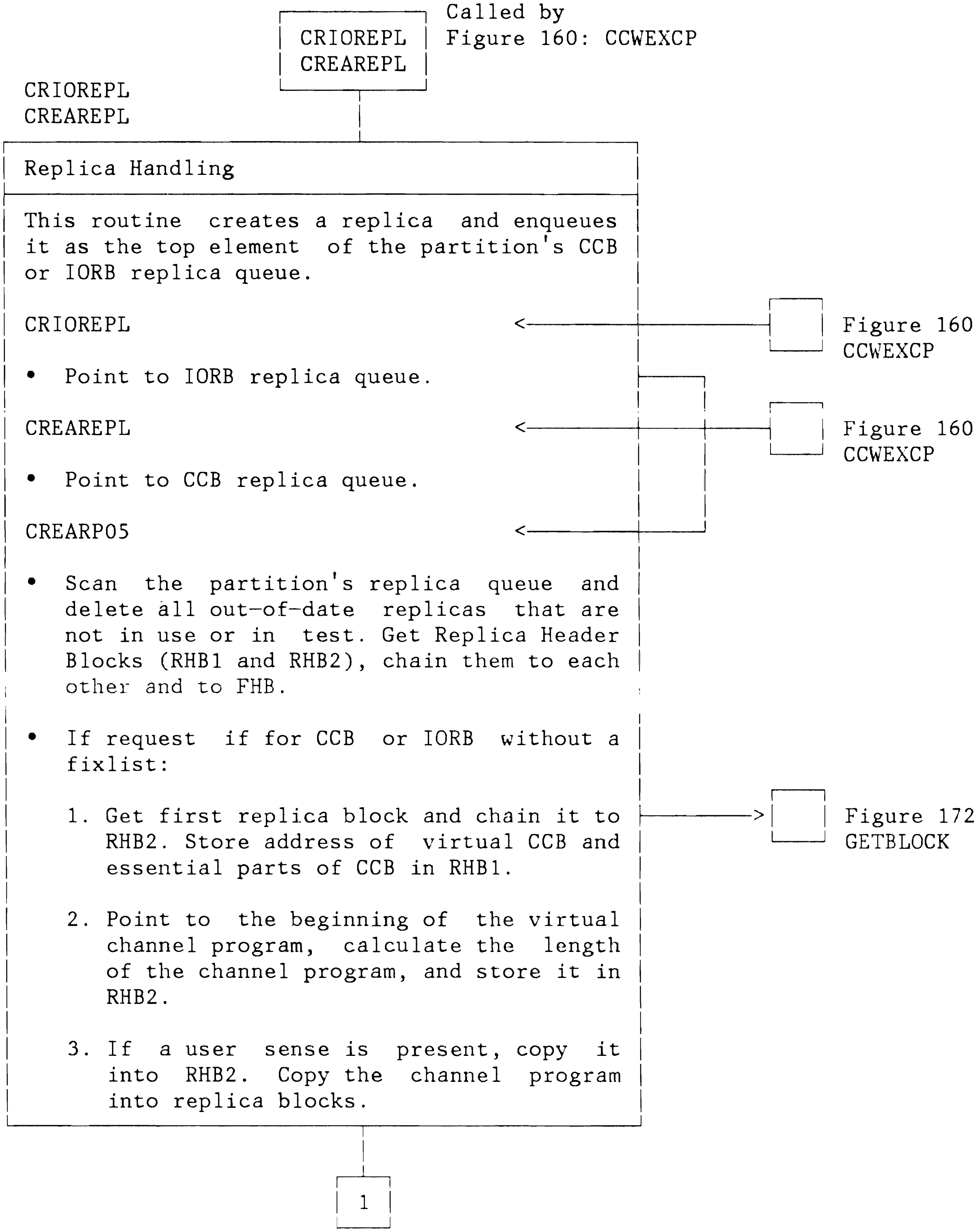
This routine merges a new entry (NBA, NEA) into the fixlist. NBA is the begin address of the data area to be merged. NEA is the address of the last byte of the data area to be merged.

- If the data area (NBA+NEA) is already contained in the fixlist, return to caller.
- If NBA is already contained in the fixlist but NEA is not, extend the corresponding fixlist entry. Return to caller.
- If NEA is already contained in the fixlist but NBA is not, extend the corresponding fixlist entry. Return to caller.
- If the entry must be inserted, adjust the existing fixlist and return to caller.

  If a new copy block is needed to extend the fixlist, GETBLOCK is called.

  If the existing fixlist can be compressed as a consequence of adding the new entry, a copy block is released.

Figure 165. Channel Program Fixing (ECPS:VSE Mode): DATMERGE Routine



CRIOREPL
CREAREPL

Called by
Figure 160: CCWEXCP

CRIOREPL
CREAREPL

Replica Handling

This routine creates a replica and enqueues it as the top element of the partition's CCB or IORB replica queue.

CRIOREPL

- Point to IORB replica queue.

CREAREPL

- Point to CCB replica queue.

CREARP05

- Scan the partition's replica queue and delete all out-of-date replicas that are not in use or in test. Get Replica Header Blocks (RHB1 and RHB2), chain them to each other and to FHB.

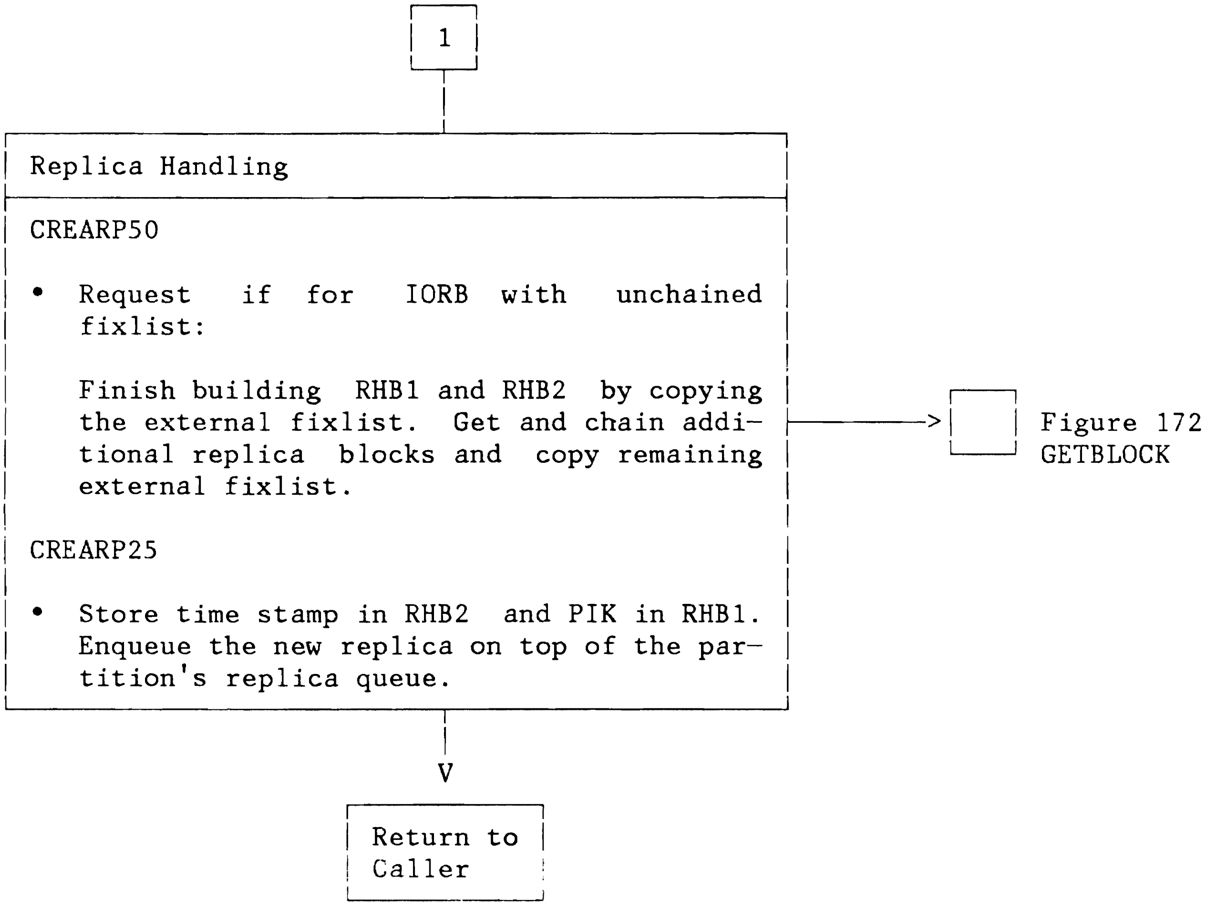
- If request if for CCB or IORB without a fixlist:

  1. Get first replica block and chain it to RHB2. Store address of virtual CCB and essential parts of CCB in RHB1.

  2. Point to the beginning of the virtual channel program, calculate the length of the channel program, and store it in RHB2.

  3. If a user sense is present, copy it into RHB2. Copy the channel program into replica blocks.

Figure 166 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): CREAREPL Routine



Replica Handling

CREARP50

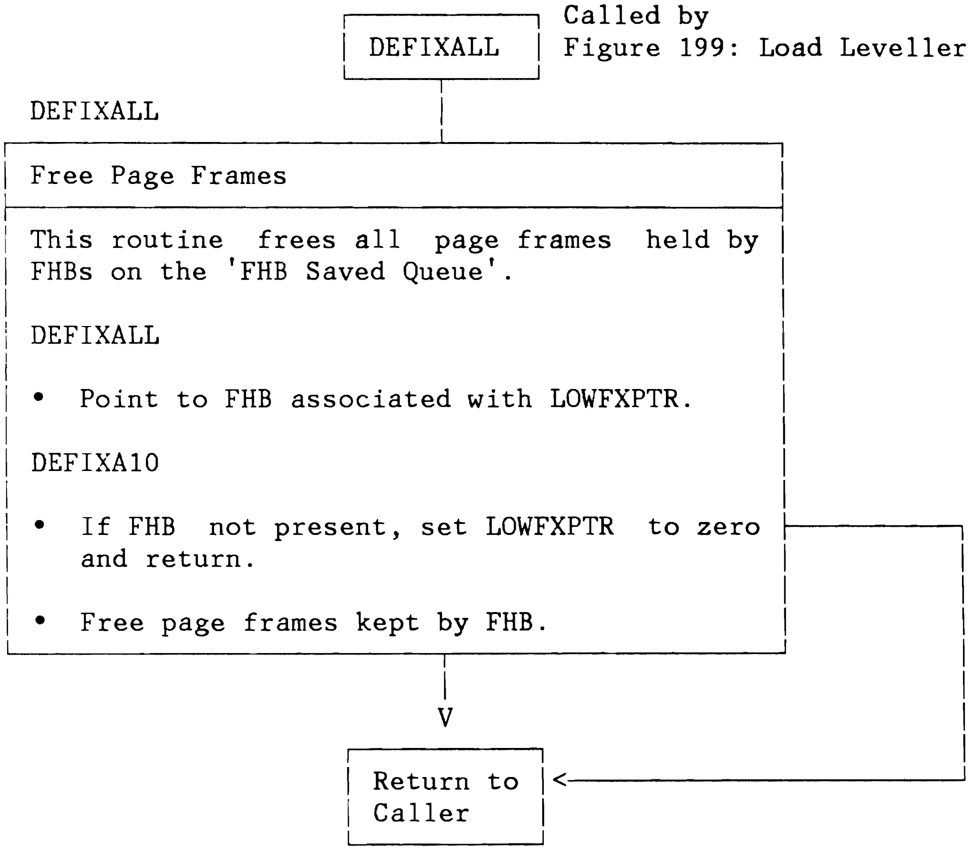
- Request if for IORB with unchained fixlist:

  Finish building RHB1 and RHB2 by copying the external fixlist. Get and chain additional replica blocks and copy remaining external fixlist.

CREARP25

- Store time stamp in RHB2 and PIK in RHB1. Enqueue the new replica on top of the partition's replica queue.

Figure 166 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): CREAREPL Routine



This routine frees all page frames held by FHBs on the 'FHB Saved Queue'.

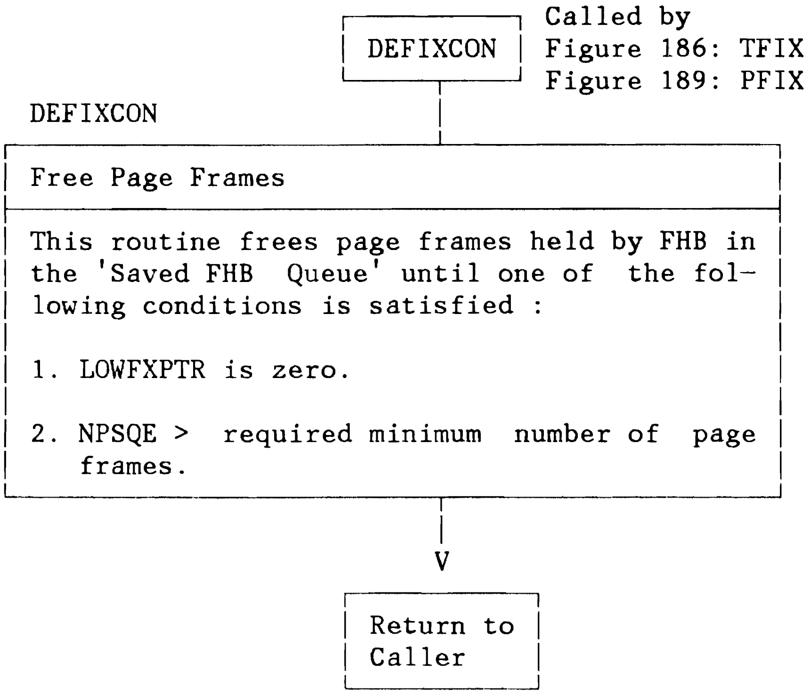
DEFIXALL

- Point to FHB associated with LOWFXPTR.

DEFIXA10

- If FHB not present, set LOWFXPTR to zero and return.
- Free page frames kept by FHB.

Figure 167. Channel Program Fixing (ECPS:VSE Mode): DEFIXALL Routine



DEFIXCON

Called by
Figure 186: TFIX
Figure 189: PFIX

DEFIXCON

Free Page Frames

This routine frees page frames held by FHB in the 'Saved FHB Queue' until one of the following conditions is satisfied :

1. LOWFXPTR is zero.

2. NPSQE > required minimum number of page frames.

Return to Caller

Figure 168. Channel Program Fixing (ECPS:VSE Mode): DEFIXCON Routine

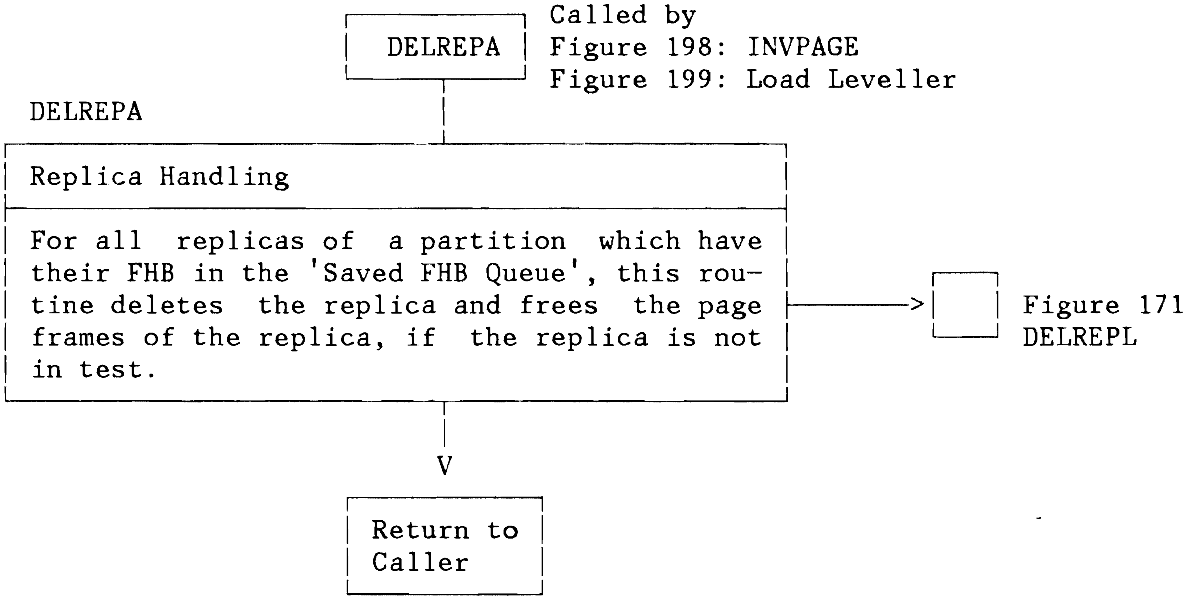
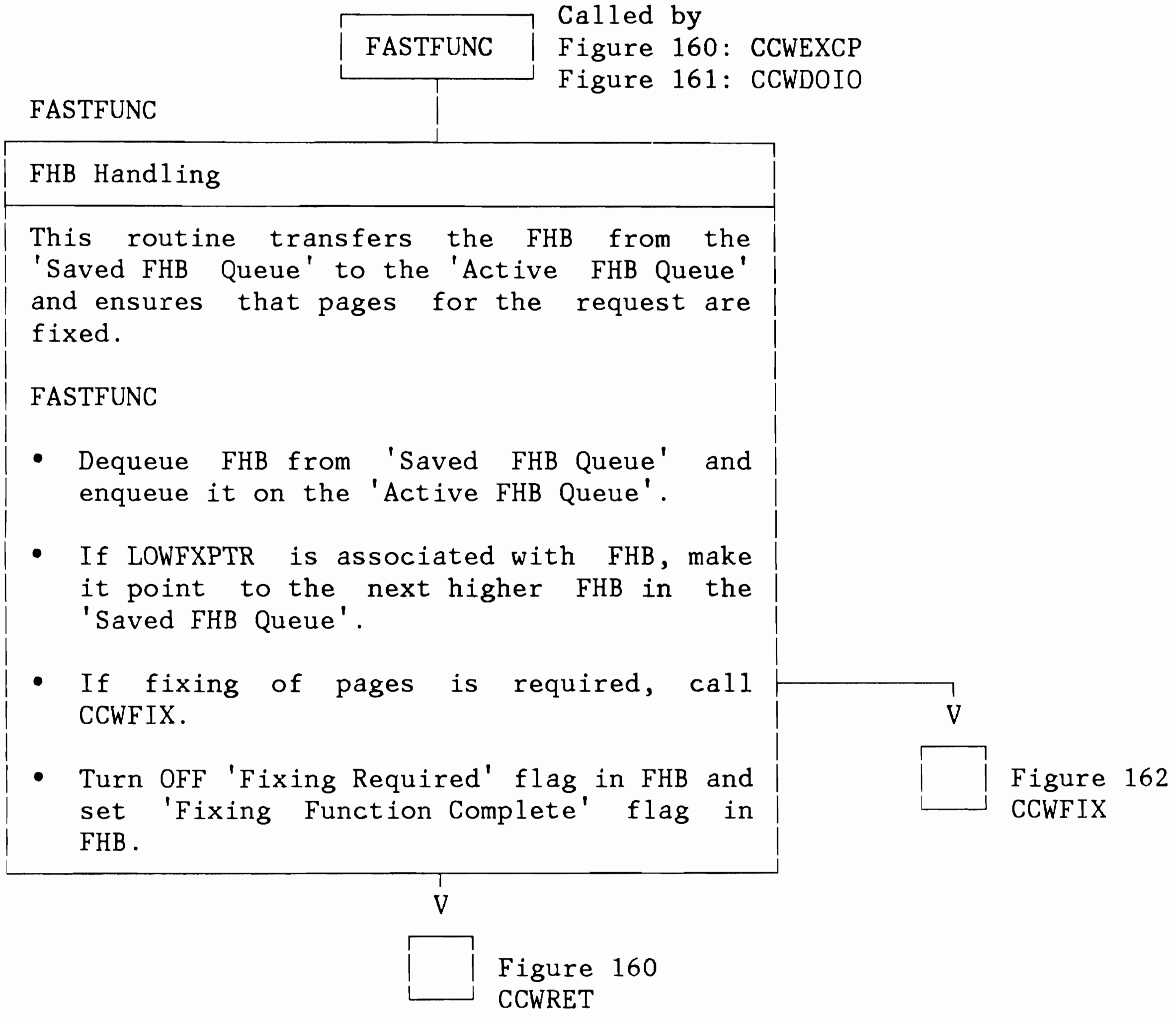

DELREPA
Called by
Figure 198: INVPAGE
Figure 199: Load Leveller
DELREPA
Replica Handling
For all replicas of a partition which have their FHB in the 'Saved FHB Queue', this routine deletes the replica and frees the page frames of the replica, if the replica is not in test.
Figure 171
DELREPL
V
Return to Caller

Figure 169. Channel Program Fixing (ECPS:VSE Mode): DELREPA Routine



FASTFUNC

Called by
Figure 160: CCWEXCP
Figure 161: CCWDOIO

FASTFUNC

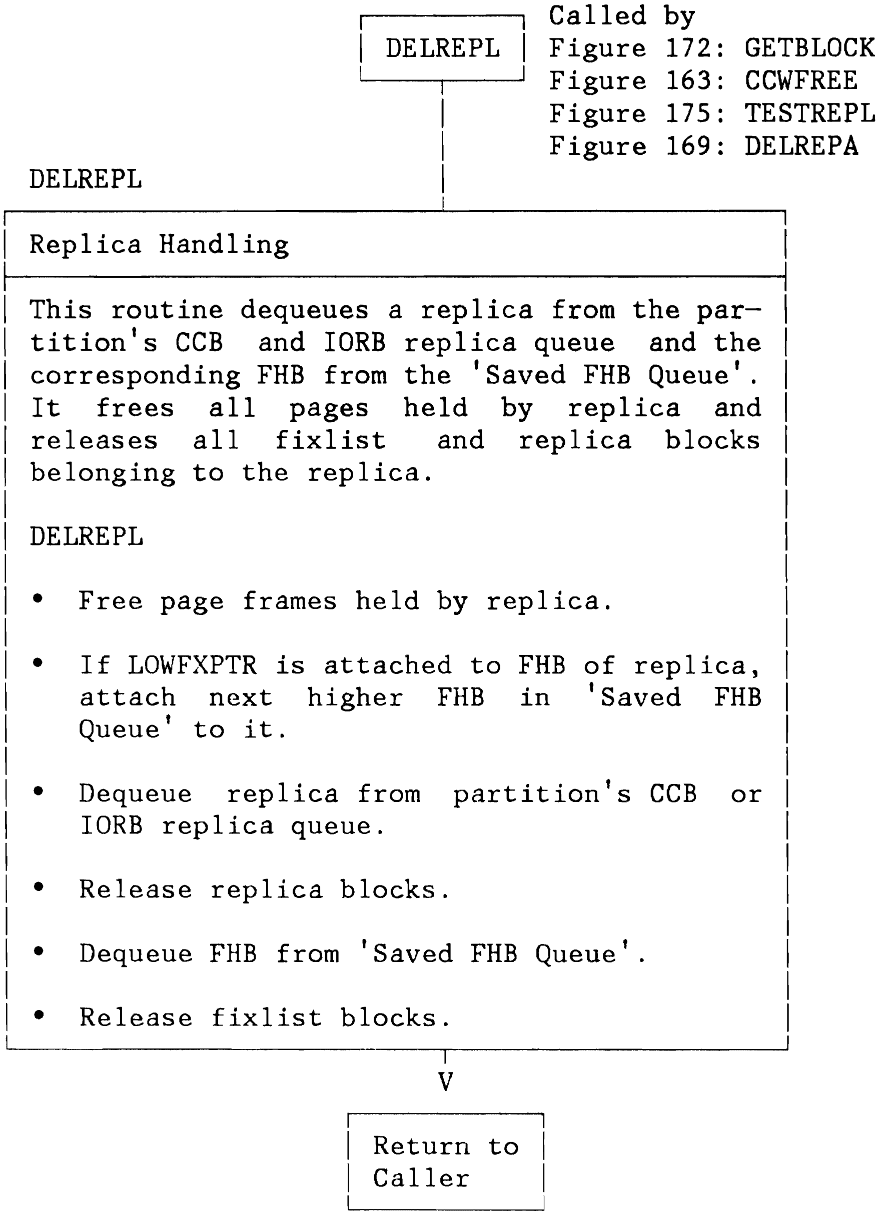
FHB Handling

This routine transfers the FHB from the 'Saved FHB Queue' to the 'Active FHB Queue' and ensures that pages for the request are fixed.

FASTFUNC

- Dequeue FHB from 'Saved FHB Queue' and enqueue it on the 'Active FHB Queue'.
- If LOWFXPTR is associated with FHB, make it point to the next higher FHB in the 'Saved FHB Queue'.
- If fixing of pages is required, call CCWFIX.
- Turn OFF 'Fixing Required' flag in FHB and set 'Fixing Function Complete' flag in FHB.

Figure 170. Channel Program Fixing (ECPS:VSE Mode): FASTFUNC Routine



DELREPL

Called by
Figure 172: GETBLOCK
Figure 163: CCWFREE
Figure 175: TESTREPL
Figure 169: DELREPA

DELREPL

Replica Handling

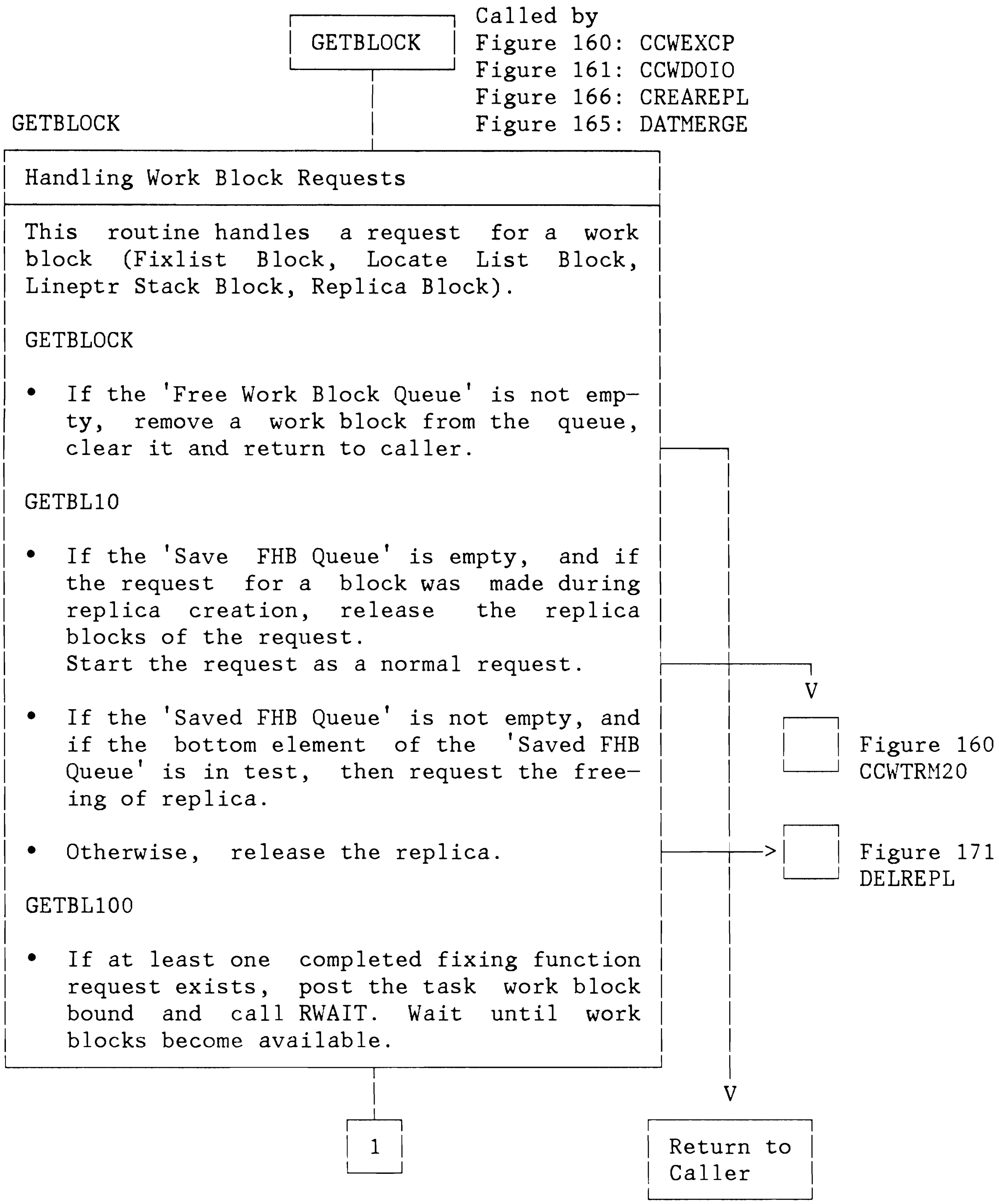
This routine dequeues a replica from the partition's CCB and IORB replica queue and the corresponding FHB from the 'Saved FHB Queue'. It frees all pages held by replica and releases all fixlist and replica blocks belonging to the replica.

DELREPL

- Free page frames held by replica.
- If LOWFXPTR is attached to FHB of replica, attach next higher FHB in 'Saved FHB Queue' to it.
- Dequeue replica from partition's CCB or IORB replica queue.
- Release replica blocks.
- Dequeue FHB from 'Saved FHB Queue'.
- Release fixlist blocks.

Return to Caller

Figure 171. Channel Program Fixing (ECPS:VSE Mode): DELREPL Routine



GETBLOCK

Called by
Figure 160: CCWEXCP
Figure 161: CCWDOIO
Figure 166: CREAREPL
Figure 165: DATMERGE

GETBLOCK

Handling Work Block Requests

This routine handles a request for a work block (Fixlist Block, Locate List Block, Lineptr Stack Block, Replica Block).

GETBLOCK

- If the 'Free Work Block Queue' is not empty, remove a work block from the queue, clear it and return to caller.

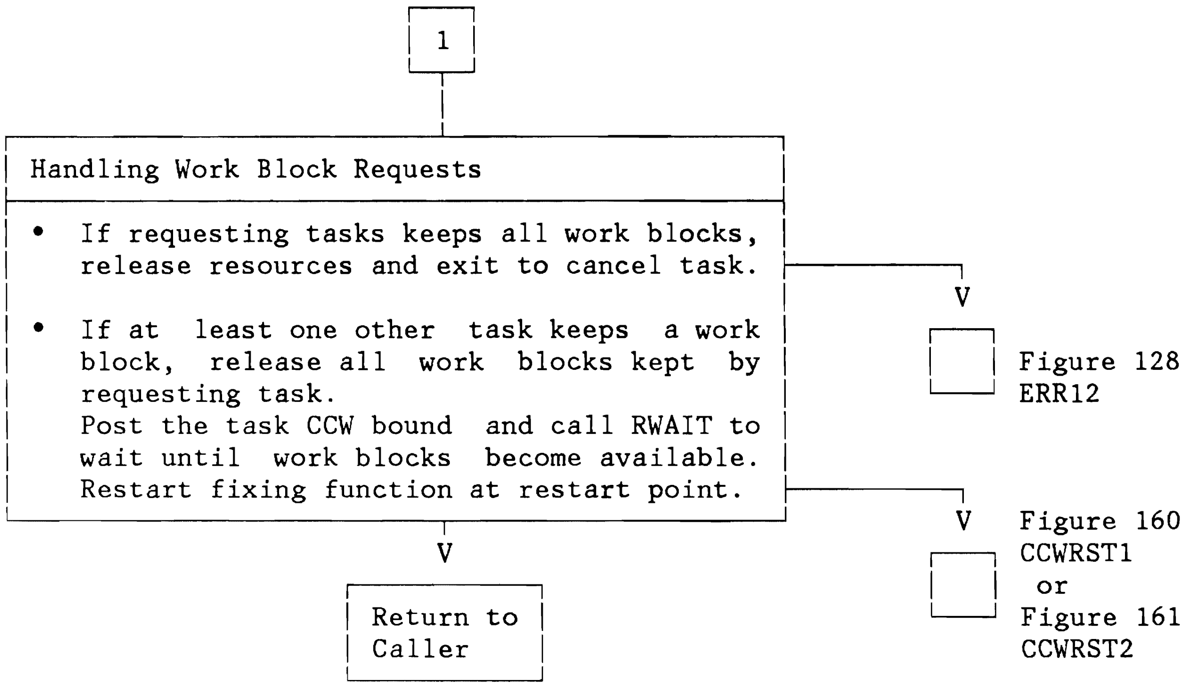
GETBL10

- If the 'Save FHB Queue' is empty, and if the request for a block was made during replica creation, release the replica blocks of the request.
  Start the request as a normal request.
- If the 'Saved FHB Queue' is not empty, and if the bottom element of the 'Saved FHB Queue' is in test, then request the freeing of replica.
- Otherwise, release the replica.

GETBL100

- If at least one completed fixing function request exists, post the task work block bound and call RWAIT. Wait until work blocks become available.

Figure 172 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): GETBLOCK Routine



1

Handling Work Block Requests

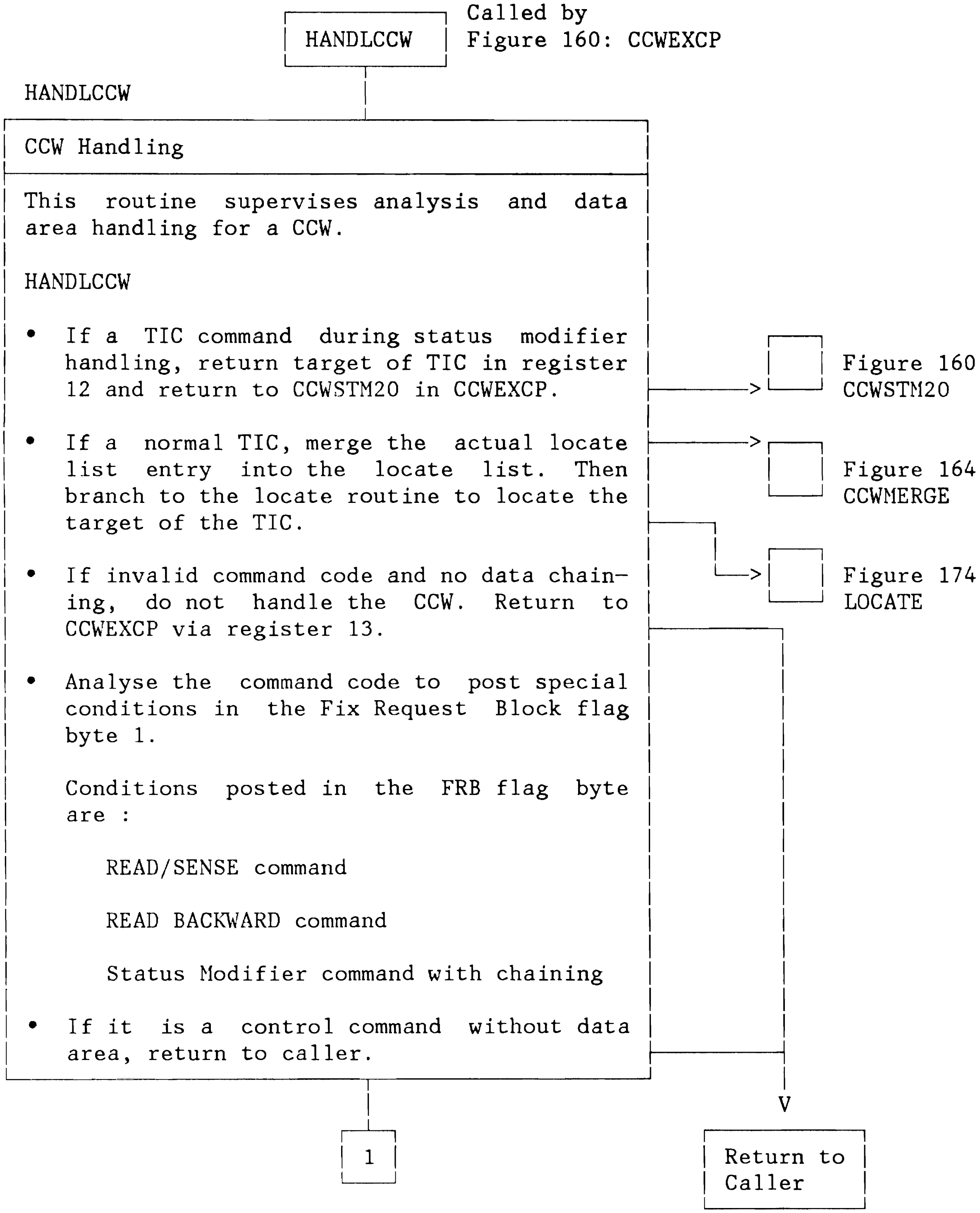
- If requesting tasks keeps all work blocks, release resources and exit to cancel task.
- If at least one other task keeps a work block, release all work blocks kept by requesting task.
  Post the task CCW bound and call RWAIT to wait until work blocks become available.
  Restart fixing function at restart point.

Figure 128
ERR12

Figure 160
CCWRST1
or
Figure 161
CCWRST2

Return to Caller

Figure 172 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): GETBLOCK Routine



HANDLCCW

Called by
Figure 160: CCWEXCP

HANDLCCW

CCW Handling

This routine supervises analysis and data area handling for a CCW.

HANDLCCW

- If a TIC command during status modifier handling, return target of TIC in register 12 and return to CCWSTM20 in CCWEXCP.
- If a normal TIC, merge the actual locate list entry into the locate list. Then branch to the locate routine to locate the target of the TIC.
- If invalid command code and no data chaining, do not handle the CCW. Return to CCWEXCP via register 13.
- Analyse the command code to post special conditions in the Fix Request Block flag byte 1.

  Conditions posted in the FRB flag byte are :

  READ/SENSE command

  READ BACKWARD command

  Status Modifier command with chaining
- If it is a control command without data area, return to caller.

Figure 160
CCWSTM20

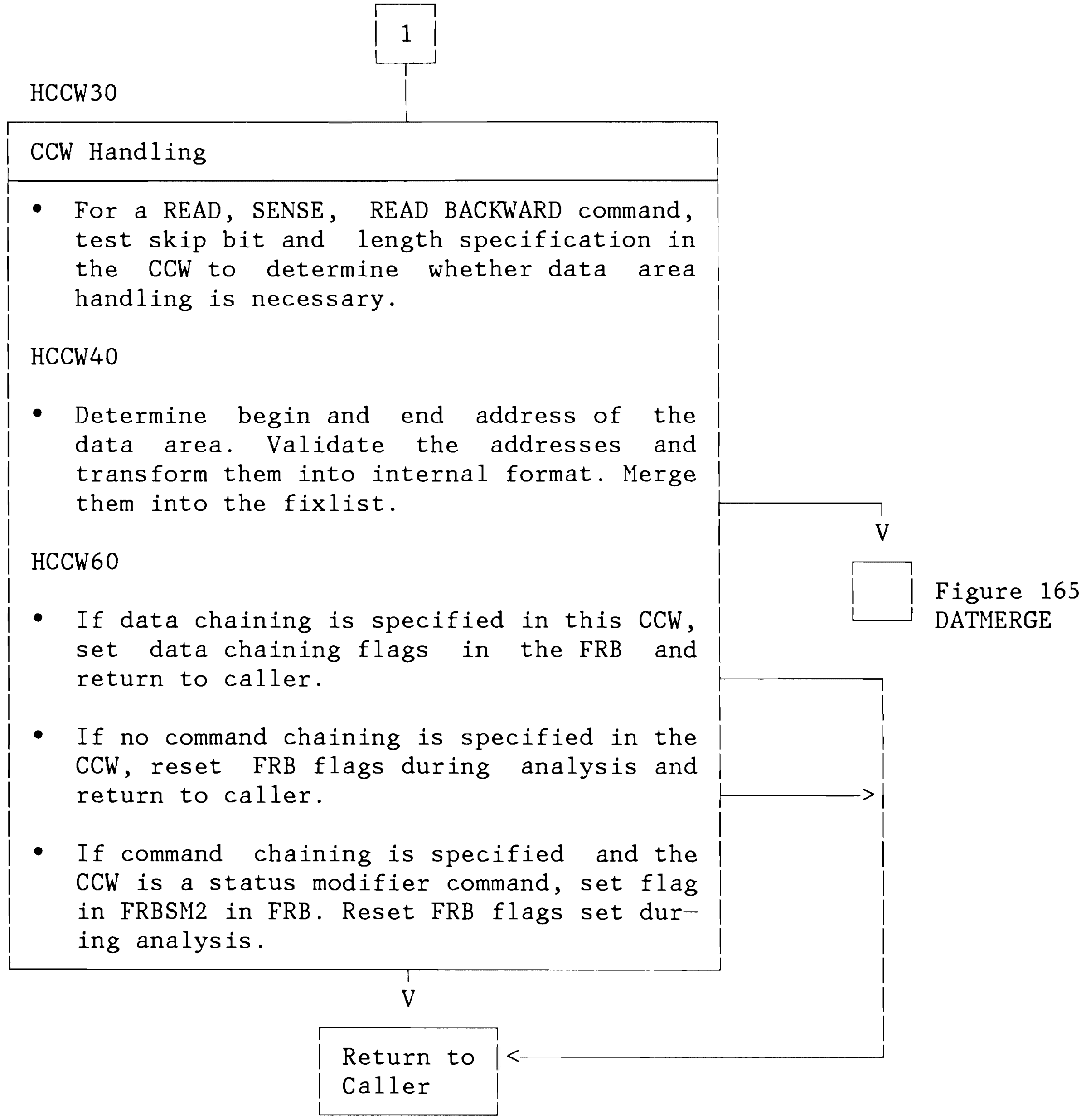
Figure 164
CCWMERGE

Figure 174
LOCATE

1

Return to Caller

Figure 173 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): HANDLCCW Routine



1

HCCW30

CCW Handling

- For a READ, SENSE, READ BACKWARD command, test skip bit and length specification in the CCW to determine whether data area handling is necessary.

HCCW40

- Determine begin and end address of the data area. Validate the addresses and transform them into internal format. Merge them into the fixlist.

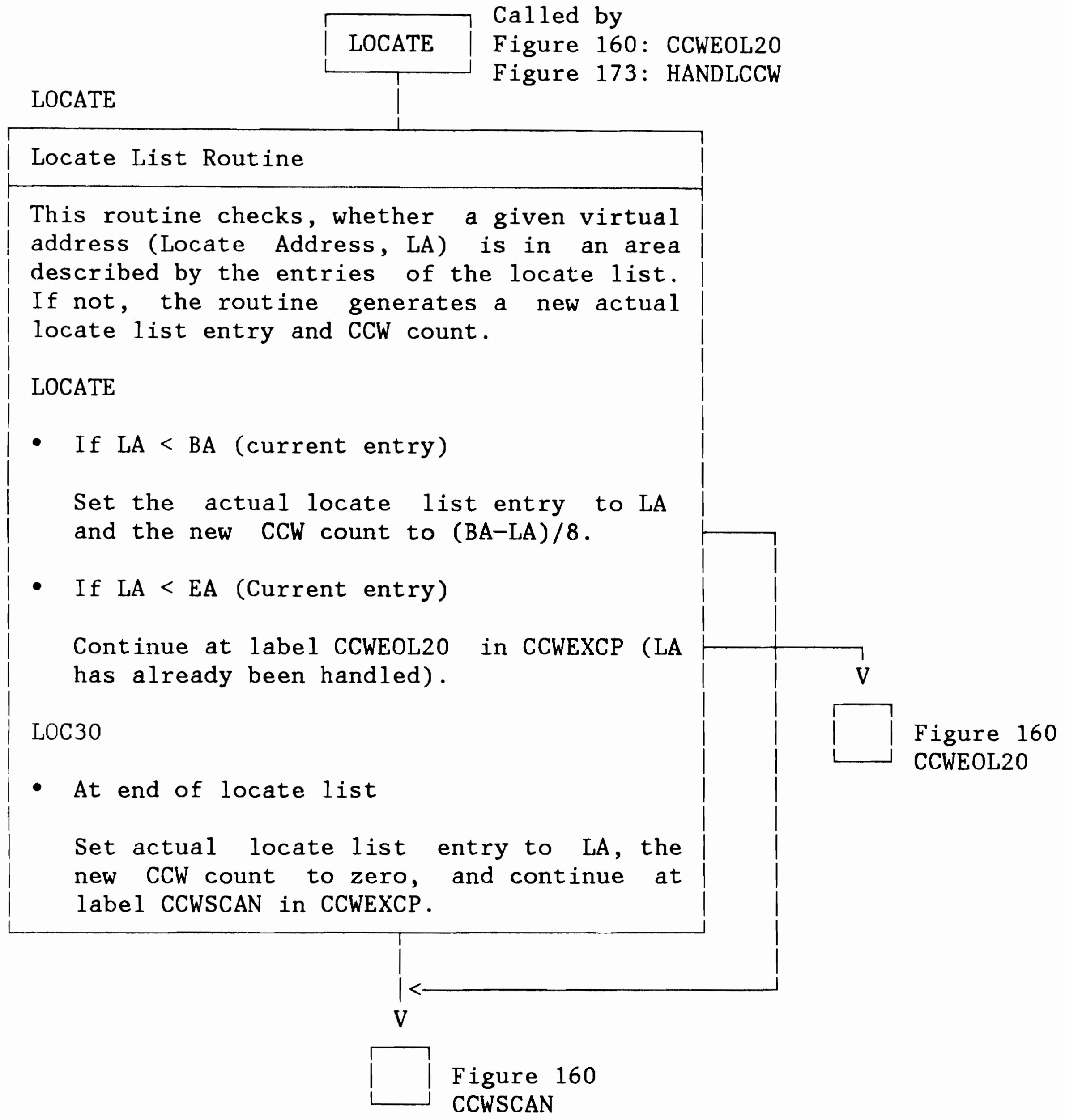
HCCW60

- If data chaining is specified in this CCW, set data chaining flags in the FRB and return to caller.

- If no command chaining is specified in the CCW, reset FRB flags during analysis and return to caller.

- If command chaining is specified and the CCW is a status modifier command, set flag in FRBSM2 in FRB. Reset FRB flags set during analysis.

Figure 173 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): HANDLCCW Routine



Locate List Routine

This routine checks, whether a given virtual address (Locate Address, LA) is in an area described by the entries of the locate list. If not, the routine generates a new actual locate list entry and CCW count.

LOCATE

- If LA < BA (current entry)

  Set the actual locate list entry to LA and the new CCW count to (BA-LA)/8.

- If LA < EA (Current entry)

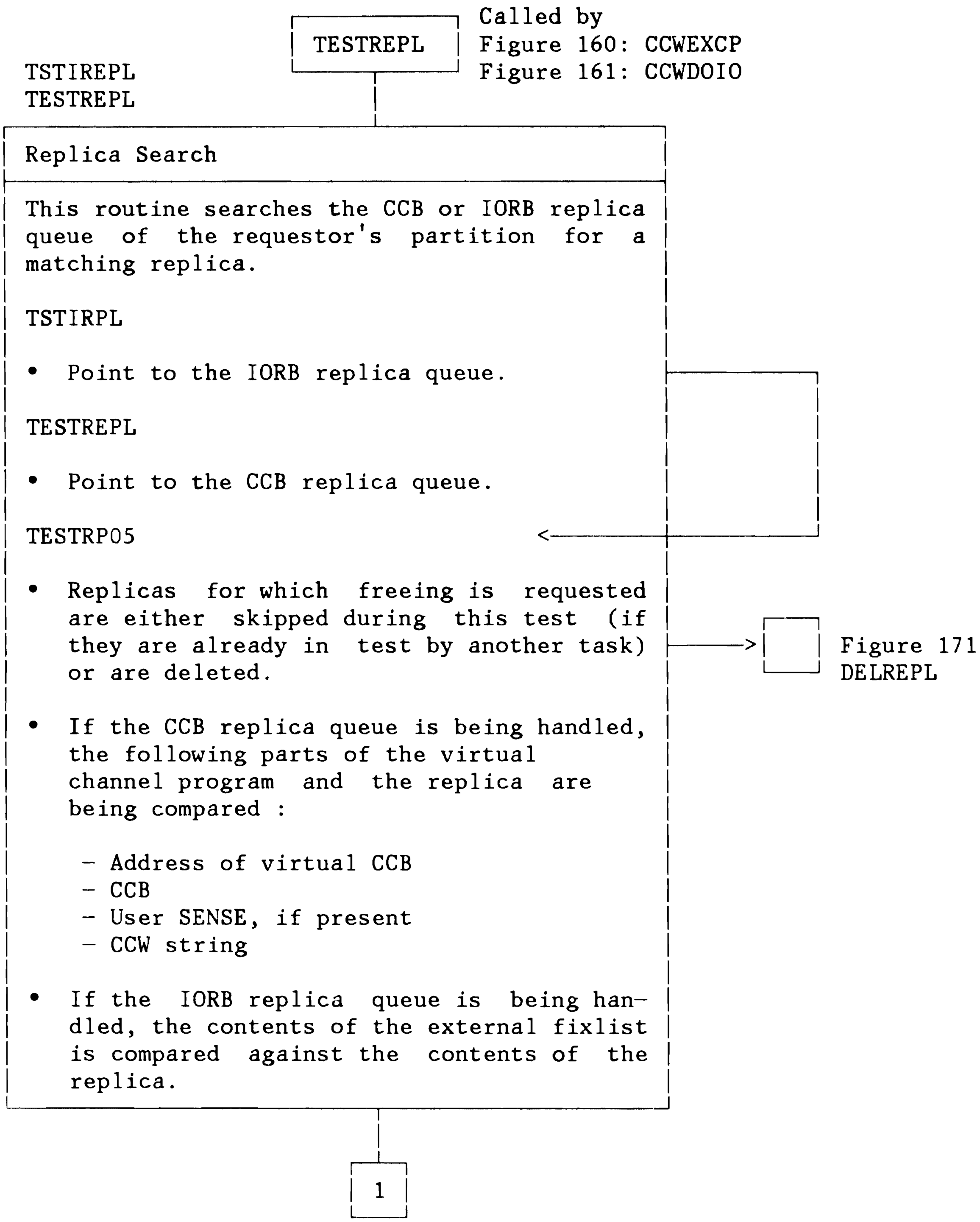
  Continue at label CCWEOL20 in CCWEXCP (LA has already been handled).

LOC30

- At end of locate list

  Set actual locate list entry to LA, the new CCW count to zero, and continue at label CCWSCAN in CCWEXCP.

Figure 174. Channel Program Fixing (ECPS:VSE Mode): LOCATE Routine



TESTREPL

Called by
Figure 160: CCWEXCP
Figure 161: CCWDOIO

TSTIREPL
TESTREPL

Replica Search

This routine searches the CCB or IORB replica queue of the requestor's partition for a matching replica.

TSTIRPL

- Point to the IORB replica queue.

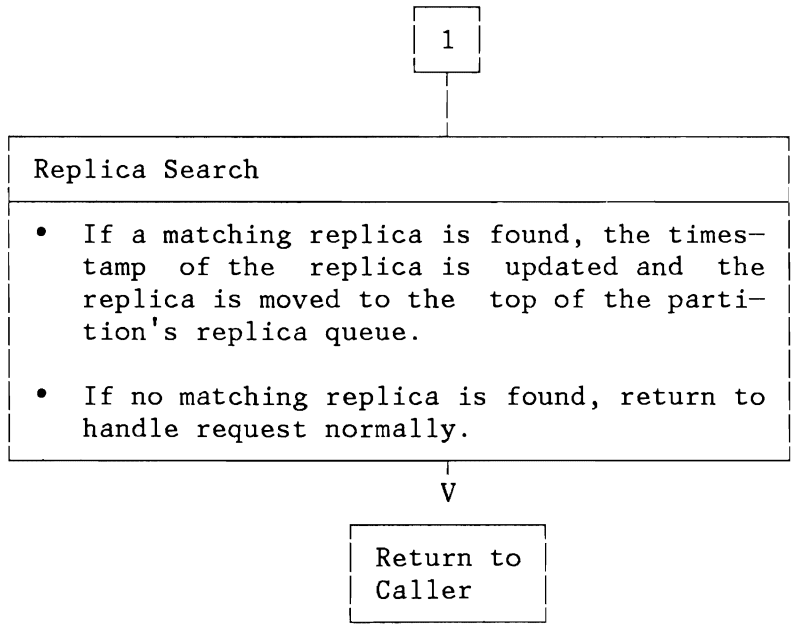
TESTREPL

- Point to the CCB replica queue.

TESTRP05

- Replicas for which freeing is requested are either skipped during this test (if they are already in test by another task) or are deleted.
- If the CCB replica queue is being handled, the following parts of the virtual channel program and the replica are being compared :
  - Address of virtual CCB
  - CCB
  - User SENSE, if present
  - CCW string
- If the IORB replica queue is being handled, the contents of the external fixlist is compared against the contents of the replica.

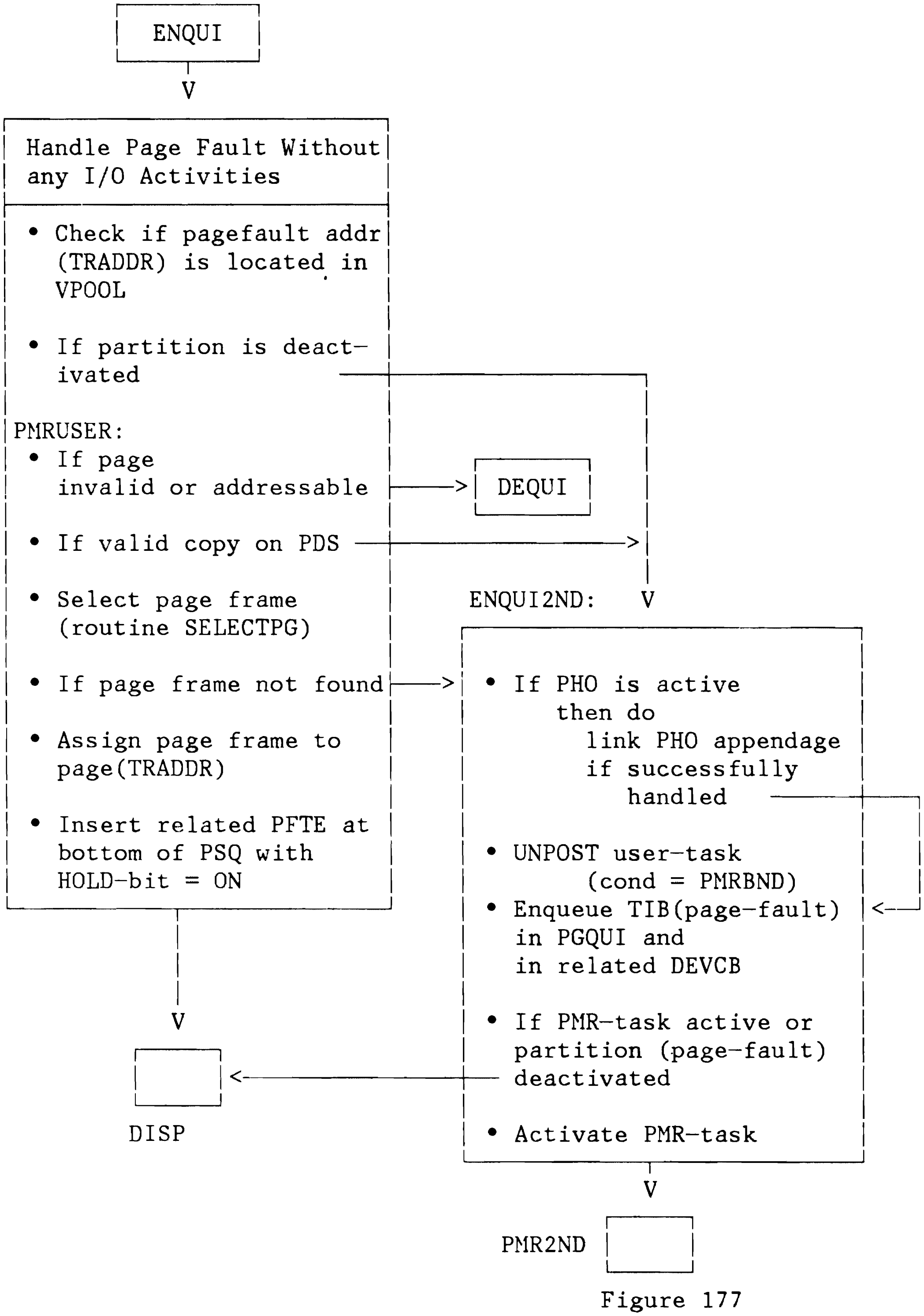
Figure 175 (Part 1 of 2). Channel Program Fixing (ECPS:VSE Mode): TESTREPL Routine



Replica Search

- If a matching replica is found, the timestamp of the replica is updated and the replica is moved to the top of the partition's replica queue.
- If no matching replica is found, return to handle request normally.

Figure 175 (Part 2 of 2). Channel Program Fixing (ECPS:VSE Mode): TESTREPL Routine



ENQUI

Handle Page Fault Without any I/O Activities

- Check if pagefault addr (TRADDR) is located in VPOOL
- If partition is deactivated

PMRUSER:

- If page invalid or addressable
- If valid copy on PDS
- Select page frame (routine SELECTPG)
- If page frame not found
- Assign page frame to page(TRADDR)
- Insert related PFTE at bottom of PSQ with HOLD-bit = ON

DEQUI

ENQUI2ND:

- If PHO is active
    then do
      link PHO appendage
      if successfully
        handled
- UNPOST user-task
      (cond = PMRBND)
- Enqueue TIB(page-fault) in PGQUI and in related DEVCB
- If PMR-task active or partition (page-fault) deactivated
- Activate PMR-task

DISP

PMR2ND

Figure 177

Figure 176. Page Management: ENQUI Routine

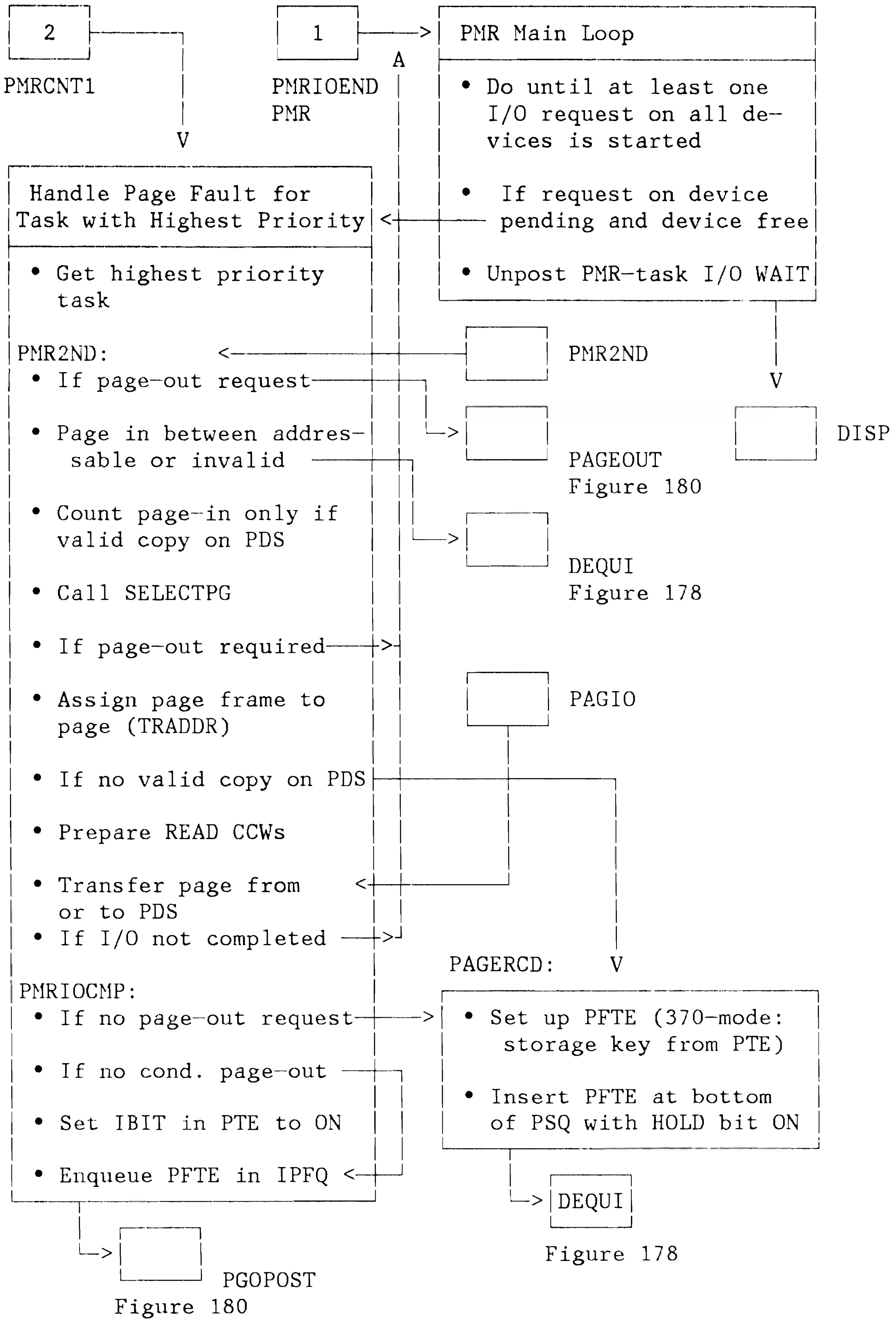

2
PMRCNT1
1
PMRIOEND
PMR
A
PMR Main Loop
• Do until at least one I/O request on all devices is started
• If request on device pending and device free
• Unpost PMR-task I/O WAIT
Handle Page Fault for Task with Highest Priority
• Get highest priority task
PMR2ND:
• If page-out request
• Page in between addressable or invalid
• Count page-in only if valid copy on PDS
• Call SELECTPG
• If page-out required
• Assign page frame to page (TRADDR)
• If no valid copy on PDS
• Prepare READ CCWs
• Transfer page from or to PDS
• If I/O not completed
PMRIOCMP:
• If no page-out request
• If no cond. page-out
• Set IBIT in PTE to ON
• Enqueue PFTE in IPFQ
PMR2ND
PAGEOUT
Figure 180
DISP
DEQUI
Figure 178
PAGIO
PAGERCD:
• Set up PFTE (370-mode: storage key from PTE)
• Insert PFTE at bottom of PSQ with HOLD bit ON
DEQUI
Figure 178
PGOPOST
Figure 180

Figure 177. Page Management: PMR Routine

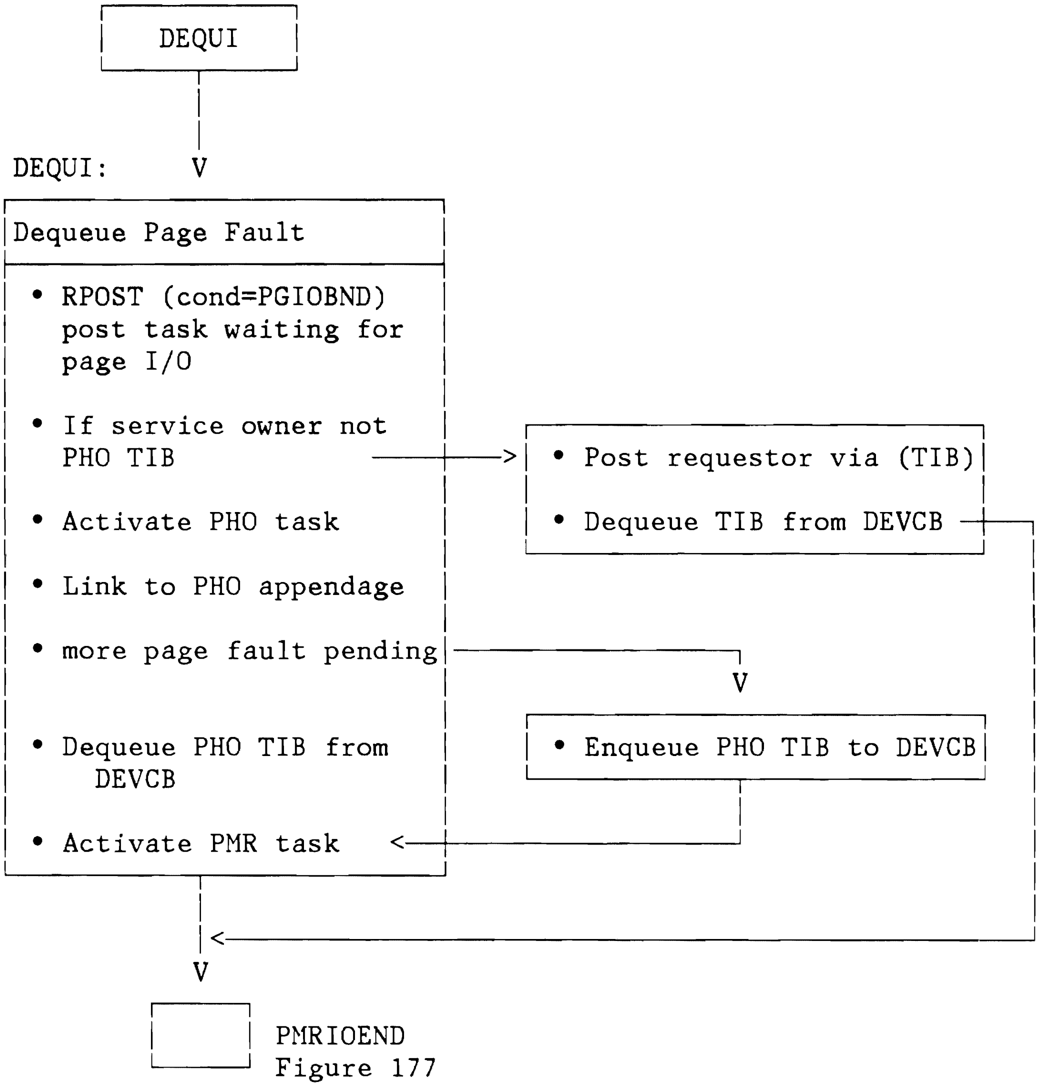

DEQUI
DEQUI:
Dequeue Page Fault
• RPOST (cond=PGIOBND) post task waiting for page I/O
• If service owner not PHO TIB
• Activate PHO task
• Link to PHO appendage
• more page fault pending
• Dequeue PHO TIB from DEVCB
• Activate PMR task
• Post requestor via (TIB)
• Dequeue TIB from DEVCB
• Enqueue PHO TIB to DEVCB
PMRIOEND
Figure 177

Figure 178. Page Management: DEQUI Routine

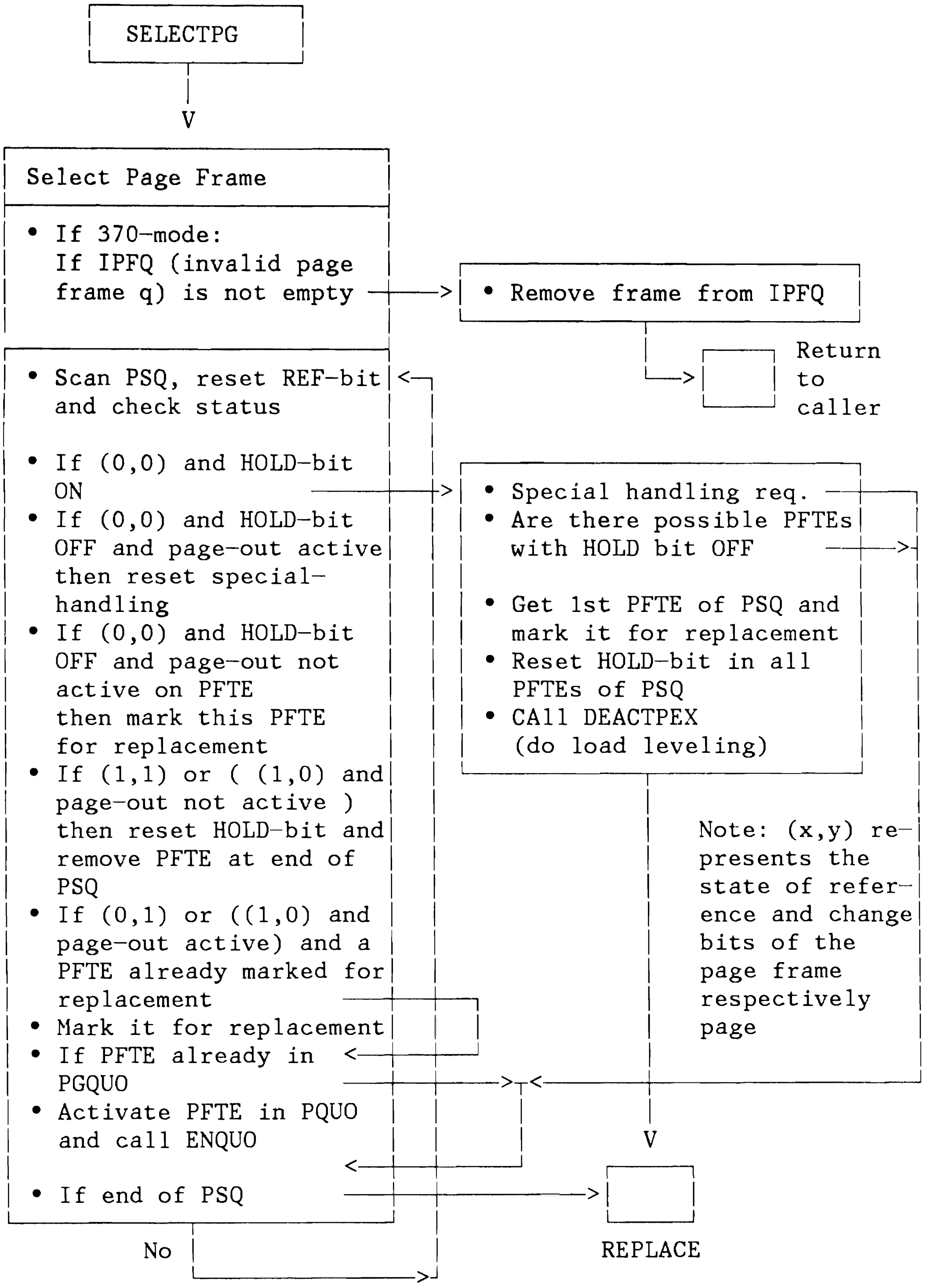


Figure 179 (Part 1 of 2). Page Management: SELECTPG Routine

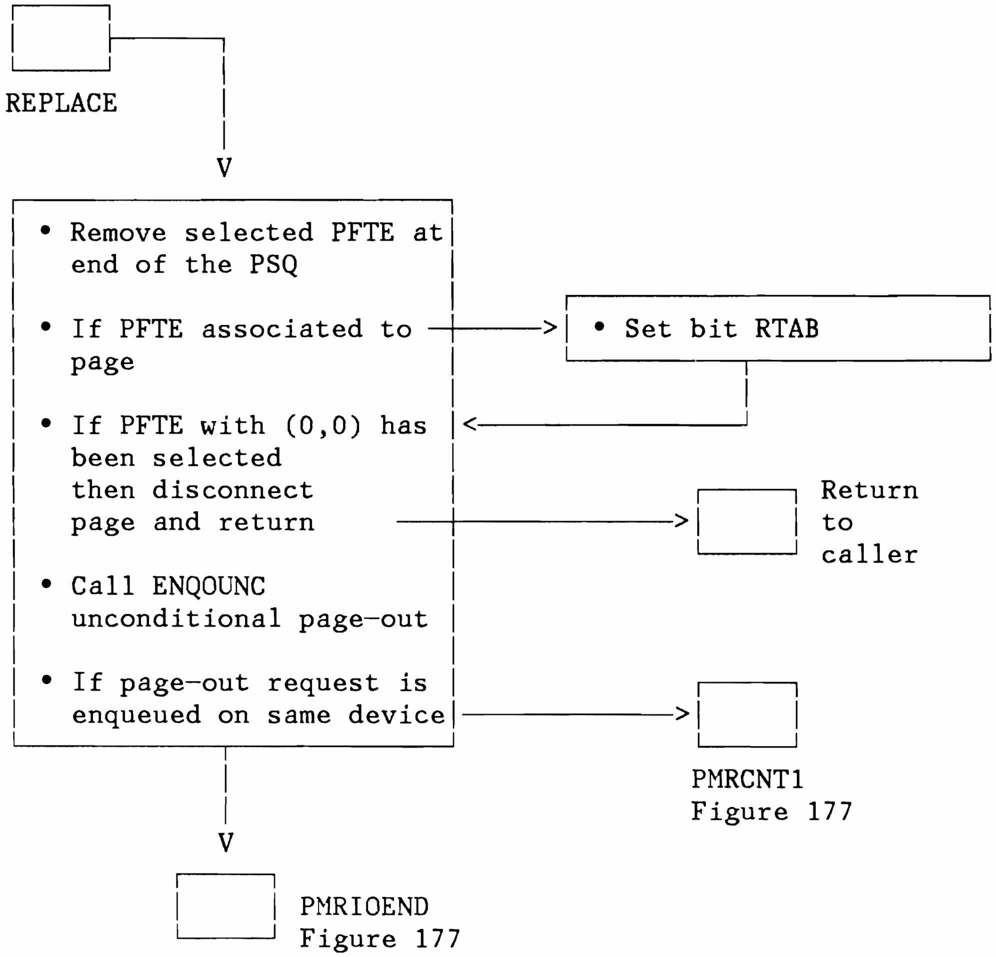
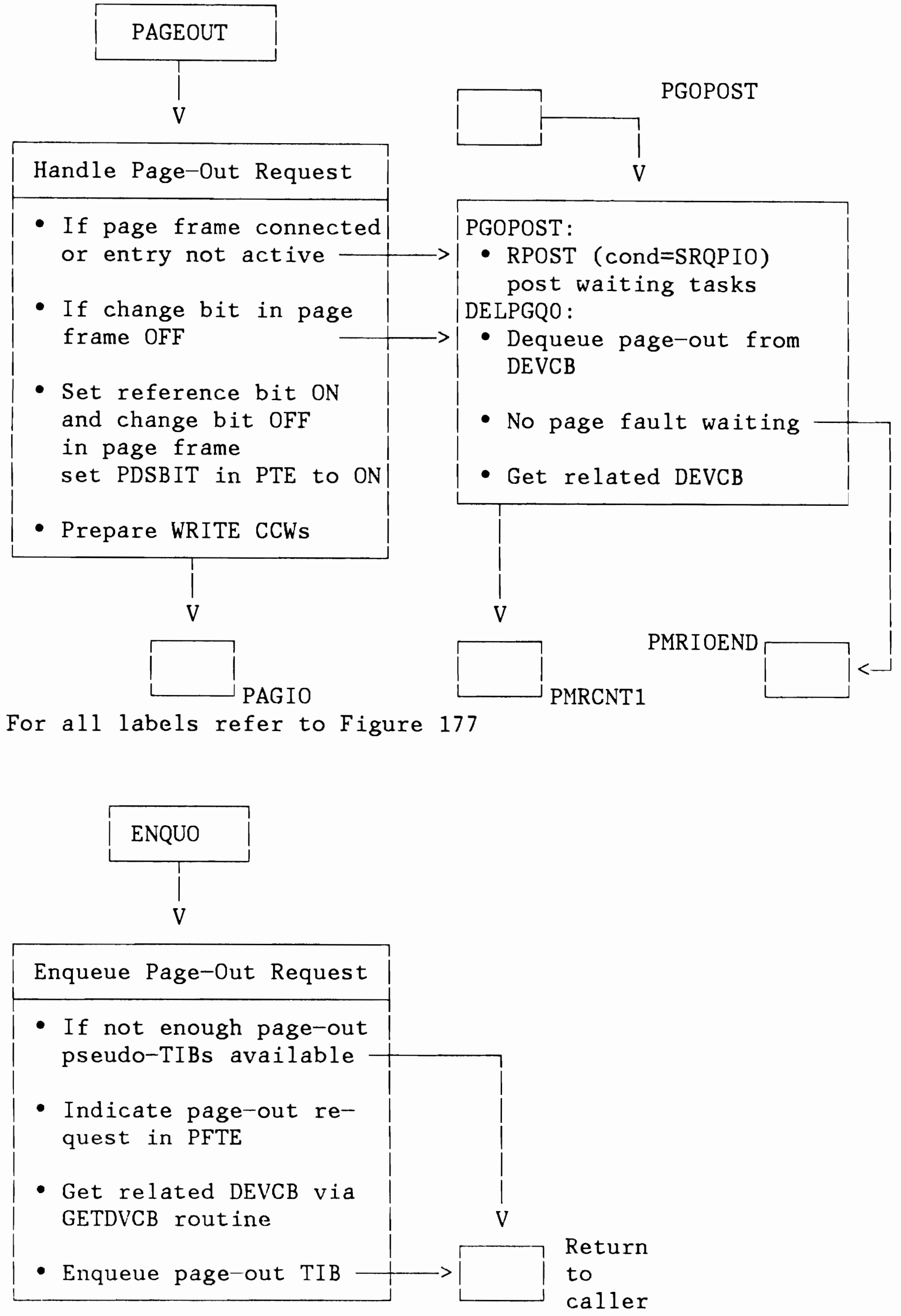

REPLACE
• Remove selected PFTE at end of the PSQ
• If PFTE associated to page
• Set bit RTAB
• If PFTE with (0,0) has been selected then disconnect page and return
Return to caller
• Call ENQOUNC unconditional page-out
• If page-out request is enqueued on same device
PMRCNT1 Figure 177
PMRIOEND Figure 177

Figure 179 (Part 2 of 2). Page Management: SELECTPG Routine



Handle Page-Out Request

- If page frame connected or entry not active
- If change bit in page frame OFF
- Set reference bit ON and change bit OFF in page frame set PDSBIT in PTE to ON
- Prepare WRITE CCWs

PGOPOST:
- RPOST (cond=SRQPIO) post waiting tasks

DELPGQ0:
- Dequeue page-out from DEVCB
- No page fault waiting
- Get related DEVCB

For all labels refer to Figure 177

Enqueue Page-Out Request

- If not enough page-out pseudo-TIBs available
- Indicate page-out request in PFTE
- Get related DEVCB via GETDVCB routine
- Enqueue page-out TIB

Figure 180. Page Management: PAGEOUT and ENQUO Routines

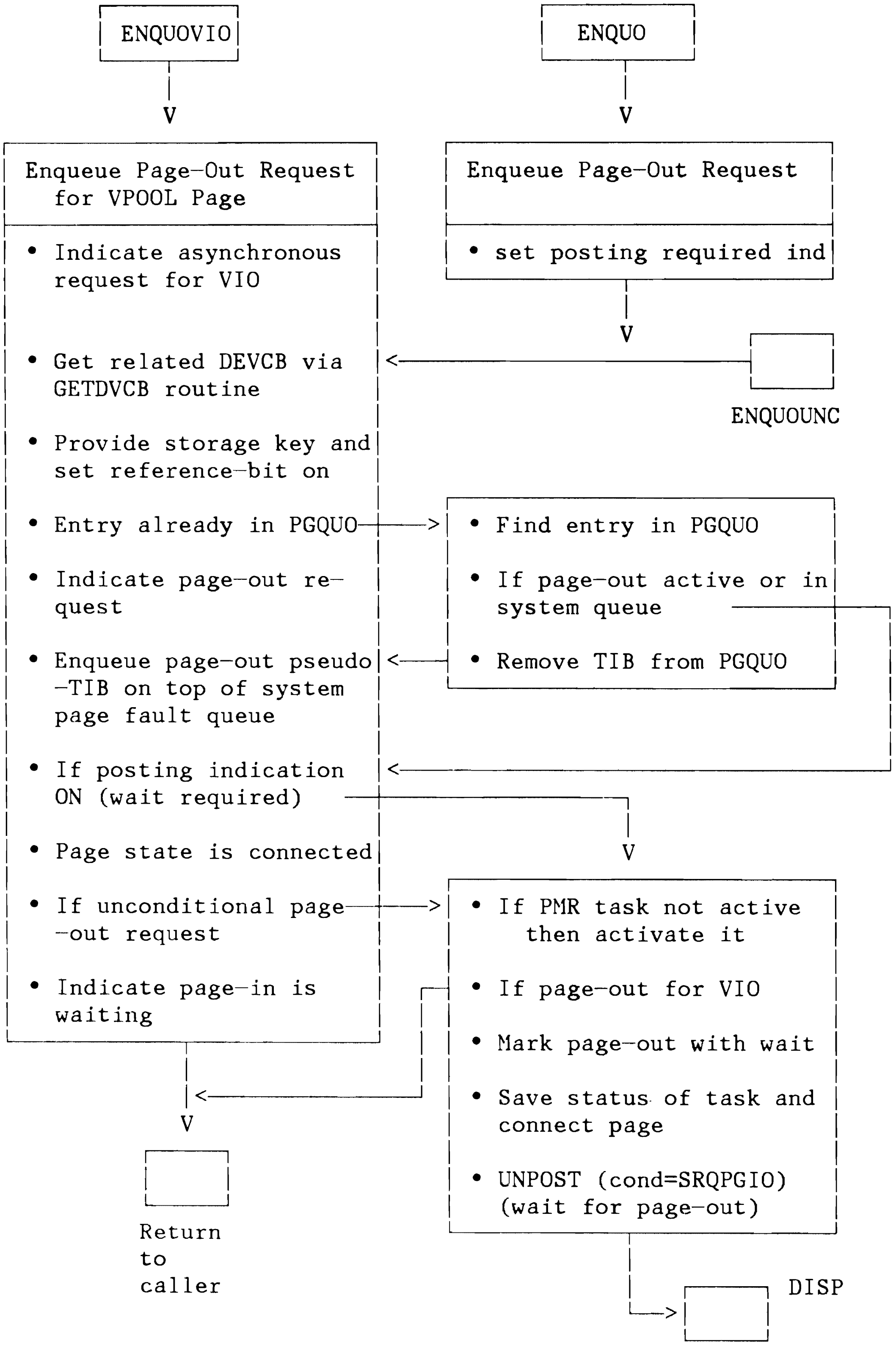

ENQUOVIO
ENQUO
Enqueue Page-Out Request for VPOOL Page
• Indicate asynchronous request for VIO
• Get related DEVCB via GETDVCB routine
• Provide storage key and set reference-bit on
• Entry already in PGQUO
• Indicate page-out request
• Enqueue page-out pseudo-TIB on top of system page fault queue
• If posting indication ON (wait required)
• Page state is connected
• If unconditional page-out request
• Indicate page-in is waiting
Enqueue Page-Out Request
• set posting required ind
ENQUOUNC
• Find entry in PGQUO
• If page-out active or in system queue
• Remove TIB from PGQUO
• If PMR task not active then activate it
• If page-out for VIO
• Mark page-out with wait
• Save status of task and connect page
• UNPOST (cond=SRQPGIO) (wait for page-out)
Return to caller
DISP

Figure 181. Page Management: ENQUOW Routine

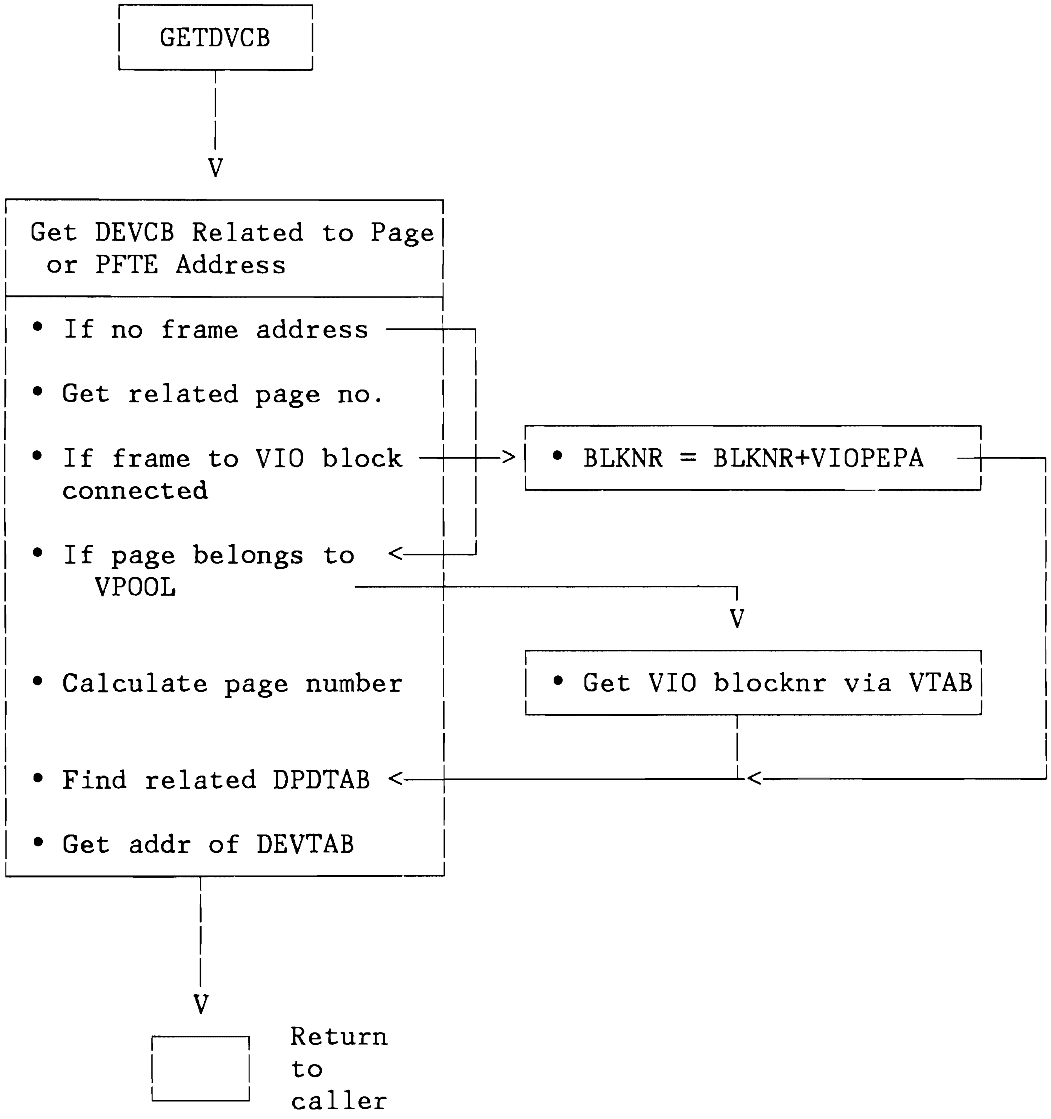


Figure 182. Page Management: GETDVCB Routine

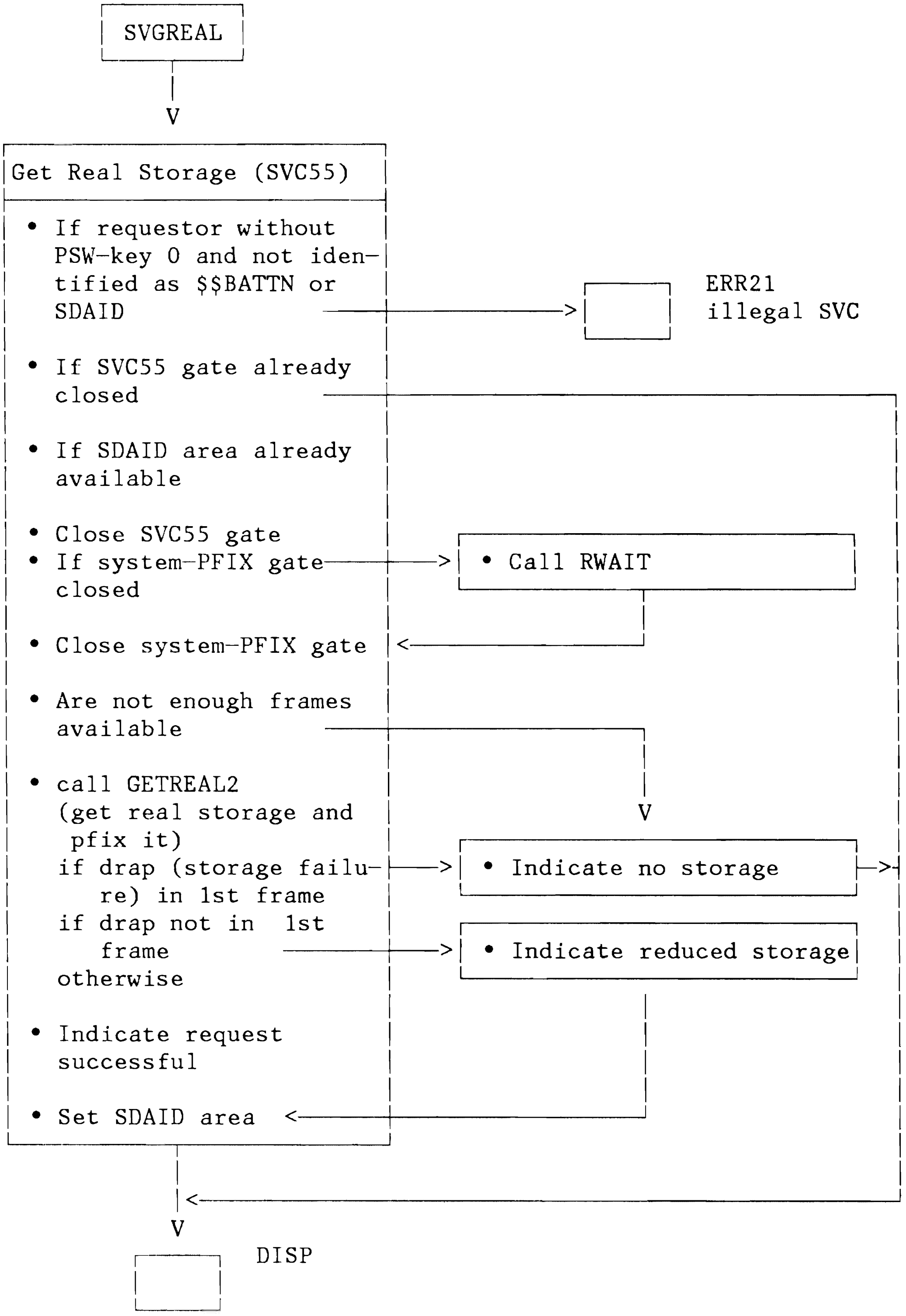
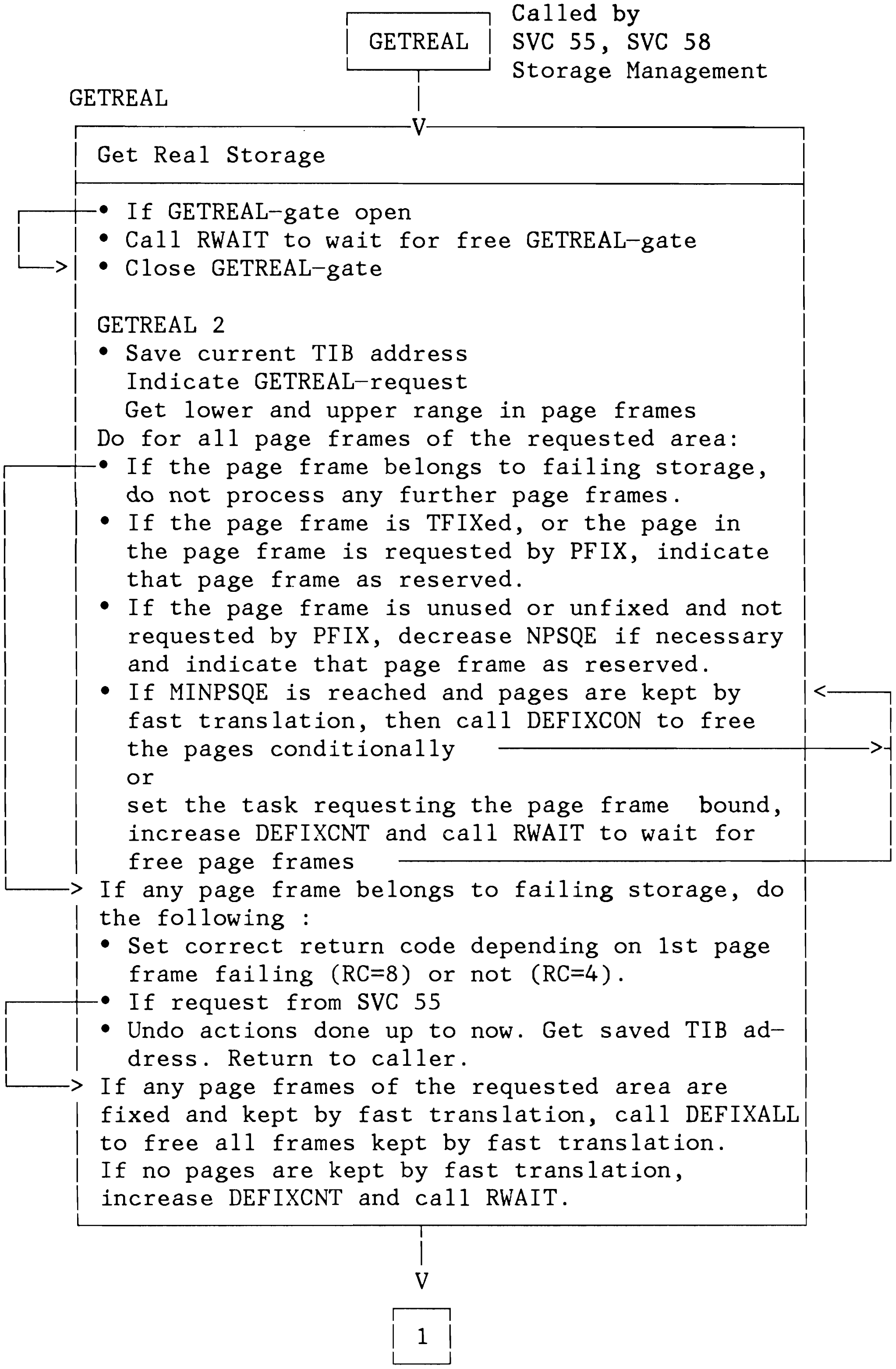

SVGREAL
Get Real Storage (SVC55)
• If requestor without PSW-key 0 and not identified as $$BATTN or SDAID
ERR21
illegal SVC
• If SVC55 gate already closed
• If SDAID area already available
• Close SVC55 gate
• If system-PFIX gate closed
• Call RWAIT
• Close system-PFIX gate
• Are not enough frames available
• call GETREAL2 (get real storage and pfix it)
if drap (storage failure) in 1st frame
if drap not in 1st frame
otherwise
• Indicate no storage
• Indicate reduced storage
• Indicate request successful
• Set SDAID area
DISP

Figure 183. Page Management: SVGREAL Routine (370 Mode only)



GETREAL

Called by
SVC 55, SVC 58
Storage Management

GETREAL

Get Real Storage

- If GETREAL-gate open
- Call RWAIT to wait for free GETREAL-gate
- Close GETREAL-gate

GETREAL 2

- Save current TIB address
  Indicate GETREAL-request
  Get lower and upper range in page frames

Do for all page frames of the requested area:

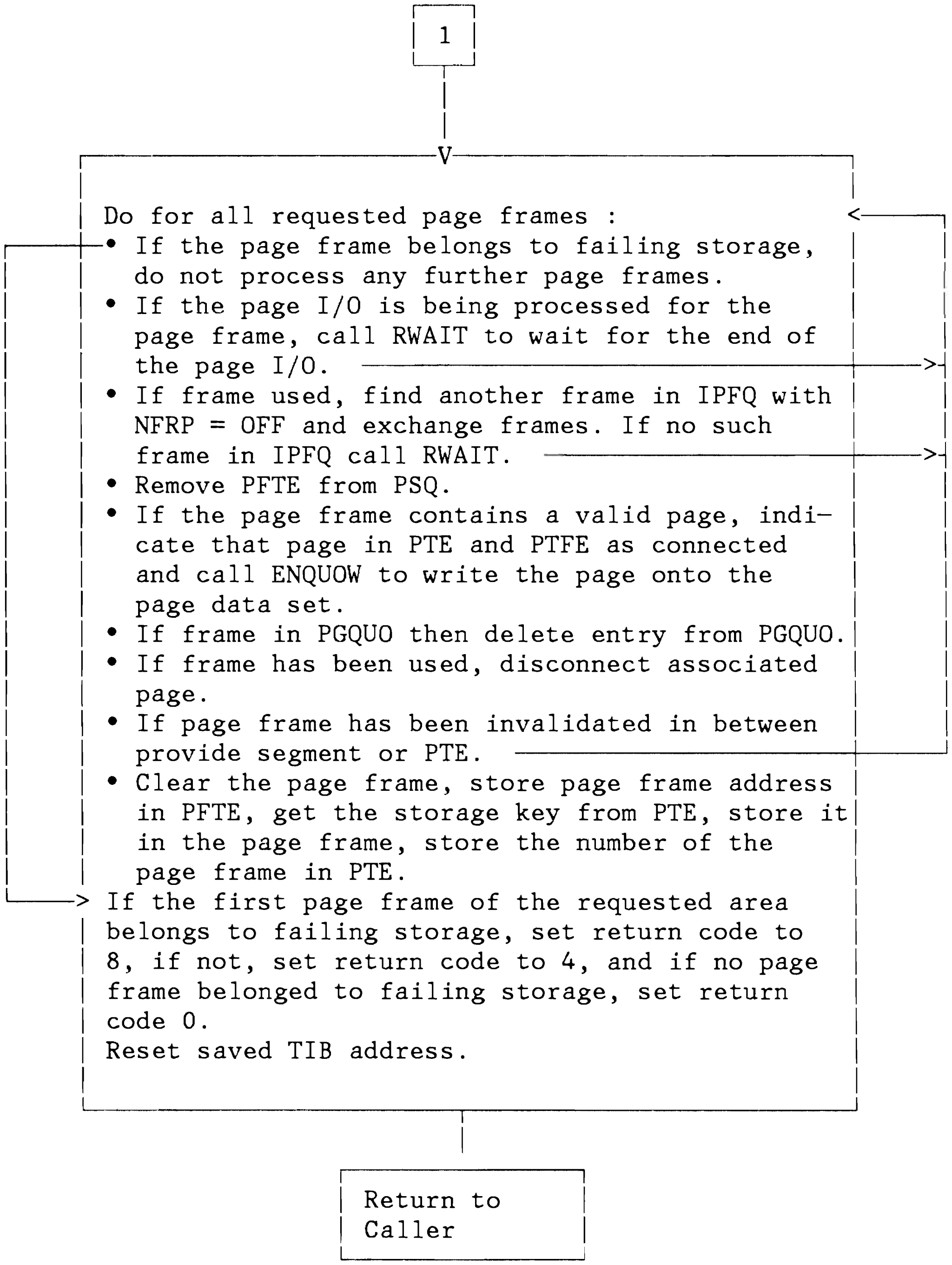
- If the page frame belongs to failing storage, do not process any further page frames.
- If the page frame is TFIXed, or the page in the page frame is requested by PFIX, indicate that page frame as reserved.
- If the page frame is unused or unfixed and not requested by PFIX, decrease NPSQE if necessary and indicate that page frame as reserved.
- If MINPSQE is reached and pages are kept by fast translation, then call DEFIXCON to free the pages conditionally
  or
  set the task requesting the page frame bound, increase DEFIXCNT and call RWAIT to wait for free page frames

If any page frame belongs to failing storage, do the following :

- Set correct return code depending on 1st page frame failing (RC=8) or not (RC=4).
- If request from SVC 55
- Undo actions done up to now. Get saved TIB address. Return to caller.

If any page frames of the requested area are fixed and kept by fast translation, call DEFIXALL to free all frames kept by fast translation.
If no pages are kept by fast translation, increase DEFIXCNT and call RWAIT.

Figure 184 (Part 1 of 2). Page Management: GETREAL Routine (370 Mode only)



1

Do for all requested page frames :

- If the page frame belongs to failing storage, do not process any further page frames.
- If the page I/O is being processed for the page frame, call RWAIT to wait for the end of the page I/O.
- If frame used, find another frame in IPFQ with NFRP = OFF and exchange frames. If no such frame in IPFQ call RWAIT.
- Remove PFTE from PSQ.
- If the page frame contains a valid page, indicate that page in PTE and PTFE as connected and call ENQUOW to write the page onto the page data set.
- If frame in PGQUO then delete entry from PGQUO.
- If frame has been used, disconnect associated page.
- If page frame has been invalidated in between provide segment or PTE.
- Clear the page frame, store page frame address in PFTE, get the storage key from PTE, store it in the page frame, store the number of the page frame in PTE.

If the first page frame of the requested area belongs to failing storage, set return code to 8, if not, set return code to 4, and if no page frame belonged to failing storage, set return code 0.

Reset saved TIB address.

Return to Caller

Figure 184 (Part 2 of 2). Page Management: GETREAL Routine (370 Mode only)

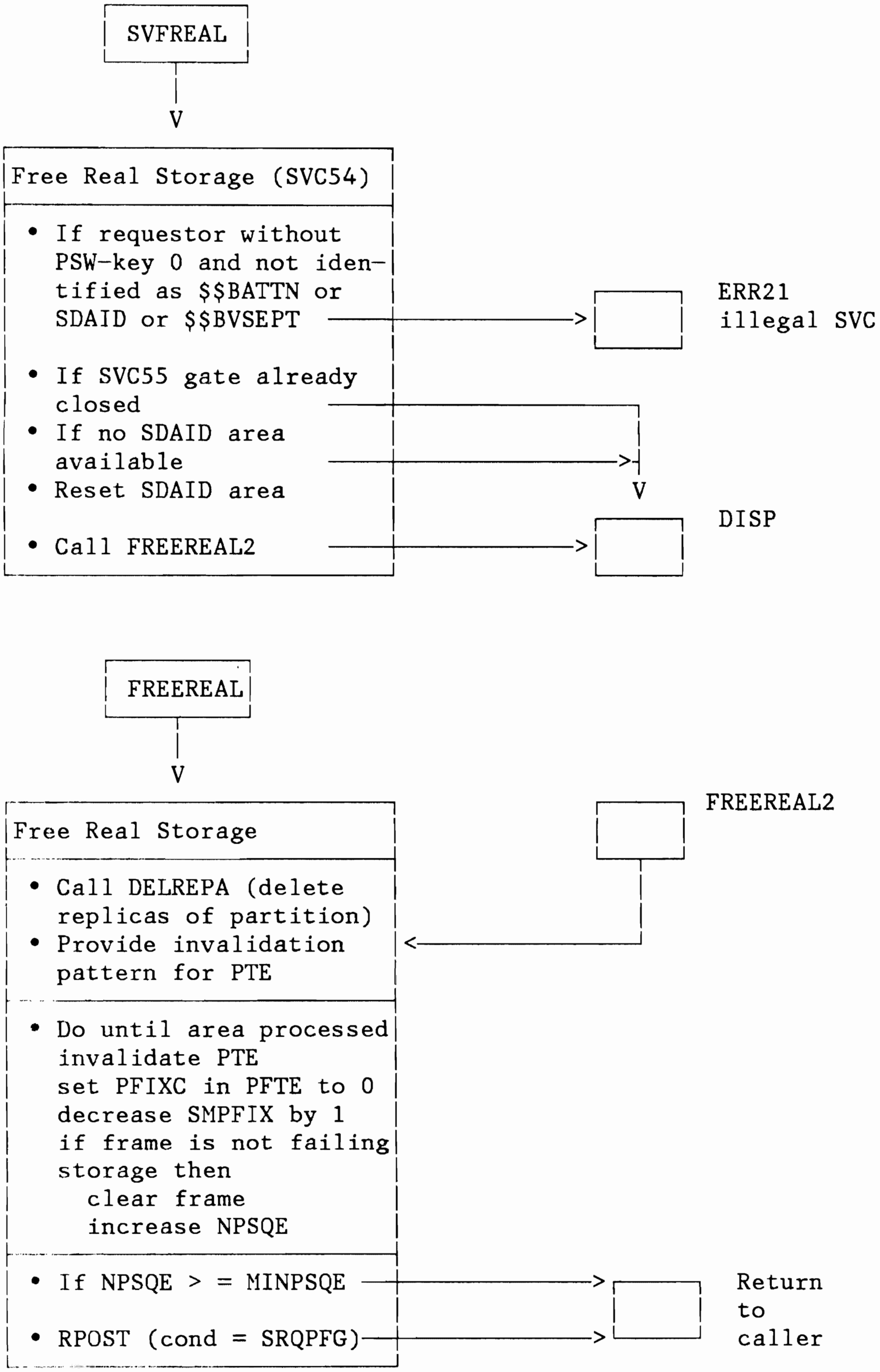
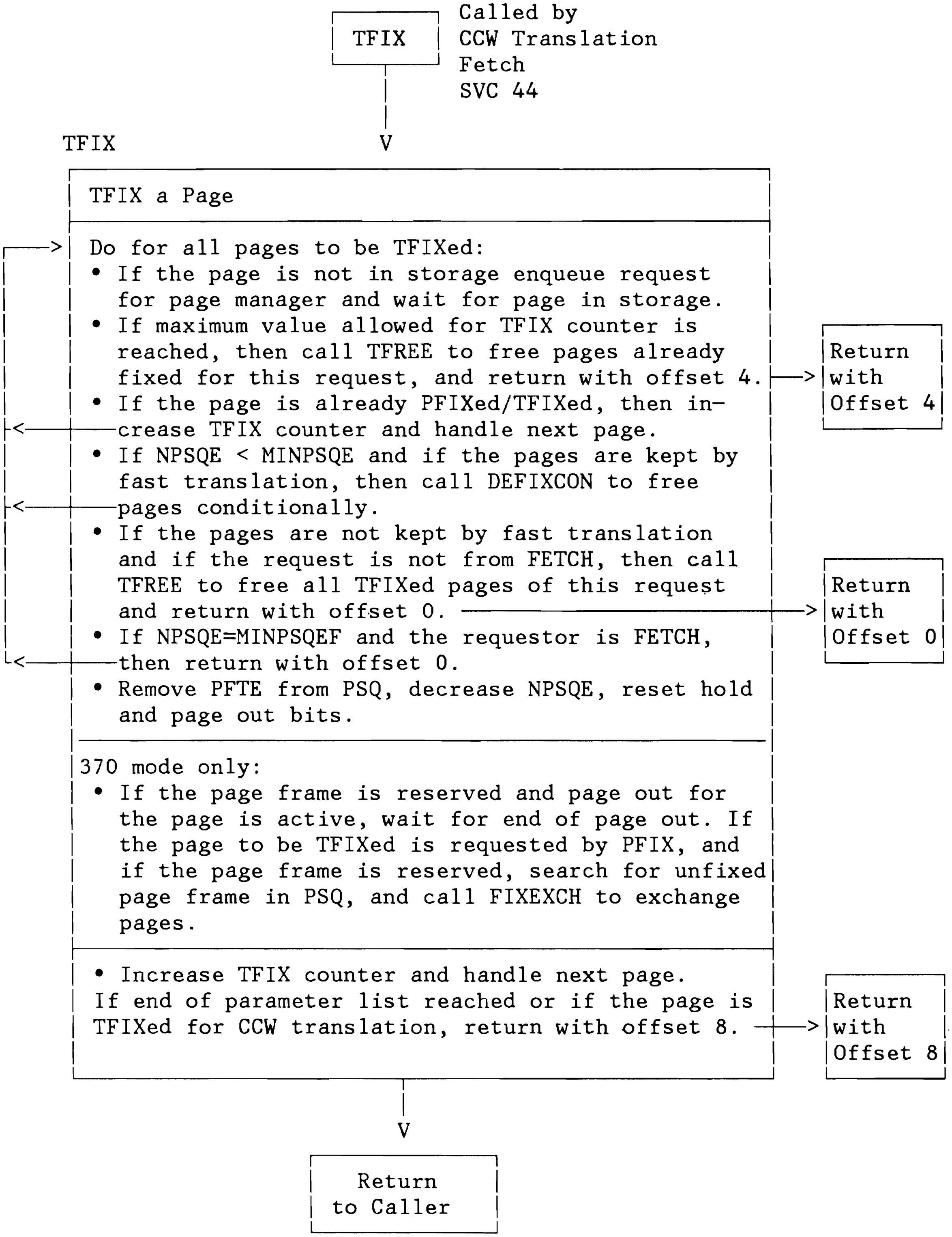

SVFREAL
Free Real Storage (SVC54)
• If requestor without PSW-key 0 and not identified as $$BATTN or SDAID or $$BVSEPT
ERR21 illegal SVC
• If SVC55 gate already closed
• If no SDAID area available
• Reset SDAID area
• Call FREEREAL2
DISP
FREEREAL
Free Real Storage
FREEREAL2
• Call DELREPA (delete replicas of partition)
• Provide invalidation pattern for PTE
• Do until area processed
invalidate PTE
set PFIXC in PFTE to 0
decrease SMPFIX by 1
if frame is not failing
storage then
clear frame
increase NPSQE
• If NPSQE > = MINPSQE
• RPOST (cond = SRQPFG)
Return to caller

Figure 185. Page Management: SVFREAL and FREEREAL Routines (370 Mode only)



TFIX

Called by
CCW Translation
Fetch
SVC 44

TFIX

TFIX a Page

Do for all pages to be TFIXed:

- If the page is not in storage enqueue request for page manager and wait for page in storage.
- If maximum value allowed for TFIX counter is reached, then call TFREE to free pages already fixed for this request, and return with offset 4.
- If the page is already PFIXed/TFIXed, then increase TFIX counter and handle next page.
- If NPSQE < MINPSQE and if the pages are kept by fast translation, then call DEFIXCON to free pages conditionally.
- If the pages are not kept by fast translation and if the request is not from FETCH, then call TFREE to free all TFIXed pages of this request and return with offset 0.
- If NPSQE=MINPSQEF and the requestor is FETCH, then return with offset 0.
- Remove PFTE from PSQ, decrease NPSQE, reset hold and page out bits.

370 mode only:

- If the page frame is reserved and page out for the page is active, wait for end of page out. If the page to be TFIXed is requested by PFIX, and if the page frame is reserved, search for unfixed page frame in PSQ, and call FIXEXCH to exchange pages.

- Increase TFIX counter and handle next page.

If end of parameter list reached or if the page is TFIXed for CCW translation, return with offset 8.

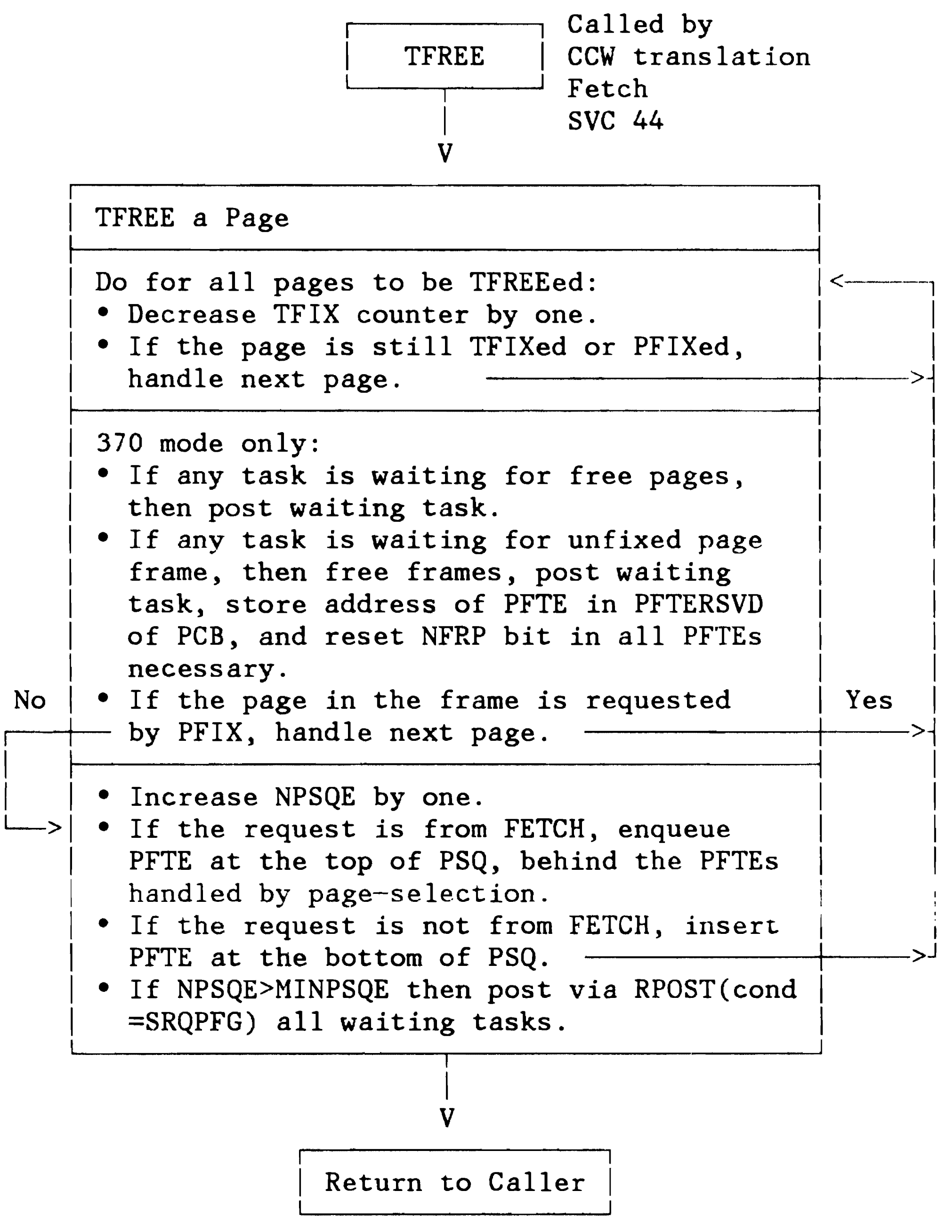
Return with Offset 4

Return with Offset 0

Return with Offset 8

Return to Caller

Figure 186. Page Management: TFIX Routine



TFREE

Called by
CCW translation
Fetch
SVC 44

TFREE a Page

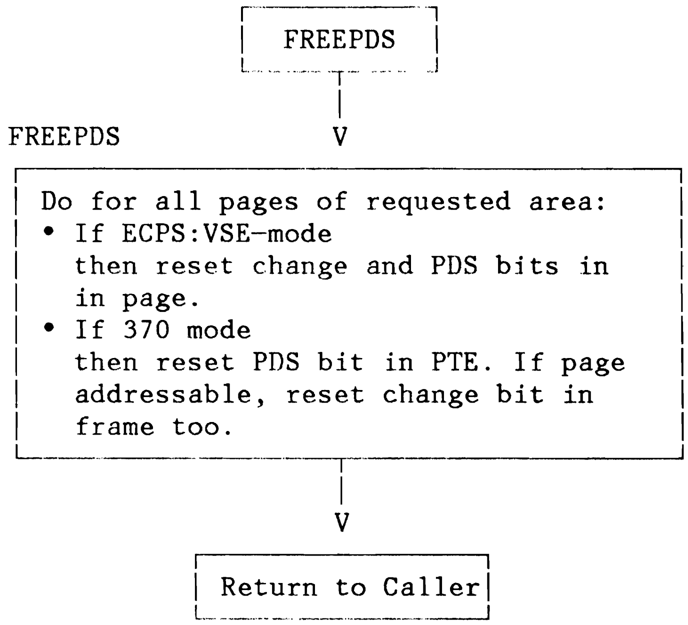
Do for all pages to be TFREEed:
- Decrease TFIX counter by one.
- If the page is still TFIXed or PFIXed, handle next page.

370 mode only:
- If any task is waiting for free pages, then post waiting task.
- If any task is waiting for unfixed page frame, then free frames, post waiting task, store address of PFTE in PFTERSVD of PCB, and reset NFRP bit in all PFTEs necessary.
- If the page in the frame is requested by PFIX, handle next page.

- Increase NPSQE by one.
- If the request is from FETCH, enqueue PFTE at the top of PSQ, behind the PFTEs handled by page-selection.
- If the request is not from FETCH, insert PFTE at the bottom of PSQ.
- If NPSQE>MINPSQE then post via RPOST(cond =SRQPFG) all waiting tasks.

Return to Caller

Figure 187. Page Management: TFREE Routine



FREEPDS

FREEPDS

Do for all pages of requested area:

- If ECPS:VSE-mode then reset change and PDS bits in in page.
- If 370 mode then reset PDS bit in PTE. If page addressable, reset change bit in frame too.

Return to Caller

Figure 188. Page Management: FREEPDS Routine

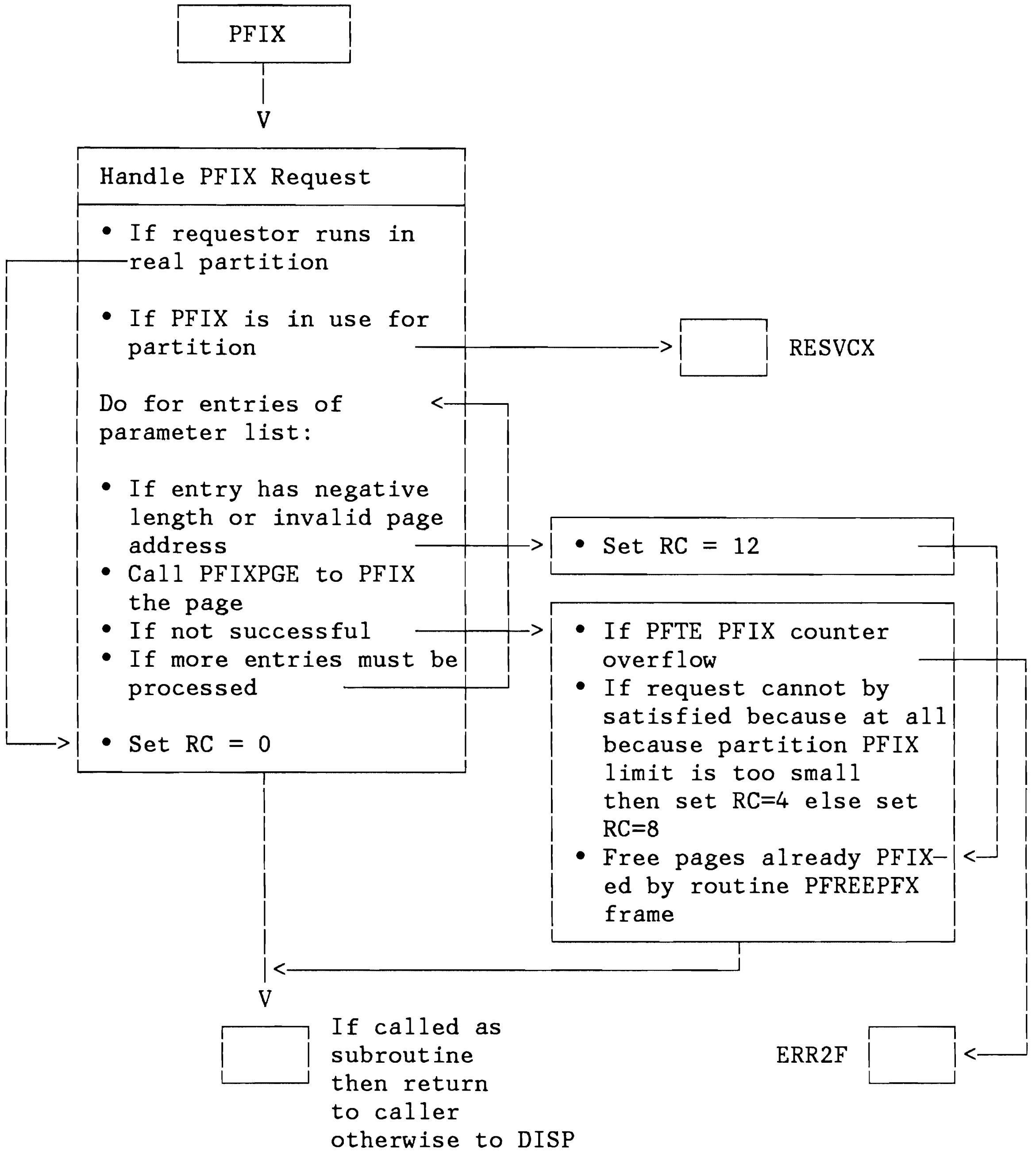

PFIX
Handle PFIX Request
• If requestor runs in real partition
• If PFIX is in use for partition
RESVCX
Do for entries of parameter list:
• If entry has negative length or invalid page address
• Call PFIXPGE to PFIX the page
• If not successful
• If more entries must be processed
• Set RC = 0
• Set RC = 12
• If PFTE PFIX counter overflow
• If request cannot by satisfied because at all because partition PFIX limit is too small then set RC=4 else set RC=8
• Free pages already PFIX-ed by routine PFREEPFX frame
If called as subroutine then return to caller otherwise to DISP
ERR2F

Figure 189. Page Management: PFIX Routine

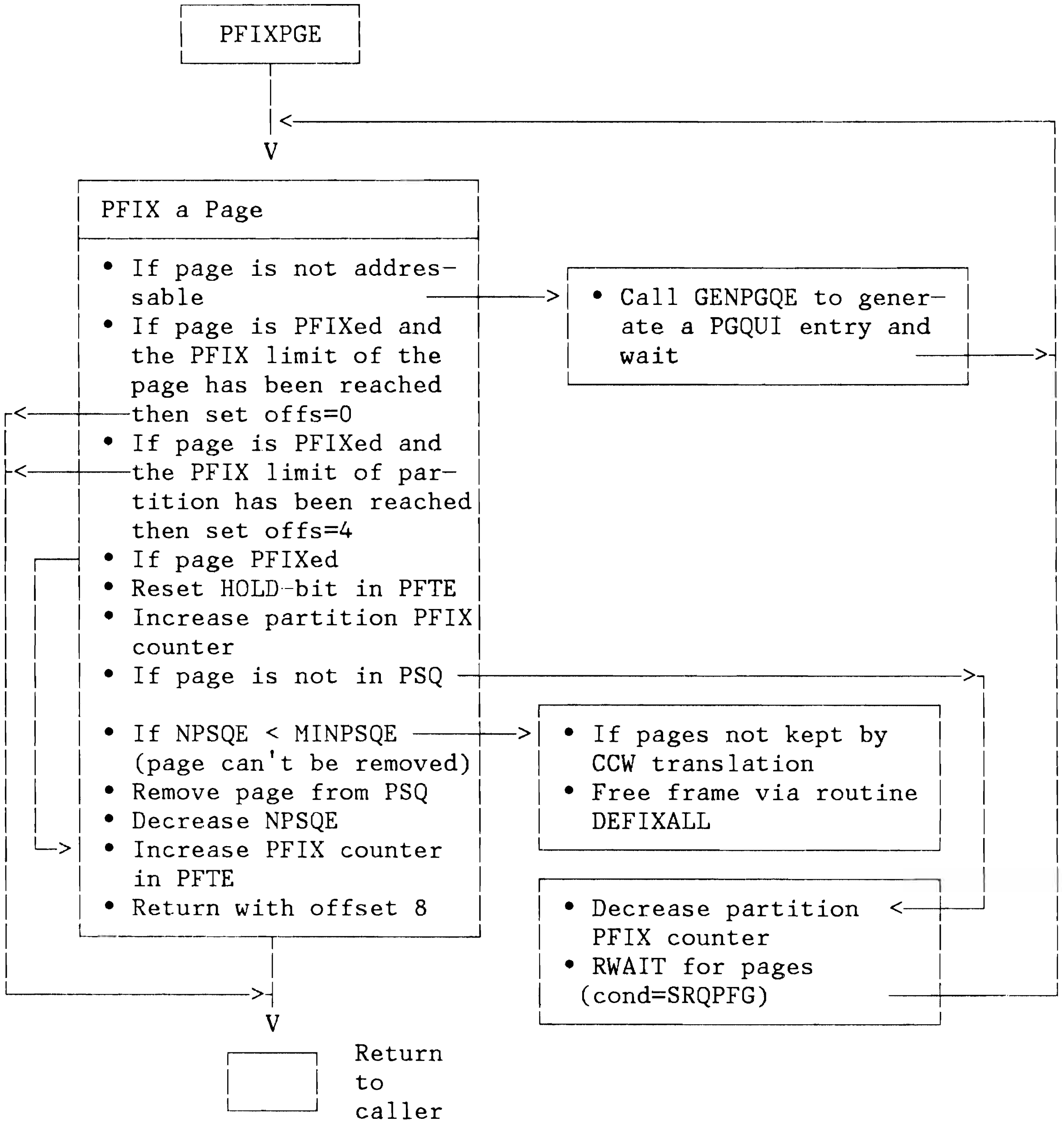

PFIXPGE
PFIX a Page
• If page is not addressable
• If page is PFIXed and the PFIX limit of the page has been reached then set offs=0
• If page is PFIXed and the PFIX limit of partition has been reached then set offs=4
• If page PFIXed
• Reset HOLD-bit in PFTE
• Increase partition PFIX counter
• If page is not in PSQ
• If NPSQE < MINPSQE (page can't be removed)
• Remove page from PSQ
• Decrease NPSQE
• Increase PFIX counter in PFTE
• Return with offset 8
• Call GENPGQE to generate a PGQUI entry and wait
• If pages not kept by CCW translation
• Free frame via routine DEFIXALL
• Decrease partition PFIX counter
• RWAIT for pages (cond=SRQPFG)
Return to caller

Figure 190. Page Management: PFIXPGE Routine (ECPS:VSE Mode)

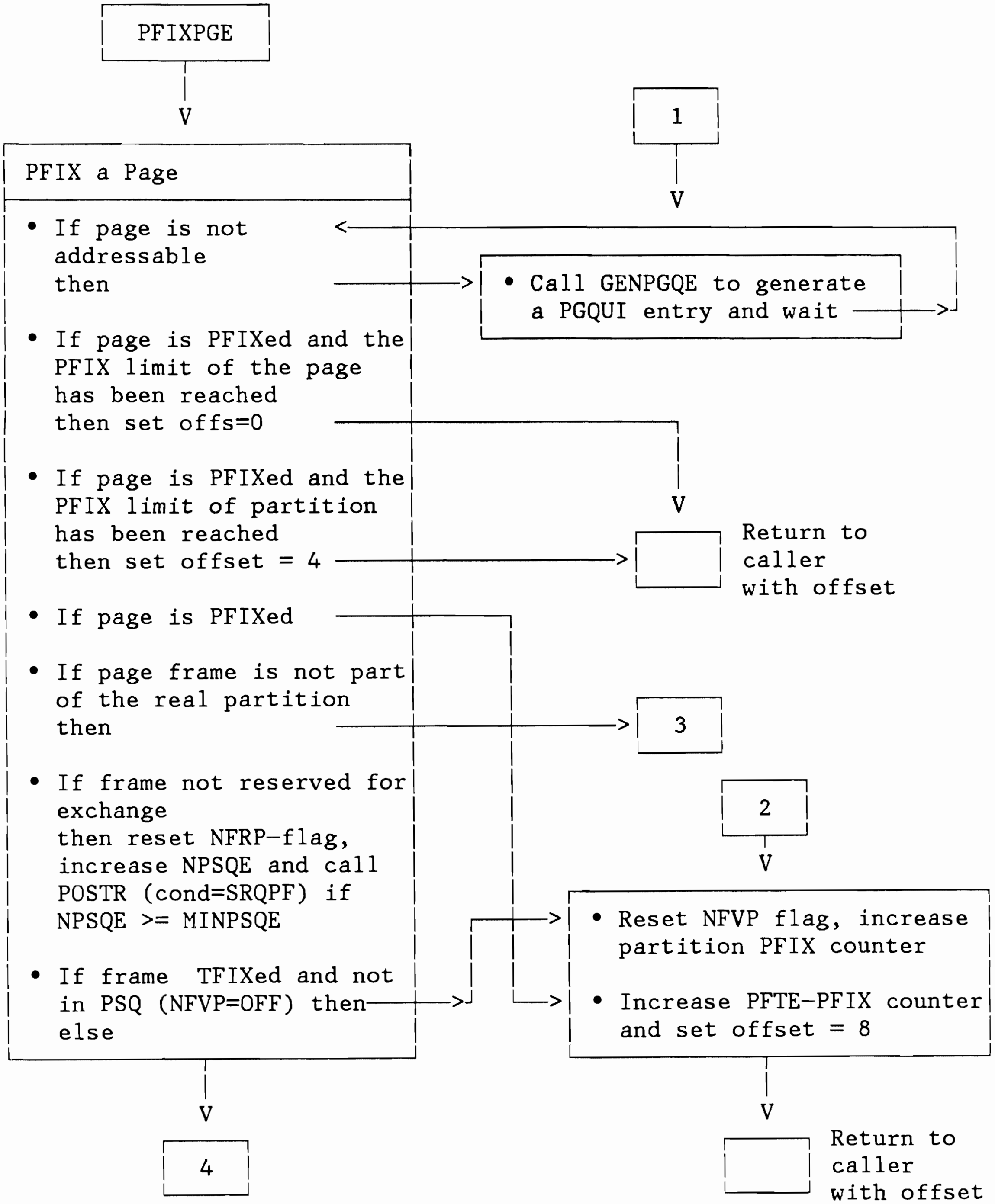

PFIXPGE
PFIX a Page
• If page is not addressable then
• If page is PFIXed and the PFIX limit of the page has been reached then set offs=0
• If page is PFIXed and the PFIX limit of partition has been reached then set offset = 4
• If page is PFIXed
• If page frame is not part of the real partition then
• If frame not reserved for exchange then reset NFRP-flag, increase NPSQE and call POSTR (cond=SRQPF) if NPSQE >= MINPSQE
• If frame TFIXed and not in PSQ (NFVP=OFF) then else
1
• Call GENPGQE to generate a PGQUI entry and wait
Return to caller with offset
3
2
• Reset NFVP flag, increase partition PFIX counter
• Increase PFTE-PFIX counter and set offset = 8
4
Return to caller with offset

Figure 191 (Part 1 of 2). Page Management: PFIXPGE Routine (370 Mode)

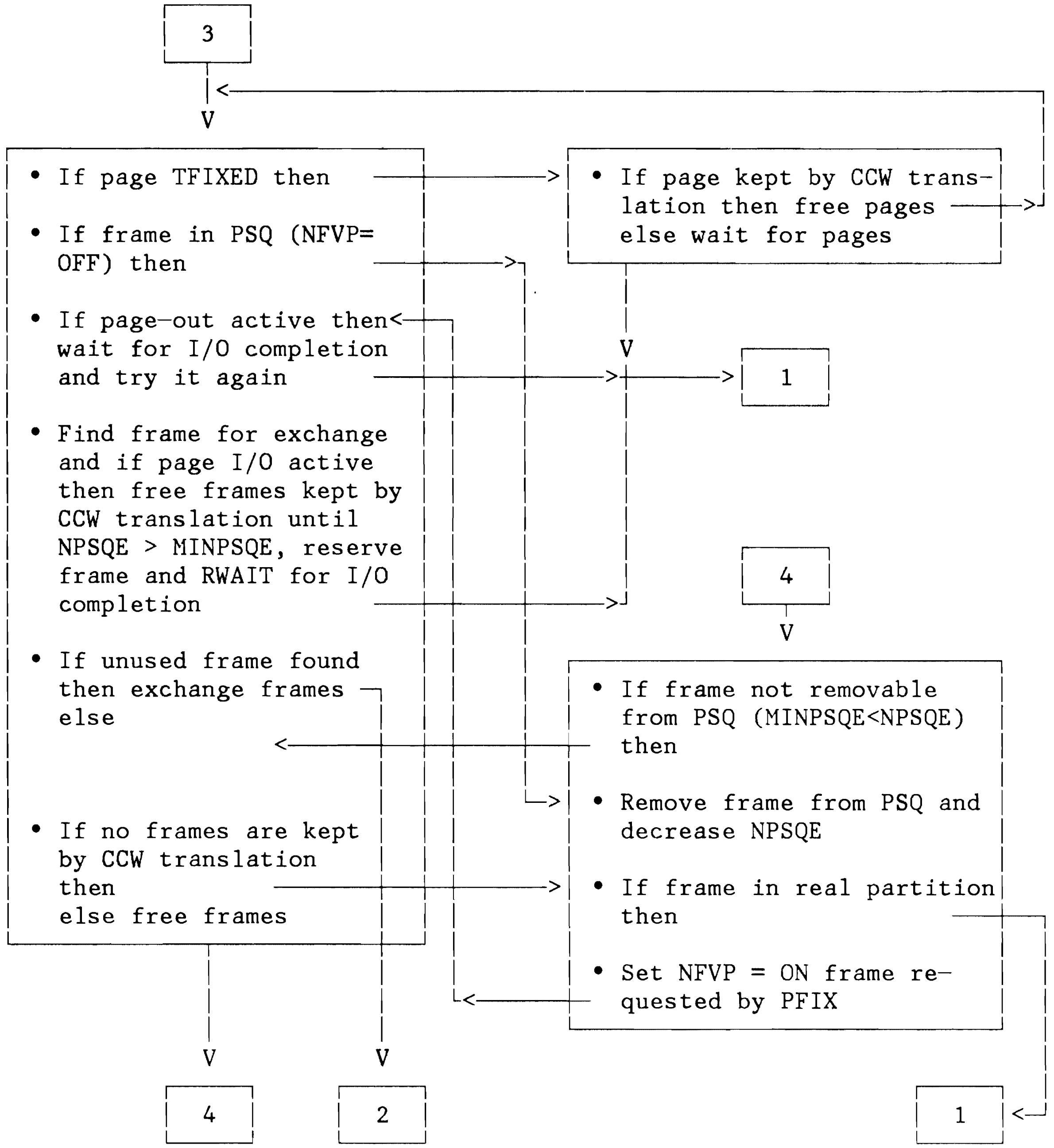
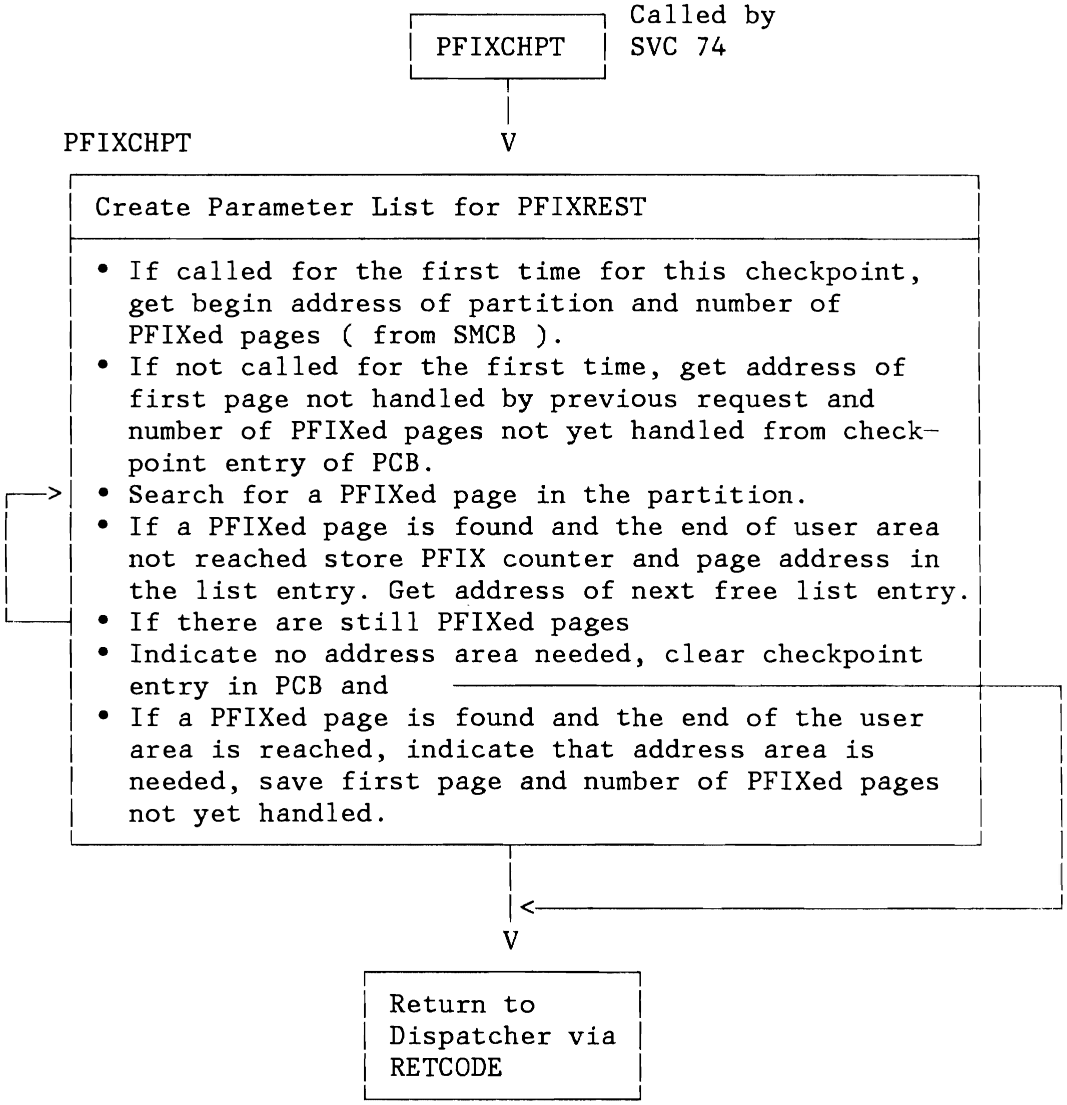

3
• If page TFIXED then
• If frame in PSQ (NFVP= OFF) then
• If page-out active then wait for I/O completion and try it again
• Find frame for exchange and if page I/O active then free frames kept by CCW translation until NPSQE > MINPSQE, reserve frame and RWAIT for I/O completion
• If unused frame found then exchange frames else
• If no frames are kept by CCW translation then else free frames
• If page kept by CCW translation then free pages else wait for pages
1
4
• If frame not removable from PSQ (MINPSQE<NPSQE) then
• Remove frame from PSQ and decrease NPSQE
• If frame in real partition then
• Set NFVP = ON frame requested by PFIX
4
2
1

Figure 191 (Part 2 of 2). Page Management: PFIXPGE Routine (370 Mode)



PFIXCHPT

Called by SVC 74

PFIXCHPT

Create Parameter List for PFIXREST

- If called for the first time for this checkpoint, get begin address of partition and number of PFIXed pages (from SMCB).
- If not called for the first time, get address of first page not handled by previous request and number of PFIXed pages not yet handled from checkpoint entry of PCB.
- Search for a PFIXed page in the partition.
- If a PFIXed page is found and the end of user area not reached store PFIX counter and page address in the list entry. Get address of next free list entry.
- If there are still PFIXed pages
- Indicate no address area needed, clear checkpoint entry in PCB and
- If a PFIXed page is found and the end of the user area is reached, indicate that address area is needed, save first page and number of PFIXed pages not yet handled.

Return to Dispatcher via RETCODE

Figure 192. Page Management: PFIXCHPT Routine

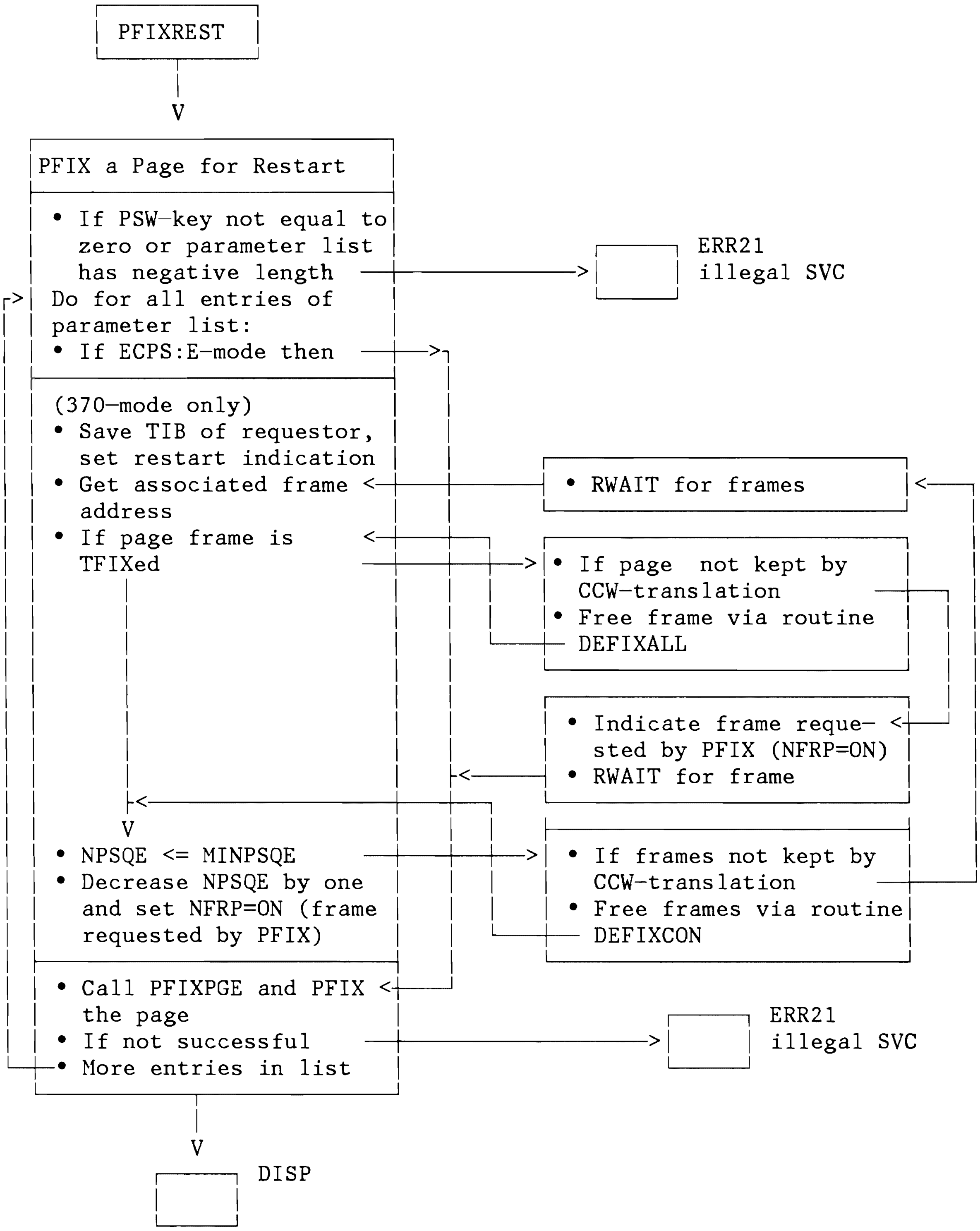

PFIXREST
PFIX a Page for Restart
• If PSW-key not equal to zero or parameter list has negative length
Do for all entries of parameter list:
• If ECPS:E-mode then
ERR21 illegal SVC
(370-mode only)
• Save TIB of requestor, set restart indication
• Get associated frame address
• If page frame is TFIXed
• RWAIT for frames
• If page not kept by CCW-translation
• Free frame via routine DEFIXALL
• Indicate frame requested by PFIX (NFRP=ON)
• RWAIT for frame
• NPSQE <= MINPSQE
• Decrease NPSQE by one and set NFRP=ON (frame requested by PFIX)
• If frames not kept by CCW-translation
• Free frames via routine DEFIXCON
• Call PFIXPGE and PFIX the page
• If not successful
• More entries in list
ERR21 illegal SVC
DISP

Figure 193. Page Management: PFIXREST Routine

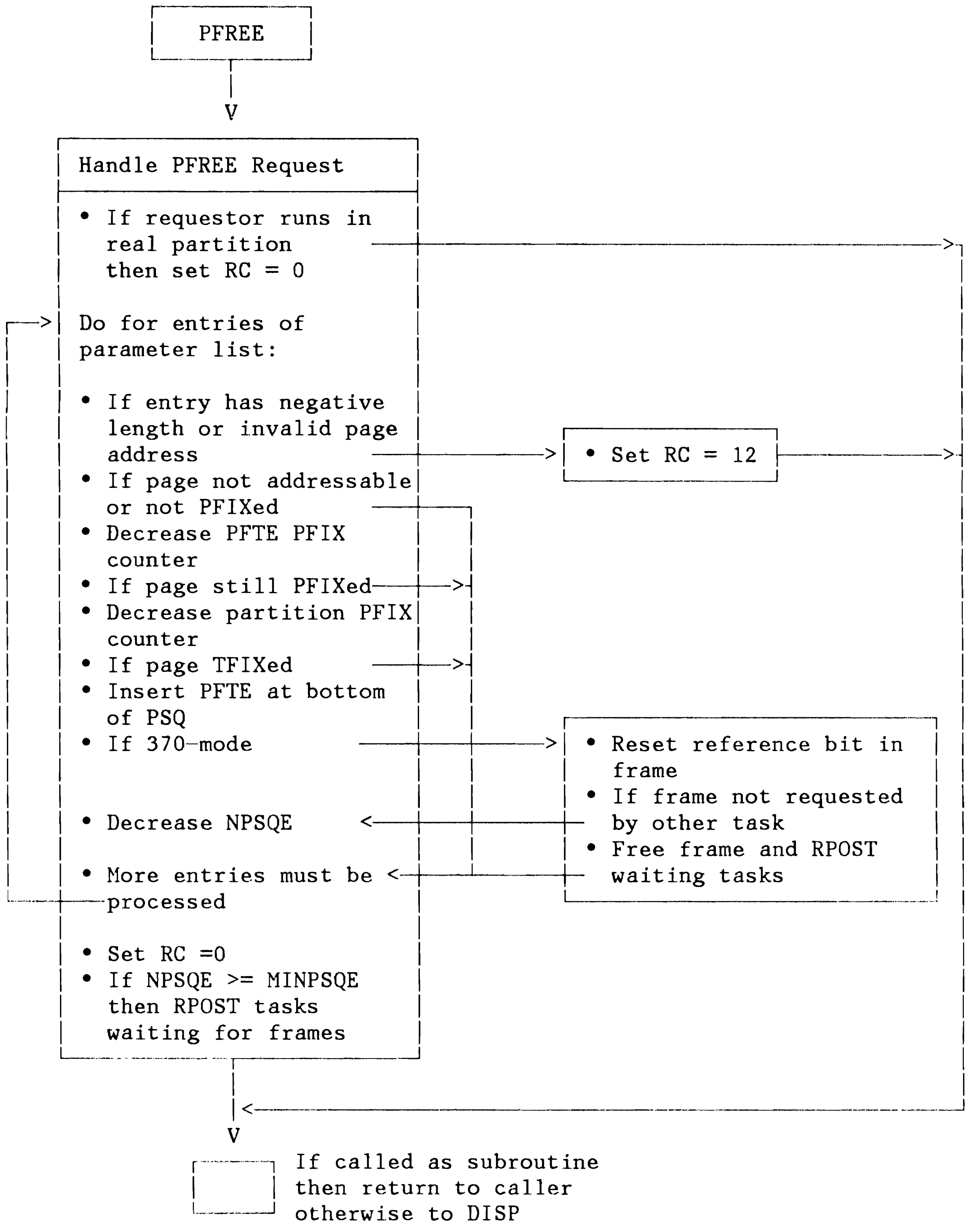

PFREE
Handle PFREE Request
• If requestor runs in real partition then set RC = 0
Do for entries of parameter list:
• If entry has negative length or invalid page address
• If page not addressable or not PFIXed
• Decrease PFTE PFIX counter
• If page still PFIXed
• Decrease partition PFIX counter
• If page TFIXed
• Insert PFTE at bottom of PSQ
• If 370-mode
• Decrease NPSQE
• More entries must be processed
• Set RC =0
• If NPSQE >= MINPSQE then RPOST tasks waiting for frames
• Set RC = 12
• Reset reference bit in frame
• If frame not requested by other task
• Free frame and RPOST waiting tasks
If called as subroutine then return to caller otherwise to DISP

Figure 194. Page Management: PFREE Routine

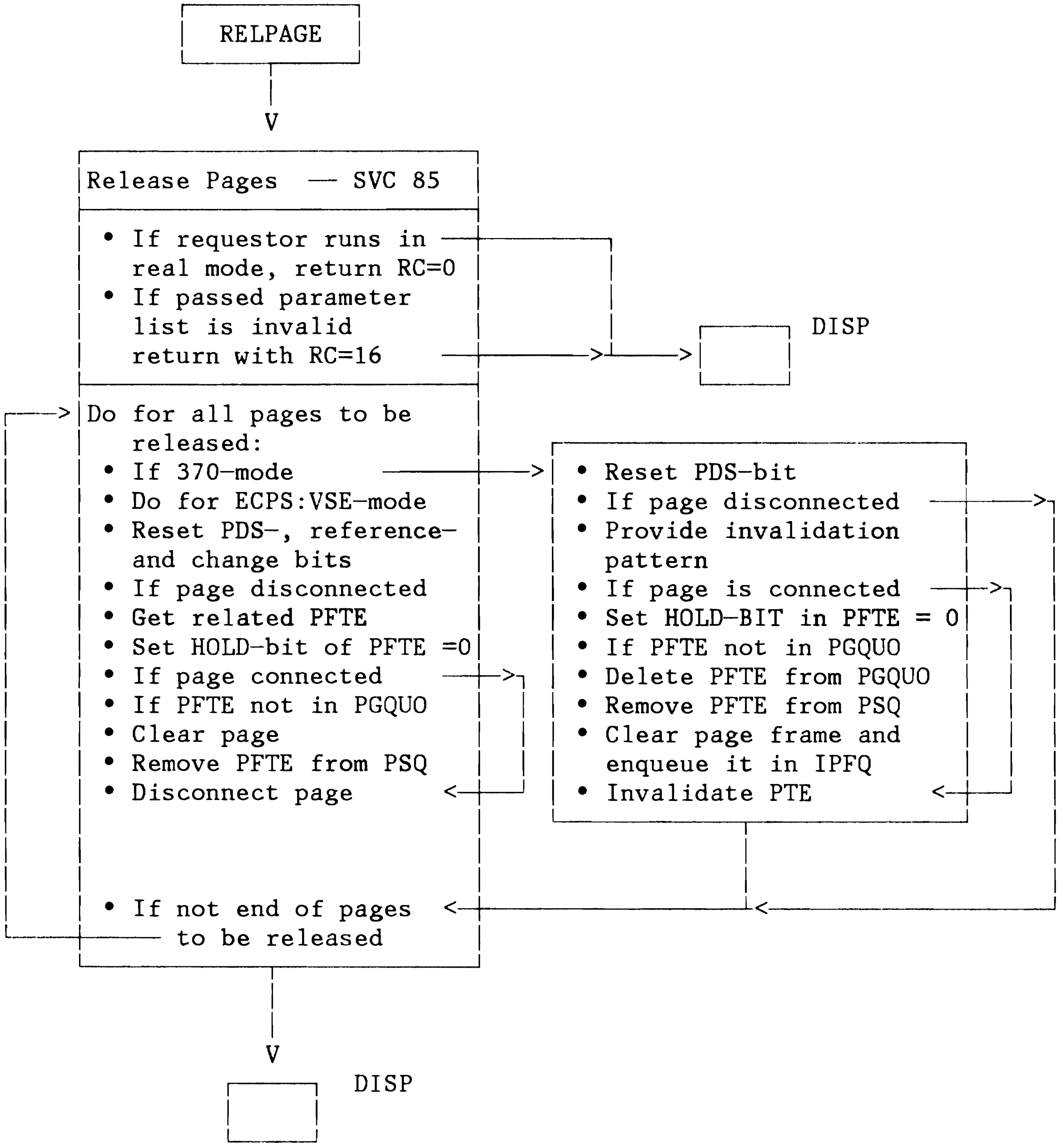


Figure 195. Page Management: RELPAGE Routine

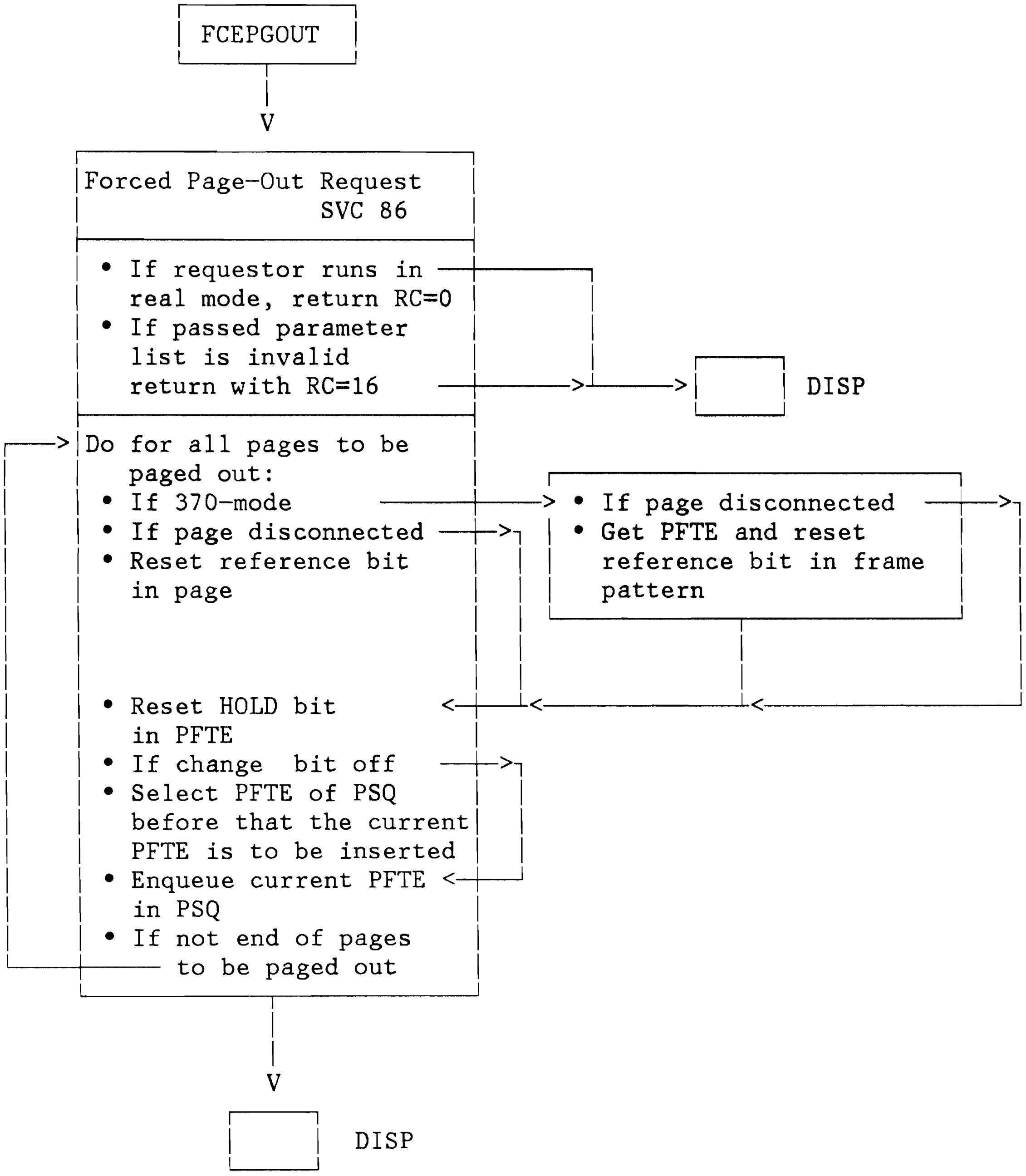

FCEPGOUT
Forced Page-Out Request SVC 86
• If requestor runs in real mode, return RC=0
• If passed parameter list is invalid return with RC=16
DISP
Do for all pages to be paged out:
• If 370-mode
• If page disconnected
• Reset reference bit in page
• If page disconnected
• Get PFTE and reset reference bit in frame pattern
• Reset HOLD bit in PFTE
• If change bit off
• Select PFTE of PSQ before that the current PFTE is to be inserted
• Enqueue current PFTE in PSQ
• If not end of pages to be paged out
DISP

Figure 196. Page Management: FCEPGOUT Routines

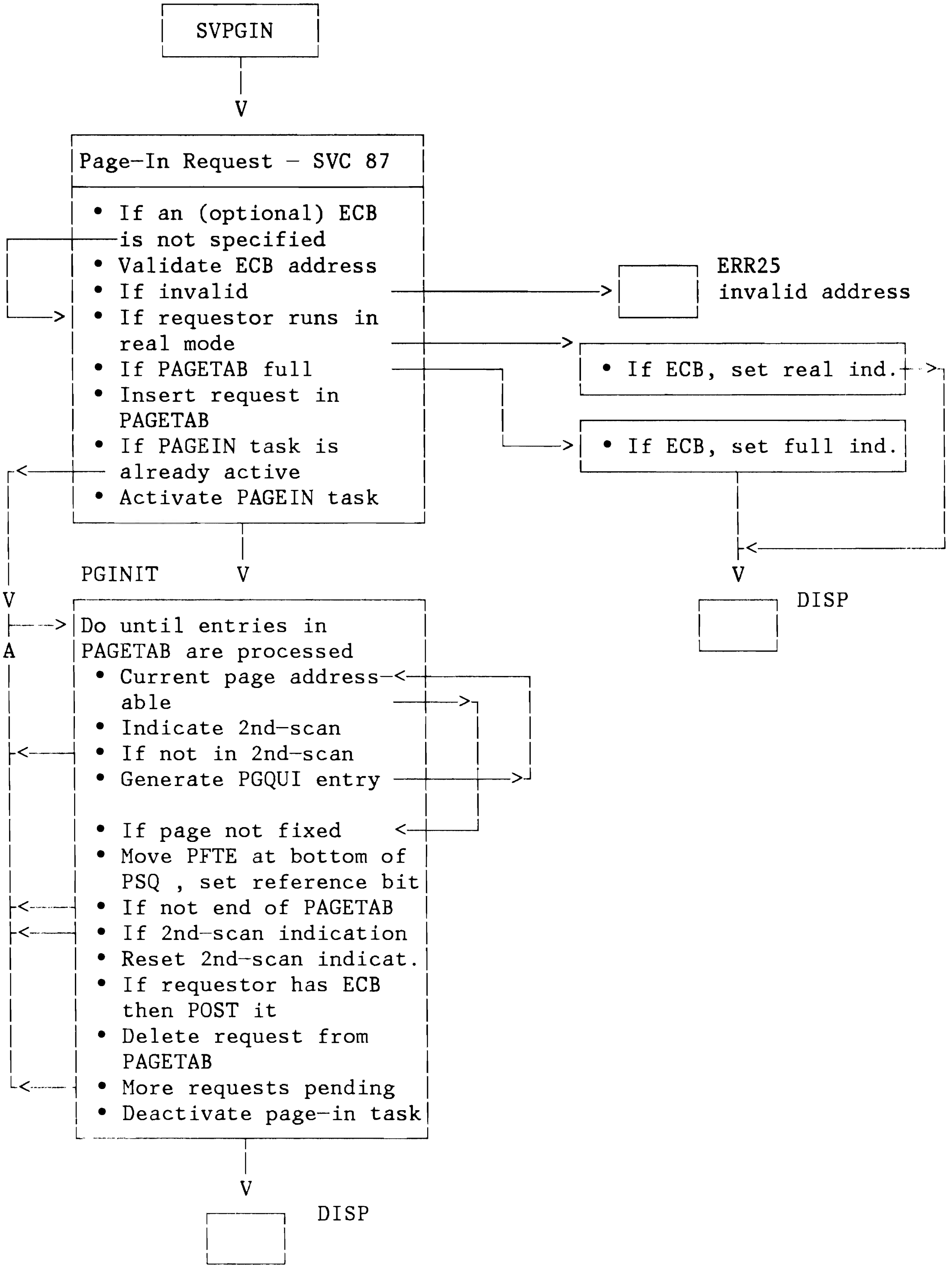

SVPGIN
Page-In Request - SVC 87
• If an (optional) ECB is not specified
• Validate ECB address
• If invalid
• If requestor runs in real mode
• If PAGETAB full
• Insert request in PAGETAB
• If PAGEIN task is already active
• Activate PAGEIN task
ERR25
invalid address
• If ECB, set real ind.
• If ECB, set full ind.
DISP
PGINIT
Do until entries in PAGETAB are processed
• Current page address-able
• Indicate 2nd-scan
• If not in 2nd-scan
• Generate PGQUI entry
• If page not fixed
• Move PFTE at bottom of PSQ , set reference bit
• If not end of PAGETAB
• If 2nd-scan indication
• Reset 2nd-scan indicat.
• If requestor has ECB then POST it
• Delete request from PAGETAB
• More requests pending
• Deactivate page-in task
DISP

Figure 197. Page Management: PAGEIN Routine

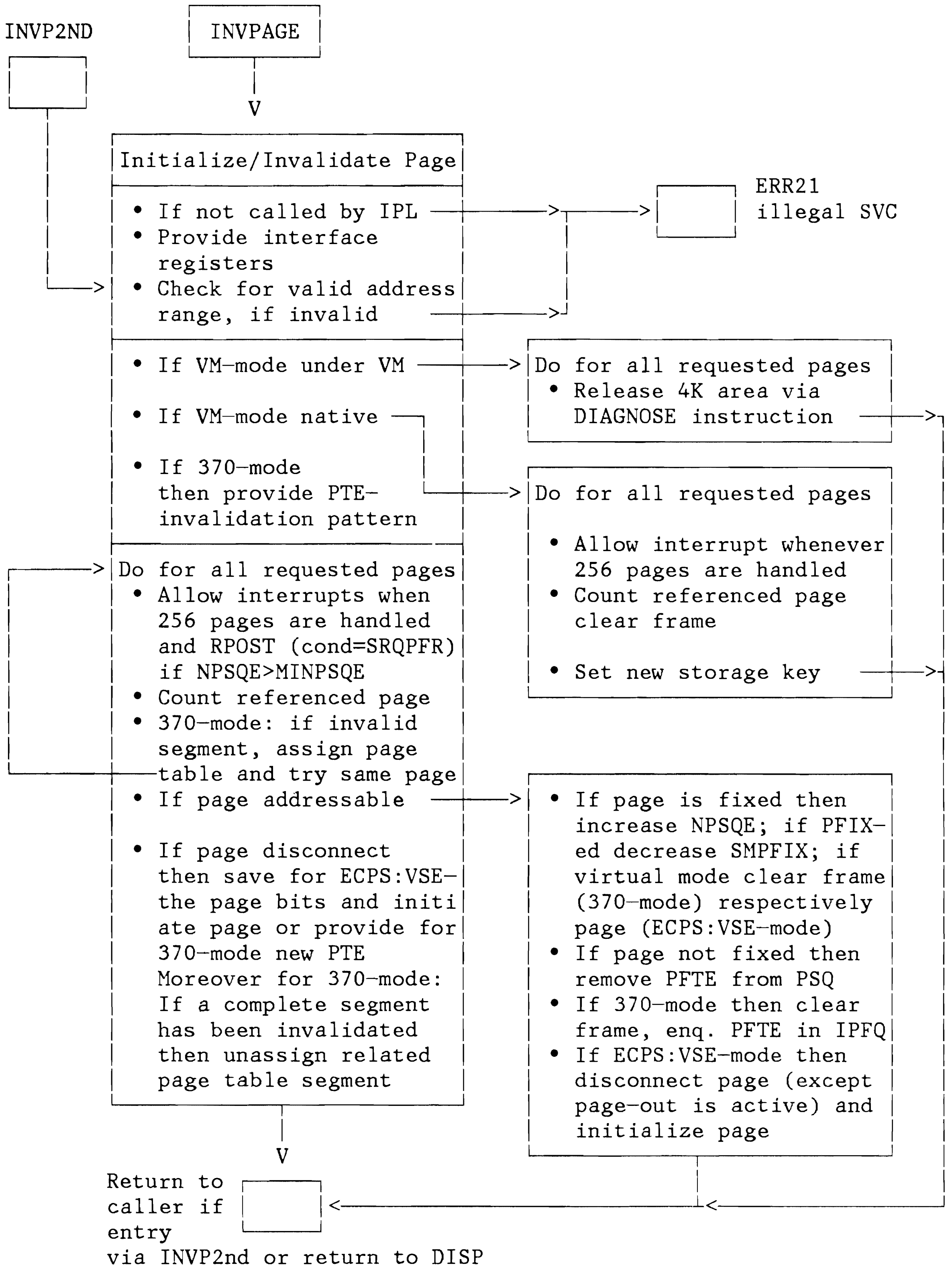
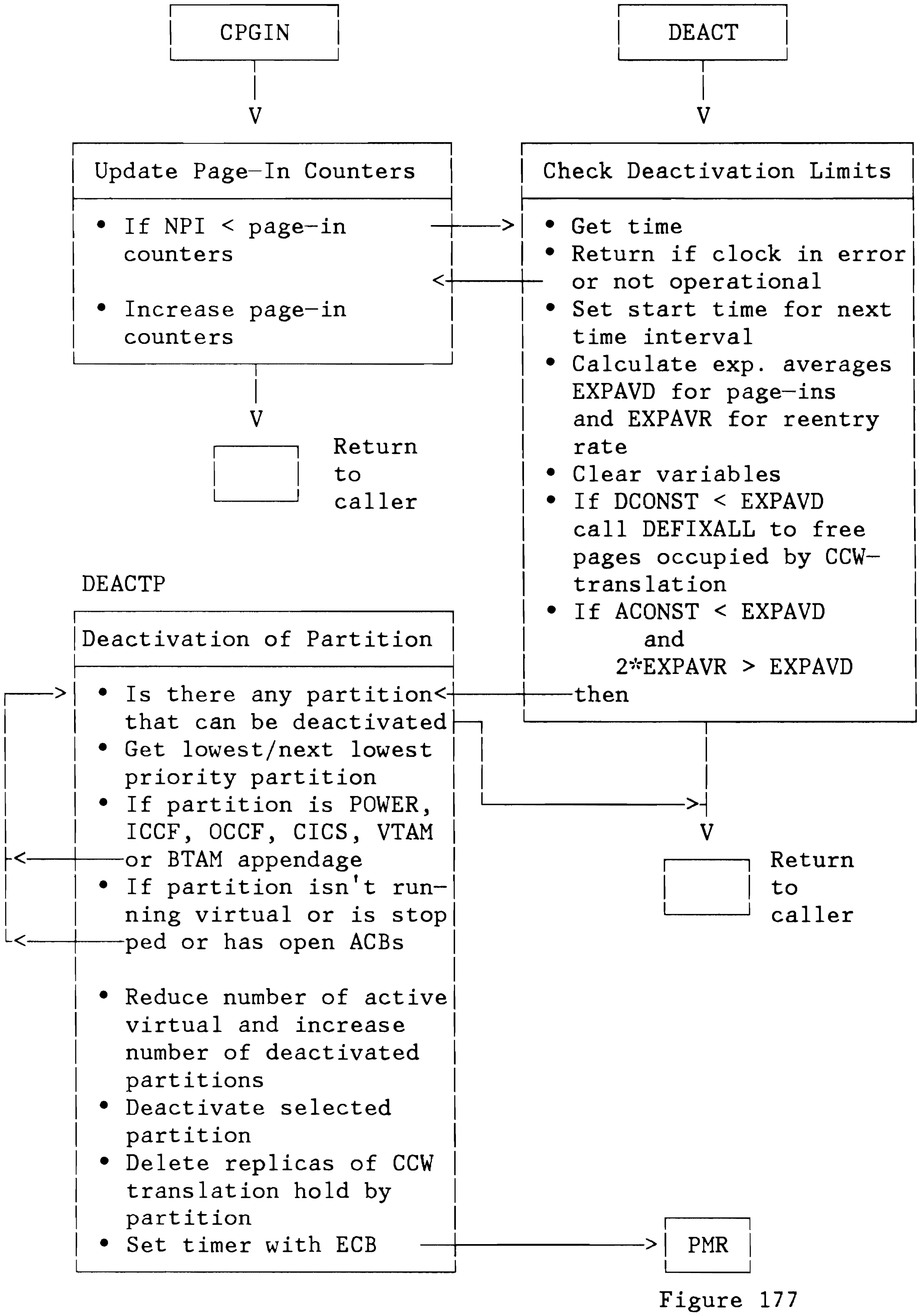

INVP2ND
INVPAGE
Initialize/Invalidate Page
• If not called by IPL
• Provide interface registers
• Check for valid address range, if invalid
ERR21 illegal SVC
• If VM-mode under VM
• If VM-mode native
• If 370-mode then provide PTE-invalidation pattern
Do for all requested pages
• Release 4K area via DIAGNOSE instruction
Do for all requested pages
• Allow interrupt whenever 256 pages are handled
• Count referenced page clear frame
• Set new storage key
Do for all requested pages
• Allow interrupts when 256 pages are handled and RPOST (cond=SRQPFR) if NPSQE>MINPSQE
• Count referenced page
• 370-mode: if invalid segment, assign page table and try same page
• If page addressable
• If page disconnect then save for ECPS:VSE-the page bits and initi ate page or provide for 370-mode new PTE Moreover for 370-mode: If a complete segment has been invalidated then unassign related page table segment
• If page is fixed then increase NPSQE; if PFIX-ed decrease SMPFIX; if virtual mode clear frame (370-mode) respectively page (ECPS:VSE-mode)
• If page not fixed then remove PFTE from PSQ
• If 370-mode then clear frame, enq. PFTE in IPFQ
• If ECPS:VSE-mode then disconnect page (except page-out is active) and initialize page
Return to caller if entry via INVP2nd or return to DISP

Figure 198. Page Management: INVPAGE Routine



Update Page-In Counters

- If NPI < page-in counters
- Increase page-in counters

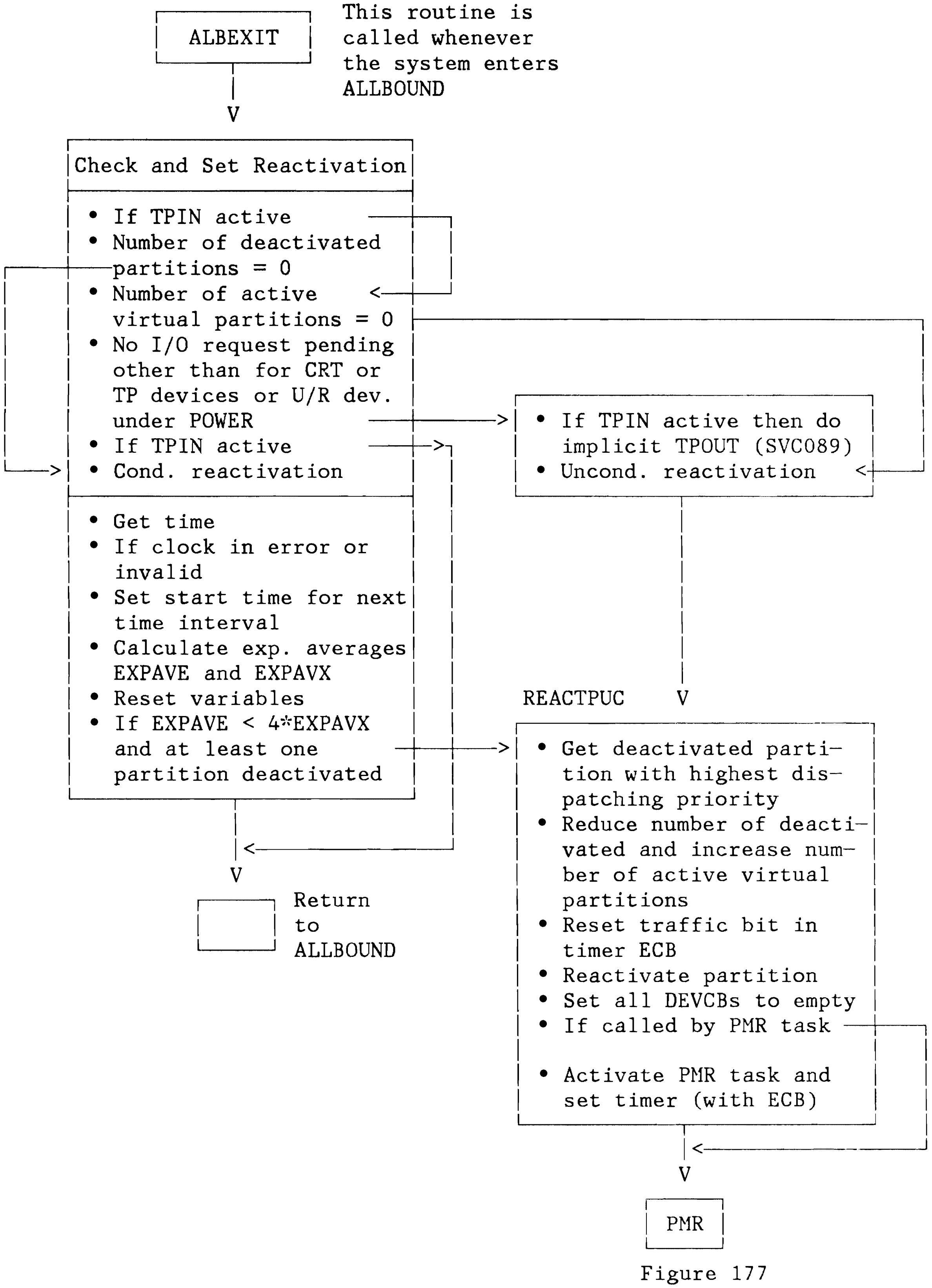
Check Deactivation Limits

- Get time
- Return if clock in error or not operational
- Set start time for next time interval
- Calculate exp. averages EXPAVD for page-ins and EXPAVR for reentry rate
- Clear variables
- If DCONST < EXPAVD call DEFIXALL to free pages occupied by CCW-translation
- If ACONST < EXPAVD and 2*EXPAVR > EXPAVD then

Deactivation of Partition

- Is there any partition that can be deactivated
- Get lowest/next lowest priority partition
- If partition is POWER, ICCF, OCCF, CICS, VTAM or BTAM appendage
- If partition isn't running virtual or is stopped or has open ACBs
- Reduce number of active virtual and increase number of deactivated partitions
- Deactivate selected partition
- Delete replicas of CCW translation hold by partition
- Set timer with ECB

Figure 199. Load Leveling: Deactivation of Partition



ALBEXIT

This routine is called whenever the system enters ALLBOUND

Check and Set Reactivation

- If TPIN active
- Number of deactivated partitions = 0
- Number of active virtual partitions = 0
- No I/O request pending other than for CRT or TP devices or U/R dev. under POWER
- If TPIN active
- Cond. reactivation

- Get time
- If clock in error or invalid
- Set start time for next time interval
- Calculate exp. averages EXPAVE and EXPAVX
- Reset variables
- If EXPAVE < 4*EXPAVX and at least one partition deactivated

Return to ALLBOUND

- If TPIN active then do implicit TPOUT (SVC089)
- Uncond. reactivation

REACTPUC

- Get deactivated partition with highest dispatching priority
- Reduce number of deactivated and increase number of active virtual partitions
- Reset traffic bit in timer ECB
- Reactivate partition
- Set all DEVCBs to empty
- If called by PMR task

- Activate PMR task and set timer (with ECB)

PMR

Figure 177

Figure 200. Load Leveling: Reactivation of Partition

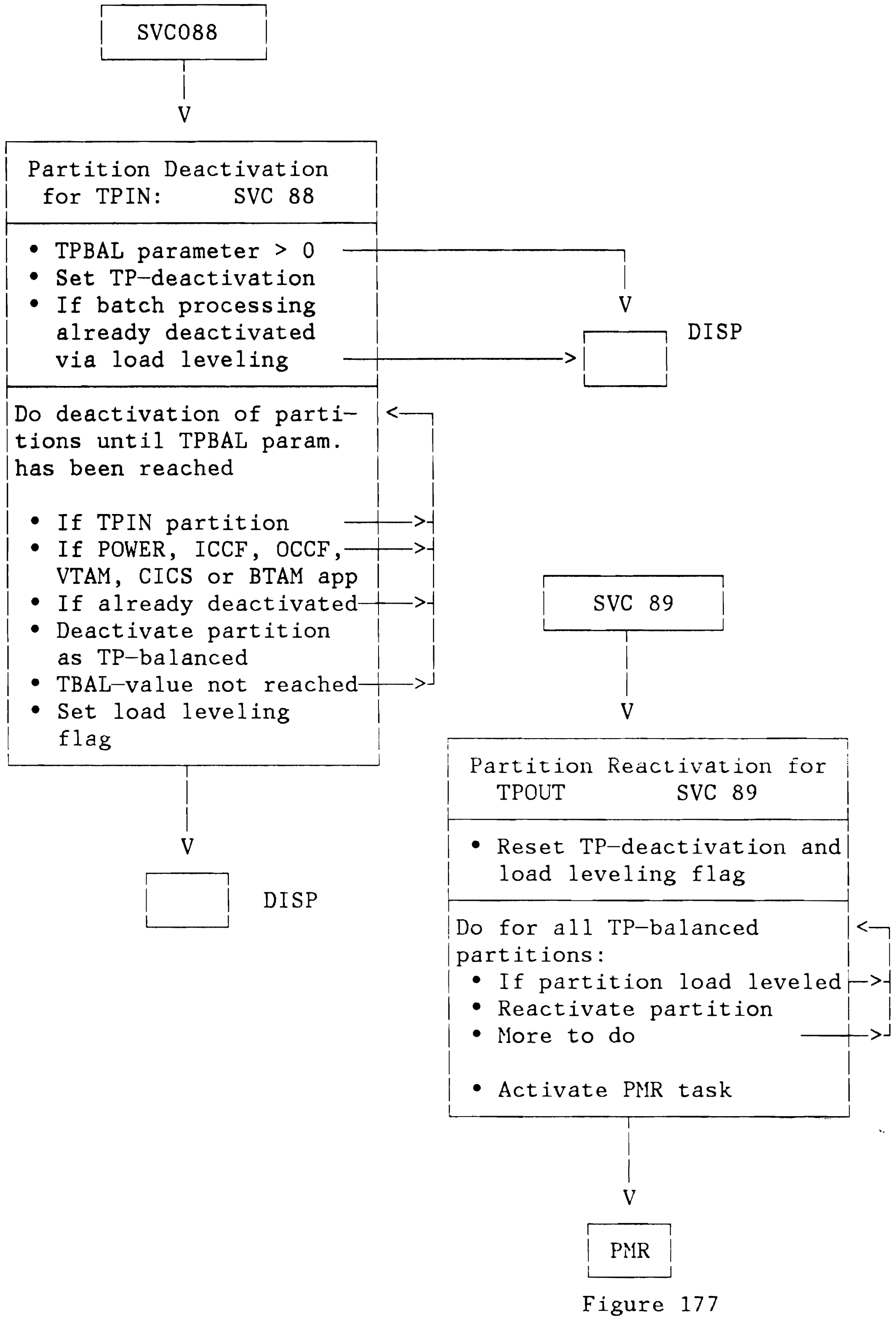

SVC088
Partition Deactivation for TPIN: SVC 88
• TPBAL parameter > 0
• Set TP-deactivation
• If batch processing already deactivated via load leveling
DISP
Do deactivation of partitions until TPBAL param. has been reached
• If TPIN partition
• If POWER, ICCF, OCCF, VTAM, CICS or BTAM app
• If already deactivated
• Deactivate partition as TP-balanced
• TBAL-value not reached
• Set load leveling flag
DISP
SVC 89
Partition Reactivation for TPOUT SVC 89
• Reset TP-deactivation and load leveling flag
Do for all TP-balanced partitions:
• If partition load leveled
• Reactivate partition
• More to do
• Activate PMR task
PMR
Figure 177

Figure 201. TP Balancing of Partitions

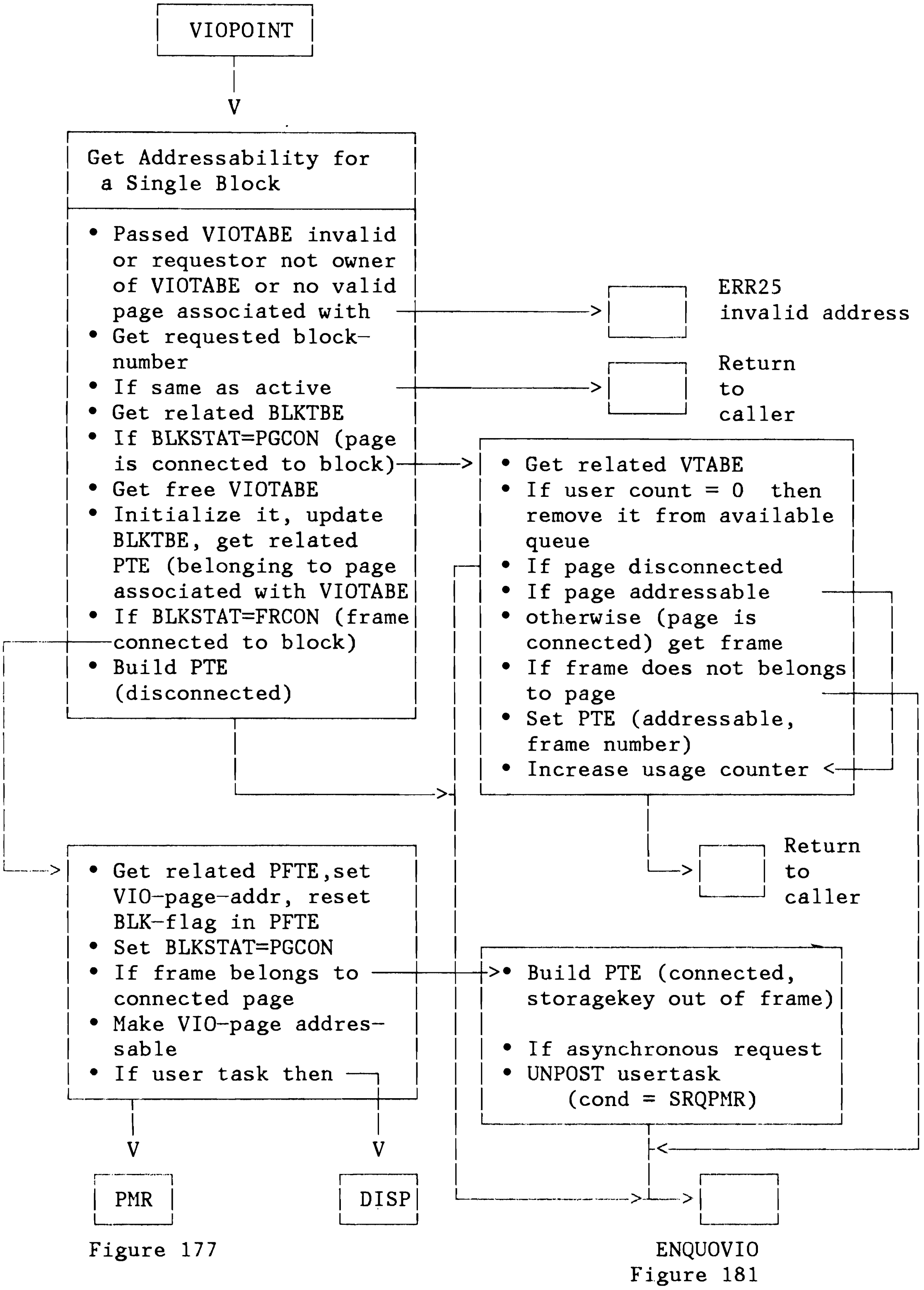
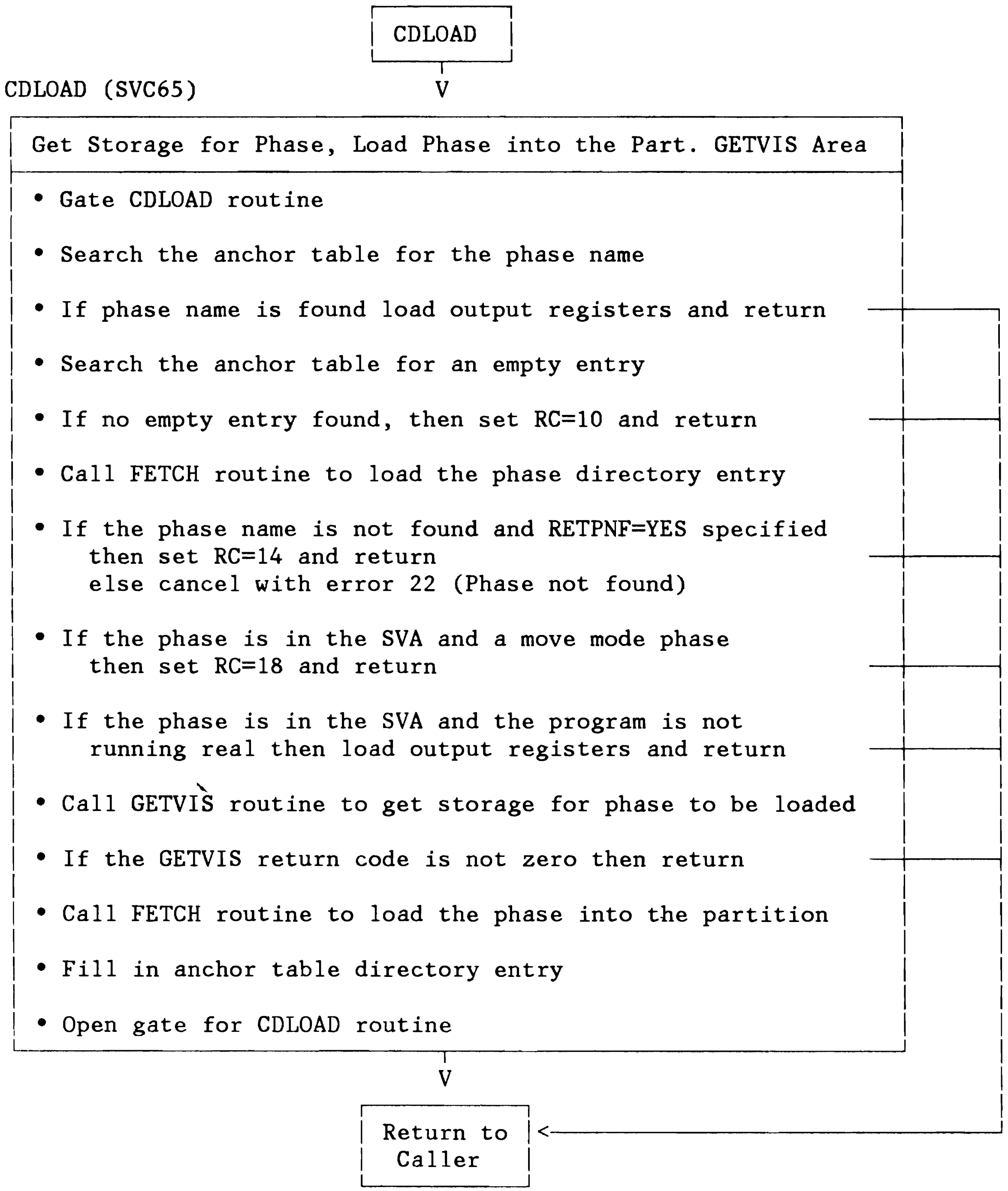

VIOPOINT
Get Addressability for a Single Block
• Passed VIOTABE invalid or requestor not owner of VIOTABE or no valid page associated with
• Get requested block-number
• If same as active
• Get related BLKTBE
• If BLKSTAT=PGCON (page is connected to block)
• Get free VIOTABE
• Initialize it, update BLKTBE, get related PTE (belonging to page associated with VIOTABE
• If BLKSTAT=FRCON (frame connected to block)
• Build PTE (disconnected)
ERR25 invalid address
Return to caller
• Get related VTABE
• If user count = 0 then remove it from available queue
• If page disconnected
• If page addressable
• otherwise (page is connected) get frame
• If frame does not belongs to page
• Set PTE (addressable, frame number)
• Increase usage counter
Return to caller
• Get related PFTE,set VIO-page-addr, reset BLK-flag in PFTE
• Set BLKSTAT=PGCON
• If frame belongs to connected page
• Make VIO-page addres-sable
• If user task then
• Build PTE (connected, storagekey out of frame)
• If asynchronous request
• UNPOST usertask (cond = SRQPMR)
PMR
Figure 177
DISP
ENQUOVIO
Figure 181

Figure 202. Virtual I/O: VIOPOINT Service

CDLOAD and GETVIS/FREEVIS Routines



CDLOAD

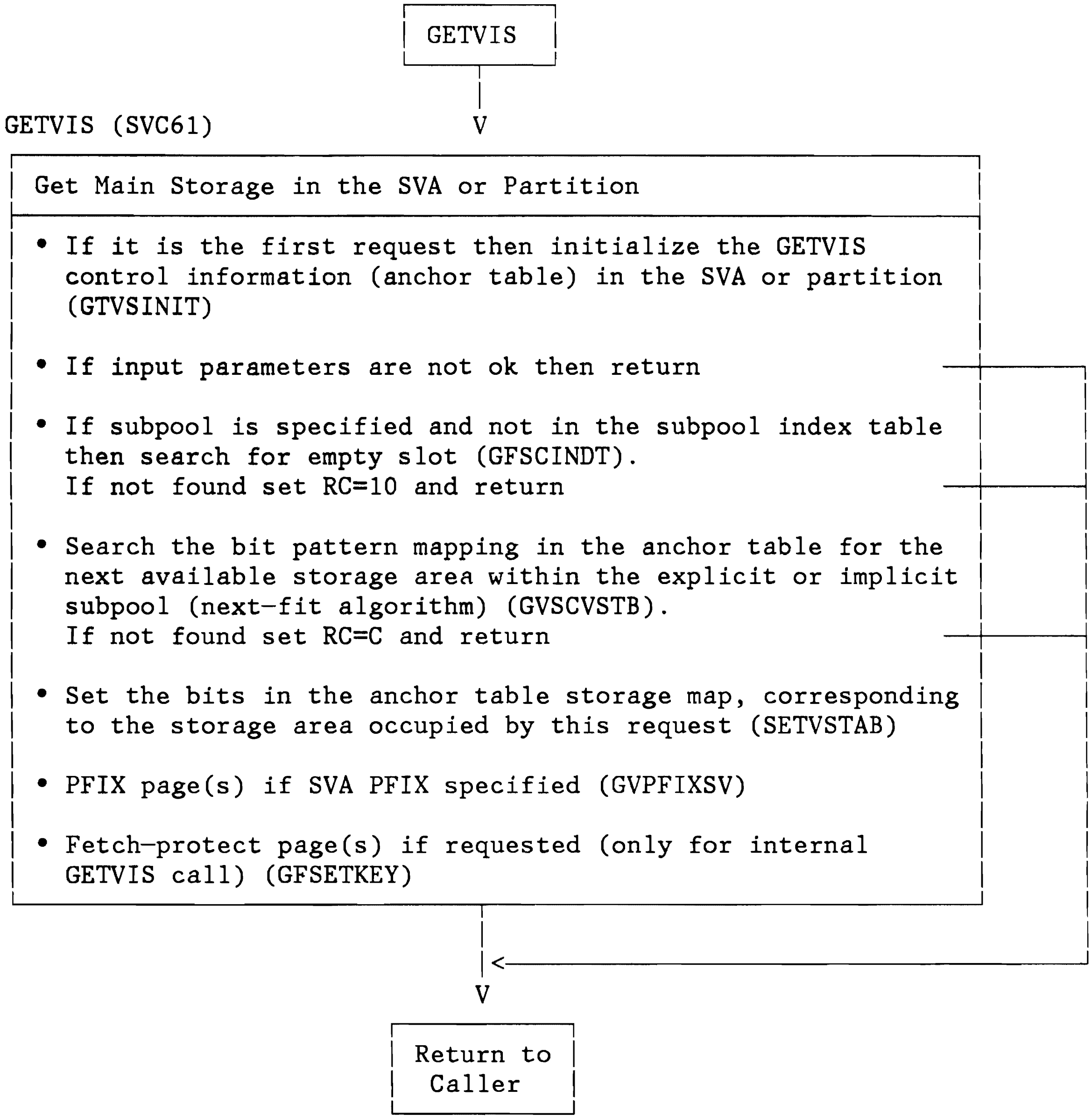
CDLOAD (SVC65)

Get Storage for Phase, Load Phase into the Part. GETVIS Area

- Gate CDLOAD routine
- Search the anchor table for the phase name
- If phase name is found load output registers and return
- Search the anchor table for an empty entry
- If no empty entry found, then set RC=10 and return
- Call FETCH routine to load the phase directory entry
- If the phase name is not found and RETPNF=YES specified then set RC=14 and return else cancel with error 22 (Phase not found)
- If the phase is in the SVA and a move mode phase then set RC=18 and return
- If the phase is in the SVA and the program is not running real then load output registers and return
- Call GETVIS routine to get storage for phase to be loaded
- If the GETVIS return code is not zero then return
- Call FETCH routine to load the phase into the partition
- Fill in anchor table directory entry
- Open gate for CDLOAD routine

Return to Caller

Figure 203. Storage Management: CDLOAD Routine



GETVIS

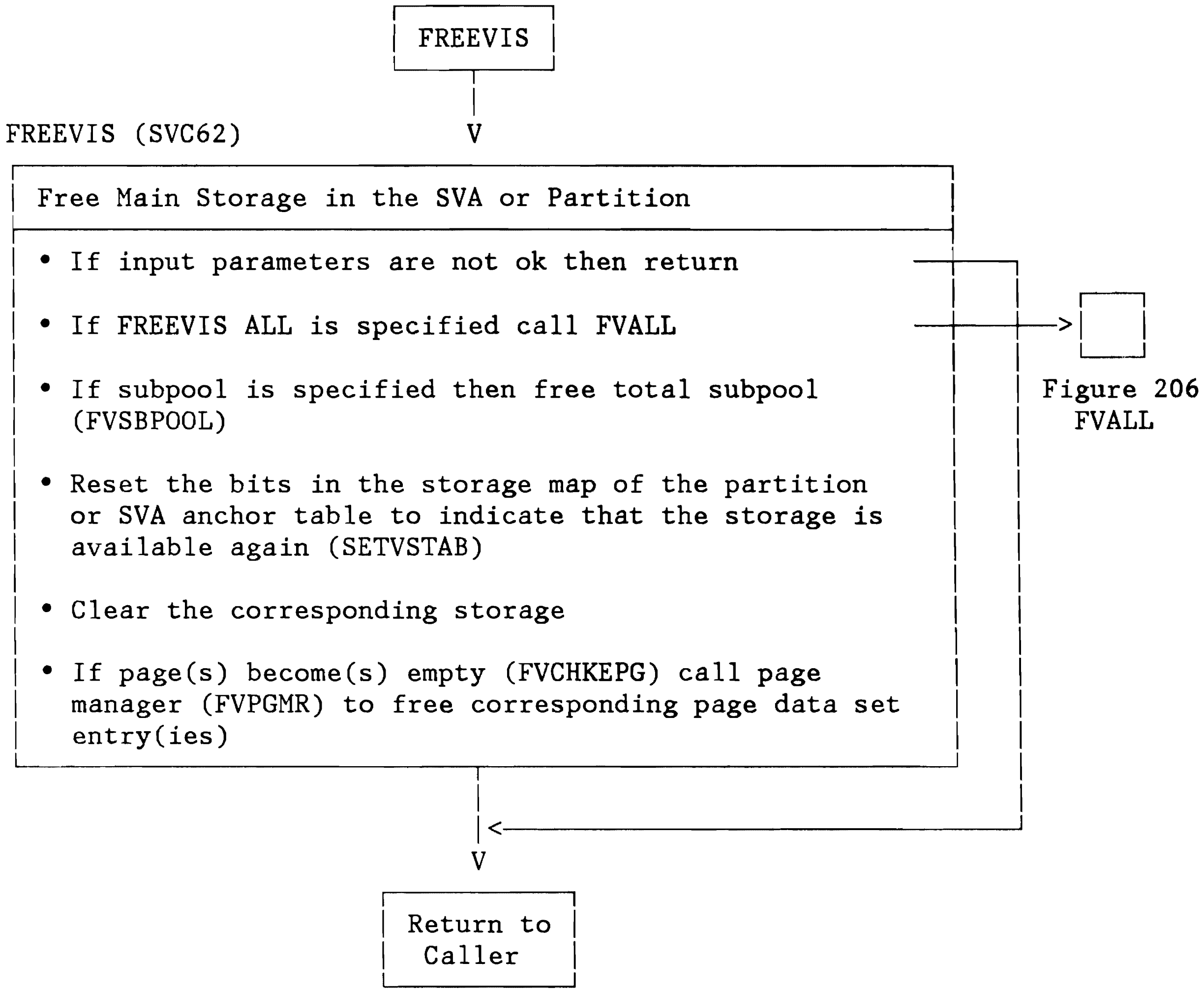
GETVIS (SVC61)

Get Main Storage in the SVA or Partition

- If it is the first request then initialize the GETVIS control information (anchor table) in the SVA or partition (GTVSINIT)
- If input parameters are not ok then return
- If subpool is specified and not in the subpool index table then search for empty slot (GFSCINDT).
  If not found set RC=10 and return
- Search the bit pattern mapping in the anchor table for the next available storage area within the explicit or implicit subpool (next-fit algorithm) (GVSCVSTB).
  If not found set RC=C and return
- Set the bits in the anchor table storage map, corresponding to the storage area occupied by this request (SETVSTAB)
- PFIX page(s) if SVA PFIX specified (GVPFIXSV)
- Fetch-protect page(s) if requested (only for internal GETVIS call) (GFSETKEY)

Return to Caller

Figure 204. Storage Management: GETVIS Routine



FREEVIS

FREEVIS (SVC62)

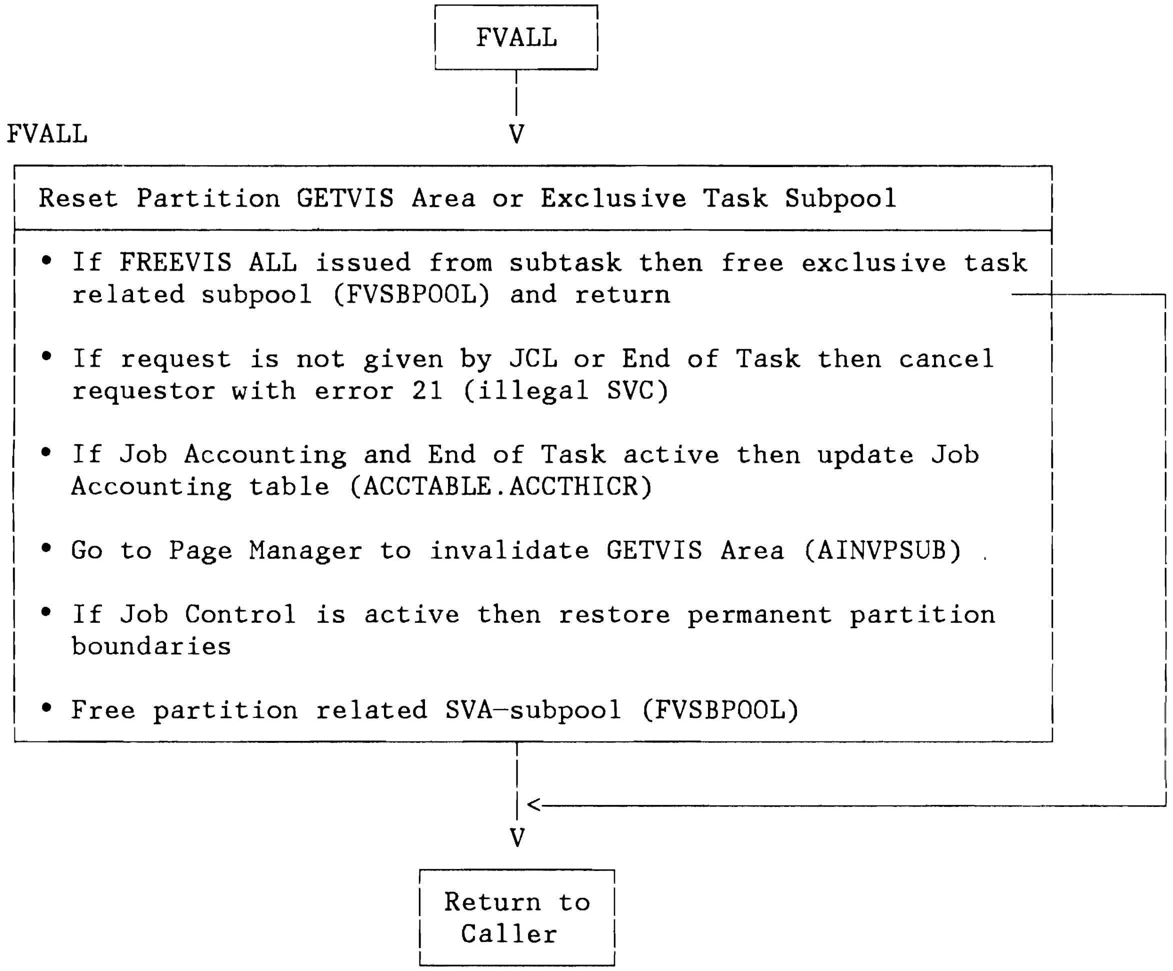
Free Main Storage in the SVA or Partition

- If input parameters are not ok then return
- If FREEVIS ALL is specified call FVALL
- If subpool is specified then free total subpool (FVSBPOOL)
- Reset the bits in the storage map of the partition or SVA anchor table to indicate that the storage is available again (SETVSTAB)
- Clear the corresponding storage
- If page(s) become(s) empty (FVCHKEPG) call page manager (FVPGMR) to free corresponding page data set entry(ies)

Figure 206
FVALL

Return to Caller

Figure 205. Storage Management: FREEVIS Routine

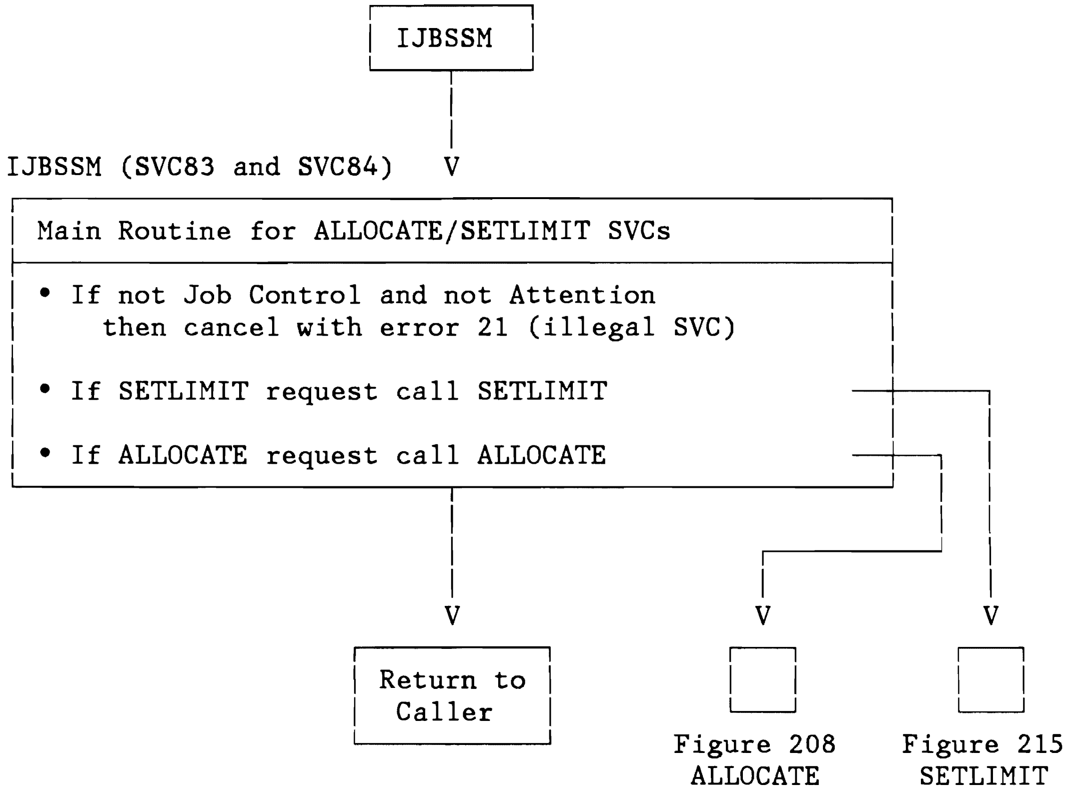


Reset Partition GETVIS Area or Exclusive Task Subpool

- If FREEVIS ALL issued from subtask then free exclusive task related subpool (FVSBPOOL) and return
- If request is not given by JCL or End of Task then cancel requestor with error 21 (illegal SVC)
- If Job Accounting and End of Task active then update Job Accounting table (ACCTABLE.ACCTHICR)
- Go to Page Manager to invalidate GETVIS Area (AINVPSUB)
- If Job Control is active then restore permanent partition boundaries
- Free partition related SVA–subpool (FVSBPOOL)

Figure 206. Storage Management: FREEVIS ALL Routine

ALLOCATE and SETLIMIT Routines



IJBSSM

IJBSSM (SVC83 and SVC84)

Main Routine for ALLOCATE/SETLIMIT SVCs

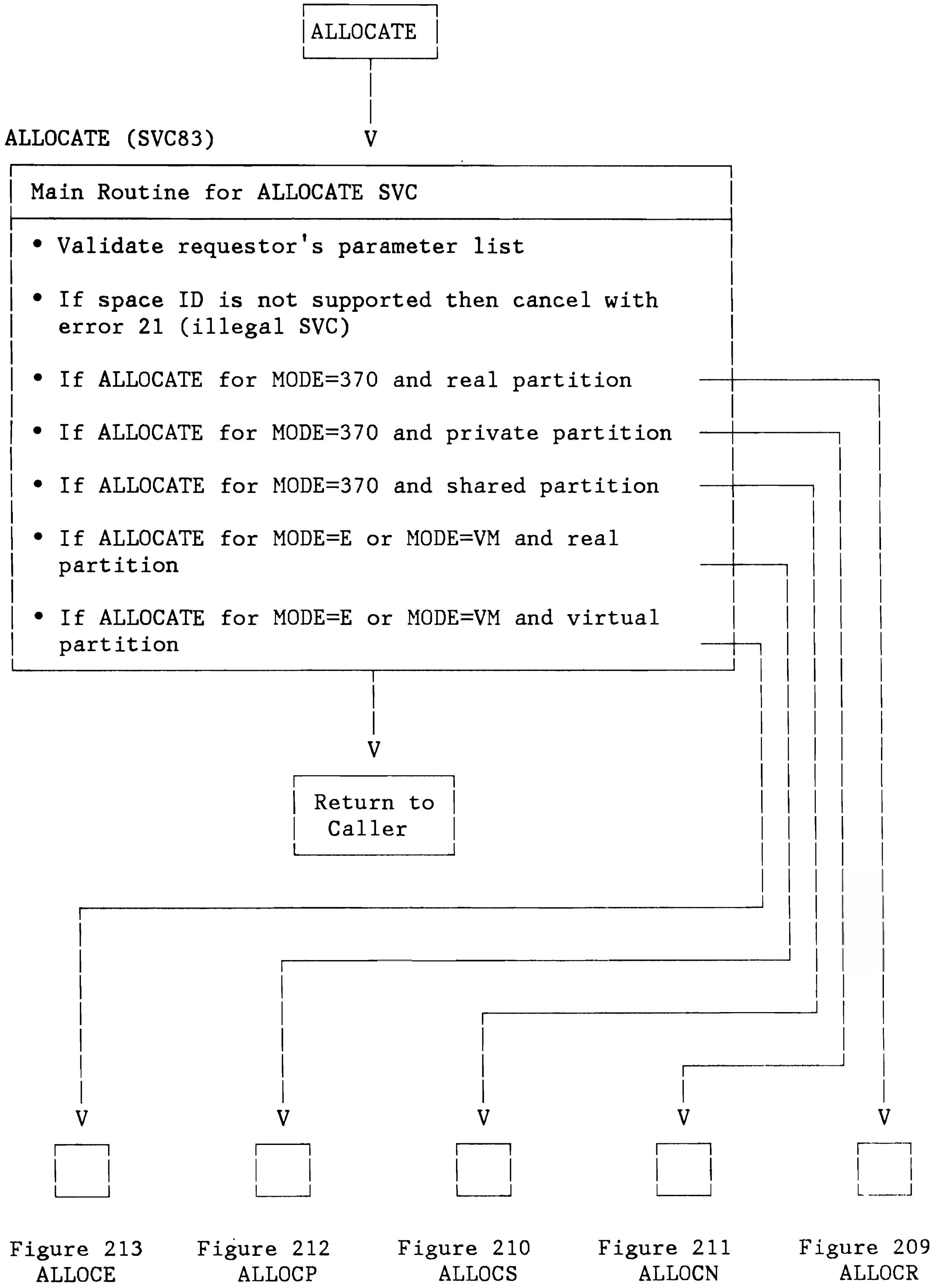
- If not Job Control and not Attention then cancel with error 21 (illegal SVC)
- If SETLIMIT request call SETLIMIT
- If ALLOCATE request call ALLOCATE

Return to Caller

Figure 208 ALLOCATE

Figure 215 SETLIMIT

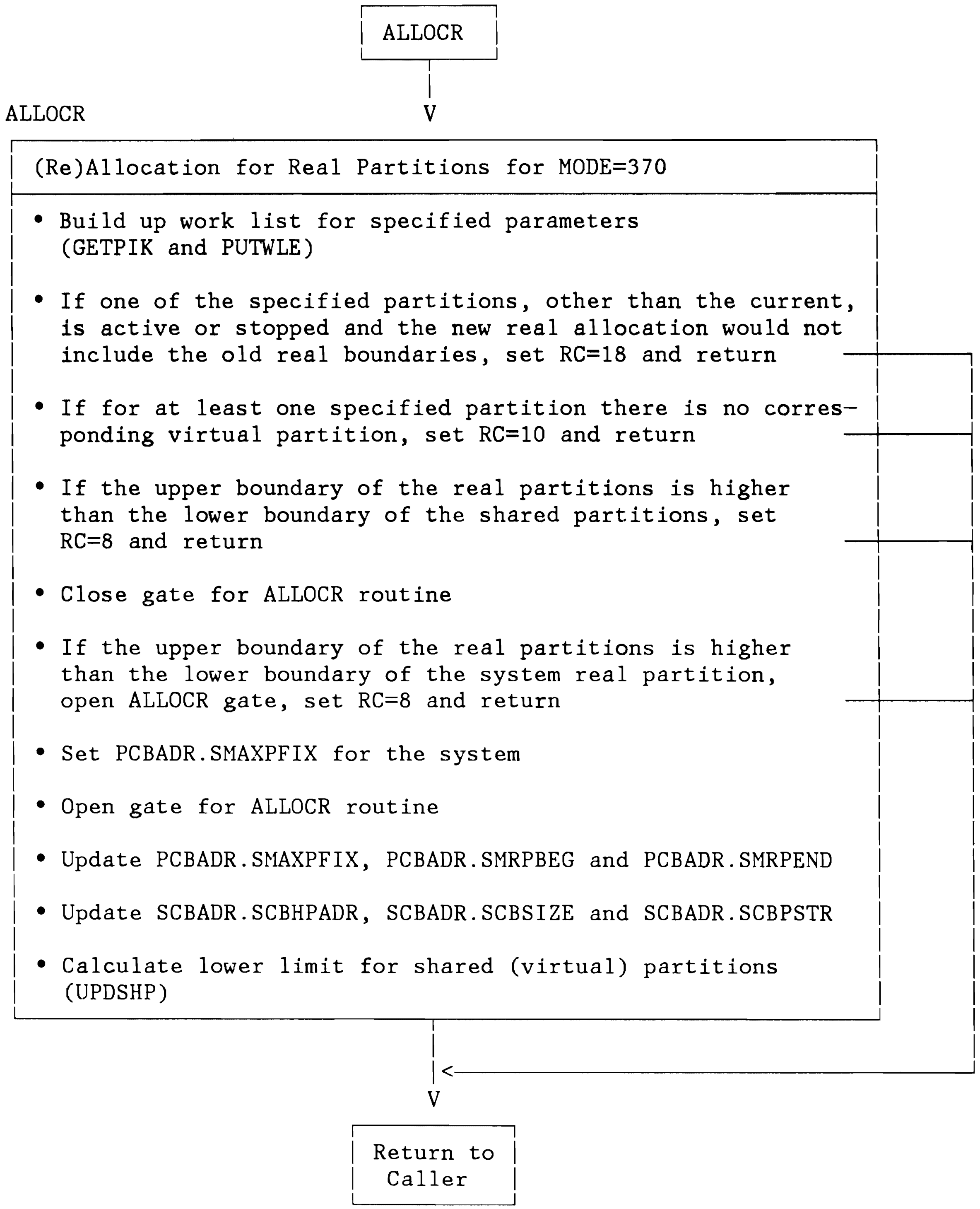
Figure 207. Storage Management: Entry Routine for ALLOCATE/SETLIMIT



Main Routine for ALLOCATE SVC

- Validate requestor's parameter list
- If space ID is not supported then cancel with error 21 (illegal SVC)
- If ALLOCATE for MODE=370 and real partition
- If ALLOCATE for MODE=370 and private partition
- If ALLOCATE for MODE=370 and shared partition
- If ALLOCATE for MODE=E or MODE=VM and real partition
- If ALLOCATE for MODE=E or MODE=VM and virtual partition

Figure 208. Storage Management: ALLOCATE Main Routine



ALLOCR

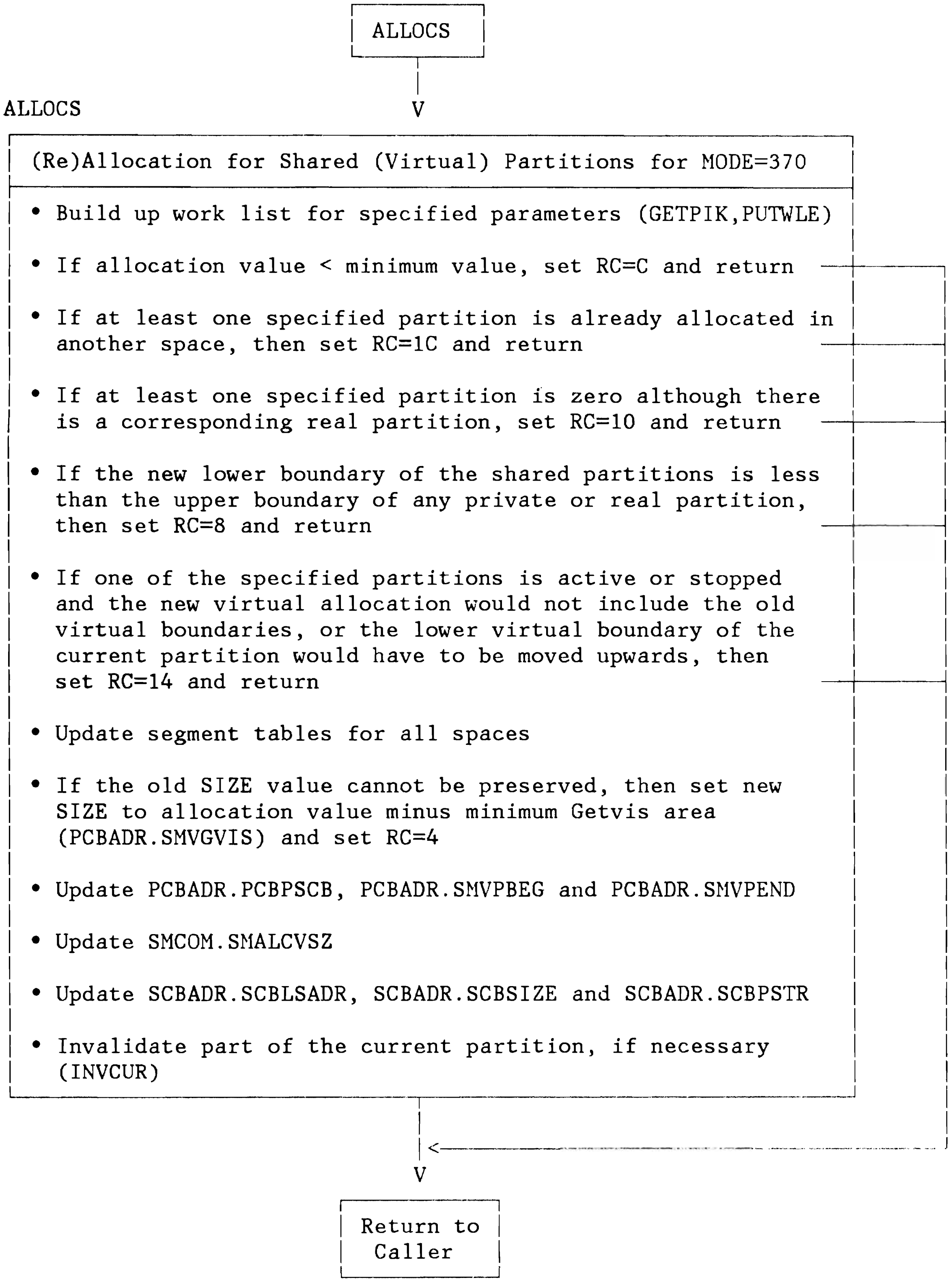
ALLOCR

(Re)Allocation for Real Partitions for MODE=370

- Build up work list for specified parameters (GETPIK and PUTWLE)
- If one of the specified partitions, other than the current, is active or stopped and the new real allocation would not include the old real boundaries, set RC=18 and return
- If for at least one specified partition there is no corresponding virtual partition, set RC=10 and return
- If the upper boundary of the real partitions is higher than the lower boundary of the shared partitions, set RC=8 and return
- Close gate for ALLOCR routine
- If the upper boundary of the real partitions is higher than the lower boundary of the system real partition, open ALLOCR gate, set RC=8 and return
- Set PCBADR.SMAXPFIX for the system
- Open gate for ALLOCR routine
- Update PCBADR.SMAXPFIX, PCBADR.SMRPBEG and PCBADR.SMRPEND
- Update SCBADR.SCBHPADR, SCBADR.SCBSIZE and SCBADR.SCBPSTR
- Calculate lower limit for shared (virtual) partitions (UPDSHP)

Return to Caller

Figure 209. Storage Management(ALLOCR): Allocation Routine for Real Partitions for MODE=370



ALLOCS

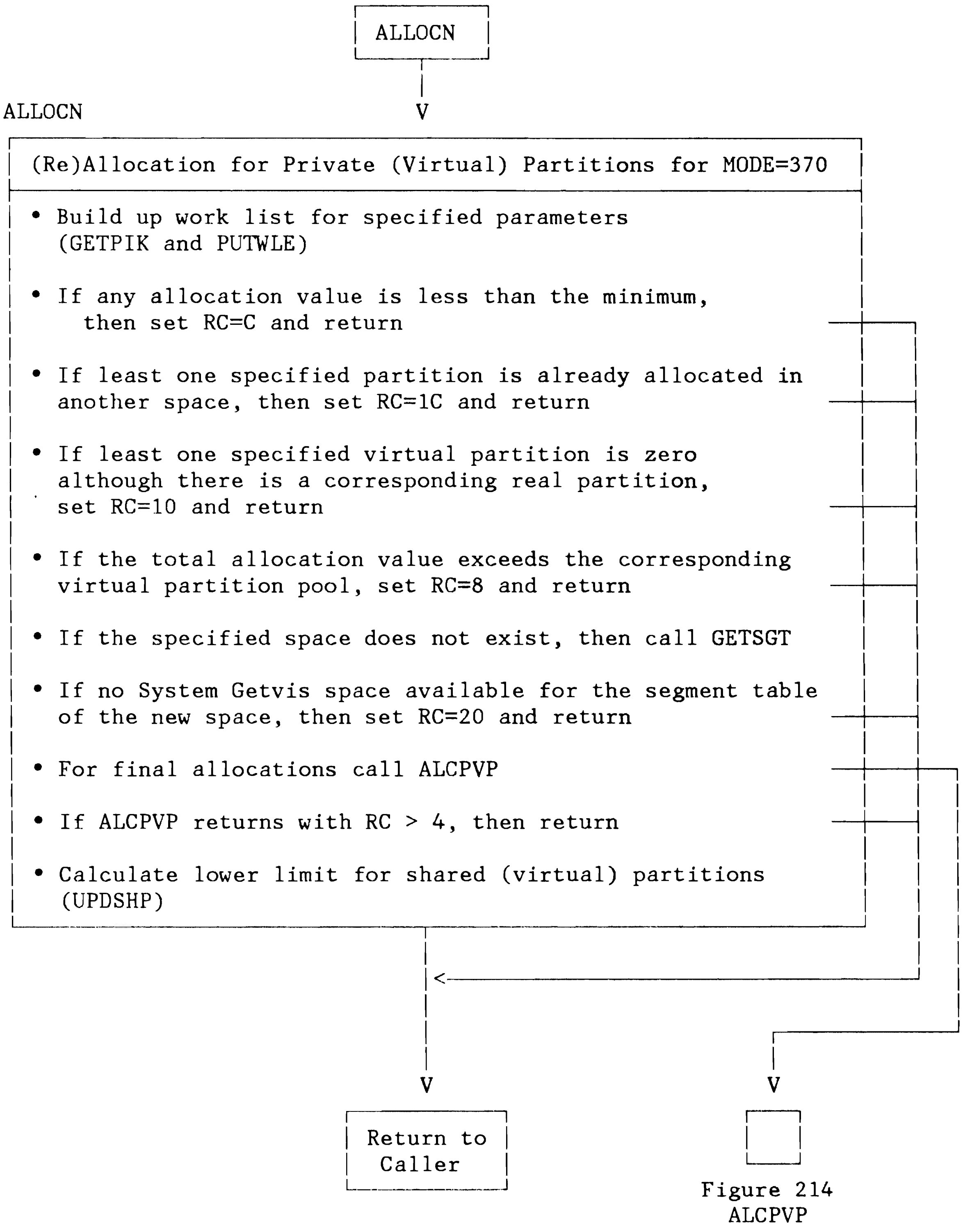
ALLOCS

(Re)Allocation for Shared (Virtual) Partitions for MODE=370

- Build up work list for specified parameters (GETPIK,PUTWLE)
- If allocation value < minimum value, set RC=C and return
- If at least one specified partition is already allocated in another space, then set RC=1C and return
- If at least one specified partition is zero although there is a corresponding real partition, set RC=10 and return
- If the new lower boundary of the shared partitions is less than the upper boundary of any private or real partition, then set RC=8 and return
- If one of the specified partitions is active or stopped and the new virtual allocation would not include the old virtual boundaries, or the lower virtual boundary of the current partition would have to be moved upwards, then set RC=14 and return
- Update segment tables for all spaces
- If the old SIZE value cannot be preserved, then set new SIZE to allocation value minus minimum Getvis area (PCBADR.SMVGVIS) and set RC=4
- Update PCBADR.PCBPSCB, PCBADR.SMVPBEG and PCBADR.SMVPEND
- Update SMCOM.SMALCVSZ
- Update SCBADR.SCBLSADR, SCBADR.SCBSIZE and SCBADR.SCBPSTR
- Invalidate part of the current partition, if necessary (INVCUR)

Return to Caller

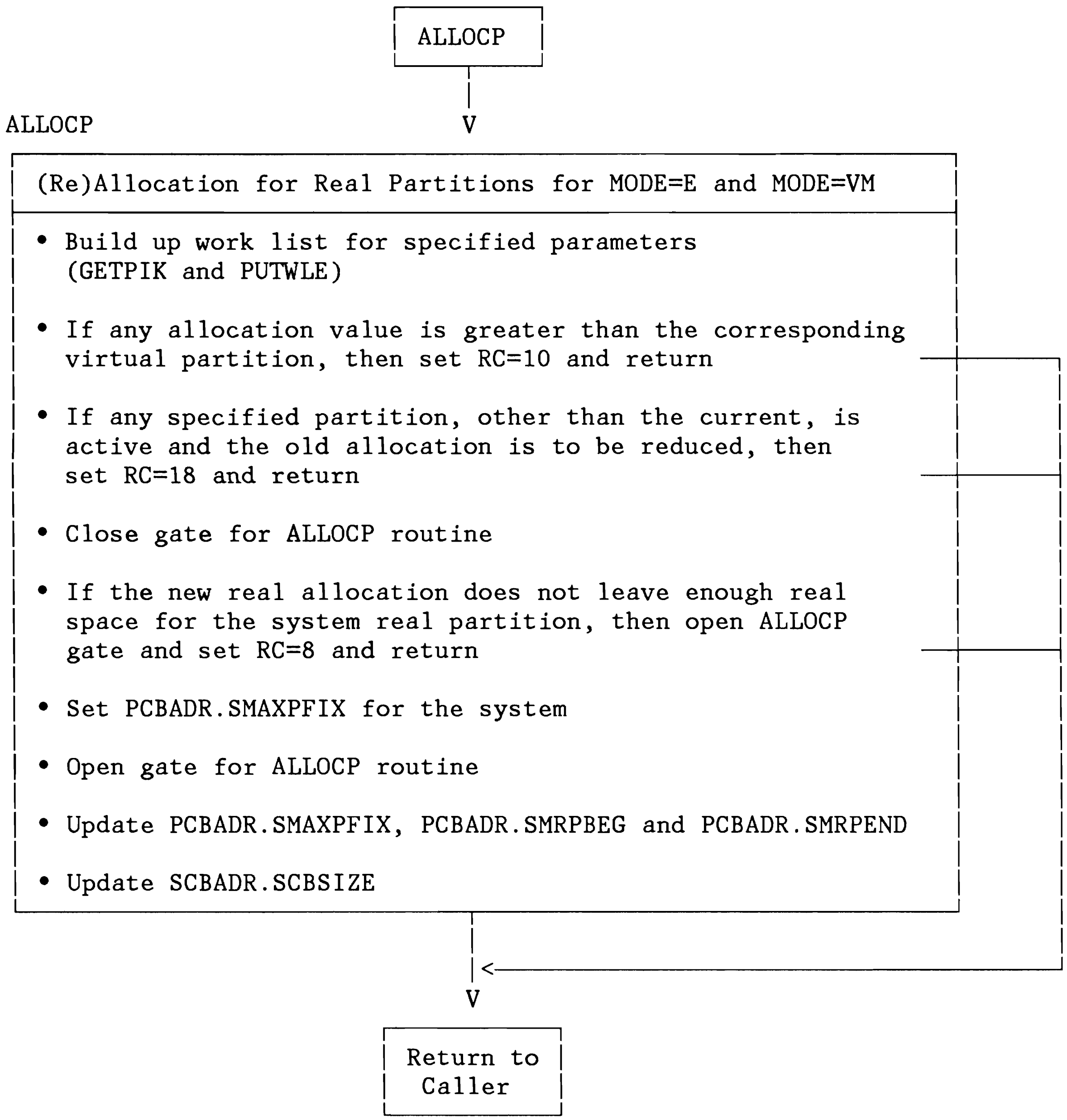
Figure 210. Storage Management(ALLOCS): Allocation Routine for Shared (Virtual) Partitions for MODE=370



(Re)Allocation for Private (Virtual) Partitions for MODE=370

- Build up work list for specified parameters (GETPIK and PUTWLE)
- If any allocation value is less than the minimum, then set RC=C and return
- If least one specified partition is already allocated in another space, then set RC=1C and return
- If least one specified virtual partition is zero although there is a corresponding real partition, set RC=10 and return
- If the total allocation value exceeds the corresponding virtual partition pool, set RC=8 and return
- If the specified space does not exist, then call GETSGT
- If no System Getvis space available for the segment table of the new space, then set RC=20 and return
- For final allocations call ALCPVP
- If ALCPVP returns with RC > 4, then return
- Calculate lower limit for shared (virtual) partitions (UPDSHP)

Figure 211. Storage Management(ALLOCN): Allocation Routine for Private (Virtual) Partitions for MODE=370



ALLOCP

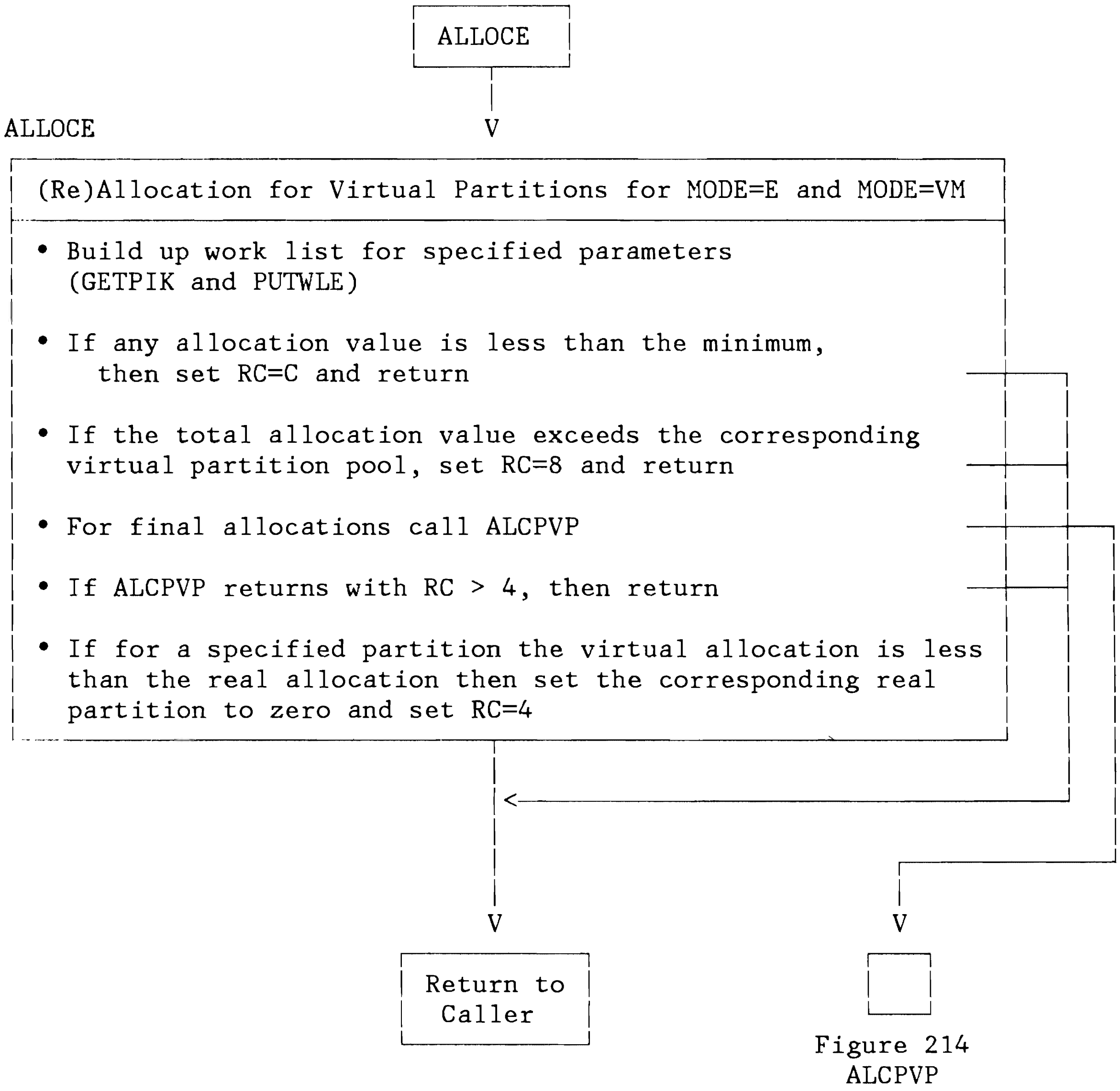
ALLOCP

(Re)Allocation for Real Partitions for MODE=E and MODE=VM

- Build up work list for specified parameters (GETPIK and PUTWLE)
- If any allocation value is greater than the corresponding virtual partition, then set RC=10 and return
- If any specified partition, other than the current, is active and the old allocation is to be reduced, then set RC=18 and return
- Close gate for ALLOCP routine
- If the new real allocation does not leave enough real space for the system real partition, then open ALLOCP gate and set RC=8 and return
- Set PCBADR.SMAXPFIX for the system
- Open gate for ALLOCP routine
- Update PCBADR.SMAXPFIX, PCBADR.SMRPBEG and PCBADR.SMRPEND
- Update SCBADR.SCBSIZE

Return to Caller

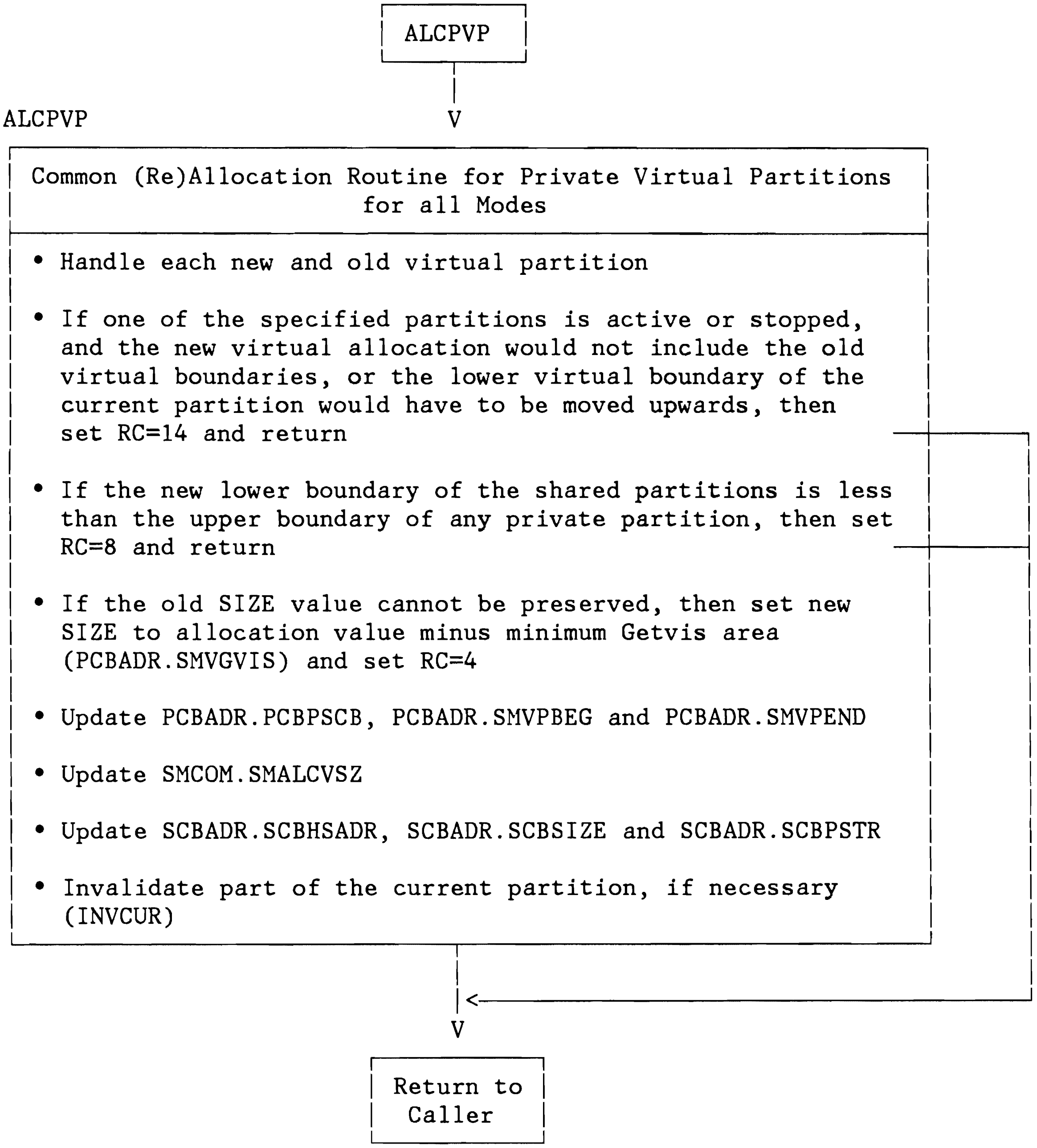
Figure 212. Storage Management(ALLOCP): Allocation Routine for Real Partitions for MODE=E and MODE=VM



(Re)Allocation for Virtual Partitions for MODE=E and MODE=VM

- Build up work list for specified parameters (GETPIK and PUTWLE)
- If any allocation value is less than the minimum, then set RC=C and return
- If the total allocation value exceeds the corresponding virtual partition pool, set RC=8 and return
- For final allocations call ALCPVP
- If ALCPVP returns with RC > 4, then return
- If for a specified partition the virtual allocation is less than the real allocation then set the corresponding real partition to zero and set RC=4

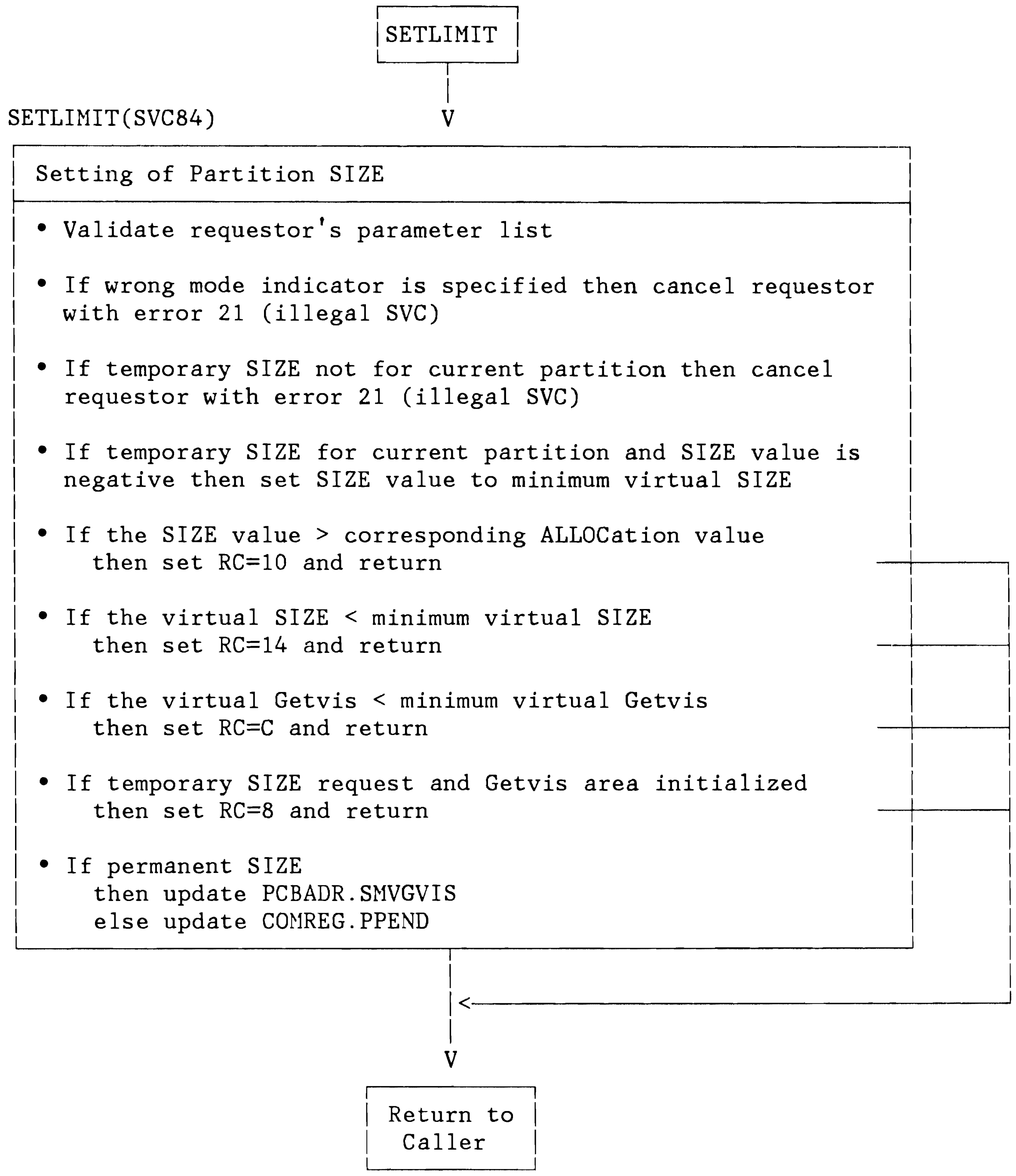
Figure 213. Storage Management(ALLOCE): Allocation Routine for Virtual Partitions for MODE=E and MODE=VM



Common (Re)Allocation Routine for Private Virtual Partitions for all Modes

- Handle each new and old virtual partition
- If one of the specified partitions is active or stopped, and the new virtual allocation would not include the old virtual boundaries, or the lower virtual boundary of the current partition would have to be moved upwards, then set RC=14 and return
- If the new lower boundary of the shared partitions is less than the upper boundary of any private partition, then set RC=8 and return
- If the old SIZE value cannot be preserved, then set new SIZE to allocation value minus minimum Getvis area (PCBADR.SMVGVIS) and set RC=4
- Update PCBADR.PCBPSCB, PCBADR.SMVPBEG and PCBADR.SMVPEND
- Update SMCOM.SMALCVSZ
- Update SCBADR.SCBHSADR, SCBADR.SCBSIZE and SCBADR.SCBPSTR
- Invalidate part of the current partition, if necessary (INVCUR)

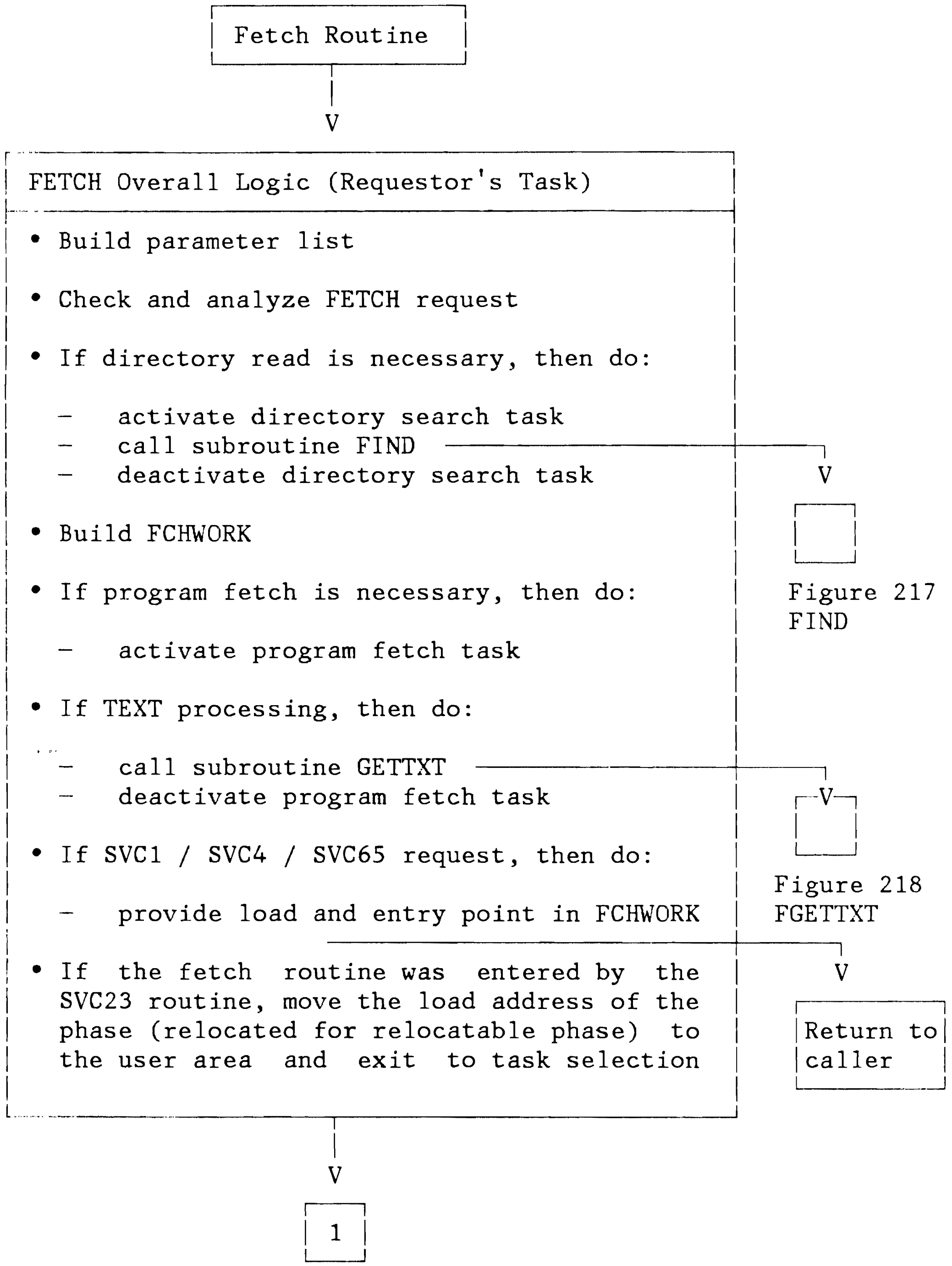
Figure 214. Storage Management(ALCPVP): Common Allocation Routine for Virtual Partitions for All Modes



- Validate requestor's parameter list
- If wrong mode indicator is specified then cancel requestor with error 21 (illegal SVC)
- If temporary SIZE not for current partition then cancel requestor with error 21 (illegal SVC)
- If temporary SIZE for current partition and SIZE value is negative then set SIZE value to minimum virtual SIZE
- If the SIZE value > corresponding ALLOCation value
  then set RC=10 and return
- If the virtual SIZE < minimum virtual SIZE
  then set RC=14 and return
- If the virtual Getvis < minimum virtual Getvis
  then set RC=C and return
- If temporary SIZE request and Getvis area initialized
  then set RC=8 and return
- If permanent SIZE
  then update PCBADR.SMVGVIS
  else update COMREG.PPEND

Figure 215. Storage Management: Setting of Partition Size

FETCH ROUTINES



FETCH Overall Logic (Requestor's Task)

- Build parameter list
- Check and analyze FETCH request
- If directory read is necessary, then do:
  - activate directory search task
  - call subroutine FIND
  - deactivate directory search task
- Build FCHWORK
- If program fetch is necessary, then do:
  - activate program fetch task
- If TEXT processing, then do:
  - call subroutine GETTXT
  - deactivate program fetch task
- If SVC1 / SVC4 / SVC65 request, then do:
  - provide load and entry point in FCHWORK
- If the fetch routine was entered by the SVC23 routine, move the load address of the phase (relocated for relocatable phase) to the user area and exit to task selection

Figure 216 (Part 1 of 2). Fetch Routine

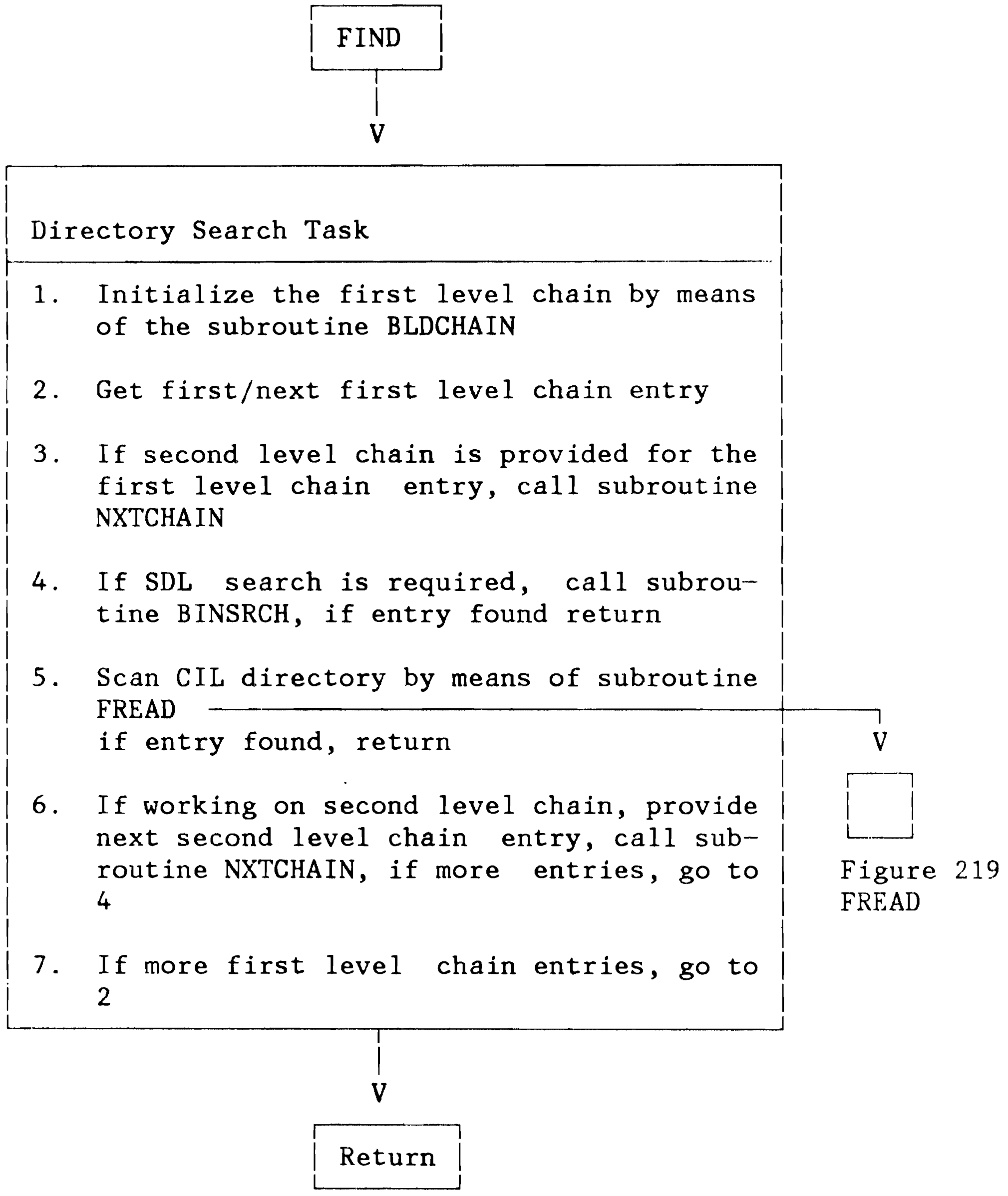
1

- If the fetch routine was entered by the SVC51 routine, move the requested number of halfwords of the directory entry of the phase to the area and exit to task selection.

- If the phase to be fetched is not found, then do:

1. For SVC23 and SVC51 return to task selection
2. If an in-storage directory is found and TXT=NO specified
   - post directory entry active and phase not found
   - return to task selection
3. In the other cases:
   - If $$A phase – Cancel due to I/O error
   - If CRT phase – Hardwait x'FF8'
   - Otherwise:
     - If return code requested, then return to requestor with return code in register 15
     - If no return code requested then cancel with cancel code x'22'

Figure 216 (Part 2 of 2). Fetch Routine



FIND

Directory Search Task

1. Initialize the first level chain by means of the subroutine BLDCHAIN

2. Get first/next first level chain entry

3. If second level chain is provided for the first level chain entry, call subroutine NXTCHAIN

4. If SDL search is required, call subroutine BINSRCH, if entry found return

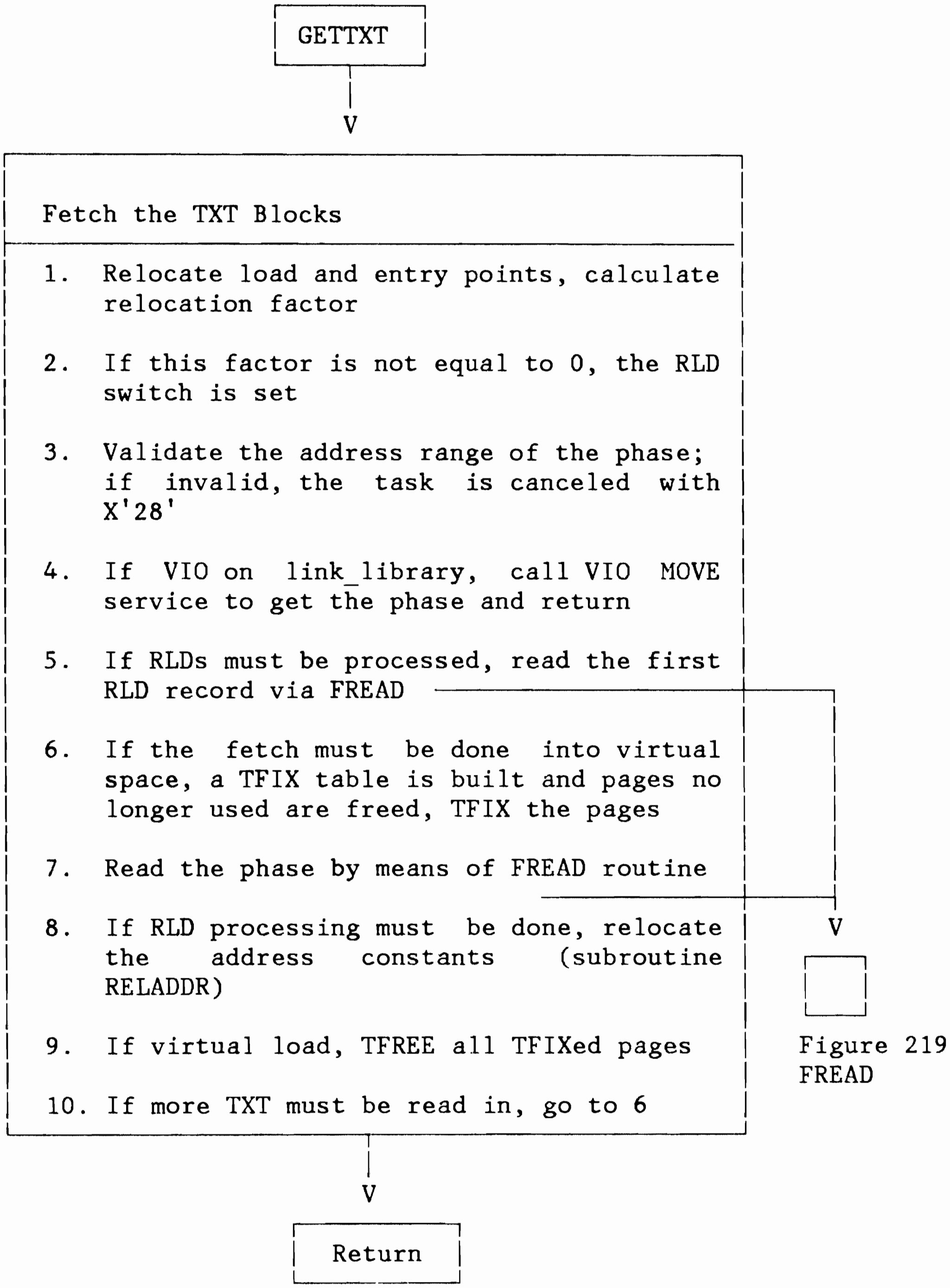
5. Scan CIL directory by means of subroutine FREAD
   if entry found, return

6. If working on second level chain, provide next second level chain entry, call subroutine NXTCHAIN, if more entries, go to 4

7. If more first level chain entries, go to 2

Return

Figure 217. Fetch Routine: Directory Search Task



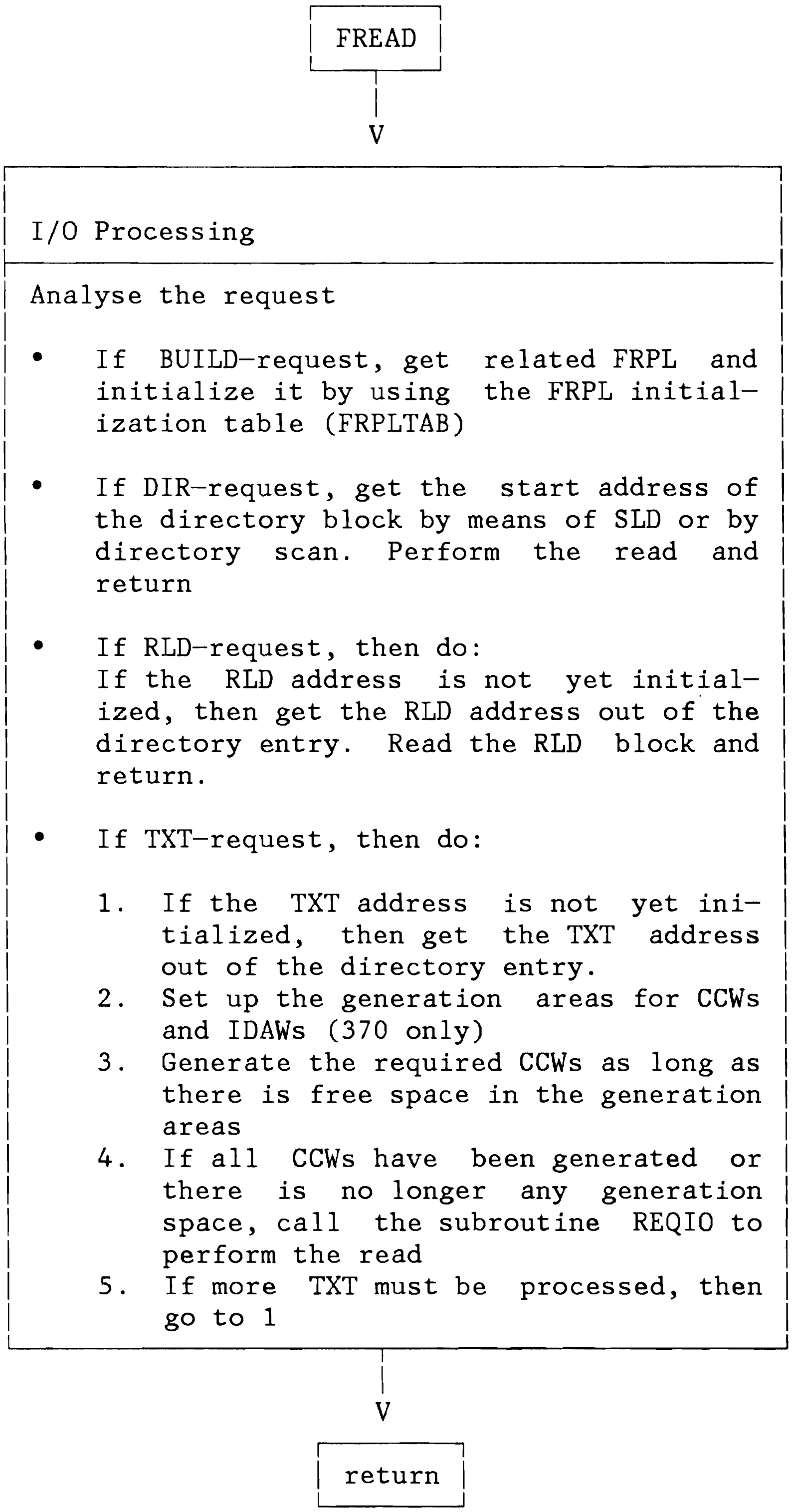
GETTXT

Fetch the TXT Blocks

1. Relocate load and entry points, calculate relocation factor
2. If this factor is not equal to 0, the RLD switch is set
3. Validate the address range of the phase; if invalid, the task is canceled with X'28'
4. If VIO on link_library, call VIO MOVE service to get the phase and return
5. If RLDs must be processed, read the first RLD record via FREAD
6. If the fetch must be done into virtual space, a TFIX table is built and pages no longer used are freed, TFIX the pages
7. Read the phase by means of FREAD routine
8. If RLD processing must be done, relocate the address constants (subroutine RELADDR)
9. If virtual load, TFREE all TFIXed pages
10. If more TXT must be read in, go to 6

Return

Figure 218. Fetch Routine: GETTXT Routine



FREAD

I/O Processing

Analyse the request

- If BUILD-request, get related FRPL and initialize it by using the FRPL initialization table (FRPLTAB)

- If DIR-request, get the start address of the directory block by means of SLD or by directory scan. Perform the read and return

- If RLD-request, then do:
  If the RLD address is not yet initialized, then get the RLD address out of the directory entry. Read the RLD block and return.

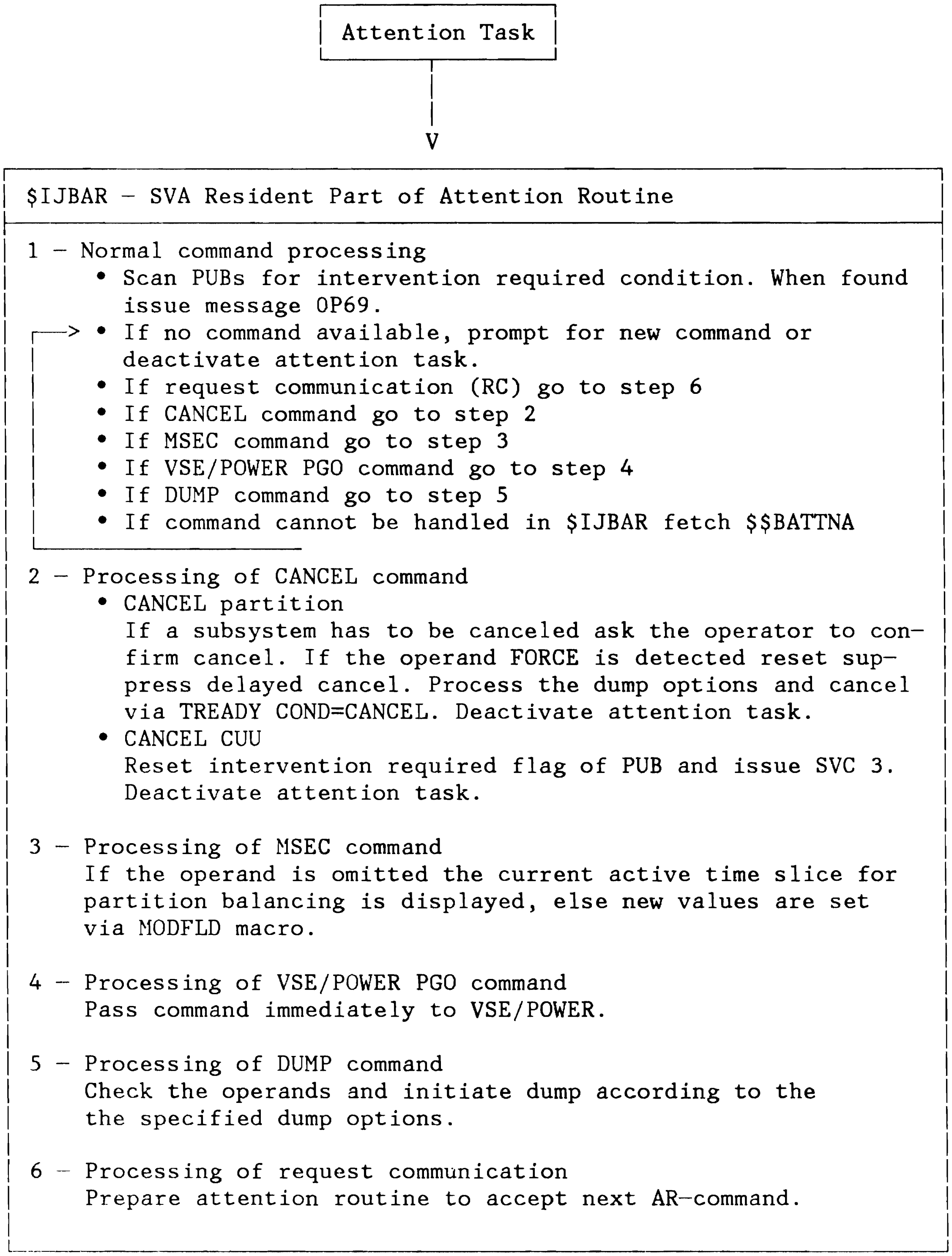
- If TXT-request, then do:

  1. If the TXT address is not yet initialized, then get the TXT address out of the directory entry.
  2. Set up the generation areas for CCWs and IDAWs (370 only)
  3. Generate the required CCWs as long as there is free space in the generation areas
  4. If all CCWs have been generated or there is no longer any generation space, call the subroutine REQIO to perform the read
  5. If more TXT must be processed, then go to 1

return

Figure 219. Fetch Routine: I/O Processing

ATTENTION MAIN ROUTINE



Attention Task

$IJBAR - SVA Resident Part of Attention Routine

1 - Normal command processing
- Scan PUBs for intervention required condition. When found issue message 0P69.
- If no command available, prompt for new command or deactivate attention task.
- If request communication (RC) go to step 6
- If CANCEL command go to step 2
- If MSEC command go to step 3
- If VSE/POWER PGO command go to step 4
- If DUMP command go to step 5
- If command cannot be handled in $IJBAR fetch $$BATTNA

2 - Processing of CANCEL command
- CANCEL partition
  If a subsystem has to be canceled ask the operator to confirm cancel. If the operand FORCE is detected reset suppress delayed cancel. Process the dump options and cancel via TREADY COND=CANCEL. Deactivate attention task.
- CANCEL CUU
  Reset intervention required flag of PUB and issue SVC 3. Deactivate attention task.

3 - Processing of MSEC command
If the operand is omitted the current active time slice for partition balancing is displayed, else new values are set via MODFLD macro.

4 - Processing of VSE/POWER PGO command
Pass command immediately to VSE/POWER.

5 - Processing of DUMP command
Check the operands and initiate dump according to the the specified dump options.

6 - Processing of request communication
Prepare attention routine to accept next AR-command.

Figure 220. Attention Main Routine

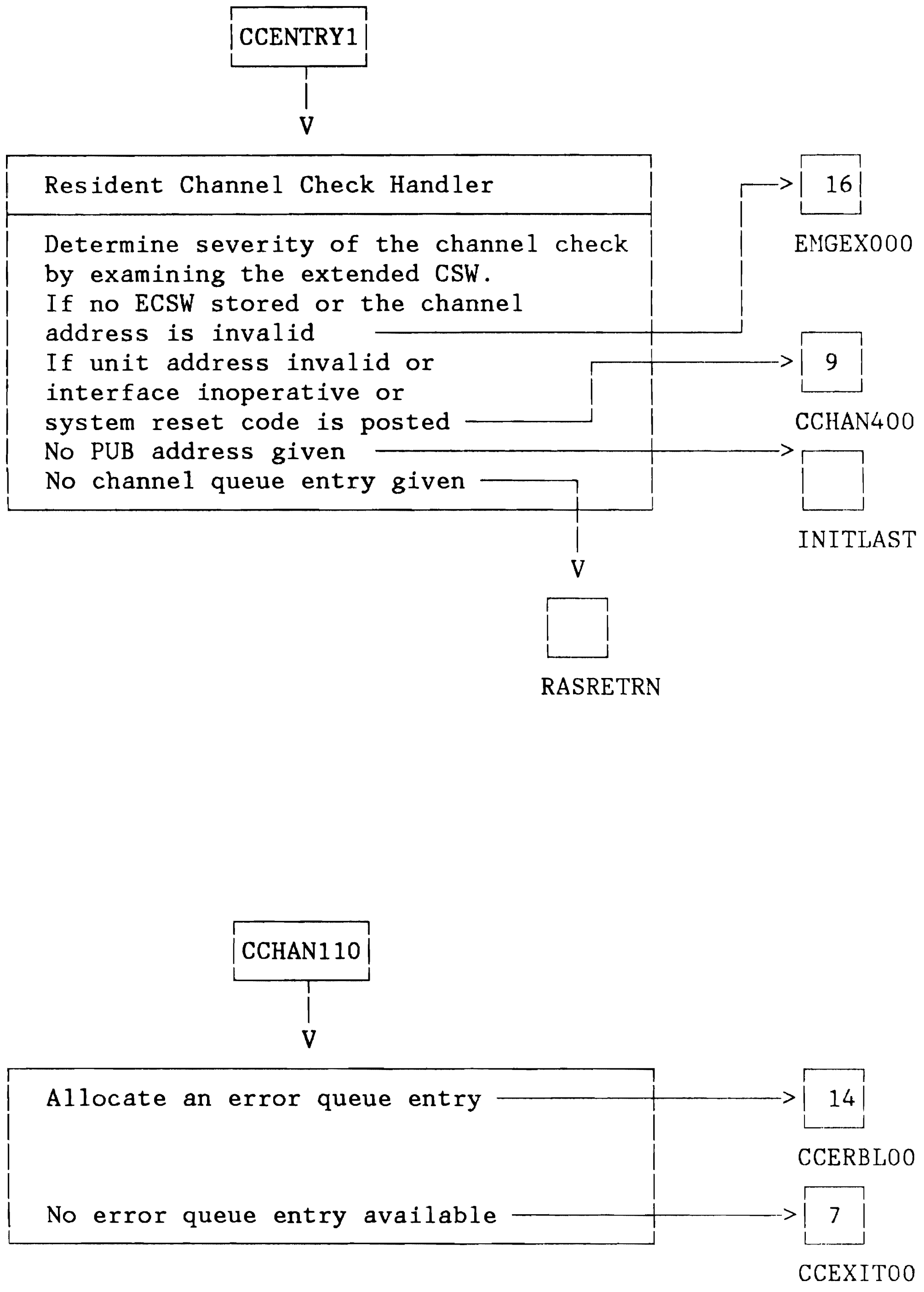

CCENTRY1
Resident Channel Check Handler
Determine severity of the channel check by examining the extended CSW.
If no ECSW stored or the channel address is invalid
16
EMGEX000
If unit address invalid or interface inoperative or system reset code is posted
9
CCHAN400
No PUB address given
INITLAST
No channel queue entry given
RASRETRN
CCHAN110
Allocate an error queue entry
14
CCERBL00
No error queue entry available
7
CCEXIT00

Figure 222 (Part 1 of 5). MCH/CCH: Channel Check Handler

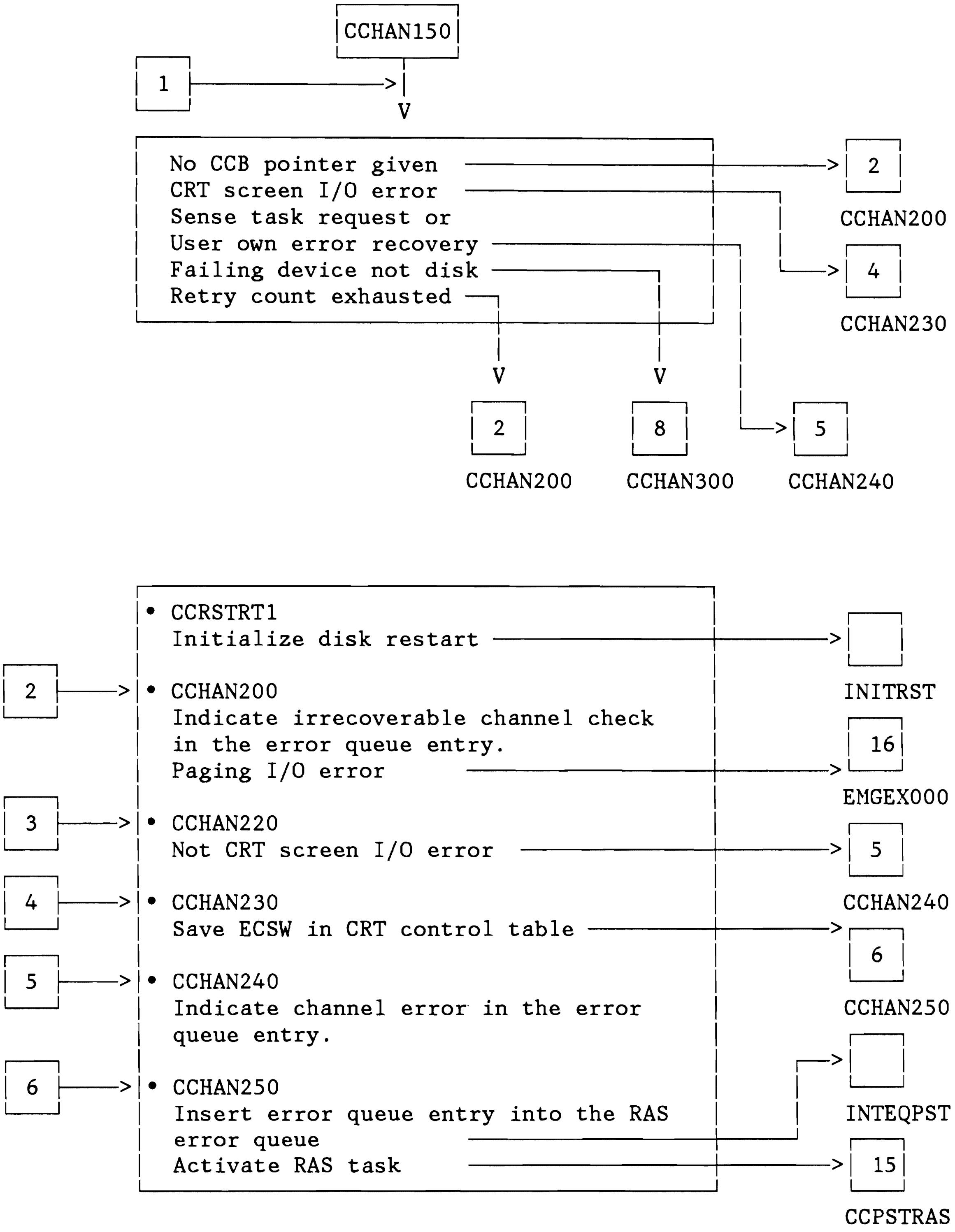
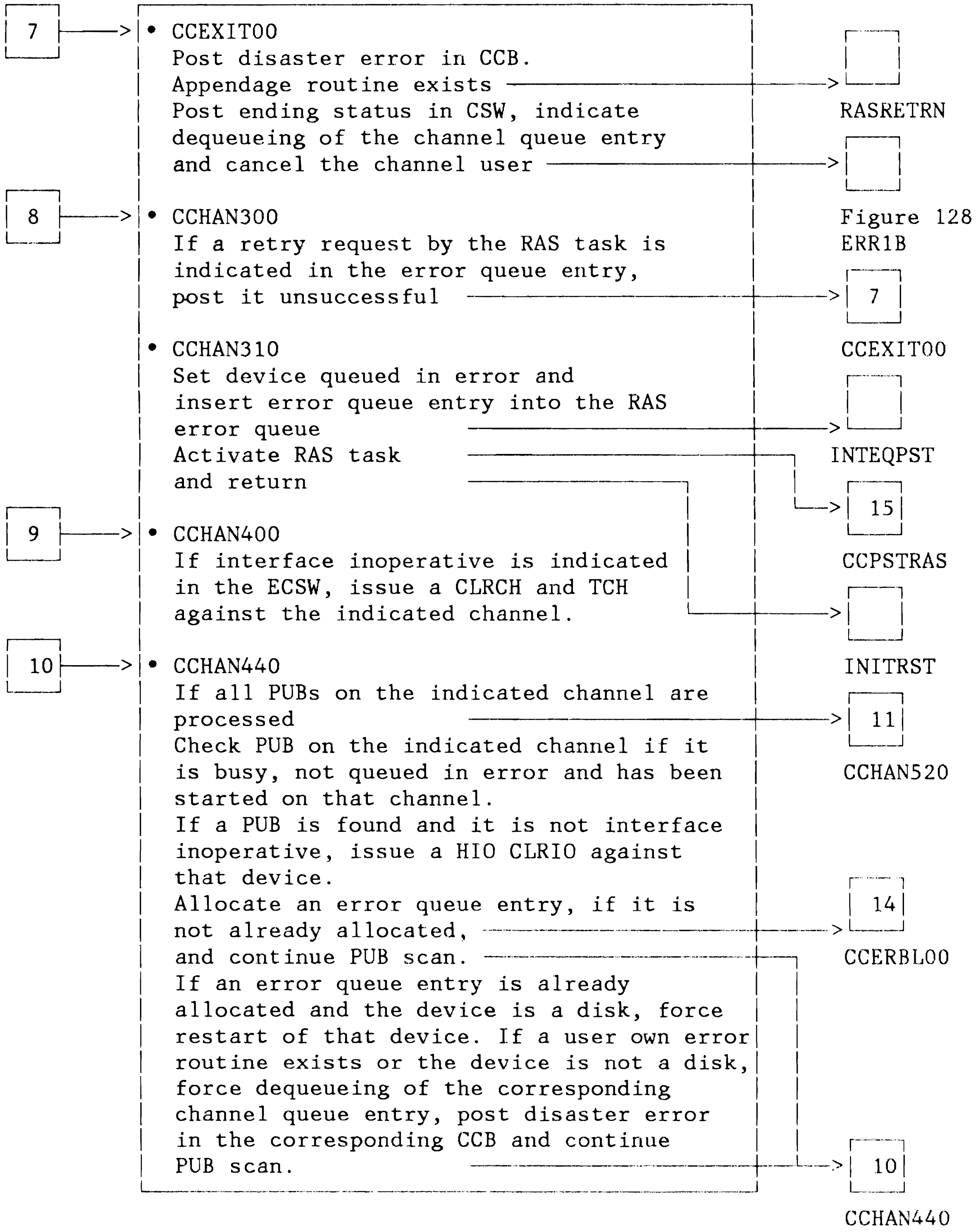


Figure 222 (Part 2 of 5). MCH/CCH: Channel Check Handler



7 → • CCEXIT00
Post disaster error in CCB.
Appendage routine exists → RASRETRN
Post ending status in CSW, indicate dequeueing of the channel queue entry and cancel the channel user → Figure 128 ERR1B

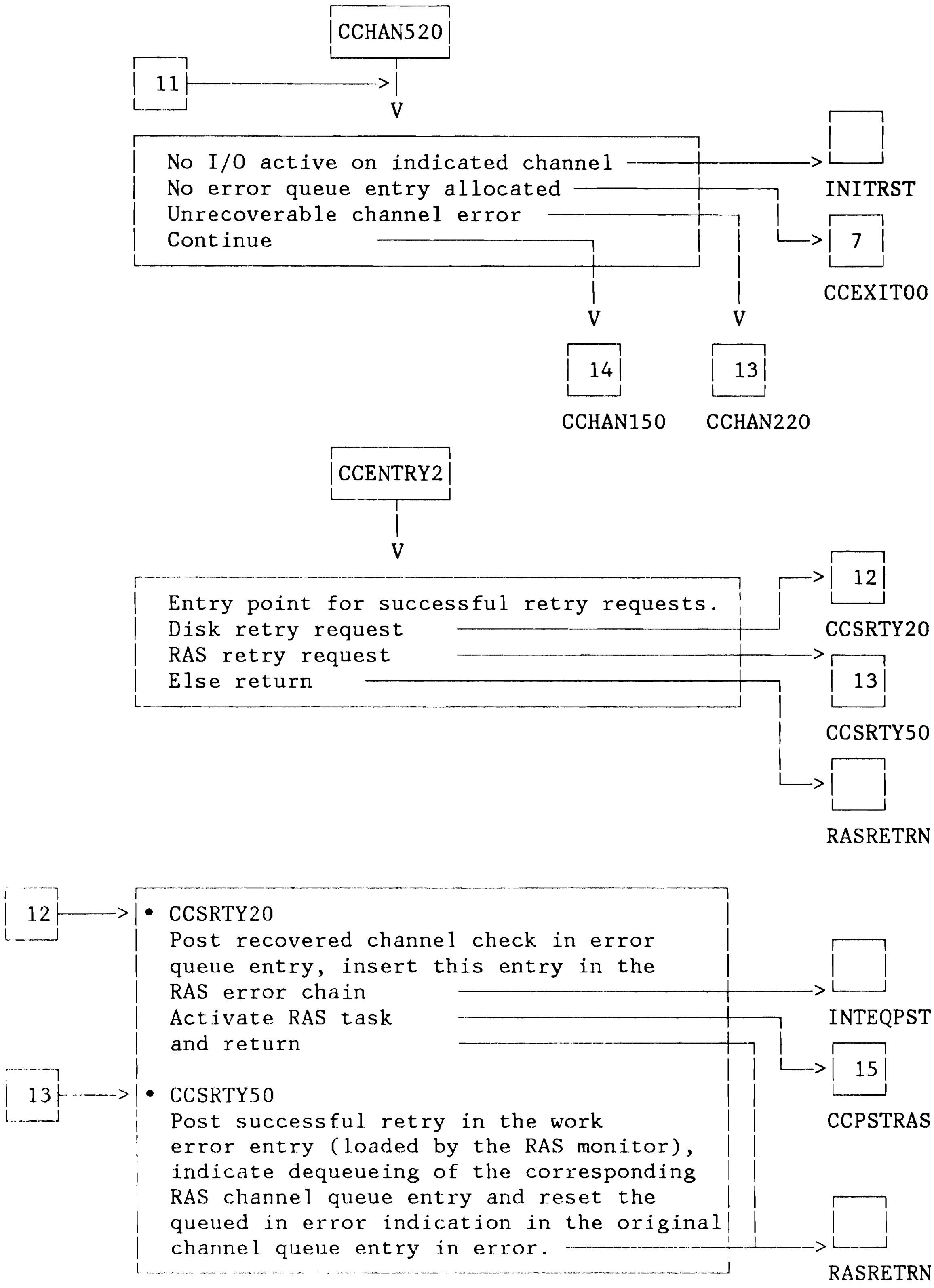
8 → • CCHAN300
If a retry request by the RAS task is indicated in the error queue entry, post it unsuccessful → 7 CCEXIT00

• CCHAN310
Set device queued in error and insert error queue entry into the RAS error queue → INTEQPST
Activate RAS task → 15 CCPSTRAS
and return → INITRST

9 → • CCHAN400
If interface inoperative is indicated in the ECSW, issue a CLRCH and TCH against the indicated channel.

10 → • CCHAN440
If all PUBs on the indicated channel are processed → 11 CCHAN520
Check PUB on the indicated channel if it is busy, not queued in error and has been started on that channel.
If a PUB is found and it is not interface inoperative, issue a HIO CLRIO against that device.
Allocate an error queue entry, if it is not already allocated, → 14 CCERBL00
and continue PUB scan. → 10 CCHAN440
If an error queue entry is already allocated and the device is a disk, force restart of that device. If a user own error routine exists or the device is not a disk, force dequeueing of the corresponding channel queue entry, post disaster error in the corresponding CCB and continue PUB scan. → 10 CCHAN440

Figure 222 (Part 3 of 5). MCH/CCH: Channel Check Handler



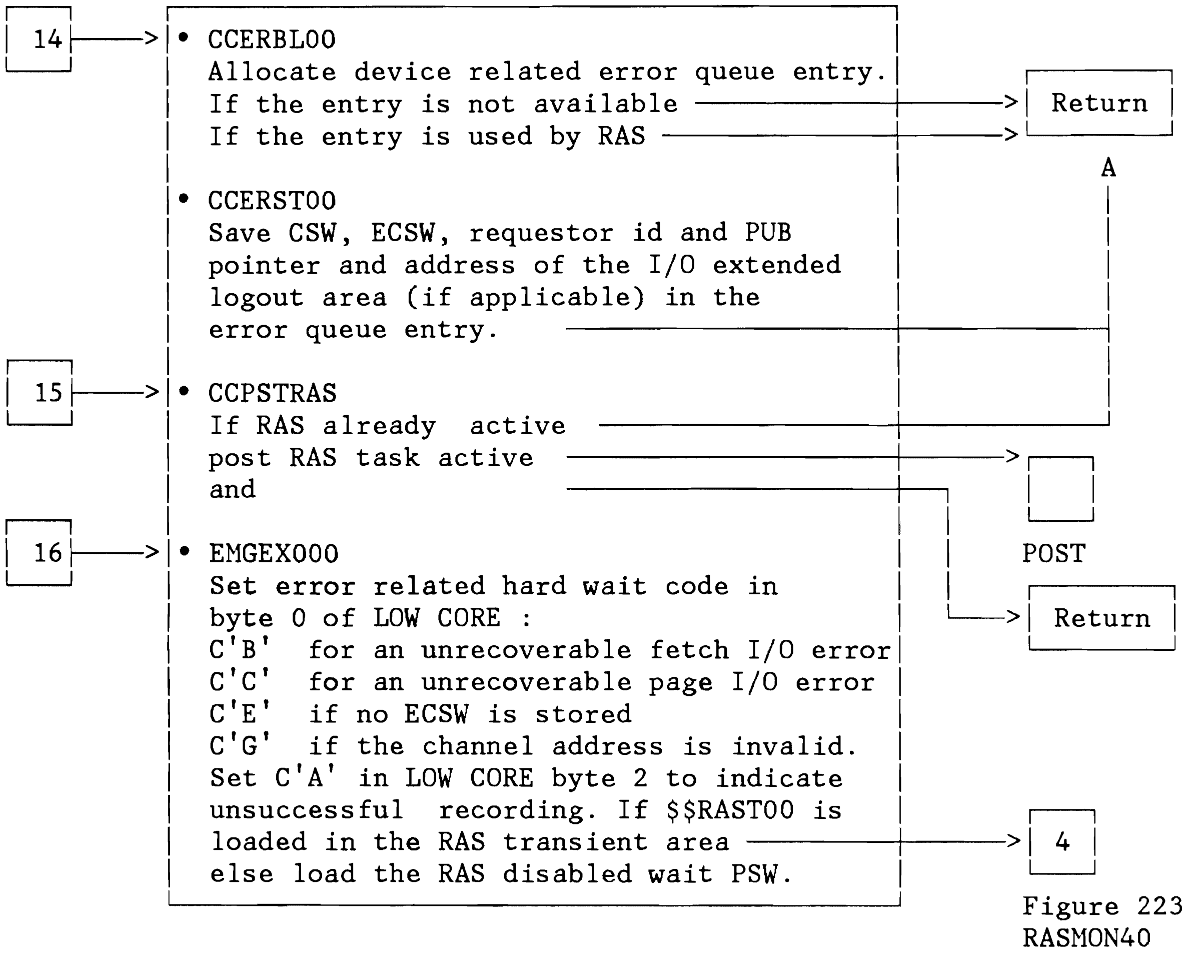
No I/O active on indicated channel
No error queue entry allocated
Unrecoverable channel error
Continue

Entry point for successful retry requests.
Disk retry request
RAS retry request
Else return

- CCSRTY20
  Post recovered channel check in error queue entry, insert this entry in the RAS error chain
  Activate RAS task
  and return

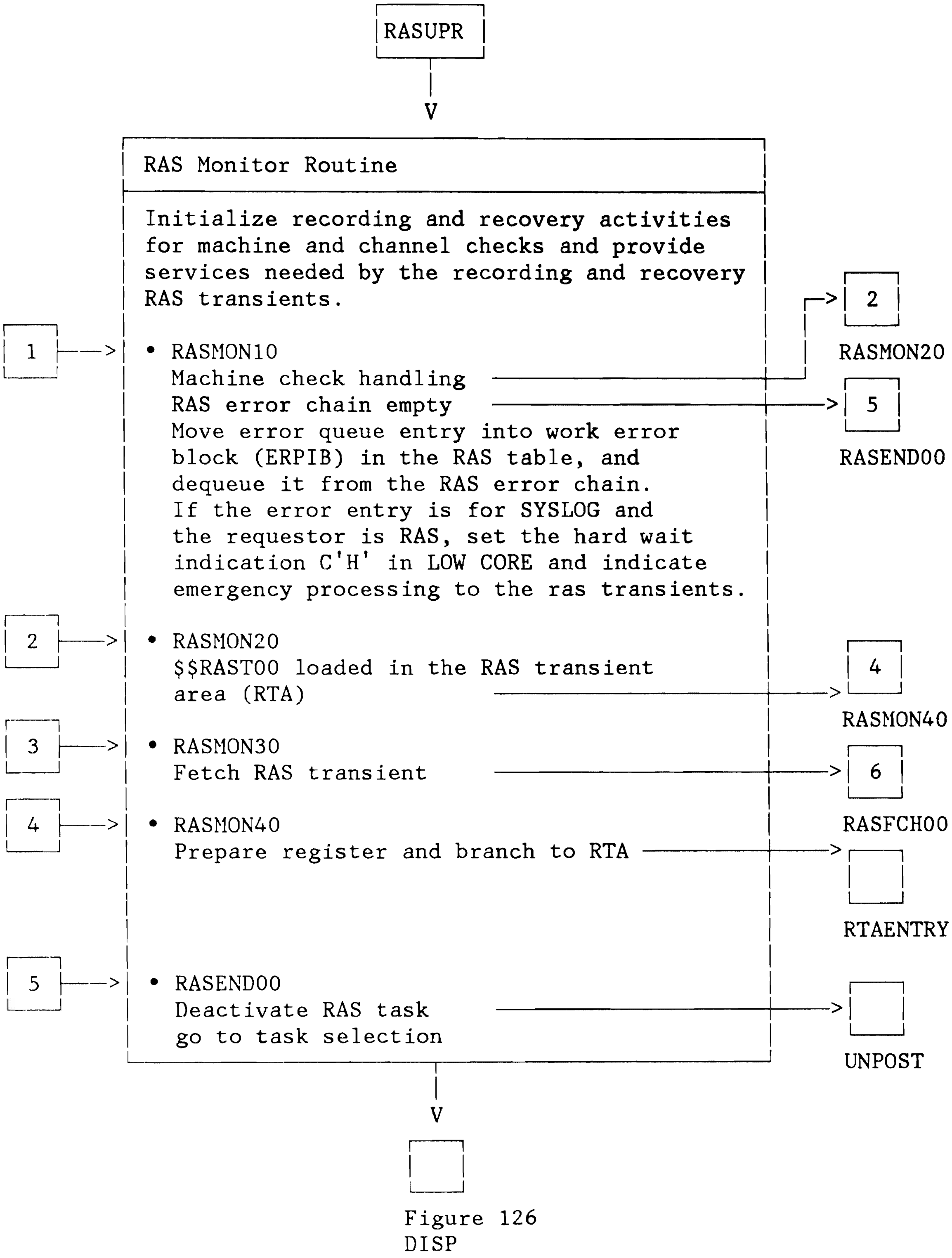
- CCSRTY50
  Post successful retry in the work error entry (loaded by the RAS monitor), indicate dequeueing of the corresponding RAS channel queue entry and reset the queued in error indication in the original channel queue entry in error.

Figure 222 (Part 4 of 5). MCH/CCH: Channel Check Handler



- CCERBL00
  Allocate device related error queue entry.
  If the entry is not available
  If the entry is used by RAS
- CCERST00
  Save CSW, ECSW, requestor id and PUB pointer and address of the I/O extended logout area (if applicable) in the error queue entry.
- CCPSTRAS
  If RAS already active
  post RAS task active
  and
- EMGEX000
  Set error related hard wait code in byte 0 of LOW CORE :
  C'B' for an unrecoverable fetch I/O error
  C'C' for an unrecoverable page I/O error
  C'E' if no ECSW is stored
  C'G' if the channel address is invalid.
  Set C'A' in LOW CORE byte 2 to indicate unsuccessful recording. If $$RAST00 is loaded in the RAS transient area
  else load the RAS disabled wait PSW.

Figure 222 (Part 5 of 5). MCH/CCH: Channel Check Handler



RAS Monitor Routine

Initialize recording and recovery activities for machine and channel checks and provide services needed by the recording and recovery RAS transients.

- RASMON10
  Machine check handling
  RAS error chain empty
  Move error queue entry into work error block (ERPIB) in the RAS table, and dequeue it from the RAS error chain.
  If the error entry is for SYSLOG and the requestor is RAS, set the hard wait indication C'H' in LOW CORE and indicate emergency processing to the ras transients.
- RASMON20
  $$RAST00 loaded in the RAS transient area (RTA)
- RASMON30
  Fetch RAS transient
- RASMON40
  Prepare register and branch to RTA
- RASEND00
  Deactivate RAS task
  go to task selection

Figure 223 (Part 1 of 4). MCH/CCH: RAS Monitor

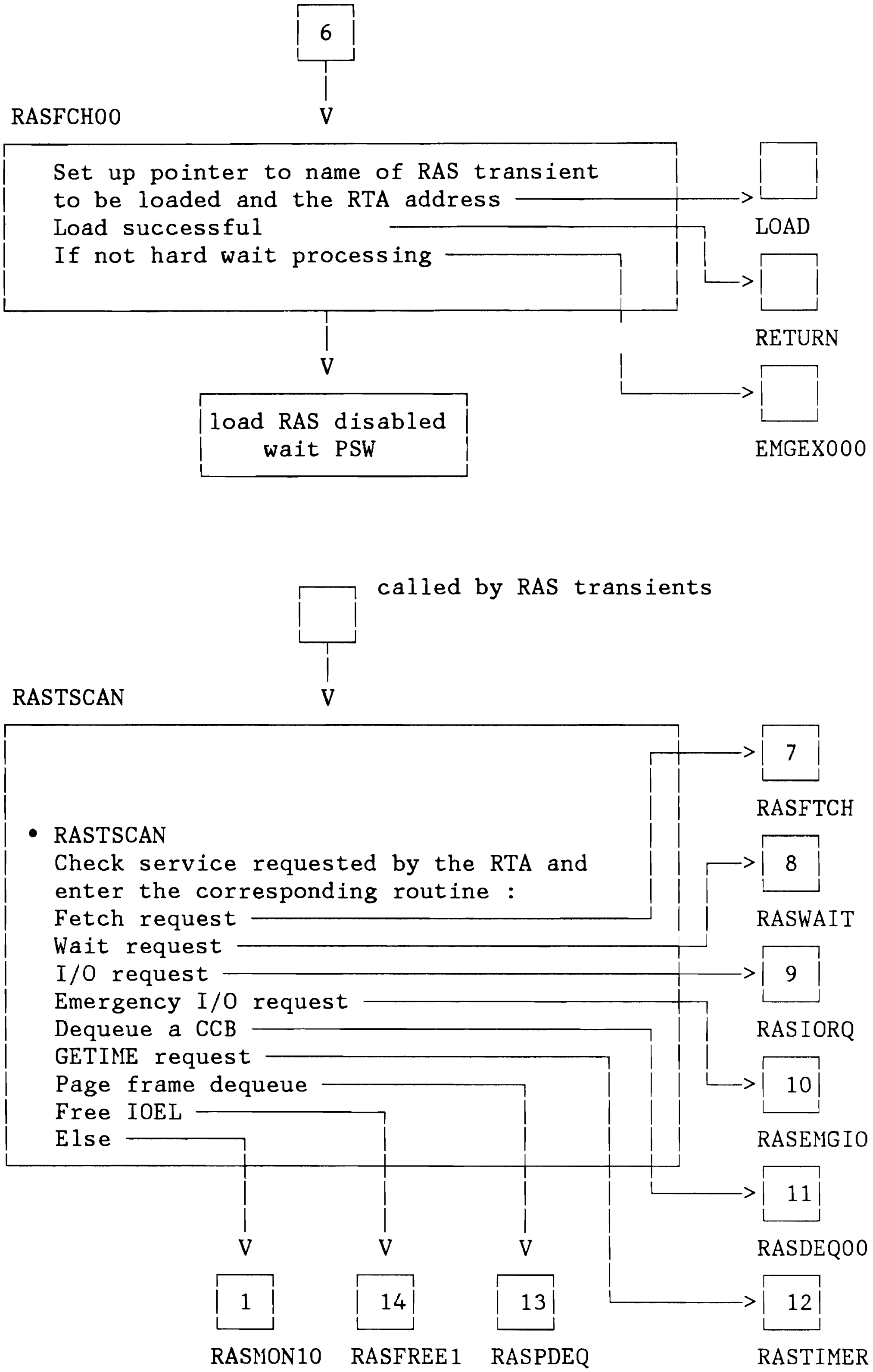
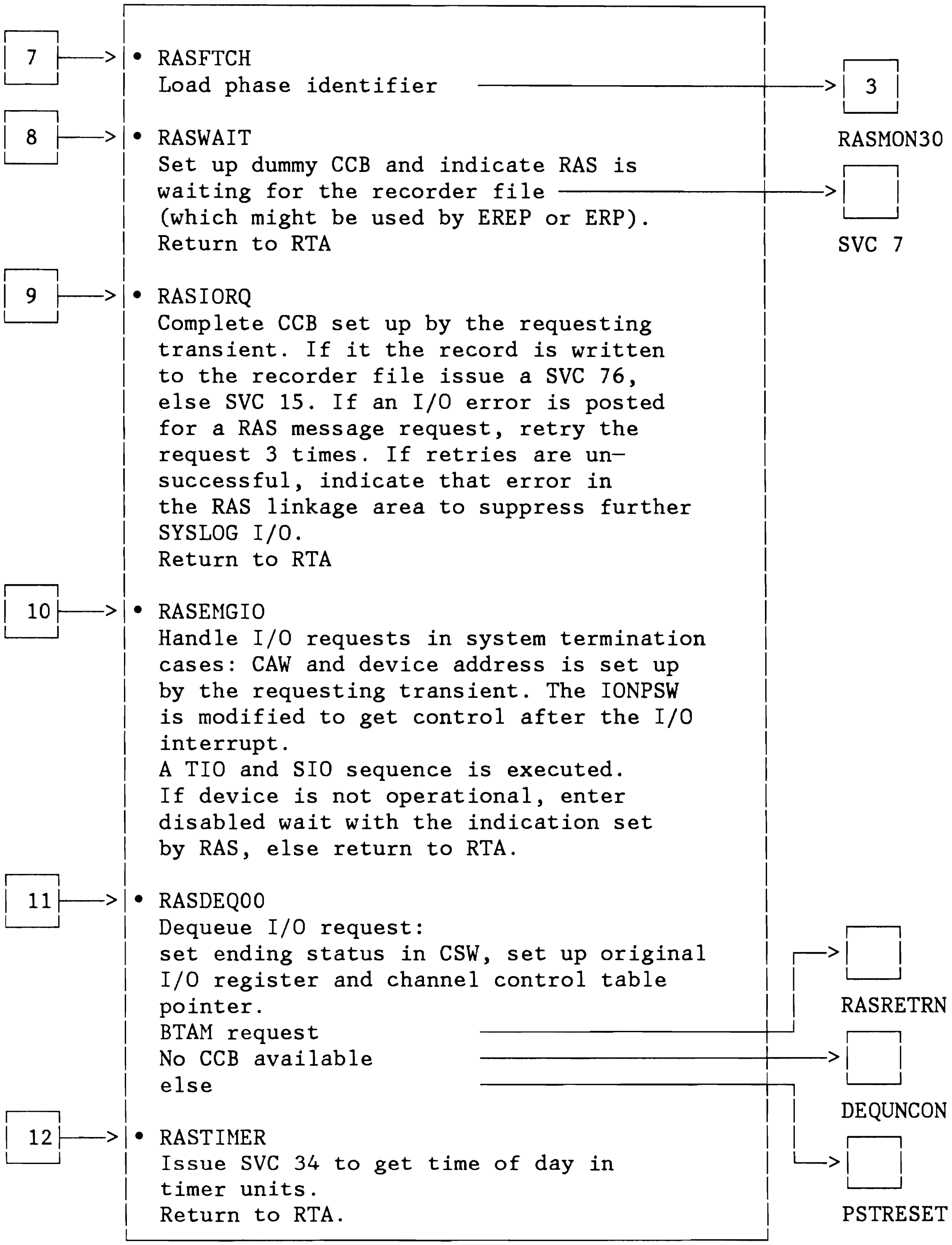


Figure 223 (Part 2 of 4). MCH/CCH: RAS Monitor



- RASFTCH
  Load phase identifier

- RASWAIT
  Set up dummy CCB and indicate RAS is waiting for the recorder file (which might be used by EREP or ERP).
  Return to RTA

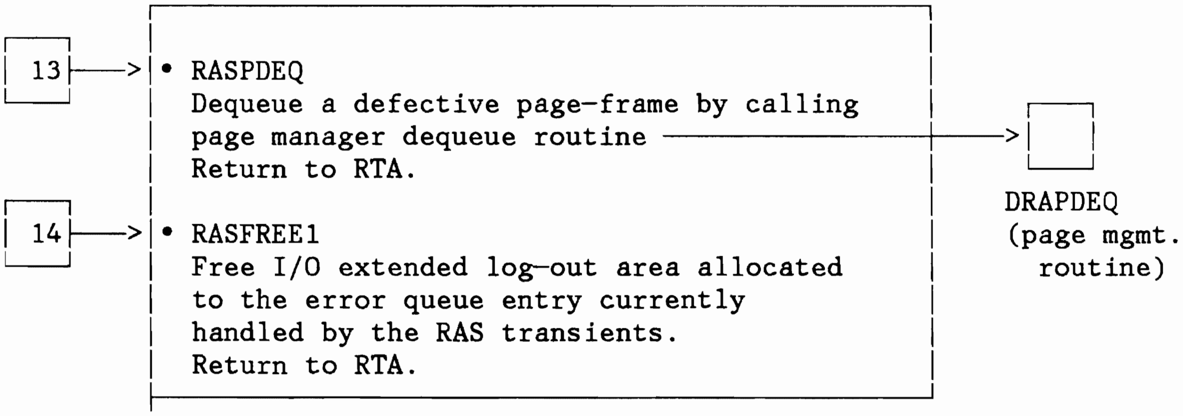
- RASIORQ
  Complete CCB set up by the requesting transient. If it the record is written to the recorder file issue a SVC 76, else SVC 15. If an I/O error is posted for a RAS message request, retry the request 3 times. If retries are unsuccessful, indicate that error in the RAS linkage area to suppress further SYSLOG I/O.
  Return to RTA

- RASEMGIO
  Handle I/O requests in system termination cases: CAW and device address is set up by the requesting transient. The IONPSW is modified to get control after the I/O interrupt.
  A TIO and SIO sequence is executed.
  If device is not operational, enter disabled wait with the indication set by RAS, else return to RTA.

- RASDEQ00
  Dequeue I/O request:
  set ending status in CSW, set up original I/O register and channel control table pointer.
  BTAM request
  No CCB available
  else

- RASTIMER
  Issue SVC 34 to get time of day in timer units.
  Return to RTA.

Figure 223 (Part 3 of 4). MCH/CCH: RAS Monitor



- RASPDEQ
  Dequeue a defective page-frame by calling page manager dequeue routine
  Return to RTA.

- RASFREE1
  Free I/O extended log-out area allocated to the error queue entry currently handled by the RAS transients.
  Return to RTA.

Figure 223 (Part 4 of 4). MCH/CCH: RAS Monitor

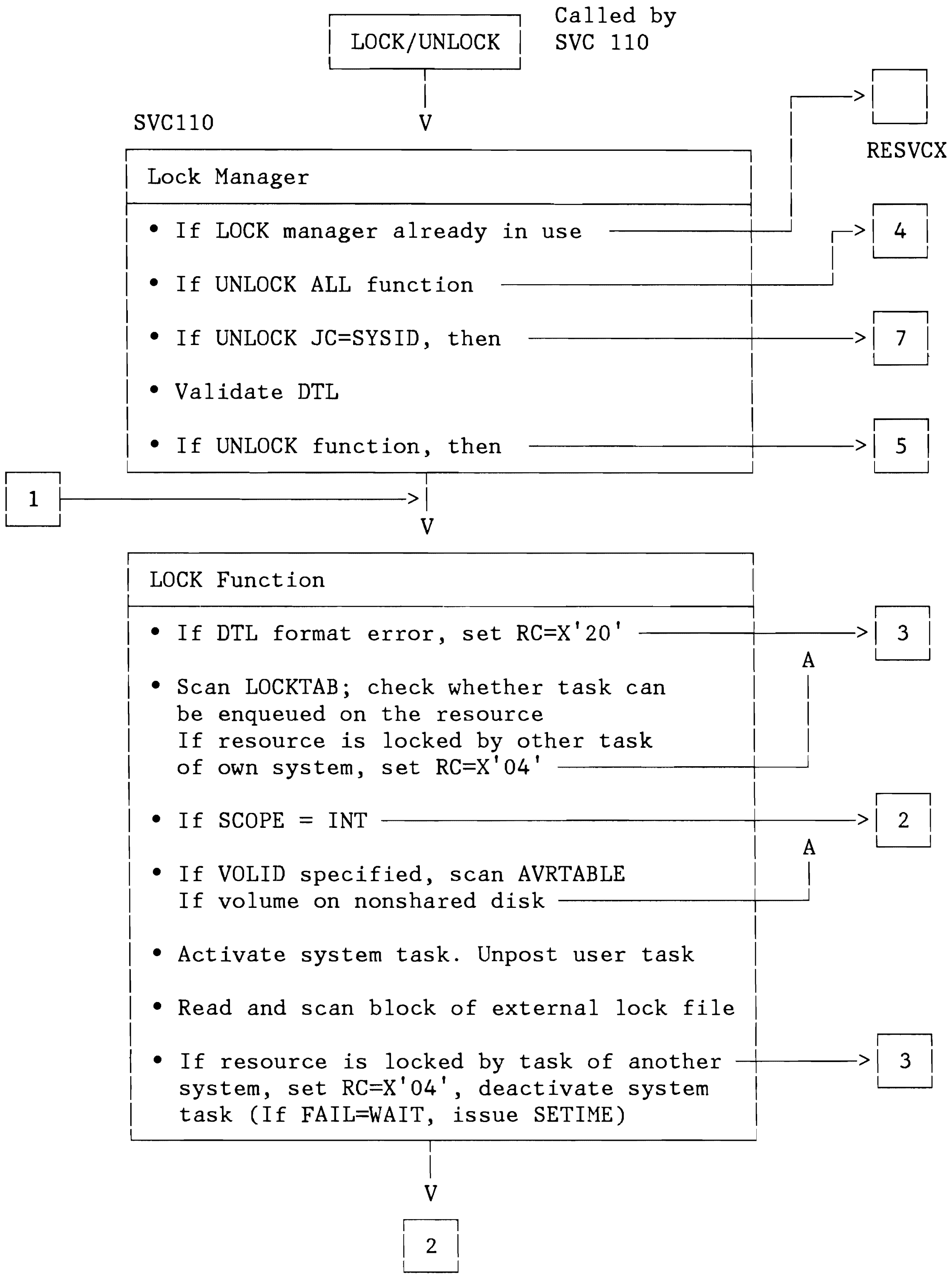
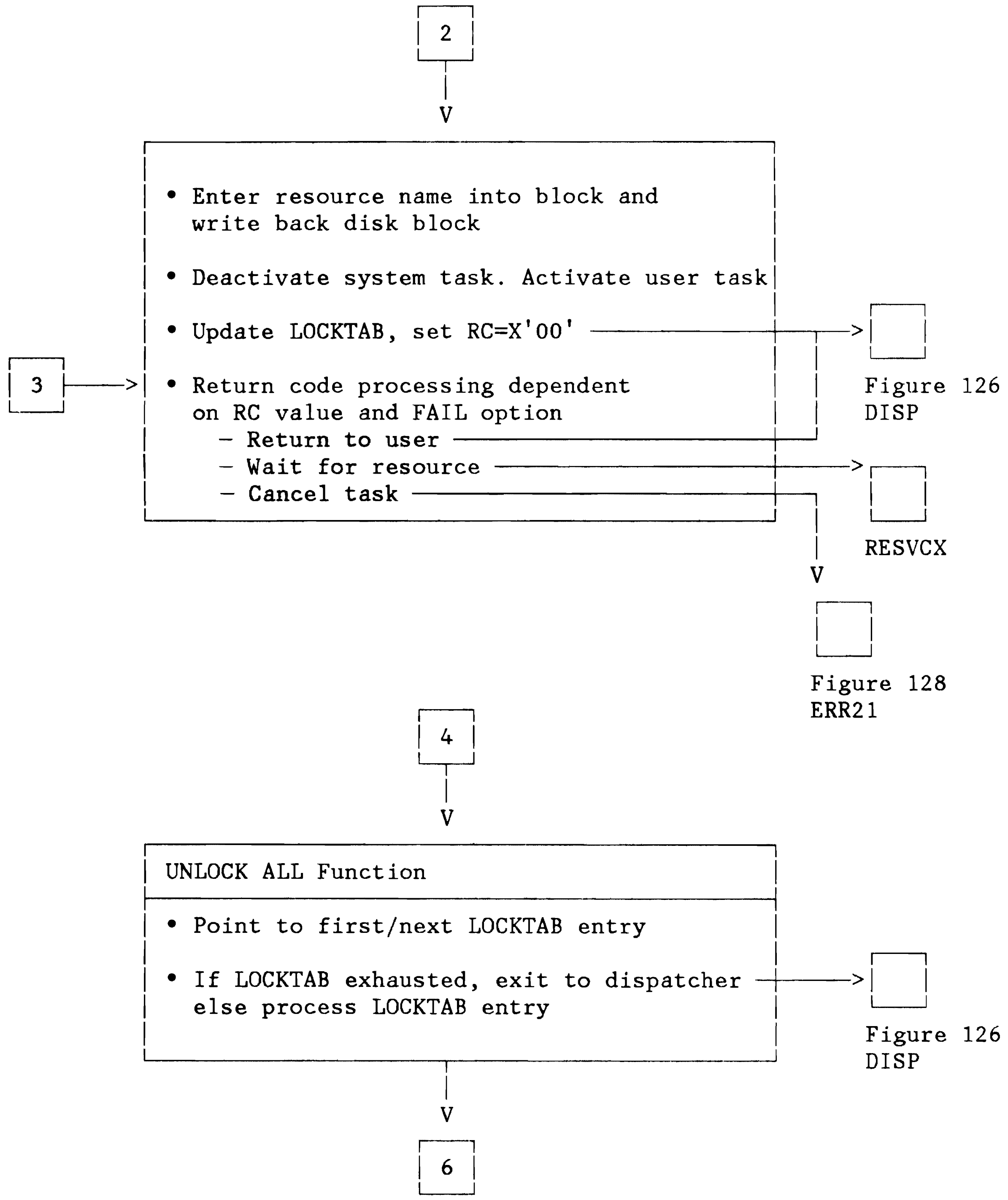

LOCK/UNLOCK
Called by
SVC 110
SVC110
RESVCX
Lock Manager
• If LOCK manager already in use
• If UNLOCK ALL function
4
• If UNLOCK JC=SYSID, then
7
• Validate DTL
• If UNLOCK function, then
5
1
LOCK Function
• If DTL format error, set RC=X'20'
3
A
• Scan LOCKTAB; check whether task can be enqueued on the resource
If resource is locked by other task of own system, set RC=X'04'
• If SCOPE = INT
2
A
• If VOLID specified, scan AVRTABLE
If volume on nonshared disk
• Activate system task. Unpost user task
• Read and scan block of external lock file
• If resource is locked by task of another system, set RC=X'04', deactivate system task (If FAIL=WAIT, issue SETIME)
3
2

Figure 224 (Part 1 of 5). Lock Manager

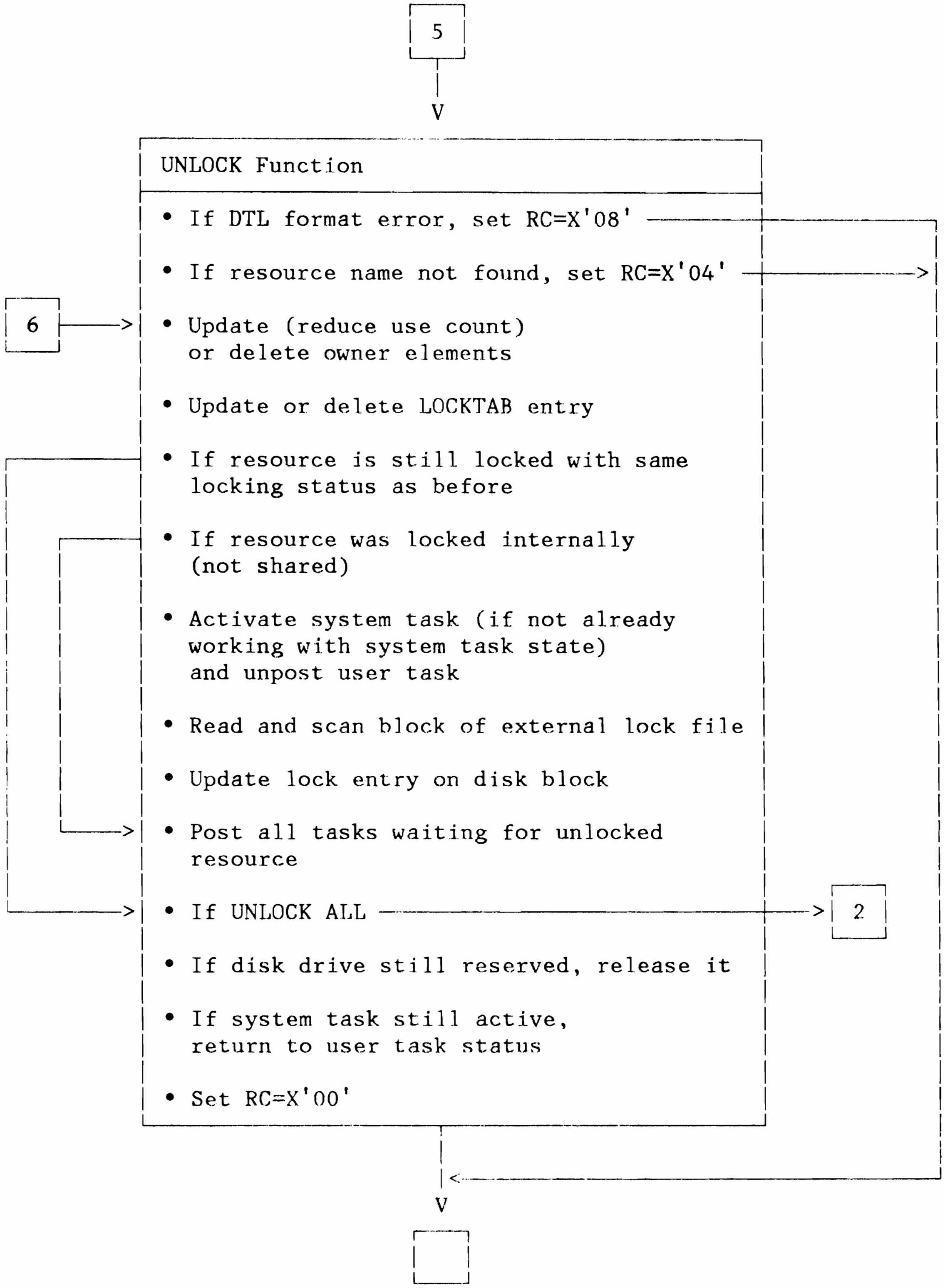


- Enter resource name into block and write back disk block
- Deactivate system task. Activate user task
- Update LOCKTAB, set RC=X'00'
- Return code processing dependent on RC value and FAIL option
  - Return to user
  - Wait for resource
  - Cancel task

UNLOCK ALL Function

- Point to first/next LOCKTAB entry
- If LOCKTAB exhausted, exit to dispatcher else process LOCKTAB entry

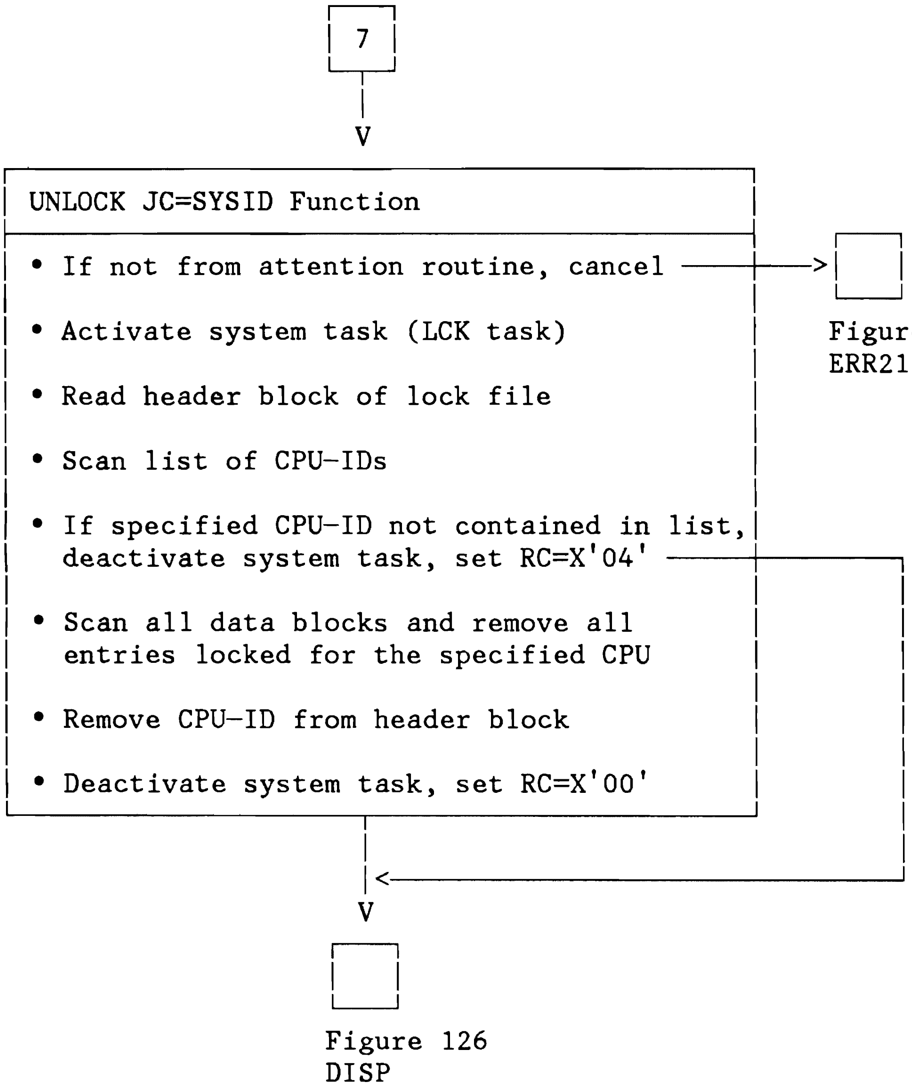
Figure 224 (Part 2 of 5). Lock Manager



UNLOCK Function

- If DTL format error, set RC=X'08'
- If resource name not found, set RC=X'04'
- Update (reduce use count) or delete owner elements
- Update or delete LOCKTAB entry
- If resource is still locked with same locking status as before
- If resource was locked internally (not shared)
- Activate system task (if not already working with system task state) and unpost user task
- Read and scan block of external lock file
- Update lock entry on disk block
- Post all tasks waiting for unlocked resource
- If UNLOCK ALL
- If disk drive still reserved, release it
- If system task still active, return to user task status
- Set RC=X'00'

Figure 224 (Part 3 of 5). Lock Manager



- If not from attention routine, cancel
- Activate system task (LCK task)
- Read header block of lock file
- Scan list of CPU-IDs
- If specified CPU-ID not contained in list, deactivate system task, set RC=X'04'
- Scan all data blocks and remove all entries locked for the specified CPU
- Remove CPU-ID from header block
- Deactivate system task, set RC=X'00'

Figure 224 (Part 4 of 5). Lock Manager

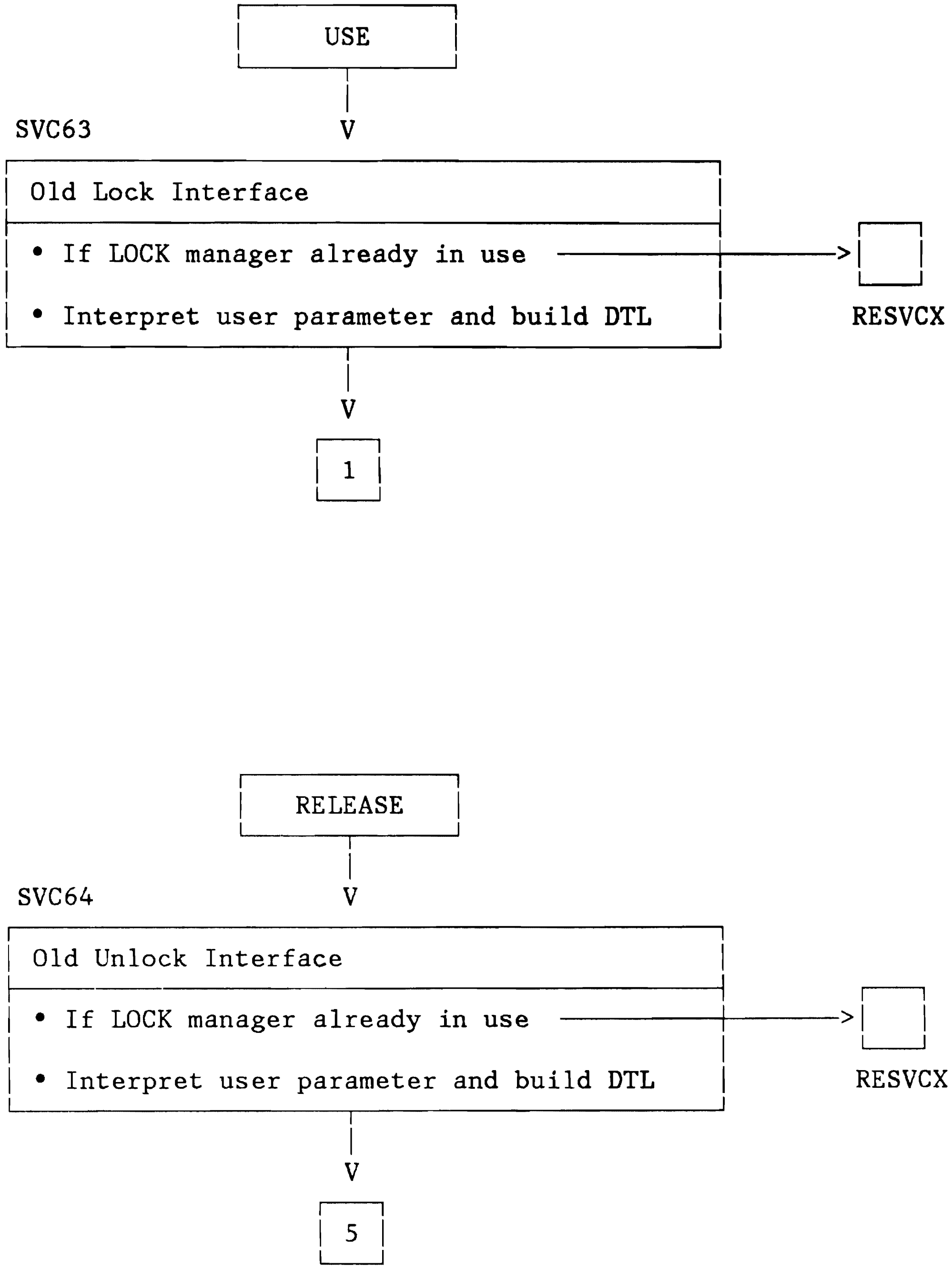

USE
SVC63
Old Lock Interface
• If LOCK manager already in use
• Interpret user parameter and build DTL
RESVCX
1
RELEASE
SVC64
Old Unlock Interface
• If LOCK manager already in use
• Interpret user parameter and build DTL
RESVCX
5

Figure 224 (Part 5 of 5). Lock Manager

CHAPTER 4: DATA AREA INFORMATION

Data Areas Introduction

This chapter provides detailed information on supervisor data areas (tables, regions, save areas etc.) which are commonly used. It does however not show all supervisor data areas.

The following is a list of data areas contained in this chapter:

Input/Output Control Words, Blocks and Areas

- "Machine and Channel Check Control Blocks" on page 603
- "Track Hold Table (THTAB)" on page 609

Flags and Function Codes

- "Task Status Flags" on page 611
- "SVC 107 (X'6B') Function Codes" on page 613

For the recorder file table (RFTABLE) and PUB2 table format, refer to:

VSE/Advanced Functions Diagnosis Reference: Error Recovery and Recording Transients

SYSTEM COMMUNICATION REGION (SYSCOM)

Bytes			
Dec	Hex	Label	Description
0-3	0-3	IJBERBLC	Address of error block
4-7	4-7	IJBHWC	Hard Wait Code (Note 2)
8-11	8-B	IJBERR19	Address of CANCEL exit for ERP message writer
12-15	C-F	IJBPUBRS	Pointer to SYSRES PUB (set by IPL)
16-19	10-13		Reserved
20-23	14-17	IJBSPAVT	Address of supervisor address vector
24-27	18-1B	IJBPTCOM	Address of VSE/PT communication area
28-31	1C-1F	IJBLTA	Address of Logical Transient Area
32-35	20-23	IJBPPBEG	Begin of Problem Program Area (set by IPL)
36	24	IJBFLPTR	Free list pointer
37-39	25-27	IJBCHANQ	Address of Channel Queue
40-41	28-29	IJBQSIZE	Number of Channel Queue entries
42-43	2A-2B	IJBQLNG	Length of ERP Error Queue entry
44-45	2C-2D	IJBNPART	Number of partitions
46	2E	IJBFLG05	Flag byte
		IJBAF	X'80' AF System (always on)
			40 DOS/VSE System (always on)
			20 TP Balancing not active
			10 Reserved
			08 Console Buffering (CBF) active
			04 Reserved
		IJBCSCDS	02 CS and CDS supported
		IJBSIPOF	01 SIPO format flag
47	2F	IJBFLG06	Flag byte
		IJBEMODE	X'80' ECPS:VSE mode
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
		IJBCKD	04 CKD support generated (always on)
		IJBFBA	02 FBA support generated (always on)
		IJB3800	01 3800 support generated (always on)

Notes:

1. The address of SYSCOM can be found at fixed location X'80' - X'83'.
2. "Hard Wait Codes" on page 624.

Figure 225 (Part 1 of 5). System Communication Region (SYSCOM)

Bytes			
Dec	Hex	Label	Description
48-51	30-33	IJBVSIZE	Total virtual storage size
52	34	IJBCONSP	
		IJBOCFLG	DOC configuration byte
			X'80' CRT support initialized
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Support for 3277 screen (always on)
			01 CRT support generated (always on)
53-55	35-37		Address of Console Communication Area (Address of CRT Table)
56-59	38-3B	IJBOCFCM	Address of OCCF Communication Area
60-63	3C-3F	IJBVIOCM	Address of VIO Communication Area
64	40	IJBFLG01	RMS flag byte
		IJBRMSR	X'80' RMSR support generated (always on)
		IJBRMS	40 Full RMS support (always on)
			20 Always off
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
		IJBITDWN	01 IT support down (Clock damage)
65	41	IJBFLG02	Flag byte
		IJBCBF	X'80' Console buffering active
		IJBJA	40 Job Accounting support active (SYS JA=YES)
		IJBDSDFP	20 DASD File Prot. support active (SYS DASDFP=YES)
		IJBSEC	10 Access Control support active (SYS SEC=YES)
			08 Reserved
			04 Channel Scheduler entered after interrupt
		IJBMPXGT	02 Byte MPX channel gating (switched on/off by AR MPXGTN ON/OFF)
		IJBIPLAC	01 IPL in progress

Figure 225 (Part 2 of 5). System Communication Region (SYSCOM)

Bytes			
Dec	Hex	Label	Description
66	42	IJBFLG03	Flag byte
			X'80' Reserved
			40 RAS in special WAIT state
			20 RAS IPL in progress
		IJBIPLV	10 Virtual storage has been initialized by IPL
			08 VSE/POWER supported (always on)
			04 VSE/POWER initialized
			02 GETREAL in progress
			01 Reserved
67	43	IJBFLG04	Flag byte
			X'80' System GETVIS Area initialized
			40 EXCP REAL supported (always on)
			20 CDLOAD supported (always on)
		IJBVMBTM	10 BTAM AUTOPOLL enabled
			08 XECB supported (always on)
			04 Reserved
			02 Batch deactivated by TPIN
		IJBVMLE	01 VM Linkage Enhancements (MODE=VM)
68-69	44-45	IJBHSTID	Highest system task TID
70-71	46-47	IJBHMTID	Highest maintask TID
72-75	48-4B	IJBVPBEG	Begin of V-Pool for VIO
76	4C	IJBTHPTR	Track Hold free list header
77-79	4D-4F	IJBTKHLD	Address of Track Hold Table
80-87	50-57		Reserved
88-89	58-59	IJBLIK	Task ID of LTA owner
90-91	5A-5B	IJBTIK	Task ID of current task
92-95	5C-5F	IJBPWR	Address of POWER Table
96-99	60-63	IJBTCAVT	Address of VTAM Address Vector Table
100-103	64-67	IJBRFTAB	Address of RF Table
104-107	68-6B	IJBEUECB	Reserved
108-111	6C-6F	IJBOLTEP	Flag byte and address of OLTEP Bucket
108	6C	IJBOLTSW	Flag byte
		IJBOLTAC	X'80' OLTEP is active
109-111	6D-6F	IJBOLTPT	Address of OLTEP Bucket
112-115	70-73	IJBRASLN	Address of RAS Linkage Area
116-119	74-77	IJBTRTAB	Address of ASCII Table
120-123	78-7B	IJBPBOWN	Address of PUB Ownership Table
124-127	7C-7F	IJBJATAB	Address of Job Accounting Common Table
128-131	80-83	IJBPROCT	Address of Procedure Common Table
132-135	84-87	IJBIJBSD	Used by SDAID
136-139	88-8B	IJBSAVSD	Address of SDAID Area
140-143	8C-8F	IJBLNSTB	Address of Line Mode Table

Figure 225 (Part 3 of 5). System Communication Region (SYSCOM)

Bytes		Label	Description
Dec	Hex		
144-147	90-93	IJBARBUF	Address of AR input buffer
148-151	94-97	IJBAPTA	Address of Physical Transient Area
152-153	98-99	IJBNDEV	Number of ADD-ed devices
154-155	9A-9B	IJBNSDEV	Number of ADD-ed partition sharable devices
156-157	9C-9D	IJBVTPIK	VTAM PIK (set by SUBSID)
158-159	9E-9F	IJBPWPIK	POWER PIK (set by SUBSID)
160-161	A0-A1	IJBICPIK	ICCF PIK (set by SUBSID)
162-163	A2-A3		Reserved
164-165	A4-A5	IJBLPBDV	PUB pointer of printer buffer load
166-167	A6-A7	IJBPHLSL	Length of phase load list
168-169	A8-A9	IJBDMPDV	cuu of SYSDMP device (from IPL DEF command)
170-171	AA-AB	IJBRECDV	cuu of SYSREC device (from IPL DEF command)
172-173	AC-AD	IJBCATDV	cuu of SYSCAT device (from IPL DEF command)
174-175	AE-AF	IJBRESDV	cuu of SYSRES device
176-179	B0-B3	IJBTTAB	Address of Task Timer Table
180-183	B4-B7	IJBSMCOM	Addr. of storage management comm.area
184-187	B8-BB	IJBPMCOM	Addr. of page management comm. area
188-189	BC-BD	IJBTPBAL	TP Balancing parameter
190-191	BE-BF	IJBTTPID	PIK of partition owning the Task Timer
192-202	C0-CA	IJBMFCER	Repositioning information for 2560/5424/5425 ERP
203	CB	IJBNERQ	Number of ERP Error Queue entries (always=1)
204-205	CC-CD	IJBPUBLN	Length of PUB Table
206-207	CE-CF	IJBAPNO	Number of active virtual partitions
208-211	D0-D3	IJBSEGT	Address of Segment Table (only for MODE=370)
212-215	D4-D7	IJBAPT	Address of page table
216-217	D8-D9	IJBNPGR	Total number of programmer LUB's (NPGR parameter)
218-219	DA-DB	IJBGHLUB	Highest used BG programmer Logical Unit
220-223	DC-DF	IJBASMCB	Address of SMCB Address Table
224-227	E0-E3	IJBDPDTB	Address of DPD Table
228-229	E4-E5	IJBOCDEV	cuu of SYSLOG device
230-231	E6-E7	IJBNTASK	Number of subtasks supported
232-235	E8-EB	IJBSSBEG	Addr. of first byte after supervisor
236-239	EC-EF	IJBEOR	End of real storage (only for MODE=370)
240-243	F0-F3	IJBFTTAB	Address of system library offsets for FETCH

Figure 225 (Part 4 of 5). System Communication Region (SYSCOM)

Bytes Dec	 Hex	Label	Description
244-247	F4-F7	IJBSVA	Flag and address of SVA
244	F4	IJBSVAFL	Flag byte for Shared Virtual Area
			X'80' Reserved
			40 SDL active
			20 Reserved
			10 SDL build in progress
			08 SDL overflow
			04 High level SDL search
			02 Reserved
			01 Reserved
245-247	F5-F7	IJBSVAAD	Address of Shared Virtual Area
248-251	F8-FB	IJBSVIS	Address of System Getvis Area
252-255	FD-FF	IJBARPSL	Address of RPS Local Directory List
256-259	100-103	IJBARPSR	Address of RPS Sector Calculation Routine
256	100	IJBRPSIS	RPS flag byte
260-263	104-107	IJBDLAB	Address of System Code
264-267	108-10B	IJBASY	Flag and addr. of Asynch. Operator Communication Table
264	108	IJBASYFL	Flag byte for Asynchroneous Operator Communication
			X'80' Reserved
			40 ASYNOC task is active
			20 Read is requested
			10 Reply or command is already in input buffer
			08 Reserved
			04 Print message 0D13D
			02 Message 0D13D has been printed
			01 Reserved
265-267	109-10B		Address of Asynchronous Operator Communication Table
268-271	10C-10F	IJBSLACB	Address of SLA work areas
272-275	110-113	IJBSVIPL	Address of Supervisor-IPL Communication Area
276-279	114-117	IJBAMSVA	Address of SVA module area
280-283	118-11B	IJBNPDA	Address of NPDA appendage
284-287	11C-11F	IJBETSS	Address of ICCF Vector Table
288-291	120-123	IJBSCTAB	Address of Security Vector Table
292-295	124-127	IJBPCSAV	Address of Special Save Area for error in system code
296-299	128-12B	IJBINSTR	Pointer to instrumentation data
300-303	12C-12F	IJBPLCT	Address of Librarian Control Table
304	130	IJBFINSC	End of system communication area

Figure 225 (Part 5 of 5). System Communication Region (SYSCOM)

PARTITION COMMUNICATION REGION (COMREG)

Bytes			
Dec	Hex	Label	Description
0-7	0-7	JOBDATE	MM/DD/YY or DD/MM/YY Updated by GETIME macro or set by DATE. Format is controlled by bit 0 of byte 53, see below.
8-11	8-B		Reserved
12-22	C-16	COMUSCR	User area
23	17	UPSI	User program switch indicator (UPSI byte)
24-31	18-1F	COMNAME	Job name from JOB statement
32-35	20-23	PPEND	End address of program space within partition
36-39	24-27	HIPHAS	End address of last phase loaded
40-43	28-2B	HIPROG	End address of largest phase for a multi-phase program (see SVC 51-X'33'
44-45	2C-2D	LABLEN	Length of Problem Program label area (always 0)
46-47	2E-2F	PID	Partition identifier (PIK), same as PIB offset BG COMREG: PIK of active partition
48-51	30-33	EOCADR	End address of virtual storage
52	34	CONFIG	Machine configuration byte X'80' Standard storage protection (always on)
			40 Decimal feature (always on)
			20 Floating point feature (always on)
			10 Physical transient overlap option (always on)
			08 Standard timer feature (always on)
			04 Channel switching supported (always on)
			02 Support for burst mode on byte MPX (always on)
		RMSBIT	01 RMS support available (always on)

Note: The address of the communication region of the active partition can be found at fixed location X'14' - X'17'.

Figure 226 (Part 1 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
53	35	LTACT	System configuration byte
		DDMMYY	X'80' DDMMYY date format convention
			40 Two or more partitions (always on)
		DASDFPSW	20 DASD file protect active (SYS command)
			10 SYSFIL support (always on)
			08 Teleprocessing support (always on)
			04 Two more partitions (always on)
			02 Multitasking support (always on)
			01 Track Hold support (TRKHLD parameter)
54	36	SOB1	Standard language translator options (generated value 1100110, changed by STDOPT statement
			X'80' DECK option, object modules on SYSPCH
			40 LIST option, source listings and diagnostic on SYSLST
			20 LISTX option, hexadecimal object modules listings on SYSLST
			10 SYM option, symbol tables on SYSLST/SYSPCH
			08 XREF option, cross reference list on SYSLST
			04 ERRS option, diagnostics on SYSLST
			02 CHARSET option, 60 character set (else 48)
			01 Reserved
55	37	SOB2	Flag byte
			X'80' Always on
			40 STDOPT DUMP=YES or PART
			20 Partition waiting for volume mount (Job Control)
			10 STDOPT LOG=YES
		DUMDVC	08 Dummy device search in progress
			04 Reserved
			02 Relocating loader supported (always on)
			01 ASCII supported (always on)

Figure 226 (Part 2 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
56	38	JCSW1	Flag byte
		JASWITCH	X'80' Job Accounting not active (SYS command)
		JCOPEN	40 Return to caller on LIOCS disk open failure)
		JCINRDR	20 Job Control input from SYSRDR (else SYSLOG)
			10 Job Control output on SYSLOG
		JOBEND	08 Skip to end of job
			04 Pause at end of job step (JC PAUSE statement)
			02 Always 0
			01 SYSLOG assigned to same device as SYSLST
57	39	JCSW2	Linkage Editor control byte
			X'80' SYSLNK open for output
		IJBACTCL	40 Action clear indicator
		ALLOWEX	20 Allow EXEC
			10 Catalog Linkage Editor output
		IGNTESTM	08 Ignore test mode
			04 Reserved
			02 Reserved
			01 Reserved
58	3A	JCSW3	Non-standard language translator options (set by OPTION statement)
			X'80' DECK option, object modules on SYSPCH
			40 LIST option, source listings and diagnostic on SYSLST
			20 LISTX option, hexadecimal object modules listings on SYSPCH
			10 SYM option, symbol tables on SYSLST/SYSPCH
			08 XREF option, cross reference list on SYSLST
			04 ERRS option, diagnostics on SYSLST
			02 CHARSET option, 60 character set (else 48)
			01 Rewind/unload option

Figure 226 (Part 3 of 9). Partition Communication Region (COMREG)

Bytes		Label	Description
Dec	Hex		
59	3B	JCSW4	Job Control flag byte
			X'80' Job in progress
		OPTDUMP	40 OPTION DUMP
			20 Pause at end of job step (AR PAUSE command)
			10 OPTION LOG
			08 Temporary assignment for SYSRDR
		TESTMODE	04 SDL scanned as specified by LIBDEF
		DATEBIT	02 DATE statement processed for current job
		BATINIT	01 START/BATCH command just issued
60	3C	JCSW7	Job control flag byte
		OPCNCL	X'80' Indicator for operator cancel
		JCLTSTRN	40 OPTION TSTRUN
		LIBPCHNG	20 LIBDEF PROC change
		PRCFRSTL	10 Used to control check for PROC statement
		PRCFRSTH	08 Used to control check for PROC statement
		IJBOVLOG	04 Procedure overwrite statements to be read from SYSLOG
		IJBJCCNL	02 Job Control CANCEL issued
		IJBUSRMD	01 User mode
61	3D	NSTLEVEL	Procedure nesting level
62	3E	JCSW8	Job control flags
		IJBCNCPD	X'80' Operator cancel pending
		IJBRCCNC	40 RC operator cancel
		IJBARCNA	20 Delay AR cancel
		IJBEOPDL	10 EOP delayed
		IJBABTRM	08 Abnormal termination
			04 Reserved
			02 Reserved
			01 Reserved
63	3F		Reserved
64-65	40-41	PUBPT	Address of PUB Table
66-71	42-47	IJBJOBST	Job start time
72-73	48-49	FICLPT	Address of FICL
74-75	4A-4B	NICLPT	Address of NICL
76-77	4C-4D	LUBPT	Address of partition LUB Table
78	4E	SYSLINE	SYSLST line count as specified by STDOPT LINES=nn
79-87	4F-57	SYSDATE	System date, MMDDYYDDD or DDMMYYDDD
79-82	4F-52	MMDD	MMDD or DDMM
83-87	53-57	YYDDD	YYDDD portion of date

Figure 226 (Part 4 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
88	58	LIOCSCOM	LIOCS communication byte 1
			X'80' Reserved
		LIOCSRDS	40 Return to $$BODSMO
			20 Reserved
			10 Reserved
			08 Reserved
		LIOCSOIP	04 Open/close in progress
		LIOCSCPO	02 CP/DI open indicator
			01 Reserved
89	59		LIOCS communication byte 2
		LIOCSRSV	X'80' Return from SVA
			40 Reserved
			20 Reserved
			10 Reserved
		LIOCSRTM	08 Reserved
		LIOCSRLB	04 Return from $$BOPLBL
		LIOCSQMT	04 QTAM DTF
		LIOCSRLK	02 Return from 'LOCK'
			01 Reserved
90-91	5A-5B	PIBPT	Address of PIB Table
92-93	5C-5D	CHKPTID	ID of last checkpoint
94-95	5E-5F	JOBZON	Job zone in minutes. Value is positive for ZONE=EAST and negative for ZONE=WEST.
96-97	60-61	DIBPT	Address of partition DIB Table
98	62	DEVFLG1	Flag byte
		OPN3800	X'80' One or more 3800 extended buffering DTF's open
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
99	63	OPNBYT2	Flag byte
		BTAMFLG	X'80' BTAM active in partition
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
100-105	64-69		Reserved
106-107	6A-6B	PWTIMS	PIK of partition
108-109	6C-6D	IJBSPID	Space id (370 only)

Figure 226 (Part 5 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
110-111	6E-6F	LTK	PIK of part. owning the LTA
			(set only in BG COMREG)
112-115	70-73	SYSPAR	Address of SYSPARM field
116-119	74-77	JAPART	Address of Job Accounting Table
120-123	78-7B	TODCOM	Address of TOD common area
124-125	7C-7D	PIB2PTR	Address of PIB2 Table
126-127	7E-7F	PDTABB	Address of MICR DTF Table
128-131	80-83	LABELPTR	Reserved for LIOCS
132-133	84-85	BGCOMPT	Address of BG COMREG
134	86	OPTNBYTE	Flag byte
			X'80' Reserved
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
		JAPGCIND	02 Count pages for Job Accounting
		ANCHTBIT	01 GETVIS area initialized
135	87	RMSROPEN	Flag byte
			X'80' PCIL support (always on)
		TODBIT	40 TOD support (always on)
			20 PFIX support (always on)
			10 Fetch $$BOPEN by $JOBCTLJ
			08 Fetch $$BOPEN by $JOBCTLD
			04 Fetch $$BOPEN by $JOBCTLJ
			02 Reserved
			01 RPS support
136-139	88-8B	IJBJCWA	Addr. of job control work area
140	8C	STDOPT	Job Control standard option
			(STDOPT statement)
			Generated value is 010000-0
			X'80' EDECK
			40 ALIGN
		OPTPDUMP	20 PARTDUMP
			10 RLD
			08 SXREF
			04 TERM
			02 Reserved
			01 ACANCEL

Figure 226 (Part 6 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
141	8D	TEMOPT	Job Control temporary option (OPTION statement)
			X'80' EDECK
			40 ALIGN
			20 PARTDUMP
			10 RLD
			08 SXREF
			04 TERM
			02 SUBLIB=DF
			01 Reserved
142	8E	DISKCONF	Disk configuration byte
			X'80' Reserved
			40 Reserved
			20 Reserved
			10 Reserved
			08 3350 support (always on)
			04 3340 support (always on)
			02 3330 support (always on)
			01 2311 and 2314/2319 support (always on)
143-150	8F-96	PROCNAM	Procedure name
151	97	PSWTCH	Interface byte for Cataloged Procedures
		IJBPCALL	X'80' Procedure being executed
		IJBPOVMD	40 Overwrite processing
		IJBPDATA	20 SYSIPT data present
		IJBPOVRQ	10 Overwrite request for Job Control
		IJBPINST	08 Insert request for Job Control
		IJBPNDMK	04 End of procedure
		IJBPSLOG	02 Called from SYSLOG
		IJBPOVBT	01 Overwrite request for supervisor
152-158	98-9E	POVNAM	JCL statement name for Cataloged Procedure
159	9F	INSIZE	Flag byte
			X'80' Permanent 81 bytes on SYSRDR
			40 Permanent 81 bytes on SYSIPT
		RDR81T	20 Temporary 81 bytes on SYSRDR
		IPT81T	10 Temporary 81 bytes on SYSIPT
			08 Reserved
			04 Reserved
			02 Reserved
		CATALSA	01 Allow /& within procedure to be catalogued

Figure 226 (Part 7 of 9). Partition Communication Region (COMREG)

Bytes		Label	Description
Dec	Hex		
160-163	A0-A3	POWPCB	Pointer to VSE/POWER Partition Control Block
164	A4	POWFLG1	VSE/POWER flag byte
		POWACCT	X'80' POWER accounting supported
		POWUPART	40 POWER controlled partition
		POWPART	20 POWER partition
		POWPDORM	10 POWER partition dormant
		POWWPART	08 POWER controlled partition waiting for work
			04 Reserved
			02 Reserved
			01 Reserved
165	A5	POWFLG2	Reserved for VSE/POWER
166-167	A6-A7	IJBVSSNP	VSAM snap dump function bytes
166	A6	IJBSNP01	X'80' SNAP dump indicator 1
			X'40' SNAP dump indicator 2
			X'20' SNAP dump indicator 3
			X'10' SNAP dump indicator 4
			X'08' SNAP dump indicator 5
			X'04' SNAP dump indicator 6
			X'02' SNAP dump indicator 7
			X'01' SNAP dump indicator 8
167	A7	IJBSNP09	X'80' SNAP dump indicator 9
			X'40' SNAP dump indicator 10
			X'20' SNAP dump indicator 11
			X'10' Reserved
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
168-171	A8-AB	LUBEXT	Address of LUB Extension Table
172	AC	JCSW5	Flag byte
			X'80' EXEC LNKEDT statement to be generated
			40 EXEC statement to be generated
			20 Skip link and execution, except for OPTION LINK
			10 NEWVOL ignored
		LSTLOG	08 LISTLOG called for cancel
		ASIPL	04 ASI IPL
		ASICONT	02 Job Control first time activation passed
		JCLACTIV	01 Job Control active

Figure 226 (Part 8 of 9). Partition Communication Region (COMREG)

Bytes			
Dec	Hex	Label	Description
173	AD	JCSW6	Flag byte
			X'80' Reserved
		ONLNSYSG	40 On-line system generation
			20 Reserved
		JOBLOGSW	10 Write job statement to HC file
		JCLOUTA	08 Alternate assignments exist for SYSOUT
		SLAACTIV	04 SLA active
		SYSPROC	02 System procedure library in use
		IJBFNSLB	01 Allow to add system labels from this partition (Fn)
174	AE	STDOPT2	Reserved for Job Control standard options (STDOPT)
		OPTNFSTR	X'80' NOFASTTR
		OPTSDUMP	40 SYSDMP
		OPTPROC	20 PROC
		OPTPARM	10 PARM
		OPTJCNCL	08 JCANCEL
		OPTNHCTR	04 NOHCTRAN
			02 Reserved
			01 Reserved
175	AF	TEMOPT2	Job Control temporary options (OPTION statement)
			X'80' NOFASTTR
			40 SYSDMP
			20 PROC
			10 PARM
			08 JCANCEL
			04 NOHCTRAN
			02 Reserved
			01 Reserved
176–179	B0–B3	IJBJPL	Address of JPL of partition
180–183	B4–B7	IJBAFCB	Reserved for CICS
184–187	B8–BB	IJBPHLST	Address of Fetch/Load Trace Table
188–195	BC–C3	IJBJOBLG	Address of last job statement on
188	BC		Cycle byte of job statement
189–193	BD–C1	IJBDSKAD	Disk address of job statement
194–195	C2–C3	IJBDSKLR	Logical record of job statement
196–199	C4–C7	IJBASPF	Address of SPF control information
200–203	C8–CB	IJBGVCTL	Address of GETVIS control information
204–207	CC–CF	IJBIJJT	Address of Tape Open control block
208–215	D0–D7	IJBSPNAM	System GETVIS partition pool
216–223	D8–DF	IJBPHNAM	Exec phase name
224–227	E0–E3	IJBDECPY	Mirror DE entry chain
228	E4	COMREND	End of partition communication region

Figure 226 (Part 9 of 9). Partition Communication Region (COMREG)

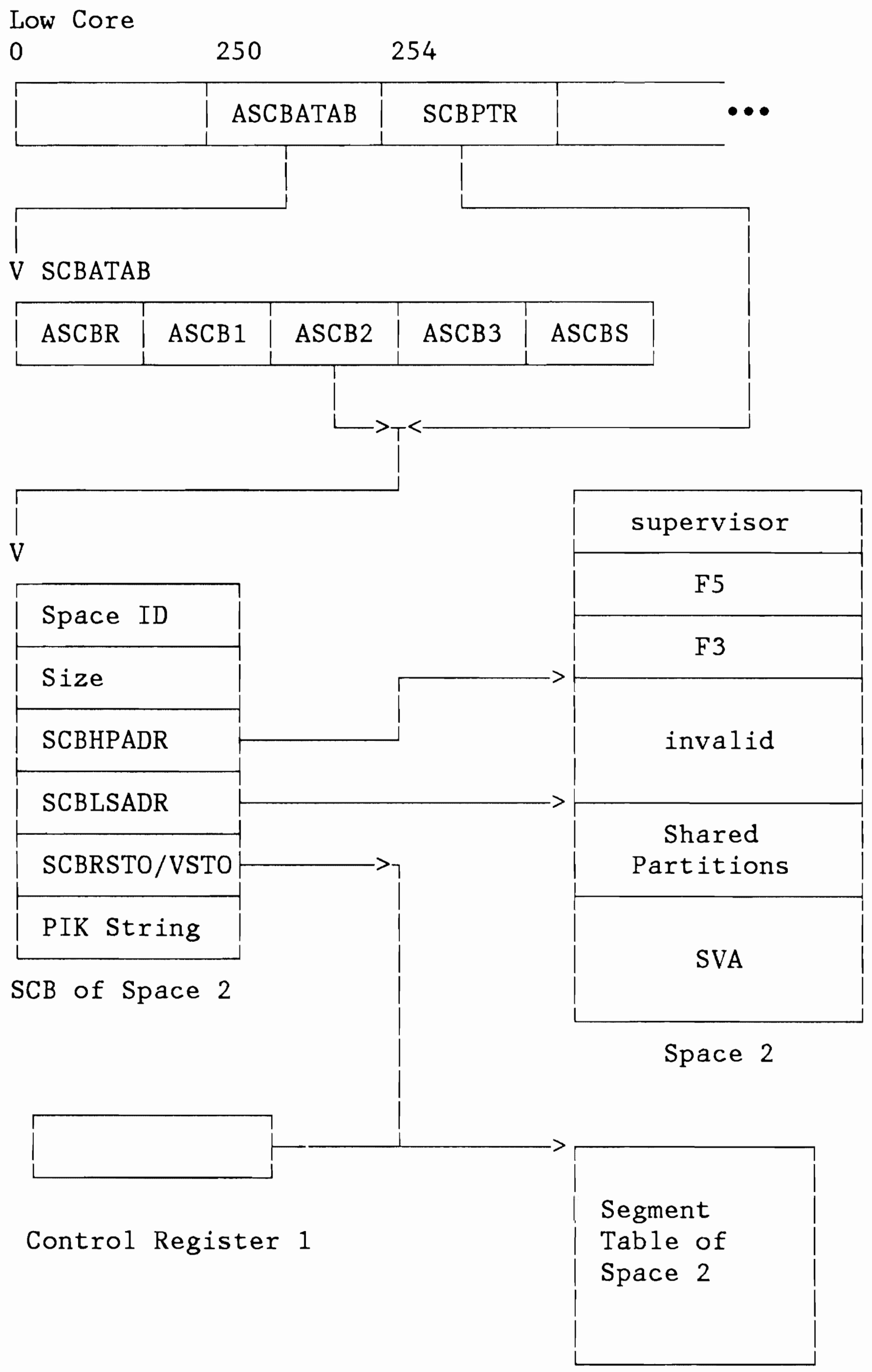


Figure 227. Space Control Block (SCB) Data Relationship for a /370 VAE System

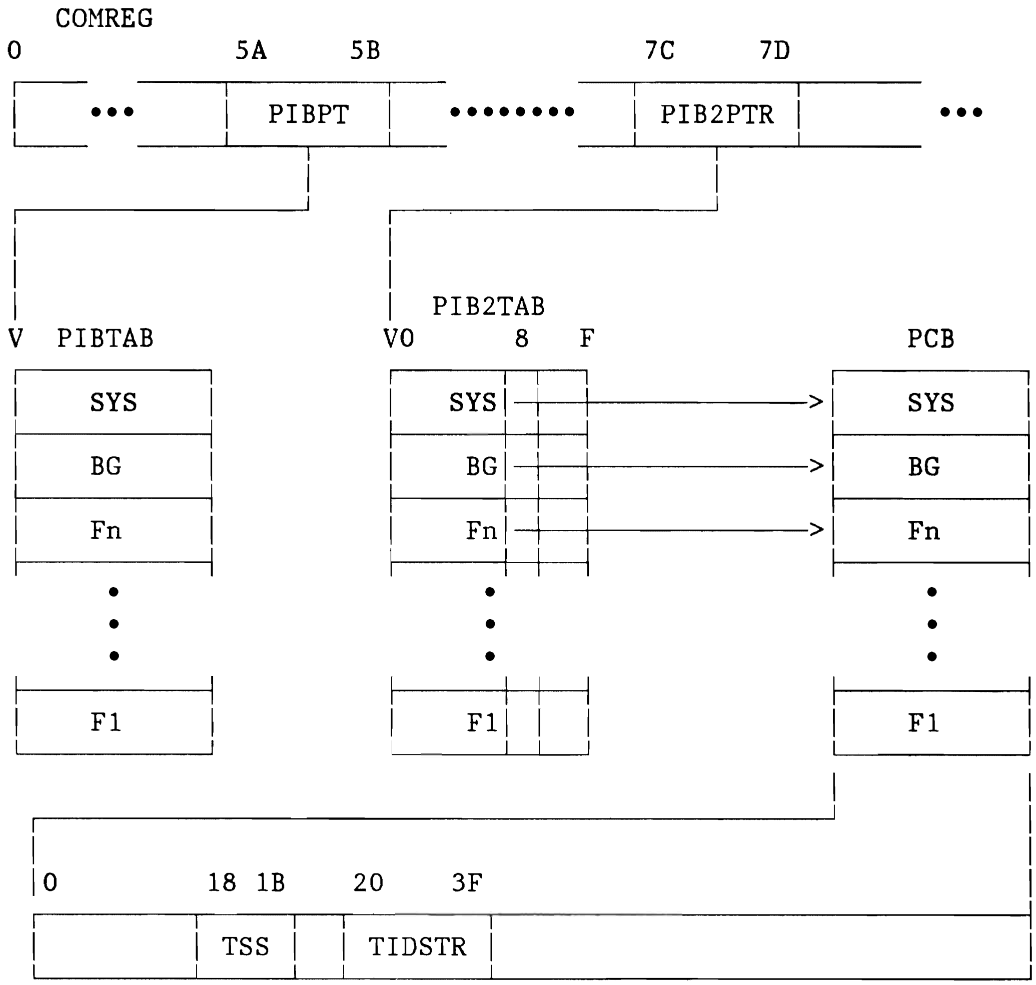
Bytes Dec	Hex	Label	Description
0	0	SCBID	Symbolic space identifier predefined values: 'R ' for real space '1 ' for primary virtual spaces 'N ' for add. virtual spaces (370) 'S ' for shared virtual spaces (370)
2	2	SCBSPN	Space number
3-5	3-5		Reserved
6-7	6-7	SCBSIZE	Size of allocated part. space in K
..................End of SCB of E and VM mode.........................			
8-11	8-B	SCBHPADR	Upper limit of private area
12-15	C-F	SCBLSADR	Lower limit of shared area
16-19	10-13	SCBVSTO	Virtual address of segment table
20	14	SCBRSTO	Segment table origin for DAT
		SCBSTL	(length of segment table)/64-1
21-23	15-17		Reserved
24-55	18-37	SCBPSTR	PIK list of allocated partitions

Figure 228. Space Control Block (SCB)

PARTITION CONTROL BLOCKS (PIB, PIB2, PCB)

PIB	(Partition Information Block)
PIB2	(Partition Information Block Extension)
PCB	(Partition Control Block)

The PIB, the PIB2 and the PCB contain static and dynamic status information about the system and about partitions. There is one set of these control blocks for the system and one for each partition generated (NPARTS specification), see Figure 229.



Notes:

1. Fn = Foreground partition "n"
2. @ = Address
3. @(FnPIB) = @(PIBTAB) + FnPIK
4. @(FnPIB2) = @(PIB2TAB) + FnPIK
5. @(@(FnPCB)) = @(FnPIB2) + 8

Figure 229. Partition Control Blocks Interrelationship

The PIBs and the PIB2s are each 16 bytes long. They are arranged into tables (PIBTAB and PIB2TAB) with NPARTS+1 entries. Each Partition Communication Region contains the address of PIBTAB in bytes 90-91 (X'5A'-X'5B') and the address of the PIB2TAB in bytes 124-125 (X'7C'-'X7D'). The first entry of each table belongs to the system. The PIB/PIB2 for a given partition is found by adding the PIK of this partition to the begin address of the appropriate table. For details see Figure 230 on page 497 and Figure 231 on page 498. The system and partition PCBs have different length, some of their common fields also have different contents. The actual length of a PCB is contained in its first two bytes. Figure 232 on page 499 shows the layout of the PCB.

Partition Information Block (PIB)

Bytes		Label	Description
Dec	Hex		
0	0	PIBFLG	Partition status byte
		STOPPED	X'82' Partition is stopped
		INACT	80 Partition is inactive/unbatched
			00 Partition is active
1	1	PIBOLDST	Maintask status at operator cancel time
2-3	2-3	PIBLOGID	SYSLOG ID (AR,BG,F1,...,Fn)
4	4	PIBDATFL	Flag byte
		PIBTRAM	X'80' Partition running in virt. mode
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
5-7	5-7	PIBSVADD	Begin address of virtual partition
8	8		Reserved
9-11	9-B	PIBSAV2	Problem Program save area of LTA owner (system PIB)
12	C	PIBPUBAS	Flag byte
			X'80' Reserved
		APPEN	40 Channel appendage allowed
			20 Reserved
			10 Reserved
			08 Foregr. assignments to be hold
			04 Reserved
			02 Reserved
			01 Reserved
13	D	PIBLUBID	Number of System LUB's (AR PIB: Number of BG system LUB's)
14	E	PIBLUBNO	Number of Progr.LUB's (AR PIB: Number of BG progr. LUB's)
15	F	PIBFLG2	Flag byte
			X'80' Reserved
			40 Reserved
		JOBDUN	20 End of Job indicator
			10 Partition stopped (set by Job Control)
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved

Figure 230. Program Information Block (PIB)

Partition Information Block Extension (PIB2)

Bytes Dec	 Hex	Label	Description
0-1	0-1	PIBCOMRA	Address of Communication Region (AR PIB: BG COMREG)
2-3	2-3	PIBSLUB	Index of System LUB's relative to LUB table (always 0)
4-5	4-5	PIBMTID	Task ID of main task
6-7	6-7		Reserved
8-11	8-B	PIBPCB	Address of Partition Control Block (PCB)
12-13	C-D	PIBPRTID	PIK of partition (0 for AR PIB2)
14-15	E-F		Bytes 2 and 3 of ECB for ATTACH limit within partition
15	F	PIBFLG3	Extension flags

Figure 231. Partition Information Block Extension (PIB2)

Partition Control Block (PCB)

Bytes		Label	Description
Dec	Hex		
0-1	0-1	PCBLNGTH	Length of PCB
2	2	PCBFLAG	Flag byte
		BALANCED	X'80' Balanced partition
		PERACT	40 Reserved
		SUPPRPFH	20 Suspend page fault handling (load leveller)
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
		PWSRVFLG	01 Some task within partition waiting for POWER
3	3		Reserved
4-7	4-7	PCBPMASK	Partition priority mask
8-11	8-B	RUNTIME	Time counter for part.balancing and job accounting
12-15	C-F	PBALTIME	Initial value of part.balancing time
16-19	10-13	PCBJAPTR	PCB pointer for time accounting
20-21	14-15	PCBPIK	PIK of partition
22-23	16-17	PCBLCTSS	Active length code for TIDSTR
23	17	PCBSUBS	Number of attached subtasks
24-27	18-1B	TSS	Task selection bit string
28-31	1C-1F		Reserved
32-63	20-3F	TIDSTR	TID's of attached tasks in priority order
64-67	40-43		Reserved
68-69	44-45	PCBNTASK	Counter of used subtasks
70-71	46-47	CDLDTID	TID of CDLOAD owner within partition
71	47	CDLDBYTE	Significant portion of CDLDTID
72-75	48-4B	PCBPSCB	SCB pointer of allocation space
76-79	4C-4F	PCBASCB	SCB pointer of active space

Figure 232 (Part 1 of 4). Partition Control Block (PCB)

Bytes			
Dec	Hex	Label	Description
Begin of SMCB			
80-81	50-51	SMAXPFIX	PFIX limit in pages (system PCB: SVA PFIX limit)
82-83	52-53	SMPFIX	PFIX count in pages (system PCB: SVA PFIX count)
84-87	54-57	SMPSAVE	Partition PCB: Address of main task save area
		PCBAPBEG	= Active partition begin address
88	58	SMVFLAG	Storage management flag byte X'80' SETLIMIT given indicator 40 Reserved 20 Reserved 10 Reserved 08 Reserved 04 Reserved 02 Reserved 01 Reserved
88-91	58-5B	SMVGVIS	Partition PCB: Addr. of GETVIS area
		SMSGVIS	System PCB: Address of system GETVIS area
92-95	5C-5F	SMVPBEG	Partition PCB: Begin of virtual partition
		SMSVABEG	System PCB: Begin of SVA
96-99	60-63	SMVPEND	Partition PCB: End of virtual partition + 1
		SMSVAEND	System PCB: End of SVA + 1
100-103	64-67	SMRPBEG	Partition PCB: Begin of real partition System PCB: Begin of real area for system PFIX
104-107	68-6B	SMRPEND	Partition PCB: End of real partition + 1 System PCB: End of real area for system PFIX + 1
(28)	(1C)	SMCBLNG	length of SMCB
End of SMCB			

Figure 232 (Part 2 of 4). Partition Control Block (PCB)

Bytes Dec	Bytes Hex	Label	Description
108-111	6C-6F	PCBAPEND	End address + 1 of user key area within partition
112	70	PCBSSCNT	Subsystem counter within partition
113	71		Reserved
114	72	PCBSSFL1	Subsystem flag byte
			X'80' Reserved
			40 Reserved
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
		NPDA	01 NPDA partition
115	73	PCBSSFLG	Subsystem flag byte
		PWR	X'80' POWER partition
		VTAM	40 VTAM partition
		ICCF	20 ICCF partition
		CICS	10 CICS partition
		VCNA	08 VCNA partition
		OCCF	04 OCCF partition
		DS2	02 DS2 partition
		SSX	01 SSX partition
116-121	74-79	CHPTENT	Checkpoint PFIX entry
116-119	74-77	CHPTPAGE	First PFIXCHPT page not yet handled
120-121	78-79	CHPTCNT	Remaining number of PFIX-ed pages for PFIXCHPT
122-123	7A-7B		Reserved
124-127	7C-7F	PCBOCPTR	Address of OC exit routine (partition PCB only)
128-131	80-83	PCBOCSAV	Address of OC exit save area (partition PCB only)
132-139	84-8B	PCPUTIME	CPU time counter
140-147	8C-93	POVHTIME	Overhead time counter
148-155	94-9B	PBNDTIME	Allbound time counter
156-163	9C-A3	PCBRQ	Begin of PCB resource descriptors
		SRQGTV	GETVIS/FREEVIS resource queue header
162	A2	PCBRBGTV	GETVIS/FREEVIS resource byte
164-171	A4-AB	SRQCDL	CDLOAD resource queue header
170	AA	PCBRBCDL	CDLOAD resource byte
172-179	AC-B3	SRQPFX	PFIX resource queue header
178	B2	PCBRBPFX	PFIX resource byte
180-183	B4-B7	PCBCNT	Address of usage and SIO counters for partition sharable devices

Figure 232 (Part 3 of 4). Partition Control Block (PCB)

Bytes			
Dec	Hex	Label	Description
184	B8	FIXTYPE	PFIX flag byte
		GTRBIT	X'80' GETREAL request
		RSTRTBIT	40 PFIXREST request
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
184-187	B8-BB	FIXTIB	TIB pointer of PFIX/GETREAL requestor
188-191	BC-BF	PFTERSVD	Address of reserved PFTE for PFIX/GETREAL
.... End of system PCB			
184-192	B8-C0	PCBPFIXL	Count of tasks with open VTAM ACB's
193-194	C1-C2		Reserved
195	C3	PCBVTCNT	Count of tasks with open VTAM ACB's
(196)	(C4)	PCBVMLNG	Length of PCB for VM
196-215	C4-D7	PHOTIB	Pseudo-TIB for PHO

Figure 232 (Part 4 of 4). Partition Control Block (PCB)

TASK CONTROL BLOCKS (TIB, TCB)

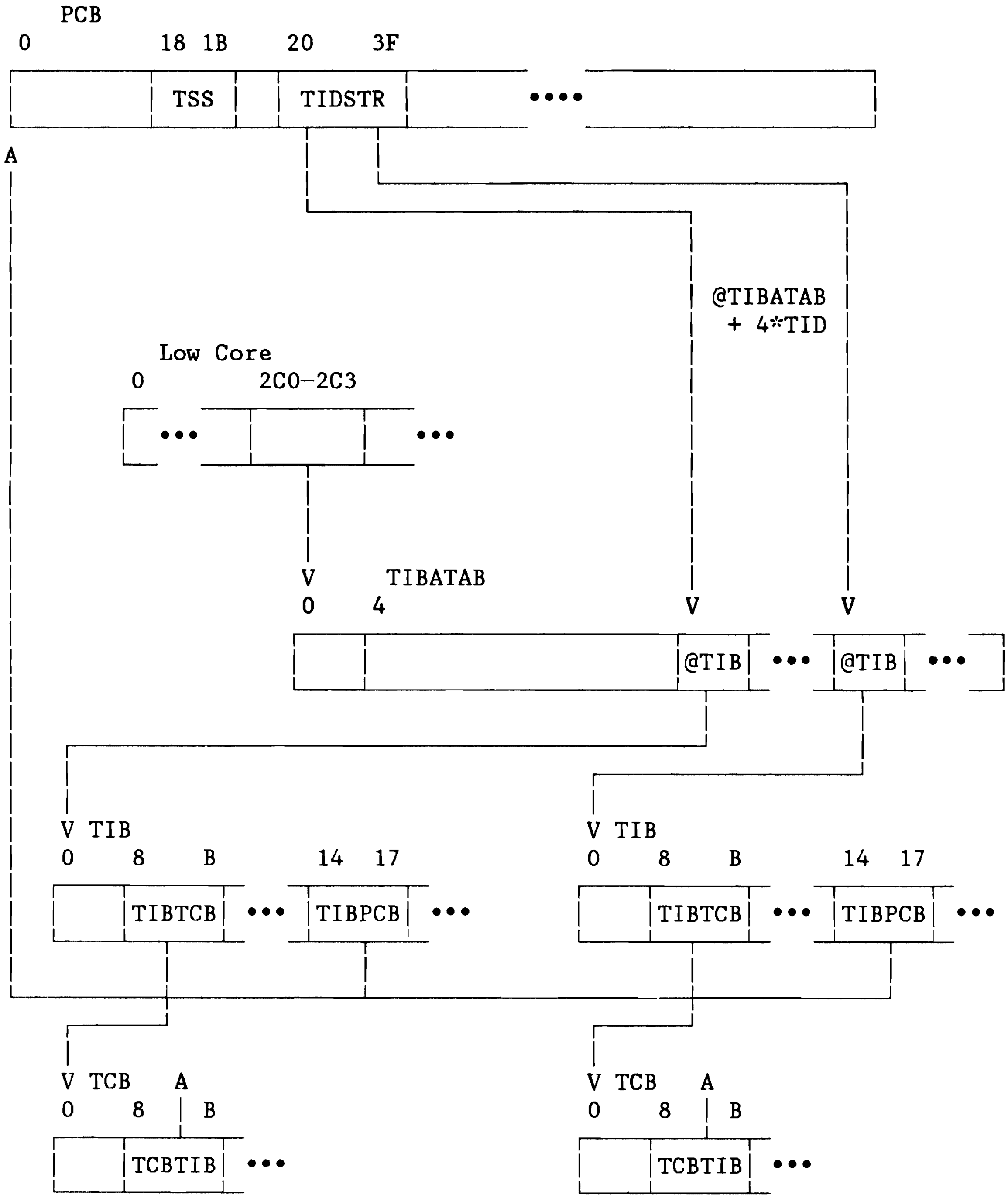
TIB	(Task Information Block)
TCB	(Task Control Block)

The TIB and TCB contain static and dynamic status information on system tasks and user tasks. One set of these control blocks exists for each system task, for each main task (NPARTS specification), and for each generated user subtask (NTASKS specification).

- For the layout of the TIB, see Figure 234 on page 505.
- For the layout of the TCB, see Figure 235 on page 508.

There is a predefined relationship between task identifiers (TID values) and task control blocks. This is obvious for system tasks and user main tasks which are statically assigned to specific supervisor functions or partitions, and always exist in the system. As far as user subtasks are concerned, the control block structure also exists for subtasks currently detached. In order to minimize the real storage requirements subtask TIBs and TCBs are allocated in the SVA.

The TIBs have a fixed length of 36 bytes. They are addressed via an address table (TIBATAB) with offset TID*4. The TIB contains all task-related information which has to be kept in fixed storage, either for logical or for performance reasons.
The length of TCBs is different for system task not using FETCH (short system task TCBs), for system task using FETCH (long system task TCBs), and for the attention and user tasks. The length and layout of the long system task and of the attention and user task TCBs also depend on the supervisor options MODE, FASTTR, and TP. The actual length of a TCB is contained in the field TIBTCB of the corresponding TIB.



Note: @TIB = Address(TIB)

Figure 233. Partition/Task Control Table Relationship

Bytes Dec	 Hex	Label	Description
0	0	TIBCHAIN	Wait chain indicator
			X'00' Task is enqueued in wait chain
			FF Last TIB in resource wait chain
1-3	1-3		Pointer to next TIB in a resource wait chain
4-7	4-7	TIBSTATE	Resource identifier within generic wait chain
8	8	TIBFLAG1	Flag byte
		PHOIND	X'80' Pseudo-TIB for PHO or VIO
		PHOACT	40 PHO initialized for this task
		PHOREQ	40 (PHO TIB only) PHO request enqueued
		EOTACT	20 EOT active
		VIOREQ	20 VIO pseudo-TIB (if PHOIND on)
		EOTINPR	10 EOT subsystem clean-up active
		LTAACT	08 LTA active
		LTAOWNER	04 LTA owner
		TERMACT	02 Terminator active
		SYSACT	01 System code active
		PRIVILEG	1B
8-11	8-B	TIBTCB	TCB pointer
		TIBPFAPP	PHO TIB: Address of PHO appendage
		TIBVIOTB	VIO TIB: VIOTAB pointer
12	C	PGQTYP	Type of page I/O request
		PGSEL	X'80' Page selection required
		PGNCNT	40 Counting already done
		PGO	10 Page-out request
		PGOWAIT	18 Page-out request with waiting task
		PGOPGIN	14 Page-out request with waiting Page-in
		PGOVIO	12 Asynchr. Page-out requ. for VIO
12-15	C-F	PGINF	Address of PDS device control block or of PFTE
16-17	10-11	TIBRTID	User tasks: Task ID
			Syst.tasks: Task ID of service owner
			PHO TIB: Task ID of PHO owner within partition
17	11	TIBRBYTE	Significant byte of TIBRTID
18-19	12-13	TIBRPIK	User tasks: PIK of owner partition
			Syst.Tasks: PIK of serviced partition
(20)	(14)	TIBPFLNG	Length of PHO/VIO TIB
.... End of Pseudo-TIB			

Figure 234 (Part 1 of 3). Task Information Block (TIB)

Bytes			
Dec	Hex	Label	Description
20-23	14-17	TIBPCB	Pointer to PCB of owner partition
24-27	18-1B	TIBPMASK	Priority mask of task within partition
28	1C	TIBFLAG	Dispatcher exit flags
		CSVRET	X'80' Return to supervisor routine
		RETRYSVC	40 Restart SVC pre-processing
		TIBDELMV	20 General delayed move processing
		FETCHEOJ	10 Task termination to be initialized
		ROLLOUT	08 ICCF inter.part. eligible for roll-out
		CDELEX	04 Delayed timer interrupt processing
		OCPEND	02 OC exit to be scheduled
		APSEXFLG	01 Call VTAM exit
29	1D	TIBFLAG2	Flag byte
		ICCFPP	X'80' ICCF Interactive Partition
		PWRMTASK	40 POWER main task
		OVHIND	20 Account CPU-time as overhead
		SVPCCNCL	10 Status saved in special save area
		OCCFACT	08 OCCF service request pending
		ASYOCACT	04 ASYNOC request pending
		VTOPEN	02 At least one VTAM ACB open
		LIBRSERA	01 Librarian service active
30	1E	TIBCNCL	First cancel code
31	1F	TIBCNCL2	Last cancel code
		TERMCNL	X'80' Terminator cancelled
32	20	TIBRQID	Task status flag
33	21	TIBFLAG3	Flag byte
			X'80' Reserved
			40 Reserved
			20 Reserved
		SEIZEBIT	10 Task is seizing the system (see SVC-16)
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
34	22	TIBCNCL3	Terminator cancel code

Figure 234 (Part 2 of 3). Task Information Block (TIB)

Bytes Dec	 Hex	Label	Description
35	23	TIBDMFLG	Del. move flag, used with TIBDELMV
		TIBCMVEX	X'80' Invoke CCB delayed posting
		TIBXPCEX	40 Invoke XPCC delayed move exit
		TIBSFLEX	20 Return to SYSFIL FBA processing
		TIBPERST	10 Invoke PER bit update
		TIBDMALL	F0 Invoke previous services
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
36-41	24-29	TIBITREQ	Significant part of timer interrupt
42-43	2A-2B	TIBITCHN	Address of Y-pointer to next TIB in IT chain
44-47	2C-2F	TIBSCB	Current SCB pointer for task
48	30	TIBLNG	Length of TIB

Figure 234 (Part 3 of 3). Task Information Block (TIB)

Task Control Block (TCB)

Bytes Dec	Bytes Hex	Label	Description
0-1	0-1	TCBLNGTH	Length of TCB
2	2	TCBAUTHF	Authorization flag
		TCBFLAG3	
			X'80' Reserved
			40 Reserved
		CICSMT	20 CICS 'maintask' from SUBSID
		DLIMT	10 DLI 'maintask' from SUBSID
		ISPFMT	08 ISPF 'maintask' from SUBSID
		FTPTSK	04 FTP task allowed to use CPCOM
			02 Reserved
			01 Reserved
3	3	TCBRID	RID saved on interrupt in supervisor service
4-5	4-5	FATHERID	Task ID of attaching task (user subtask only)
6	6	TCBFLAGS	Flag byte
		SYSRESW	X'80' DASD File Protect to be skipped
		SKIPMSG	40 OPTION=NODUMP for STXIT AB
		EARLYAB	20 OPTION=EARLY for STXIT AB
		ACLOSE	10 VSAM Automatic Close in progress
		VSMOPEN	08 VSAM ACB'S open in partition (set for main tasks and ICCF IP's)
			04 Reserved
		ICCFSVC	02 ICCF SVC screening flag
		OWNTIMER	01 Task Timer owner (main task only)
7	7	TCBFLAG2	Flag byte
		CNCLRTRN	X'80' Terminator to be reentered
			40 Reserved
			20 Reserved
		OPENSVA	10 OPEN active in SVA
		SELFTERM	08 Task terminating by itself (EOJ, CANCEL, DUMP, JDUMP, DETACH by user code)
			04 Reserved
		CNCLALL	02 CANCEL ALL request
		NOPAGING	01 No page faults allowed (system tasks)

Figure 235 (Part 1 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
8-11	8-B	TCBTIB	TIB pointer
12-15	C-F	TCBSAVE	Address of current save area
16-19	10-13	INTINFO	Saved interrupt information
19	13	SVCIC	SVC interruption code
20-23	14-17	AERREXIT	Address of cancel exit (used for system tasks only)
24-27	18-1B	TCBERBLK	Address of head queue error entry (system tasks only)
28-31	1C-1F	TCBSAV2	Address of second save area
.... End of TCB for system tasks without second save area.			
24-103	18-67	TCBSSAVE	Second save area
104-251	68-FB		TCB work area (FETCH, CCW Translation, CCW fixing, SVC)
252-255	FC-FF	TCBCINF	Fetch cancel information
		TCBCALIB	Pointer to library name
256-259	100-103	TCBCASLB	Pointer to sublibrary name
260-263	104-107	TCBCANAM	Pointer to phase name
264-267	108-10B		Used for move mode
(268)	(10C)	TCBWLEN	Total length of TCB work area
.... End of TCB for system tasks without exits............			
268-299	10C-12B	TCBEXTAB	AB, IT, PC exit information
268	10C		AB exit flag byte
		EXITACT	X'80' AB exit routine active
268-271	10C-10F	TCBABPTR	Address of AB exit routine
272-275	110-113	TCBABSAV	Address of AB exit save area
276-283	114-11B	TCBABSEC	Address and save area of secondary AB exit
284	11C		PC exit flag byte
		EXITACT	X'80' PC exit routine active
284-287	11C-11F	TCBPCPTR	Address of PC exit routine
288-291	120-123	TCBPCSAV	Address of PC exit save area
292	124		IT exit flag byte
		EXITACT	X'80' IT exit routine active
		DELINT	40 IT interrupt processing delayed
292-295	124-127	TCBITPTR	Address of IT exit routine
296-299	128-12B	TCBITSAV	Address of IT exit save area
300-303	12C-12F	TCBEOTAD	Continuation address for End of Task clean-up

Figure 235 (Part 2 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
304	130	VTAMBGIN	AR TCB: Begin address of VTAM partition (set by VTAM)
		APSFLAG	Flag byte
			X'80' Reserved
			40 Reserved
		VTLTDLY	20 VTAM user exit delayed while task owns the LTA
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
305-307	131-133		User task TCB: Pointer to VTAM APT (set/used by VTAM)
308-311	134-137	VTAMEND	AR TCB: End address of VTAM partition (set by VTAM)
308	134	APSCNT	Count of open VTAM ACB's (maintained by VTAM)
309-311	135-137		Reserved for VTAM
312-315	138-13B	VTPCINF	Program check information (VTAM)
316	13C	VTAMFLG	Flag byte
		VTABEND	X'80' AR TCB: TPBAL issued (set by VTAM)
			User task TCB: Abnormal term. of a VTAM process
		VTSPSAV	40 PSW + registers in SVPCSAVE
		VTCDLY	20 Cancel delayed for VTAM
		VTAPDEL	10 VTAM AP exit delayed while terminator is active
		VTURX	08 VTAM user exit in control
		VTSVC	04 VTAM SVC active
		VTAPP	02 VTAM process active
		VTAMK0	01 Key 0 / supervisor state required for VTAM
317-319	13D-13F		Reserved
320-323	140-143	TCBECB	Address of ATTACH ECB (used only for user subtasks)
324-325	144-145	TCBSPOFF	Identification of dedicated GETVIS subpool
326-327	146-147		Reserved
328-331	148-14B	TCBCRCBC	Anchor of CRCB chain (XPCC exit)
(332)	(14C)	TCBLNG	Length of AR and main task TCB
.... End of user main task TCB's			

Figure 235 (Part 3 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
332-335	14C-14F	TCBSTADR	Address of system task deactivation routine
.... End of TCB's for system tasks with exits			
336-455	150-1C7	TCBUSAVE	Subtask save area in case of ATTACH without SAVE
(446)	(1C8)	STCBLNG	Length of subtask's TCB
.... End of subtask's TCB			

Figure 235 (Part 4 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
.... FETCH work area			
104-143	68-8F	DFCBSAV	Save area (Registers 0-2,8-14)
144-147	90-93	DFWKLPNT	Phase load point
148-151	94-97	DFWKEPNT	Phase entry point
152-155	98-9B	DFWKUSEN	Pointer to user's directory entry
156	9C	DFCBSW1	Flag byte
		FIRSTDIR	X'80' First directory record
		INVUSEN	40 Invalid local list
		FLABMASK	20
		USERMASK	10 User task
		NODEVALD	08 No validation required
		PARTLOAD	04 Load into partition
		REALMASK	02 Request for real partition
		SYSAMASK	01 System task request
157	9D	DFCBSW2	Flag byte
		FIXPAG	X'80' Pageable FETCH part is fixed
		USERDUPD	40 Update user directory entry
		GENINT	20 CCW generation area exhausted
		FIXTXT	10 Area for phase read in is fixed
		ACTDIR	08 Directory entry active
		LASTTBL	04 Last text block is read in
		IDERR	02 ID mismatch during dir. read
		FLNKVIO	01 LNKEDT with option LINK

Figure 235 (Part 5 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
158	9E	DFWKRCOD	Return code
159	9F		Reserved
160-163	A0-A3	DFWKPHPT	Address of phase name
164	A4	DFWKFLAG	Option byte
		FLRETCOD	X'80' Return code required
		SVAUPD	40 Load/update SVA phase
		SDLUPD	20 Update SDL
			10 Reserved
		SDLFORM	08 Directory entry has SDL format
		SYSLIST	04 Search SYSLIB first
		DENTRY	02 Directory entry option
		NTXTNTRY	01 No text load option
165-167	A5-A7	DFWKLIST	Pointer to local list
168-175	A8-AF	DFWKNAME	Phase name
176	B0	DFWKEGEN	Reserved
177	B1		Phase attributes
178-179	B2-B3		Offset of PRBA-ADDR
180-183	B4-B7	DFWKERBA	Relative block address
184-185	B8-B9	DFWKECON	Number of contiguous blocks
186-187	BA-BB		Reserved
188	BC	DFWKESWT	Indicators
189	BD	DFWKEMVS	Status MOVE-MODE
190-191	BE-BF		Reserved
192-195	C0-C3	DFWKEPLN	Length of phase in bytes
196-199	C4-C7	DFWKELPL	Load point at LNKEDT time
200-203	C8-CB	DFWKEEPL	Entry point at LNKEDT time
204-207	CC-CF	DFWKEBGP	Part. start address at LNKEDT time
208-209	D0-D1	DFWKERLD	Number of RLD items
210-215	D2-D7	DFWKERDA	PRBA of RLD item
216-217	D8-D9		Reserved
218-223	DA-DF		Reserved
224-227	E0-E3	DFWKEVLE	Entry point in SVA
228-231	E4-E7	DFWKLBID	Librarian identifier of phase
232-235	E8-EB	DFWKALIB	Address of Library Definition Table
236-239	EC-EF	DFWKASLB	Addr. of Sublibrary Definition Table
(240)	(F0)	DFWKEND	End of DE lay-out
240-243	F0-F3	DFWKANAM	Address phase name
244-247	F4-F7	DFWKCOMG	Pointer to actual COMREG
248-251	F8-FB	ANCSAV	Pointer to Anchor table
(148)	(94)	LFCHWORK	Length of fetch work area
.................End of FETCH work area			

Figure 235 (Part 6 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
.... CCW Translation work area (/370 mode)			
104	68	TCBFLAG	Flag byte
		TCBDC	X'80' Data chaining
		TCBRDS	40 Read/Sense command
		TCBRDB	20 Read Backward command
		TCBSM1	10 Status modifier command and command chaining
		TCBSM2	08 Status modifier command
		FXGETBL	04 Request for FIXINF block
		CHKSTM	02 Check status modifier 1287/3890
		GETDCBL	01 Double copy block request
105	69	ADBTAMCB	No. of addit. blocks needed by BTAM
106	6A		Reserved
107	6B	CCWTFLG2	CCW-Translation second flag byte
		CCWTFIDA	X'80' Fix IDAL request
		CCWTPAF	40 Page already fixed
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
108-111	6C-6F	DEVSTPTR	Pointer to status modifier list
112-115	70-73	DEVCDPTR	Pointer to control command list
116-119	74-77	LINEPTR	Pointer to next line
120-123	78-7B	BENDPTR	Block end pointer
124-127	7C-7F	TCBACCB	Address of copied CCB
128-131	80-83	IDALCNT	Number of free IDAL's
132-139	84-8B	DYNAREA1	Dynamic save area
140-147	8B-93	DYNAREA2	Dynamic save area
148-151	94-97	DYNAREA3	Dynamic save area
152-155	98-9B	DYNAREA6	Dynamic save area
136-159	9C-9F	DYNAREA7	Dynamic save area
160-163	A0-A3	FIXADDR	Address of last TFIX request
164-167	A4-A7	CCWTFREP	Address of free fix list entry
168-171	A8-AB	TCBDCB	Pointer to DIDAL block chain FASTTR
172-175	AB-AF	DDALBLAD	Address of current DIDAL block FASTTR
176	B0	TCBFLAG1	Flag byte FASTTR
		RFALG	X'80' REPLICA creation request FASTTR
			40 Reserved
		REPLCR	20 REPLICA block request FASTTR
		DIDALCR	10 Request for DIDAL block FASTTR
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved

Figure 235 (Part 7 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
177	B1		Reserved FASTTR
178-179	B2-B3	DIDALCNT	No. of free DIDAL double words FASTTR
180-183	B4-B7	DIDAWAD	Address of current DIDAL FASTTR double word FASTTR
184-187	B8-BB	VCCWAD1	Stored virtual CCW address FASTTR
188-215	BC-D7	SAVEREG2	Save area for registers 2-8
216-219	D8-DB	SAVEREG9	Save area for register 9
220-223	DC-DF	SAVEREGA	Save area for register 10
224-227	E0-E3	SAVEREGB	Save area for register 11
228-231	E4-E7	SAVEREGC	Save area for register 12
232-235	E8-EB	SAVEREGD	Save area for register 13
236-239	EC-EF	SAVEREGD	Save area for register 14
240-243	F0-F3	SAVEREGD	Save area for register 15
(140)	(8C)	LCCWTAR	Length of CCW Translation work area
.... End of CCW Translation work area			

Figure 235 (Part 8 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
.... CCW Fixing work area (ECPS:VSE mode)			
104	68	FRBFLAG1	Flag byte
		FRBDC	X'80' Data chaining specified
		FRBRDS	40 Read/Sense command
		FRBRDB	20 Read Backward command
		FRBSM1	10 Status modifier command and data chaining
		FRBSM2	08 Status modifier command and command chaining
		FRBSM3	04 Status modifier handling in process
			02 Reserved
		FRBDOIO	01 DOIO request
105	69	FRBFLAG2	Flag byte for FASTTR
		FRBRRQ	X'80' REPLICA creation required
		FRBCFL	40 Chained fixlist
		FRBVAL	20 Valid fixlist entry
		FRBRCS	10 Replica creation suppressed
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved

Figure 235 (Part 9 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
106-107	6A-6B		Reserved
108-111	6C-6F	FRBSFADR	Address of SETFLAG routine
112-115	70-73	FRBSMPTR	Address of status modifier list
116-119	74-77	FRBCDPTR	Address of control command list
120-123	78-7B	FRBAFHB	Address of FHB (fixlist)
124-127	7C-7F	FRBLNPTR	Address of LINEPTR stack
128-131	80-83	FRBLLPTR	Address of Locate list
132-139	84-8B	FRBSAVLE	Save field for locate list entry
132-135	84-87	FRBALLE	Actual Locate list entry
136-139	88-8B	FRBWRK1	Work field 1
140-143	8C-8F	FRBWRK2	Work field 2
144-147	90-93	FRBWRK3	Work field 3
148-151	94-97	FRBWRK4	Work field 4
152-155	98-9B	FRBWRK5	Work field 5
156-159	9C-9F	FRBSAV	Save area for register 15
160-163	A0-A3	FRBSAV0	Save area for register 0
164-167	A3-A7	FRBSAV1	Save area for register 1
168-199	A8-C7	FRBSAV2	Save area for registers 2-9
200-203	C8-CB	FRBSAVA	Save area for register 10
204-211	CC-D3	FRBSAVB	Save area for registers 11-12
212-219	D4-DB	FRBSAVD	Save area for registers 13-14
(220)	(DC)	FRBEND	End of FRB
(116)	(74)	LCCWFAR	Length of CCW Fixing work area
. ... End of CCW Fixing area			

Figure 235 (Part 10 of 11). Task Control Block (TCB)

Bytes			
Dec	Hex	Label	Description
.... SVC work area			
104-167	68-A7	SVCSV3	Save area for registers 9-8
168-215	A8-D7	SVCWORK	Work area
(112)	(70)	LSVCWORK	Length of SVC work area
.... End of SVC work area			

Figure 235 (Part 11 of 11). Task Control Block (TCB)

LAYOUT OF PSW

```
 ________________________________________________________________________
|        I |    |     |    |     |        |        |                    |
|0R000T/E  | KEY| IMWP|00CC|PROG |00000000|00000000|INSTRUCTION ADDRESS |
|        0 |    |     |    |MASK |        |        |                    |
|__________|____|_____|____|_____|________|________|____________________|
BIT
 01234567   8    12    16   20    24       32       40                 63
```

Bits	Apprev.	Description
0		Always zero
1	R	Program Event Recording Mask
2 - 4		Reserved (must be zero)
5	T	Translation Mode or Zero in ECPS:VSE Mode)
6	I/O	I/O interrupt mask
7	E	External interrupt mask
8 - 11		CPU protection key
12	I	Always one (EC mode)
13	M	Machine Check mask
14	W	Wait state
15	P	Problem Program State
16 - 17		Reserved (must be zero)
18 - 19	CC	Condition code
20		Fixed-point overflow mask
21		Decimal overflow mask
22		Exponent overflow mask
23		Significance mask
24 - 31		Reserved (must be zero)
32 - 39		Reserved (must be zero)
40 - 63		Instruction address

Figure 236. Program Status Word (PSW)

SAVE AREAS

Problem Program (PP) Save Area
User Supplied Save Area (STXIT)
LTA Save Area
System Save Area
Logical Transient Area Occupancy and Activity

The addresses of the various Save Areas allocated by the System can be found in the appropriate TCB table. The layout of the different Save Areas is shown in Figure 237 through Figure 241 on page 521.

Problem Program (PP) Save Area

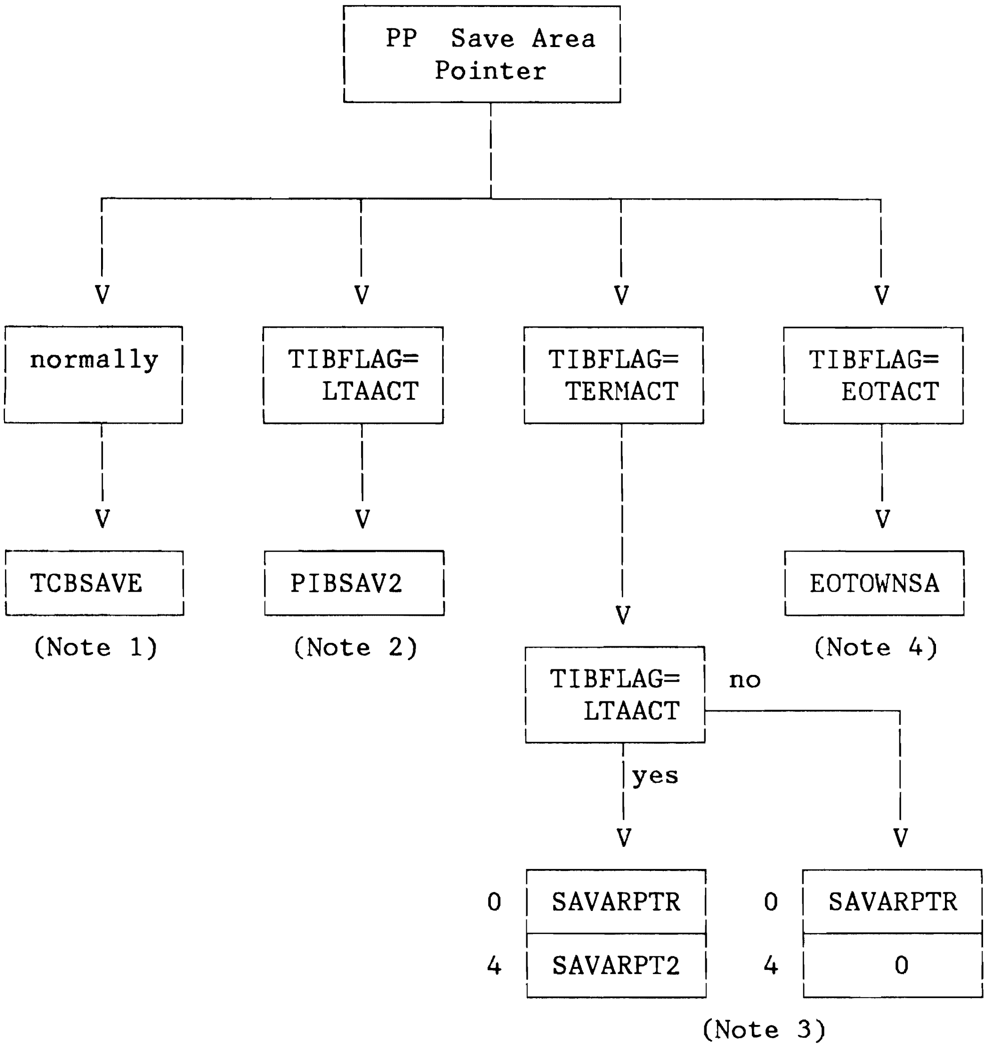
```
┌────────────────────────────────────────────────────────────────────┐
│                            Program Name                            │
│0 (0)                                                          7 (7)│
├────────────────────────────────────────────────────────────────────┤
│                   Program Status Information                       │
│                            (Note 1)                                │
│8 (8)      │9 (9)       │10 (A)     │11 (B)     12 (C)│13 (D)    15 (F)│
├───────────┼────────────┼───────────┼─────────────────┼───────────────┤
│X'40' PER  │Protection  │ (Note 2)  │ Zero            │Instruction Address│
│X'04' DAT  │Key and     │           │                 │               │
│X'02' I/O  │Mask (CMWP) │           │                 │               │
│X'01' EXT  │Bits        │           │                 │               │
├───────────┴────────────┴───────────┴─────────────────┴───────────────┤
│16 (10)   General Register save area (Reg. 9 through Reg. 8)   79 (4F)│
├────────────────────────────────────────┬───────────────────────────┤
│80 (50) Reserved  81 (51)│82 (52)        (Note 3)              87 (57)│
├────────────────────────────────────────┴───────────────────────────┤
│88 (58)  Floating Point Reg. save a. (Reg. 0 through Reg. 6)  119 (77)│
└────────────────────────────────────────────────────────────────────┘
```

Notes:

1. EC Mode PSW see Figure 236 on page 516
2. Byte 10
   - bits 0-1 = Reserved (zero)
   - bits 2-3 = Condition Code
   - bits 4-7 = Program Mask
3. Bytes 82 - 87
   - main task: Date of job begin
   - subtask: 82 (52) - 83 (53) : Reserved
   - 84 (54) - 85 (55) : Task id
   - 86 (56) : Key of ICCF pseudo-partition
   - 87 (57) : Reserved

Figure 237. Problem Program Save Area

Where to Find the PP Save Area Pointer in Case of Termination



Notes:

1. Located in Task Control Block (TCB).
2. Located in Partition Information Block (PIB).
3. Identified via "eye catcher" 'CNCLINFO' in the supervisor. If TIBFLAG=LTAACT and TERMACT, the LTA save area pointer will be found in SAVARPTR, the PP save area ptr. in SAVARPT2. Otherwise (TERMACT) the PP save area ptr. will be found in SAVARPTR.
4. Identified via "eye catcher" 'EOT SAVE' in the supervisor.

Figure 238. Problem Program (PP) Save Area Pointer in Case of Termination

User Supplied Save Area (STXIT)

<table>
<tr><td colspan="5">Interrupt Status Information</td></tr>
<tr><td>0 (0)</td><td>1 (1)</td><td>2 (2) 3 (3)</td><td>4 (4)</td><td>5 (5) 7 (7)</td></tr>
<tr><td>Reserved</td><td>Protection Key and Mask bits from PSW byte 1</td><td>Interruption Code</td><td>(Note 1)</td><td>Instruction Address</td></tr>
<tr><td colspan="5">8 (8) General Register save area (Reg. 0 through Reg. 15) 71 (47)</td></tr>
</table>

Notes:

1. Byte 4
   - bits 0-1 = Instruction Length Code
   - bits 2-3 = Condition Code
   - bits 4-7 = Program Mask

Figure 239. Format of the User's Save Area for AB, PC, OC, IT, and TT Routines

The address of the save area specified by the user in the STXIT macro parameter is stored in the appropriate table (TCB, PCB or TTTAB).

LTA Save Area

Logical Transient Phase name 0 (0) 7 (7)				
Program Status Information (Note 1)				
8 (8)	9 (9)	10 (A)	11 (B) 12 (C)	13 (D) 15 (F)
X'40' PER X'04' DAT X'02' I/O X'01' EXT	Protection Key and Mask (CMWP) Bits	(Note 2)	Zero	Instruction Address
16 (10) General Register save area (Reg. 9 through Reg. 8) 79 (4F)				
80 (50) Reserved 87 (57)				
88 (58) Floating Point Reg. save a. (Reg. 0 through Reg. 6) 119 (77)				

Notes:

1. EC Mode PSW see Figure 236 on page 516
2. Byte 10
   - bits 0-1 = Reserved (zero)
   - bits 2-3 = Condition Code
   - bits 4-7 = Program Mask

Figure 240. LTA Save Area

Program Status Information (Note 1)				
0 (0)	1(1)	2 (2)	3 (3) 4 (4)	5 (5) 7 (7)
X'40' PER X'04' DAT X'02' I/O X'01' EXT	Protection Key and Mask (CMWP) Bits	(Note 2)	Zero	Instruction Address
8(8) General Register save area (Reg. 9 through Reg. 8) 71 (47)				

Notes:

1. EC Mode PSW see Figure 236 on page 516
2. Byte 2
   - bits 0-1 = Reserved (zero)
   - bits 2-3 = Condition Code
   - bits 4-7 = Program Mask

Figure 241. System Save Area

Logical Transient Area Occupancy and Activity

Indications of Logical Transient Area Occupancy and Activity

Status	BGCOMREG	Attention PIB	Problem PIB	Condition of LTA	Notes
SVCs issued	Contents of LTK + 1 (1 Byte)	Address in ARFLG + 1 (3 Bytes)	Addr. in PIBSAVE+1 (3 Bytes)		
	zero	Logical Transient Save Area (LTASAVE)		Free	Initial condition before issuing SVC-02
SVC-02	Owner's Partition Identific. Key	Problem Program Save Area	Logical Transient Save Area (LTASAVE)	Active	
SVC-02 SVC-0B	zero	Logical Transient Save Area (LTASAVE)	Problem Program Save Area	Free	Restored to (1)
SVC-02 SVC-08	Owner's Partition Identific. Key	Logical Transient Save Area (LTASAVE)	Problem Program Save Area	Occupied but Inactive	SVC-08 may be issued only from LTA. General register 14 contains address of entry point to the user routine
SVC-02 SVC-08 SVC-09	Owner's Partition Identific. Key	Problem Program Save Area	Logical Transient Save Area (LTASAVE)	Active	Restored to (2). SVC-09 may be issued only from Problem Program.
SVC-02 SVC-08 SVC-09 SVC-0B	zero	Logical Transient Save Area (LTASAVE)		Free	Restored to (1)

Figure 242. Indications of Logical Transient Area Occupancy and Activity

JOB ACCOUNTING TABLES (ACCTCOMN, ACCTABLE)

Job Accounting Common Table (ACCTCOMN)
Job Accounting Partition Table (ACCTABLE)

Bytes 124-127 (X'7C' - X'7F') of the System Communication Region (SYSCOM) contain the address of the Job Accounting interface common table. Label ACCTCOMN identifies the first byte of the table.

Bytes Dec	 Hex	Label	Description
0–3	0–3	ACCTPCNT	Count of active partitions
4	4	ACCTSWCH	Job control switches
		ACCTCTSW	X'20' Catal switch
5–7	5–7		Reserved
8–11	8–B	ACCTABLN	Length of JA partition table
12–15	C–F	ACCTUSEP	Address of JA user save area
16–19	10–13	ACCTUSEL	Length of JA user save area

Figure 243. Job Accounting Common Table (ACCTCOMN)

Bytes Dec	 Hex	Label	Description
0	0	ACCTSWTC	Accounting partition switches
		ACCTACTV	X'80' Indicate JCL–N/$JOBACCT active
1–7	1–7		Reserved
8–11	8–B	ACCTSVPT	Address of job card field following job name
12–13	C–D	ACCTNSIO	Current number of SIO count fields
14–15	E–F	ACCTLEN	Length of SIO area = 6n+1, where n is the number of devices accessed by the job step
16–23	10–17	ACCTCLCK	Time field in seconds
24–27	18–1B	ACCTLADD	Address of label area
28–31	1C–1F	ACCTCPUT	Partition CPU time counter for current step
32–35	20–23	ACCTOVHD	Overhead time counter for current job step (distributed in proportion to CPU time)

Note: Bytes 116–119 (X'74'–X'77') of the Partition Communication Region (COMREG) contain the address of the ACCTABLE.

Figure 244 (Part 1 of 2). Job Accounting Partition Table (ACCTABLE)

Bytes			
Dec	Hex	Label	Description
36-39	24-27	ACCTBNDT	System wait time for current job step (distributed in equal parts to activate partition) (note)
40-47	28-2F	ACCTSVJN	Save area for job name during simulated EOJ
.... Following information passed to the user			
48-55	30-37	ACCTJBNM	Job name from job card
56-71	38-47	ACCTUSRS	User information from job card
72-73	48-49	ACCTPTID	Partition ID
74	4A	ACCTCNCL	Cancel code for job step
75	4B	ACCTYPER	Type of record: 'L'=last job step, else 'S'
76-83	4C-53	ACCTDATE	Date of end of job step in the format MM/DD/YY or DD/MM/YY, depending on the DATE standard option
84-87	54-57	ACCTSTRT	Stop time of previous job step, in packed decimal
88-91	57-5B	ACCTSTOP	Stop time of job step, in packed decimal
92-95	5C-5F	ACCTDUR	Step duration time in seconds, in binary
96-103	60-67	ACCTEXEC	Phase name taken for EXEC statement
104-107	68-6B	ACCTHICR	length of page * number of partition pages referenced (or PFIXed for real execution) in the current job step. For MODE=VM, this field contains the highest virtual storage address allocated to this partition
108-111	6C-6F	ACCTIMES	Same as ACCTCPUT at the end of the job step
112-115	70-73		Same as ACCTOVHD at the end of the job step
116-119	74-77		Same as ACCTBNDT at the end of the job step
120	78	ACCTSIOS	Six bytes for each device accessed by the job step, as follows: 2 bytes for device addr.(0cuu) 4 byte for SIO count in current job step
last byte			Overflow byte: always X'20', indicating no overflow

Figure 244 (Part 2 of 2). Job Accounting Partition Table (ACCTABLE)

EVENT CONTROL BLOCK (ECB)

```
┌──────────┬──────────┬──────────┬──────────┐
│          │          │          │          │
│ Reserved │ Reserved │    •     │ Reserved │
│          │          │    |     │          │
└──────────┴──────────┴────┼─────┴──────────┘
     0          1          2      3
                           |
                           V
                         X'80'   Event completed or terminated normally
                         X'C0'   Abnormal termination
```

Figure 245. Event Control Block (ECB)

AB, IT, OC, PC EXIT ROUTINE ENTRY

FLAG BYTE	EXIT ROUTINE ADDRESS	PSW KEY	SAVE AREA ADDRESS
0	1 3	4	5 7

Bytes	Description
0	Flag byte
	X'80' User EXIT routine already active
	40 Reserved
	20 Reserved
	10 Reserved
	08 Reserved
	04 Reserved
	02 Reserved
	01 Reserved
1 - 3	User's STXIT AB, IT, OC, PC routine address
	Zero if STXIT not yet issued
4	PSW key of user
5 - 7	Address of users save area
	Zero if STXIT not yet issued

Figure 246. AB, IT, OC, PC Exit Routine Entry

FETCH CONTROL BLOCKS (DSRCHNX, FRPL)

DSRCHNx searching chain control block
Fetch request block (FRPL)

Layout of a FETCH-CHAIN Entry

Bytes			
Dec	Hex	Label	Description
0	0	DSRCHID0	Identification
		ENDENT	X'FF' End identification
		NORMENT	X'00' Normal entry
1	1	DSRCHID1	Status flag 1
		SDLID	X'80' SDL search
		DIRID	X'10' Directory search
		LINKID	X'0C' Link search
2	2	DSRCHID2	Status flag 2
		PCILID	X'08' Phase not found in SYSLIB
		SCILID	X'00' Phase found in SYSLIB
3	3	DSRCHID3	Status flag 3
		SCDLLNK	X'0C' Search on Link-VIO
		SCDLJOB	X'08' Search on permanent job chain
		SCDLPRT	X'04' Search on temporary partition chain
		FSTLCHN	X'00' Search on SDL or SYSLIB
4	4		Total Length

Figure 247. Layout of the DSRCHNx Entry

Layout of the FRPL

The FRPL describes the interface between the logical level of the FETCH processing and the I/O level. It is provided by the FETCH I/O layer and must initialized before any read request can be performed. Its layout is as follows:

Bytes			
Dec	Hex	Label	Description
0 - 1	0 - 1	DRPLID	Identification
		DIRRQID	X'000C' Directory read
		RLDRQID	X'0008' RLDreaD
		TXTRQID	X'0004' TXTread
2 - 3	2 - 3	DRPOCCW	Offset of CCW program
4 - 5	4 - 5	DRPPHBL	Physical block-length
6 - 7	6 - 7	DRPLGRL	Logical record length
8 -11	8 - B	DRPLADA	Addr. of disk-addr area
12 -15	C - F	DRPLINP	Addr. of input area
16 -19	10 -13	DRPLCIF	Addr. of LBCIF
20 -21	14 -15	DRPLNRC	Number of req. records
22 -23	16 -17	DRPLOPC	Operation field
		NCONTTXT	X'0080' No contiguous TXT
24	18	DRPLFLG	Flags
25	19	DRPLCMD	Read op-code
26 -27	1A -1B	DRPLLRC	TXT-length in last TXT-LB
28 -31	1C -1F	DRPLAGM	Addr. of CCW-generation model
32 -35	20 -23	DRPLTIC	Addr. of related TIC
36 -37	24 -25	DRPNIDAW	Number of IDAWs (370 only)
38 -39	26 -27	DRPLIDAL	Length of IDAW list (370 only)
40	28		Total Length

Figure 248. Layout of the FRPL

LOCK MANAGEMENT AREAS (DTLADR, LOCKADR, LOKOADR, DLFADR)

Define the Lock (DTLADR)
LOCKTAB Entry (LOCKADR)
Owner Element (LOKOADR)
DASD Sharing Dsect (DLFADR)

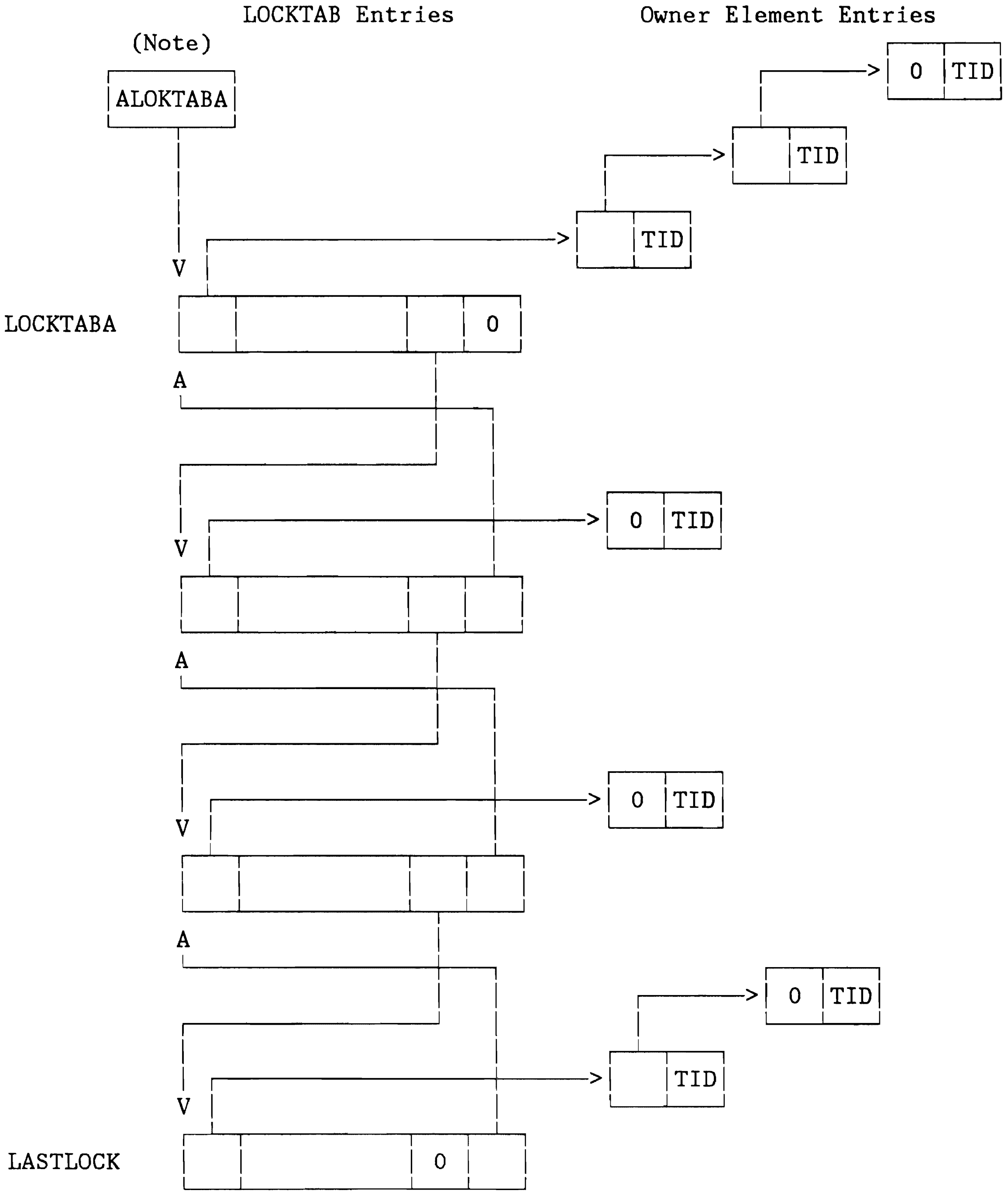
Define the Lock (DTLADR)

On entry to SVC 110 (X'6E') register 1 contains the address of the DTL. This control block describes a resource to be locked/unlocked with SVC 110 (X'6E').

Bytes			
Dec	Hex	Label	Description
0 - 1	0 - 1	DTLLENF	Length of DTL
2	2	DTLFLG1	Flag Byte 1:
			CONTROL option
			X'80' Reserved
			X'40' Reserved
			X'20' Reserved
		DTLEXC	X'10' CONTROL=E(xclusive)
			LOCKOPT option
			X'08' Reserved
		DTLOPT4	X'04' LOCKOPT=4
		DTLOPT2	X'02' LOCKOPT=2
		DTLOPT1	X'01' LOCKOPT=1
3	3	DTLFLG2	Flag Byte 2:
		DTLKEEP	X'80' KEEP=YES
		DTLPART	X'40' OWNER=PARTITION
		DTLREDC	X'20' CHANGE=ON
		DTLEXTR	X'10' SCOPE=EXT
		DTLVOL	X'08' VOLID specified
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
4 - 15	4 - F	DTLNAME	Resource Name
16 - 21	10 - 15	DTLVOLID	Volume Identification
22	16	DTLLEN	Length of DTL

Figure 249. Define the Lock (DTL)

LOCKTAB Entry (LOCKADR) and Owner Element (LOKOADR)



Note: Identified via eye catcher 'LOCKSP' + 8 in pageable part of the supervisor.

Figure 250. Relationship Between LOCKTAB and Owner Elements

LOCKTAB Entry (LOCKADR)

A LOCKTAB entry contains a chain pointer to owner elements, a resource name two flag bytes, an exclusive usage counter, a forward and backward chain pointer to the next resp. foregoing LOCKTAB entry.

The layout is shown below:

Bytes			
Dec	Hex	Label	Description
0 - 3	0 - 3	LOCKCHN	Chain pointer to Owner elements
4 - 15	4 - F	LOCKRESN	Resource Name
16	10	LOCKFLG1	Flag Byte 1:
			CONTROL option
			X'80' Reserved
			X'40' Reserved
			X'20' Reserved
		LOCKEXC	X'10' CONTROL=E(xclusive)
			LOCKOPT option
			X'08' Reserved
			X'04' LOCKOPT=4
			X'02' LOCKOPT=2
			X'01' LOCKOPT=1
17	11	LOCKFLG2	Flag Byte 2:
		LOCKUSED	X'80' LOCKTAB entry in use
		LOCKPART	X'40' LOCK owned by partition
		LOCKWAIT	X'20' Task waits for resource
		LOCKEXT	X'10' Cross system lock
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
18 - 19	12 - 13	LOCKCNTE	Number of exclusive users
20 - 23	14 - 17		Reserved
24 - 27	18 - 1B	LOCKPTR	Forward chain pointer
28 - 31	1C - 1F	LOCKBPTR	Backward chain pointer
32	20	LOCKLEN	Length of LOCKTAB Entry

Figure 251. LOCKTAB Entry

Owner Element (LOKOADR)

An owner element contains a forward pointer to the next owner element, two flag bytes, usage counters, and the task identifier (TID) of the owning task.

The element's layout is shown below:

Bytes			
Dec	Hex	Label	Description
0 - 3	0 - 3	LOKOCHN	Chain pointer to next Owner Element
4 - 5	4 - 5	LOKOTID	Task Identification of owning task
6 - 7	6 - 7	LOKOCNTS	Number of shared users
8 - 9	8 - 9	LOKOCNTE	Number of exclusive users
10	A	LOKOFLG	Flag Byte:
		LOKOKEEP	X'80' Keep until end of job
			X'40' Reserved
			X'20' Reserved
		LOKOEXC	X'10' Exclusive usage
11	B	LOKOFLG2	Flag Byte 2
12 - 15	C - F		Reserved
16	10	LOKOLEN	Length of Owner Element

Figure 252. Owner Element

If an owner element is freed, it will be put in front of the free-list.

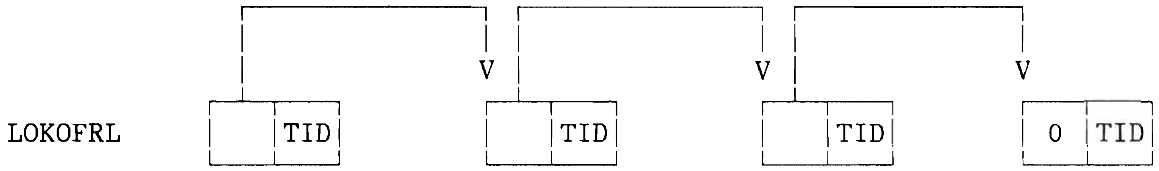


Figure 253. Free-list of Owner Elements

DASD Sharing Dsect

Bytes Dec	Hex	Label	Description
.... First 20 bytes of lock file header record			
0 - 1	0 - 1	DLFCHAR	Block identification
2 - 3	2 - 3	DLFNCPUS	Number of sharing CPU's
4 - 5	4 - 5	DLFLBLK	Physical block length
6 - 7	6 - 7	DLFNBLK	No. of physical blocks in data area
8 - 9	8 - 9	DLFNENT	No. of entries per block
10 - 11	A - B	DLFLENT	Length of one lock entry (12+NCPU)
12 - 13	C - D	DLFBLKLL	Lower limit on FBA
		DLFCYL	Cyl. address of external file (CKD)
14 - 15	E - F	DLFREC#	Number of blocks per track (CKD)
16 - 17	10 - 11	DLFTRCK#	Number of tracks per cylinder (CKD)
18	12	DLFDEVT	Flag - device type
		DLFRPS	X'03' External file on RPS CKD
		DLFCKD	02 External file on CKD
		DLFFBA	01 External file on FBA
19	13	DLFDEVC	Device code
.... Start of 8 byte CPU field			
20	14	DLFCPUS	Start of 8 byte CPU field
		DLFCPUF1	Flag byte 1 in CPU entry
		DLFCPUUS	X'80' CPU field in use
20 - 21	14 - 15	DLFUNT	Channel and unit of external file
22 - 23	16 - 17	DLFPUB	PUB index (for physical addressing)
24	18	DLFFLG1	Flag - byte 1
		DLFINT	X'80' DSHRINIT processed successful
		DSHRDOWN	40 DASD sharing support down (I/O error)
		DLFACT	20 DASD sharing support is active
		DSDWNMSG	10 DASD-SHR-DOWN message to be displayed
		DLFCHAIN	08 write chained to device release
25	19	DLFFLG2	Flag - byte 2 (reserved)
26 - 27	1A - 1B	DLFINDEX	Number of this CPU (0 until NCPU-1)
.... End of IPL DLF table			
28	1C	DLFLENI	Length of DLF table (for IPL)
....			
28 - 31	1C - 1F	DLFAREA	I/O area for external file
32 - 33	20 - 21	DLFHBLK	Actual block in lock file (hash no.)
34	22	DLFLEN	Length of DLF table (full length)

Figure 254. DASD Sharing Dsect (DLFADR)

PAGE MANAGEMENT COMMUNICATION AREA (PMCOM)

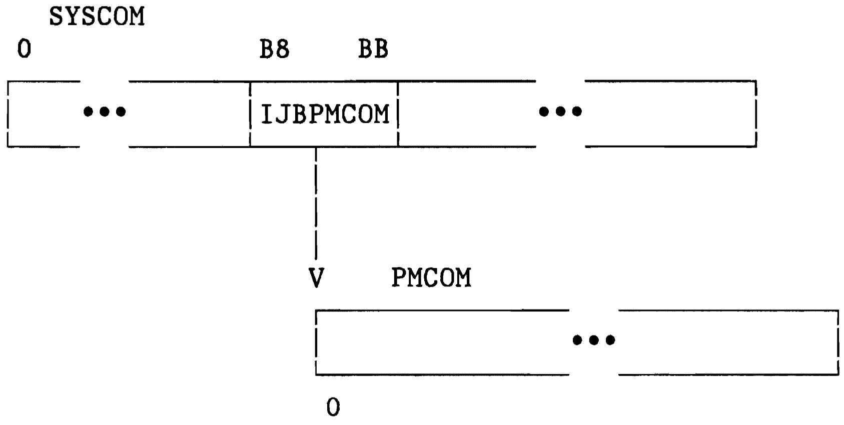


Figure 255. PMCOM Relationship

Bytes			
Dec	Hex	Label	Description
0 - 3	0 - 3	PMPGSIZE	Page size in bytes
4 - 7	4 - 7	PMPAGMSK	Pattern for page boundary
8 - 11	8 - B	PMDISMSK	Pattern for displacem.in page
12 - 15	C - F	PMPTEMSK	Pattern for page number in PTE
16 - 19	10 - 13	PMPNRMSK	Pattern for page number in PFTE
20 - 21	14 - 15	PMADPN	Shift amount addr. to page number
22 - 23	16 - 17	PMADPFTO	Shift amount addr. to PFT offset
24 - 25	18 - 19	PMADPTO	Shift amount addr. to PT offset
.... End of PMCOM for VM Mode			
26 - 27	1A - 1B	DEVCBNUM	Number of device control blocks
28 - 31	1C - 1F	PSQPTR	A(page selection queue header)
32 - 35	20 - 23	ARTAB	A(reentry-rate table)
36 - 39	24 - 27	ARTABX	A(reentry-rate table)
40 - 43	28 - 2B	LRTAB	Length of reentry-rate table
44 - 47	2C - 2F	ADEVCB	A(paging device control blocks)
48 - 51	30 - 33	PMMAXEPA	Max. extended page addr. from vsize
52 - 53	34 - 35	MINPSQEF	Min. PSQ contents for fetch TFIX
54 - 55	36 - 37	PGQOMIN	Min. avail. pseudo-TIBs for page-out
.... End of PMCOM for ECPS:VSE			
56 - 57	38 - 39	PMADSN	Shift amount addr. to segment number
58 - 59	3A - 3B	PMADSTO	Shift amount addr. to ST offset
60 - 63	3C - 3F	AAPTAS	A(A(page table allocation string))
64 - 67	40 - 43	LPTAS	Number of PTAS entries
68 - 71	44 - 47	APTR	A(page table for real partition)
72 - 75	48 - 4B	PMSGSIZE	Segment size in bytes
76 - 79	4C - 4F	PMSGMSK	Mask for segment boundary
80 - 83	50 - 53	PMSGDIS	Mask for displacement in segment
84 - 87	54 - 57	PMPTOMSK	Page table offset mask
88 - 91	58 - 5B	PMINVSTE	Prototype for invalid STE
92 - 95	5C - 5F	PMSTEMSK	Mask for segment table entry
96 - 97	60 - 61	PMINVPTE	Invalidation pattern for PTE
98 - 99	62 - 63	PMIBIT	Invalid bit
100	64	PMSTECOM	Common segment mask
101	65	PMOPFLAG	Operation flag
		PMOPIPTE	X'80' IPTE support available
.... End of PMCOM for 370 Mode			

Figure 256. Page Management Communication Area (PMCOM)

RESOURCE CONTROL BLOCK (RCB)

X'FF' or X'00'	RESERVED	X'80' or X'00'	ECB ADDRESS
0	1 3	4	5 7

Bytes	Description
0	X'FF' resource is in use X'00' resource is not in use
1 - 3	Reserved
4	X'80' Another task waiting for this resource
5 - 7	ECB address of current resource owner

Figure 257. Resource Control Block (RCB)

TASK TIMER TABLE (TTTAB)

FLAG BYTE	EXIT ROUTINE ADDRESS	PSW KEY	SAVE AREA ADDRESS	TASK TIMER INTERVAL
0	1 3	4	5 7	8 15

Bytes	Description
0	Flag byte
	X'80' User EXIT routine already active
	40 Reserved
	20 Reserved
	10 Reserved
	08 Reserved
	04 Reserved
	02 Reserved
	01 Reserved
1 - 3	User's STXIT TT routine address
	Zero if STXIT not yet issued
4	Caller's PSW key
5 - 7	Address of users save area
	Zero if STXIT not yet issued
8 -15	SETT issued:
	Interval time still left
	Bits 0-51 contain the time in microseconds
	Bits 52-63 are ignored
	No SETT issued:
	Zero or negative

Figure 258. Task Timer Table (TTTAB)

VIO CONTROL BLOCKS (VIOCM, VIOPL, VTABE, VIOTABE, BLKTBE)

VIO Communication Area	(VIOCM)
VIO Parameter List	(VIOPL)
VIO Table Entry	(VTABE)
VIO File Identification Entry	(VIOTABE)
VIO Block Table Entry	(BLKTBE)

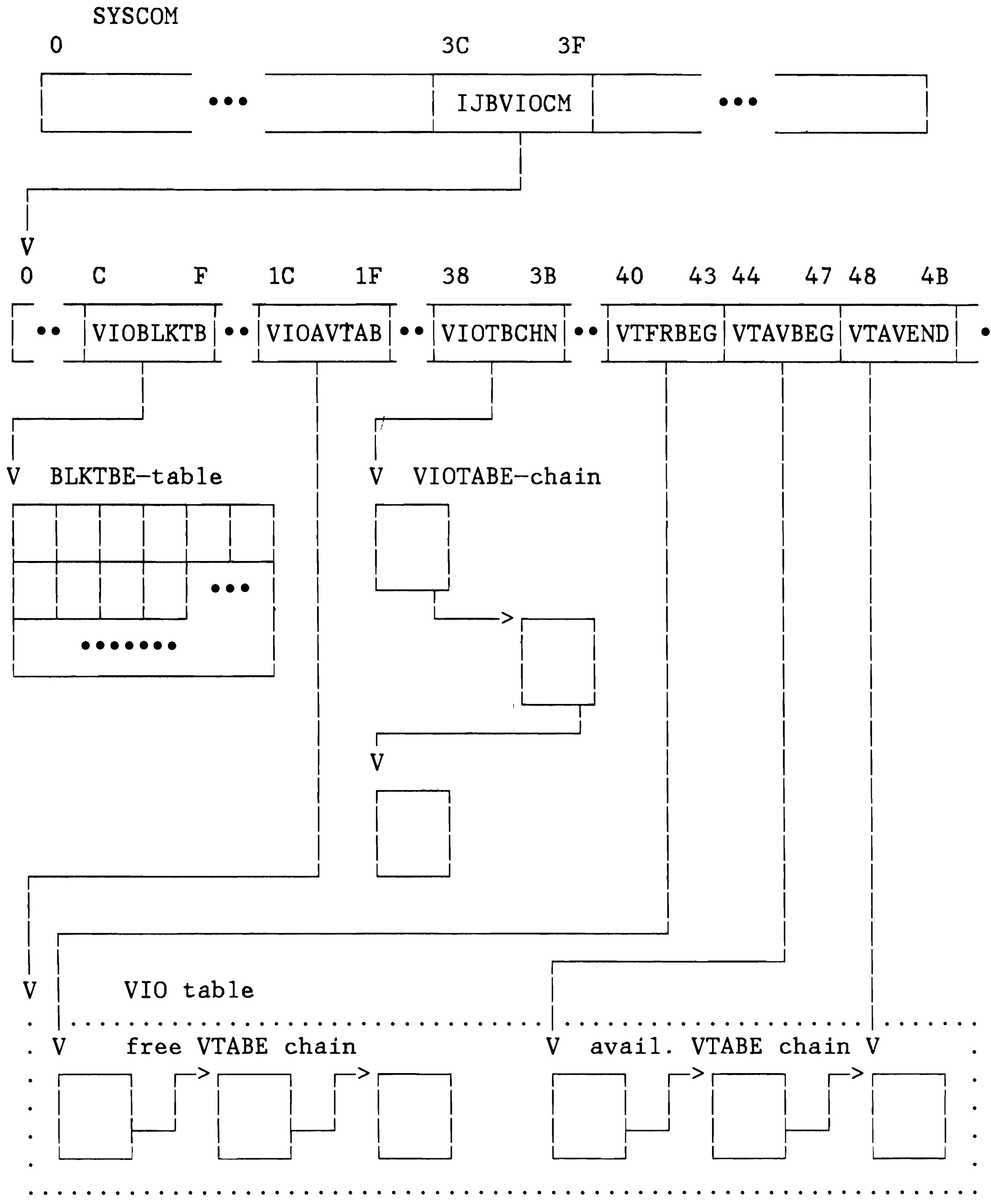


Figure 259. VIO Control Block Relationship (after IPL)

VIO Communication Area (VIOCM)

Bytes			
Dec	Hex	Label	Description
.... Temporary space used during IPL			
0-3	0-3	VIOARBEG	Begin of VIO tables (set by IPL)
4-5	4-5	VIOSGM#	Number of VIO segments (set by IPL)
6-7	6-7	VIOSPSIZ	Number of bytes allocated per VIO segment
8-9	8-9	VIOVPSIZ	Number of bytes to be allocated per page in VPOOL
10-11	A-B	VIOKSGSH	Shift value: K-bytes and segment no.
.... Normal layout of VIOCM after IPL			
0-3	0-3	VIOSPBEG	Begin of VIO allocation string
4-7	4-7	VIOSPEND	End of VIO allocation string
8-11	8-B	VIOSPNXT	Next segment slot to check
12-15	C-F	VIOBLKTB	Address of VIO Block Table
16-19	10-13	VIOVPPSZ	Size of VPOOL in pages
20-23	14-17	VIOVPNFP	Number of first VPOOL page
24-27	18-1B	VIOVPEPA	VIOVPNFP * pagesize
28-31	1C-1F	VIOAVTAB	Address of VTAB
32-35	20-23	VIOBLKSZ	Size of a VIO block
36-39	24-27	VIOSEGSZ	Size of a VIO segment
40-43	28-2B	VIOBLSFT	Page address from block number
44	2C	VIOBLDSC	OR-byte for disconnected page/frame
45-47	2D-2F		Reserved
48-55	30-37	VIOPLID	VIO Getvis subpool ID
56-59	38-3B	VIOTBCHN	VIOTAB chain header
60-63	3C-3F	VIOOPCNT	Number of VIOTAB entries
.... Header for queue of free VTAB entries			
64-67	40-43	VTFRBEG	Address of first element in chain
.... Header for queue of available VTAB entries			
68-71	44-47	VTAVBEG	Address of first element in chain
72-75	48-4B	VTAVEND	Address of last element in chain
....			
76-79	4C-4F	AVIOFBLK	Entry address of VIOFRBLK routine
80-83	50-53	AVIOFPAG	Entry address of VIOFRPAG routine
84	54		Length of VIO communication area

Figure 260. VIO Communication Area (VIOCM)

VIO Parameter List (VIOPL)

Bytes			
Dec	Hex	Label	Description
0-1	0-1	VIOPLOPT	Option bytes
0	0	VIOPLLFT	Scope option byte
		VIOPLJOB	X'10' Job
		VIOPLSTP	X'08' Step
1	1	VIOPLPRC	Processing option byte
		VIOPLASY	X'80' Asynchronous
2-3	2-3		Reserved
4-7	4-7	VIOPLRSZ	Requested size in K bytes
8	8	VIOPLLNG	Length of parameter list

Figure 261. VIO Parameter List (VIOPL)

VIO Table Entry (VTABE)

Bytes			
Dec	Hex	Label	Description
0	0	VTFLG	Flag byte
		VTPAGERR	X'80' Page in error
1	1	VTUSCNT	Usage count
2-3	2-3	VTOWNER	Partition ID of requestor
4-5	4-5	VTFRAM	Frame number belonging to page
6-7	6-7	VTPFCNT	Number of pending page-faults
8-11	8-B	VTBLKN	Total block number of page
12-15	C-F	VTFPTR	Forward pointer
16	10	LVTABE	Length of VTABE

Figure 262. VIO Table Entry (VTABE)

VIO File Identification Entry (VIOTABE)

Bytes			
Dec	Hex	Label	Description
0–1	0–1		Reserved
2	2	VIORBCM1	Communication byte
		VIORBTRB	X'80' VIO POINT request complete
3	3	VIORBRTC	Return code
		VIORBEOF	X'04' Requested block outside area
		VIORBERR	X'08' Unrecoverable error
		VIORBINC	X'0C' Inconsistent state
4–7	4–7	VIORBASZ	Actual size of area in bytes
8–11	8–B	VIORBBSZ	Size of a block in bytes
12–15	C–F	VIORBPNT	Virtual address of current block = 0 : No VIO POINT given up to now < 0 : VIO POINT in process
16–19	10–13	VIORBRBA	Relative byte addr. of current block
20–23	14–17	VIORBASR	Address of service routine
24–31	18–1F	VIOTBSID	Storage ID for validation
....	....	VIOTIB	Pseudo–TIB for VIO ....
32–39	20–27		1st two fullwords of TIB
40	28	VIOTIBFL	TIBFLAG in VIOTIB
		VIOIND	X'A0' Indication for VIO TIB
40–47	28–2F		Next two fullwords of TIB
48–49	30–31	VIORTID	TID of VIO POINT requestor
50–51	32–33	VIOOWNER	PIK of owner partition
....	....	.... End of	Pseudo–TIB for VIO ....
52–55	34–37	AFLSEGTB	Address of 1st file segment block
56–59	38–3B	VTABEACT	Address of VTABE belong. to VIORBPNT
60–75	3C–4B	VIORBSAV	Register save area
76–79	4C–4F	VIOBLKN	Save area for total block number
80–81	50–51	VIOTBOPT	Option bytes from VIOPL
80	50	VIOTBLFT	Scope option from VIOPL
81	51	VIOFLAG	Flag byte
		ASYNCH	X'80' Asynchronous request
		VIOSTSAV	X'40' Status already saved
82–83	52–53		Reserved
84–87	54–57	VIOTBNXT	VIOTABE chain pointer
87	57	VIOTABLN	Length of VIOTABE – 1

Figure 263. VIO File Identification Entry (VIOTABE)

VIO Block Table Entry (BLKTBE)

Bytes			
Dec	Hex	Label	Description
0-1	0-1	BLKPAG	Page / frame addr. belonging to block
0	0	BLKKEY	Storage key
1	1	BLKSTAT	Status indication
			370 and VM mode:
		FRCON	X'0A' Frame connected to block
		BLKDISC	X'08' Whether page nor frame conn.
			ECPS:VSE mode:
		FRCON	X'06' Frame connected to block
		BLKDISC	X'04' Whether page nor frame conn.
			All modes:
		BLKERR	X'02' Error on block
		BLKPDS	X'01' Copy on external storage
2	2	LBLKTBE	Length of block table entry

Figure 264. VIO Block Table Entry (BLKTBE)

XPCC CONTROL BLOCKS (IDCB, CRCB)

Identification Control Block (IDCB)
Connect Request Control Block (CRCB)

Identification Control Block (IDCB)

Bytes Dec	Bytes Hex	Label	Description
0 - 3	0 - 3	XPIDPT	Pointer to next ID-CB
4 - 7	4 - 7	XPICRPT	Pointer to first CR-CB
8 - 9	8 - 9	XPIPART	Offset to that part of CR-CB which belongs to current application
10 - 11	A - B	XPITID	TID of ID-CB owner
12 - 13	C - D	XPIMTID	TID of corresponding maintask
14 - 21	E - 15	XPIMTID	Identification key (token)
22 - 29	16 - 1D	XPIAPPL	Application name
30 - 31	1E - 1F	XPICRQS	Number of requested connections
32 - 33	20 - 21	XPICNTR	Number of open connections
34	22	XPIFLG1	Flag byte
		XPISUBS	X'80' IBM-subsystem
		XPITMQ	40 Application issued TERMQSCE
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
35	23	XPIFLG2	Flag byte (reserved)
36	24	XPIDEND	Length of IDCB

Figure 265. Identification Control Block (IDCB)

Connection Request Control Block (CRCB)

Bytes			
Dec	Hex	Label	Description
0 - 3	0 - 3	XPZTCBC	TCB chain pointer
4 - 11	4 - B	XPZCRTK	Path-id (connection request token)
12 - 15	C - F	XPZBUFAD	SEND buffer address
12	C		X'80' Last buffer of a list
16 - 19	10 - 13	XPZBUFLN	Buffer length
20 - 67	14 - 43		max. 7 entries in list
68 - 71	44 - 47	XPZTOTAL	Total buffer length
72 - 75	48 - 4B	XPZREPLY	Address of reply area
76	4C	XPZFLAG	Flag in user area
77 - 79	4D - 4F	XPZRPYLN	Reply area length
80 - 87	50 - 57	XPZUSER	User data
88 - 93	58 - 5B	XPZSPACE	SCB pointer of partner
92	5C	XPZFCT	Function code
93	5D	XPZFLG1	Flag byte
		XPCONCL	X'80' Connection is completed
		XPCONBSY	40 Connection is busy
		XPCINTCB	20 In TCB chain
			10 Reserved
			08 Reserved
			04 Reserved
		XPZCONPE	02 Connection exit pending
		XPZRPOST	01 Post at receive after SENDR
94	5E	XPZFLG2	Flag byte (reserved)
		XPTERMAB	X'80' Other side terminated abnorm.
		XPTERMNO	40 Other side terminated normally
		XPDSCED	20 Other side disconnected
			10 Reserved
			08 Reserved
			04 Reserved
		XPINOVM	02 Partner in other VM machine
		XPCURRSP	01 Both part. in current space
95	5F	XPZREAS	Reason code
96	60	XPZCEND	Length of Common Part

Figure 266 (Part 1 of 2). Connection Request Control Block (CRCB)

Bytes			
Dec	Hex	Label	Description
Block of First Communication Partner			
96 - 99	60 - 63	XPZNXTCR	Pointer to next CR-CB
100 -101	64 - 65	XPZPART	Offset to that part of CR-CB which belongs to current application
102 -103	66 - 67	XPZTID	TID of connect owner
104 -107	68 - 6B	XPZPCCB	XPCCB address
108 -111	6C - 6F	XPZIDADR	Address of corresponding IDCB
112 -119	70 - 77	XPZTOAP	To-Application name
120	78	XPZFLG3	Flag byte
		XPSEND	X'80' SEND pending
		XPSENDR	40 SENDR pending
		XPCLEAR	20 Sender cleared request
		XPRECVE	10 Receive after SENDR executed
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
121 -123	79 - 7B		Reserved
End of First Part			
124	7C	XPZFEND	Len. of First Partner + Common Sect.
Block of Second Communication Partner			
124 -127	7C - 7F		Pointer to next CR-CB
128 -129	80 - 81		Offset to that part of CR-CB which belongs to current application
130 -131	82 - 83		TID of connect owner
132 -135	84 - 87		XPCCB address
136 -139	88 - 8B		Address of corresponding IDCB
140 -147	8C - 93		To-Application name
148	94	XPZFLG3#	Flag byte
			X'80' SEND pending
			40 SENDR pending
			20 Sender cleared request
			10 Receive after SENDR executed
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
149 -159	95 - 9F		Reserved
160	A0	XPZREND	Total Length of CRCB

Figure 266 (Part 2 of 2). Connection Request Control Block (CRCB)

INPUT/OUTPUT CONTROL WORDS, BLOCKS AND AREAS

BASIC INPUT/OUTPUT CONTROL WORDS (CAW, CSW, CCW)

Figure 267 to Figure 269 on page 549 show the layout of the Channel Address Word (CAW), the Channel Status Word (CSW) and the Channel Command Word (CCW). For more information refer to the appropriate 'PRINCIPLES OF OPERATION' manual.

Layout of CAW

KEY	0000	COMMAND ADDRESS	
0	4	8	31

Bits	Description
0 - 3	Storage protection key
4 - 7	Reserved (must be zero)
8 -31	Address of first/only CCW

Figure 267. Channel Address Word (CAW)

Layout of CSW

```
┌────┬────┬──────────────────────────┬──────────────────┬─────────────────────┐
│    │    │                          │                  │                     │
│ KEY│0000│     COMMAND ADDRESS      │   STATUS BYTES   │     BYTE COUNT      │
│    │    │                          │                  │                     │
└────┴────┴──────────────────────────┴──────────────────┴─────────────────────┘
 0    4    8                          32                 48                  63
```

Bits	Apprev.	Description
0 - 3		Storage protection key
4		Reserved (must be zero)
5		Logout pending
6 - 7		Deferred condition code
8 - 31		Address+8 of last CCW executed
32	ATTN	Attention
33	SM	Status modifier
34	CUE	Control unit end
35	BSY	Busy
36	CE	Channel end
37	DE	Device end
38	UC	Unit check
39	UX	Unit exception
40	PCI	Program controlled interruption
41	IL	Incorrect length
42		Channel program check
43		Channel protection check
44		Channel data check
45		Channel control check
46		Interface control check
47		Channel chaining check
48 - 63		Residual byte count

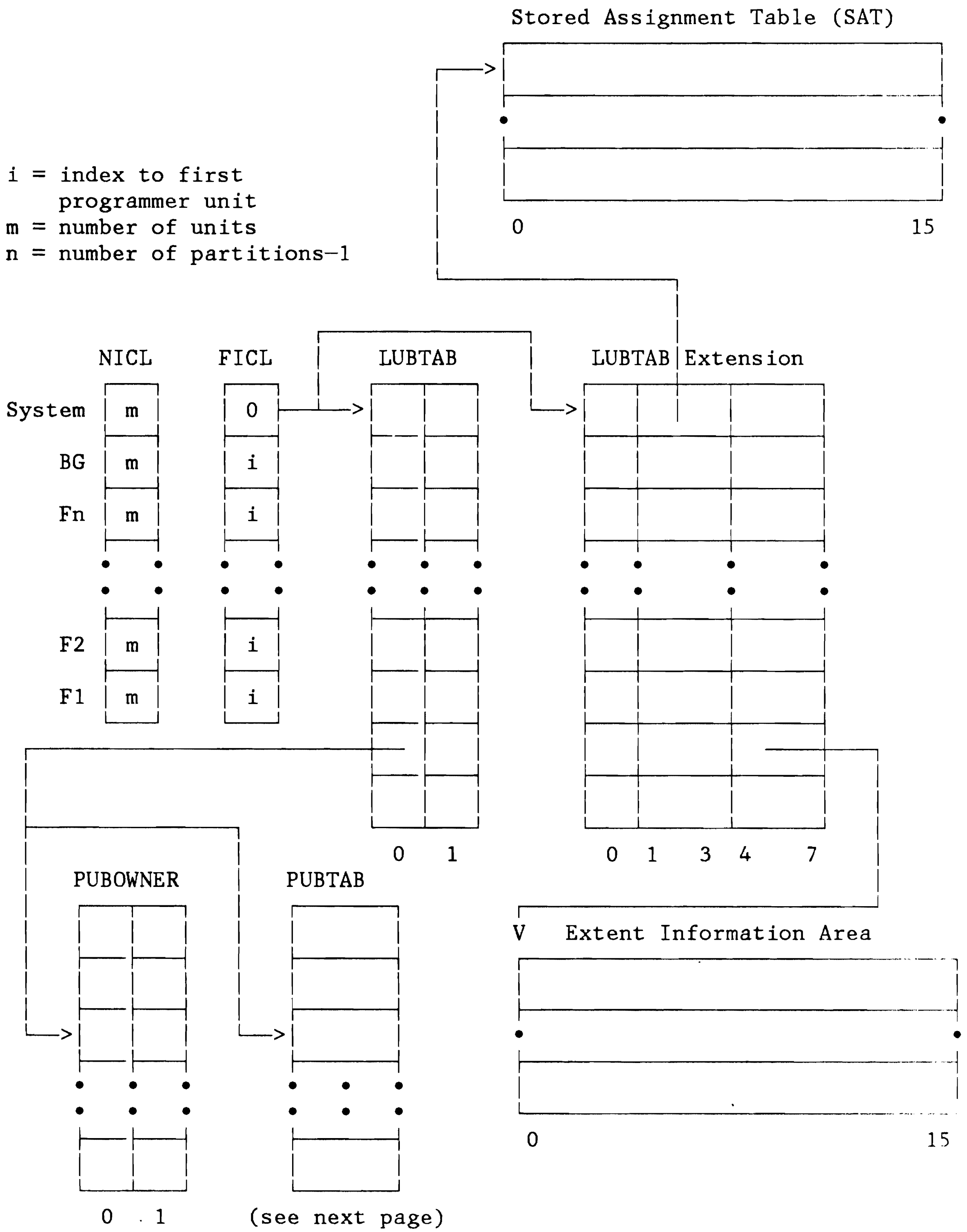
Figure 268. Channel Status Word (CSW)

Layout of CCW

COMMAND CODE	I/O AREA ADDRESS	FLAGS	00	TP OPERATION CODE	BYTE COUNT
0	8	32	38	40	48 63

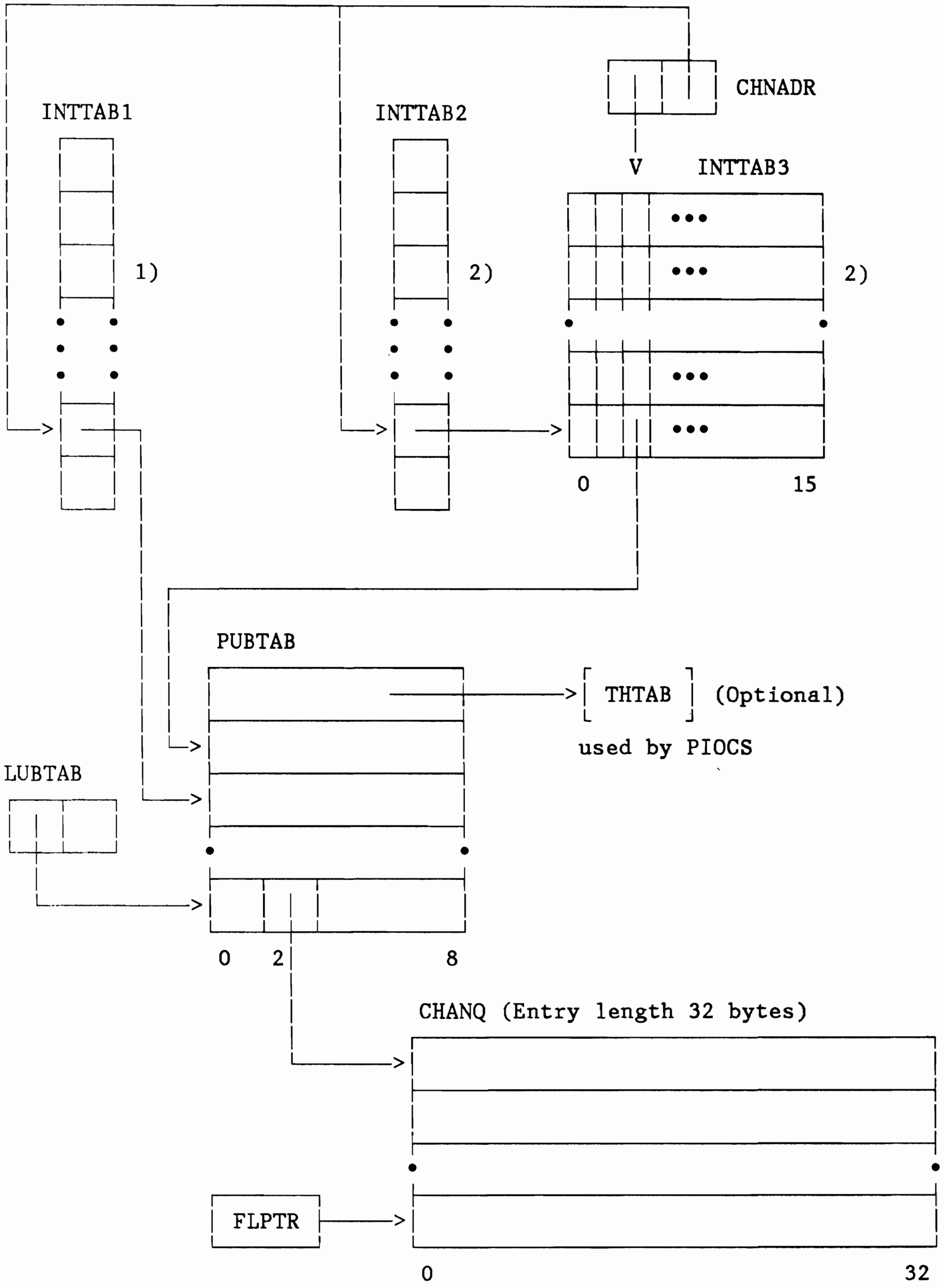
Bits	Apprev.	Description
32	CD- bit	X'80' Causes use of address portion of next CCW (data chaining)
33	CC- bit	40 Causes use of next CCW (command chaining)
34	SLI- bit	20 Causes Suppression of incorrect length indication
35	Skip bit	10 Suppresses transfer of data to processor storage
36	PCI- bit	08 Cause channel to generate a program controlled interruption
37	IDA- bit	04 Specifies indirect data addressing, 370 Mode only (can only be specified for REAL addr.)
38		02 Must initially be zero
39		01 Must initially be zero

Figure 269. Channel Command Word (CCW)



i = index to first programmer unit
m = number of units
n = number of partitions-1

Figure 270 (Part 1 of 4). I/O Table Interrelationship



1) Initialized by IPL.
2) Optionally allocated and initialized by IPL.

Figure 270 (Part 2 of 4). I/O Table Interrelationship

Label	Description
NICL (Number in Class List)	Byte 0 contains the No. of System Class LUBs. The remaining bytes contain the No. of Programmer Class LUBs for each partition (BG, ..., F3, F2,F1)
FICL (First in Class List	Byte 0 is an index pointer to both, the first System Class LUB entry within each Partition LUB Table as well as to the LUBTAB Extension (BG, ..., F3, F2, F1). The remaining byte contain the entry number of the first Programmer Class LUB entry within the Partition LUBTAB and LUBTAB-Extension.
LUBTAB (Logical Unit Block Table)	Byte 0 of each entry is an index ptr. to both, an entry in the PUB Table (PUBTAB) and PUB Ext. Area (PUBXAREA) as well as to the PUB OWNERSHIP Table (PUBOWNER). X'FF' indicates that no logical unit is assigned. X'FE' indicates that I/O requests are to be ignored.
LUBTAB EXTENSION	Bytes 1-3 point to first STORED ASSIGNMENT TABLE entry. Bytes 4-7 point to first EXTENT INFORMATION AREA entry. Zero indicates no extent information available.
EXTENT INFORMATION	Bytes 1 - 3 point to the next EXTENT INFORMATION ENTRY. Zero identifies this entry as the last one in the chain.
STORED ASSIGNMENT ENTRY	Bytes 1 - 3 point to the next STORED ASSIGNMENT TABLE entry.
PUBTAB (Physical Unit Block Table)	Byte 2 is index ptr. to the CHANNEL QUEUE TABLE (CHANQ) X'FF' indicates that no request is queued to the PUB. Byte 5 is an index pointer which for: DASD points to the entry in the TRACK HOLD TABLE (THTAB)
PUBXAREA (Physical Unit Block Extension Area)	Bytes 0 - 3 contain the address of the associated PUBX entry.
PUBX (Physical Unit Block Extension Entry)	For DASD devices: Bytes 16 - 19 point to the CCW chain that is to be used in case DASDFP=YES was specified in the IPL SYS command.

Figure 270 (Part 3 of 4). I/O Table Interrelationship

Key	Description
INTTAB1 (Interrupt Processing Table 1)	Consists of 255 one-byte entries that contain an index to a related PUB entry, or zero, if an entry is to be obtained from the INTTAB2 Table. The one-byte device address stored in low core at interrupt time is used as an index.
INTTAB2 (Interrupt Processing Table 2)	Consists of 255 one-byte entries that contain an index to an entry in the INTTAB3 table, or zero, if such an entry does not exist. The one-byte device address stored in low core at interrupt time is used as an index.
INTTAB3 (Interrupt Processing Table 3)	Contains one 16-byte entry for each PUB entry that has a device address (PUB byte 1), which exists at least two time on different channels. The Channel-ID stored in low core at interrupt time is used to index a byte within the entry. This byte contains an index to the related PUB, or zero, if the PUB is not defined.
FLPTR (Free List Pointer)	This one-byte pointer contains the entry index of the next free entry in the Channel Queue Table (CHANQ).
CHANQ (Channel Queue Table)	Byte 0 in each entry is an index to the next entry in sequence, or it contains X'FF' if the entry is the last in a chain. There are two types of chains: The DEVICE CHAIN is based on a PUB entry, the FREELIST CHAIN is based on the FLPTR entry.
THTAB (Track Hold Table)	Byte 0 on each entry points to the next entry in the chain of requests for a track/block to be held on a specific DASD (or the next free entry if in the free list) or it contains X'FF' if the entry is the last in a chain. Byte 12 contains a backward pointer. The backward pointer of the first Track Hold Table entry contains the PUB index.

Figure 270 (Part 4 of 4). I/O Table Interrelationship

Logical Unit Block Tables (LUBTAB, LUBX, SAT, Ext.Inf.)

Logical Unit Block Table (LUBTAB)
LUBTAB Extension Table
Stored Assignment Table Entry (SAT)
Extent Information Entry

Logical Unit Block Table (LUBTAB)

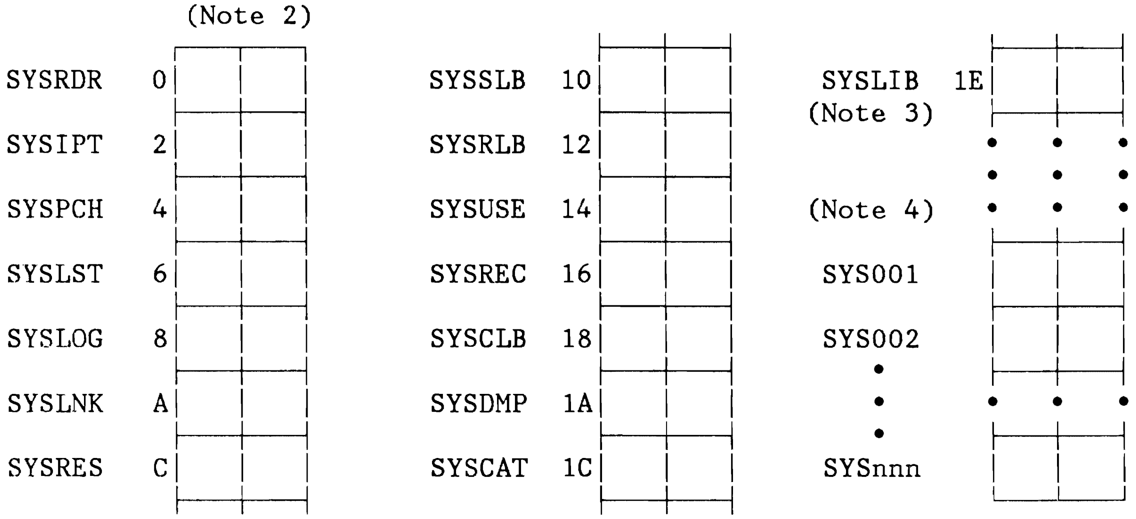
Bytes 76-77 (X'4C' - X'4D') of the Partition Communication Region contain the address of the LUB table. Label LUBTAB identifies the first byte of the table for the BG partition, label FnLUB for the other partitions (n = 1 - B).

Logical Unit Block Entry (Note 1):

Bytes		
Dec	Hex	Description
0	0	PUB index of device assigned to this logical unit X'FF' if no PUB is assigned or X'FE' if I/O is to be ignored for this log. unit
1	1	Reserved

Figure 271 (Part 1 of 2). Logical Unit Block (LUB) Entry and TABLE

Logical Unit Block Table:



Notes:

1. Null entries X'FFFF' are generated at supervisor generation time.
2. There are 14 externally known system LUBs and one internally used for label access method.
3. System LUBs used by dynamic assignments.
4. The total number of system LUBs is a constant.

Figure 271 (Part 2 of 2). Logical Unit Block (LUB) Entry and TABLE

LUBTAB Extension Table

The LUB Extension Table for each Partition is initialized by IPL. It has as many entries as allocated to the LUB table of that Partition. Each entry is 4 bytes long except the user did specify DASDFP=YES (IPL SYS-command) in which case each entry is 8 bytes in length. The start address of the LUB Extension table is stored by IPL in bytes 168-171 (X'A8-AB') of the Partition Communication Region.

Bytes			
Dec	Hex	Label	Description
0	0	LUBXFLG	Flag Byte
		LUBXPA	X'80' Permanent alternate assignment stored
		LUBXTA	40 Temporary alternate assignment stored
		LUBXPE	20 Permanent assignment stored
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
			(LUBXPA and/or LUBXTA is on)
1-3	1-3	LUBXADR	Pointer to first Stored Assignment Table entry (SAT)
			(LUBXPA and LUBXTA both off)
1	1		Reserved
2-3	2-3	LUBXPER	Stored permanent assignment
OPTIONAL DASDFP=YES			
4-7	4-7	LUBXEPT	Pointer to first EXTENT INFORMATION chain entry or zero if no EXTENT INFORMATION available

Figure 272. Logical Unit Block (LUB) Extension Entry

Stored Assignment Table Entry (SAT)

The LUB Extension table entry may contain a pointer to a chain of assign entries, each containing additional information on stored assignments. Each entry is fixed length and is allocated in the System GETVIS area.

Bytes			
Dec	Hex	Label	Description
0	0	SATFLG	Flag byte
			X'80' Reserved
			40 Reserved
		SATPE	20 Permanent Assignment saved in this entry
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
1-3	1-3	SATNEXT	Pointer to next assign entry in the chain
4	4	SATEOCH	Offset within SATSAV of next free entry
5	5	SATEOPCH	Offset within SATSAV of saved permanent assignment
6-7	6-7	SATSAV	Space for saving permanent assignment (max. of 5)
•••	•••		
14-15	E-F		

Figure 273. Stored Assignment Table Entry (SAT)

Extent Information Entry

The LUB extension table entry contains a pointer to a chain of Extent entries for DASD File Protection. Each entry is fixed length and is allocated in the System GETVIS area.

Bytes			
Dec	Hex	Label	Description
0	0	EXBFLG	Flag Byte
		EXBREAD	X'80' Allow READ access only
			(no multi-track operation)
		EXBSHORT	40 Extent information is CC only
			20 Reserved
			10 Reserved
			08 Reserved
			04 Reserved
			02 Reserved
			01 Reserved
1-3	1-3	EXBNXT	Pointer to next Extent entry in the chain or zero if this is the last Extent entry
4-7	4-7	EXBHI	High Extent Limit
			CKD Device Cylinder+Head No.
			FBA Device Physical Block No.
8-11	8-B	EXBLOW	Low Extent Limit
			CKD Device Cylinder+Head No.
			FBA Device Physical Block No.
12-13	C-D	EXBCOUNT	Usage count for this extent
14-15	E-F		Reserved

Figure 274. Extent Information Entry

Physical Unit Block Tables (PUBTAB, PUBX, PUB2, PUBOWNER)

Physical Unit Block Table (PUBTAB)
Physical Unit Block Extension (PUBX)
Physical Unit Block 2 (PUB2)
PUB Ownership Table (PUBOWNER)

Physical Unit Block Table (PUBTAB)

Bytes 64-65 (X'40'-X'41') of the Partition Communication Region contain the address of the PUB table. Label PUBTAB identifies the first byte of the table.

Bytes			
Dec	Hex	Label	Description
0	0	PUBCHANN	Channel number of device (Hex 0-F) X'FF' indicates end of PUBTAB
1	1	PUBDEVNO	Unit number
2	2	PUBCHQPT	Index to first CHANQ entry X'FF' indicates no request enqueued
3	3		Reserved
4	4	PUBDEVTY	Device type code (see Appendix C)
5	5	PUBOPTN	For TAPE devices: Tape Mode from ADD or ASSGN For DASD-Devices: Index of TRKHLD Table entry or X6 For MICR devices: External line in use For 3704/3705: Type of channel adapter For 2560 or 5424/5425: X'80' Repositioning required (used for ERP) 40 SYSPCH temporarily assigned to hopper 2 20 SYSIPT temporarily assigned to hopper 2 10 SYSRDR temporarily assigned to hopper 2 08 Reserved 04 SYSPCH permanently assigned to hopper 2 02 SYSIPT permanently assigned to hopper 2 01 SYSRDR permanently assigned to hopper 2 For 3800: Bit 0-1 00 3800 01 3800 B 10 3800 C 11 3800 BC

Note: A PUB entry must be added during IPL for any device of the installation

Figure 275 (Part 1 of 2). Physical Unit Block (PUB) Entry

Bytes			
Dec	Hex	Label	Description
6	6	PUBCSFLG	Channel Scheduler flags
		DEVBSY	X'80' Device is active
		SWITCH	40 Device is switchable
			20 Reserved
		QEDERR	10 I/O error queued for recovery
		OPINTV	08 Operator intervention required
		INTPEND	04 Interrupt was trapped by SDAID
		BRSDEV	02 Burst or overrunable device
		SVNTRK	01 7-track tape unit
7	7	PUBJCFLG	Job Control flags
			Bits 0-4: TAPE : Standard MODE assignment
			Not TAPE : All ones if device is up
			Device DOWN: All zeros
			5: Device supports RPS
			6: Alternate path is not operational
			7: Primary path is not operational

Note: A PUB entry must be added during IPL for any device of the installation

Figure 275 (Part 2 of 2). Physical Unit Block (PUB) Entry

Physical Unit Block Extension (PUBX)

The PUBX table is a logical extension of the PUB table. There is one PUBX entry for each device added at IPL. A PUBX entry is addressed via address table APBXAREA at offset 4*PUB index (see Figure below). The PUBX entries have variable length and contair device related error information. Their layouts are shown in the paragraph on I/O error processing under 'Error Entries'.

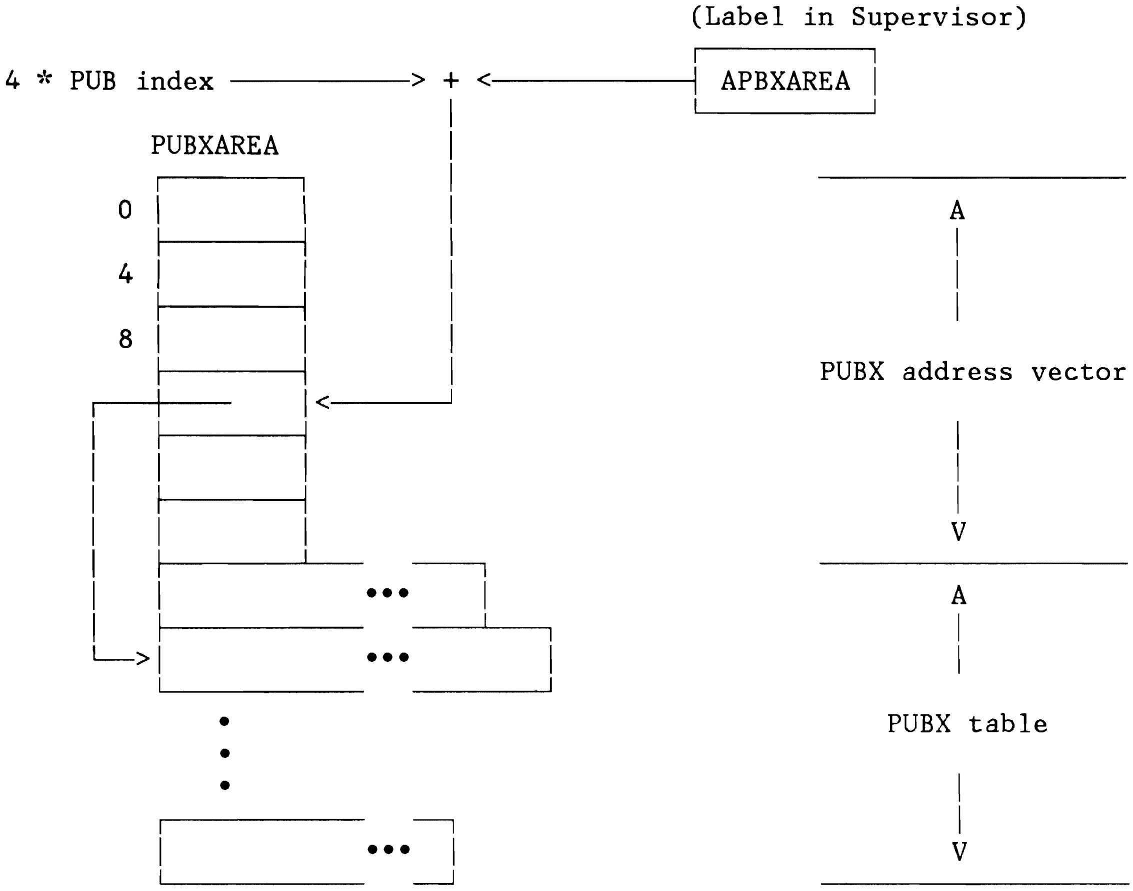


Figure 276. PUBX Table Interrelationship

Bytes			
Dec	Hex	Label	Description
0	0	PBXFLG	Flag byte
		PBXDASD	X'80' DASD device
		PBXTAPE	40 Tape device
		PBXUR	20 Unit record device
			10 – 02 Reserved
		PBXSLOG	01 SYSLOG device
1	1	PBXFLAG1	Flag byte
		PBXSHR	X'80' Partition sharable device Is on for DASD devices, for the SYSLOG device and for unit record devices, which are sharable as POWER dummy devices
		PBXMTFLG	40 Mount request pending
			20 – 01 Reserved
2 – 3	2 – 3	PBXCUU	CUU address
4	4	PBXPUBCD	VSE device type code
5 – 11	5 – B	PBXSNSID	Sense device type information
5	5		X'FF' If entry is valid
6 – 7	6 – 7	PBXCUTYP	Control unit type number
8	8	PBXCUMOD	Control unit model number
9 – 10	9 – A	PBXDVTYP	Device type number
11	B	PBXDVMOD	Device type model number
12 – 13	C – D	PBXOWNER	PIK of partition owning the device, if applicable
14 – 15	E – F		Reserved
			(if PBXSHR OFF)
16 – 19	10 – 13	PBXUSCNT	Device usage counters
20 – 23	14 – 17	PBXJACNT	Job Accounting SIO counters
			(if PBXSHR ON)
16 – 19	10 – 13	PBXUSOFF	Offset of usage counters within partition string
20 – 23	14 – 17	PBXJAOFF	Offset of SIO counters within partition string
24 – 27	18 – 1B	PBXERBLK	Addr. of Error Entry for this device
28	1C	PBXCLNG	End of common section
			End of section common to all devices
28 – 31	1C – 1F	PBXCCW	DASD devices: Address of Set File Mask CCW's TAPE devices: Addr. of Set Mode CCW's
32	20	PBXTLNG	End of tape device section
32	20	PBXDLNG	End of DASD device section

Figure 277. Physical Unit Block Extension (PUBX)

Physical Unit Block Table 2 (PUB2)

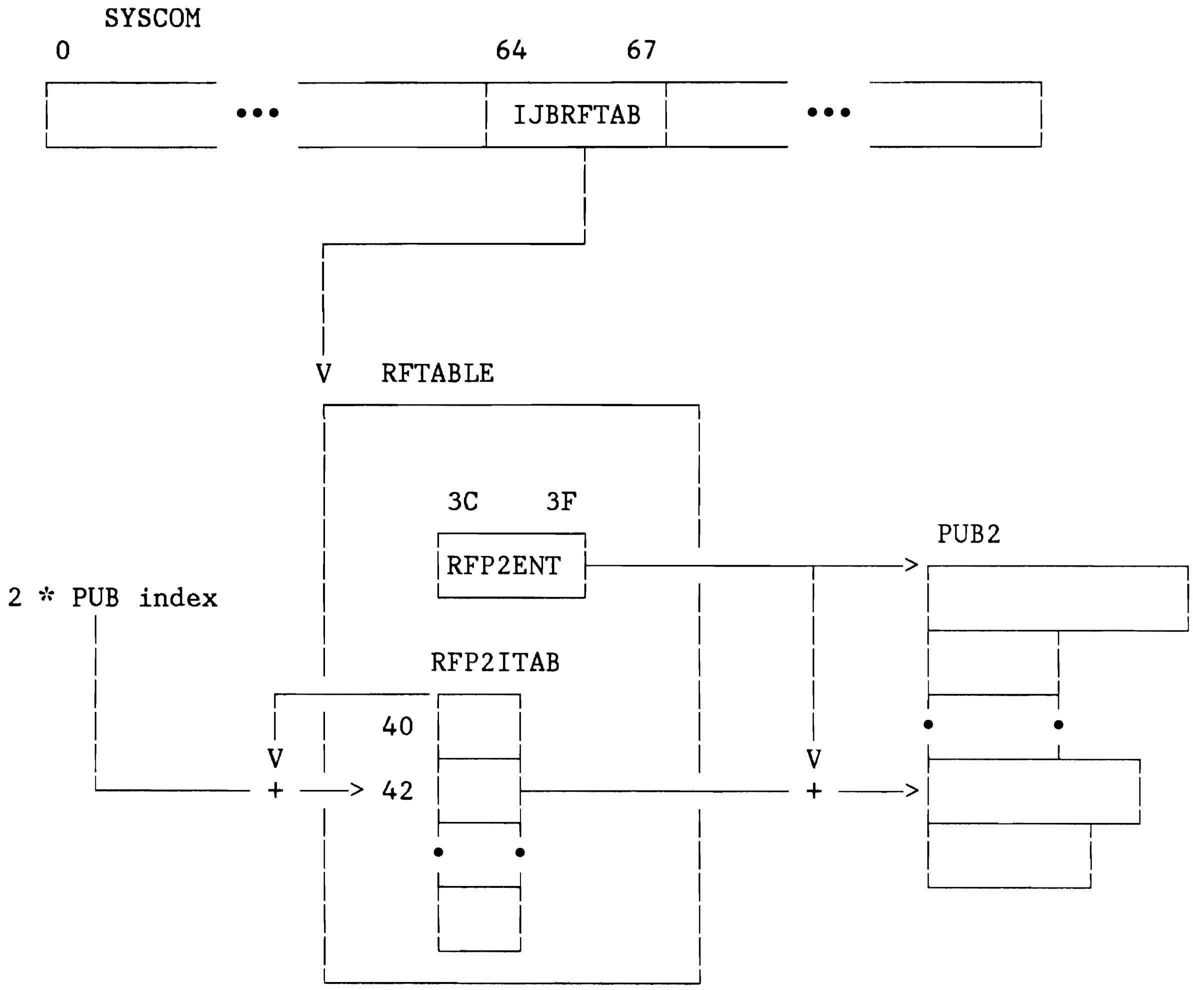


Figure 278. PUB2 Relationship

Bytes Dec	Hex	Label	Description
0 - 3	0 - 3	P2USAGE	Usage count (number of non-ERP SIO)
4	4	P2FLAGS	Flag byte common to all PUB2 entries
		P2INTSM	X'80' Device is in intensive mode
		P2DIAGM	40 Device is in diagnostic mode
		P2NORCM	20 No recording mode
		P2STAT2	10 Call statistics transient 2
		P2NAMEF	08 Use PUB2 name completion field
		P2OPEN	04 Volume opened on this device
			02 Reserved
			01 Reserved
5	5	P2LIMIT	CE mode limit byte
		P2BBMASK	CE mode byte/bit mask
6	6	PUB2EXT	End of basic PUB2

Figure 279. Physical Unit Block Table 2 (PUB2)

Physical Unit Block Table 2 Extensions

Bytes Dec	Hex	Label	Description
6	6	P2UNITX	Start of unit record PUB2
6 - 11	6 - B	SDRUNITR	SDR counters for unit record devices
12	C	P2UNITE	End of unit record PUB2

Figure 280. Unit Record and Unsupported Device Extension

Bytes Dec	Hex	Label	Description
6	6	P23540X	Start of PUB2 extension
6 - 13	6 - D	SDR3540	SDR counters
14 - 15	E - F	P23540R	Reserved
16	10	P23540E	End of 3540 PUB2

Figure 281. 3540 Diskette Extension

Bytes			
Dec	Hex	Label	Description
6	6	P23211X	Start of PUB2 extension
6 - 11	6 - B	SDR3211	SDR counter area
12	C	P23211E	End of 3211 PUB2

Figure 282. 3211 Printer Extension

Bytes			
Dec	Hex	Label	Description
6	6	P23800X	Start of PUB2 extension
6	6	PB2SDR1	Channel data checks
7	7	PB2SDR2	Cont forms stacker misfolds
8	8	PB2SDR3	Burster/trimmer jams
9	9	PB2SDR4	No burst check
10	A	PB2SDR5	Burster/stacker jams
11	B	PB2SDRE	End of counters area
11	B	PB2DFLG	Default flags
		PB2DBRST	X'80' Default spec.=burst
12 - 15	C - F		Reserved
16 - 19	10 - 13	PB2DFCB	Default fcb id
20 - 23	14 - 17	PB2DCHAR	Default char. arrangement table id
24 - 27	18 - 1B	PB2DMDFY	Default copy modific. id
28 - 31	1C - 1F	PB2DFLSH	Default forms overlay frame id
32 - 35	20 - 23	PB2DFORM	Default paper forms id
36	24	PB2DFTE	End of default area
36 - 39	24 - 27	PB2WCGMS	Character sets presently load

Figure 283 (Part 1 of 2). 3800 Printer Extension

Bytes			
Dec	Hex	Label	Description
40	28	PB2WMOD	WCGM# with modified character sets
		PB2WMOD0	X'80' WCGM0 contains a modified character set
		PB2WMOD1	40 WCGM1 cont. a mod.chr set
		PB2WMOD2	20 WCGM2 cont. a mod.chr set
		PB2WMOD3	10 WCGM3 cont. a mod.chr set
41	29	PB2FLAG1	First byte of flags
		PB2BURY	X'30' Burst = Y last specified
		PB2BURN	10 Burst = N last specified
		PB2UDCHK	08 DCHK=U was specified
42	2A	PB2FLAG2	Second byte of flags
		PB2TRCY	X'30' TRC=Y was specified
		PB2TRCN	10 TRC=N was specified
		PB2DEBTR	0E Debug = trac last specified
		PB2DEBDU	0A Debug = dump last specified
		PB2DEBTE	06 Debug = term last specified
		PB2DEBNO	02 Debug = none last specified
43	2B		Reserved
44 - 47	2C - 2F	PB2FCB	Currently loaded FCB id
48 - 63	30 - 3F	PB2CHAR	Character arrangement tables (CAT)
48 - 51	30 - 33	PB2CHAR1	Id of 1st CAT currently loaded
52 - 55	34 - 37	PB2CHAR2	Id of 2nd CAT currently loaded
56 - 59	38 - 3B	PB2CHAR3	Id of 3rd CAT currently loaded
60 - 63	3C - 3F	PB2CHAR4	Id of 4th CAT currently loaded
64 - 67	40 - 43	PB2CMCHR	Id of CAT used when loading current copymod
68 - 71	44 - 47	PB2CPMOD	Id of copymod currently loaded into the printer
72 - 75	48 - 4B	PB2FORMS	Id of paper form currently loaded
76 - 79	4C - 4F	PB2FLASH	Id of current forms overlay frame
80 - 87	50 - 57	PB2COPYG	Eight copy group count last received by setprint
88	58	PB2CINDX	Copy group id last received by setprint
89	59	PB2FLSHC	Flash count last received by setprint
90 - 91	5A - 5B		Reserved
92	5C	P23800E	End of 3800 PUB2

Figure 283 (Part 2 of 2). 3800 Printer Extension

Bytes			
Dec	Hex	Label	Description
6 - 25	6 - 19	SDR3886	SDR counter area
26	1A	P23886E	End of 3886 PUB2

Figure 284. 3886 Optical Character Reader Extension

Bytes			
Dec	Hex	Label	Description
6	6	P23890X	Start of PUB2 extension
6 - 15	6 - F	SDR3890	SDR counter area
16	10	P23890E	End of 3890 PUB2

Figure 285. 3890 Document Reader Extension

Bytes			
Dec	Hex	Label	Description
6	6	P2DISKX	Start of PUB2 extension
6	6	P2DFLG	Disk flags
		P2SDERRQ	X'80' Soft DASD error is queued
		P2DLOG	40 ERP requests error logged
7 - 8	7 - 8		Reserved
9	9	P2DMOD	Physical module identifier
10 - 15	A - F	P2DVOL	Volume serial number
16	10	P23330E	End of 3330 PUB2
16	10	P23340E	End of 3340 PUB2
16	10	P23350E	End of 3350 PUB2
16	10	P2fBAE	End of FBA PUB2
16 - 23	10 - 17	SDRDISK	SDR counters for 23xx
24	18	P2DISKE	End of 23xx PUB2

Figure 286. Disk Device Extension

Bytes			
Dec	Hex	Label	Description
6	6	P2TAPEX	Start of PUB2 extension
6 - 7	6 - 7	P2TNAME	Name of ERP that wants control
8	8	P2TFLG1	Tape flags 1
		P2TUNSOL	40 Unsolicited interrupt for tapes
		P2TERP	20 ERP is in control
		P2TREPO	10 ERP requests repositioning
		P2TIEORG	08 Use original tie byte; if off the opposite tie is used
		P2TECPT	04 Intercept next SIO request
		P2TROR	02 ERP read opposite request
		P2TREST	01 Restart users CCW chain
9	9	P2TFLG2	Tape flags 2
10	A	P2TFLG3	Tape flags 3
11	B	P2TEMPR	Temporary read count
12	C	P2TEMPW	Temporary write count
13	D	P2NOISE	Noise record count
14 - 15	E - F	P2ERG	Erase gap count
16 - 17	10 - 11	P2CLEAN	Cleaner action counts
18	12	P2PRD	Permanent read errors
19	13	P2PWT	Permanent write errors
20	14	P2ORGTIE	Tie original direction
21	15	P2OPPTIE	Tie opposite direction
22	16	P2ECTR0	ERP counter
23	17	P2ECTR1	ERP counter
24 - 31	18 - 1F	P2TWORKA	ERP work area
32 - 37	20 - 25	P2TVOL	Tape serial number
38 - 39	26 - 27	P2TBLK	Block length
40 - 43	28 - 2B	P2CCWAD	
44	2C	P2CSWRES	
45 - 47	2D - 2F		Reserved
48 - 63	30 - 3F	P2RUNSAV	Save area for run ERP
48 - 55	30 - 37	P2TSCSW	For CSW in error
56 - 57	38 - 39	P2TSSNS0	For sense bytes 0,1
58	3A	P2TSSNS5	For sense byte 5
59 - 63	3B - 3F		Reserved
.... 2400 Extension..			
64 - 73	40 - 49	SDR2400	2400 SDR area
74 - 75	4A - 4B		Reserved
4C	4C	P22400E	End of 2400 PUB2
.... 3420 Extension..			
64 - 83	40 - 53	SDR3420	3420 tape drive counter area
84	54	P23420E	End of 3420 PUB2

Figure 287. Tape Device Extension

PUB Ownership Table (PUBOWNER)

Bytes 120 - 123 (X'78'-X'7B') of the System Communication Region (SYSCOM) contain the address of the PUB Ownership Table. Label PUBOWNER identifies the first byte of the table. One fixed length entry is associated to each PUB and has the following layout.

Bits	Description
0	Device is owned by ACF/VTAM
1-2	Reserved
3	Device is owned by the system (e.g. contains PDS extent)
4-15	Identifier of Partition owning the PUB

Bit setting	Partition owning the PUB if number of partitions is										
	2	3	4	5	6	7	8	9	10	11	12
X'000'	UNASSIGNED										
001	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG
002	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
004		F1	F2	F3	F4	F5	F6	F7	F8	F9	FA
008			F1	F2	F3	F4	F5	F6	F7	F8	F9
010				F1	F2	F3	F4	F5	F6	F7	F8
020					F1	F2	F3	F4	F5	F6	F7
040						F1	F2	F3	F4	F5	F6
080							F1	F2	F3	F4	F5
100								F1	F2	F3	F4
200									F1	F2	F3
400										F1	F2
800											F1

Figure 288. Physical Unit Block Ownership Table (PUBOWNER) Entry

Device Usage Counters (DVCUSCNT)

For devices, which are not partition sharable (PBXSHR=0), the usage and SIO counters are included in the PUBX, see Figure 277 on page 563. For partition sharable devices (PBXSHR=1), one set of usage and SIO counters is needed for every partition. All usage counters belonging to one partition are allocated as a string. The address of the string can be found in PCB.PCBCNT, see Figure 232 on page 499. The offset of the usage counters of a given device within the partition string can be found in fields PUBX.PBXUSOFF and PUBX.PBXJAOFF.
The SIO counter for Job Accounting is a single 4-byte field. For partition sharable devices, SIO counters are included in the partition string only if SYS JA=YES was specified at IPL time. Device usage counters are always allocated. There structure and meaning are described below.

Bytes			
Dec	Hex	Label	Description
0-1	0-1	DVCPUCNT	(if DVCPWRSP OFF) Physical usage counter Gives the number of times a device is physically accessed in a partition, either via a Logical Unit assignment or via physical addressing.
0-1	0-1	DVCPWRTD	(if DVCPWRSP ON) This field contains the TID of the task, which has a spooling request pending. If no request is pending, it contains X'0000'.
2	2	DVCUSFLG	Flag byte X'80' Reserved
		DVCPWRSP	40 Used as a dummy device for POWER 20 Reserved 10 Reserved 08 Reserved 04 Reserved 02 Reserved 01 Reserved
2-3	2-3	DVCLUCNT	Logical usage counter Gives the total number of Logical Unit assignments to this device within a partition.

Figure 289. Device Usage Counters (DVCUSCNT)

Channel Control Table (CHNTAB)

Label CHNTAB identifies the first byte of the Channel Control Table.

Bytes			
Dec	Hex	Label	Description
0	0	CHNTYPE	Channel Flag Byte
		NTOPCHN	X'80' Channel not operational or not present
			40 Reserved
		BLCKCHN	20 Block multiplexor channel
		MPXCHN	10 Byte multiplexor channel
			08 Reserved
			04 Reserved
		BRSTCHN	02 Byte multiplexor running in burst mode
		BMPXCHN	01 Byte multiplexor with burst devices attached
1	1	CHNTERR	Number of unit checks pending on this channel
2	2	CHNTFLG1	Processing Flag Byte
		CHNRSTRT	X'80' Channel must be restarted
		CHNRSDEV	40 At least one device busy during restart
			20 Reserved
			10 Reserved
		CHNISBSY	08 Channel is busy
			04 Reserved
			02 Reserved
			01 Reserved
3	3	CHNTFLG2	Channel ID (Channel No.)
4-7	4-7	CHNTPUBF	Address of first PUB on channel
8-11	8-B	CHNTPUBL	Address of next PUB to be started on channel
12-15	C-F	CHNTPUBB	Address of PUB that needs channel exclusively

Figure 290. Channel Control Table (CHNTAB)

Channel Queue Table (CHANQ)

Bytes 37-39 (X'25'-X'27') of the System Communication Region (SYSCOM) contain the address of the Channel Queue Table. Label CHANQ identifies the first byte of the Table. Each entry is fixed length and its layout is as follows:

Bytes			
Dec	Hex	Label	Description
0	0	CHQCHAIN	Index of next entry in free list or device queue. X'FF' indicates the last entry.
0-3	0-3	CHQCCBAD	Address of CCB/IORB associated with I/O request
4	4	REQID	PIK of service owner
5	5	CHQPROC	Logical processing flag required
		CHQDOINT	X'80' Interrupt not yet processed
		CHQDQUNC	40 Dequeue unconditional
		CHQNODEQ	20 Do not dequeue entry
		CHQPRCBF	10 Console buffering request
		CHQPROCF	08 OCCF request
		CHQDASFP	04 DASD file protect needed
		CHQFILE	02 SYSFIL on CKD device
		CHQSFFBA	01 SYSFIL on FBA device
6	6	CHQSLUB	System logical unit number associated with request X'FF' if this is a programmer unit (SYS000-SYS254)
7	7	TKREQID	Task ID (TID) of request owner
8	8	CHQCCSIO	SIO flag byte
		CHQCCACT	X'80' Device is running
		CHQCCALT	40 Alternate channel I/O
		CHQCCPRI	20 Primary channel I/O
		CHQCCLTE	10 Long time entry (Missing Interrupt Handler)
		CHQCCRUN	08 Condition Code 0
		CHQCCCSW	04 Condition Code 1
		CHQCCBSY	02 Condition Code 2
		CHQCCNOP	01 Condition Code 3
9	9	CHQCCBB1	Copied from byte 2 of CCB/IORB
10	A	CHQCCBB2	Copied from byte 3 of CCB/IORB
11	B	CHQCCBB3	Copied from byte 12 of CCB/IORB
12	C	CHQPFIX	Reserved for page fixing routine.
13-15	D-F	CHQPFIXL	Address of user specified or internal fixlist

Figure 291 (Part 1 of 2). Channel Queue Table (CHANQ)

Bytes			
Dec	Hex	Label	Description
16	10	CHQERRCT	Error retry count
17-18	11-12		Reserved
19	13	CHQPUBNO	PUB entry number
20	14	CHQFLG1	Flag byte
		CHQHQU	X'80' Unconditional request
		CHQHQA	40 Head queue request
		CHQCSBSY	20 Device busy status from PUB
		CHQCSQED	10 Device queued-in-error from PUB
		CHQDIDJA	08 Request was already accounted
			04 Reserved
		CHQFSIO2	02 Start on alternate channel only
		CHQFSIO1	01 Start on primary channel only
21	15	CHQGRP	Requestor flag
		CHQGROLT	X'80' OLTEP request
		CHQGRBTM	40 BTAM request
		CHQGRVTM	20 VTAM request (new interface)
			10 Reserved
		CHQGRRAS	08 RAS request
		CHQGRROK	04 Successful retry
			02 Reserved
			01 Reserved
22	16	CHQDEV	Device group indicator
		CHQDASD	X'80' CKD device or diskette
		CHQFBA	40 FBA device
		CHQTAPE	20 TAPE device
		CHQTP	10 TP (teleprocessing) device
		CHQCRT	08 2260 or 3277 device
		CHQURC	04 Unit record device
			02 Reserved
			01 Reserved
23	17	CHQIOINF	Delayed interrupt exit indicator
			X'00' Dispatcher (DISP)
			04 I/O initiator (INITRG)
			08 I/O interrupt handler (INTRTN)
			0C Error ignore routine (IGNORE)
			10 Cancel with code X'1A' (ERR1A)
			14 Reserved
			18 Dequeue routine (DEQUNCON)
			1C Post routine (PSTRESET)
			20 Emergency MSG writer (EMWINTRQ)
24	18	CHQCAWKY	Storage protect key
24-31	18-1F	CHQCSW	Accumulated status information from CSW

Figure 291 (Part 2 of 2). Channel Queue Table (CHANQ)

Command Control Block (CCB)

The CCB establishes communication between the problem program and physical IOCS. The CCB is two double words in length with eight major fields and an optional field, as shown in Figure 293 on page 581.

Count	Trans-mission Informa-tion	CSW Status Bits	Type Code and Logical Unit	Used by LIOCS or 3895 PIOCS	CCW Addr.	Used by Physical IOCS	CCW Address in CSW	Optional Sense CCW
0 1	2 3	4 5	6 7	8	9 11	12	13 15	16 23

Byte(s)	Description	
0-1 RESIDUAL COUNT	Number of bytes that have not been processed by the channel BTAM (370 mode only): Number of needed copy blocks	
2-3 TRANSMITTING INFORMATION between PIOCS and PROBLEM PROGRAM (Pr.Pr.)	Byte 2	set on by
	Bit 0: Traffic Bit (WAIT). (Note 5)	PIOCS
	Bit 1: End-of-File, (Note 2) PRT1-UCSB Parity Check.	PIOCS
	Bit 2: Irrecoverable I/O error was encountered.	PIOCS
	Bit 3: Prevent Cancelation on Irrecoverable I/O error.	Pr.Pr.
	Bit 4: Return DASD and/or DISKETTE Data Checks, Return 1017,1018 and 2671 (Paper Tape) errors, Return 5424/5425 not ready, Indicate action-type messages for DOC.	Pr.Pr.
	Bit 5: Post at Device End. (Note 5)	Pr.Pr.
	Bit 6: Return TAPE or DASD Read Data Check, (Note 3) Return 1018 or 2560 Data Check, (Note 6) Return 2520, 2540, 2560, 3881, or 5424/5425 Equipment check, Return 3504, 3505, or 3525 Permanent error, (Note 8) Return 3203, PRT1, or 5203 errors, Return 3895 errors. (Note 10)	Pr.Pr.
	Bit 7: User handles I/O errors. (Note 9)	Pr.Pr.
Note: Pr.Pr. stands for Problem Program		

Figure 292 (Part 1 of 5). Command Control Block (CCB)

Count	Trans-mission Information	CSW Status Bits	Type Code and Logical Unit	Used by LIOCS or 3895 PIOCS	CCW Addr.	Used by Physical IOCS	CCW Address in CSW	Optional Sense CCW
0 1	2 3	4 5	6 7	8	9 11	12	13 15	16 23

Byte(s)	Description	
2-3 (cont.)	Byte 3	set on by
	Bit 0: DASD Data Check in Count Area, 3330, 3340 or 3350 permanent error, 1287/1288 Data Check, 1419D SCU Not Operational, 3203, PRT1 or 5203 Print check/equipment check, 3540 Special Record transferred.	PIOCS
	Bit 1: DASD Track Overrun, 1419 Intervention required, 1287 Keyboard Correction in Journal Tape Mode, 1017 Broken Tape, PRT1 Print Quality/Equipment check	PIOCS
	Bit 2: DASD End-of-Cylinder, 1419, 1287 or 1288 Hopper Empty (Note 4) PRT1/2245 Line position error.(Note 7)	
	Bit 3: 1287, 2520, 2540 or 3881 Equipment Equipment Check, 2560, 3203, 5203, 5424/5425 Data/equipment check, 33504, 3505 or 3525 Permanent Error, (Note 8) TAPE Read Data check, DASD Data Check, 1017/1018 Data Check, PRT1 Print Check/Data Check, Diskette Data Check.	PIOCS
	Bit 4: CARD Unusual command sequence, DASD No Record Found, 1287/1288 Document Jam or Torn Tape, PRT1 UCSB, PRT1 UCSB Parity Check (Command retry), 5424/5425 not ready.	PIOCS
	Bit 5: user does not expect NO RECORD FOUND condition,	Pr.Pr.
Note: Pr.Pr. stands for Problem Program		

Figure 292 (Part 2 of 5). Command Control Block (CCB)

Count 0 1	Transmission Information 2 3	CSW Status Bits 4 5	Type Code and Logical Unit 6 7	Used by LIOCS or 3895 PIOCS 8	CCW Addr. 9 11	Used by Physical IOCS 12	CCW Address in CSW 13 15	Optional Sense CCW 16 23

Byte(s)	Description		
2-3 (cont.)	Byte 3		set on by
	Bit 6: PRINTER Carriage Channel 9 Overflow, DASD Verify error; 1287 Late Stacker select (Document Mode), 1288 End of Page.		PIOCS
	Bit 7: Channel Program is not retryable (Command Chain - Retry will be started from failing CCW).		Pr.Pr.
4-5 CSW STATUS BYTES	Byte 4	Byte 5 (Note 1)	
	Bits:	Bits:	
	0 (32): Attention	0 (40): Program Controlled Interruption	
	1 (33): Status Modifier	1 (41): Incorrect Length	
	2 (34): Control Unit End	2 (42): Program Check	
	3 (35): Busy	3 (43): Protection Check	
	4 (36): Channel End	4 (44): Channel Data Check	
	5 (37): Device End	5 (45): Channel Control Check	
	6 (38): Unit Check	6 (46): Interface Control Check	
	7 (39): Unit Exception	7 (47): Chaining Check	
6-7 TYPE code and LOGICAL UNIT	Byte 6		
	B'1x00x00x' = User-translated CCB		
	B'x1x0x00x' = BTAM CCB		
	B'0x10x00x' = System-translated CCB		
	B'xxx0100x' = CCB for physical unit		
	B'xxx00001' = CCB for program logical unit		
	B'xxx00000' = CCB for system logical unit		
Note: Pr.Pr. stands for Problem Program			

Figure 292 (Part 3 of 5). Command Control Block (CCB)

Count	Trans-mission Information	CSW Status Bits	Type Code and Logical Unit	Used by LIOCS or 3895 PIOCS	CCW Addr.	Used by Physical IOCS	CCW Address in CSW	Optional Sense CCW
0 1	2 3	4 5	6 7	8	9 11	12	13 15	16 23

Byte(s)	Description
6-7 TYPE code and LOGICAL UNIT (cont.)	Byte 7

	Hexadecimal representation of SYSnnn:
	SYSRDR = 00 SYSSLB = 07 SYSLUB = 0e-ff
	SYSIPT = 01 SYSRLB = 08 SYS000 = 00
	SYSPCH = 02 SYSUSE = 09 SYS001 = 01
	SYSLST = 03 SYSREC = 0A SYS002 = 02
	SYSLOG = 04 SYSCLB = 0B .
	SYSLNK = 05 SYSDMP = 0C .
	SYSRES = 06 SYSCAT = 0D SYS255 = FF

8 LIOCS Information	Buffer Offset: ASCII Input Tapes X'00' — X'63' ASCII Output Tapes Fixed X'00' Variable X'00' or X'04' Undefined X'00' 2501 Read ahead support X'80' (2501 Read ahead support is active) SNS task I/O request X'80' (I/O error on alternate channel) 3895 Error information (Note 10)
9-11 CCW ADDRESS	Virtual or real addr. of CCW associated with this CCB (Byte 6 bit 0 = 1 Address is a REAL address) (Byte 6 bit 0 = 0 Address is a VIRTUAL address)
12 PIOCS Information	X'80' CCB is used by ERP X'40' Channel Appendage Routine present X'20' Sense Information desired (Note 9) X'10' Reserved X'08' Reserved X'04' OLTEP Appendage available X'02' TAPE ERP Read Opposite Recovery in progress X'01' Reserved

Figure 292 (Part 4 of 5). Command Control Block (CCB)

Count	Transmission Information	CSW Status Bits	Type Code and Logical Unit	Used by LIOCS or 3895 PIOCS	CCW Addr.	Used by Physical IOCS	CCW Address in CSW	Optional Sense CCW
0 1	2 3	4 5	6 7	8	9 11	12	13 15	16 23

Byte(s)	Description
13-15 CCW ADDRESS from CSW	Address of CCW pointed to by CSW at Channel End, (Byte 6 bit 0 = 1 Address is real) (Byte 6 bit 0 = 0 Address is virtual) or address of the appendage routine.
OPTIONAL 16-23 Sense CCW	8 bytes appended to the CCB when Sense Information is desired.

Figure 292 (Part 5 of 5). Command Control Block (CCB)

Notes:

1. Bytes 4 and 5 contain the status bytes of the CSW (Bits 32-47). If byte 2, bit 5 is ON and Device End occurs as a separate interrupt, bytes 4 and 5 will contain the accumulated status information. A tape read-backward I/O operation reading into loadpoint will force the UNIT EXCEPTION (Bit 47) to be turned on and the unit check bit to be reset (assuming byte 2 bit 7 and byte 12 bit 2 are both off).

2. Indicates /* or /& statement read on SYSRDR or SYSIPT. Byte 4, bit 7 (Unit Exception) is also on.

3. DASD data checks on count not returned.

4. For 1255/1259/1270/1275/1419, disengage. For 1275/1419D, I/O error in external interrupt routine (Channel Data Check or Bus-out check).

5. The traffic bit (Byte 2, bit 0) is normally set on at channel end to signify that the I/O was completed. If byte 2, bit 5 has been set on, the traffic bit and bits 2 and 6 in byte 3 will be set on at device end. See also Note 1.

6. 1018 ERP does not support the Error Correction Function.

7. This error occurs as an equipment check, data check or FCB parity check. For 2245, this error occurs as a data check or FCB parity check.

8. Byte 2, bit 6 must be set on to allow you to accept 3504, 3505, 3525 permanent errors. This bit is forced on by LIOCS if the

user specified ERROPT for his input or output files. Byte 3, bit 3 is set on if a permanent error was encountered.

9. If User Error Routine is specified and the user needs the sense information to further process the error, byte 12, bit 2 must also be set. Otherwise, the supervisor error routine will clear off the status on return and the sense information is not available.

10. 3895 error codes are returned in CCB byte 8. Refer to 3895 Document Reader/Inscriber Machine and Programming Description for information on these error codes.

Input/Output Request Block (IORB)

The IORB establishes communication between the problem program and physical IOCS. The IORB consists of a fixed length part (24-bytes) and some optional extension fields each of it fixed length (4-bytes), which are all appended to each other.

Bytes			
Dec	Hex	Description	
0- 1	1- 1	Residual count, Number of Bytes which where not transferred by the channel	
2	2	Communication Byte 1 Set by Physical IOCS:	
		X'80' WAIT Bit, Traffic Bit	(Note 1)
		X'40' End-of-File on SYSRDR or SYSIPT, /* or /&	(Note 2)
		X'20' Irrecoverable I/O error encountered	
		Set by Problem Program	
		X'10' Prevent Cancelation in case of irrecoverable I/O Error	
		X'08' Reserved	
		X'04' User wants to be posted at Device End	(Note 1)
		X'02' Reserved	
		X'01' Skip system error Recovery (no Recovery Action)	
3	3	Communication Byte 2 Reserved for ERP return information.	
4	4	Device Status Information	(Note 3)
		X'80' Attention	
		X'40' Status modifier	
		X'20' Control unit end	
		X'10' Busy	
		X'08' Channel end	
		X'04' Device end	
		X'02' Unit check	
		X'01' Unit exception	
5	5	Channel Status Information	(Note 3)
		X'80' Program controlled Interrupt	
		X'40' Incorrect length	
		X'20' Program check	
		X'10' Protection check	
		X'08' Channel data check	
		X'08' Channel control check	
		X'02' Interface control check	
		X'01' Channel Chaining check	

Figure 293 (Part 1 of 3). Input/Output Request Block (IORB)

Bytes Dec	 Hex	Description
6	6	IORB and device identification Information X'80' Reserved X'40' Reserved X'20' Copied IORB (370 mode only) X'10' Reserved X'08' Device is identified by PUB entry number X'04' Control Block is an IORB X'02' Reserved X'01' Device is identified by Programmer Logical Unit
7	7	LUB or PUB entry number in the appropriate table

Byte 6 bit 4 off + 7 off		Byte 6 Bit 4 off 7 on	Byte Bit 4 on
SYSRDR=00	SYSRLB=08	SYS000=00	PUB entry No.
SYSIPT=01	SYSUSE=09	SYS001=01	00
SYSPCH=02	SYSREC=0A	SYS002=02	.
SYSLST=03	SYSCLB=0B	.	.
SYSLOG=04	SYSDMP=0C	.	.
SYSLNK=05	SYSCAT=0D	.	.
SYSRES=06	SYSLUB=0E-FF	.	.
SYSSLB=07		SYS255=FF	FF

Bytes Dec	 Hex	Description
8	8	Reserved for Logical Input Output Control System (LIOCS)
9-11	9- B	Virtual address of the CCW associated with this IORB
12	C	Reserved for physical Input Output Control System (PIOCS) X'80' IORB is used by Error Recovery Procedure X'40' Reserved X'20' This IORB has an extension X'10' Reserved X'08' Reserved X'04' Reserved X'02' Tape ERP read opposite Recovery in progress X'01' Reserved
13-15	D- F	Address+8 of last CCW that was executed

Figure 293 (Part 2 of 3). Input/Output Request Block (IORB)

Bytes		
Dec	Hex	Description
16	10	Fix Flag
		X'80' Fix List is already in compressed format (Each page to be fixed for Channel Program execution is covered only once within the FIXLIST)
		X'40' All pages are FIXED (The user has already fixed all the pages need for channel program execution)
		X'20' Reserved
		X'10' Reserved
		X'08' Reserved
		X'04' Reserved
		X'02' Reserved
		X'01' Reserved
17–19	11–13	Address of FIXLIST
20–21	14–15	IORB Version identification code
22–23	16–17	Special processing flags set by LIOCS
		Bit 0 SYSFIL request for FBA Device
		Bits 1–15 Reserved
OPTIONAL		
24	18	Parameter ID:
		Bit 0 Identifies the last optional Parameter
		Bits 1–7 Parameter ID
		B'0000000' ECB ID
		B'xxxxxxX' Reserved
25–27	19–1B	Address portion of optional Parameter
•		•
•		•
•		•
		Parameter ID:
		Parameter ID:

Notes:

1. The WAIT Bit (byte 2, bit 0) is normally set on at Channel End to to signify that at least the data transfer is completed.
   If byte 2, bit 5, has been set on, the WAIT Bit is set at Device End.

2. Unit Exception (Byte 4, bit 7) is also turned on.

3. Bytes 4 and 5 contain the status bytes of CSW (Bits 32-47) which is always the accumulated status information received so far.

Figure 293 (Part 3 of 3). Input/Output Request Block (IORB)

Disk Information Block (DIB) Tables

DIB Table for CKD and DISKETTE
DIB for FBA Device
DIB Extension (DIBX) Table (required by FBA)

Disk Information Block (DIB) Table for CKD and DISKETTE

Bytes 96-97 (X'60' - X'61') of the Partition Communication Region contain the address of the DIB Table. Each DIB table for a partition comprises a number of single entries. There is one entry for each, SYSLNK (open information), SYSIN, SYSPCH and SYSLST. There are different formats of the DIB entries:

Current	Length of		End of Extent	Head No.		Max	Notify	Flag	
Record Address	Key	Data	Address	High	Low	Rec.	Rec.No.		
0 6	7	8 9	10 16	17	18	19	20 21	22	23

Byte(s)	Description
0 CURRENT RECORD ADDRESS	Specifies the disk address of the next sequential record. The format differs slightly depending on the file and the device. For CKD-devices: SYSIN : BBCCHHR, SYSPCH : BBCCHHR, SYSLST : BBCCHHR For DISKETTE-devices: SYSIN : 0000CHR, SYSPCH : 0000CHR, SYSLST : 0000CHR
7 KEY LENGTH	Always zero
8 DATA LENGTH	Data length of record to be processed For CKD-devices: SYSIN : X'0050' or X'0051', SYSPCH : X'0051', SYSLST : X'0078' For DISKETTE-devices: SYSIN : X'0000', SYSPCH : X'0000', SYSLST : X'0000'
10 END of EXTENT ADDRESS	Specifies the disk address of the last record within the given Extent. For CKD-devices: SYSIN : BBCCHHR, SYSPCH : BBCCHHR, SYSLST : BBCCHHR For DISKETTE-devices: SYSIN : 0000CHR, SYSPCH : 0000CHR, SYSLST : 0000CHR

Note: The DIB is initialized by Job Control with Extent Info. and updated by PIOCS on every I/O oper. to the appropriate device.

Figure 294 (Part 1 of 2). Disk Information Block Table (DIB) for CKD Devices and Diskette

Current Record Address	Length of Key	 Data	End of Extent Address	Head No. High	 Low	Max Rec.	Notify Rec.No.	Flag	
0 6	7	8 9	10 16	17	18	19	20 21	22	23

Byte(s)	Description
17 HIGHEST HEAD NO.	Highest head number accessible on this device
18 LOWEST HEAD NO.	Lowest head number accessible on this device
19 MAXIMUM NO. of RECORDS	Maximum number of records that fit on one track
20 NOTIFY RECORD NUMBER	This field specifies the number of records that the user wants to be checked at EOJ time of whether they still fit into the specified Extent (applicable for output only). This field is set by the JCL SET statement (RCLST or PCPCH). A warning message will be issued when this minimum number has been reached or exceeded during the previous JOB.
22 FLAG BYTE	Flag byte: X'40' Device with RPS feature
23 RESERVED	Not used

Note: The DIB is initialized by Job Control with Extent Info. and updated by PIOCS on every I/O oper. to the appropriate device.

Figure 294 (Part 2 of 2). Disk Information Block Table (DIB) for CKD Devices and Diskette

Disk Information Block Table (DIB) for FBA Device

Bytes Dec	 Hex	Label	Description
0-3	0-3	ULPBN	End address of extent. Upper limit of physical block number
4-7	4-7	CRPBN	Current address. Current physical block number
8-9	8-9	CIOFF	Offset of current record within control interval
10-11	A-B	LNGCI	Length of control intervals in bytes
12	C	PBPERCI	Number of physical blocks per control interval
13-15	D-F	PBUFFER	Pointer to data buffer
16	10	DIBFLAGS	X'80' DIB gate flag X'40' Task waiting for DIB X'20' Reserved X'10' Source begin readjustment required X'08' Reserved X'04' Force write out X'02' End of extent reached X'01' Buffer-in-use flag
17-19	11-13	PDIBX	Pointer to DIB extension (DIBX)
20-21	14-15	DIBRSCNT	Residual count for JCL message
22-23	16-17		Reserved

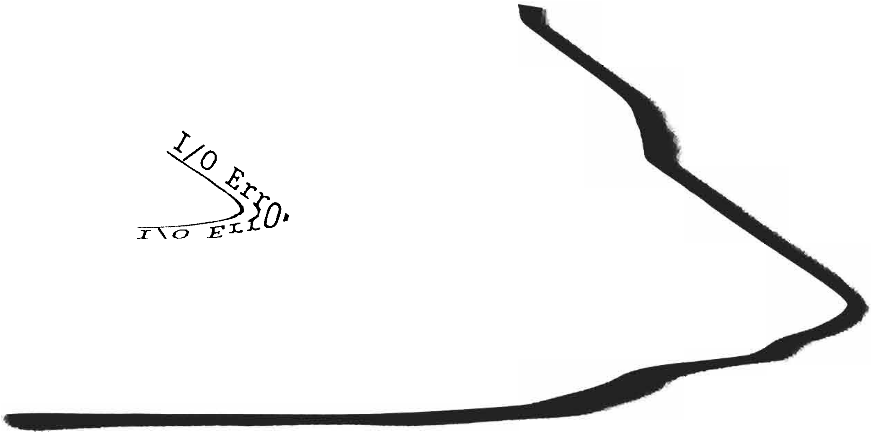
Figure 295. Disk Information Block Table (DIB) for FBA Devices

The FBA device also requires a DIB Extension (DIBX) Table.

Bytes Dec	 Hex	Description
0-23	0-17	Input Output Request Block (IORB)
24-31	18-1F	Fixlist first area
32-39	20-27	Fixlist second area
40-47	28-2F	DEFINE EXTENT CCW
48-55	30-37	LOCATE CCW
56-63	38-3F	READ/WRITE CCW
64-79	40-4F	DEFINE EXTENT Parameter list
80-87	50-57	LOCATE Parameter list

Figure 296. DIB Extension Table (DIBX) for FBA Devices

0 all system


I/O Erro.
I\O Erro

ERBLOC Area

The ERBLOC area is used as a common interface between all system components involved in I/O Error Recovery and/or Recording processing. Byte 0-3 (X'00 - X'03') of the System Communication Region contain a pointer to the ERBLOC area.

Bytes Dec	Hex	Label	Description
0-7	0-7	SVC5NM	Name of first/next ERP Transient to be fetched
8-11	8-B	YRETRY	Continuation address for retry I/O request (INITRG)
12-15	C-F	YIGNORE	Continuation address to ignore I/O error (IGNORE)
16-19	10-13	ACANCEL	Continuation address to cancel I/O request (ERR1A)
20-23	13-17	YERPEXIT	Common DSK/ERP return address (ERPEXIT)
24-75	18-4B	ERQ1	Area to pass recovery and recording information to the ERP. Its lay-out is the same as for a single error block, except for the 8-byte header (see note)
76-111	4C-6F	SNSSDAID	Sense data saved by SDAID
112-119	70-77	ERCHNOFT	Chain header offset table, used to address the following error chains
120-123	78-7B	RASERCHN	Address of first RAS error entry
124-127	7C-7F		Pointer to RAS TIB
128-131	80-83	ERPERCHN	Address of first ERP error entry
132-135	84-87		Pointer to ERP TIB
136-139	88-8B	DSKERCHN	Address of first DSK error entry
140-143	8C-8F		Pointer to DSK TIB
144-147	90-93	SNSERCHN	Address of first SNS error entry
148-151	94-97		Pointer to SNS TIB

Note:

- See Figure 298 on page 588.

Figure 297. ERBLOC Area

I/O Error Block

There is one I/O error block for each device. Field PBXERBLK in the PUBX contains a pointer to this block. An additional error block exists for some system tasks. The address of this block is contained in field TCBERBLK of the system task TCB.

Bytes Dec	 Hex	Label	Description
0-3	0-3	ERBLKPTR	Pointer to next error block in a chain or 0
4	4	ERBLKFLG	Flag byte
		HQERBLK	X'80' System task error block
		ALTCHANN	40 Error on alternate channel
		ERSNSDAV	20 Sense data available
		ERACTIVE	10 Error block active
		ERQUEUED	08 Error block is enqueued in some error chain
			04 Reserved
			02 Reserved
			01 Reserved
5	5	ERBLKFLG1	Flag byte
			X'80' Reserved
			40 Reserved
			20 Reserved
			10 Reserved
		NEEDSNS	08 Must be processed by SNS task
		NEEDDSK	04 Must be processed by DSK task
		NEEDERP	02 Must be processed by ERP task
		NEEDRAS	01 Must be processed by RAS task
6	6		Reserved
7	7	ERBLKSNL	Number of sense bytes
.... End of error block header			

Figure 298 (Part 1 of 3). I/O Error Recovery/Recording Block

Bytes			
Dec	Hex	Label	Description
.... Layout of UNIT CHECK entry			
8-15	8-F	ERRQCSW	CSW of I/O error
16-17	10-11	ERRQPUB	PUB pointer of affected device
18	12	ERRQFLG	Flag Byte
		TRUNRF	X'80' No record found on DASD
			40 Intervention required (set by ERP)
			20 Pass back error information (set by ERP)
		IGNERR	10 Channel program is not retryable (IGNORE)
		SUCCESS	08 Error successfully recovered (IGNORE)
		RTYERR	04 Channel program is retryable (RETRY)
			02 Reserved (RETRY)
		OCCUP	01 Error block is in use (set only for error block in ERBLOC area)
19	13	ERRQMSG	Message Code
20-23	14-17	ERRQSEK	Used for disk devices only CKD: Failing Seek address FBA: OS device type codes
24	18	ERRQCQPT	Index of channel queue entry X'FF' for unsolicited error
24-27	18-1B	ERRQCCB	CCB pointer (address is 0 if not available)
28-...	1C-...	ERRQSNS	Sense data
.... End of UNIT CHECK entry			
.... Layout of RECORDING entry			
8-11	8-B	ERQAEADR	SD record address
		ERQAEINF	SD record information
12	C	ERQAELEN	Length of SD record
13	D	ERQAETYP	Type of SD record
14	E	ERQAESW1	Record dependent switch 1
15	F	ERQAESW2	Record dependent switch 2
16-17	10-11	ERQAEPUB	PUB pointer of affected device
18	12	ERQAEFLG	Flag Byte X'80' SD record is TFIX-ed
			02 Must be 0 for recording info.
		OCCUP	01 Error block is in use (set only for error block in ERBLOC area)
19	13	ERQAEMSG	Contains X'AE' for Alternate Entry
20-23	14-17	ERQAETIB	TIB of requesting task
24-27	18-1B		Reserved
28-...	1C-...	ERQAECOM	Communication information
.................End of RECORDING entry.............................			

Figure 298 (Part 2 of 3). I/O Error Recovery/Recording Block

Bytes			
Dec	Hex	Label	Description
.................Layout for CHANNEL CHECK entry			
8-31	8-1F		see note

Note: Byte 8-31 same as ERPIB control block, Figure 309 on page 608.

Figure 298 (Part 3 of 3). I/O Error Recovery/Recording Block

PDTABB and PDTABA Tables

Bytes 126 and 127 (X'7E'-X'7F') of the partition communication region contain the address of the Paper Document processing Table. Label PDTABB identifies the first byte of the table. The tables are used for handling external interrupts on magnetic ink or optical character recognition devices.
PDTABBB contains six 8-byte entries; one for each line of the direct control feature on the system.

BYTE		AND INSTRUCTION		OWNER	DTF ADDRESS for MICR
DEC	HEX				
0	0	NI	PDSTAT+1,X'FE'	TID	Device on LINE 7
8	8	NI	PDSTAT+1,X'FD'	TID	Device on LINE 7
16	10	NI	PDSTAT+1,X'FB'	TID	Device on LINE 7
24	18	NI	PDSTAT+1,X'F7'	TID	Device on LINE 7
32	20	NI	PDSTAT+1,X'EF'	TID	Device on LINE 7
40	28	NI	PDSTAT+1,X'DF'	TID	Device on LINE 7

Figure 299 (Part 1 of 2). Table for MICR DTF Addresses Entries (PDTABB)

Bytes	Bits	Description
0 - 3		The NI instruction is executed in the External Signal Interrupt handler to turn off the external line status as soon as this line interrupt is being processed (any other External line signal remains affective). PDSTAT+1 is the fixed main STORAGE location 135 (X'87') and contains the External Signal codes that have not yet been processed in Bits 2-7.
	Bits	Description
	7	External signal from line 7
	6	External signal from line 6
	5	External signal from line 5
	4	External signal from line 4
	3	External signal from line 3
	2	External signal from line 2
4		Contains the PIK of the partition containing the DTF
5 - 7		Contain the address of the DTF table

Figure 299 (Part 2 of 2). Table for MICR DTF Addresses Entries (PDTABB)

Note: The contents of PDSTAT+1 (bits 2-7) is used to index a one byte entry in table PDTABA which, in turn indexes the DTF address entry, whithin table PDTABB of the external signal line with the currently highest priority. (Line 2 has highest, line 7 lowest priority).

Recorder File Table (RFTABLE)

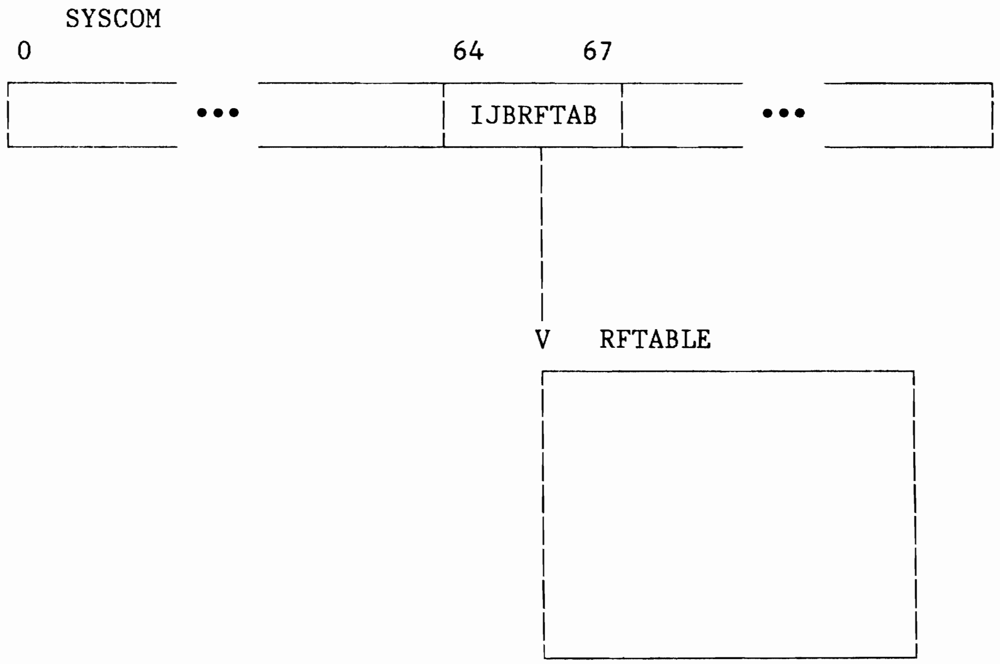


Figure 300. Recorder File Table Relationship

Bytes			
Dec	Hex	Label	Description
		RFTABLE	Label of Starting Address
0	0	RFFLAGS1	Flag byte 1
		RFFULL	X'80' File full
		RFRDE	40 RDE option included
		RFIPL	20 Initial IPL
		RFNO	10 RF=No option
		RFCREATE	08 File is to be created
		RFBUILT	04 File has been created
		RFONFBA	02 File on FBA device
		RFREADY	01 File ready
1	1	RFFLAGS2	Flag byte 2
		FFMSG	X'80' File full message request
		LTMSG	40 Last track message request
		IEMSG	20 I/O error message request
		DLMSG	10 Data lost message request
		RFEVA	08 EVA message request
		RFRTAOWN	04 File owned by RTA recorder
		RFPTAOWN	02 File owned by PTA recorder
		RFEREP	01 File being accessed by EREP
2	2	RFFLAGS3	Flag byte 3
		LTMISSUD	X'80' Last track msg issued once
		RECDERR	40 Error is to be recorded
		RECDSF	20 Short form record request
		RFIRULT	10 Individual records for unlabeled tapes
			08 Reserved
		RFHIOERR	04 Error in writing RFHEADER
		RFBOMT05	02 Exit to $$BOMT05 indicator for $$BOPEN
		RFBOMT01	01 Exit to $$BOMT01 indicator for $$BOPEN
3	3	RFFLAGS4	Flag byte 4
			X'80' - X'02' Reserved
		RFRNW	01 No record written
4	4	RFFLAGS5	Flag byte 5
			X'80' - X'02' Reserved
		RFFLG5BD	01 BOPEND called by OPEN
5	5	RFNOFN	N of N for records (low order 4 bits contain the number of records to be recorded and high order 4 bits contain the number of the record being recorded)
6	6	RFRECTYP	Record type code
7	7	RFREL	Release level code of VSE/Adv.Funct.
8	8	RFRDSW1	Record dependent bit 1
		RFTEMP	X'40' Temporary error

Figure 301 (Part 1 of 2). Recorder File Table (RFTABLE)

Bytes			
Dec	Hex	Label	Description
9	9	RFRDSW2	Record dependent bit 2
10 - 11	A - B	RFBUFLG	Length of data buffer (FBA)
.... CKD Device Related Information			
12 - 13	C - D	RFMCONST	Multiplier for track balance
14 - 15	E - F	RFDCONST	Divisor for track balance
16 - 17	10 - 11	RFOCONST	Overhead for track balance
18 - 19	12 - 13	RFRECLEN	Length of record
20	14	RFRDSW3	Record dependent switch 3
21 - 23	15 - 17		Reserved
24 - 27	18 - 1B	RFRECADR	Address of record
28 - 34	1C - 22	RFSEEK	Work area for seek addr.BBCCHHR
28 - 29	1C - 1D	RFSEEKBB	BB portion of seek
30 - 31	1E - 1F	RFSEEKCC	CC portion of seek
32 - 33	20 - 21	RFSEEKHH	HH portion of seek
34	22	RFSEEKR	R portion of seek
35	23	RFEREPK	Key of EREP partition
36 - 39	24 - 27	RFHDRCH	SYSREC cylinder/head
36 - 37	24 - 25	RFHDRCYL	Cyl. address of file start
38 - 39	26 - 27	RFHDRTRK	Head address of file start
.... End of CKD Device Related Information			
.... FBA Device Related Information			
12 - 15	C - F	RFBUFAD	Address of data buffer
16 - 17	10 - 11	RFNAVR	Displacement of next available RDF in buffer (FBA)
18 - 19	12 - 13	RFRECLEN	Length of record
20	14	RFRDSW3	Record dependent switch 3
21 - 23	15 - 17		Reserved
24 - 27	18 - 1B	RFRECADR	Address of record
28 - 31	1C - 1F	RFCUBL	Work area for block number
32 - 34	20 - 22		Reserved
35	23	RFEREPK	Key of EREP partition
36 - 39	24 - 27	RFHDRBL	SYSREC block number
.... End of FBA Device Related Information			
40 - 41	28 - 29	RFCHMAP	Map of supported channels
42 - 49	2A - 31	RFCHIDC	Channel ID codes
50	32	RFRDSW0	Record dependent switch 0
51	33		Reserved
52 - 55	34 - 37	RFEXIT	Exit phase name or exit address
56	38	RFEVARTH	EVA read threshold
57	39	RFEVAWTH	EVA write threshold
58 - 59	3A - 3B	RFP2ENTL	Length of PUB2 table
60 - 63	3C - 3F	RFP2ENT	Address of PUB2 table
64 - ...	40 - ...	RFP2ITAB	PUB2 index table (see Note)

Note: Two bytes are generated for each PUB2 index entry. See also Figure 278 on page 564.

Figure 301 (Part 2 of 2). Recorder File Table (RFTABLE)

CONSOLE BUFFER TABLE (CBTAB)

Label CBTAB identifies the first byte of the Console Buffer Table. Label CBNEXT points to the next free entry within this table.

Bytes			
Dec	Hex	Label	Description
0-7	0-7	CBCCW	CCW: Command code, chain byte and count have been copied from the user's CCW. The data address is always the addr. of CBDATA (byte 24-103 see below).
8-23	8-17	CBCCB	CCB: An area in CCB format whose CCW address field always points to CBCCW (see byte 0-7 above).
24-103	18-67	CBDATA	Console Buffer: An output area in which the users users data is kept
⋮	⋮	⋮	⋮

Figure 302. Console Buffering Table (CBTAB)

CRT AREAS (CRTTAB, CRTSAV)

CRT Constant Table (CRTTAB)
CRT Save Area (CRTSAV)

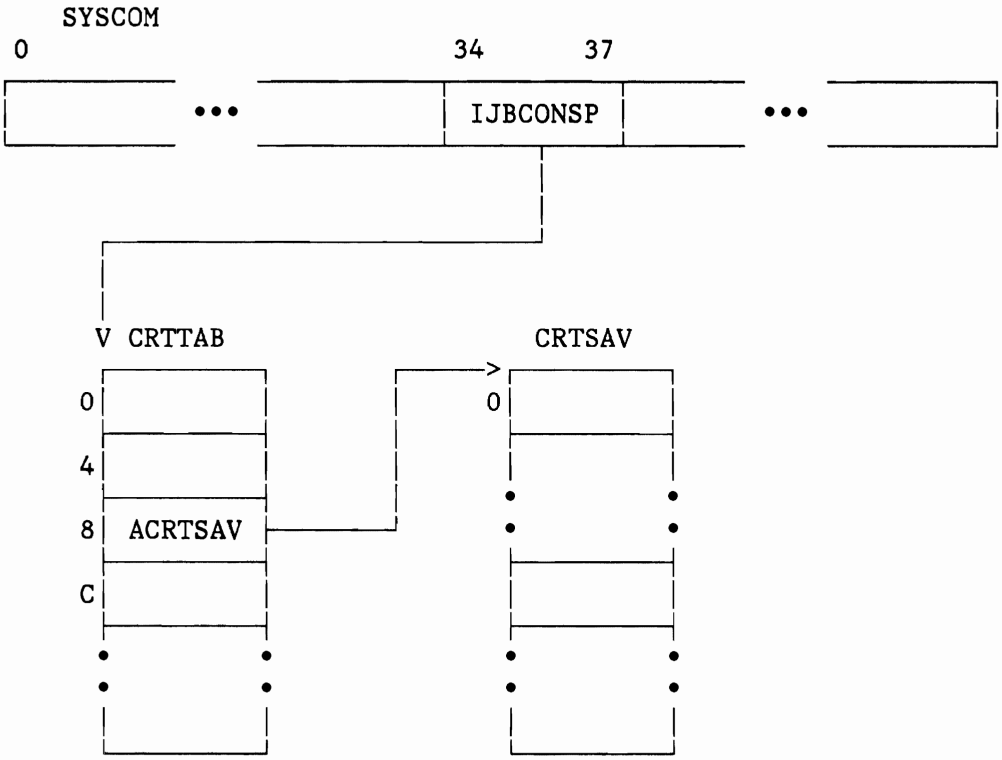


Figure 303. Relationship of CRT Areas

CRT Constant Table

Bytes			
Dec	Hex	Label	Description
0-7	0-7	CRTNAME	Name of CRT routine
8-11	8-B	ACRTSAV	Address of CRT save area (CRTSAV)
8	8	CRTNAM1	Phase Identifier Last character of phase that is to regain control after Attention Interrupt or I/O error are processed.
8	8	SENSEBT	Dummy sense byte
			X'80' Command reject
			40 Operator intervention required
			20 Reserved
			10 Equipment check
			08-02 Reserved
			01 Operation check
12-15	C-F	ACRTTRNS	Address of C-Transient area (CRTTRNS)
16-19	10-13	AATTNINT	Address of Attention interface rout.
20-23	14-17	ACRTUNPS	Address of CRT deactivation routine
24-27	18-1B	ACRTNWSO	Service owner of CRT
28	1C	CRTFLG1	Flag byte 1
		CRTERPBT	X'80' ERP message
		CRTUNITC	40 Unit check for CRT SYSLOG requ.
		CRTFETCH	20 Fetch of $$BOCRTK is in progr.
		CRTATTH	10 Device end simulated
		CRTERADR	08 Validation error
		CRTREDSP	04 Redisplay in progress
		CRTERR	02 CRT I/O error
		CRTBUSY	01 CRT busy
29	1D	CRTFLG2	Flag byte 2
			X'80' Reserved
		CRTSENS	40 Sense Byte (see byte 8) was set up by CRT
		CRTEOJ	20 End of CRT routines
		CRTDATRD	10 Data already read
		CRTATTPD	08 Attention Interrupt pending
		CRTRQPD	04 Request pending
		CRTATTRQ	02 Attention request being handled
		CRTEOJO	01 EOJ on CRT
30-35	1E-23	CRTEINF	CRT error information
36-39	24-27	AHCFIOMD	Entry address of I/O module for hardcopy file access

Figure 304. CRT Constant Table (CRTTAB)

CRT Save Area (CRTSAV)

Used to store control information for CRT system task processing.

Bytes			
Dec	Hex	Label	Description
0	0	CRTSAV	CRT save area - Set on doubleword boundary
0-7	0-7	SAVOLDP	Save area for old SVC PSW
8-11	8-B	ACRTSAVA	Address of problem program save area
12-55	C-34	CSAVEAR	Channel scheduler save area
12-15	C-F	CRTSV1 CCBSAVAR	save area for register 1 CCB address
16-19	10-13	CRTSV2	save area for register 2
20-23	14-17	CRTSV3	save area for register 3
24-27	18-1B	CRTSV4	save area for register 4
28-31	1C-1F	CRTSV5	save area for register 5
32-35	20-23	CRTSV6	save area for register 6
36-39	24-27	CRTSV7 CRTPSWM	save area for register 7 Save area for CRT system mask
40-43	28-2B	CRTIOSB	save area for IOS base register
44-47	2C-2F	CRTINTER	Address of I/O interrupt routine
48	30	CRTCCBB1	save CCB communication byte 1
49	31	CRTCCBB2	save CCB communication byte 2
50	32	CRTCCBB3	save CCB communication byte 3
51	33	CRTATTRB	Message attribute byte
Constants needed for CCW processing			
56-63	38-3F	CRTCCW0	CCW for write screen control char.s
64-71	40-47	CRTCCW	CCW built by CRT routines
72-79	48-4F	CRTCCB	CCB modified by CRT routines
80-83	50-53		CCW address
84	54		Flag byte
85-87	55-57		CSW CCW address
88-95	58-5F		CCW
96-99	60-63	CRTSNSI	CRT sense information
100-103	64-67	ASUPSAVA	Address of SUP system task save area
104-107	68-6B	CRTNEXT	Next CCW to process
108-111	6C-6F	ACTLCCW	Address of actual CCW
112-115	70-73	CONTCCW	Address of cont. CCW
116-117	74-75	CONTRDSV	Save byte count of cont. CCW
118	76	ATTLENG	Length of attention input
119	77	CRTNAM2	Save area for CRT char. in error case
120-121	78-79	CRTUTID	TID of task requesting CRT
122-123	7A-7B	CRTUPIK	PIK of task requesting CRT

Figure 305 (Part 1 of 4). Layout of CRT Task Save Area (CRTSAV)

Bytes			
Dec	Hex	Label	Description
Constants needed for hard copy processing			
124	7C	CRTHCPIK	Translation PIK for cont.-lines
125	7D	CRTFLGHC	Flags for Hard copy file (HC)
		CRTHCOPN	X'80' HC opened
		CRTHCOVR	X'40' HC in overlay mode
		CRTHCWRN	X'20' Warning (2 tracks left) sent
		CRTIPL	X'10' HC IPL switch
		HFTOOPEN	X'08' HC must be created
		HFEQUNO	X'04' HC not in use
		HCERR	X'02' HC has unrecoverable error
		HCINCL	X'01' Incorrect length during HC disk I/O
126	7E	PRTLOCK	Lock for PRINTLOG function X'00' - open, X'FF' - closed
127	7F	HCFLG	Flags for Hard copy file support
		OVERLAY	X'80' HCF in overlay mode
		PRINTLOG	X'40' PRINTLOG no select active
		HCINCERR	X'20' Inconsistent state in HC-supp.
		NOTCMPLT	X'10' HC file not yet full
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
HCFCB extension			
128-131	80-83	HCFCBWRT	Address of write HCFCB
132-135	84-87	HCFCBHDR	Pointer to HCFCB for write header
136-137	88-89	HCBOWNER	Owner of HC file
138	8A	HCFDEVTP	Device type (GETVCE output)
139	8B		Reserved
140-141	8C-8D	HCFBLKLN	Physical block length
142-149	8E-95	CWRPDADR	Addr. of last 'print logged' HCF rec.
150-155	96-9B	CWARNSKA	Disk address of warning message in HCF overlay mode
156-163	9C-A3	IPLDADR	Address of IPL-record on HCF
164-165	A4-A5	HCMSGLNG	Length of message 3277
166-167	A6-A7	HCFNRTR	Tracks/Cylinder
168-169	A8-A9	HCFNRBLK	Number of physical records/track
		HCFBLFBA	Block length of FBA device
170-177	AA-B1	HCCSW	CSW without 1st byte
178-201	B2-C9	HCSNS	HC file sense bytes

Figure 305 (Part 2 of 4). Layout of CRT Task Save Area (CRTSAV)

Bytes			
Dec	Hex	Label	Description
Constants used by CRT-redisplay feature			
204-205	CC-CD	PARTRED	Current partition redisplaying
206-207	CE-CF	PARTRED1	Partition id unchecked
208	D0	OCCFLG	Current OCCF options redisplay
209	D1	OCCFLG1	OCCF options specification unchecked
210-211	D2-D3	MSGACOO	Residual lines on screen
212-215	D4-D7	SCREENAD	Address of screen buffer save area
216-219	D8-DB	AHCFCBRD	Address of HCFCB for redisplay
220-221	DC-DD	LINEAL1	Line count 1. screen line - all msg.
222-223	DE-DF	LINECOA	Actual line counter
224-225	E0-E1	LINEPA1	Line count 1. screen line - partition
226-227	E2-E3	LINECOP	Actual line counter for selection
228-229	E4-E5	LINECNT	Line count indicated by command
230-231	E6-E7	LINECNT1	Line count unchecked command
232	E8	DISPF	Display flag
		BW	X'80' Actual reading is backward
		OCCFPAR	X'40' OCCF options specified
			X'20' Reserved
			X'10' Reserved
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
233	E9	REDISFLG	Communication redisplay routines
		SCRSAVE	X'80' Save current display
		SCRREST	X'40' Restore current display
		SCRRET	X'20' Return to start point
		SCRFW	X'10' Forward redisplaying
		PARCHG	X'08' Partition changed
		DISPCNT	X'04' Display content of part. line
		BYPSCOM	X'02' Bypass command checking
			X'01' Reserved
234	EA	FLG1	Communication byte command checking
		NOFRST	X'80' No first parameter indicated
		PARTPAR	X'40' partition parameter indicated
		DIRPAR	X'20' Direction parameter indicated
		RETURPAR	X'10' Return parameter indicated
		COUNTPAR	X'08' Count parameter indicated
		NOSEC	X'04' No second parameter indicated
		SCRFW1	X'02' Forward redisplay indicated
		ERRRET	X'01' Error return indicator
Temp. save area for SCT-pointer and screen buffer addr. used by $$BOCRTC and $$BOCRTD			
236-239	EC-EF	R3SAV	Save area for register 3
240-243	F0-F3	RDSAV	Save area for register D

Figure 305 (Part 3 of 4). Layout of CRT Task Save Area (CRTSAV)

Bytes Dec	 Hex	Label	Description
Constants needed for screen management			
244-247	F4-F7	SEGVAL0	Auto. del. default for 3277
248-251	F8-FB	SEGVAL1	K-command default S/125
252	FC	ACTCCW	Actual CCW indicator for screen cmd.
253	FD	MSGIND	Message indicator in HEX
254-257	FE-101	SEGVAL2	Work segment value
258-259	102-103		Reserved
260-263	104-107	CRTPOS1	Position avail.for data in curr.line
264	108	CRTPOS2	
265	109	CRTPOS3	
266-282	10A-11A	DELTAB	Deletion table for ASY OC
283	11B		End of deletion table (X'0F)
284-286	11C-11E	POWERCUU	For PGO commands CUU is stored
287	11F		Reserved
288-259	120-127	CRTCCWS	CCW save area
End of Constants needed for screen management			
296-319	128-13F		Interphase communication flags
322-418	142-1A2	AUXTAB	Auxiliary screen description table
419-503	1A3-1F7	IOAREA	Hard copy file I/O area
504-623	1F8-26F	PRINTSNS	Support for 3284/86/87 printer
.... Layout for 3277 screen			
624-748	270-2EC	SCRNCTL	Screen control table
749-2619	2ED-A3B	SCRIMG	Buffer for screen image
2620-2667	A3C-A6B	CRTBUAD	Device buffer line addresses
2668-2749	A6C-ABD	TABASE	Device buffer line offsets
2750-3078	ABE-C06	BLKLNE	Line frames
3079	C07		Reserved
3080-3143	C08-C47	CRTMVCSA	Move routine save area

Figure 305 (Part 4 of 4). Layout of CRT Task Save Area (CRTSAV)

MACHINE AND CHANNEL CHECK CONTROL BLOCKS

RAS Linkage Area (RASLINK)
RAS Monitor Table (RASTAB)
Error Recovery Procedure Information Block (ERPIB)

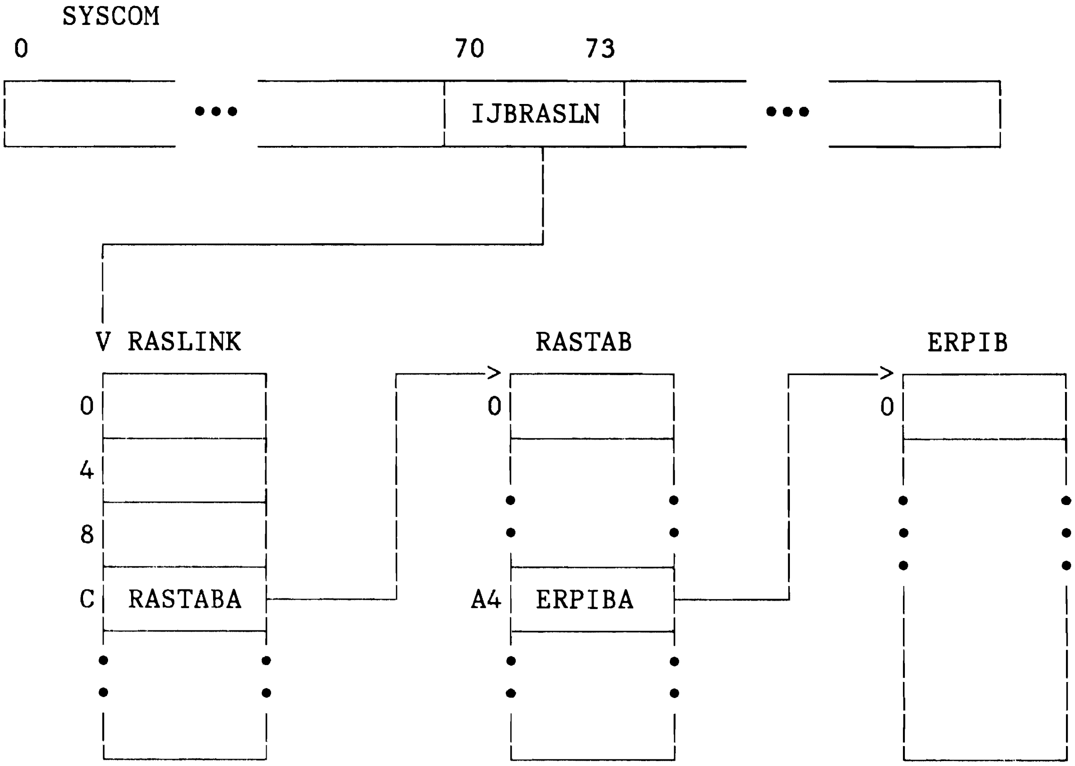


Figure 306. Machine/Channel Check Control Block Relationship

RAS Linkage Area (RASLINK)

Bytes Dec	Hex	Label	Description
0-3	0-3	CPUIDW1	First part of CPUID field
4-7	4-7	CPUIDW2	Second part of CPUID field
5	5	CPUID	Model number in CPUID field
6	6	RASMCELL	Length of machine check extended logout area
8	8	RASDMC	Damaged channel ID
9	9	RASFLAGS	RAS flag byte
		RASACT	X'80' RAS task activated
		RASMCACT	40 Machine check handling
		RASCCACT	20 Channel check handling
		RASEMGEX	10 Emergency handling
		RASSTERM	08 System termination
			04 Reserved
		RASNORTY	02 Retry not possible
		RTAIOA	01 RAS task I/O active
10	A	MCFLAGS	Machine check flags
		MCHARD	X'04' Hard machine check
11	B	RASRSFLG	RAS recording status flag
		RASNOFCH	X'80' Fetch of R-transient fails
			40 Reserved
		RASNOMSG	20 Unrecoverable channel check on SYSLOG
			10 Reserved
			08 Reserved
		RASBTDEQ	04 BTAM dequeue request
		RASMSGRT	02 Return from RAS message writer
		RASMSGIO	01 RAS message I/O
12-15	C-F	RASTABA	Address of RAS monitor table (RASTAB)
16-19	10-13	RASBASE	RAS base address
20-21	14-15	RASIMOD	Internal model number
22-23	16-17	RASIOELL	Length of I/O extended logout area
24-27	1C-1F	RASMCELA	Address of machine check extended logout area X'80' Indicates field contents not valid

Figure 307. RAS Linkage Area (RASLINK)

RAS Monitor Table (RASTAB)

Bytes			
Dec	Hex	Label	Description
0–3	0–3	LD00SLOT	$$RAST00 communication bytes
4–7	4–7	LD01SLOT	$$RAST01 communication bytes
8–11	8–B	LD02SLOT	$$RAST02 communication bytes
12–15	C–F	LD03SLOT	$$RAST03 communication bytes
16–19	10–13	LD04SLOT	$$RAST04 communication bytes
20–23	14–17	LD05SLOT	$$RAST05 communication bytes
24–27	18–1B	LD06SLOT	$$RAST06 communication bytes
28–31	1C–1F	LD07SLOT	$$RAST07 communication bytes
32–35	20–23	LD08SLOT	$$RAST08 communication bytes
36–39	24–27	LD09SLOT	$$RAST09 communication bytes
40–43	28–2B	LD10SLOT	$$RAST10 communication bytes
44–47	2C–2F	LD11SLOT	$$RAST11 communication bytes
48–51	30–33	LD12SLOT	$$RAST12 communication bytes
52–55	34–37	LD13SLOT	$$RAST13 communication bytes
56–59	38–3B	LD14SLOT	$$RAST14 communication bytes
60–63	3C–3F	LD15SLOT	$$RAST15 communication bytes
64–67	40–43	LD16SLOT	$$RAST16 communication bytes
68–71	44–47	LD17SLOT	$$RAST17 communication bytes
72–75	48–4B	LD18SLOT	$$RAST18 communication bytes
76–99	4C–63		reserved
100–103	64–67	LD25SLOT	$$RAST25 communication bytes
104–115	68–77	RASCCB	RAS CCB
116–147	78–97	RASCCWS	RAS CCW chain
148–154	98–9E	RASEEK	Seek address of RAS seek
155	9F	RTAOWN	R-transient identifier
156–157	A0–A1	MCPIK	Index to PIB active at machine check time
158–159	A2–A3	MCTIK	Index to TIK active at machine check time
160–163	A4–A7	ERPIBA	Address of work ERPIB
164–167	A8–AB	CCENTADR	Address of channel check routine
168	AC	RTAID	Requestor ID for RTA I/O
		RASRECID	X'08' RAS recording request
		RASRTYID	X'04' Channel retry request
169	AD	ERPID	Return load index for WTOR
170–171	AE–AF	RASRES	Device address of SYSRES
172–173	B0–B1	RASREC	Device address of SYSREC
174–175	B2–B3	RASLOG	Device address of SYSLOG

Figure 308 (Part 1 of 3). RAS Monitor Table (RASTAB)

Bytes			
Dec	Hex	Label	Description
176-243	B4-F3	TRANSAV	RTA register save area, Register 0 to Register 15
244-307	F4-133	SYSREGS	RAS monitor register save area, Register 0 to Register 15
308-311	134-137	SUPLINK	Service routine address for RTA in RAS monitor
308	137	LINKFLAG	Flag byte indicating requested service
		RASLIO	X'80' Perform normal I/O
		RASLEMIO	40 Perform emergency I/O
		RASLFTCH	20 Fetch another transient
		RASLWAIT	10 Perform wait
		RASLPDEQ	08 Dequeue page frame
		RASLDEQ	04 Dequeue CCB/IORB
		RASLFREE	02 Free I/O extended logout area
		RASLTIME	01 Get timer value for RTA
		RASLEXIT	00 Exit from RAS transient
312-323	138-13F	HIR	Hardware instr. retry accumulator
312-313	138-139	HIRACNT	Accumulated HIR count
314-315	13A-13B	HIRLCNT	Threshold value for count
316-319	13C-13F	HIR1TME	Time of day for first error of group
320-323	140-143	HIRLTME	Time threshold value in timer units
324-335	144-14F	ECCMAIN	Main storage error accumulators
324-325	144-145	ECCACNT	Accumulated ECC count for main stor.
326-327	146-147	ECCLCNT	Threshold value for count
328-331	148-14B	ECC1TME	Time of day for first error of group
332-335	14C-14F	ECCLTME	Time threshold value in timer units
336	150	MCMODE	Hardware operation mode
337	151	BUFDEL	Count of buffers deleted
338	152	RASMSG1	RAS Message byte 1
		MTICLDMG	X'10' Clock and or timer damage
		MTIMDMG	08 Timer damage
		MECQUIET	04 Control storage ECC in quiet mode
		MPERFDEG	02 System performance degradation
		MEFLOVFL	01 EFL overflow

Figure 308 (Part 2 of 3). RAS Monitor Table (RASTAB)

Bytes Dec	Bytes Hex	Label	Description
339	153	RASMSG2	RAS Message byte 2
		MCLOKDMG	X'80' Clock damage, all modes quiet
		MLASTTR	40 Threshold on recorder file reached
		MPAGEDEL	20 Buffer pages deleted
		MHIR	10 Soft MCI disabled
		MECC	08 ECC MCI disabled
		MFILEFL	04 Recorder file full
		MUNRCIO	02 Error on recorder file
		MCRECOV	01 Successful recovery from machine check
340-341	154-155	RASIND	RAS indicators
		RASNODEQ	X'80' Page frame not dequeued
342-343	156-157		Reserved
344-347	158-15B	RASPFT	Page frame table pointer
348-356	15C-164	INTERSEG	Interface segment build area
348-350	15C-15E	ILOGADR	Address of logout
351	15F	INOFN	Sequence number: record one of n
352	160	ILOGL	Logout length in record one
353	161	IRECL	Total length of record one
354	162	NNOFN	Sequence number record n of n
355	163	NLOGL	Logout length in record n
356	164	NRECL	Total length of record n

Figure 308 (Part 3 of 3). RAS Monitor Table (RASTAB)

Error Recovery Procedure Information Block (ERPIB)

Bytes			
Dec	Hex	Label	Description
0-7	0-7	ERPIBCSW	Saved CSW
0	0	ERPIBSTC	ERPIB status codes
		ERPIBFRE	X'FE' Indicate free ERPIB
		ERPIBCNC	X'FD' Indicate task is to be canceled
		ERPIBCCR	X'FC' Indicate retry unsuccessful
		ERPIBCCS	X'FB' Indicate retry successful
0-3	0-3	ERPIBCCW	Address of failing CCW + 8
4	4	ERPIBST1	First status byte
5	5	ERPIBST2	Second status byte
6-7	6-7	ERPIBCNT	Residual count in CSW
8-11	8-B	ERPIBIOE	Pointer to corresponding I/O extended logout area
12	C		Reserved
13	D	ERPIBDMC	Damaged channel ID
14-15	E-F	ERPIBPUB	PUB address of failing device
16	10	ERPIBCQP	Channel queue pointer from the PUB
17	11	ERPIBRTC	RAS retry counter
18	12	ERPIBMSG	Message indicator
		ACTMSG	X'80' Wait for operator response
		CCDONE	40 Channel check handling complete
		CCNODEQ	20 PUB not queued in error
			10 Reserved
			08 Reserved
			04 Reserved
		RECCC	03 Recovered channel check
		ERRCC	02 Channel check
		HRDCC	01 Unrecoverable channel check
19	13	ERPIBREQ	Requestor ID
20	14	ERPIBFLG	Flag byte
		CCSIO	X'80' Channel check on SIO
		CCDAM	40 Channel damage
			20 Reserved
			10 Reserved
		CCREC	08 Record build or written
			04 Reserved
		CCDSK	02 Channel check on disk device
		CCSKM	01 Skip message writer
21-23	15-17	ERPIBESW	Extended CSW
24	18	ERPIBEND	X'FF' End of ERPIB

Figure 309. Error Recovery Procedure Information Block (ERPIB)

TRACK HOLD TABLE (THTAB)

Bytes 76-79 (X'4C' - X'4F') of the System Communication Region (SYSCOM) contain both, the free list pointer and the address of the Track Hold Table. Label THFLPTR identifies the free list pointer and label THTAB identifies the first byte of the table.
The halfword at THTAB-2 contains the total number of 16-byte entries comprising the track hold table.

Bytes			
Dec	Hex	Label	Description
0	0	THPTR	Index of next entry in the chain (forward pointer) X'FF' indicates last entry
1–3	1–3	THCCB	Address of CCB/IQRB
4–11	4–B	THTRK	CKD devices: Address of track in BBCCHH00 format FBA devices: Physical block numbers of first and
12	C	THBWPTR	Index of previous entry in the chain (backward pointer)
13	D	THFLG	Flag and count byte X'80' Another task is waiting for this track/block 40 First entry within a PUB chain 20 Reserved 10 Reserved
		THCTR	Bits 4–7: Number of concurrent holds – 1
14–15	E–F	THTID	Task ID of track/block owner

Figure 310. Track-Hold Table (THTAB)

FLAGS AND FUNCTION CODES

Task Status Flags and Resource Gates
SVC 107 (X'6B') Function Codes

TASK STATUS FLAGS

Type	Value	Name	Usage
S	55	RSGTBND	Gate for real space segment table
S	56	SPFIXBND	Gate for PFIX in SVA processing
S	57	PWSRVBND	Gate for usage of POWER service
S	58	GQMGBND	Gate for usage of LOG queue manager
S	59	G117BND	Gate for usage of LOG service
S	5A	NPGRBND	Gate for usage of LUB allocation services
S	5B	VIOBND	Gate for virtual I/O support
O	5C	CONDRDY	Flag for conditional ready state
S	5D	IUCVBND	Gate for IUCV support for VCNA
S	5E	G108BND	Gate for usage of SVC-6C
S	5F	SATBND	Gate for usage of stored assign.table
S	60	CRTSVBND	Gate for CRTSAV usage
S	61	HCFCBBND	Gate for HC-file control block usage
S	62	ERQBND	Gate for error queue entry
S	63	G133BND	Gate for XPCC processing
S	64	OCFBND	Gate for operator comm. facility
S	65	OREBND	Gate for operator request element
S	66	EOTBND	Gate for EOT routine
C	67	SCYBND	Gate for security task
C	68	LCKBND	Gate for LOCK file I/O
C	69	PGFXBND	Gate for page to be freed
S	6A	GSMBND	Gate for ALLOCATE processing
S	6B	THTABBND	Gate for track hold table
C	6C	SFILBND	Gate for SYSFIL I/O
S	6D	SGTVSBND	Gate for GETVIS SVA

Type: O = permanently opened gate
C = permanently closed gate
I = I/O chain with permanently closed gate
W = wait chain with permanently closed gate
P = partition chain with switchable gate,
P gates located in Partition Control Block (PCB)
S = system chain with switchable gate

Figure 311 (Part 1 of 2). Task Status Flags and Resource Gates

Type	Value	Name	Usage
S	6E	LQBND	Gate for security logger queue
S	6F	CBFBND	Gate for console buffers
C	70	MICRBND	Gate for MICR I/O
S	71	GETRBND	Gate for GETREAL processing
S	72	FDIRBND	Gate for program fetch directory
S	73	SEIZEBND	Gate for SEIZE to be freed
S	74	CILBND	Gate for CIL update
S	75	BUFBND	Gate for copy blocks
C	76	ICCFBND	Gate for ICCF high priority task
S	77	PFRBND	Gate for page frames
S	78	PFGBND	Gate for page frames (occupied by TFIX)
S	79	CHQBND	Gate for channel queue entry
S	7A	DIBBND	Gate for DIB access
S	7B	CCWBND	Gate for CCW translation
W	7C	TRKBND	Gate for track to be freed
W	7D	AVRBND	Gate for AVR processing
S	7E	G41BND	Gate for ENQ/DEQ processing
S	7F	G92BND	Gate for XECB processing
C	80	NOTACT	Flag for inactive tasks
C	80	SYSBND	Flag for inactive system tasks
S	81	LTABND	Gate for LTA use
I	82	WAITBND	Gate for ECB/XECB (I/O or TIMER or POST)
O	83	READY	Flag for ready to run state
S	84	IDRABND	Gate for program fetch IDRA (old gate)
S	84	FPGMBND	Gate for program fetch IDRA (new gate)
C	85	FETCHBND	Gate for program fetch processing
W	86	PGIOBND	Gate for page I/O
C	87	PMRBND	Gate for page fault processing
I	88	ENQBND	Gate for RCB to be freed
S	89	TERMBND	Gate for terminator processing
C	8A	PGINBND	Gate for page-in
S	8B	USEBND	Gate for LOCK/UNLOCK processing
C	8C	CNCLBND	Gate for subtask to be cancelled
S	8D	SSIDBND	Gate for subsystem id processing
W	8E	RURBND	Gate for LOCK to be freed
S	8F	EXNTBND	Gate for EXTENT processing
P	90	GTVBND	Gate for partition GETVIS
P	91	CDLBND	Gate for CDLOAD
P	92	PFXBND	Gate for PFIX

Type: O = permanently opened gate
C = permanently closed gate
I = I/O chain with permanently closed gate
W = wait chain with permanently closed gate
P = partition chain with switchable gate,
P gates located in Partition Control Block (PCB)
S = system chain with switchable gate

Figure 311 (Part 2 of 2). Task Status Flags and Resource Gates

SVC 107 (X'6B') FUNCTION CODES

MACRO	OPTION	FUNCTION CODE DEC/HEX		SERVICE CLASS	AUTHORIZATION
TREADY	LQ	00	00	A	LOG-TASK
TREADY	NO	01	01	A	IPL+LOG-TASK
TREADY	IO	02	02	A	KEY 0 PROGRAMS
TREADY	VTAM	03	03	A	VTAM
TREADY	CANCEL	04	04	A	VTAM+POW+ICCF
TREADY	VCANCEL	05	05	A	VTAM
GETFLD	SAVAR	06	06	B	CURR. TASK
GETFLD	PPSAVAR	07	07	B	CURR. TASK
GETFLD	LTAPTR	08	08	A	
GETFLD	CNCLCODE	09	09	A	
GETFLD	PIK	10	0A	A	
GETFLD	MAINTASK	11	0B	A	
GETFLD	VTAMOPEN	12	0C	A	VTAM
GETFLD	VTAMDISP	13	0D	A	VTAM
GETFLD	AOTPTR	14	0E	A	VTAM
MODFLD	SYSRESW	15	0F	A	KEY 0 PROGRAMS
MODFLD	CNCLCODE	16	10	A	VTAM+POWER+EOJ
MODFLD	VTAMOPEN	17	11	A	VTAM
MODFLD	VTAMDISP	18	12	A	VTAM
TREADY	START	19	13	A	JCL+POWER
TREADY	OC	20	14	A	JCL+POWER
TREADY	CANCEL	21	15	A	POWER
TSTOP	SYSBND,NO	22	16	C	SYSTEM-TASKS
TSTOP	SYSBND,YES	23	17	A	SYSTEM-TASKS
TSTOP	STOP	24	18	C	JCL
TSTOP	UNBATCH	25	19	C	JCL
GETFLD	CNCLALL	26	1A	B	TERMINATOR
GETFLD	ICCFPP	27	1B	A	ICCF
MODFLD	SAVAR	28	1C	B	IPL+EOJ
MODFLD	CNCLALL	29	1D	B	TERMINATOR
GETFLD	SYSRESW	30	1E	B	
GETFLD	ICCFRO	31	1F	A	ICCF
GETFLD	ACLOSE	32	20	B	CURR. TASK
GETFLD	STATUS	33	21	A	ICCF
MODFLD	ICCFPP	34	22	A	ICCF
MODFLD	ICCFRO	35	23	A	ICCF
MODFLD	ACLOSE	36	24	B	EOJ
GETFLD	NSUB	37	25	A	
GETFLD	CPUTIME	38	26	A	
MODFLD	VSAMOPEN	39	27	B	OPEN/CLOSE

Figure 312 (Part 1 of 3). SVC 107 (X'6B') Function Codes

MACRO	OPTION	FUNCTION CODE DEC/HEX		SERVICE CLASS	AUTHORIZATION
GETFLD	ABINPR	40	28	B	ICCF
TREADY	ICCF	41	29	A	ICCF
GETFLD	LTAACT	42	2A	A	
GETFLD	OPENSVA	43	2B	B	CURR. TASK
MODFLD	OPENSVA	44	2C	B	CURR. TASK
MODFLD	ICCFSVC	45	2D	B	ICCF
GETFLD	PAGEIN	46	2E	A	
GETFLD	PAGEOUT	47	2F	A	
GETFLD	TERMACT	48	30	A	ICCF
GETFLD	EOTACT	49	31	A	ICCF
GETFLD	PCEXIT	50	32	B	CURR. TASK
GETFLD	ITEXIT	51	33	B	CURR. TASK
GETFLD	CNCLCOD2	52	34	A	
GETFLD	OCEXIT	53	35	B	CURR. TASK
TREADY	OCCF	54	36	A	OCCF
RLOCK	CRTSAV	55	37	C	OCCF
RLOCK	HCFCB	56	38	C	OCCF
TREADY	CRTSAV	57	39	A	OCCF
TREADY	HCFCB	58	3A	A	OCCF
TREADY	ATTINT	59	3B	A	OCCF
TREADY	OCCFIO	60	3C	A	OCCF
GETFLD	OCCFACT	61	3D	A	OCCF
GETFLD	BALANCE	62	3E	A	BAM
GETFLD	SSFLAGS	63	3F	A	SYSTEM-TASKS
GETFLD	COMRGPTR	64	40	A	
GETFLD	OWNER	65	41	A	
SRCHFLD	CHNUNIT	66	42	A	
SRCHFLD	DEVTYPE	67	43	A	
DEVUSE	PU	68	44	A	
DEVREL	PU	69	45	A	
SENTER	LIBR	70	46	A	
SLEAVE	LIBR	71	47	B	SYSTEM-TASKS
VIO	POINT	72	48	B	
GETFLD	USECNT	73	49	A	
GETFLD	PUSECNT	74	4A	A	
GETFLD	MOUNTFLG	75	4B	A	
MODFLD	MOUNTFLG	76	4C	A	JCL
TREADY	POWER	77	4D	A	POWER
GETFLD	PUBXPTR	78	4E	A	
GETFLD	PCBPTR	79	4F	A	

Figure 312 (Part 2 of 3). SVC 107 (X'6B') Function Codes

MACRO	OPTION	FUNCTION CODE DEC/HEX		SERVICE CLASS	AUTHORIZATION
GETFLD	TCBPTR	80	50	A	
GETFLD	ABEXIT	81	51	B	CURR. TASK
GETFLD	MSECS	82	52	A	SYST.-TASKS+JCL
MODFLD	MSECS	83	53	A	SYST.-TASKS+JCL
VALID	READ	84	54	A	
VALID	WRITE	85	55	A	
GETFLD	VSAMOPEN	86	56	B	
MODFLD	PERBIT	87	57	B	SDAID
GETFLD	PU	88	58	A	
GETJA	PART	89	59	C	
MODFLD	RUNMODE	90	5A	A	SYSTEM
MODFLD	SASCOPE	91	5B	A	SYSTEM
MODFLD	PASCOPE	92	5C	A	SYSTEM+POWER+ OCCF
RLOCK	ALLOCR	93	5D	A	SYSTEM
RLOCK	RSGT	94	5E	A	SYSTEM
TREADY	ALLOCR	95	5F	A	SYSTEM
TREADY	RSGT	96	60	A	SYSTEM
MODFLD	LIBRSERV	97	61	A	KEY 0

Figure 312 (Part 3 of 3). SVC 107 (X'6B') Function Codes

CHAPTER 5: DIAGNOSTIC AIDS

This chapter contains:

FIXED STORAGE LOCATIONS IN PROCESSOR STORAGE (LOW CORE)

The allocation of the first 512 bytes of processor storage is standard for any IBM System/370 CPU or any IBM 4300 processor. Fixed storage locations 513-1024 (X'200'-X'3FF') have been assigned to contain standard VSE Supervisor information. In Figure 313 the use of the fixed storage locations in processor storage are shown.

HEX	Label	Description
0- 7		Restart PSW if restart is possible (SDAID, DEBUG...)
0- 3		Hard wait message codes (MCH, CCH, IPL), if any
0- 4		Device error message codes if I/O error, and SYSLOG device is also in error.
10- 13		In a system with ACF/VTAM, the address of the VTAM communications vector table (ATCVT)
14- 17	CRADDR	Addr.of Communications Region of act. part.(COMREG)
18- 1F	EXOLDP	External Old PSW
20- 27	SVOLDP	Supervisor call old PSW
21	SVOLDKEY	Location of SVC old PSW key
24- 27	SVOLDADR	Address in SVC old PSW
28- 2F	PCOLDP	Program check old PSW
29	PCOLDKEY	Location of PC old PSW key
2C- 2F	PCKADR	Address in PC old PSW
30- 37	MCOLDP	Machine check old PSW
38- 3F	IOOLDP	I/O old PSW
40- 47	CSW	Channel status word
41- 43	CCWDRS	CSW channel command word (CCW) address
44	DEVSTA	Device status in CSW
45	CHNSTA	Channel status in CSW
46- 47	CSWCNT	Residual count
48- 4B	CAW	Channel address word
4C- 4F		Job duration
50- 53	TIMER	Hardware timer - no longer used
54- 57		Time of day - no longer used
58- 5F	XTNPSW	External new PSW
60- 67	SCNPSW	Supervisor call new PSW
64- 67	SCNADR	Address in SVC new PSW
68- 6F	PCNPSW	Program check new PSW
70- 77	MCNPSW	Machine check new PSW
78- 7F	IONPSW	I/O new PSW
80- 83	ASYSCOM	Address of System Communication Region (SYSCOM)
84- 85	EXTINF	External interrupt information
86- 87	EXTINF	External interrupt code
88- 89	SVCINF	SVC interrupt information
8A- 8B	SVCINTC	SVC interrupt code
8C- 8D	PGMINF	Program check interrupt information
8E- 8F	PGMINTC	Program check interrupt code

Figure 313 (Part 1 of 5). Fixed Storage Locations in Processor Storage

HEX	Label	Description	Bit
90- 93	TRADDR	Address which caused a page fault	
94- 95	MONCLASS	Monitor class	
96- 9B		Reserved	
9C- 9F	MONCADR	Monitor call address field	
A0- A7	PCOLDPS	Saved program check old PSW	
A1	PCOLDKYS	Key in saved PC old PSW	
A4- A7	PCKADRS	Address in saved PC old PSW	
A8- AB		Target of STIDC instruction	
AC- AF	AIOEL	Extended I/O logout address	
B0- B3	EXCSW	Limited channel logout	
B0	ECSWDET	Error detector	
	ECSWSTAT	X'80' ECSW stored if bit=off	0
		X'40' Storage control unit (SCU) id	1
		X'20' dto.	2
	ECSWSCUS	X'10' SCU validity	3
	ECSWDCPU	X'08' Error detected by CPU	4
	ECSWDCHN	X'04' Error detected by channel	5
	ECSWDSCU	X'02' Error detected by SCU	6
	ECSWDSTO	X'01' Error detected by storage	7
B1	ECSWSRC	Source of error	
	ECSWSCPU	X'80' Source is CPU	8
	ECSWSCHN	X'40' Source is channel	9
	ECSWSSCU	X'20' Source is storage control	10
	ECSWSSTO	X'10' Source is storage	11
	ECSWSCNU	X'08' Source is control unit	12
		X'04' Reserved	13
		X'02' Reserved	14
	ECSWVLOG	X'01' Channel logout stored	15
B2	ECSWVAL	Field validity flag	
		X'80' Reserved	16
		X'40' Reserved	17
		X'20' Reserved	18
	ECSWVSEQ	X'10' Valid sequence code	19
	ECSWVUNS	X'08' Valid unit status	20
	ECSWVCAK	X'04' Valid CCW address and key	21
	ECSWVCHA	X'02' Valid channel address	22
	ECSWVUNA	X'01' Valid unit address	23

Figure 313 (Part 2 of 5). Fixed Storage Locations in Processor Storage

HEX	Label	Description	Bit
B3	ECSWTSC	Termination and sequence code	
		Bit 24 + 25 = Termination code	
		Interface disconnected - code: 00	
	ECSWTSER	X'80' Selective reset - code: 10	24
	ECSWTSSN	X'40' Stop, stack or normal term. - code: 01	25
	ECSWTSYR	X'C0' System reset - code: 11	
		X'20' Reserved	26
	ECSWTNOP	X'10' Interface inoperative	27
	ECSWTIOA	X'08' I/O error alert	28
		Bit 29 - 31 = Sequence code	
	ECSWTSC0	X'00' Error during TIO CLRIO - code: 000	
	ECSWTSC1	X'01' Command out status in error - code: 001	29
	ECSWTSC2	X'02' No data transfer - code: 010	30
	ECSWTSC3	X'03' Data transfer error - code: 011	
	ECSWTSC4	X'04' Command out not accepted - code: 100	31
	ECSWTSC5	X'05' Unpredicted data transfer - code: 101	
		X'06' Reserved - code: 110	
	ECSWTSC7	X'07' No other codes apply - code: 111	
B4- B7		Reserved	
B8- B9	IOINF	Saved I/O interrupt information	
BA- BB	CHNADR	I/O address	
BB	DEVADR	Device address	
BC- E7		Reserved	
E8- EF	MCIC	Machine check interruption code	
E8	MCICB00	MCIC byte 0	
	SDBIT	X'80' System damage	0
	PDBIT	X'40' Instruction processing damage	1
	SRBIT	X'20' System recovery	2
	TDBIT	X'10' Interval timer damage	3
	CDBIT	X'08' Timing facility damage	4
	EDBIT	X'04' External damage	5
		X'02' Unused	6
	DGBIT	X'01' Degradation	7
E9	MCICB01	MCIC byte 1	
	WABIT	X'80' Warning	8
		X'40' Unused	9
		X'20' Unused	10
		X'10' Unused	11
		X'08' Unused	12
		X'04' Unused	13
	BUBIT	X'02' Backed up	14
	DLBIT	X'01' Delayed	15

Figure 313 (Part 3 of 5). Fixed Storage Locations in Processor Storage

HEX	Label	Description	Bit
EA	MCICB02	MCIC byte 2	
	SEBIT	X'80' Storage error uncorrected	16
	SCBIT	X'40' Storage error corrected	17
	KEBIT	X'20' Storage key error uncorrected	18
		X'10' Unused	19
	WPBIT	X'08' PSW EMPW validity	20
	MSBIT	X'04' PSW mask and key validity	21
	PMBIT	X'02' PSW program mask and condition code val.	22
	IABIT	X'01' PSW instruction address validity	23
EB	MCICB03	MCIC byte 3	
	FABIT	X'80' Failing storage address validity	24
	RCBIT	X'40' Region code validity	25
	EDRBIT	X'20' External damage code validity	26
	FPBIT	X'10' Floating point register validity	27
	GRBIT	X'08' General register validity	28
	CRBIT	X'04' Control register validity	29
	LGBIT	X'02' Log-out validity	30
	STBIT	X'01' Storage logical validity	31
EC	MCICB04	MCIC byte 4 - unused Bit: 32-39	
ED	MCICB05	MCIC byte 5	
		X'80' Unused	40
		X'40' Unused	41
		X'20' Unused	42
		X'10' Unused	43
		X'08' Unused	44
		X'04' Unused	45
	CTBIT	X'02' CPU timer validity	46
	CCBIT	X'01' Clock comparator validity	47
EE- EF	MCELL	MC extended log-out length Bit: 48-63	
F0- F3		Reserved	
F4	EDRCODE	External damage reason code	
		X'80' Reserved	
		X'40' Reserved	
	ESRBIT	X'20' Secondary report	
		X'10' Channel not operational	
		X'08' Channel control failure	
		X'04' I/O instruction time-out	
		X'02' I/O interruption time-out	
		X'01' Reserved	
F5- F7		Reserved	
F8- FB	FSA	Failing storage address	
FC- FF	REGCODE	Region Code (model dependent)	
100-1FF	FLOGA	Fixed logout area	
100-15F		Store status or machine check save areas	
160-17F	FPRSAVE	Floating point registers 0 - 6	
180-1BF	GRSAVE	General registers 0 - F	
1C0-1FF	CRSAVE	Control registers 0 - F	
200-203	IJBPATCH	Address of patch area (see note)	

Figure 313 (Part 4 of 5). Fixed Storage Locations in Processor Storage

HEX	Label	Description
204-23D		CE patch area
23E-23F	CHNADRSA	CUU addr.from X'BA' at time system entered hard wait
240	SUPFLAG	Supervisor communication flag
		X'80' Reserved
	PMRINIT	X'40' Page manager initialized
	SUPNFIX	X'20' Supervisor pageable
	VMSYS	X'10' System running under VM
		X'08' Reserved
		X'04' Reserved
	KLLEDBT	X'02' BATCH deactivated
	TPBIT	X'01' TPIN in progress
241	SUPVFLAG	Supervisor internal flag
	JAACT	X'80' Job accounting active
	PBALACT	X'40' Partition balancing active
	TTIMEACT	X'20' Timer is active
		X'10' Reserved
		X'08' Reserved
	TTIMESET	X'04' Timer set
		X'02' Reserved
	TFREEPH	X'01' TFREE user phase area
242-243	RID	Routine identifier (RID)
244-247	ARUNTIME	Address of PCB which will be charged for accounting
248-24B	APIBTAB	Address of PIBAREA (PIB2TAB, PIBTAB)
24C-24F	ASYSPCB	Address of System PCB
250-253	ASCBATAB	Address of SCB address vector
254-257	SCBPTR	Address pf current SCB
258-25F	PSS	Partition selection string (PSS)
260-263	TCBPTR	Address of currently active Task Control Block (TCB)
264-267	TIBPTR	Addr.of curr. active Task Information Block (TIB)
268-26B	PIBPTR	Addr.of curr. active Part. Information Block (PIB)
26C-26F	PCBPTR	Addr.of curr. active Partition Control Block (PCB)
270-273	XXARPTR	Address of currently active DEBUG area
274-277	XXPARMAD	Address of DEBUG Parameter area
278-27B	AFLIH	Address of First Level Interrupt Handler (FLIH)
27C-27F	DISPAD	Address of Dispatcher
280-2BF	ERA	Save area for general registers 9 through 8
290-293	ERARD	Save area for general registers D
294-297	TINFSAVE	Save area for general registers E
298-29B	TINFSAVF	Save area for general registers F
29C-29F	TINFSAV0	Save area for general registers 0
2A0-2A3	TINFSAV1	Save area for general registers 1
2C0-2C3	ATIBATAB	Address of TIB table
2C4-2C7	APCBATAB	Address of Partition Priority Table (PPRTYOWN)
2C8-2C9	NPSQE	Number of available page frames
2CA-2CB	MINPSQE	Minimum number of reserved (unfixed) page frames
2CC-2CD	TINFRID	Save area for RID
2CE-2FF		Supervisor Level Identification information
300-3FF	SADUMPLA	Reserved area for Stand alone DUMP

Figure 313 (Part 5 of 5). Fixed Storage Locations in Processor Storage

Note:

Supervisor patch areas are provided for use by IBM programming support representatives. They use those areas if there is a need for installing a local fix to (usually a bypass of) a problem.

There is one 62-byte patch area within the supervisor at label IJBPATCH (X'200' in low core). The first four bytes of this area point to a 300-byte patch area in the high address range of virtual storage. The small area within the supervisor allows coding of a limited number of instructions without the need for a base register in operand addresses.

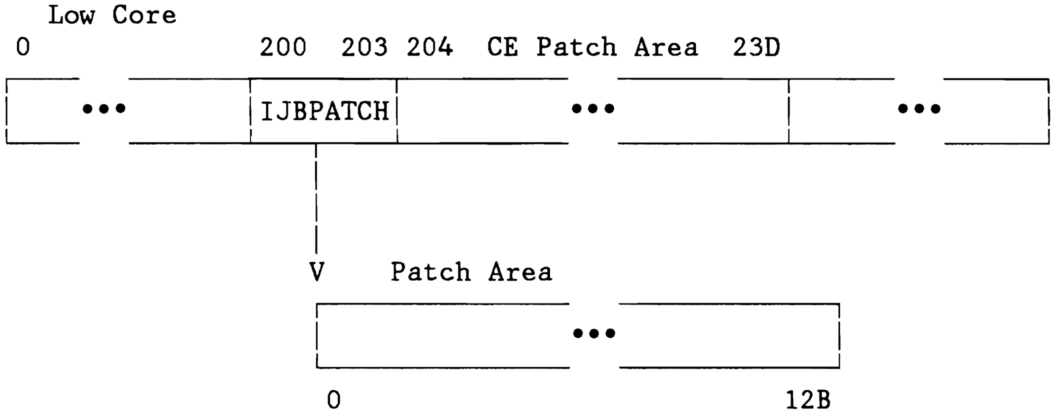


Figure 314. Patch Area Relationship

HARD WAIT CODES

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'C1'	X'00'	A,I,S	not used	Irrecoverable machine check.
X'C2'	X'00'	A,I,S	not used	Irrecoverable channel check during FETCH.
X'C3'	X'00'	A,I,S	not used	Irrecoverable channel check on paging channel.
X'C5'	X'00'	A,I,S	not used	No ECSW stored.
X'C7'	X'00'	A,I,S	not used	Channel failure; channel address invalid.
X'C8'	X'00'	A,I,S	not used	Channel failure on SYSLOG

Notes:

A	X'C1' - SYSREC recording unsuccessful (No record written)
I	X'C9' - SYSREC recording incomplete (Not all records written)
S	X'E2' - SYSREC recording successfully completed

Figure 315. MCH/CCH Wait Codes

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'07'	X'E6'	Channel	Unit or X'00'	IPL input/output error: • I/O error on SYSRES • I/O error on communication device • Equipment malfunction during STORE-CHANNEL-ID
X'C1'	X'E2'	not used	not used	Irrecoverable machine check
X'cc'	X'00'	X'0F'	X'D0'	Error during IPL. IPL canceled. (cc=cancel code)
X'F0'	X'C9'	X'F0'	X'F0'	See message 0I00
X'F0'	X'C9'	X'F0'	X'F2'	This code means that the requested supervisor cannot be loaded. (see message 0I03)
X'F0'	X'C9'	X'F0'	X'F6'	The device type of SYSRES can not be identified. The volume label (VOL1) or format-4 record contains invalid information. The pack was not initialized correctly.
X'F0'	X'C9'	X'F0'	X'F7'	See message 0I07
X'F0'	X'C9'	X'F0'	X'F8'	See message 0I08
X'F0'	X'C4'	X'F3'	X'F8'	See message 0D38
X'F0'	X'D1'	X'F5'	X'F0'	Unsupported SYSLOG device, see message 0J50

Figure 316. IPL Hard Wait Codes. For IPL Wait State Messages in low core refer to the VSE/Advanced Functions Message Manual.

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'08' to X'60'	X'C1' or X'C4'	Channel	Unit	Error recovery messages.

Figure 317. Device Error Recovery Wait Codes. For Error Recovery Messages refer to **OP**... messages in the VSE/Advanced Functions Message Manual.

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'62'	X'C5'	Not used	Not Used	SDAID output device became unready. Make printer ready and press the EXTERNAL INTERRUPT key.
X'00'	X'00'	X'00'	X'00'	SDAID stop on event. To continue, press the EXTERNAL INTERRUPT key.

Figure 318. SDAID Soft Wait Code. (Identified by EEEE in the address part of the WAIT PSW).

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'00'	X'00'	X'0C	X'CC'	No recovery possible from CRT errors.
X'00'	X'00'	X'0F'	X'ED'	System error condition (e.g.control block inconsistency). General Register 5 contains the address of the location where the System inconsistency was determined.
X'00'	X'00'	X'0F'	X'F1'	System error detected by the page manager.
X'00'	X'00'	X'0F'	X'F2'	Unused
X'00'	X'00'	X'0F'	X'F3'	Unused
X'00'	X'00'	X'0F'	X'F4'	$$A transient not found (the transient name can be found in ERBLOC).
X'00'	X'00'	X'0F'	X'F5'	TFIX count outside limits.
X'00'	X'00'	X'0F'	X'F6'	I/O error during update of the SLD.
X'00'	X'00'	X'0F'	X'F7'	No copy blocks available for BTAM appendage I/O request.

Figure 319 (Part 1 of 2). General Hard Wait Codes

Byte 0	Byte 1	Byte 2	Byte 3	Description
X'00'	X'00'	X'0F'	X'F8'	CRT phase not found.
X'00'	X'00'	X'0F'	X'F9'	Paging I/O error.
X'00'	X'00'	X'0F'	X'FA'	Translation specification exception.
X'00'	X'00'	X'0F'	X'FB'	Page fault in supervisor routine with identifier RID=X'00'.
X'00'	X'00'	X'0F'	X'FC'	Unused
X'00'	X'00'	X'0F'	X'FD'	Unused
X'00'	X'00'	X'0F'	X'FE'	I/O error during fetch from SYSLIB.
X'00'	X'00'	X'0F'	X'FF'	Program check in supervisor or or SDAID

Notes:

General Hard Wait Codes will be set by the VSE Supervisor or related routines.

Figure 319 (Part 2 of 2). General Hard Wait Codes

CANCEL CODE TO MESSAGE CODE CROSS REFERENCE

Cancel Code (Hex)	Message Code	Descriptive Part of Message (or Condition)
00	——	In all cases default value except those listed
08	0V16I	CANCEL request from subsystem
09	0V15I	CANCEL request from LIOCS
0A	0S21I	Processing error in access control
0B	0S20I	Access control violation
0C	0S19I	Execution failure in ICCF interactive partition
0D	0V13I	Program check in subsystem or appendage
0E	0V14I	Page fault in subsystem or appendage
0F	0P80I	Invalid 'read from/or write to' system file on FBA device
10	——	Normal EOJ
11	0V07I	No channel program translation for unsupported device
12	0V06I	Insufficient buffer space for channel program translation
13		reserved
14	0V04I	Page pool too small
15	0V02I	Page fault in disabled program
16	0V11I	Error in privately translated CCW
17	0S02I	(Same as 23 but causes dump because subtasks were attached when maintask issued CANCEL macro)
18	——	Eliminates cancel message when task issues DUMP macro
19	0P74I	I/O operator option
1A	0P73I	I/O Error
1B	0P82I	Channel failure
1C	0S14I	CANCEL ALL macro
1D	0S12I	Maintask termination
1E	0S13I	I/O error on lock file
1F	0P81I	CPU failure
20	0S03I	Program check
21	0S04I	Illegal SVC
22	0S05I	Phase not found
23	0S02I	Program request
24	0S01I	Operator intervention (cancel)
25	0P77I	Invalid address
26*	0P71I	SYSxxx not assigned (unassigned LUB code)

* If the CCB/IORB is unavailable, the logical unit is SYSxxx.

Figure 320 (Part 1 of 2). Cancel Code to Message Code Cross-Reference

Cancel Code (Hex)	Message Code	Descriptive Part of Message (or Condition)
27	0P70I	Undefined logical unit (invalid LUB code in CCB)
28	0S35I	Phase too long (does not fit in LTA or partition)
29	0P92I	Invalid Sub-library structure
2A	0V10I	I/O error on page data set
2B	0P84I	I/O error during fetch from private core image library
2C	0V09I	Illegal parameter passed by PHO routine
2D	0P881	Failing storage block (program cannot be executed)
2E	0S16I	Invalid resource request (possible deadlock)
2F	0V03I	More than 255 PFIX requests for 1 page
30	0P72I	Reading past /& statement (on SYSRDR or SYSIPT)
31		Reserved
32	0P76I	Invalid DASD address
33	0P79I	Invalid first CCW
34	0P93I	GETVIS space exhausted
35	0P85I	Job control open failure
36	0V08I	Program check or page fault in I/O appendage routine
37		Reserved
38	0V11I	Wrong privately translated CCW
39	0V12I	Invalid CCW chain for SYSLOG
3A	0V17I	Spool request out of sequence
40	0V95I	ACF/VTAM error (termination of task)
41	0V96I	ACF/VTAM error (invalid condition code)
42	0P86I	Violated DASD file protection
FF		Multiple cancel condition (see SYSLST for details)
xx	0P78I	Unrecognized cancel code
	0P83A*	Supervisor catalog failure
	0P87A*	IPL failure

* This cancel code is not significant in case of a supervisor catalog or IPL failure, because the System is placed in a WAIT state without any further processing by the Terminator.

Figure 320 (Part 2 of 2). Cancel Code to Message Code Cross-Reference

APPENDIXES

This publication contains appendixes as follows:

- Appendix A: Supervisor Generation

  A description of the supervisor generation macros and their functions and a list of globals set in the supervisor depending on parameters set in the supervisor generation macros.

- Appendix B: Macro Description

  A description of the internal VSE macros; these macros, mainly used by the VSE system and its components, perform a variety of functions within the system.

- Appendix C: Device Type Codes

  This table lists the device types and their VSE internal codes supported by VSE system.

- Appendix D: Supervisor Calls

  This table lists the supervisor calls (SVCs) supported by VSE system.

- Appendix E: Samples

  Track hold processing examples.

APPENDIX A. SUPERVISOR GENERATION MACROS AND GLOBAL SETTINGS

The supervisor is assembled with a series of macros that describe the installation's functional requirements and its configuration. At supervisor generation time, the supervisor generation macros are assembled into an object deck.

The following descriptions of supervisor generation macros show:

- Required generation macro sequence (as listed).
- Supervisor generation macro names.
- A brief description of what the generation macro does and which globals may be set.

The code generated by the assembler is a function of the generation macros described below and a group of inner macros which are called by these generation macros. The specific instructions assembled depend on the global settings which is finally a result of the options specified by the user. For a list of global settings refer to Figure 321 on page 635.

For a detailed description of the supervisor macros and their parameters refer to VSE/Advanced Functions System Generation.

SUPVR

The SUPVR generation macro describes the system environment:

ID	Supervisor identification character.
MICR	Support for magnetic ink or optical characters readers / sorters.
MODE	Which machine environment is to be supported.
NPARTS	How many partitions are to be supported.

The following globals are set dependent on the specified options:

```
MICR:   BG35   BG36
MODE:   BG370  BGVM
NPARTS: NPART  CGP(n)  P(n)
```

FOPT

The FOPT generation macro describes the functional supervisor options:

DASDSHR DASD sharing support.

FASTTR Fast CCW translation.

RPS Support for the rotational position sensing (RPS) capabilities.

TRKHLD Track/hold feature for DASD.

TTIME Timer support.

USERID Print supervisor ID at IPL completion.

The following globals are set dependent on the specified options:

DASDSHR:	BGDSHR	
FASTTR:	BGFASTT	
RPS:	BG31	
TRKHLD:	AG27	BG16
TTIME:	AGTTMR	BGTT
USERID:	CGUSID	

IOTAB

The IOTAB generation macro describes installation requirements for I/O tables:

IODEV Number of I/O devices attached to the system.

NPGR Number of programmer logical units for all partitions.

The following globals are set dependent on the specified options:

IODEV:	AG1
NPGR:	AGNPGR

Global	Set by Option	Purpose
AGDAT1	(IOTAB)	Defines the minimum size of the real address area, as generated by IOTAB.
AGDAT2	(IOTAB)	Defines the minimum size of the virtual address area, as generated by IOTAB.
AGDEFLB	(IOTAB)	Default programmer LUBs per partition.
AGDFPSZ	(IOTAB)	GETVIS size for DASDFP.
AGEPMSK	(SGPDSECT)	Page mask.
AGIBIT	(SGPDSECT)	Invalid bit (IBIT).
AGINVPT	(SGPDSECT)	Invalitation pattern for PTE.
AGJCLAR	(IOTAB)	Size of JCL buffers per partition.
AGMAXEX	(IOTAB)	Percentage value of maximum SVA GETVIS space for excessive requestors.
AGMAXLB	(IOTAB)	Maximum programmer LUBs per partition.
AGMINGV	(IOTAB)	Minimum partition GETVIS in K.
AGMINLB	(IOTAB)	Minimum programmer LUBs per partition.
AGMINSG	(IOTAB)	Minimum system GETVIS for vital requestors
AGMINSZ	(IOTAB)	Minimum partition size in K.
AGNPGR	NPGR= (IOTAB)	Number of programmer LUBS for all partitions.
AGOFMSK	(SGPDSECT)	Page table offset mask.
AGPHLSL	(IOTAB)	Length of load table per partition.
AGPGPSG	(SGPDSECT)	Number of pages per segment.
AGPGMSK	(SGPDSECT)	PTE mask.

Figure 321 (Part 1 of 3). Global Settings

Global	Set by Option	Purpose
AGPNSFT	(SGPDSECT)	Shift value of page size and page number.
AGPRNXT	(SGPDSECT)	Next page in TFIX table.
AGPSIZB	(IOTAB)	Page size in bytes.
AGPTSPT	(SGPDSECT)	Shift value of PTAS and page table.
AGSGMSK	(SGPDSECT)	Segment mask.
AGSGSIZ	(SGPDSECT)	Segment size in bytes.
AGSVIS	(IOTAB)	Minimum system GETVIS space in K.
AGSYSLB	(IOTAB)	Number of system class LUBs.
AGTASK	(SUPVR)	Number of subtasks to be supported.
AGTIME	(FOPT)	Balancing time slice.
AGTTMR	TTIME= (FOPT)	PIK of partition owning task timer.
AG1	IODEV=n (IOTAB)	Number of entries for PUB table.
AG13	(IOTAB)	See AGSYSLB.
AG15	(all)	Checks macro sequence of supervisor macros
AG27	TRKHLD=n (FOPT)	Indicates number of tracks/blocks that can be held.
BGDSHR	DASDSHR=YES(FOPT)	Indicates that DASD sharing is supported.
BGFASTT	FASTTR=YES (FOPT)	Indicates that fast CCW translation is supported.

BGTT	TTIME=BG (FOPT)
	or =Fn
		Indicates that the task timer is supported.

BGVM	MODE=VM (SUPVR)	Indicates VSE/Advanced Functions-VM/370 linkage improvements support.
BG16	TRKHLD=n (FOPT)	Determines if the track hold function is supported.
BG31	RPS=YES (FOPT)	Determines if rotational position sensing is supported.

Figure 321 (Part 2 of 3). Global Settings

Global	Set by Option	Purpose
BG35	MICR=1419, 1419D (SUPVR)	Determines if any MICR type device is supported.
BG36	MICR=1419D (SUPVR)	Determines if a magnetic ink or optical character reader with dual address adapter is supported.
BG370	MODE=370 (SUPVR)	Indicates 370 support to be generated.
CGP(n) *	NPARTS (SUPVR)	Partition identifier in sequence BG, F1, ...
CGUSID	USERID= (FOPT)	Print supervisor ID at IPL completion.
IJBSGEN	(SUPVR)	Indicate supervisor generation in progress
NPART	NPARTS (SUPVR)	Number of partitions supported by this VSE supervisor.
P(n) *	NPARTS (SUPVR)	Partition identifiers in sequence BG, Fnparts−1 ... F1.

* n=1 if the global refers to BG, and 2 or more if the global refers to a foreground partition. As many globals of this name are set as partitions are being generated.

Figure 321 (Part 3 of 3). Global Settings

APPENDIX B. MACRO DESCRIPTIONS

SUPERVISOR INTERFACE MACROS

The macros described on the following pages represent a symbolic interface between the VSE/AF Supervisor and other SCP subcomponents, such as non-resident system tasks, IPL, EOT, etc., or IBM licensed programs, like ACF/VTAM, VSE/POWER, etc. They are not to be considered as new general purpose user interfaces and will not be included in the SRL documentation.

A specific authorization is required for most of the described functions. This is so because the related interfaces are only committed to restricted classes of users and because the integrity of the system may be affected by an inappropriate usage of some of the functions. In some cases, authorization is restricted to system tasks and to one or more other known components. An easy and fast identification of the requestor is useful for this type of authorization checks. Components, which are not initialized by the VSE/AF supervisor itself, are therefore requested to identify themselves to the supervisor by means of a SUBSID NOTIFY macro during their initialization. Note, however, that authorization is related to the code being executed, and may therefore change dynamically. Note also that a protection key of 0 is in many cases neither necessary nor sufficient as authorization criterion. A list of authorized components (besides system tasks) is given whenever applicable. It may be extended in the future.

ALLOCATE

The ALLOCATE macro (see also SVC 83) allocates or reallocates real or virtual partitions. The macro is used by the job control ALLOC(R) command processing, and the attention ALLOC(R) command processing. It may also be used by other system components if applicable. The format is as follows:

```
[name]    ALLOCATE    [APL={name1|(1)}]
```

APL Defines the parameter list into which the specified operands have to be placed before issuing this macro. The address of the parameter list may be supplied either as an operand (name1) or in a register.

The format of the parameter list is as follows:

APL

```
┌───────┬───────┬───────┬───────┬───────┬───────┐
│ APL-  │ APL-  │ APL-  │ APL-  │ APL-  │ APL-  │
│ SID   │ NO    │ PID   │ SIZE  │ PID   │ SIZE  │
└───────┴───────┴───────┴───────┴───────┴───────┘
 0       2       4       6       8       A    B
                                 |<────────────>|

                       Repetitive; depending on the
                       number of elements specified
                       in the command.
```

APLSID It contains the space identifier for which the partitions are to be allocated, with the following values:

```
'1 ' for the primary space   (all modes)
'n ' for the spaces 2 to 3   (370 mode only)
'R ' for real (re)allocation (all modes)
'S ' for the shared area     (370 mode only)
```

APLNO It contains the number of operands which were specified in the ALLOC command, and which is equal to the number of elements in the parameter list.

APLPID It contains the partition id 'BG' or 'Fn'

APLSIZE It contains the partition size in K-bytes.

To generate the layout of the parameter list, the following macro may be used:

```
 [name]   APL    [DSECT=YES]
```

Output: Register 15 contains one of the following return codes:

For MODE=370:

0 (X'00') The partitions are (re)allocated.

4 (X'04') The partitions are reallocated.
But the size of the program area in at least one partition has been decreased because it does not leave the minimum GETVIS area.
Display the new partition characteristics by a MAP command.

8 (X'08') The partitions are not (re)allocated.
The requested (rounded) allocation exceeds the corresponding allocation pool.

12 (X'0C') The partitions are not (re)allocated.
At least one specified (rounded) virtual partition allocation value is smaller than 128K.

16 (X'10') The partitions are not reallocated.
At least one specified virtual partition is zero although there is no corresponding real partition, or for at least one specified real partition there is no corresponding virtual partition.

20 (X'14') The virtual partitions are not reallocated.
At least one of the affected partitions is active or stopped and the new virtual allocation would not include the old virtual boundaries,
or the lower virtual boundary of the current partition would have to be moved upwards.

24 (X'18') The real partitions are not reallocated.
At least one of the affected partitions, other than the current, is active or stopped and the new real allocation would not include the old real boundaries.

28 (X'1C') The partition are not reallocated.
At least one of the specified partitions is already allocated in another space.

32 (X'20') The partition are not reallocated.
There is not enough System GETVIS space available to allocate a segment table for a new address space or the system PFIX counter is exhausted.

For MODE=E:

0 (X'00') The partitions are (re)allocated.

4 (X'04') The partitions are (re)allocated.
But the size of the program area in at least one partition has been decreased because it does not leave the minimum GETVIS area,
and/or at least one real partition has been reset to zero because it exceeds the virtual partition size.
Display the new partition characteristics by a MAP command.

8 (X'08') *** See MODE=370 ***

12 (X'0C') *** See MODE=370 ***

16 (X'10') The partitions are not (re)allocated.
At least one specified real partition value is larger than its corresponding virtual partition.

20 (X'14') *** See MODE=370 ***

24 (X'18') The real partitions are not reallocated.
At least one of the specified partitions, other than the current, is active or stopped and the new real allocation would reduce the old real size.

28 (X'1C') *** not given ***

32 (X'20') *** not given ***

For MODE=VM:

0 (X'00') The partitions are (re)allocated.

4 (X'04') *** See MODE=E ***

8 (X'08') *** See MODE=370 ***

12 (X'0C') *** See MODE=370 ***

16 (X'10') *** See MODE=E ***

20 (X'14') *** See MODE=370 ***

24 (X'18') *** See MODE=E ***

28 (X'1C') *** not given ***

32 (X'20') *** not given ***

ASYSCOM

The ASYSCOM macro returns the address of the system communication region to the user. The macro has the following format:

```
[name]   ASYSCOM   [(1)]
```

The operand specifies the general register that is to be loaded with the address of the System Communication Region (SYSCOM).

CLOSEHCF

The macro CLOSEHCF must be issued to terminate accessing of the HCF started by the POINTHCF macro.

The macro has the following format:

```
[name]   CLOSEHCF [{(hcfreg)|(1)}]
```

HCFREG Is the general register containing the address of the HCFCB control block returned by the corresponding POINTHCF macro.

Note: If no operand is specified, the WRITE HCFCB will be closed.

Output: Register 15 contains one of the following return codes:

0 (X'00')	Normal processing successfully completed.
4 (X'04')	Inconsistent input.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

The format is as follows:

```
[name]   CPCOM   ACMD={name1|(r1)|(1)}
                ,LCMD={n|(r0)|(0)}
```

The operands have the following meaning:

ACMD   Address of command text

LCMD   Length of command in bytes, must be between 1 and 240

Output: Register 15 contains one of the following return codes.

0 (X'00')   Command successfully completed

1 (X'01')   Is returned if the supervisor is not running under VM (corresponds to CP completion code for 'Invalid Command').

2 (X'02')   Is returned if any parameter is invalid (corresponds to CP completion code for 'Invalid Parameter').

In all other cases, the CP completion code is returned unchanged in Register 15.

Cancel conditions:

The requestor is cancelled with 'Invalid SVC' (Error 21) if he is not authorized.

Register Usage:

Reg. 0	Length of command
Reg. 1	Address of command text
Reg.15	Return code

DEVREL

DEVREL decrements the physical usage counter and in addition resets the ownership for the specified partition as soon as the decremented physical unit counter reaches zero.

GETFLD PU=...,FIELD=OWNER can be used to obtain the current ownership status.

The macro can only be used by IPL and VSE/POWER and has the following format:

```
ASSEMBLER:

   [name]   DEVREL    PU={name1|(r1)|(0)},PART={name2|(r2)|(1)}

PLS:

  ?[name:]  DEVREL    PU{(name1)|((r1))|(0)} PART{(name2)|((r2))|(1)};
```

PU — Name of a 2-byte field or register containing the physical unit number of the device (same as PUB-index = PUB-offset/8).

PART — Name of a 2-byte field or register containing the identifier (PIK) of the applicable partition. A value of 0 is interpreted as a request for system ownership.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Request complete
4 (X'04')	SIO-count for JA must be saved
8 (X'08')	Device not owned by specified partition

Register Usage:

Reg. 0	Physical unit number for input
Reg. 1	PIK for input
Reg.15	Function code for input, return code for output

DEVUSE

DEVUSE increments the physical usage counter and sets ownership for the specified partition.

GETFLD PU=...,FIELD=OWNER can be used to obtain the current ownership status.

The macro can only be used by IPL and VSE/POWER and has the following format:

```
ASSEMBLER:

   [name]   DEVUSE    PU={name1|(r1)|(0)},PART={name2|(r2)|(1)}

PLS:

  ?[name:]  DEVUSE    PU{(name1)|((r1))|(0)} PART{(name2)|((r2))|(1)};
```

PU   Name of a 2-byte field or register containing the physical unit number of the device (same as PUB-index = PUB-offset/8).

PART   Name of a 2-byte field or register containing the identifier (PIK) of the applicable partition. A value of 0 is interpreted as a request for system ownership.

<u>Output:</u> Register 15 contains one of the following return codes.

0 (X'00')	Request complete
8 (X'08')	Non-DASD device owned by other partition
12 (X'0C')	Device is down

<u>Register Usage:</u>

Reg. 0	Physical unit number for input
Reg. 1	PIK for input
Reg.15	Function code for input, return code for output

DSPLOG

The DSPLOG macro provides layouts for the various records on the LOG-DATA-SET. It describes the header as well as the detail records.

The macro has the following format:

```
[name]   DSPLOG
```

The macro has no operands.

DSPLPAR

The DSPLPAR macro provides layouts for the communication areas used between the LOG system task and the components issuing LOG requests.

The macro has the following format:

```
 [name]   DSPLPAR
```

The macro has no operands.

DTSAPL

The DTSAPL macro generates or describes the layout of the Authorization Parameter List (APL).

The macro has the following format:

```
[name]   DTSAPL    [DSECT=YES]
```

For detailed description of the various input and output fields see DTSAPL macro expansion.

DTSJPL

The DTSJPL macro provides the layout of the Job Control Parameter List (JPL) build by JCL and for which storage was reserved at IPL time to allow access control.

When a job control ID-statement is detected, the access control information is extracted from the access control resource table (DTSECTAB) and transferred to the JPL.

The macro has the following format:

```
[name]   DTSJPL
```

For detailed description of the various fields see DTSJPL macro expansion.

EXTENT

This macro provides a DASD file protect interface. The facility supports all DASD (FBA and CKD) devices and can perform the following functions.

- An extent can be ADDED to the Extent Information for a LUB.
- An extent can be DELETED if a matching extent is found for this LUB.
- ALL extents for a given LUB can be DELETED.
- Given a LUB, it can CHECK for an already existing matching extent.

The macro has the following format:

```
[name]   EXTENT   {(1)|name1|DSECT=YES}
```

name1 Address of a parameter list as described by the DSECT. In case register notation was used, the register must contain the address of the parameter list.

Output: Register 15 contains one of the following return codes:

0 (X'00')	Request successfully processed.
4 (X'04')	The LUB is invalid.
8 (X'08')	No matching extent found (DELETE or CHECK).
12 (X'0C')	No more extent entries available (ADD).
16 (X'10')	Parameter list contains invalid data.

EXTRACT

The EXTRACT macro (see also "SVC 98 (X'62' - EXTRACT/MODCTB)" on page 73) provides the following information:

- Partition boundaries from the Storage Management Control Block (SMCB)
- Unit information from the PUB, the PUB extension or the PUB2 table entries
- Control registers
- CPU identifier

The macro has the following format:

```
[name]   EXTRACT    ID={BDY|CPUID|CR|DEVICE|DVTY|MAP|PUB|PUB2}
                    ,AREA={name1|(S,name1)|(r1)}
                    ,LEN={n|(r2)}
                    [,{SEL={name3|(S,name3)|(r3)}|
                     SEP={name4|(S,name4)|(r4)}|
                     PU={name5|(S,name5)|( r5)}}]
                    [,PID={name6|(S,name6)|(r6)}]
                    [,SID={name7|(S,name7)|(r7)}]
                    [,DISP={0|n|name8|(r8)}]
                    [,MODE={P|S|T}]
                    [,MFG={name9|(r9)}]
```

ID — Identifies the requested information. Valid parameters are:

BDY Returns the boundaries of a partition.
CPUID Returns CPUID and the partition prefix for SYSLOG.
CR Returns the contents of control registers.
DEVICE Returns device specific information as retrieved by means of the SENSE-ID command.
DVTY Returns the device type code of specified device.
MAP Returns a list of partitions allocated in the specified space.
PUB Returns the contents of PUB of specified device.
PUB2 Returns the contents of PUB2 of specified device.

AREA — Address of the user area where the extracted information is to be stored.

LEN — Length of user area in bytes.

SEL — Points to a halfword containing the logical unit information in the same format as the logical unit number in the CCB.

SEP Points to a halfword containing the device address (cuu).

PU Points to a halfword containing the PUB-index (X'0000'-X'00FE').

PID Points to a two-byte field containing the PIK of the partition the information belongs to. The default is the identifier of the partition issuing the request.

SID Address of a two byte symbolic space identifier. Supported values for 370-mode 'R ', 'n ' (n = 1..3) and 'S '. For ECPS:VSE and VM mode only '1 ' is supported.

Default value for all modes is '1 '.

DISP Specifies the offset within the specified field where EXTRACT is to start. The default value is zero. DISP may either be specified as a number or as a register containing the displacement value. If DISP is not a number, nor a register, it is assumed to be the name of a field defined in the MAPPUB DSECT.

MODE Qualifies the type of information that the requestor wants to be returned.

S Indicates that the requester wants SYSTEM specific information to be returned.
P Indicates that the requester wants PERMANENT information to be returned.
T Is the default value and indicates that the requester wants TEMPORARY information to be returned.

MFG Points to a work area where the parameter list is to be generated by the EXTRACT macro (for re-entrant coding). If register notation is used, register 1 may point to this area.

The following list shows which operands are required with the different IDs:

ID	AREA	LEN	DISP	SEL	SEP	PU	PID	SID	MODE	MFG
BDY	R	R	N/A	N/A	N/A	N/A	O*	N/A	O	O
CPUID	R	R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O
CR	R	R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O
DEVICE	R	R	N/A	R+	R+	R+	O	N/A	N/A	O
DVTY	R	R	O	R	N/A	N/A	O	N/A	N/A	O
MAP	R	R	N/A	N/A	N/A	N/A	N/A	O	N/A	O
PUB	R	R	O	R	N/A	N/A	O	N/A	N/A	O
PUB2	R	R	O	R+	R+	N/A	O	N/A	N/A	O

R = Required Parameter O = Optional Parameter * = If PID given, MODE = P required + = Mutual exclusive

Input:

R0 Is a work register if S-type operands are used (for self-relocating programs)

R1 Is used as a pointer to a parameter list (PARMLIST) built by the EXTRACT macro prior to calling SVC 98 to process the request.

For a description of the layout of the parameter list see SVC 98 in Chapter 2, "Interrupt Processors".

Output:

The user's area is cleared to binary zeros and the information requested by ID is moved to the user's area in the specified length.

Reg.15 Contains one of the following return codes.

0 (X'00') The requested information has been returned.

4 (X'04') The partition specified is not supported by the system or the specified SID is not supported or invalid for the current system mode.

8 (X'08') The logical unit specified exceeds the range of the logical units for the specified partition or the specified PUB index is not

	within the range of PUBs valid for this system.
12 (X'0C')	The LUB is not assigned or ignored.
16 (X'10')	The length parameter is found to be zero, or negative, or below the minimum; or the DISP specification exceeds the length of the PUB or PUB2 entry.

An illegal SVC is forced, if one or more of the following conditions are true:

- ID specification is invalid.
- PIK has been specified and the user does not have key 0 (ID=PUB2 only).
- SEP has been specified and the user does not have key 0.

ID=PUB Minimum length = default length = 1 byte

The appropriate PUB bytes are copied to the user area. The layout of a PUB entry is described by the DSECT macro MAPPUB. If the logical unit specified by EXTRACT ID=PUB is unassigned or assigned ignore, a return code 12 is presented in Register 15 and, in addition, an indicator is stored in byte 0 of the area specified by the AREA operand. The indicator is X'FF' if the logical unit is unassigned, and X'FE' if it is assigned ignore.

ID=PUB2 The PUB2 table entry is moved to the user area, either in the specified range, or in its complete length.

ID=CPUID

A length of 10 bytes is required.

The area must start on a doubleword boundary.

The user area pointed to by the AREA parameter will contain the CPUID as stored by the CPU, followed by the SYSLOG ID of the issuing partition. The SYSLOG ID is given in the form (BG|F1|...|FB).

ID=BDY MODE=S

A length of 20 bytes is required.

The output is as follows:

DEC	HEX	Description
0 – 3	0 – 3	Size of minimum page pool (K-bytes)
4 – 7	4 – 7	Size of fixed supervisor (K-bytes)
8 – 11	8 – B	Amount of real storage available for real allocations (K-bytes)
12 – 15	C – F	Size of system 'real partition' for PFIX in SVA (K-bytes)
16 – 19	10 – 13	Start address of shared area (S-partitions and SVA)

Figure 322. Output with MODE=S

MODE=P

A length of 20 bytes is required.

'MODE=P' indicates that the permanent boundaries of the issuing partition or the partition indicated by PID are to be returned. They correspond to the latest allocation and may not yet have been used by the active job.

The output is described by the DSECT MAPBDYVR which will be generated when using the appropriate macro.

The format is as follows:

```
[name]  MAPBDYVR   [DSECT=YES]
```

If 'name' is omitted the default name generated MAPBDYVR. DSECT=YES generates a DSECT; if omitted, in-line code is generated.

DEC	HEX	Label	Description
0 - 3	0 - 3	VPBEGIN	Virtual partition start address
4 - 7	4 - 7	VPEND	Virtual partition logical end address (last addressable byte, GETVIS area excluded).
8 - 11	8 - B	VPGEND	Virtual partition physical end address (last addressable byte, GETVIS area included)
12 - 15	C - F	RPBEGIN	Real partition start address
16 - 19	0 - 13	RPEND	Real partition end address (last addressable byte)
20	14	VBDYLEN	Length of MAPBDYVR area.

Figure 323. Output as Described by Macro MAPBDYVR

MODE=T

A length of 20 bytes is required.

'MODE=T' indicates that the temporary boundaries of the issuing partition are to be returned. This is also the default value. PID may not be specified in this case, since a snapshot of any other partition's temporary boundaries is unreliable.

If the partition is executing in real mode the boundaries of the real partition (which in ECPS:VSE mode is contained in the corresponding virtual partition) will be returned; i.e., PBEGIN defines the begin address of the real partition (Problem Program Save Area). PENDLOG defines the logical end of the real partition which is identical with the allocated (or PFIXed) partition end address (if no SIZE was specified in the EXEC statement), and which is the real GETVIS area begin address (if SIZE was specified). PGEND is the allocated (or PFIXed) real partition end address (if no SIZE was specified in the EXEC statement), and it is the highest used GETVIS area address (rounded to the next page boundary) (if SIZE was specified). PENDLOG and PGEND are equal if no SIZE was specified in the EXEC statement, or if no GETVIS request has been issued so far.

The output is described by the DSECT MAPBDY which will be generated when using the appropriate macro.

The format is as follows:

```
[name]  MAPBDY     [DSECT=YES]
```

If 'name' is omitted the default name generated MAPBDY. DSECT=YES generates a DSECT; if omitted, in-line code is generated.

DEC	HEX	Label	Description
0 - 3	0 - 3	PBEGIN	Partition start address, corresponding to problem program save area address (field PIB SAVE).
4 - 7	4 - 7	PENDLOG	Logical end of partition (last addressable byte, GETVIS area excluded), corresponding to field PPEND in the partition communication region.
8 - 11	8 - B	PGEND	Physical end of partition (last addressable byte, GETVIS area included).
12 - 15	C - F	PFIXLMT	PFIX limit (K-bytes) or zero (real mode).
16 - 19	10 - 13	PFIXCNT	PFIX count (number of PFIXed pages).
20	14	MBDYLEN	Length of MAPBDY area.

Figure 324. Output as Described by Macro MAPBDY

ID=MAP

A length of 32 bytes is required.

A list of two byte PIK values of partitions allocated in the specified space will be returned. The list PIK values is ordered by increasing partition begin address. The ALLOCATE algorithm makes sure that all partitions are contiguous. The end of the PIK list is indicated by zero.

ID=DEVICE

A length of 10 bytes is required.

Device specific information is to be returned.

The output is described by the mapping macro MAPDEVIN.

The format is as follows:

```
[name]  MAPDEVIN   [DSECT=YES|NO]
```

If 'name' is omitted the default name generated MAPDEVIN. DSECT=YES generates a DSECT; if omitted, in-line code is generated.

DEC	HEX	Label	Description
0 – 1	0 – 1	IJBDVCUU	CUU address
2	2	IJBDVDTC	VSE device type code
3 – 9	3 – 9	IJBDVSNS	Sense device type information
3	3		Validity flag
		IJBDVVAL	X'FF' Entry is valid
4 – 5	4 – 5	IJBDVCUN	Control unit type number
6	6	IJBDVCUM	Control unit model number
7 – 8	7 – 8	IJBDVDTN	Device type number
9	9	IJBDVDTM	Device type model number
10	A	IJBDVLEN	Length of MAPDEVIN area.

Figure 325. Output as Described by Macro MAPDEVIN

FREECBUF (370 Mode Only)

The FREECBUF returns a Copy Buffer back to the supervisor. It is used by TAPE-ERP and has the following format.

```
[name]   FREECBUF  {name1|(1)}
```

name1 Address of copy buffer that is to be returned to the supervisor.

GETCBUF (370 Mode Only)

The GETCBUF macro returns the address of a 72-byte area (one Copy Buffer) to the requester. It is used by the TAPE-ERP and has the following format.

```
[name]   GETCBUF
```

Output: The address of a Copy Buffer (72-byte area in supervisor) is returned in register 1. If none is available, zero is returned.

GETDADR

The GETDADR macro returns the virtual address of an I/O area pointed to by a CCW.

The macro is issued by system tasks, e.g. ERP, RAS, CRT and it has the following format:

```
[name]   GETDADR  {ccwaddr|(8)},{offset|(0)}
```

ccwaddr Pointer to a CCW containing the address of the I/O area. If the CCW address is a real address, the issuing task will be canceled.

offset Offset within the I/O area, which is to be added to the I/O area before translation to virtual takes place.

Output: Register 15 contains the converted (virtual) address if the page frame indicated by the real address is connected to a virtual page. If the address of the CCW is invalid, zero is returned.

GETFLD

The GETFLD macro retrieves system, partition, task or device specific information. The information is returned right adjusted in register 1.

```
[name]   GETFLD FIELD=name1,
                [,{PART|TASK|PU}={name2|(0)}]|
                [,PU={name3|(r3)|(0)},PART={name4|(r4)|(1)}]|
                [,LU={name5|(r5)|(0)},PART={name6|(r6)|(1)}]
```

FIELD= Symbolic identification of the field to be retrieved. Valid symbols, and their interpretation are as follows.

ABEXIT Pointer to standard AB exit routine and save area (TASK). The pointers are returned in register 0 and 1 respectively. Bit 0 in register 0 is on if the exit routine is currently active. Zero values are returned if no exit is defined The standard exit is associated with STXIT AB, OPTION=DUMP or NODUMP. No interface is available to retrieve the addresses of the exit associated with OPTION=EARLY.

ABINPR (ICCF only)
A value of 1 if the AB exit routine is active for the specified task, else 0 (TASK).

ACLOSE A value of 1 if VSAM Automatic Close is currently in process for this partition, else 0 (PART).

AOTPTR (ACF/VTAM only)
Pointer to the AOT of the specified task.

BALANCE (BAM, RAS, ERP only)
New track balance (number of available bytes per track) for a specified device is calculated and returned.

<u>Input:</u>

R0 Must contain record descriptive information:

Bytes	0-1:	Length of one fixed data record
Byte	2:	Zero or length of key if keyed records
Byte	3:	Record number

R1 Must contain device descriptive information:

Bytes	0-1:	Logical unit identifier (as byte 6-7 of CCB)

Bytes 2-3: Current or old balance

Output:

R0 If not zero, the record does not fit in this track.

R1 Contains the remaining bytes available on the current track.

CANCLALL (TERMINATOR only)
A value of 1 if cancel has to be propagated to all tasks belonging to the same partition as the specified task, else 0 (TASK).

CNCLCODE (EOT, TERMINATOR, VSE/POWER, ACF/VTAM only)
First cancel code (TASK).

CNCLCOD2 (EOT, TERMINATOR, VSE/POWER, ACF/VTAM only)
Last cancel code (TASK).

COMRGPTR (VSE/POWER, ICCF, OCCF)
Address of Partition communication region (PART).

CPUTIME Amount of CPU time, in units of 16 micro-seconds, charged to this partition since begin of job step (PART). This function is available only if JA support was activated in the IPL SYS command.

Note: This counter overflows and is reset to zero each time it reaches a value of about 18 hours.

EOTACT (ICCF only)
A value of 1 if End of Task (EOT) is active for the specified task, else 0 (TASK).

ICCFPP A value of 1 if the specified task is assigned to an ICCF pseudo partition, else 0 (TASK).

ICCFRO (ICCF only)
A value of 1 if the specified task is assigned to an ICCF pseudo partition and is eligible for ICCF roll-out, else 0 (TASK).

ITEXIT Pointer to IT exit routine and save area (TASK). The pointers are returned in register 0 and 1 respectively. Bit 0 in register 0 is on, if the exit routine is currently active. Zero values are returned, if no exit is defined.

Note: Cross partition requests are not supported.

LTAACT (ICCF only)
A value of 1 if the LTA is active for the specified task, else 0 (TASK).

LTAPTR (VSE/POWER, ACF/VTAM only)
Pointer to the Logical Transient Area if currently owned by the specified task (TASK). (The LTA must not necessarily be active). If not owned by the specified task, a value of zero is returned.

MAINTASK The task identifier (IJBTIK) of the maintask of the partition to which the specified task belongs.

MOUNTFLG A value of 1 if the specified device has been reserved, to allow a volume change, else 0 (PU).

MSECS Current time slice for partition balancing in milli seconds.

NSUB Number of attached subtasks (PART)

OCCFACT (ICCF only)
A value of 1 if at least one OCCF request is pending for the specified task, else 0 (TASK).

OCEXIT Pointer to OC exit routine and save area (PART). The pointers are returned in register 0 and 1 respectively. Zero values are returned, if no exit is defined. Bit 0 in register 0 is on, if the exit routine is currently active.

Note: Cross partition requests are not supported.

OPENSVA A value of 1 if OPEN routines are executing in the SVA for this task, else 0 (TASK).

Note: Cross partition requests are not supported.

OWNER Owner of specified device (PU).
If a unique owner exists, the 2-byte PIK of the owning partition is returned (see also return code).
The information is taken from the PUB Ownership Table entry of the device. For a ACF/VTAM-owned device, the PIK of the ACF/VTAM partition is returned.

Output: Register 15 contains one of the following return codes.
0 (X'00') No owner
4 (X'04') Unique device owner
8 (X'08') Multiple device owners (DASD)

PAGEIN Number of page-in I/Os in whole system since IPL or last wrap around of the 3-byte counter. The function depends on JA support being active.

PAGEOUT Number of page-out I/Os in whole system since IPL or last wrap around of the 3-byte counter. The function depends on JA support being active

PCBPTR (SYSTEM only)
Pointer to the Partition control block (PCB) (PART).

PCEXIT Pointer to PC exit routine and save area (TASK). The pointers are returned in register 0 and 1 respectively. Bit 0 in register 0 is on, if the exit routine is currently active. Zero values are returned, if no exit is defined.

Note: Cross partition requests are not supported.

PIK PIK of the partition to which the specified task belongs (TASK).

PPSAVAR Pointer to the problem program save area (TASK). This is the partition save area for maintasks; whereas for subtasks it is the save area specified in the ATTACH macro.

Note: Cross partition requests are not supported.

PU PUB index of the specified logical unit (LU, PART).

Note: PART required for this option.

PUBXPTR Pointer to Physical Unit Block Extension (PU).

PUSECNT Number of users that access this device using "Physical addressing". The returned value applies to the specified partition only (PU,PART).

The operand PART is required for this option.

SAVAR Pointer to the current save area (TASK).

Note: Cross partition requests are not supported.

SSFLAGS (SYSTEM only)
A bit-string identifying the subsystem (if any) active in the specified partition (PART).

HEX	Subsystem
0100	NPDA
0080	VSE/POWER
0040	ACF/VTAM
0020	ICCF
0010	CICS
0008	VCNA
0004	OCCF
0002	SQL/DS
0001	SSX

STATUS (ICCF only)
A 1-byte identification of the current status of the specified task (TASK).

Currently committed values are:
X'81' Task is waiting for the LTA
X'82' Task is waiting for a CCB/IORB/ECB to be posted

SYSRESW A value of 1 if DASD file protection is being bypassed for this task, else 0 (TASK).

Note: Cross partition requests are not supported.

TCBPTR (SYSTEM only)
Pointer to Task Control Block (TASK).

TERMACT (ICCF only)
A value of 1 if the Terminator is active for the specified task, else 0 (TASK).

USECNT Number of current or eventually stored assignments plus number of users that access this device using "Physical addressing". The returned value applies to the specified partition only (PU,PART).

VSAMOPEN A value of 1 if there is at least one open VSAM ACB for the specified partition, else 0.

VTAMDISP (ACF/VTAM only)
A value of 1 if the ACF/VTAM AP exit is scheduled for the specified task, else 0.

VTAMOPEN (ACF/VTAM only)
A value of 1 if there is at least one open ACF/VTAM ACB for this task, else 0.

The operand PART is required for this option.

PART= Name of a 2-byte field or a register containing the PIK of the partition to which the specified field belongs. The default value is the PIK of the requester.

TASK= Name of a 2-byte field or a register containing the TID of the task to which the specified field belongs. The default value is the TID of the requester.

PU= Name of a 2-byte field or register containing the physical unit number (PUB-index) of the device to which the specified field is to be returned.
This operand is mandatory for device related fields.

LU= Name of a 2 byte field or a register containing the logical unit.

Note: Format as in CCB.

Output:

Reg. 0 Will contain the first full word of a double word field whenever applicable (PCEXIT and ITEXIT). In this case register 1 will contain the contents of the second full word.

Reg. 1 Contains the requested field contents. The requested field contents is returned right-adjusted, which, for double word fields is the second full word.

Reg.15	Will contain the return code whenever applicable.

Register Usage:

Reg. 0	Is used to pass the PIK, the TIK or the PU number. A value of 0 is passed if the TASK or PART operand is omitted.
Reg. 1	PIK for input when PU and PART are combined.
Reg.15	Is used to pass the function code.

GETIME

The GETIME macro allows VSE components to compute the time and date from the eight-byte value of the Clock Comparator. The macro has the following format:

```
[name]   GETIME     [STANDARD|BINARY|TU]
                    [,LOCAL|GMT]
                    [,MFG={area|(S,area)|(r1)}]
                    [,CLOCK=YES]
```

CLOCK Identifies that registers 0 and 1 contain a value which was obtained by means of a STCK instruction. This value is transferred into time and date as defined through the other parameters and the associated JCL options. Register 1 contains the time and register 15 and 0 the date (in the form mmddyy00 or ddmmyy00).

All other operands have the same meaning as described in the VSE/Advanced Functions, Application Programming: Macro Reference, SC33-6197.

GETJA

The GETJA macro updates, clears or resets supervisor maintained information in the partition's job accounting tables. It is issued by job control at the end of a job step just before the user-written phase $JOBACCT is fetched. The format of the macro is as follows:

```
 [name]   GETJA   [PART={name1|(r1)|1}
                  [,ACTION={UPDATE|CLRTIME|RESET}]
```

PART Name of a 2-byte field or register containing the PIK of the partition, to which the request refers.
If PART is omitted the current partition is assumed.

ACTION Defines the function that is to be performed. This operand is required for JOB CONTROL and it is obsolete for other programs, since other programs usage is defaulted to ACTION=UPDATE.

UPDATE The time and SIO related accounting fields residing in supervisor storage are updated.

CLRTIME The time related accounting fields residing in supervisor storage are cleared.

The start I/O related accounting fields residing in supervisor storage are updated.

RESET The time and SIO related accounting fields residing in supervisor storage are reset and in addition, CPU time is moved to OVERHEAD time.

GETPRTY

The GETPRTY macro displays the dispatching and balancing sequence of the partitions in the system. The macro has the following format:

```
[name]   GETPRTY  {(1)|area},{(0)|length}
```

area   Address of an area where the information is to be stored.

length   Length of area where information is to be stored. It should be at least 3 times NPARTS. It has to be a self-defining term or loaded into a register (not register 1).

Output: The 2-byte PIKs of the partitions are placed into the user supplied area in ascending order of dispatching priorities. The PIKs are separated by a comma (X'6B'), if the partitions do not participate in balancing whereas the keys of balanced partitions are separated by an equal-sign (X'7E'). The output will be truncated if the specified length is too small.

GETVCE

The GETVCE macro returns to the user a specific volume characteristic entry, retrieved from the Volume Characteristics Table (VCT). It also returns information about the track capacity or track balance. Refer also to "Automatic Volume Recognition (AVR)" on page 122.

The macro has the following format:

```
[name]  GETVCE  AREA={name1|(S,name1)|(r1)},
                {DEVICE={SYSxxx|X'cuu'|DASD}
                     |VOLID={name2|(S,name2|(r2)}
                [,DEVTYPE={name3|(S,name3)|(r3)}]
                     |LOGUNIT={name4|(S,name4)|(r4)}
                     |CHNUNIT={name5|(S,name5)|(r5)}}
                [,LENGTH=n]
                [,MFG={name6|(S,name6)|(r6)}]
                [,REQUEST={TRKBAL|TRKCAP}]
                [,DATALEN={name7|(S,name7)|(r7)}]
                [,KEYLEN={name8|(S,name8)|(r8)}]
                [,RECNO={name9|(S,name9)|(r9)}]
                [,BALANCE={name10|(S,name10)|(r10)}]
                [,OPTION ={(REMOVE{,MAXSIZE{,LAST}}})]
```

AREA   Address of an area where the user wants the specified volume characteristic entry to be stored.

DEVICE   Identifies the device whose volume characteristics entry the user wants to be returned. The specification may refer to a logical unit as well as to a physical one.

DASD   To retrieve (cuu,VOLID) for all DASDs.

VOLID   Address of the VOLID. The volume characteristic entry that the user wants to be returned is identified by its volume serial number. This is a 6-byte field and when specified with register notation also points to a 6-byte field.

If duplicate VOLIDs are present in the system, the first one found by GETVCE is saved and returned to the user if the associated device is not DOWN or, if all the associated devices are DOWN. In all other cases the first device which is not DOWN will be returned to the user.

The VOLID can, however, be further qualified by the DEVTYPE parameter which limits the search to all 3330s or all 3340s, etc.

DEVTYPE Address of device type (1-byte field). If VOLID is given , it can be qualified by the PUB device type code. With this parameter, GETVCE will return volume and device information for the volume with matching VOLID and device type (for example 3340 or 3330).

LOGUNIT Address of device description in logical unit format. The volume characteristic entry that the user wants to be returned is identified by its logical assignment. This operand points to a halfword with the same format as a logical unit number in a CCB.

CHNUNIT Address of device description in physical unit (channel, unit) format. The volume characteristic entry that the user wants to be returned is identified by its physical device address. This operand points to a halfword with the physical device address.

LENGTH Length of data to be placed in AREA by GETVCE. AREA is cleared for this length and the data is moved in. Default is the maximum defined length.

MFG Address of a dynamic storage area that is to be used for parameter list construction for re-entrant programs.

REQUEST

TRKBAL The user requests the track balance (number of remaining data bytes on a track) of the specified DASD device to be returned.

TRKCAP The user requests the number of whole records to be returned that will fit into the given or calculated track balance.

DATALEN Pointer to a two-byte field containing the length of one fixed length data record. This field is processed as a unsigned binary value.

KEYLEN Pointer to a one-byte field containing the key length of one fixed-length data record. This field is processed as an unsigned binary value. If non-keyed records are processed, this byte must either be set to zero or the keyword must be omitted.

RECNO Pointer to a one-byte field containing the record number. A record number of zero results in the maximum track balance being returned to the user or being used for track capacity calculations.

BALANCE Pointer to a two-byte field containing the track balance that is to be used for calculation. This balance field is processed as an unsigned binary value. If the balance is not known, this parameter must be omitted.

OPTION

REMOVE The given record with the specified DATALEN and KEYLEN is assumed as having been removed from the track by the user. GETVCE processing will calculate and/or increment the track balance or track capacity. If the user also provided a balance, it must always equal the number of bytes available on the track before the record was removed.

MAXSIZE The maximum data record length is to be returned to the user (keylength has already been taken into account).

LAST The caller requests that the actual physical record size of the last record is to be used as the decrement/increment to obtain the output balance.

Output:

AREA Is initially cleared, before the requested information is moved in the specified length. The 'AVRLIST DSECT=YES,DEVICE=YES' macro describes the layout of the max. information returned to the user.

R0 Feedback information varies depending on the function (TRKCAP or TRKBAL) and the results.

TRKCAP Set to the number of whole records that will fit on the remainder of a track (input track balance).

TRKBAL If a new record fits or an old record is removed, R0 contains the updated track balance. If a whole record would not fit (R15=X'24') and MAXSIZE was specified, R0 is set to the number of maximum writable DATA bytes. (KEY bytes have already been taken care off). If a whole record would not fit and MAXSIZE was not requested, R0 is set to zero.

R15 Contains one of the following return codes:

0 (X'00') Successful completion.

4 (X'04') Successful completion, but some data is not valid (described by AVRFLG)

8 (X'08') The volume specified is not mounted, or the logical unit specified is not assigned, or the specified unit is not included in the system.

12 (X'0C') The logical unit specified is assigned 'IGNORE'.

16 (X'10') The device is not operational.

20 (X'14') The parameter list is invalid (e.g. logical unit number too high).

24 (X'18') The given logical unit or device is not a DASD.

28 (X'1C') The device is not ready.

36 (X'24') For REQUEST=TRKBAL or TRKCAP only: The input balance is not sufficient to accommodate a record of the specified key and data length. MAXSIZE was specified and at least one byte of data could be written. R0 is set to the maximum number of DATA bytes that will fit.

AVRLIST and DCTENTRY

The contents of the VCT entry is described by the AVRLIST and the contents of the DCT entry is described by the DCTENTRY macro:

```
[name]   AVRLIST  [DSECT={YES|NO}][,DEVICE={NO|YES}]

[name]   DCTENTRY [DSECT={YES|NO}]
```

DSECT   Determines whether a DSECT statement is to be generated or not,

DEVICE   Determines whether DCTENTRY is called within AVRLIST thus describing all the output within one DSECT.

Defaults are: DSECT=YES, DEVICE=NO.

INVPAGE

The INVPAGE macro sets a number of virtual pages (see also SVC 59) into the hardware state 'Disconnected' and into the software state 'No copy on page-data set'. The next reference to such a page will get a cleared page (binary zeros).

The function is restricted to programs with a PSW key of zero.

The macro has the following format:

```
[name]  INVPAGE  {begadr|(3)},{endadr|(4)}
```

begadr   Address within the first page to be invalidated.

endadr   Address within the last page to be invalidated.

INVPART

The INVPART macro invalidates old address space and initializes address space for the execution of the user program (see also SVC 58). This includes GETREAL for real execution, for example GETREAL in 370 mode, and PFIX in ECPS:VSE mode. The function is restricted to programs with a PSW key of zero.

The macro has the following format:

```
[name]  INVPART  {begadr|(3)},{endadr|(4)},REAL={YES|NO}
```

begadr   Begin address of the area to be initialized.

endadr   End address of the area to be initialized.

REAL

- YES: Switch to 'real' mode required.
- NO: No switch to 'real' mode required.

General register 2 will be destroyed.

LOG

The LOG macro allows VSE components to log various information like access control information or ICCF auditing information to the LOG DATA SET, which is maintained and processed by VSE/Access-Control-PP. For ICCF a special handling of a record zero is provided.

The supervisor function is standard, but only usable if VSE/Access-Control-PP is installed. Log access authorization is checked only if SEC specification was given at IPL.

The macro has the following format:

```
[name]  LOG      [LOGID=name1]
                 [,FC={TST|RHD|WHD|WRC|WRF|WFC|RFB|INT|TRM}]
                 [,AREA={name2|(r2)}]
                 [,LEN={n3|(r3)}]
                 [,MFG={name4|(r4)|(1)}]
```

LOGID Identifies the type of logging information. LOGID=AUDIT is currently supported. This operand is mandatory if MFG is omitted.

FC Identifies the request type.

TST The status of the logging function is checked and indicated by the return code.

RHD A leading header record segment fitting into a specified area of record zero of the LOG DATA SET is retrieved. Note that locking of the header record for update is not supported.

WHD The header record segment is overwritten with the specified data and forced to permanent storage.

WRC The specified record is added to the LOG DATA SET at the next free location.

WRF The specified record is added to the LOG DATA SET at the next free location and forced to permanent storage together with any previous records.

WFC All pending records are forced to permanent storage.

RFB The feedback information related to a pending request is retrieved by in-line code.

This is the default function in case the FC operand has been omitted. In this case MFG must be specified or register 1 must point to the appropriate parameter list.

INT Initialize and start the logger system task.

TRM Deactivate the logger system task. No action if ACF is active.

AREA Address of record to be logged or address of area into which a record is to be retrieved. This parameter is mandatory for all functions except TST,INT,TRM and RFB.

LEN Length of record to be logged or length of area into which a record is to be retrieved. The maximum supported record length is 2033 bytes. The length of the header record for ICCF-audit is 116 bytes (see DSPLOG macro). This parameter is mandatory for all functions except TST,INT,TRM and RFB.

MFG Address of a 16-bytes area into which the parameter list of a LOG request is to be build. If omitted, an in-line parameter list is generated. A parameter list defined with MFG is only updated with the specified operands. The requestor is responsible for initializing to binary zeros, when it is used first time.

Output: Register 15 contains one of the following return codes.

0 (X'00') Logging function is active (TST) or request successfully completed.

4 (X'04') Request accepted but not yet completed. A WAIT macro (or any equivalent function) must be issued. Register 1 points on return of the LOG request to an ECB which is posted when the request is completed. A LOG FC=RFB must the be issued to obtain the final return code in register 15.

8 (X'08') The LOG DATA SET is full. VSE/Access-Control reporting is requested.

12 (X'0C') Unrecoverable error. May also affect previous WRC requests processed with return code X'00'. Subsequent requests will present normally return code X'10'.

16 (X'10') Logging function not active. It indicates either that the function was not yet initialized since the last IPL or that the function has been deactivated due to some previous failure(e.g. OPEN failure or unrecoverable I/O error).

20 (X'14') Logging function not available. (VSE/Access Control PP not installed).

24 (X'18') Authorization check failed.

28 (X'1C') Length specification invalid.

32 (X'20') Invalid address of parameter list or record area.

36 (X'24') LOGID or FC (function code) invalid.

40 (X'28') LOG request currently not possible due to initialization in process.

44 (X'2C') Termination request not accepted due to ACF being active.

48 (X'30') Record zero request not accepted because record zero update in process (Try later).

MAPXPCCB

This macro describes the parameter list which must be submitted to XPCC. It contains a detailed description of the various fields like IJBXRETC (return codes), IJBXREAS (reason codes, set when an ECB is posted), IJBXFCT (function codes) etc. The macro is bilingual and has the following format:

```
[name]    MAPXPCCB
```

The macro has no operands. The most important fields are described within the following figures.

Reg. 15	IJBXRETC	(Symbolic Name)	Reason
X'00'	X'00'	(IJBXREOK)	Request handled normal
X'04'	X'01'	(IJBXDAPP)	IDENTIFY with same APPL name was done previously in different partition. ID granted.
	X'02'	(IJBXAPSP)	IDENTIFY with same APPL name was was done previously in same partition. ID granted.
	X'03'	(IJBXFCRQ)	MORE than one CONNECT request pending for this sub-system
	X'04'	(IJBXNIDN)	Other side did no IDENT until now
	X'05'	(IJBXNCNN)	Other side did no CONNECT FOR this APPL until now.
	X'1B'	(IJBXOICL)	Request already cleared.
X'08'	X'06'	(IJBXWCBK)	XPCCB control block format error
	X'07'	(IJBXWIDK)	Wrong IDENTIFY token. (Token is invalid or, APPL issued already TERMQSCE or CONNECT from pseudo partition without an IDENTIFY.)
	X'08'	(IJBXWPID)	Wrong PATH ID token.
	X'09'	(IJBXWOWN)	REQUEST WAS DONE UNDER A TASK which has not the correct Task-ID.

Figure 326 (Part 1 of 2). MAPXPCCB Macro Return Codes

Reg. 15	IJBXRETC	(Symbolic Name)	Reason
X'08'	X'0A'	(IJBXWIND)	Invalid buffer list indicator.
	X'0B'	(IJBXWLST)	Too many buffers or buffer length exceeds 16M bytes or a single buffer length is zero.
	X'0C'	(IJBXWRAR)	Receiving buffer is too small.
	X'0D'	(IJBXTMCR)	Too many CONNECTs for a user APPL.
	X'0E'	(IJBXNSTO)	Try later, not sufficient storage to allocate system control blocks.
	X'0F'	(IJBXNOSY)	None of the partners of a connection is a sub-system (as required).
	X'10'	(IJBXNREQ)	No request pending (line not busy or, SEND was from other side or data already cleared).
	X'11'	(IJBXCCLR)	Request was already cleared.
	X'12'	(IJBXCBSY)	Line already busy
	X'13'	(IJBXWSEQ)	REPLY was issued before data was received.
	X'14'	(IJBXNTRM)	At least one connection is still busy for APPL.
	X'15'	(IJBXNDC1)	Busy from own SEND
	X'16'	(IJBXNDC2)	SEND from other side pending.
	X'17'	(IJBXQSCE)	Other side did issue TERMQSCE.
	X'18'	(IJBXNOC1)	A connection was never existing
	X'19'	(IJBXNOC2)	The other side terminated normally
	X'1A'	(IJBXNOC3)	The other side terminated abnormally.
	X'1C'	(IJBXWCBA)	XPCCB address of this request differs from the one given with CONNECT

Figure 326 (Part 2 of 2). MAPXPCCB Macro Return Codes

One of the following reason codes will be set up in IJBXREAS.

Symbolic Name	Hex Value	Posted ECB	Description
One of the following reason codes will be set up in IJBXREAS.			
IJBXCPRG	X'01'	IJBXSECB	After SEND/SENDR the receiver issued PURGE.
IJBXCLEA	X'02'	IJBXRECB	Sender issued CLEAR before receiver was able to receive/reply
IJBXRECX	X'03'	IJBXCECB	After SENDR command when the RECEIVE is executed by partner.
The next two reason codes are OR'ed to the reason code field.			
IJBXDISC	X'40'	IJBXSECB IJBXRECB	Other side issued DISCONNECT
IJBXABDC	X'80'	IJBXSECB IJBXRECB	Other side was disconnected due to abnormal termination

Figure 327. MAPXPCCB Reason Codes

One of the following function codes will be set up in IJBXFCT.

Symbolic Name	Hex Value	Description
IJBXID	X'01'	Identify
IJBXCON	X'02'	Connect
IJBXSND	X'03'	Send
IJBXSNDR	X'04'	Send with reply
IJBXRCV	X'05'	Receive
IJBXREP	X'06'	Reply
IJBXCLR	X'07'	Clear
IJBXPRG	X'08'	Purge
IJBXDSC	X'09'	Disconnect
IJBXDSCP	X'0A'	Disconnect and purge
IJBXDSCA	X'0B'	Disconnect all
IJBXTRM	X'0C'	Terminate
IJBXTRMP	X'0D'	Terminate and purge
IJBXTRMQ	X'0E'	Terminate and quiesce

Figure 328. MAPXPCCB Reason Codes

MODCTB

The MODCTB macro (see also SVC 98) modifies a PUB2 table entry for a 3800 printer device.

```
[name]   MODCTB    ID=PUB2
                ,AREA={name1|(r1)}
                ,LEN={name2|(r2)}
                ,DISP={name3|(r3)}
                ,{SEL={name4|(r4)}|SEP={name4|(r4)}}
                [,PID={name5|(r5)}]
                [,MFG={name6|(r6)}]
```

ID=PUB2 Defines the information to be modified. The PUB2 information of the physical device connected to the logical unit as specified in the SEL operand is retrieved by the system from the user area in the specified length.

AREA Address of the user area where the PUB2 information is to be retrieved from.

LEN Length of the user area.

DISP Offset within the PUB2 table entry of the specified device. If omitted, the whole PUB2 table entry is modified.

SEL Address of a halfword containing the logical unit (same as in CCB) assigned to the physical device for the partition specified by PID.

SEP Address of a halfword containing the physical unit (cuu). Either the SEL or the SEP operand must be specified.

PID Address of a halfword containing the PIK of the partition the logical unit belongs to. Default is the PIK of the partition of the issuing program. If SEP is specified, PID is ignored.

MFG Address of a 16-byte work area where the parameter list is to be generated by the MODCTB macro (refer to the MFG description for the EXTRACT macro).

Output: Register 15 contains one of the following return codes.

0 (X'00')	The PUB2 table entry has been updated.
4 (X'04')	The specified PIK is invalid for this supervisor.
8 (X'08')	The specified logical or physical unit does not exist.
12 (X'0C')	The logical unit specified in SEL is not assigned or it is assigned IGNore.

16 (X'10')	The length specified is zero, or the DISP specification exceeds the length of the PUB2 table entry for the specified device, or the range defined by DISP and LEN exceeds the range of the PUB2 table entry for the specified device.

An illegal SVC is forced when:

- ID specification is invalid.
- PIK is specified and the user does not have key 0.
- SEP is specified and the user does not have key 0.

Note: The register usage and the layout of the parameter list (see "SVC 98 (X'62' - EXTRACT/MODCTB)" on page 73) are the same as for the EXTRACT macro.

MODESET

The MODESET macro performs key switching. There are two different formats available:

Format 1:

```
[name]     MODESET          EXTKEY={n|(reg)}
                           ,WORKREG=(reg)
                           ,SAVEKEY={old-key-addr|(reg)}
```

EXTKEY Specifies the new key directly (n), or it specifies the register (reg) which does contain the new program key.

WORKREG Specifies a register which the service routine will use as a work register, thus destroying its contents.

SAVEKEY Specifies, where the old key is to be saved (one byte address or register)

The Format 1 is authorized for programs running in supervisor state that need a fast key switch facility.

Format 2:

```
[name]     MODESET          KEY={ZERO|USER}
```

KEY KEY=ZERO means, that the issuing task wants to run with key 0.

KEY=USER means that the issuing task wants a key switch back from key 0 to the user program key.

The Format 2 is intended for non-privileged programs only and register 1 is used as a work register.

Note: This macro will be restricted by a capability in future.

MODHCF

The macro MODHCF provides the ability to change the direction in which the HCF is to be retrieved.

The macro has the following format:

```
[name]   MODHCF   {(hcfreg)|(1)},{BW|FW|UNC}
```

hcfreg Is the general register containing the address of the HCFCB returned by the corresponding POINTHCF macro.

BW Starting from the current position, changes the direction of the next READ to backwards.

FW Starting from the current position, changes the direction of the next READ to forwards.

UNC Starting from the current position, it reverses the READ direction unconditionally.

> **Note:** If the MODHCF results in an actual change of direction, then the next subsequent READHCF returns the record already provided by a preceding READHCF.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Normal processing successfully completed.
4 (X'04')	Inconsistent input.
8 (X'08')	No record found, incorrect length.
12 (X'0C')	Unrecoverable I/O error.
16 (X'10')	HCF device is not ready.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

MODFLD

The MODFLD macro must be used whenever a field, maintained by the supervisor has to be modified or updated. Each field is described in detail below and is either Partition (PART), Task (TASK) or Device (PU) related.

The MODFLD macro has the following format:

```
[name]     MODFLD   FIELD=name1
                   ,NEWVAL={name2|(r2)|(1)}
                   [,{PART|TASK|PU}={name3|(r3)|(0)}]
```

FIELD= Identification of the field to be modified. Valid symbols and their interpretation are given below. The specification PART, TASK or PU in brackets denotes Partition, Task or Physical Unit specific information and is, as well as the required authorization, given for each valid field specification.

ACLOSE (EOJ only)
A value of 1 if VSAM automatic close is being started for this partition, a value of 0 if VSAM automatic close processing has completed.

CNCLALL (terminator only)
A value of 1, if cancel has to be propagated to all tasks belonging to the same partition as the specified task, else 0 (TASK).

CNCLCODE (EOJ, ACF/VTAM and VSE/POWER only)
Set cancel code. (TASK)

ICCFPP (ICCF only)
A value of 1 if the specified task is assigned to an ICCF interactive partition, else 0 (TASK).

ICCFRO (ICCF only)
A value of 1 if the specified task is assigned to an ICCF interactive partition and is eligible for ICCF roll-out, else 0 (TASK).

ICCFSVC (ICCF only)
A value of 1 if SVCs issued by the task are to be intercepted by ICCF, else 0 (TASK). :dt,LIBRSERV This service is provided for the LIBRARIAN routines to indicate that the LIBRARIAN clean up routines need to be called at EOT. The flag is set (NEWVAL¬=0) for the current task and the corresponding maintask but reset (NEWVAL=0) for the current task only.

MSECS Time slice for partition balancing in milliseconds. The specified value must be within 100 and 1000.

Return codes:

0 (X'00')	Modification completed
1 (X'01')	Specified value not within supported range.

MOUNTFLG (JOB CONTROL only)
A value of 1 if the specified device is to be reserved to allow a volume change, else 0 to indicate that the specified device is to be released.

Return codes:

0 (X'00')	No owner of device or flag set/reset.
8 (X'08')	Multiple device owner or not owned by requestor, flag not set/reset.

OPENSVA A value of 1 if OPEN routines are executing in the SVA for this task, else 0 (TASK).

Note: Cross task requests are not supported.

PASCOPE The addressability scope of the current shared task is changed according to the partition (PIK) specified by NEWVAL. The new scope is shared/private, if the specified partition is shared/private. If the specified partition is executing real, the new scope is for space R. This service is supported only in 370 mode.

Return codes:

0 (X'00')	Successful
8 (X'08')	Not successful, the specified partition is inactive

PERBIT (SDAID)
Modify PER active indication in the partition control block (PCB) and the save area PSW of the specified partition to on or off as specified by NEWVAL-operand (PART).

RUNMODE This service is provided for the INVPART routine to switch the partition specified by PART from virtual to real (NEWVAL=0) and vice versa (NEWVAL¬=0), before and after an EXEC REAL job step. The counter of active virtual partitions (IJBAPNO) and the run mode flag in the PIB (TRAM bit) are updated. If the specified partition was deactivated, it is reactivated before switching to real. In 370 mode, the addressability scope of the partition is switched to the R space or back to the virtual allocation space.
This service is supported only for the current partition in 370 and E mode (EXEC REAL is ignored in VM mode).

SASCOPE The addressability scope of the current shared task is changed according to the space ID specified by NEWVAL. The new scope is private,

if the specified space ID is 'R ' or 'n ', and shared for 'S '.
This service is supported only in 370 mode.

Return codes:

0 (X'00')	Successful
8 (X'08')	Not successful, the specified space is not allocated

SAVAR (IPL only)
Modify the current save area pointer (TASK).

SYSRESW (KEY 0 is required)
A value of 1 if DASD file protection is to be bypassed for this task, else 0 (TASK).

Note: Cross task requests are not supported.

VSAMOPEN (OPEN, CLOSE ONLY)
A value of 1 if there is at least one open VSAM ACB for this partition, else 0 (PART).

VTAMDISP (ACF/VTAM only)
A value of 1 if the ACF/VTAM AP exit is scheduled for this task, else 0 (TASK).

VTAMOPEN (ACF/VTAM only)
A value of 1 if there is at least one open ACF/VTAM ACB for this task, else 0 (TASK).

NEWVAL= The name of a 4-byte field or a register containing the new value to be stored in the specified field. Only the right adjusted significant portion of this argument is used. Register 0 must not be used for register notation.

PART= Name of a 2-byte field or register containing the PIK of the partition to which the specified field belongs. The default value is the requester's PIK.

TASK= Name of a 2-byte field or register containing the TIK of the task to which the specified field belongs. The default value is the requester's TIK (IJBTIK).

PU= Name of a 2-byte field or register containing the physical unit number of the device to which the specified field is to be applied.

This field is mandatory for device related fields (MOUNTFLG).

<u>Register Usage:</u>

R0 Is used to pass the PIK, TIK or PU value. A value of 0 is passed if the PART or TASK operand is omitted.

R1 Is used to pass the new value for the specified field. The value must be right adjusted.

R15 Is used to pass the function code and return code.

MODVCE

This macro indicates to the supervisor the changing of a volume serial number of a DASD device. The supervisor reads the new volume serial number and updates the appropriate entry in the Volume Characteristics Table (VCT).

The macro has the following format:

```
[name]   MODVCE   {LOGUNIT={name1|(r1)}|CHNUNIT={name2|r2}}
                  [,RESERVE={YES,NO}]
                  [,SHARE={YES,NO}]
```

The operands have the following meaning:

LOGUNIT Address of device description in logical unit format.

The volume characteristics entry that the user wants to be updated is identified by its logical assignment. This operand points to a halfword with the same format as a logical unit number in a CCB.

CHNUNIT Address of device description in physical unit (channel, unit) format (as in the PUB).

The volume characteristic entry that the user wants to be updated is identified by its physical device address. This operand points to a halfword with the physical device address.

RESERVE YES: Do not allow the specified device to be assigned until a volume is mounted.

SHARE YES: The device is defined shareable among different CPUs.

Output: Register 15 contains one of the following return codes.

0 (X'00') Request successfully processed.

4 (X'04') The logical unit specified is not assigned.

8 (X'08') The physical unit specified is not in the system or the device is not a DASD.

12 (X'0C') The device is not ready.

16 (X'10') The VOL1 label has not been found or is not valid.

20 (X'14') Some other irrecoverable I/O error occurred.

24 (X'18') The device is not operational.

MSAT

The MSAT macro is used to manipulate stored assignment information.

It has the following format:

```
[name] MSAT  ID={ALT|ALP|CKU|DEL|INQ|NXT|PER|RSA|RSU|RTL|RTP|
                                 DRL|DVR|DVU|NPM|NTM|PSP|PST}
             [,LOGUNIT={name1|(S,name1)|(r1)}]
             [,CHNUNIT={name2|(S,name2)|(r2)}]
             [,PHYUNIT={name3|(S,name3)|(r3)}]
             [,AREA={name4|(S,name4)|(r4)}]
             [,LEN={name5|(S,name5)|(r5)}]
             [,PID={name6|(S,name6)|(r6)}]
             [,MFG={name7|(r7)}]
```

The operands have the following meaning:

ID Specifies the function required

PER The current LUB value is saved as permanent and changed to UA, provided it is the first one to be saved. If there is already a permanent assignment saved, a return will be provided and the saved assignment will not be overwritten.

INQ An indicator byte is returned as leftmost byte of register 15 indicating the types of assignments that have been stored for the specified logical unit. The bits of the indicator byte have the following meaning:

X'80'	Permanent alternate assignment stored
X'40'	Temporary alternate assignment stored
X'20'	Permanent assignment stored
X'10'	Reserved
X'08'	Reserved
X'04'	Reserved
X'02'	Reserved
X'01'	Reserved

If no assignment is stored, return code 0 is given.

DEL All assignments, if any, of the specified logical unit in the specified partition are deleted, device ownership information is updated and the current assignment is set to UA.

ALP The physical device specified in CHNUNIT is noted as permanent, alternate assignment for the logical device specified by LOGUNIT and the device ownership information is updated, but only if the current assignment is also a permanent one. Only one chain of alternates is maintained for SYSPCH and SYSLST, if both are combined to SYSOUT.

ALT The physical device specified in CHNUNIT is noted as temporary, alternate assignment for the logical device specified by LOGUNIT and the device ownership information is updated, but only if the current assignment is also temporary.

NXT The current assignment (value in LUB) is saved as the last alternate one (in the pertinent chain i.e. either temporary or permanent). The first alternate assignment is made the current one provided the associated device is not down. The mode byte is moved from the original current one to the new current one. If the device associated with the new assignment is down, the process is repeated until all alternates have been tried. If all alternate devices are down, return code 4 is returned. If the logical unit specified is SYSPCH or SYSLST and both are combined to SYSOUT then both LUBs are updated with the new alternate assignment found for the logical unit just being processed.

RSU The LUB of the logical unit specified is reset to the saved permanent assignment or unassign. Any temporary, alternate assignments are deleted. For each deleted assignment, the device ownership information is updated.

RSA Starting with the logical unit specified, the LUBs of the higher system logical units or programmer logical units of the specified partition are reset to the saved permanent assign or unassign. Any temporary alternate assignments are deleted. For each deleted assignment, the device ownership information is updated.

CKU Starting with the logical unit specified, all higher system logical units or programmer logical units of the specified partition are checked if they are assigned to the physical device specified in CHNUNIT. If at least one logical unit is assigned to the device, return code 0 is given. Return code 4 indicates that no logical unit in the range is assigned to the device.

NPM (JOB CONTROL only)
All assignments of the specified logical unit are deleted and the specified PHYUNIT is noted as the current permanent assignment. Device ownership is updated for any deleted and new assignment.

NTM (JOB CONTROL, VSE/POWER, ACF/VTAM only)
All temporary assignments of the specified logical unit are deleted, all permanent assignments are saved and the specified PHYUNIT is noted as the current temporary

assignment. Device ownership information is updated for any deleted and new assignment.

PST (VSE/POWER only)
Indicates that the specified device will be used in the specific partition as a dummy unit record device for spooling by VSE/POWER. All assignments are left unchanged, but device ownership for the specified partition is reset.

PSP (VSE/POWER only)
Indicates that the specified device is no longer being used as a VSE/POWER dummy device in the specified partition. All assignments to this device are reset to UA.

DVU (LIBRARIAN, ACF/VTAM only)
Indicates that the specified device is to be accessed by physical addressing in the specified partition. Device ownership information is updated and a 2-byte physical unit number is returned in the specified area.

DVR (LIBRARIAN, ACF/VTAM only)
Indicates that a physical addressing access to the device with the specified physical unit number in the specified partition is released. Device ownership information is updated and the field specified by the PHYUNIT parameter is changed to X'FFFF'. Each such request has to be paired with a previous ID=DVU request.

The following functions ID=RTL|RTP|DRL require the additional parameters AREA and LEN because they return retrieved information in a user defined area. These functions are used by LISTIO and DVCDN command processors.

RTL For the specified logical unit, the current assignment and all stored assignments are returned together with an indication of their type. Output for each assignment is of the form: flag byte/00/PUBindex. The bits of the flag byte have the following meaning:

X'80'	Permanent alternate assignment
X'40'	Temporary alternate assignment
X'20'	Permanent assignment
X'10'	Temporary assignment
X'08'	Reserved
X'04'	Reserved
X'02'	Reserved
X'01'	Reserved

If all retrieved assignments fit into the user specified area, return code 0 is given. Return code 32 indicates that the area is too small. In both cases the number of existing assignments for the logical unit is returned in the first two bytes of the user area.

RTP The LUB index of all higher system logical units or programmer units of the specified partition being assigned to the physical device specified in CHNUNIT is returned together with an indication of the type of assignment. Output for each logical unit in the range is of the form flag byte/00/LUBindex. The bits in the flag byte have the same meaning as for RTL. Several bits may be on if different types of assignment of the logical unit exist for the specified physical device.

If all retrieved assignments fit into the user specified area, return code 0 is given. Return code 32 indicates that the area is too small. In both cases the number of existing assignments for the specified physical device is returned in the first two bytes of the user area.

DRL (JOB CONTROL only)
Any permanent or temporary assignment of the specified logical unit in the specified partition to the specified device is changed to UA and its alternate assignments are deleted. Any alternate assignment to the specified device is deleted. For all changed or deleted assignments, device ownership information is updated. All changed or deleted assignments are returned together with an indication of their type. Output for each assignment is of the format flagbyte/00/PUBindex. The bits in the flagbyte have the following format:

X'80'	Permanent alternate assignment
X'40'	Temporary alternate assignment
X'20'	Permanent assignment
X'10'	Temporary assignment
X'08'	Reserved
X'04'	Reserved
X'02'	Reserved
X'01'	Reserved

The number of affected assignments is returned in the first two bytes of the user area. If all returned assignments fit into the specified area, return code 0 is given. Return code 32 indicates that the area is too small. In this case, all assignments remain unchanged.

LOGUNIT Address of device description in logical unit format. This operand points to a halfword with the same format as a logical unit number in a CCB.

CHNUNIT Address of device description in physical unit format (channel, unit). This operand points to a halfword containing the physical device address.

PHYUNIT Address of a halfword containing a physical unit number (PUB-index) to be used as new current assignment (NPM, NTM) or to be released (DVR). X'FFFF' is interpreted as UA, X'FEFF' as IGN.

AREA Address of the area where the retrieved information is to be returned. Only valid for ID=RTL|RTP.

LEN Length of the area where retrieved information is to be returned. The length specification must be at least two bytes. Only valid for ID=RTL|RTP.

PID Points to a two byte field containing the PIK of the partition the logical unit belongs to. The default is the PIK of the issuing partition.

MFG Address of dynamic storage area that is to be used for construction of parameter list for reentrant programs.

Output: Register 15 contains one of the following return codes.

0 (X'00')	No permanent assignment stored (RSU). No assignment to device found (CKU). No alternates present or all devices down or all devices assigned to same physical unit (NXT).
4 (X'04')	Requested function complete.
8 (X'08')	No more space available (ID=ALT|ALP|PER).
12 (X'0C')	Status of alternate assignment incompatible with status of current assignment (ID=ALT|ALP). Permanent assignment already saved (ID=PER). Device is already spooled (PST) or not spooled (PSP). Device not in use by specified partition (DVR).
16 (X'10')	Logical unit specified exceeds the range of logical units
20 (X'14')	Physical unit specified not supported in the system. Physical unit specified is not a unit record device (PST,PSP).
24 (X'18')	Partition specified not supported by the system.
28 (X'1C')	Function requested not supported.
32 (X'20')	User area too small.
36 (X'24')	Specified device is already owned by or reserved for another partition (ALP|ALT|NPM|NTM|DVU).
40 (X'28')	Specified device is down (ALP|ALT|NPM|NTM|DVU).

The following list shows which operands are required with the different IDs:

ID	LOGUNIT	CHNUNIT	PHYUNIT	AREA	LEN	PID	MFG
PER	R	N/A	N/A	N/A	N/A	O	O
INQ	R	N/A	N/A	N/A	N/A	O	O
DEL	R	N/A	N/A	N/A	N/A	O	O
ALP	R	R	N/A	N/A	N/A	O	O
ALT	R	R	N/A	N/A	N/A	O	O
NXT	R	N/A	N/A	N/A	N/A	O	O
RSU	R	N/A	N/A	N/A	N/A	O	O
RSA	R	N/A	N/A	N/A	N/A	O	O
CKU	R	R	N/A	N/A	N/A	O	O
RTL	R	N/A	N/A	R	R	O	O
RTP	R	R	N/A	R	R	O	O
NPM	R	N/A	R	N/A	N/A	O	O
NTM	R	N/A	R	N/A	N/A	O	O
DRL	R	R	N/A	R	R	O	O
PST	N/A	R	N/A	R	R	O	O
PSP	N/A	R	N/A	R	R	O	O
DVU	N/A	R	N/A	R	N/A	O	O
DVR	N/A	N/A	R	N/A	N/A	O	O

R = Parameter is required O = Parameter is optional N/A = Parameter not applicable

NPGR

The NPGR macro causes the number of programmer LUBs to be set to the specified value. The macro is used by the job control NPGR command processing routine. The format of the NPGR macro is as follows:

```
[name]  NPGR   [NPGRLST={name1|(r1)}]
```

NPGRLST Is the address of the parameter list into which the specified operands have to be placed before issuing the macro. The address of the parameter list may be supplied either as an operand or in a register. If the operand is omitted, register 1 is used.

The format of the parameter list is as follows:

```
NPGRLST
| NPGRNOP | NPGRPID | NPGRVAL | reserved | NPGRPID | NPGRVAL |
 0         1         2         4          5         6        7
                               |<------------------------------>|
                             Repetitive; depending on the number
                             of elements specified in the command.
```

NPGRNOP It contains the number of operands which were specified in the NPGR command and which is equal to the number of elements in the parameter list.

NPGRPID It specifies the partition number from 0 to n for which the following number of programmer LUBs is to be allocated, where 0 is the background, and 1 to n are the foreground partitions F1 to Fn.

NPGRVAL It contains the number of programmer LUBs.

Output: Register 15 contains one of the following return codes:

0 (X'00') The specified partition programmer LUB values are accepted.

8 (X'08') The NPGR command is rejected. The sum of all partition programmer LUBs is larger than the supervisor generated NPGR Value.

12 (X'0C') The NPGR command is rejected. At least one of the specified NPGR value is either below the minimum of 20 or above the maximum of 255.

16 (X'10') The NPGR command is rejected. At least one of the specified partition has been started before (may be unbatched now).

20 (X'14') The NPGR command is rejected. NPGR for BG was specified but another partition was already started before (may be unbatched now).

24 (X'18') The NPGR command is rejected. Reallocation of BG LUBs is below highest assigned BG LUB.

28 (X'1C') The NPGR command is rejected. A partition was specified, which is not supported.

NPGRLST

The macro NPGR generates a NPGR parameter list. Its format is as follows.

```
[name]  NPGRLST   [DSECT=YES]
```

DSECT   YES: Forces the layout of the parameter list to be generated.

PAGESTAT

The PAGESTAT macro returns the status of an area. Each page of the specified area is checked for validity and whether the page is used or not.

The macro has the following format:

```
[name]   PAGESTAT    {name1|(0)},{name2|(1)}
```

name1 Address within the first page to be handled.

name2 Address within the last page to be handled.

Output: Register 15 contains return information as described below.

On return byte 0 contains an identification about the status of the first page of the specified area and bytes 1-3 contain the address of the first page of the area that has a different status.

If bytes 1-3 of register 15 contain zeros all pages of the area have the same status indicated by byte 0.

Byte 0=0	Address is valid and page is used.
Byte 0=4	Address is valid and page is not used.
Byte 0=8	Address is invalid which means that the appropriate task is canceled whenever it attempts to reference this page or address. This may be due to one of the following reasons: • Address beyond virtual storage. • In ECPS:VSE mode: page belongs to a partition in 'real' mode and page is not addressable. • In 370 mode: HABIT (bit 10) and IBIT (bit 12) are on in corresponding page table entry.

Execution of this macro with name1 higher than name2 results in 'cancel due to invalid address' (ERR25).

PFIXCHPT

The macro ensures that during checkpointing the parameter list for PFIXREST is built (see also "SVC 74 (X'4A' - PFIXCHPT/PFIXREST)" on page 57). The function is restricted to programs with a PSW key of zero.

The macro has the following format:

```
[name]   PFIXCHPT   {name1|(1)},{length|(0)}
```

name1 The symbolic name of an area where the entries have to be inserted. For the layout of the entries refer to the description of the "SVC 74 (X'4A' - PFIXCHPT/PFIXREST)" on page 57.

length Length of the area provided by the user.

Note: A length of 0 is not allowed.

Output: Register 2 contains zero if no additional area is needed; register 2 contains 4 if an additional area is needed. A non-zero byte is placed right after the last generated entry in each area.

PFIXREST

The macro ensures that during RESTART the pages which were permanently fixed at checkpoint time are PFIXed again with the same value of the PFIX counter (see also "SVC 74 (X'4A' - PFIXCHPT/PFIXREST)" on page 57). The function is restricted to programs with a PSW key of zero.

The macro has the following format:

```
[name]  PFIXREST {name1|(1)}
```

name1 Is the symbolic name for a list of consecutive 6-byte entries built during checkpointing. For the layout refer to the description of the "SVC 74 (X'4A' - PFIXCHPT/PFIXREST)" on page 57.

POINTHCF

The macro POINTHCF opens the HCF and provides the interface for accessing the HCF. According to the parameters passed, the logical record pointer is initialized and the specific logical file limits are determined.

The macro has the following format:

```
[name]   POINTHCF {WRITE[,{CONTINUE|CREATE}]|
                  READ,{LISTLOG|REDISPL|PRINTLOG[,{ALL|NEW}]
                       [,{NOSELECT|SELECT}]}}
                 ,SAVE={name|13}
```

WRITE   Opens the HCF for the output.

CONTINUE This option ensures writing onto the HCF behind the last previously written record.

CREATE   Initializes the HCF (as a result of the SET RF=CREATE command).

READ   Opens the HCF for the input.

LISTLOG Opens the HCF for the LISTLOG utility program.

REDISPL Opens the HCF for the redisplay function of CRT.

PRINTLOG Opens the HCF for the PRINTLOG utility program.

> ALL   All available data is to be retrieved.
>
> NEW   Only the new data (since last IPL) is to be retrieved.
>
> NOSELECT No special records are to be selected.
>
> SELECT   Special record selection is to take place.

SAVE   Specifies a 68-byte register save area in which the caller's registers are saved.

Output: Register 1 points to the control block (HCFCB), and this address must be provided in subsequent READHCF, MODHCF, SKIPHCF or CLOSEHCF requests.
If POINTHCF returns in error, register 1 will be set to zero.
For POINTHCF with WRITE, the HCFCB address is not returned in register 1.

Register 15 contains one of the following return codes.

0 (X'00')	Normal processing successfully completed.
8 (X'08')	No record found, incorrect length.
12 (X'0C')	Unrecoverable I/O error.
16 (X'10')	HCF device is not ready.
20 (X'14')	READ HCF not yet opened. WRITE Incorrect HCF format.
24 (X'18')	Unauthorized POINTHCF request.
28 (X'1C')	HCF not accessible. Byte 0 of register 15 contains the original return code of the GETVCE/GETVIS/EXTENT request. READ No JOB statement for partition, or EXTENT failed. Byte 0 of register 15 is zero if no JOB statement was detected for the partition. WRITE Open unsuccessful, or GETVCE failed. Byte 0 of register 15 is zero if OPEN was unsuccessful.
32 (X'20')	HCF too small.
36 (X'24')	GETVIS failed. Byte 0 of register 15 contains the original return code of the GETVCE/GETVIS/EXTENT request.
40 (X'28')	HCF does not meet extent specifications.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

The format is as follows:

```
[name]   PWROFF
```

There are no operands.

The PWROFF macro allows authorized subsystems (SSX) to power-off a 4361 CPU via a SVC interface.

If there is no error situation, the supervisor does not pass control back to the requestor.

Cancel conditions:

The requestor is cancelled with 'Invalid SVC' (Error 21) if he is not authorized.

Output: Register 15 contains the following return code:

X'08' Is returned if the supervisor is running under VM or the CPU is not a 4361.

Register Usage: None.

READHCF

The macro READHCF ensures that one HCF record is provided in the I/O area, specified by the second operand. The direction of the READ operation depends on the associated POINTHCF macro which implies this information or it depends on the last MODHCF macro given (if any).

The macro has the following format:

```
[name]   READHCF   {(hcfreg)|(1)},{ioarea|(0)}
```

hcfreg Is the general register containing the address of the HCFCB control block returned by the corresponding POINTHCF macro.

ioreg Is the symbolic name of the I/O area to where the HCF record is to be moved.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Normal processing successfully completed.
4 (X'04')	Inconsistent input.
8 (X'08')	No record found, incorrect length.
12 (X'0C')	Unrecoverable I/O error.
16 (X'10')	HCF device is not ready.
20 (X'14')	Begin of HCF (first record in file).
24 (X'18')	End of HCF (last record in file).
28 (X'1C')	Record has already been overwritten by a new one.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

RLOCK

The RLOCK macro obtains access to a specified resource. If the resource is not available, the issuing task is set into the appropriate resource-bound condition.

The macro has the following format:

```
 [name]   RLOCK COND={name1}
```

The operands have the following meaning:

COND= Specifies the resource that the requestor wants to access.

ALLOCR Allows to lock or to wait for the access to fields related to the system 'real partition' and will be used in 370 and ECPS:VSE mode by the ALLOCR sub-function of ALLOCATE.

CRTSAV (OCCF only)
Indicates that the CRT save area has to be accessed.

HCFCB (OCCF only)
Indicates that the CRT hard copy file has to be accessed.

RSGT Allows to lock or to wait for the access to the segment table of the R-space and will be used by the INVPART routine for EXEC REAL in 370 mode.

Register Usage: R15 is used to pass the function code.

SECHECK

The SECHECK macro can be used to perform an access control check. The issuer of that macro is checked whether he is authorized to access the specified resource or not. After execution of the macro register 15 contains the return code.

The macro has the following format:

```
[name]   SECHECK    AREA={name1|(r1)}
                  [,NAME={name2|(r2)}]
                  [,TYPE={LIB|SUBLIB|MEMBER|FILE}]
                  [,MODE={READ|UPDATE|CONNECT|ALTER}]
                  [,RETN={NO|YES}]
```

AREA Points to the Authorization Parameter List (see DTSAPL macro). It is a 24-byte control block which contains information that can be specified by the other four parameters of the SECHECK macro.

NAME Specifies the name of the resource to be checked. The length of the name depends on the TYPE specification.

TYPE Specifies the type of the resource to be checked. TYPE is required when NAME is specified.

LIB The library is to be checked. The length of the resource name is 57 bytes (containing VOLID, FILE-ID and library-name).

SUBLIB The sublibrary is to be checked. The length of the resource name is 15 bytes (containing library-name and sublibrary-name).

MEMBER The member is to be checked. The length of the resource name is 23 bytes (containing library-name, sublibrary-name and member-name).

FILE The whole file is to be checked. The length of the resource name is 50 bytes (containing VOLID and FILE-ID).

MODE Specifies the access mode of the specified resource.

READ The user requires READ access

UPDATE The user requires WRITE access

CONNECT The user requires the authorization to ACCESS a member in a library or sublibrary. Applicable only for TYPE=LIB or TYPE=SUBLIB.

ALTER The user requires the authorization to CREATE or DELETE a library or sublibrary. Applicable only for TYPE=LIB or TYPE=SUBLIB.

RETN YES: Specifies that control is to be returned to the user after an access control violation.

NO: (default) Specifies, that the job is to be canceled in case an Access control violation is determined.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Access allowed
4 (X'04')	No access control support
8 (X'08')	Access control violation
12 (X'0C')	In a protected library: the sub-library is not in the access control resource table (DTSECTAB). In a protected sub-library: the member is not in the DTSECTAB.

SENTER

Pass control to the entry point of a predefined SVA-resident phase of a system component and associate the component capability with the issuing task.

The address of the instruction following the macro call is passed to the entered phase in register 14, the entry point itself in register 15. SENTER is therefore equivalent to a BALR 14,15 instruction. All other registers are passed unchanged.

The macro has the following format:

```
ASSEMBLER:

  [name]   SENTER   LIBR

PLS:

 ?[name:]  SENTER(LIBR);
```

LIBR   The librarian component is to be entered. The assumed entry phase is a module within phase $IJBLBR.

Register Usage:

R14	Return address
R15	Function code

SETLIMIT

The SETLIMIT macro (see also SVC 84) changes partitions sizes. The macro is used by the job control SIZE command processing, the attention SIZE command processing, and job control EXEC statement processing, if the SIZE parameter is specified. It may also be used by other system components if applicable.

The format is as follows:

```
[name]    SETLIMIT   [SLPL={name1|(1)}][,MODE=({PERM|TEMP},{V|R})]
```

SLPL   Defines the parameter list into which specified operands have to be placed before issuing this macro. The address of the parameter list may be supplied either as an operand or in register 1.

The parameter list has the following format:

```
┌──────────┬──────────┐
│ SLPLPID  │ SLPLSIZE │
│          │          │
└──────────┴──────────┘
0          2          3
```

SLPLPID It specifies the partition ID 'BG' or 'Fn'. X'FFFF' means that the operand is omitted, and that the limits are reset for the partition issuing the command.

SLPLSIZE It specifies in K-bytes the amount of contiguous virtual storage of a partition which is used for job execution. The remaining space is the partition GETVIS area. X'FFFF' means that the minimum SIZE value should be taken.

To generate the layout of the parameter list, the SLPL macro may be used:

```
[name]    SLPL   [DSECT=YES]
```

MODE   indicates whether the limit is to be changed permanently or temporarily, and whether it is a virtual or real mode partition.

PERM,V The limit is to be changed permanently. A permanent limit value is applicable for a virtual partition only. It is retrieved from the SIZE operand.

PERM,R Invalid specification. For a real partition the limit can be changed temporarily only.

TEMP,V The limit is to be changed temporarily for a partition which will execute in virtual mode.

TEMP,R The limit is to be changed temporarily for a partition which will execute in real mode. For a real partition the limit can be changed temporarily only. Its value is submitted on the job control EXEC statement.

If the MODE parameter is omitted, the parameter is expected to be supplied via register 0.

A value of zero means PERM,V; a value of one means TEMP,V; a value of three means TEMP,R.

Output: Register 15 contains one of the following return codes:

0 (X'00')	The specified limits are stored.
8 (X'08')	The new SIZE limit is not stored. The partition occupies at present dynamic storage. Re-issue the command right after End-Of-Job or End-Of-Job Step.
12 (X'0C')	The new SIZE limit is not stored. The SIZE specification does not leave the minimum GETVIS space. Reduce the SIZE value for this partition.
16 (X'10')	The new SIZE limit is not stored. The SIZE value exceeds the virtual storage of the partition. Reduce the SIZE value for this partition.
20 (X'14')	The new SIZE limit is not stored. The address space specified by the SIZE value is below minimum. Increase the SIZE specification for this partition.

SGENL

The SGENL macro provides the ability to generate a local directory list of SDL-like directory entires and is intended to be used by the librarian only.

The macro has the following format:

```
 [name]   SGENL  name1(,name2(,name3(,... ))
```

name1 Name of a phase that is to be included in the local directory list.

Up to 200 phase-names may be specified. The phase-names will be alphamerically sorted by the macro expansion.

SGETVIS

The macro SGETVIS allows supervisor components to request System GETVIS (SVA) storage.

- Registers 0, 1 and 15 are destroyed.
  All other registers are saved in, and restored from the TCB - SVC work area (SVCSV3), or in the area specified by the SAVE operand.

- It will be forced that a requested area will not cross a page boundary unless the requested area is larger than a page.

The format of the SGETVIS macro is as follows:

```
[name]   SGETVIS  [LENGTH={name1|(0)}]
                  [,SPID={name2|(1)}]
                  [,ADDRESS={name3|(1)}]
                  [,SAVE={name4|(r4)}]
                  [,PAGE={YES|NO}]
                  [,PFIX={YES|NO}]
                  [,FTCHPR={YES|NO}]
                  [,EXCREQ={YES|NO}]
```

SAVE Is the area where the requester wants the general registers to be stored (in case the TCB SVC work area is not available). The save area must be 18 fullwords long, according to the first part of the DSECT SVEARA.

FTCHPR Specifies whether the area is to be fetch protected.

YES The corresponding GETVIS storage will be fetch-protected. Fetch-protection is a property of the subpool, i.e. for all requests for that subpool FTCHPR=YES must be specified.

NO The corresponding GETVIS storage will not be fetch-protected.

EXCREQ Specifies whether the requestor may use SVA GETVIS space excessively.

YES The requestor identifies itself as a SVA GETVIS user, who may occupy SVA GETVIS space in an excessive manner, triggered by various user functions such as FETCH, LOCK, XECB, etc. Requestors, who specify this parameter, should tolerate a GETVIS return code for this request. The GETVIS function will check in this case, if the current GETVIS request exceeds a predefined high water mark (defined by the global AGMAXEX in IOTAB).

NO The Requestor may exceed high water mark (default value).

Note: All other operands have the same meaning as described in the GETVIS macro, VSE/Advanced Functions, Application Programming: Macro Reference, SC33-6197.

SKIPHCF

The macro SKIPHCF provides the ability to skip the number of specified records, or to skip to the end or the begin of of the file, depending on the current direction of reading (initially set by POINTHCF or set by MODHCF). Skipping to the end or to the begin of the file implies an unconditionally READ direction change.

The macro has the following format:

```
[name]   SKIPHCF  {(hcfreg)|(1)},{{count|(0)}|EOF}
```

hcfreg Is the general register containing the address of the HCFCB control block returned by the corresponding POINTHCF macro.

count Is the symbolic name of a 2-byte field containing the number of records to be skipped or the value is given in a register. A negative number is not allowed.

EOF Forces a skip to the begin-of-file if the direction of reading is backward or to the end-of-file if the direction of reading is forward.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Normal processing successfully completed.
4 (X'04')	Inconsistent input.
8 (X'08')	No record found, incorrect length.
12 (X'0C')	Unrecoverable I/O error.
16 (X'10')	HCF device is not ready.
20 (X'14')	Begin of HCF (first record in HCF). Register 0 contains the number of records that the service was unable to skip due to this begin-of-file.
24 (X'18')	End of HCF (last record in HCF). Register 0 contains the number of records that the service was unable to skip due to this end-of-file.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

SLEAVE

Release capability currently associated with issuing task and, optionally, return to caller with specified return code.

Must be issued by system components called via the SENTER macro. If the RETURN parameter is specified (see below), SLEAVE is equivalent to a   BR 14   with a return code in register 15.

The macro has the following format:

```
ASSEMBLER:

   [name]   SLEAVE [RETADD={name1|(r1)|(14)}][,RETCOD={name2|(r2)|(0)}]

PLS:

  [name:]   SLEAVE [RETADD{(name1)|(r1)|(14)}][ RETCOD{(name2)|(r2)|(0)}];
```

RETADD   Name of a fullword or register containing the address to which control is to be passed after execution of LEAVE. If the parameter is omitted, control is returned to the next sequential instruction.

RETCOD   Name of a fullword or register containing the return code to be passed in register 15. If the parameter is omitted, a return code of zero is assumed.

<u>Return Codes:</u>

As specified by RETCOD.

<u>Register Usage:</u>

R0:   Return code for input.
R14:   Return address for input and output
R15:   Function code for input, return code for output.

SLOAD

The SLOAD macro can be used to LOAD a phase into the SVA, or to retrieve SDL-like directory entries.

The macro has the following format:

```
[name]   SLOAD  phname|(r1)[,{loadadr|(r0)}]
                [,DE={YES|NO|SDLFORM|VSEFORM}]
                [,TXT={YES|NO}]
                [,SYS={YES|NO}]
                [,SVAUPD={NO|YES}]
                [,SDL={YES|NO}]
                [,RET={NO|YES}]
                [,MFG={name1|(S,name1)|(r2)}]
                [,LIST={name2|(r3)}]
```

The operands have the following meaning:

phname   Name of the phase that is to be loaded.

loadadr   Address where the phase is to be loaded.

DE   Directory entry information

This option should be used to determine whether a phase is available in the system and / or to avoid the directory search in case the phase is to be loaded more than once during program execution.

NO   No directory entry is available.

YES   A valid directory entry in the length of 38 bytes and the indication X'0000000D' in offset 8 is provided.

VSEFORM   A VSE directory entry in the length of 40 bytes with the indication X'FFFFFF0E' in offset 8 has been provided. VSEFORM may be abbreviated VSE and it must be used when the System Directory List (SDL) is somehow affected by the SLOAD function.

SDLFORM   A SDL (System Directory List) like directory entry of the length of 68 bytes has been provided. SDLFORM may be abbreviated SDL.

TXT   Text processing information.

NO   The phase is not to be loaded. Useful with DE=YES to determine whether the requested phase is available and/or for accessing directory entry information only in case the phase is to be re-loaded during program execution.

YES The phase is to be loaded. The search sequence is taken as for $-phases.

SYS Search sequence information.

YES Search sequence as for $-phases.

NO The normal search sequence which is: SDL, temporarily concatenated sublibraries (if any), permanently concatenated sublibraries (if any) and at last SYSLIB sublibrary is taken.

SDL

YES Default value. The currently active library concatenation chain for this partition is to be used.

NO The directory entry is or will be a part of the SDL, the SDL is not searched.

SVAUPD

NO Default value. The phase is to be loaded into the user partition.

YES The phase is to be loaded into the SVA and the associated SDL entry is to be updated accordingly.

LIST If specified, a pointer to a local list of directory entries is passed. It is recommended to generate this list by the GENL macro for directory entries in VSE format, respectively by the SGENL macro for entries in the SDL format.

RET Return error information

NO Default value The user does not want return codes to be passed back. The user Program will be canceled in case of permanent errors.

YES The user request return codes to be passed back in register 15. The service caller is not canceled in the case of error situation.

0 (X'00')	SLOAD completed successfully
4 (X'04')	Phase not found (preventing cancel code X'22'). Reasons: 1. No directory entry was found during directory search. 2. A directory entry was provided by the user but the corresponding phase is already deleted (the directory search is not restarted).

8 (X'08') Unrecoverable I/O error during SLOAD service (preventing cancel code X'2B').

12 (X'0C') Invalid library structure detected by SLOAD (preventing cancel code X'29').

16 (X'10') Invalid address provided by SLOAD caller (preventing cancel code X'25').

20 (X'14') Security violation (preventing cancel code X'0B').

24 (X'18') Inconsistent directory entry. SLOAD cannot use the directory entry and confirms an inconsistency between provided directory entry and sublibrary entry. The provided DE is replaced by the sublibrary entry. This return code may by used by programs with own storage management to ensure that the SLOAD service does not overwrite any storage when a phase has been replaced by a longer version in the meantime.

28 (X'1C') Phase does not fit in partition OR LTA (Logical Transient Area, preventing cancel code X'28').

MFG Macro format information.
Address of a sufficiently large work area (due to the various directory entry formats) where the parameter list is to be generated.

Notes:

1. All registers must be different from each other and register 0 must not be used.
2. If the phase name is specified via register notation, a valid directory entry as specified by DE must be provided.
3. The SVA parameter is still allowed but has no effect.

SRCHFLD

Return the physical unit number of a device which is identified by its cuu-address or by its device type code.

The physical unit number of the matching device is returned right adjusted in register 1.

The macro has the following format:

```
ASSEMBLER :

   [name]   SRCHFLD FIELD={CHNUNIT|DEVTYP},VALUE={name1|(r1)|(1)}
                    [,PU={name2|(r2)|(0)}]

PLS :

  ?[name:]  SRCHFLD FIELD(CHNUNIT|DEVTYP) VALUE{(name1)|((r1))|(1)}
                   [PU{(name2)|((r2))|(0)}];
```

FIELD Symbolic identification of the field to be searched for.

CHNUNIT 2-byte channel and unit address in the form cuu.

DEVTYPE 1-byte device type code.

VALUE Name of a 4-byte field or a register containing the right adjusted value of the field to be searched for.

Register 0 may not be used for register notation.

PU Name of a 2-byte field or register containing the physical unit number (same as PUB-index = PUB-offset/8) of the device at which searching has to start. Search is in ascending order and stops at the first match. If this parameter is omitted or zero, search starts with the lowest physical unit number.

Output:

Register 15:	contains one of the following return codes.	
	0 (X'00')	An appropriate pub index was returned in register 1.
	4 (X'04')	No matching PUB found.
Register 1:	Physical unit number	

Register Usage:

R0: Physical Unit number for input.
R1: Search argument for input
R15: Function code for input

STARTP

The STARTP macro is used to call the phase $IJBSTR5 which starts the partition requested in the macro.

The macro has the following format:

```
[name]   STARTP PART={number|(r1)}
                SAVE={address|(r2)}
```

The operands have the following meaning:

PART This keyword operand specifies a partition number (0 for BG, 1 for F1, etc.) If number is specified, the operand contains a decimal number up to 15. If (r1) is specified, the item within the brackets specifies a register that contains a binary value.

SAVE If address is specified, the operand contains the assembler label of an 18-fullword save area. If (r2) is specified, the item within the brackets specifies a register that contains the address of the save area at execution time.

Output: Register 15 contains one of the following return codes:

0 (X'00')	Successful execution
4 (X'04')	Area not available
6 (X'06')	No ASI bit on for partition to be started
8 (X'08')	No GETVIS in issuing partition

SUBSID

The SUBSID macro must be used to keep the supervisor informed about the state (active, inactive) of a specific subsystem thus allowing other subsystems to inquire the state of another subsystem or the currently loaded supervisor itself.

The following table lists the SUBSID names defined for the subsystems which are handled by the supervisor. The four byte SUBSID name is a unique name and must always be used for the SUBSID services.

Subsystem	SUBSID Name	Active Once in	Parameter passed 'SPARM'	
VSE/POWER	PWR	System	No	-
ACF/VTAM	VTAM	System	No	-
ICCF	ICCF	System	No	-
SUPERVISOR	SUP	System	No	-
CICS	CICS	Partition	Yes	Ptr to IJBAFCB
SQL/DS	ARI	Partition	No	-
OCCF	OCCF	System	Yes	Ptr to OCCF COMREG in IJBOCFCM
VM/VCNA	VCNA	System	No	-
SPF	ISPF	Task	No	-
DL/1	DLI	Task	No	-
NPDA	NPDA	System	No	-
SSX	SSX	Partition	No	-
FTP	FTP	Task	No	-

The macro has the following format:

```
[name]   SUBSID   {NOTIFY|REMOVE|INQUIRY}
                  ,NAME={name1|(r1)|(S,name1)}
                  [,SPARM={name2|(r2)|(S,name2)}]
                  [,AREA={name3|(r3)|(S,name3)}
                  ,LEN={n|name4|(r4)|(S,name4)}
                  [,PID={name5|(r5)|(S,name5)}]
                  [,MFG={name6|(r6)}]
                  [,LVLTEST={NO|YES}]]
```

NOTIFY The subsystem NOTIFies the supervisor about the existence of itself. This option always applies to the issuing task/partition.

REMOVE The subsystem notifies the supervisor that it is to be considered inactive from now on.

INQUIRY This option can be used to determine the state and probably the level of a defined subsystem or the supervisor itself.

The layout of the supervisor entry is described by the mapping DSECT generated by the macro MAPSSID.

The format is as follows:

```
[name]   MAPSSID
```

DEC	HEX	Label	Description
0 – 1	0 – 1	IJBSSID1	Partition id
2 – 5	2 – 5	IJBSNAME	Program name
6	6	IJBSVERS	Version number
7	7	IJBSREL	Release number
8	8	IJBSMOD	Modification number
9	9	IJBSVARL	Length of variable part
10	A	IJBSFLAG	Flags (varying length)
		IJBSFL01	Flag byte 1
		IJBSF370	X'80' 370 support
		IJBSFEXX	X'40' E support
		IJBSFCKD	X'20' CKD support
		IJBSFFBA	X'10' FBA support
		IJBSFAPR	X'08' 3800 support
		IJBRCHAN	X'04' Relocating channels
		IJBSVMLE	X'02' VMLE support generated
		IJBSVMAC	X'01' Running on VM
11	B	IJBSFL02	Flag byte 2
		IJBSFAF	X'80' AF support
		IJBSFPAG	X'40' 4K page size used
			X'20' Reserved
			X'10' Reserved
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved

Figure 329 (Part 1 of 2). Output as Described by Macro MAPSSID

DEC	HEX	Label	Description
12	C	IJBSFL03	Flag byte 3
		IJBSFSEC	X'80' Security support
		IJBSFSHR	X'40' DASD sharing support
		IJBSFSAT	X'20' JIB replaced by SAT
			X'10' Reserved
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
13	D	IJBSFL04	Flag byte 3
			X'80' Reserved
			X'40' Reserved
			X'20' Reserved
			X'10' Reserved
			X'08' Reserved
			X'04' Reserved
			X'02' Reserved
			X'01' Reserved
14 - 15	E - F	IJBSLCON	Library concatenation chain length
10	A	IJBSFIXL	Length of fixed part
16	10	IJBSSLEN	Total length of DSECT

Figure 329 (Part 2 of 2). Output as Described by Macro MAPSSID

NAME Is the address of a field describing the subsystem

The field contains information such as:

DEC	HEX	Description
0 - 3	0 - 3	Name field containing the unique subsystem name (this field is the only one required for REMOVE or INQUIRY)
4 - 6	4 - 6	Subsystem specific information (applies to NOTIFY only)
7	7	Containing the length (0-24 bytes) of the optional variable part appended to the required fixed part (byte 0 - 7 — applies to NOTIFY only)
8 - 31	8 - 1F	Maximum of 24 bytes containing subsystem parameters. This field may contain such information as version and release number as well as features supported by this subsystem. (applies to NOTIFY only)

Figure 330. Subsystem Descriptor

SPARM Is the address of a field containing information that is to be saved in a predefined field within the supervisor. This operand is accepted from special subsystems only (see table above)

AREA Is the address of an area where the user want the information describing the subsystem to be stored.

The returned information does contain the PIK (byte 0-1) followed by at least 8 bytes (byte 2-9) containing the subsystem specific information as passed with the NOTIFY option. (Macro MAPSSID describes the layout of the supervisor entry).

LEN Specifies the area length either as an integer, a self-defining term, or as value in a register, or, in S-type notation as the name of a halfword containing the value.

The value must be in the range from 10 to 34 bytes.

PID Is the address of a halfword containing the PIK of the partition which is to be interrogated whether the specified subsystem is active. In case this operand is omitted, or if the PIK is zero, the whole internal subsystem table is scanned until the first matching entry is found which will then be returned to the requester.

MFG Specifies the address of a work area where to build the parameter list.

LVLTEST YES: Generates code which ensures that the IPLed supervisor does support the SUBSID.

Output: Register 15 contains one of the following return codes.

NOTIFY

0 (X'00')	Information stored
4 (X'04')	Subsystem name already exists in system
8 (X'08')	Byte string too long, SUBSID rejected
12 (X'0C')	Subsystem table is full
16 (X'10')	Subsystem name not known to supervisor

REMOVE

0 (X'00')	Information for the specified subsystem removed
4 (X'04')	No matching subsystem entry found

INQUIRY

0 (X'00')	Information returned
4 (X'04')	Information returned, however, the same subsystem is currently active in another partition. Register 0 contains the PIK of that partition in its high-order two bytes.
8 (X'08')	Returned information truncated, the area is too short. Register 0 contains the total length in the low-order two bytes.
12 (X'0C')	Return codes 4 and 8 together.
16 (X'10')	Name not found in subsystem table.
20 (X'14')	Supervisor does not support SUBSID service (LVLTEST=YES only)

Register Usage:

R0 contains the function code and return information.
R1 pointer to the parameter list, byte string or name.
R15 pointer to the special parameter and return code register.

SUPRET

The SUPRET macro is used by all A-transient routines (error recovery and recording transients) to properly return to the supervisor.

The macro has the following format:

```
[name]   SUPRET  [ENTRY=][,PLSBASE=]
```

The operands have the following meaning:

ENTRY Identifies to the supervisor the type of transient that wants to return control to the supervisor (error recovery or recording transient). If nothing is specified, an error recovery transient is assumed. A recording transient must specify ENTRY=AE .

PLSBASE Indicates for PL/S coding the register containing the address of the error block, if the PL/S version of the ERBLOC macro has been used. The register must be enclosed by parentheses.

Output: In-line code for return to supervisor is generated.

SVALLIST

This macro produces the assembler source code of a load-list. The load-list contains the names of the phases that are to be loaded automatically into the SVA during IPL under control of one or more load conditions.

The macro has the following format:

```
[name]   SVALLIST  [llname,]
                   (phname1[,cond1][,cond2]...[,condn]),
                  {,(phname2[,cond1][,cond2]...[,condn])
                  {,(phnamen[,cond1][,cond2]...[,condn])}}
```

The operands have the following meaning:

llname Specifies the phase name of the LOADLIST to be generated. This name has to be predefined in the master LOADLIST $$A$SVA. Llname is mandatory in the first SVALLIST macro call within one assembly; subsequent SVALLIST macro calls ignore the llname specification and assume to be a continuation of the first macro call or load-list, respectively.

phname Specifies the names of the phases that are to be included into LOADLIST specified by llname.

cond Specifies the condition that the IPL'ed supervisor must meet in order to get the corresponding phase automatically loaded into the SVA.

FBA Supervisor must provide FBA support.
RPS Supervisor must support rotational position sensing
SEC Security was activated at IPL time.
If none of the load conditions is specified the corresponding phase is loaded unconditionally.

SYSIO

The SYSIO macro requests the initiation of an I/O operation ahead of all other I/O operations requested by EXCP(SVC 0). It will observe the priority for headqueuing assigned to the different system tasks. The WAIT for the completion of the I/O request is implied when using SYSIO.

The macro has the following format:

```
[name]  SYSIO  {name1|(1)}
```

The operands have the following meaning:

name1 Is the address of a CCB or IORB established for the device. It can be given as a symbol or in register notation.

Output: The traffic bit in the CCB or IORB is posted when the system task gets back control. If an error occurred the disastrous error indicator in the CCB is posted and must be checked by the system task. Transient error recovery procedures are skipped for headqueue requests and headqueue errors will not be recorded onto the record file. A normal user task issuing SYSIO or SVC 15 is canceled due to illegal SVC, cancel code 21.

TRANSCCW (370 Mode Only)

The TRANSCCW macro returns the address of the original users CCW that corresponds to the address of a given, copied CCW (only in 370 mode, illegal SVC in ECPS:VSE mode - see "Channel Program Translation (370 Mode)" on page 157). It is intended to be used by the ERP routines only.

The macro has the following format:

```
[name]  TRANSCCW  {ccwaddr|(0)},{ccbaddr|(1)}
```

The operands have the following meaning:

ccwaddr Contains the copied CCW address.

ccbaddr Contains the address of the copied CCB. The address may be passed in any register except R0.

Output: Register 0 points to the address in the user's program.

TREADY

The TREADY macro must be used to set a specified task "ready-to-run" which includes the ability to abnormally terminate the task(s).

The macro has the following format:

```
[name]   TREADY COND={LQ|VCANCEL|OCCF|CRTSAV|HCFCB|ATTINT|ALLOCR|RSGT}|

                PART={name1|(r1)|(0)}
                [,COND={START|OC|
                       {CANCEL,CODE={name2|(r2)|(1)}}]|

                TASK={name3|(r3)|(0)}
                [,COND={IO|NO|VTAM|ICCF|OCCFIO|
                       {CANCEL,CODE={name4|(r4)|(1)}}|
                       {POWER,PU={name5|(r5)|(1)}}]
```

The Operands have the following meaning:

COND= Specifies the condition that one or more tasks or even a partition must meet in order to be set "ready-to-run".

LQ (LOG task only)
All tasks waiting for the LOG task are to be set ready.

VCANCEL (ACF/VTAM only)
Indicates that all tasks, which are communicating with ACF/VTAM (at least one ACF/VTAM ACB is open) are to be cancelled with cancel code X'40'.

OCCF (OCCF only)
Indicates that all tasks, which are waiting for OCCF service, are to be set ready.

CRTSAV (OCCF only)
Indicates that all tasks, which are waiting to access the CRT save area (CRTSAV) are to be set ready.

HCFCB (OCCF only)
Indicates that all tasks, which are waiting to access the hard copy file control block, are to be set ready.

ATTINT (OCCF only)
Indicates that an attention interrupt has to be simulated. 'Register 0 must be set by the caller and indicates to process to be performed.

R0 1 Indicates command available
2 Request cancel
R1 Must contain the address of a field containing the length of the command in bytes 0-1 immediately followed by the command itself.

ALLOCR Unlock accessed fields related to the system 'real partition' and posts all tasks waiting for accessing the segment table.

RSGT Unlocks the access to the segment table of the R-space and posts all tasks waiting for accessing the segment table.

PART= Name of a 2-byte field or register containing the identifier (PIK) of the partition to be started or canceled. The PIK is available in field PID of the corresponding COMREG, or in the same field of the BG COMREG, if the partition is active.
This operand is required for COND=START and COND=OC. It is also required for COND=CANCEL in case TASK is omitted.

START (JOB CONTROL and VSE/POWER only)
Valid with PART only.
Indicates that the partition is to be removed from the unbatched or stopped state. This is the default option if PART was specified. The main task of the partition is scheduled for EOJ with cancel code X'10' and the number of active virtual partitions (field IJBAPNO in SYSCOM) is incremented. The user must ensure that the partition area has been allocated.

OC (System internal use only)
Valid with PART only.
Indicates that the operator communication exit for the specified partition has to be activated, if available.

Output:

0 (X'00')	Activation successful
4 (X'04')	OC exit routine already active
8 (X'08')	No OC exit routine available

CANCEL (VSE/POWER, ACF/VTAM, ICCF only)
Valid with PART or TASK only.
Indicates that the maintask of the specified partition or the specified task is to be canceled with the cancel code specified in the CODE operand.

CODE= The CODE operand refers to the name of a 1-byte field or to a register containing the cancel code. Register 0 must not be used.

TASK= Name of a 2-byte field or register containing the TID of the task to be posted or cancelled. The TID is available in the SYSCOM field TID when the task is active.
This operand is required for COND=(NO, IO, VTAM, ICCF, OCCFIO and POWER).
It is also required for COND=CANCEL in case PART is omitted.

IO (Key zero)
Valid with TASK only.
Indicates that the task is to be made ready only if it is waiting as a result of a WAIT or WAITM macro.

COND=IO is the default value if TASK is specified

NO (IPL, LOG task only)
Valid with TASK only.
Indicates that the task is to be made ready unconditionally.

VTAM (ACF/VTAM only)
Valid with TASK only.
Indicates that the task is to be made ready as for COND=IO and, in addition, that the ACF/VTAM AP exit has to be taken the next time the task is dispatched.

ICCF (ICCF only)
Valid with TASK only.
Indicates that the task is to be put into the ready state from the ICCF wait state.

OCCFIO (OCCF only)
Valid with TASK only.
Indicates that a SYSLOG request serviced by OCCF is completed for the specified task, and that the task has to be posted if it is waiting on a WAIT or WAITM macro.

CANCEL (VSE/POWER, ACF/VTAM, ICCF only)
Valid with PART or TASK only.
Indicates that the maintask of the specified partition or the specified task is to be cancelled with the cancel code specified in the CODE operand.

CODE= The CODE operand refers to the name of a 1-byte field or to a register containing the cancel code. Register 0 must not be used.

POWER (VSE/POWER only)
Valid with TASK only.
Indicates that an I/O request spooled by VSE/POWER is completed for the specified task and that the

task has to be posted if it is waiting on a WAIT or WAITM macro. In addition, all tasks waiting for VSE/POWER service within the same partition are posted.

PU= Name of 2-byte field or register containing the physical unit number (PUB-index) of the device for which posting is requested.

Register Usage:

R0	PIK or TID for input.
R1	Cancel code for input, whenever applicable.
R15	Function code for input, return code for output whenever applicable.

TSTOP

Deactivate the current task or partition.

The macro has the following format:

```
 [name]   TSTOP [COND={SYSBND|STOP|UNBATCH}]
                [,RETURN={NO|YES}]
```

The operands have the following meaning:

COND   Specifies the condition into which the issuing task is to be set.

SYSBND   (System only)
This is the default value and indicates that the issuing task is to be set into the "system-bound" condition. It is to be used by non-resident system tasks to deactivate themselves.

STOP   (JOB CONTROL only)
Indicates that processing has to be stopped in the current partition. The status is saved at the invocation point, and processing will resume at the next sequential instruction, as soon as the partition is started again (TREADY COND=START macro, see above). The main task of the partition is made undispatchable and the number of active virtual partitions (field IJBAPNO in SYSCOM) is decremented.

UNBATCH   (JOB CONTROL only)
Indicates that processing has to be stopped in the current partition and, in addition, that the partition has to be invalidated. The status at the invocation point is not saved in this case. The partition has to be reinitialized before it is started again (TREADY COND=START macro, see above). The main task of the partition is made undispatchable and the counter of active virtual partitions (field IJBAPNO in SYSCOM) is decremented.

RETURN   (Valid with COND=SYSBND only)

NO   This is the default option and indicates that the status as present at the time this service was invoked is to be saved and that processing is to be resumed at the next sequential instruction, as soon as this task is activated again.

YES	Control returns immediately to the calling program without status saving.

Register Usage:

R15	Function code for input.

VALID

The VALID macro can be used to check if addresses of user specified storage area is contained within the user's addressing limits. For each page of the area specified by BEGIN and END (see below) it is checked whether the service owner is allowed to access it in the requested way. An appropriate return code is returned in register 15.

The macro has the following format:

```
 [name]   VALID     BEGIN={addr|(1)}
                    ,END={addr|(2)}
                    ,CHECK={READ|UPD}]
```

The operands have the following meaning:

BEGIN   Specifies the begin address of the area to be handled.

END   Specifies the end address of the area to be handled.

CHECK   Specifies the type of check to be done

READ   User wants to check if read access within the specified area is possible. This assumes that,

- None of the pages within the specified area is fetch protected.
- None of the pages within the specified area is flagged invalid and any of the pages has a storage protection key that is valid to be accessed by the issuing task.

UPDate   User wants to check if write access within the specified area is possible. This assumes that,

- None of the pages within the specified area is flagged invalid and any of the pages has a storage protection key that is valid to be accessed by the issuing task.

<u>Output:</u>

Register 15 contains one of the following return codes.

0 (X'00')	Requested access is allowed to the total area
4 (X'04')	Reserved
8 (X'08')	CHECK=READ: storage is fetch protected or key mismatch CHECK=UPD: key mismatch

12 (X'0C')   Addressed area is invalid

VIO

For a description of the VIO macro refer to "SVC 114 (X'72' - VIO)" on page 85.

VIO CLOSE

This macro has the following format:

```
Assembler :

   [name]   VIO      CLOSE
                    [,{VIORB={(r1)|(1)}|
                     SCOPE={STEP|JOB}}]

PLS :

  ?[name:]   VIO      (CLOSE)
                    [{VIORB({(r1)|(1)})|
                     SCOPE({STEP|JOB})}];
```

The operands have the following meaning:

VIORB   Register containing the VIORB pointer (as returned by VIO OPEN) of the work area to be deallocated. If both VIORB and SCOPE are omitted, Reg.1 is assumed to contain the VIORB pointer.

SCOPE   Unconditional deallocation of all VIO work areas belonging to the issuing partition with the specified lifetime (see also VIO OPEN). This operand is reserved for system usage. The specification is passed in Reg.0.

Return Codes in Register 15:

0 (X'00')   Successful deallocation.

VIO EXTND

```
Assembler :

   [name]   VIO      EXTND
                     ,SIZE={nK|(r1)|(0)}
                     [,VIORB={(r2)|(1)}]

PLS :

  ?[name:]   VIO     (EXTND)
                     SIZE{(nK)|((r1))|(0)}
                     [VIORB{((r2))|(1)}];
```

The operands have the following meaning:

SIZE Amount of additional space to be allocated. The unit is K-bytes for absolute notation and bytes for register notation. The specified value is passed in Reg.0. The requested increment is interpreted relative to the actually allocated size available in the VIORB. The additional space is logically contiguous to the existing area.

VIORB Register containing the VIORB pointer (as returned by OPEN) to the work area to be extended. If the parameter is omitted, Reg.1 is assumed to contain the VIORB pointer.

Return Codes in Register 15:

0 (X'00')	Additional space allocated
8 (X'08')	No more space available

VIO MOVE

```
Assembler :

   [name]   VIO       MOVE
                      ,FROM={name1|(r1)|(r11,r12)}
                      ,TO={name2|(r2)|(r21,r22)}
                      ,LEN={n3|(r3)}
                      [,MFG={name4|(r4)|(1)}]

PLS :

  ?[name:]   VIO      (MOVE)
                      FROM{(name1)|((r1))|((r11,r12))}
                      TO{(name2)|((r2))|((r21,r22))}
                      LEN{(n3)|((r3))}
                      [MFG{(name4)|((r4))|(1)}];
```

The operands have the following meaning:

FROM — Address of source data area.
If symbolic or single register notation is used, an address in virtual storage is assumed. Double register notation must be used to specify a VIO address. In this case, r11 is interpreted as the VIORB pointer of a VIO area and r12 as an offset (RBA) within the VIO area.

TO — Address of target area.
The notation convention is the same as for the FROM operand.

LEN — Number of contiguous bytes to be moved.
The maximum specification for absolute notation is 4095. With register notation, any number compatible with the size of program/VIO areas may be specified. Crossing of VIO block boundaries is supported.

MFG — Address of a 20-byte area, in which the parameter list is to be build. If omitted, an in-line parameter list area is generated.

Register Usage:

R0	Not used.
R1	Address of parameter list.
R2-R12	May be used for register notation.
R13	Assumed to contain the address of a 72-byte save area.

R14	Link register.
R15	Address of MOVE routine (from the VIORB) for input. Return code for output.

Return Codes in Register 15:

0 (X'00')	Operation successful
4 (X'04')	End of File reached on VIO file
8 (X'08')	Unrecoverable error
12 (X'0C')	Invalid input

Restrictions:
At least one of the FROM/TO parameters must designate a VIO address.
If both are VIO addresses, they may not refer to the same VIO area
(r11 ¬= r21).
Control returns to the caller only after completion of the MOVE
operation, independently of VIO PROC options.

VIO OPEN

```
 Assembler :

    [name]   VIO       OPEN
                       ,SIZE={nK|(r1)|(0)}
                       [,SCOPE={STEP|JOB}]
                       [,PROC={SYNCH|ASYNCH}]

 PLS:

   ?[name:]   VIO      (OPEN)
                       SIZE{(nK)|((r1))|(0)}
                       [SCOPE{(STEP)|(JOB)}]
                       [PROC={(SYNCH)|(ASYNCH)}];
```

The operands have the following meaning:

SIZE Amount of space to be allocated.
The unit is K-bytes for absolute notation and bytes for register notation. The specified value is passed in Reg.0.

SCOPE Lifetime of the work area.

STEP The work area is automatically deallocated at end of job step (default).

JOB The work area is automatically deallocated at end of job.

PROC Processing mode.

SYNCH After VIO POINT (see below), the issuing task is implicitly set to wait until the block is available (default).

ASYNCH After VIO POINT, control returns immediately to the issuing task and WAIT or WAITM must be issued before accessing the requested block.

Output: An address is returned in reg.1, which points to a system control block (VIORB). The VIORB address uniquely identifies the allocated work area and must be specified for all subsequent requests referring to this area. The VIORB contains the actual size of the allocated area, which can be larger then the requested size, depending on the internal allocation unit. The VIORB also contains the size of a VIO block, which is identical with the page size. The user is recommended to treat the block size as a variable, to be obtained from the VIORB after VIO OPEN. Note that a POINT request is necessary before a block of a VIO area becomes addressable to the program.

Return Codes in Register 15:

0 (X'00')	Successful allocation
8 (X'08')	No more space available

VIO POINT

```
Assembler :

   [name]   VIO       POINT
                     ,RBA={(r1)|(0)}
                     [,{VIORB={(r2)|(1)}

PLS :

  ?[name:]   VIO      (POINT)
                     RBA{((r1))|(0)}
                     [{VIORB{((r2))|(1)};
```

The operands have the following meaning:

RBA Register containing a relative byte address within the VIO area, to which addressability is requested. The specified value is passed in Reg.0.

VIORB Register containing the VIORB pointer (as returned by OPEN). of the work area to be accessed. If the parameter is omitted, Reg.1 is assumed to contain the VIORB pointer.

Output: The results of a VIO POINT are returned to the requestor in the VIORB. The lay-out of the VIORB is described by the MAPVIORB macro, see below.

Bit 0 in byte 2 of the VIORB indicates completion of a POINT request (same as the traffic bit in a CCB). If PROC=SYNCH was specified for OPEN, the request is always complete when the task regains control after a POINT request. If PROC=ASYNCH was specified, the traffic bit must be checked (usually by WAIT or WAITM) before processing based on a previous POINT request can continue. Note that the traffic bit is exclusively maintained by the system. Since the VIORB is allocated in protected storage, the user program can only retrieve information from it.

Error conditions are indicated by a return code in field VIORBRTC (see MAPVIORB).

After a successfully completed POINT request (no error flag on), a VIO block containing the requested RBA is addressable under the virtual address returned in field VIORBPNT. The RBA of the first byte of this block is returned in field VIORBRBA. The block remains addressable to the user program until a POINT request for another RBA is issued or the work area is deallocated by VIO CLOSE see below. Any later attempt to access the block leads to unpredictable results.

When a block is accessed the first time after VIO OPEN or EXTND it is cleared to binary 0's.

Performance Note:
If PROC=ASYNCH was specified for OPEN, POINT is always executed on a fast path without redispatching. For PROC=SYNCH, the fast path is taken whenever the requested block is already available in real storage.

Path length figures are not yet available.

MAPVIORB Macro

MAPVIORB generates a DSECT describing the VIORB.

This macro has the following format:

```
Assembler :

   [name]   MAPVIORB

PLS :

  ?[name:]   MAPVIORB;
```

Bytes Dec	 Hex	Labels	Description
0 - 1	0 - 1		Reserved
2	2	VIORBCM1	Communication byte
		VIORBTRB	X'80' Point request complete
3	3	VIORBRTC	Return code
		VIORBEOF	X'04' Requested block outside area
		VIORBERR	X'08' Unrecoverable error
		VIORBINC	X'0C' Inconsistent state
4 - 7	4 - 7	VIORBASZ	Actual size of area in bytes
8 - 11	8 - B	VIORBBSZ	Size of a block in bytes
12 - 15	C - F	VIORBPNT	Virtual address of current block
16 - 19	10 - 13	VIORBRBA	Relative byte address of current block

Figure 331. DSECT Generated by Macro MAPVIORB

VSIUCVU, VSIUCVPL, VSIUCV

Macros for VM/VCNA (VTAM Communication Network Application) Support

There are control blocks by which an application program passes requests to the subsystem support for VM/VCNA (also referred to as VSE/Advanced Functions IUCV). Macros are provided to build these control blocks and to map them for symbolic reference within a program. The blocks are defined by the macros VSIUCVU and VSIUCVPL.

A single macro, VSIUCV, is used to request subsystem support functions provided by VSE/Advanced Functions.

VSIUCV

The VSIUCV macro is used to request subsystem support functions provided by VSE/Advanced Functions.

The macro has the following format:

```
[name]   VSIUCV CB={addr|(r1)}
               ,OP={OPEN|CLOS|CONN|ACPT|SEVR|SSTE}
              [,VSIUCVU={addr|(r2)}]
              [,{TRGTID=(id,name)|PATH=number|(r3)}]
              [,UDATA={data|(r4)}]
              [,PRTY={YES|NO}]
              [,QUIES={YES|NO}]
              [,MSGLIM={value|(r5)}]
              [,ID=name]
              [,EXIT={addr|(reg)}]
```

The operands have the following meaning.

CB — Is the address of the function related control block. It Is the address of either the VSIUCVU control block, (OP=OPEN, OP=CLOS or OP=SSTE), or it is the address of the VSIUCVPL control block (OP=CONN, OP=ACPT or OP=SEVR).

OP — Specifies the requested service.

OPEN — Identifies a program to VSE/Advanced Functions IUCV and establishes the environment necessary to connect one program to another program.

CLOS — Indicates that an application program wants to drop the connection with the VSE/Advanced Functions IUCV. It immediately severs all paths associated with this user ID.

CONN — Initiates a connection between the issuing user and another user of IUCV.

ACPT — Indicates that a program accepts a connection request initiated by another user.

SEVR — Terminates a previously established connection request or it forces an incoming connection request to be rejected.

SSTE — Issuer requests supervisor state. This enables the requester to issue macros for VM/VCNA (VTAM communication Network Application) support.

VSIUCVU The address of the VSIUCVU block that identifies the user.

trgtid 'id' is the VMID of the virtual machine of the target user.
'name' is the identifier by which the target user is known.
(TRGTID and PATH are mutually exclusive).

PATH Is the path ID or the path number.
It is the path ID passed when the user is notified of the incoming connection request or it is the path ID of the path being terminated or rejected.
(PATH and TRGTID are mutually exclusive).

UDATA Any 4 bytes of user information.

PRTY Specifies whether priority messages will be used.
(Default=NO).

QUIES A connection is to be initiated in quiesced mode.
(Default=NO).

MSGLIM The message limit value.

ID Is a VSE/Advanced Functions provided password.

EXIT The address of the exit routine that is to be given control when an interrupt-related event for this user occurs.

Operands Interrelationship with Keywords

OP	CB	VSIUCVU	TRGTID	PATH	UDATA	PRTY	QUIES	MSGLIM	ID	EXIT
OPEN	R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O	O
CLOS	R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CONN	R	O	O	N/A	O	O	O	O	N/A	N/A
ACPT	R	O	N/A	O	O	N/A	O	N/A	N/A	N/A
SEVR	R	O	N/A	O	N/A	N/A	N/A	N/A	N/A	N/A
SSTE	R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O	N/A

R = Required parameter O = Optional parameter N/A = Not Applicable

Register Usage:

R0 Used by the macro to pass the operation to the SVC.
R1 Used for the address of the appropriate control block.

Output: Register 15 contains one of the following return codes.

0 (X'00')	No error	All
4 (X'04')	Issuer not in SUPVR state	All
8 (X'08')	Unknown operation code	All
12 (X'0C')	User already active	OPEN

16 (X'10')	Insufficient resources	OPEN
20 (X'14')	User inactive	CONN, ACPT, SEVR, CLOS
24 (X'18')	IUCV detected error	CONN, ACPT, SEVR
28 (X'1C')	No connect pending	ACPT
32 (X'20')	Invalid path ID	ACPT, SEVR
36 (X'24')	Issuer is not path owner	ACPT, SEVR
40 (X'28')	Path inactive	SEVR

A requester of the subsystem support for VM/VCNA (SVC 141) will be canceled with 'ILLEGAL SVC' (ERROR 21) under the following conditions:

- The requester is not authorized (not VM/VCNA).
- The requester for supervisor state is not a main task.
- IUCV is not present in the VM/System Product.
- IUCV is present in the VM/System Product, but IUCV 'QUERY' failed during IPL of VSE/Advanced Functions.

A requester of the subsystem support for VM/VCNA (SVC 141) will be canceled with 'Invalid Address' (ERROR 25) when either the VSIUCVU or VSIUCVPL are not in the corresponding partition. The following examples show how the VSIUCV macro should be used to perform the various operations.

(1) VSIUCV OPEN

OPEN is used to identify a program to VSE/Advanced Functions IUCV support and to establish an environment by which a program can be connected to another program.

```
[name]  VSIUCV CB=addr|(reg),OP=OPEN
               [,ID=name]
               [,EXIT=addr|(reg)]
```

CB must point to a VSIUCVU block.

(2) VSIUCV CLOS

CLOS is used to stop usage of VSE/Advanced Functions IUCV by an application. It immediately severs all paths associated with this user ID.

```
[name]  VSIUCV CB=addr|(reg),OP=CLOS
```

CB must point to a VSIUCVU block.

(3) VSIUCV CONN

CONN is used to initiate a connection between the issuing user and another user of IUVC.

```
   [name]  VSIUCV CB=addr|(reg),OP=CONN
                  [,VSIUCVU=addr|(reg)]
                  [,TRGTID=(id,name)]
                  [,UDATA=data|(reg)]
                  [,PRTY=YES|NO]
                  [,QUIES=YES|NO]
                  [,MSGLIM=value|(reg)]
```

CB points to a VSIUCVPL block.

(4) VSIUCV ACPT

ACPT is used to accept a connection request initiated by another user.

```
   [name]  VSIUCV CB=addr|(reg),OP=ACPT
                  [,VSIUCVU=addr|(reg)]
                  [,PATH=number|(reg)]
                  [,UDATA=data|(reg)]
                  [,QUIES=YES|NO]
```

CB points to an VSIUCVPL block.

PATH is the path ID passed when the user is notified of the incoming connection request.

(5) VSIUCV SEVR

SEVR is used to terminate a previously established connection or to reject an incoming connection request.

```
   [name]  VSIUCV CB=addr|(reg),OP=SEVR
                  [,VSIUCVU=addr|(reg)]
                  [,PATH=number|(reg)]
```

CB points to a VSIUCVPL block.

PATH is the path ID of the path being terminated or rejected.

(6) VSIUCV SSTE

SSTE is used to give the requester supervisor state. This enables the requester to issue macros for VM/VCNA (VTAM communication Network Application) support.

```
   [name]  VSIUCV CB=add|(reg),OP=SSTE
                  [,ID=name]
```

CB must point to a VSIUCVU block.

ID must be the VSE/Advanced Functions provided password.

VSIUCVPL

This macro is used to create or map the control block used to request the subsystem support for VM/VCNA to perform connection-related services.

The macro has the following format:

```
[name]   VSIUCVPL  [DSECT={YES|NO}]
                   [,VSIUCVU=addr]
                   [,TRGTID={(id,name)|PATH=number}]
                   [,UDATA=data]
                   [,PRTY={YES|NO}]
                   [,QUIES={YES|NO}]
                   [,MSGLIM=value]
```

The operands have the following meaning:

DSECT If DSECT =NO is coded or defaulted to, the macro produces a 48-byte area with a label as specified. If DSECT=YES is coded, the macro produces a DSECT of the area with a label as specified.

VSIUCVU =addr
The address of the VSIUCVU block that identifies the user.

TRGTID 'id' is the VMID of the virtual machine of the target user. 'name' is the identifier by which the target user is known.

PATH number
The path ID or number. PATH and TRGTID are mutually exclusive.

UDATA data
Any 4 bytes of information.

PRTY Specifies whether priority messages will be used.
Default=NO.

QUIES Specifies whether a connection is to be initiated in quiesced mode.
Default=NO.

MSGLIM value
The message limit value.

VSIUCVU

This macro is used to create or map the control block used to identify the using program to VSE/Advanced Functions IUCV. It is pointed to when the program executes VSIUCV with OP=OPEN, OP=CLOS or OP=SSTE.

The macro has the following format:

```
[name]  VSIUCVU  [ID=name]
                 [,EXIT=addr]
                 [,DSECT={YES|NO}]
```

The operands have the following meaning:

ID An 8-character name that is the identifier by which this user is to be known to VSE/Advanced Functions and to other IUCV users to which this program will be connected. The default is all blanks.

EXIT The address of the exit routine that is to be given control when an interrupt-related event for this user occurs. The default is zero.

DSECT YES: Forces a mapping of the control block to be generated.

NO: Forces a CSECT of the control block to be generated.

WRITEHCF

The macro WRITEHCF ensures that the record specified by the first operand will be written onto the HCF. The record address has initially been provided by the POINTHCF macro and is automatically updated by any subsequent WRITEHCF request (except FORCE=YES). The HCF is written in wrap around mode.

The macro has the following format:

```
[name]   WRITEHCF {ioarea|(0)}
                  [,FORCE={NO|YES}]
                  [,DUMP=YES]
```

ioarea Symbolic name of the I/O area which contains the record to be written on the HCF.

FORCE YES: Forces the I/O buffer to be immediately written onto the HCF without the necessity that the I/O buffer is full. The first operand will be ignored, that is, no record is inserted into the I/O buffer.

NO: The I/O buffer will not be written onto the HCF before the I/O buffer is full.

DUMP YES: Indicates that the DUMP program is the issuer of the WRITEHCF macro.

Output: Register 15 contains one of the following return codes.

0 (X'00')	Normal processing successfully completed.
4 (X'04')	Inconsistent input, no WRITE authority
8 (X'08')	No record found, incorrect length.
12 (X'0C')	Unrecoverable I/O error.
16 (X'10')	HCF device is not ready.
20 (X'14')	HCF has just entered the overlay mode.
24 (X'18')	Warning message (HCF is close to overlay mode) must be issued.

Register Usage: The contents of general register 14 through 2 are destroyed by this macro.

XPCC, XPCCB, MAPXPCCB

Cross Partition Communication Macros

The cross-partition communication service (XPCC) allows cross-partition communication between VSE subsystems and also between VSE subsystems and application programs.

The XPCC is IPLed as part of the supervisor. No SYSGEN option is required. Its main functions are:

- An identify function (IDENT)
  This function allows application programs and subsystems to 'log-on' to the XPCC. The XPCC recognizes the names of the applications and uses those names later to set-up the corresponding communication link.

- Set-up of a communication link
  Before data can actually be transmitted between two XPCC users, a communication link has to be established. The applications have to build this link via the CONNECT function. In order to have a complete link, both applications have to request the communication link set-up. Only then can they start exchanging data via this link. This link always is a two-way-only communication path, that is, a data transmission request is always directed to only one other application.
  For synchronization purposes a WAIT capability is provided.

- Once the data transmission link is completed, the two applications can start exchanging data. XPCC will make sure, that the sender of data does not overwrite any data in case where data sending is done faster than the data receiver is processing the data. Whenever a request is issued, return information about the other side of the communication link is provided.

  Sending and receiving data is done asynchronously with a WAIT and POST capability.

- Special commands are provided, in order to clear a connection from a data transmission request.

- Applications may disconnect a communication link and terminate their communication (DISCONN, DISCPRG, TERMINATE). They are not known to XPCC anymore.

- In case a partition ABENDs (or is canceled), the XPCC will disconnect any outstanding communication links. In addition, the XPCC does a 'log-off' (TERMINATE) for the corresponding application.

The cross-partition communication service is invoked via the XPCC macro.

All XPCC requests are associated with a program defined control block (XPCCB - cross-partition communication control block)

The XPCCB is used

- To define request options
- To set up pointers to buffer areas and ECBs
- To receive system return information.

The XPCCB may be generated statically at assembly time or it may be set up and/or modified at execution time. Referencing the fields is done via the mapping macro MAPXPCCB.

Each XPCC request expands into an SVC 113 (X'71'), which performs the required function. Control is returned to the requester via the dispatcher.

Register 15 indicates whether the request was successful.

0 (X'00') Request was started successfully

4 (X'04') Same as X'00', but additional return information stored in IJBXRETC.

8 (X'08') Request rejected. IJBXRETC in the XPCCB contains a code which defines the reason. (for detailed description see macro MAPXPCCB)

XPCC

Invokes the cross-partition communication service.

The macro has the following format:

```
[name]  XPCC    XPCCB={addr|(1)|(S,addr)},
                FUNC={keyword|(reg)}
                [,BUFFER={addr|(reg)|(S,addr)}]
```

The operands have the following meaning:

XPCCB Defines the address of the XPCCB control block containing all request-related information. Depending on the request, only certain fields are used (for details, refer to the description of the various functions below).

FUNC Defines the specific function to be requested from the XPCC service. Depending on the type of request, the following keywords may be used:

1. Initializing requests

   IDENT
   Does a 'log-on' of an application to the XPCC service.

   TERMIN
   An application terminates its XPCC usage (if all links are already disconnected).

   TERMPRG
   Unconditional termination, data transmission may get interrupted.

   TERMQSCE
   No termination yet, but new connections to this application are not granted anymore.

2. Connection-related requests

   CONNECT
   Connect an application to another application

   DISCONN
   Terminate a connected link to another application (if no data transmission is going on at the moment).

DISCPRG
Terminate connection unconditionally. A data transmission may get interrupted.

DISCALL
The system disconnects unconditionally all connections for a certain application.

3. Data transmission related requests

SEND
Send data to another application

SENDR
Send data and request a reply back from the receiver.

RECEIVE
Receive data

REPLY
Send a reply back to the sender

CLEAR
Purge a previously initiated SEND request from the connection (used by the sender)

PURGE
The receiver purges the data, because he is not able to receive it.

If the format **FUNC=(reg)** is used, the register specified must have been loaded with the corresponding function byte value. These values can be found in the program listing under label IJBXFCT in the mapping macro MAPXPCCB.

BUFFER This parameter may optionally be used in connection with SEND, SENDR, RECEIVE, and REPLY requests to dynamically provide a data area, from where (SEND, SENDR, REPLY) or to which (RECEIVE) data are moved. It overwrites the corresponding entry in the XPCCB control block (see XPCCB macro below).

For a RECEIVE and REPLY request the BUFFER parameter address must point to an 8-byte area with the following format:

Bytes	Description
0	X'80'
1 – 3	data area address
4 – 7	length of data area

With SEND or SENDR the BUFFER parameter address must point to an address list as shown below:

Bytes	Description
0	indicator byte X'00' : not last entry in list X'80' : last entry in list
1 - 3	data area address
4 - 7	length of data area

Up to 7 entries of the format described above may be specified in an address list. If for a SEND(R) request the indicator byte is other than X'00' or X'80', the request is rejected with 'Format Error'.

XPCCB

This macro is used to set up the cross-partition communication control block. The corresponding DSECT is generated by means of the MAPXPCCB macro instruction. Each XPCCB represents one connection, its address is saved by XPCC. The mapping DSECT (MAPXPCCB) may be used to reference or modify the control block fields at execution time when setting up a VSE-XPCC request or when checking the return code information.

The macro has the following format:

```
[name] XPCCB    APPL=name
                TOAPPL={name|any}
                [BUFFER={addr|(addr,length)}]
                [REPAREA={(addr,length)}]
```

The operands have the following meaning:

APPL   The name of the application requesting XPCC service.

TOAPPL   'name' is the name of the application, to which communication should be established. 'ANY' means that an open communication link is set up. It cannot be the name of an application and can only be used by an IBM subsystem.

BUFFER   Defines buffer area(s) for the data transmission request, where **addr** is a pointer to a list of 8-byte fields, described under 'BUFFER parameter' in the XPCC macro. If the list contains only one entry, it can be specified directly by **(addr,length)** instead of **addr**.

REPAREA   Defines for the SENDR and for the REPLY request the data area for the reply.

Return Codes and Reason Codes:
Value and meaning of the return codes (returned in field IJBXRETC) and of the reason codes (returned in field IJBXREAS) are defined in the MAPXPCCB macro. They are also briefly described under the various XPCC function examples, which follow.

Identification of Communication User

In order to set up a connection between two applications, the applications should be known by the system.

Therefore, before requesting any XPCC services, the application has to identify itself to the XPCC. This identification process will be done via the IDENTIFY function.

The macro has the following format:

```
[label]  XPCC   XPCCB={addr|(1)|(S,addr)}
                FUNC={IDENT|(reg)}
```

The request uses the following fields in the XPCCB control block:

Input

Name	Description
IJBXFCT	Function byte
IJBXAPPL	Application name of the requesting application.

Output

Name	Description
IJBXRETC	Return codes
IJBXREAS	Reason code
IJBXITID	TID of identify requestor It may be up to 8 bytes long.
IJBXIDK	ID-token returned back by the system after the IDENTIFY is completed.

Special naming conventions have to be observed by IBM subsystems in order to permit usage of unlimited number of connections.

Subsystem names must all start with a 'SYS' prefix. The next 3 or 4 characters must be identical to those submitted with the SUBSID NOTIFY function. The remaining 1 (or 2) characters are optional.

Product	ICCF	SQL/DS	VSE/POWER	CICS	SSX
Name provided via SUBSID NOTIFY	ICCF	ARI	PWR	CICS	SSX
Name provided via IDENTIFY	SYSICCFx	SYSARIxx	SYSPWRxx	SYSCICSx	SYSSSXxx

User application names must not start with a 'SYS' , else the IDENTIFY request will result in a program cancel condition ('ILLEGAL SVC').

Function: The XPCC will associate the name with a unique IDENTIFY token, which is returned back to the user into the IJBXIDK field.

All succeeding CONNECT requests, issued by the application, must use this IDENTIFY token (thereby indicating the requestor of the connection).

One program may issue several IDENTIFYs with different names. This means, that a program is known under two (or more) different names, each representing one application.

Ownership: If the IDENTIFY request is issued by the maintask, it is regarded as being owned by the corresponding VSE partition.

If the partition ABENDs, all IDENTIFYs issued in this partition and not yet terminated, are terminated by the system.

If the IDENTIFY request is issued under control of a VSE subtask, the IDENTIFY is regarded as being owned by the subtask. If the subtask terminates, the system will terminate all IDENTIFYs belonging to this subtask. Furthermore all connections of this subtask issued under control of maintasks IDENTIFYs are terminated.

Authorization: Before an IBM subsystem does an IDENTIFY, it has to establish its identity via the SUBSID supervisor service.

IDENTIFY Names Starting with 'SYS': The XPCC compares byte 4 to 6, or byte 4 to 7 of the IDENTIFY name with the names stored in the SUBSID table. If a match is found, it checks whether the IDENTIFY was issued in the same VSE partition as the SUBSID NOTIFY request for this name. If yes, the XPCC will treat the application as an authorized subsystem.

If the IDENTIFY name does not match an IBM subsystem name in the SUBSID table, or if the partition-IDs of IDENTIFY and SUBSID NOTIFY do not match, the requesting program will be canceled (illegal SVC).

IDENTIFY Names not Starting with 'SYS': Such names are treated as 'normal' user applications

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'ILLEGAL SVC' - an application name starting with 'SYS' was submitted but no equivalent name for this partition is stored in the SUBSID system table.
- Cancel due to 'invalid address' address validation error.

Defining a Communication Path

Before starting with the actual data transmission, a unique communication path has to be established via the CONNECT request.

A communication path is always built between two applications. In order to get completed, both applications have to request the communication path link-up via the CONNECT request.

Two formats are possible:

- Format 1 defines a connection directed to a specific application.
- Format 2 defines an 'open-ended' connection, to which any other subsystem or user can link up.

To prevent a user-application from monopolizing the XPCC services, the number of connections, which can be set up for one IDENTIFY will be restricted to 512. Subsystems have no restriction.

Defining a Specific Connection

Format:

```
[label] XPCC    XPCCB={addr|(1)|(S,addr)}
                FUNC={CONNECT|(reg)}
```

The CONNECT request uses the following fields in the XPCCB.

Input:

Name	Description
IJBXFCT	Function byte
IJBXTOAP	Up to 8-Byte long application name, to which connection is to be established. It is the name used by the other application at IDENTIFY time. If an 'open' ended connection is to be set up, the user has to initialize this field to 0.
IJBXIDK	Provided by the system at IDENTIFY time of requesting application.
IJBXSUSR	Moved to JBXSUSR of the other side at connection completion time.

Output:

Name	Description
IJBXRETC	Return codes
IJBXREAS	Reason code
IJBXCTID	TID of CONNECT requestor
IJBXPID	Returned when connection is completed. Identifies the connection and has to be used on all succeeding data transmission requests.
IJBXCECB and IJBXSECB	Set to F'00'. They are posted, whenever the connection is complete (data transmission may start).
IJBXRECB	Set to F'00' .
JBXCNTL	Set to 4F'00'.
IJBXRUSR	User data from IJBXSUSR of the other side is filled in at connection completion time.

Additional comments:

IJBXCECB: The 'CONNECT' ECB (IJBXCECB)

The link connection can be set up right away, if the application at the other end already issued a CONNECT, waiting for the link to be completed.
The connection may not yet be complete, because the other side did not yet issue a CONNECT request or did not yet identify itself to the XPCC. In such a case, the CONNECT requestor may wait for the connection to be completed via the specified IJBXCECB.

The IJBXCECB will be posted by the XPCC as soon as the other side completes the connection and IJBXREAS will be set with the appropriate reason code (see macro MAPXPCCB).

Functional Characteristics

CONNECT will try to establish a link to the application requested in IJBXTOAP.

If the other side already issued the corresponding CONNECT, the connection is established right away and IJBXCECB and IJBXSECB are posted both.

If the other application did not yet issue the CONNECT or is even not yet active, the request will return a return code into field IJBXRETC, but the connection is granted. IJBXCECB and IJBXSECB will be posted and the task will be taken out of wait state, if the other side issues the corresponding connect, which completes the connection.

When the other side DISCONNECTs from this link, other applications cannot CONNECT to this link.

Link Ownership and VSE Tasks

If an IDENTIFY was issued under control of the maintask, any subtask may set up a connection for this IDENTIFY.

If an IDENTIFY was issued under control of a VSE subtask, the CONNECT requests for this IDENTIFY can only be issued under control of the same subtask.

Connections are owned by the task, which issued the CONNECT request. All data transmission requests and DISCONNECT must be issued by the same task, which did the CONNECT.

The system cleans up the connection at task termination time

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

Defining an Open-Ended Connection

In many cases, the subsystem trying to establish a connection, does not know the application, which wants to set up a communication protocol. In such a case, the subsystem would provide a connection, to which any application could connect.
If a subsystem establishes both types of connections (specific and open-ended) and if it wants that the specific connections should be used first, then it must issue the connect-specific prior to the CONNECT-any.

Format:

```
[label]    XPCC      XPCCB={addr|(1)|(S,addr)}
                     FUNC={CONNECT|(reg)}
```

The usage of the different XPCCB fields is the same as in CONNECT-specific.

Function:
The XPCC connects this link to the first application, which requests a connection with this subsystem. At that time it will post the IJBXCECB and IJBXSECB, take the task out of wait state and the connection is completed.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address': Address validation error.

Data Transmission

Once a connection is established, the two applications can start exchanging data via this link.

Principle of Data Transfer

The sender of data builds up a list of data areas (and their length) and - via the SEND request - passes them to the XPCC.
At the other end of the connection, the XPCC moves the corresponding control information (message length) into the IJBXCNTL field and posts the 'Receive' ECB (IJBXRECB). The other application would then realize, that there is a pending SEND request. It would first inspect the control buffer for the length of the message (obtain optionally buffer space), and would then ask for the data transfer via the RECEIVE request indicating, where the data are to be stored.

The system transfers the data into the corresponding buffer space. At the same time it posts the 'SEND' ECB associated with the SEND request at the sender's side (IJBXSECB), to indicate, that the data SEND request is successfully completed. At the receiver's side, the IJBXRECB is reset, in order to be ready for the next SEND request.

Connection 'BUSY' Status

As soon as a SEND or SENDR request is started on an available link, the connection is considered to be 'busy'. In case of a SEND, the connection is busy until the receiver issues the RECEIVE (this is also valid for a SEND with zero data length). In case of a SENDR, the connection is busy until the receiver accepts the data via RECEIVE and sends a reply back via REPLY. The receiver may also 'free' the connection by purging the Connection via the PURGE command.

As long as a connection is in such a 'busy' status, no new data transmission request can be started (will be rejected by a return code).

As soon as a connection is 'free' again, the sender's IJBXSECB is posted (either by the RECEIVE or by the REPLY or by the PURGE).

The sender of data may clear a connection after issuing a SEND command, however, the connection is still regarded as 'busy' until the receiver acknowledges the CLEAR command by a RECEIVE, REPLY or PURGE.

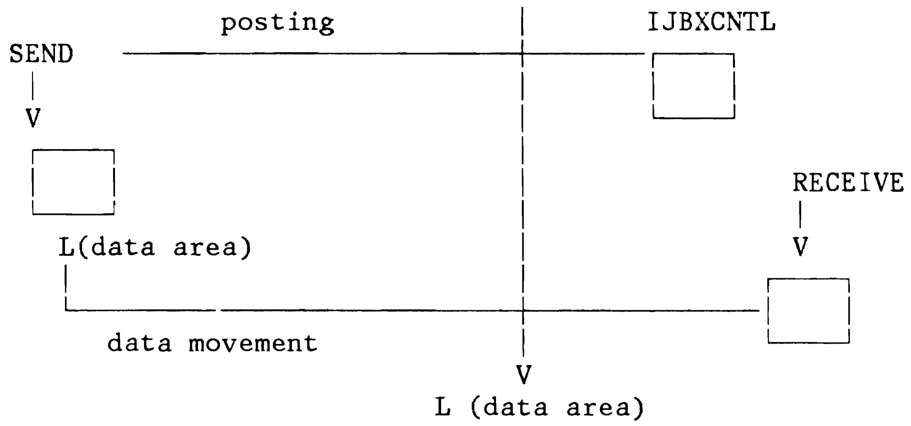
Link Ownership and Data Transmission

All data transmission requests and the DISCONNECT must be done by the task, which did the CONNECT.

Overview on Data Exchange Functions

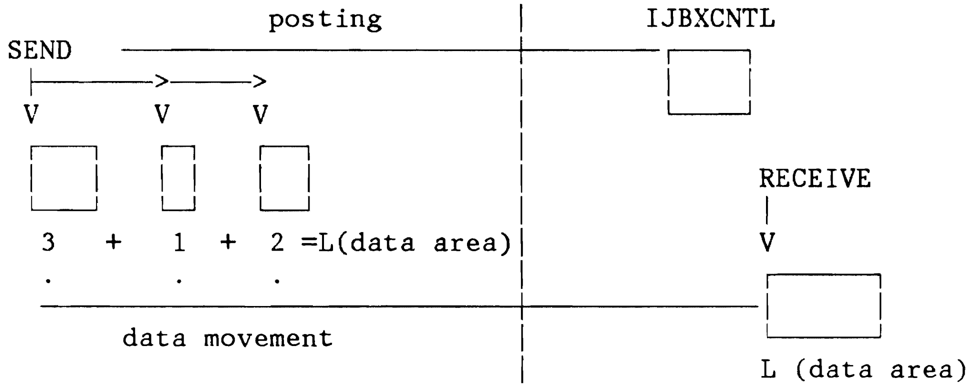
Two data transmission methods are possible:

1. Transmission of data such, that one data record on the sender's side is received as the same data record on the receiver's side.



The SEND request will post on the receiver's side the length of the data area to be sent (in the IJBXCNTL field).
The receiver obtains the needed storage and issues RECEIVE.

2. Transmission of data such, that the sender provides a list of data areas to be sent, and the receiver collects the concatenated data into one data area (the length being the sum of the sender's data area lengths).



The concatenated data length is posted at SEND time on the receiver's side. The receiver obtains dynamically the needed storage and issues RECEIVE.

The programs may choose at SEND/RECEIVE time, whether to use method 1 or method 2 .

Two protocols are available for data exchange:

a. A SEND - RECEIVE protocol:
If this protocol is used, the sender is posted when the receiver accepts the sent data via the Receive request. At this time, the connection is available for the next data transmission request.

b. A SENDR - RECEIVE - REPLY protocol.
When using this protocol, the sender requests a reply from the receiver. The sender is posted when the receiver sends a reply back upon receiving the data. The XPCC transfers the reply data into a reply area, which has to be provided by the sender at SEND time.

Transmitting Data Without Reply Request

Format:

```
[label]    XPCC   XPCCB={addr|(1)|(S,addr)}
                  FUNC={SEND|(reg)}
                  [BUFFER={addr|(reg)|(S,addr)}]
```

Fields in the XPCCB used by SEND on the sender's side.

Input:

Name	Description
IJBXFCT	Function byte
IJBXPID	Path-ID as provided by CONNECT request
IJBXBUF	Consists of IJBXIWP, IJBXADR, and IJBXBLN.
IJBXIND	X'80' If only one data area is to be transmitted X'00' If a list of data area addresses is provided.
IJBXADR	Address of data area to be transmitted or address of list of data area addresses.
IJBXBLN	Length of data area to be transmitted not used in case of data area list Note: If BUFFER is used in the XPCC macro the addr must point to a list of max. 7 entries each entry consisting of (IJBXIND,IJBXADR,IJBXBLN).
IJBXSUSR	8 bytes of user data. This data is moved at the receiver's side into field IJBXRUSR.

Output:

Name	Description
IJBXRETC	Return codes
IJBXREAS	Reason codes
IJBXSECB	Reset to F'00'. Posted, if other side issues RECEIVE or PURGE

Fields in the XPCCB used on the receiver's side.

Output:

Name	Description
IJBXRECB	Posted by system
IJBXRUSR	8 bytes of data from the sender's IJBXSUSR field
IJBXSLN	Length of message to arrive or – in case of an address list – the sum of the length of the sender's data areas
IJBXFLG	X'01' For a 'normal' SEND request

Additional Information:

If a list of data areas is provided, the IJBXADR field points to an array of 8 byte fields where each entry has the following format

Bytes	Description
0	X'00' If this is not the last entry X'80' If this is the last entry in the array
1 – 3	Address of data area to be sent
4 – 7	Length of data area to be sent

The list may have up to 7 entries.

Functional Characteristics

The addresses of the data areas to be transmitted and their length are passed to the XPCC, which will do an address validation. If the BUFFER parameter is not used, the XPCCB information stored in the IJBXBUF field is used for data transmission. If the buffer parameter is specified, it overwrites the information stored in the IJBXBUF field.

The XPCC will calculate the length of the data to be transmitted and moves this information into field IJBXSLN at the receiver's side. The information provided in the field IJBXSUSR of the sender will be moved into field IJBXRUSR at the receiver's side (send user data).

If the connection is still 'busy', the SEND request is rejected and has to be retried. Such a case might occur, if the other side did not yet issue a RECEIVE for a previous SEND request or the other side has issued SEND. In the first case the connection would be free again, when the IJBXSECB associated with the previous SEND request is posted in the second case first a RECEIVE or PURGE must be issued.

The SEND function might be issued with the IJBXBLN field (length of data) initialized to zero and IJBXIND initialized to X'80'. In such a case, only user data are transmitted from IJBXSURS to IJBXRUSR The receiver can recognize such a condition by getting posted (IJBXRECB) and finding a zero data length value in the IJBXSLN field.

Note, however, that the connection would still remain busy until the other side acknowledges the posting via a RECEIVE, or PURGE.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.
- Reason codes posted back together with IJBXSECB by system (see macro MAPXPCCB).

Data Transmission with Reply Request

Via the SENDR the sender posts in the usual way a data transmission request to the receiver. The receiver accepts the data via RECEIVE, processes them, and sends a reply back to the sender via the REPLY request. The connection is now free to handle the next data transmission request.

Format:

```
[label]   XPCC   XPCCB={addr|(1)|(S,addr)}
                 FUNC={SENDR|(reg)}
                 [BUFFER={addr|(reg)|(S,addr)}]
```

Fields in the XPCCB used on the sender's side.

Input:

Name	Description
IJBXFCT	Function byte
IJBXFDSC	If IJBXPOST is on, IJBXCECB will be posted. If the other side receives the data, buffers are free for usage from now.
IJBXPID	Path-ID of connection
IJBXBUF	Buffer area for data transmission. The fields are used in the same way as described under SEND.
IJBXSUSR	8 bytes of user data to be posted into IJBXRUSR field at other end of connection
IJBXRADR	Address of area, into which system transfers the reply data from the other side
IJBXRLNG	Length of reply area

Output:

Name	Description
IJBXRETC	Return codes returned by system
IJBXREAS	Reason codes
IJBXSECB	Reset to F'00'. Posted if other side issues REPLY, PURGE.
IJBXCECB	Reset to F'00'. Posted when the other side issued RECEIVE (if IJBXPOST was set at SENDR time).

Fields in the XPCCB used on the receiver's side.

Output:

Name	Description
IJBXRECB	The IJBXRECB is posted.
IJBXRUSR	8 bytes of data from the sender's IJBXSUSR field
IJBXSLN	Length of data, being sent. It is the sum of the length of all data areas.
IJBXSLNR	Length of data which is expected to be replied
IJBXFLG	Flag bytes X'02' For a SENDR request. This means, that a REPLY is requested. Note, that programs should test only the corresponding bit.

Function:
The SENDR function is equivalent to the SEND function. IJBXCECB at the sender's side will be posted and the task will be taken out of wait state, when the receiver issues the RECEIVE (together with IJBXRECX in IJBXREAS), if IJBXPOST is on at SENDR.

IJBXSECB will be posted and the task will be taken out of wait state when the receiver issued REPLY. At SENDR time, the XPCC will validate the area, which will receive the reply (IJBXREPA) and the length of the REPLY-area is moved to IJBXSLNR at receiver's side.

The SENDR function might be issued with the IJBXBLN field (length of data) initialized to zero and IJBXIND initialized to X'80'. In such a case, only validation of the reply-area is done and the user data is transmitted from IJBXSURS to IJBXRUSR. The receiver can recognize such a condition by getting posted (IJBXRECB) and finding a zero data length value in the IJBXSLN field. He can execute immediately the REPLY (he is not forced to execute a RECEIVE before REPLY)

Return Codes and Reason Codes Provided:

Same as for the SEND request. In addition the reason code IJBXRECX may be posted back together with IJBXCECB by the system when the RECEIVE is executed. Note, that an address validation error may be caused here due to the IJBXREPA address.

Receiving Data

The target application will consider a SEND request at the other side in the following way:

- IJBXRECB of the connection will be posted.
- The IJBXCNTL field will contain control information defining the SEND request.
- The IJBXFLG flag area contains a flag indicating, whether this is a SEND or a SENDR request.

With the RECEIVE request the application prompts the system for the actual data transfer.

Format:

```
[label]   XPCC     XPCCB={addr|(1)|(S,addr)}
                   FUNC={RECEIVE|(reg)}
                   [BUFFER={addr|(Reg)|(S,addr)}]
```

Fields in the XPCCB used by RECEIVE on the receiver's side:

Input:

Name	Description
IJBXFCT	Function Byte
IJBXPID	Path-ID as returned by the CONNECT
IJBXADR	Address of area, to where data are to be moved
IJBXBLN	Length of RECEIVE data area (IJBXIND eq X'80')
IJBXSUSR	8 bytes of user data to be posted into IJBXRUSR field at sender's side

Output:

Name	Description
IJBXRETC	Return codes
IJBXRECB	System resets the IJBXRECB in case of a SEND. In case of a SENDR, the IJBXRECB is reset at REPLY time. The IJBXRECB was posted at SEND (SENDR) time

Fields in the XPCCB used on the sender's side:

Output:

Name	Description
IJBXRUSR	8 bytes of data from receiver's IJBXSUSR field
IJBXSECB	Posted, if this is SEND/RECEIVE protocol
IJBXCECB	Posted, if this is SEND/RECEIVE protocol and posting was required at SENDR time
IJBXREAS	Set, if posting was required at SENDR time
IJBXFLG	X'08' indicates RECEIVE was last XPCC function executed by other side.

If the XPCC RECEIVE macro is used with the BUFFER parameter, it will overwrite the IJBXBUF field information stored in the XPCCB.

Function:
A RECEIVE is requested on a connection upon being posted by a SEND (SENDR).

The program will request the data transfer via the RECEIVE request and the XPCC will move the data into the input area.

If the RECEIVE is executed after a SEND request IJBXRECB at the receiver's side will be reset, and the IJBXSECB at the sender's side is posted. The sender is also taken out of wait state.
The connection is now ready to handle the next SEND (or SENDR) request.
Note, that such a RECEIVE (or PURGE) is also needed in case of SEND with zero data in order to free the connection.

If the RECEIVE is executed after a SENDR request the connection remains 'busy', until the receiver responds back to the sender via the REPLY request.

The RECEIVE request will perform an address validation of the receiver's data area. No padding is performed if the data to be moved are shorter than the input area.

If the data to be moved are longer than the input area, the RECEIVE request is rejected with a return code.
The program can then decide, whether to obtain a longer input area and retry the RECEIVE request, or it may decide, that the incoming data block is too long to be handled and purges the connection from the sent data (refer to PURGE). This PURGE command will post the IJBXSECB at the sender's side with a return code indicating that the receiver issued a PURGE.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

The REPLY Function

The REPLY function is used by the receiver, when receiving data which was sent via a SENDR request.
It allows to send response data back to the sender without going through the normal SEND/RECEIVE function protocols.

Format:

```
[label]   XPCC   XPCCB={addr|(1)|(S,addr)}
                 FUNC={REPLY|(reg)}
                 [BUFFER={addr|(reg)|(S,addr)]
```

Fields in the XPCCB used on the replier's side:

Input:

Name	Description
IJBXFCT	Function byte
IJBXPID	Path-ID as returned from CONNECT
IJBXADR	Address of reply message area (IJBXIND=X'80')
IJBXBLN	Length of reply message
IJBXSUSR	8 bytes of user data to be posted into IJBXRUSR field at sender's side (useful for reply with length 0)

Output:

Name	Description
IJBXRETC	Return codes
IJBXRECB	Reset to F'00' (posted at SENDR time).

Fields in the XPCCB used on the sender's side:

Output:

Name	Description
IJBXRUSR	8 bytes of data from the replier's IJBXSUSR field
IJBXFLG	X'20' indicates last XPCC function executed by other side was REPLY
IJBXSECB	Posted

Function:
The XPCC will get the reply data from the IJBXADR address with the specified length and moves them into the AREA at the sender's side defined at SENDR time via the IJBXREPA field.
If reply area length is 0 only the sender's IJBXSECB is posted and user data is transmitted from IJBXSUSR at replier's side to IJBXRUSR at sender's side. If the sender's area is too long, the remaining bytes will not be padded. If it is too short, the REPLY request will be rejected. The actual data length is moved into IJBXSLN on sender's side.

IJBRECB at the receiver's side will be reset, and IJBXSECB at the sender's side will be posted. The sender will also be taken out of wait state.

Note, that after a SENDR request the connection is 'busy' until the receiver answers with the REPLY request.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

Clearing a Pending SEND Request on the Sender's Side

The sender of data may decide, that he wants to cancel an outstanding SEND/SENDR request. For this purpose he can use the CLEAR function.

Format:

```
[label]  XPCC    XPCCB={addr|(1)|(S,addr)}
                 FUNC={CLEAR|(reg)}
```

XPCCB fields used at the sender's side:

Input:

Name	Description
IJBXFCT	Function byte
IJBXPID	Path-ID as returned by CONNECT
IJBXSUSR	8 bytes of user data to be posted into IJBXRUSR field at receiver's side

Output:

Name	Description
IJBXRETC	Return codes

XPCCB fields used on the receiver's side:

Output:

Name	Description
IJBXRUSR	8 bytes of data from the sender's IJBXSUSR field
IJBXRECB	The 'RECEIVE' ECB is posted
IJBXREAS	IJBXCLEA is posted to the reason code field
IJBXFLG	X'04' indicates last XPCC function executed by the other side was CLEAR

Function:
The XPCC will set a 'SEND cleared' flag for the connection in the XPCCB, post IJBXRECB and store reason code IJBXCLEA into IJBXREAS on the receiver side in the following cases:

- If there is a SEND request pending for the connection, for which the other side did not yet issue a RECEIVE.

- If there is a SENDR request pending for this connection, for which the other side did not yet issue a RECEIVE, or the requested REPLY.

In order to free the connection for the next SEND request, the receiver has to issue a RECEIVE or PURGE. The RECEIVE will return a return code indicating a cleared connection and it will post the associated IJBXSECB at the sender's side. The connection is then ready for the next SEND (SENDR) request.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

Clearing a Pending SEND Request on the Receiver's Side

The receiver might receive messages, which he is unable to handle (for example, because the message length exceeds the available buffer storage). He can reject those messages via the PURGE request.

Format:

```
[label]  XPCC  XPCCB={addr|(1)|(S,addr)}
               FUNC={PURGE|(reg)}
```

XPCCB fields used at the receiver's side:

Input:

Name	Description
IJBXFCT	Function byte
IJBXPID	Path-ID as returned by CONNECT
IJBXSUSR	8 bytes of user data to be posted into IJBXRUSR field at sender's side

Output:

Name	Description
IJBXRECB	Reset
IJBXRETC	Return codes

XPCCB fields used at the sender's side:

Output:

Name	Description
IJBXRUSR	8 bytes of data from the receiver's IJBXSUSR field
IJBXSECB	Posted
IJBXREAS	Appropriate reason code is set
IJBXFLG	X'10' Indicates that last XPCC function executed by the other side was PURGE

Function:
The XPCC clears the connection from the pending SEND request. The IJBXRECB receiver's side is reset. At the sender's side the IJBXSECB (associated with the SEND) is posted with the reason-code IJBXCPRG indicating the PURGE request and the sender is taken out of wait state.

With PURGE, the receiver may also acknowledge a CLEAR request from the sender. In this case no reason code is posted back.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

DISCONNECTing from a Communication Link

If a data communication path is not needed any more, an application can break it via the DISCONNECT function.

Format:

```
[label]     XPCC  XPCCB={addr|(1)|(S,addr)}
                  FUNC={DISCONN|DISCPRG|DISCALL|(reg)}
```

Fields in the XPCCB used by the request:

Input:

Name	Description
IJBXFCT	Function byte
IJBXIDK	IDENTIFY token returned at IDENTIFY time. This field is only used with DISCALL.
IJBXPID	Path-ID which is to be disconnected. This field is not used in case of a DISCALL request.
IJBXSUSR	8 bytes of user data moved into IJBXRUSR of the other side

Output:

Name	Description
IJBXRETC	Return codes

XPCCB fields used at the partner's side:

Output:

Name	Description
IJBXCECD	posted
IJBXSECB	posted
IJBXRECB	posted
IJBXREAS	proper reason code set (see macro MAPXPCCB)
IJBXRUSR	8 bytes of user data from IJBXSUSR of the other side

Function:

DISCONN Will check, whether the link is still 'busy'. If YES, it will reject the request with a return code. If NO, it will disconnect the link on the requestor's side.

DISCPRG Will disconnect the link unconditionally, regardless whether the link is still 'busy'. If the other side has still an outstanding SEND (SENDR) request on this link, the request will be purged, and the IJBXSECB will be posted together with a reason code (IJBXDISC 'ored' to IJBXCPRG). If an own SEND is pending this send request is cleared.

DISCALL Will unconditionally disconnect all connections set up by the corresponding application. It can only be issued by that task which issued the corresponding IDENTIFY. The DISCALL command implies a CLEAR/PURGE if necessary.

If the other side is in the wait state at the moment, it will be posted (IJBXCECB, IJBXSECB and IJBXRECB) and reason code IJBXDISC (if program requested disconnect) or IJBXABDC (if disconnect due to abnormal task termination) are 'ored' to IJBXREAS field.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

Terminating XPCC Usage

If an application does not need any more the XPCC services, it can do an XPCC 'log-off' by issuing a TERMINATE request.

Format:

```
[label]  XPCC XPCCB={addr|(1)|(S,addr)}
              FUNC={TERMIN|TERMQSCE|TERMPRG|(reg)}
```

XPCCB fields used by the system:

Input:

Name	Description
IJBXFCT	Function byte
IJBXIDK	IDENTIFY token of application

Output:

Name	Description
IJBXRETC	Return codes

Function:

TERMIN Checks first, if there are connections still available for the application. If YES, the request is rejected with a return code. If NO, the XPCC purges all internal knowledge of this application.

TERMPRG Will unconditionally execute the TERMINATE request. All available links will be unconditionally disconnected and all pending data requests will be unconditionally terminated (via DISCALL).

TERMQSCE The application indicates that it is shortly going to perform a shut-down operation. The existing connections may be still be used for data transmission, however, the XPCC will not grant any more a CONNECT request to/from the quiescing application. All still 'open-ended' connections from this application are disconnected.

Return Information Provided:

- Reg.15 and IJBXRETC in the XPCCB are set (see macro MAPXPCCB)
- Cancel due to 'invalid address'. Address validation error.

Abnormal End Processing

For clean-up purposes, the system associates the IDENTIFYs and the CONNECTs with work units, which it knows.

1. An IDENTIFY may be issued either by a VSE maintask or under control of a subtask. In the first case, the IDENTIFY is regarded as being owned by the VSE partition. If this partition terminates, the system will issue a TERMPRG for this IDENTIFY (if not already terminated).

   In the second case, the IDENTIFY is regarded as being owned by the VSE subtask. If this subtask terminates, the system will issue a TERMPRG for this IDENTIFY.

2. Each CONNECT is associated with a VSE task-ID, under which control the CONNECT was requested.

   If the task terminates, the system will disconnect all connections, which were set up by this task.

If the subsystems are using smaller work units for their applications, the corresponding subsystem has to do the DISCONNECT and TERMINATE requests for the ABENDed application ('private' subtasking).

APPENDIX C. DEVICE TYPE CODES

IPL Device Code	Actual IBM Device	PUB Device Type X'nn'	Device Type
7770	7770 Audio Response Unit	D3	Audio Response Units
7772	7772 Audio Response Unit	D4	
2501	2501 Card Reader	10	Card Reader
2540R	2540 Card Reader	11	
3504	3504 Card Reader	12	
3505	3505 Card Reader	12	
1442N2	1442N2 Card Punch	22	Card Punch
2520B2	2520B2 Card Punch	20	
2520B3	2520B3 Card Punch	20	
2540P	2540 Card Punch	21	
3525P	3525 Card Punch	23	
1442N1	1442N1 Card Read Punch	30	Card Read Punch
2520B1	2520B1 Card Read Punch	31	
2560	2560 Multifunction Card Machine	33	
2596	2596 Card Read Punch	30	
3525RP	3525 Card Punch (with read feature)	32	
5425	5424 Multifunction Card Unit	34	
	5425 Multifunction Card Unit		
FBA	3310 Fixed Block Storage Device	90	DASD
	3370 Fixed Block Storage Device	90	
	3370-2 Fixed Block Storage Device	90	
2311	2311 Disk Storage Device	60	
2314	2314 Disk Storage Device	62	
2314	2319 Disk Storage Device	62	
3330	3330 Disk Storage, Model 1 and 2	63	
	3333-1		
3330B	3330 Disk Storage Model 11	65	
3340	3340 Disk Storage without RPS Feature	69/6A	
	3344 Disk Storage w/o RPS Feature	6A	
3340R	3340 Disk Storage with RPS Feature	69/6A	
	3344 Disk Storage with RPS Feature	6A	

Figure 332 (Part 1 of 5). Device Type Codes

IPL Device Code	Actual IBM Device	PUB Device Type X'nn'	Device Type
3350	3350 Disk Storage	67	DASD (cont.)
3375	3375 Disk Storage	6B	
3380	3380 Disk Storage	6C	
3540	3540 Diskette Input/Output Unit	80	Diskette
7443	7443 System Recording File	88	
3277	3277 Display Operator Console	B0	Display Operator Console and Console Printers
3277	3284 Console Printer 3286 (the MODE operand must be 3287 entered as X'02')	B0	
1050A	3210 Console Printer Keyboard	00	Printer Keyboard
	3215 Console Printer Keyboard	00	
	3286-2 in Printer Keyboard Mode	00	
2260	2260 Display-Station	C0	Display Station
3277 (local)	3277 Display Station (MODE operand 3278 must be omitted) 3279	B0	
3277B (local)	3277 Display Station, attached in 3278 Burst Mode to a Multiplexer 3279 Channel (Mode operand must be omitted)	B0	
3277	3277 Display Units attached via 3274-1D Control Unit, mode=X'05'	B0	
2400T7	2400 7-track Magnetic Tape Unit	50	Magnetic Tape
2400T9	2400 9-track Magnetic Tape Unit	50	
3410T7	3410 7-track Magnetic Tape Unit	53	
3410T9	3410 9-track Magnetic Tape Unit	53	
3420T7	3420 7-track Magnetic Tape Unit	52	
3420T9	3420 9-track Magnetic Tape Unit	52	
3430	3430 9-track Magnetic Tape Unit	53	
8809	8809 Magnetic Tape Unit	5A	

Figure 332 (Part 2 of 5). Device Type Codes

IPL Device Code	Actual IBM Device	PUB Device Type X'nn'	Device Type
1419	1255 Magnetic Character Reader	72	MICR (Magnetic Ink Character Recognition Device)
1419	1259 Magnetic Character Reader	72	
1419	1419 Magnetic Character Reader	72	
1419P	1419 Dual Address Adapter Primary Control Unit	73	
1419S	1419 Dual Address Adapter Secondary Control Unit	74	
3890	3890 Document Reader/Inscriber	7E	Reader/ Inscriber
3895	3895 Document Reader/Inscriber	7D	Reader/ Inscriber
1287	1287 Optical Reader	77	Optical Reader
1288	1288 Optical Page Reader	77	
1419	1270 Optical Reader Sorter	72	
1419P	1275 Optical Reader Sorter Primary Control Unit	73	
1419S	1275 Optical Reader Sorter Secondary Control Unit	74	
3881	3881 Optical Mark Reader	11	
3886	3886 Optical Character Reader	7C	
1017	1017 Paper Tape Reader with 2826 Control Unit Model 1	78	Paper Tape Reader
1017TP	1017 Paper Tape Reader with 2826 Control Unit Model 2	D5	
2671	2671 Paper Tape Reader	70	
1018	1018 Paper Tape Punch with 2826 Control Unit Model 1	79	Paper Tape Punch
1018TP	1018 Paper Tape Punch with 2826 Control Unit Model 2	D6	
PRT1	3211 Printer	43	Printer
	3203-4 Printer		
	3203-5 Printer		
	3262-1 Printer		
	3262-5 Printer		
	3262-11 Printer		
	3289-4 Printer		
	4245 Printer		
	4248 Printer		

Figure 332 (Part 3 of 5). Device Type Codes

IPL Device Code	Actual IBM Device		PUB Device Type X'nn'	Device Type
1403	1403	Printer	40	Printer
1403U	1403	Printer with UCS feature	42	
1443	1443	Printer	41	
3203	3203	Printer Models 1 and 2	4A	
3211	3211	same as PRT1	43	
3800	3800	Printing Subsystem	45	
3800B	3800	Printing subsystem with Burster–Trimmer–Stacker (BTS)	45	
3800BC	3800	Printing subsystem BTS and additional CGS	45	
3800C	3800	Printing subsystem with additional Character Generation Storage (CGS)	45	
5203	5203	Printer	4C	
5203U	5203	Printer with UCS Feature	4D	
3277 (local)	3284 3286 3287 3288	Printers with 3277 or 3274–1B Control Unit (MODE operand must be entered as X'01')	B0	Terminal Printer
	3289	Printer with 3274–1B Control Unit (MODE operand must be entered as X'01')		
3277B (local)	3284 3286 3287 3288	Printers with 3277 or 3274–1B Control Unit (MODE operand must be entered as X'01')	B0	
	3289	Printer with 3274–1B Control Unit (MODE operand must be entered as X'01') attached in burst mode to a multiplexor channel		
3277	3277 4550	Printers attached via 3274–1D Control Unit, mode=X'06'	B0	
2701	2701	Data Adapter Unit	D0	Teleprocessing Lines
	2715	Data Adapter Unit	D0	
2701	Model 135	Integrated Communication Adapter (ICA)	D0	
2702	2702	Transmission Control Unit	D1	
2703	2703	Transmission Control Unit	D2	
2703	Model 138	Integrated Communication Adapter (ICA)	D2	

Figure 332 (Part 4 of 5). Device Type Codes

IPL Device Code	Actual IBM Device	PUB Device Type X'nn'	Device Type
2703	4331 Communications Adapter (ICA) for BSC or Start/Stop lines	D2	Teleprocessing Lines
2703	3704 Communications 3705 Controller in Emulation Mode	D2	
3704	3704 Communications Controller	DC	
3705	3705 Communications Controller	DC	
3705	4331 Communication Adapter (ICA) for SDLC Mode = X'10'	DC	
3725	3725 Communications Controller	DC	
3791L	3791 Local Communications	DE	
	3791 Controller	DE	
3791L	3274-1A Local Communications Contr.	DE	
UNSP	Unsupported Device	FF	Unsupported Device
UNSPB	Unsupported Device (burst)	FF	

Figure 332 (Part 5 of 5). Device Type Codes

APPENDIX D. VSE SUPERVISOR CALL TABLE

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
0	00	EXCP	none	Execute channel program
1	01	FETCH	none	Fetch a phase, except a transient phase
2	02		none	Fetch a logical transient phase ($$B.....)
3	03		none	Quiesce I/O
4	04	LOAD/SLOAD	none	Load a phase
5	05	MVCOM	none	Modify the partition communication-region
		if issued by ERP-Task	none	Fetch a physical transient ($$A.....)
6	06	CANCEL	none	Cancel a problem program or a task
7	07	WAIT	none	Wait for the posting of a control block (CCB, IORB, ECB, TECB)
8	08		none	Transfer control from a logical transient to a problem program
9	09	LBRET	none	Return from the problem program to the logical transient which issued SVC 8
10	0A	SETIME	none	Set interval timer
11	0B		none	Final return from a logical transient

Figure 333 (Part 1 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
12	0C		none	Reset switches in the partition communication region (COMREG)
13	0D		none	Set switches in the partition communication region (COMREG)
14	0E	EOJ	none	Terminate a job and go to job control for end of job step processing
15	0F	SYSIO	none	Head queue I/O request and execute the channel program
16	10	STXIT PC	none	Establish/reset linkage to user's PC routine for program check interrupts
17	11	EXIT PC	none	Return from the user's PC routine
18	12	STXIT IT	none	Establish/reset linkage to user's IT routine for interval timer interrupts
19	13	EXIT IT	none	Return from the user's IT routine
20	14	STXIT OC	none	Establish/reset linkage to user's OC routine in case of attention MSG command
21	15	EXIT OC	none	Return from the user's OC routine
22	16		none	SEIZE or RELEASE the system; enable or disable for external and I/O interrupts; set the key in a user's PSW

Figure 333 (Part 2 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
23	17		none	Store the LOAD ADDRESS of a phase at a defined user address
24	18	SETIME	none	Set TIMER INTERVAL and establish accessibility to user's TECB
25	19	HALTIO	none	Issue an HDV for a telecommunication device or for any device if issued by OLTEP.
26	1A		none	Validate address limits
27	1B		none	Issue an HDV for a telecommunication device without dequeueing the the CHANQ entry
28	1C	EXIT MR	MICR=type in SUPVR	Return from user's stacker select routine
29	1D	WAITM	none	Wait for the posting of one of the control blocks specified
30	1E		none	Reserved
31	1F		none	Reserved
32	20		none	Reserved
33	21	COMRG	none	Force task selection
34	22	GETIME	none	Provide the time and update
35	23		TRKHLD=YES in FOPT	Hold a track for exclusive use by the requesting task
36	24	FREE	TRKHLD=YES in FOPT	Free a track held by the requesting task

Figure 333 (Part 3 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
37	25	STXIT AB	none	Establish/reset linkage to user's AB routine for abnormal termination of a task
38	26	ATTACH	none	Initialize a subtask and establish its processing priority
39	27	DETACH	none	Terminate a subtask; free resources that might be held by the subtask
40	28	POST	none	Indicate occurrence of an event and ready any waiting task
41	29	DEQ	none	Indicate that a previously enqueued resource is available again
42	2A	ENQ	none	Prevent two or more task from simultaneously manipulating a shared resource (e.g. data area)
43	2B		none	Reserved.
44	2C		none	Force a unit check record to be written onto the recorder file
45	2D		none	Reserved.
46	2E		none	Allow OLTEP to run in supervisor state
47	2F	WAITF	MICR=type in SUPVR	Support the multiple wait macro WAITF for MICR type I/O routines
48	30		none	Fetch a CRT-transient phase

Figure 333 (Part 4 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
49	31		none	Allow ACF/VTAM to initiate the execution of a channel program
50	32		none	Used by LIOCS to cancel user indicating illegal SVC
51	33		none	Make directory entry information for a phase available to the requesting task
		HIPROG	none	Calculate the highest address of an overlay structure of phases or of one phase only and store it in the COMREG
52	34	TTIMER	none	Return the remaining time interval or cancel a time interval
53	35		none	Allow ACF/VTAM to schedule a user exit in an application program
54	36		none	Release page frames to selection pool (applies only to 370 mode of operation)
55	37		none	Allow SDAID to acquire processor storage needed for program initialization (applies only to 370 mode of operation)
56	38	CPCLOSE	MODE=VM or MODE=370 in SUPVR	Support the VSE/POWER-CP interface when VSE operates under VM/370.
57	39	GETPRTY	none	Return partition priorities to the requesting task

Figure 333 (Part 5 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
		SETPRTY	none	Change partition priorities as specified
58	3A	INVPART	none	Initialize partition
59	3B	INVPAGE	none	Initialize tables or invalidate pages
60	3C	GETDADR	none	Return the virtual equivalent of a real I/O area plus offset
61	3D	GETVIS	none	Request allocation of storage within the same partition or within the SVA
62	3E	FREEVIS	none	Free storage requested through a GETVIS macro
63	3F	USE	none	Indicate system resource is in USE
64	40	RELEASE	none	RELEASE a system resource
65	41	CDLOAD	none	Load a phase in the requesting partition's GETVIS area unless that phase is already in the SVA
66	42	RUNMODE	none	Return the system's operating mode
67	43	PFIX	none	FIX pages in processor storage
68	44	PFREE	none	FREE pages in processor storage
69	45	REALAD	none	Return the REAL address corresponding to a given virtual address

Figure 333 (Part 6 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
70	46	VIRTAD	none	Return the virtual address corresponding to a given real address
71	47	SETPFA	none	Establish or terminate linkage to a user Page Fault Appendage routine
72	48	GETCBUF	none	GET Copy buffer for IDAL of tape ERP
		FREECBUF		FREE Copy BUFfer for IDAL of tape ERP
73	49	SETAPP	none	Allow linkage to channel-end appendage routines
74	4A	PFIXREST	none	Fix page(s) in processor storage for restart
		PFIXCHPT	none	Build parameter list for PFIXREST during checkpointing
75	4B	SECTVAL	RPS=YES in FOPT	Calculate a sector value for a disk device with the RPS feature
76	4C		none	Initiate recording on VM recorder file
77	4D	TRANSCSW	none 370 mode only	Returns the virtual address of an ERP CCW address copied from the pertinent CSW
78	4E	CHAP	none	Change the processing priority of the requesting task
79	4F		none	Reserved
80	50	SETT	TTIME=part-id in FOPT	Set task time interval

Figure 333 (Part 7 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
81	51	TESTT	TTIME=part-id in FOPT	Return remaining task time interval or cancel the time interval
82	52		none	Set monitor call and/or branch, for ICCF
83	53	ALLOCATE	none	Allocate real or virtual partitions
84	54	SETLIMIT	none	Set partition sizes
85	55	RELPAG	none	Release the contents of one or more pages
86	56	FCEPGOUT	none	Force a page-out operation for more pages
87	57	PAGEIN	none	Request a page-in operation for more pages
88	58	TPIN	none	Start TP balancing
89	59	TPOUT	none	Stop TP balancing
90	5A	PUTACCT	JA=YES in IPL SYS-CM)	Provide interface with VSE/POWER for additional, user-provided account information
91	5B		JA=YES in IPL SYS-CMD	Provide interface with VSE/POWER for standard account information
92	5C	XECBTAB	none	Define, delete, or check an entry in the cross-partition ECB table
93	5D	XPOST	none	Set the traffic bit in a cross-partition ECB and ready any waiting tasks
94	5E	XWAIT	none	Wait for a cross-partition ECB to be posted

Figure 333 (Part 8 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
95	5F	EXIT AB	none	Return from a user's abnormal termination routine
96	60	EXIT TT	TTIME=part-id of FOPT	Return from a user's task timer exit routine
97	61	STXIT TT	TTIME=part-id of FOPT	Establish/reset linkage of user task's timer exit routine for task time interval end
98	62	EXTRACT	none	Extract system control information
		MODCTB	none	Modify a PUB2 table entry
99	63	GETVCE	none	Return a specific volume characteristics and/or track balance information
100	64	PFIX PFREE	none (ECPS:VSE mode only)	Fix or free a page in the SYSTEM GETVIS area
101	65	MODVCE	none	Update the volume characteristics table
102	66	GETJA	JA=YES in IPL SYS-CMD	Update the fields in the requesting partition's job accounting table
103	67		none	Execute I/O operations for SYSFIL on on FBA device, if FBA supported
104	68	EXTENT	none	Add, return, or delete DASD extent information
105	69	SUBSID	none	Accept, return, and delete subsystem identification information.

Figure 333 (Part 9 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
106	6A		none	Set the storage key for a specific area to the value in Register 0 (ICCF)
107	6B	DEVREL	none	Release a device that was "in use"
		DEVUSE		Force a device to be set "in use"
		GETFLD		Retrieve task-related information
		MODFLD		Modify task-related information
		RLOCK		Obtain access to a specified resource or wait for it
		SENTER		Enter a sub-system
		SLEAVE		Leave a sub-system
		TREADY		Post or cancel a task
		TSTOP		Deactivate current task or partition
		VIO POINT		Piont to VIO control block (VIORB)
108	6C	SECHECK	SEC=nn in IPL SYS-CMD	Check user's authority for accessing the specified resource
109	6D	PAGESTAT	none	Return status of a page or a set of pages
110	6E	LOCK/UNLOCK	none	Protect or release a serially re-usable resource against concurrent access of two or more tasks
111	6F		none	Reserved
112	70	MSAT	none	Build, return, or delete stored assignment information
113	71	XPCC	none	Cross-partition communication services

Figure 333 (Part 10 of 11). VSE Supervisor Calls

SVC Code		Imperative Macro that Issues the SVC	Activation Option to be Specified	Function
DEC	HEX			
114	72	VIO	none	Allocate, deallocate or extend VIO file
115	73	PWROFF	none	Software initiated power-off for 4361
116	74	NPGR	none	Allocate or reallocate programmer LUB's
117	75		none	Reserved
118	76	CPCOM	MODE=VM or MODE=370 in SUPVR	CP command interface (CPCOM macro)
		• • •	• • •	• • •
140	8C		none	Reserved
141	8D	VSIUCV	MODE=VM in FOPT	Provide subsystem support for VM/VCNA (VTAM Communication Network Application)
142	8E		none	Reserved
		• • •	• • •	• • •
255	FF		none	Reserved

Figure 333 (Part 11 of 11). VSE Supervisor Calls

APPENDIX E. SAMPLES

Track Hold Processing

Figure 334 on page 820 shows the initialized significant bytes of the track hold mechanism.

Figure 335 on page 821 illustrates the pointers and table entries after several track hold requests have been issued.

Figure 336 on page 822 summarizes the sequence of events leading to the situations shown in Figure 335 on page 821, Figure 337 on page 823, and Figure 338 on page 824.

When a task requests a hold on a track/block that is already held by another task, the high-order bit of the flag-and-counter byte is turned on (for example, entry No. 1 in Figure 335 on page 821). When a task requests a hold on a track/block it holds itself, the flag-and-counter byte is incremented by one (for example, entry No. 4 in Figure 335 on page 821).

On release of a track/block by the holding task, and provided the counter is zero before the release, any task or tasks that are waiting for that track/block are brought out of the wait state. The supervisor then returns to task selection and, if the next selected task was waiting for this track/block, its hold request is honored. Any other task or tasks that were waiting for the track/block now remain ready-to-run, but if such a task gains control before the track/block has again been released, that task returns to the wait state.

If the counter is not zero before the release, then only the counter is decremented by one so that the track/block remains held by the same task. For illustrations of these operations, compare Figure 335 on page 821 and Figure 337 on page 823, entries number one and four.

Track Hold Table Entries

Free List Pointer THFLPTR	Entry No.	Chain Byte	CCB/IORB Addr.	BBCCHH00 or LLPBN+ULPBN	Backward Pointer	Flag and Counter	Task ID
00——>	0	01	zeros	zeros	zeros	zeros	zeros
	1<	02	zeros	zeros	zeros	zeros	zeros
	2<	03	zeros	zeros	zeros	zeros	zeros
PUBS Track Hold Pointers (PUBOPTN)	3<	04	zeros	zeros	zeros	zeros	zeros
	4<	05	zeros	zeros	zeros	zeros	zeros
FF	5<	06	zeros	zeros	zeros	zeros	zeros
FF	6<	07	zeros	zeros	zeros	zeros	zeros
FF	7<	08	zeros	zeros	zeros	zeros	zeros
FF	8<	09	zeros	zeros	zeros	zeros	zeros
FF	9<	FF	zeros	zeros	zeros	zeros	zeros

Notes:

THFLPTR: The track hold free list pointer (1 byte) contains a pointer to the first entry in the free list or X'FF' when the track hold table is full.

BBCCHH00:

- 00,
- cylinder cylinder,
- head head,
- 00

LLPBN: Low limit physical block number

ULPBN: Upper limit physical block number

Figure 334. Track Hold Table Example. Initial contents of significant bytes used by track hold requests.

Track Hold Table Entries

```
┌───────────┬─────┬─────┬────────┬────────────┬────────┬────────┬──────┐
│Free List  │     │     │        │            │        │        │      │
│Pointer    │Entry│Chain│CCB/IORB│BBCCHH00 or │Backward│Flag and│Task  │
│THFLPTR    │ No. │Byte │ Addr.  │LLPBN+ULPBN │Pointer │Counter │ ID   │
├───────────┼─────┼─────┼────────┼────────────┼────────┼────────┼──────┤
│   05──────┐  0       01 │  xxx   │ Track  1A  │PUB ptr.│   40   │  aa  │
│           │  A       │  │        │            │        │        │      │
│┌──────────┴──┘  ┌────┘  │        │            │        │        │      │
││          │  1<─┘    02 │  xxx   │ Track  1B  │   00   │   80   │  aa  │
││          │     ┌────┘  │        │            │        │        │      │
├│──────────│  2<─┘    04 │  xxx   │ Track  1C  │   01   │   00   │  bb  │
││ PUBS     │     ┌────┘  │        │            │        │        │      │
│Track HoldA>>>3  │    FF │  xxx   │ Track  2A  │PUB ptr.│   40   │  aa  │
│Pointers  A│     │       │        │            │        │        │      │
│(PUBOPTN) A│  4<─┘    FF │  xxx   │ Track  1D  │   02   │   40   │  aa  │
├│─────────A│             │        │            │        │        │      │
││         A└─>5       06 │ zeros  │   zeros    │ zeros  │ zeros  │zeros │
││2nd Dev.>>           │  │        │            │        │        │      │
││  03      │     ┌────┘  │        │            │        │        │      │
││          │  6<─┘    07 │ zeros  │   zeros    │ zeros  │ zeros  │zeros │
││  FF      │     ┌────┘  │        │            │        │        │      │
││          │  7<─┘    08 │ zeros  │   zeros    │ zeros  │ zeros  │zeros │
││  FF      │     ┌────┘  │        │            │        │        │      │
││1st Device│  8<─┘    09 │ zeros  │   zeros    │ zeros  │ zeros  │zeros │
│└<──00     │     ┌────┘  │        │            │        │        │      │
│           │  9<─┘    FF │ zeros  │   zeros    │ zeros  │ zeros  │zeros │
│   FF      │             │        │            │        │        │      │
└───────────┴─────────────┴────────┴────────────┴────────┴────────┴──────┘
```

Figure 335. Track-Hold Table Example. Task aa holding tracks 1A, 1B, and 1D (2 holds) on 1st device, and track 2A on 2nd device; task bb holding track 1C on first device; a task is waiting to hold track 1B on first device.

Sequence of Requests	Tasks			Remarks
	aa	bb	cc	
	Hold 1A			Entry queued
	Hold 1B			Entry queued
		Hold 1C		Entry queued
	Hold 2A			Entry queued
	Hold 1D			Entry queued
V	Hold 1D			Counter incremented
			Hold 1B	Entry flagged and requester put into into wait state.

The table entries and pointers at this stage are illustrated by Figure 335.

	Free 1B			Flag turned off, waiting task (cc) made ready-to-run, and task selection entered; if task cc is selected, its request for track 1B is honored
	Free 1A			Entry dequeued.
V	Free 1D			Counter decremented.

The table entries and pointers at this stage are illustrated by Figure 337 on page 823.

	Free 2A			Entry dequeued.
	Free 1D			Entry dequeued.
		Free 1C		Entry dequeued.
V			Free 1B	Entry dequeued.

All tracks have now been freed as shown in Figure 338 on page 824.

Figure 336. Example of Tracks Held and Freed by Three Tasks

Track Hold Table Entries

```
┌────────────┬──────┬──────┬────────┬───────────┬────────┬────────┬─────┐
│Free List   │      │      │        │           │        │        │     │
│Pointer     │Entry │ Chain│CCB/IORB│BBCCHH00 or│Backward│Flag and│Task │
│THFLPTR     │ No.  │ Byte │ Addr.  │LLPBN+ULPBN│Pointer │Counter │ ID  │
├────────────┼──────┴──────┼────────┼───────────┼────────┼────────┼─────┤
│  00──────────>0       05 │  xxx   │ Track 1A  │  zero  │   00   │ 00  │
│            │          │  │        │           │        │        │     │
│            ┌──────────┘  │        │           │        │        │     │
│            │ 1<──┐    FF │  xxx   │ Track 1B  │   04   │   00   │ cc  │
│            │     │       │        │           │        │        │     │
├┌───────────┼─>2  │    04 │  xxx   │ Track 1C  │PUB ptr.│   40   │ bb  │
││ PUBS      │     │       │        │           │        │        │     │
│Track HoldA>>>3   │    FF │  xxx   │ Track 2A  │PUB ptr.│   40   │ aa  │
│Pointers   A│     │       │        │           │        │        │     │
│(PUBOPTN)  A│ 4   └────01 │  xxx   │ Track 1D  │   02   │   00   │ aa  │
├│──────────A│             │        │           │        │        │     │
││2nd       A└─>5       06 │ zeros  │   zeros   │ zeros  │   00   │ 00  │
││  device A            │  │        │           │        │        │     │
││   03>>>>>       ┌────┘  │        │           │        │        │     │
││           │ 6<──┘    07 │ zeros  │   zeros   │ zeros  │   00   │ 00  │
││   FF      │     ┌────┘  │        │           │        │        │     │
││           │ 7<──┘    08 │ zeros  │   zeros   │ zeros  │   00   │ 00  │
││   FF      │     ┌────┘  │        │           │        │        │     │
││1st Device │ 8<──┘    09 │ zeros  │   zeros   │ zeros  │   00   │ 00  │
│└<──02      │     ┌────┘  │        │           │        │        │     │
│            │ 9<──┘    FF │ zeros  │   zeros   │ zeros  │   00   │ 00  │
│   FF       │             │        │           │        │        │     │
└────────────┴─────────────┴────────┴───────────┴────────┴────────┴─────┘
```

Figure 337. Track-Hold Table Example. Task aa has released holds on tracks 1B and 1A, and one of the holds on track 1D; task cc has been taken out of the wait state and has been selected to run, so it now holds track 1B.

Track Hold Table Entries

```
┌────────────┬──────┬──────┬────────┬────────────┬─────────┬─────────┬──────┐
│Free List   │      │      │        │            │         │         │      │
│Pointer     │Entry │ Chain│CCB/IORB│BBCCHH00 or │Backward │Flag and │Task  │
│THFLPTR     │ No.  │ Byte │ Addr.  │LLPBN+ULPBN │Pointer  │Counter  │ ID   │
├────────────┼──────┼──────┼────────┼────────────┼─────────┼─────────┼──────┤
│   01       │  0<.    05  │  xxx   │ Track 1A   │   00    │   00    │  00  │
│   │        │┌────.─────┤  │        │            │         │         │      │
│   │        │├────.─────   │        │            │         │         │      │
│   └────────┼─>1 .    02  │  xxx   │ Track 1B   │   00    │   00    │  00  │
│            │    . ┌────┘  │        │            │         │         │      │
├─────────   │  2<.─┘   04  │  xxx   │ Track 1C   │   00    │   00    │  00  │
│  PUBS      │    . ┌────┘  │        │            │         │         │      │
│Track HoldA>>>3 ......00  │  xxx   │ Track 2C   │   00    │   00    │  00  │
│Pointers   A│      │      │        │            │         │         │      │
│(PUBOPTN)  A│  4<──┘   03  │  xxx   │ Track 1D   │   00    │   00    │  00  │
├─────────   <<<<<<<<<<<<<<  │        │            │         │         │      │
│            └─>5      06  │ zeros  │   zeros    │   00    │   00    │  00  │
│ 2nd Device         │   │        │            │         │         │      │
│    FF      │      ┌────┘  │        │            │         │         │      │
│            │  6<──┘   07  │ zeros  │   zeros    │   00    │   00    │  00  │
│    FF      │      ┌────┘  │        │            │         │         │      │
│            │  7<──┘   08  │ zeros  │   zeros    │   00    │   00    │  00  │
│    FF      │      ┌────┘  │        │            │         │         │      │
│ 1st Device│  8<──┘   09  │ zeros  │   zeros    │   00    │   00    │  00  │
│    FF      │      ┌────┘  │        │            │         │         │      │
│            │  9<──┘   FF  │ zeros  │   zeros    │   00    │   00    │  00  │
│    FF      │             │        │            │         │         │      │
└────────────┴─────────────┴────────┴────────────┴─────────┴─────────┴──────┘
```

Figure 338. Track-Hold Table Example. Situation after total release.

INDEX

VSE/Advanced Functions
Diagnosis Reference
Supervisor
Order No. LY33-9107-0

READER'S
COMMENT
FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

____

____

____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? ______________________________

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

LY33-9107-0

Reader's Comment Form

Cut or Fold Along Line

Fold And Tape **Please Do Not Staple** **Fold And Tape**

International Business Machines Corporation
Department 6R1
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** **Fold And Tape**

IBM®

VSE/Advanced Functions Diagnosis Reference Supervisor (File No. S370/4300-36) Printed in U.S.A. LY33-9107-0

VSE/Advanced Functions
Diagnosis Reference
Supervisor
Order No. LY33-9107-0

READER'S
COMMENT
FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*
Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

____
____
____

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? ________________________

Number of latest Newsletter associated with this publication: ____________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

LY33-9107-0

Reader's Comment Form

Fold And Tape Please Do Not Staple Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape Please Do Not Staple Fold And Tape

IBM®

Cut or Fold Along Line

VSE/Advanced Functions Diagnosis Reference Supervisor (File No. S370/4300-36) Printed in U.S.A. LY33-9107-0

IBM

LY33-9107-0