GC33-5380-1
File No. S370-37

Systems

DOS/VS Serviceability Aids and Debugging Procedures

Release 29

IBM

Second Edition (November, 1973)

This edition applies to Version 5, Release 29, of the IBM Disk Operating System/Virtual Storage, DOS/VS, and to all subsequent versions and editions until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein. Before using this publication in connection with the operation of IBM systems, consult the *IBM System/360 and System/370 Bibliography,* GA22-6822, for the editions that are applicable and current.

This is a major revision of, and obsoletes, GC33-5380-0. It includes changes reflecting support for the System/370 Models 115, 155-II, and the Model 158. It also includes changes reflecting support for new devices and other DOS/VS enhancements.

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 reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratory, Publications Department, P.O. Box 24, Uithoorn, The Netherlands. Comments become the property of IBM.

PREFACE

THIS MANUAL . . .

. . . is intended to guide System/370 operators and programmers using DOS/VS in determining and isolating the cause of a system malfunction.

METHOD OF PRESENTATION

Serviceability aids and how to use them are described in this manual through extensive use of diagrams and examples. This enables fast retrieval of information and largely avoids the need to use other publications in order to analyze the dumps and printouts discussed.

Contents and addresses shown in the illustrations are subject to change and are shown only as an aid to offline debugging of DOS/VS release 29. IBM will not be responsible for any system malfunction resulting from a change made by the user of any contents or addresses of the tables and blocks described.

SUBJECTS COVERED

There are four major sections;

SECTION 1: *Introduction*, introduces the serviceability aids detailed in Section 2, and the debugging procedures described in Sections 3 and 4.

SECTION 2: *Serviceability Aids*, describes in detail the serviceability aids, showing in flowchart form how to use them, and recommending when to use them. Examples show how to analyze dumps and printouts in conjunction with the debugging procedures of Section 3 and 4.

SECTION 3: *Debugging for Operators*, consists of flowcharts that help the operator to isolate the cause of a system malfunction. The operator is instructed when to use the procedures of Section 2 to ensure that information is gathered from the system.

SECTION 4: *Debugging for programmers,* this section is divided into two parts:

Part 1 consists of checklists in flowchart form that recommend the method of analysis and choice of serviceability aids best suited to isolate the cause of a given type of system malfunction. An indication is made on the flowcharts when it is considered necessary to inform your IBM customer engineer when it is not possible to isolate the cause of an error. System information to be saved for the IBM CE is also listed at these points in the flowcharts.

Part 2 is a general description of the DOS/VS supervisor/problem program interface tables, information blocks and save areas. It shows how to locate these areas in a dump, and how to analyze the data during offline program debugging. Debugging aids for high level languages are described in publications dealing with the specific language.

PREREQUISITE KNOWLEDGE

Operators using this manual must be familiar with the following IBM publications:

DOS/VS Operating Procedures	*GC33 – 5378*
DOS/VS Messages	*GC33 – 5379*

Programmers using Section 4 must be familiar with the following IBM publications:

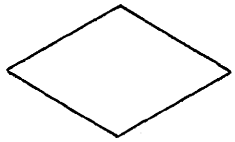
IBM System/370 Principles of Operation	*GA22 – 7000*
DOS/VS System Management Guide	*GC33 – 5371*

Other IBM publications referenced in this manual are listed in the bibliography at the back.

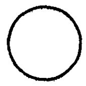
OVERVIEW OF CONTENTS

SYMBOLS USED

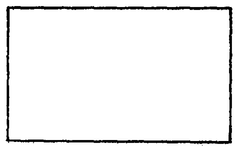
Start or finish



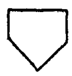
Decision to determine which alternative path to follow



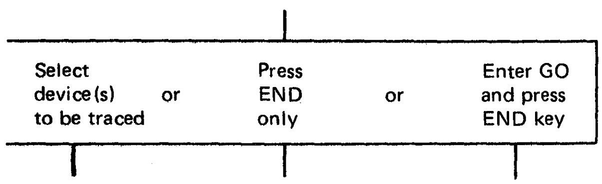
Exit to, or entry from another part of the flowchart on the same page



Process or action

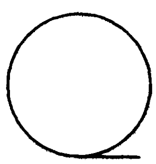


Entry to, or exit from a flowchart to link with a flowchart on another page

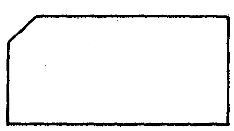


Multiple choice

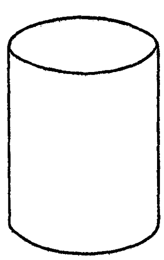
Console printer keyboard: operator input, or message output



Magnetic tape



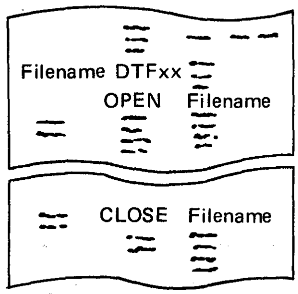
Card file



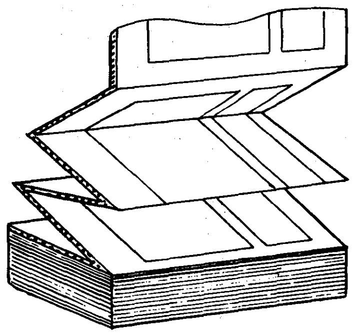
Disk drive or pack

SYMBOLS USED

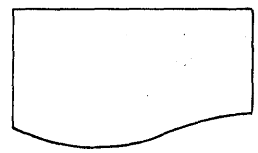
Program listings



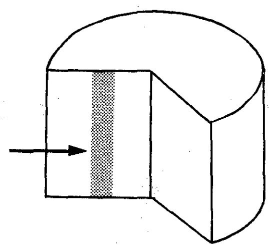
Line printer output



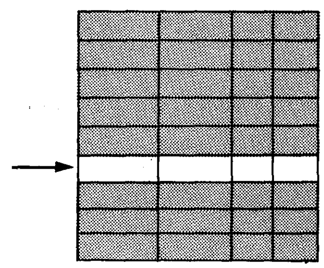
or



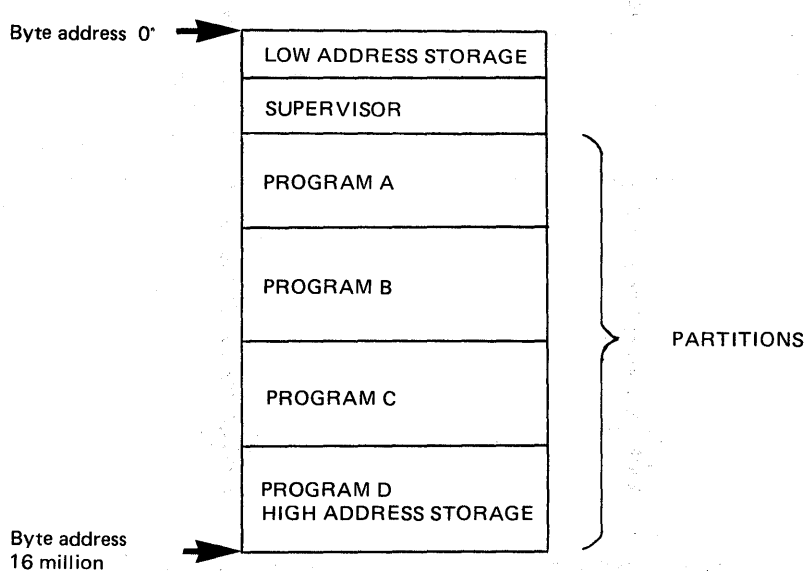
Data areas on disk packs

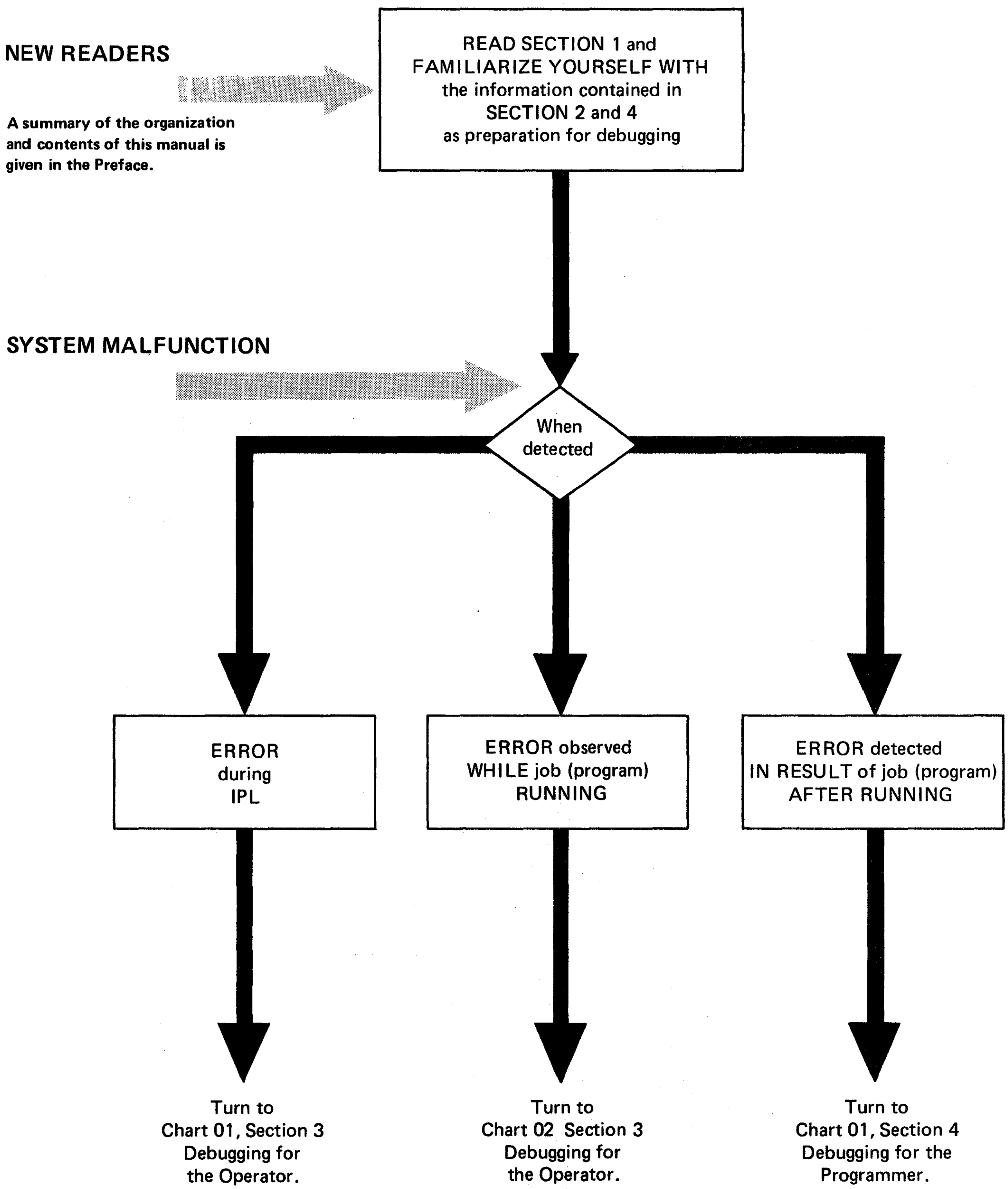


or



Virtual storage




NEW READERS
A summary of the organization and contents of this manual is given in the Preface.
READ SECTION 1 and FAMILIARIZE YOURSELF WITH the information contained in SECTION 2 and 4 as preparation for debugging
SYSTEM MALFUNCTION
When detected
ERROR during IPL
ERROR observed WHILE job (program) RUNNING
ERROR detected IN RESULT of job (program) AFTER RUNNING
Turn to Chart 01, Section 3 Debugging for the Operator.
Turn to Chart 02 Section 3 Debugging for the Operator.
Turn to Chart 01, Section 4 Debugging for the Programmer.

ABBREVIATIONS

**CE, SE, IBM CE/SE is the IBM representative*

AB	Abnormal Termination
ACB	Access Method Control Block
ADDR	Address
AMS	Access Method Services
AP	Asynchronous Processing
AR	Attention Routine
ASCII	American National Code for Information Interchange
BC	Basic Control
BBOX	Boundary Box
BG	Background Partition
BIN	Binary
BSC	Binary Synchronous Communication
BTAM	Basic Telecommunication Access Method
CAW	Channel Address Word
CC	Chain Command
CCB	Command Control Block
CCH	Channel Check Handler
CD	Chain Data
CE	Customer Engineer*
CHANQ	Channel Queue
CNT	Count
COBOL	Common Business Oriented Language
COMREG	Communication Region
CPU	Central Processing Unit
CR	Combined Recording
CR	Control Register
CRT	Cathode Ray Tube
CSECT	Control Section
CSW	Channel Status Word
CUA	Area Station Address
CUU	Channel and Device Unit Number
CYL	Cylinder (Disk Extent)
DASD	Direct Access Storage Device
DAT	Dynamic Address Translation
DEC	Decimal
DOC	Display Operators Console
DTF	Define the File
DIB	Disk Information Block
EBCDIC	Extended Binary-Coded-Decimal Interchange Code
EC	Extended Control
ECB	Event Control Block
ECC	Error Checking and Correction
ECSW	Extended Channel Status Word
EFL	Error Frequency Limit
EOB	End of Block (Press End/Enter)
EOD	End of Day (End of Shift/System Switch Off)
EOF	End of File
EOJ	End of Job
EREP	Environmental Recording, Editing and Printing
ERP	Error Recovery Procedure
ERPIB	Error Recovery Program Interface Bytes
EVA	Error Volume Analysis
EXT	External
FAVP	First Available Pointer
FCB	Forms Control Buffer
FG	Foreground Partition
FICL	First in Class List
F/L	Fetch/Load
FLPTR	Free List Pointer
FOCL	First on Channel List
FORTRAN	Formula Translation
FP	Floating Point
GPR or GR	General Purpose Register
GSVC	Generalized Supervisor Call
HD	Head (Disk Extent)
HEX	Hexadecimal
HIO	Halt I/O

HIR	Hardware Instruction Retry
ICA	Integrated Communications Adapter
ID	Identifier
IDAL	Indirect Data Address List
ILC	Instruction Length Code
IMPL	Initial Micro-Program Load
INT	Interrupt
INTVN	Intervention
INVAL	Invalid
I/O	Input/Output
IOCS	Input/Output Control System
IPL	Initial Program Load
IR	Individual Recording
IT	Interval Timer
JAI	Job Accounting Interface
JCC	Job Control Command
JCL	Job Control Language
JCS	Job Control Statement
JIB	Job Information Block
K	1024 Bytes (Dec)
KBD	Keyboard
LIK	Logical Transient Owner Identification Key
LIOCS	Logical Input/Output Control System
LMT	Line Mode Table
LOC	Location
LTA	Logical Transient Area
LTK	Logical Transient Key
LUB	Logical Unit Block
MCAR	Machine Check Analysis and Recovery
MCI	Machine Check Interrupt
MCK	Machine Check
MDR	Miscellaneous Data Record
MFCM	Multifunction Card Machine
MICR	Magnetic Ink Character Reader
MPS	Multiprogramming System
MPX	Multiplexer
MSG	Message
NICL	Number in Class List
NSD	Non Sequential Disk
OC	Operator Communication
OCR	Optical Character Reader
OD	Output Device
OLTEP	Online Test Executive Program
OLTS	Online Test System
PART	Partition
PC	Program Check
PCI	Program Controlled Interrupt
PCIL	Private Core Image Library
PD	Problem Determination
PDAID	Problem Determination Aid
PDS	Page Data Set
PER	Program Event Recording
PF	Page Frame
PFT	Page Frame Table
PFTX	Page Frame Table Extension
PG	Page
PGM	Program
PHO	Page Fault Handling Overlap
PIB	Program Information Block
PIB2	Program Information Block Extension
PIK	Partition Identification Key
PIOCS	Physical Input/Output Control System
PMGR	Page Manager
POWER	Priority Output Writers, Execution Processors, and Readers
POWER RJE	Power Remote Job Entry
PP	Page Pool
PPBEG	Start of Problem Program Area

ABBREVIATIONS

PRT	Partition
PSLD	Private second level directory
PSW	Program Status Word
PT	Page Table
PTA	Physical Transient Area
PTF	Program Temporary Fix
PTR	Pointer
PUB	Physical Unit Block
QTAM	Queued Telecommunication Access Method
RAS	Reliability, Availability, and Serviceability
RDE	Reliability Data Extractor
REQID	I/O Requestor Partition or System Task Identity
REQD	Required
RF	Recorder File
RID	Routine Identifier
RLD	Relocation Dictionary
RMS	Recovery Management Support
RMSR	Recovery Management Support Recorder
RPG	Report Program Generator
RPG	Report Program Generator
RTN	Routine
SAB	Seek Address Block
SCP	System Control Program
SCU	Secondary Control Unit
SDAID	System Debugging Aid
SDL	System Directory List
SE	System Engineer*
SEREP	Stand-Alone EREP
SIO	Start I/O
SLD	Second Level Directory
SLI	Suppress Length Indication
SPVR	Supervisor
SRI	System Recovery Incident
STAB	Segment Table
STMT	Statement
SVA	Shared Virtual Area
SVC	Supervisor Call
SYSCOM	System Communication Region
SYSREC	System Recorder File
SYSRES	System Residence Unit
SYSVIS	Page Data Set
TCB	Translation Control Block
TES	Tape Error Statistics
TIB	Task Information Block
TIC	Transfer in Channel
TIK	Task Interrupt Key
TKREQID	I/O Requestor's Task Identity
TOD	Time of Day
TP	Teleprocessing
TPER	Teleprocessing Error Record
TXT	Text
UCS	Universal Character Set
UCSB/UCB	Universal Character Set Buffer
UPSI	User Program Switch Indicator
VDU	Virtual Display Unit
VSAM	Virtual Storage Access Method
VS	Virtual Storage
VTOC	Volume Table of Contents
WTM	Write Tape Mark
X' '	Hexadecimal Value
YR	Year

**CE, SE, IBM CE/SE is the IBM representative*

LIST OF TABLES ILLUSTRATIONS AND FLOWCHARTS

Tables

Illustrations and flowcharts

Section 1: Introduction

Section 2-A: Dumps of, and changes to Real and Virtual Address Areas

Section 2-B: Trace Routines

Section 2-C: Library Display Programs and Utilities

LIST OF TABLES ILLUSTRATIONS AND FLOWCHARTS

LIST OF TABLES
ILLUSTRATIONS
AND FLOWCHARTS

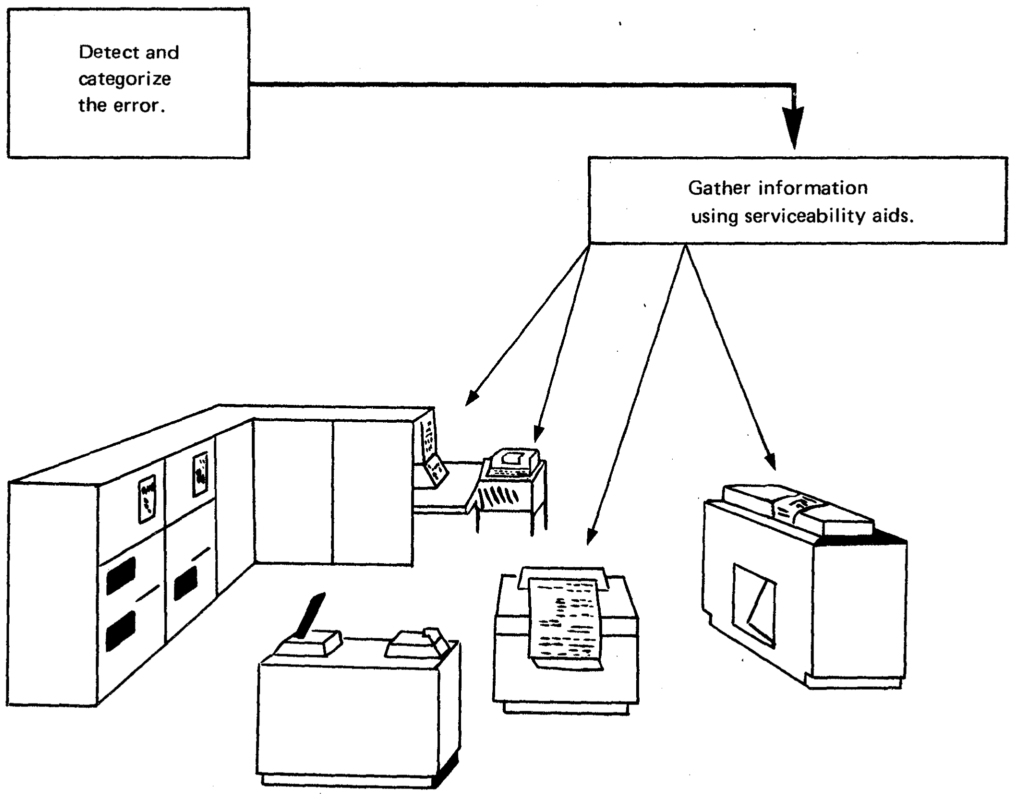
Section 4: Debugging for Programmers, part 2

LIST OF TABLES ILLUSTRATIONS AND FLOWCHARTS

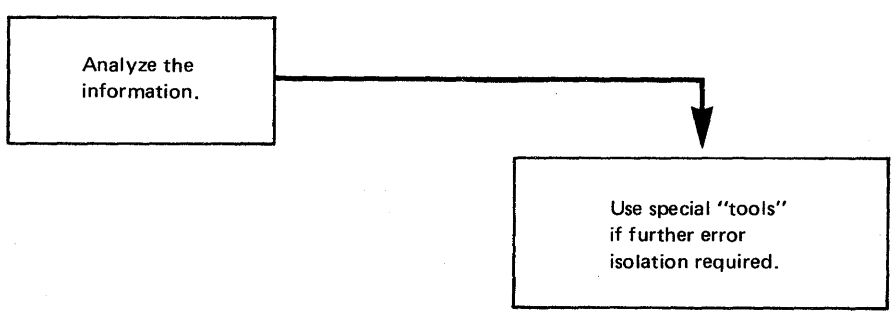
Appendixes

CONTENTS

Hands-on debugging



Offline debugging



Serviceability aids are "tools" offered by IBM and are designed to gather system information whenever a malfunction occurs on a System/370.

A malfunction can be caused by a programming error or by a hardware failure.

Some of the serviceability aids that gather system information when programming errors occur are:

- DUMPS of specified real and virtual address areas
- DUMPS or DISPLAYS of general registers,
  control registers,
  floating point registers, and
  program status words
- FORMATTED PRINTOUT of the DOS/VS supervisor tables and information blocks
- The ability to ALTER any register or any area of virtual storage
- Problem determination aids, PDAIDS (event tracing routines)
- System debugging aids, SDAIDS (program event recording and tracing routines)
- Disk and tape LABEL INFORMATION display programs
- LISTIO and MAP commands (aids that list devices used per partition, and that map virtual storage organization during system operation)
- Commands that allow information contained on disk files using VSAM (Virtual Storage Access Method) to be printed, listed, or verified
- Programs that display libraries and allow them to be edited and maintained
- Error messages issued by the system that inform the operator about the nature of an error.

The serviceability aids that detect hardware failures and produce formatted output concerning this failure are:

- RMS – Recovery Management Support
- EREP – Environmental Recording, Editing, and Printing
- OLTEP – Online Test Executive Program
- SEREP – Stand-alone EREP
- Micro-program diagnostic aids, (Models 115, 125, and 158)

In addition to the above aids, the Models 115 and 125 are provided with a micro-program recording facility that records certain types of hardware errors on DISKETTE. The errors recorded on DISKETTE can be displayed and analyzed by the IBM CE using the Maintenance Program Selection and Log Analysis displays.

A similar facility is provided on the Model 158 in the form of displays obtained by the use of the Service function.

A reference chart at the front of Section 2 lists the IBM serviceability aids, which are described in detail in that section.

Serviceability aids offered by IBM that are designed to gather system information specifically for use with high-level languages (RPG II, PL/I, American National Standard COBOL, and FORTRAN) are not described in this manual. Details about these aids are found in the corresponding manuals for the processor being used.

What is Debugging?

Debugging is a procedure that is followed to isolate an error (sometimes referred to as a bug) that prevents programs from being correctly executed by a computer system.

Debugging requires the coordinated efforts of operators and programmers, and is divided into two distinct actions:

- Hands-on debugging
- Offline debugging.

Hands-on debugging entails the examination of available symptoms and indications and the saving of information by the operator when a system malfunction occurs.

Offline debugging requires the analysis and the isolation of an error by the programmer, using data gathered during hands-on debugging.

IBM has provided special programs, commands, and procedures called serviceability aids, or tools, to help in gathering information about a system malfunction. These aids can be initialized by the operator and are of special interest when an error is obscure.

The two debugging actions (hands-on and offline) can be divided into the following four phases as shown in figure 1.1:

1. Determine the type of malfunction.
2. Gather information.
3. Analyze the information.
4. Use aids for further error isolation if required.

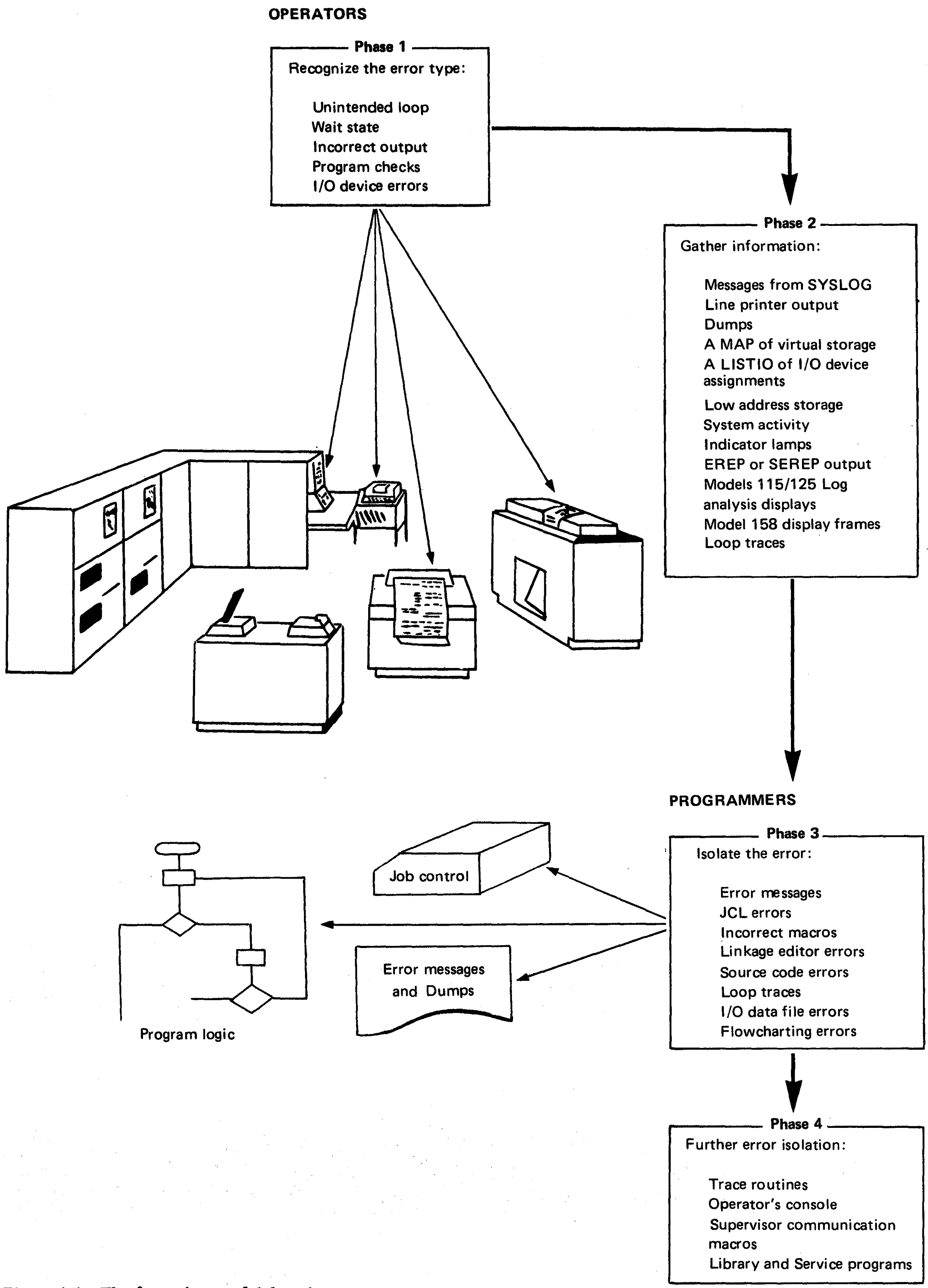


Figure 1-1. The four phases of debugging

System Malfunctions

Generally speaking, a system malfunction is said to have occurred whenever a program did not do what it was expected to do. A system malfunction can be due to one or more of the following:

- An operator error or job set (JCL)
- An error in the program logic, a coding error, or the misuse of instructions
- A hardware failure
- An unusual circumstance during program execution.

Because of the many circumstances in which errors may occur, system malfunctions manifest themselves in different ways.

The physical size of a given system, its environment, and the type of programs used also play a part in how a particular error affects system operation.

During single-partition batch jobs (BJ)

In this type of environment, the easiest way to recover from an error is to cancel the job and begin it again from the IPL procedure. However, to deal with a program that has been operating successfully for several hours prior to the error, alternative methods must be used.

Also in this type of environment, operators "get to know" the programs and can recognize when the programs do not appear to be performing the same as before. Hands-on debugging can be performed without interfering with the execution of other jobs.

Multiprogramming environment (MPS)

To cancel, re-IPL, and restart jobs after a malfunction in this type of environment would delay both production and debugging procedures. Hands-on debugging is more difficult than with BJ, and the method used to gather information must be carefully chosen. It is also impossible to "get to know," by repeated use of the same programs, exactly what each job should be doing at any given time, and so it is more difficult to recognize a system malfunction.

Teleprocessing (TP)

Since teleprocessing is normally executed on multi-programming systems, the same problems are met as those described under MPS. Additionally, the cancellation of jobs is more difficult. The difficulty increases in proportion to the number of terminals online, and the number of active partitions, when the malfunction occurred. Hands-on debugging cannot be attempted without informing and affecting all the terminals.

During program testing

Although systems may not be large or complex in this type of environment, it is less likely that the operator will know the programs. In this environment, the testing of new programs and the simulation of space flights, aircraft structures, traffic controls systems, etc., are carried out daily, with unpredictable results in most cases. Hands-on debugging can be done only by the programmer. Even recognizing a system malfunction is in itself difficult. Gathering the right information is of paramount importance, to enable the programmer to debug offline.

The previous paragraphs indicate that when a system malfunction occurs, the operator must be able to recognize it as such, decide on whether or not to use hands-on debugging to make a possible recovery, and decide on the best method of gathering information that will help the programmer.

A description follows of the main types of system malfunctions, how to recognize them, and how to treat them.

Type of Malfunctions

LOOPS

Definition

A loop in a program is the repetitive execution of a sequence of CPU (central processing unit) instructions.

If the number of instructions in the loop is small, the loop is referred to as being small, short, or tight. When a loop consists of many instructions, which may also include input/output operations, the loop is often referred to as long.

Types of loops

A part of a program may be repeated a number of times, thus creating a programmed loop. A programmed loop is often referred to as a processing loop. Sometimes a program error causes the CPU to repeat part of a program endlessly. Such a loop is never intended and requires debugging procedures to isolate the error.

Recognizing a loop

One or more of the following may indicate that a job/program is in an unintended loop:

- A steady glow of lights on the system Models 115, 125, and 158, control panel with the SYS indicator on, or for the System/370 one address will appear to remain displayed on the video display unit. (This depends on whether the loop is long or short.)
- A rhythmic pattern in the lights on the system control panel, or for the Models 115, 125, and 158, the word WAIT may flicker on the video display unit.
- A pointless recurrence of I/O (input/output) activity.
- A job (program) that does not change status for a long time (for example, an absence of I/O activity).

A note to the operator: When a loop is recognized, the operator must first try to establish whether the loop is unintended or has been programmed, before beginning with hands-on debugging.

If the programmer has not warned the operator about a programmed loop, or given a time estimate for the program, it will be very difficult to differentiate between an unintended loop and a programmed loop.

Even when time estimates are given, job or program time may increase because of any one or more of the following:

- Priority of the partition in which the job is running (multiprogramming system)
- CPU retry and error logging routines
- The use of slower speed input/output units than those for which the job was originally planned.

Causes of an unintended loop

- A coding or logic error in the program may cause an unintended loop.
- The operator may have set the job up incorrectly, thus causing the program to loop at some stage during execution.
- An input/output device malfunction.
- A JCL (job control language) error.

Operator action

If the operator is not sure whether the loop is unintended, the programmer must be contacted before any debugging procedures can begin. If this is not possible, the only action the operator can take is to let the job run on for a time, depending on system commitments, and to make notes of any further system activity. If the loop is programmed, no time would have been lost by allowing it to run on. In multiprogramming environments a loop in one partition will affect the run times of programs in other partitions.

Flowcharts in Section 3 will help the operator in gathering information at the time the error occurs, and Section 4 provides a guide for programmers in how to analyze this information.

WAIT STATES

Definition

There are occasions when an error in the program or the machine causes the system to stop. This means that no I/O activity is occurring and no instructions are being executed.

In this state the hardware circuitry turns on the WAIT indicator, or on the System/370 Models 115, 125, and 158, displays the word WAIT on the video display unit, and the system is said to be in a wait state.

Types of wait states

The impact of a wait state on system operation depends on the cause of the wait and the operator action required to recover from it. The following terminology is used for describing a wait state:

- Hard wait
- Soft wait
- Normal wait

Essentially, the difference between the first two waits is that the system recovery from a hard wait is impossible without executing a system IPL, whereas recovery from a soft wait may be accomplished without impairing program or system operation. The operator can easily determine the type of wait state by pressing the REQUEST key. If the wait is soft the following message may be issued:

AR 1I60A READY FOR COMMUNICATIONS

When the system is waiting for operator response to a message printed on the console printer or for an I/O device to be made ready by operator action, the wait state is sometimes referred to as normal.

Recognizing a Wait State

Any of the following observations confirm that the System is in a wait state:

- WAIT indicator remains on, or for a System/370 Models 115, 125, and 158, the word WAIT remains displayed on the video display unit.
- SYS indicator remains off (See Figure 1-2).
- No I/O device activity occurs.
- One or more SYSTEM CHECK indicators are on.
- A HARD MACHINE CHECK message is printed on the console printer.
- A HARD WAIT coded message in general register 11 (X'B').
- A HARD WAIT coded message in bytes 0-3 of low address storage.

Causes of a soft wait

A soft wait may be the result of an I/O operation performed on a malfunctioning device that is unable to complete an operation.

A system waiting for a magnetic tape unit to rewind a tape reel or for a disk unit to finish a seek before continuing a program, is in a temporary soft wait.

Recovery from a soft wait

If the system is in a soft wait, it is waiting for an interrupt to signal the completion of an event. Although the expected interrupt may be from the timer or external interrupt key, a missing "device-end" caused by hardware is the most frequent cause. The operator can make each device not-ready, then ready, to generate a device-end interrupt from each device. The system light flashes briefly as the supervisor examines and discards interrupts for which it was not waiting. The interrupt from the device for which the system is waiting causes normal processing to continue. (The occurrence should be brought to the attention of the customer engineer as a possible hardware failure.)

It may be possible to isolate the cause of the wait and take alternative action, such as using a different I/O device.

Recovery from a wait state becomes more important on large online multi-programming systems where to cancel programs or to re-IPL may be disastrous.

Causes of a hard wait

Hard waits can be caused by machine failure and programming errors. Possible programming errors that cause hard waits are:

- Supervisors errors as the result of a program check while in the supervisor state
- Coding errors in transient routines
- Incorrect use of transient routines.

Operator action

If the hard wait has been caused by a hard machine check shown by a message on SYSLOG and/or a coded message in bytes 0-3 of low address storage (see note), the operator must gather information from the system to help the IBM customer engineer locate the error.

If, however, there is no indication that the wait has been caused by a hard machine check, some information as to the cause of the wait can be obtained before retrying the job or starting a new one.

In any case certain initial checks must be made on the setup procedures for the job, the input media in use, and I/O devices in use.

Flowcharts in Section 3 will help operators in carrying out initial system checks and in gathering data about the wait state, and Section 4 provides a guide for programmers in how to analyze the data.

Note: For the Models 115 and 125 that are not supporting MCAR/CCH, a coded message is placed in GR11. A complete list of coded messages is given in Section 2.

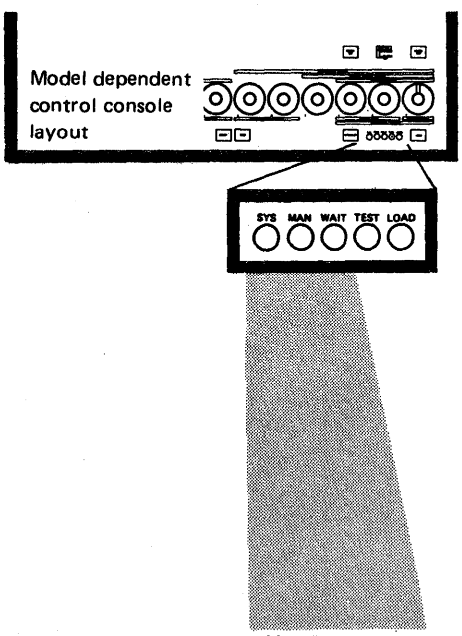
Types of Malfunctions

INDICATORS FOR LOOPS AND WAIT STATES

The SYS, MAN, and WAIT indicators show the CPU and I/O operating states as follows:

INDICATOR	FUNCTION
SYS *	The SYSTEM indicator is on when CPU operations are in progress and either use meter is running.
MAN *	The MANUAL indicator is on when the CPU clock is stopped or the system is in a stop state. All pending interrupts are handled. Manual store/display operations are possible only when the MANUAL indicator is on.
WAIT *	The WAIT indicator is on when the system is in a wait state (CPU clock running but no instruction processing taking place). If the wait is a soft wait state and an interrupt occurs, the CPU is taken out of wait state and processing started under control of the program being executed.

*** These indicators do not exist on System/370 Models 115, 125, and 158. Instead the corresponding words are displayed on the display unit.**



SYS	MAN	WAIT	CPU State	I/O State	
O	O	O	Abnormal condition	Abnormal condition	
O	O	●	Wait	Not working	WAIT STATE
O	●	O	Stopped	Not working	
O	●	●	Stopped/Wait	Not working	
●	O	O	Running	Undetermined	LOOP
●	O	●	Wait	Working	
●	O	●	Running	Working	LOOP (with I/O activity).
			Stopped/Wait	Working	

Legend

O - Off ● - On

Figure 1-2. System indicators.
Aids for recognizing a loop or a wait state.

Definition

Incorrect output can range from incorrect line spacing on the printed output from a line printer to incorrect results of calculation written on a disk file.

Recognizing incorrect output

Incorrect output may be detected by:

1. Operator
- Invalid messages — on console printer(s)
- Unidentified data — line printer(s)
- Duplication of data — display unit(s)
- Lack of activity on I/O devices assigned as output units
- Either more or less I/O activity than expected.

2. Programmer
If the execution of a program has been apparently successful, incorrect results will not be detected until the data is used at some future time.
Incorrect output can be categorized as:
- Missing records
- Duplicate records
- Invalid data that has sequence errors, incorrect values, format errors, or meaningless information.

Causes of incorrect output

As well as errors in the program logic, mistakes in setting up the system for the program will cause errors in the output. For example, use of incorrect data for input files, mistakes in device assignments, and incorrect job control statements and commands in the job stream will cause unexpected output.

Operator action

If the programmer cannot be contacted, the operator must save the output (whatever it is) or make a note of system activity before cancelling the job, or both.

The work files and input data should be given to the programmer together with any dumps executed. It may also be necessary to re-submit the job and trace the logic flow by using the SDAID BR and/or IF trace.

Flowcharts in Section 3 and 4 indicate the serviceability aid to use for isolating the cause of this type of system malfunction.

INTERMITTENT ERRORS

Definition

An error which occurs once and then seems not to recur for some time, is said to be intermittent. The frequency of the error may be a fraction of a second in the case of a high-speed computer like the System/370, or a week, a month, a year, or even longer. Intermittent errors can be caused by hardware failures or by programming errors.

Hardware failures

IBM provides serviceability aids that record and analyze hardware failures and attempt to recover from them. The routines that perform these functions are collectively termed RMS (Recovery Management Support). If online recovery is impossible, the system may be placed in a hard wait state. A message is issued to the system operator to run either the SEREP or EREP program. The output obtained from either of these programs is a listing of the statistical data accumulated up to and including the time of the error. This information serves not only as an aid in diagnosing machine errors, but also helps IBM customer engineers to increase the Reliability, Availability, and Serviceability (RAS) of the system.

RMS does not affect system operation, except for the time required to record the failure and issue an informatory message on SYSLOG.

- *Note: By use of the MODE command the recording and printing of soft machine checks can be suppressed. (This is not applicable to the Models 115 and 125).*

If the retry of an error is not successful and the severity of the error prevents system operation, the machine attempts to issue the message

0T11W HARD WAIT CODE = X (where X is an alpha character A thru I)

and the system is placed in a hard wait state. Diagnosing this condition is described in Sections 2 and 3.

On the System/370 Models 115 and 125 statistical data about the hardware is recorded on the DISKETTE by micro-program. The recorded data on the DISKETTE can be displayed on the video display unit by selecting one of the LOG ANALYSIS displays. This is described more fully in Section 2 – F in this manual. The information displayed supplements the EREP/SEREP program output that may also be required, depending on the type of I/O units attached to the systems. If a hard wait occurs with no message on the console printer, there may be a message in "low address storage" that will indicate an operator action. Low address storage and its meaning is fully discussed in Section 2.

Intermittent programming errors

After writing a program, it is in most cases quite impossible to test it under all combinations of circumstances that may occur during its use. Therefore, programs may contain coding errors that become evident only under particular circumstances, even after years of error-free use.

Since the error does not occur every time the program is executed, and the EREP printout or Log Analysis display indicates no hardware failures, this type of system malfunction is regarded as an intermittent software error.

Such an error can be caused by a combination of the following:

- A change in the input data (a new card deck)
- Poor quality input media (cards, tape, data transmission)
- An existing coding error in a routine that is not normally executed
- A change of routines called by the supervisor
- The use of a new software routine
- New operating procedures
- Changes in the job control language.

An error of this type is difficult to isolate, and requires the use of special debugging techniques.

PROGRAM CHECK INTERRUPT

Definition

There are three types of program check interrupts:

1. A page translation exception. This occurs when an instruction or data is not in the real address area. A page from the page data set must be 'paged in' to the real address area before the program can continue. This is not an error condition.
2. Program check interrupt resulting from the use of the MC (monitor call) instruction. This is not an error condition.
3. Program check interrupt resulting from incorrect specification or use of an instruction or data by the problem program. This is an error condition, and is always reported by a message issued on SYSLOG at the time of the program check as shown below:

```
BG 0S03I PROGRAM CHECK INTERRUPTION – HEX LOCATION 0406E0 –
        CONDITION CODE 3 – DATA EXCEPTION
   0S00I JOB DEBUGEXS CANCELED
```

The program is automatically cancelled by the supervisor and depending on the use of the job control statement // OPTION DUMP, or the DUMP option being supported by the supervisor, a dump of the supervisor and of the partition owning the program is executed. This automatic program cancellation is termed abnormal EOJ (end of job), or program abnormal end. The program check message gives the location of the failing operation and the condition code. This gives the programmer a starting point for offline program debugging.

Causes of a program check

The most probable cause is improper specification or incorrect use of instructions or data in the program.

Program checks occur most frequently during program testing, because of incorrect coding or errors in the program logic.

Operator action

No action can be taken by the operator other than saving for the programmer the console printer log sheet, the dump (if executed), job stream, and any input data files used by the failing program. Flowcharts in Section 4 will help the programmer to analyze the information and isolate the error.

Types of Malfunctions

I/O DEVICE MALFUNCTIONS

A device malfunction either will be seen immediately as an incorrect physical operation, or will cause the system to enter a wait state, loop, or produce incorrect output as already discussed. Normally an error message will be issued on SYSLOG.

Examples of device malfunctions

Some obvious device malfunctions are:

- Mechanical noises not normally present
- Lamps either on or off which the operator recognizes as not normal conditions
- A lack of movement of input/output media which the operator knows to be incorrect at the time
- "Tape Runaway," a special type of error that occurs on magnetic tape drive unit (A mounted tape winds forwards at a higher speed than normal.)
- Incorrect "form skipping" on the line printer.

Operator action

If there is no obvious action that can be taken such as pressing the device STOP and/or OFF buttons, consult the device component manual before informing your IBM customer engineer (unless the nature of the malfunction constitutes a danger to human lives and equipment).

When a system malfunction is recognized it is important that the operator obtain information from the system. The information helps the programmer and the IBM customer engineer during offline program debugging. Whatever the system malfunction, the operator must always save error messages issued on SYSLOG and/or on SYSLST, and in some cases save the I/O media (card files).

The operator can obtain information by doing one or more of the following:

- Issue the MAP command.
- Make a note of system activity.
- Display low address storage, the current PSW, the control registers and general registers.
- Execute a storage dump.
- Take a trace of a loop.
- On the Models 135, 145, 155-11, and 158, initiate the EREP or SEREP programs
- On the Models 115 and 125 on the advice of the IBM CE use the Log Analysis to display hardware errors recorded on the DISKETTE, and on the Model 158 use the display frames.

Many factors must be considered when gathering information, and Section 3 and 4 cover this subject in detail.

LOW ADDRESS STORAGE

This area of low real storage (as defined in the *Introduction to DOS/VS*) is one of the important sources of system information used to aid offline program debugging. The contents of the low address storage can be dumped (printed out) or displayed by using job control commands or console aids. Details about the format and contents of the low address storage are given in Section 2-E of this manual. Figure 1-3 illustrates the location of low address storage in relation to other areas in virtual storage.

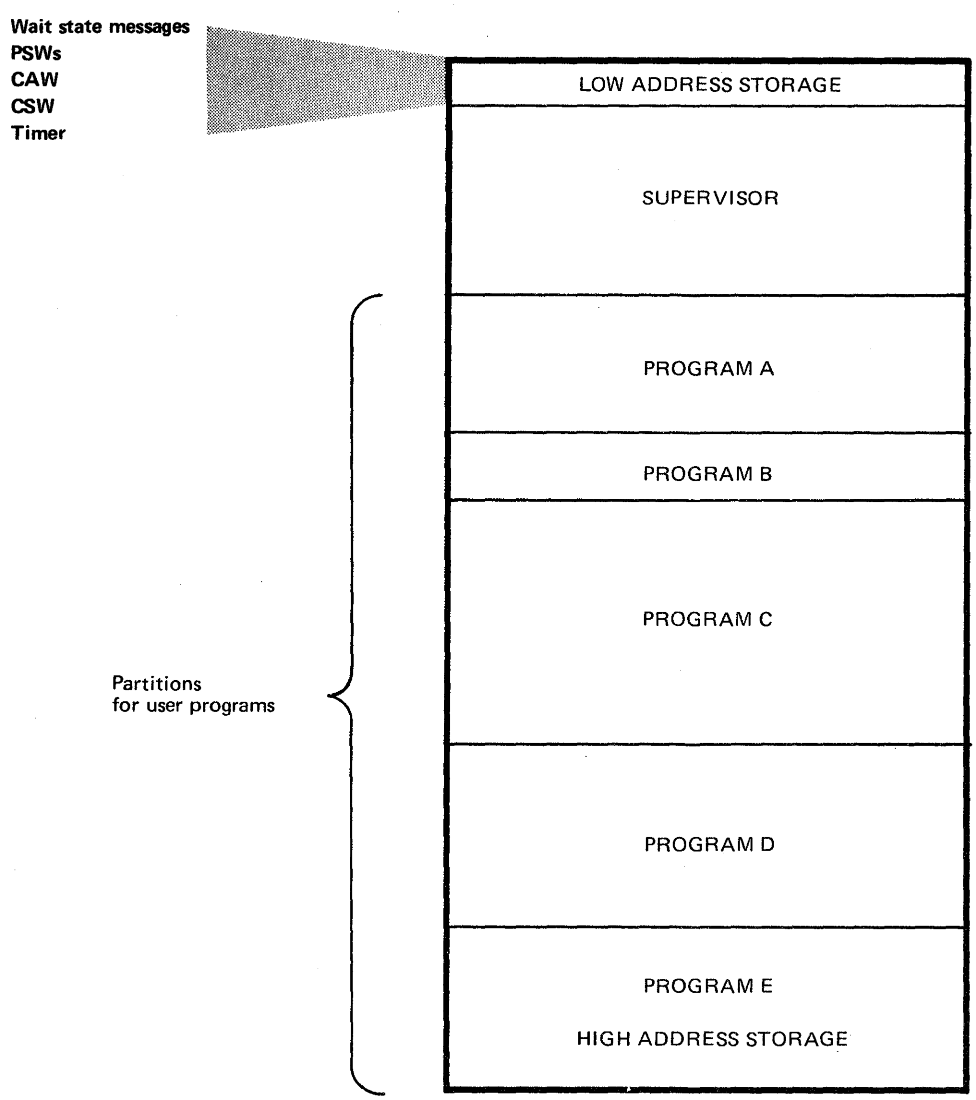


Figure 1-3. Relative location of low address storage.
Low address storage contains information to aid offline debugging.
(Size relationships in this figure are purely illustrative.)

STORAGE DUMPS

A dump is a program or an operation that prints the image, in hexadecimal format, of a selected area of virtual storage. This term is also used when an area of virtual storage is recorded or stored on magnetic tape of disk pack.

Figure 1-4 illustrates the various type of dumps offered by IBM. Section 2-A of this manual describes how to execute the dump programs and operations, and discusses the meaning of dump output that is useful during offline program debugging.

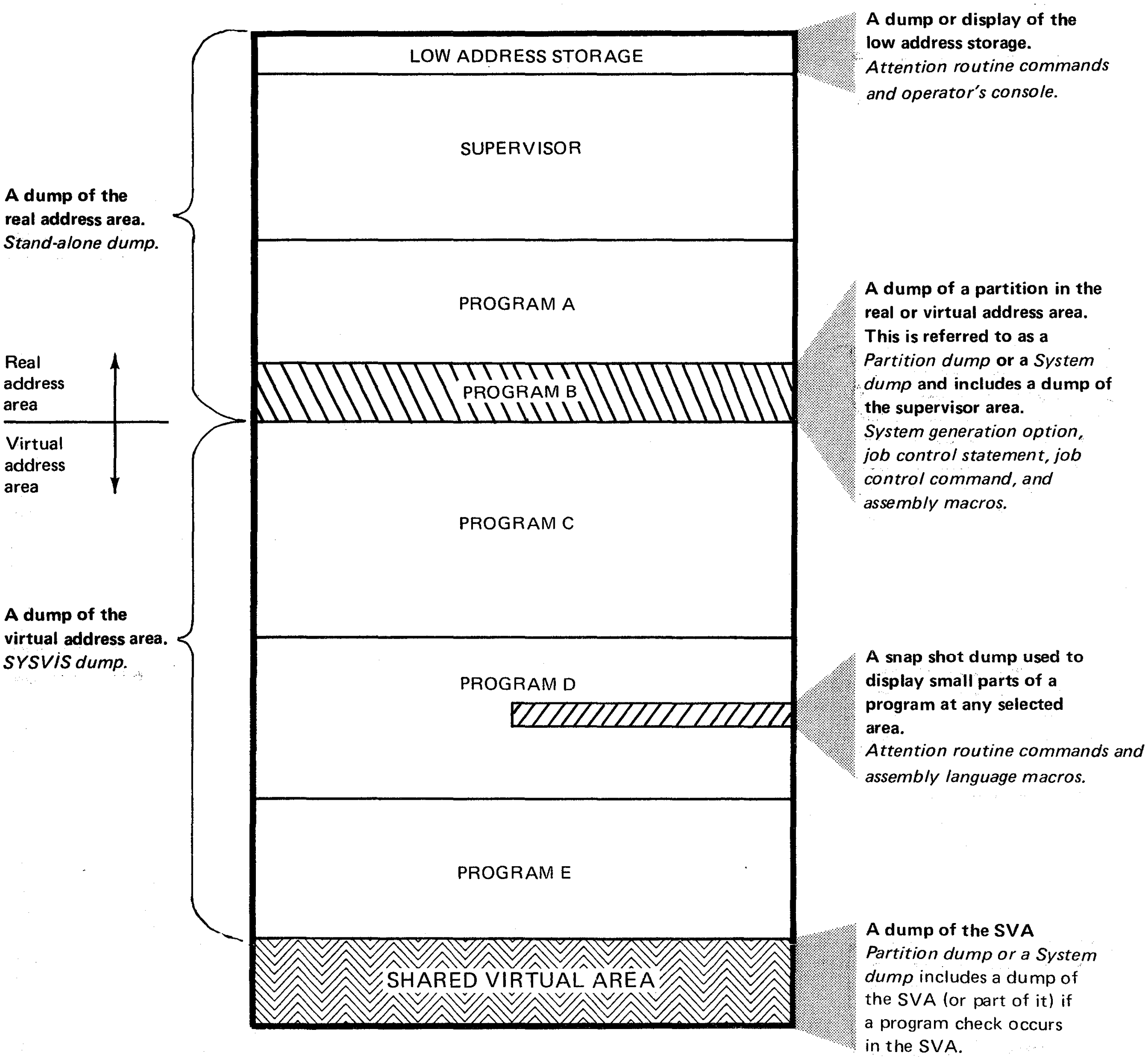


Figure 1-4. Storage dumps.
Various areas of storage can be dumped or displayed using the IBM dump programs and console aids.

(The dividing point between real and virtual address areas depends on the size of the hardware memory on your System/370).

LOOP TRACING

Three methods of tracing or recording the path of a loop are provided on the System/370:

1. By using the facilities provided by the operator's console, the operator can list the addresses of the instructions used by the loop.
2. By using the successful branch routine of the SDAIDS.
3. By using the instruction fetch trace of the SDAIDS.

All three methods are described in Section 2. The first method is useful to trace small loops during hands-on debugging. However, the amount of time that may be spent tracing a loop by this method depends on the answer to the following:

1. How important is it to system operation that the loop be fully traced?
2. How will the time spent tracing the loop affect system commitments?

Normally the operator is not in a position to answer these questions and if the programmer or the DP manager is not available, he can only take a short trace. The second and third methods can be used either during hands-on debugging, or during re-runs of the program generating the loop.

A note to operators

Before tracing a loop by using any of the above methods, you must consider their effects on time-dependent programs currently running in the system. Such programs are, for example, those using magnetic ink character recognition or teleprocessing equipment as input/output devices.

Guidelines on how to isolate an unintended loop and trace it are given in flowcharts in Sections 2 and 3.

RMS (Recovery Management Support)

The functions employed in recording a hardware error and recovering from it are collectively termed RMS (Recovery Management Support). RMS was introduced under "Hardware failures" in this Section. RMS software routines record hardware failures on the system recorder file, located on SYSREC (SYSREC can be either an area on SYSRES, or an individual disk pack.)

For the System/370 Models 115 and 125, errors in the CPU and natively attached input/output devices (except tape units and teleprocessing terminals), are recorded on the DISKETTE. Recording is performed by microprograms and is independent of the RMS software routines.

Figure 1-5 contains an overview of RMS, which is a part of the total RAS (Reliability, Availability, and Serviceability) concept. RMS uses a monitor and several transient routines that check the severity of the error, record it (if possible), and print informatory messages.

Using an IBM program called EREP (Environmental Recording, Editing, and Printing) the data on the recorder file can be printed on a line printer. This data is used to investigate the nature and cause of a system malfunction. For the Models 115 and 125, information will be printed by EREP only if the system supports RMS. (Refer to Section 2-F for details.)

If the severity of a hardware error is such that EREP can not be executed, the IBM-supplied program SEREP must be executed. SEREP is a stand-alone version of EREP that formats and prints the data held in the logout areas of real storage.

On the System/370 Models 115 and 125, the LOG ANALYSIS displays hardware statistical data recorded on the DISKETTE. This is additional to the EREP program that can be executed after using the log analysis displays.

A similar facility is provided on the Model 158 in the form display frames obtained by the use of the SERVICE function.

How to execute EREP and SEREP, and how to use the log analysis displays and display frames feature is described in Section 2. The components of RMS are fully described in Section 2-F.

HARDWARE ERROR RECORDING AND RECOVERY

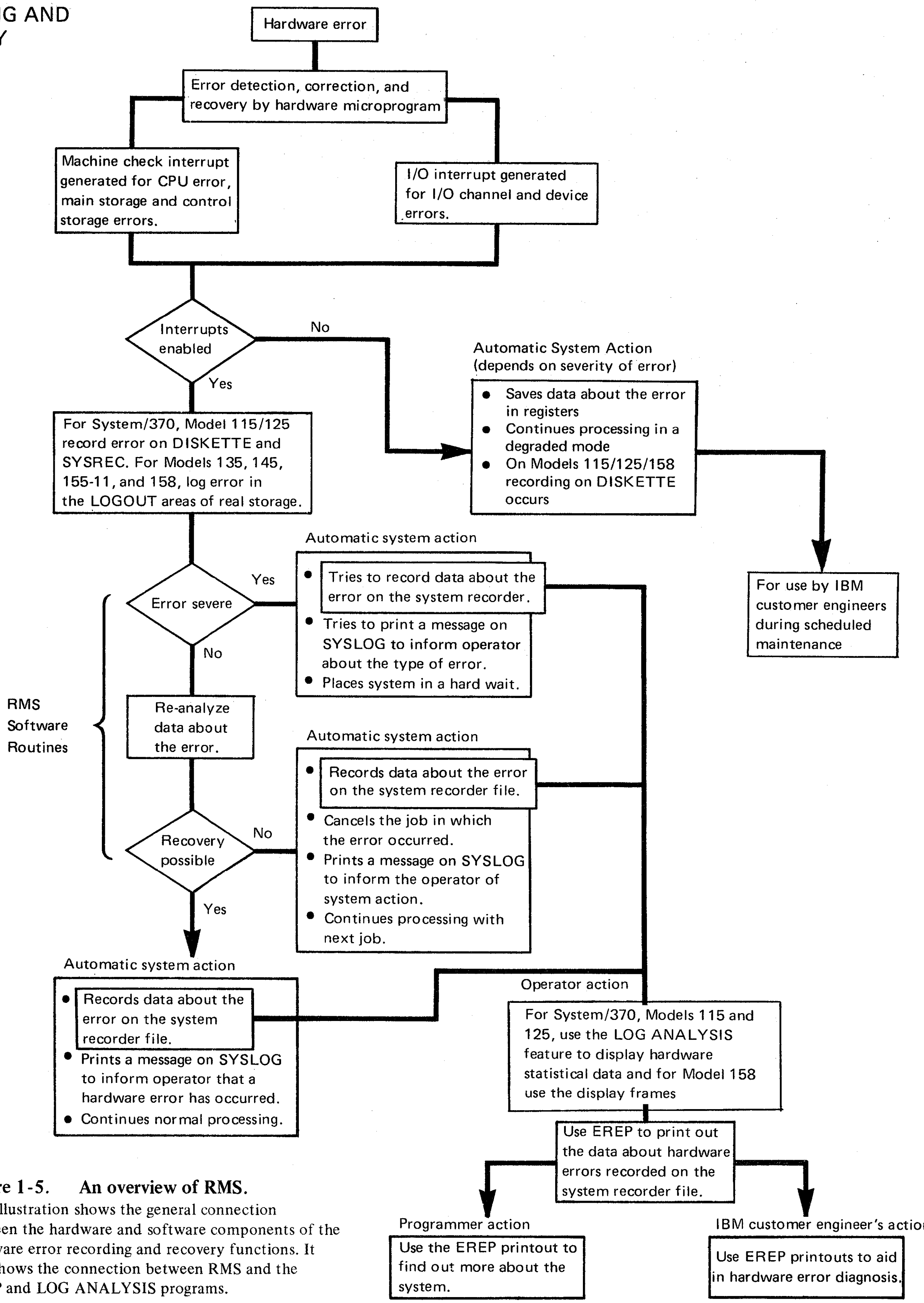


Figure 1-5. An overview of RMS.
This illustration shows the general connection between the hardware and software components of the hardware error recording and recovery functions. It also shows the connection between RMS and the EREP and LOG ANALYSIS programs.

EREP (Environmental Recording, Editing, and Printing)

This program edits and prints information about hardware failures that are recorded on the system recorder file (SYSREC).

There are several options of EREP that enable the operator to select SYSREC records for editing and printing. These options are detailed in Section 2. By using the EREP program output, the IBM customer engineer can recognize hardware failures. During scheduled maintenance periods he can then perform preventive maintenance on the parts of the system causing hardware failures.

Because the EREP program can be initiated by the operator, it is a useful aid for gathering data about the condition of the hardware at any time during system operation.

Some messages issued on SYSLOG tell the operator when to execute EREP.
For example:

```
0T11W HARD WAIT CODE = D
      RUN EREP RECORDING SUCCESSFUL
```

Other occasions when EREP should be executed are indicated in *DOS/VS Messages.*
For example:

```
0T05E ERROR ON RECORDER FILE – RUN EREP
```

Operator action:
Schedule the EREP program to display the information on SYSREC.

Either the operator action listed under the appropriate message will indicate the EREP option to select, or your IBM customer engineer will advise you on the option to select.

Flowcharts in Sections 3 and 4 also indicate when to execute EREP.

HARDWARE ERROR RECORDING AND RECOVERY

SEREP (Stand-alone EREP)

This is a stand-alone program that edits and prints hardware failure data either stored in the logout area of real storage or, for the Model 158, recorded on the log recording console file.

SEREP provides a means of printing system status information stored in the real storage logout areas at the time of the machine malfunction. The SEREP printout is analyzed by your IBM customer engineer.

For the Models 135, 145, and 155-11, SEREP is initiated using the standard IPL procedure. The SEREP program consists of a card deck and must be executed when the message issued on SYSLOG indicates "RUN SEREP."
For example:

```
0T11W HARD WAIT CODE = H
        RUN SEREP RECORDING UNSUCCESSFUL
```

For the model 158 SEREP is contained on the Log Recording Console File which is loaded by using the Service and Index frames.

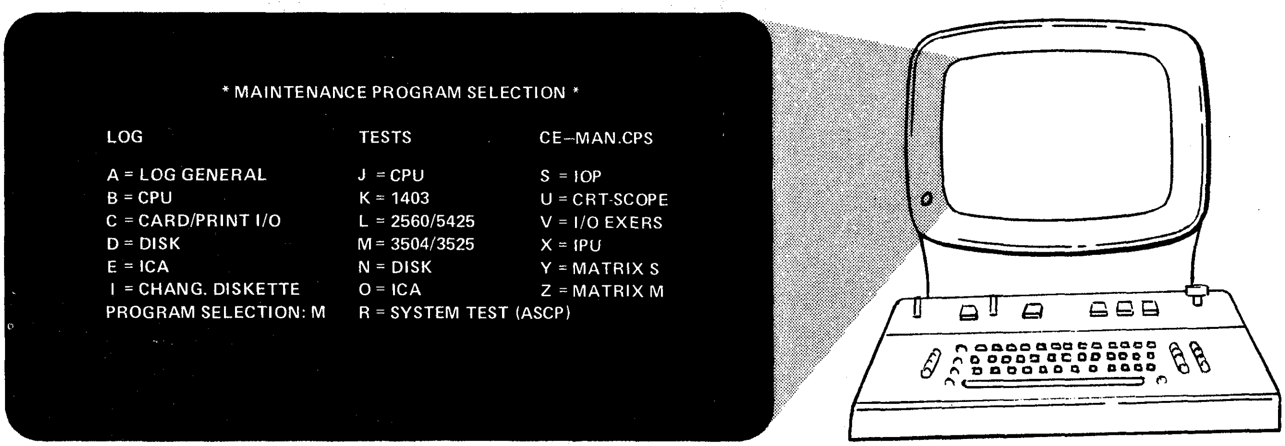
If a hard wait condition occurs and no message is printed, a wait message in low address storage will inform the operator if SEREP is to be initiated.

Flowcharts in Sections 2 and 3 indicate how and when to use SEREP.

Log Analysis Displays (Models 115 and 125)

This aid, provided for IBM CE use, enables the condition of the internal hardware to be displayed, and, if required for offline analysis, to be printed on the 5213 printer, if attached. On advice from the CE, an operator is able to obtain "hard copies" of the displays if a hardware error is the cause of a system malfunction.

Maintenance (M)
When the mode selection display is on the video screen and the operator enters selector character 'M' against 'Mode Specification,' the screen displays the maintenance repertoire. This repertoire consists of log analysis, micro tests, and CE manual operations, as shown in the figure below. The cursor is positioned next to the preselected 'M' so that any one of the maintenance modes can be selected.



Note: E = ICA is displayed only when the system supports the Integrated Communications Adapter

Figure 1-6. Model 125 maintenance program selection display.
On the Model 115 the entry M = 3540/3525 and entry K = 3203/5203

Log Analysis (A-E)
When a parameter 'A' through 'E' is entered into the maintenance display, log information is brought to the screen. Entering 'A' for instance, causes a display of general log information that informs the operator if any logging occurred, and if so, which part of the system caused it. From this report, the operator can select a detailed log by keying in one of the four characters 'B' through 'E'. For example, 'B' provides log information for the CPU.

A "hard copy" printout of the displayed information can be obtained and saved for your IBM customer engineer by pressing the copy key, if a 5213 matrix printer is attached to your system.

Further details are given in Section 2, F-5.

HARDWARE ERROR RECORDING AND RECOVERY

Display Frames (Model 158 only)

A serviceability aid provided on the Model 158 allows the operator to display and obtain "hard copies" (on the 3213 printer) of the condition of the hardware.

The information displayed or printed is used by the IBM CE to diagnose the cause of a permanent hardware error. A hardware error of this type will be recognised by the operator by an error message displayed on the program frame, for example the words STOR CHECK displayed in the lower right corner of the frames.

Having recognized the existence of a hardware error, the operator can either inform the IBM CE immediately, or can "look at the hardware" by scanning the information on the display frames and obtaining a "hard copy" of them if desired.

The type of information displayed is listed under numbers 6 to 26 in the INDEX frame shown below.

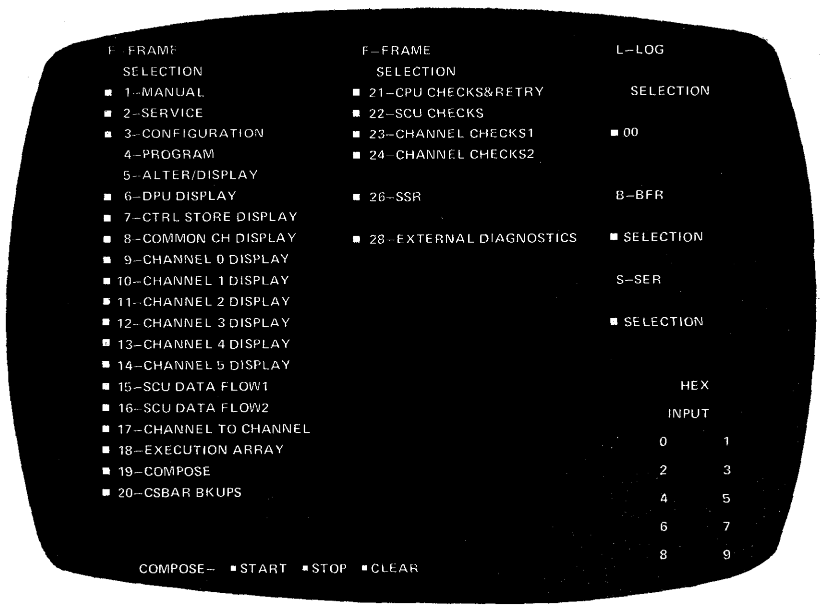


Figure 1-7. The Model 158 Index Frame

This frame is obtained via the manual and service frames. With this additional information about the hardware, the IBM CE will be able to advise on further system operation. How to use this serviceability aid is described in Section 2-F.

If the error cannot be isolated by analyzing the information obtained from the procedures already mentioned, other programs and tools must be employed during program reruns.

Trace routines

The trace routines supplied by IBM are special programs that "look inside" the central processing unit during system operation.

Traces of program execution are especially valuable on the larger multiprogramming systems.

Traces can indicate the phases used, the supervisor calls required, the types of interrupts encountered, and the I/O activity during program execution. Trace routines can also indicate paging activity and successful branching, and produce a printout of instructions fetched, and storage and general register alterations, during program execution.

Supervisor communication macros

Certain DOS/VS macros can be written into programs to provide more information about the state of the system at the time of an error. One such macro is PDUMP, which will give a dump of any specified real or virtual address area.

Operator's console

A useful tool for hands-on debugging is the operator's console. This is used for "tracing a loop" and displaying or altering registers and small areas of storage.

Library and Service programs

DOS/VS library and service programs are useful when information is required about previously written programs that are used by problem programs. These DOS/VS programs will list volume directories, print listings of the programs contained in the libraries, and display file label information. Such information is required, for instance if a particular phase on a private core image library causes incorrect results of calculations when used by one of the tested problem programs.

VSAM Programming Aids

VSAM (Virtual Storage Access Method) provides aids that print, list, and verify data recorded on VSAM files. Assembler macro instructions for VSAM are also provided to allow the programmer to obtain information about I/O operations (OPEN, GET, PUT, CLOSE) during execution of VSAM programs.

Analyzing the Information

The analysis to be made depends on when the job failed and how much pertinent information the operator obtained from the system at the time of failure. It also depends a great deal on the system environment. The first step is to examine messages printed on the console log sheet, and to look for any messages on the output printer.

The next step is to examine any other printed output, for instance, program output and storage dumps.

For a successful analysis, the programmer, who should be familiar with the program, will require:

- The program listings
- The linkage editor map
- Flowcharts of the failing program.

In the more difficult cases of program errors, the programmer will also require:

- The supervisor listing
- Output of the trace routines
- Dumps of the data input files
- The input media
- Listings (or displays) of file label information
- Listings (or displays) of the libraries.

If the program failure was caused by a hard wait, the programmer should scan the EREP printout (if one was obtained) to eliminate any possibility of a hardware failure, and examine the stand-alone dump.

Section 4 describes in detail how to use the above listed output during offline debugging.

SERVICEABILITY AIDS

Section 2

SERVICEABILITY AIDS

How to use this section.

Familiarize yourself with the contents of this section, which gives details about the operation and execution of the serviceability aids offered by IBM.

The reference chart shown on the opposite page lists the aids described in this section in groups according to type.

All right-hand pages in this section have running-tabs numbered to correspond to this chart. The chart helps you to locate details about any aid described in this section.

For example:
SEREP is described on page marked F – 4

Another example:
When a dump of the low address storage is required in an operators flowchart of Section 2 and 3 it is indicated as:

Dump, or display and note the
- Current PSW
- 16 control registers
- 16 general registers
- Low address storage

See D-1-F for Models 135, 145, 155-11
and D-2-F for Models 115, 125
and D-3-F for Model 158

D-I-F, D-2-F and D-3-F refer to tab numbers that identify unique pages in this section.

In this example it guides the operator to the flowchart procedure for dumping low address storage.

Table referencing in this Section

Illustrations in this Section do not have figure numbers, but are referenced by the text. For example, "the next illustration shows. . . . ." or "the figure on the opposite page shows the . . . .".

However, because tables in this section are often referenced from other parts of this manual the tab referencing system is used for the Tables. For example: A table in sub-section B-6 is given the reference of B-6. If there is more than one table in any sub-section the first table is given the reference B-6-A and the second table B-6-B, and so on.

SERVICEABILITY AIDS

A — Dumps of, and changes to the real and virtual address areas

1 Operator commands ALTER DISPLAY DUMP
2 System dump
3 DUMPGEN and the stand-alone dump program
4 Transient dump
5 Supervisor communication macros

B — Trace routines

1 PART1 PDAIDS Introduction, system requirements, and terminating
2 PDAIDS Description, operation, and examples
3 The PD area
4 Initializing PDAIDS
5 PDAID Job stream examples.
6 PART 2 SDAIDS Introduction, System requirements, terminating, and output information
7 SDAIDS Description and operation
8 SDAIDS Stop and dump routines
9 The SD area
10 Initializing SDAIDS
11 Examples of SDAID job entry and output

C — Library display programs and utilities

1 LSERV
2 Library display programs
3 ESERV
4 VTOC display program
5 Reserved for future use
6 SYSVIS dump Utility

D — Hardware aids through the operator's console

1 ALTER/DISPLAY Models 135, 145 and 155-11
2 ALTER/DISPLAY Models 115 and 125
3 ALTER/DISPLAY Model 158
4 Instruction stepping Models 135, 145 and 155-11
5 Instruction stepping Models 115 and 125
6 Instruction stepping Model 158
7 Stop on address Models 135, 145 and 155-11
8 Stop on address Models 115 and 125
9 Stop on address Model 158
10 Models 115 and 125 Console dump operation
11 Store status and clear real storage (all Models), and Save Usage Counters (Models 115 and 125 only)

E — Other aids

Job control commands and statements

Low address storage

Wait state messages

Linkage Editor Map

F — Hardware error recovery and recording

1 Recovery Management Support (A general description)
2 Recovery Management Support (A detailed description)
3 EREP
4 SEREP
5 Maintenance Log Analysis Models 115 and 125
6 Display frames (Model 158 only)
7 OLTEP

Intentionally Blank

CONTENTS

Dumps of, and changes to Real and Virtual Address Areas.

OPERATOR COMMANDS (ALTER)

The ALTER Command

To activate the ALTER command press the REQUEST key and enter ALTER. The command is used to alter from 1 to 16 bytes of virtual storage starting at the specified address.

Operation	Operand
ALTER	xxxxxx

The operand xxxxxx is a six-digit hexadecimal address. Six digits must be entered regardless of the size of the address; addresses of less than six digits must be preceded by zeros.

After the command has been entered and the END key pressed, the hexadecimal representation of the information to be placed in storage should be entered. Two hexadecimal characters (0 to F) must be entered for each byte to be changed. If an odd number of characters is entered, the last character is ignored and its associated byte is unaltered.

Examples are shown below.

```
AR  1I60A  READY FOR COMMUNICATIONS.
AR  alter 040820
AR  00622000
AR  alter 000430
AR  1I42D  ADDRESS WITHIN SUPERVISOR OR SVA
AR
AR  1I60A  READY FOR COMMUNICATIONS.
AR  alter 0404e0
AR  8a800004
AR
```
Ex1 KT

Restrictions

1. If the bytes to be altered cross the boundary from a valid to an invalid address space (see the third restriction, below), only the bytes in the valid address space are changed, and the following message is issued on SYSLOG:

    1147I XX BYTES COULD ONLY BE ALTERED

2. If the highest available virtual storage address is exceeded before sixteen bytes are printed, the command is terminated and no alteration can occur.

3. If the specified address is within an invalid address space, message

    1141D INVALID ADDRESS

   is issued on SYSLOG.

   An INVALID ADDRESS is one of the following:

   - The address of a location in the gap between real and virtual address areas.
   - The address of a location beyond the end of virtual storage.
   - The address of a location in the page pool.
   - The address of a location in a virtual partition whose real partition contains a program running in real mode.

4. Altering the Supervisor area or SVA

   If the address entered falls within the supervisor area or within the shared virtual area (SVA), a warning message is issued on SYSLOG:

    1142D ADDRESS WITHIN SUPERVISOR OR SVA

   To respond to this message, press END/ENTER to terminate the ALTER command or reply with IGNORE to allow alteration.

When to use

This aid is primarily a hands-on debugging aid. The programmer can use it in conjunction with program listings to modify any part of the programs presently running in virtual storage. This enables immediate checks on results of program changes during execution of the program.

OPERATOR COMMANDS (DSPLY)

The DISPLAY Command

To activate the DSPLY command press the REQUEST key and enter DSPLY. The command allows the operator to display on the console printer keyboard 16 bytes of virtual storage starting at the specified hexadecimal address. Two hexadecimal characters (0 to F) are printed for each byte of information; these characters represent the hexadecimal equivalent of the current information in the virtual storage.

Operation	Operand
DSPLY	xxxxxx

The operand xxxxxx is a six-digit hexadecimal address. Six digits must be entered regardless of the size of the address; addresses of less than six digits must be preceded by zeros.

After the command is entered and the END/ENTER key is pressed, the hexadecimal representation (two characters for each byte) of sixteen bytes of virtual storage will be printed.

Examples are shown below.

```
AR   1I60A  READY FOR COMMUNICATIONS.
AR   dsply 00052d                                  16 bytes from
AR     C00340 40404040 40404000 40404040 40           52D
AR   dsply 040820
AR   00622000 0C000103 00040858 00040860
AR   dsply 03fffe
AR   1I41D  INVALID ADDRESS
AR   dsply 07afff
AR          00 00000000 00000000 00000000 000000
AR
                                                        (Ex2K7)
```

Restrictions

1. If the sixteen bytes cross the boundary from a valid to an invalid address space (see the third restriction, below) only the bytes in the valid address space are displayed, and the following message is issued on SYSLOG:

   1148I XX BYTES COULD ONLY BE DISPLAYED

2. If the highest available virtual storage address is exceeded before sixteen bytes are printed, the command is terminated. However, the contents of those bytes that fall within the virtual address area are printed.

3. If the specified address is within an invalid address space the following message is issued on SYSLOG:

   1141D INVALID ADDRESS

   The definition of invalid address space is listed under item three of "Restrictions" in the description of the ALTER command.

When to use

This aid can be used during hands-on debugging, or an operator can be instructed to use it at specific addresses in a program.

For instance, loop count areas, small areas modified by loops, or parts of I/O areas can be dumped or displayed during program execution. The dump information will help during offline program debugging.

OPERATOR COMMANDS (DUMP)

The DUMP Command

To activate the DUMP command press the REQUEST key and enter DUMP. The command allows the operator to display large areas of virtual storage on SYSLST. The SYSLST used may be assigned to any partition, but it must be a printer and it should not be in use by the partition. If the same printer is being used by the partition, the printed output will be a mixture of dump and partition output.

Operation	Operand
DUMP	[S / BG / Fn / BGS / FnS / PDAREA / xxxxxx, xxxxxx] [(BG) / (Fna')] * Note 1

n,n' = 1,2,3,4

Note:

If the first operand is omitted, the general registers, control registers, and all storage that is currently used by programs, except that used by the supervisor (unless the operand BGS or FnS is specified), will be dumped. See note 2. The storage used consists of:

1. *Real storage not belonging to the page-pool*
2. *The virtual partitions in which a program is currently running.*

Description of the operands:

Operand	**Meaning**
S	Causes a dump of storage used and the supervisor area. See note 2
[BG / Fn]	Causes a dump of the specified partition and its associated registers. If a real-mode program is running in the specified partition, the temporary real partition is dumped. If a virtual-mode program is executed in the specified partition, the whole virtual partition is dumped.
[BGS / FnS]	Causes a dump of the same areas as described for the BG/Fn operand. However, the dump will include the supervisor area.
PDAREA	The PD area and the registers will be dumped (See Section 2, B-3 for details and a description of the PD area.)
XXXXXX, XXXXXX	Specifies the starting and ending address of virtual storage, with associated registers, that is to be printed. If the starting address is not on a fullword boundary, the address is rounded down to the first fullword boundary; if the ending address is not on a fullword boundary, the address is rounded up to the first fullword boundary. A minimum of one fullword is dumped, beginning at the start address.

Note 1:

BG, F1, F2, F3, F4: *When any of these additional operands are specified, the area of virtual storage specified by the first operand is dumped on the SYSLST assigned to the partition specified by this operand. SYSLST must be a printer and should not be in use by its assigned partition. If the same printer is being used by the partition, the printed output will be a mixture of dump and partition output.*
(If this operand is not specified, the SYSLST printer assigned to BG is used. See note 2.)

Note 2:

An indication is given on the dump output whether the dump includes storage areas considered to be invalid address spaces.
The definition of invalid address spaces is listed under item three of Restrictions in the description of the ALTER command.

When to use

This command is useful in circumstances similar to those described for the DSPLY command: to obtain information about I/O areas, or areas modified by loops or transients during program execution. The only difference between this command and DSPLY is in the size of the area that can be dumped.

Note: Logical transient routines can not be checked because the LTA is used by the DUMP transient. The information in the LTA is therefore overwritten by the DUMP routine.

An example is shown below of a dump of the PD area using the DUMP command, when SYSLST is assigned to a line printer.

```
CR 0-7   804000E0 0000F600 FFFFFFFF FFFFFFFF  00000000 00000000 00000000 00000000
CR 8-F   0000E000 00000000 00000000 00000000  00000000 00000000 C2000000 00000200

PDAREA

009480   E3D37EF3 F8F07000 000099FF 0000479C  0000163E 0000844E 00000C56 00009F02   TL.380.......... .......+........
0094A0   00000000 00004798 00003AA4 00001000  D7C4C1C9 C4C9E3E6 FFFF0009 FFFFFFFF   ................ PDAIDITW........
0094C0   FFFFFFFF FFFFFFFF FFFFFFFF 3FDEFFFF  FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   ................ ................
0094E0   FFFFFF80 800047AE 47F0915C 58B091B8  D20FB002 0038D201 B00000BA 50A0905C   .........0.*.... K.....K......#..*
009500   50C091C4 58A00014 48A0A040 492A0000  4780909C 95FFA000 4780910C 41AA0008   #..D....... .... ................
009520   47F09084 95FFA002 4780910C 1BCC43CA  000239C0 000358A0 00805AC0 A024581C   .0.............. ................
009540   00004111 000058C0 91C49120 100647E0  910C50C0 91C41BAA BFA70041 47809110   .........0...... ..#..D..........
009560   58B10020 19AB4740 90FE41CB 004819AC  472090FE 1BAB5AAB 003C58B0 9188BEA7   ....... ....... ................
009580   B00347F0 910CBFB7 B0454770 900CD702  B00BB008 58C091C4 58A0905C 49209036   ...0...........P. ........D...*....
0095A0   078A070A 49209038 078A070A 4920903A  078A58B0 008058B0 B01C0502 B00391C0   ................ ..........N.....
0095C0   078A0501 B00691C8 078A58B0 918841BB  00125980 918C4740 915658B0 91845080   ..N....H........ ....... .......#.
0095E0   918807FA 58B091B8 44009068 502B0002  501B0006 D201B000 918050A0 905C88A0   .............#... #...K......#..*..
009600   001842A3 00029430 B0029120 10064710  919C9130 B0024770 9110D702 B00BB00B   ................ ...........P.....
009620   47F09110 58A1001C 50A80005 91300302  47809192 47F090CA E2400000 00009654   .0.......#....... ......0..S ......
009640   00009666 000099FC C4E4D400 00002000  D5E7FFFF 000E070C 20000000 09620000   .........DUM..... NX..............
009660   8AF00800 0000000E 070F2000 00000962  00000000 04000000 E2400000 000EFF00   .0.............. .........S ......
009680   8AF00000 00000400 0000000E 070C2000  00000962 00008AF0 08000000 000E070F   .0.............. ........0........
0096A0   20000000 09620000 00000400 0000E240  0000000E FF008AF0 00000000 04000000   ...............S ........0........
0096C0   000E070C 20000000 09620000 8AF00800  0000000E 070F2000 00000962 00000000   .............0.. ................
0096E0   04000000 E2400000 000EFF00 8AF00000  00000400 0000000E 070C2000 00000962   ....S ........0.. ................
009700   00008AF0 08000000 000E070F 20000000  09620000 00000400 0000E240 0000000E   ...0............ ...........S ....
009720   FF008AF0 00000000 04000000 000E070C  20000000 09620000 8AF00800 0000000E   ...0............ ..........0......
009740   070F2000 00000962 00000000 04000000  E2400000 000EFF00 8AF00000 00000400   ................ S ........0......
009760   0000000E 070C2000 00000962 00008AF0  08000000 000E070F 20000000 09620000   ...............0 ................
009780   00000400 0000E240 0000000E FF008AF0  00000000 04000000 000E070C 20000000   ......S ........0 ................
0097A0   09620000 8AF00800 0000000E 070F2000  00000962 00000000 04000000 E2400000   .....0.......... .............S ..
0097C0   000EFF00 8AF00000 00000400 0000000E  070C2000 00000962 00008AF0 08000000   .....0.......... ...........0....
0097E0   000E070F 20000000 09620000 00000400  0000E240 0000000E FF008AF0 00000000   ................ ..S ........0....
009800   04000000 000E070C 20000000 09620000  8AF00800 0000000E 070F2000 00000962   ................ .0..............
009820   00000000 04000000 E2400000 000EFF00  8AF00000 00000400 0000000E 070C2000   ........S ...... .0..............
009840   00000962 00008AF0 08000000 000E070F  20000000 09620000 00000400 0000E240   ........0........ ...............S
009860   0000000E FF008AF0 00000000 04000000  000E070C 20000000 09620000 8AF00800   ........0........ ..............0..
009880   0000000E 070F2000 00000962 00000000  04000000 E2400000 000EFF00 8AF00000   ................ .....S ........0..
0098A0   00000400 0000000E 070C2000 00000962  00008AF0 08000000 000E070F 20000000   ................ ....0............
0098C0   09620000 00000400 0000E240 0000000E  FF008AF0 00000000 04000000 000E070C   ..........S .... ....0............
0098E0   20000000 09620000 8AF00800 0000000E  070F2000 00000962 00000000 04000000   ..........0...... ................
009900   E2400000 000EFF00 8AF00000 00000400  0000000E 070C2000 00000962 00008AF0   S ........0...... ................0
009920   08000000 000E070F 20000000 09620000  00000400 0000E240 0000000E FF008AF0   ................ .......S ........0
009940   00000000 04000000 000E070C 20000000  09620000 8AF00800 0000000E 070F2000   ................ ......0..........
009960   00000962 00000000 04000000 E2400000  000EFF00 8AF00000 00000400 0000000E   ............S .. ......0..........
009980   070C2000 00000962 00008AF0 08000000  000E070F 20000000 09620000 00000400   ............0.... ................
0099A0   0000E240 0000000E FF008AF0 00000000  04000000 000E070C 20000000 09620000   ..S ........0.... ................
0099C0   8AF00800 0000000E 070F2000 00000962  00000000 04000000 E2400000 000EFF00   .0.............. .........S ......
0099E0   8AF00000 00000400 0000000E 070C2000  00000962 00008AF0 08000000 00000000   .0.............. ........0........
```

(Ex 4 KT)

The example below shows the beginning of a dump of the BG partition using the command DUMP BG when SYSLST is assigned to a line printer.

```
CR 0-7   004000FF 0000E640 FFFFFFFF FFFFFFFF  00000000 00000000 00000000 00000000
CR 8-F   0000E000 00000000 00000000 00000000  00000000 00000000 C2000000 00000200

BGVPSW   071D0000 0004044A
GP 0-7   00000000 00D2F000 0000002B 00000000  00000000 0025CC2B 000006A0 000000D4
GP 8-F   00057D53 00000009 0000000E 4004007A  0004107A D7C8C1E2 B0040426 000422F8

FP 0-6   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

040000   D7C8C1E2 C55C5C5C 071D0000 0004044A  00000009 0000000E 4004007A 0004107A   PHASE***........ ........ .......
040020   D7C8C1E2 B0040426 000422F8 00000000  00D2F000 0000002B 00000000 00000000   PHAS.......8.... .KO............
040040   0025CC2B 000006A0 000000D4 00057D53  0000849D CA205500 00000000 00000000   ...........M.... ................
040060   00000000 00000000 00000000 00000000  00000000 00000000 05B041CB 0FFF41CC   ................ ................
```

Dump command output
(DUMP BG)

(Ex 4 KT)

Dumps of, and changes to Real and Virtual Address Areas

SYSTEM DUMP

// OPTION DUMP statement

If this statement is included in the job stream a dump of the partition, the supervisor and/or the SVA will be printed on SYSLST, whenever a program is canceled either by the operator or by other cancel conditions. The dump includes a printout of the supervisor area, and if the program check is within the SVA, a dump of the SVA or parts of it. For this reason the dump is often referred to as a SYSTEM DUMP.

Note: There is no need to insert the // OPTION DUMP statement in a job stream if the default option of the DUMP parameter has been specified in the STDJC macro during system generation. If however the dump is not required when abnormal termination occurs during execution of a particular job, the // OPTION NODUMP statement must be included in the job stream.

To enable the // OPTION DUMP statement to operate, one of two types of system dumps must be cataloged into your system during system generation:

- A standard system dump, whose printed output is in hexadecimal code
- A translating system dump, whose output is printed in hexadecimal and alphameric codes.

The dump output includes the following:

- General registers
- Floating point registers, if FP is specified for the system
- Control registers
- Active communication region address (See Section 4 for a description)
- Supervisor (see note below)
- PD area, if PD is specified for the system (see B-3 in this section for details)
- Label length
- Partition identifier: BG, F4, F3, F2, or F1
- Temporary real or virtual partition.
- The Shared Virtual Area (See note 2 below).

Note 1: The LTA (Logical Transient Area) is used to contain the dump program. Therefore, the LTA printed in the dump will always contain the B-transients $$BDMPBC (if the dump is directed to a line printer or tape unit) or $$BDMPDC (if the dump is directed to a disk).

Note 2: If a program check occurs in the SVA before the dump logical transients are loaded and have control, a dump of the SVA is executed. Even if the program check occurs in a phase within the SVA, the phase name of which has been deleted from the SDL (System directory list) before the dump transients have control, parts of the SVA are dumped that contain the phase in which the program check occurred.

When to use

Insert this statement into job streams for new, modified, or untested programs. The partition dump and general register dump can be analyzed and the information obtained will help during offline program debugging, *How to use the dump output*

Begin the analysis by examining the error message issued on SYSLOG and/or on SYSLST. If the program check occured within the SVA, the message

PART OF SVA WHICH CAUSES THE ERROR

is printed after the dump of the partition in which the program is running, or if the entire program is running in the SVA, the message is printed after the dump of the program save area. A hexadecimal dump, including storage of the SVA, or a dump of the phase running in the SVA addresses in which the program check occurred – follows the message.

The example below illustrates how to use the system dump in conjunction with program listings and the linkage editor map in order to isolate a data exception program check occurring in a program running in a BGV partition.

Step 1: Check for messages on SYSLOG and/or SYSLST. From the message obtain the address at which the interrupt occurred and reason for the program check.

```
BG   0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 040712 -
          CONDITION CODE 3 - DATA EXCEPTION
     0S00I JOB BUGPRGCK CANCELED
BG   EOJ BUGPRGCK
     DATE 04/20/73,CLOCK 02/38/02,DURATION 00/10/59
```

(Ex 5A KT)

```
0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 040712 - CONDITION CODE 3 - DATA EXCEPTION
0S00I JOB BUGPRGCK CANCELED
```

(Ex 5B KT)

Step 2: Locate the register values printed at the beginning of the system dump. Scan the contents of register for unreasonable values. (This may help in later problem analysis.)

```
         BUGPRGCK          12/06/73                                          19.35.18                  PAGE   1
GR 0-F   0000001B 00040910 00000021 00000002   0000002E 00000000 00000000 00000000
         00000000 00000000 00000000 4004007A   0004107A D7C8C1E2 A00406EE 000422A0
FP REG   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
CR 0-F   004000FF 0000E640 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
         0000FFFF 00000000 00000000 00FFFFFF   00000000 00000000 C2000000 00000200
COMREG   BG ADDR IS 0004A0

000000   00000000 00000000 00000000 00000000   00000000 000004A0 40000000 00000000   ...............  ....... .......
000020   470D0000 000085DE 440C0000 000009D2   00000000 00000000 470C2000 0000090C   ..............K  ...............
000040   0000E720 08000000 0000E710 00000000   FB61C700 023E32CD 440C0000 00000C14   ..X.......X.....  ./G............
000060   440C0000 00000BCC 000C0000 0003E800   04080000 0000D13A 440C0000 00000B10   ..............Y.  ......J........
000080   00000540 00000000 00020007 00040000   12042003 00020000 00000000 000001CC   ... ...........  ...............
0000A0   00000000 00000000 20000060 000002C0   00000000 00000100 0000000E 00000000   ..........-....  ...............
0000C0   00000000 --SAME--                                                           ....
0004A0   F1F261F0 F661F7F3 70007000 00000000   00000000 00000000 C2E4C7D7 D9C7C3D2   12/06/73........  .......BUGPRGCK
0004C0   00060FFF 0004232F 0004232F 00000010   0008FFFF FD7FCED3 38A0CED0 002E40FD   ...............  .......L...... .
0004E0   41044296 42974389 3F003F06 3F0C38F1   F2F0F6F7 F3F3F4F0 00003CD4 0000003C   ...............1  20673340...M....
000500   46300000 3DCC3E4C 3EBC0010 00000010   00008090 00007118 00003864 3B045A30   ...............  .............M..
```

GR 1 may be interesting

R 13 looks strange ?

Address of active comreg

Job name

(Ex 6 KT)

Dumps of, and changes to Real and Virtual Address Areas

SYSTEM DUMP

Step 3: Locate the linkage editor map and obtain the relocation factor. Subtract this value from the address given in the program check message.

In case of a program check in the SVA, the relocation factor applicable is obtained as follows:

1. Subtract the instruction address printed in the program check message from the first address printed in the dump of the SVA.
2. Add the result to the start address printed in the assembly listing for the program or phase causing the error.

```
12/06/73   PHASE      XFR-AD  LOCORE  HICORE  DSK-AD    ESD TYPE  LABEL     LOADED  REL-FR
           PHASE***   040078  040078  04232F  00D 0D 2  CSECT     BEGIN     040078  03C878  RELOCATABLE
                                                        CSECT     IJFFZZWZ  041ED8  041ED8
                                                        * ENTRY   IJFFZZZZ  041ED8
                                                        CSECT     IJCFZIZ0  042230  042230
                                                        CSECT     IJDFCZZZ  0422A0  0422A0
                                                        * ENTRY   IJDFZZZZ  0422A0
                                                        CSECT     IJ2L0067  0422F8  0422F8
```

PC address

40712 −
3C878
3E9A =
address in listing

Ex 7

Step 4: Locate the resulting address (from step 3) in the program listing. This will give the failing instruction.

```
003E88 D203 C682 BBD2 04E84 043D4   573   MVC   PLN,MASKER
003E8E D203 C686 BBD2 04E88 043D4   574   MVC   AND,MASKER
003E94 D203 C68A BBD2 04E8C 043D4   575   MVC   TOT,MASKER
003E9A DE03 C676 C5EC 04E78 04DEE   576   ED    EDB,DECE+6
003EA0 DE03 C67A C5F4 04E7C 04DF6   577   ED    KNG,DECK+6
```

Label for the data area used as the "SOURCE" for the EDIT instruction

(Ex 8 KT)

Step 5: Compare the hexadecimal code for the instruction in the program listing with the code in the dump. It should be the same.

```
040640  CCAF5810 CDDA58F1 001045EF 000CD24F   C2BFCCFF 5810CDDA 58F10010 45EF000C   .......1......K.  A........1......
040660  47F0B2C2 4780B302 95E0B985 4780B75C   47F0B24C 4E50C5E6 4E60C5EE 4E70C5F6                                .E6
040680  4E80C5FE 4E90C606 4EA0C60E 1A561A57   1A584E50 C6165810 CDAE4100 001B58F1                                ..1
0406A0  001005EF 95D7C83F 4780B4F6 5850CDB2   5860CDB2 5870CDB2 5880CDB2 5890CDB2                                ...
0406C0  58A0CDB2 D203C672 BBD2DE03 C672C61C   D283B936 B935D203 B951C672 D215B93B                                ...
0406E0  CE105810 CDAE58F1 001045EF 000CD203   C676BBD2 D203C67A BBD2D203 C67EBBD2                                ..K
040700  D203C682 BBD2D203 C686BBD2 D203C68A   BBD2DE03 C676C5EC DE03C67A C5F4DE03
040720  C67EC5FC DE03C682 C604DE03 C686C60C   DE03C68A C614D283 B936B935 D212B93B   F.E...F.F...F.F
040740  CE29D202 B94DC677 D20BB952 CDEED202   B95DC67B D210B962 CD9ED202 B972C67F   ..K...F.K.....K.  ...............F.
```

40712

Hex code is OK } therefore instruction not overwritten

(Ex 9)

Step 6: Locate the location in the program listing that defines the data area used by the failing instruction. Use the relocation factor from the linkage editor map to calculate the address of the data area in the dump.

```
0043E1 4040404040404040   1325 TPTOPR   DC   CL80' '
004431 4040404040404040   1326 CDTOTP   DC   CL80' '
004481                    1327 READTAPE DS   10CL80
0047A1                    1328 RITETAPE DS   10CL80
004AC1                    1329 BUGSWARN DS   10CL80
004DE8                    1330 DECE     DS   D
004DF0                    1331 DECK     DS   D
004DF8                    1332 DECM     DS   D
004E00                    1333 DECP     DS   D
004E08                    1334 DECA     DS   D
004E10                    1335 DECT     DS   D
004E18                    1336 DECW     DS   D
004E20                    1337 R1SAVE   DS   F          SAVE AREA FOR REGISTER ONE
004E24                    1338 LOOPCNT  DS   F
004E28                    1339 TIMSA    DS   9D
004E70 E3D6D4E2           1340 INSERT   DC   C'TOMS'
```

4DE8 +
3C878
41660
relocation factor

41660   41666   41667

DECE

DECE+6 and 7 used by the EDIT instruction

Ex 10

Step 7: Locate the data area used by the instruction in the dump. Check if the data is specified for the failing instruction. If it is not, continue by identifying the point in the program listing at which the data is prepared.

```
041600  E2E5C340 D5C5E640 50000048 92201004   47F0301C 40404040 40404040 F3841078   SVC NEW &.......  .0..        3...
041620  00609240 1080DC07 10785000 F3841081   00649240 1089DC07 10815000 9D004000   .-. ......&.3...  ... ......&... .
041640  4770303C 9C004000 9D004000 47303048   47F01010 40404040 40404040 40404040   ...... ... .....  .0..
041660  00000000 000000E9 00000000 0000004C   00000000 0000010C 00000000 0000004C   .......Z........  ................
041680  00000000 0000042C 00000000 0000068C   00000000 0000026C 8004033C 92601004   ..............
0416A0  F3841078 00                                                                   3....... .....
0416C0  40009D00 40                                                                   ... ......0..
0416E0  50000048 47F0301C E3D6D4E2 4040F2F6   40202                                  .0..TOMS
041700  40202120 40202120 D3D6D5C4 D6D54040   40400                                  ...LONDON
041720  17800000 84000C00 00040004 17800004   17880                                  ..........
041740  090417A4 60000020 090417C4 6000002F   090417F3 6000001F 09041812 60000021   ....-......D-...  ....-........ ...
```

DECE

4 left most bits of byte used as SOURCE in EDIT

(This is cause of prog check)

Now, continue debugging to see how it got there.

Ex 11 KT

DUMPGEN

The IBM program DUMPGEN allows you to generate a stand-alone dump program that must be used to obtain information about the system under certain conditions of system malfunction.

The dump consists of a printout of real storage (except bytes X'00'–X'17', X'40'–X'4B', X'BA'–X'BB' and 214 bytes of a non-critical area in the supervisor). Two types of dump programs can be generated using DUMPGEN:

- Translating dump
- Formatting dump.

Both programs produce a conventional dump with translation. In addition, the formatting dump produces a pre-formatted printout of the DOS/VS interface tables. This dump is generated if the DUMPGEN option FORMAT=YES is specified.

Executing DUMPGEN

Before being able to execute DUMPGEN you must catalog it to the core image library. Execute it in any partition by the job control statement or command:

```
// EXEC DUMPGEN
```

You enter DUMPGEN and read its control statements from SYSIPT.

i.e. For Example See Page 2.17

Note that SDAIDS may not be initiated during execution of DUMPGEN. (SDAIDS are described in Section 2-B.) The two types of control statements used with DUMPGEN are ASSGN and OPTN, described as follows:

```
// JOB DGEN
// EXEC DUMPGEN
ASSGN SYSLST,X'00E'
OPTN INTR=YES
OPTN FORMAT=YES
```

ASSGN Statement: ASSGN defines the output device for the stand-alone dump program.

Name	Operation	Operand
(blank)	ASSGN	SYSLST, X 'cuu'

see p2.16

SYSLST The only valid logical unit assignment.

X 'CUU' Must define the address of the SYSLST printer. If the ASSGN statement is omitted, then X'00E' is assumed.

DUMPGEN AND STAND-ALONE DUMP

OPTN statement: OPTN defines the type of output generated by the DUMPGEN program.

Name	Operation	Operand
(blank)	OPTN	INTR= NO YES DECKS= nnnnnn PPOOL= NO YES FORMAT= NO YES TAPEIPL= NO YES

Operands for the DUMPGEN option statement.

INTR
INTERRUPT

YES produces a DUMP program that, when loaded, enters the WAIT state. Either press the INTERRUPT button on the CPU operating panel to print the output on X'00E', or press the STOP button and then START button of the printer desired for the output device.
NO produces a DUMP program that, when loaded, prints out the contents of real storage either on the SYSLST printer defined with the ASSGN statement or on X'00E'.

DECKS

Specifies the number of DUMP card decks (punched out on SYSPCH) desired. nnnnnn may be any decimal number from 1 to 99,999,999. A blank card separates each deck produced. If DECKS is omitted, one deck is produced.

PPOOL

YES produces a dump program that, after printing out real storage, will print the formatted contents of the Boundary Box and the contents of the real storage in sequence of ascending virtual addresses. If NO is specified, the last two items are not printed.

FORMAT

YES produces a translating stand-alone dump that formats and displays the DOS/VS supervisor tables after displaying the contents of real storage. This formatted display depends upon the location of the communications region.
If the communications region cannot be related, the program is terminated when the formatted display is to occur.
In this case the following message is printed on the dump output:

COULD NOT FIND COMREG BETWEEN C0 AND A00,
FORMATTING WILL NOT OCCUR

If NO is specified or FORMAT is omitted, a non-formatting translating dump is generated.

TAPEIPL

If YES is specified and SYSPCH is assigned to a tape unit, the stand-alone dump written on tape may be IPLed directly from the tape unit. If NO is specified, or TAPEIPL is omitted and SYSPCH is assigned to a tape unit, the stand-alone dump records are written on tape preceded by an ASA control character.

Control statements for the DUMPGEN operands

Control statements may be specified in any order; however, the following rules apply:

- All statements may be omitted, but if they are, DUMPGEN assigns printer X'00E', INTR=NO, FORMAT=NO, and PPOOL=NO options.
- Only one operation and only one operand per control statement is allowed.
- The last statement processed of a duplicate operation overrides all previous statements of the same operation with similar operands (if DECKS=2 is followed by DECKS=5, five decks are punched).
- The name field must be blank.
- Decimal operands may contain leading zeros.
- One of more blanks must follow the operand if comments are to be made.

Job stream example

The following example is a typical job used to create a stand-alone dump.

```
// JOB
// EXEC DUMPGEN
Col. 2

ASSGN SYSLST, X'00F'
OPTN FORMAT=YES
OPTN PPOOL=YES
OPTN DECKS=5
/*
/&
```

DUMPGEN messages

The functions of DUMPGEN-to-operator error message routines are:

- Cancel the job if SYSLOG is not a 3215/3210 or a System/370 Model 125/115 video display unit.
- Reissue the message if operator response is to press the CANCEL key
- Process an operator response of END/ENTER as IGNORE
- Cancel the job if operator response is CANCEL.
- Ignore the control card in question when the operator response is IGNORE.

If none of the preceding operator responses is issued, then DUMPGEN assumes that a correction has been made and processes it.

DUMPGEN AND STAND-ALONE DUMP

Stand-alone Dump Program

This program is generated for your installation using the IBM program DUMPGEN.

DUMPGEN produces a dump program that is either punched into a card deck or stored on magnetic tape. When required, the dump program thus generated can be loaded into the system via the standard IPL procedure.

The stand-alone dump program that is generated by DUMPGEN provides either a conventional dump or a formatted dump, depending on the FORMAT option used in the DUMPGEN program.

Operation

During execution of the stand-alone dump program, a non-critical area in the supervisor is used to load the program. The LOAD ADDRESS of the non-critical area is punched (in decimal) in the first card of the stand-alone dump card deck punched by the DUMPGEN program. Because of this use of the non-critical area it is recommended to use the stand-alone program for a system using a supervisor that was used for the generation of that dump.

The conventional dump prints the contents of real storage locations, but does not dump the floating point registers. In addition to the areas dumped by the conventional dump, the formatted dump prints the DOS/VS interface tables in a more readable form.

For both types of dump the following is printed:

1. The contents of the general registers, the old and new PSWs, the interruption codes, CSW, CAW, and TIMER.
2. The contents of real storage in 2k blocks. Each block is preceded by a sequence number.
3. At the end of the real storage dump, page address and status information is printed that contains the following information for each page frame:
   - The virtual address
   - The real address of the associated page
   - The sequence number of the 2k block
   - Information that indicates whether the contents of the page frames has been changed.
4. The contents of the control registers are printed after page address and status information.
5. Depending on the options selected, the following then occurs:

   If PPOOL=YES
   - The formatted contents of the boundary box is printed after the control registers.
   - The contents of real storage is printed in 2k blocks in sequence of ascending virtual addresses.

If FORMAT=YES,
the formatted contents of the tables listed below are printed at the end of the dump.

COMREGS
PIBs
AP SUBTASK PIBs (if AP supported)
PARTITION SAVE AREAS
LUBs
PCIL LUBs (if PCIL supported)
PUB
ERROR RECOVERY BLOCK
CHANNEL QUEUE
FLOATING POINT REGISTERS
COPIED AND TRANSLATED CCB
FIXINF EXT. BLOCKS
COPIED AND TRANSLATED CHANNEL PROGRAM
IDAL BLOCK QUEUE
FIXINF BLOCK
BOUNDARY BOX
SEGMENT TABLE
PAGE TABLE
PAGE FRAME TABLE and PAGE FRAME TABLE EXT
SELECTION POOL

The full names of these tables, their contents, locations, and meaning to the system programmer during offline program debugging are found in Section 4.

An example of the formatted output of these tables is given in Appendix G.

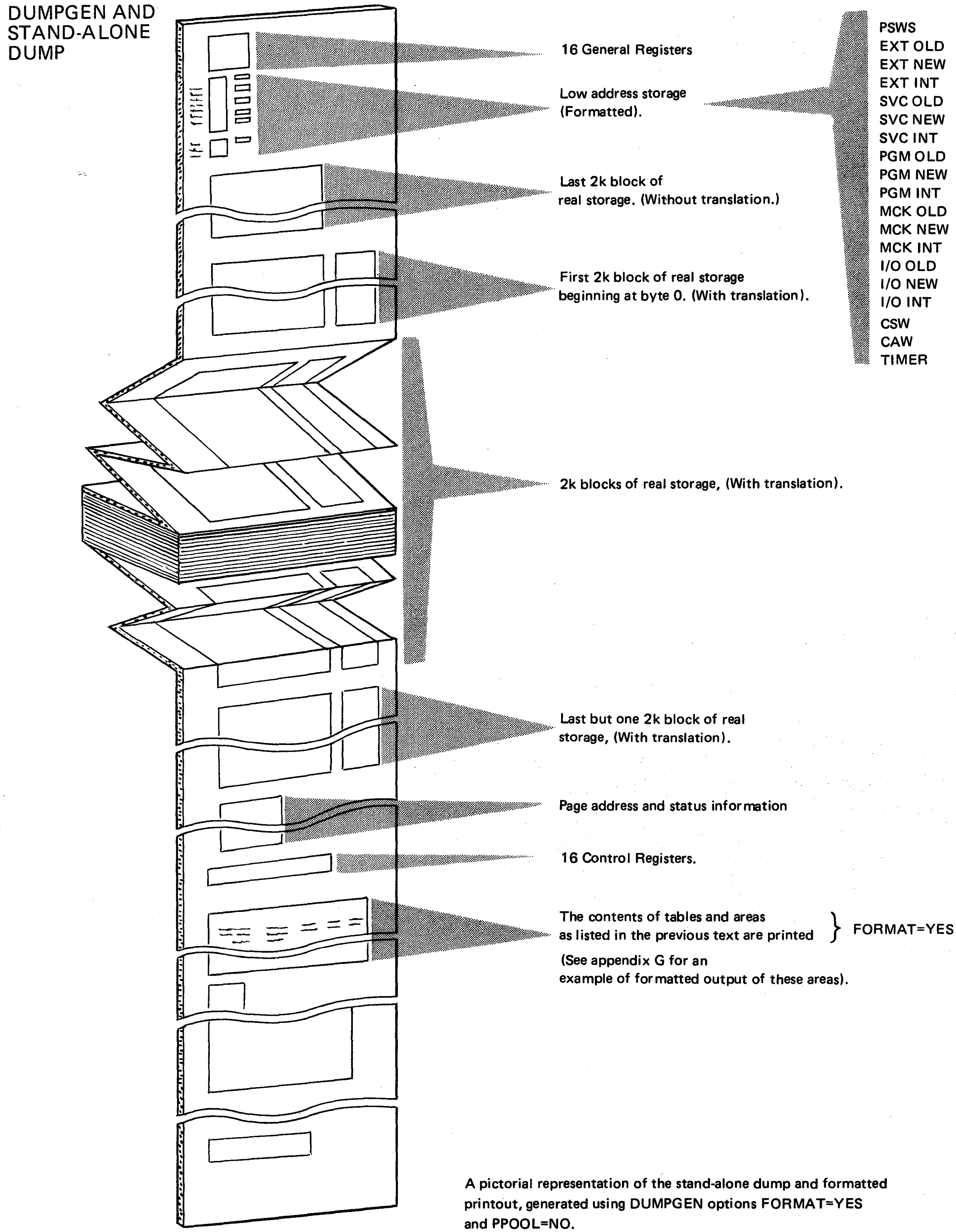
The two following illustrations show the information that is printed after executing the dump program.

When to use

The stand-alone dump program must be used whenever the severity of a system malfunction, such as a loop or hard wait state, prevents alternative methods of obtaining system information that aids offline debugging.

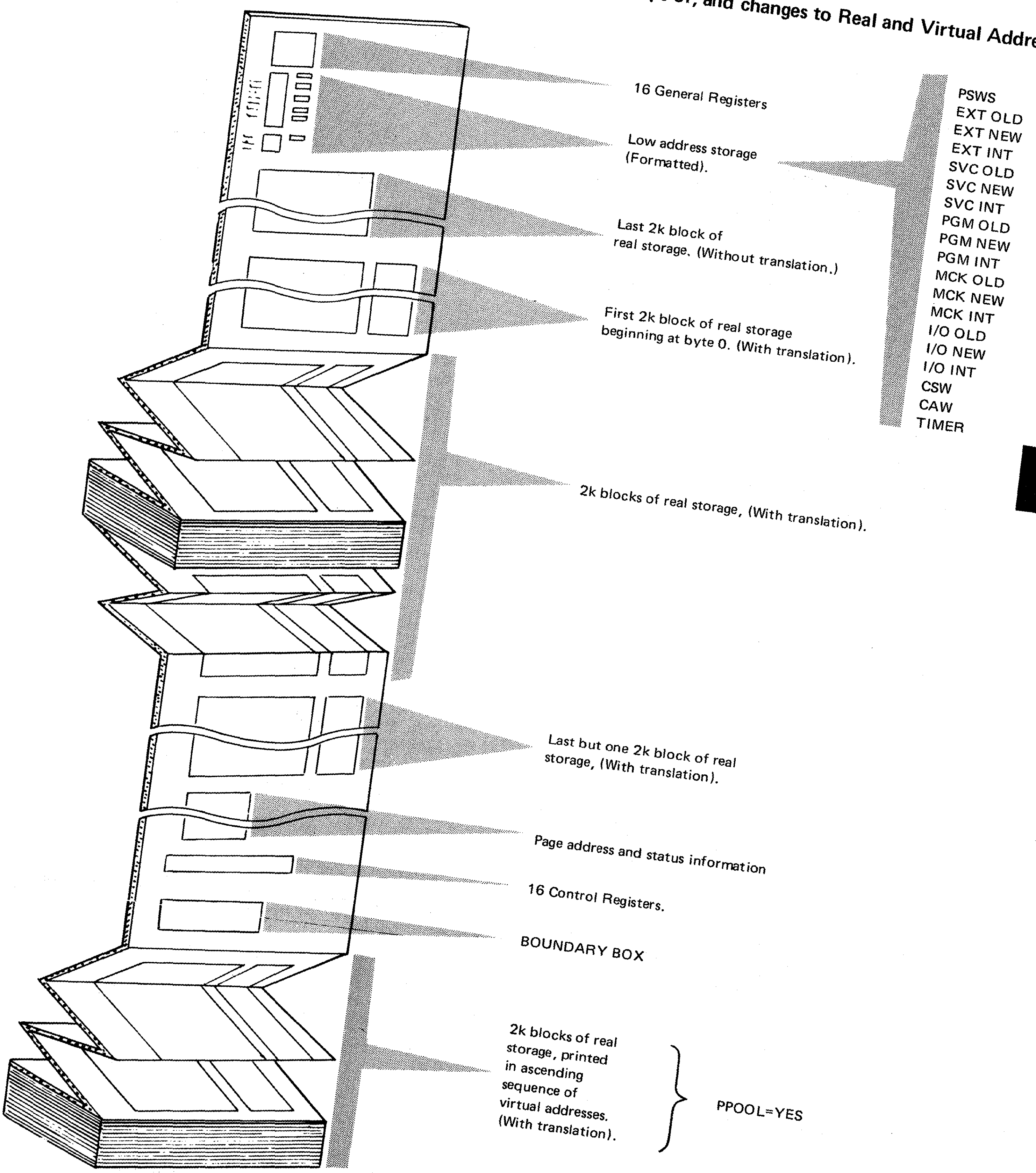
Flowcharts in Section 3 indicate when to execute the stand-alone dump program.

Dumps of, and changes to Real and Virtual Address Areas



A pictorial representation of the stand-alone dump and formatted printout, generated using DUMPGEN options FORMAT=YES and PPOOL=NO.

Dumps of, and changes to Real and Virtual Address Area



A pictorial representation of the stand-alone dump unformatted printout, generated using DUMPGEN options FORMAT=NO, and PPOOL=YES

A-3

DUMPGEN AND STAND-ALONE DUMP

A note to the operators

Before the stand-alone dump program is executed, the operator must dump, or display and note, the contents of bytes X'00' through X'17', X'40' through X'4B', X'BA', and X'BB' of low address storage. (The contents of these bytes will be destroyed when the dump program is loaded.) It may be important to the programmer to have a note of the contents of the control registers at the time of the error. This can be done by (1) executing the store status function or (2) dumping or displaying the control registers using the ALTER/DISPLAY feature described in Section 2-D. The operator should also display and note the current PSW before executing the dump. Also there may be a need to dump the page data set after executing the stand-alone dump. For example, the programmer may have made a request for a "SYSVIS dump" after the execution of a stand-alone dump. The flowchart shown opposite indicates the procedure for loading and executing the stand-alone program.

A note to the programmers

To ease the task of locating and interpreting the contents of control blocks and tables during offline debugging, generate the formatted dump program (FORMAT=YES).

A note to IBM SE/CE

For any System/370 supporting RAS the serial number and System/370 Model type is stored in the first 8 bytes of the RAS linkage area, the address of which is located at displacement X'70' of SYSCOM.

How to use

Initially the following listed areas should be examined, for what appears to be unexpected information.

1. General registers. } See note.
2. Current PSW and old PSWs. }
3. From the PIB table locate the partition in control at the time of the error.
4. Registers and the PSW in the partition save area.
5. Using the program listing of the program that was running in the failing partition, scan the I/O areas, instructions, intermediate results, and operands in areas you consider critical to the program.
6. Use the linkage editor map to locate where the system should load the phases.
7. For further analysis, check the PUB table for any I/O request left outstanding, and, depending on the type of the program in error, check the CCB/DTF table and label save areas.

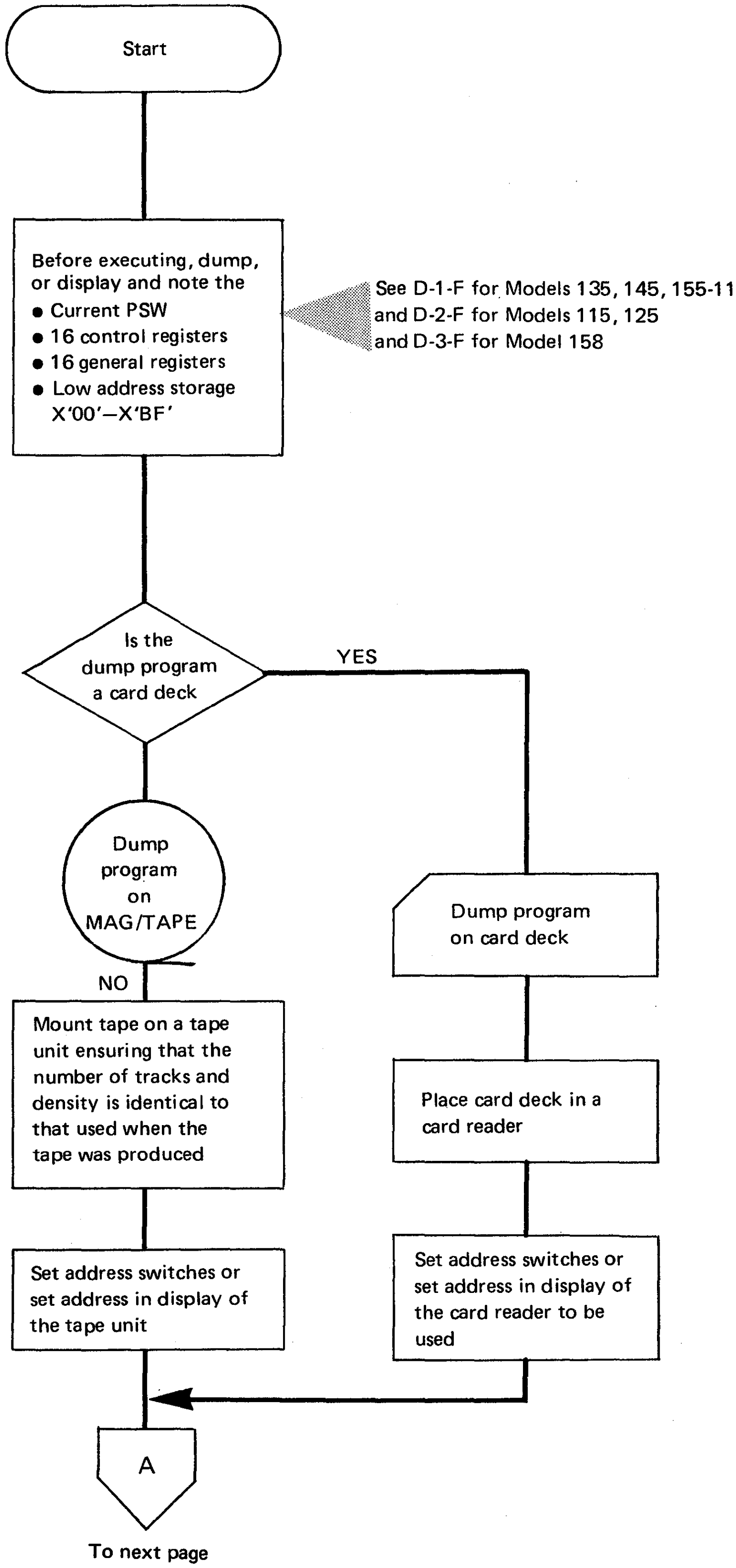
The order in which these areas are examined rests with you, who as programmer, will know what the program was expected to do, and approximately what the contents of certain registers and I/O areas should contain.

An example of a stand-alone output is given in appendix G.

By examining the dump in this way and by consulting the program listing you will be able to form an idea of the cause of the error, and to discover where to look for further clues or pointers.

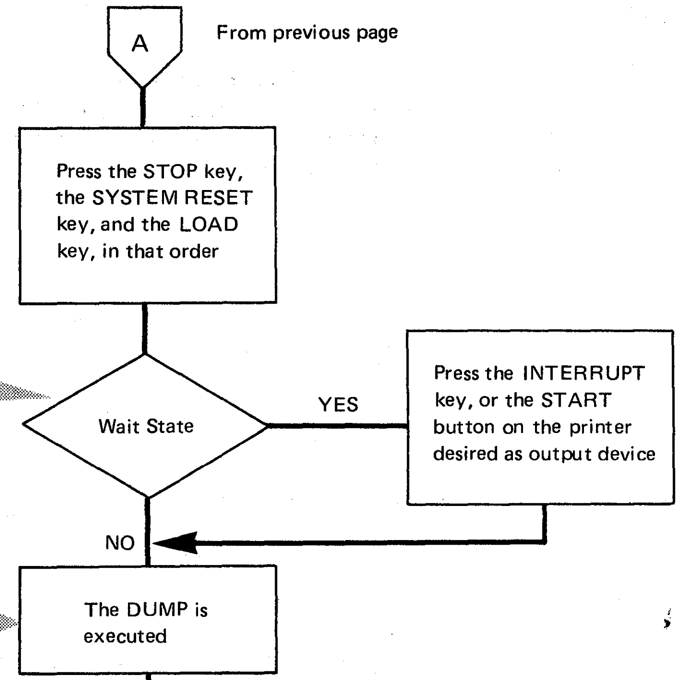
However, the first pointer will depend on the symptoms of the error, the environment, what the program was expected to do, any output that became available, and incorrect results of calculations.

Note: Use the information printed at the start of the dump, that is before the print out of the last block of real storage. (The contents of low address storage is not reliable after the execution of a stand-alone dump program.)


Start
Before executing, dump, or display and note the
● Current PSW
● 16 control registers
● 16 general registers
● Low address storage X'00'–X'BF'
See D-1-F for Models 135, 145, 155-11
and D-2-F for Models 115, 125
and D-3-F for Model 158
Is the dump program a card deck
YES
NO
Dump program on MAG/TAPE
Dump program on card deck
Mount tape on a tape unit ensuring that the number of tracks and density is identical to that used when the tape was produced
Place card deck in a card reader
Set address switches or set address in display of the tape unit
Set address switches or set address in display of the card reader to be used
A
To next page

Dumps of, and changes to Real and Virtual Address Areas

DUMPGEN AND STAND-ALONE DUMP



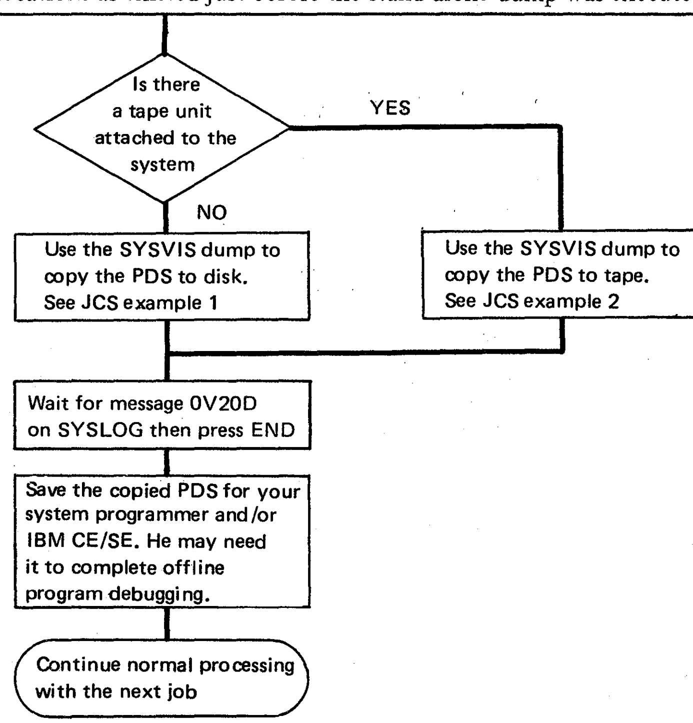
A wait state occurs if INTR YES is specified during generation of the dump program using DUMPGEN.

The contents of REAL storage is printed in blocks of 2K bytes on the line printer specified in the ASSGN statement of DUMPGEN or the line printer made ready after entering the WAIT STATE. Other information is also printed depending on the type of dump program generated by DUMPGEN.

The SYSVIS dump utility program is described under C-6 in this Section

The following procedure executes the SYSVIS dump utility program. This procedure copies the PDS to tape or disk which can then be dumped to SYSLIST if required during offline debugging. It is also possible to dump the PDS directly to SYSLST. To ensure that the contents of the PDS and the allocations of the virtual address area are the same as they were just prior to the execution of the stand-alone dump the following instructions must be adhered to:

1. Re-IPL using identical parameters for the DPD command as specified in the previous system IPL. **However you must specify N to the parameter TYPE=.**
2. Check for any previous ALLOC commands. You must specify identical virtual partition allocations as existed just before the stand-alone dump was executed.



Example 1

```
// JOB DUMPPDS
// ASSGN SYS000,X'cuu'
// ASSGN SYS001,X'cuu'
// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// DLBL S01DISK,'BACKUP FOR PDS',
// EXTENT SYS001,111111,1,,0020,180
// EXEC PDSDM
/&
```

Example 2

```
// JOB DUMPPDS
// ASSGN SYS000,X'cuu'
// ASSGN SYS001,X'cuu'
// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// TLBL S01TAPE,'BACKUP FOR PDS'
// EXEC PDSDM
/&
```

PDAIDS

PDAIDS (problem determination aids), are routines especially designed to provide specific information useful during offline debugging. These routines consist of four trace routines, which are described in Section 2-B of this manual, and one routine called the transient dump program.

Transient Dump

This program is designed to dump, on a program check, areas of the supervisor before they are altered. The dump provides:

- the 16 general registers
- the 16 control registers
- the first X'20F' bytes of low address storage
- the logical transient area (with the label LTA)
- the physical transient area (with the label PTA).

This information may be provided on either a printer or a tape unit. When tape is used, the tape must be processed by the PDLIST utility program to provide readable output data. PDLIST is described in Section 2-B. Both the printer and tape modules are reusable, that is a dump occurs with each program check until the function is reset.

The printed dump output is non-translating.

System requirements

Because the Transient Dump program is a PDAID function, it requires the PDAID initializing phase and a PD area. Refer to "System Requirements" for Trace Routines in Section 2-B-1.

TRANSIENT DUMP

Initializing the transient dump

Initializing is done by calling the PDAID program via the job control statement // EXEC PDAID.

The parameters for the dump may be entered through SYSLOG or through the card reader assigned to SYSIPT.

Note: No other PDAID can be executed when the transient dump program is initiated.

If SYSIPT is to be used, the card deck must be punched as follows.

Punch desired keywords and parameters, as shown in example 1, into cards. Entries may be punched one-per-card, or as multiple entries (separated by commas) in a single card. An entry may not be split between two cards. All 80 columns of a card may be used, but a card is terminated either by the first blank following an entry, or by a GO entry. The last entry of the last card must be GO and the last card must be followed by a /* card.

Note: If an incorrect parameter is read from a card, corrections are requested on SYSLOG.

When the main phase (PDAID) has been loaded into any free partition (one must be made available), the following message is issued on SYSLOG:

4C10D PDAID=

The operator must respond to this message with one of the following:

- TD Initiates transient dump.
- XX Terminates PDAID.
- END key Indicates that the parameters are to be entered via SYSIPT.

Selecting the Output Device

The initializer keyword OUTPUT DEVICE selects an output device, which must be specified by channel and unit address, not by symbolic unit. When an output device is specified, the initializer checks the address against the supervisor PUB and automatically selects the appropriate module for the unit type (tape or printer).

Once the transient dump program has been initiated, it is given control each time a program check interrupt occurs. When it has control, the transient dump program has highest priority, and the system accepts only external interrupts. All system processing is suspended and the operator must ready the transient dump output device.

When to use

Use the Transient dump when you suspect coding errors in the transient routines, for example, your own error recovery routines for devices not supported by IBM. The information obtained from the dumps will help during offline program debugging.

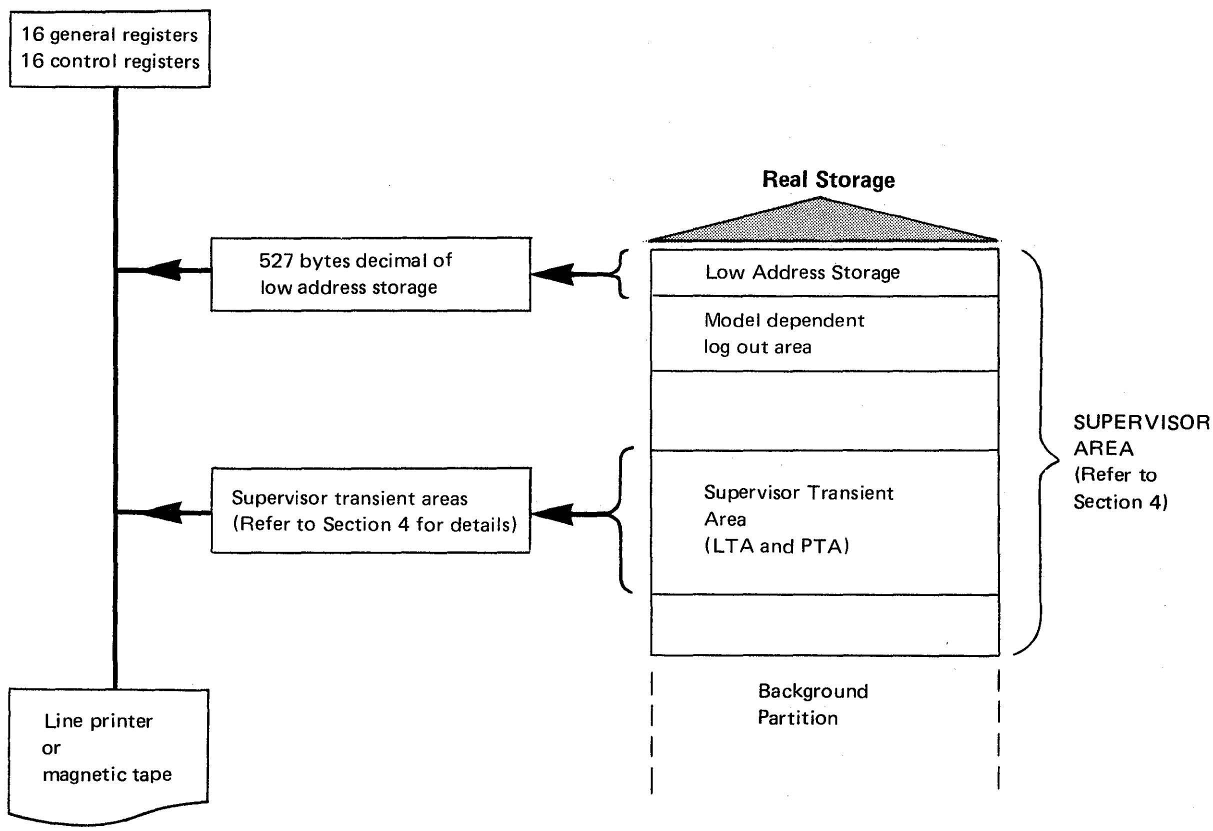
Terminating the transient dump

The transient dump program can be terminated by re-initializing the PDAID program (// EXEC PDAID), and responding to the message PDAID= with XX. It is also possible to reset (terminate) the transient dump by loading one of the PDAID trace routines.

The following illustration represents the action of the transient dump program if a program check interrupt occurs in the LTA.

Dumps of, and changes to Real and Virtual Address Areas

TRANSIENT DUMP



Pictorial representation of the information that is dumped.

Keyword	Parameter	Meaning	Default
PDAID	TD XX	Initiate transient dump. Terminate function	 Function continues.
OUTPUT DEVICE (Note 2)	CUU or X'cuu'	Use specified output device for output of transient dump function.	
GO (Note 1)		End of initializer keyword entries.	

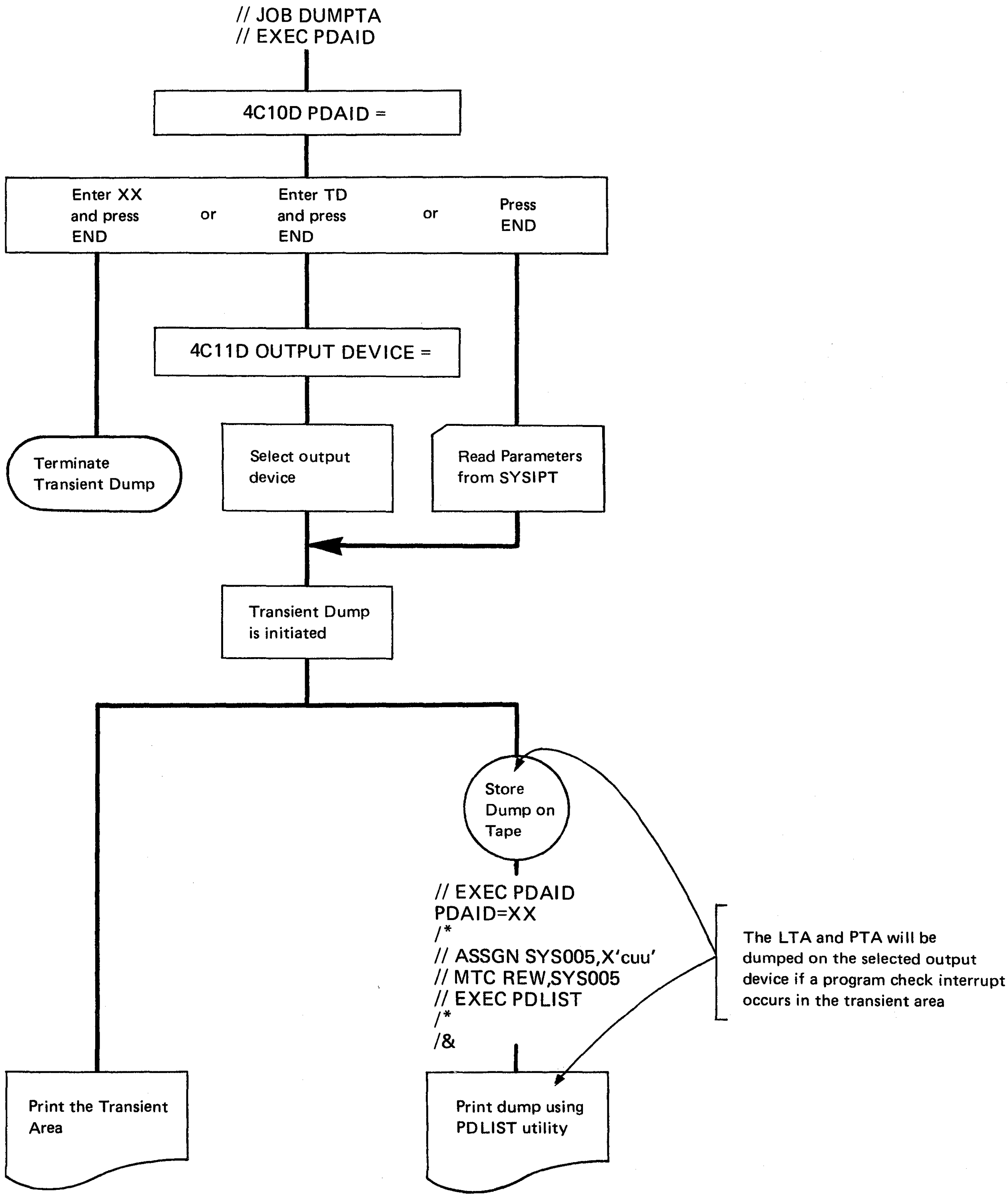
Note 1: *GO is an invalid response to a request for a console correction to card input.*

Note 2: *A printer or tape output device must be specified for transient dump. CUU or X'CUU' notation must be specified by channel and unit address, and not by symbolic unit.*

Two examples of initiating the transient dump immediately follow the flowchart shown opposite.

Table A-4. Table of parameters for initializing the Transient Dump

TRANSIENT DUMP
Initializing the Transient Dump



Dumps of, and changes to Real and Virtual Address Areas

TRANSIENT DUMP

Job stream examples

The following two examples show job stream to initiate the transient dump program.

Example 1, via SYSIPT:

// JOB CARDINP7	
// EXEC PDAID	Calls for initializer.
PDAID=TD	Calls for transient dump function.
OUTPUT DEVICE=00E	Specifies printer output.
GO	Signals end of input.
/*	
/&	

Note: A dump is given on all program checks.

Example 2, via SYSLOG:

// JOB TYPINPT6	
// EXEC PDAID	Calls for initializer.
PDAID=	Console requests function.
TD and END key	Operator specifies transient dump function.
OUTPUT DEVICE=	Console requests output device.
00E and END key	Operator specified printer output.

Note: An output device must be specified for the transient dump function.

PDAID Transient Dump

```
        KENSDUMP            12/06/73

GR 0-F  00040684 00008620 00040298 00040308   000085B1 08040300 00000000 000084DA
        000084BC 00000008 00040280 00040688   00040078 00000008 80008300 000082E8

CR 0-F  004000FF 0000E640 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
        0000FFFF 00000000 00000000 00000000   00000000 00000000 C2000000 00000200
```

Program check old PSW

```
000000  00000000 00000000 00000000 00000000   00000000 000004A0 071D0000 0004044A   040C0000 00002876 070D0000 000082F6
000030  00000000 00000000 070F2000 0000090C   000087A0 0C000000 00003A88 00000000   FE3BD600 055C0E26 040C0000 00000C14
000060  040C0000 00000BCC 000C0000 0000A5DC   04080000 0000013A 040C0000 00000810   00000540 00000080 00020007 00060001
000090  12042003 00020000 00000000 000001CC   00000000 00000000 20000060 000002C0   00000000 00000100 00000130 00000000
0000C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0000F0                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
000120                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
000150                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
000180                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0001B0                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0001E0                                        00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
```

First 8 bytes of LTA (give phase name of transient causing the error

Failing instruction address = 82F6

$$BOPEN

```
LTA
0082E8  5B5BC2D6 D7C5D540 C7F0F00E 28004590   F2669101 30044710 F09495FE 70004780   F0389551 80044780 F038D300 F4188004
008318  9550F418 4780F0B6 91013002 4710F08A   91023002 4710F080 9108A087 4710F076   9110A087 47E0F05E 94EFA087 0A0B9550
008348  F4184770 F0705860 2008D201 40266006   4110F41A 0A024110 F42A94F7 A0870A02   4110F43A 94FD3002 0A024110 F43294FE
008378  30020A02 4110F422 0A020A00 91801002   4710F0A6 0A079120 10024710 F0B007F9   92FFF419 07F94160 F3F04060 100A4590
0083A8  F09A9108 F2D74710 F0DCD605 40204020   4780F038 96044003 47F0F038 9104A087   47E0F0EC 94FBA087 47F0F104 4160F3F8
0083D8  4060100A 4590F09A 4160F400 4060100A   4590F09A 95FFF419 4780F038 9118A087   4770F11C 91112014 4710F126 D503F2EE
008408  F40E4780 F12CD205 F2F2F412 D5054020   F4084780 F234D505 4020F2F2 4770F15A   D505F2F2 F4124770 F2349110 30024710
008438  F15A9112 201447E0 F234D76B F26AF26A   9280F288 9203F282 D201F286 8004920A   F28AD201 F2848000 9206F291 D205F284
008468  F2D69201 F2899550 80044780 F1889218   F2919214 F28AD217 F2BEF2D6 9204F289   926CF3DC 95538004 4770F188 9209F291
008498  9205F289 925DF3DC D205F292 4020D200   F2A58005 D201F29E 400BD201 F2A24012   D201F2A0 4001D200 F2A4400D D203F2A6
0084C8  400ED201 F2984026 95508004 4770F214   D209F2AA 4030D602 40004000 4780F234   D7024000 4000D733 40084008 9250F3DC
0084F8  47F0F230 D213F2AA 4030D602 40004000   4780F234 D7024000 4000D73B 40084008   4590F24A 9110A087 4710F240 96044003
008528  D2054020 F2F247F0 F038D203 F266F3DC   18514110 F2660A2C 9180F268 4710F262   0A071815 07F958A0 001458B0 A0885830
008558  B01C4110 F3E09118 A0874780 F2A24000   10064590 F3821821 45E0F394 18444347   00008940 00014844 30345A40 303047F0
008588  F0B69180 A0594710 F0701820 91033002   4740F34E 9104A058 4710F32E 91013004   4710F320 1BDD18C0 412CD000 95002000
0085B8  4770F320 58220000 95FF2007 4780F2F0   95202014 4740F2F6 95282014 4720F2F6   96013004 07F99532 20144770 F31845E0
0085E8  F39495FE 700047B0 F318D300 F4188004   9560F418 4780F2F0 41DD0004 47F0F2C8   94FE3004 18205822 000047F0 F34E5820
008618  F4AC9500 20044780 F34A9102 A05847E0   F0705822 000047F0 F34E5822 000495FF   20074780 F0709520 20144740 F3669528
008648  201447D0 F070D201 10062006 45E0F394   18664360 70008960 00014846 30345A40   30304160 F2D64060 F3F24160 F2EE4060
008678  F3FA07F9 487A007C 488A002E 48887002   487A004A D200F3BB 70001B77 43720007   91012006 47E0F3BC 41770000 1A781A77
0086A8  4A7A004C 18884387 00008980 00034A8A   004007FE 00000000 00000000 00300400   00001500 00000000 00000000 20000000
0086D8  040002D6 20000018 020002EE 2000000C   07000400 20000001 00000000 0000E506   D3F14040 40404040 00005B5B C2D6D7C5
008708  D5F15B5B C2D6D7C5 D5C45B5B C2D1C3D6   D7E35B5B C2D6D4E3 F0F15B5B C2D6D4E3   F0F50000 00000000 00005C00 012C0000
008738  84000000 92600004 00006060 C2C76060   C3D6D4D9 C5C74040 C2C740C1 C4C4D940   C9E240F0 F0F0F4C1 F0D3C2D3 E3E8D740
008768  40C8C5E7 40D3C5D5 C7E3C840 C9E240F0   F0F0F060 D74BC44B 604040C1 D9C5C100   000020F2 F14BF1F3 4BF1F660 80041224
```

(Non-translating dump)

First byte of failing instruction = X'C7'

a non-existent op code (thus cause of program check)

```
PTA
008E60  5B5BC1C2 C5D9E9F1 28005890 00144890   908458C0 008048A0 905A9287 A0009102   903547E0 B02E48A0 C05847F0 B03248A0
008E90  906E4AA0 905A1B88 BF87A005 5810802C   4570B068 48F0C046 58E0C004 9280A000   9127A008 4780B05C 9283A000 5860C000
008EC0  D2016006 B0C407FE 95FF1002 4780B09A   1B554350 10028950 00035A50 C0249500   50044780 B0B895FF 50004780 B09A1B44
008EF0  43405000 185447F0 B0764111 000895FF   10004770 B0689283 A0009680 A00148A0   905A94DF A00807F7 5010802C 0A0348A0
008F20  905A07F7 D9C1D201 00BA3000 4A20905A   18A248F0 C00847F0 B23C4100 B29094AD   D0084570 B14AD508 B02AB342 4780B21A
008F50  D505B02A B3364770 B14A18AD 4850C02C   4860907C 41AA0010 41660010 D501B031   B3424780 B120D501 B031A002 4770B142
008F80  9180600F 4780B1A8 4140B21A 4100B2D8   9226B267 9220B264 4570B14A 9260B264   07F44650 B10492F2 B2ABBE07 B2614100
008FB0  B260BE07 B31F4100 B02ABE07 B271D708   B02AB02A 4100B34C BE07B269 4110B316   0A0F9180 10024710 B17C0A07 9120B318
008FE0  4710B21A 9218B267 9101B31A 4780B19C   4100B270 BE07B31F 47F0B15E D608B02A   B3424100 B2A807F7 4100B2C0 91029035
009010  47E0B1E6 4120000F 4B20C02C 4830C02C   89300004 4A30905A 41330010 D5013002   A0024770 B1E29601 3000960C 300F921D
009040  30014620 B1C89580 A0004780 B14A9110   A00F4770 B14A9601 A00048F0 C00A47F0   B23C4100 B2FE9218 B2679220 B2644570
009070  B14A9260 B26447F0 B03E48F0 C04658E0   C0049280 D0009127 D0084780 B2329283   D0005860 C00092C1 600707FE 9140D008
0090A0  4710B248 94F8D008 4BA0905A 42A0902F   91029035 4780B21E 42A0C05B 47F0B21E   090090F0 60000018 010091AC 60000004
0090D0  0A008E8A 20000009 F0D7F6F0 C440C9D5   E3C5D9E5 40D9C5D8 40404040 40F0F0F0   F1C9F4F0 C440D9C5 D8E4C5E2 E340C3C1
009100  D5C3C5D3 40404040 F1E2F0F1 C440C9D5   E5C1D3C9 C440E2E3 C1E3C5D4 C5D5E340   F1C9F3F2 C440C1D9 C5C140D5 D6E340C1
009130  C3E3C9E5 C5404040 F1D8F6F2 C940C3C1   D5C3C5D3 40C6D6D9 40D7D6E6 C5D940D7   D9D6C7D9 C1D440C9 C7D5D6D9 C5C4F0D7
009160  F8F9C940 C961D640 C3C1D5C3 C5D340C9   C7D5D6D9 C5C40003 81000800 00040000   90C00000 90D8F0F1 F2F3F4F5 F6F7F8F9
009190  C1C2C3C4 C5C6C3C1 D5C3C5D3 C9C7D5D6   D9C54040 40404040 40404000 C1D94040   00000000 00000000 00000000 00000000
0091C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0091F0  00000000 00000000 00000000 00000000   00000000 00004040 40404040 40C3C3E2   E67EF0F1 F1F0F0F4 F0F4F2F0 F0F2F0F0
009220  F0F0F0F0 40C3C3C2 7EF0F4F0 F4F0F040   E2D27EF6 F1F6F1F0 F0F0F0F0 F0F0F000   F0D7F0F8 C1404040 40C9D5E3 C5D9E540
009250  D9C5D840 E2E8E2D3 E2E37EF0 F0C56000
```

(Ex 12

A-4

Transient Dump output

This example shows the output obtained from the transient dump program. (The dump is output when a program check occurs in the transient area.)

The programmer's remarks on this example indicate the main points of interest to aid offline debugging, however this depends on the error symptoms and system environment.

SUPERVISOR COMMUNICATION MACROS

PDUMP (partial dump) macro

This macro instruction provides a hexadecimal dump of:

- The general purpose registers
- The floating point registers (if FP is supported)
- The control registers
- The virtual storage area contained between two address expressions.

The addresses can be expressed in decimal or hexadecimal or in register notation and need not be confined to any one partition.

Name	Operation	Operand
(name)	PDUMP	address 1 , address 2 (r) (r)

Address 1 **specifies the start address of the storage to be dumped.**

Address 2 **specifies the end address of the storage to be dumped.**

(r) **one or both of the addresses can be specified in any of the general registers.**

The contents of registers 0-1 are destroyed, but the CPU status is retained. Thus, PDUMP furnishes a dynamic dump (snapshot) that is useful for program checkout. Processing continues with the next user instruction.

The dump is always provided by SYSLST on 121-byte records. The first byte is an ASA control character. If SYSLST is a disk drive, the user must issue an OPEN macro to any DTF assigned to SYSLST after each PDUMP that is executed. The OPEN macro updates the disk address maintained in the DTF table to agree with the address where the PDUMP output ends. If OPEN is not issued, the address is not updated, and the program is canceled when the next PUT is issued.

The specified addresses are checked against the end address of virtual storage. If address 1 is higher than the end address of virtual storage, or if address 1 is higher than address 2, the PDUMP macro results in no operation. If address 2 is higher than the end address of virtual storage, address 2 is automatically set to that address.

If address 1 and 2 are identical, only the contents of the general registers, the control registers and floating point registers are dumped. (Floating point registers are dumped only when the supervisor supports the floating point option.) The dump output can be either standard (non-translating) or translating, depending on the dump program cataloged in your system transient library.

Note: Addresses for this macro may not be specified by register notation for programs eligible to run in the SVA (shared virtual area).

When and how to use

PDUMP is useful when you need to know the contents of specific virtual storage areas at specified points in the program during program execution. For example, you may want to examine the contents of storage areas that are being modified during program execution, such as I/O areas.

The following example illustrates the use of the PDUMP macro.

```
                                         400 OUT      CANCEL
                                         405 ************************************************************************
                                         406 *LOOP TIMER ROUTINE*
                                         407 ************************************************************************
003C7E D24F C2BF C2BE 04AC1 04AC0        408 TIMINTR  MVC   BUGSWARN,BUGSWARN-1
003C84 D227 C2BF CD4F 04AC1 05551        409          MVC   BUGSWARN(40),CHKPTWRN
                                         410          PUT   TYPOUT1
                                         415          PDUMP LOOPCNT,LOOPCNT+3        DUMP LOCATIONS USED AS COUNTER
                                         420 *                                       FOR THE LOOP AT EACH TIMER INT.
                                         421          EXIT  IT
                                         424 ************************************************************************
                                         425 *END OF LOOP ROUTINE*
                                         426 ************************************************************************
```

(Ex 13

This source code listing shows the PDUMP macro where the programmer needed to know the contents of an area used as a counter.

The following dump printout was obtained when the program was executed.

```
        KENTOMS1          12/06/73                                                                  PAGE   1

GR 0-7  00041E10 00041E08 00000049 00000000   00000000 00000050 00000010 00000002
GR 8-F  00000001 0000CA34 00000013 4004007A   0004107A D7C8C1E2 8004050E 000422F8
FP REG  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
CR 0-7  004000FF 0000E640 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
CR 8-F  0000FFFF 00000000 00000000 00000000   0000,000 00000000 C2000000 00000200

041680                                                                0000CA34                   ....
```

(Ex 14 KT)

Restriction:

The message INVALID ADDRESS SPACE is printed on the dump output if the dump includes storage areas considered to be invalid address space.

The definition of invalid address space is listed under "Restrictions" in the description of the ALTER command.

A-5

SUPERVISOR COMMUNICATION MACROS

DUMP macro

This macro, when assembled into your program and executed, will dump the following system information:

- The general registers
- The floating point registers (if FP is supported)
- The control registers
- The active communication region address (see Section 4 for a description)
- The supervisor
- The PD area (if PD is specified for the system; see B-3 in this Section)
- The label length
- The partition identifier BG, F4, F3, F2, or F1
- The temporary real or virtual partition issuing the macro.

Name	Operation	Operand
(name)	DUMP	

If the program or main task issued the macro, the job step is terminated

JDUMP macro

Name	Operation	Operand
(name)	JDUMP	

If the program of main task issued the macro, the main task (the whole job) is terminated and a dump is made of those areas listed in the description of the DUMP macro.

The following considerations apply to both the DUMP and JDUMP macros:

1. If a subtask issues these macros, the subtask is detached, the job step or job, respectively, is not terminated, and the dump described above is executed.
2. The dump is always provided on SYSLST, which, if disk or tape, must be OPENED.
3. If either macro is issued by a program running in real mode, the temporary real partition is dumped. However, if these macros are issued by a program running in virtual mode, the whole virtual partition is dumped.
4. The dump output can be either standard or translated, depending on the type of dump cataloged into your system during system generation.
5. The LTA (Logical Transient Area) is used to contain the dump program; therefore, the LTA printed in the dump will always contain a B-transient $$BDUMPB (if the dump is directed to a line printer or tape unit), or $$BDMPDC (if the dump is directed to a disk drive).

When to use

By coding these macros into your source listing you can ensure that a dump of the supervisor and of the partition issuing the macro is executed.

For example, you may require a partition dump when certain conditions arise during program execution. This is accomplished by programming a branch to the DUMP (or JDUMP) macro written in the source listing. The JDUMP macro must be used when it is necessary to terminate the job after entering the routine that issued the macro. The DUMP macro is used when termination only of the job step is required, for example, during program testing. After termination of the job step in which the macro was issued the job steps after that are still executed.

Intentionally Blank

CONTENTS

CONTENTS (continued)

B-1

Two series of trace routines are provided on the System/370: PDAIDS and SDAIDS

These aids enable information to be obtained from the system at the time of a malfunction. They are aids for further error isolation, and are usually initiated during a rerun of a troublesome program after a first analysis of the problem. The type of trace to use for a particular problem depends on the result of the first analysis and how much more information is required to help in further isolation of the error.

This section is divided into two parts:

Part 1 describes the PDAIDS, and part 2 describes the SDAIDS.

PART 1 PDAIDS

General description

There are four trace routines that can produce printed output of certain events which occur during the execution of programs.

The trace routine will:

- Record I/O operations (I/O trace)
- Record the order in which phases and transients are called (Fetch/Load trace)
- Record the order in which supervisor calls (SVCs) are executed (Generalized SVC trace)
- Record the order in which either an SVC 0 or an SVC 31, and I/O interrupts occur. (QTAM trace).

On the occurrence of an event, an entry is generated which, by selection of the trace, can be recorded on magnetic tape, printed on a line printer, or preserved either in the PD area or, if specified, in an alternate area of real storage.

Caution

The effect on the operation of programs currently running in the system that are time dependent, for example, a program using MICR or teleprocessing as input/output, must be considered before using this serviceability aid.

System requirements

Before any PDAID function can be executed, the following requirements must be met:

- During the system generation, specify a minimum value of 1400 in the PD parameter of the FOPT macro. (The maximum value is 10,240).
- If data provided by the trace routines is recorded on magnetic tape, use the PDLIST program after tracing is complete to obtain a printout of the tape.

All PDAID modules are distributed by IBM in the relocateable library and must be cataloged into the core image library before execution. They are self-relocating for initialization in any real or virtual partition (6K or greater) of a multiprogramming system.

Restrictions: More than one PDAID trace routine cannot operate concurrently. This also applies to the PDAID Transient Dump program described in Section 2-A-4. Therefore, more than one program rerun must be executed if more than one PDAID function is used to gather information about a failing program.

Using PDAID and SDAID concurrently: IF SDAID is active it must first be terminated before initiating a PDAID trace in core-wrap output mode in an alternate area.

Modes of output

Line printer: (not available as output mode for QTAM trace) Examples in this section show the trace outputs when the output device is a line printer. An asterisk on the print-out indicates that at least one event (trace entry) has been overwritten. This occurs when an overflow is caused in the trace table in the PD area (described in B-3) or in an alternate area. This may occur when the trace output device, or its control unit, or channel, is shared with other programs running simultaneously.

If the printer is not ready or has an error condition, message 4C24A NO I/O TO OD is printed on SYSLOG and the system waits for the END/ENTER key to be pressed after the printer is made READY.

Magnetic tape: This mode of output collects and writes on an unlabled tape the trace entries that occur during execution of a job stream.

The events are written on tape in core image (unprintable) format.

The tape must be processed using the PDLIST utility. The tape unit must be assigned temporarily or permanently to SYS005 and SYSLST assigned temporarily or permanently to a line printer in order to obtain readable listings of the events traced. Examples in this section show the output format after using the PDLIST utility.

If the tape unit is not ready or has an error condition the message

4C24A NO I/O TO OD

is issued on SYSLOG and the system waits for the END/ENTER key to be pressed after the tape drive is made READY.

Core-wrap: This mode of output preserves a fixed number of trace entries in either the PD area buffer or an alternate area taken from the main page pool. If the alternate area is specified, the PD area buffer is not used. When the area is full, the oldest entry is overwritten by each new entry.

When core-wrap in the PD area is specified, the PD area must be dumped. The dump should normally be executed on the occurrence of a system malfunction when the last few trace event entries are required to aid offline debugging. Dumping and locating the PD area is described under B-3 in this section.

Table B-3 lists the length of each type of trace entry, the locations, and the maximum number of entries that can be preserved in the minimum PD area buffer size. Use the table and a dump of the PD area to locate the oldest and newest trace entries.

Core-wrap in an alternate area: If many events are to be recorded in the core-wrap output mode and the PD area is considered to be too small, specify an alternate area large enough to contain the trace event entries.

Specifying and dumping an alternate area is described under B-4 in this section.

When an alternate area is specified, the real storage taken from the main page pool is returned to the main page pool on termination of PDAIDS. Before the alternate area is released, its contents are dumped on the device assigned to SYSLST. (See "Termination of PDAIDS.")

When to use the core-wrap output mode: This output mode is useful when no output device is available, or when time required by the output operation is not available. This would be the case for example, when a PDAID output device interferes with time-dependent programs example, when a PDAID output device interferes with time-dependent programs using the I/O channels. It should also be specified when only the last few trace event entries are necessary to aid in offline debugging. (This reduces the task of searching through masses of output.)

Terminating PDAIDS

Any trace routine can be terminated by re-initializing the PDAID program with the job control statement // EXEC PDAID, and responding to the message PDAID= with XX. It is also possible to reset (terminate) one trace routine by loading another.

Terminating core-wrap output in an alternate area

When the core-wrap output is selected, SYSLST must be assigned to either a line printer, a tape unit, or a disk drive, before responding with XX to the message 4C10D PDAID=.

For example:

```
// ASSGN SYSLST, X'00E'
// EXEC PDAID
```

If SYSLST is unassigned, the contents of the alternate area is overwritten when it is returned to the main page pool.

PDLIST

Whether the PDAID function uses a printer for its output device, or the PDLIST program prints the output of a tape unit, the data printed out is identical.

PDLIST is initiated by the command:

```
// EXEC PDLIST
```

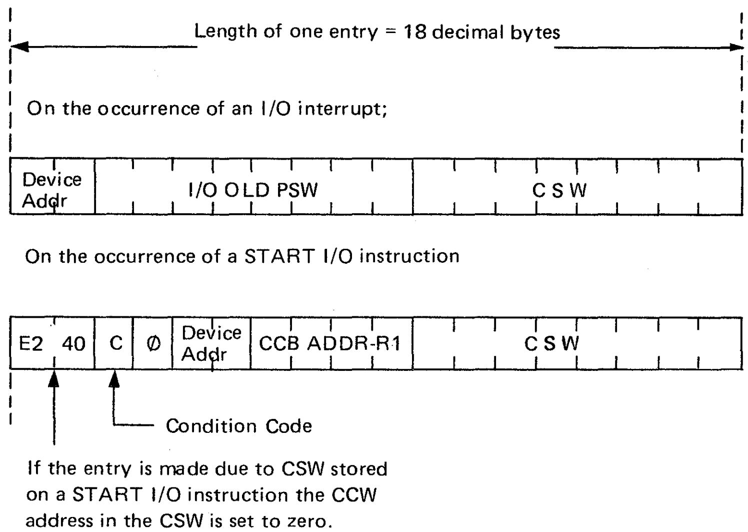
PDLIST then prints on SYSLST the contents of the tape reel (it can include the output of more than one PDAID function) mounted on SYS005. No tape labels are required.

Note: The data can only be printed using PDLIST if the device assigned to SYSLST is a line printer.

Description and operation

Input/Output Trace

This trace enables the I/O activity of programs run under DOS/VS to be recorded for offline analysis. The format of the data recorded in the PD area either in a trace table or, when using the core-wrap output mode in a rotating buffer, is as follows:



Notes:
1. *The PSW and CSW are described in E-2 of this Section.*
2. *The CCB is described in chapter 6 of Section 4.*
3. *General purpose register usage is described in chapter 10 of Section 4.*
4. *The CCB address and the CCW address in the CSW are virtual addresses.*

Either of these occurrences is referred to as an I/O event.

By selection of the trace output device, the event can be:
- Recorded on magnetic tape
- Printed on a line printer
- Preserved in the PD area
- Preserved in an alternate area

When magnetic tape output is used, the tape must be processed by the PDLIST utility program to provide a formatted output on a line printer.

The modes of output and PDLIST are described under B-1 in this Section.

Tracing Options: The I/O trace function provides the following options:

- Trace all I/O activity on the system.
- Eliminate a maximum of three devices.
- Limit trace to a maximum of three devices.

The trace limiting options are specified by the initializer keywords IGNORE DEVICE= or TRACE DEVICE=. All I/O activity is traced if one of these keywords is not specified. The two keywords are mutually exclusive: when one is specified, the other becomes invalid.

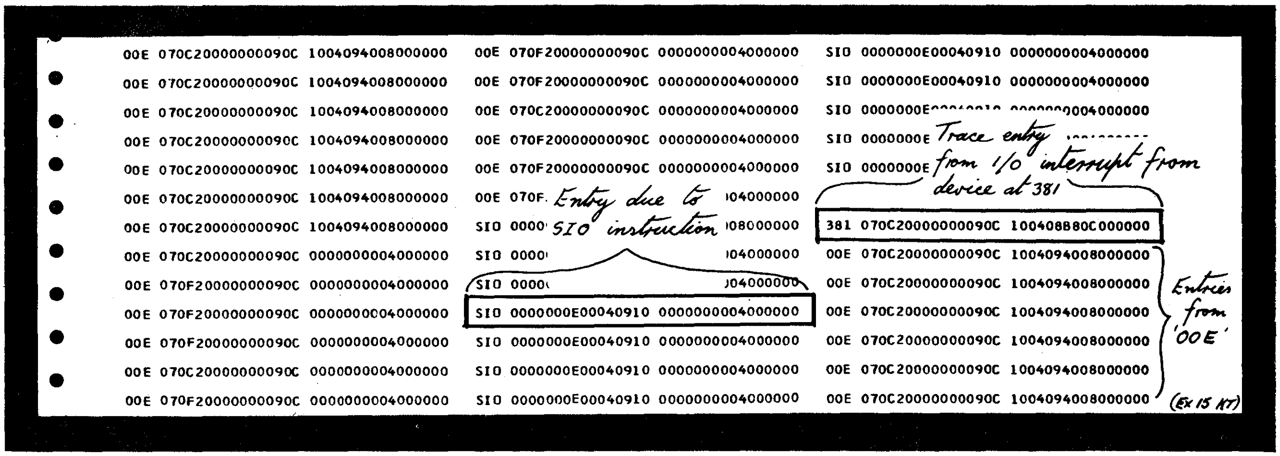
The trace limiting options are invoked by specifying the channel and unit addresses (X'CUU' or CUU) of the appropriate devices. Symbolic device references (SYSxxx) are invalid.

Note: If the trace output device is being used by a problem or control program simultaneously with the PDAID program, I/O events for the PDAID program are ignored (not traced). Because of this, it is not necessary to ignore the trace output device.

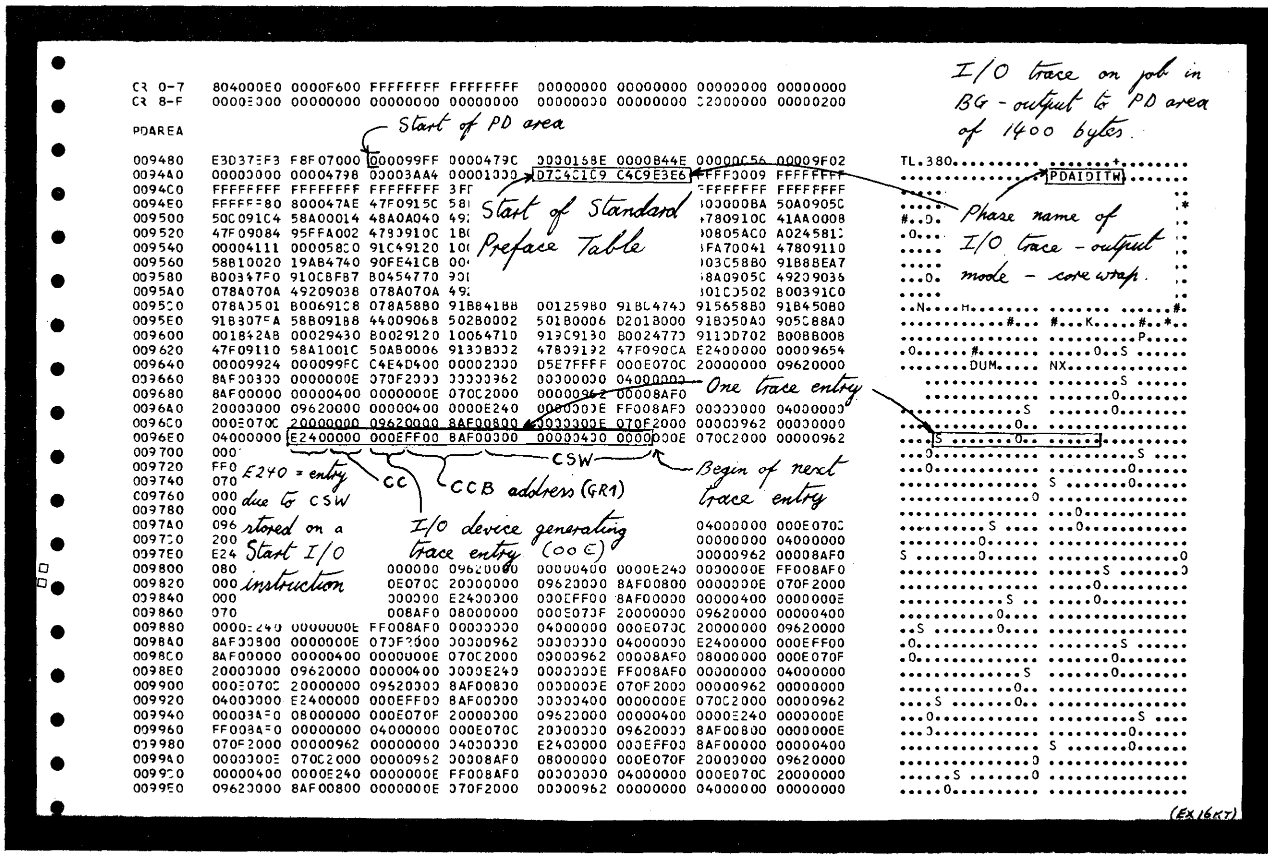
When to use: Use the I/O trace to check that the I/O interrupts within your system are correct during the execution of programs.

You could use it, for example, in a multiprogramming system where the status of I/O units is suspected of causing incorrect I/O interrupts. An I/O trace output will inform you about the sequence of SIO/ I/O interrupts and about the status of I/O units at the time of interrupt.

The next two examples show the output obtained from an I/O trace.



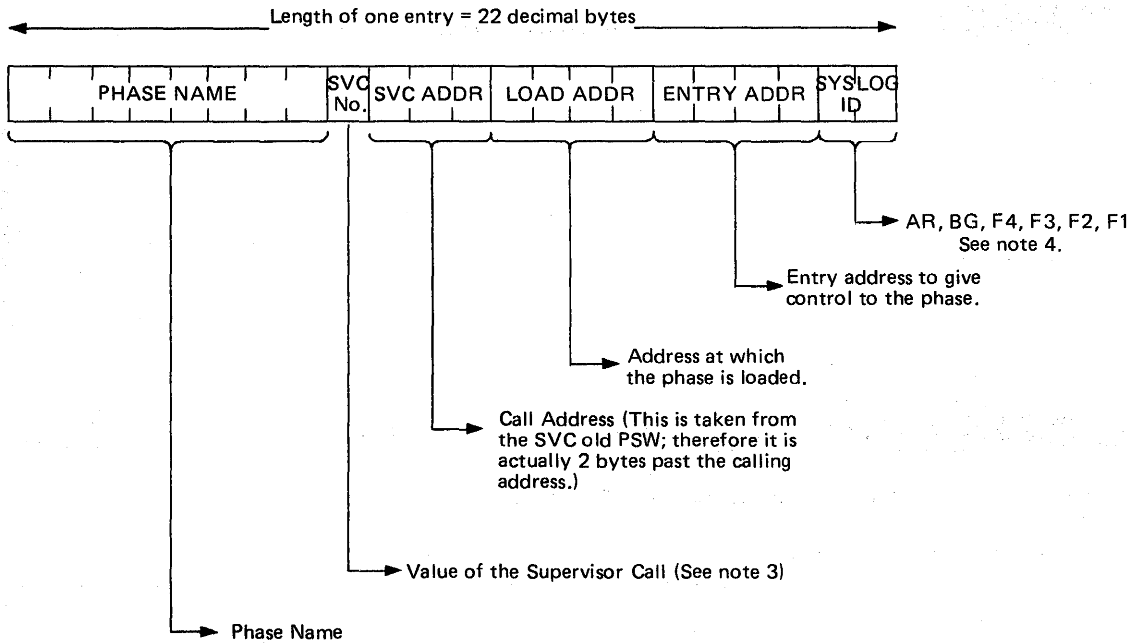
An example showing an I/O trace output printed on a line printer used as the PDAID output device.



An example of an I/O trace executed in core-wrap output mode in the PD area. The PD area was dumped using the DUMP command.

Fetch/Load trace

The F/L (fetch/load) trace records the order in which phases and transients are called from the core image library under the control of DOS/VS. Issuing a fetch or load causes an SVC 1, 2 or 4, and the format of data recorded is as follows:



Notes:

1. *At times, SVC 5, 6, 11, and 14 branch directly into the supervisor fetch or load routine. These are traced whenever they occur, and appear in the output of the trace; however, the calling address and SVC values do not indicate the actual fetch or load.*
2. *Use of the REQUEST key during the operation of the F/L trace may result in apparently erroneous data due to the supervisor action required to handle the request.*
   In particular, supervisor calls that have already been recorded may not be completed, and part of the data put out by the specific phase may pertain to these incomplete SVCs.
3. *A list DOS/VS SVCs can be found in Section 4.*
4. *The SYSLOG ID is described in appendix B.*

When the data is recorded in the PD area, either in a trace table or, when using the core-wrap output mode, in a rotating buffer, the two bytes used for the SYSLOG ID is recorded between SVC ADDR and the LOAD ADDR.

PDAIDS

On the occurrence of an event, an entry is generated. By selection of the trace output device, the event can be:

- Recorded on magnetic tape
- Printed on a line printer
- Preserved in the PD area
- Preserved in an alternate area.

When magnetic tape output is used, the tape must be processed by the PDLIST utility program to provide a formatted output on a line printer.

The modes of output and PDLIST are described under B-1 in this Section.

Tracing Options: The F/L trace functions are:

- Trace all SVC 1, 2, 4, and certain SVC 5, 6, 11, and 14 interruptions.
- Limit the trace by partition (multiprogramming systems only).

Trace limiting options are specified by the initializer keyword TRACE PARTITION=

These options are useful only when the user runs several partitions at once, and does not wish to trace all of them. If only one partition is operating at a given time, the default (trace all partitions) allows both the single partition and the supervisor to be traced.

When to use: Use the F/L trace if you are not certain which phases are required for a particular program, or in which sequence they are called by the program. From the trace output you can see where the phases were loaded and their entry addresses. In addition you can check the logical use of the phases for the program.

The next two examples show the output obtained from an F/L trace.

B-2

```
// JOB PDLIST                                                           DATE 12/06/73,CLOCK 20/31/54
MAP
// ASSGN SYS005,X'2A1'
// MTC REW,SYS005
// EXEC PDLIST


$$BCLOSE 2 C-04084A BG L-0082E8 E-0082F0  $$BCLOS2 2 C-0084D6 BG L-0082E8 E-0082F0

$$BEOJ3  E C-04084C BG L-0082E8 E-0082E8  $$BEOJ4  2 C-00837C BG L-0082E8 E-0082F0

$$BEOJ7  2 C-0084BA BG L-0082E8 E-0082F0  $$BEOJ   2 C-008372 BG L-0082E8 E-0082F0

$$BEOJS1 2 C-00847A BG L-0082E8 E-0082F0  $JOBCTLA 4 C-00839E BG L-040078 E-041888

$JOBCTLG 4 C-040958 BG L-041888 E-041888  $JOBCTLN 4 C-040958 BG L-041888 E-041888

$JOBACCT 4 C-041AE0 BG L-041B80 E-041B80  $JOBCTLG 4 C-040958 BG L-041888 E-041888

$JOBCTLF 4 C-040958 BG L-041888 E-041888  $JOBCTLG 4 C-040958 BG L-041888 E-041888

$JOBCTLJ 4 C-040958 BG L-041888 E-041888  $$BOPEN  2 C-040842 BG L-0082E8 E-0082F0
```

F/L trace on BG job to see when $$BOPEN is used. Trace output to tape then using PDLIST.

($$BOPEN phase called by SVC 2 issued by job in BG at address 40B42-2 =40B40

(Ex 17 KT)

An example showing an F/L trace output as printed on a line printer using PDLIST. (A tape unit was selected as output device for the PDAID.)

Job Name

F/L trace on job in F2.

End address of PD area

Start of PD area

PDAID Phase Name

```
           DEBUGEX3          10/05/73                                                      PAGE  50

016C00   00000000 --SAME--
016FE0   00000000 00000000 00000000           )00 00000000 00000000                        .........

-P.D.-   AREA
009480                     000099FF 0000479C   0000168E 0000B44E 00000C56 00009F02                          +........
0094A0   00000000 00006940 00003AE4 00001000   D7C4C1C9 C4C6E3E6 FFFF0009 FFFF5880   ....... ...U....  PDAIDFTW........
0094C0   90ECD202 B0090025 D200B008 008B07FA   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   ..K.....K.......  ................
0094E0   FFFFFF30 FFFFFFFF 07FA070A 07FA070A   47F09036 428090F5 58800080 91018042   ................  .0.....5........
009500   4710909A 95FF90F5 078A95FF 90F5078A   95FF90F5 078A95FF 90F5078A 95FF90F5   ........5.....5..  ...5.....5.....5
009520   078A58B0 90EC501B 000C500B 0010D207   B0005000 58800014 4880805A 4A8090F4   ......#...#...K.  ..#............4
009540   D2018014 80029502 B0084770 90CE4111   0008501B 0010581B 000C418B 001659B0   K...............  ..#.............
009560   90F04740 90E258B0 90E850B0 90EC07FA   00009584 0000973C 000099FC 0040FFFF   .0. .S...Y#.....  ............. ..
009580   FFFFFFFF 5B5BC2C5 D6D1F340 0B008908   000087C0 000087C0 C6F25B5B C2C5D6D1   ....&&BEOJ3 ....  .........F2&&BEOJ
0095A0   F4400200 88860000 87C00000 87C8C6F2   5B5BC2C5 D6D1F540 020087D4 000087C0   4 ..........HF2  &&BEOJ5 ...M....
0095C0   000087C8 C6F25B5B C2C5D6D1 F4400200   881A0000 87C00000 87C8C6F2 5B5BC2C5   ...HF2&&BEOJ4 ..  .........HF2&&BE
0095E0   D6D1F540 020087D4 000087C0 000087C8   C6F25B5B C2C5D6D1 F4400200 881A0000   OJ5 ...M.......H  F2&&BEOJ4 ......
009600   87C00000 87C8C6F2 5B5BC2C5 D6D1F540   020087D4 000087C0 000087C8 C6F25B5B   .....HF2&&BEOJ5   ...M.......HF2&&
009620   C2C5D6D1 F4400200 881A0000 87C00000   87C8C6F2 5B5BC2C5 D6D1F540 020087D4   BEOJ4 ..........  .HF2&&BEOJ5 ...M
009640   000087C0 000087C8 C6F25B5B C2C5D6D1   F4400200 881A0000 87C00000 87C8C6F2   .......HF2&&BEOJ  4 ..........HF2
009660   5B5BC2C5 D6D1F540 020087D4 000087C0   000087C8 C6F25B5B C2C5D6D1 F4400200   &&BEOJ5 ...M....  ...HF2&&BEOJ4 ..
009680   881A0000 87C00000 87C8C6F2 5B5BC2C5   D6D1F540 020087D4 000087C0 000087C8          HF2&&BE  OJ5 ...M.......H
0096A0   C6F25B5B C2C5D6D1 F4400200 881A0000   87C00000 87C8C6F2 5B5BC2                      ......  .....HF2&&BEOJ7
0096C0   02008996 000087C0 000087C8 C6F25B5B   C2C5D6D1 40400200 885E00                      .HF2&&  BEOJ .........
0096E0   87C8C6F2 5B5BC2D7 C3C8D240 02008A3E   000087C0 000087C8 C6F25B                      < ....  .......HF2&&BDUM
009700   D7400200 894C0000 87C00000 87C8C6F2   5B5BC2C4 E4D4D7C2 02008AA0 000087C0   P ..........HF2  &&BDUMPB........
009720   000087C8 C6F25B5B C2C4D4D7 C2C30200   88EE0000 87C00000 87C8C6F2 5B5BC2D6   ...HF2&&BDMPBC..  .........HF2&&BO
009740   D7C5D540                                                                    PEN ...........H  F2&&BOPEN1......
009760   87C00000                                                                    .....HF2&&BOPIGN  ............F2&&
009780   C2D6D4E3                                                                    BOMT05..........  .HF2&&BOPEN ....
0097A0   000087C0                                                                    .......HF2&&BOPE  N1...........HF2
0097C0   5B5BC2D6                                                                    &&B3PIGN........  ....F2&&BOUR01..
0097E0   89EE0000                                                                    ..........HF2&&BC  LOSE...........H
009800   C6F25B5B                                                           D3D6E2F3   F2&&BCLOS2......  .....HF2&&BCLOS3
009820   02008A02                                                           87C00000   ...<.......HF2&&  3CMT05..........
009840   87C8C6F2 5B5BC2C3 D3D6E2C5 02008A90   000087C0 000087C8 C6F25B5B C2D6D7C5   .HF2&&BCLOSE....  .......HF2&&BOPE
009860   D540020B 12C20000 87C00000 87C8C6F2   5B5BC2D6 D7C5D5F1 02008862 000087C0   N ...B.......HF2  &&BOPEN1........
009880   000087C8 C6F25B5B C2D6D7C9 C7D50400   89460000 87C00000 87C0C6F2 5B5BC2D6   ...HF2&&BOPIGN..  ..........F2&&BO
0098A0   D4E3F0F5 020089EE 000087C0 000087C8   C6F25B5B C2D6D7C5 D5400200 89220000   MT05...........H  F2&&BOPEN ......
0098C0   87C00000 87C8C6F2 5B5BC2D6 D7C5D5F1   02008862 000087C0 000087C8 C6F25B5B   .....HF2&&BOPEN1  ...........HF2&&
0098E0   C2C3C5D6 E5F1020B 315A0000 87C00000   87C8C6F2 5B5BC2C3 D4E3F0F1 02008986   BCE3V1..........  .HF2&&BCMT01....
009900   000087C0 000087C8 C6F25B5B C2C3D3D6   E2C5020B 17C20000 87C00000 87C8C6F2   .......HF2&&BCLO  SE...B.......HF2
009920   5B5BC2C3 D3D6E2F2 020089CE 000087C0   000087C8 C6F25B5B C2C3D3D6 E2F30200   &&BCLOS2........  ...HF2&&BCLOS3..
009940   8AD20000 87C00000 87C8C6F2 5B5BC2C3   D4E3F0F5 02008882 000087C0 000087C8   .K.......HF2&&BC  MT05...........H
009960   C6F25B5B C2C3D3D6 E2C50200 8A900000   87C00000 87C8C6F2 5B5BC2D6 D7C5D540   F2&&BCLOSE......  .....HF2&&BOPEN
009980   020B1702 000087C0 000087C8 C6F25B5B   C2D6D7C5 D5F10200 88620000 87C00000   ...<.......HF2&&  BOPEN1..........
0099A0   87C8C6F2 5B5BC2D6 D7C9C7D5 04008946   000087C0 000087C0 C6F25B5B C2D6D4E3   .HF2&&BOPIGN....  .........F2&&BOMT
0099C0   F0F50200 89EE0000 87C00000 87C8C6F2   5B5BC2D6 D7C5D540 02008922 000087C0   05...........HF2  &&BOPEN ........
0099E0   000087C8 C6F25B5B C2D6D7C5 D5F10200   88620000 87C00000 87C8C6F2 00000000   ...HF2&&BOPEN1..  .........HF2....

LBLTYP   HEX LENGTH IS 0000
--F2--
081000   D7C8C1E2 C55C5C5C 074D3000 000B1718   00000000 00000000 4008107A 0003207A   PHASE***........  ........ .......
081020   D7C8C1E2 A00B16EE 000B32A8 00000018   000B1910 00000049 00000002 0000002E   PHAS............  ................
081040   00000000 00000000 00000000 00000000   00008450 5E0E6CAF 00000000 00000000   ................  ...#............
```

One trace entry (22 bytes)

Phase Name ($$BDMPBC)

SVC (2)

SVC Call Address

Load Address

Entry Address

SYSLOG ID (F2)

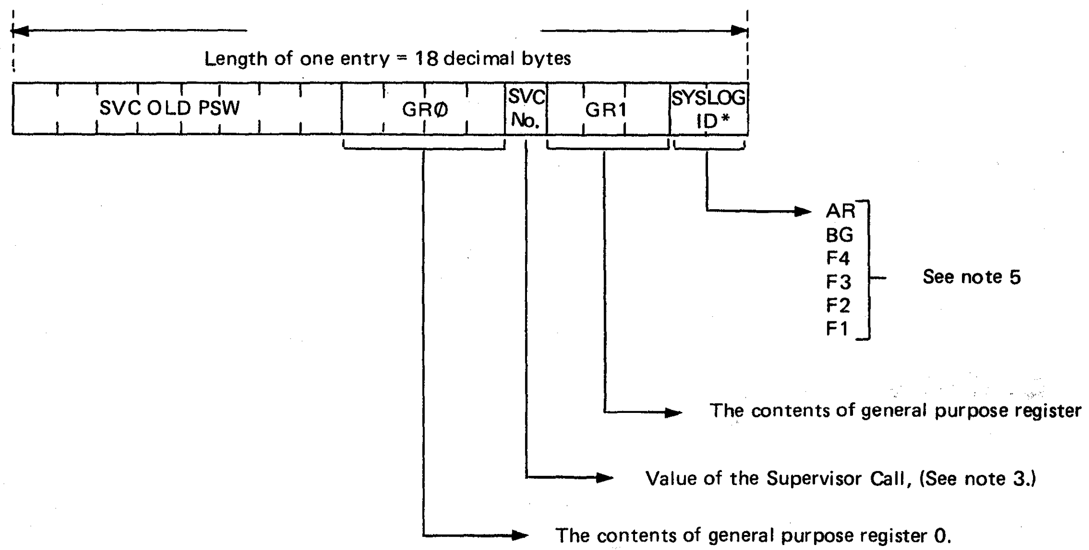
Start of F2 partition

(Ex 18 KT)

An example showing the PD area in a system dump after executing the F/L trace in core-wrap output mode in the PD area.

Generalized Supervisor Call trace

The GSVC trace records SVC interrupts as they occur. All SVCs, or a selected group of SVCs, may be traced. The format of the data recorded in the PD area either in a trace table or, when using the core-wrap output mode, in a rotating buffer, is as follows:



Notes:

1. *If PTO=YES in the FOPT macro, then SVCs issued when the physical transient area is busy are not traced.*
2. *The PSW is described in E-2 of this Section.*
3. *A list of DOS/VS SVCs can be found in Section 4.*
4. *General purpose register usage is described in Section 4.*
5. *The SYSLOG ID is described in appendix B.*

On the occurrence of an event, an entry is generated. By selection of the trace output device, the event can be:

- Recorded on magnetic tape
- Printed on a line printer
- Preserved in the PD area
- Preserved in an alternate area.

When the magnetic tape output is used, the tape must be processed by the PDLIST utility program to provide a formatted output on a line printer.

The modes of output and PDLIST are described under B-1 in this Section.

Tracing Options: The GSVC function provides the following options:

- Trace all SVCs that occur.
- Trace up to six SVCs selectively.
- Eliminate up to six SVCs selectively, and trace all others.
- Trace all partitions.
- Trace up to five partitions selectively.

SVC limiting options are specified by the initializer keywords IGNORE SVC= or TRACE SVC=. All SVC activity is traced if one of these option keywords is not specified. The two keywords are mutually exclusive: when one is specified, the other becomes invalid.

The partition limiting options are specified by the initializer keyword TRACE PARTITION=. This is useful only when the user must run several partitions at once, and does not wish to trace all of them.

When reading the output from this trace routine you may see more SVCs listed than expected. This is because an SVC already traced and recorded may be reset by the supervisor SVC routine, and then re-issued by the program being traced. For example, your program may issue an SVC 0, which is traced. But the channel queue may be full at that point in time, and so the supervisor can not handle the SVC 0. When your program has control again it will issue the SVC 0 which will of course be traced again.

When to use: Use the GSVC trace when a particular SVC issued by a troublesome program is suspected of causing the errors.

The values of registers 0 and 1 are printed on the trace output and these can be important for certain SVCs.

The trace output also shows the current PSW at the time the SVC was issued. Therefore, the instruction and routine issuing the SVC in the program can be located.

The next two examples show the output obtained from a GSVC trace.

```
// ASSGN SYS005,X'2A1'
// MTC REW,SYS005
// EXEC PDLIST
                 SVC old PSW          GRØ   GR1      SVC0
                                                     BG Partition
PSW-070D100000008376 0-000092C0 1-000092C0 N-08 T-BG  PSW-071D200000040B4A 0-00040844 1-00041508 N-02 T-BG
PSW-070D10000000831C 0-00040844 1-000004A0 N-21 T-BG  PSW-070D000000008484 0-00040844 1-00041488 N-1A T-BG
PSW-070D0000000084D6 0-00040844 1-000082E8 N-02 T-BG  PSW-070D000000008304 0-00040844 1-000004A0 N-21 T-BG
PSW-070D000000008422 0-00040844 1-000004A0 N-08 T-BG  PSW-071D20000004084C 0-00040844 1-00041488 N-0E T-BG
PSW-070D0000000082FC 0-00040844 1-000004A0 N-21 T-BG  PSW-070D00000000837C 0-00040844 1-000082E8 N-02 T-BG
PSW-070D000000008306 0-00040844 1-000004A0 N-21 T-BG  PSW-070D2000000084BA 0-00040840 1-000082E8 N-02 T-BG
```

One trace entry from SVC X'21' issued by job in BG

(Ex 19 KT)

An example showing a GSVC trace output as printed on a line printer using PDLIST. (A tape unit was selected as output device for the PDAID.)

```
        TERMTOMS            12/06/73        GSVC trace on all partitions in core-wrap   PAGE  12
                                            output to an alternate area.
03F980  22560000 001B0704 08D8C2C7 071D1000  0004224C 00040CA9 000408D8 C2C7071D
03F9A0  00000004 22560004 0CA90704 08D8C2C7  071D1000 0004224C 00040CA9 000408D8
03F9C0  C2C7071D 00000004 22560004 0CA90704  08D8C2C7 071D1000 0004224C 00040CA9
03F9E0  000408D8 C2C7071D 00000004 22560004  0CA90704 08D8C2C7 071D3000            ...QBG..........  .....QBG.......K
03FA00  00040CA9 00040910 C2C7071D 00000004  22DC0004 0CA90704 0910C2C  One trace  ........BG......  ..........BG....
03FA20  000422EE 0000000B 00040910 C2C7071D  00000004 22C40000 000B07C    entry    ............BG..  .....D.........BG
03FA40  071D3000 000422D2 0000000B 00040910  C2C7071D 00000004 22DC00             .......K........  BG..............
03FA60  0910C2C7 071D3000 000422EE 0000000B  00040910 C2C7071D 000000 04 22C40000 ..BG............  ....BG.......D..
03FA80  000B0704 0910C2C7 071D3000 000422D2  0000000B 00040910 C2C7071D 00000004 .....BG.......K  ........BG......
03FAA0  22DC0000 000B0704                                     0910 C2C7071D     ..........BG....  .............BG..
03FAC0  00000004 22E80000  SVC old PSW        GRØ    GR1      0001B 00040910    .....Y........BG  ................
03FAE0  C2C7071D 30000004                                     42256 0000001B    BG..............  .QBG............
03FB00  070408D8 C2C7071D              SVC0   SYSLOG ID       D0000 00042256    ...QBG..........  .....QBG........
03FB20  00040CA9 070408D8 C2C7071D 10000004  224C0004 0CA90004 08D8C2C7 071D0000 .......QBG......  ..........QBG....
03FB40  00042256 00040CA9 070408D8 C2C7071D  00000004 224C0004 0CA90004 08D8C2C7 ...........QBG..  ..............QBG
03FB60  071D0000 00042256 00040CA9 070408D8  C2C7071D 00000004 20100004 0CA90004 ...............Q  BG..............
03FB80  0820C2C7 071D0000 0004201A 00040CA9  07040820 C2C7071D 30000004 22D20004 ..BG............  ....BG.......K..
03FBA0  0CA90004 0910C2C7 071D3000 000422EE  0000000B 00040910 C2C7071D 00000004 ......BG........  ........BG......
03FBC0  22C40000 000B0704 0910C2C7 071D3000  000422D2 0000000B 00040910 C2C7071D .D.........BG....  ...K........BG..
03FBE0  00000004 22DC0000 000B0704 0910C2C7  071D3000 000422EE 0000000B 00040910 ...............BG  ................
03FC00  C2C7071D 00000004 22C40000 000B0704  0910C2C7 071D3000 000422D2 0000000B BG.......D......  ..BG.......K....
03FC20  00040910 C2C7071D 00000004 22DC0000  000B0704 0910C2C7 071D3000 000422EE ....BG..........  ......BG........
03FC40  0000001B 00040910 C2C7071D 00000004  22E80000 001B0704 0910C2C7 071D3000 .........BG......  .Y........BG....
03FC60  000422EE 0000001B 00040910 C2C7071D  30000004 224C0000 001B0004 08D8C2C7 .............BG..  ..............QBG
03FC80  071D0000 00042256 0000001B 070408D8  C2C7071D 10000004 224C0004 0CA90004 ...............Q  BG..............
03FCA0  08D8C2C7 071D0000 00042256 00040CA9  070408D8 C2C7071D 10000004 224C0004 .QBG............  ...QBG..........
03FCC0  0CA90004 08D8C2C7 071D0000 00042256  00040CA9 070408D8 C2C7071D 10000004 .....QBG........  .......QBG......
03FCE0  224C0004 0CA90004 08D8C2C7 071D0000  00042256 00040CA9 070408D8 C2C7071D .........QBG....  ...........QBG..
03FD00  30000004 22D20004 0CA90004 0910C2C7  071D000                              ........BG  ................
03FD20  C2C7071D 30000004 22EE0000           0910C2C   Entry generated by SVC 7   ........  ..BG.......D....
03FD40  07040910 C2C7071D 30000004 22D20000  000B000                              ......K..  ......BG........
03FD60  0000000B 07040910 C2C7071D 30000004  22EE000                              .BG......  ..........BG....
03FD80  000422C4 0000000B 07040910 C2C7071D  30000004 22D20000 000B0004 0910C2C7 ...D........BG..  .....K........BG
03FDA0  071D0000 000422DC 0000000B 07040910  C2C7071D 30000004 22EE0000 001B0004 ................  BG..............
03FDC0  0910C2C7 071D0000 000422E8 0000001B  07040910 C2C7071D 30000004 22EE0000 ..BG.......Y....  ....BG..........
03FDE0  001B0004 0910C2C7 071D3000 0004224C  0000001B 000408D8 C2C7071D 00000004 ......BG........  ........QBG......
03FE00  22560000 001B0704 08D8C2C7 071D0000  0004224C 00040CA9 000408D8 C2C7071D .........QBG....  ............QBG..
03FE20  00000004 22560004 0CA90704 08D8C2C7  071D1000 0004224C 00040CA9 000408D8 .............QBG  ...............Q
03FE40  C2C7071D 00000004 22560004 0CA90704  08D8C2C7 071D1000 0004224C 00040CA9 BG...             ......
03FE60  000408D8 C2C7071D 00000004 22560004  0CA90704 08D8C2C7 071D3000 000422D2 ...Q  The last two bytes ....K
03FE80  00040CA9 00040910 C2C7071D 00000004  22DC0004 0CA90704 0910C2C7 071D3000 .....  of entries are  G....
03FEA0  000422EE 0000000B 00040910 C2C7071D  00000004 22C40000 000B0704 0910C2C7 .....  recognized by the ....BG
03FEC0  071D3000 000422D2 0000000B 00040910  C2C7071D 00000004 22DC0000 000B0704 .....  SYSLOG ID. (SP    ...D.
03FEE0  0910C2C7 071D3000 000422EE 0000000B  00040910 C2C7071D 00000004 22C40000 ..BG.             BG   .....
03FF00  000B0704 0910C2C7 071D3000 000422D2  0000000B 00040910 C2C7071D 00000004 .....      F1 F2 F3 F4) BG..
03FF20  22DC0000 000B0704 0910C2C7 071D3000  000422EE 0000001B 00040910 C2C7071D .....      or AR
03FF40  00000004 22E80000 001B0704 0910C2C7  071D3000 000422EE 0000001B 00040910 .....
03FF60  C2C7071D 30000004 224C0000 001B0004  08D8C2C7 071D0000 00042256 0000001B BG..............  .QBG
03FF80  070408D8 C2C7071D 10000004 224C0004  0CA90004 08D8C2C7 071D0000 00042256 ...QBG..........  ....
03FFA0  00040CA9 070408D8 C2C7071D 00000004  20100004 0CA90004 0820C2C7 071D0000 .......QBG......  ....
03FFC0  0004201A 00040CA9 07040820 C2C7071D  00000004 224C0004 0CA90004 08D8C2C7 .............BG..  ....
03FFE0  00000000 00000000 00000000 00000000  00000000                            ................  ....
                                                                                                  (Ex 20 KT)
```

An example showing a GSVC trace in core-wrap output in an alternate area. The alternate area is dumped on termination of PDAID and is given back to the main page pool. (The beginning of the alternate area is not shown).

QTAM Trace

This trace records the sequence of SIO instructions issued to channels and devices. The data recorded is similar to that of the I/O trace, but gives more details about the type of I/O interrupt.

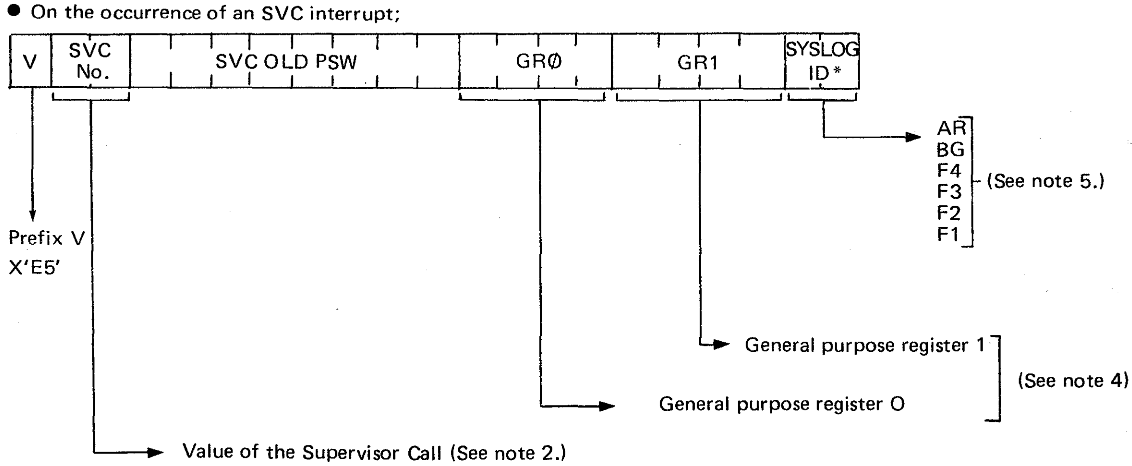
This routine is designed to trace programs running in real mode. However, it can be used to trace virtual mode programs provided the following is considered when reading the trace output:

- If the program being traced is running in real mode the CCB address and the CCW address in the CSW are real addresses.
- If the program being traced is running in virtual mode, the CCB address and the CCW address in the CSW are, respectively, the address of the CCB copy block. (Refer to Section 4 Chapter 13 for a description of CCB and CCW copy blocks.)

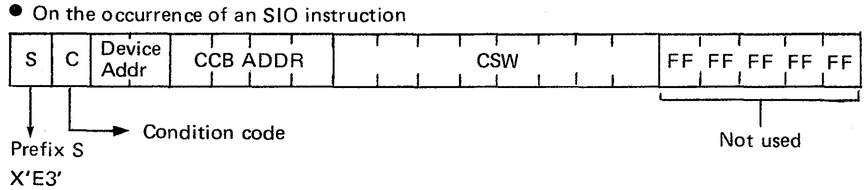
There are three types of trace events and each type is recorded, having a prefix that defines the type.

The data is recorded in the PD area, either in a trace table or, when using the core-wrap mode of output, in a rotating buffer. The format of the data recorded is as follows:

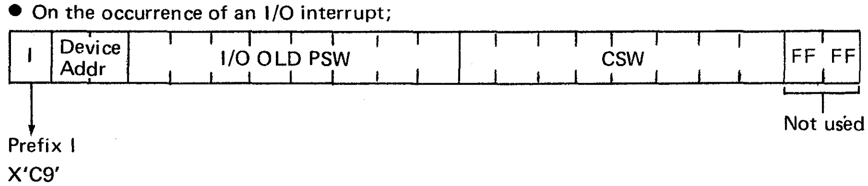
- On the occurrence of an SVC interrupt;



- On the occurrence of an SIO instruction



- On the occurrence of an I/O interrupt;



Notes:

1. *The PSW and CSW is described in E-2 of this Section.*
2. *A list of DOS/VS SVCs can be found in Section 4.*
3. *The CCB is described in Section 4.*
4. *General purpose register usage is described in Section 4*
5. *The SYSLOG ID is described in appendix B.*

Any of these occurrences is referred to as an event. On the occurrence of an event, an entry is generated.

By the selection of the trace output device, the event can be:

- Recorded on magnetic tape
- Preserved in the PD area
- Preserved in an alternate area.

When magnetic tape output is used, the tape must be processed by the PDLIST utility program to provide a formatted output on a line printer.

The modes of output and PDLIST are described under B-1 in this Section.

Tracing Options: The QTAM trace function provides the following options:

- Trace all SVC 0 and 31, SIO, and I/O interrupts.
- Trace SVC 0 and 31, SIO, and I/O interrupts from any three devices.
- Ignore SVC 0 and 31, SIO, and I/O interrupts from and three devices.
- Trace in all partitions
- Selectively trace up to five partitions.

Trace limiting options are specified by the initializer message parameters IGNORE DEVICE= or TRACE DEVICE=. (The device options are invoked by specifying the three devices to be traced or ignored.) All SVC 0 and 31, SIO, and I/O interrupt activity is traced in all partitions of core if one of these options is not specified. They are mutually exclusive: when one is specified, the other becomes invalid.

The partition limiting options are specified by the initializer keyword TRACE PARTITION=.

When to use: Use the QTAM trace to check the sequence of SIO instructions to the channels and devices. Use this trace if you suspect errors in I/O interrupt handling routines or in program routines issuing SVC 0 and SVC 31, or if you suspect errors in the sequence of I/O interrupts being returned from channels or devices. The next two illustrations are examples of output from a QTAM trace.

The next example shows a dump of the real address area containing trace output when the core-wrap output mode in an alternate area is selected.

B-2

PDAID QTAM trace in core-wrap in alternate area of 2K.

```
          TERMKEN            12/06/73                                                                       PAGE   1

GR 0-7    8004080C 00041500 0003F800 00000000   00000050 00041084 00042084 00040FB8
GR 8-F    800400A4 90041086 00000000 182F07F1   00040078 40040084 0003FFF5 08041518
FP REG    00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
CR 0-7    004000FF 0000E640 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
CR 8-F    0000FFFF 00000000 00000000 00000000   00000000 00000000 C2000000 00000200

03F800    C9013007 0F200000 00090C00 003AF00C   000000FF FFE20001 30000039 A000003A    I.............0.  .....S..........
03F820    F00C0000 00FFFFFF FFFFC901 30070C20   00000009 0C00003A C0080000 00FFFFC9    0.........I.....  ...............I
03F840    0130070F 20000000 090C0000 00000400   0000FFFF E2000130 000039A0 00000000    ................  ....S...........
03F860    04000000 FFFFFFFF FFC90130 070F2000   0000090C 000087A0 0C000000 FFFFC900    .........I......  ..............I.
03F880    09070F20 00000009 0C000000 00040000   00FFFFE2 00000900 0078D000 00000004    ................  ...S............
03F8A0    000000FF FFFFFFFF C9000907 0F200000   00090C10 0078D008 000000FF FFC90009    ........I.......  .............I..
03F8C0    070F2000 0000090C 00000000 04000000   FFFFE200 00090000 79380000 00000400    ................  ..S.............
03F8E0    0000FFFF FFFFFFC9 0009070F 20000000   090C1000 79380800 0000FFFF C9000907    .......I........  ............I...
03F900    0F200000 00090C00 00000004 000000FF   FFE20000 09000076 60000000 00040000    ................  .S......-.......
03F920    00FFFFFF FFFFC900 09070F20 00000009   0C100076 60080000 00FFFFC9 0009070F    ......I.........  ....-......I....
03F940    20000000 090C0000 00000400 0000FFFF   E2000009 000076C8 00000000 04000000    ................  S......H........
03F960    FFFFFFFF FFC90009 070F2000 0000090C   100076C8 08400000 FFFFC900 09070F20    .....I..........  ...H. ....I.....
03F980    00000009 0C000000 00040000 00FFFFE2   00000900 00E6C800 00000004 000000FF    ...............S  .....WH.........
03F9A0    FFFFFFFF C9000907 0F200000 00090C10   00E76008 400050FF FFE20001 30000039    ....I...........  .X-. .&..S......
03F9C0    A01000E7 60084000 50FFFFFF FFFFC901   30070C20 00000009 0C00003A C0080000    ...X-. .&.....I.  ................
03F9E0    00FFFFC9 0130070F 20000000 090C0000   00000400 0000FFFF E2000130 000039A0    ...I............  ........S.......
03FA00    00000000 04000000 FFFFFFFF FFC90130   070F2000 0000090C 00003AF0 0C000000    .............I..  ...........0....
03FA20    FFFFE200 01300000 39A00000 3AF00C00   0000FFFF FFFFFFC9 0130070C 20000000    ..S..........0..  .......I........
03FA40    090C0000 3AC00800 0000FFFF C9013007   0F200000 00090C00 00000004 000000FF    ............I...  ................
03FA60    FFE20001 30000039 A0000000 00040000   00FFFFFF FFFFC901 30070F20 00000009    .S..............  ......I.........
03FA80    0C000087 A00C0000 00FFFFE2 00013000   0039A000 0087A00C 000000FF FFFFFFFF    ...........S....  ................
03FAA0    C9013007 0C200000 00090C00 003AC008   000000FF FFC90130 070F2000 0000090C    I...............  .....I..........
03FAC0    00000000 04000000 FFFFE200 01300000   39A00000 00000400 0000FFFF FFFFFFC9    ..........S.....  ...............I
03FAE0    0009070F 20000000 090C0000 00000400   0000FFFF C9013007 0F200000 00090C00    ................  ....I...........
03FB00    003AF00C 000000FF FFE20001 30000039   A000003A F00C0000 00FFFFFF FFFFC901    ..0......S......  ....0.........I.
03FB20    30070F20 00000009 0C00003A F00C0000   00FFFFE2 00013000 0039A000 003AF00C    ............0...  ...S..........0.
03FB40    000000FF FFFFFFFF C9013007 0C200000   00090C00 003AC008 000000FF FFC90130    ........I.......  .............I..
03FB60    070F2000 0000090C 00000000 04000000   FFFFE200 01300000 39A00000 00000400    ................  ..S.............
03FB80    0000FFFF FFFFFFC9 0130070F 20000000   090C0000 87A00C00 0000FFFF E2000130    .......I........  ............S...
03FBA0    000039A0 000087A0 0C000000 FFFFFFFF   FFC90130 070C2000 0000090C 00003AC0    ................  .I..............
03FBC0    08000000 FFFFC901 30070F20 00000009   0C000000 00040000 00FFFFE2 00013000    ......I.........  ...........S....
03FBE0    0039A000 00000004 000000FF FFFFFFFF   C9013007 0F200000 00090C00 003AF00C    ................  I.............0.
03FC00    000000FF FFE20001 30000039 A000003A   F00C0000 00FFFFFF FFFFC901 30070F20    .....S..........  0.........I.....
03FC20    00000009 0C00003A F00C0000 00FFFFE2   00013000 0039A000 003AF00C 000000FF    ........0......S  ..........0.....
03FC40    FFFFFFFF C9013007 0C200000 00090C00   003AC008 000000FF FFC90130 070F2000    ....I...........  .........I......
03FC60    0000090C 00000000 04000000 FFFFE200   01300000 39A00000 00000400 0000FFFF    ..............S.  ................
03FC80    FFFFFFC9 0130070F 20000000 090C0000   87A00C00 0000FFFF E2000130 000039A0    ...I............  ........S.......
03FCA0    000087A0 0C000000 FFFFFFFF FFC90130   070C2000 0000090C 00003AC0 08000000    .............I..  ................
03FCC0    FFFFC901 30070F20 00000009 0C000000   00040000 00FFFFE2 00013000 0039A000    ..I.............  .......S........
03FCE0    00000004 000000FF FFFFFFFF C9013007   0F200000 00090C00 003AF00C 000000FF    ............I...  ..........0.....
03FD00    FFE20001 30000039 A000003A F00C0000   00FFFFFF FFFFC901 30070F20 00000009    .S..........0...  ......I.........
03FD20    0C000087 A00C0000 00FFFFE5 0000070D   00000000 83840004 10700000 86C8C2C7    ...........V....  .............HBG
03FD40    E20002A2 0000E6C8 000087A0 0C0                                     90C    S.....WH........  .....I..........
03FD60    0000E838 0C000012 FFFFE200 013                                     FC9    ..Y.......S.....  ....Y..........I
03FD80    0130070F 20000000 090C0000 3AF                                     AF0    ..............0.  ...S...........0
03FDA0    0C000000 FFFFFFFF FFC90130 070F                                    200    .........I......  ..............S.
03FDC0    01300000 39A00000 87A00C00 000                                            ................  ...I............
03FDE0    3AF00C00 0000FFFF E2000130 000                                            .0......S.......  ...0.........I..
```

I type entry

S type entry

SVC 0

SVC old PSW

GR0 GR1

SYSLOG ID = BG

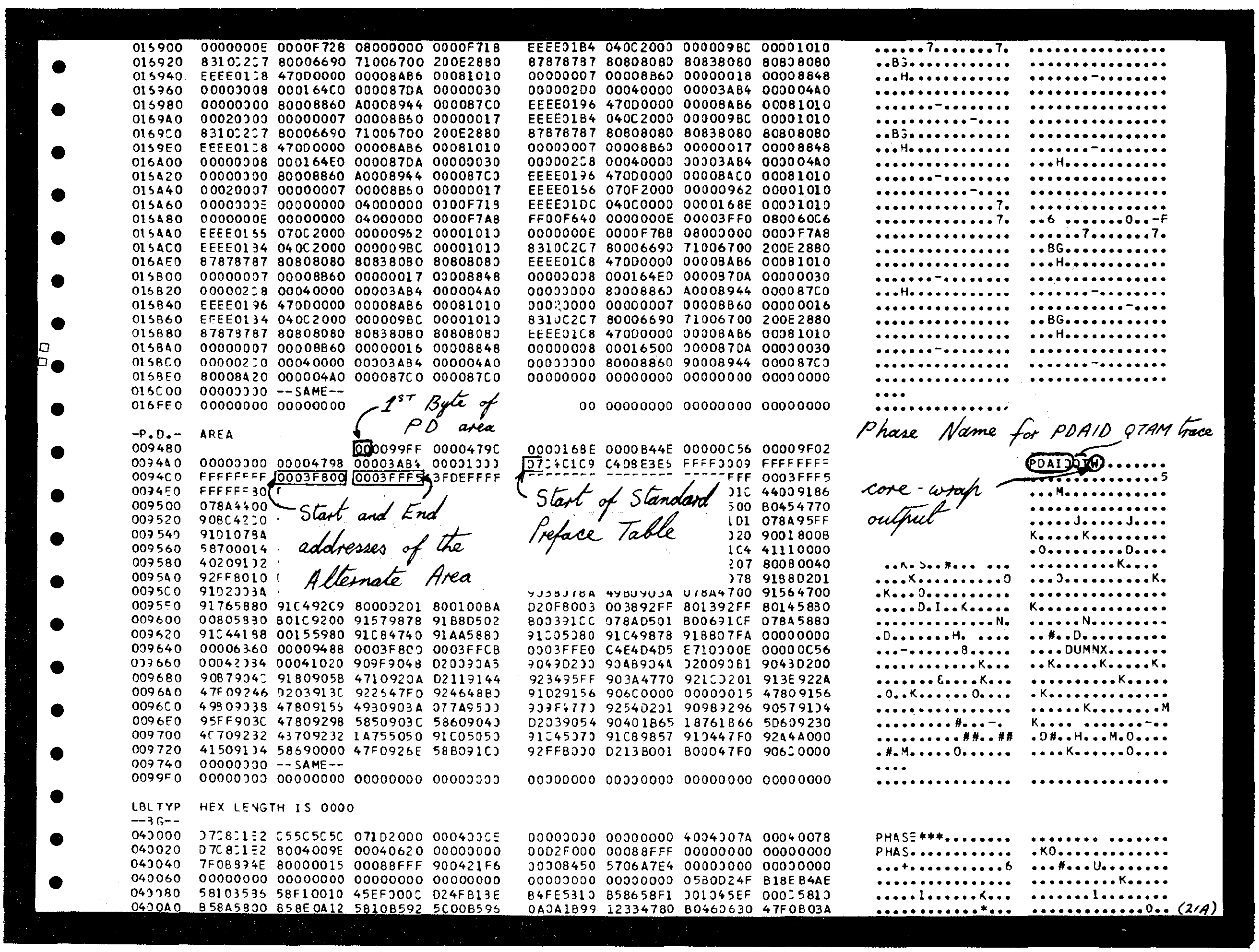
V type entry

```
          TERMKEN            12/06/73                                                                       PAGE   2

03FE00    070F2000 0000090C 000087A0 0C000000   FFFFE200 01300000 39A00000 87A00C00    ................  ..S.............
03FE20    0000FFFF FFFFFFC9 0130070F 20000000   090C0000 3AF00C00 0000FFFF E2000130    .......I........  .....0......S...
03FE40    000039A0 00003AF0 0C000000 FFFFFFFF   FFC90130 070F2000 0000090C 000087A0    .......0........  .I..............
03FE60    0C000000 FFFFE500 00071D00 00000415   42A00410 6C000414 88C2C7E2 0002A200    ......V.........  .........BGS....
03FE80    00E6C800 0087A00C 000000FF FFFFFFFF   C902A207 0F200000 00090C10 00E8380C    .WH.............  I............Y..
03FEA0    000001FF FFE50000 071D0000 00041542   A004106C 00041488 C2C7E200 02A20000    .....V..........  ........BGS.....
03FEC0    E6C81000 E8380C00 0001FFFF FFFFFFC9   02A2070F 20000000 090C1000 E7600C00    WH..Y..........I  ............X-..
03FEE0    0001FFFF E5000007 1D000000 041542A0   04106C00 041488C2 C7E20002 A20000E6    ....V...........  .......BGS.....W
03FF00    C81000E7 600C0000 01FFFFFF FFFFC902   A2070F20 00000009 0C1000E8 380C0000    H..X-.........I.  ...........Y....
03FF20    01FFFFE2 00013000 0039A010 00E8380C   000001FF FFFFFFFF C9013007 0F200000    ...S.........Y..  ........I.......
03FF40    00090C00 003AF00C 000000FF FFE20001   30000039 A000003A F00C0000 00FFFFFF    ......0......S..  ........0.......
03FF60    FFFFC901 30070F20 00000009 0C00003A   F00C0000 00FFFFE2 00013000 0039A000    ..I.............  0......S........
03FF80    003AF00C 000000FF FFFFFFFF C9013007   0F200000 00090C00 003AF00C 000000FF    ..0.........I...  ..........0.....
03FFA0    FFE20001 30000039 A000003A F00C0000   00FFFFFF FFFFC901 30070F20 00000009    .S..........0...  ......I.........
03FFC0    0C00003A F00C0000 00FFFFE2 00013000   0039A000 003AF00C 000000FF FFFFFFFF    ....0......S....  ......0.........
03FFE0    00000000 00000000 00000000 00000000   00000000 00000000                      ................  ........
```

(Ex21KT)

An example showing a dump of the alternate area used for a QTAM trace in core-wrap output mode.



```
015900  0000000E 0000F728 08000000 0000F718   EEEE01B4 040C2000 000009BC 00001010   ......7.......7.  ................
016920  8310C2C7 80006690 71006700 200E2880   87878787 80808080 80838080 80808080   ..BG............  ................
015940  EEEE01C8 470D0000 00008AB6 00081010   00000007 00008B60 00000018 00008848   ...H............  .......-........
015960  00000008 000164C0 000087DA 00000030   000002D0 00040000 00003AB4 000004A0   ................  ................
016980  00000000 80008860 A0008944 000087C0   EEEE0196 470D0000 00008AB6 00081010   .......-........  ................
0169A0  00020000 00000007 00008B60 00000017   EEEE01B4 040C2000 000009BC 00001010   ...........-....  ................
0169C0  8310C2C7 80006690 71006700 200E2880   87878787 80808080 80838080 80808080   ..BG............  ................
0159E0  EEEE01C8 470D0000 00008AB6 00081010   00000007 00008B60 00000017 00008848   ...H............  .......-........
016A00  00000008 000164E0 000087DA 00000030   000002C8 00040000 00003AB4 000004A0   ................  ...H............
015A20  00000000 80008860 A0008944 000087C0   EEEE0196 470D0000 00008AC0 00081010   .......-........  ................
015A40  00020007 00000007 00008B60 00000017   EEEE0156 070F2000 00000962 00001010   ...........-....  ................
015A60  0000000E 00000000 04000000 0000F718   EEEE01DC 040C0000 0000168E 00001010   ..............7.  ................
015A80  0000000E 00000000 04000000 0000F7A8   FF00F640 0000000E 00003FF0 080060C6   ..............7.  ..6 .......0..-F
015AA0  EEEE0155 070C2000 00000962 00001010   0000000E 0000F7B8 08000000 0000F7A8   ................  ......7.......7.
015AC0  EEEE0134 040C2000 000009BC 00001010   8310C2C7 80006690 71006700 200E2880   ................  ..BG............
016AE0  87878787 80808080 80838080 80808080   EEEE01C8 470D0000 00008AB6 00081010   ................  ...H............
015B00  00000007 00008B60 00000017 00008848   00000008 000164E0 000087DA 00000030   .......-........  ................
016B20  000002C8 00040000 00003AB4 000004A0   00000000 80008860 A0008944 000087C0   ...H............  .......-........
015B40  EEEE0196 470D0000 00008AB6 00081010   00000000 00000007 00008B60 00000016   ................  ............-...
015B60  EFEE0134 040C2000 000009BC 00001010   8310C2C7 80006690 71006700 200E2880   ................  ..BG............
015B80  87878787 80808080 80838080 80808080   EEEE01C8 470D0000 00008AB6 00081010   ................  ...H............
015BA0  00000007 00008B60 00000016 00008848   00000008 00016500 000087DA 00000030   .......-........  ................
015BC0  000002C0 00040000 00003AB4 000004A0   00000000 80008860 90008944 000087C0   ................  .......-........
015BE0  80008A20 000004A0 000087C0 000087C0   00000000 00000000 00000000 00000000   ................  ................
016C00  00000000 --SAME--                                                           ....
016FE0  00000000 00000000                              00 00000000 00000000 00000000   ................

-P.D.-  AREA
009480                    000099FF 0000479C   0000168E 0000B44E 00000C56 00009F02
0094A0  00000000 00004798 00003AB4 00001000   D7C4C1C9 C4D8E3E5 FFFF0009 FFFFFFFF                      PDAIDQTH........
0094C0  FFFFFFFF 0003F800 0003FFF5 3FDEFFFF   -------- -------- -----FFF 0003FFF5                      ...............5
0094E0  FFFFFF30 F                                                    01C 44009186                      ...M............
009500  078A4400                                                      500 80454770                      ................
009520  908C4200                                                      1D1 078A95FF                      .....J.....J....
009540  9101078A                                                      020 90018008                      K.....K.........
009560  58700014                                                      1C4 41110000                      .0.........D....
009580  40209102                                                      207 80080040   ..K.S..#... ...    ..........K....
0095A0  92FF8010                                                      078 91880201   ....K..........0   ...0...........K.
0095C0  9102003A                              9038078A 49B0903A 078A4700 91564700   .K...0..........  ................
0095E0  91765880 91C492C9 80000201 800100BA   D20F8003 003892FF 801392FF 801458B0   .....D.I..K.....  K...............
009600  00805830 801C9200 91579878 91B8D502   B00391CC 078AD501 B00691CF 078A5880   ..............N.  ......N.........
009620  91C44188 00155980 91C84740 91AA5880   91C05080 91C49878 918807FA 00000000   .D.......H. ....  ..#..D..........
009640  00006360 00009488 0003F800 0003FFCB   0003FFE0 C4E4D4D5 E710000E 00000C56   ...-......8.....  ....DUMNX.......
009660  00042034 00041020 909F9048 D20090A5   90490200 90AB904A 020090B1 90430200   .............K...  ..K.....K.....K.
009680  90879040 9180905B 4710920A D2119144   923495FF 903A4770 921C0201 913E922A   .......&.....K...  ..........K.....
0096A0  47F09246 D203913C 922547F0 924648B0   91D29156 906C0000 00000015 47809156   .0..K.......0....  .K..............
0096C0  49809038 47809155 4930903A 077A9500   909F4770 92540201 90989296 90579104   ................  .......K........M
0096E0  95FF903C 47809298 5850903C 58609040   D2039054 90401B65 18761B66 5D609230   ..........#...-.  K.... .......-..
009700  4C709232 43709232 1A755050 91C05050   91C45070 91C89857 910447F0 92A4A000   ...........##..##  .D#..H...M.0....
009720  41509104 58690000 47F0926E 58B091C0   92FFB000 D213B001 B00047F0 90600000   .#.M.....0......  ....K.......0....
009740  00000000 --SAME--                                                           ....
0099F0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................  ................

LBLTYP  HEX LENGTH IS 0000
--BG--
040000  D7C8C1E2 C55C5C5C 071D2000 000400CE   00000030 00000000 4004007A 00040078   PHASE***........  ........ ......
040020  D7C8C1E2 8004009E 00040620 00000000   00D2F000 00088FFF 00000000 00000000   PHAS............  .K0.............
040040  7F08894E 80000015 00088FFF 900421F6   00008450 5706A7E4 00000000 00000000   ...+...........6  ...#...U........
040060  00000000 00000000 00000000 00000000   00000000 00000000 05B0D24F B18EB4AE   ................  ...........K.....
040080  58103535 58F10010 45EF000C D24FB13E   B4FE5310 B58658F1 001045EF 00005810   .....1.......K...  ........1.......
0400A0  B58A5800 858E0A12 5810B592 5C00B596   0A0A1B99 12334780 B0460630 47F0B03A   ............*...  ..............0..  (21A)
```


The example above shows part of a system dump output. By examination of the PDAREA printed in the dump (when PDAID is supported by the system) it can be seen that a QTAM trace was active when the dump was executed. From the PDAID phase name in the PD area, the output mode for the trace was core-wrap. However, no trace entries are seen in the PD area, which indicates that the an alternate area had been specified for the trace entries. The start address of the alternate area is contained at displacement decimal 20 from the start of the PD standard preface table. (The standard preface table starts at the PDAID phase name.)

B-2

Job Name

System dump output – QTAM trace to PD area of 1400 bytes – trace on job in F2

```
         DUMPEXS             10/05/73                                                                      PAGE  50

016BC0   000005A8 00081000 00003AE4 00004360    00000000 80008860 90008944 000087C0    ..........U...-  .......-........
015BE0   000B207A D7C8C1E2 800B136C 000B32A8    00000000 00000000 00000000 00000000    ....PHAS.......  ................
016C00   00000000 --SAME--                                                             ....
016FE0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000    ...............  ................

-P.D.-   AREA
009480                     000099FF 0000479C    0000168E 0000B44E 00000C56 00009F02           .......  ........+.......
0094A0   00000000 00004798 00003AE4 00001000    D7C4C1C9 C4D8E3E6 FFFF0009 FFFFFFFF    ..........U....  PDAIDQTW........
0094C0   FFFFFFFF FFFFFFFF FFFFFFFF 3FDEFFFF    FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF    ...............  ................
0094E0   FFFFFF30 FFFFFFFF 47F090EC 47F09132    470091D4 58B00080 58B0B01C 44009186    .........0...0..  ...M............
009500   078A4400 918E078A 9500008B 4780908E    951F008B 077A58B0 00809500 B0454770    ...............  ................
009520   90BC4200 91D195FF 91D1078A 95FF91D1    078A95FF 91D1078A 95FF91D1 078A95FF    .....J...J.....J  .....J.....J....
009540   91D1078A 907891B8 588091C4 92E58000    D2018001 008AD207 80030020 9001800B    .J.........D.V..  K.....K.........
009560   58700014 4870705A 1A7CD201 80137002    47F0919A 907891B8 588091C4 41110000    ..........K.....  .0.........D....
009580   40209102 92E28000 50180004 40280002    187A8870 00184278 0001D207 80080040     ..K.S..#... ...  ..........K....
0095A0   92FF8010 D2038011 80109430 800192F0    915747F0 913C9200 91579078 91B8D201    ....K..........0  ...0...........K.
0095C0   91D2003A 47F09156 49B09036 078A49B0    9038078A 49B0903A 078A4700 91564700    .K...0..........  ................
0095E0   917658B0 91C492C9 8000D201 800100BA    D20F8003 003892FF 801392FF 801458B0    .....D.I..K.....  K...............
009600   008058B0 B01C9200 91579878 91B8D502    B00391CC 078AD501 B00691CF 078A5880    ..............N.  ......N.........
009620   91C441B8 00155980 91C84740 91AA5880    91C05080 91C49878 91B807FA 00000000    .D.......H. ....  ..#..D..........
009640   00006360 00009488 0000965C 0000993B    000099E3 C4E4D4D5 E740000E C9000E07    ...-.......*....  ...TDUMNX ..I...
009660   0F200000 00096200 00000004 000000FF    FFE20000 0E0000F8 C8000000 00040000    ................  .S.....8H.......
009680   00FFFFFF FFFFC900 0E070C20 00000009    620000F7 B8080000 00FFFFE5 0000470D    ......I.........  ...7.......V....
0096A0   20000000 09620000 F9200800 0000FFFF    C9000E07 0F200000 00096200 00000004    .........9.......  I...............
0096C0   008B60C6 F2C9000E 070F2000 00000962    00000000 04000000 FFFFE200 000E0000    ..F2I...........  ...........S.....
0096E0   F8C80000 00000400 0000FFFF FFFFFFC9    000E070C 20000000 09620000 F7B80800    8H.............I  ...........7....
009700   0000FFFF E5000047 0D000000 008AB600    00000700 008B60C6 F2C9000E 070F2000    ....V...........  .......F2I......
009720   00000962 00000000 04000000 FFFFE200    000E0000 F8C80000 00000400 0000FFFF    ..............S.  ....8H..........
009740   FFFFFFC9 000E070C 20000000 09620000    F9200800 0000FFFF E5000047 0D000000    ...I............  9.......V.......
009760   008AB600 00000700 008B60C6 F2C9000E    070F2000 00000962 00000000 04000000    ...........F2I..  ................
009780   FFFFE200 000E0000 F8C80000 00000400    0000FFFF FFFFFFC9 000E070C 20000000    ..S.....8H......  ........I.......
0097A0   09620000 F7B80800 0000FFFF E5000047    0D000000 008AB600 00000700 008B60C6    ....7.......V...  ...............-F
0097C0   F2C9000E 070F2000 00000962 00000000    04000000 FFFFE200 000E0000 F8C80000    2I..............  ......S......8H..
0097E0   00000400 0000FFFF FFFFFFC9 000E070C    20000000 09620000 F9200800 0000FFFF    ...........I....  .........9.......
009800   E5000047 0D000000 008AB600 00000700    008B60C6 F2C9000E 070F2000 00000962    V...............  ..F2I...........
009820   00000000 04000000 FFFFE200 000E0000    F8C80000 00000400 0000FFFF FFFFFFC9    ..........S.....  8H.............I
009840   000E070C 20000000 09620000 F7B80800    0000FFFF E5000047 0D000000 008AB600    ............7...  ....V...........
009860   00000700 008B60C6 F2C9000E 070F2000    00000962 00000000 04000000 FFFFE200    ......-F2I......  ...............S.
009880   000E0000 F8C80000 00000400 0000FFFF    FFFFFFC9 000E070C 20000000 09620000    ....8H..........  ...I............
0098A0   F9200800 0000FFFF E5000047 0D000000    008AB600 00000700 008B60C6 F2C9000E    9.......V.......  ...........F2I..
0098C0   070F2000 00000962 00000000 04000000    FFFFE200 000E0000 F8C80000 00000400    ................  ..S......8H......
0098E0   00040000 00FFFFE2 00000E00 00F8C800    00000004 000000FF FFFFFFFF C9000E07    .......S......8H.  .............I...
009900   0C200000 00096200 00F92008 000000FF    FFE50000 470D0000 00008AB6 00000007    .........9.......  .V..............
009920   00008B60 C6F2C900 0E070F20 00000009    62000000 00040000 00FFFFE2 00000E00    ...F2I..........  .............S....
009940   00F8C800 00000004 000000FF FFFFFFFF    C9000E07 0C200000 00096200 00F7B808    .8H.............  I.............7..
009960   000000FF FFE50000 470D0000 00008AB6    00000007 00008B60 C6F2C900 0E070F20    .....V..........  ........F2I.....
009980   00000009 62000000 00040000 00FFFFE2    00000E00 00F8C800 00000004 000000FF    ...............S  ......8H.........
0099A0   FFFFFFFF C9000E07 0C200000 00096200    00F92008 000000FF FFE50000 47000000    ....I...........  .9.......V.......
0099C0   00008AB6 00000007 00008B60 C6F2C900    0E070F20 00000009 62000000 00040000    ............-F2I.  ................
0099E0   00FFFF88 C6F25BD1 D6C2C3E3 D3C7040B    1958000B 2888000B 2888C6F2 00000000    ....F2&JOBCTLG..  ...........F2....

LBLTYP   HEX LENGTH IS 0000
--F2--
081000   07C8C1E2 C55C5C5C 074D0000 000B146E    00000001 00000013 400B107A 000B207A    PHASE***........  ........ .......
```

Phase Name of QTAM trace

1[ST] byte of F2 partition save area

QTAM trace entries indentified by SYSLOG ID

(Ex 2/BKT)

This example shows part of a system dump output that by examination of the PD area indicates QTAM trace entries in the PD area. Compare this example with the previous one and note the difference between the information contained in the PD area.

Trace Routines

PDAIDS

The PD area

The PD area is located in the supervisor and consists of four separate parts described below and shown in

1. PD Address Table
   This table is built up during system generation if the system is to support PDAIDS. It contains the addresses of the supervisor hooks that provide the interface between the PDAID routines and the supervisor.

2. PD Standard Preface Table
   This table is built up by the PDAID initializing phase, and is used by the PDAID event handling routines.

3. PDAID Event Handling Area
   This area is occupied by the PDAID event handling routines specified by the type of trace requested by the operator.

4. PD Buffer Area.
   This area is used in the following two ways:

   When core-wrap output mode in the PD area is specified it is used as a rotating buffer which preserves events (trace entries). PDAID event handling routines use this area as temporary storage for events. This storage area is called the trace table. Data is transferred from this table to an output area, which is either printed out or dumped on a tape unit, depending on the output device selected for the trace routine.

Locating the PD area

The start address of the PD area can be located by:

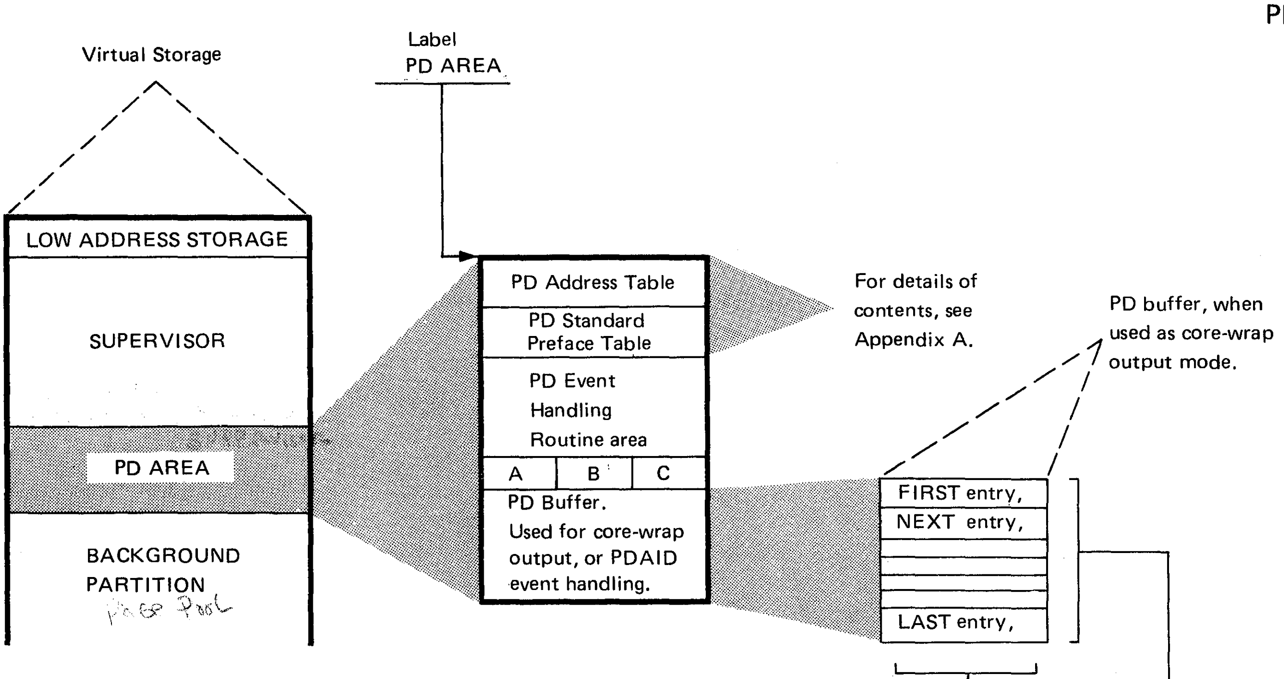
1. Using any dump containing the supervisor area to find the address of SYSCOM (system communication region) in bytes 80 to 83 of low address storage. (See E-2 in this Section.)

   The address contained in bytes X'48' to '4B' (label PDARPTR) of SYSCOM contains the address of the PD area.

2. Using the supervisor listings to find the address of the label PDAREA. This label is the name given to the first byte of the PD area.

Dumping the PD area

The easiest method is to use the PD AREA operand of the DUMP command. (See A-1 in this Section.) Alternatively, any dump of real storage that includes the supervisor area will also include the PD area.



Note:
See Section 4 for details about the supervisor organization.

	A	B	C	D	
Trace type.	**Address of FIRST entry, (SLOT1) is found at location**	**Address of NEXT entry is found at location; (See Note 1.)**	**Address of LAST entry, (WRAPADR) is found at location; (See also note 2.)**	**Length of one entry in bytes;**	**Maximum number of entries in the minimum PD area of 1200 bytes (dec)**
Input/output	PDAREA + X'198'	B = A + 4	C = A + 8	18 (dec)	42
Fetch/load	PDAREA + F4	B = A + 4	C = A + 8	22 (dec)	42
GSVC	PDAREA + X'114'	B = A + 4	C = A + 8	18 (dec)	50
QTAM	PDAREA + '1CØ'	B = A + 4	C = A + 8	21 (dec)	35

Notes:

1. *For the I/O trace*
   NEXT — address of the next available slot in the save area. (Because NEXT is filled with unchecked new information, it may contain either the oldest entry in the table, or the most recent activity of a device not being traced. If the latter is the case, ignore the entry.)

   For the F/L, GSVC, and QTAM trace
   NEXT — address of the next available slot in the save area. (Because NEXT is filled with unchecked new information, it may contain either the oldest entry in the table, or the SVC number and calling address from a partition not being traced. If the latter is the case, ignore the entry.)

2. *When LAST entry is filled, next becomes first and the buffer is overwritten by new entries.*

Table B-3 Trace entry locations and lengths for core-wrap output mode in the PD area.

PDAIDS

Initiating the PDAID trace routines

You can initiate PDAID trace routines by using standard DOS/VS job control languages from either SYSLOG or SYSIPT. The statement

// EXEC PDAID

causes the main phase (PDAID) to be loaded at the address of the initiating partition. Control is given to the PDAID for further specifications to indicate the type of trace to be performed.

The options and control statements for the trace routines may be entered through SYSLOG or through the device assigned to SYSIPT.

If a card reader is used as SYSIPT, the card deck must be punched as follows:

Entries may be punched one-per-card, or as multiple entries (separated by commas) in a single card. An entry may not be split between two cards. All 80 columns of a card may be used, but a card is terminated either by the first blank following an entry, or by a GO entry. The last card must be followed by a /* CARD.

Note: If an incorrect parameter is read from SYSIPT, corrections are requested on SYSLOG.

When the initializing phase (PDAID) has been loaded, the following message is issued on SYSLOG:

4C10D PDAID=

The operator must respond to this message with one of the following:

IT Specifies an I/O Trace (See note 1.)
FT Specifies an F/L Trace (See note 1.)
GT Specifies a GSVC Trace (See note 1.)
QT Specifies a QTAM Trace (See note 1.)
TD Specifies the Transient Dump
(refer to A-4 in this Section)
XX Terminates the PDAID presently running
Pressing the END or ENTER key indicates that PDAID control statements are entered through SYSIPT (See note 2.)

Notes:
1. *When IT, FT, GT, or QT is specified, the operator must provide additional PDAID control statements through SYSLOG.*
2. *The END response is valid only for SYSLOG and cannot be used as a SYSIPT operand.*
3. *Multiple operands or operator responses to PDAID control statements for traces with a variable number of functions (such as ignoring SVCs) are not allowed. Repeat each parameter with each variable). Repeat each message until either the maximum number of variables is reached or an END response is given.*
4. *GO terminates the PDAID control input, and the default is taken for any PDAID options that are not specified. When you use SYSIPT, GO should be the last parameter, and it has no operand associated with it. A /* card must follow the GO operand.*

Selecting the output mode

Selection of an output device:

The PDAID message/parameter OUTPUT DEVICE= permits the selection of an output device. Specify the device by channel and unit, not by symbolic unit. If an output device is specified, PDAID checks the address against the supervisor PUB and selects the appropriate phase for the unit type (tape or printer). If the output is to be magnetic tape, you must use the PDLIST program after tracing is complete to obtain a printout of the tape.

Selection of core-wrap mode: If an output device is not specified, core-wrap mode is assumed. The event trace table (see Table B-3) is in the PD buffer in PD area. The number of events (trace entries), contained in this area depends on its size as generated at system generation time with the option of the FOPT macro. PD=YES or 1400 is the minimum, and 10,240 is the maximum that can be selected.

The table shown in the previous illustration lists the maximum number of events that can be preserved in this area, for each of the four trace routines. If core-wrap mode is selected, an alternate area can be used.

Specifying an Alternate Area

An alternate area may be specified for core-wrap output through the message/ parameter AAA= (alternate area address). AAA= and OUTPUT DEVICE= are mutually exclusive: when one is specified, the other cannot be used. The operator specifies an alternate area by responding to AAA= with nk.

n should be an even integer but if an odd integer is specified, n+1 is assumed. n specifies the number of thousand (1024) bytes to be allocated to the alternate area, which is taken from the main page pool.

After AAA=nk has been entered, one of four messages is printed on SYSLOG:

1. If the requested size of the alternate area is accepted, the message is

   4C50E ADDRESS OF AAA= xxxxxx

2. If space could not be allocated from the main page pool, the message is

   4C52E NO SPACE AVAILABLE FOR AAA. PDAID IS TERMINATED

   The size of the page pool must be increased and the PDAID must be re-initialized.

3. If the space requested is larger than the space that can be allocated from the page pool, the message is

   4C51D SIZE OF AAA=nK, ADDRESS OF AAA=XXXXXX. END/CANCEL

   If the space allocated is sufficient, the operator need only press the END/ENTER key. However, if the space allocated is not sufficient, the operator must respond with CANCEL, and the size of the page pool must be increased before re-initializing the PDAID.

4. If a second or duplicate request is made for an alternate area, or if a request is made for a PDAID using an alternate area while any SDAID function is running, the second request is automatically terminated, and the message is 4C70E
   4C70E DUPLICATE REQUEST FOR PDAID AND/OR SDAID
   The above message is also issued if a second or duplicate request is made for SDAIDS.

Dumping the alternate area

The contents of the alternate area is automatically dumped on the device assigned to SYSLST upon termination of the PDAID. (See "Terminating core-wrap in an alternate area" for details.) However, if a dump of an alternate area is required without terminating the PDAID, use the xxxxxx, xxxxxx operand of the DUMP command. (See A-1 in this Section for details.)

Note: If this command is used, the trace output will include the fetch and execute of the DUMP transient. Specify the address of AAA in the first operand of the command, and calculate the value of the second operand from the value of nk, given in the message 4C51D or specified in the message 4C27D during trace initialization.

Use Table B-3 and the dump to locate the oldest and newest trace entries.

PDAID error messages

PDAID routines issue error messages on SYSLOG if incorrect or duplicate parameters are specified, or if selected output devices are not ready. The PDAID error messages together with recommended actions for operators and programmers are listed in the *DOS/VS Messages* manual.

The following list is a table of options and control statements for executing the trace routines. The statements in the table are shown in the sequence in which they must be used. Five flowcharts follow the table of options. These flowcharts show how to execute the trace routines.

Six examples of initiating trace routines via SYSIPT, followed by five examples of initiating via SYSLOG, immediately follow the last of those flowcharts.

PDAIDS

Initializing PDAIDS

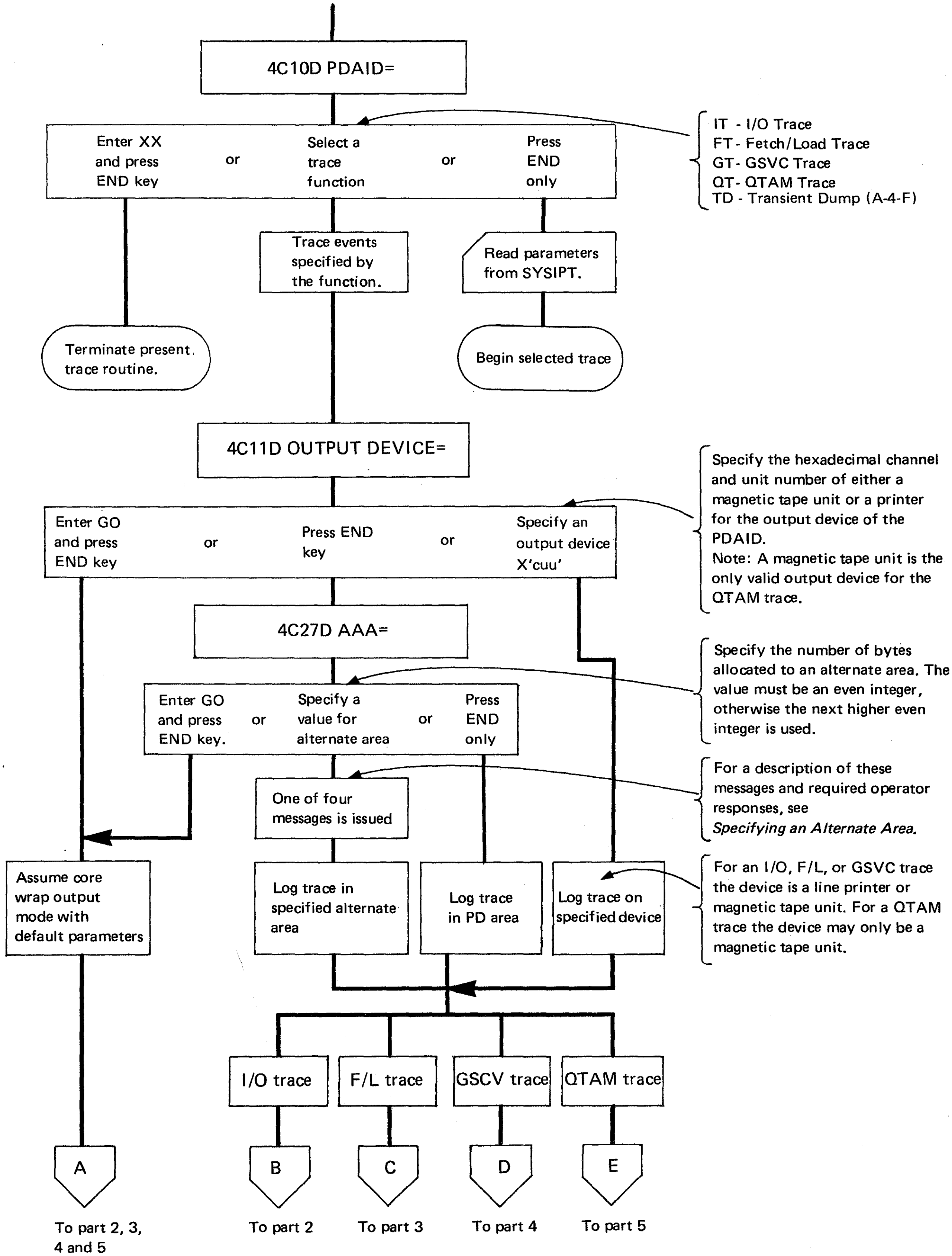
SYSLOG Message / SYSIPT Parameter	SYSLOG Response / SYSIPT Operand	Meaning	Default
PDAID =	{ FT, GT, IT, QT, TD, XX, END }	FT - Fetch/Load Trace GT - GSVC Trace IT - I/O Trace QT - QTAM Trace TD — Transient Dump, refer to A-4 in this Section XX - Terminate present PDAID function. END - Additional PDAID control input through SYSIPT (See note 5)	None.
OUTPUT DEVICE = (see note 3)	{ cuu, X'cuu', END, GO }	Specify the hexadecimal channel and unit number of either a magnetic tape unit or a printer for the output device of the PDAID. (see note 6)	Core-wrap mode. (See note 7)
AAA = (see note 3)	{ nK, END, GO }	The parameter nK specifies the number of bytes to be allocated as alternate address area. This area will be allocated storage from the main page pool. The value n must be an even integer. If it is not an even integer, (n+1) K is allocated.	Core-wrap mode using PD area
TRACE PARTITION= (Valid for Fetch/Load, SVC, and QTAM Trace)	[{ SP, BG, F4, F3, F2, F1, END, GO }]	SP - Supervisor BG - Background F4 - Foreground 4 F3 - Foreground 3 F2 - Foreground 2 F1 - Foreground 1 (see note)	Trace all partitions and the supervisor.
IGNORE DEVICE = (See notes 2 and 7)	[{ cuu, X'cuu', END, GO }]	Specify the hexadecimal channel and unit number of the device to be ignored by the I/O and QTAM trace. A maximum of 3 may be specified.	Trace all devices.
TRACE DEVICE= (See notes 2 and 7)	[{ cuu, X'cuu', END, GO }]	Specify the hexadecimal channel and unit number of the device to be traced by the I/O and QTAM trace. A maximum of 3 may be specified.	Trace all devices.
IGNORE SVC= (See notes 2 and 7)	[{ nn, END, GO }]	Specify the hexadecimal SVC number to be ignored by the GSVC trace. A maximum of 6 may be specified.	Trace all SVCs.
TRACE SVC= (See notes 2 and 7)	[{ nn, END, GO }]	Specify the hexadecimal SVC number to be traced by the GSVC trace. A maximum of 6 may be specified.	Trace all SVCs.
GO (Valid SYSIPT Parameter) (See note 4)	GO (Valid SYSLOG Response) (See note 4)	GO terminates the PDAID control input and the default is used for those options that are not specified.	None.

Notes: 1. Specification of F1 or F2 is valid for MPS supervisor only. Only SVCs 0 and 31 are recorded for the QTAM trace.
2. The trace and ignore options are mutually exclusive.
3. The output device and AAA options are mutually exclusive.
4. GO will generate default parameters.
5. END means 'Press the END key', or for Models 115, 125, and 158 press the ENTER key.
6. A magnetic tape unit is the only valid output device for the QTAM trace.
7. Not applicable to the Transient dump.

Table B-4 Options and control statements for executing the PDAID trace routines.

Six examples of initiating trace routines via SYSIPT, followed by five examples of initiating via SYSLOG, immediately follow the last of those flowcharts.

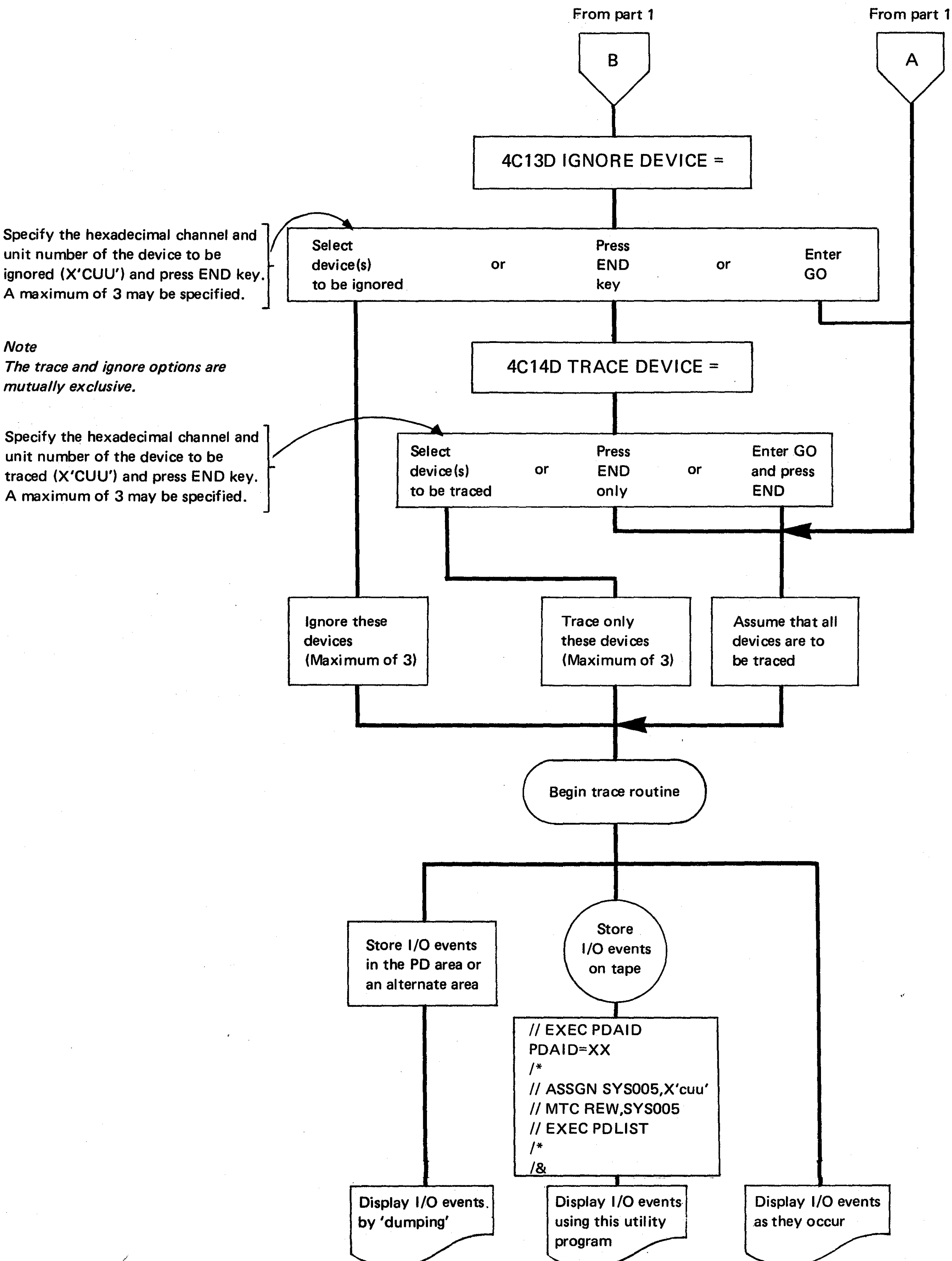
//JOB PDTRACE
//EXEC PDAID

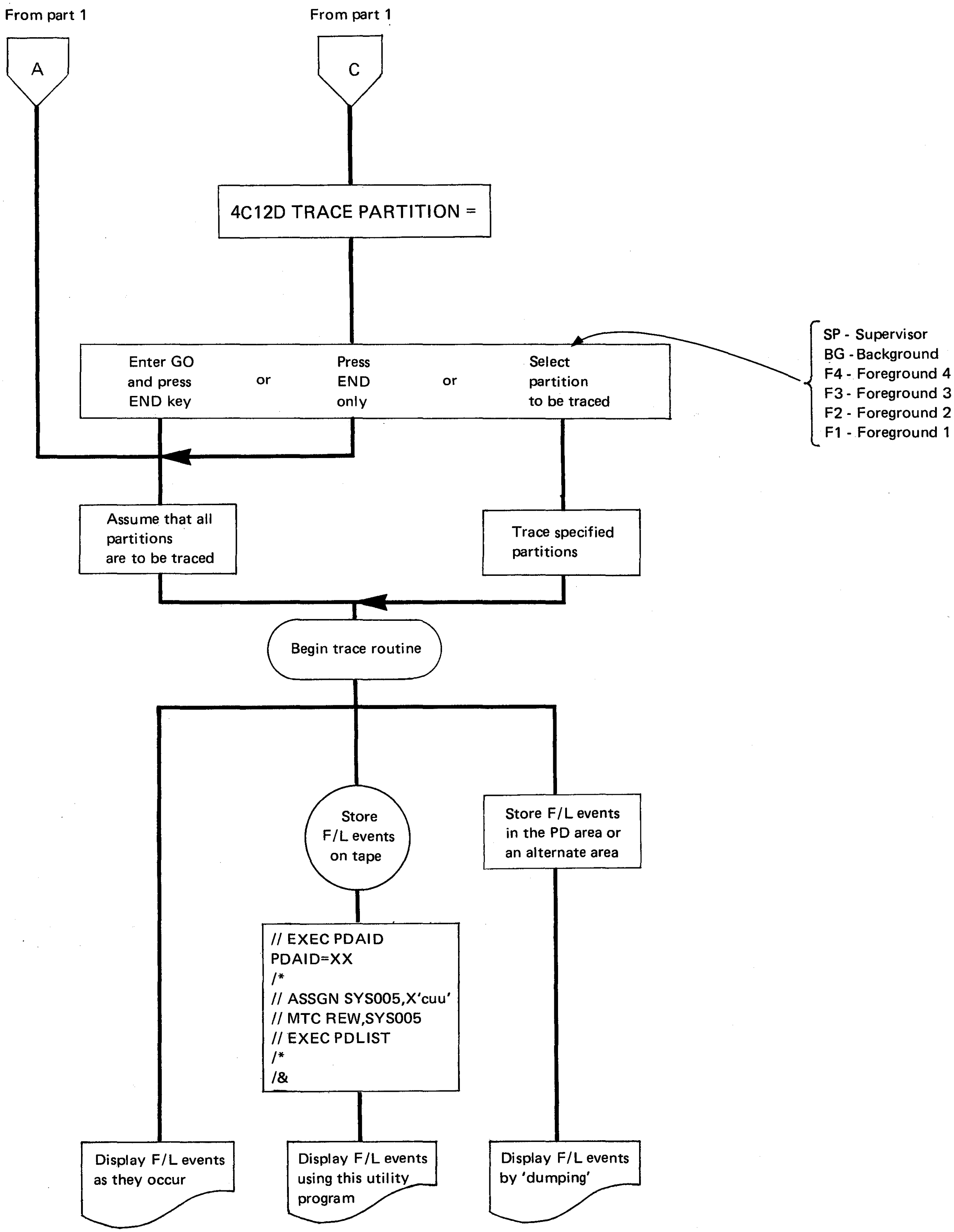


B-4-F

PDAIDS

Initializing PDAIDS part 2 of 5

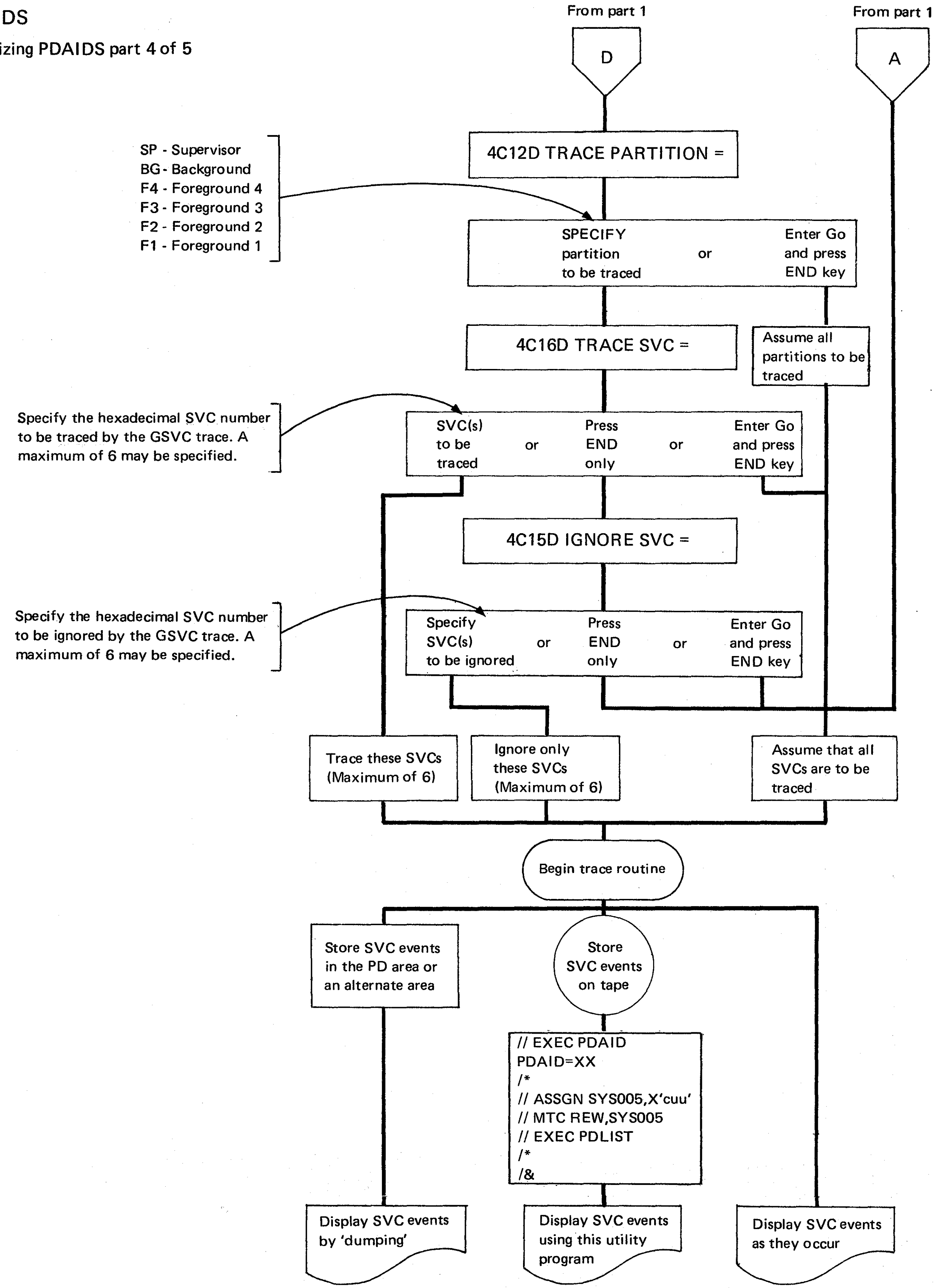


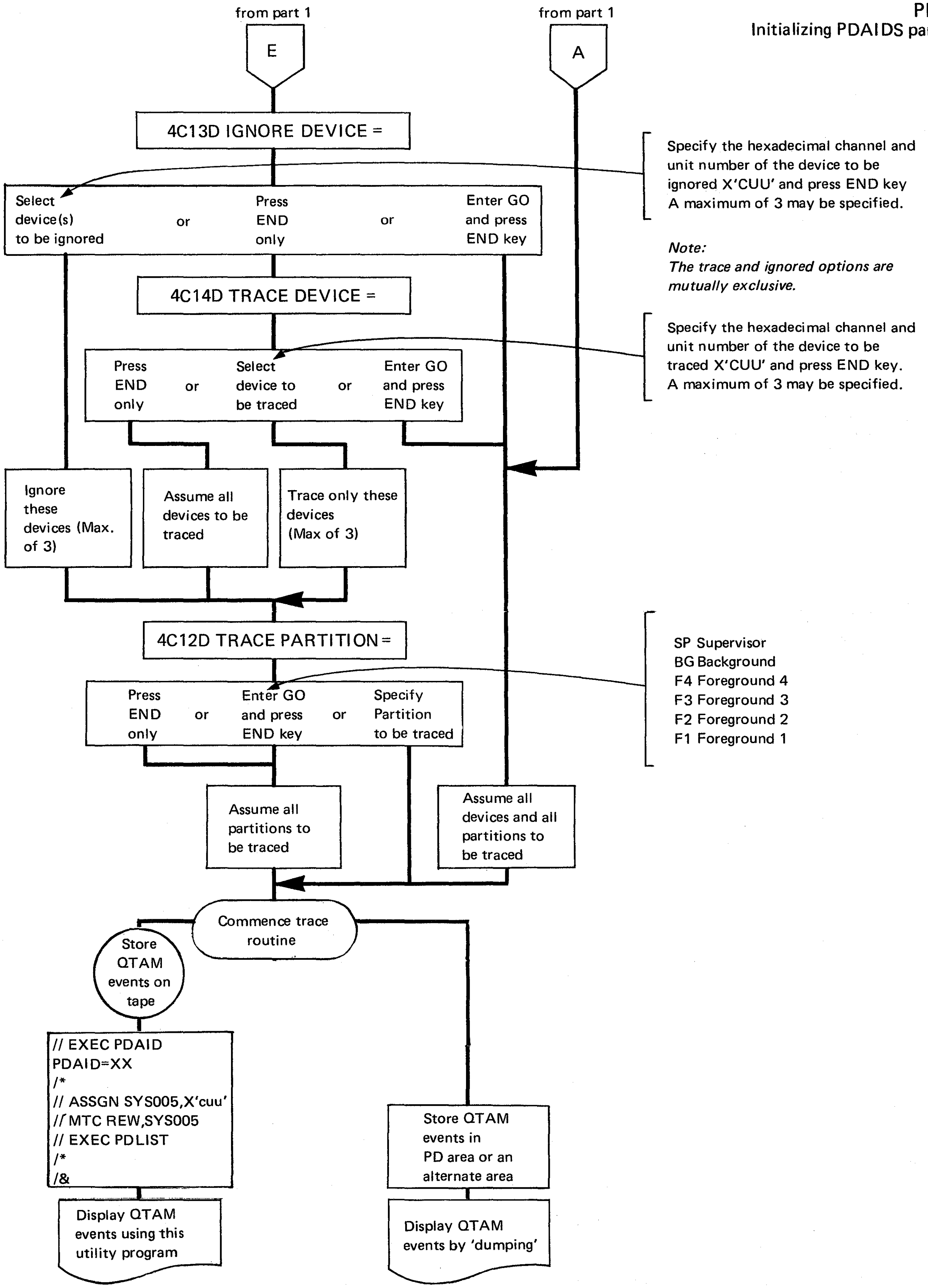

From part 1
A
From part 1
C
4C12D TRACE PARTITION =
Enter GO and press END key
or
Press END only
or
Select partition to be traced
SP - Supervisor
BG - Background
F4 - Foreground 4
F3 - Foreground 3
F2 - Foreground 2
F1 - Foreground 1
Assume that all partitions are to be traced
Trace specified partitions
Begin trace routine
Store F/L events on tape
Store F/L events in the PD area or an alternate area
// EXEC PDAID
PDAID=XX
/*
// ASSGN SYS005,X'cuu'
// MTC REW,SYS005
// EXEC PDLIST
/*
/&
Display F/L events as they occur
Display F/L events using this utility program
Display F/L events by 'dumping'

B-4

PDAIDS

Initializing PDAIDS part 4 of 5




from part 1
E
from part 1
A
4C13D IGNORE DEVICE =
Select device(s) to be ignored
or
Press END only
or
Enter GO and press END key
Specify the hexadecimal channel and unit number of the device to be ignored X'CUU' and press END key A maximum of 3 may be specified.
Note:
The trace and ignored options are mutually exclusive.
4C14D TRACE DEVICE =
Press END only
or
Select device to be traced
or
Enter GO and press END key
Specify the hexadecimal channel and unit number of the device to be traced X'CUU' and press END key. A maximum of 3 may be specified.
Ignore these devices (Max. of 3)
Assume all devices to be traced
Trace only these devices (Max of 3)
4C12D TRACE PARTITION =
Press END only
or
Enter GO and press END key
or
Specify Partition to be traced
SP Supervisor
BG Background
F4 Foreground 4
F3 Foreground 3
F2 Foreground 2
F1 Foreground 1
Assume all partitions to be traced
Assume all devices and all partitions to be traced
Commence trace routine
Store QTAM events on tape
// EXEC PDAID
PDAID=XX
/*
// ASSGN SYS005,X'cuu'
// MTC REW,SYS005
// EXEC PDLIST
/*
/&
Display QTAM events using this utility program
Store QTAM events in PD area or an alternate area
Display QTAM events by 'dumping'

B-4

Trace Routines

PDAIDS

The following six examples show job streams to initiate trace routines through SYSIPT.

Examples 1 – I/O Trace Function (single entry per card):

// JOB CARDINP1	
// EXEC PDAID	Calls for initializer.
PDAID=IT	Calls for I/O trace function.
AAA=2K	Specifies alternate save area.
IGNORE DEVICE=190	Ignores events from 190.
IGNORE DEVICE=191	Ignores events from 191.
GO	Signals end of input.
/*	
/&	

Note: No output device is specified; therefore, core-wrap is selected by default. To obtain the data held in the alternate area, SYSLST must be assigned to either a line printer, tape unit, or disk drive. Exercise care, therefore, during termination of PDAID.

For example:

```
// ASSGN SYSLST, X'191'
// EXEC PDAID
```

ensures that the alternate area is dumped on device 191 before responding XX to the message 4C10D PDAID=.

Example 2 – I/O Trace Function (multiple entries):

// JOB CARDINP2	
// EXEC PDAID	Calls for initializer.
PDAID=IT, IGNORE DEVICE=00E, OUTPUT DEVICE=180, GO	Calls for I/O trace function. Specifies that the function ignore interrupts from 00E and record I/O events on 180. (Assume 180 is a tape unit) Signals end of input.
/*	
// EXEC ASSEMBLY Source Deck	I/O activity of assembler will be traced; output will be on tape drive 180.
/*	
// EXEC PDAID	
PDAID=XX,	Terminates I/O trace function.
/*	
// ASSGN SYS005,X'180'	Assigns tape to SYS005.
// MTC REW,SYS005	Rewind the tape.
// EXEC PDLIST /*	Print out contents of tape on the printer using the PDLIST program. program.
/&	

Tape is formatted and listed on SYSLST.

Example 3 – Fetch/Load Trace Function (partitions specified):

// JOB CARDINP3	
// EXEC PDAID	Calls for initializer.
PDAID=FT	Calls for F/L trace function.
TRACE PARTITION=F2	Trace foreground 2 partition.
TRACE PARTITION=BG	Trace background partition.
GO	Signals end of input.
/*	
/&	

Note: Because no output device (OUTPUT DEVICE=) is specified, core-wrap is selected by default.

Example 4 – Fetch/Load trace Function:

// JOB CARDINP4	
// EXEC PDAID	Calls for initializer.
PDAID=FT	Calls for F/L trace function.
OUTPUT DEVICE=00E	Specifies printer output.
GO	Signals end of input.
/*	
/&	

Note: All partitions are traced if this is a multiprogramming system.

Example 5 – GSVC Trace Function:

// JOB CARDINP5	
// EXEC PDAID	Calls for initializer.
PDAID=GT	Calls for GSVC trace function.
OUTPUT DEVICE=00E	Specifies printer output.
TRACE PARTITION=BG	Trace background partition.
TRACE PARTITION=F2	Trace foreground 2 partition.
TRACE SVC=01	Trace SVC 1.
TRACE SVC=04	Trace SVC 4.
GO	Signals end of input.
/*	
/&	

B-5

Example 6 – QTAM Trace Function:

// JOB CARDINP6	
// EXEC PDAID	Calls for initializer.
OUTPUT DEVICE=180	Specifies tape output.
TRACE DEVICE=183	Trace events on tape drive 183.
TRACE DEVICE=00E	Trace events on printer.
GO	Signals end of input.
/*	
/&	

Note: All partitions are traced if this is a multiprogramming system.

PDAIDS

The following five examples show job streams to initiate trace routines through SYSLOG.

Example 1 – Store all I/O events in core using PD area for tables:

// JOB TYPINPT1	
// EXEC PDAID	Calls for initializer.
4C10D PDAID=	Console requests function.
IT and press END	Operator response: I/O trace function.
OUTPUT DEVICE=	Console requests output device.
GO	Operator response: end of input (PD area is used for output).

Note: Because no output device is specified, core-wrap mode is selected by default.

Example 2 – Trace I/O events from three specified devices, using printer output:

// JOB TYPINPT2	
// EXEC PDAID	Calls for initializer.
4C10D PDAID=	Console requests function.
IT and Press END	Operator response: I/O trace function.
OUTPUT DEVICE=	Console requests output device address.
00E and press END	Operator response: printer output.
IGNORE DEVICE=	Console requests IGNORE parameters.
Press END	Operator response: no devices to be ignored.
TRACE DEVICE=	Console requests devices to be traced and the operator specifies them.
180 and press END	
TRACE DEVICE=	
090 and press END	
TRACE DEVICE=	
01F and press END	

Note: GO does not have to be specified here. The initializer knows this is the end of input because three TRACE entries have been made.

Example 3 – Trace only the background partition and store the F/L events in the PD area:

// JOB TYPINPY3	
// EXEC PDAID	Calls for initializer.
4C10D PDAID=	Console requests function.
FT and press END	Operator response: F/L trace function.
OUTPUT DEVICE=	Console requests output device.
END	Operator response: core-wrap mode.
AAA=	Console requests alternate area.
Press END	Operator response: no AAA; store events in PD area.
TRACE PARTITION=	Console requests partition to be traced.
BG and press END	Operator response: background.
TRACE PARTITION=	Console requests second partition.
GO and press END	Operator response: end of input.

B-5

Example 4 – Trace all SVC's in both foreground partitions and list events on printer.

// JOB TYPINPT4	
// EXEC PDAID	Calls for initializer
PDAID=	Console requests function
GT and press END	Operator response. Generalized SVC trace function
OUTPUT DEVICE=	Console requests output device
00E and press END	Operator response: Printer output
TRACE PARTITION=	Console requests partition to be traced
F1 and press END	Operator response: foreground 1
TRACE PARTITION=	Console requests second partition to be traced
F2 and press END	Operator response: foreground 2
TRACE PARTITION=	Console requests third partition to be traced
Press END	Operator response: no more partitions to be traced
IGNORE=	Console requests first SVC to be ignored
Press END	Operator response: No SVCs to be ignored
TRACE SVC=	Console requests first SVC to be traced
GO and press END	Operator response: Trace all SVCs: end of input

Example 5 – Trace interrupts on tape drive 180 and printer 00E using the QTAM trace function and store the events in the PD area:

// JOB TYPINPT5	
// EXEC PDAID	Calls for initializer.
4C10D PDAID=	Console requests function.
QT and press END	Operator response: QTAM trace.
OUTPUT DEVICE=	Console requests output device address.
Press END	Operator response: PD area.
AAA=	Console requests alternate area.
Press END	Operator response: no alternate area.
IGNORE DEVICE=	Console requests device to be ignored.
Press END	Operator response: no device to be ignored.
TRACE DEVICE=	Console requests device to be traced.
180 and press END	Operator response: Trace interrupts on device 180.
TRACE DEVICE=	Console request second device to be traced.
00E and press END	Operator response: trace interrupts on device 00E.
TRACE DEVICE=	Console requests third device to be traced.
Press END	Operator response: no third device; end of input.
TRACE PARTITION=	Console requests first partition to be traced.
F4 and press END	Operator response foreground 4.
TRACE PARTITION=	Console requests second partition to be traced.
Press END	Operator response: end of input.

B-5

PART 2 SDAIDS

General description

SDAIDS provide further tracing facilities to supplement those already provided by the PDAIDS. While the PDAIDS produce a predefined output for each type of trace, as described in Part 1 of this Section, most of the SDAID trace functions can be initiated to produce information that is more defined for a given type of system malfunction. The SDAID printout ranges from one printed line for each event up to a dump of the complete real storage for each event. (No events will be lost as they may be with PDAID output.) SDAIDS also provide special dumping facilities that enable non-destroying dumps to be executed on the occurrence of specific events during program operation.

CAUTION

The effect on the operation of programs currently running in the system that are time dependent, for example, a program using MICR or teleprocessing as input/output, must be considered before using this serviceability aid.

The SDAID trace functions are as follows:

1. A page trace, consisting of
   - a page translation exception trace (when a page fault occurs)
   - a page enque trace (when a page is placed in the page queue)
   - a page handling trace (when a page is removed from the page queue)
2. An instruction trace that records instructions in the order in which they are executed between any selected addresses.
3. A main storage alter trace that records the address of the instruction that altered the contents of any or all byte locations between any selected addresses.
4. A general register alter trace that records any alteration made to any one, or any selected, general registers.
5. A successful branch trace that records the address at which a successful branch is made, ~~between any selected addresses.~~

The stop and dump facilities are:

1. Stop on event: On the occurence of one or any of the following specified events, all system activity is suspended after SDAID output is complete.
   - at any specified instruction address
   - on alteration of any byte location between any selected addresses
   - on alteration of one or more specified general registers
   - on any successful branch that occurs between any selected addresses
   - on the occurrence of a page translation exeption
   - on the occurrence of a program check code X'01'–X'10' and X'12'
   - on the occurrence of a request for a page to be placed in the PG queue (page fault enqueued).
   - on the occurrence of a request for a page to be removed from the PG queue by the page handler.

2. As well as being able to obtain a dump of areas specified by the output class at the stop event, it is possible to obtain a dump of real and virtual address areas after the specified output class has been dumped.

   The types of dumps that can be obtained in this way are:

   - Non-destroying dump: This is a dump of all real storage. It can be obtained if required after a stop on event. The dump is non-destroying because system status information is preserved, thus enabling system operation to continue after execution of the dump.
   - Dump on a program check: On the occurrence of a program check interrupt (codes X'01' to X'0F', X'10', and X'12'), a non-destroying dump of the complete supervisor area is automatically executed.
   - PDUMP: Enables a dump of a minimum area of 32 bytes (one print line) between two virtual address limits. The maximum area that can be dumped depends only on the size of virtual storage, and only virtual address area information that is in real storage is dumped.

System requirements

The SD area need not be specified during system generation, but the SDAID initializing and terminating programs must be cataloged in the core image library.

SDAIDS make use of program event recording and monitoring, described in Appendix E.

Output from all SDAIDS routines is directed to a line printer. The line printer is non-dedicated, meaning that the same printer may be used as an output device for other programs as well as for the SDAIDS. Therefore, SDAID output may be interspersed with job output.

Note: the following restriction, if the printer is connected via a selector or block multiplexer mode channel:

No other devices must be running on the same channel as the printer at the moment when SDAID attempts to write to the printer.

SDAIDS

SDAID Characteristics

- SDAIDS reside in the SD area, which must occupy at least 6K bytes of the real address area.
  The storage assigned to the SD area is taken from the page pool.
- SDAID is initialized by // EXEC SDAID, and requires 12K of a real or virtual partition (only during initialization of any SDAID function). Parameters, specified either at initialization time or later, must be entered on the console.
- After initialization, SDAID does not use DOS/VS services.
- SDAID has immediate control in case of a program check interruption.
- SDAID runs with DAT (Dynamic Address Translation) off, disabled for I/O and external interrupts.
- After SDAID handled event, processing continues as if event handling had not occurred.
- Only the contents of the real address area is dumped with SDAID. (Pages that currently reside only on the page data set will not be dumped.)
- SDAID may not be used to debug time-dependent programs because it runs disabled while recording events and thus delays processing.
- Because SDAIDS use the program event recording PER facility, and because time is required to print SDAID output, program execution time is increased. Its effect on the operation of time-dependent programs must therefore be considered before using this serviceability aid. Performance degradation when using SDAIDS will be reduced when the FASTREC output class is selected.
- Debugging of printer error recovery routines is possible only if the FASTREC output class is used.
- If, during the printing of SDAID output, the line printer is stopped for any reason or becomes not ready, the system will enter a wait state with a message in low address storage. To continue printer operation, press the EXTERNAL INTERRUPT key.
- When initialization is complete, the event handling routines within the SDAID initiating program partition are transferred to the SD area. The 12K partition can then be re-used, but the pages occupied at the end of the page pool by the SD area are not released for normal program use until all SDAID functions are terminated.

Terminating the SDAID routines

The tool SDAID is terminated, and the SD area is released to the page pool by one of the following:

1. The AR (attention routine) command ENDSD
2. The job control statement // EXEC ENDSD

Note: Depending on the events being traced and the event limits specified, it may take some time before the attention routine or job control becomes active. One method to avoid this delay is to clear control register 9 using the ALTER/DISPLAY console feature before requesting the attention routine.
This de-activates all PER event tracing.

Using SDAID and PDAID concurrently

If the system has been generated to accept PDAIDS, any one of the PDAID trace routines may run concurrently with SDAID. However, if the PDAID currently running is using an alternate area, it must first be terminated before an SDAID routine can run.

SDAID Events

SDAID events are recognized as program checks. There are two groups of events: elementary events and dedicated events.

Elementary events are:

Mnemonic	Event
BR	successful branching
IF	instruction fetching
SA	storage alteration
GA	general register alteration
TE	page translation exception

Dedicated events are:

Mnemonic	Event
PGMCHK	program interruption codes X'01'–X'0F' and X'10', X'12'
PAGENQ	request for page is enqueued
PAGEHDL	request for page is handled

B-6

SDAID output information

When an event occurs, the SDAID event handling routines will record either the information specified by output class parameter (for elementary events), or predefined data (for dedicated events).

By using the output class parameter of the SDAID operand OUTCL= the amount and and type of information required for offline program debugging can be selected for the elementary event during initialization of the SDAIDS. After initialization, the output class can also be re-specified if required.

For elementary events the output class can be specified according to Table B-6-A. However, if more than one elementary eve nt is being traced simultaneously, the output class will be the same for all events. For each dedicated event, a predefined output is obtained as shown in Table B-6-B.

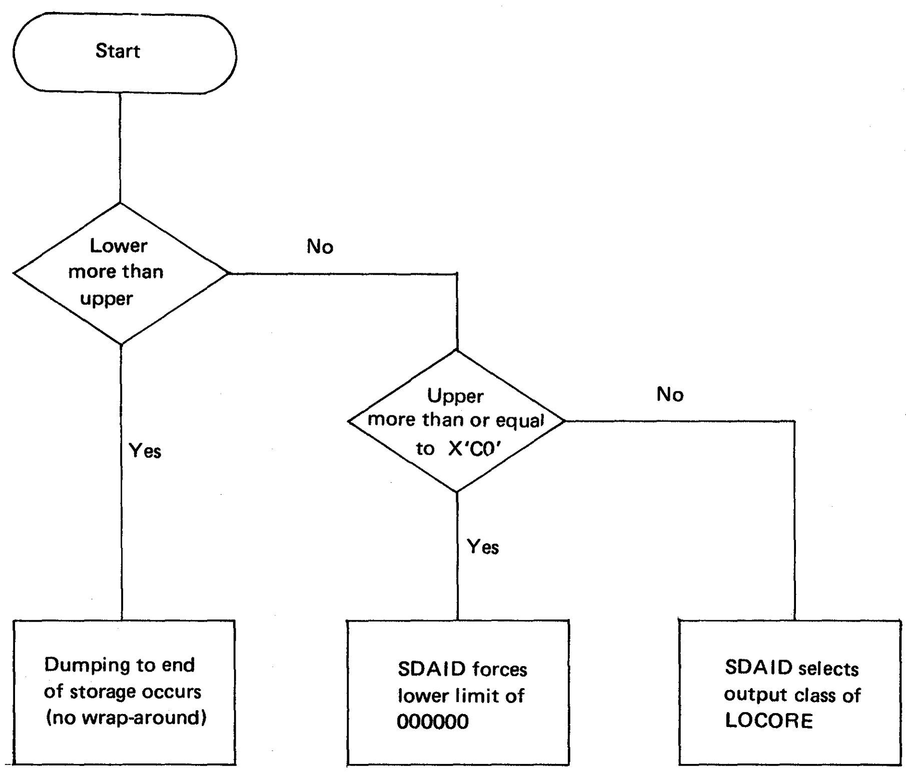
Output if more than one event is being traced:

Events can be enabled individually or in combination with one another. If some or all of the events BR, IF, SA, and GA happen concurrently, the output class listed in Table B-2-A is printed only once. The event ID, however, contains the mnemonics of all current events.

If any of the other events happen concurrently, even if they occur together with one of the events BR, IF, SA, or GA, the output is printed for each event that occurs.

PDUMP output class: A PDUMP is triggered by events just as the other output classes. It dumps a minimum of 32 Bytes (one print line) between two virtual address limits. The maximum area that can be dumped depends only on the size of virtual storage. Any are a between the two limits, not in real storage, will be indicated by a message.

Any PDUMP limits may be specified. However, the value of the limits in relation to the value X'C0' and to each other determines the output.



Note: The defaults for the PDUMP limits are the EVENT limits (X'llllll', 'hhhhhh') specified in answer to message 4C61D, refer to Table B-10.

Recorded Information / Output classes and Mnemonic	OUTCL 1 PSW 01	OUTCL 2 GPR 02	OUTCL 3 LOCORE 03	OUTCL 4 COMREG 04	OUTCL 5 PAGETAB 05	OUTCL 6 SUPVISOR 06	OUTCL 7 DUMPREAL NDD 07	OUTCL 8 PDUMP 08	OUTCL Q FASTREC** 00
Event ID* program old PSW, and time of day in microseconds	X	X	X	X	X	X	X***	X	X
Instruction causing event	X	X	X	X	X	X	X***	X	
General purpose registers		X	X	X	X	X	X	X	
Low core (X'000'–X'11F')			X	X	X	X	X		
Current COMREG and SYSCOM				X		X	X		
Control registers,			X	X	X	X	X	X	
segment tables, page tables, page frame table					X	X	X		
Complete supervisor						X	X		
Complete real address area							X		
Virtual dump between PDUMP address limits								X	
TE-MASK PER mask (control register 9) GPR mask (control register 9) PER start address (control register 10) PER end address (control register 11) general purpose registers 13, 14, 15, 0, 1, 2									X

Notes

* *Event ID for BR, IF, SA, and GA – event mnemonic and instruction address.*
Event ID for TE – mnemonic TE and address of the page causing TE.

** *FASTREC is an output class that stores the described information into an SDAID internal buffer. Information for several events is stored and printed as one block.*

*** *INSTR and PSW are not printed if NDD is forced after STOP ON EVENT VIA NDD BYTE X'FF'.*

Table B-6-A . Output class options for SDAID elementary events.

Examples at the end of this section 2-F show several types of output specified by the output class parameter.

Recorded Information	Dedicated Event		
	PGMCHK	PAGENQ	PAGEHDL
Event-Mnemonic Program old PSW, Time of day in microseconds, Complete supervisor, Instruction at time of PGMCHK Control registers General purpose registers	X		
Event-Mnemonic Requestor-ID (TE/GETR/TFIX/PFIX) Task-ID Address of page to be handled		X	X
Protection key associated with page to be handled		X	X
Address of page frame to which the page is assigned			X

Table B-6-B. Predefined output obtained from SDAID dedicated events.

Specification of area to be traced

For elementary events, two addresses may be specified during SDAID initialization as the start and end addresses of the area to be traced or monitored. These address limits are interpreted as virtual addresses if the DAT bit in the PSW is on. Address limits are not applicable to dedicated events, for which the SDAID program includes all real and virtual address areas.

If the start address specified is higher than the end address, tracing commences from the higher address and continues to the end of virtual storage (the maximum address being 16,777,215). Tracing continues from address 000000 up to the end address (the lower address specified). This is termed “wrap around tracing”.

Translation exception trace

This occurs when an instruction requires a page to be paged in from the page data set in order for the instruction to be completed. An example is an MVC instruction whose address 1 is in page frame x in real storage, and whose address 2 is in page y that is not in real storage.

When this trace is initialized, any page fault generated because of such an instruction is printed along with the instruction and its address that caused the page fault, plus the output of the specified output class.

Page enqueue trace

This trace enables the sequence to be traced in which programs are calling for pages. Page faults caused by translation exceptions will also be traced with this routine.

Page handling trace

This trace provides information about the sequence in which pages are paged in from the page data set. After a page is handled, a trace output is printed.

When to use: Use this trace if you suspect that the loss of a page, or the sequence of page usage by a program, is causing programming errors. This trace gives you page management information during program execution.

Instruction trace

This trace records information about the order of instruction execution within any selected area of storage during program execution. The amount and type of information provided depends on the output class selected during initialization of the trace.

When to use: If an unintended loop develops during program execution, this trace can be initiated and the program re-run. During the re-run, a list of all the instructions executed within the loop will be traced. This is an efficient method to obtain a loop trace.

Storage alter trace

This trace records information about instructions that alter one or more locations in virtual storage between address limits that can be specified. The amount and type of information provided depends on the output class selected during initialization of the trace routine.

When to use: If, for example, you suspect I/O areas or count locations for loops, information obtained from this trace output will show the instructions that are altering the areas. The SA trace will not record changes in the contents of locations that are changed directly by I/O channel operations.

General register alter trace

This is similar to the virtual storage alter trace. It should be used when information about changes to any GR during program execution is required to help during offline program debugging. Any GR or any combination of GRs can be traced.

Successful branch trace

This trace provides a check on the logical path of a program during its execution in any selected part of virtual storage.

When to use: Use this trace if the actual path taken by a program cannot be analysed from the program flowcharts and listings. You can also use it to provide information about the path taken, for example, by a long loop.

Stop and dump routines

Stop on event

This facility stops all system activity on the occurrence of a specified event. At the stop on event, the system is held in a wait state.

Processing continues via external interrupt.

With the system in this wait state, the operator or programmer can either use hands-on debugging aids or obtain a non-destrcying dump.

The specified event can be one or more of the elementary or dedicated events.

When to use:

1. Use this routine if hands-on debugging is necessary on the occurrence of one of the specified events. For example, when a change occurs in a general register, you may want to look through the program listings to enable you to decide on the next step in isolating an error. When the stop occurs, it is also possible to initiate another SDAID routine that will provide additional system information for offline program debugging.
2. When no time is available for hands-on debugging, the non-destroying dumps obtained when the stop on event occurs will provide a great deal of additional information for offline program debugging.

Stop on address

This facility provides a stop on address on any specified (real or virtual) address. When the stop occurs, the system is held in a wait state, and the operator or programmer can use hands-on debugging aids or obtain a non-destroying dump.

When and how to use: This facility is used under conditions similar to those for the hardware stop on address compare, that is, hands-on debugging is to be carried out when a program has reached some specific point during its operation. However, this aid enables a stop on all SDAID events.

The stop on address is accomplished by initiating the instruction trace, specifying stop on event, and entering the address at which the stop is required as the address supplied within the event limit field during initialization of the trace.

SDAIDS

Non-destroying dump: This is a dump of real storage that can be obtained after the occurrence of a specified event during the stop on event. The dump is non-destroying because the system is placed in a wait state on the occurrence of the specified event, and because SDAIDS do not destroy system status during execution of the dump.

How to obtain the dump: The following procedure describes how to obtain the non-destroying dump:

1. When the system is in the stop-on-event wait state, locate the real storage address of the NDD (non-destroying dump) byte switch.
   The address of this program switch is printed during SDAID initialization. Refer to point 3 of the example in this section which shows the SDAID initializing output part 2.

2. To ensure that the wait state is the true stop-on-event wait, use the ALTER/DISPLAY console feature to display the PSW. The instruction address part of the WAIT PSW will be OOOOEEEE.

3. To obtain the dump, set the NDD byte to X'FF', using the ALTER/DISPLAY console feature as described in this Section 2-D.

4. Press the START key and then the EXTERNAL INTERRUPT key. A non-destroying dump will be printed and processing continues.

When the dump is complete, the NDD byte is reset by the SDAID program, and so a dump will not occur at the next stop on event. To obtain another dump at any following stop, the NDD byte must again be set on.

Note: The dump can be discontinued by the following procedure:

1. Make the line printer used as SDAID output device unready.
2. Now make the printer ready.
3. Press the EXTERNAL INTERRUPT key two times within one second.

When to use: This SDAID facility enables you to obtain the information needed for problem analysis without having to take dumps of real storage at every occurrence of an event. Therefore this decreases the amount of paper to be searched through during offline debugging. For example you may consider it sufficient for offline debugging to take a dump at every twenty seventh occurrence of an event.

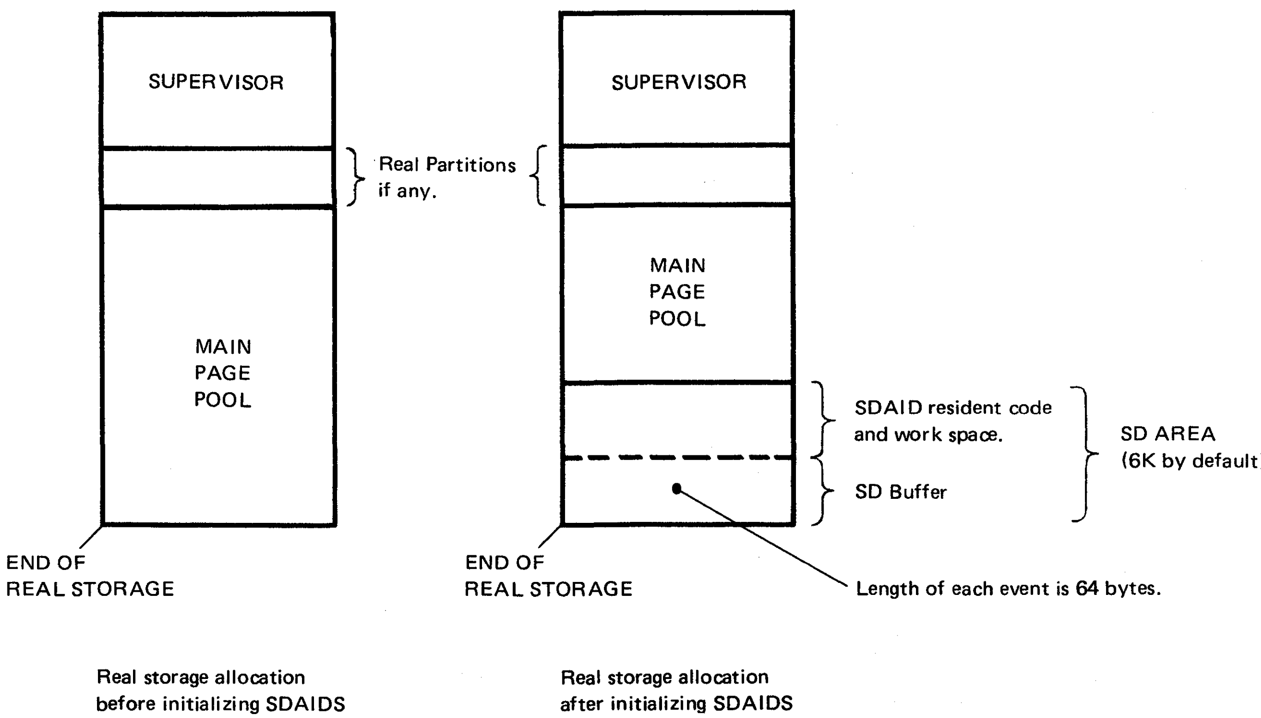
Dump on a program check: On the occurrence of a program check interrupt codes X'01' – X'0F', X'10', and X'12', the following information is dumped automatically:

- Event ID
- Program old PSW
- Time of day in microseconds
- Control registers
- General purpose registers
- Real storage from byte location 0 to the end of the supervisor area, and the contents of the SDAID buffer.

After this automatic non-destroying dump is executed, the DOS/VS program check handler routine will be entered.

When to use: If PDAIDS are not available on your system, the use of the SDAID dump on a program check is the only way to obtain a non-destroying dump of the supervisor transient area at the time of a program check interrupt.

The SD area



How to locate

The address of the beginning of the SD area is printed on the device assigned to SYSLST during initialization of SDAID. Refer to page 2.90 for an example of SDAID initializing output.

B-9

SDAIDS

Initializing SDAID

SDAID may be initialized in any real or virtual partition by entering the following execute statement via SYSLOG or SYSRDR:

// EXEC SDAID

An operator/system dialog follows, beginning with the message:

4C55D GIVE SPACE FOR SDAID=

The operator may respond by pressing the END key (which gives a default value of 6K to SDAID), or he may specify a value nK, where n represents a multiple of 1024 bytes. The maximum value that may be specified is 999K. If an odd number is specified, the value is incremented to the next even number. The SDAID space is taken from the main page pool. If the main page pool is not large enough to accept the area specified, the minimum area of 6K is automatically taken. If the page pool is not large enough to accept the minimum (6K), the following message is printed on SYSLOG.

4C56E INSUFFICIENT SDAID SPACE, REALLOCATE

The MAP command should be issued before reallocating real partition areas in order to increase the size of the page pool before re-initializing SDAIDS.

The following message will be issued if this is a second request for SDAID space:

4C70E DUPLICATE REQUEST FOR PDAID AND/OR SDAID

This message is also issued if PDAID using the core wrap output mode in an alternate area is active in the main page pool and a request for SDAIDS is made. When the space allocated to SDAID is accepted, a message dialog follows that allows the operator to select one or more events to be traced and to specify between which address limits of real or virtual storage the events are to be traced. (Event limits do not apply to event PAGENQ, event PAGEHDL, and event PGMCHK.) The dialog also enables the selection of a line printer at a device address other than X'00E', which is the device address by default. However, the device must be a line printer.

An output class may also be specified (refer to Tables B-6-A and B-6-B in this chapter). A response of EOB (pressing the END key) to all SDAID messages will give default values.

When the SDAID message dialog is complete, the SDAID initializing outputs part 1 and 2 are issued to the device assigned as SYSLST. This need not be the same device on which SDAID trace output is printed. The SDAID trace output is printed immediately after the initializing output on the device at the address specified in the reply to message

4C58D OUTPUT DEVICE=

(Address X'00E' is taken as default.) After initialization, the partition used for the initialization is given back to the main page pool.

The table shown in Figure B-3 lists all SDAID messages in the order in which they are issued and describes the responses.

SDAID job entry examples are shown after the example of the SDAID initializing output. Operator flowcharts follow.

SDAID messages after initialization time

4C71I SDAID FOUND PRTR STATUS CSW SENSE

This message may be written out on the printer. It is accompanied by the CSW and SENSE information if applicable. It indicates that the previous printer operation which was started may not have been completed successfully.

Altering SDAID functions and/or address limits after initialization

When the SDAID is initialized, trace functions and events limits, where applicable, can be changed by altering the SDAID program parameters directly in storage. The contents of the parameters at the addresses printed on part 2 of the SDAID initializing output, and of control registers 8, 9, A, and B, must be altered to predetermined values. Their values are also printed in the initializing output.

To make SDAID parameter changes:

- Press the STOP key.
- Use the console ALTER/DISPLAY feature to alter the contents of the program parameters.
- Press the START key.

Note: When SDAID is terminated and later re-initialized, new SDAID parameters are printed in the SDAID initializing output.

Note: SDAID requires SYSLST for the initializing output. Therefore, if you intend to change SDAID parameters after initializing SDAIDS, you should ensure that the SYSLST device is a line printer on the partition used for SDAID initialization.

A note to programmers

SDAIDS are primarily designed to be initialized before re-running failing programs. If you, as the programmer, are debugging on the system (hands-on debugging), it is recommended that you initiate SDAIDS without specifying any events. (Press the END key as a response to all SDAID messages.) SDAID is then retained in the page pool ready to be activated. The failing programs can then be executed and SDAID events made active by entering event parameters directly into control registers 8, 9, 10, and 11. For example, altering the contents of the high-order byte of control register 9 (by the console ALTER/DISPLAY feature) enables you to activate any one or all of the events BR, IF, SA, and GA.

You can also specify which general registers are to be traced by entering values into the lower 2 bytes of control register 9. Control registers 10 and 11 contain, respectively, the start and end addresses for the event limits. The output of the MAP command will tell you the partition in which the failing programs reside.

From the MAP output you can also obtain the addresses of the upper and lower limit of the partition, which can then be used as the event limits for the SDAID trace. (Note that addresses printed by the MAP command are decimal.)

If you are unable to use the system for hands-on debugging, you as the programmer must specify clear instructions to the operator about the events to be traced and the event limits to be used.

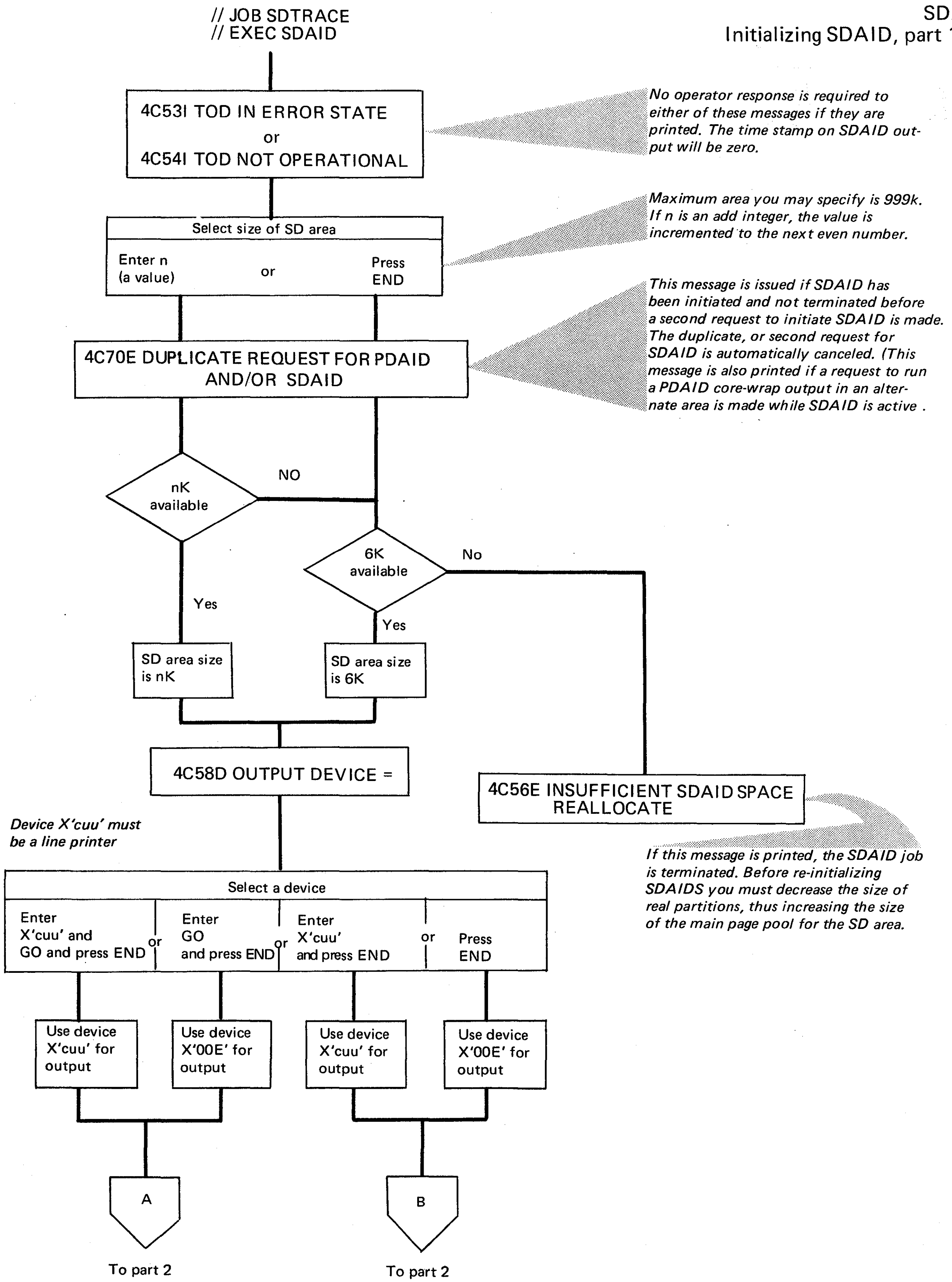
GO RESPONSE CAUSES SYSTEM TO TAKE DEFAULT OPTION.

Message Code	Message issued on SYSLOG	Parameters entered by Operator	Remarks
4C58D	OUTPUT DEVICE=	[X'00E' [,GO] X'cuu' [,GO] GO] END/ENTER	
4C60D	STOP ON EVENT=	[YES NO] [,GO] END/ENTER	
4C61D	EVENT LIMITS=	[X'000000',X'FFFFFF' [,GO] X'llllll' [,X'llllll' [,GO] ,X'hhhhhh' [,GO] ,GO]] END/ENTER	X'llllll',X'hhhhhh': Lower and upper limit of virtual storage to be traced with events BR, IF, SA and TE.
4C59D	OUTCL=	[PSW GPR LOCORE COMREG PAGETAB SUPVISOR DUMPREAL [,GO] PDUMP [,lower and upper event limit [,GO] ,X'aaaaaa' [,upper event limit [,GO] ,X'bbbbbb' [,GO] ,GO] ,GO] FASTREC [,PGMCHK [,GO] ,AUTOMATIC [,GO] ,GO]] END/ENTER	Valid output classes for the events BR, IF, SA, GA, and TE. END/ENTER PGMCHK: Causes wrap around mode of internal buffer. It is written each time a PGMCHK event occurs. AUTOMATIC: If the internal buffers is full, it is written out.
4C62D	EVENT BR=	[YES NO] [,GO] END/ENTER	
4C63D	EVENT IF=	[YES NO] [,GO] END/ENTER	
4C64D	EVENT SA=	[YES NO] [,GO] END/ENTER	
4C65D	EVENT GA=	[X'012...EF' [,GO] NO] END/ENTER	Designate the general purpose registers to be traced. At least one must be specified.
4C66D	EVENT TE=	[YES NO] [,GO] END/ENTER	
4C67D	EVENT PGMCHK=	[YES NO] [,GO] END/ENTER	
4C68D	EVENT PAGENQ=	[YES NO] [,GO] END/ENTER	
4C69D	EVENT PAGEHDL=	[YES NO] END/ENTER	

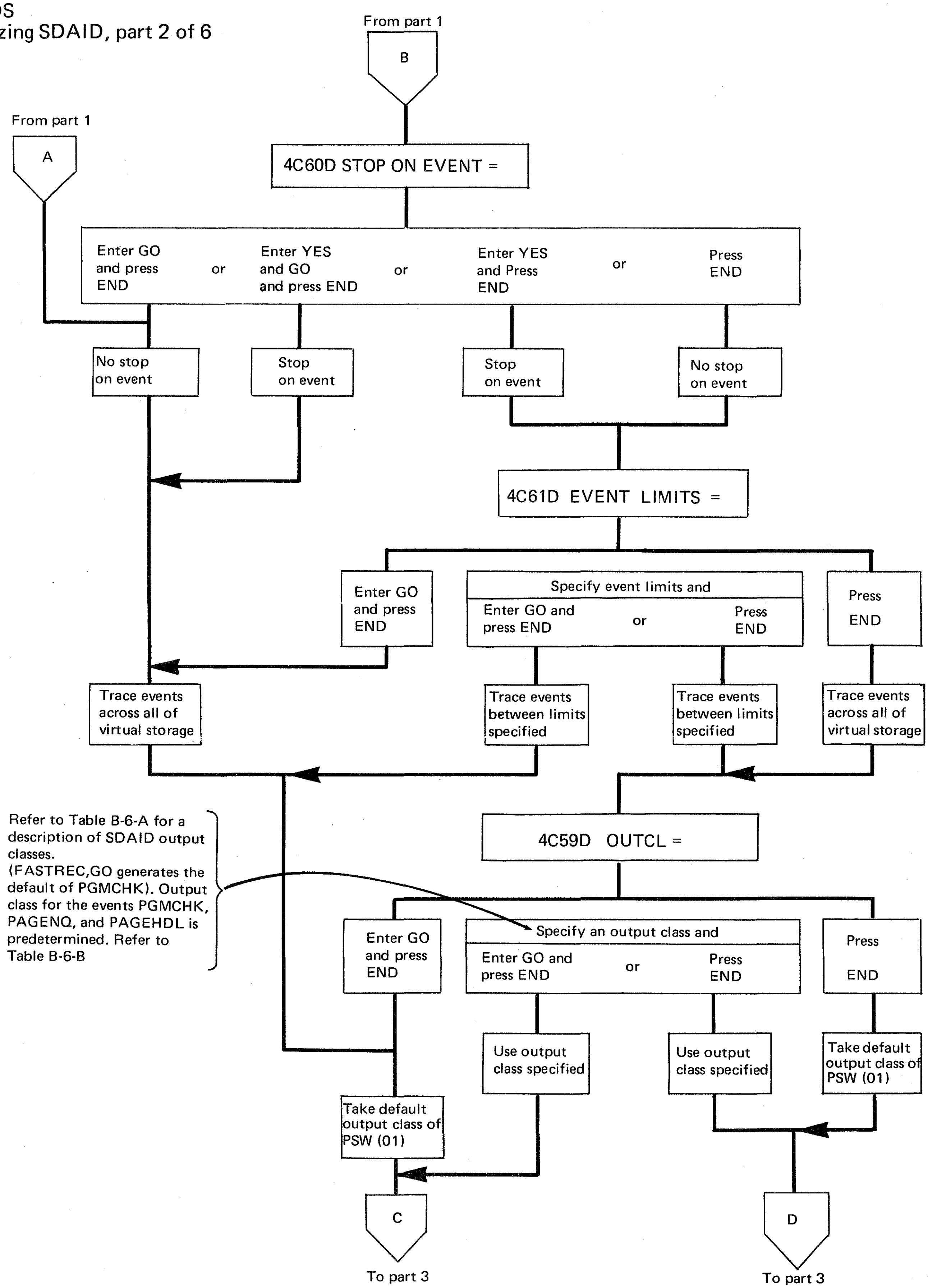
Default is NO

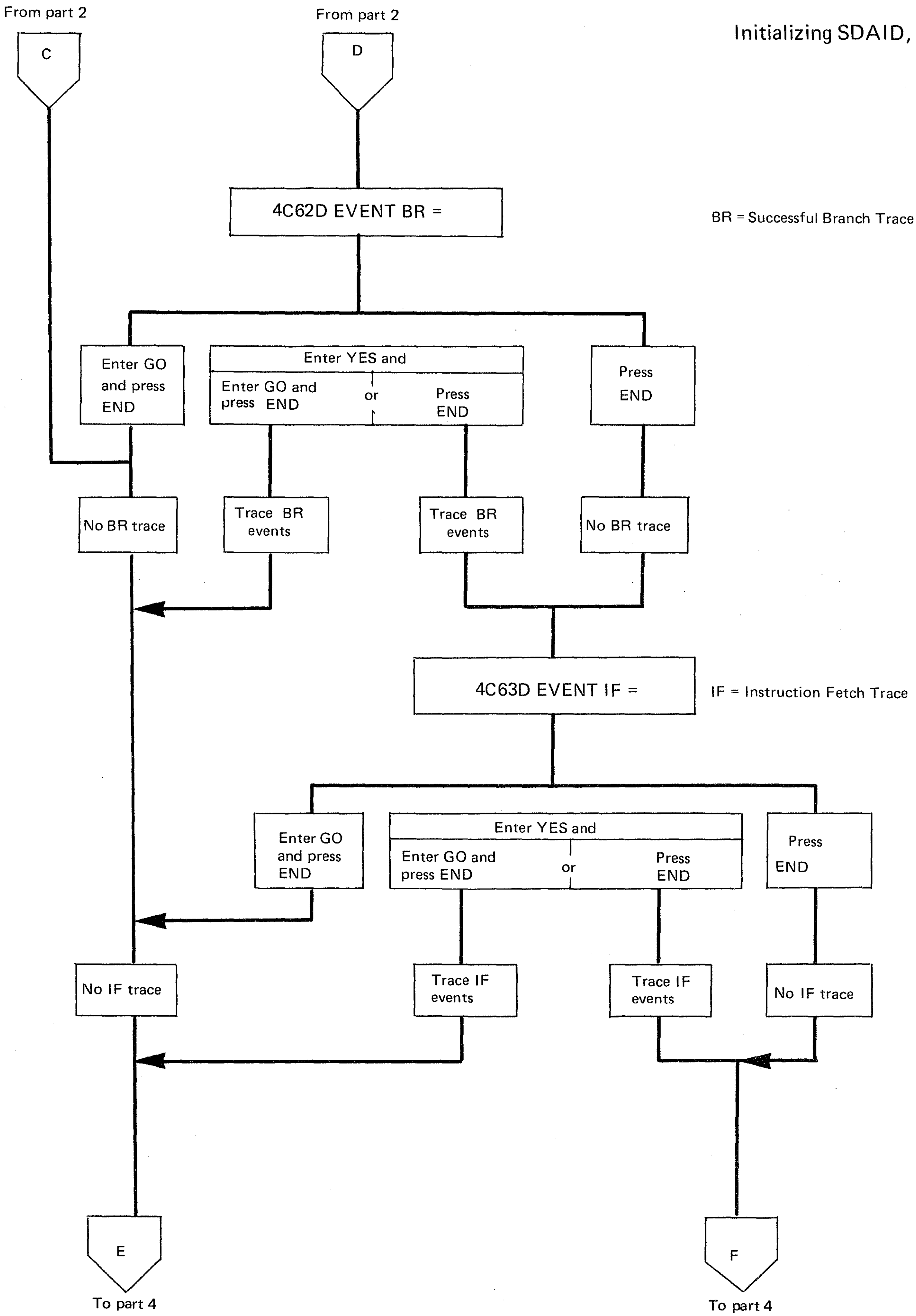
Note: If GO is entered for a parameter, the dialogue is terminated immediately and defaults (underlined) are taken for the remaining parameters. DEFAULTS

Table B-10. The parameters required to initialize SDAID event tracing.

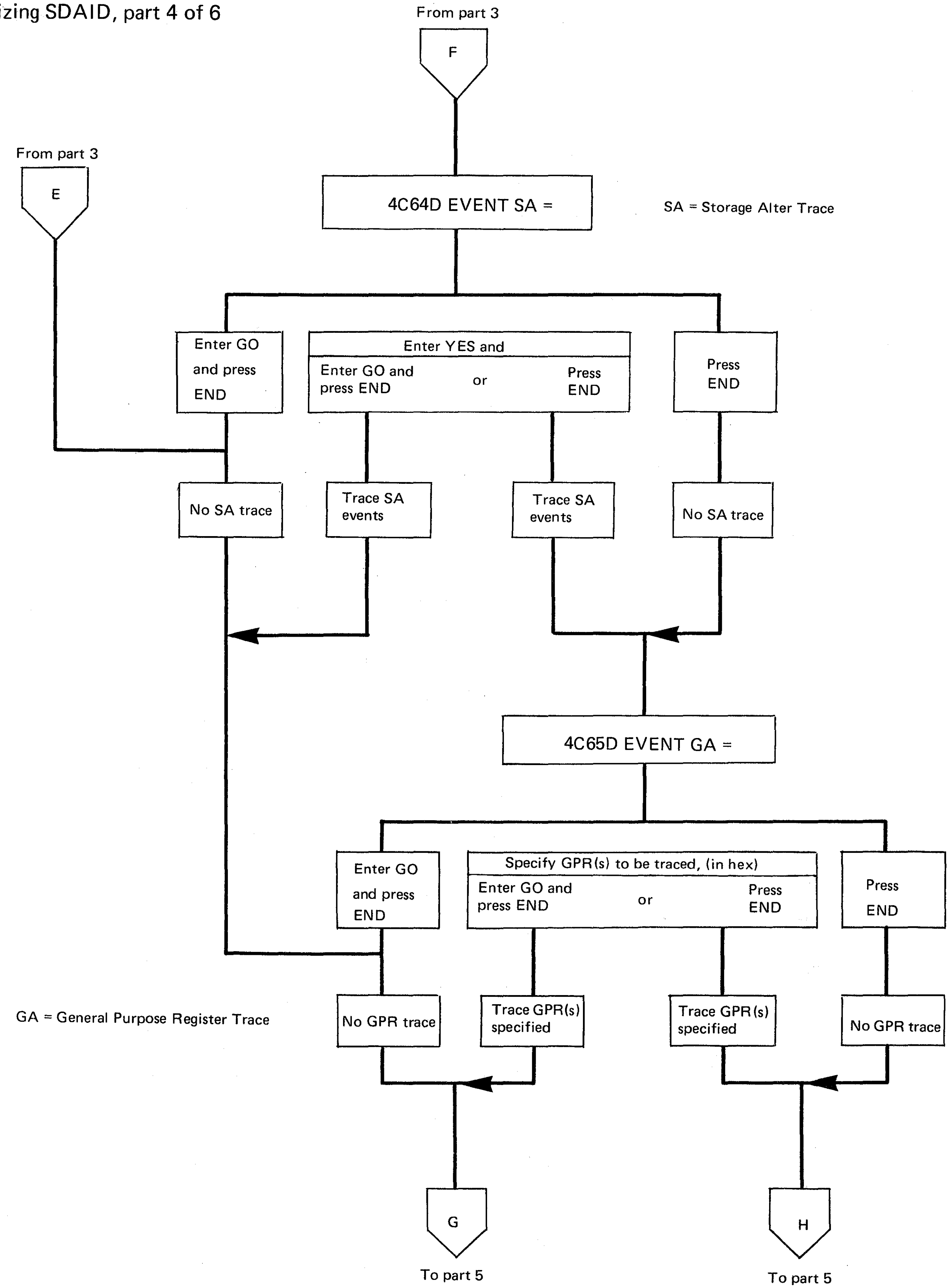

// JOB SDTRACE
// EXEC SDAID
4C53I TOD IN ERROR STATE
or
4C54I TOD NOT OPERATIONAL
No operator response is required to either of these messages if they are printed. The time stamp on SDAID output will be zero.
Select size of SD area
Enter n (a value)
or
Press END
Maximum area you may specify is 999k. If n is an add integer, the value is incremented to the next even number.
4C70E DUPLICATE REQUEST FOR PDAID AND/OR SDAID
This message is issued if SDAID has been initiated and not terminated before a second request to initiate SDAID is made. The duplicate, or second request for SDAID is automatically canceled. (This message is also printed if a request to run a PDAID core-wrap output in an alternate area is made while SDAID is active .
nK available
NO
Yes
6K available
No
Yes
SD area size is nK
SD area size is 6K
4C58D OUTPUT DEVICE =
4C56E INSUFFICIENT SDAID SPACE REALLOCATE
If this message is printed, the SDAID job is terminated. Before re-initializing SDAIDS you must decrease the size of real partitions, thus increasing the size of the main page pool for the SD area.
Device X'cuu' must be a line printer
Select a device
Enter X'cuu' and GO and press END
or
Enter GO and press END
or
Enter X'cuu' and press END
or
Press END
Use device X'cuu' for output
Use device X'00E' for output
Use device X'cuu' for output
Use device X'00E' for output
A
B
To part 2
To part 2

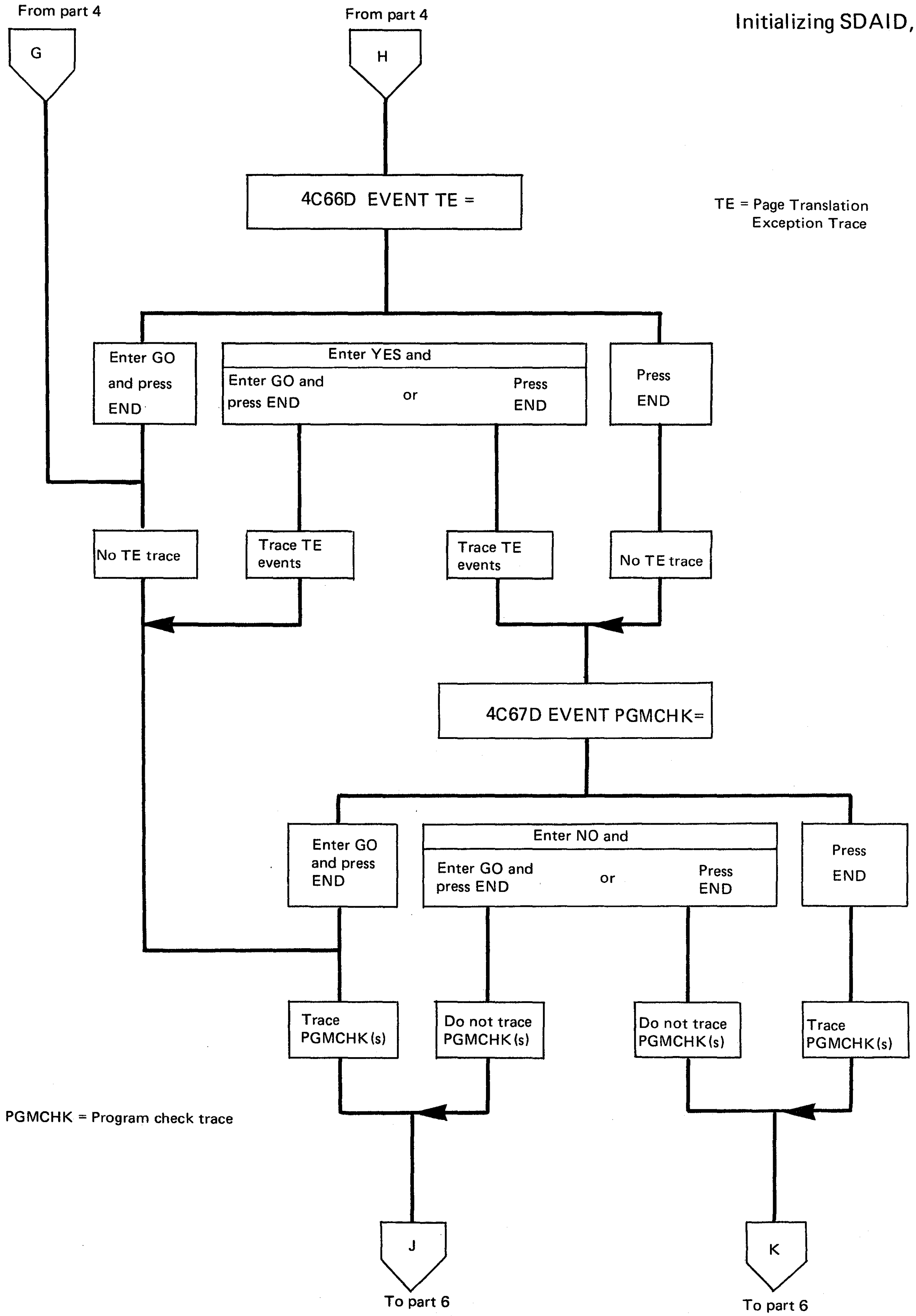
SDAIDS
Initializing SDAID, part 2 of 6



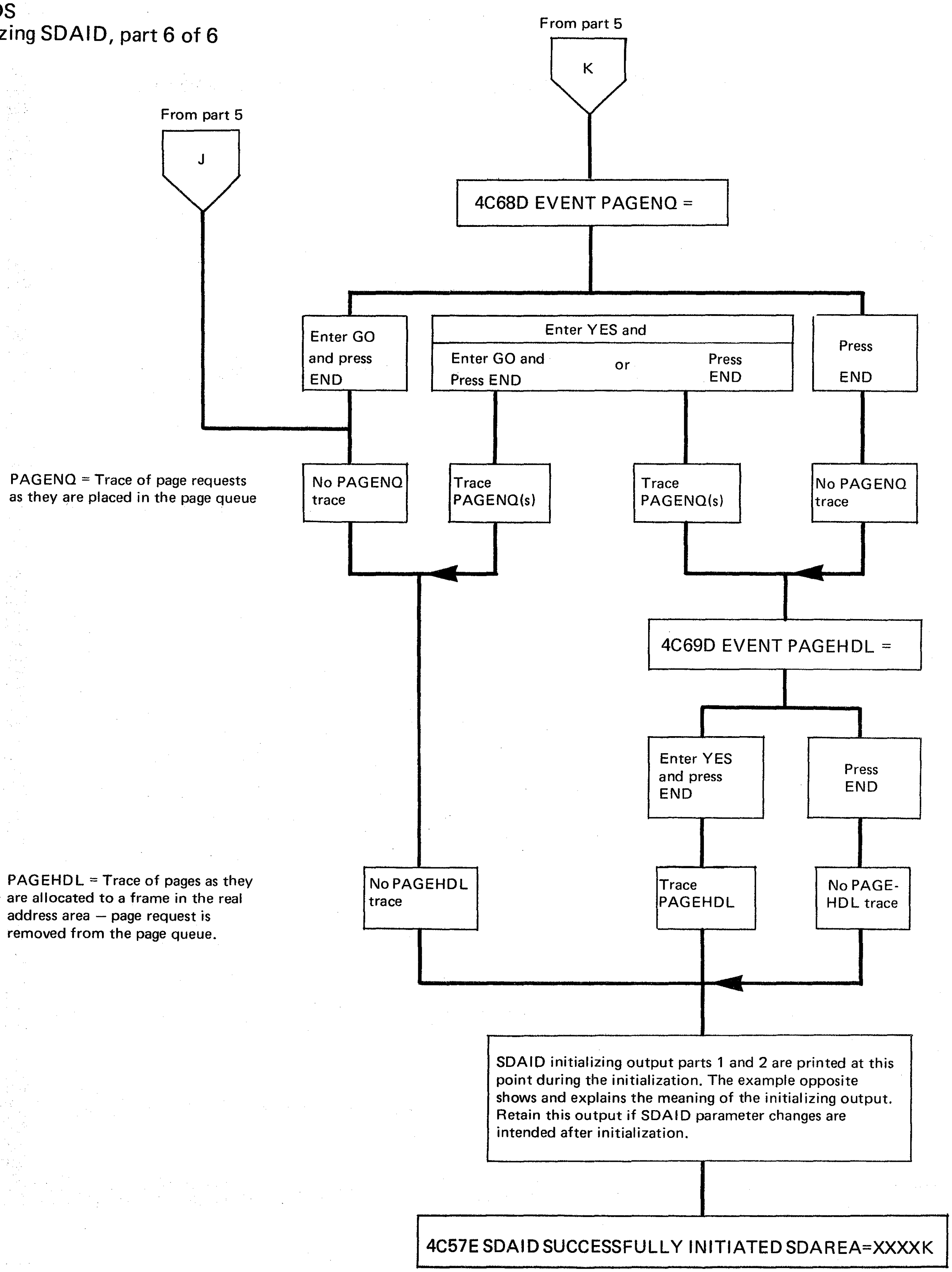

From part 2
C
From part 2
D
4C62D EVENT BR =
BR = Successful Branch Trace
Enter GO and press END
Enter YES and
Enter GO and press END
or
Press END
Press END
No BR trace
Trace BR events
Trace BR events
No BR trace
4C63D EVENT IF =
IF = Instruction Fetch Trace
Enter GO and press END
Enter YES and
Enter GO and press END
or
Press END
Press END
No IF trace
Trace IF events
Trace IF events
No IF trace
E
To part 4
F
To part 4

SDAIDS
Initializing SDAID, part 4 of 6




From part 4
G
From part 4
H
4C66D EVENT TE =
TE = Page Translation Exception Trace
Enter GO and press END
Enter YES and
Enter GO and press END
or
Press END
Press END
No TE trace
Trace TE events
Trace TE events
No TE trace
4C67D EVENT PGMCHK=
Enter GO and press END
Enter NO and
Enter GO and press END
or
Press END
Press END
Trace PGMCHK(s)
Do not trace PGMCHK(s)
Do not trace PGMCHK(s)
Trace PGMCHK(s)
PGMCHK = Program check trace
J
To part 6
K
To part 6

SDAIDS
Initializing SDAID, part 6 of 6



SDAID initializing output, part 1

```
       SDAID FOUND/ASSUMED THE FOLLOWING PARAMETERS:

    SDAID START ADDRESS IS ....................... HEX 0003E800
    SDAID END   ADDRESS IS ....................... HEX 0003FFFF
    SDBUFFER START ADDRESS IS .................... HEX 0003FDC0
    SDBUFFER END   ADDRESS IS .................... HEX 0003FFFF
    SDBUFFER CAN KEEP THIS NR OF EVENTS .......... DEC 00000009

  1 PRINTER ADDRESS IS ........................... HEX     000E
  2 OUTPUT CLASS FIELD IS ........................     PSW (01)
  3 STOP ON EVENT ................................     NO
  4 EVENT LIMITS  START .......................... HEX 00000000
                  END   .......................... HEX 00FFFFFF
  5 EVENT BR .....................................     NO
  6 EVENT IF .....................................     NO
  7 EVENT SA .....................................     NO
  8 EVENT GA .....................................     NO
    MASK BITS FOR GENERAL PURPOSE REGISTERS ......0123456789ABCDEF
                                                  0000000000000000
  9 EVENT TE .....................................     NO
 10 EVENT PGMCHK .................................     YES
 11 EVENT PAGENQ .................................     NO
 12 EVENT PAGEHDL.................................     NO
```

3FFFF − 3E800 = 17FF = 6143 dec = Default value of 6K

Default values

Last 2 bytes of CR9

Example of use:
To select GA trace on GR4 only enter 10000800 in CR9
To select GA trace on GRs 3, 5 and 7 enter 10001500

Default values

(Ex 23 KT)

SDAID initializing output, part 2

```
**** THESE ARE THE PARAMETERS THAT YOU CAN CHANGE FROM CONSOLE:           *****
*    PROCEDURE: MACHINE IN STOPPED STATE, HIT ALTER/DSPLY,  EXAMPLES BELOW:   *
*                                                                             *
*                TO ALTER CTL REG      9 TO 40000000                          *
*                .........TYPE IN  AC  9    40000000  END KEY                 *
*                                                                             *
*                TO ALTER MS LOCATION  03CC8C  TO FF                          *
*                .........TYPE IN  AM  03CC8C     FF  END KEY                 *
*                                                                             *
*                                                                             *
**1 PRINTER ADDRESS .................... ON HEX 03FD88   FORMAT 0XXX          *
*                                                                             *
**2 OUTPUT CLASS ....................... ON HEX 03FD98   FORM   UUYY          *
*                              WHERE IF: UU=00 FASTREC                        *
*                                        UU=01 PSW                            *
*                                        UU=02 GPR                            *
*                                        UU=03 LOCORE                         *
*                                        UU=04 COMREG                         *
*                                        UU=05 PAGETAB                        *
*                                        UU=06 SUPVISOR                       *
*                                        UU=07 DUMPREAL                       *
*                                        UU=08 PDUMP                          *
*                                        YY=00 AUTOMATIC SDBUFFER OUTPUT      *
*                                        YY=FF SDBUFFER OUTPUT ON PGMCHK      *
*                                                                             *
**3 STOP ON EVENT ...................... ON HEX 03FD94   X'FF'=YES, X'00'=NO  *
*   NONDESTROYING DUMP ................. ON HEX 03FD96   X'FF'=YES            *
*                                                                             *
**4 EVENT LIMITS    START ADDRESS .... CTL REG A   HEX 00XXXXXX  ADDR         *
*                   END   ADDRESS .... CTL REG B   HEX 00XXXXXX  ADDR         *
*                                                                             *
*   PDUMP LIMITS    START ADDRESS .... ON HEX 03FD9C   00XXXXXX  ADDR         *
*                   END   ADDRESS .... ON HEX 03FDA0   00XXXXXX  ADDR         *
*                                                                             *
*                                                                             *
*   SDAID EVENTS:  TO ENABLE --SWITCHON THE BIT/BYTE--                        *
*                                                                             *
**5 BR EVENT ......................... CTL REG 9   HEX 80000000  BIT          *
**6 IF EVENT ......................... CTL REG 9   HEX 40000000  BIT          *
**7 SA EVENT ......................... CTL REG 9   HEX 20000000  BIT          *
*                                                                             *
**8 GA EVENT ......................... CTL REG 9   HEX 10000000  BIT          *
*                   GPR MASK (0-15) .. CTL REG 9   HEX 1000XXXX  POSITIONAL BITS*
*                                                                             *
**9 TE EVENT .................... MAIN STORAGE 03FD8C    HEX FF  BYTE         *
*                                                                             *
*10 PGMCHK EVENT ................ MAIN STORAGE 03FD8E    HEX FF  BYTE         *
*                                                                             *
*11 PAGENQ EVENT ................ MAIN STORAGE 03FD90    HEX FF  BYTE         *
*12 PAGEHDL EVENT ................MAIN STORAGE 03FD92    HEX FF  BYTE         *
*                                                                             *
*NOTE: SDAID NEEDS EXTERNAL INTERRUPT IF PRINTER BECAME UNREADY               *
*                                                                             *
*                                                                             *
*******************************************************************************
    CPU-ID IS ................................. HEX 0001004301450OC0
4C57E  SDAID SUCCESSFULLY INITIATED  SDAREA=0006K
```

Applicable to FASTREC output only

Event limits are not applicable to events PAGENQ, PAGEHDL and PGMCHK.

Example of use:
To trace events BR and IF = 8 + 4 = C. Enter C0 in byte 0 of CR 9.
Events IF and GA = 4 + 1 = 5. Enter 50.
To select GR's for the GA trace see part 1 above.

Max length of logout area

CPU Model type

CPU Serial number

(Ex 24 KT)

The summary of parameters printed on the line printer after successful initialization of any SDAID routine.

ALL ADDRESSES ARE REAL.

SDAIDS

```
AR  1I60A  READY FOR COMMUNICATIONS.
AR  batch f2
F2  assgn sysist,x'00e'  ---------------- SDAID requested by F2
F2  // exec sdaid
F2  4C55D GIVE SPACE FOR SDAID =
F2  10k
F2  4C56E INSUFFICIENT SDAID SPACE, REALLOCATE  <-- { Page Pool not enough
F2  1I00A  READY FOR COMMUNICATIONS.                { SDAID cancels.
F2  map
F2  AREA     K-REAL UPPER LIMIT  K-VIRT UPPER LIMIT    NAME
F2
F2   SP          92K      94207             262144
F2   BG   V5A    30K     124927      292K   561151     KENSLOOP
F2   F4   I4     10K     135167       80K   643071              } MAP command
F2   F3   I3     10K     145407       80K   724991              } issued to check
F2   F2   V2A    14K     159743      100K   827391     NO NAME  } real storage
F2   F1   I1     82K     243711      120K   950271              } allocations
F2   SVA                              96K  1048575
F2   VIS                              68K  1048575
F2   PP          18K     262143
F2  allocr f1r=14k  <------------------ { Decrease F1R
F2  // exec sdaid   <------------------ { to create a
F2  4C55D GIVE SPACE FOR SDAID =         { larger page pool
F2  10k                              SDAID re-requested
F2  4C58D  OUTPUT DEVICE =
F2
F2  4C60D  STOP ON EVENT =           Output device is 00E
F2
F2  4C61D  EVENT LIMITS =
F2  x'040000',x'050fff'  <----- { Area specified to be traced
F2  4C59D  OUTCL =              { (was BG V partition)
F2
F2  4C62D  EVENT BR =   }
F2  yes                 }
F2  4C63D  EVENT IF =   }
F2  yes                 }
F2  4C64D  EVENT SA =   }---- Events selected: Successful Branch
F2                      }     Instruction, and
F2  4C65D  EVENT GA =   }     General register
F2  x'345678'           }     alteration of
F2  4C66D  EVENT TE =   }     GPRs 3,4,5,6,7 & 8.
F2
F2  4C67D  EVENT PGMCHK =      { Event program
F2                             { check selected
F2  4C68D  EVENT PAGENQ =      { by default.
F2
F2  4C69D  EVENT PAGEHDL =
F2
F2  4C57E  SDAID SUCCESSFULLY INITIATED  SDAREA=0010K
F2  1I00A  READY FOR COMMUNICATIONS.
F2  *  <-------------------------- SDAID begins immediately

DC 9
[D0001F80] [00040000] [00050FFF] 00000000 00000000 C2000000 00000200 80400040
 0000F600   FFFFFFFF   FFFFFFFF  00000000 00000000 00000000 00000000 0000FFFF

  CR 11 X'B' contains the upper event limits
  CR 10 X'A'   - " -    - " - lower   - " -    - " -
AC A  CR 9     - " -    - " - events (BR + IF + GA)
[0004043C] [000404EA]

DC 9
D0001F80  [0004043C] [000404EA] 00000000 00000000 C2000000 00000200 80400040
0000F600   FFFFFFFF   FFFFFFFF  00000000 00000000 00000000 00000000 0000FFFF

  Printer made           CR 10 and 11 altered to decrease
  READY and              the area to be traced.
  EXT INT key pressed.
F2  *
    <----------- SDAID continues
AC 9
00000000  <-- Clear CR 9 to turn off events, except event PGMCHK
                                                        (Ex25 R...)
```

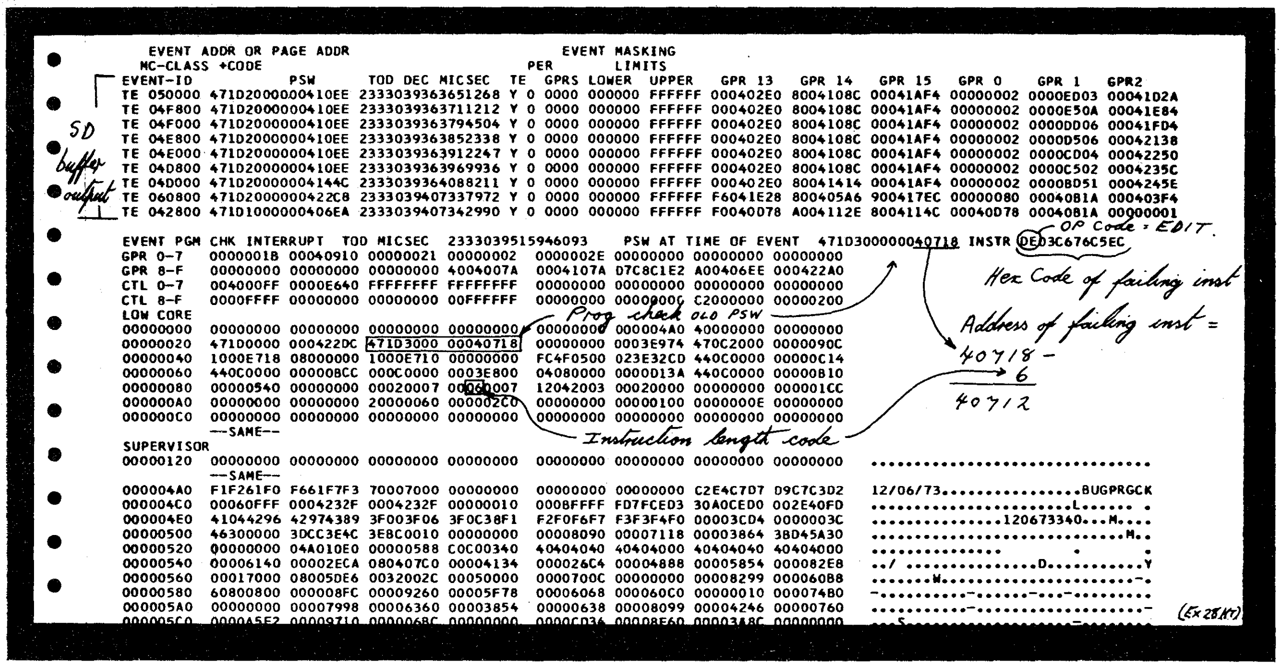
The example above shows the operator-system dialogue on a 3215 PRKB during SDAID initialization.

After initialization the contents of the control registers have been changed to alter event limits and traces.

```
  IF   GA    0004043C    TOD MICSEC   2333024913754652    PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
GPR 0-7     00000000 00D2F000 00000030 00000000  00000000 000000BE 00000010 00000002
GPR 8-F     00000001 00000833 0000000F 4004007A  0004107A D7C8C1E2 B0040426 000422F8
CTL 0-7     004000FF 0000E640 FFFFFFFF FFFFFFFF  00000000 00000000 00000000 00000000    LTR 3.3
CTL 8-F     0000FFFF D0001F80 0004043C 000404EA  00000000 00000000 C2000000 00000200    at address 4043C
LOW CORE
00000000    00000000 00000000 00000000 00000000  00000000 000004A0 470F2000 0000090C
00000020    471D3000 0004043A 471D0000 0004043E  00000000 0003B174 471D0000 0004044A    Output class PAGETAB
00000040    00000000 04000000 10003B70 00000000  00B62900 02FDB4D4 440C0000 00000C14
00000060    440C0000 00000BCC 000C0000 0003B000  04080000 0000D13A 440C0000 00000B10
00000080    00000540 00000080 0002000A 00020080  12042003 00025000 0004043C 000001CC    on events IF and GA
000000A0    Seg 0       00000 20000060 000002C0  00000000 00000100 00000009 00000000
000000C0                00000 00000000 00000000  00000000 00000000 00000000 00000000    (GA on general register 8)

SEGM TABLE   0000E640
0000E640    F000E330 F000E370 F000E3B0 F000E3F0  F000E430 F000E470 F000E4B0 F000E4F0  }  Segment  .U.O.U.O.UO
0000E660    F000E530 F000E570 F000E5B0 F000E5F0  00000001 00000001 00000001 00000001  }           ..........
PAGE TABLES  0000E330
0000E320    00000000 00000000 00000000 00000000  00000008 00100018 00200028 00300038   ................................
0000E340    00400048 00500058 00600068 00700078  ^^^^^^^^ ^^900098 00A000A8 00B000B8   . ...&...-......................
0000E360    00C000C8 00D000D8 00E000E8 00F000F8  Page 0     100118 01200128 01300138   ...H...Q...Y.O.8................
0000E380    01400148 01500158 01600168 80118011               118011 80118011 80118011   . ...&...-......................
0000E3A0    80118011 80118011 80118021 80218021  80218021 80318031 80318031 80318041   ................................
0000E3C0    80418041 80418041 80418041 80518051  80518051 80518051 80518001 80018001   ................................
0000E3E0    80018001 80018001 80018001 80018001  80018001 80018001 80018001 80018001   ................................
0000E400    80018001 80018001 80018001 80018001  80018001 80018001 80018001 038103B9   ................................
0000E420    03C103C9 03D103D9 03E103E9 03F103F9  01790171 01A10199 01890015 00150015   .A.I.J.R...Z.I.9................
0000E440    00150015 00150015 00150015 00150015  00150015 00150015 00150015 00150015              ............
            --SAME--
0000E4A0    00150015 00150015 00150015 00150015  00150015 00250025 00250025 00250025   Page Table ............
0000E4C0    00250025 00250025 00250025 00250025  00250025 00250025 00250025 00250025   ................................
            --SAME--
0000E500    00250025 00350035 00350035 00350035  00350035 00350035 00350035 00350035   ................................
0000E520    00350035 00350035 00350035 00350035  00350035 00350035 00350035 00350035   ................................
0000E540    00350035 00350035 00350035 00350035  00350035 00450045 00450045 00450045   ................................
0000E560    00450045 00450045 00450045 00450045  00450045 00450045 00450045 00450045   ................................
            --SAME--
0000E5A0    00450045 00450045 00450045 00450045  00450045 00450045 00550055 00550055   ................................
0000E5C0    00550055 00550055           550055  00550055 00550055 00550055 00550055   ................................
            --SAME--           Page Fr 0
0000E620    00550055 00550055           550055  00000000 00000000 00000000 00000000   ................................
PAGEFR TAB   0000DE88
0000DE80    00000000 00000000 00020000 00000000  00020000 00010000 00020000 00020000   ................................
0000DEA0    00020000 00030000 00020000 00040000  00020000 00050000 00020000 00060000   ................................
0000DEC0    00020000 00070000 00020000 00080000  00020000 00090000 00020000 000A0000   ................................
0000DEE0    00020000 000B0000 00020000 000C0000  00020000 000D0000 00020000 000E0000   ................................
0000DF00    00020000 000F0000 00020000 00100000  00020000 00110000 00020000 00120000   ................................
0000DF20    00020000 00130000 00020000 00140000  00020000 00150000 00020000 00160000   ................................
0000DF40    00020000 00170000 00020000 00180000  00020000 00190000 00020000 001A0000   ................................
0000DF60    00020000 001B0000 00020000 001C0000  00020000 001D0000 00020000 001E0000   ................................
0000DF80    00020000 001F0000 00020000 00200000  00020000 00210000 00020000 00220000   ................................
0000DFA0    00020000 00230000 00020000 00240000  00020000 00250000 00020000 00260000   ................................
0000DFC0    00020000 00270000 00020000 00280000  00020000 00290000 00020000 002A0000   ................................
0000DFE0    00020000 002B0000 00020000 002C0000  00020000 002D0000 000001A0 00810410   ................................
0000E000    00000418 00800188 000001A8 FFFF0190  00000178 00840418 00000180 FFFF0400   ................................
0000E020    00000410 008301A0 00000198 00820170  000001B0 FFFF0180 000001B8 FFFF01A8              ...........
0000E040    000001C0 FFFF01B0 000001C8 FFFF01B8  000001D0 FFFF01C0 000001D8 FFFF01C8   Page Frame .......Q...H
0000E060    000001E0 FFFF01D0 000001E8 FFFF01D8  000001F0 FFFF01E0 000001F8 FFFF01E8              .......8...Y
0000E080    00000200 FFFF01F0 00000208 FFFF01F8  00000210 FFFF0200 00000218 FFFF0208              ............
0000E0A0    00000220 FFFF0210 00000228 FFFF0218  00000230 FFFF0220 00000238 FFFF0228   ................................
0000E0C0    00000240 FFFF0230 00000248 FFFF0238  00000250 FFFF0240 00000258 FFFF0248   ... ...............&... ........
0000E0E0    00000260 FFFF0250 00000268 FFFF0258  00000270 FFFF0260 00000278 FFFF0268   ...-...&...............-........
0000E100    00000280 FFFF0270 00000288 FFFF0278  00000290 FFFF0280 00000298 FFFF0288   ................................
0000E120    000002A0 FFFF0290 000002A8 FFFF0298  000002B0 FFFF02A0 000002B8 FFFF02A8   ................................
0000E140    000002C0 FFFF02B0 000002C8 FFFF02B8  000002D0 FFFF02C0 000002D8 FFFF02C8   ...........H...............Q...H
0000E160    000002E0 FFFF02D0 000002E8 FFFF02D8  000002F0 FFFF02E0 000002F8 FFFF02E8   ...........Y...Q...O.......8...Y
0000E180    00000300 FFFF02F0 00000308 FFFF02F8  00000310 FFFF0300 00000318 FFFF0308   .......O.......8................
0000E1A0    00000320 FFFF0310 00000328 FFFF0318  00000330 FFFF0320 00000338 FFFF0328   ................................
0000E1C0    00000340 FFFF0330 00000348 FFFF0338  00000350 FFFF0340 00000358 FFFF0348   ... ...............&... ........
0000E1E0    00000360 FFFF0350 00000368 FFFF0358  00000370 FFFF0360 00000378 FFFF0368   ...-...&...............-........
0000E200    00000380 FFFF0370 00000388 FFFF0378  00000390 FFFF0380 00000398 FFFF0388   ................................
0000E220    000003A0 FFFF0390 000003A8 FFFF0398  00000400 FFFF03A0 000303B8 007603A8   ................................
0000E240    000303C0 007703A8 000303C8 007803A8  000303D0 007903A8 00030400 007A03A8   ...........H....................
0000E260    00030190 007B03E0 00030190 007C03E8  00030190 007D0188 00030198 007E03F8   ...............Y...............8
                                                                                          (Ex 30 KT)
```

An example of dumping the page tables and low address storage on an event.

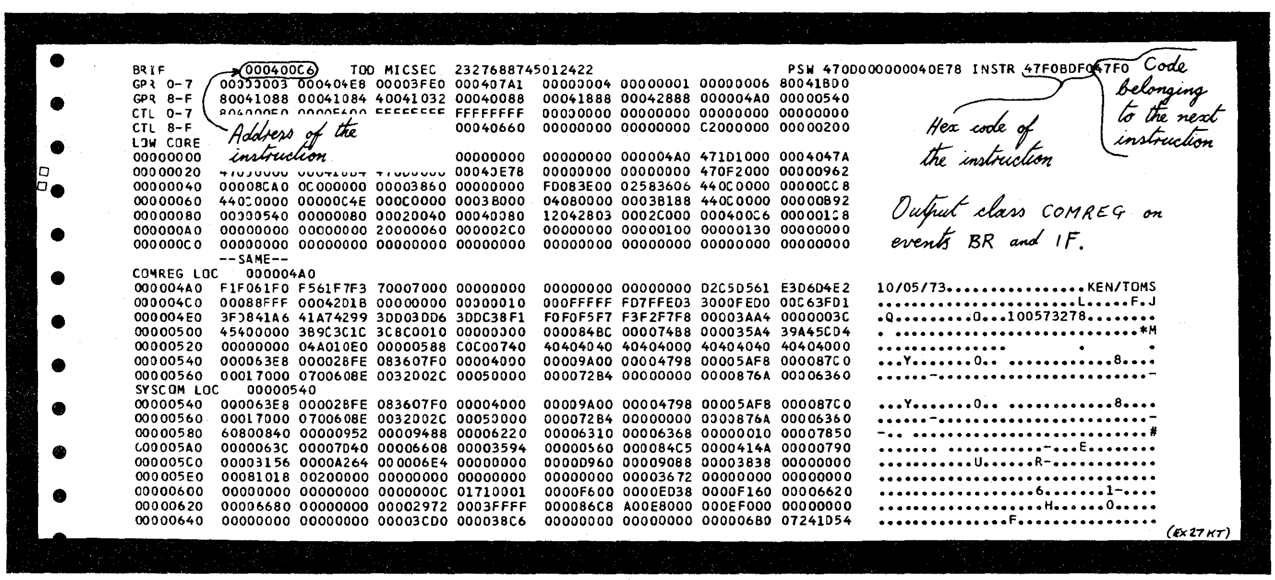


```
      EVENT ADDR OR PAGE ADDR                        EVENT MASKING
     MC-CLASS +CODE                               PER        LIMITS
   EVENT-ID            PSW        TOD DEC MICSEC  TE  GPRS LOWER  UPPER    GPR 13    GPR 14    GPR 15    GPR 0     GPR 1     GPR2
   TE 050000 471D2000000410EE 2333039363651268 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000ED03 00041D2A
   TE 04F800 471D2000000410EE 2333039363711212 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000E50A 00041E84
   TE 04F000 471D2000000410EE 2333039363794504 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000D006 00041FD4
   TE 04E800 471D2000000410EE 2333039363852338 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000D506 00042138
   TE 04E000 471D2000000410EE 2333039363912247 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000CD04 00042250
   TE 04D800 471D2000000410EE 2333039363969936 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000C502 0004235C
   TE 04D000 471D20000004144C 2333039364088211 Y 0 0000 000000 FFFFFF 000402E0 80041414 00041AF4 00000002 0000BD51 0004245E
   TE 060800 471D2000000422C8 2333039407337972 Y 0 0000 000000 FFFFFF F6041E28 800405A6 900417EC 00000080 000408IA 000403F4
   TE 042800 471D1000000406EA 2333039407342990 Y 0 0000 000000 FFFFFF F0040078 A004112E 8004114C 00040D78 000408IA 00000001

   EVENT PGM CHK INTERRUPT  TOD MICSEC  2333039515946093    PSW AT TIME OF EVENT  471D300000040718 INSTR DE03C676C5EC
   GPR 0-7    0000001B 00040910 00000021 00000002   0000002E 00000000 00000000 00000000
   GPR 8-F    00000000 00000000 00000000 4004007A   0004107A D7C8C1E2 A00406EE 000422A0
   CTL 0-7    004000FF 0000E640 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
   CTL 8-F    0000FFFF 00000000 00000000 00FFFFFF   00000000 0000000C C2000000 00000200
   LOW CORE
   00000000   00000000 00000000 00000000 00000000   00000000 000004A0 40000000 00000000
   00000020   471D0000 000422DC 471D3000 00040718   00000000 0003E974 470C2000 0000090C
   00000040   1000E718 08000000 1000E710 00000000   FC4F0500 023E32CD 440C0000 00000C14
   00000060   440C0000 00000BCC 000C0000 0003E800   04080000 0000D13A 440C0000 00000B10
   00000080   00000540 00000000 00020007 00060007   12042003 00020000 00000000 000001CC
   000000A0   00000000 00000000 20000060 000002C0   00000000 00000100 0000000E 00000000
   000000C0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
              --SAME--
   SUPERVISOR
   00000120   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................................
              --SAME--
   000004A0   F1F261F0 F661F7F3 70007000 00000000   00000000 00000000 C2E4C7D7 D9C7C3D2   12/06/73................BUGPRGCK
   000004C0   00060FFF 0004232F 0004232F 00000010   0008FFFF FD7FCED3 30A0CED0 002E40FD   .....................L...... .
   000004E0   41044296 42974389 3F003F06 3F0C38F1   F2F0F6F7 F3F3F4F0 00003CD4 0000003C   ...............120673340...M....
   00000500   46300000 3DCC3E4C 3E8C0010 00000000   00008090 00007118 00003864 3BD45A30   ............................M..
   00000520   00000000 04A010E0 00000588 C0C00340   40404040 40404000 40404040 40404000   ..............          .     .
   00000540   00006140 00002ECA 080407C0 00004134   000026C4 00004888 00005854 000082E8   ../ ................D...........Y
   00000560   00017000 08005DE6 0032002C 00050000   0000700C 00000000 00008299 000060B8   .......W......................-.
   000005A0   60800800 000008FC 00009260 00005F78   00006068 000060C0 00000010 000074B0   -.........-......-...-........
   000005A0   00000000 00007998 00006360 00003854   00000638 00008099 00004246 00000760   ..........-....................-
   000005C0   0000A5E2 00009710 0000068C 00000000   0000CD36 00008E60 00003A8C 00000000   ...S.....................-.......
```


An example showing the output to device X'00E' (a line printer) after specifying a "dump on program check" and a page translation exception trace using output class FASTREC, AUTOMATIC

Notice that the SD buffer is dumped before the dump of the supervisor and real storage. (A system dump would follow the SDAID dump if it was requested, refer to A-2 in this Section for a description of the system dump.)



```
   BRIF       000400C6     TOD MICSEC   2327688745012422                      PSW 470D000000040E78 INSTR 47F0BDF047F0
   GPR 0-7    00000003 000404E8 00003FE0 000407A1   00000004 00000001 00000006 80041BD0
   GPR 8-F    80041088 00041084 40041032 00040088   00041888 00042888 000004A0 00000540
   CTL 0-7    [illegible] FFFFFFFF   00000000 00000000 00000000 00000000
   CTL 8-F                                00040660   00000000 00000000 C2000000 00000200
   LOW CORE
   00000000                               00000000   00000000 000004A0 471D1000 0004047A
   00000020   [illegible]                 00040E78   00000000 00000000 470F2000 00000962
   00000040   00008CA0 0C000000 00003860 00000000   FD083E00 02583606 440C0000 00000CC8
   00000060   440C0000 00000C4E 000C0000 0003B000   04080000 0003B188 440C0000 00000B92
   00000080   00000540 00000080 00020040 00040080   12042803 0002C000 000400C6 000001C8
   000000A0   00000000 00000000 20000060 000002C0   00000000 00000100 00000130 00000000
   000000C0   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
              --SAME--
   COMREG LOC    000004A0
   000004A0   F1F061F0 F561F7F3 70007000 00000000   00000000 00000000 D2C5D561 E3D6D4E2   10/05/73...............KEN/TOMS
   000004C0   00088FFF 000420IB 00000000 00000010   000FFFFF FD7FFED3 3000FED0 00C63FD1   .......................L.....F.J
   000004E0   3F0841A6 41A74299 3DD03DD6 3DDC38F1   F0F0F5F7 F3F2F7F8 00003AA4 0000003C   .Q.........0...100573278........
   00000500   45400000 389C3C1C 3C8C0010 00000000   000084BC 000074B8 000035A4 39A45CD4   . ...........................*M
   00000520   00000000 04A010E0 00000588 C0C00740   40404040 40404000 40404040 40404000   ..............          .     .
   00000540   000063E8 000028FE 083607F0 00004000   00009A00 00004798 00005AF8 000087C0   ...Y.......0.. ............8....
   00000560   00017000 0700608E 0032002C 00050000   000072B4 00000000 0000876A 00006360   ......-........................-
   SYSCOM LOC    00000540
   00000540   000063E8 000028FE 083607F0 00004000   00009A00 00004798 00005AF8 000087C0   ...Y.......0.. ............8....
   00000560   00017000 0700608E 0032002C 00050000   000072B4 00000000 0000876A 00006360   ......-........................-
   00000580   60800840 00000952 00009488 00006220   00006310 00006368 00000010 00007850   -.. ............................#
   000005A0   0000063C 00007D40 00006608 00003594   00000560 000084C5 0000414A 00000790   ....... ............-....E........
   000005C0   00003156 0000A264 000006E4 00000000   0000D960 00009088 00003838 00000000   ..........U.......R-.............
   000005E0   00081018 00200000 00000000 00000000   00000000 00003672 00000000 00000000   ................................
   00000600   00000000 00000000 0000000C 01710001   0000F600 0000ED38 0000F160 00006620   ..................6.......1-....
   00000620   00006680 00000000 00002972 0003FFFF   000086C8 A00E8000 000EF000 00000000   ..................H......0.....
   00000640   00000000 00000000 00003CD0 000038C6   00000000 00000000 00000680 07241D54   ...............F...............
```


An example of dumping the active communication region and low address storage on an event.

```
// JOB DEBUGEX1                                                       DATE 12/06/73,CLOCK 11/49/33
// OPTION NODECK,LIST,XREF,LINK,LOG,DUMP
// EXEC ASSEMBLY
PAGENQ   PAGE 00040800                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00040800   FRAME 00018800 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00041000                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00041000   FRAME 00019000 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00041800                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00041800   FRAME 00019800 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00042000                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00042000   FRAME 0001A000 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00042800                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00042800   FRAME 00018000 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00043000                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00043000   FRAME 00017800 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00043800                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00043800   FRAME 0001A800 KEY 1   TASKID 1   REQUID 03   TFIX
PAGENQ   PAGE 00044000                  KEY 1   TASKID 1   REQUID 03   TFIX
PAGEHDL  PAGE 00044000   FRAME 0001B000 KEY 1   TASKID 1   REQUID 03   TFIX
TE PAGE ADDR 0005D000    TOD MICSEC  2333011788410161    PSW AT TIME OF EVENT  471D200000041632 INSTR 92FF2002943F
PAGENQ   PAGE 0005D000                  KEY 1   TASKID 1   REQUID 10   TE
PAGEHDL  PAGE 0005D000   FRAME 0001B800 KEY 1   TASKID 1   REQUID 10   TE
TE PAGE ADDR 0005F000    TOD MICSEC  2333011788621154    PSW AT TIME OF EVENT  471D00000004166E INSTR D706A000A000
PAGENQ   PAGE 0005F000                  KEY 1   TASKID 1   REQUID 10   TE
PAGEHDL  PAGE 0005F000   FRAME 0001C000 KEY 1   TASKID 1   REQUID 10   TE
TE PAGE ADDR 0005F800    TOD MICSEC  2333011790227387    PSW AT TIME OF EVENT  471D300000040644 INSTR 0EE0D501D686
PAGENQ   PAGE 0005F800                  KEY 1   TASKID 1   REQUID 10   TE
PAGEHDL  PAGE 0005F800   FRAME 0001C800 KEY 1   TASKID 1   REQUID 10   TE
TE PAGE ADDR 00060000    TOD MICSEC  2333011793770641    PSW AT TIME OF EVENT  471D300000040644 INSTR 0EE0D501D686
PAGENQ   PAGE 00060000                  KEY 1   TASKID 1   REQUID 10   TE
PAGEHDL  PAGE 00060000   FRAME 0001D000 KEY 1   TASKID 1   REQUID 10   TE
TE PAGE ADDR 00060800    TOD MICSEC  2333011796139258    PSW AT TIME OF EVENT  471D300000040644 INSTR 0EE0D501D686
PAGENQ   PAGE 00060800                  KEY 1   TASKID 1   REQUID 10   TE
PAGEHDL  PAGE 00060800   FRAME 0001D800 KEY 1   TASKID 1   REQUID 10   TE
```

SDAID Page trace output
Translation Exception,
Page Handle, and
Page enqueue
during an Assembly

(Ex 29 KT)

An example of SDAID page tracing during an assembly job.

```
   EVENT ADDR OR PAGE ADDR                        EVENT MASKING
  MC-CLASS +CODE                              PER        LIMITS
EVENT-ID           PSW         TOD DEC MICSEC TE  GPRS LOWER  UPPER   GPR 13   GPR 14   GPR 15   GPR 0    GPR 1    GPR2
TE 043000 470D000000041C62 2333035396598690 Y 0 0000 000000 FFFFFF 00042888 000004A0 00000760 00000003 00003F0C 0000410C
TE 05D000 471D200000041632 2333035416187226 Y 0 0000 000000 FFFFFF 000402E0 000414F8 00060FFE 80041934 00040C20 0005D5D7
TE 05F000 471D00000004166E 2333035416192369 Y 0 0000 000000 FFFFFF 000402E0 80041646 0005F255 80041934 00040C20 00040A7F
TE 05F800 471D300000040644 2333035417576315 Y 0 0000 000000 FFFFFF 000402E0 0005F7FD 00000019 0005F255 00000019 00040B39
TE 060000 471D300000040644 2333035420950387 Y 0 0000 000000 FFFFFF 000402E0 0005FFEA 0000001F 0005F255 0000001F 00040B3F
TE 060800 471D300000040644 2333035423146414 Y 0 0000 000000 FFFFFF 000402E0 000607F4 0000003F 0005F255 0000003F 0005F289
TE 059800 471D200000041E2C 2333035449944730 Y 0 0000 000000 FFFFFF 000402E0 00040F58 0005F255 00040A7C 00059B41 00001C7E
TE 057800 471D200000041E6E 2333035449949477 Y 0 0000 000000 FFFFFF 000402E0 00040F58 0005F255 00040A7C 00001C7E 00059B32
TE 05B800 471D300000040644 2333035453007449 Y 0 0000 000000 FFFFFF 000402E0 0005B7EE 00000039 0005D447 00000039 00000004
TE 05C000 471D300000040644 2333035453671223 Y 0 0000 000000 FFFFFF 000402E0 0005BFF6 00000021 0005D447 00000021 00000005
TE 05C800 471D300000040644 2333035454009747 Y 0 0000 000000 FFFFFF 000402E0 0005C7CA 00000046 0005D447 00000046 00000005
TE 050800 471D2000000410EE 2333035623279245 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000F504 00041BD4
TE 050000 471D2000000410EE 2333035623336942 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000ED03 00041D2A
TE 04F800 471D2000000410EE 2333035623396991 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000E50A 00041E84
TE 04F000 471D2000000410EE 2333035623480401 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000DD06 00041FD4
TE 04E800 471D2000000410EE 2333035623538305 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000D506 00042138
TE 04E000 471D2000000410EE 2333035623598321 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000CD04 00042250
TE 04D800 471D2000000410EE 2333035623656083 Y 0 0000 000000 FFFFFF 000402E0 8004108C 00041AF4 00000002 0000C502 0004235C
TE 04D000 471D20000004144C 2333035623774539 Y 0 0000 000000 FFFFFF 000402E0 80041414 00041AF4 00000002 0000BD51 0004245E
TE 060800 471D2000000422C8 2333035667085247 Y 0 0000 000000 FFFFFF F6041E28 800405A6 900417EC 00000080 00040B1A 000403F4
TE 042800 471D1000000406EA 2333035667090270 Y 0 0000 000000 FFFFFF F0040D78 A004112E 8004114C 00040D78 00040B1A 00000001
TE 043000 470D000000041C62 2333035783718692 Y 0 0000 000000 FFFFFF 00042888 000004A0 00000760 00000003 00003F0C 0000410C
SDBUFFER END
```

(Ex 26 KT)

A dump of the SD buffer after executing a page trace (TE) using the FASTREC output class. (The SD buffer is dumped on termination of SDAID).

```
  IF  GA   0004043C   TOD MICSEC  2333012672382809   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
BRIF       0004043E   TOD MICSEC  2333012672458551   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630
  IF  GA   00040448   TOD MICSEC  2333012672534442   PSW AT TIME OF EVENT  471D00000004044A INSTR 12444780B3DA
BRIF       0004044A   TOD MICSEC  2333012672610919   PSW AT TIME OF EVENT  471D000000040454 INSTR 4780B3DA0640
  IF  GA   00040454   TOD MICSEC  2333012672685341   PSW AT TIME OF EVENT  471D200000040456 INSTR 12554780B3E6
  IF       00040456   TOD MICSEC  2333012672761969   PSW AT TIME OF EVENT  471D20000004045A INSTR 4780B3E60650
  IF  GA   0004045A   TOD MICSEC  2333012672838460   PSW AT TIME OF EVENT  471D20000004045C INSTR 065047F0B3C2
BRIF       0004045C   TOD MICSEC  2333012672913531   PSW AT TIME OF EVENT  471D20000004043C INSTR 47F0B3C21266
  IF  GA   0004043C   TOD MICSEC  2333012672990157   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
BRIF       0004043E   TOD MICSEC  2333012673065877   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630
  IF  GA   00040448   TOD MICSEC  2333012673140296   PSW AT TIME OF EVENT  471D00000004044A INSTR 12444780B3DA
BRIF       0004044A   TOD MICSEC  2333012673216921   PSW AT TIME OF EVENT  471D000000040454 INSTR 4780B3DA0640  BZ CLEAR 5
  IF  GA   00040454   TOD MICSEC  2333012673293403   PSW AT TIME OF EVENT  471D200000040456 INSTR 12554780B3E6  LTR 5,5
  IF       00040456   TOD MICSEC  2333012673368510   PSW AT TIME OF EVENT  471D20000004045A INSTR 4780B3E60650  BZ CLEAR 6
  IF  GA   0004045A   TOD MICSEC  2333012673445086   PSW AT TIME OF EVENT  471D20000004045C INSTR 065047F0B3C2  BCTR 5,0
BRIF       0004045C   TOD MI                    78   PSW AT TIME OF EVENT  471D20000004043C INSTR 47F0B3C21266  B CLEAR 3
  IF  GA   0004043C   TOD MI                    98   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE  LTR 3,3
BRIF       0004043E   TOD MI                    90   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630  BZ CLEAR4
  IF  GA   00040448   TOD MI                    15   PSW AT TIME OF EVENT  471D00000004044A INSTR 12444780B3DA  LTR 4,4
BRIF       0004044A   TOD MI                    19   PSW AT TIME OF EVENT  471D000000040454 INSTR 4780B3DA0640  BZ CLEAR 5
  IF  GA   00040454   TOD MI                    91   PSW AT TIME OF EVENT  471D200000040456 INSTR 12554780B3E6
  IF       00040456   TOD MI                    52   PSW AT TIME OF EVENT  471D20000004045A INSTR 4780B3E60650
  IF  GA   0004045A   TOD MI                    31   PSW AT TIME OF EVENT  471D20000004045C INSTR 065047F0B3C2
BRIF       0004045C   TOD MI                    16   PSW AT TIME OF EVENT  471D20000004043C INSTR 47F0B3C21266
  IF  GA   0004043C   TOD MI                    37   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
BRIF       0004043E   TOD MI                    62   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630
  IF  GA   00040448   TOD MI                    96   PSW AT TIME OF EVENT  471D00000004044A INSTR 12444780B3DA
BRIF       0004044A   TOD MICSEC  2333012674430955   PSW AT TIME OF EVENT  471D000000040454 INSTR 4780B3DA0640
  IF       00040454   TOD MICSEC  2333012879517750   PSW AT TIME OF EVENT  471D200000040456 INSTR 12554780B3E6
  IF       00040456   TOD MICSEC  2333012879592055   PSW AT TIME OF EVENT  471D20000004045A INSTR 4780B3E60650
  IF       0004045A   TOD MICSEC  2333012879668631   PSW AT TIME OF EVENT  471D20000004045C INSTR 065047F0B3C2
BRIF       0004045C   TOD MICSEC  2333012879745185   PSW AT TIME OF EVENT  471D20000004043C INSTR 47F0B3C21266
  IF       0004043C   TOD M                    357   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
BRIF       0004043E   TOD M                    535   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630  BR +
  IF       00040448   TOD M                    310   PSW AT TIME OF EVENT  471D00000004044A INSTR 12444780B3DA  IF
BRIF       0004044A   TOD M                    351   PSW AT TIME OF EVENT  471D000000040454 INSTR 4780B3DA0640  Trace
  IF       00040454   TOD M                    .97   PSW AT TIME OF EVENT  471D200000040456 INSTR 12554780B3E6
  IF       00040456   TOD M                    332   PSW AT TIME OF EVENT  471D20000004045A INSTR 4780B3E60650
  IF       0004045A   TOD M                    299   PSW AT TIME OF EVENT  471D20000004045C INSTR 065047F0B3C2
BRIF       0004045C   TOD MICSEC  2333012880353883   PSW AT TIME OF EVENT  471D20000004043C INSTR 47F0B3C21266
  IF       0004043C   TOD MICSEC  2333012880427507   PSW AT TIME OF EVENT  471D00000004043E INSTR 12334780B3CE
BRIF       0004043E   TOD MICSEC  2333012880501878   PSW AT TIME OF EVENT  471D000000040448 INSTR 4780B3CE0630  (Ex 31 KT)
```

Program Loop on this address

GA trace taken off via CR 9

Address in program listing = 4044A − 3C878 = 3BD2 relocation factor

This example shows an SDAID BR, IF and GA trace used to trace a loop using output class PSW.

The GA trace was "taken off" by changing the contents of control register 9, and the programmers remarks on this example show how the information is interpreted.

CONTENTS

Library Display Programs and Utilities

LSERV

The label information cylinder is on the first full cylinder after the last system library on SYSRES. A display of all labels on the cylinder, with the exception of Data Set Secured labels, can be obtained by executing LSERV. Illustrations in this section show the location of the label information cylinder on SYSRES, and the layout of label information and record format.

System requirements

LSERV may be executed in any partition, with a minimum of 8192 bytes of the real or virtual address areas. LSERV assumes that the SYSRES label cylinder is formatted as described in *DOS/VS DASD Labels.*

Executing LSERV

The control statements necessary to execute LSERV in a virtual partition are:

From the console:

```
// EXEC LSERV
```

From the reader:

```
// JOB jobname
// EXEC LSERV
/*
/&
```

LSERV can also be executed in a real partition. The output of LSERV shows the contents of the label cylinder on the device assigned to SYSRES. The output is directed to the device assigned to SYSLST.

When and How to use

1. Operator action given in *DOS/VS Messages* indicates when LSERV must be executed.
   Programmer action, also given in *DOS/VS Messages,* explains how to use the LSERV printout.
   For example, under the message:

   ```
   0P36 NO REC FND
   ```
2. It is useful to execute LSERV prior to running a program that is known to have been run sometime in the past, but whose workfile assignments and partition allocations are unknown.
3. LSERV can be used for error analysis. LSERV displays the TLBL and the DLBL and EXTENT information contained on the SYSRES label cylinder. Information about secured data files is not displayed.

Summary of information provided

The printout of LSERV will show you the following details about the previous run:

- Whether the correct DLBL/EXTENT information is still on the label cylinder
- The permanent files
- The temporary files
- Extent type
- File type.

An example shown at the end of this chapter relates the data printed by the LSERV program to the DLBL/EXTENT job control statements.

C-1

The Label Information Cylinder

The label information cylinder contains standard and user label information for background and foreground partitions.

19 tracks on the 3330/3333, 12 on the 3340, or 20 on the 2314/2319 are allocated to the label information cylinder.

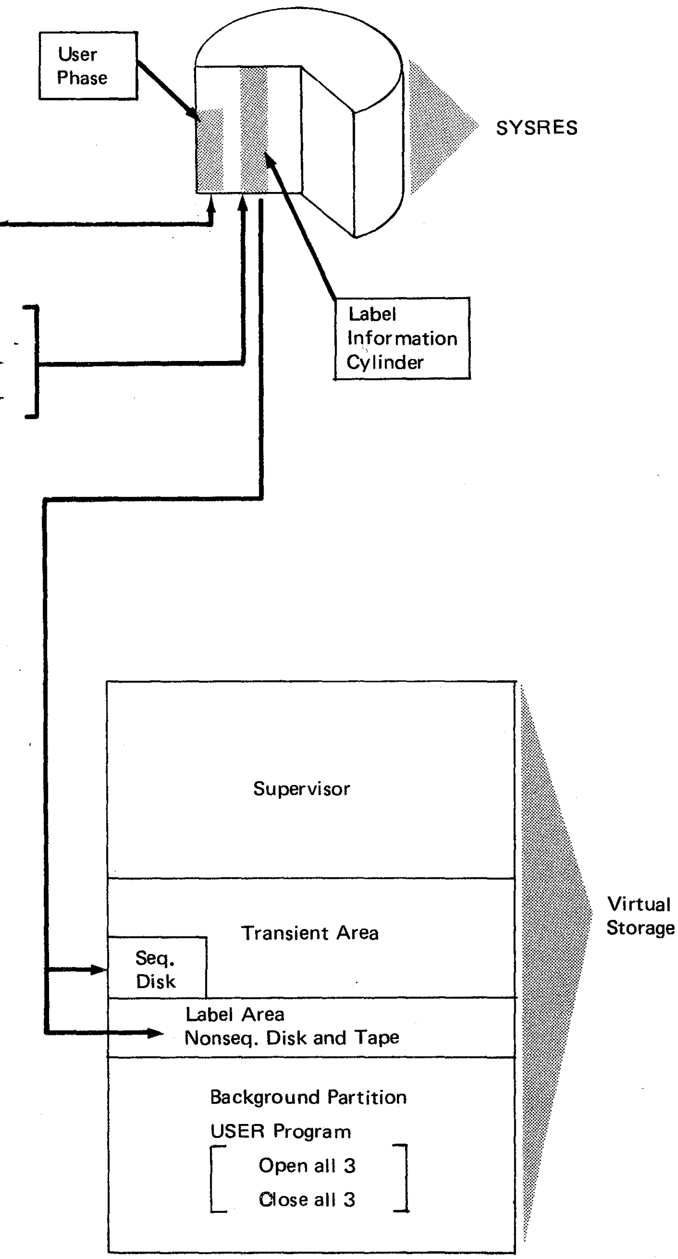
The purpose of the label cylinder is to enable label information to be placed in the label save areas during program execution.

Job control stores label information, which is contained in the input job stream, on the label cylinder.

During program execution, label information on the label cylinder is used by the OPEN and CLOSE routines.

SAMPLE JOB STREAM

```
       // JOB CATALOG USER PROGRAM
       // OPTION CATAL
        [Object module preceded ]
        [by PHASE and INCLUDE   ]
1      /*
       // LBLTYP NSD (01)
2      // EXEC LNKEDT
       /&
       // JOB EXECUTE USER PROGRAM
       // DLBL         File 1 (Seq. Disk)
       // EXTENT
3      // DLBL         File 2 (Nonseq. Disk)
       // EXTENT
       // TLBL         File 3 (Labeled Tape)
4      // ASSGN        File 1
       // ASSGN        File 2
       // ASSGN        File 3
5      // EXEC USER
       /&
```



Sequence of Events

1 LBLTYP reserves label area in virtual storage for longest label block

2 Link edit and catalog user phase to core image library

3 Write label information to label cylinder

4 Assign user files

5 Bring program into virtual storage and process OPEN/CLOSE
   a. Bring label storage data into virtual storage
   b. Bring file labels into virtual storage
   c. Compare input, create new output labels

An overview showing how DOS/VS uses the label information cylinder.

<table>
<tr><th rowspan="2">NO.</th><th rowspan="2" colspan="2">COMPONENT</th><th colspan="4">STARTING DISK ADDRESS</th><th rowspan="2">NUMBER OF TRACKS (Allocation)</th><th rowspan="2">R = REQUIRED
O = OPTIONAL</th></tr>
<tr><th>BB</th><th>CC</th><th>HH</th><th>R</th></tr>
<tr><td rowspan="4">1</td><td colspan="2">IPL Bootstrap Record 1 (SSASIPL1)</td><td>00</td><td>00</td><td>00</td><td>1</td><td rowspan="4">1</td><td>R</td></tr>
<tr><td colspan="2">IPL Bootstrap Record 2 (SSASIPLA)</td><td>00</td><td>00</td><td>00</td><td>2</td><td>R</td></tr>
<tr><td colspan="2">Volume Label</td><td>00</td><td>00</td><td>00</td><td>3</td><td>R</td></tr>
<tr><td colspan="2">User Volume Label</td><td>00</td><td>00</td><td>00</td><td>4</td><td>O</td></tr>
<tr><td rowspan="5">2</td><td rowspan="4">System Directory</td><td>Record 1</td><td>00</td><td>00</td><td>00</td><td>1</td><td rowspan="5">1</td><td>R</td></tr>
<tr><td>Record 2</td><td>00</td><td>00</td><td>00</td><td>2</td><td>R</td></tr>
<tr><td>Record 3</td><td>00</td><td>00</td><td>00</td><td>3</td><td>R</td></tr>
<tr><td>Record 4</td><td>00</td><td>00</td><td>00</td><td>4</td><td>R</td></tr>
<tr><td colspan="2">IPL Retrieval Program (SSASIPL2)</td><td>00</td><td>00</td><td>01</td><td>5</td><td>R</td></tr>
<tr><td>3</td><td colspan="2">Core Image Library Detector</td><td>00</td><td>00</td><td>02</td><td>1</td><td>*</td><td>R</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2" colspan="2">Core Image Library</td><td rowspan="2">00</td><td colspan="2">End of CI Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">R</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">5</td><td rowspan="2" colspan="2">Relocatable Library Directory</td><td rowspan="2">00</td><td colspan="2">End of CI Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2" colspan="2">Relocatable Library</td><td rowspan="2">00</td><td colspan="2">End of RL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2" colspan="2">Source Statement Library Directory</td><td rowspan="2">00</td><td colspan="2">End of RL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">8</td><td rowspan="2" colspan="2">Source Statement Library</td><td rowspan="2">00</td><td colspan="2">End of SS Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">9</td><td rowspan="2" colspan="2">Procedure Library Directory</td><td rowspan="2">00</td><td colspan="2">End of SS Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">10</td><td rowspan="2" colspan="2">Procedure Library</td><td rowspan="2">00</td><td colspan="2">End of PL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">11</td><td rowspan="2" colspan="2">Label Information Cylinder</td><td rowspan="2">00</td><td colspan="2">End of P Library</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
</table>

*Allocation Dependent On User Requirements
X = Ending CC of the Preceeding Directory
Y = Ending HH of the Preceeding Directory
Z = Ending CC of the Preceeding Library

The location of the label information cylinder on the SYSRES extent

DISK PACK TYPE: 2314 or 2319 (tracks 0–19); 3330 or 3333 (tracks 0–18); 3340 (tracks 0–11)

TRACK ALLOCATION
for a 1, or 2, or 3, partition system.
(First Full Cylinder after Last Library on SYSRES)

For a 1 partition system, information on tracks 2, 3, 4 and 5 is ignored, and for a 2 partition system the tracks for the F1 partition follows BG and information on tracks 4 and 5 is ignored

Track		
0	BG (USRLABEL) *	Temporary
1	BG (PARSTD)	Standard (Permanent)
2	F2 (USRLABEL) *	Temporary
3	F2 (PARSTD)	Standard (Permanent)
4	F1 (USRLABEL) *	Temporary
5	F1 (PARSTD)	Standard (Permanent)
6	(STDLABEL) Standard Labels for all partitions	Standard (Permanent)
7		
8		
9		
10		
11		
12		
13		
14	Additional Standard Labels	
15		
16		
17		
18		
19		

TRACK ALLOCATION
for a 4 partition system

Track		
0	BG (USRLABEL) *	Temporary
1	BG (PARSTD)	Standard (Permanent)
2	F3 (USRLABEL) *	Temporary
3	F3 (PARSTD)	Standard (Permanent)
4	F2 (USRLABEL) *	Temporary
5	F2 (PARSTD)	Standard (Permanent)
6	F1 (USRLABEL) *	Temporary
7	F1 (PARSTD)	Standard (Permanent)
8	Standard for all	Standard (Permanent)
9	Partitions	
10		
11		
12		
13		
14	Additional Standard Labels	
15		
16		
17		
18		
19		

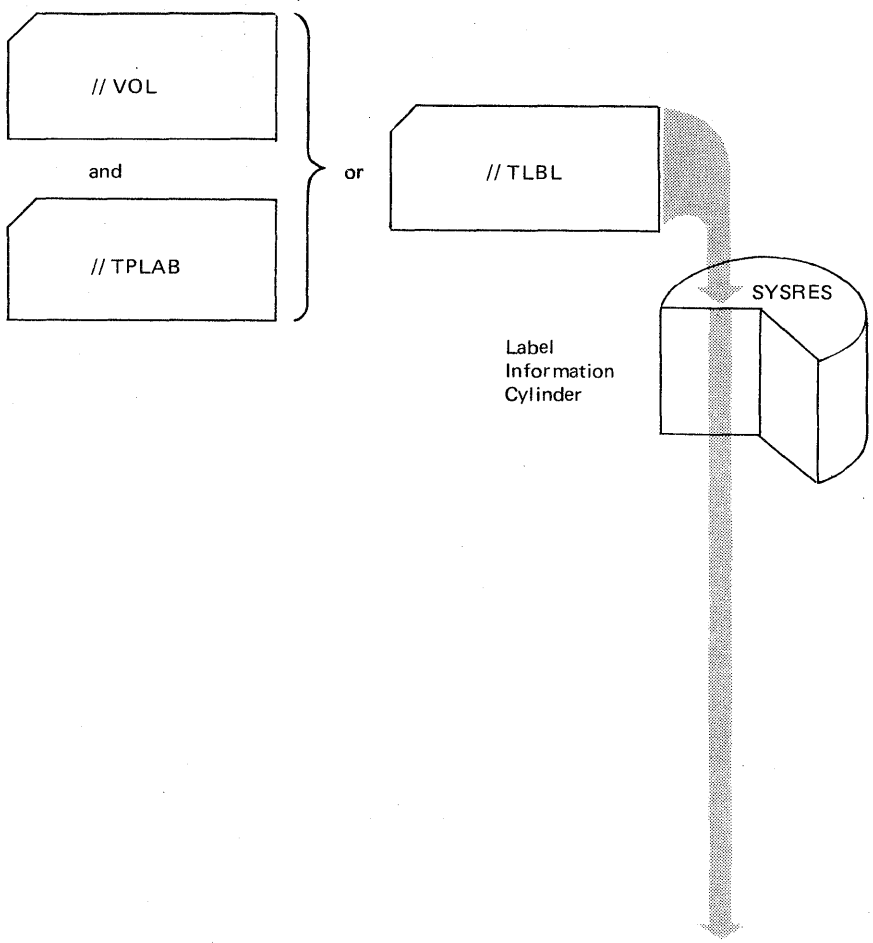
TRACK ALLOCATION
for a 5 partition system

Track		
0	BG (USRLABEL) *	Temporary
1	BG (PARSTD)	Standard (Permanent)
2	F4 (USRLABEL) *	Temporary
3	F4 (PARSTD)	Standard (Permanent)
4	F3 (USRLABEL) *	Temporary
5	F3 (PARSTD)	Standard (Permanent)
6	F2 (USRLABEL) *	Temporary
7	F2 (PARSTD)	Standard (Permanent)
8	F1 (USRLABEL) *	Temporary
9	F1 (PARSTD)	Standard (Permanent)
10		Standard (Permanent)
11		
12		
13		
14	Additional Standard Labels	
15		
16		
17		
18		
19		

*Label sets submitted in a job stream without a // OPTION PARSTD,STDLABEL are written to the temporary area for the partition being used.

Track allocations of the Label Information Cylinder for a 1, 2, 3, 4, and 5 partition system.

Job Control Statements or Operator Initiation Commands



// TLBL

// VOL

// TPLAB

Reserved Byte	Filename	Auxiliary Byte	File ID	File Serial Number	Volume Sequence Number	File Sequence Number	Generation Number	Version Number	Creation Date	Expiration Date	File Security Number	Block Count	System Code	Flag Bytes
1	7	1	17	6	4	4	4	2	6	6	1	6	13	2
0	1	8	9	26	32	36	40	44	46	52	58	59	65	78

Byte Count

Displacement in virtual storage

Format and contents of the label information cylinder for tape labels

(Shaded areas are not processed by DOS/VS Logical IOCS)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		19	Field
DLBL-EXTENT Indicator	Filename	DAM/ISAM Switch	File ID	Format ID	File Serial Number	Volume Seq. No.	Creation Date	Expiration Date	Reserved	Open Code	System Code	Volume Serial Number	Extent Type	Extent Seq. No.	Extent Lower Limit	Extent Upper Limit	Logical(Symbolic) Unit Address	2321 Lower Cell	2321 Upper Cell	Another Extent if DAM, ISAM or VSAM
1	7	1	44	1	6	2	3	3	2	1	13	6	1	1	4	4	2	1	1	Bytes
0	1	8	9	53	54	60	62	65	68	70	71	84	90	91	92	96	100	102	103	Displacement

C-1

Field	Name	Description
1.	DLBL-EXTENT	SAM Bit 0: 1 = Next extent on a new pack. Bit 1: 1 = Last extent. Bit 2: 1 = Bypass extent. Bit 3: 1 = New volume on same unit. Bit 4: 1 = Extent limits omitted. Bit 5: 1 = Extent converted to DASD address. Bit 6: 1 = No EXTENT/XTENT card. Bit 7: 1 = Unused. DAM, ISAM, or VSAM Number of extents.
2.	Filename	
3.	DAM/ISAM Switch	Bits 0-3: Unused. Bit 4: 1 = Extent limits omitted. Bit 5: 1 = Extent converted to DASD address. Bits 6 & 7: Unused.
4.	File ID	File identifier including generation and version numbers. If field is missing on DLBL card, Filename padded with blanks is inserted. This field is not used when a VSAM data space or Unique file is defined. Also, filename is not substituted for file-I-D with VSAM
5.	Format ID	Numeric 1 is inserted.
6.	File Serial No.	Volume serial number from first extent.
7.	Volume Seq. No.	Always initialized to X'0001'.
8.	Creation Date	Initialized with 3 bytes of X'00'.
9.	Expiration Date	If date is in the form YYDDD, it is converted to YDD. If date is in retention period form, 1 to 4 characters, the field is padded with binary zeros.
10.	Reserved	The retention period, if specified is converted to a 2-byte number and inserted in this field.
11.	Open Code	DLBL type: S = SAM D= DAM A= VSAM C or E = ISAM System where: C = Load create function E = Load extend function
12.	System Code	Initialized to contain one character string IBMDOSVS. This field is not processed by DOS/VS.
13.	Volume Serial No.	Volume serial number for extent.
14.	Extent Type	Same codes as in Format-1 label: X'00' = Next three fields do not indicate any extent. X'01' = Prime data area ISAM, data area (SAM,DAM) or data space (VSAM), (that is the extent containing the user's data records). X'02' = Overflow area of an ISAM file. X'04' = Cylinder index or master index of an ISAM file. X'40' = User label track area. X'8n' = Shared cylinder indicator, where n = 1,2, or 4.
15.	Extent Seq. No.	Number of extent as determined by the extent card sequence.
16. & 17.	Extent Lower and Upper Limits	Before the OPEN, DLBL/EXTENT information is in the relative track form of HHNNT followed by three bytes of binary zeros. HH = Relative (to 0) start address in tracks. NN = Number of tracks. T = 0 or upper track number for split cylinder in SAM files. Following an OPEN on DLBL/EXTENT cards, or whenever DLAB/XTENT cards are used, the extent lower and upper limits are each in the CCHH format.
18.	Logical (Symbolic) Unit Address	This 2-byte field identifies the logical unit with the same code as that used in a CCB. The first byte identifies the unit class: X'00' = System Logical Unit X'01' = Programmer Logical Unit The second byte identifies the logical unit within its class. Thus X'0003' denotes SYSLST and X'0103' denotes SYS003.
19.	2321 Lower Cell 2321 Upper Cell	2321 extent lower and upper cell limit. This 2-byte field contains zeros for disk devices.

NOTE: For SAM files, a complete 104-byte block is repeated for each new EXTENT.
For DAM, VSAM, and ISAM files, only fields 13 through 18 are repeated for each EXTENT.

Format and contents of the label information cylinder for DASD and DISKETTE labels.

```
                                        DOS LABEL CYLINDER DISPLAY

                                   SYSRES VOLUME SERIAL NUMBER - 111111

BG USER LABELS (TEMPORARY PER PARTITION) TRACK 0

   NONE

BG PARTITION STANDARD LABELS (PERMANENT) TRACK 1

   IJSYSPH
          FILE IDENTIFIER                        BUG EXAMPLES (K.TOMS, IBM UITHOORN HOLLAND)
          FILE SERIAL NUMBER                     111111
          VOLUME SEQUENCE NUMBER                 01
          CREATION DATE                          OMITTED
          EXPIRATION DATE                        73/249
          FILE TYPE                              SEQUENTIAL

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                 00
          EXTENT TYPE                            1 (PRIME DATA)
          EXTENT LOWER LIMIT                     CYLINDER     014
                                                 HEAD          00
          EXTENT UPPER LIMIT                     CYLINDER     063
                                                 HEAD          19
          SYMBOLIC UNIT                          SYSPCH      CCB FORMAT  0002
          VOLUME SERIAL NUMBER                   111111

F4 USER LABELS (TEMPORARY PER PARTITION) TRACK 2

   NONE

F4 PARTITION STANDARD LABELS (PERMANENT) TRACK 3

   IJSYSIN
          FILE IDENTIFIER                        SYSRDR40
          FILE SERIAL NUMBER                     OMITTED
          VOLUME SEQUENCE NUMBER                 01
          CREATION DATE                          OMITTED
          RETENTION PERIOD (DAYS)                0007
          FILE TYPE                              SEQUENTIAL

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                 00
          EXTENT TYPE                            1 (PRIME DATA)
          EXTENT LIMITS OMITTED
          SYMBOLIC UNIT                          SYSRDR      CCB FORMAT  0000
          VOLUME SERIAL NUMBER                   OMITTED

F3 USER LABELS (TEMPORARY PER PARTITION) TRACK 4

   NONE

F3 PARTITION STANDARD LABELS (PERMANENT) TRACK 5

   IJSYSIN
          FILE IDENTIFIER                        SYSRDR30
          FILE SERIAL NUMBER                     OMITTED
          VOLUME SEQUENCE NUMBER                 01
          CREATION DATE                          OMITTED
          RETENTION PERIOD (DAYS)                0007
          FILE TYPE                              SEQUENTIAL

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                 00
          EXTENT TYPE                            1 (PRIME DATA)
          EXTENT LIMITS OMITTED
          SYMBOLIC UNIT                          SYSRDR      CCB FORMAT  0000
          VOLUME SERIAL NUMBER                   OMITTED

F2 USER LABELS (TEMPORARY PER PARTITION) TRACK 6

   NONE

F2 PARTITION STANDARD LABELS (PERMANENT) TRACK 7

   IJSYSIN
          FILE IDENTIFIER                        SYSRDR20
          FILE SERIAL NUMBER                     OMITTED
          VOLUME SEQUENCE NUMBER                 01
          CREATION DATE                          OMITTED
          RETENTION PERIOD (DAYS)                0007
          FILE TYPE                              SEQUENTIAL
```

(Ex 32 RNT)

An example of some of the output to a line printer after executing the LSERV program, part 1.

C-1

```
      EXTENT INFORMATION
           EXTENT SEQUENCE NUMBER                                  00
           EXTENT TYPE                                             1 (PRIME DATA)
           EXTENT LIMITS OMITTED
           SYMBOLIC UNIT                                           SYSRDR       CCB FORMAT  0000
           VOLUME SERIAL NUMBER                                    OMITTED


F1 USER LABELS (TEMPORARY PER PARTITION) TRACK 8

   NONE

F1 PARTITION STANDARD LABELS (PERMANENT) TRACK 9

   IJSYSIN
           FILE IDENTIFIER                                         SYSRDR10
           FILE SERIAL NUMBER                                      OMITTED
           VOLUME SEQUENCE NUMBER                                  01
           CREATION DATE                                           OMITTED
           RETENTION PERIOD (DAYS)                                 0007
           FILE TYPE                                               SEQUENTIAL

      EXTENT INFORMATION
           EXTENT SEQUENCE NUMBER                                  00
           EXTENT TYPE                                             1 (PRIME DATA)
           EXTENT LIMITS OMITTED
           SYMBOLIC UNIT                                           SYSRDR       CCB FORMAT  0000
           VOLUME SERIAL NUMBER                                    OMITTED


 STANDARD LABELS (ALL PARTITIONS-PERMANENT) TRACKS 10-19 FOR 2314,10-18 FOR 3330, OR 10-11 FOR 3340

   IJSYSRS
           FILE IDENTIFIER                                         DOS.SYSRES.FILE
           FILE SERIAL NUMBER                                      111111
           VOLUME SEQUENCE NUMBER                                  01
           CREATION DATE                                           OMITTED
           EXPIRATION DATE                                         99/365
           FILE TYPE                                               SEQUENTIAL

      EXTENT INFORMATION
           EXTENT SEQUENCE NUMBER                                  01
           EXTENT TYPE                                             1 (PRIME DATA)
           RELATIVE (TO ZERO) START ADDRESS IN TRACKS              0001
           NUMBER OF TRACKS                                        3959
           SYMBOLIC UNIT                                           SYSRES       CCB FORMAT  0006
           VOLUME SERIAL NUMBER                                    111111

   IJSYSLN
           FILE IDENTIFIER                                         DOS.WORKFILE.NO.0
```

(Ex 33A KT)

```
                                   DOS LABEL CYLINDER DISPLAY


   STAND
           FILE IDENTIFIER                                         COMPARE-FILE FOR TESTCASEOUTPUT
           FILE SERIAL NUMBER                                      TESTV1
           VOLUME SEQUENCE NUMBER                                  01
           CREATION DATE                                           OMITTED
           EXPIRATION DATE                                         69/001
           FILE TYPE                                               DIRECT ACCESS

      EXTENT INFORMATION
           EXTENT SEQUENCE NUMBER                                  00
           EXTENT TYPE                                             1 (PRIME DATA)
           RELATIVE (TO ZERO) START ADDRESS IN TRACKS              0006
           NUMBER OF TRACKS                                        3874
           SYMBOLIC UNIT                                           SYS005       CCB FORMAT  0105
           VOLUME SERIAL NUMBER                                    TESTV1

   WORKAR
           FILE IDENTIFIER                                         WORK FILE FOR SLIB
           FILE SERIAL NUMBER                                      TESTV1
           VOLUME SEQUENCE NUMBER                                  01
           CREATION DATE                                           OMITTED
           EXPIRATION DATE                                         69/001
           FILE TYPE                                               DIRECT ACCESS

      EXTENT INFORMATION
           EXTENT SEQUENCE NUMBER                                  00
           EXTENT TYPE                                             1 (PRIME DATA)
           RELATIVE (TO ZERO) START ADDRESS IN TRACKS              3880
           NUMBER OF TRACKS                                        0100
           SYMBOLIC UNIT                                           SYS010       CCB FORMAT  010A
           VOLUME SERIAL NUMBER                                    TESTV1

 END OF LABEL CYLINDER DISPLAY
```

(Ex 33B KT)

An example of some of the output to a line printer after executing the LSERV program, part 2.

LSERV

// DLBL filename, 'file ID', date, codes, data security
// EXTENT symbolic unit, serial number, type, sequence number, relative track, number of tracks, split cylinder track, B bins

JOB CONTROL statements,
for details see *DOS/VS System Control Statements*

// DLBL IJSYSRS,'DOS.RECORDER.FILE',99/365,SD
// EXTENT SYSRES, 111111,1,1,3960,20,,

May also be printed as RETENTION PERIOD (DAYS) Depends on the manner used to specify date in the DLBL statement.

```
                                                    DOS LABEL CYLINDER DISPLAY

          FILE SERIAL NUMBER                                 111111
          VOLUME SEQUENCE NUMBER                             01
          CREATION DATE                                      OMITTED
          EXPIRATION DATE                                    99/365
          FILE TYPE                                          SEQUENTIAL

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                             01
          EXTENT TYPE                                        1 (PRIME DATA)
          RELATIVE (TO ZERO) START ADDRESS IN TRACKS         3880
          NUMBER OF TRACKS                                   0060
          SYMBOLIC UNIT                                      SYS004      CCB FORMAT  0104
          VOLUME SERIAL NUMBER                               111111

   IJSYSRC
          FILE IDENTIFIER                                    DOS.RECORDER.FILE
          FILE SERIAL NUMBER                                 111111
          VOLUME SEQUENCE NUMBER                             01
          CREATION DATE                                      OMITTED
          EXPIRATION DATE                                    99/365
          FILE TYPE                                          SEQUENTIAL

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                             01
          EXTENT TYPE                                        1 (PRIME DATA)
          EXTENT LOWER LIMIT                                 CYLINDER     198
                                                             HEAD          00
          EXTENT UPPER LIMIT                                 CYLINDER     198
                                                             HEAD          19
          SYMBOLIC UNIT                                      SYSREC      CCB FORMAT  000A
          VOLUME SERIAL NUMBER                               111111

   DIRECT
          FILE IDENTIFIER                                    NAME-LIST
          FILE SERIAL NUMBER                                 HELP#1
          VOLUME SEQUENCE NUMBER                             01
          CREATION DATE                                      OMITTED
          EXPIRATION DATE                                    69/001
          FILE TYPE                                          DIRECT ACCESS

     EXTENT INFORMATION
          EXTENT SEQUENCE NUMBER                             00
          EXTENT TYPE                                        1 (PRIME DATA)
          RELATIVE (TO ZERO) START ADDRESS IN TRACKS         0020
          NUMBER OF TRACKS                                   0003
          SYMBOLIC UNIT                                      SYS005      CCB FORMAT  0105
          VOLUME SERIAL NUMBER                               HELP#1
                                                                              (Ex 34KT)
```

Relationship between DLBL/EXTENT card data and the information printed by the LSERV program.

LIBRARY DISPLAY

Under certain circumstances knowing the contents of libraries can be helpful during program debugging. The library display programs DSERV, CSERV, PSERV, SSERV, and RSERV enable you to print an image of:

- Any library directory
- Any library
- Any program in any library
- Any phase in any library.

When using DSERV, a System Status Report is always printed before the specified directory. A private status report is also printed when private libraries are used with the system.

An example of a system status report is shown in two examples at the end of the section describing the Linkage Editor Map (E-4 of this manual).

When and how to use

Control statements required to execute the library display programs are shown in the next two tables.

The following list gives some examples of when to use the various library display programs:

1. The operator action given under the appropriate message in *DOS/VS Messages* indicate when to execute DSERV.

   For example, under the message:

   1C33A PROGRAM NOT FOUND

When error message instructions include a library display, enter cards that correspond to the library and type of display. Be sure to substitute the actual program module, book, sublibrary or phase name for the words phase 1, module 1, book 1, sublib 1, or prog 1.

Note: If you assign a private library and display that type of library, only the private library will be displayed. To obtain a display of the system library, the private library must be unassigned.

Additional information on the display program is found in *DOS/VS System Control Statements.*

Further recommendations as to when to use the library display programs are given after the two tables following.

LIBRARY DISPLAY

Control cards required to execute the Library display programs

<table>
<tr><th>Unit</th><th>Element</th><th colspan="2">Control Statements Required</th></tr>
<tr><td rowspan="4">Core Image Library</td><td>Phase</td><td colspan="2">// JOB jobname
// EXEC CSERV
DSPLY phase1 [,phase2,...]
/*
/&</td></tr>
<tr><td>Program</td><td colspan="2">// JOB jobname
// EXEC CSERV
DSPLY prog1.ALL[,prog2.ALL,...]
/*
/&</td></tr>
<tr><td>Library</td><td colspan="2">// JOB jobname
// EXEC CSERV
DSPLY ALL
/*
/&</td></tr>
<tr><td>Directory</td><td>Unsorted
// JOB jobname
// EXEC DSERV
DSPLY CD
/*
/&</td><td>Sorted
// JOB jobname
// EXEC DSERV
DSPLYS CD
/*
/&</td></tr>
<tr><td rowspan="4">Relocatable Library</td><td>Module</td><td colspan="2">// JOB jobname
// EXEC RSERV
DSPLY module1 [,module2,...]
/*
/&</td></tr>
<tr><td>Program</td><td colspan="2">// JOB jobname
// EXEC RSERV
DSPLY prog1.ALL[,prog2.ALL,...]
/*
/&</td></tr>
<tr><td>Library</td><td colspan="2">// JOB jobname
// EXEC RSERV
DSPLY ALL
/*
/&</td></tr>
<tr><td>Directory</td><td>Unsorted
// JOB jobname
// EXEC DSERV
DSPLY RD
/*
/&</td><td>Sorted
// JOB jobname
// EXEC DSERV
DSPLY RD
/*
/&</td></tr>
</table>

Table C-2 Library Display Control Cards (Part 1 or 2)

Note: To execute DSERV, SYSIN must be assigned to a card reader, a tape unit, or a disk drive. SYSLOG must be assigned to a 1052, 3210 or 3215 console printer, or for the Model 125 it must be assigned to the CRT.

Unit	Element	Control Statements Required	

Source Statement Library	Book	// JOB jobname
		// EXEC SSERV
		DSPLY sublib.book1 [,sublib.book2,...]
		/*
		/&	

	Sub-library	// JOB jobname
		// EXEC SSERV
		DSPLY sublib1.ALL[,sublib2.ALL,...]
		/*
		/&	

	Library	// JOB jobname
		// EXEC SSERV
		DSPLY ALL
		/*
		/&	

	Directory	Unsorted
		// JOB jobname
		// EXEC DSERV
		DSPLY SD
		/*
		/&
			Sorted
			// JOB name
			// EXEC DSERV
			DSPLYS SD
			/*
			/&

Procedure Library	Directory	Unsorted
		// JOB name
		// EXEC DSERV
		DSPLY PD
		/*
		/&
			Sorted
			// JOB name
			// EXEC DSERV
			DSPLYS PD
			/*
			/&

	Library	// JOB jobname
		// EXEC PSERV
		DSPLY ALL
		/*
		/&	

	Procedure	// JOB jobname
		// EXEC PSERV
		DSPLY procedure1 [,procedure2,..]	

Directories	All	Sorted
		// JOB name
		// EXEC DSERV
		DSPLY ALL
		/*
		/&
			Sorted
			// JOB jobname
			// EXEC DSERV
			DSPLYS ALL
			/*
			/&

Systems Directory		// JOB jobname
		// EXEC DSERV
		DSPLY SD
		/*
		/&	

Table C-2 Library Display Control Cards (Part 2 of 2)

LIBRARY DISPLAY

2. Execute DSERV when you require details about the core image library.

```
SYSTEM  CORE IMAGE DIRECTORY--------------------------------------   SVA----------------         10/05/73 PAGE   10

 PHASE     DISK     TXT   BTS LST  RLD   RLD      LOAD    ENTRY   PART     ENTRY   SVA    IN
 NAME      ADDR     RCDS  TXT RCD  RCDS ITEMS     ADDR    ADDR    ADDR     ADDR    ELIG   SDL

         --START ADDR<---------DEC---------------    ------------HEX--------------

          C  H  R
&&RAST13  005 00 02  001   0582                   000000  000000
&IJKS00   033 06 04  001   0536                   000000  000000
&IJKS10   033 06 05  001   0284                   000000  000000
&IJKS20   033 06 06  001   0488                   000000  000000
&IJKS30   033 07 01  001   0592                   000000  000000
&IJKS40   033 07 02  001   0392                   000000  000000
&IJKS50   033 07 03  001   0360                   000000  000000
&IJKS60   033 07 04  001   0672                   000000  000000
&IJKS70   033 07 05  001   0584                   000000  000000
&IPLRT2   005 02 06  006   0072                   000000  000C68
&IPLRT3   005 03 06  003   0248                   000C68  000C68
&IPLRT4   005 04 03  006   0416                   000C68  000C68
&IPLRT5   005 05 03  005   0240                   000C68  000C68
&JOBACCT  005 09 03  001   0002                   000000  000000
&JOBCTLA  000 17 03  009   0720                   000000  001810
&JOBCTLB  000 18 06  004   0340                   001810  001810
&JOBCTLD  000 19 04  011   0504                   001810  001810
&JOBCTLE  001 01 03  007   0610                   001810  001810
&JOBCTLF  001 02 04  006   0444                   001810  001810
&JOBCTLG  001 03 04  006   0148                   001810  001810
&JOBCTLJ  001 04 04  008   0208                   001810  001810
&JOBCTLK  001 05 06  005   0972                   001810  001810
&JOBCTLM  001 06 05  007   0136                   000000  000000
&JOBCTLN  001 07 06  001   0784                   001810  001810
&LIBSTAT  005 02 01  004   0984     001   00011   040800  040800  040078           YES
&LNKEDT   005 06 02  019   0491                   000000  003EC8
&MAINDIR  005 00 03  010   0976     000   00011   040800  040800  040078           YES
ALIST     010 12 05  012   0628     001   00100   040078  040300  040078
ALTDC     010 08 03  004   0700     000   00053   040078  04031A  040078
                                                                                                 Ex 35K7
```

PHASE NAME	The names of programs (phases)
DISK ADDR	The disk address of the phase on the core image library (Disk address of first text, TXT, record)
TST RCDS	The number of records belonging to the phase (Number of TXT records)
BTS LST TXT RCD	The number of TXT bytes in the last TXT record
RLD RCDS (see note 1)	The number of additional relocation list dictionary (RLD) records, referring to the address constants in the text that will be modified by the relocating loader
RLD ITEMS (see note 1)	Total number of RLD items that show the total number of TXT modifications due to relocating load
VER MOD LEV LEV (see note 2)	The version and modification level of phases, modules, and books in the core image, relocatable, and source statement libraries respectively.
LOAD ADDR	The load address of the phase at the time it is link-edited to core image library (Link-edit time)
ENTRY ADDR	Entry address of the phase at link-edit time
PART ADDR (see note 1)	Start address at link-edit time of the partition to which the phase is link-edited
ENTRY ADDR	The entry address in the SVA of the phase contained in the SVA. (These phases can be loaded into the SVA after IPL or by cataloging them into the SVA during system operation.)
SVA ELIG	A YES in this column indicates that it is permissible to load the phase in the SVA either after IPL time or during system operation using the linkage editor. (It indicates that the phase is reentreable and relocatable which it must be to enable its use by the system when it resides in the SVA.)
IN SDL	A YES in this column indicates that the name of the phase is in the SDL. Note: A phase name can be present in the SDL but need not necessarily be present in the system core image library.

Notes 1. Entries are printed in these columns only when a relocatable phase is found in the library.

2. Version and modification levels are always listed for modules and books displayed, but are listed for phases only when displaying a specific phase. This information is required under some conditions of system malfunctions that may be caused by the use of programs at different levels of modification.

Most IBM-supplied programs have a 2-byte VM (version and modification level) number. The number may be in decimal or hexadecimal form in a storage dump, depending on the input format. It is in decimal form in a DSERV printout of the source statement or relocatable library. For example, version 5 modification level 0 appears at 2800 or F2F8 in a storage dump and a 5.0 in a DSERV printout. The VMs for phases and transients are contained within the phase or transient.

Your IBM CE/FE can also check your library by using DSERV to examine it for the applicability of an IBM-supplied program temporary fix (PTF).

The modification level of your library is also required if an authorized program analysis report (APAR) must be submitted to IBM for analysis of a particularly difficult programming error.

Similar information is displayed by the DSERV program for the following directories:

- Transient
- Relocatable
- Source statement
- Procedure.

3. Use CSERV to display a phase and compare it with the program listings for the phase when, for example, a program using that phase produces incorrect results after having run successfully in the past. RLD information for any relocatable phases will also be displayed.
4. Use PSERV to check job streams cataloged in the procedure library if they are suspected of causing program errors. Before modifying any job stream on this library, use PSERV to enable checks to be carried out on the modifications. This helps to reduce errors when using the overwrite function of the catalog procedure.

Note: The procedure library itself can be used as a secondary debugging aid. It is useful to catalog job streams that initiate or execute the more frequently used debugging aids, or aids requiring complex job streams. For example, time will be saved if EREP or trace routine job streams are kept on the procedure library.

5. Use RSERV when, for example, a modification must be made (via REP cards) to an object module in the relocatable library.
   An example of an RSERV output is shown below:

```
PDAIDGTW  V.M     29.0                00004 BLOCKS      SYSTEM  RELOCATABLE LIBRARY
 PHASE PDAIDGTW,+0,NOAUTO
          ESD             016  0001               PC  000060  00017C
          TXT     000060  136  0001   07FA070A 07FA070A 47F09120 42C0910B 58C00080 9500C045 4770909A 9500910B
                                      078A9500 910B078A 9500910B 078A9500 910B078A 9500910B 078A9078 910C4700
                                      90CA4180 903B4170 000695FF 80004780 90CAD500 8000008B 478090C0 468090C6
                                      9878910C 07FA4670 90AA5870 9118D207 70000020 90017008 D200700C 008B5880
                                      00144880 805A4A80
          TXT     0000E8  040  0001   910AD201 70108002 41770012 5970911C 47409100 58709114 50709118 9878910C
                                      07FA0000 00000012
          TXT     000114  136  0001   00000000 00000000 00000000 92009069 D200907D 9048D200 90839049 D2009089
                                      904AD200 908F904B D2009095 904C95FF 903B4770 915492F0 909F47F0 917090CA
                                      9180905B 47109170 D2019080 90BAD201 908A9152 927090BD 928090C7 9500907D
                                      4770917E D201907A 91DA9057 912095FF 903C4780 91C05850 903C5860 9040D203
                                      90549040 1B651876
          TXT     00019C  064  0001   1B665D60 910C4C70 910E4B70 910E1A75 50509114 50509118 5070911C 98579120
                                      47F091CC 41509120 58690000 47F09198 58B09114 D711B00D B00047F0 906CA000
          END     404040
```

(Ex 36K7)

6. Use SSERV to examine the contents of a book. Use it, for example, to check if changes are required to privately written macros cataloged into the source statement library, in unedited macro format.

ESERV

This service program de-edits macros contained in the source statement library. When the source macro definition is not available, ESERV enables you to de-edit the edited macro to source format. Several macros can be de-edited at one time. ESERV also combines the function of de-editing with that of updating the source macro definition.

The ESERV program allows you to:

- De-edit and update a macro definition
- Obtain a printout of the de-edited macro definition
- Obtain a printout and a punched card deck of the de-edited macro definitions
- Verify and update statements from a printout of the source macro definition.

For a detailed description of the control statements used by this program, refer to the *Guide to the DOS/VS Assembler.*

Input consists of ESERV control statements entered via SYSIPT, and the edited macro definition in the macro library.

Output consists of:

- The selected macro definitions in source format (and updated) on the device assigned to SYSPCH and/or the device assigned to SYSLST.

Note: To allow immediate editing of the updated macro, an END card can be generated at the end of the update run.

- Update information
- Error diagnostics.

The illustration opposite shows the input and output of the ESERV program, and includes a list of control statements for the program.

How to use

Six control statements are available to update the de-edited source macro definitions. These are described in the table below:

COL:	for specifying the columns containing the sequence numbers in the statements of a macro definition
ADD:	for adding source statements to a macro definition
VER:	for verifying the contents of a specific source statement of a macro definition
DEL:	for deleting source statements from a macro definition
REP:	for replacing source statements in a macro definition
RST:	for specifying a new number sequence
END:	for indicating the end of an update to a macro definition.

Note: Each update control statement is printed on SYSLST followed by (except for RST) the affected source statement. This printout is known as the update survey. For ADD, the de-edited source record preceding the records to be added is printed, followed by the added records. For VER, the character string to be compared is printed, followed by the de-edited source record to be verified. For DEL, the de-edited source records to be deleted are printed. For REP, the de-edited source records to be replaced are printed, followed by the records to be inserted.

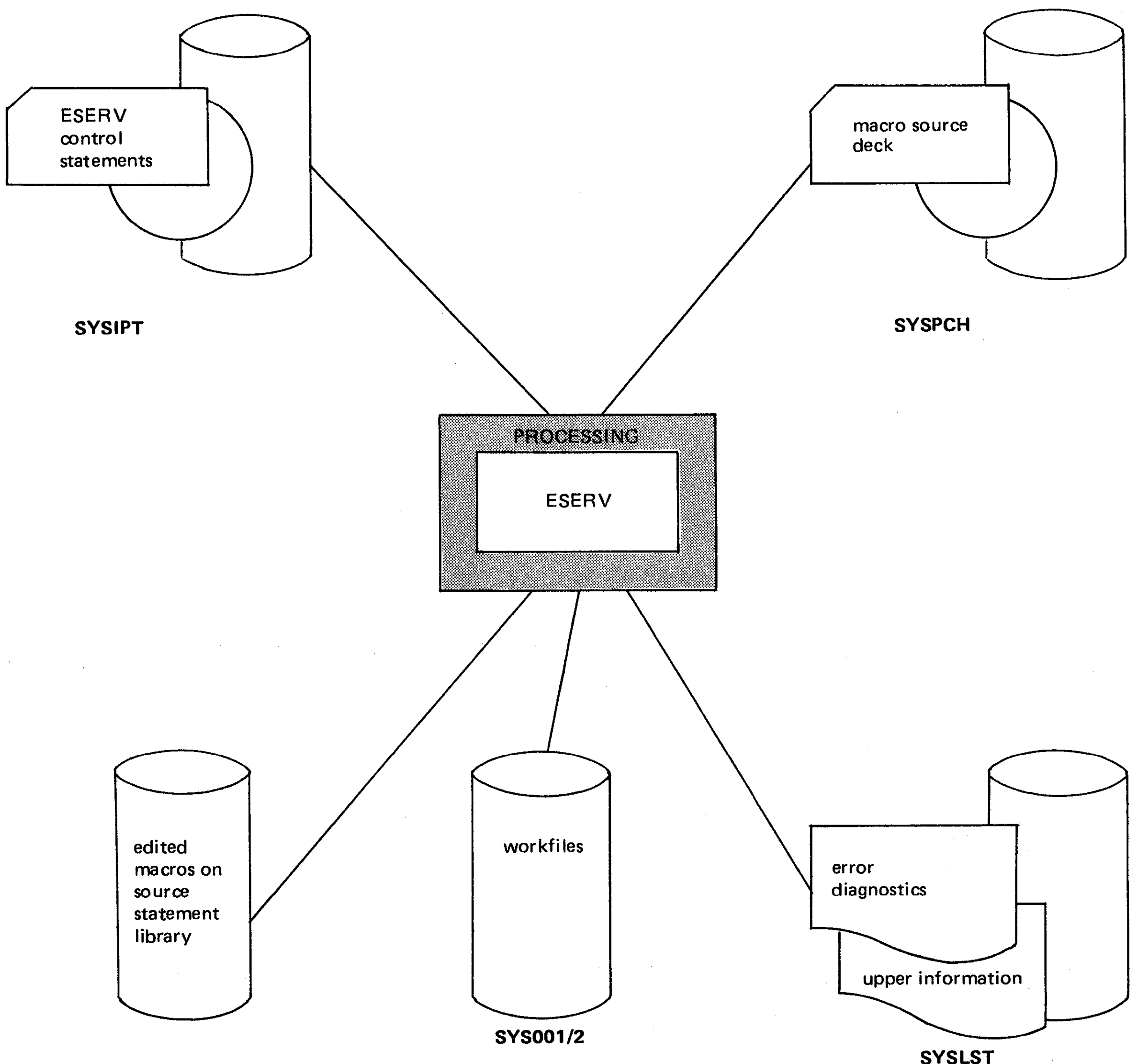
Errors During Update

If an error is detected during updating, a message is printed on the update survey. The requested update action will not be performed. If possible, updating will continue with the next update control card. Otherwise, a termination message is given and only the remaining update control cards are printed on the survey. De-editing of the macro will always be completed. Th e job will be cancelled at the end of the ESERV run; that is, remaining jobsteps to edit the macro and catalog the EDECK again will not be executed.

Updating will continue with the next update control card for all errors except when:

- The COL statement has invalid operands.
- COL statement is not the first update control statement.
- The macro is completely de-edited without all update control statements being completely processed.
- An RST statement has an invalid operand.

Appendix D shows two ESERV job stream examples.



This illustrates the input to and output from the ESERV

When to use

- Use ESERV to punch up a new card deck.
- ESERV can be used to list the source code of an edited macro.
- If an IBM program tempory fix (PTF) is to be installed in your library, use ESERV to de-edit and update the macro. An example of installing a PTF using ESERV is given in appendix D.

Note: Before installing a PTF use either ESERV or SSERV to display the macro in order to check if the PTF is applicable.

LVTOC

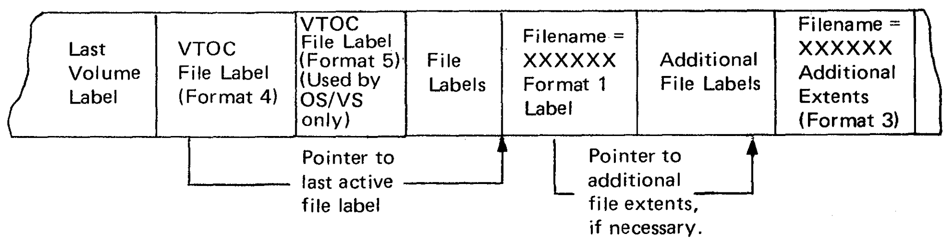
The LVTOC program enables you to print out VTOC (volume table of contents) of a DASD disc pack.

From the printout (the VTOC display), you can see the names of files, contained on any disc pack, their extents, and addresses. A VTOC display, therefore, enables you to keep track of data sets and files on all your packs.

Additional information on the VTOC display program can be obtained from *DOS/VS System Control Statements*

Information in the VTOC

All standard file labels are grouped together and stored in a specific area on a disk pack or data cell. This group of labels essentially a directory of all data records on the volume because each file label contains file limits. Therefore, this group of labels is called the volume table of contents, or VTOC. Because the VTOC itself is a file of records containing one or more standard label records for each logical file, it is defined as such with its own file lable.



Function of the VTOC

Before a DASD file can be processed by logical IOCS, the file must be opened to permit transfer of data. The open routines check the DASD labels identifying the file. This is accomplished by comparing the information from the actual file labels in the VTOC with the label information in the SYSRES label information cylinder. (See LSERV in this Section for a description of the label cylinder.)

The illustration opposite is a overview of how DOS/VS uses the VTOC.

DASD Label Formats

The VTOC contains all format labels. Each format label points to an area of DASD storage on the volume and indicates what the area is currently being used for. A format 1 label describes one to three physical areas (extents) on the volume. It is the first format label used to describe each file.

A format 2 label describes a file as being indexed sequential. If a format 2 label is used, there is always a format 1 label describing the same file.

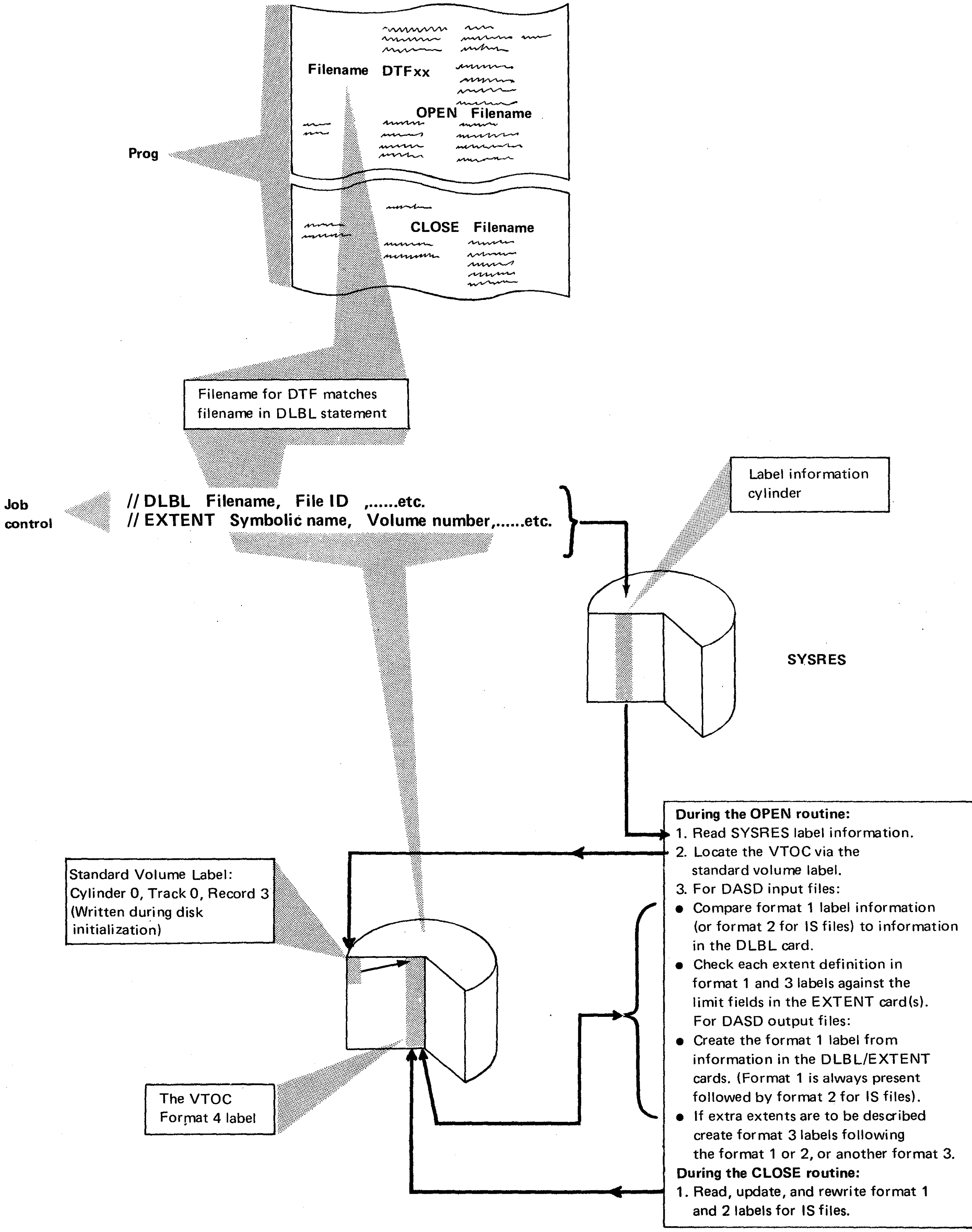
A format 3 label describes from one to thirteen extents on the volume. It is used when a file is made up of four to sixteen extents (the format 3 label is always associated with a format 1 label).

A format 4 label describes the VTOC.

A format 5 label is not used by DOS/VS.

The following illustrations show the layout of format labels 1 to 4, with examples of printouts from the VTOC display program.

A detailed description of these label formats is given in *DOS/VS Data Management Guide.*



An overview showing how the DOS/VS uses the VTOC. (Not applicable to VSAM files) Details are found in DOS LIOCS Vol 1

LVTOC

Executing the VTOC display program

The control cards necessary to execute VTOC in a virtual partition are:

```
// JOB jobname
// ASSGN SYS004,X'cuu' (input)
// ASSGN SYS005,X'cuu' (output)
// EXEC LVTOC
/&
```

The operator commands necessary are:

1. Press REQUEST on the console printer keyboard
2. PAUSE (BG F4, F3, F2, F1), EOJ
3. // ASSGN SYS004,X'cuu' (input device)
4. // ASSGN SYS005,X'cuu' (output device)
5. // EXEC LVTOC

Where:
- SYS004 is assigned to the channel and unit address (cuu) of the DASD on which the disk pack is mounted.
- SYS005 is the output device, normally a printer. If the output device is not a printer, TLBL, DLBL, and EXTENT cards must be included to describe the output device. The filename for these cards is UOUT.

The first of the two examples on the opposite page is a VTOC display using the LVTOC program.

Two other methods of obtaining a VTOC display are as follows:
- Instead of typing CANCEL to terminate the job, the operator can type CANCELV to get a VTOC dump on SYSLST, if SYSLST is a printer. Refer to the second example on the opposite page.
- The operator can display the VTOC by typing DSPLYV (in response to an error message). This reply does not terminate the job, but reissues the same message issued prior to the VTOC display request. (The output can be directed to SYSLOG if SYSLST is not assigned.) Refer to the example at the end of this chapter.

Operator actions given in *DOS/VS Messages* indicate methods to obtain a VTOC display for particular messages.

When to use

The five examples listed below illustrate when and how the VTOC display program can aid program debugging by providing details about your disk volumes:
1. During disk pack initialization, the VTOC label is checked. A message is printed on SYSLOG if there is an unexpired file in the pack. If the contents of the pack are unknown or its validity is in doubt, a VTOC listing will enable you to check the unexpired file. You can then decide if the unexpired file is to be retained or replaced by different extents.
2. Before copying a volume it is useful to keep a record of the contents of the volume to be copied and the volume that receives the copy. Having a record will reduce debugging time if an error occurs in a program that uses one or both of these packs.
3. A VTOC display enables you to monitor and keep track of volume areas, thus allowing economical use of your packs.
4. If the input data contained on a pack is causing program errors, a VTOC listing of the input volumes enables you to check for the presence (or absence) of data sets.
5. If, during program execution, a system malfunction prevents workfiles from being properly closed by the CLOSE macro, it is probably that volumes used

during the program will cause program errors when used for future jobs. This is because the VTOC label would not have been updated by the CLOSE macro. It is useful, therefore, to obtain a VTOC listing of the volumes affected.

Examples of the LVTOC output

```
                                              VTOC DISPLAY UTILITY

      LABEL IDENT - VOL1     VTOC LOCATION IS CYLINDER - 199   SERIAL NUMBER - 111111     OWNER IDENT -

  T 0,R 1 FORMAT 4    LAST ACTIVE     UNUSED       ALTERNATE TRACK INFORMATION          EXTENT INFORMATION
                       FORMAT 1       LABELS         LOCATION     NO.OF TRACKS          SEQ  TYPE  LOW   HIGH

                                       473          200    0            60               0  PRIME 199  0 199 18

  T 0,R 2 FORMAT 5    USED ONLY BY O/S 360,NOT SUPPORTED BY DOS

  T 0,R 3 FORMAT 1          SERIAL NUMBER - 111111      LENGTHS             DATES          SYSTEM     EXTENT INFORMATION
                            FILE ID                  KEY BLOCK  RCD   CREATE  EXPIRE       IDENT    SEQ  TYPE  LOW   HIGH

  PAGE DATA SET                                        0   2048  2048  73 278  99 365 IBMDOSVS       0          180  0 186 19

  T 0,R 4 FORMAT 1          SERIAL NUMBER - 111111      LENGTHS             DATES          SYSTEM     EXTENT INFORMATION
                            FILE ID                  KEY BLOCK  RCD   CREATE  EXPIRE       IDENT    SEQ  TYPE  LOW   HIGH

  DOS.RECORDER.FILE                                    0      0     0  73 278  99 365 IBMDOSVS       1  PRIME 198  0 198 19

  T 0,R 5 FORMAT 1          SERIAL NUMBER - 111111      LENGTHS             DATES          SYSTEM     EXTENT INFORMATION
                            FILE ID                  KEY BLOCK  RCD   CREATE  EXPIRE       IDENT    SEQ  TYPE  LOW   HIGH

  BUG.GENERATOR,WORKFILE.K.TOMS  File Identifier       0      0     0  73 278  73 249 IBMDOSVS       1  PRIME  21 14  41 13
                                 (max = 44 characters)                                                          CY  HD  CY  HD

  T 0,R 8 FORMAT 1          SERIAL NUMBER - 111111      LENGTHS             DATES          SYSTEM     EXTENT INFORMATION
                            FILE ID                  KEY BLOCK  RCD   CREATE  EXPIRE       IDENT    SEQ  TYPE  LOW   HIGH

  DOS.WORKFILE.NO.0                                    0      0     0  73 278  99 365 IBMDOSVS       1  PRIME   1  0  20 19

  END OF VTOC LISTING                               Applicable to ISAM files
                                                                                                               (Ex 37 KT)
```

An example of the output on a line printer after executing the LVTOC program

```
  VOLUME SERIAL NO. IS 111111                                                                                 12/06/73

  00C7000001    FORMAT 4 LABEL

  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  04040404  F4000000
  000001F2  00C80003  001B8001  000000CB  00141C7E  92202D01  02161911  00000000  00000000  00000000  00000000  00000000
  00000000  00000000  00010000  C7000000  C7001300  00000000  00000000  00000000  00000000  00000000  00000000

  00C7C00002    FORMAT 5 LABEL

  05050505  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  F5000000
  00000000  00000000  0000000C  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000

  00C7000003    FORMAT 1 LABEL

  DOS.RECORDER.FILE                        SERIAL NO. 111111  VOL NO. 0001  490154-63016D  010700
                                                                                      SYS. CODE IS  IBMDOSVS
  00000000  00000040  00000000  00000000  00000000  00000000  00000000  0000
  0101 00C60000-C0C60013  0000 00000000-00000000  0000 00000000-00000000                 POINTER IS  0000000000

  00C7000004    FORMAT 1 LABEL

  DOS RECORDER FILE                        SERIAL NO. 111111  VOL NO. 0001  490034-63016D  010700
                                                                                      SYS. CODE IS  IBMDOSVS
  00000000  00000040  00000000  00000000  00000000  00000000  00000000  0000
  0101 00C40000-C0C40013  0000 00000000-00000000  0000 00000000-00000000                 POINTER IS  0000000000
                                                                               Creation date
  00C7000006    FORMAT 1 LABEL     File identifier                                   Expiration date
  PAGE DATA SET       Lower limit                SERIAL NO. 111111  VOL NO. 0001  490154-63016D  014040
                                                                                      SYS. CODE IS  IBMDOSVS
  40404040  40404000  00000008  00080000  00008040  40404000  00000000  4040
  2100 00B40000-C0B80013  0000 00000000-00000000  0000 00000000-00000000                 POINTER IS  0000000000
        CYL HD   CY HD      Upper limit of extent                                  Pointer to format 2 label
  VTOC LISTING COMPLETED                                                           for ISAM files or format 3
      Extent sequence number                                                       for multiple extent files
                                                                                                               (Ex 38 KT)
```

An example of the output on a line printer after issuing the CANCELV command

```
BG
BG   ASSGN SYSPCH,X'131'
BG   4444A OVERLAP ON UNEXPRD FILE      IJSYSPH   SYSPCH=131   111111
     BUG.GENERATOR,WORKFILE.K.TOMS
BG   dsplyv
BG   4V95A SYSLOG OR SYSLST
BG   syslog
BG

BG   VOLUME SERIAL NO. IS 111111                                       10/04/13

BG   PAGE DATA SET                                 1111111    0001    0D0115-63016D
BG   2100 00B40000-00BA0013

BG   DOS.RECORDER.FILE                             1111111    0001    0D0115-63016D
BG   0101 00C60000-00C60013

BG   BUG.GENERATOR,WORKFILE.K.TOMS                 1111111    0001    0D0115-4900F9
BG   0101 0015000E-0029000D
               CY  HD   CY  HD
BG   DOS.WORKFILE.NO.0                             1111111    0001    0D0115-63016D
BG   0101 00010000-00140013

BG   VTOC DISPLAY COMPLETED
                                                                               Ex39KT
```

DSPLYV command

output to SYSLOG

Lower limit

File identifier

Creation date

Expiration date

Higher limit of extent

Volume Serial Number

Volume

Extent Sequence Number

An example of the output on a 3215 console printer keyboard after issuing the DSPLYV command.

```
VOLUME SERIAL NO. IS 111111                                    12/06/73

DOS.RECORDER.FILE                        1111111    0001    490154-63016D
0101 00C60000-00C60013

DOS RECORDER FILE                        1111111    0001    490034-63016D
0101 00C40000-00C40013

PAGE DATA SET                            1111111    0001    490154-63016D
2100 00B40000-00B80013

VTOC DISPLAY COMPLETED
                                                                  (Ex 40 KT)
```

Upper limit of extent

Creation date

Lower limit.

Expiration date

An example of the output on a line printer after issuing the DSPLYV command.

Intentionally Blank

C-5

Intentionally Blank

The SYSVIS DUMP program copies the contents of the page data set (PDS) contained on the system logical unit SYSVIS on to magnetic tape or disk pack. A printout on SYSLST can then be obtained for use during offline program debugging. The utility also enables you to dump the contents of the PDS directly to SYSLST, which can be assigned to a tape unit, a disk drive, or a line printer.

The SYSVIS dump may also be referred to as the Page Data set dump.

Restrictions

This utility program can be used only to copy or dump the contents of the PDS contained on SYSVIS. Any other use is automatically rejected by the system. Because paging must not occur during execution of this utility when dumping from SYSVIS do not start or run any other jobs either before or during its execution.

Description and operation

This utility program is initiated by normal JCL through SYSLOG or SYSIPT by the execute statement // EXEC PDSDM. Parameters entered either through SYSLOG or SYSIPT enable you to select areas of SYSVIS, thus avoiding the need to dump all of the virtual address area contained on SYSVIS.

The following areas can be selected:

- The entire PDS, that is, all the virtual address area
- Any specified virtual partition
- One or more pages contained within any virtual partition.

Multiple parameters can be specified but they must be confined to one card image. Multiple cards are possible.

For example:

BG, (089ABC,08ABCD), F4 punched in a card or entered through SYSLOG causes a dump of the whole of the background and foreground 4 virtual partitions, and the pages on the PDS that contain any addresses between the address limits 089ABC and 08ABCD.

(Addresses are specified by six hexadecimal digits.)

The dump output is directed to SYSLST or SYS001, depending on the parameters specified. For example, a response to SYSLST to the message 0V23D TO= causes the dump to be directed to the device assigned as SYSLST.

If SYS001 is used as the input or output device, tape or disk label information must be supplied in the job stream.

If the dump is from SYSVIS, it is accessed by assigning SYS000 to it. The necessary disk label information must then also be supplied in the job stream.

Job stream examples are shown on the following pages.

The format of the dump output is similar to the output obtained from the stand-alone dump, that is, each 2K of virtual storage contained on the PDS is separated and given a block number starting with BLOCK 0000. Blocks containing only zeros are suppressed. An example of the stand-alone dump output is shown in Appendix G.

SYSVIS DUMP

How to execute

Because this utility consists effectively of three separate utility programs, it is necessary to show three sample job streams.

Example 1 shows the job stream required to copy SYSVIS to SYS001, where SYS001 can be assigned to either a tape unit or a disk drive.

Example 2 shows the job stream required to dump SYS001 to the device assigned to SYSLST.

Example 3 shows the job stream required dump SYSVIS directly to the device assigned to SYSLST.

To ensure that the contents of the PDS and the allocations of the virtual address area are the same as they were just prior to the execution of the stand-alone dump the following instructions must be adhered to:

1. Re-IPL using identical parameters for the DPD command as specified in the previous system IPL. However you must specify N to the parameter TYPE=.
2. Check for any previous ALLOC commands. You must specify identical virtual partition allocations as existed just before the stand-alone dump was executed.

Example 1: Copying SYSVIS to SYS001 on tape or disk.
(SYS001 must be a DASD device)

// JOB COPYPDS
// ASSGN SYS000,X'cuu' — where CUU is the physical address of SYSVIS.

// ASSGN SYS001,X'cuu' — where CUU is the physical address of the device to be used as temporary storage for the PDS.

// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// DLBL S01DISK,'BACKUP FOR PDS' [,date]
// EXTENT SYS001,vol ID, ,relative starting address, number of tracks

If the PDS copy is to be on tape, replace the previous two statements by the following:

// TLBL S01TAPE,'BACKUP FOR PDS'

followed by:

// EXEC PDSDM — where PDSDM is the phase name for the utility contained on the system core image library.

Example 2: Dumping SYS001 to SYSLST

```
// JOB DUMPPDS
// ASSGN SYS001,X'cuu'
```

where CUU is the physical address of the device containing the copied PDS.

```
// DLBL PDSDISK,'BACKUP FOR PDS'
// EXTENT SYS001
```

If the copied PDS is on tape, replace the previous two statements by the following:

```
// TLBL PDSTAPE,'BACKUP FOR PDS'
```

followed by:

```
// EXEC PDSDM
```

If parameters are to be read through SYSIPT respond to message 0V20D with IPT, press the END key and use the following statement:

TO=SYSLST, T followed by the cards containing the parameters and

/&

Respond to message 0V20D with LOG and press END if parameters are to be read through SYSLOG.

Only pressing the END key as the answer to message 0V20D causes a dump of the whole PDS to SYSLST.

If LOG is entered followed by END key the following message is issued on SYSLOG:

0V23D TO=

Respond to this with:

SYSLST,T — This selects SYS001 as input device and SYSLST as output device for the dump.

This is followed by the message:

0V21D GIVE PARAMETERS

Pressing the END key after entering parameters causes an immediate dump of the areas specified followed by the message:

0V21D GIVE PARAMETERS

Further parameters can be entered but if no more areas of the PDS are to be dumped, either enter EOJ or press the END key. This terminates the job.

Note: On Models 115 and 125 the END key is replaced by the ENTER key.

C-6

SYSVIS DUMP

Example 3: Dumping SYSVIS direct to SYSLST.

```
// JOB DUMPPDS
// ASSGN SYS000,X'cuu'        where CUU is the physical address of
                              SYSVIS.
// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// EXEC PDSDM
```

If parameters are to be read through SYSIPT this must be followed by:

```
TO=SYSLST    followed by the cards containing the parameters and
/&
```

Respond to message 0V20D with either LOG or IPT and press the END key. (END key only is an invalid response.) If LOG is entered followed by END key the following message is printed on SYSLOG:

```
0V23D TO=
```

Respond to this with SYSLST. This selects SYSVIS as input device and SYSLST as output device for the dump.

This is followed by the message:

```
0V21D GIVE PARAMETERS
```

Pressing the END key after entering parameters causes an immediate dump of the specified areas of SYSVIS and is followed by the message:

```
0V21D GIVE PARAMETERS
```

Further parameters can be entered, but if no more areas of SYSVIS are to be dumped either enter EOJ or press the END key to terminate the job.

Pressing the END key before entering parameters causes the whole PDS to be dumped on SYSLST.

Note: On Models 115 and 125 the END key is replaced by the ENTER key.

Error messages

The list below summarizes the error messages that are printed on SYSLOG to inform the operator of incorrect job stream input:

- Invalid parameters are specified.
- SYSLST or SYS001 is incorrectly specified.
- Start address is greater than end address.
- Partition is not allocated.
- Address or partition is in real storage.
- Address is greater than end of virtual storage.
- Partition ID is invalid or greater than number of partitions allocated.
- Incorrect assignments for SYS000 and/or SYS001.
- Attempt to dump a file other than the PDS.

Incorrect addresses and partition IDs are flagged by an asterisk* printed on the line below. For example:

```
BG bg,0809ab,f4,148000,05f5ee,(0809ab,096000)f2,
BG                        *
BG 0V40I ADDRESS IS OUTSIDE OF VIRTUAL PARTITIONS
```

Pressing the END key causes the areas that are specified correctly to be dumped up to the first invalid parameter. The incorrect parameters can be corrected through SYSLOG. If the input is through SYSLOG, further parameters can be specified after the message:

```
0V21D GIVE PARAMETERS
```

If the input is through SYSIPT, you can switch back to SYSIPT as input device for specification of further parameters by entering IPT to the message:

```
0V21D GIVE PARAMETERS
```

Terminating the dump

This can be done in any of the three ways given below:

- Enter EOJ on SYSLOG
- Having a /* or a /& card at the end of the job stream when entering parameters through SYSIPT.
- Pressing the END key in response to the message: 0V21D GIVE PARAMETERS after at least one address has been processed.

Note: On Models 115 and 125 the END key is replaced by the ENTER key.

C-6

SYSVIS DUMP

When to execute the dump

It is recommended to obtain a dump of SYSVIS whenever a stand-alone dump is executed. SYSVIS DUMP should not be executed until the stand-alone dump has been completed, and should be initiated during the system re-IPL. To help your analysis of the information contained in the SYSVIS dump it is also recommended to execute a formatted stand-alone dump as described in A-3 of this Section. For this reason, execution of this utility is included in the flowchart A-3-F "Executing the Stand Alone Dump".

How to use the dump output

During analysis of a system malfunction, such as a HARD WAIT STATE using a stand-alone dump output, it may be necessary to analyze the coding in a page belonging to a virtual partition which was not in real storage when the stand-alone dump was executed.

The virtual address allocations can be obtained from the BOUNDARY BOX, and pages not in real storage can be found by analyzing the contents of the PAGE TABLE. The format and contents of the boundary box and the page table are described in Section 4, Chapter 12 of this manual.

Therefore, the SYSVIS dump should be used in conjunction with the stand-alone dump output. It is recommended to always use a stand-alone dump generated with the DUMPGEN parameter FORMAT=YES. DUMPGEN is described in A-3 of this Section.

It is essential that the operator save the copy of the PDS after executing the stand-alone dump. You as the system programmer, or the IBM CE/SE will then be able to print out all or any part of the PDS to complete problem analysis.

CONTENTS

Aids provided by the Operator's Console

Aids provided by the Operator's Console

ALTER/DISPLAY FEATURE (ALL MODELS)

The ALTER/DISPLAY facility allows the operator to dump or display, and change the contents of various parts of the CPU storage (depending on the CPU Model) and of real or virtual storage.
For the purpose of hands-on debugging, the following areas may need to be displayed:

- Any selected area of virtual storage
- General registers
- Floating-point registers
- Current PSW
- Control registers

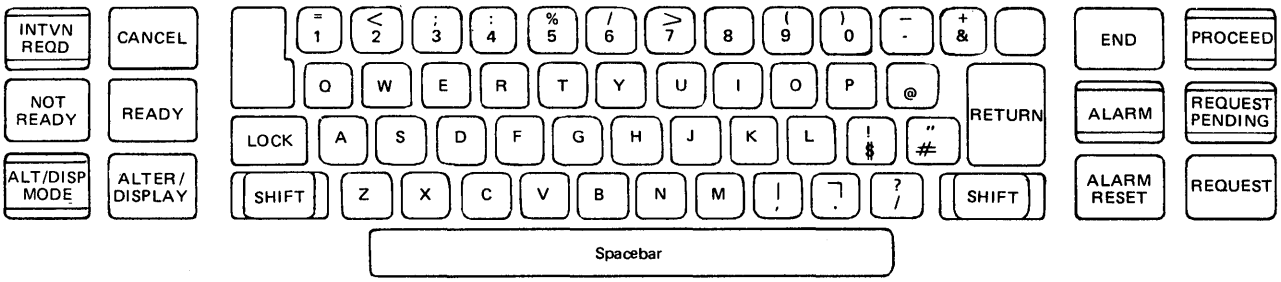
When to use

ALTER/DISPLAY is useful for hands-on debugging, and enables the operator to obtain information about the system at the time a malfunction occurs. It must be used whenever a display of the low address storage is required, for example, to record a wait state message (see E-3 in this section).

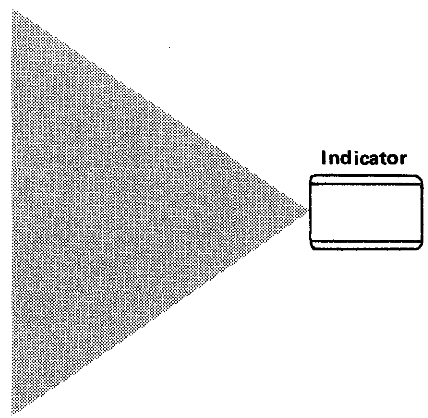
Flowcharts in section 3 indicate when to use this facility, and which option to choose for a particular system malfunction

CAUTION

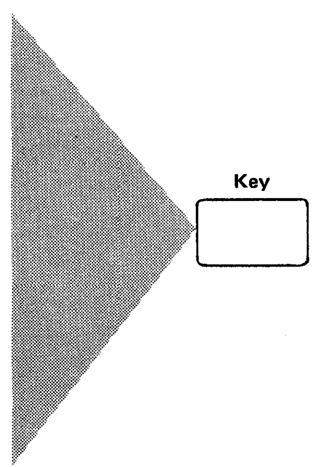
The effect on the operation of programs currently running in the system that are time dependent, for example, a program using MICR or teleprocessing as input/output, must be considered before using this serviceability aid.



Indicator	Condition
INTVN REQD	The console printer is out of forms or the PR-KB is not ready.
ALT/DISP MODE	A request for an alter/display operation was accepted.
ALARM	An alarm command was issued, and manual intervention is required by the operator.
PROCEED	The PR-KB is unlocked and ready to accept characters. This indicator is turned on by the ALTER/DISPLAY key or by a read command.
REQUEST PENDING	A request operation was initiated. The indicator is turned off when the attention status is accepted by the CPU.



Key	Function
NOT READY	Places the console printer in a not-ready condition.
CANCEL	Used to terminate a read command when the operator has made an error in data entry. Normally, the program will issue the same read command again.
READY	Places the PR-KB in the ready state when forms are in the printer and the cover is closed.
ALTER/ DISPLAY	Requests or ends an alter/display operation. When used to end an alter/display operation, the PR-KB remains in alter /display mode.
END	Terminates a read, write, or alter/display operation.
ALARM RESET	Resets the ALARM indicator.
REQUEST	Initiates the attention routine to enable operator/ system communication



Indicators and Control Keys (3210 and 3215 printer keyboard)

ALTER/DISPLAY
MODELS 135 145
AND 115-11

Table D-1 below summarizes the ALTER/DISPLAY options that can be selected when using the flowchart shown in D-1-F.

Mnemonic		Storage Area	Address Range
Alter	Display		
AM	DM	Real address area	000000–03BFFF*
AV	DV	Virtual address area	000000–16 megabytes
†	DS	Control storage	0000–DFFE*+
AG	DG	General register	0–F
AF	DF	Floating-point register	0,2,4,6
AP	DP	PSW	None
AC	DC	Control register	0–F
AK	DK	Storage key	000000–03BFFF*
AX	DX	Transmission speed††	1–8 (line number)

Use address length shown; if necessary, fill-up with leading zeros.

* Model-dependent.

† You cannot alter control storage data.

+ Control storage addresses are not continuous. For control storage address to be valid, leftmost (fourth-highest) digit must be:
    1. For 24K control storage size; 0–5
    2. For 36K control storage size, 0–5, 8, A, or D(hex)
    3. For 48K control storage size, 0–5 or 8–D(hex).

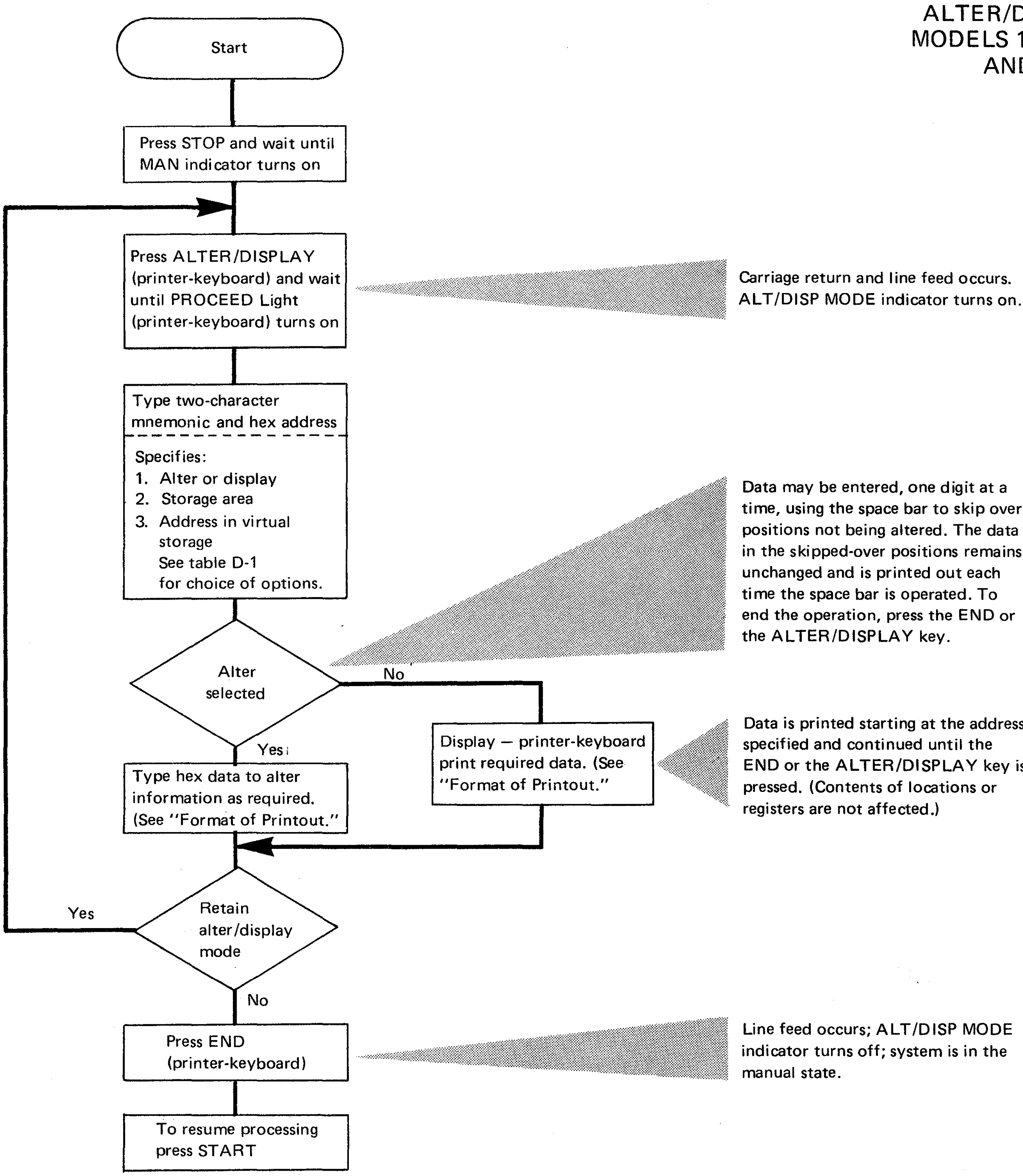
†† Line speed can only be changed if, with your ICA feature, you have the SDA II subfeature with clocking provided by the Model 135. 0 = 600 bits per second, 1 = 1200 bits per second.

Table D-1. Options for the ALTER/DISPLAY feature.

Notes: When the operation is ended with the ALTER/DISPLAY key, the keyboard remains in ALTER/DISPLAY mode (ALT/DISP MODE indicator on). When the operation is terminated with the END key, ALTER/DISPLAY mode is terminated.

For ALTER/DISPLAY of general and floating-point registers, a wraparound is performed (F to 0 for general registers, and 6 to 0 for floating-point registers). When addressing virtual storage, either a word or byte address may be used. If the starting address is not on a word boundary, the console printer spaces and aligns at the byte addressed.

ALTER/DISPLAY MODELS 135, 145, AND 155-11



ALTER/DISPLAY MODELS 135, 145, AND 155-II

Format of printout

Starting at the specified address, the display is printed in groups of eight characters with up to eight groups per line. Depending on your starting address, the initial group might not contain eight characters. When general and floating-point registers are displayed, the address sequence 'wraps,' that is, the highest available address is followed by the lowest address (zero).

When altering, enter new hex characters in the positions occupied by the characters to be replaced. Reach the required positions by repeating characters to be retained.

Examples are shown below of the printout (reduced in size) from a 3215 console printer by using the ALTER/DISPLAY feature.

Example 1
This example shows a display of the
- current PSW (DP)
- general purpose registers (DG)
- control registers and
- low address storage.

```
DP
074D0000  00089ABC
DC
004000FF  0000E640  FFFFFFFF  FFFFFFFF  00000000  00000000  00000000  00000000
0000FFFF  00000000  00000000  00000000  00000000  00000000  C2000000  00000200

DG
00089A7C  00089E80  00089E78  00089EAC  00089E80  00000019  A0089EDC  0008A157
00089978  00089E80  0009E447  40089F5A  4008AD9A  000892E0  80089EF0  00089AA0

DM 000000
00000000  00000000  00000000  00000000  00000000  00004450  00000000  00000000
074D0000  00089B62  040C0000  000009D2  00000000  00000000  070F2000  0000090C
4000E6F0  0C000000  400070B8  00000000  FAE40800  020AFE5D  040C0000  00000C14
040C0000  00000BCC  000C0000  0000A5DC  04080000  0000D13A  040C0000  00000B10
00000540  00000000  00020007  00040000  4009D840  00020000  00000000  000001CC
00000000  00000000  20000060  000002C0  00000000  00000100  00000130  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000  00000000  00000000
```

(Ex 41 KY)

Example 2
In this example, the contents of control registers 9, 10, and 11 were altered. First the operator displayed the contents by using the DC option. Then using the AC option, he entered the new data. To ensure that the data change was successful, the operator has displayed the control registers again.

```
DC 9
0F123D00  00000000  00000000  00000000  00000000  C2000000  00000200  004000FF
0000E640  FFFFFFFF  FFFFFFFF  00000000  00000000  00000000  00000000  0000E000

AC 9
000000AA  AA896300  DEF12301

DC 9
000000AA  AA896300  DEF12301  00000000  00000000  C2000000  00000200  004000FF
0000E640  FFFFFFFF  FFFFFFFF  00000000  00000000  00000000  00000000  0000E000
```

(Ex 42 KY)

Error Messages

An ALTER/DISPLAY operation is terminated when an ALTER/DISPLAY error message occurs or when the end of a storage area or register is reached.

Model 135 Alter/Display Error

Invalid character: An invalid character is created when you use the mnemonic not shown in the table, when you address a feature not installed on your system, or when you enter an address or data character that is not a hex digit. An invalid character is ignored (no print or space occurs). Continue by entering the correct character—it is not necessary to restart the whole operation.

Invalid address: An invalid address is created when your address is not addressable location (the address might be outside the storage capacity of your system or you may be trying to address a virtual address that is not in the real address area) or when you address an ICA line either not installed or not fitted with the SDA II subfeature with clocking by the Model 135. An invalid address terminates the operation with the message '?ADR.' You must start again.

Invalid Data: When changing the transmission speed for a communications link (AX or DX mnemonics), the only valid data characters are '0' or '1.' When any other hex character is entered, the operation is terminated with the message '?DATA.' The transmission speed remains unaltered.

Invalid-Format PSW: When you enter an invalid-format PSW, the PSW is altered but an interruption is generated when the invalid PSW is subsequently used.

Model 145 Alter/Display Error

Invalid Character: INVAL CHAR is printed if one of the following occurs:
- The first character of a mnemonic is not A, D, or T (see Keyboard Test Mode Operation).
- The second character is not M, S, L, K, C, G, F, or P. S and L are reserved for service personnel.
- An invalid digit is typed when addressing or altering data.
- The CANCEL key is pressed.

Invalid Address: INVAL ADDR is printed if one of the following errors occurs:
- Invalid starting address.
- The updated address exceeds the capacity of specified storage.
- The operator performs an AS or AL operation.
- You may be trying to address a virtual address that is not in the real address area.

Model 115-11

As a result of the editing function, the following indications are given:
1. If an invalid character is detected in the op code, storage mnemonic, or hex digit (0-9 and A-F), the printer does not respond. The operator can then rekey the correct character.
2. If an invalid address (beyond the physical storage) is detected the error message '?' is printed.
3. If an alter PSW operation is invalid, the PSW is restored to its original value and '?' is printed.

Aids provided by the Operator's Console

ALTER/DISPLAY MODELS 115 AND 125

The ALTER/DISPLAY facility allows the operator to display or change the contents of the following parts of the CPU (Central Processing Unit), and of real or virtual storage areas:

- General registers
- Floating-point registers
- Current PSW
- Control registers
- Protection keys
- Real storage areas
- Virtual storage areas.

A "hard copy" of all information displayed on SYSLOG can be obtained on a Model 115 and 125 with a 5213 printer attached by pressing the COPY key after the information is displayed.

CAUTION

The effect on the operation of programs currently running in the system that are time dependent, for example a program using MICR or teleprocessing as input/ output, must be considered before using this serviceability aid.

How to use

Before the ALTER/DISPLAY feature can be used, the mode select display shown below must be brought to the screen by pressing the MODE SELECT key.

```
                    * MODE SELECTION *

    R  SYSTEM RESET            A  ALTER/DISPLAY
    C  ADDRESS COMPARE         I  INSTRUCTION STEP
    L  PROGRAM LOAD            P  RESTART
    T  INTERVAL TIMER          M  MAINTENANCE
    K  CHECK-CONTROL           S  STORE STATUS
    D  STORAGE DUMP            U  SAVE USAGE COUNTERS
    E  ICA LINE MODES
MODE SPECIFICATION:
```

To select the ALTER/DISPLAY feature:
1. Type A into the mode select display.
2. Press the ENTER key.

The ALTER/DISPLAY picture as shown below is brought to the screen and shows those parts of the CPU and real/virtual address areas that can be altered and/or displayed.

```
                    * ALTER/DISPLAY *

    G  GENERAL REGISTERS
    C  CONTROL REGISTERS
    P  CURRENT PSW
    F  FLOATING POINT REGISTERS    STORAGE ADDRESS
    K  PROTECTION KEY              000000 -- FFFFFF
    M  MAIN STORAGE REAL           000000 -- FFFFFF
    V  MAIN STORAGE VIRTUAL        000000 -- FFFFFF

  MODE SPECIFICATION:         ADDRESS:
```

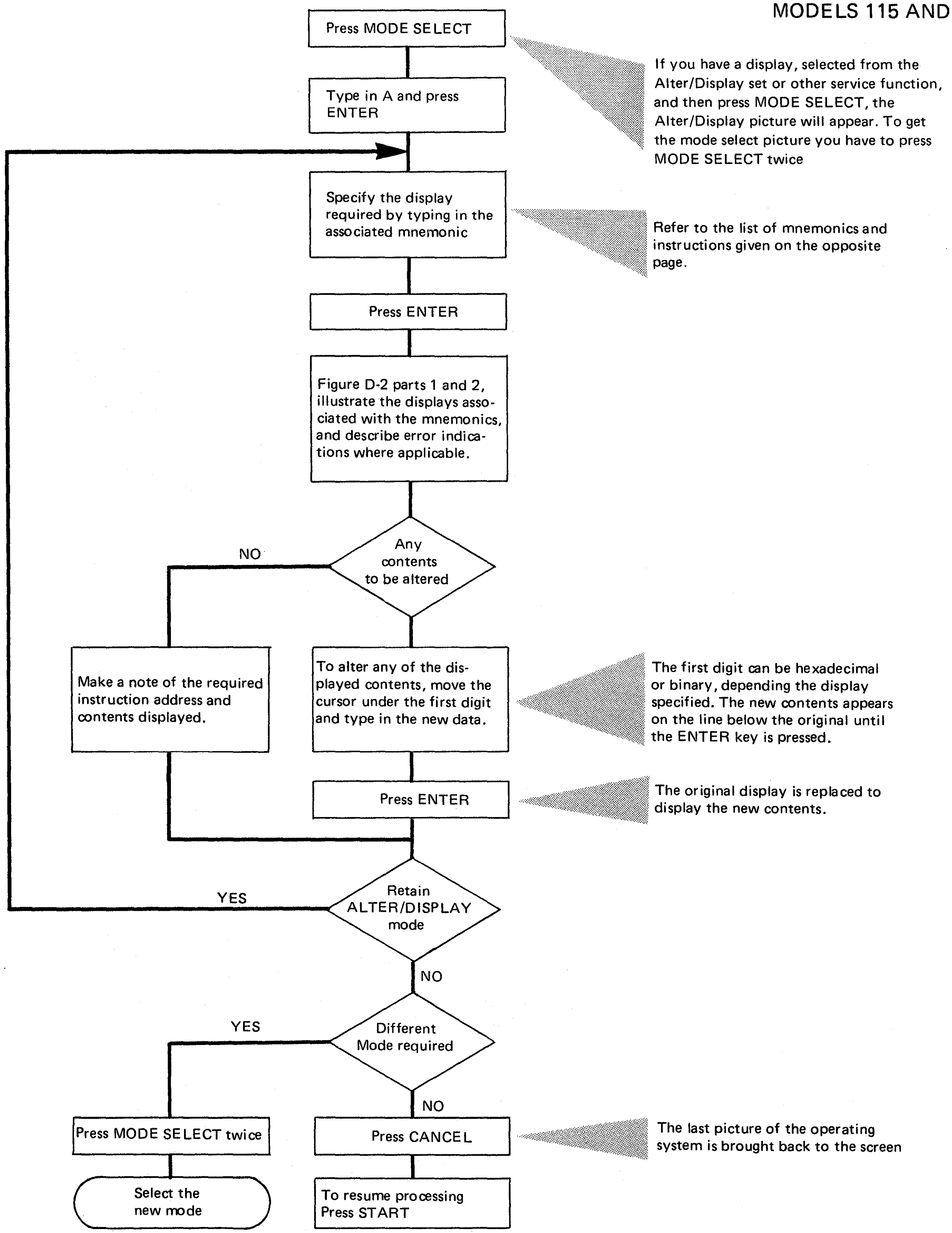
To select a particular display:
1. Type in the associated mnemonic according to the instruction given in the next flowchart.
2. Press the ENTER key.
   Before ENTER is pressed, you can still change your input by using the cursor keys and entering the changes in the usual way. As soon as ENTER is pressed, the new data replaces the old. The display remains on the screen and the cursor is at the next ALTER/DISPLAY line. Because there is an A (for ALTER/DISPLAY on this line, you need only enter F (for floating point registers) or P (for PSW), and so on.

ALTER/DISPLAY
MODELS 115 AND 125

G GENERAL REGISTERS	To display: 1. Type G into the alter/display picture. 2. Press ENTER. All general registers appear at once.
F FLOATING- POINT REGISTERS	To display: 1. Type F into the alter/display picture. 2. Press ENTER. All floating-point registers appear at once.
P CURRENT PSW	To display: 1. Type P into the alter/display picture. 2. Press ENTER. The current PSW is displayed in binary notation, except for the instruction address, which is in hex. BC or EC mode is indicated in the machine status area, line 14, and in the E-bit in the PSW. o BC Mode: The system is in basic control mode when the E-bit is zero. o EC Mode: The system is in extended control mode when the E-bit is 1.
C CONTROL REGISTERS	To display: 1. Type C into the alter/display picture. 2. Press ENTER. All control registers appear at once.
K PROTECTION KEY	To display: 1. Type K into the alter/display picture. 2. Type in the main storage address in hex. 3. Press ENTER. In the protection key display: o The address is in hex. o The key is in binary. o The reference (R), the change (C), and the protection (P) bits are in binary.
M MAIN STORAGE REAL	To display: 1. Type M into the alter/display picture. 2. Type in the main storage address in hex. 3. Press ENTER. The display shows 32 halfwords of main storage at once. The Y characters in the format illustration represent, in hex, the main storage address without its low-order digit. The missing low order digit of the address is shown above each leftmost byte of each halfword.
V MAIN STORAGE VIRTUAL	To display: 1. Type V into the alter/display picture. 2. Type in the address. 3. Press ENTER. The display shows 32 halfwords of virtual storage at once. The R char-

Examples following the flowchart opposite show the format of the various displays and describe error indications where applicable.

Table D-2 Options for the ALTER/DISPLAY console feature (Models 115 and 125)


Press MODE SELECT
If you have a display, selected from the Alter/Display set or other service function, and then press MODE SELECT, the Alter/Display picture will appear. To get the mode select picture you have to press MODE SELECT twice
Type in A and press ENTER
Specify the display required by typing in the associated mnemonic
Refer to the list of mnemonics and instructions given on the opposite page.
Press ENTER
Figure D-2 parts 1 and 2, illustrate the displays associated with the mnemonics, and describe error indications where applicable.
Any contents to be altered
NO
Make a note of the required instruction address and contents displayed.
To alter any of the displayed contents, move the cursor under the first digit and type in the new data.
The first digit can be hexadecimal or binary, depending the display specified. The new contents appears on the line below the original until the ENTER key is pressed.
Press ENTER
The original display is replaced to display the new contents.
Retain ALTER/DISPLAY mode
YES
NO
Different Mode required
YES
NO
Press MODE SELECT twice
Select the new mode
Press CANCEL
The last picture of the operating system is brought back to the screen
To resume processing Press START

ALTER/DISPLAY MODELS 115 AND 125

Error messages

If logical errors are made while altering the current PSW, one or any of the following error indications may be displayed:

1. EC-PSW ERROR
2. INVALID ADDRESS LOADED
3. ADDRESS NOT TRANSLATE-ABLE

Message 1 indicates PSW rejection, which is caused if bit 12 of the PSW is set to zero. Messages 2 and 3 indicate that the PSW has been loaded, but a program check will occur when an attempt is made to continue operation.

Message 3 occurs in case of invalid page or segment table address specification exception.

Format of displays and error indications

The following illustrations show the amount and format of information displayed with the associated mnemonic. When a wrong character (either a non-hex or a non-binary as the case may be) is entered, INVALID CHARACTER appears. The cursor marks the first invalid character.

G GENERAL REGISTERS

```
        * ALTER/DISPLAY GENERAL REGISTERS (HEXADECIMAL) *

  0 0000 0000    1 0000 0000    2 0000 0000    3 0000 0000

  4 0000 0000    5 0000 0000    6 0000 0000    7 0000 0000

  8 0000 0000    9 0000 0000    A 0000 0000    B 0000 0000

  C 0000 0000    D 0000 0000    E 0000 0000    F 0000 0000

NEXT ALTER/DISPLAY: A
```

F FLOATING POINT REGISTERS

```
        * ALTER/DISPLAY FLOATING POINT REGISTERS (HEX) *

                 0 0000 0000 0000 0000

                 2 0000 0000 0000 0000

                 4 0000 0000 0000 0000

                 6 0000 0000 0000 0000

   NEXT ALTER/DISPLAY: A
```

P Current PSW

```
              * ALTER/DISPLAY CURRENT PSW *

  SYST.MASK   KEY    EMWP   ILC   CC   PROGRMASK
  0000 0000   0000   0000   00    00   0000

  INSTRUCTION ADDRESS: 000000
  ADDRESS IN HEX, OTHER DATA IN BINARY
NEXT ALTER/DISPLAY: A
```

Figure D-2, part 1 of 2. Format of the displays for Models 115 and 125.

C CONTROL REGISTERS

```
* ALTER/DISPLAY CONTROL REGISTERS (HEXADECIMAL) *

0 0000 00E0   1 0000 0000   3 FFFF FFFF   3 FFFF FFFF
4 0000 0000   5 0000 0000   6 0000 0000   7 0000 0000
8 0000 0000   9 0000 0000   A 0000 0000   B 0000 0000
C 0000 0000   D 0000 0000   E C200 0000   F 0000 0200
NEXT ALTER/DISPLAY: A
```


K PROTECTION KEY

```
* ALTER/DISPLAY PROTECTION KEY *

            HEX           BIN          BIN
ADDRESS: 00002F      KEY: 0000     FRC: 010

NEXT ALTER/DISPLAY: A
```

Error messages

INVALID ADDRESS appears if the address is larger than the real storage size.

The address has to be typed in with leading zeros. When selecting the alter/display protection key display, do not use any commas or blanks.

M MAIN STORAGE REAL

```
* ALTER/DISPLAY MAIN STORAGE REAL (HEXADECIMAL) *

        0     2     4     6     8     A     C     E
00012  0000  0000  0000  0000  0000  0000  0000  0000
00013  0000  0000  0000  0000  0000  0000  0000  0000
00014  0000  0000  0000  0000  0000  0000  0000  0000
00015  0000  0000  0000  0000  0000  0000  0000  0000
NEXT  ALTER/DISPLAY:  A
```

V ALTER/DISPLAY MAIN STORAGE VIRTUAL

```
* ALTER/DISPLAY MAIN STORAGE VIRTUAL (HEX) *

REAL: 03E28  0     2     4     6     8     A     C     E
      61A94 D0E3  AB13  5478  4ABE  0000  0000  0000  0000
      61A95 0000  0000  0000  0000  0000  0000  0000  0000
      61A96 0000  0000  0000  0000  0000  0000  0000  0000
      61A97 0000  0000  0000  0000  0000  0000  0000  0000
NEXT ALTER/DISPLAY: A
```

Error messages

If the contents of the virtual address entered is not in real storage the virtual storage area will not be displayed. Instead one of the following messages will be displayed:

OUTSIDE PAGE TABLE
OUTSIDE SEGMENT TABLE
PAGE ENTRY INVALID
SEGMENT ENTRY INVALID
SPECIFICATION EXCEPTION
ADDRESSING EXCEPTION

Figure D-2, part 2 of 2. Format of the displays for Models 115 and 125

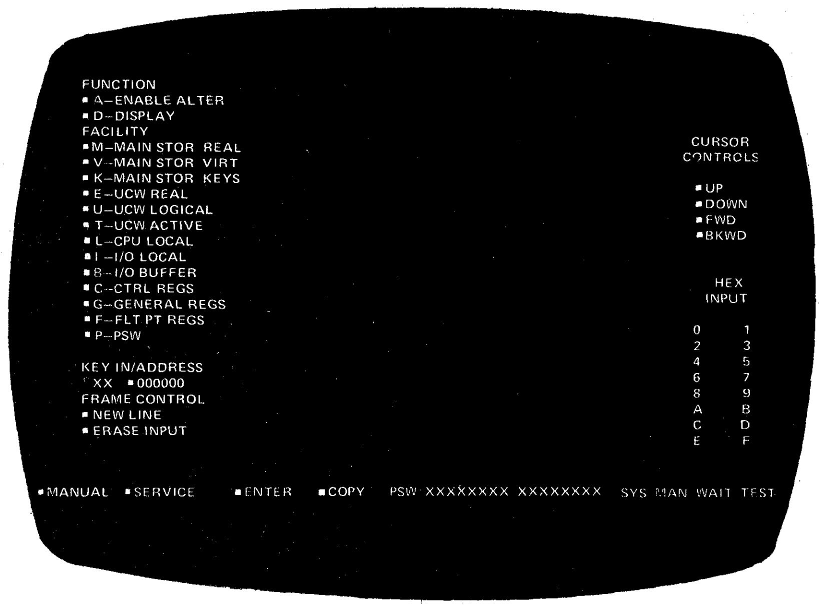
ALTER/DISPLAY MODEL 158

The ALTER/DISPLAY facility allows the operator to display or change the contents of the following parts of the CPU (Central Processing Unit), and of real or virtual storage areas:

- General registers
- Floating-point registers
- Current PSW
- Control registers
- Protection keys
- Real storage areas
- Virtual storage areas
- Real Channel UCWs
- Logical Channel UCWs
- Active UCWs
- CPU Local Storage
- I/O UCW Local Storage
- I/O Buffer Local Storage

How to use

The ALTER/DISPLAY frame, shown below, can be entered only from the MANUAL or SERVICE frame when the CPU is in the stopped (manual) state.



A procedure for using this facility is shown in the flowchart on the opposite page.

Error Indications

All address characters are checked as they are entered for hex values 0-F. Invalid characters are not displayed and the console alarm sounds.

If the cursor stays in the reset position (under the Y in Key In/Address) and if the console alarm sounds, an invalid function code has been entered. A valid function code must be entered before the cursor will reposition to the right (one position).

Printing displayed information

A "hard copy" of all information displayed can be obtained on a Model 158 with a 3213 printer attached by pressing the COPY key after the information is displayed.

Note however, that the following frames cannot be printed:

- PROGRAM
- ALTER/DISPLAY
- INDEX

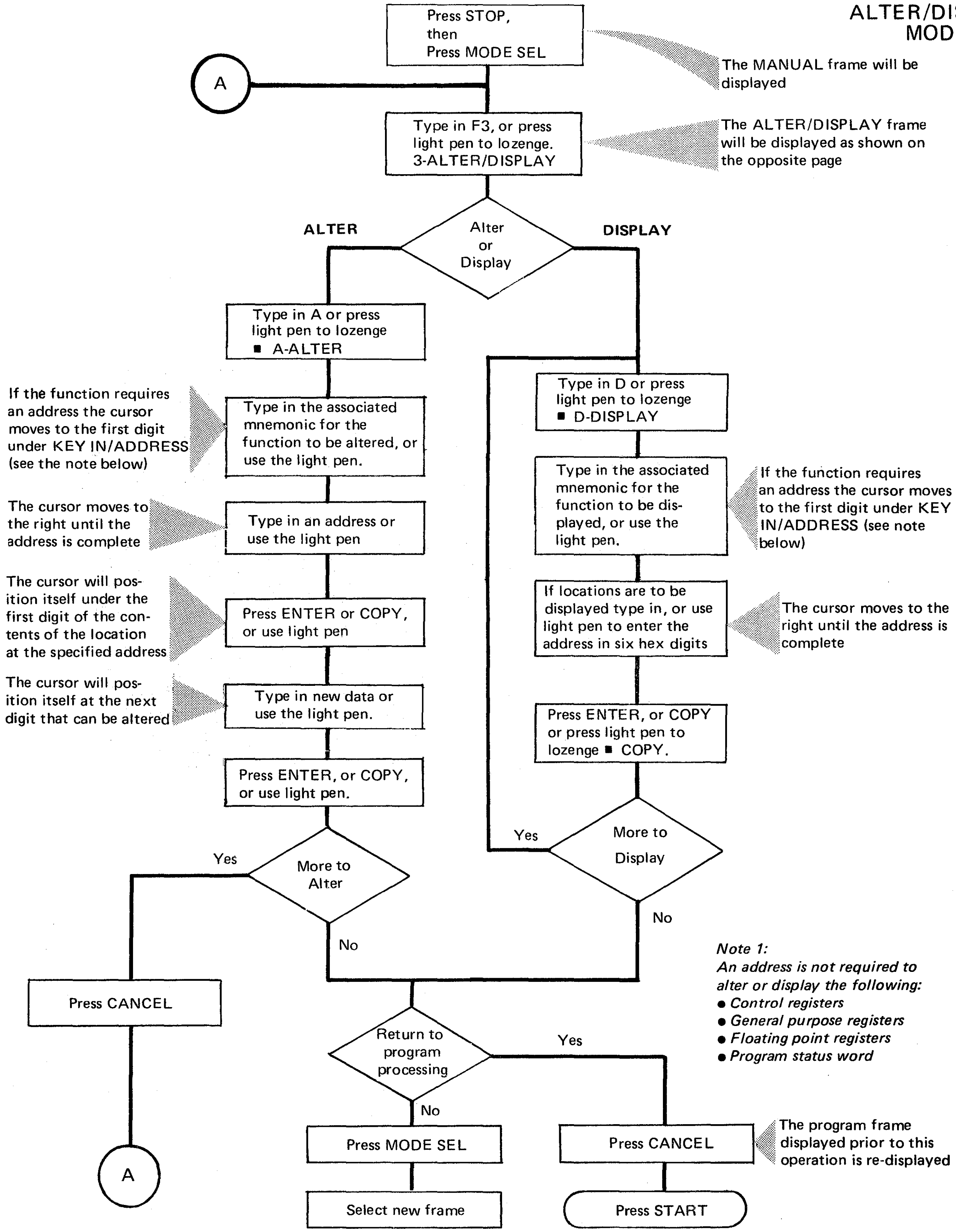
(and when using the ALTER facility, only lines that have been changed by entering new data will be printed on the 3213 printer.)

CAUTION

The effect on the operation of programs currently running in the system that are time dependent, for dxample a program using MICR or teleprocessing as input

ALTER/DISPLAY MODEL 158

D-3-F



When using COPY and ENTER:

Using COPY will only produce a "hard copy" of lines that have been changed by entering new data.
Using ENTER in display mode will not produce a "hard copy".

INSTRUCTION STEPPING (ALL MODELS)

This console facility allows the operator to check and obtain a *hard copy* of each instruction address executed during program operation.

Combining this facility with the console printer ALTER/DISPLAY feature described in D-1 of this section, provides a procedure to trace and record the path of a short loop.

Note: The different types of loops and their causes are described in Section 1.

When to use (all Models)

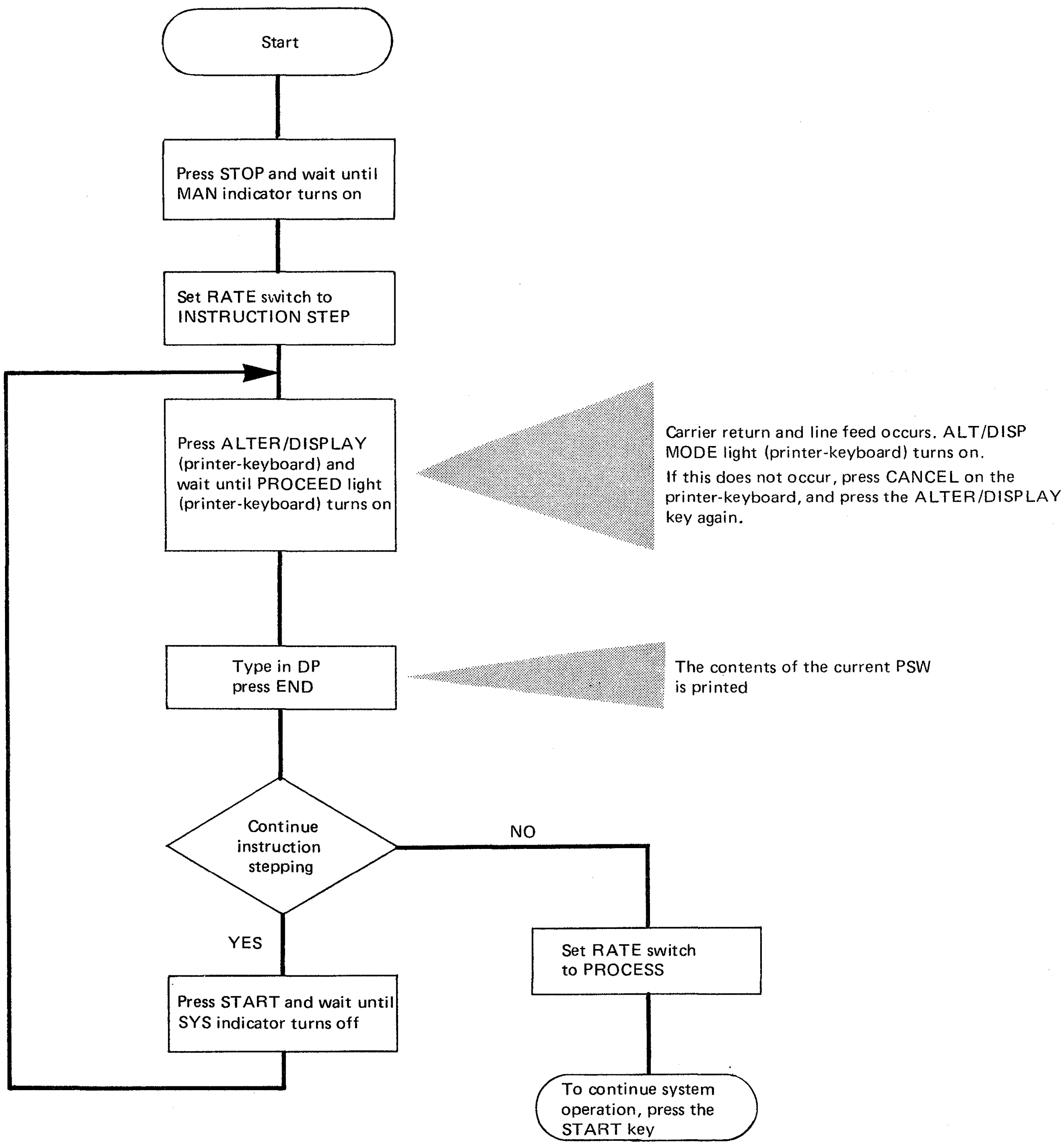
This facility should be used when the system malfunction prevents the use of SDAIDS to trace the loop. It is also useful during hands-on debugging when only small parts of a program require accurate program flow analysis.

Flowcharts in Section 3 indicate to the operator when a loop is to be traced using this console facility.

INSTRUCTION STEPPING MODELS 135, 145, AND 155-11

How to use

A procedure for tracing and recording the path of a loop using the instruction step facility of the Models 135, 145 and 155-II is shown in the flowchart opposite.



The procedure for tracing a loop using instruction step method

INSTRUCTION STEPPING MODELS 115 AND 125

How to use

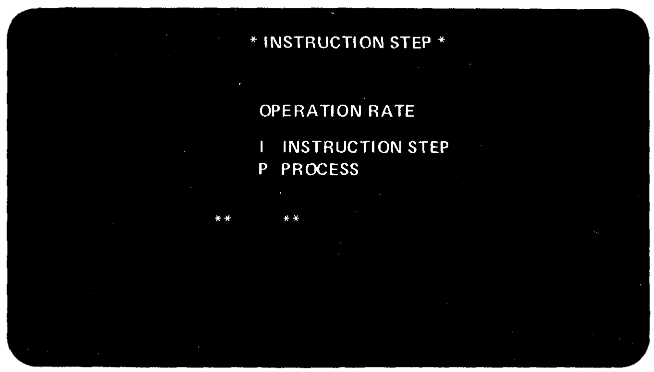
The INSTRUCTION STEP display allows the operator to check and make a note of each instruction address during program operation.

Making a note of the instruction addresses executed each time the START button is pressed provides a procedure to trace and record the path of a short loop.

INSTRUCTION STEP offers two modes: I and P

To select the instruction step display shown below:

1. Type I into the mode select display.
2. Press ENTER.



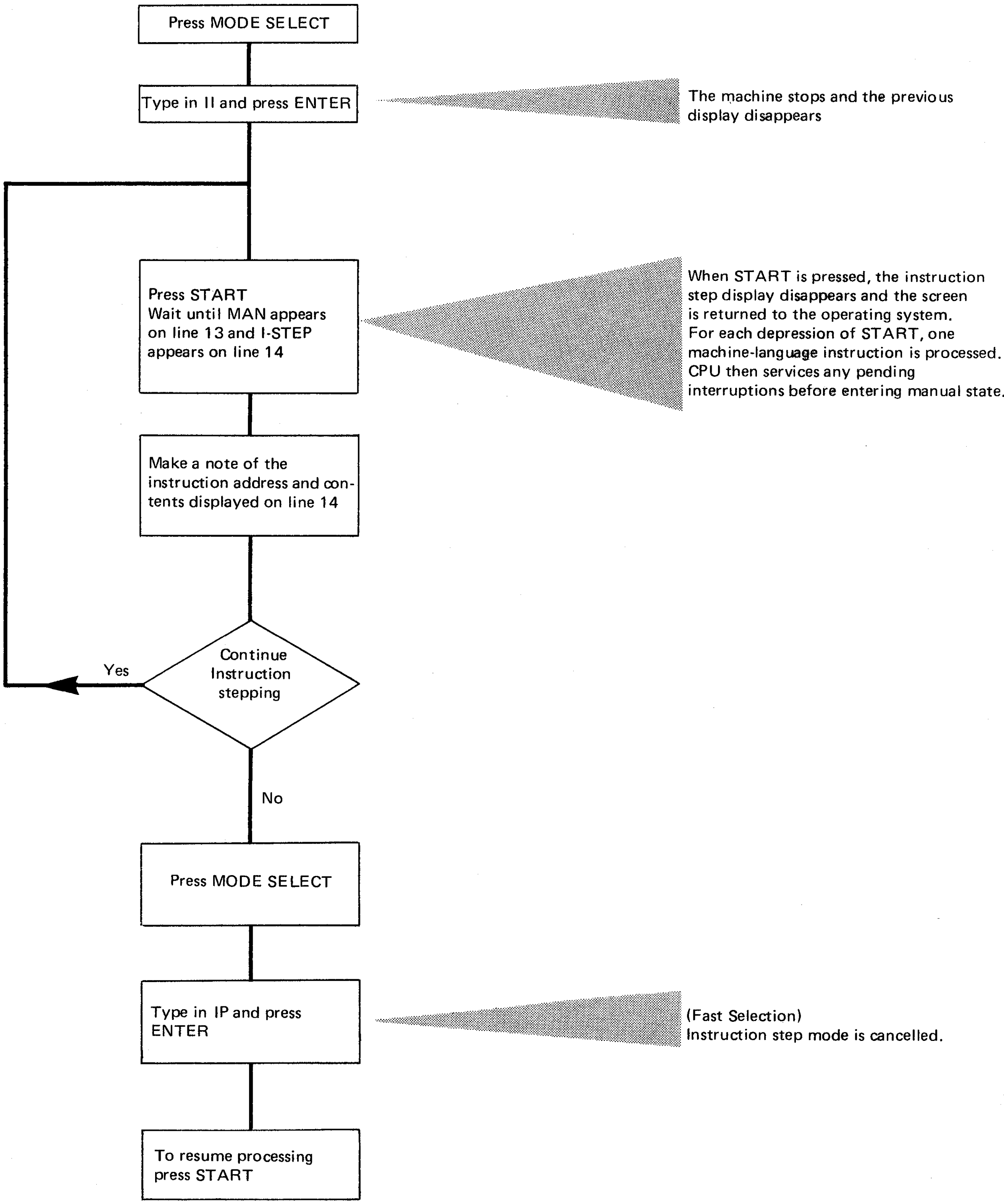
I INSTRUCTION STEP . . . . If I is typed into the instruction step display and ENTER is pressed, the new data can be seen as soon as the stop occurs. Line 14 (in the machine status area) shows the address and the data at this address.

P PROCESS . . . . As soon as START is pressed the screen is returned to the operating system and operating messages can be traced with each step. Instruction step mode is indicated by I-STEP on line 15 (in the machine status area). Entering P is used to end the instruction step mode.

1. Type in P.
2. Press ENTER

The last picture of the operating system is brought back to the screen.
Press START to continue processing.

D-5-F



The procedure for tracing a loop using the instruction step method

INSTRUCTION STEPPING MODEL 158

The INSTRUCTION STEP display allows the operator to check and make a note of, or obtain a "hard copy" of each instruction address during program operation.

Making a note of the instruction addresses executed each time the START button is pressed provides a procedure to trace and record the path of a short loop.

How to use

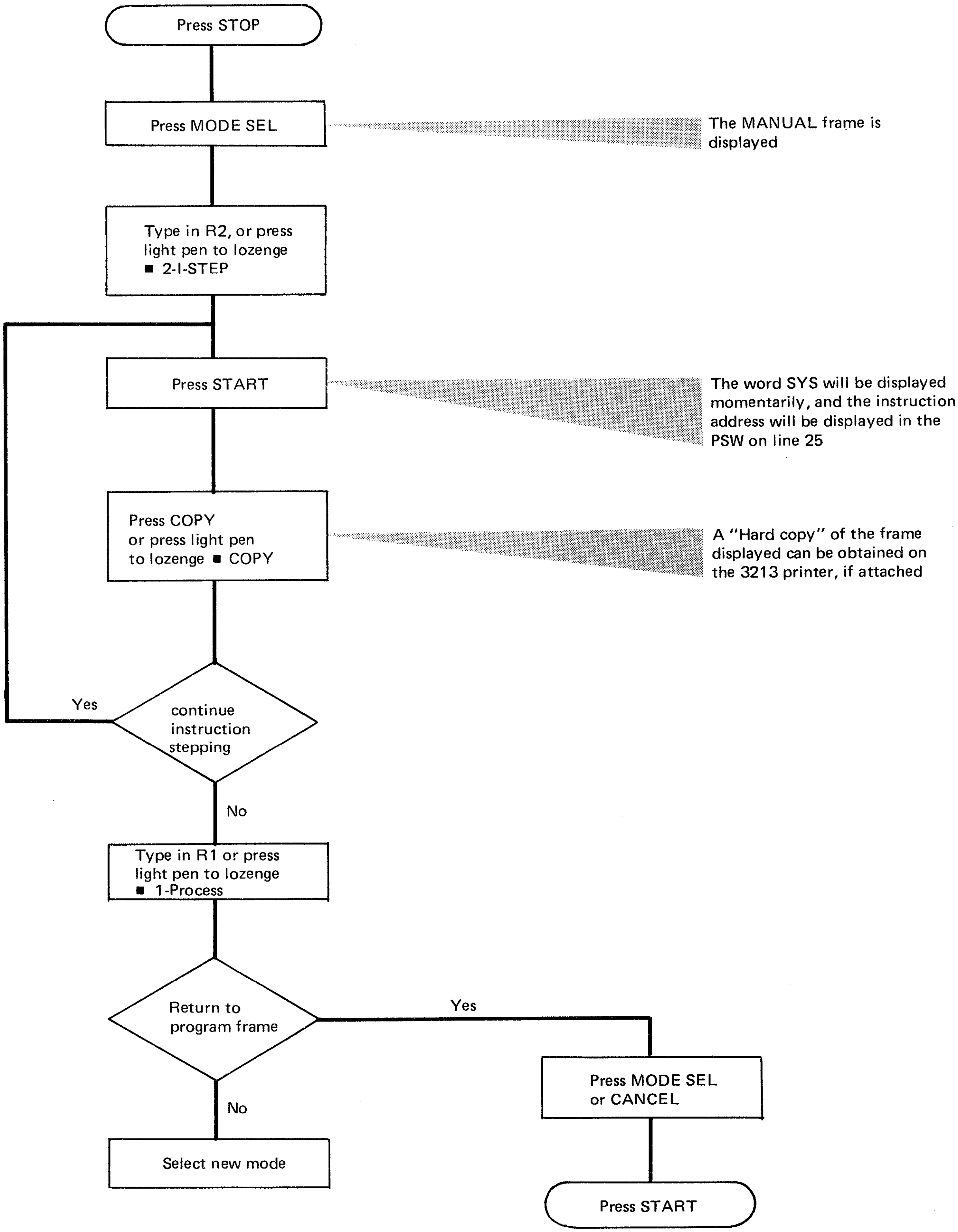
With the manual fram displayed, shown below, after pressing MODE SEL, the R-RATE switch must be set to I-STEP by either typing in R2 or by pressing the light pen to lozenge ■ 2-I-STEP. The selection is indicated by an arrow displayed as shown in the example below.

The instruction address of the current instruction will be in the address part of the PSW displayed on line 25.

Pressing the START key will cause the CPU to execute the next instruction in logical sequence, the address of which will be displayed in the PSW as before.

In display mode of operation a "hard copy" of the PSW displayed can be obtained on the 3213 printer, if attached, by pressing the COPY key or by pressing the light pen to lozenge COPY.

To return to normal CPU processing rate, type in R1 or hold the light pen against ■ 1-PROCESS and press MODE SEL followed by START.



The procedure for tracing a loop using the instruction step method on the Model 158

STOP ON ADDRESS COMPARE (ALL MODELS)

This facility enables you to stop all system activity at any selected storage address during system operation. Two methods are provided on the System/370 that enable both hardware and software-controlled stops:

1. By using switches on the system control panel
2. By using the SDAID stop on event facility.

For the Models 145 and 155-11 the system control panel switches enable a stop on real or virtual address.

The Model 115, 125 and 135 have system control panel switches that do not allow for a stop on a virtual address.

Stop on event for all models is described under SDAIDS, Section 2-B-8. A flow-chart in Section 2-B-10 shows how to initiate and execute this aid.

When to use (all Models)

This facility is a hands-on debugging aid for the programmer, and permits him to stop system operations at selected addresses in the program listings. He can use it, for example, in conjunction with either the ALTER, DSPLY and DUMP commands, or the console ALTER/DISPLAY feature, to change the contents or display particular areas of storage at selected addresses in a program. The operator is also able to use this facility if, for example, the programmer requests a dump of certain storage locations at particular points in a program during execution of the program.

STOP ON ADDRESS COMPARE MODELS 135, 145, AND 115-11

How to use

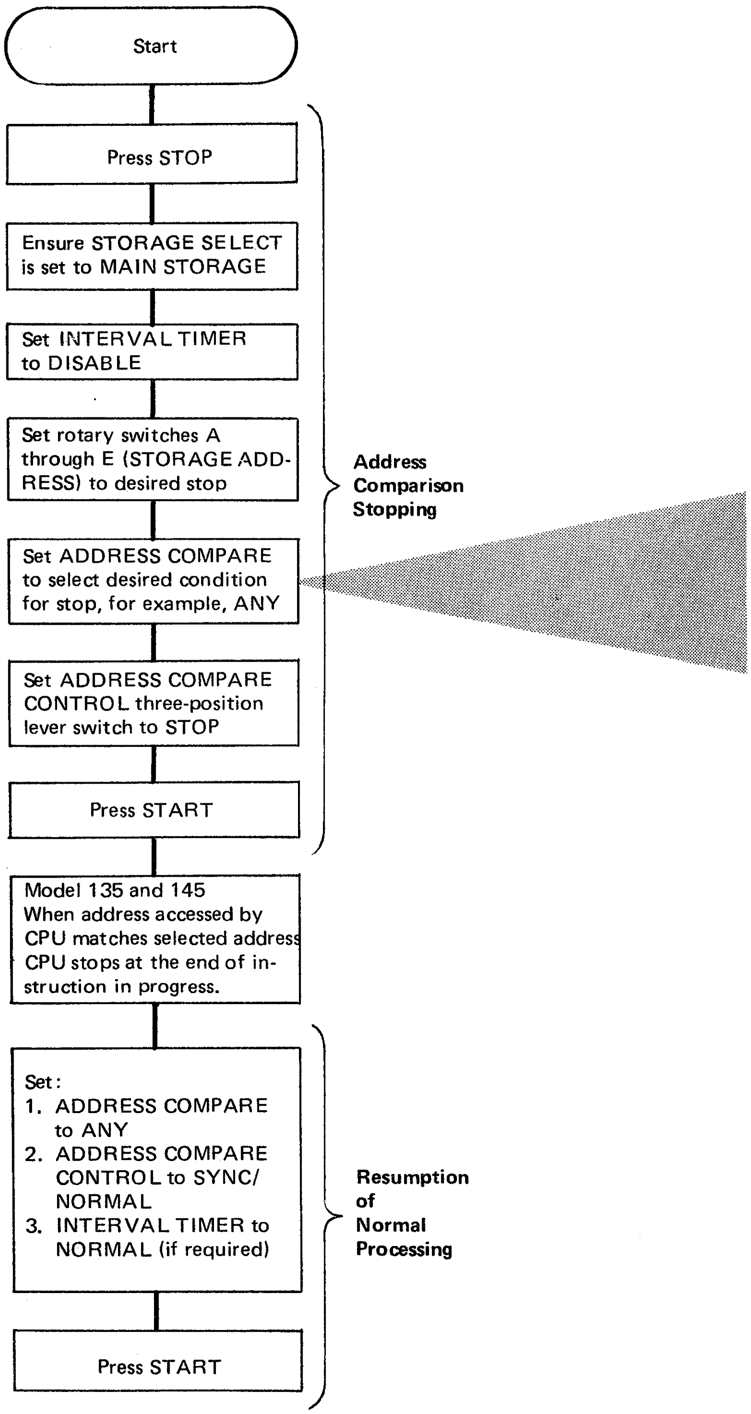
Four switches on the system control panel are used during address compare operations:

- ADDRESS COMPARE CONTROL (Toggle)
- ADDRESS COMPARE (Rotary)
- STORAGE SELECT (Rotary)
- INTERVAL TIMER (Toggle).

The flowchart opposite shows the procedure for stop on address compare applicable to System/370 Models 135, 145 and 155-11. However, because the ADDRESS COMPARE rotary switch differs between models, the IBM operating procedures for the model on which the operation is to be executed must be consulted.

The flowchart on the following page shows how to invoke the displays required to execute the "Stop on Address Compare" on the Models 115 and 125.

STOP ON ADDRESS COMPARE MODELS 135, 145, AND 155-11



Switch positions for selecting the conditions for the stop are different for each of the System/370 Models. Consult *Compare Type Selection* on the next two pages and the note below.

D-7-F

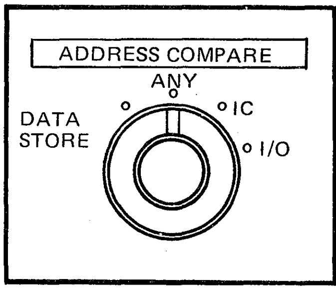
Note:
See ADDRESS COMPARE and ADDRESS COMPARE CONTROL CONTROL switch setting for each System/370 model as described in the IBM-supplied Operating Procedures manual belonging to the system.

A procedure for using the stop on address compare facility.

Aids provided by the Operator's Console

STOP ON ADDRESS COMPARE MODELS 135, 145, AND 155-II

Compare type selection (Model 135)



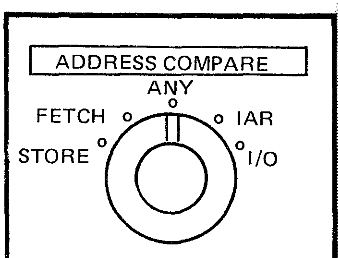
DATA STORE:
A match occurs when the selected location is addressed to store data.

ANY:
The normal operating position – a match occurs when the selected location is addressed for any type of operation

IC:
A match occurs when the selected location is addressed by an instruction

I/O:
A match occurs when the selected location is addressed for an I/O data transfer

Compare type selection (Model 155-11)



ANY (Real Address)

This position of the switch is used for normal program processing. With the switch in this position, a match occurs for main storage access when the storage address matches the address set in console switches CDEFGH.

FETCH

This position causes a match when the storage address matches the address set in console switches CDEFGH, and the operation is a data fetch from main storage.

STORE

This position allows a match when the storage address matches the address set in console switches CDEFGH during a data store operation.

IAR

This position of the switch allows a match when the IAR (instruction address register) address matches the address set in console switches CDEFGH.

I/O (Input/Output)

This position of the switch allows a match when the storage address matches the address set in console switches CDEFGH, and the operation is a data store or fetch for an I/O operation.

STOP ON ADDRESS COMPARE MODELS 135, 145, AND 155-II

Compare type selection (Model 145)

ANY (Logical Address)

This position of the switch allows a match when the logical main storage address used to access storage matches the address set in console switches CDEFGH.

ANY (Real Address)

This position of the switch is used for normal program processing. With the switch in this position, a match occurs for main storage access when the storage address matches the address set in console switches CDEFGH.

DATA STORE

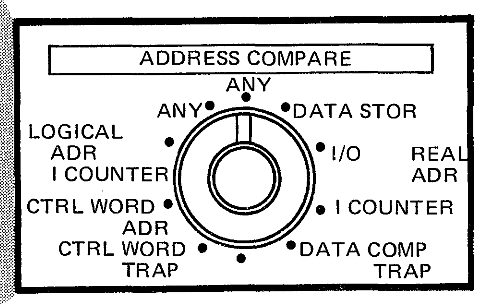
This position allows a match when the sotrage address matches the address set in console switches CDEFGH during a data store operation.

I/O (INPUT/OUTPUT)

This position of the switch allows a match when the storage address matches the address set in conssole switches CDEFGH, and the operation is storing or fetching data for an I/O operation.

I-COUNTER (Real or Logical Address)

This position causes a match when the real or logical main storage address matches the address in console switches CDEFGH, and the operation is an instruction fetch from main storage.



NOTE: Significant throughput degradation can occur while processing with this switch set to the I-COUNTER REAL ADR position.

Data compare trap (Model 145)

This facility is useful during hands-on debugging to determine what instruction is causing a particular storage byte location to be modified.

1. Press STOP.
2. Set the ADDRESS COMPARE switch to DATA COMPARE TRAP.
3. Set the address of the storage byte location being modified in console switches CDEFGH.
4. Set data switches A and B to the desired byte match value.
5. Set the ADDRESS COMPARE CONTROL toggle switch to STOP.
6. Press START.

When a store operation modifies the specified storage byte location to the value set in switches A and B, the ADR COMP MATCH indicator is turned on and the CPU enters a soft-stop state.

To determine the address of the instruction that modified the storage byte, display the current PSW, and subtract the current instruction length code from the instruction address in the current PSW.

Note:
The instruction found with this procedure may not have modified the data. An I/O data trap occurring during execution of this instruction could have modified the data. To determine which I/O data trap modified the data, log the address displayed in the A-Register Display roller switch indicators and call your service representative.

D-7

Aids provided by the Operator's Console

STOP ON ADDRESS COMPARE MODELS 115 AND 125

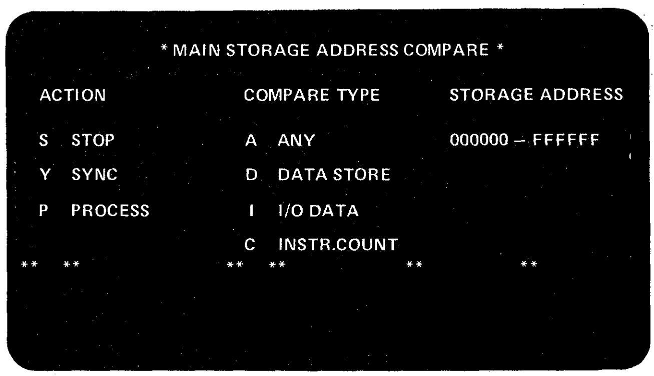
How to use

Before this facility can be used the mode select display must be brought to the screen by pressing the MODE SELECT key.

To select the storage ADDRESS COMPARE display shown below:
1. Type in 'C' beside MODE SPECIFICATION on the mode select display.
2. Press ENTER.

Error Indications
If you make an error when typing in the code or hex characters:
- The address compare display stays on the screen.
- INVALID CHARACTER appears.
- The cursor marks the location of the first error.

The display remains on the screen as long as an entry error has not been corrected. When there is no error, the display disappears after ENTER is pressed.



Three columns are displayed and an entry must be made under each column.

ACTION

S STOP: the machine stops when the address has been found.

Y SYNC: a signal for the customer engineer is given when the address has been found.

COMPARE TYPE

P PROCESS: address compare mode is turned off and normal processing continues.

A ANY: the CPU will compare your search address (the address you type into column three) against all addresses used in the system.

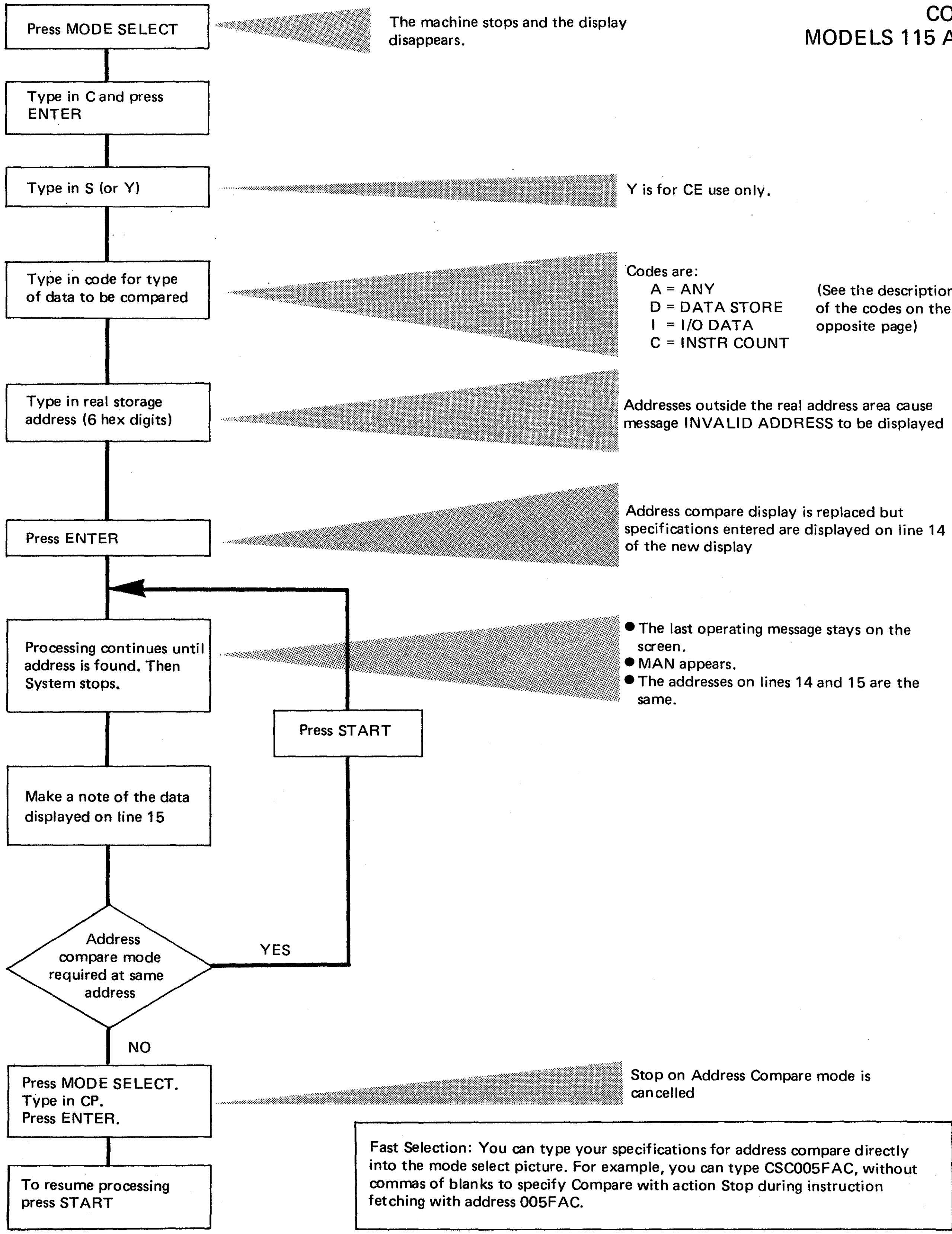
D DATA STORE: the CPU will compare your search address against only those storage addresses used to store data.
Your search address will not be compared against addresses used in transferring data to or from I/O devices.

I I/O DATA: the CPU will compare your search address against only those storage addresses used in transferring data to or from I/O devices.

STORAGE ADDRESS

C INSTR COUNT: the CPU will compare your search address against only those addresses used when fetching instructions.
The real storage address at which the stop is to occur.

STOP ON ADDRESS COMPARE MODELS 115 AND 125



Fast Selection: You can type your specifications for address compare directly into the mode select picture. For example, you can type CSC005FAC, without commas of blanks to specify Compare with action Stop during instruction fetching with address 005FAC.

D-8-F

STOP ON ADDRESS COMPARE 'MODEL 158

How to use

Before this facility can be used the MANUAL frame must be brought to the screen by pressing the MODE SEL key.

STOP ON A REAL ADDRESS

To stop on a real address S-SAR COMP SEL(REAL) must be used in conjunction with E-SAR COMP SEL(REAL).

1. Select the STOP function ■ 5-STOP or type in S5
2. Select the compare type (see below)
3. Enter a real address at E-SAR COMP SEL(REAL).

A stop will occur on any quadword boundary of a selected real address

COMPARE TYPE

Note: If STOP is not selected a 'sync' pulse will be generated on true comparisons and the CPU will not stop.

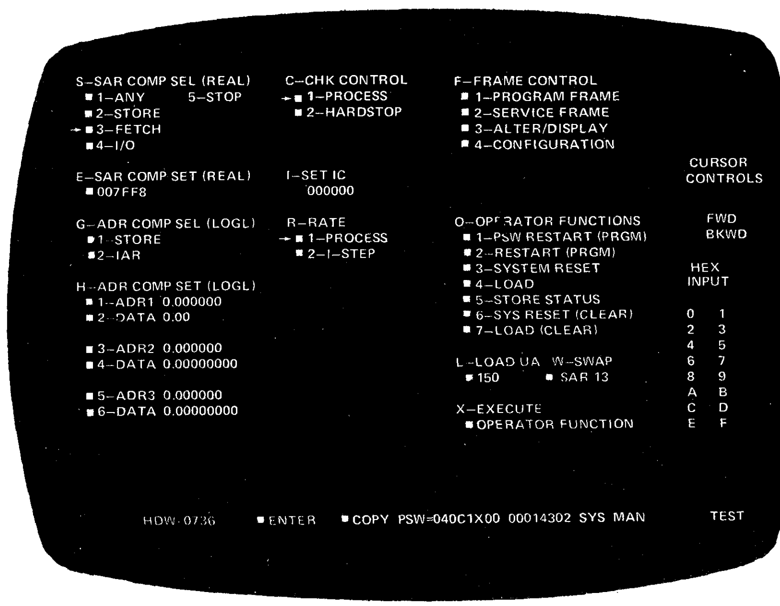
1-ANY: the CPU will compare the address set under E with all addresses used in the real addre ss area. When a true comparison is met the CPU will stop.

2-STORE: causes the CPU to stop when a STORE operation is performed on the location at the address entered under E.

3-FETCH: causes the CPU to stop when a FETCH operation is performed on the location at the address entered under E.

4-I/O: causes the CPU to stop when data is transferred either to or from the location at the address entered under E.

The example below shows the appearance of the manual frame after selecting a stop on a FETCH operation on the location at real address 007FF8.

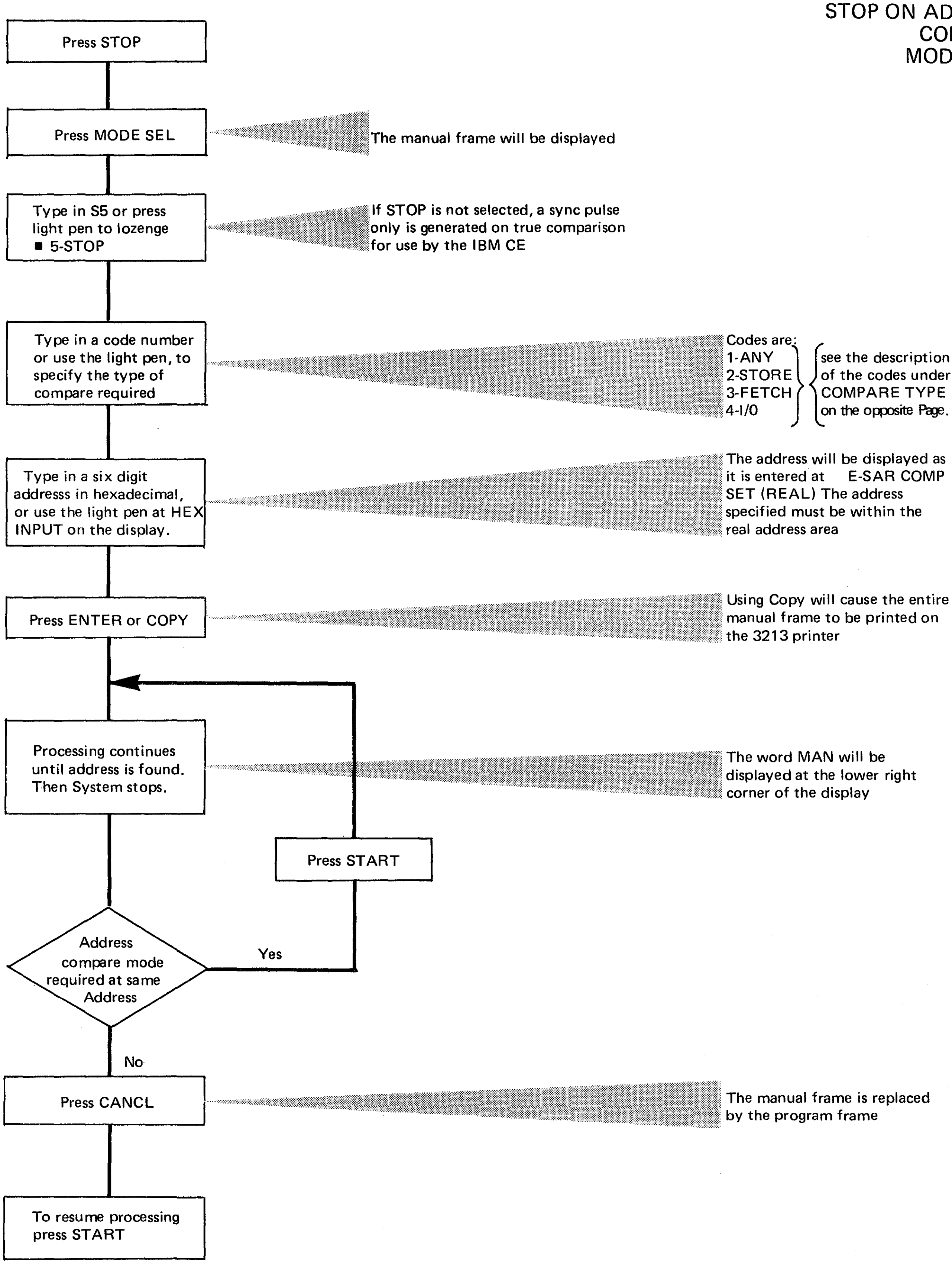


STOP ON A VIRTUAL ADDRESS AND DATA COMPARE TRAP

A facility on the System/370 Model 158 allows a stop on a virtual address with a maximum of 6 conditions described in the table below.

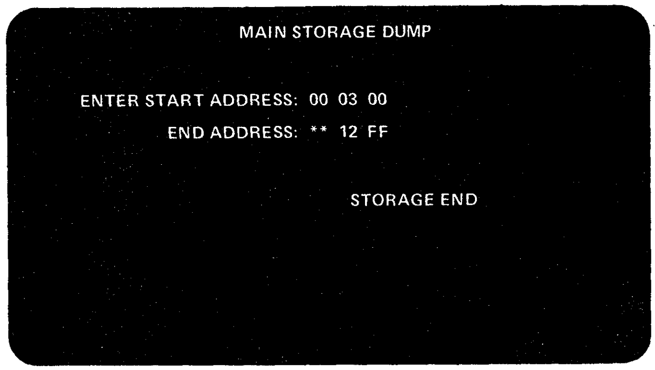
FUNCTION	CPU ACTION	USAGE
G1–STORE	Stops processing after a store to the virtual address specified in ADR1.	The cursor can be backed up under the digit to the left of the period and the base register entered in that position. The six digits to the right of the period are used to contain the displacement. The DATA fields and the ADR2 and ADR3 fields can be used to further define the stop conditions, giving a maximum of six conditions that must be satisfied before a stop occurs. The C to the left of the period in the data field signifies that the specified data will be compared. When the C is changed to a 1, the specified bits are compared. *Example: If the character to the left of the period is changed to a 1, and 1A is entered in the DATA field, a stop occurs whenever bits 3, 4, and 6 are on. Conversely, if the C is changed to a 0, a stop occurs whenever bits 3, 4, and 6 are off.*
G2–IAR	Stops processing after an instruction fetch to the virtual address specified in ADR1.	
H1–ADR1	Specifies the first condition that must be met in order for a stop to occur.	
H2–DATA	Specifies the data that must be found at the address in ADR1 for a stop to occur. **If DATA is not pressed, the stop occurs whenever the location specified by ADR1 is addressed.**	
H3–ADR2 H4–DATA	Specifies additional conditions that must be met if a stop is to occur.	
H5–ADR3 H6–DATA	Specifies additional conditions that must be met if a stop is to occur.	

STOP ON ADDRESS COMPARE MODEL 158



CONSOLE DUMP OPERATION MODELS 115 AND 125 ONLY

This operation provides a non-destructive readout and printout of any real storage area (up to 64K bytes at a time). The command can be executed at any time, and the (dumped) program can continue as soon as dumping is completed (no IPL or restart required).
To select the storage dump display:
1. Type D into the mode select display.
2. Press ENTER.
The display shown below appears on the screen.



Error Indications:
PRINTER NOT READY appears if the line printer is not ready.
STORAGE END appears if the start address is greater than the physical size.
INVALID CHARACTER appears for any non-hex digits.
INCOMPLETE ENTRY appears when necessary.

How to use

1. Type in the start and end addresses. Remember that:
- The low order halfword must be two zeros.
- The end address must be within 64K bytes from the start address, and the low-order halfword must be FF.
- If the dump required is more than 64K bytes, repeat the operation with a new start and end address.

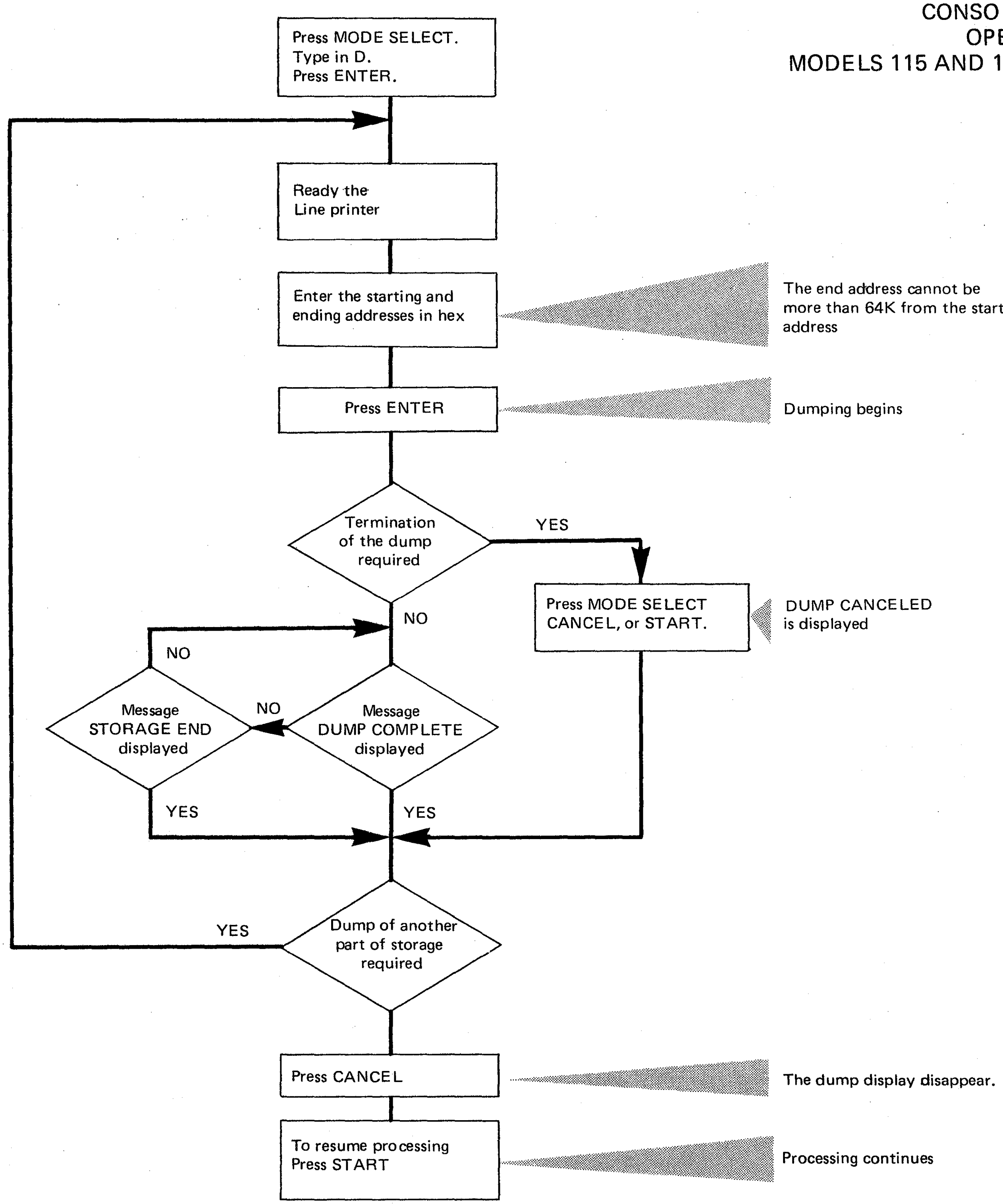
2. Press ENTER.
- Dumping can be stopped at any time by pressing MODE SELECT, CANCEL, or START. If one of these keys is pressed, DUMP CANCELLED appears on the screen.
- DUMP COMPLETE appears on the scrren when the selected dump range has been printed.
- STORAGE END appears on the screen when the upper boundary of storage has been reached.

The flowchart shown on the opposite page shows the procedure for using this command.

When to use

This is a useful aid to use when large areas of real storage must be recorded for later analysis. It can also be used instead of the ALTER/DISPLAY feature to record low address storage before executing the stand-alone dump program.

CONSOLE DUMP OPERATION MODELS 115 AND 125 ONLY



NOTE: If a log or retry operation takes place at the same time as a dump request, PRESS CANCEL appears on the screen. After pressing CANCEL, the message LOG IN PROGRESS appears in the machine status area. You can repeat dumping as soon as the log message disappears.

D-10-F

STORE STATUS (ALL MODELS)

This function enables certain control information to be stored and preserved for analysis by the IBM CE.

Models 135, 145 and 155-11

1. Press the console printer keyboard ALTER/DISPLAY key.
2. Type in ST.

The information saved is identical to that listed below for the Model 125

Models 115 and 125

There is no display for STORE STATUS.
To store the status:
1. Type S into the mode select display.
2. Press ENTER.

The following information is stored.
- CPU Timer
- Clock Comparator
- The current PSW
- Floating-Point Registers
- Control Registers
- General Registers.

After ENTER has been pressed:
- The mode select display remains on the screen and STATUS STORED appears.
- The system goes into the stopped state.
- The S has disappeared from the mode specification field, so this field is free and another operation can be specified.

Note: This function must not be used on a Model 115 or 125 that does not support MCH (Machine Check Handling).

```
O–OPERATOR FUNCTIONS
   1–PSW RESTART(PRGM)
   2–RESTART(PRGM)
   3–SYSTEM RESET
   4–LOAD
   5–STORE STATUS
   6–SYS REST(CLEAR)
   7–LOAD(CLEAR)

I–LOAD UA        W–SWAP
   000              SAR 13

X–EXECUTE
   OPERATOR FUNCTION
```

Model 158

1. Press MODE SEL to obtain the manual frame.
2. Type in 03 or hold light pen against ▪ 3-SYSTEM RESET.
3. Type in X or press light pen to lozenge ▪ OPERATOR FUNCTION.
4. Type in 05 or press light pen to lozenge ▪ 5-STORE STATUS.
5. Type in X or use light pen at ▪ OPERATOR FUNCTION.

A new function may now be selected.

When to use

This function should be used before executing the stand-alone dump program.

CLEAR REAL STORAGE (ALL MODELS)

Models 135, 145 and 155-11

Real storage can be cleared to zeros by the following procedure:
1. Press and hold in ENABLE SYSTEM CLEAR.
2. Press SYSTEM RESET or LOAD.
3. Re-IPL to continue processing new job.

Note: Control storage is unaffected.

Models 115 and 125

1. Press the MODE SELECT key.
2. Type RC into the mode select display.

Note: At IPL time one of the LOAD parameters is NORMAL or CLEAR

Model 158

Two methods are available on the Model 158 to clear real, or main storage.

1. *System Reset (Clear):* In addition to performing the reset function, this causes main storage and the storage-protect arrays to be validated (cleared to zeros with good parity).

- Press MODE SEL to obtain the manual frame.
- Type in 06 or hold light pen against ▪ 6-SYS RESET (CLEAR).
- Type in X or hold light pen against ▪ OPERATOR FUNCTION.

2. *Load (Clear):* In addition to performing the load function, this causes the IPL function to be preceded by an initial program reset, and clears main storage and the storage-protect arrays.

- Press MODE SEL to obtain the manual frame
- Enter a load unit address
- Type in 07 or hold the light pen against ▪ 7-LOAD (CLEAR).
- Type in X or use light pen at ▪ OPERATOR FUNCTION.

When to use

This facility should be used with caution. An example of its use is to reset the hardware after a machine check is caused by a parity error in real storage. It must be used after you have gathered all the information from the system.

SAVE USAGE COUNTERS

There is no display for SAVE USAGE COUNTERS.
To select the save usage counters operation:
1. Type U into the mode select display.
2. Press ENTER.

When to use

This operation saves the usage counters of all disk drives. The operation should always be performed before you turn the power off so that the information can be used by the CE for maintenance. The message 'counter saved' appears for each counter that is recorded.

CONTENTS

Other Aids

JOB CONTROL COMMANDS AND STATEMENTS

The following commands and statements are not primarily designed as serviceability aids, but enable useful information to be obtained from the system during program execution.

For example, it is useful to place the LISTIO statement and command in job streams where device assignments are suspected of causing errors. The LOG command enables you to record job control statements and commands issued during a job, and the MAP command enables you to check partition allocation. These three commands, LISTIO, LOG, and MAP can be used therefore as a "job stream trace," as shown in the example opposite.

In certain cases of system malfunctions, this information, used in conjunction with dumps and trace routine output, will help during offline debugging.

LISTIO or // LISTIO

The LISTIO command or statement (List I/O Assignment) causes the system to print a listing of I/O assignments. The listing appears on SYSLOG (command) or SYSLST (statement). If SYSLST is not assigned, the LISTIO statement is ignored.

Statement Format

```
              ┌  SYS     ┐
              │  PROG    │
              │  BG      │
              │  F4      │
              │  F3      │
              │  F2      │
// LISTIO  ───┤  F1      ├─── (output on SYSLST)
              │  ALL     │
              │  SYSxxx  │
              │  UNITS   │
              │  DOWN    │
              │  UA      │
              └  X'cuu'  ┘
```

Command Format

```
              ┌  SYS     ┐
              │  PROG    │
              │  BG      │
              │  F4      │
              │  F3      │
              │  F2      │
LISTIO     ───┤  F1      ├─── (output on SYSLOG)
              │  ALL     │
              │  SYSxxx  │
              │  UNITS   │
              │  DOWN    │
              │  UA      │
              └  X'cuu'  ┘
```

The table following the example opposite explains the meaning of the LISTIO options.

```
BG   // JOB KENSLOOP ✓ LOOP SIMULATOR FOR SDAID EXAMPLES
    DATE 10/04/13,CLOCK 22/08/41
BG   1I00A  READY FOR COMMUNICATIONS.
BG
BG   // OPTION LINK,DUMP
BG      INCLUDE
BG   // EXEC LNKEDT
BG   4444A OVERLAP ON UNEXPRD FILE       IJSYS01  SYS001=131  111111
    BUG.GENERATOR,WORKFILE.K.TOMS
BG   delete
BG   ALLOCR F1R=82K
BG   MAP
BG   AREA     K-REAL  UPPER LIMIT  K-VIRT UPPER LIMIT     NAME
BG
BG    SP           92K      94207             262144
BG    BG   V5A    30K     124927      292K     561151     KENSLOOP
BG    F4   I4     10K     135167       80K     643071
BG    F3   I3     10K     145407       80K     724991
BG    F2   I2     14K ✓   159743      100K     827391
BG    F1   I1     82K     243711      120K     950271
BG    SVA                              96K    1048575
BG    VIS                              68K    1048575
BG    PP          18K     262143
BG   * MOUNT SCRATCH TAPE ON 381
BG   // PAUSE
BG
BG   // ASSGN SYS003,X'381'
BG   LISTIO SYS003
BG
BG                   *** BACKGROUND ***
BG
BG   I/O UNIT    CMNT    CHNL    UNIT    MODE
BG
BG    SYS003               3       81      90
BG    SYS003      STD      1       31
BG   LISTIO SYSLST
BG
BG                   *** BACKGROUND ***
BG
BG   I/O UNIT    CMNT    CHNL    UNIT    MODE
BG
BG    SYSLST               3       80      90
BG   LISTIO DOWN
BG
BG             *** DOWN ***
BG
BG             CHNL    UNIT
BG
BG               1      97
BG               2      50
BG               2      51
BG               2      52
BG               2      53
BG   LISTIO SYS
BG
BG                   *** BACKGROUND ***
BG
BG   I/O UNIT    CMNT    CHNL    UNIT    MODE
BG
BG    SYSRDR               0       0C
BG    SYSIPT               0       0C
BG    SYSPCH               0       0D
BG    SYSLST               3       80      90
BG    SYSLOG               0       09
BG    SYSLNK               1       31
BG    SYSRES               1       30
BG    SYSSLB              ** UA **
BG    SYSRLB              ** UA **
BG    SYSREC               1       31
BG    SYSCLB              ** UA **
BG    SYSVIS               1       31
BG    SYSCAT              ** UA ** ✓
BG   * THIS JOB MUST HAVE 30K REAL, AND IT SHOULD BE RUN IN BG
BG   * IT ALSO REQUIRES A SCRATCH TAPE ON 381
BG   * CHECK THAT SYSLST IS ASSIGNED TO 380
BG   * CHECK ALLOCATIONS AND ASSIGNMENTS
BG   // PAUSE
BG
```

Name of job in BG. ✓

BG is priority 5 Job is running in virtual mode, and is active.

OK.

Ex56KT

An example of a "job stream trace" using the LOG, MAP, LISTIO and PAUSE commands and statements.

Options for LISTIO

Options	Meaning
SYS	Lists the physical units assigned to all system logical units. (See note.)
PROG	Lists the physical units assigned to all background programmer logical units. (See note.)
BG	Lists the physical units assigned to all background logical units.
F4	Lists the physical units assigned to all foreground-one logical units.
F3	Lists the physical units assigned to all foreground-two logical units.
F2	Lists the physical units assigned to all foreground-three logical units.
F1	Lists the physical units assigned to all foreground-four logical units.
ALL	Lists the physical units assigned to all logical units.
SYSxxx	Lists the physical units assigned to the logical unit specified. The assignment is given for the partition from which the command is given. (See note.)
UNITS	Lists the logical units assigned to all physical units. (See note.)
DOWN	Lists all physical units specified as inoperative. (See note.)
UA	Lists all physical units not currently assigned to a logical unit.
X'cuu'	Lists the logical units assigned to the physical unit specified.

Note: Physical units are listed with current device specification for magnetic tape units. Logical units are listed with ownership (background, or any foreground), when applicable. If a unit has a standard assignment in one mode and a temporary assignment in another mode, the CMNT column identifies the type of assignment for each indicated mode. All channel unit numbers are represented in hexadecimal.

LOG

The LOG job control command causes the system to log, on SYSLOG, columns 1-72 of all Job Control commands and statements occurring in the batched-job partition in which the LOG is issued. The AR LOG affects all the partitions. The LOG function is effective until a NOLOG command for the partition involved is sensed.

The format for the LOG job control command via attention routine is as follows:

    LOG blank

The operand field is ignored by the system.

NOLOG

The NOLOG command (suppress logging) terminates the listing, on SYSLOG, of Job Control commands and statements (except JOB, PAUSE, STOP, ALLOC, MAP, HOLD, RELSE, UNA, DVCDN, DVCUP, *, and /&) that occur in the batched-job partition in which the NOLOG is issued. The NOLOG function is effective until a LOG command for the partition involved is sensed.

The format for the NOLOG job control command via attention routine is as follows:

    NOLOG blank

The operand field is ignored by the system.

MAP command

The MAP command produces on SYSLOG a map of virtual storage areas allocated to programs.

An example of the output produced on the console printer is shown below:

```
F2  map
F2  AREA     K-REAL UPPER LIMIT  K-VIRT UPPER LIMIT    NAME
F2
F2   SP       92K      94207              262144
F2   BG  V5A  30K     124927      292K    561151      KENSLOOP
F2  F4   I4   10K     135167       80K    643071
F2  F3   I3   10K     145407       80K    724991
F2  F2   V2A  14K     159743      100K    827391      NO NAME
F2  F1   I1   82K     243711      120K    950271
F2  SVA                            96K   1048575
F2  VIS                            68K   1048575
F2  PP        18K     262143
```

	SP = Supervisor, **V** = Virtual, **PP** = Main Page Pool, **I** = Inactive, **SVA** = Shared Virtual Area, **R** = Real, **A** = Active 1, 2, 3, 4, 5 = Partition Priority (1 = highest priority), VIS = Amount of SVA reserved by GETVIS parameter of the VSTAB system generation macro
K-REAL	gives the number of bytes allocated to the real partition or the number of bytes of the main page pool. The size is given in multiples of 2K.
UPPER LIMIT	shows the highest storage addresses in decimal of the respective real partition, of the supervisor, and of the main page pool.
K-VIRT	specifies the number of bytes allocated to the respective virtual partition. The size is given in multiples of 2K. This field is blank for the supervisor and for the main page pool.
UPPER LIMIT	contains the highest storage address in decimal of the respective virtual partition. For the supervisor, this field specifies the start address of the virtual address area. This field is blank for the main page pool.
NAME	contains the name of the job which is currently executing in the corresponding partition. This field is blank for the supervisor, SVA, VIS, and for the main page pool. When the listing shows NO NAME for the background, or when the name field is blank for a foreground partition, no program is being executed in that area. However, the name field for any partition contains NO NAME when no job control statement or command was entered, but the program is active.

Note: If a program issues an SVC55, some page frames in the main page pool (PP) will also belong to that program. Therefore to calculate the total area in real storage occupied by that program, the MAP command should be issued before running it. The difference between the number of K in the PP before running the program, and the number of K during the execution of the program is the amount of K seized by the SVC55.
This also applies when PDAID output mode is an alternate area or the SDAID is initiated.
In this case, the area occupied by the PDAID or SDAID is printed during their initialization.

Note: The output does not indicate storage temporarily added to the page pool as a result of using the SIZE parameter of the EXEC statement.

JOB CONTROL COMMANDS AND STATEMENTS

OPTION

The OPTION statement specifies one or more of the Job Control options. The format of the OPTION statement is:

JCS Format

// OPTION option 1 (,option2,...)

The options that can appear in the operand field follow. Selected options can be in any order. Options are reset to the standards established at system generation time upon encountering a JOB or a /& statement.

DUMP — Causes a dump of the registers and main storage to be output on SYSLST, if assigned, in the case of an abnormal program end. (See A-2 in this section for a full description.)

NODUMP — Suppresses the DUMP option, if the latter was specified in the STDJC macro during system generation.

LOG — Causes the listing of columns 1-80 of all control statements on SYSLST. Control statements are not listed until a LOG option is encountered. Once a LOG option statement is read, logging continues from job-step to job-step until NOLOG option is encountered or until either the JOB or /& control statement is encountered.

NOLOG — Suppresses the listing of all valid control statements on SYSLST until a LOG option is encountered. If SYSLST is assigned, invalid statements and commands are listed.

LIST — Causes language translators to write the source module listing on SYSLST. In addition, it causes the Assembler to write the hexadecimal object module listing and causes the Assembler and the FORTRAN compiler to write a summary of all errors in the source program. All are written on SYSLST.

NOLIST — Suppresses the LIST option.

PAUSE

The PAUSE command causes a pause at the end of the current job step. The PAUSE Job Control statement causes a pause immediately after processing this statement. At the time, SYSLOG is unlocked for message input. END (on the Models 135 and 145) or ENTER (on the Model 125) causes processing to continue. The PAUSE statement or command always appear on SYSLOG. If a 3210 or 3215 or video console display unit is not available, the PAUSE statement or command is ignored.

This is an area of real storage, starting at byte address 000000, and permanently reserved for use by the supervisor.

For the purpose of program debugging, low address storage extends up to byte address 160 decimal (X'BF')

Displaying low address storage

Low address storage will always be dumped during the execution of:

- A stand-alone dump; see A-2 in this section.
- A system dump; see A-2 in this section.
- A transient dump (bytes 0-144 hex); see A-4 in this section.

Low address storage can also be dumped by means of DUMP or DSPLY operator commands (see A-1 in this section), or the ALTER/DISPLAY feature on the console printer or display unit keyboard (see D-1 in this section).

When to display

- Low address storage must be dumped by using the ALTER/DISPLAY console printer feature whenever a hard wait is recognized.
- When a system malfunction is recognized in one of the programs in a multiprogramming or teleprocessing environment, low address storage must be dumped by using the DUMP or DSPLY commands in order to avoid total system collapse.

Flowchart D-1-F in this section shows <u>how</u> to dump low address storage by using the ALTER/DISPLAY feature on the console printer. Flowcharts in Section 3 indicate <u>when</u> to dump low address storage.

CAW (Channel Address Word)

The CAW specifies the storage protection key and the address of the first channel command word associated with the START I/O instruction. The CAW is found at hex location 48.

KEY	0000	Channel Command Word Address
0	4	8 31

Note: After the execution of any dump program, the information in the CAW is unreliable. In this case, the start address of the CCW is found in the command control block (CCB).

Locating CCBs is described in Section 4.

psw

LOW ADDRESS STORAGE

CSW (Channel Status Word)

The CSW informs the program of the status of an I/O device or the conditions under which an I/O operation has been terminated. The CSW is formed, or parts of it are replaced, during I/O interruptions and during execution of I/O instructions. The CSW is placed in low address storage at location hex 40. It is available to the program at this location until the next I/O interruption occurs or until another I/O instruction generates a new CSW, whichever occurs first. When the CSW is stored as a result of a START I/O instruction, the I/O device is identified by the I/O address in the old PSW. The information placed in the CSW by an I/O instruction pertains to the device addressed by the instruction.

The CSW format is shown below.

Key	0 L CC	Channel Command Address	Unit Status	Channel Status	Count
0	4	8	32	40	48 … 63

Bits 0- 3	Protection key used in the last operation
Bit 5	Reserved
Bits 6- 7	Deferred condition code
Bits 8-31	Address plus 8 of the last CCW used
Bits 32-39	contain the unit status byte: Bit 32 –– attention Bit 33 –– status modifier Bit 34 –– control unit end Bit 35 –– busy Bit 36 –– channel end Bit 37 –– device end Bit 38 –– unit check Bit 39 –– unit exception
Bits 40-47	contain the channel status byte: Bit 40 –– program-controlled interruption Bit 41 –– incorrect length Bit 42 –– program check Bit 43 –– protection check Bit 44 –– channel data check Bit 45 –– channel control check Bit 46 –– interface control check Bit 47 –– chaining check
Bits 48-63	Residual count of the last CCW used

Note: After the execution of any dump program, the information in the CSW is unreliable. In this case, CSW information is found in the command control block (CCB).

Locating CCBs is described in Section 4.

E-2

PSW (Program Status Word)

The PSW contains information required for the program execution. By storing the PSW, the control program can preserve the status of the CPU for later inspection. By loading a new PSW or part of a PSW, the status of the CPU can be changed.

The format of old and new PSWs is the same as that of the current PSW, shown below:

U	R	U	U	O	T	I	E	Protection key	1	M	W	P	U	U	C	C	Program mask	Unassigned	Reserved	Instruction address
0								8					16				20	24	33	40 63

U indicates the bit is unassigned.
0 indicates the bit is set to zero.
1 indicates the bit is set to one.

PROGRAM EVENT RECORDING MASK (Bit 1).	If ON, permits interruptions subject to the program-event control bits in control register 9.
TRANSLATION MODE (Bit 5).	If ON, invokes the dynamic address translation (DAT) services.
I/O MASK (Bit 6).	It ON, enables I/O interruptions subject to the channel mask bits in control register 2.
EXTERNAL MASK (Bit 7).	If ON, enables external interruptions subject to the corresponding external sub-class mask bits in control register 0.
PROTECTION KEY (bits 8-11).	Is compared with a storage key whenever a result is stored, or information is fetched from a protected location.
EXTENDED CONTROL MODE INDICATOR (Bit 12).	If ON, indicates that the supervisor operates in Extended Control (EC) model.
MACHINE CHECK MASK (Bit 13).	If ON, enables machine check interruptions resulting from system damage or instruction-processing damage; other machine check interruptions are enabled subject to the sub-class mask bits in control register 14.
WAIT STATE (Bit 14).	If ON, indicates that the CPU is in the Wait State.
PROBLEM STATE (Bit 15).	If ON, indicates that the CPU is in the Problem State; if OFF, the CPU is in the Supervisor State.
CONDITION CODE (bits 18-19).	Is set as the result of the execution of certain instructions.
PROGRAM MASK (Bits 20-23)	comprises: Fixed-Point Overflow Mask Decimal Overflow Mask Exponent Underflow Mask Significance Mask. A Mask bit ON enables an interruption when the specified exception occurs. The Significance Mask bit also determines the manner in which floating-point addition and subtraction are completed.
INSTRUCTION ADDRESS (Bits 40-63).	For all PSWs, the address is that of the next logical instruction. In addition, for the new PSWs the address points to the routine that handles the particular interrupt, and for the old PSWs it contains the return address in the calling routine.

LOW ADDRESS STORAGE

Displacement in hexadecimal	Description (all numbers referenced are in hexadecimal).
0-7	The field is used for the following two functions: Restart New PSW: The new PSW is fetched from locations 0-7 during the restart operation. IPL PSW: The first eight bytes read during the IPL initial read operation are stored at locations 0-7. The contents of these locations are used as the new PSW at the completion of the IPL operation. These locations may also be used for temporary storage at the initiation of the IPL operation.
8-F	The field is used for the following two functions: Restart Old PSW: The current PSW is stored as the old PSW at locations 8-F during the restart operation. IPL CCW1: Bytes 8-F read during the IPL initial read operation are stored at locations 8-F. The contents of these locations are ordinarily used as the second CCW in an IPL CCW chain after completion of the IPL initial read operation.
10-17	IPL CCW2: Bytes 10-17 read during the IPL initial read operation are stored at locations 10-17. The contents of these locations may be used as the third CCW of an IPL CCW chain after completion of the IPL initial read operation. After IPL bytes 14-17 contain the address of the background partition communication region. Thereafter they contain the address of the communication region for the active partition. (Communication regions are described in Section 4).
18-3F	Interruption Old PSWs: The current PSW is stored as the old PSW at locations 18-1F, 20-27, 28-2F, 30-37, and 38-3F during the external, supervisor-call, program, machine-check, and input/output interruptions, respectively.
40-47	CSW: The channel status word (CSW) is stored at locations 40-47 during an I/O interruption. It, or portions thereof, may be stored during the execution of START I/O, START I/O FAST RELEASE, TEST I/O, HALT I/O, or HALT DEVICE, in which case condition code 1 is set.
48-4B	CAW: The channel address word (CAW) is fetched from locations 48-4B during the execution of START I/O and START I/O FAST RELEASE.
4C-4F	Save area for job duration measurement when the interval timer location is being used by the supervisor IT option routines.
50-53	Interval Timer: Locations 50-53 contain the interval timer. The timer is updated whenever the CPU is in the operation state. Depending on the resolution of the timer, the low-order locations may not be updated.
54-57	Contain the time of day.
58-7F	Interruption New PSWs: The new PSW is fetched from locations 58-5F, 60-67, 68-6F, 70-77, and 78-7F during the external, supervisor-call, program, machine-check, and input/output interruptions, respectively.
80-83	The address of the system communication region, described in Section 4.
84-87	External-interrupt Code: During an external interruption in the EC mode, the interruption code is stored at locations 86-87 and zeros are stored at locations 84-85.
88-8B	SVC-Interrupt Code: During a supervisor-call interruption in the EC mode, the instruction-length code is stored in bit positions 5 and 6 of location 89, and the interruption code is stored at locations 8A-8B. Zeros are stored at location 88 and in the remaining bit positions of 89.
8C-8F	Program Check Interrupt Code: During a program interruption in EC mode the instruction-length code is stored in bit positions 5 and 6 of location 8D, and the interruption code is stored at locations 8E-8F. Zeros are stored at location 8C and in the remaining bit positions of 8D.
90-93	Translation-Exception Address: During a program interruption due to a segment-translation exception or a page-translation exception, the translation-exception address is stored at locations 91-93, and zeros are stored at location 90.
94-95	Monitor-Class Code: During a program interruption due to a monitor event, the monitor-class number is stored at location 95, and zeros are stored at 94. This field can be stored in either the BC or EC mode.
96-97	PER-Interrupt Code: During a program interruption due to a program event, the program-event-recording (PER) code is stored at location 96, and zeros are stored at 97. This field can be stored only when the instruction causing the PER condition was executed under the control of a PSW specifying the EC mode.
98-9B	PER Address: During a program interruption due to a program event, the program-event-recording (PER) address is stored at locations 99-9B, and zeros are stored at location 98. This field can be stored only when the instruction causing the PER condition was executed under the control of a PSW specifying the EC mode.
9C-9F	Monitor Code: During a program interruption due to a monitor event, the monitor code is stored at locations 9D-9F, and zeros are stored at location 9C. This field can be stored in either the BC or EC mode.

Table E-2 Format and contents of low address storage.

WAIT STATE MESSAGES

Bytes 0 - 3 of low address storage are used to store and record coded messages when a system malfunction occurs during IPL. Other occasions when coded messages are stored in these bytes are described below under "Wait states during problem program execution."

Whenever a wait state occurs, it is imperative that these low address storage bytes are dumped by using the console printer ALTER/DISPLAY feature, described in D-1 of this section.

The table below lists all the coded wait state messages:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	EXPLANATION
MCH/CCH/IPL Hard Wait Codes placed in low address storage				
X'C1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable machine check.
X'C2'	X'E2'(2)	Not used	Not used	Irrecoverable channel failure during RMS fetch.
X'C3'	X'E2'(2)	A, I, S(1)	Not used	Channel failure on SYSLOG when RMS message scheduled.
X'C4'	X'E2'(2)	A, I, S(1)	Not used	No ECSW stored
X'C5'	X'E2'(2)	A, I, S(1)	Not used	Channel failure: ERPBs exhausted.
X'C6'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; two channels damaged or a damaged channel situation occurred while RMS was executing an I/O operation.
X'C7'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system reset was presented by a channel.
X'C8'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system codes in ECSW are invalid.
X'C9'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; channel address invalid.
X'D1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable channel failure on SYSVIS.
X'07'	X'E6'	Channel	Unit or X'00'	IPL I/O error or equipment malfunction; condition code 2 during STIDC instruction. Channel and unit indicate whether device in error is SYSRES or communication device. When byte 3 = X'00', byte 2 indicates the channel for which STIDC instruction was issued. Re-IPL system.
SDAID Hard Wait Code				
X'61'	X'E6'(3)	Channel	Unit	Another device is running in burst mode on same channel as SDAID output device. Re-IPL system.
SDAID Soft Wait Code				
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'(3)	X'00'	X'00'	SDAID Stop on Event. Press EXTERNAL INTERRUPT key to continue operations.
The following Hard Wait Codes are placed in general register 11 X'B' as well as in low address storage.				
X'00'	X'00'	X'0F'	X'FF'	Program Check in Supervisor.
X'00'	X'00'	X'0F'	X'FE'	I/O error during fetch from System CIL.
X'00'	X'00'	X'0F'	X'FD'	Channel Failure if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
X'00'	X'00'	X'0F'	X'FC'	Machine Check if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
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'FA'	Translation Specification Exception
X'00'	X'00'	X'0F'	X'F9'	Error on Paging I/O.
X'00'	X'00'	X'0F'	X'F8'	CRT phase not found.
X'00'	X'00'	X'0F'	X'F7'	No copy blocks available for BTAM appendage I/O request.
X'00'	X'00'	X'0F'	X'F6'	$MAINDR canceled during system CIL update. If this occurs, the system CIL is only partially updated and must be restored before use. This hard wait condition can also occur if the FETCH QUEUE BIT (FCHQ) is set in the linkage control byte in the partition communication region owned by the terminating partition.
Device Error Recovery Soft Wait Codes placed in low address storage.				
X'08' to	X'C1' or	Channel	Unit	Error recovery messages. Refer to 0P messages in *DOS/VS Messages.*

Notes: 1. A (X'C1') = SYSREC recording unsuccessful.
I (X'C9') = SYSREC recording incomplete.
S (X'E2') = SYSREC recording successful.
2. S(X'E2') = Run SEREP.
3. SDAID wait states are identified by X'EEEE' in the address part of the wait PSW.

Table E-3 Wait State codes

WHENEVER "IPL" LOAD KEY CAUSES DESTRUCTION OF THE FIRST 24 BYTES ∴ DISPLAY MAIN SHOULD BE DONE & THE IPL ERROR MESSAGE COPIED BEFORE RE IPL-ING WHEN SYS BLEW DURING IPL.

Wait states during IPL

If the system enters the wait state during an IPL procedure and no message is printed on SYSLOG, the operator should record at least the first five bytes of low address storage. The IPL error message number and action code are displayed in hex in these bytes. For example:

Message 0I11A appears in low address storage bytes 0-4 as

F0C9F1F1C1

0 I 1 1 A

The operator should look up this message in *DOS/VS Messages* and perform the indicated action, except for the messages noted below.

IPL error messages

If there is an equipment malfunction during IPL, or the IPL cannot be loaded, an IPL error message is placed in bytes 0-3. In this state all interrupts are disabled, and you must repeat IPL after dumping low address storage, as shown in flowchart D-1-F in this section.

E-3

Byte 0	Byte 1	Byte 2	Byte 3	Meanings:
X'07'	X'E6'	Channel	Unit or X'00'	IPL input/output error: • I/O error on SYSRES 01 - 1 • I/O error on communication device (see notes 1 and message X'F0C9F0F1' below) • Equipment malfunction during the STORE CHANNEL ID instruction (see note 3) • Supervisor entry not found.
X'F0'	X'C9'	X'F0'	X'F0'	This code indicates that less than 16K of real storage is left for problem programs. Check that the correct disk volume is mounted on the device assigned to SYSRES and re-IPL. If the error recurs, the system programmer must check the allocations of real partitions specified in the supervisor to be used, and check that at least 16K of real storage is available for execution of problem programs running in virtual mode.
X'F0'	X'C9'	X'F0'	X'F1'	If a card reader has been assigned to SYSRDR during system generation and is to be the IPL communication device, press the INTERRUPT key. If a card reader has **not** been assigned to SYSRDR during system generation and yet it is to be the IPL communication device, simply READY the reader.
X'F0'	X'C9'	X'F0'	X'F2'	This code means that the supervisor requested can not be found. Check that the correct disk volume is mounted on the device assigned to SYSRES. If it is correct, re-IPL and specify a different supervisor when message 0I03A is issued and press the END/ENTER key, or press END/ENTER key only, to load the standard supervisor. (If possible contact the system programmer and check which supervisor to use.)
X'F0'	X'C9'	X'F1'– X'F2'	X'F0'– X'F8'	Refer to messages 01100A – 0128A in *DOS/VS Messages* in the *DOS/VS Messages* manual

Note 1: *When the IPL procedure reaches the normal IPL wait state, and the IPL communication device is to be SYSLOG, press the REQUEST key on the console printer keyboard.*

Note 2: *When byte = X'00', byte 2 indicates the channel for which the STIDC instruction was issued.*

WAIT STATE MESSAGES

Wait states during program execution

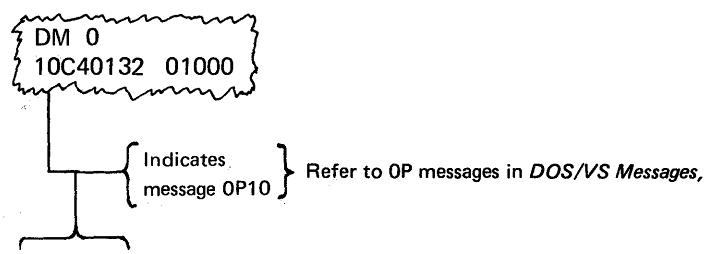
Three conditions will place a coded message in low address storage during program execution:

- I/O device error
- Hardware failures
- Unrecoverable I/O error during FETCH.

1. I/O device error messages.
   Device Error Recovery Messages.
   The example below shows the information that is placed in low address storage bytes hex 0-3 when a wait state is caused by an I/O device error, and both SYSLOG and SYSLST are inoperative, or SYSLOG is not assigned.
   An example of a coded device error recovery message as it is stored in the low address storage is shown below:

   0P08A INTERV REQ SYSLST=00E

An example of a device error recovery message is shown below:

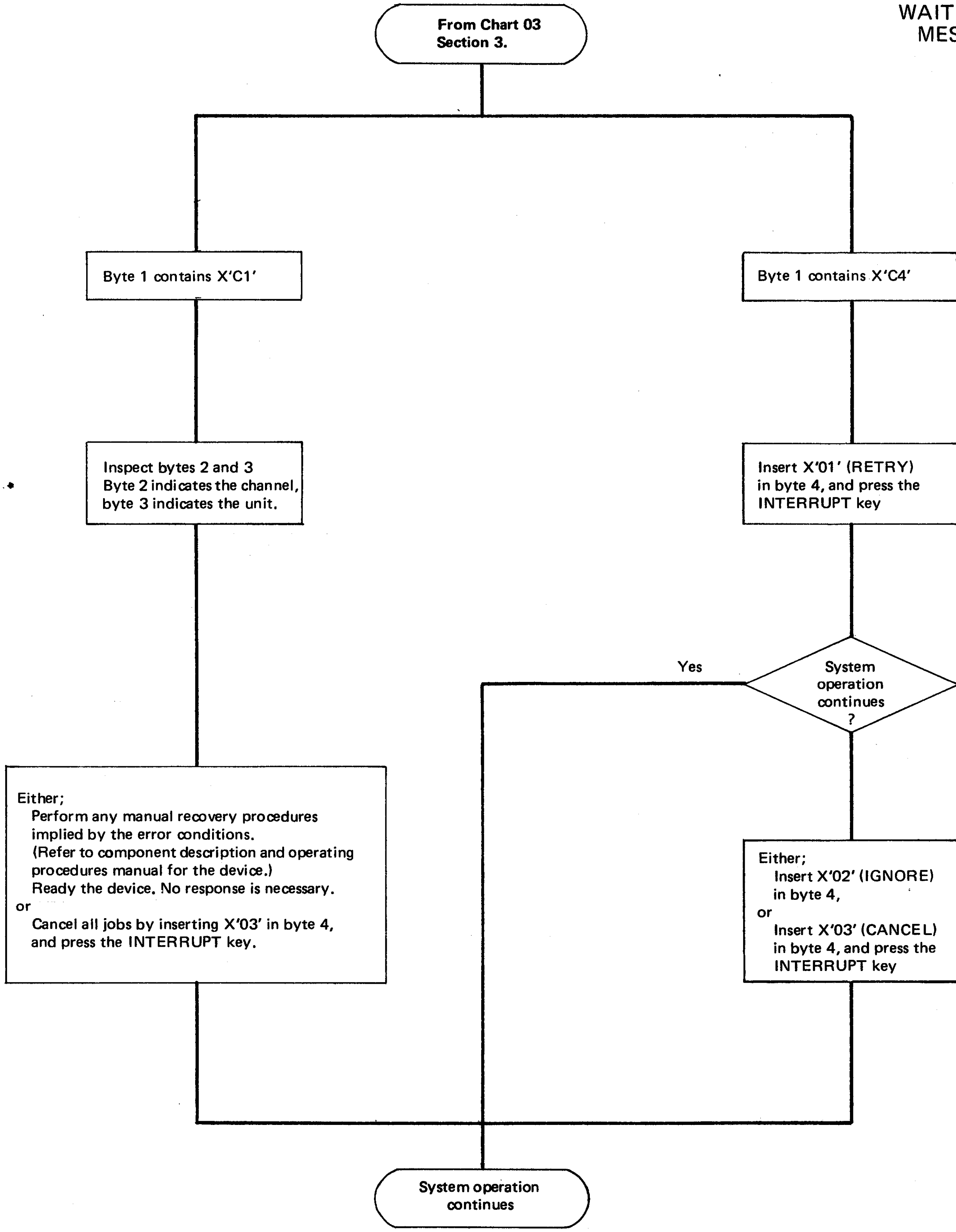


Byte 0	Byte 1	Byte 2	Byte 3
X'08' to X'60'	Action indicator 'C1' – 'C4'	Channel	Unit

If this condition occurs, the operator must dump, or display and note, bytes 0-3 of low address storage, and inspect the contents of byte 0. This byte contains a hex number that corresponds to an OP message listed in *DOS/VS Messages.*

Before proceeding with system operation, the operator will have to decide whether to continue system operations with the program currently running or to CANCEL all jobs and repeat IPL. This decision depends on system commitments and the importance of other programs that are running.

To continue operations, the operator must first inspect the contents of byte 1 for the presence of a hex C1 or C4. The flowchart opposite shows what further actions can be taken under these conditions, and a flowchart in Section 3 indicates to operators when this procedure may be required.



Operator procedure to recover from an input/output device error, when the device error recovery message cannot be printed on either SYSLOG or SYSLST.

WAIT STATE MESSAGES

2. Hardware failure

If a hardware failure occurs that cannot be corrected by the RAS transients (R-transients), a message is normally printed on SYSLOG. If this is not possible a coded message is placed in low address storage.

A complete list of wait state messages is given in table E-3.

3. Unrecoverable I/O error during FETCH

If an unrecoverable I/O error occurs during a FETCH operation, a coded message is placed in the low address storage bytes 0-3.

The contents of the following 24 bytes starting at byte 4 will contain device sense information.

(The number of sense bytes is device-dependent).

The sense data is useful to your IBM customer engineer and may also be of use to your system programmer. The component manual for the failing device provides details about the cause of failure. Before repeating IPL, try a different drive.

THE LINKAGE EDITOR MAP

A linkage editor map is an aid to program debugging. This map is obtained during link-editing when SYSLST is assigned (unless ACTION NOMAP was specified). Details about link-editing are found in *DOS/VS System Control Statements.*

Description

When used in conjunction with a storage dump and program listings, the linkage editor map will help in locating programs and subroutines that are included in the programs at object time. Common areas, load address, relocation factors, low and high addresses are also shown. In addition, the PHASE card is displayed to show where the phase was loaded, which is also helpful when working with multi-phase programs.

The linkage editor map also shows where programs should be located in virtual storage, where overlays are loaded, and whether the program is relocatable, self-relocating, non-relocatable or SVA-eligible. The example below shows a linkage editor map.

```
12/06/73   PHASE    XFR-AD  LOCORE  HICORE  DSK-AD     ESD TYPE  LABEL     LOADED  REL-FR
           PHASE***  040078  040078  04232F  047 13 3  CSECT     BEGIN     040078  03C878  RELOCATABLE

                                                       CSECT     IJFFZZWZ  041ED8  041ED8
                                                       *  ENTRY  IJFFZZZZ  041ED8

                                                       CSECT     IJCFZIZO  042230  042230

                                                       CSECT     IJDFCZZZ  0422A0  0422A0
                                                       *  ENTRY  IJDFZZZZ  0422A0

                                                       CSECT     IJ2L0067  0422F8  0422F8
```

(Ex 60 KT)

The next two illustrations show an example of the DIAGNOSTIC OF INPUT, and virtual storage MAP, which are printed on SYSLST during link-editing.

The example contains errors which are discussed in the text following each figure.

How to use

Refer to A-2 in this section for an example that shows how the map is used during debugging in conjunction with a system dump and program listing. Examples at the end of this chapter show the information reports that immediately follow the map. These reports confirm that the new phase, or phases, are correctly cataloged, and enable you to monitor the status of your libraries.

Other Aids

THE LINKAGE EDITOR MAP

```
     JOB      EXAMPLE                              DISK LINKAGE EDITOR DIAGNOSTIC OF INPUT
 1   ACTION   TAKEN    MAP    CLEAR
 2   LIST              PHASE   PHASE1,ROCT,NOAUTO
 3   LIST             INCLUDE  ,(NAMEONE)
 4   21411    EX1 0002 ESD 404040 0010 0002 POINT3  1 000244 000003 NAMETWO  2 FF0130 0000CA NAMTHREE 0 000200 0000A8
 5   LIST             PHASE     PHASE2,x,NOAUTO
 6   LIST               INCLUDE       ,(NAMEFOUR)
 7   21411    EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO   2 FF0130 0000CA NAMTHREE 0 000200 0000A8
 8   LIST               PHASE PHASE3,PHASE1+73,NOAUTO
 9   LIST               INCLUDE   ,(NAMETWO,NAMTHREE)
10   21411    EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000007 NAMETWO   0 000130 001898 NAMTHREE 0 000200 0000A8
11   LIST           INCLUDE RELMOD
12   21311            INCLUDE RELMOD
13   LIST     EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO   0 000130 001928 NAMTHREE 0 000200 0000A8
14   LIST               PHASE     PHASE4,+16500
15   LIST        INCLUDE    RELMODUL
16   21441    REL 0015 TXT 00425C 0038 F0F1 1A361A56 46D0E254 4130EF1E D500EF1E E5FA477C E2869201 EF1E0630 9509EF1D 4770E286
17   LIST     AUTOLINK     AUTOMOD2
18   LIST               PHASE PHASE5,+X'25BA',NOAUTO
19   LIST     REP 0C4018 0034130,C03A,47F0,C30E           PATCH ASSEMBLY ERRORS
20   LIST     REP 0C40CC 003D20E,
21   21021             REP 0040C0 C3F0 0003 D20EF0C5 6B404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040
22   LIST     ENTRY             INVALID TRANSFER LABEL
```

Line 1 (ACTION TAKEN). MAP and CLEAR have been specified on separate ACTION cards. Had NOAUTO been specified, it would also appear on this line.
Lines 4, 7, 10, and 13. Error 21411 (duplicated ESID number) is printed four times because the submodular structure of the phase demanded four passes over the same module. As the linkage editor processes in its own input area, the record printed may not be identical to the original input record. Lines 10 and 13 differ in content from lines 4 and 7 for this reason.
Lines 11 and 12. Line 11 is printed when the statement is read by the linkage editor. Line 12, error 21311 indicating that the requested module is not in the relocatable library, is printed after the error is detected.
Line 16. This is an example of an error detected in a TXT statement. Error 21441 indicates that the ESID number F0F1 is invalid. (It should be binary 01.)
Line 17. Indicates the AUTOLINK feature was used for relocatable library module inclusion in the phase named above it.
Line 19. An example of a valid REP statement.
Lines 20 and 21. An example of an invalid REP statement. Line 20 is printed when the statement is read by the linkage editor. Line 21, error 21021 indicating an invalid operand in the statement, is printed after the error is detected.
When a module is included from the relocatable library, it is not possible to guarantee that the sequence identification printed in columns 8-15 is that of the record printed. This occurs because the MAINT librarian program reblocks the content of the cards to a more compressed format.

```
 1          PHASE     XFR-AD  LOCORE  HICORE  DSK-AD     ESD TYPE  LABEL     LOADED  REL-FR
 2 COMMON                                                COM                 001800  0000C8
 3 ROOT     PHASE1    0018C8  0018C8  0019F7  013  2 2   CSECT     NAMEONE   0018C8  0C18C8   NOT RELOCATABLE
 4                                                         ENTRY   POINT1    0018CC
 5                                                       × ENTRY   POINT2    001930
 6          PHASE 2   0019F8  0019F8  001A87  013  3 1   CSECT     NAMEFOUR  0019F8  0C1750   NOT RELOCATABLE
 7 OVEROOT  PHASE 3   0019E8  001918  001B1F  013  3 2   CSECT     NAMETWO   001918  0017E8   NOT RELOCATABLE
 8                                                       CSECT     NAMTHREE  0019E8  0C17E8
 9                                                       × ENTRY   POINT3    001A2C
10                                                       CSECT     NAMEFOUR  001A90  0C17E8
11          PHASE4    0043A0  004140  0059A3  013  4 1   CSECT     AUTOMOD1  004140  003A98   NOT RELOCATABLE
12                                                         ENTRY   AUTOENT   0042D0
13                                                       CSECT               0043A8  0C3EF8
14                                                       CSECT     AUTOMOD2  0043C0  0003C0
15          PHASES    002688  002688  002767  013  6 1   CSECT               002688  -001B08  NOT RELOCATABLE
16                                                       CSECT     NAME5     002688  -001B08
17 *UNREFERENCED SYMBOLS                                 EXTRN     POINT2
18                                                       EXTRN     POINT4
19 ROOT STRUCTURE OVERLAID BY SUCCEEDING PHASE
20 POSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT
21 INVALID TRANSFER LABEL ON END OR ENTRY STATEMENT IGNORED
22 CONTROL SECTIONS OF ZERO LENGTH IN INPUT
23 002 UNRESOLVED ADDRESS CONSTANTS
24 003 ADDRESS CONSTANTS OUTSIDE LIMITS OF PHASE
```

E-4

Line 2 (COMMON). The entry under REL-FR contains the length instead of the relocation factor in the case of ESD-type COMMON.

Lines 5 and 9 (referring to UNREFERENCED SYMBOLS). These ENTRY labels (POINT2 and POINT3) are not referenced as external symbols, that is, by corresponding EXTRN statements.

Line 17 and 18. These labels indicate EXTRN references that cannot be matched with a corresponding entry point. In such a case *ENTRY ESD-types may be the corresponding, but misspelled, point. In the submodular structure, CSECTs not specified in any namelist appear as EXTRNs. The labels can also indicate unreferenced EXTRNs.

Line 3, 6, 7, 11, and 15. All phase origins (entries under LOCORE) are incremented by the length of COMMON.

Line 19. Warning message. When this message appears, OVEROOT is printed to the left of the name of the phase (PHASE3) that overlays the ROOT phase.

Line 20. Warning message. An entry label appeared at least twice in the input stream. At its second (or later) appearance, it was not possible to validate it as being a true duplication. The most common reason for this message is submodular structure with (source) ENTRY labels defined before the CSECT in which the entry point appears.

Line 21. An overriding transfer label in the ENTRY statement was not defined within the first phase, or a transfer label was not defined in an END statement in its module.

Line 24. Address constants had load addresses outside the limits of the phase in which they occurred. This usually occurs if the control section length is incorrectly defined in the input.

Line 22. Warning message. The COBOL, FORTRAN, RPG, and PL/I (D) compilers do not supply all of the information required by the linkage editor in the ESD records. Specifically, the control section length is provided in the END record. If a control section defined in the ESD information has a length of zero, it normally indicates that the length is to appear in the END record. It is possible to generate zero-length control sections through assembler. Such a condition produces this message. This is not an invalid condition if it is not the last control section that is of zero length. If the last control section is of zero length, the length is implied to be in the END record and, if not present, causes an error condition.

Line 23. These address constants correspond to the EXTRNs shown in line 17 and 18.

THE LINKAGE EDITOR MAP

SYSTEM DIRECTORY—SYSRES	CORE-IMAGE	RELOCATABLE	SOURCE-STATEMENT	PROCEDURE
12/06/73	DECIMAL			
	C H R E	C H R E	C H R E	C H R E
DIRECTORY STARTING ADDRESS	00 10 01	16 00 01	31 00 01	41 00 01
DIRECTORY NEXT ENTRY	00 11 15 15	16 01 07 02	31 00 08 03	41 00 03 04
DIRECTORY LAST ENTRY	00 14 15 17	16 04 17 19	31 04 27 09	41 04 27 09
LIBRARY STARTING ADDRESS	00 15 01	16 05 01	31 05 01	41 05 01
LIBRARY NEXT AVAILABLE ENTRY	13 13 02	27 04 05	38 14 05	41 09 22
LIBRARY LAST AVAILABLE ENTRY	15 19 04	30 19 16	40 19 27	45 19 40
	STATUS INFORMATION			
DIRECTORY ENTRIES ACTIVE	537	457	68	19
LIBRARY BLOCKS ALLOCATED	1220	4720	5265	3795
LIBRARY BLOCKS ACTIVE	1033	3508	4027	176
LIBRARY BLOCKS DELETED	00	00	00	00
LIBRARY BLOCKS AVAILABLE	187	1212	1238	3619
AUTOMATIC CONDENSE LIMIT	00	00	00	00
LIBRARY ALLOCATED CYLINDERS	15	15	10	05
DIRECTORY ALLOCATED TRACKS	05	05	05	05

(Ex 61 [illegible])

An example of the SYSTEM DIRECTORY status information printout that immediately follows the MAP, after cataloging a phase to the system core image library on SYSRES.

PRIVATE DIRECTORY	PRV-CORE IMAGE	PRV-RELOCATABLE	PRV-SOURCE STATEMENT
12/06/73	DECIMAL		
	C H R E	C H R E	C H R E
DIRECTORY STARTING ADDRESS	47 10 01	73 00 01	103 00 01
DIRECTORY NEXT ENTRY	47 12 06 07	73 02 09 13	103 01 15 07
DIRECTORY LAST ENTRY	47 14 15 17	73 09 17 19	103 09 27 09
LIBRARY STARTING ADDRESS	47 15 01	73 10 01	103 10 01
LIBRARY NEXT AVAILABLE ENTRY	71 09 01	98 04 06	196 05 02
LIBRARY LAST AVAILABLE ENTRY	72 19 04	102 19 16	196 19 27
	STATUS INFORMATION		
DIRECTORY ENTRIES ACTIVE	631	848	412
LIBRARY BLOCKS ALLOCATED	2020	9440	50490
LIBRARY BLOCKS ACTIVE	1812	7909	50086
LIBRARY BLOCKS DELETED	84	00	00
LIBRARY BLOCKS AVAILABLE	124	1531	404
AUTOMATIC CONDENSE LIMIT	00	00	00
LIBRARY ALLOCATED CYLINDERS	26	30	94
DIRECTORY ALLOCATED TRACKS	05	10	10

(Ex 62 [illegible])

An example of the PRIVATE DIRECTORY status information printout that immediately follows the MAP after cataloging a phase to the private core image library. SYSTEM DIRECTORY status information, shown above is printed following this report.

Note: The format of this printout depends on whether SYSCLB, SYSRLB, and SYSSLB were assigned before the linkage editor run.

Summary

The following list summarises the information contained in the map.

1. The name of each phase, the lowest and highest virtual storage locations of each phase, and an indication if the phase is relocatable, non-relocatable, self-relocating or SVA-eligible. It also shows the disk address in hex where the phase begins in the core image library.
2. An indication if the phase is a ROOT phase, or if a phase overlays the ROOT phase in any way (designated by OVERROOT).
3. The length of COMMON, if appropriate.
4. The names of all CSECTs belonging to a phase, the address where each CSECT is loaded, and the relocation factor of each CSECT.
5. All defined entry points within a CSECT. If an entry point is unreferenced, it is flagged with an asterisk (*).
6. The names of all external references that are unresolved.
7. The transfer (execute) address of each phase.
8. Warning messages are printed if:
   - The ROOT phase has been overlaid.
   - A possible invalid entry point duplication occurred.
   - The ENTRY or END statement contained an invalid (undefined) transfer label.
   - At least one control section had a length of zero.
   - The assembled origin on an RLD statement was outside the limits of the phase.
   - An address constant could not be resolved.

These messages may or may not indicate actual programming errors. If NOMAP is operational, the warning messages are not printed.

Intentionally Blank

CONTENTS

RMS

General Description of RMS

The need for the IBM serviceability aids that are collectively termed RMS (Recovery Management Support) has been described in Section 1 under the heading "Hardware Failures."

RMS consists of software routines that are grouped according to their function:
- MCAR (Machine Check Analysis and Recovery)
- CCH (Channel Check Handler)
- ERP (I/O device Error Recovery Procedures)
- RMSR (Recovery Management Support Recorder).

Each function listed above is considered to be a function of RMS, and if required, must be included in your supervisor during system generation. The function RMSR consists of several recording facilities:
- Unit check recording
- Machine check and channel check recording
- Tape/disk error statistics by volume
- MDR (Miscellaneous Data Recorder)
- IPL information } Reliability Data
- End of Day recording for devices and for the system. } Extractor (RDE)

RMS is always supported on the Models 135, 145, 155-11 and 158, and the RMSR facilities supported depends on the parameters specified during system generation. The parameters of the supervisor macros affecting the subjects described in this section are discussed here but further details required for generating a supervisor are found in the *DOS/VS System Generation* manual.

System Requirements

In order to perform its functions, RMS uses two logout areas contained in real storage:
- The fixed logout area
- The model-dependent log out area (not applicable to Models 115 and 125).

As the name "model-dependent logout area" implies, the real storage area reserved for the logout areas varies for different System/370 Model types. Therefore, if you know during supervisor generation that the supervisor will be used on a larger model, specify the larger model in the MODEL = parameter of the CONFG supervisor generation macro.

Because the Models 115 and 125 employ both software and hardware recording functions a more detailed description on these Models is given in the following paragraphs. For the Model 125, a hardware function records CPU and channel hardware failures on the DISKETTE. This also applies to hardware failures of natively attached I/O devices. Device ERP is always supported for all models. When the Models 115 and 125 support channel attached I/O devices, or magnetic tape units, or teleprocessing, RMSR support must be generated during system generation. (RMSR support records until checks on SYSREC.) RMSR support can be generated by either the parameter CHAN=YES, or RMS=YES, in the SUPVR system generation macros.

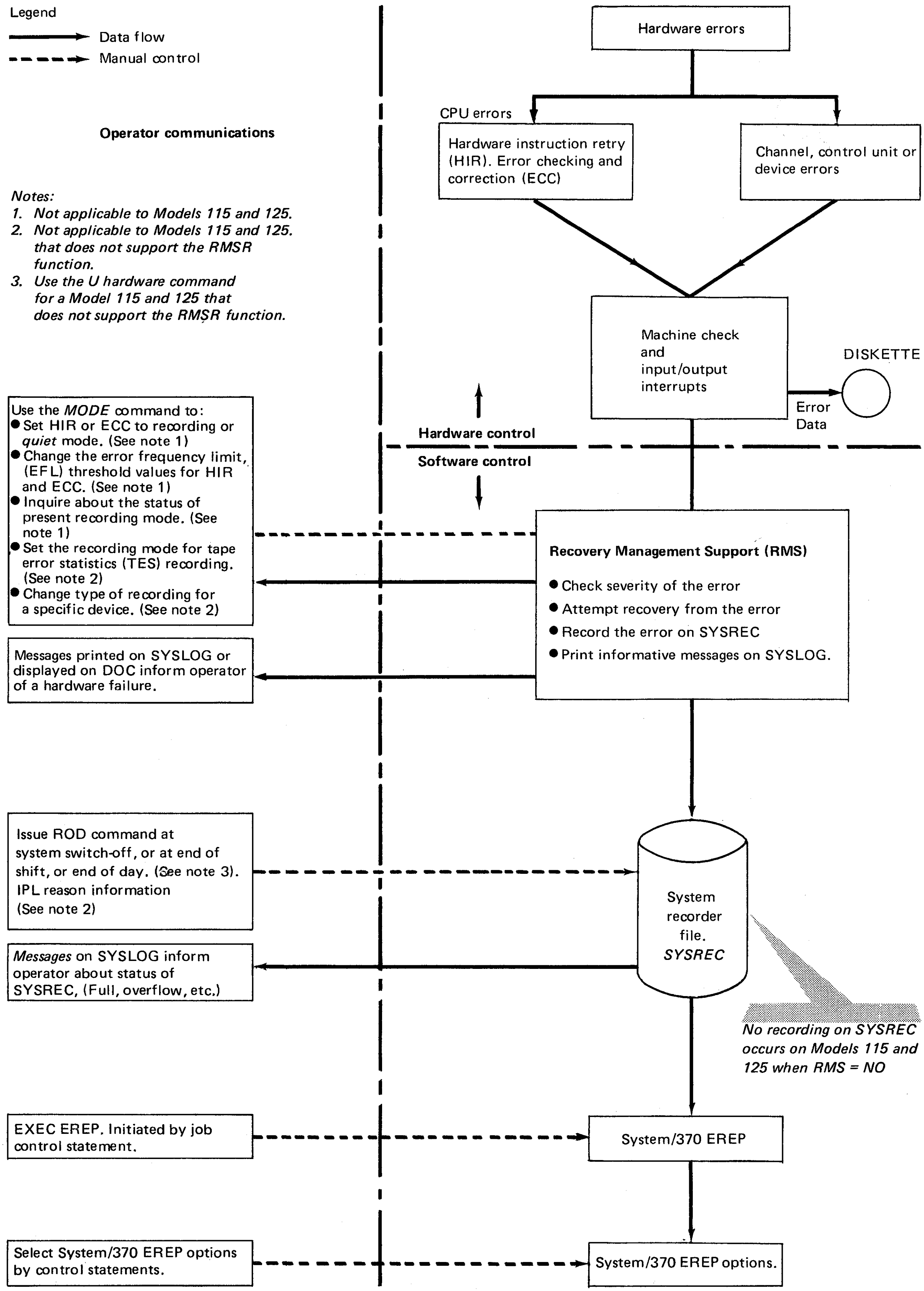
When RMS=YES: hardware failures that occur on all attached I/O devices are recorded on SYSREC by the RMSR software routines. However no error recording occurs for the Multifunction Card Machine (MFCM) if attached. Simultaneously the failures that occur on natively attached devices are also recorded on the DISKETTE by a hardware function. In the latter case of RMS=YES, MCAR/CCH records are recorded on SYSREC as well as on the DISKETTE, and the RDE facility is also supported.

When RMS=NO and CHAN=YES: the supervisor generated supports RMSR for channel attached devices, tape units, TP and MCH/CCH. Therefore hardware failures that occur on these devices are recorded on SYSREC by the software routines as well as being recorded on DISKETTE by the hardware recording function. In this case however RDE is not supported.

When RMS=NO, CHAN=NO and MCH=NO: no recording occurs on SYSREC and a hard wait is entered on the occurrence of a hardware failure

RAS = RELIABILITY
AVAILABILITY
SERVICEABILITY

F-1



RMS

Operation

An understanding of the purpose and operation of RMS will help when interpreting the EREP printout and the System/370 Models 115 and 125 Maintenance Log Analysis Feature.

The following four figures show the relationship between the hardware and software recording facilities, and show the connection between the DISKETTE (Models 115 and 125 only), the SYSREC file, and the EREP options.

Figure F-1-A shows the types of machine checks generated and the real storage used for the logout areas. Error information is first logged in this area before being used by the RMS software routines. On the System/370 Models 115 and 125, the logout area is replaced by the DISKETTE recording file.

Figure F-1-B describes the division of machine check interrupts into soft machine checks and Figure F-1-C illustrates the general flow of processing after a hard machine check occurs.

Figure F-1-D expands the RMS routines into:
- MCAR
- CCH
- Channel check ERPs that are initiated by CCH routines for device-dependent, channel error recovery.
- Unit check ERPs that handle the unit check conditions of the devices.

This figure also shows how the errors are first checked for their severity, and shows how the effect on system operation depends on the type and severity of the error.

Figure F-1-E shows the types of records that are recorded on SYSREC.

Figure F-1-F represents the EREP options that can be selected by the operator.

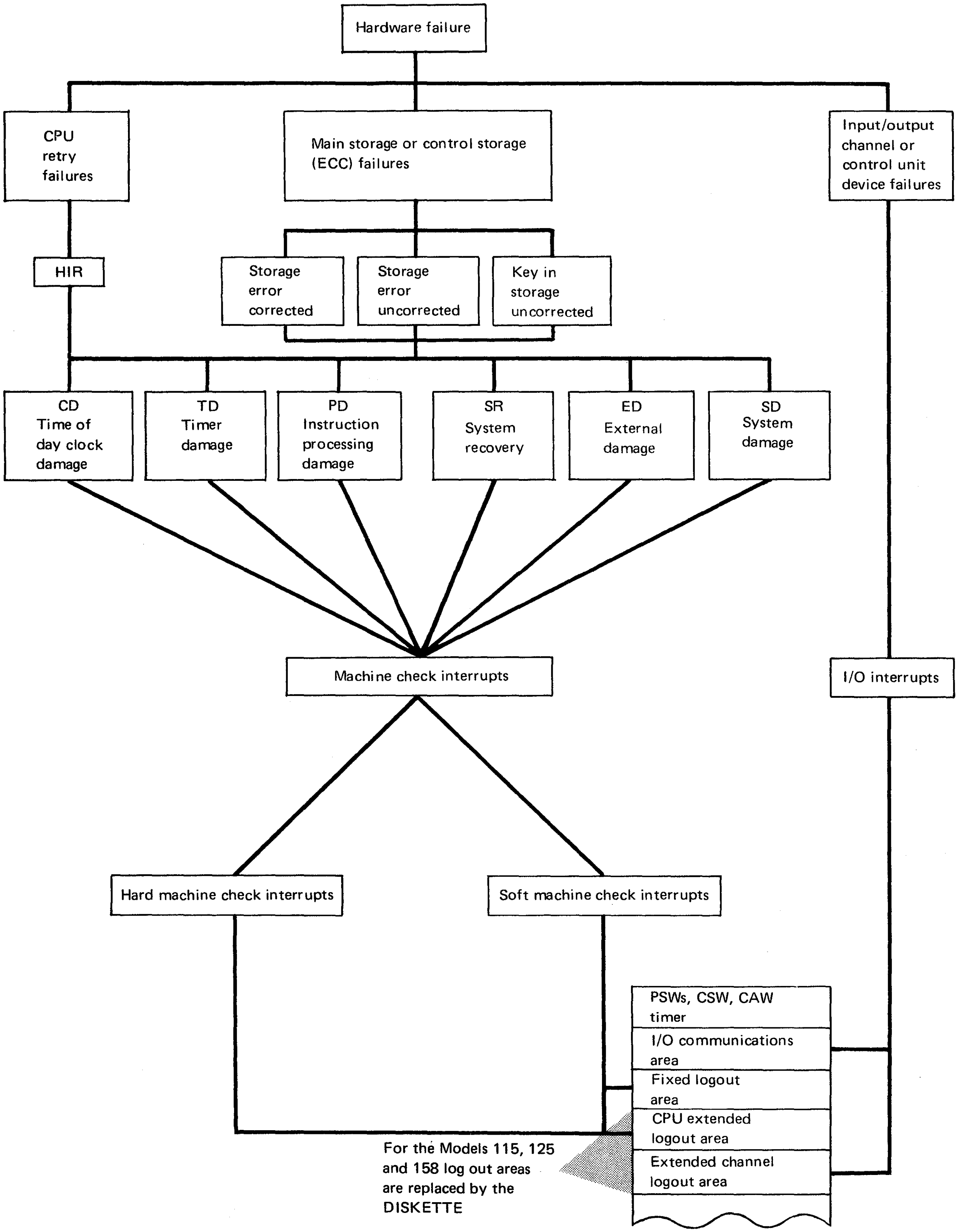


Figure F-1-A illustrates how data about a hardware error is logged in fixed areas of real storage, or on the DISKETTE.
This data is used by soft ware routines for error recovery (where possible), and for recording the data on SYSREC.

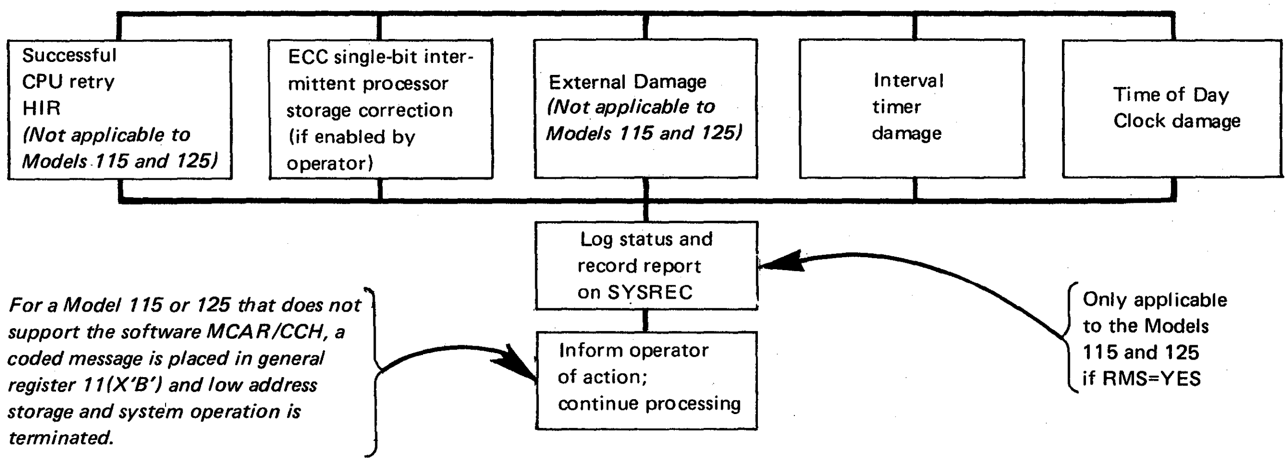


Figure F-1-B. General flow of DOS/VS MCAR processing after soft machine check interrupts

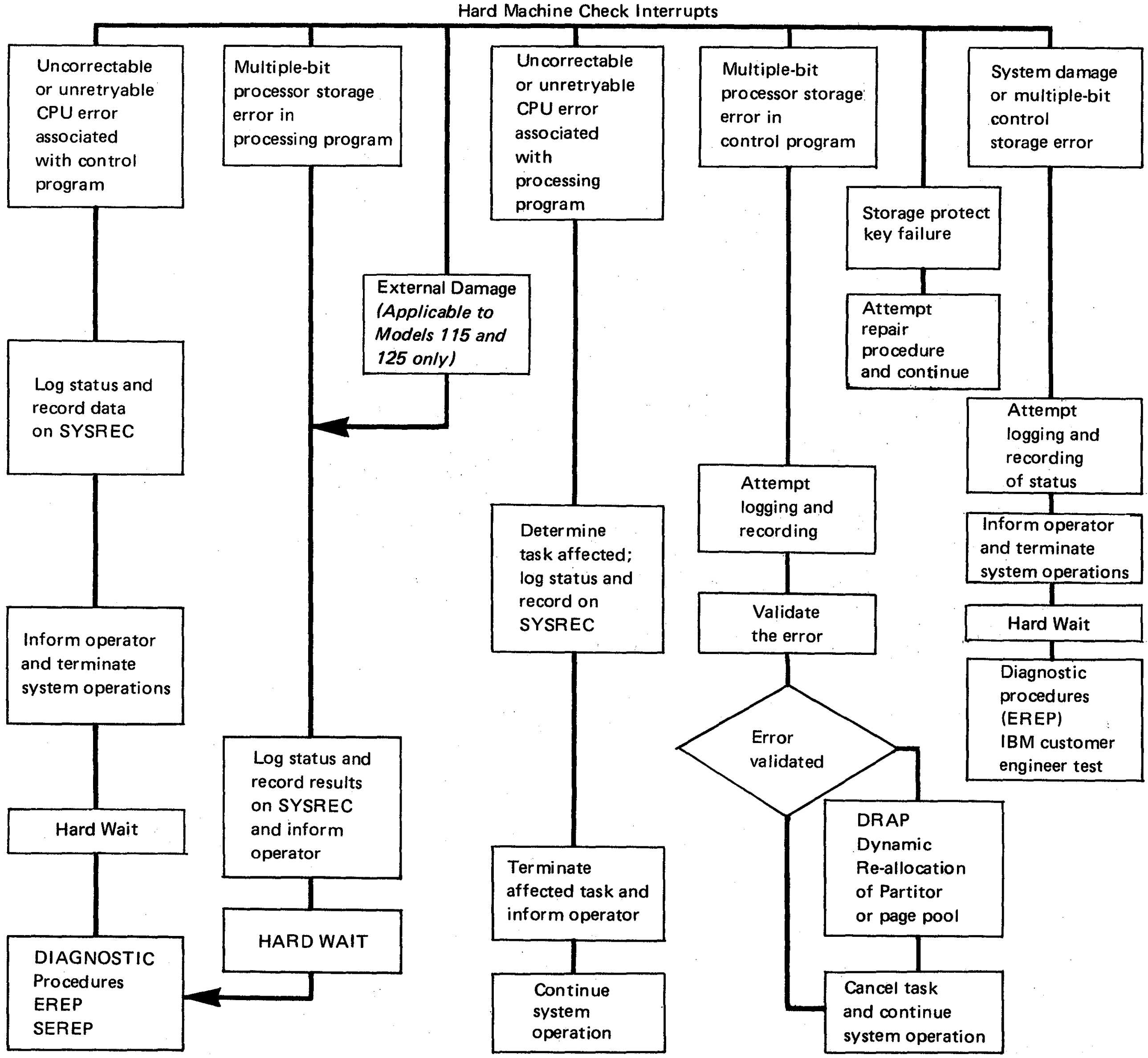


Figure F-1-C. General flow of DOS/VS MCAR processing after hard machine check interrupts

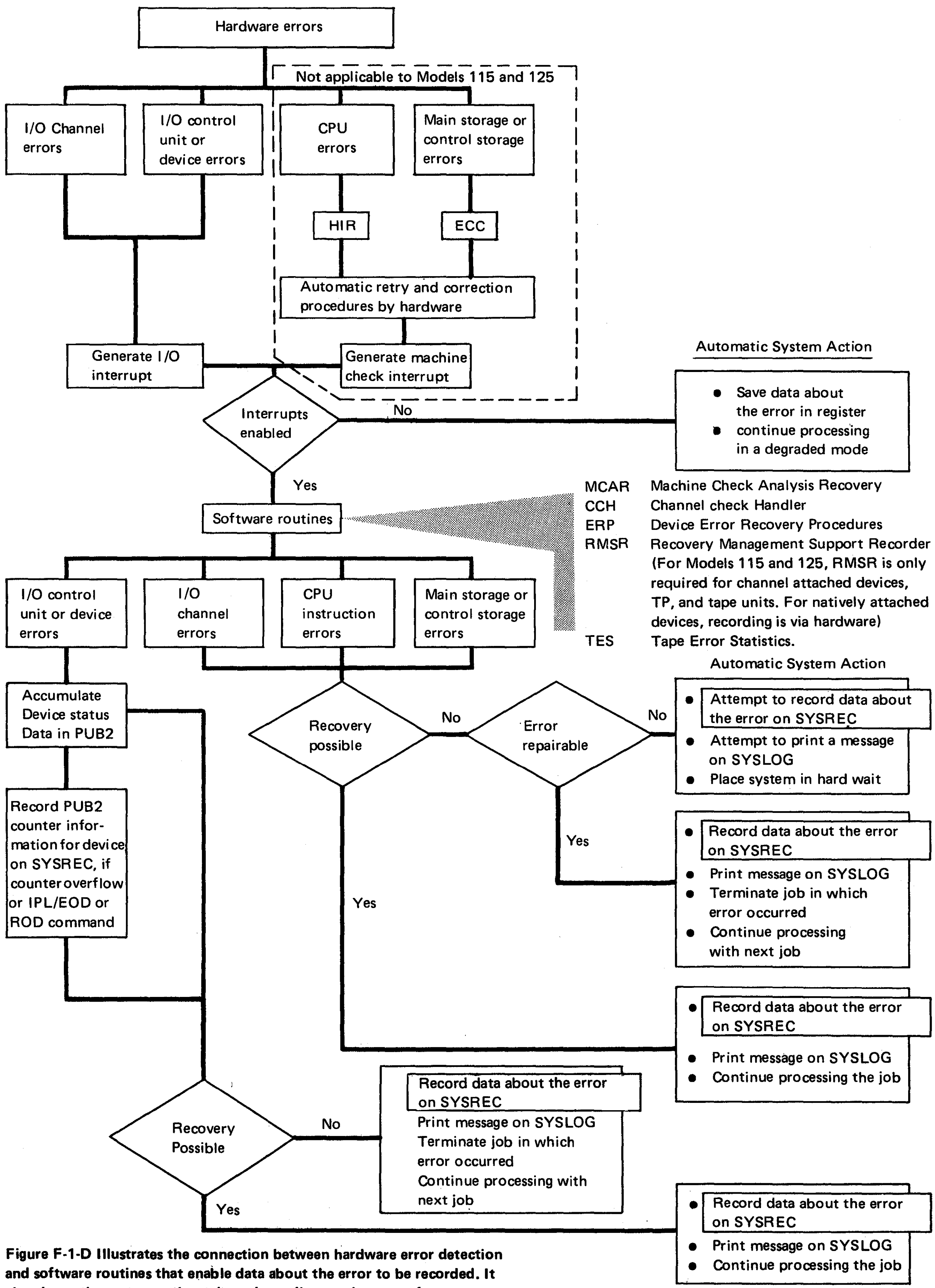
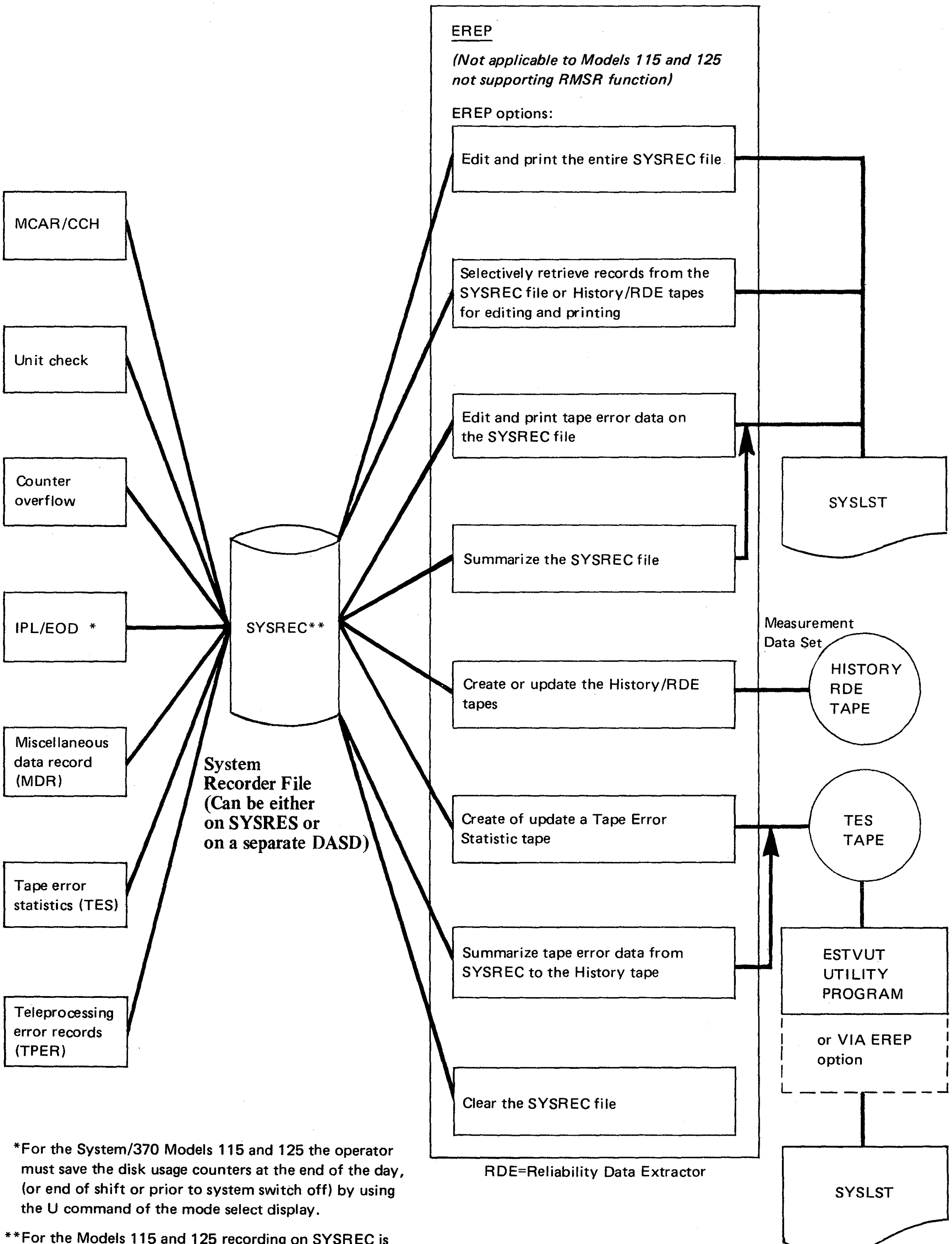


Figure F-1-D Illustrates the connection between hardware error detection and software routines that enable data about the error to be recorded. It also shows the system action taken, depending on the type of error.



*For the System/370 Models 115 and 125 the operator must save the disk usage counters at the end of the day, (or end of shift or prior to system switch off) by using the U command of the mode select display.

**For the Models 115 and 125 recording on SYSREC is applicable only when the RMSR function is generated.

Figure F-1-E Input records to SYSREC.

Figure F-1-F Output from SYSREC. (Selected by EREP options.)

Components

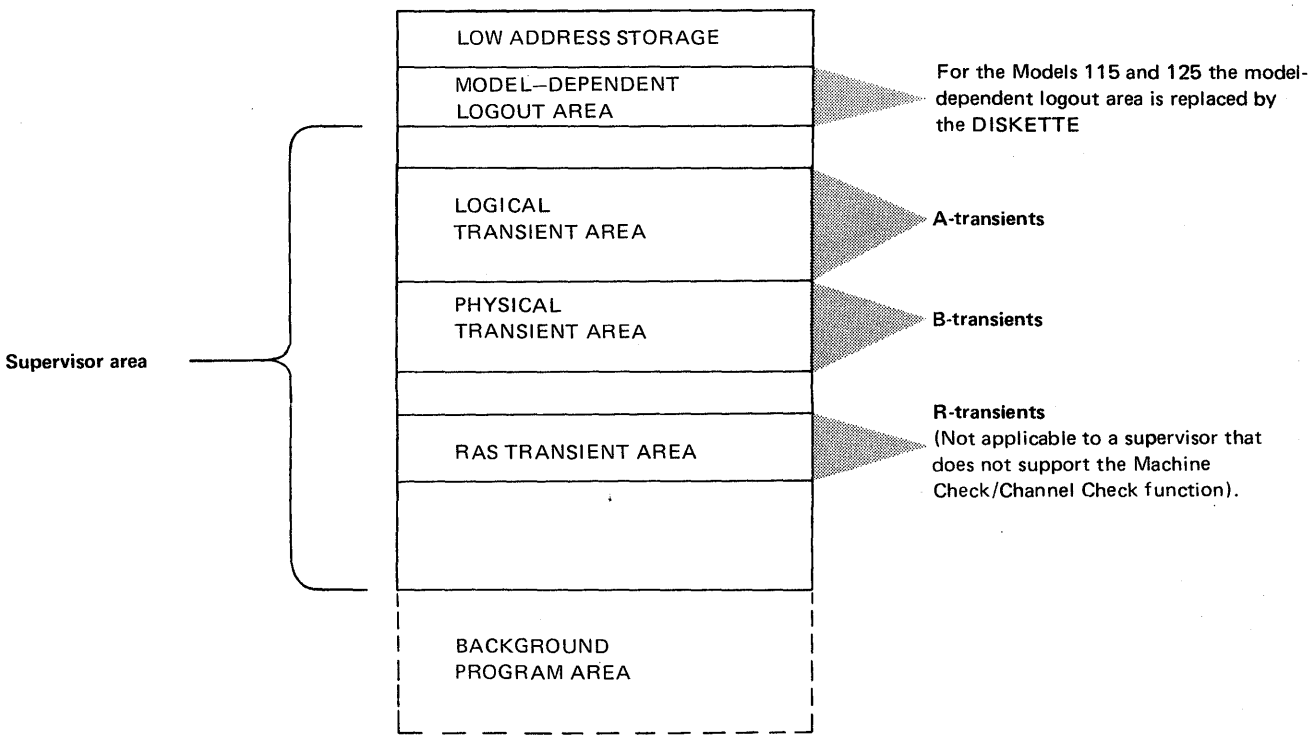
The software routines required to support the RMS options are:

- Resident MCH
- Resident CCH
- Resident RAS monitor
- Resident DASD ERP
- Device ERP transients (A transients)
- MCH/CCH transients (R transients)
- RMSR transients (A and R transients).

To record tape and disk error statistics by volume, the option RMSR also uses some B transients. A job control module is required to enable RMSR to record IPL/EOD data.

The figures below shows the relative locations, in the real address area, of the RTA (RAS transient area), LTA (Logical transient area), and PTA (Physical transient area).

Note: No error recording or recovery can be executed before IPL is successful.



RMS

Detailed Description of RMS Functions

Machine Check Analysis and Recovery (MCAR)

Two hardware error recovery features, Hardware Instruction Retry (HIR) and Error Checking and Correction (ECC), perform hardware correction for machine malfunctions. RMSR interfaces with the error correction hardware through Machine Check Analysis and Recovery (MCAR).

Note: Not applicable to the Models 115 and 125

When the CPU is in the 'recording mode,' MCAR is informed when a machine malfunction occurs and is corrected by means of a 'soft' (or recovered) Machine Check Interrupt (MCI). When the CPU is in 'quiet mode' the hardware correction routines do not generate a soft MCI if the malfunction is corrected. If the hardware correction routines cannot correct the malfunction, a 'hard' (or unrecoverable) MCI is generated regardless of the mode setting.

When a soft MCI occurs, RMSR writes a record in the recorder file containing identification information and the contents of the machine-independent logout area and the machine-dependent logout area (if available). The operator is notified that a soft MCI occurred, control is returned to the interrupted code, and system operation continues.

Dynamic Reallocation of Partition or page pool (DRAP): When a hard MCI occurs, the MCAR routine attempts to isolate the failure to a partition in order to cancel the damaged task and, if possible, continue system operation. If the system cannot continue because the failure occurred in an area critical to system operation, recording is attempted, after which the system enters a hard wait state.

When a hard MCI is caused by an unrepairable real storage position, MCAR dynamically reallocates storage. Informative messages are printed on SYSLOG that alert the operator of any action taken by DRAP.

Note: There is no need for software control or EFL for HIR on the Models 115 and 125

MCAR Modes of Operation: An error Frequency Limit (EFL) algorithm prevents SYSREC from filling up too quickly if a large number of intermittent failures occur. The initial IBM-supplied EFL and either eight (Models 155-11 and 158) or sixteen (Model 145; eight for control storage and eight for processor storage) soft ECC MCIs within an eight-hour period. These values are set at system generation time and can be changed by the MODE command. A message is issued on the first occurrence of a soft MCI on a System/370 Model 135 and all recoverable machine checks are disabled. The MODE command must be issued to re-enable reporting of soft machine checks. These values are set at system generation time but can be changed by the MODE command.

MCAR supports EFL for two hardware facilities:
- Hardware Instruction Retry
- Error Checking and Correction.

EFL Threshold Values: At IPL time, the EFL threshold values are established so that the EFL algorithm controls the number of soft MCIs recorded. These values are:
- The number of soft MCIs
- A specific time period.

When these EFL values are reached, a change in mode of operation occurs. Until the EFL threshold values are reached, the system operates in recording mode. This is the normal mode of operation in which an MCI occurs for all machine check conditions. After the EFL threshold values are reached, ECC (or ECC and HIR) is placed in quiet mode. In quiet mode, no MCIs occur for recovered errors; therefore, the number of corrected errors is unknown.

EFL threshold values are not applicable to the Models 115 and 125 owing to the recording of erros by a hardware function.

Hardware Instruction Retry (HIR) Modes: The two HIR modes are:
- Recording. A soft MCI occurs for every hardware instruction correction.
- Quiet. No soft MCI occurs for hardware instruction correction. (Recording is always quiet for the Model 125.)

Error Checking and Correction (ECC) Modes: The ECC modes are:
- Recording. A soft MCI occurs for every main or control storage correction.
- Quiet. No soft MCI occurs for real or control storage correction. (Recording is always quiet for the Model 125.)
- Threshold (Model 145 only). A soft MCI occurs after a predetermined number of unrecorded control storage errors have occurred within a given time period. Threshold mode is a hardware function and is not affected by EFL threshold values.

If HIR is in quiet mode, ECC is also in quiet mode. When ECC is in quiet mode, HIR can still be in recording mode.

At IPL time the system assumes the IBM-supplied EFL threshold values; these values can be changed by the MODE command. When IPL is completed:
- For the Model 145, recording mode is entered for HIR, quiet mode is entered for main storage ECC, and threshold mode is entered for control storage ECC.
- For the Models 155-II and 158, full recording mode is entered for both HIR and ECC

Channel Check Handler (CCH)

When a channel check occurs, channel error information is logged and an interrupt is generated. The CCH resident program investigates the severity of the malfunction. If the severity is such that system operation need not be immediately terminated, the CCH resident program:
- Builds the Error Recovery Program Interface Bytes (ERPIB) containing error information for use by the appropriate CCH/ERP
- Records the error information on the recorder file
- Attempts to isolate the error to a device.

If the error cannot be isolated to a device, CCH cancels all problem programs that use the malfunctioning channel. If the error can be isolated to a device and the device is supported by a CCH/ERP, the appropriate CCH/ERP is loaded into the R-transient Area (RTA). Then ERP examines the ERPIB supplied by CCH and determines the severity of the error. Whenever possible, the failing channel command is retried. If the command cannot be retried, or if retry fails, a message is written on SYSLOG, and all problem programs using the failing device are cancelled. If recovery is successful, a message is also written on SYSLOG, unless SYSLOG was the failing device. Certain retry conditions require manual operator intervention to enable proper retry.

Note: If the 'accept unrecoverable error' bit in the CCB is on, the error is posted and control is returned to the problem program.

If no CCH/ERP is available for an error isolated to a device, all problem programs using that device are cancelled.

RMS

Error Recovery Procedures (ERP) for I/O Devices:

Each I/O device or class of I/O devices has a unique device error recovery routine. The appropriate routine is entered from the channel scheduler upon detection of an error. The function of the error recovery procedures (ERP) is to attempt recovery from the error either through programmed recovery or by operator intervention. If recovery is not possible, the following choices are available, where applicable:
1. The error can be ignored.
2. The task can be terminated.
3. The problem program can take action (exit to a user routine).
4. The record in error can be bypassed.

Depending on the type of error, the type of device, and whether Logical IOCS is used, some or all of these options are available. Choices 3 and 4 are available through LIOCS only. In the absence of any other options, only choice 2 is available.

At the time the error is first detected, before ERP is called to attempt recovery, RMSR accumulates certain information relating to the status of the device in the PUB2 for the device. The device ERP then gets control and tries to correct the error. If the ERP cannot recover, RMSR builds and writes the unit check record, containing the statistical data from the PUB2 and the status and sense information at the time the ERP determined the error was unrecoverable. If the ERP recovers, the statistical data in the PUB2 is not cleared. This information is recorded at the next permanent error for that device, at the next statistical counter overflow for that device, or at end-of-day when the operator issues the ROD command.

Besides the unit check record (written for every permanent error) and the counter overflow record (written when a statistical counter becomes full or when the operator issues the ROD command), RMSR also writes Tape Volume Dismounts records. The data recorded in the Tape Volume Dismount records corresponds to the data formerly accumulated in the TEBV table; the EREP TES (Tape Error Statistics) options are used to format and summarize this data.

SYSREC (System Recorder File)

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.

The recorder file must be created and assigned to a disk device that is always on line. It is assigned after IPL, before the first job.

The recorder file is not CPU or SYSREC dependent. Thus it can contain records from more than one DOS/VS system.

Once the file is created, no further operator intervention is required, unless the recorder file is damaged or the operator action listed in the *DOS/VS Messages* manual specifically requests the file to be re-created. For example, message

0T03I ERROR ON RECORDER FILE.

On subsequent IPLs the system opens the recorder file and continues to update it.

Note: Recording on the recorder file is suppressed during execution of the EREP program.

Creating the Recorder File: The method of creating SYSREC and the job stream needed depends on whether the file is to be part of the system residence unit SYSREC, or whether it is to reside on a separate disk volume.

For details and job stream examples, refer to the *DOS/VS System Management Guide.*

SYSREC Record Types: SYSREC contains variable-length records with a maximum size of 200 bytes (including a standard 24-byte header). The types of recording that RMSR performs are:

- MCAR recordings
- CCH recordings
- Unit check recordings
- Counter overflow recordings
- Tape volume statistics recordings
- IPL/EOD recordings
- Miscellaneous Data Recorder (MDR) recordings
- Teleprocessing error records (TPER).

MCAR: Formats an environment record (recovery report) after a soft machine check.

The record is written on the environment recorder data set (ERDS), which has the symbolic name SYSREC. The record contains the following pertinent information about the error:

- Status information from the fixed logout area in real storage
- Recovery action
- Program identification
- Date
- Time of day.

CCH: Formats an error information block for use by the ERP routines after an I/O interrupt, caused by a channel check.

The record is written on SYSREC, and contains the following information:

- Status information from the logout area
- The ECSW (extended channel status word)
- Date
- Time of day.

MCAR also records data on SYSREC about hard machine checks.

RMS

UNIT CHECK RECORD: Device ERPs attempt recovery from an error, usually by retrying the failing channel command. If the error is not corrected after a certain number of retries, RMSR writes a unit check record which contains hardware information (sense data), statistical data, and identification data. All information relevant to the status of the device at the time the failure is recognized as permanent is contained in this record. One unit check record is written for each permanent error. RMSR resets the statistical counters in the PUB2 table at the same time.

COUNTER OVERFLOW RECORD: Whenever a statistical counter in the PUB2 table fills up, a counter overflow record is written on SYSREC. The counter overflow record is also written for each device that has unrecorded statistics when the operator issues the ROD command. The statistical counters in the PUB2 table for the device are cleared at the same time.

IPL/EOD: I/O error logging for System/370 users includes RDE (Reliability Data Extractor). If ERRLOG=RDE is specified during system generation, RDE gathers hardware reliability data that IBM personnel use to evaluate hardware performance. Two types of records are written on SYSREC by RDE:

- An IPL record. This specifies the reason for IPL.
- An EOD (End-of-Day) record. This is initiated by the ROD command, which should be issued before the system is shut down.

EREP uses these records to identify RDE data.

For the System/370 Models 115 and 125 the operator must save the disk usage counters at the end of the day (or end of shift or prior to system switch-off) by using the U command of the mode select display.

MISCELLANEOUS DATA RECORDER (MDR): RMSR makes recordings on the SYSREC file for the 3211 printer buffer errors and the 3330 and 3340 Disk Storage errors.

TAPE VOLUME DISMOUNT RECORD: When processing standard labelled tapes using LIOCS, RMSR makes a recording on SYSREC each time a new volume serial number is detected. When the tape is opened, the number of the current tape is compared with the serial number in the PUB2 for that tape drive. If the serial numbers are different, a volume dismount record, containing volume usage and Tape ERP recovery statistics, is written on SYSREC. The statistical counters in the PUB2 relating to usage and error recovery action are cleared and the serial number is updated. Processing continues and statistical data for the new tape is accumulated in the PUB2 table.

TES (Tape Error Statistics)

A major factor affecting the quality of an operating system is the condition of the volume stored on a magnetic medium, such as tape. Such a medium is subject to contamination from dust, foreign materials, fingerprints, and particles of oxide coating.

Because of these environmental factors, it is desirable to record the number of read and write errors occurring on each tape volume. By monitoring the error rate, a report can be kept on the condition of each tape volume in a tape library.

System Requirements: For Tape Cartridge Readers. When error statistics are required to monitor tape cartridges used on the 2495 Tape Cartridge Reader specify TEB = n in the FOPT macro. (n specifies the number of tape cartridge readers attached to the system.)

For magnetic tape volumes when error statistics are required to monitor tape volumes, specify TEVB=IR or CR in the FOPT macro.

For all standard labeled tapes, tape statistics are accumulated by volume. For unlabeled and nonstandard labeled tapes two types of error recording are available:

- Combined Recording (CR)
- Individual Recording (IR).

When TEBV=CR is specified, the error statistics for all unlabeled and nonstandard labeled tapes on a specific tape unit are accumulated until a labeled tape is mounted and opened on that unit. Then, one recording for the unlabeled and nonstandard labeled tapes is made, and the counters are reset in the PUB2 table.

Specify TEBV=IR to record tape error statistics on the SYSREC file and to reset the PUB2 table counters at each OPEN for unlabeled and nonstandard labeled tapes.

The mode of recording for nonstandard labeled and unlabeled tapes can be changed with the tape options of the operator's MODE command.

RMS

EVA (Error Volume Analysis)

This option of RMSR enables you to specify the number of temporary read/write errors that can occur on a tape volume before an informatory message is printed on SYSLOG.

Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function

System Requirements: The number of temporary read/write errors needed to print the informatory message must be specified by EVA=r and/or w the FOPT macro during system generation. (r specifies the threshold level of temporary read errors, and w specifies the threshold level of temporary write errors.)

Description and operation: The message that EVA issues to SYSLOG contains the number of temporary read errors, temporary write errors, and START I/Os, the physical unit identification, and if standard labeled tape is used, the volume serial number.

The message format is:

4E10I xxxxxx cuu TR-nnn TW-nnn SIO-nnnnn

where:
xxxxxx Serial number of standard label volume (blank when nonstandard or unlabeled volume is being used)
cuu Channel/unit address (physical unit)
TR-nnn Number of temporary read errors
TW-nnn Number of temporary write errors
SIO-nnnn Number of START I/Os.

Either the TR=nnn or the TW=nnn field contains one or more than the predetermined error threshold specified in the FOPT macro. When the threshold is reached, a notification, for example, 0P11, is sent to the system operator. There is no interruption in the execution of the problem program.

How and when to use: When using an unlabeled or nonstandard labeled tape, a note can be made of the volume identification of the volume in use when the message is received in order to monitor it.

By monitoring your magnetic tape volumes, a record can be accumulated of read/write errors per volume.

Operational delays caused by old or worn tapes can be avoided by cleaning and erasing the volume, or by cutting off the first ten yards of the volume that indicates read/write errors.

Note: The first part of a tape volume contains label information and is the part of the tape that suffers more from mechanical friction. Therefore, the oxide coating is more likely to cause read/write errors on the first part of a tape than on any other part.

Operator commands for controlling RMS

The error recording facility is under the control of the operator. In addition to creating the recorder file (SET RF=CREATE) and responding to error messages, the operator has the following responsibilities:

- Matching PUB2 space to devices attached
- Issuing the ROD command in response to the problem determination action of an error message, or prior to turning the system off or performing a re-PIL
- Providing IPL reason information (RDE users only)
- Issuing the MODE command to set the type of recording accomplished by the MCAR/CCH, CE, and tape error statistics portions of RMSR
- Executing the EREP program and directing EREP to perform the correct function.

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.

The following sections describe these items more fully.

Matching PUB2 space to devices attached to the system

During IPL, the following message may be issued:

0I29I INSUFFICIENT PUB2 SPACE AVAILABLE, RE-IPL

IPL is automatically canceled, and during the re-IPL the operator must delete devices until the above message is no longer issued.

The reason for this message may be a change in system configuration since supervisor generation, or it may be that the supervisor in use has not been generated for the same amount and type of disk and tape devices. PUB2 (a table in the supervisor) contains a counter for statistical data on device operation and is used by the RMSR routines.

Parameters of the IOTAB supervisor generation macro increase the size of PUB2 to accommodate the counters for devices attached to your system that require a larger field than the standard 12-byte PUB2 field.

The PUB2 table is described in more detail in the *DOS/VS Supervisor* manual.

IPL/EOD (End-of-Day) recording

This RMSR facility enables data to be recorded on the system recorder file (SYSREC) about the reason for, and time between, operator IPLs.

This allows IBM and installation management to monitor IPLs for any selected time period, for example, during an 8 hour operator shift.

When RDE is required, specify ERRLOG=RDE in the SUPVR macro during system generation.

The ROD command (Record on Demand)

Using this command will ensure that any statistical data held in the PUB2 table is added to the recorder file. For System/370 RDE users, the ROD command also writes the EOD (End-of-Day) record on SYSREC. The command ROD has no operand. BTAM and QTAM use their own separate methods of updating all disk counters during closedown or cancel.

For the System/370 Models 115 and 125, the operator must save the disk usage counters at the end of the day (or end of shift or before system switch-off) by using the U command of the mode select display.

RMS

When to use:
1. Operator actions listed under appropriate messages in *DOS/VS Messages* indicate when to issue this command.
2. In order to create meaningful END-OF-DAY records on the system recorder file, you must respond with y to the message END-OF-DAY= at system shutdown or at the end of every shift.

(If the END-OF-DAY record is not required, respond with n to the END-OF-DAY= message.)

IPL reason information

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.

During the processing of the first // JOB statement after IPL, RDE users must provide additional information about the system. Message 1I90D IPL REASON CODE= is issued on SYSLOG. You must respond to message 1I90D with a Reason Code followed by End.

If a Reason Code is not entered (only the END key is pressed) or if SYSLOG is down or not assigned to a 3210, a 3215 or a Model 125 video display unit, then the default, DF, is assumed. However, if an invalid code is entered, message 1I89I is issued and message 1I90D is reissued until a valid response is made.

After the Reason Code is entered, message 1I91D SUB-SYSTEM ID= is issued. You must respond to message 1I91D with one of the ID codes followed by END. The ID codes further identify the reason for performing IPL. The ID codes and the reasons are shown in the table below.

IPL REASON CODE		SUB-SYSTEM ID	
CE	IBM CE/SE has control of the system and is not doing user work.	00	Unknown. Must be used with Reason Codes DF, EN, NM, OP, UN and UP. 00 is the default.
DF	Default.	10	Processor. CPU, channel (integrated), storage unit, etc. failure.
EN	Environmental problem (such as: power, overheating, etc.) caused failure.	20	DASD. A failure occurred in a DASD unit or its associated control unit (2311, 2314, 2319, 2841, 3330/3333, etc.)
IE	IBM hardware or a IBM-supplied-program error that did not require an IBM CE/SE.	30	Other. A device without an ID code (such as a paper tape unit) caused the failure.
IM	IBM hardware of IBM-supplied-program error that required an IBM CE/SE.	40	Magnetic Tape. A failure occurred in a magnetic tape unit or its associated control unit (2400, 3400, etc.)
ME	Media. Hardware error caused by a faulty disk pack, reel of tape, cards, etc.	50	A failure occurred in a card reader/punch, a printer or in its associated control unit (2540, 3525, 1403, 2821, etc.)
NM	Normal IPL.	60	MICR/OCR. A magnetic ink character reader (1412, 1419, etc.) or an optical character reader (1285, 1287, 1288, etc.) failure.
OP	Operational problem. Operator error or procedural problem.	70	A teleprocessing failure occurred in a teleprocessing control unit (2701, 2702, 3705, etc.)
UN	Unknown. Undetermined error.	80	Graphic. A video display unit (2260, etc.) or its associated control unit failure.
UP	A user (non-IBM-supplied) program caused the failure.	90	An IBM-supplied SCP Type 1 or Type 2 program (such as the DOS/VS system or one of its components) failure.
		91	An IBM Programming Product failure.

Table F-2-A IPL reason codes

If the ID code is not entered (only the END/ENTER key is pressed) or if SYSLOG is down or not assigned to a 3210, a 3215 or a Model 125 video display unit then the default, 00, is assumed. However, if an invalid ID code is specified, message 1I89I is issued and message 1I91D is repeated until a valid response is made.

Notes:
1. *Always use ID code 00 with Reason Codes DF, EN, NM, OP, UN, and UP.*
2. *ID codes 10, 20, 30, 40, 50, 60, 70, 80, 90, and 91 should be used with Reason Codes CE, IE, IM, and ME.*

Normal processing continues after the IPL information has been specified. In order to create meaningful data on average running time per IPL, you must issue the ROD command before the system is shut down at the end of the working day, or at the end of a shift. By this procedure, an accurate record can be maintained on the system recorder file, which is printed periodically using an EREP option explained later in this section.

The MODE command

This command should be used only at the request of your IBM customer engineer.

```
        { IR
        { CR
        { CE,cuu  [ ,I [,xx,y] ]
        {         [ ,D [,xx,y] ]
        {         [ ,N         ]
MODE    {
        { R
        { STATUS      *                              *
        { HIR [ ,{M} ] { [ ,{R } ]                   }
        { ECC [  {C} ] { [  {Q } ] [,E=eeee] [,T=tttt] }
        {              { [  {TH} ]                   }
```

Not applicable to the Model 125 (R, STATUS, HIR, ECC)

**Note: When either HIR or ECC is specified, at least one of the optional operands within these braces must be selected. TH is only valid for the Model 145 when ECC,C is specified with the MODE command.*

F-2

The mode command provides the following options for controlling RMSR:

- Reset the recording mode for unlabeled and nonstandard tapes.
- Set recording mode for a particular device to intensive, diagnostic, or no mode.

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.

- Initiate or suppress HIR (Hardware Instruction Retry) and ECC (Error Correction Code) error recording.
- Request the Mode that the system is operating in (the status of the system).
- Change the mode of operation from Q (quiet) to R (recording) or from R to Q.
- Specify EFL threshold value to override the IBM-supplied value.
- Place the Model 145 Control Storage ECC in threshold mode.

Note: Not applicable to the Models 115 and 125.

The MODE command is a notational command. Operands of the MODE command can be entered in any order and must be continuous with no blanks between or within operands). The STATUS operand cannot have any other operands before or after it.

The total length of the MODE command must not exceed 30 characters.

The table below describes the parameters for the MODE command:

Operand	Description
IR CR	Recording mode for nonstandard labeled and unlabeled tape. Specify Individual Recording (IR) if you wish to record and then reset the tape error statistics at each tape OPEN. Specify Combined Recording (CR) to accumulate all the statistics from nonstandard labeled and unlabeled tape on a specific tape unit until a standard labeled tape is opened. Then one recording of the statistics from all the nonstandard labeled and unlabeled tapes is made on SYSREC, and the statistical counters are reset in the PUB2 table.
CE	The recording mode for a device at physical location X'cuu' may be reset. The possible recording modes are: b Normal. The default, normal, is assumed. I Intensive. Normal recording continues. In addition, the next seven errors of a particular type (xx,y) or the next seven errors of any type (if xx,y is not specified) are recorded. The number of I/O retries required for success is also recorded. D Diagnostic. Normal recording continues. In addition, the next seven errors of a particular type (xx,y) or the next seven errors of any type (if xx,y is not specified) are recorded. The number of I/O retries required for success is also recorded. N No recording. When the recording mode parameter is the last parameter of the MODE command, a check is made to see if all errors are recorded. When in intensive or diagnostic mode, it is possible to check for only one type of error. Indicate the bit to be examined with: (xx,y) where y is the bit (0–7) and xx the byte (0–31) of sense data to be checked.

Parameters for the MODE Command, (part 1 of 2).
Table F-2-B

RMS

Operand	Description
STATUS *Not applicable to Models 115 and 125*	On SYSLOG a report is printed which indicates: ● The type of facility used (HIR,ECC) ● System mode of operation ● Current error count ● Error count threshold ● Current elapsed time ● Time threshold ● Number of buffer pages deleted. The status report formats are: HIR,$\left\{\begin{matrix}R\\Q,\end{matrix}\right\}$,aaaa/eeee,bbbb/tttt **For the Model 135:** ECC,$\left\{\begin{matrix}R\\Q\end{matrix}\right\}$ **For the Model 145:** R M **ECC, Q, C,aaaa/eeee,bbbb/tttt** **For the Models 155-11 and 158:** ECC,$\left\{\begin{matrix}R\\Q,\end{matrix}\right\}$,aaaa/eeee,bbbb/tttt BUF DLT=XXX where: aaaa = Current error count eeee = Error count threshold bbbb = Current elapsed time tttt = Time threshold xxx = Total number of inoperable buffer pages deleted.
HIR *Note applicable to Models 115 and 125*	Hardware Instruction Retry. This operand changes the mode of the HIR facility to R or Q and/or modifies the error count threshold and/or time threshold. *Note: When HIR is placed in quiet mode, ECC also goes into quiet mode.*
ECC *Not applicable to Models 115 and 125*	Error Correction Code. This operand changes the mode of the ECC facility to R or Q, and/or modifies the error count threshold and/or time threshold. ECC,R and ECC,Q are the only valid modes of diagnosis for the Model 135. If ECC is specified for a Model 145, M or C must also be specified. ECC can also place the Model 145 control storage in threshold mode. *Note: Use of the Error Correction Code (ECC) in full recording mode may cause severe system degradation. Thus, the (ECC, M/C,R) operand combination of the mode command should only be used by the customer engineer or at his request.*
R	Recording Mode MODE R – places both HIR and ECC in recording mode. MODE HIR,R – places HIR in recording mode. MODE ECC,M,R (Model 145) – if HIR is already in recording mode, main storage is placed in recording mode. MODE ECC,C,R (Model 145) – if HIR is already in recording mode, control storage is placed in recording mode. MODE ECC, R (Models 155-11 and 158) – if HIR is already in recording mode, it places ECC in recording mode.
Q *Not applicable to Models 115 and 125*	Quiet Mode MODE HIR,Q – places both HIR and ECC in quiet mode. MODE EEC,Q (Model 135, 155-11 and 158) places ECC in quiet mode. MODE ECC,M,Q (Model 145) – places main storage in quiet mode. MODE ECC,C,Q (Model 145) – places control storage in quiet mode.
M or C *Note applicable to Models 115 and 125*	Main or control storage: M or C is only valid for the Model 145. M or C must be specified when ECC is specified for the Model 145. M indicates main storage and C control storage.
TH	Treshold Mode: on the next occurrence of an ECC control storage error, control storage is placed in quiet mode. TH is only valid for the Model 145 if ECC,C is specified. TH places the Model 145 control storage ECC in threshold mode.
E=eeee T=tttt	Values entered for E and T must be within the following decimal ranges: E–8 (initial value) through 9999 (Error Count threshold) T–8 (initial value) through 9999 (Time threshold) The IBM-supplied value is 8. *Note: Whenever HIR is in quiet mode, ECC mode must not be changed.* For the Model 135, the only valid mode commands are: MODE CE,... MODE STATUS MODE ECC,Q MODE ECC,R

Table F-2-B Parameters of the MODE Command, (part 2 of 2).

The EREP program edits and prints error statistics records that have been stored on the recorder file (SYSREC) by RMSR.

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.

System Requirements

Before it can be executed, EREP must be cataloged to the core image library. Check with the person in your installation who is responsible for creating or maintaining the core image library to ensure that the EREP program is cataloged. The link-edit statements for cataloging EREP are in the *DOS/VS System Generation* manual.

The EREP program is a modular, self-relocating program. If the supervisor is batched-job only, however, EREP must be link-edited to the end-of-supervisor address. It can run in a real or virtual partition using standard job control statements. When the environmental data is needed or the SYSREC file becomes full, EREP can be executed from SYSLOG or SYSIPT.

EREP can perform any combination of the following options:
- Edit/print the entire SYSREC file
- Create or update the history/RDE tapes
- Selectively retrieve records from the SYSREC file or history/RDE tapes for editing and printing
- Summarize the SYSREC file
- Create or update a TES history tape
- Edit/print TES data from the SYSREC file
- Summarize TES data from the SYSREC file or history tape
- Clear the SYSREC file.

Tables F-3-A and B show how the options are selected and table F-3-C lists the logical unit assignments required by EREP. Table F-3-D and the text following gives a detailed description of these options. Flowchart F-3-F shows the procedure for executing EREP.

```
        ( EDIT          )
        ( CLEAR         )
        ( SUM-----------)-----------GROUP
OPTION  ( HIST [ [,NEW] [,2] ] )
        (      [ UPNEW       ] )                [ [,NEW] [,PRINT[,SUM[,SUMTAPE]] [,VOL]]        ]
        ( TES-----------)----------- OPTION TES [        [,SUM[,SUMTAPE] [,VOL]          ]        ]
        ( SELECT        )                       [ ,NOTAPE {,PRINT[,SUM[,SUMTAPE]] [,VOL]}        ]
        ( RDESUM        )                       [         {,SUM[,SUMTAPE] [,VOL]        }        ]
```

Table F-3-A. The options for TES (Tape Error Statistics)

```
         ( DISK     )
         ( DISKETTE )       CPU=xxxxxx,yyyy
         ( TAPE     )
         ( MICR/OCR )       where xxxxxx = the CPU volume serial
GROUP=   ( UNITREC  )                      number
         ( TP       )             yyyy   = the CPU model number.
         ( CPU      )
         ( 2715     )
         ( ALL      )
```

Table F-3-B. Parameters for the SUM option.

LOGICAL UNIT	COMMENTS
SYSIPT	Optional
SYSLOG	Required, must be assigned to a 3210, 3215 or a Model 125 video display unit
SYSREC	Required
SYS007 SYS008	Optional; must be assigned to a magnetic tape unit when a TES option is specified.
SYS009	Optional; must be assigned to a magnetic tape unit for history/RDE options.

Table F-3-C. Logical units required by EREP.

OPTION	RESULT
OPTION EDIT	Edits and prints SYSREC onto SYSLST.
OPTION CLEAR	1. Edits and prints SYSREC onto SYSLST 2. Clears SYSREC.
OPTION SUM GROUP= { DISK / DISKETTE / TAPE / MICR/OCR / UNITREC / TP / CPU / 2715 / ALL } CPU=xxxxxx,yyyy	Prints the summarization of SYSREC onto SYSLST. The file is summarized by the hardware group(s) listed in the GROUP parameter. If records from multiple CPUs appear on the SYSREC file, specify the serial number (xxxxxx) and model number (yyyy) of the CPU whose records you wish to have summarized. If CPU data is not supplied, records from all CPUs appearing on the SYSREC file are summed together.
OPTION HIST,NEW[,2]	1. Creates the history/RDE tape on SYS009 2. Clears SYSREC.
OPTION HIST[,2]	1. Updates the history/RDE tape on SYS009 2. Clears SYSREC.
OPTION EDIT followed by OPTION HIST,NEW or OPTION HIST	1. Edits and prints SYSREC onto SYSLST 2. Creates or updates the history/RDE tape on SYS009 3. Clears SYSREC.
OPTION TES,NEW	Creates a TES history tape on SYS007.
OPTION TES	Updates a TES history on SYS007.
OPTION TES,NOTAPE,PRINT	Edits and prints tape error data from SYSREC onto SYSLST. The data is printed in the detail tape unit format.
OPTION TES,PRINT,NEW	1. Creates a TES history tape on SYS007 2. Edits and prints tape error data from SYSREC onto SYSLST in the detail tape unit format.
OPTION TES,PRINT	1. Updates the TES history tape on SYS007 2. Edits and prints tape error data from SYSREC onto SYSLST in the detail tape unit format.
OPTION TES,NOTAPE,SUM	Prints the summarized tape data from SYSREC onto SYSLST in the detail tape unit format.
OPTION TES,NOTAPE,PRINT,SUM	1. Edits and prints the tape error data from SYSREC onto SYSLST in the detail tape unit format. 2. Prints the summarization of the tape data from SYSREC onto SYSLST in the summarized tape unit format.
OPTION TES,SUM,VOL	1. Updates the TES history tape on SYS007 2. Summarizes the tape error data on SYSREC by volume serial number.
OPTION TES,PRINT,VOL	1. Updates the TES history tape on SYS007 2. Edits and prints the tape error data from SYSREC onto SYSLST in the detail volume serial number format.
OPTION TES,PRINT,SUM,SUMTAPE,VOL	1. Updates the TES history tape on SYS007. 2. Edits and prints the tape error data from SYSREC onto SYSLST in the detail volume serial number format 3. Summarizes the tape error data on the history tape and prints it on SYSLST in the summarized volume serial number format.
OPTION TES,NOTAPE,SUM,SUMTAPE	Summarizes the tape error data on the history file and prints it on SYSLST in the summarized tape unit format.
OPTION SELECT (see note 1)	Selectively prints records from SYSREC onto SYSLST.
OPTION SELECT,TAPE (see note 1)	Selectively prints records from the history/RDE tape onto SYSLST.
OPTION RDESUM	Summarizes the IPL, EOD, MCAR, CCH, and Unit Check records for a specified period of from one to 30 days. These records are on the history/RDE tape (see note 2).
(none)	Edits and prints SYSREC onto SYSLST.

Note 1. *Records are selected by specifying select parameters.*

Note 2. *RDESUM does not summarize across multiple volumes. If EOF is encountered before the entire requested reporting period has been covered (this can be checked through the end date printed on the RDESUM listing), rerun RDESUM using the next volume history/RDE file nad the same reporting period you specified during the first RDESUM execution. A listing with the remainder of the requested information is thus generated.*

Table F-3-D. The EREP options

EREP

Description of EREP Options

EDIT

The EDIT option causes EREP to edit and print the contents of the SYSREC file on SYSLST. The unit check records are displayed first and are grouped by CUA (channel and unit address) within each device group (unsupported, tape, disk, TP, unit record, MICR/OCR).

After the unit check records, the channel check, machine check, 2715, and IPL/EOD records are displayed. Retain these printouts for problem determination.

EREP displays IBM 2715 error records from the SYSREC file in this order:
1. Disk adapters
2. 2790 loop adapter
3. MPX adapters
4. 2750 adapters.
5. BSC adapters.

The special code records are grouped for editing and printing by area station address, CUA, and special code. All area station records on SYSREC are summarized by device address, area station, ID, and CUA during editing and printing.

EREP EDIT can execute in a 14K partition, but performance may be improved by allocating more than 14K (up to 42K) to the EREP partition. Storage allocation should be increased in blocks of 4K because the tables that EREP EDIT uses are each 3.5K in size. This applies only when EREP is executed in real mode.

CLEAR: The CLEAR option causes EREP to clear (reset) the entire SYSREC file for RMSR recording. If the CLEAR option is specified by itself, the EDIT option is forced. CLEAR is always the last EREP function performed. CLEAR is forced if HIST, or HIST with optional parameters, is specified.

Note: If a hard I/O error occurs on SYSREC while the CLEAR function of EREP is running, EREP will abnormally end and the operator should re-IPL the system. In a MPS system, it may be undesirable to re-IPL. If you do not re-IPL, however, the contents of the SYSREC file will be unpredictable.

SUM

The SUM option allows hardware groups on SYSRES file to be summarized. This function can:

- Accumulate certain bits and bytes in CPU logouts within MCAR/CCH records
- Accumulate statistical and sense byte data from unit check records
- Summarize area station data in 2715 error records by device address, area station, ID, and channel and unit address.

Note: This option is only applicable to the Model 145.

The SYSREC file may be summarized, or one or more hardware groups may be summarized. The GROUP parameter should immediately follow OPTION SUM.

```
        ⎧ DISK      ⎫
        ⎪ DISKETTE  ⎪
        ⎪ TAPE      ⎪
        ⎪ MICR/OCR  ⎪
GROUP= ⎨ UNITREC   ⎬
        ⎪ TP        ⎪
        ⎪ CPU       ⎪
        ⎪ 2715      ⎪
        ⎩ ALL       ⎭
```

F-3

These entries separated by commas may be made in any order. If the GROUP parameter does not follow OPTION SUM or if it contains an error which the operator does not correct, the EREP program summaries the SYSREC file for the tape hardware group.

If the SYSREC file contains records of multiple CPUs, the CPU whose records are to be summarized must be defined by entering

CPU=xxxxxx,yyyy

in which xxxxxx = the CPU serial number
yyyy = the CPU model number.

If no CPU is provided, records from all CPUs appearing in the SYSREC file are summed together.

You can execute the SUM option more than once during an EREP run if you enter the option and parameter control statements via SYSLOG. After the summary is performed with one set of parameters, the message

3E05A ENTER SUMMARY PARAMETERS

is printed on SYSLOG. You may enter the parameters for another summary at this time, or end execution of the SUM function by responding with CANCEL and pressing the END or ENTER key.
If GROUP=ALL is specified, EREP does not ask for additional parameters because a summary of all records is made.

When the summary has been completed, EREP processes the next option, if any.

It is possible to reduce the processing time for the SUM function by allocating more main storage (in blocks of 8K) to the partition in which EREP is to run. The root phase requires 2K and each transient 8K. The disk, tape, and unit record group use two transients; other hardware groups require only one transient.

When 2715 is specified in the GROUP parameter, the 2715 records are summarized before any other hardware group. The 2715 group uses 10K of storage, even if more storage is available. If not all the 2715 records can be processed in the 10K partition, those that can be are processed, after which the transient is reloaded and the next 2715 records are processed. This is done until all 2715 records are processed.

Note: This applies only to EREP executed in real mode.

In the 8K partition, the TP group can be process records for up to 60 distinct terminal names at one time. If more than 60 terminals are to be summarized, the file must be read more than once. If more than one hardware group is specified in a 10K partition, the transients overlay each other and the file must be read as many times as there are transients.

EREP

SELECT

By means of the specified search parameters, EREP selects records to be printed. The SELECT option initiates the search for these records on SYSREC; for example, SELECT, TAPE causes a search of the history tape to be performed. The parameters of the SELECT option are called select parameters; they are checked for validity but not for logical relationship. For example, although an MCAR record has no VOL field, the parameters

TYPE=MCAR
VOL=123456

are considered valid.

The possible select parameters are listed in the table below:

SELECT PARAMETER	RESULT
CPU=xxxxxx	All error records associated with a CPU may be selected for printing by entering the six digit CPU serial number.
TYPE= {MCAR, CCH, IPL, EOD, TP, UNIT, 1275}	A specific type of error record may be selected for printing. Any number of different types may be selected for each search.
DATE= {yyddd,yyddd / yyddd}	All recordings made within a time span (measured in days) may be selected for printing. If two dates, separated by a comma, are specified, all recordings made in that time span are selected. If only one date is specified, all recordings made on that day are selected for printing.
TIME hhmm,hhmm	All recordings made within a time span (measured in hours and minutes) may be selected for printing.
JOB=xxxxxxxx	All recordings made during the execution of a specific job may be selected for printing by specifying the eight-byte jobname from the job statement.
VOL=xxxxxx	The error records for a specific volume may be selected for printing by entering the six-byte volume serial number.
TERM=xxxxxxxx	The error records for a terminal may be selected by entering the eight-byte terminal name.
CUA=xxxx	Records may be selected for printing by entering the channel and unit address (in hexadecimal) or the line number for TP.
DEVICE=xxxxxx	The records associated with a specific type of device may be selected by entering the device type code (for example, 1403, 1442N1).
FORMAT=TES	Whenever a tape (2400 or 3400-series) error record is encountered, it is printed in the detail TES format by volume serial number. If FORMAT=TES is not specified, all tape error records are printed in the unit check format.
SEL2715= {AREA, ADAPTER, SPECIAL}	The 2715 records are printed in area station format if the SEL2715 parameter is not specified. If printing by area, adapter, or special is required, however, the SEL2715 parameter must be specified.

Table F-3-E. The select parameters.

You may enter any combination of parameters; the EREP program assumes that you will only enter select parameters that apply to the records you want. If no select parameters are specified with the SELECT option, the MCAR records are selected and printed.

The SELECT option can be executed more than once during an EREP run if the option and parameter control statements are entered via SYSLOG. After selective retrieval, when one set of select parameters has been completed, the message

3E03A ENTER SELECT PARAMETERS

is printed on SYSLOG. At this time, you may enter a new set of select parameters to execute the selective retrieval or you may end selective retrieval by responding with CANCEL and pressing the END or ENTER key.

RDESUM

The RDESUM option provides a summary of information about system operation during a specified 1 to 30 day period. This summary is created by searching the history/RDE tape, mounted on SYS009, for IPL, EOD, MCAR, CCH, and unit check records, after which these records are edited and printed on SYSLST. The information provided by the RDESUM option includes:

- The starting and ending dates of the report.
- The date, time, reason, and subsystem responsibility for each IPL.
- The average run time between IPL and EOD (or between two consecutive IPLs if the ROD command was not issued to create an EOD record) for the specified interval. If specified, the number of IPL records that occur in the cluster interval, (see note)
- The subsystem responsibility and number of times a subsystem caused a System Recovery Incident (a recoverable error that may cause system degradation) or a System Incident (an unrecoverable error that caused system failure).
- If the history/RDE tape contains no records within the specified dates, an error message is printed and the report is terminated.
- IPL records are not counted in the reports of sub-systems SI (System Recovery Incidents)
- If an IPL record with a reason code of UN, IE, IM, ME or DF is immediately preceeded on the tape by an SRI that occurred within 30 minutes of the IPL, the SRI may be reclassified as an SI. The SRI is reclassified if (1) the subsystem ID specified for the IPL is the same as the device type of the SRI, or (2) if the subsystem ID is unknown (00).
- Multiple SRIs on the same device are counted as a single SRI until there is a ten minute interval without an incident or an IPL record.
- If an SI occurs within ten minutes of the IPL record following an SI, the SI is counted as a multiple occurrence of the first SI regardless of the subsystem involved. Intervening SRIs are ignored.
- If 16 sequence errors occur on the history/RDE tape, RDESUM is terminated; if fewer than 16 sequence errors occur, the out-of-sequence records are ignored.

Note: Clustering is the process of searching for multiple IPL records that have occurred within a specified number of minutes. Clustering can be used to detect multiple false starts that may distort other information provided by RDESUM.

RDESUM is executed when the appropriate option card is encountered. The control information, including the start date for the report, the end date, the clustering interval if clustering is desired, and the company name, is entered once the EREPRDE phase is in main storage. The control information is entered in response to prompter messages.

RDESUM does not summarize across multiple volumes of a history/RDE file. If EOF is encountered on the input tape, RDESUM goes to EOJ and the report printed reflects the information available from the start date to the last record on the tape. There may be some inaccuracy in the average run time per IPL (because RDESUM does not know when the EOD or next IPL record will occur, it uses the time of the last error record to compute the IPL period), but no other information is lost.

RDESUM can be executed again for the next volume in the history/RDE file to obtain the remainder of the information for the desired reporting period. The previously specified period may be used on the subsequent volume because RDESUM starts with the first record on the tape if the specified start date is earlier than the date of the first record.

The following rules govern the method for summarizing RDE information:

- If the history/RDE tape does not contain information for a portion of the required time period, only those dates on the tape that fall within the time period are processed. The actual dates processed are reflected on the summary listing.
- If the starting date is defaulted, the first record on the tape is used to start the report. The report is stopped with the specified end date or, if that date is more than 30 days from the date of the first record processed, the thirtieth day processed.
- If the end date is defaulted, the report is stopped with the last date on the tape or, if that date is more than 30 days from the starting date, on the thirtieth day processed.

HIST or HIST with Operands

This option copies the data on the SYSREC file to the history/RDE tapes. All records on the tape(s) appear in chronological order. If an unrecoverable I/O error occurs while a record is being read from the SYSREC file, the record is ignored and processing continues with the next sequential record. If the data fills the complete tape, the message

    3E15A TAPE FULL, MOUNT NEW TAPE

is printed on SYSLOG. The operator must mount a new tape and press END to continue processing, or he may respond with CANCEL and press END to cancel the HIST option.

The tape must be mounted on SYS009. SYS009 must be assigned to a tape drive before EREP is executed. The tape contains standard labels that are checked before the history/RDE tape is written. If the wrong tape is mounted, the message

    3E31A WRONG TAPE, MOUNT CORRECT TAPE

is printed on SYSLOG, Mount the correct tape and press END to continue processing, or respond CANCEL END to cancel the HIST option. When the HIST option is specified, the CLEAR option is forced. The SYSREC file is cleared after the history/RDE tape has been created or updated, thus preventing redundant data from being transferred to the history/RDE tape the next time the HIST option is executed.

HIST,NEW,[,2] : This option causes EREP to create a history file on the tape unit assigned to SYS009. If 2 is also specified, a second history file is created on the same tape unit for RDE data. The data contained on both tapes is identical. The tape(s) contain the contents of the SYSREC file. The SYSREC file is cleared after all options have been executed.

HIST,UPNEW: This option causes the tape file mounted on SYS009 (either history or RDE) to be updated, after which a new tape file is created. If UPNEW is specified, TLBL information for creation and updating must be included in the job stream. The SYSREC file is cleared when all options have been executed.

TES or TES with Operands

The TES options provide for the editing and printing of the tape error records on SYSREC and the summarizing of tape data found on either SYSREC or the history file.

To enable this option to be used a work or scratch tape must be mounted on a tape unit assigned to SYS008. This option can also select tape error data from the SYSREC file and create a TES history tape with the same format as the previously supported ESTV tape file. All records on the tape appear in chronological order. If an unrecoverable I/O error occurs while reading a record from the SYSREC file, the record is ignored and processing continues with the next sequential record. If the data fills the complete tape, the message

3E15A TAPE FULL, MOUNT NEW TAPE

is printed on SYSLOG. The operator must mount a new tape and press END, or he may respond CANCEL END; the latter response causes tape updating to be discontinued, but TES records are still printed.

The tape must be mounted on SYS009. SYS009 must be assigned to a tape drive before EREP is executed. The tape contains standard labels that are checked before the history/RDE tape is written. If the wrong tape is mounted, the message

3E31A WRONG TAPE, MOUNT CORRECT TAPE

is printed on SYSLOG. Mount the correct tape and press END to continue processing, or respond CANCEL END to cancel the TES option. The history/RDE tape and TES history tape should be created or updated during the same EREP run. If the HIST option is specified without the TES option, the SYSREC File is cleared after HIST has been executed, and the TES data is lost. If you wish to maintain both these history tapes and the TES and HIST options are not specified together in one EREP run, the data on the TES history file may be redundant or lost.

TES,NEW: This causes EREP to create a TES history file on the tape unit assigned to SYS007. The tape file contains tape error data from the SYSREC file. The tape error data on the tape has the same record format as the previously supported ESTV tape file. Use ESTVUT utility program to print this tape file.
TES: EREP updates the TES history tape on SYS007.
TES,NOTAPE,PRINT: Causes the tape data on SYSREC to be edited and printed into SYSLST. Data is printed in the detail tape unit format.
TES,PRINT,NEW: A new TES history tape is created on SYS007, after which the tape error data on SYSREC is edited and printed on SYSLST. The data is printed in the detail tape unit format.
TES,PRINT: The TES history tape, which is mounted on SYS007, is updated. The tape error data on SYSREC is then edited and printed on SYSLST in the detail tape unit format.
TES,NOTAPE,SUM: The tape error data on SYSREC is summarized by tape drive.
TES,NOTAPE,PRINT,SUM: The tape error data on SYSREC is edited and printed on SYSLST in the detail tape unit format. Then the tape error data on SYSREC is summarized by channel and unit and printed on SYSLST.
TES,SUM,VOL: The TES history tape on SYS007 is updated. Afterwards the tape error data found on SYSREC is summarized by volume serial number.
TES,PRINT,VOL: The TES history tape mounted on SYS007 is updated. The tape error data on SYSREC is edited and printed on SYSLST in the detail volume serial number format. SYS007 is used as a work tape and the detail records are printed in sequence by volume serial number.

Four examples of processing tape error statistics using EREP are given in Appendix O.

EREP History Tapes

There are three types of EREP history tapes: the History tape, the RDE tape, and the TES history tape. The History and RDE tapes are created and updated from the SYSREC file and contain all the record types found on the SYSREC file. The TES history tape is also created from the SYSREC file, but contains only tape error records. If your installation has the History/RDE tapes and a TES history tape, you should create (or update) all the history tapes in the same run. If this procedure is not followed, the TES history tape may have redundant or missing data.

Retain the History and TES history tapes for those persons who work on problem determination. The History tape can be used as input for certain online test programs of OLTEP. (See the OLTEP manual.) The TES history tape can be printed with the ESTVUT utility program. Retain the RDE tape; it will be used by IBM.

History/RDE Tape

The History/RDE tape is created and updated using the EREP history option. This tape contains RDE data only if ERRLOG=RDE is specified at system generation. A magnetic tape unit assigned to SYS007 must be used for this function. EREPNEW must be the filename that is used when a tape is created, and EREPUP when a tape is updated (both TLBL cards must be included for UPNEW). When the tape becomes full or when a second tape must be mounted, the operator is notified via SYSLOG.

Note: If EREP is link-edited as a self-relocating program, an LBLTYP card is needed when EREP builds a history/RDE tape.

TES History Tape

The TES history tape is created and updated using the EREP TES options. A magnetic tape unit assigned to SYS007 must be used for this function. The filename of the tape file must be TAPEIN when the file is created and the file is updated.

Creating the History Tapes

You can create a history tape only if DOS/VS has recorded errors on SYSREC. The EREP program allows you to create or update the three types of history tapes. three types of history tapes.

You can create the History/RDE tape by specifying OPTION HIST, NEW, and update it by specifying OPTION HIST.

If a System/370 RDE tape is to be processed, the message 3E16A is printed on SYSLOG after the History tape is written. This message instructs you to replace the History tape reel with the RDE tape reel and then respond to the message. A response of END will cause the RDE tape to be processed and response of CANCEL END will cancel only the HIST option. Any other response will cause the system to reissue message 3E16A.

In addition, you can create a TES history tape, which contains only tape error records. If you want to maintain a TES history tape, create (or update) it in the same EREP run in which you create (or update) the History/RDE tape. You can create the TES history tape by specifying OPTION TES,TAPE,NEW, and update it by specifying OPTION TES, TAPE.

Processing the tape error statistics with EREP

The EREP (Environmental Recording, Editing, and Printing) program provides processing options for the tape error statistics records on SYSREC.

Tape records can be edited and printed or summarized, together with the order records on SYSREC; you may also choose to have only the tape error records of the file selected or summarized. If the SYSREC file has been used to create a history/RDE tape, the records on that tape contain the same information as the SYSREC file contained. In this case the tape error statistics records can be selected or summarized from the history/RDE tape file.

The SYSREC file may also be used to create a TES history tape. This tape contains tape error statistics records only. These records have the same format as the records of the former ESTV disk file; thus only part of the information recorded on the SYSREC file for tape error statistics is written on the TES history file. The information written on the TES history file consists of:

- Date the record was collected
- Physical address of the device on which the tape volume was mounted
- Number of temporary read errors
- Number of temporary write errors
- Number of permanent read errors
- Number of permanent write errors
- Number of error gaps encountered
- Number of noise blocks encountered
- Number of cleaner actions taken
- Number of SIO instructions issued
- Volume serial number if the tape was a standard labeled volume
- Block length if the volume contained fixed-length blocked records
- Tape density of the tape volume.

The history/RDE tape and the TES history tape must always be updated in the same run. Failure to update both these tapes on the same run may result in redundant or lost data on the TES history tape. When PRINT is specified, the detail records on SYSREC are printed on SYSLST. When SUM is specified, the tape error statistics are summarized on either the history tape or SYSREC.

It is possible to print or summarize tape error statistics by volume serial number or by tape drive address.

When tape error statistics are summarized by volume serial number, it may be possible to reduce processing time by allocating more main storage to the EREP partition. Approximately 90 distinct volumes can be summarized in a 10K partition. When the SYSREC file contains recordings for more than 90 distinct volumes and EREP is run in a 10K partition, the SYSREC file is read and 90 volumes are summarized; then the SYSREC file is processed again and the remaining (or next 90) volumes are summarized.

If you want to reduce processing time when there are more than 90 volumes, therefore, you must allocate enough storage, thus allowing all volumes to be summarized on only one read-through of the SYSREC file. Approximately 12 additional volumes can be processed for each 1K added to the partition. To calculate the number of volumes that can be summarized in a particular partition, use the following formula:

$$N = \frac{P - \begin{array}{c}\text{80 bytes}\\ \text{if tape is}\\ \text{assigned}\end{array} - 2740}{82}$$

Processing the TES History Tape with the ESTVUT Utility Program

When a TES history tape is created from the data on SYSREC, ESTVUT (the ESTV Dump File Program) is used to process the data on the TES history tape. This utility program dumps the TES history file on SYSLST.

ESTVUT consist of one module that has to be cataloged in the core image library. The module name to be used in the INCLUDE statement for this routine is IJBTESUT.

Control Cards necessary to run ESTVUT.

ESTVUT can be executed either from a card reader or from SYSLOG. An example of the job control statements required for ESTVUT is:

```
// JOB ESTVDUMP
// ASSGN SYS005,X'181'
// ASSGN SYSLST,X'00E'
// TLBL TAPEIN
// LBLTYP TAPE
// EXEC ESTVUT
/*
/&
```


Symbolic Unit Assignments: Every sumbolic unit required for execution of the ESTVUT program must be assigned either temporarily for one job, or permanently.

- SYS005 must be assigned to the magnetic tape unit on which the TES history file is mounted.
- SYSLOG must be assigned to a 3210, a 3215 or a Model 125 video display unit for all executions of ESTVUT in order to log inquiries and accept replies.

Label information: Label information must be available to the system whenever the devices are used in the execution of ESTVUT

- The first operand of the TLBL statement for the input tape must be TAPEIN
- A LBLTYP for tape is required if the program uses tape and has been cataloged as a self-relocating program (+0 on the PHASE card). This statement reserves space for processing standard label information.

Contents and format of printed output

When the operator specifies a printer as the output device, the collected error statistics are formatted and printed as illustrated below:

VOLUME SERIAL	DATE	TIME OF DAY	CHANNEL /UNIT	TEMP READ	TEMP WRITE	PERM READ
xxxxxx	yr/day	hr.mn.sc.	cuu	nnn	nnn	nnn
PERM WRITE	NOISE BLOCKS	ERASE GAPS	CLEANER ACTIONS	SIOS USAGE	TAPE DENSITY	BLOCK LENGTH
nnn	nnn	nnn	nnn	nnnn	nnn	nnnn

Each page of output contains 50 lines of data.

On the last page, a message is printed below the last line of data. The message is:

ESTV TAPE FILE DUMPED

EREP

Executing EREP

Execute the EREP program at the request of the customer engineer or in response to an instruction in an error message. The operator commands necessary to execute EREP through either SYSLOG or SYSRDR are:

```
PAUSE BG,EOJ
// TLBL    EREPNEW          (see note 1)
//         EREPUP
// TLBL    TAPEIN           (see note 2)
           TESUP
// ASSGN   SYS007,X'cuu'
// ASSGN   SYS008,X'cuu'    (see note 3)
// LBLTYP TAPE
// EXEC EREP
```

Note 1: This card is necessary only if you want to create or update either a history tape or a history tape and a Model 145 RDE tape. Use EREPNEW when creating and EREPUP when updating.
Note 2: This card is necessary only if you are creating or updating the TES history tape. Use TAPEIN when creating and TESUP when updating.
Note 3: This control card is necessary if you are updating or creating any of the history tapes (History tape, RDE tape, or TES history tape).

Then EREP issues a message to the operator via SYSLOG or SYSIPT that is to be used for entering the EREP options.

```
3E1ID ENTER OPTION SOURCE , C=CARD,
          S=CONSOLE, N=NONE
```

The operator must respond with one of the following:
- C followed by END for SYSIPT
- S followed by END for SYSLOG
- N followed by END for the default option, EDIT.

The default will be N END or just END, and the result will be the editing and printing of the SYSREC file. If the operator response is C END or S END, the system awaits option data on either SYSIPT or SYSLOG. Enter CANCEL END if you wish to cancel the job at this time.

If any response other than C, S, N, CANCEL, or END is entered:

```
3E25I INVALID RESPONSE
```

will appear on SYSLOG and message 3E1ID is reissued.

Entering EREP options

EREP options can be entered through SYSLOG or through SYSIPT.

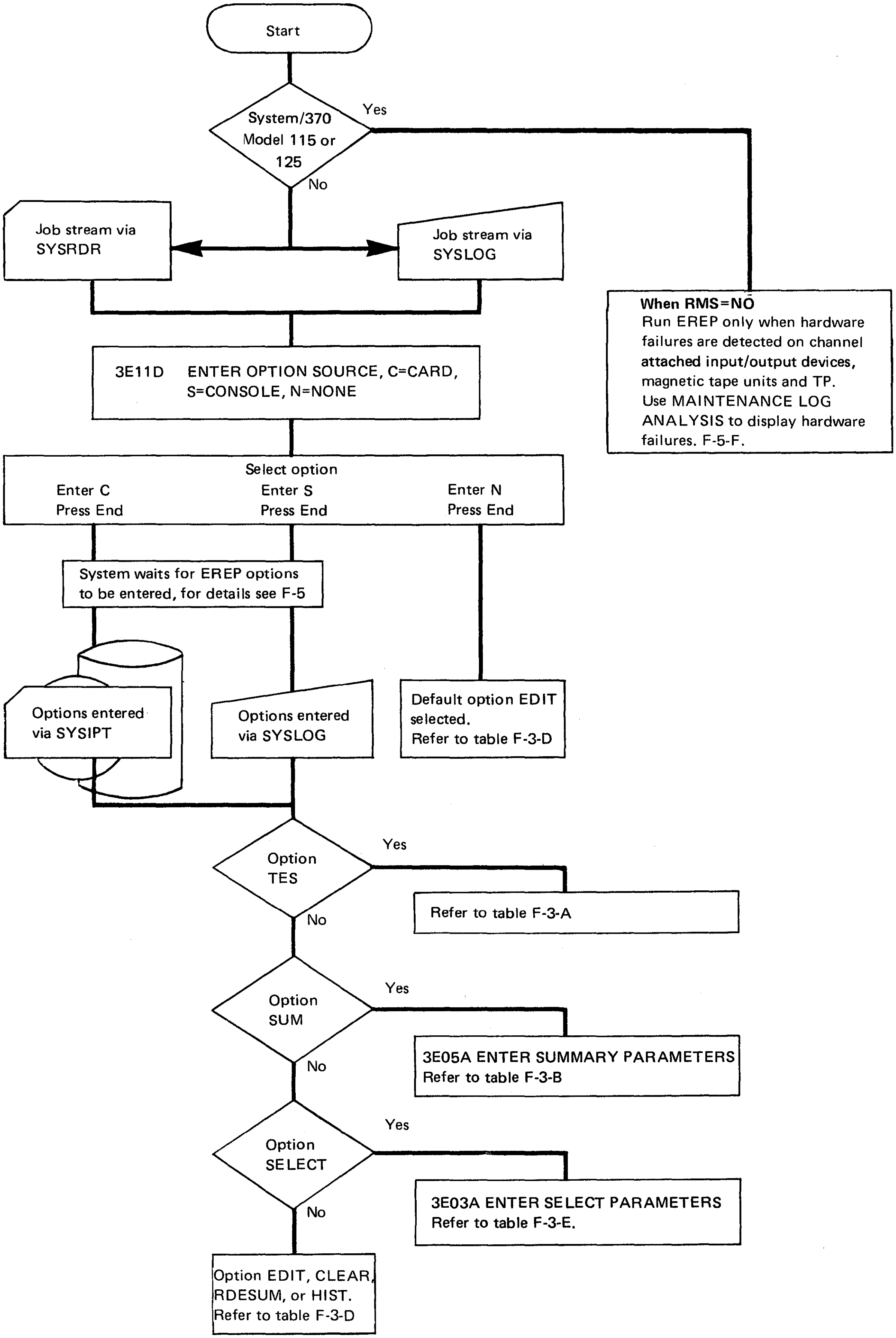
If you use the console printer-keyboard for input, you respond to the prompter messages.

There can only be one option per line (SYLOG entry) or one option per card (card entry). Only one option card for each type of option (EDIT, CLEAR, SUM, HIST, TES, and SELECT) may be entered in an EREP run. However, when entered via SYSLOG, the SUM and SELECT options may be executed more than once in a single EREP run. Table F-3 lists the EREP options.

You can alter the order of EREP actions by specifying two options. For example:

OPTION EDIT	Edit and print the SYSREC file.
OPTION HIST	Update the history tape, and then clear the file.

**The END key on the Model 125 is replaced by the ENTER key*


Start
System/370 Model 115 or 125
Yes
No
Job stream via SYSRDR
Job stream via SYSLOG
When RMS=NO
Run EREP only when hardware failures are detected on channel attached input/output devices, magnetic tape units and TP.
Use MAINTENANCE LOG ANALYSIS to display hardware failures. F-5-F.
3E11D ENTER OPTION SOURCE, C=CARD, S=CONSOLE, N=NONE
Select option
Enter C Press End
Enter S Press End
Enter N Press End
System waits for EREP options to be entered, for details see F-5
Options entered via SYSIPT
Options entered via SYSLOG
Default option EDIT selected. Refer to table F-3-D
Option TES
Yes
No
Refer to table F-3-A
Option SUM
Yes
No
3E05A ENTER SUMMARY PARAMETERS
Refer to table F-3-B
Option SELECT
Yes
No
3E03A ENTER SELECT PARAMETERS
Refer to table F-3-E.
Option EDIT, CLEAR, RDESUM, or HIST.
Refer to table F-3-D

EREP

For input from either SYSIPT or SYSLOG, embedded blanks within the operation, option, or parameter are not allowed. Misspelled words, syntax errors, duplicate option statements and unsupported options are invalid.

When input is from SYSIPT, these errors will cause 40 bytes of the card to be issued to SYSLOG along with the message:

3E04I INVALID PARAMETER

or

3E12D INVALID OPTION

At this time, you may place a corrected card in SYSIPT and then press END to process the desired option. If you do not want to process the card in error, enter END and the program will ignore that option card. However, if you wish to cancel the job, enter CANCEL END and the EREP job will be canceled. Multiple options are allowed by EREP. See figure for a summary of the EREP options.

Entering options via SYSLOG: When the EREP options are entered via SYSLOG, it is possible to execute the SUM and SELECT options more than once during an EREP run. After the SUM or SELECT function has executed, the message

3E03A ENTER SELECT PARAMETERS

or

3E05A ENTER SUMMARY PARAMETERS

is issued to SYSLOG. You may execute the SUM or SELECT function again by entering parameters at this time. If you wish to terminate the SUM or SELECT option, press END.

When entering the EREP option via SYSLOG, the entry must not exceed 80 positions. Enter, in this sequence:

1. The operation, OPTION
2. A blank
3. The option.

Any parameters should follow the OPTION statement on the next line(s). Repeat this procedure for each option; when all options have been specified, enter END to continue processing.

Note: The END key on the Model 125 is replaced by the ENTER key.

Entering options via SYSIPT: When entering the EREP options via SYSIPT, column 1 must be blank and only one option per card is allowed (for example, HIST with UPNEW or with NEW and/or 2 is considered one option). Each option may only be entered once for each execution of the EREP program.

Example job streams for executing EREP:

```
// JOB EXAMPLE1
// TLBL EREPNEW
// TLBL TAPEIN
// ASSGN SYS007,X'cuu'
// ASSGN SYS008,X'cuu'
// ASSGN SYS009,X'cuu'
// LBLTYP TAPE
// EXEC EREP
  OPTION HIST, NEW
  OPTION TES,TAPE,NEW
/*
/&
// JOB EXAMPLE2
// TLBL TESUP
// TLBL EREPUP
// ASSGN SYS007,X'cuu'
// ASSGN SYS008,X'cuu'
// ASSGN SYS009,X'cuu'
// LBLTYP TAPE
// EXEC EREP
  OPTION EDIT
  OPTION TES,TAPE
  OPTION HIST
/*
/&
```

EREPNEW and EREPUP must be the filenames for new history files or for updating. TAPEIN and TESUP must be the file names for a new TES history tape or an update TES history tape.

```
                              --- MACHINE CHECK DATA EDITING ---
*****************************************************************************************************

MODEL 0145          SERIAL NUMBER 010043        JOB IDENTITY - A            PROGRAM IDENTITY - NO NAME

                                                        DAY  YEAR                HH MM SS
                                                 DATE - 010   72         TIME - 15 13 21
                     SM KS  IC  CM   IA
OLD MACHINE CHECK PSW   FF 15 0000 40 007812

                     ***********************************************************

                     --- MACHINE CHECK INTERRUPT CODE ---

          --- SUB CLASS ---

SYSTEM DAMAGE (SD)     0                     CLOCK DAMAGE (CD)        0
PROC. DAMAGE (PD)      1                     EXTERNAL DAMAGE (ED)     0
SYSTEM RECOVERY (SR)   0                     AUTO-CONFIG (AC)         0
TIMER DAMAGE (TD)      0                     WARNING (W)              0

                   --- INTERRUPT TENSE CODES ---

BACK-UP (B)            0                     DELAYED (D)              0

                   --- STORAGE AND PROTECTION ERROR CODES ---

UNCORRECTED STORAGE ERRORS (SE)   1          UNCORRECTED PROTECTION ERRORS (PE)     0
CORRECTED STORAGE ERRORS (SC)     0

                         --- PSW VALIDITY CODES ---

AMWP BITS OF M.C. OLD ARE VALID (WP)    1     SYSTEM MASK OF M.C. OLD IS VALID (MS)   1
PROGRAM MASK OF M.C. OLD IS VALID (PM)  1     INSTR ADDR OF M.C. OLD IS VALID (IA)    1

                       --- MISC VALIDITY CODES ---

FAILING STORAGE ADDR IS VALID (FA)    1      REGION CODE VALID (RC)              1
FP REGS STORED ARE VALID (FP)         1      GP REGS STORED ARE VALID (GP)       1
CONTROL REGS STORED ARE VALID (CR)    1      EXTENDED LOGOUT AREA VALID (LG)     1  — INFO
INSTR MODIFIED STORAGE VALID (ST)     1                                            VALID

EXTENDED LOGOUT LENGTH          0000           FAILING STORAGE ADDRESS        00007812

                             --- REGION CODE ---
ERROR CORRECTION CODES        0000           CONTROL WORD ADDRESS        0000
                     ***********************************************************
```

```
                              --- MACHINE CHECK DATA EDITING ---

--- FLOATING POINT REGISTERS ---

FP REGS 0,2    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
FP REGS 4,6    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00

--- GENERAL PURPOSE REGISTERS ---

GP REGS 0-3    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
GP REGS 4-7    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
GP REGS 8-B    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
GP REGS C-F    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00

--- CONTROL REGISTERS ---

CT REGS 0-3    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
CT REGS 4-7    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
CT REGS 8-B    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00
CT REGS C-F    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00

--- MACHINE CHECK LOGOUT BYTES ---

0000  40008FFF 00000000 00000000 00000000   00007812 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0030  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0060  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0090  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
00C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
00F0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   00000000 00000000
```

An example of an EREP output obtained after a storage failure.

The programmer's marks indicate the areas of interest.

Note: the entry CONTROL WORD ADDRESS is not applicable for the Model 125.

When to use EREP

Your IBM customer engineer will usually advise you when an EREP printout is required, and tell you which option to select.

Under certain hardware failure conditions, a message issued on SYSLOG, for example, message 0T11W in the *DOS/VS Messages* manual will request you to RUN EREP.

Other DOS/VS messages that request you to RUN EREP are issued, for example, in the following cases:

- When the first record on the last track of the recorder file is reached, run EREP to avoid the risk of losing statistics.
- When an unrecoverable I/O error on the recorder file occurs while the record indicated is being accessed, the record is ignored and processing continues. If this error persists, run EREP to retrieve the information from the file and recreate the file using different disk extents.
- When SYSREC becomes full, no further recording occurs until the file is purged. To avoid the risk of losing statistics, run EREP. No recycling of the file occurs.
- For system termination situations (for example, a machine check was unrecoverable, the channel caused system reset, or two channels are damaged) encountered by MCAR/CCH, recording is attempted. Depending on the success of recording, the execution of EREP is requested. An attempt is made to write a message to the operator. If the attempt is unsuccessful, the message code is in low main storage.
- If the recorder file is more than 90% full at IPL time, the operator is requested to run EREP to prevent the loss of pertinent hardware data.

Another occasion when you may choose to execute the EREP program is when you suspect that a hardware error is causing program errors. From the EREP printout you are able to detect any hardware failure and inform your IBM customer engineer of it.

SEREP
(MODEL 135, 145 or 155-11)

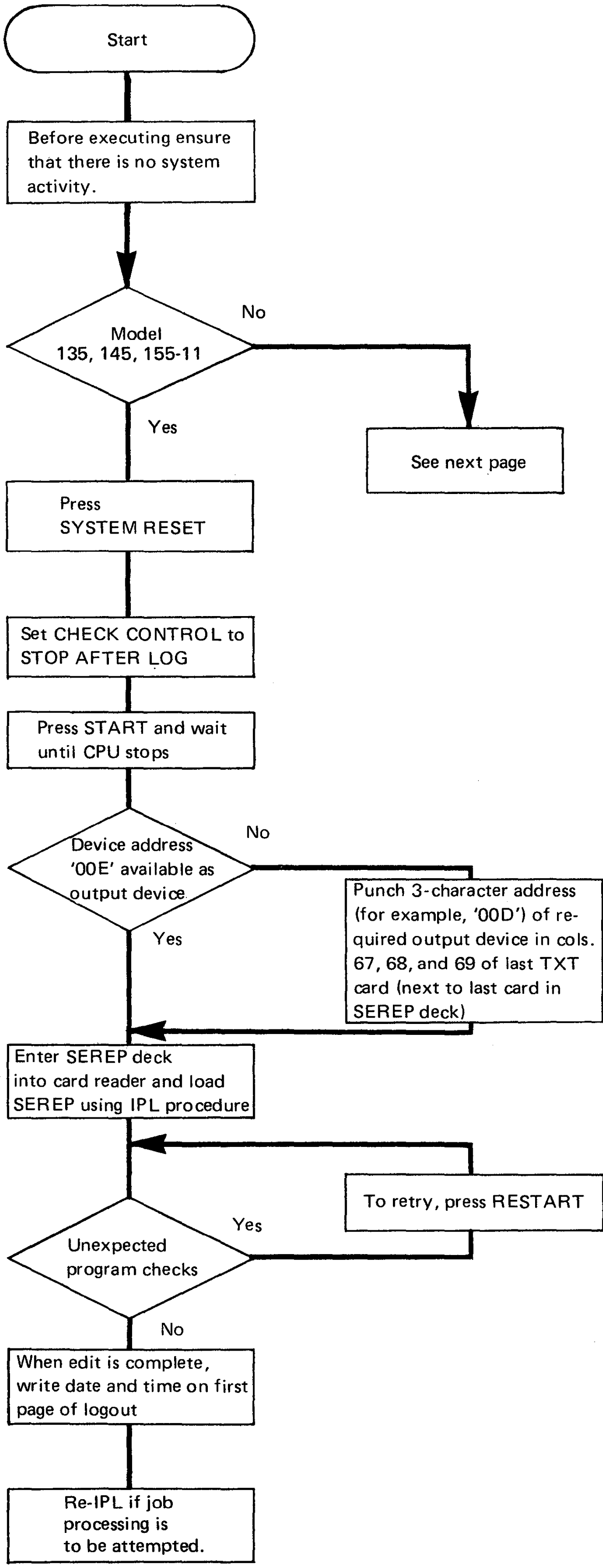
SEREP is a self-loading, stand-alone program used to:
1. Write the logout from real storage to some storage device such as tape for later use by the IBM CE
2. Perform a hard-copy Edit/Print of the logout.

When to use

SEREP is primarily an aid provided for the IBM CE to help his offline diagnosis of hardware failures. For this reason SEREP need only be executed on the advice of the IBM CE or when requested to do so by a message on SYSLOG, or when a hard wait occurs and byte 1 of low real storage contains S (X'E2').

Flowcharts in Section 3 indicate when to use SEREP during IPL errors or if the system enters a hard wait state.

Not applicable to the Models 115 and 125.



The procedure for executing the SEREP program.

Wait state while executing SEREP

Normal Waits
When no output device is specified, or the specified device is not ready, the system enters the wait state after loading SEREP

Hard Waits
An unexpected program check during execution of SEREP causes a message to be printed, and the system enters the wait state. Retry is attempted by pressing RESTART. Re-IPL should be avoided because alteration of PSWs by the SEREP program may cause that edit to be erroneous.

Termination
When logout is complete, a message is issued and the system enters the wait state. If no log is found, a message is issued and the system enters the wait state

F-4-F

Avoid new IPL procedure. Because SEREP might have altered its PSW by this time, a re-IPL can cause part of edit to be wrong

The only possible operator intervention that may be required would be for mounting the accumulation tape when the program asks for it.

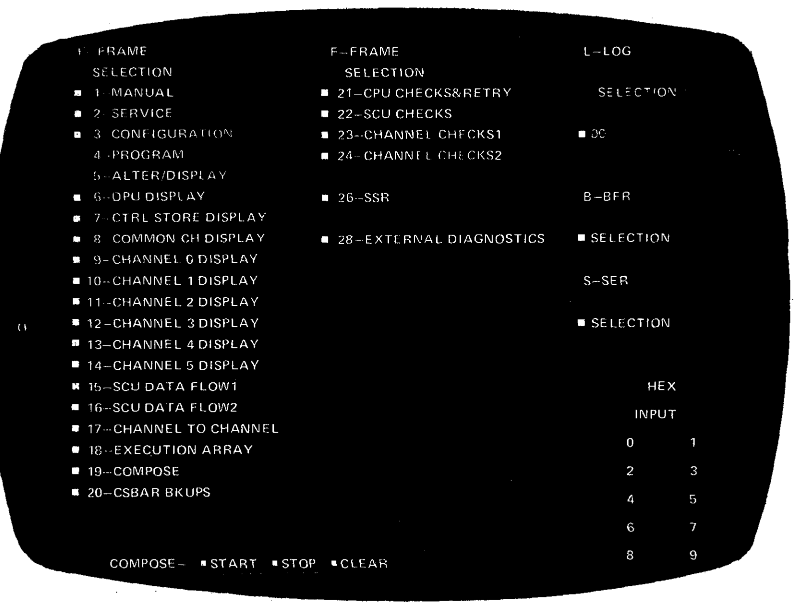
SEREP (MODEL 158)

Unlike the Models 135, 145, and 155-11, the Model 158 has no CPU logout area in real storage. Instead of being recorded in that area, certain types of hardware errors are recorded on the Log Recording Console File. The SEREP program also resides on the console file and can be loaded through its own IMPL procedure by using the MANUAL, SERVICE and INDEX frames. When SEREP is loaded, the SEREP frame is displayed to enable you to select one of the options. The options include:

- Write the log to tape
- Edit and print the log
- Select and process one of eight previous logs.

How to execute (Edit and print option)

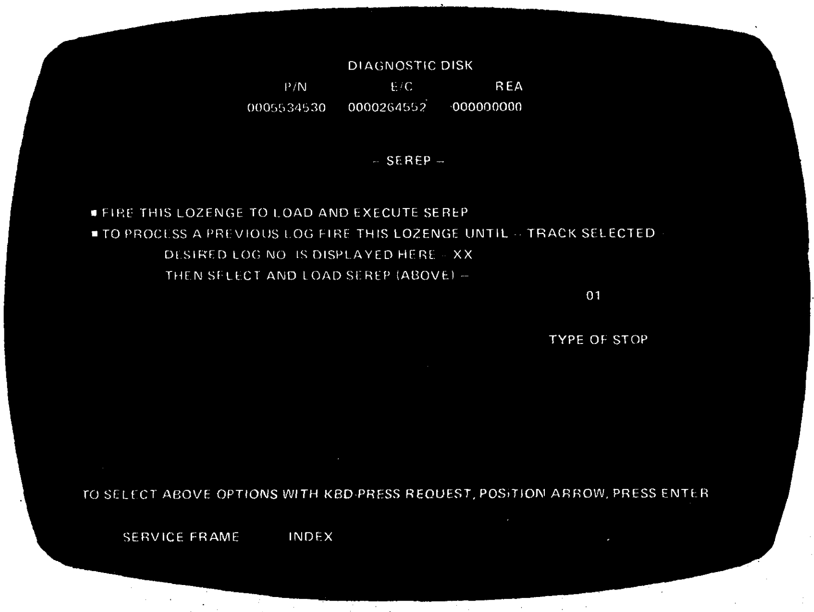
1. Press STOP followed by MODE SEL
The manual frame will be displayed
2. Type in F2 or press light pen to lozenge ▪ SERVICE FRAME
(If a "hard copy" of the service frame is required on the 3213 printer, press COPY key or press light pen to lozenge ▪ COPY)
The service frame will be displayed.
3. Type in F4 or press light pen to lozenge ▪ 4-INDEX FRAME
The index frame will be displayed as shown below.
(Using COPY will not generate a hard copy of this frame.)



How to execute . . . continued

4. Type in F28 or press light pen to lozenge
   ■ 28-EXTERNAL DIAGNOSTICS

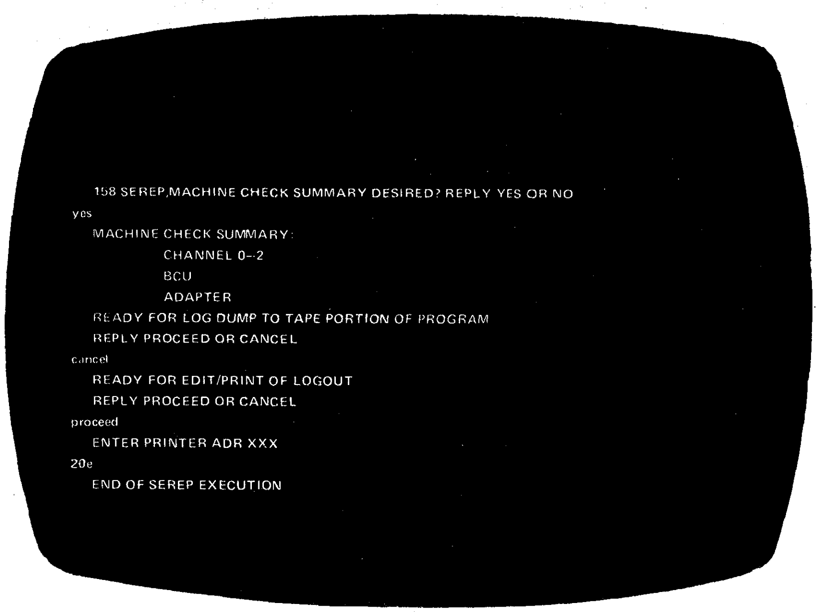
The frame shown below will be displayed.
(Using COPY will not generate a hard copy of this frame.)



5. Press light pen to the lower lozenge until number 08 is displayed at the position of the two XX in the example shown above.
6. Press light pen to upper lozenge on the display

The program frame will be displayed.

7. Press REQ key
8. Respond to messages displayed as shown in the hard copy example below.



(The operators responses are shown in lower case characters.)

The SEREP frame will be "rolled" onto the display as responses are given. After the address of the printer to be used as output device is entered, SEREP output will be observed on that printer.

Note: The output device can be a tape unit, or the console printer.

F-4

LOG ANALYSIS (MODELS 115 AND 125)

The LOG ANALYSIS facility allows the operator to display statistical data about hardware failures that are logged on the DISKETTE.

The type and amount of detail displayed is selected by entering appropriate mnemonics into the MAINTENANCE PROGRAM SELECTION display. The sequence of displays is designed to guide the operator from the initial type of display selected to displays that provide more detailed data.

For an interpretation of the data displayed refer to the Central Test Manual.

The example shown on the opposite page illustrates the sequence of displays obtained to display the errors logged by the IPU (Instruction Processor Unit).

When to use

Your IBM customer engineer will usually advise you when to use this feature, and tell you which display to select. He may require a hard copy for offline analysis of all the displays selected, therefore save the hard copy output.

Under certain hardware failure conditions, a message issued on SYSLOG, for example, message .0T11W in the *DOS/VS Messages* manual will request you to RUN EREP.

For the Models 115 and 125 you should only run EREP when requested to by a DOS/VS message. For example when the recorder file is full a message will be displayed informing you of this and re questing you to run EREP. Otherwise, before running EREP you should first contact your IBM customer engineer, who will then advise you on further action as mentioned in the previous paragraph.

How to use

To obtain the LOG ANALYSIS display required using fast selection,
1. Press the MODE SELECT key.
2. Type in M followed by the associated mnemonics of the analysis to be displayed.
3. Press the ENTER key.

By selecting and entering the appropriate mnemonics, the operator can display logged errors for a particular input/output device or a particular part of the CPU.

Press MODE SELECT

```
                 * MODE SELECTION *

   R  SYSTEM RESET          A  ALTER/DISPLAY
   C  ADDRESS COMPARE       I  INSTRUCTION STEP
   L  PROGRAM LOAD          P  RESTART
   T  INTERVAL TIMER        M  MAINTENANCE
   K  CHECK—CONTROL         S  STORE STATUS
   D  STORAGE DUMP          U  SAVE USAGE COUNTERS
   E  ICA LINE MODES
MODE SPECIFICATION:
```

Enter M and press ENTER

```
        * MAINTENANCE PROGRAM SELECTION *

LOG                      TESTS              CE—MAN.CPS

A = LOG GENERAL          J = CPU            S = IOP
B = CPU                  K = 1403           U = CRT-SCOPE
C = CARD/PRINT I/O       L = 2560/5425      V = I/O EXERS
D = DISK                 M = 3504/3525      X = IPU
E = ICA                  N = DISK           Y = MATRIX S
I = CHANG. DISKETTE      O = ICA            Z = MATRIX M
PROGRAM SELECTION: M     R = SYSTEM TEST (ASCP)
```

Enter B and press ENTER

```
          * CPU LOG ANALYSIS PROGRAMS *

   B =  SVP  BUS—0 LOG            K =  MTA   LOG DISPLAY
   C =  IPU  ANALYSIS             L =  MSCI  LOG ANALYSIS
   D =  IPU  LOG DISPLAY          M =  MSCI  LOG DISPLAY
   E =  MSC  ANALYSIS             N =  MPX   ANALYSIS
   F =  MSC  LOG DISPLAY          O =  MPX   LOG DISPLAY;
   G =  IOP  8—F ANALYSIS
   H =  IOP  8—F LOG DISPLAY

   PROGRAM SELECTION: MB

 ID:C003              M:X.DA00.4C.
```

Enter C and press ENTER

```
       * CARD/PRINT I/O LOG DISPLAY PROGRAMS *

   B=  3504  LOG  DISPLAY      H=  5425 LOG DISPLAY

   D=  3525  LOG  DISPLAY      K=  1403 LOG DISPLAY

   F=  2560  LOG  DISPLAY      M=  5203 LOG DISPLAY

   PROGRAM SELECTION: MC
```

Displaying the IPU Log Analysis

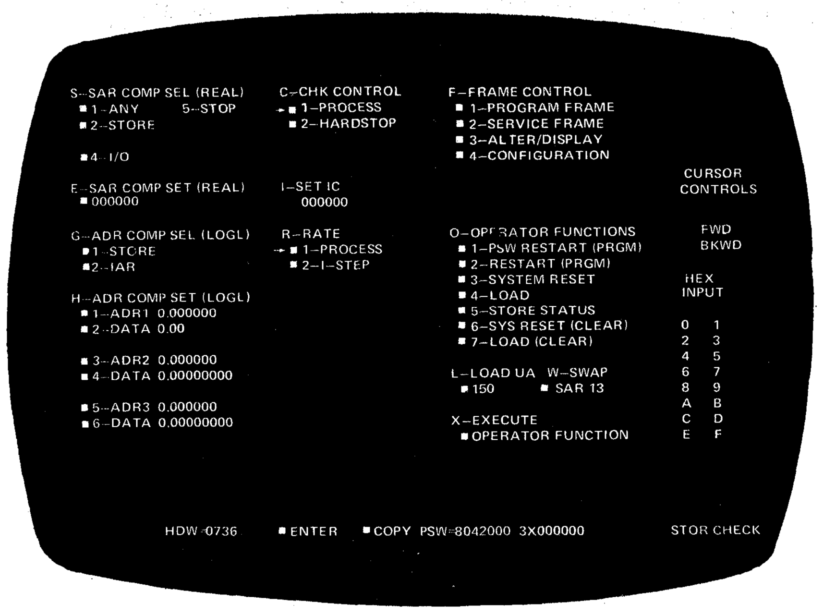
F-5

DISPLAY FRAMES (MODEL 158 ONLY)

This facility enables an operator to display information about hardware failures and warn the shift manager of IBM CE immediately about the nature and possible cause of the failure.

Recognizing a hardware failure

A hardware failure is indicated by a message which 'flashes' on and off at the lower right hand corner of the program frame or manual.
The example below shows a hardware failure indicated by the words STOR CHECK on the manual frame.



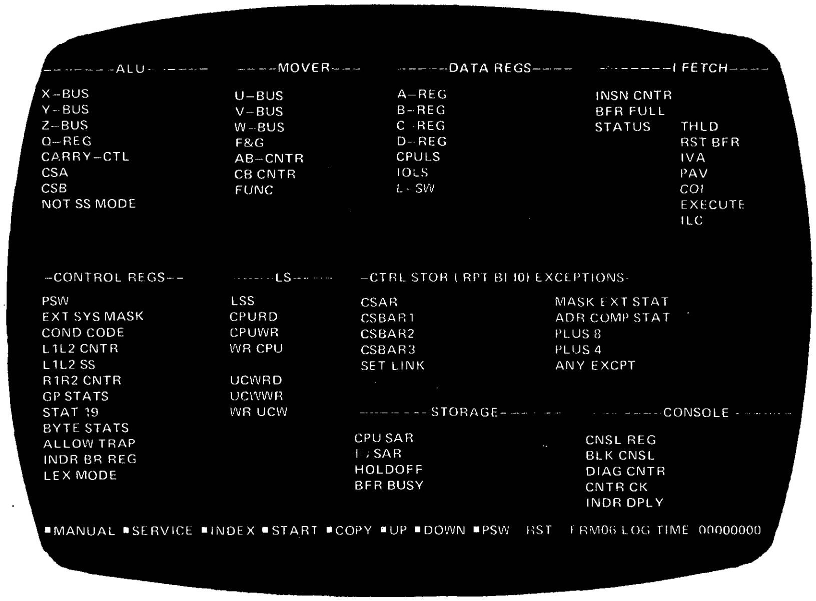
After recognizing the failure an operator is able to 'scan' the display frames and thus obtain detailed information about the condition of the hardware. This information may enable the IBM CE to diagnose the failure immediately and advise on continued system operation. He may also advise that the SEREP is executed and request 'hard copies' of the display frame on which the failure is indicated to enable an offline diagnosis of the failure.

How to use

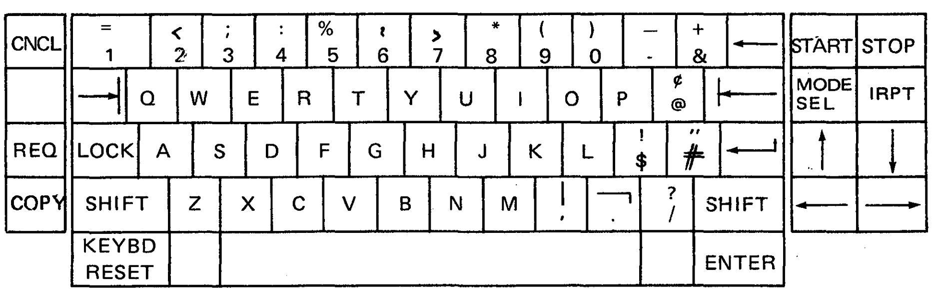
From the program frame:

1. Press MODE SEL
   The manual frame is displayed
2. Type in F2, or press light pen to lozenge ■ SERVICE FRAME
   The service frame will be displayed
3. Type in F4, or press light pen to lozenge ■ INDEX FRAME
   The index frame will be displayed, an example of which is shown in Section 1.
4. Press light pen to lozenge – CPU DISPLAY
   The first display frame will be displayed, an example of which is shown at the top of the opposite page.

How to use



5. Scan the frame for any characters that flash on and off beside an entry displayed. For example, Z-BUS 614250 indicates that the hardware failure is caused by a failure in the Z-BUS.
   If a hardware failure is indicated, press COPY to obtain a hard copy of that frame and make a note on the hard copy about the error. (Characters that indicate an error are not copied by the system.)
6. Press the key marked ↑ on the keyboard, as illustrated below.
   The next display frame will be displayed.



7. Repeat steps 5 and 6 until all display frames have been scanned and hard copies made of those containing information about the failure.
8. Press CANL to obtain the program frame.

When to use
1. After recognizing a hardware failure as shown in the example above.
2. On advice from your IBM CE.

F-6

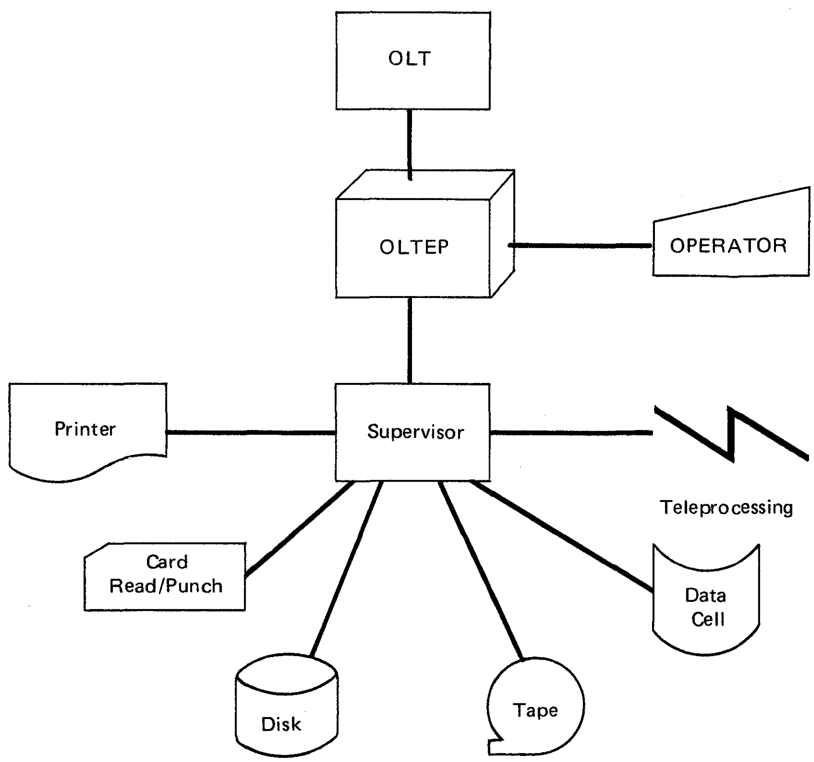
OLTEP

IBM provides a set of device test programs that run under control of DOS/VS. These test programs and the online test executive program form the online test system. The Online Test Executive program (OLTEP) is an interface between the system and the online test programs (OLTs) and communicates with the operator during the running of tests.

Some uses:
- Diagnosing I/O errors
- Verifying I/O device repairs and engineering changes
- Checking I/O devices.

Some features:
- Multiple device testing
- Data security
- Data protection
- No re-IPL time required
- Prompting
- ASCII data conversion
- Accessing of error recording information
- CDS Equate function.

Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.



OLTEP-System Relationship

Description and Operation

OLTEP operates much like other problem programs in DOS/VS. It is cataloged into the core image library and called by standard job control statements. When OLTEP is called, it notifies the operator that it is active and it communicates with him during testing. OLTEP can run in a batch-only system or as a background program in a multiprogramming environment. OLTEP must be run in the background partition in real mode and requires at least 14K.

You can test an I/O unit with minimum interference to other programs running on the system. Testing an I/O device ordinarily does not interfere with system input and output. Any unit being tested (except for direct access devices) must not be assigned to the foreground partitions. Direct access devices, however, may be shared.

An OLTEP user language defines and controls the test. With this language, you select the devices to test, the test sections to run, and the options to exercise. You enter this information via the console device or in the form of a control statement in the job input stream. This information is referred to as the test-run definition, which is common to OLTEP components for all operating systems.

You can test multiple devices of the same type with no operator interventions other than those required for data protection and data security. OLTEP loads and executes the test sections one at a time until all the tests for one device are completed. If requested, the test sections then repeat for the next available device. Testing continues in this manner until all units in the test-run definition are tested.

During testing under control of OLTEP, the system error recovery procedures are bypassed for the device being tested. OLTEP has built-in data integrity safeguards so that no data is destroyed without operator permission, and no protected data is accessed during testing.

Intentionally Blank

DEBUGGING FOR THE OPERATOR

Section 3

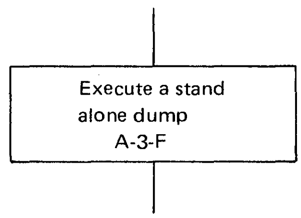
DEBUGGING PROCEDURES FOR OPERATORS

How to Use

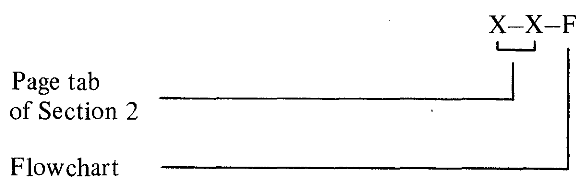
This section is in the form of flowcharts that help the operator in the initial isolation of and possible recovery from errors that occur during system operation.

- Each flowchart deals with a specific type of malfunction.
- Pointers to operator's flowcharts in Section 2 (that must be followed to complete a procedure in this section) are referenced by the page tabs used in Section 2.

  For example:



Key to references:



- When immediate recovery is not possible, offline program debugging is indicated.

Operator's Flowcharts

BYTE 0	BYTE 1	BYTE 2	BYTE 3	EXPLANATION
IPL Error Messages placed in Low Address Storage				
X'F0'	X'C9'	X'F0'	X'F0'	This code indicates that less than 16K of real storage is left for problem programs. Check that the correct disk volume is mounted on the device assigned to SYSRES, and re-IPL. If the error recurs, the system programmer must check the allocations of real partitions specified in the supervisor to be used, and check that at least 16K of real storage is available for execution of problem programs running in virtual mode.
X'F0'	X'C9'	X'F0'	X'F1'	If a card reader has been assigned to SYSRDR during system generation and is to be the IPL communication device, press the INTERRUPT key. If a card reader has **not** been assigned to SYSRDR during system generation and yet it is to be the IPL communication device, simply READY the reader.
X'F0'	X'C9'	X'F0'	X'F2'	This code means that the supervisor requested cannot be found. Check that the correct disk volume is mounted on the device assigned to SYSRES. If it is correct, re-IPL and specify a different supervisor when message 0I03A is issued and press the END/ENTER key, or press END/ENTER key only, to load the standard supervisor. (If possible contact the system programmer and check which supervisor to use.)
X'F0'	X'C9'	X'F1' X'F2'	X'F0' X'F8'	Refer to messages 01100A – 0128A in *DOS/VS Messages.*
MCH/CCH/IPL Hard Wait Codes placed in low address storage				
X'C1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable machine check.
X'C2'	X'E2'(2)	Not used	Not used	Irrecoverable channel failure during RMS fetch.
X'C3'	X'E2'(2)	A, I, S(1)	Not used	Channel failure on SYSLOG when RMS message scheduled.
X'C4'	X'E2'(2)	A, I, S(1)	Not used	No ECSW stored.
X'C5'	X'E2'(2)	A, I, S(1)	Not used	Channel failure: ERPBs exhausted.
X'C6'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; two channels damaged or a damaged channel situation occurred while RMS was executing an I/O operation.
X'C7'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system reset was presented by a channel.
X'C8'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system codes in ECSW are invalid.
X'C9'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; channel address invalid.
X'D1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable channel failure on SYSVIS.
X'07'	X'E6'	Channel	Unit or X'00'	IPL I/O error or equipment malfunction; condition code 2 during STIDC instruction. Channel and unit indicate whether device in error is SYSRES or communication device. When byte 3 = X'00', byte 2 indicates the channel for which STIDC instruction was issued. Re-IPL system.

Notes: 1. *A (X'C1') = SYSREC recording unsuccessful.*
I (X'C9') = SYSREC recording incomplete.
S (X'E2') = SYSREC recording successful.
2. *S (X'E2') = Run SEREP.*
3. *SDAID wait states are identified by X'EEEE' in the address part of the wait PSW.*

Table 3-1. WAIT STATE coded messages, part 1 of 2.

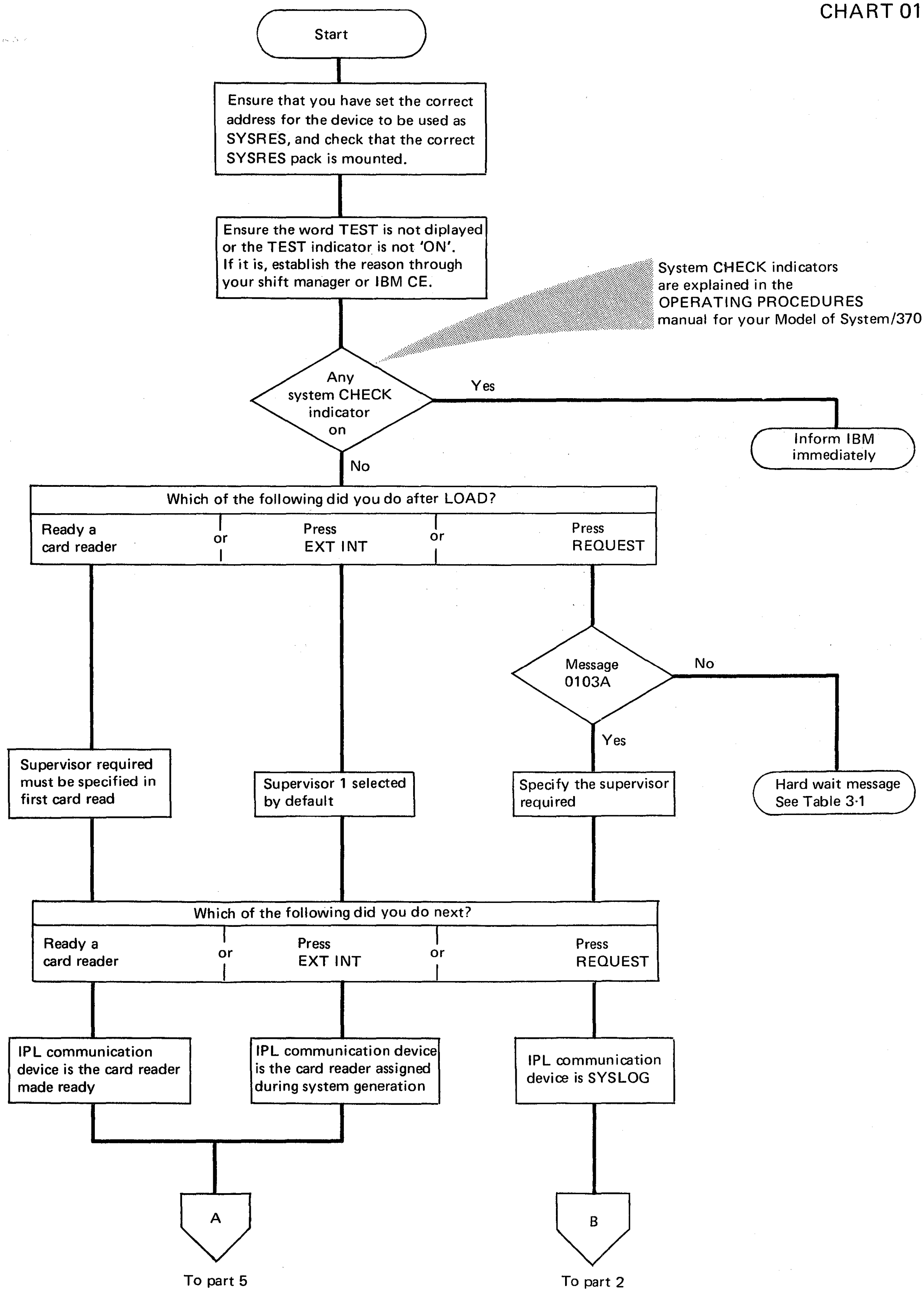
WAIT STATE CODES

BYTE 0	BYTE 1	BYTE 2	BYTE 3	
SDAID Hard Wait Code				
X'61'	X'E6'(3)	Channel	Unit	Another device is running in burst mode on same channel as SDAID output device. Re-IPL system.
SDAID Soft Wait Code				
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'(3)	X'00'	X'00'	SDAID Stop on Event. Press EXTERNAL INTERRUPT key to continue operations.
The following Hard Wait Codes are placed in general register 11 X'B' as well as in low address storage.				
X'00'	X'00'	X'0F'	X'FF'	Program Check in Supervisor.
X'00'	X'00'	X'0F'	X'FE'	I/O error during fetch from System CIL.
X'00'	X'00'	X'0F'	X'FD'	Channel Failure if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
X'00'	X'00'	X'0F'	X'FC'	Machine Check if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
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'FA'	Translation Specification Exception
X'00'	X'00'	X'0F'	X'F9'	Error on Paging I/O.
X'00'	X'00'	X'0F'	X'F8'	CRT phase not found.
X'00'	X'00'	X'0F'	X'F7'	No copy blocks available for BTAM appendage I/O request.
X'00'	X'00'	X'0F'	X'F6'	$MAINDR canceled during system CIL update. If this occurs, the system CIL is only partially updated and must be restored before use. This hard wait condition can also occur if the FETCH QUEUE BIT (FCHQ) is set in the linkage control byte in the partition communication region owned by the terminating partition.
Device Error Recovery Wait Codes placed in low address storage.				
X'08' to	X'C1' or	Channel	Unit	Error recovery messages. Refer to 0P messages in *DOS/VS Messages.*

Notes: 1. A (X'C1') = SYSREC recording unsuccessful.
I (X'C9') = SYSREC recording incomplete.
S (X'E2') = SYSREC recording successful.
2. S(X'E2') = Run SEREP.
3. SDAID wait states are identified by X'EEEE' in the address part of the wait PSW.

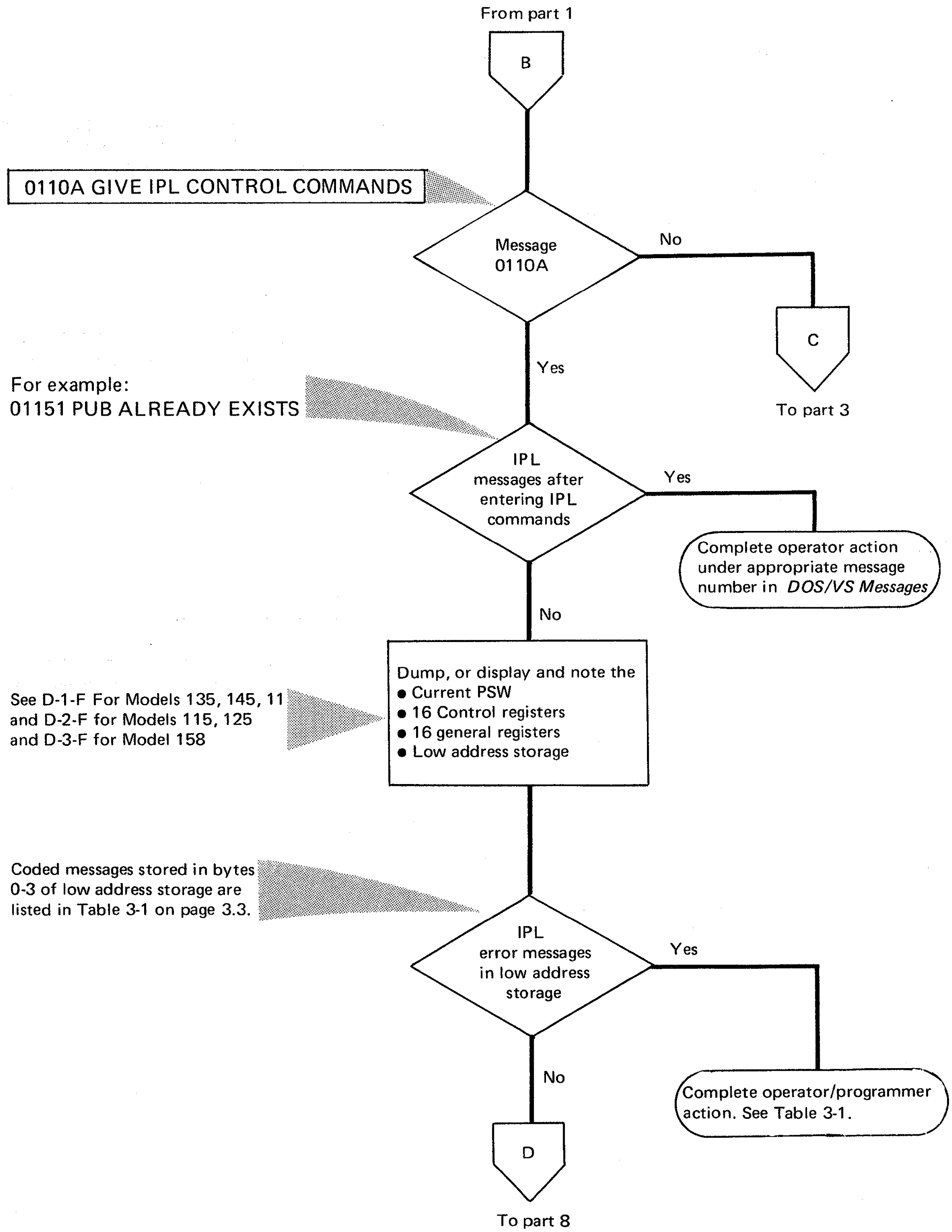
Table 3-1. WAIT STATE coded messages, part 2 of 2.

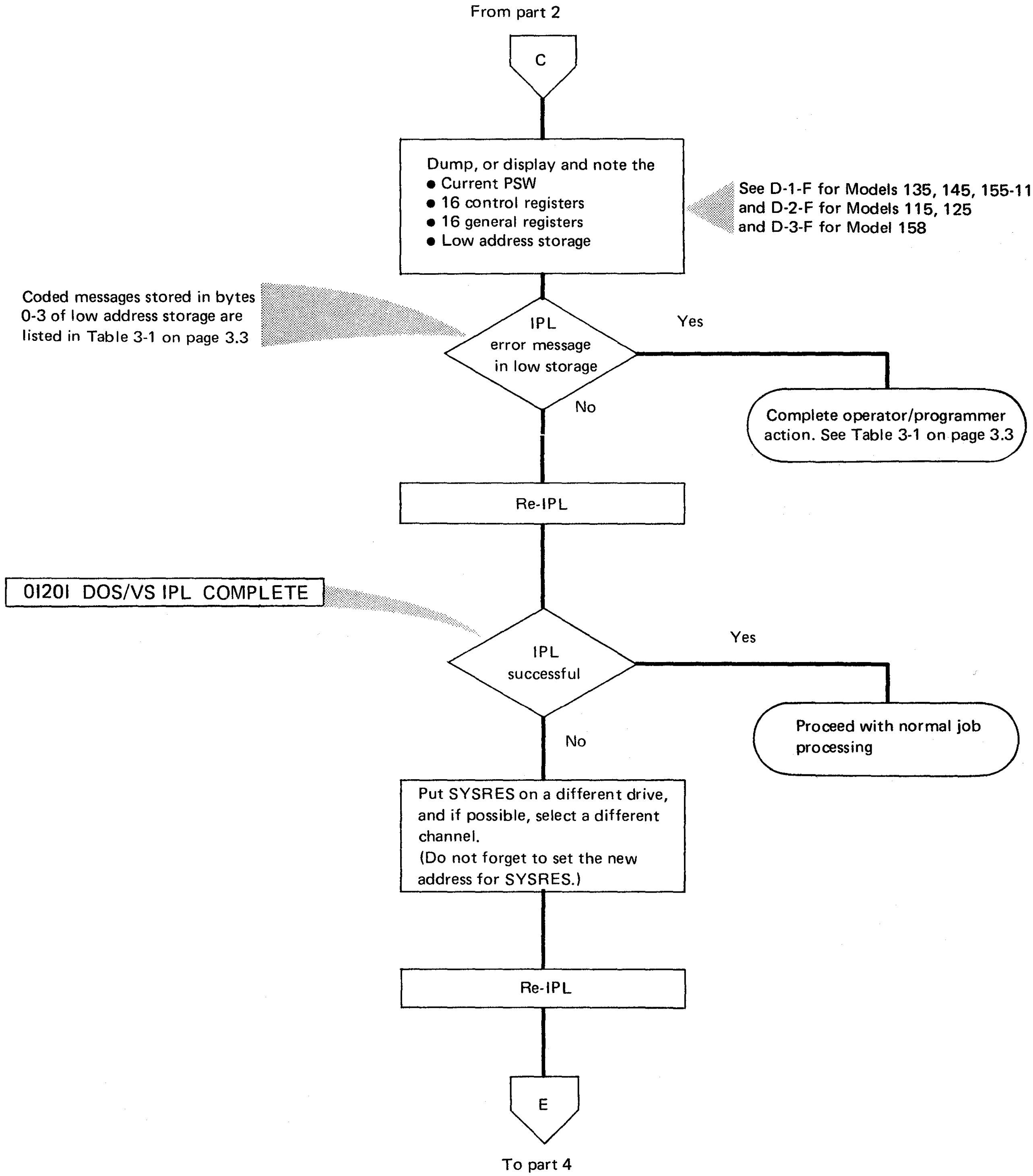
CHART 01, PART 1



Error during IPL

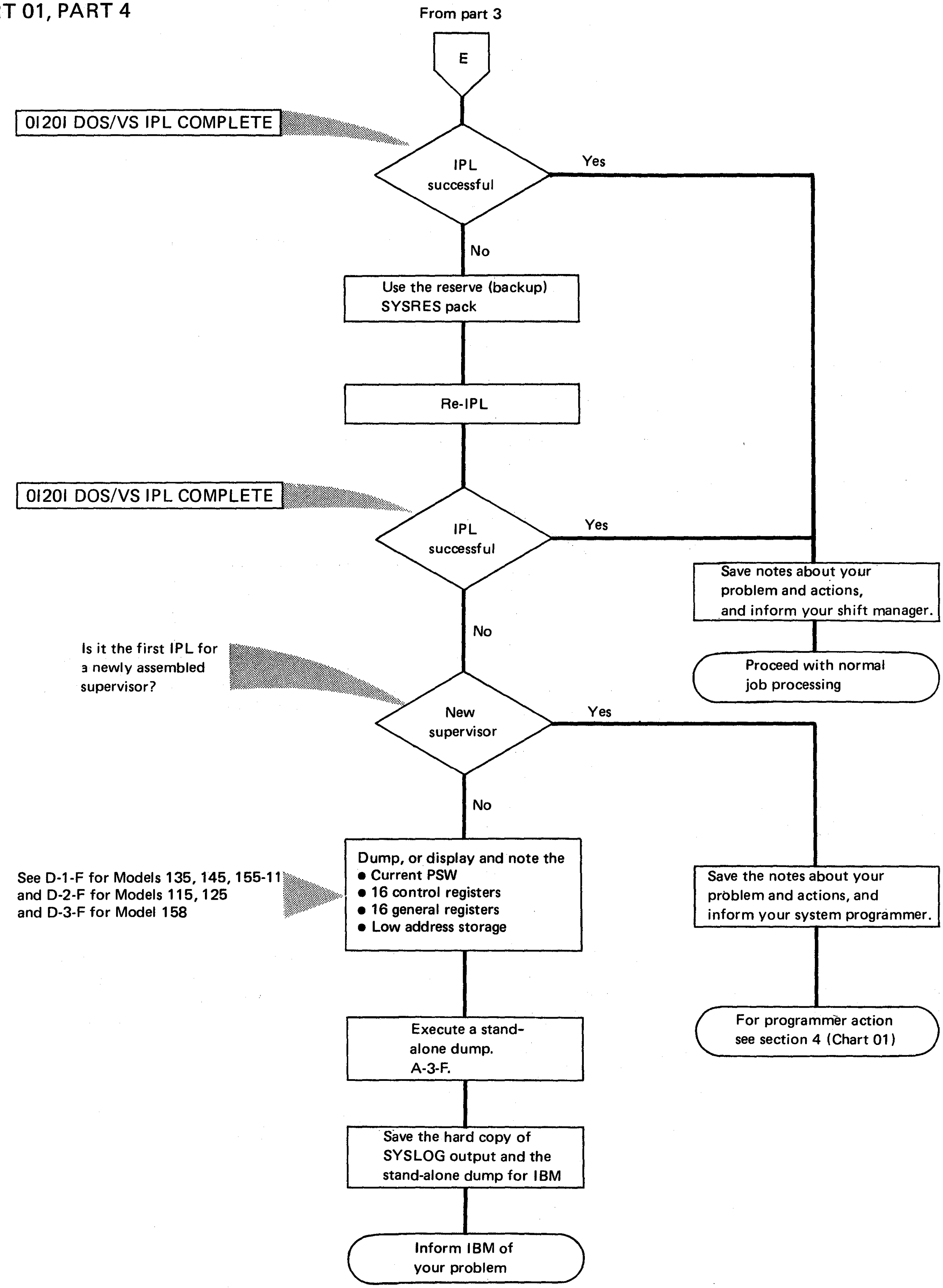
CHART 01, PART 2

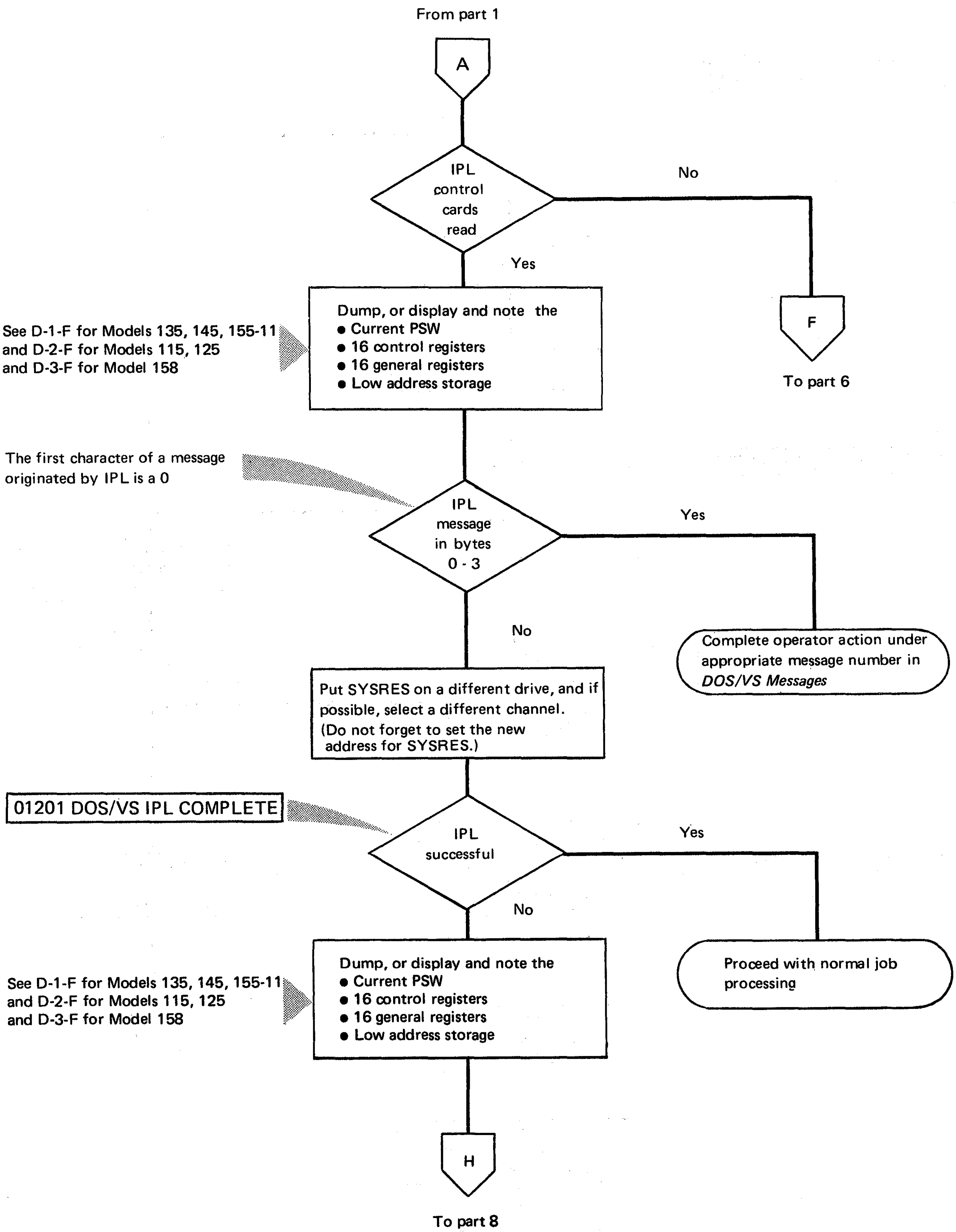



From part 2
C
Dump, or display and note the
● Current PSW
● 16 control registers
● 16 general registers
● Low address storage
See D-1-F for Models 135, 145, 155-11 and D-2-F for Models 115, 125 and D-3-F for Model 158
Coded messages stored in bytes 0-3 of low address storage are listed in Table 3-1 on page 3.3
IPL error message in low storage
Yes
No
Complete operator/programmer action. See Table 3-1 on page 3.3
Re-IPL
0120I DOS/VS IPL COMPLETE
IPL successful
Yes
No
Proceed with normal job processing
Put SYSRES on a different drive, and if possible, select a different channel.
(Do not forget to set the new address for SYSRES.)
Re-IPL
E
To part 4

Error during IPL

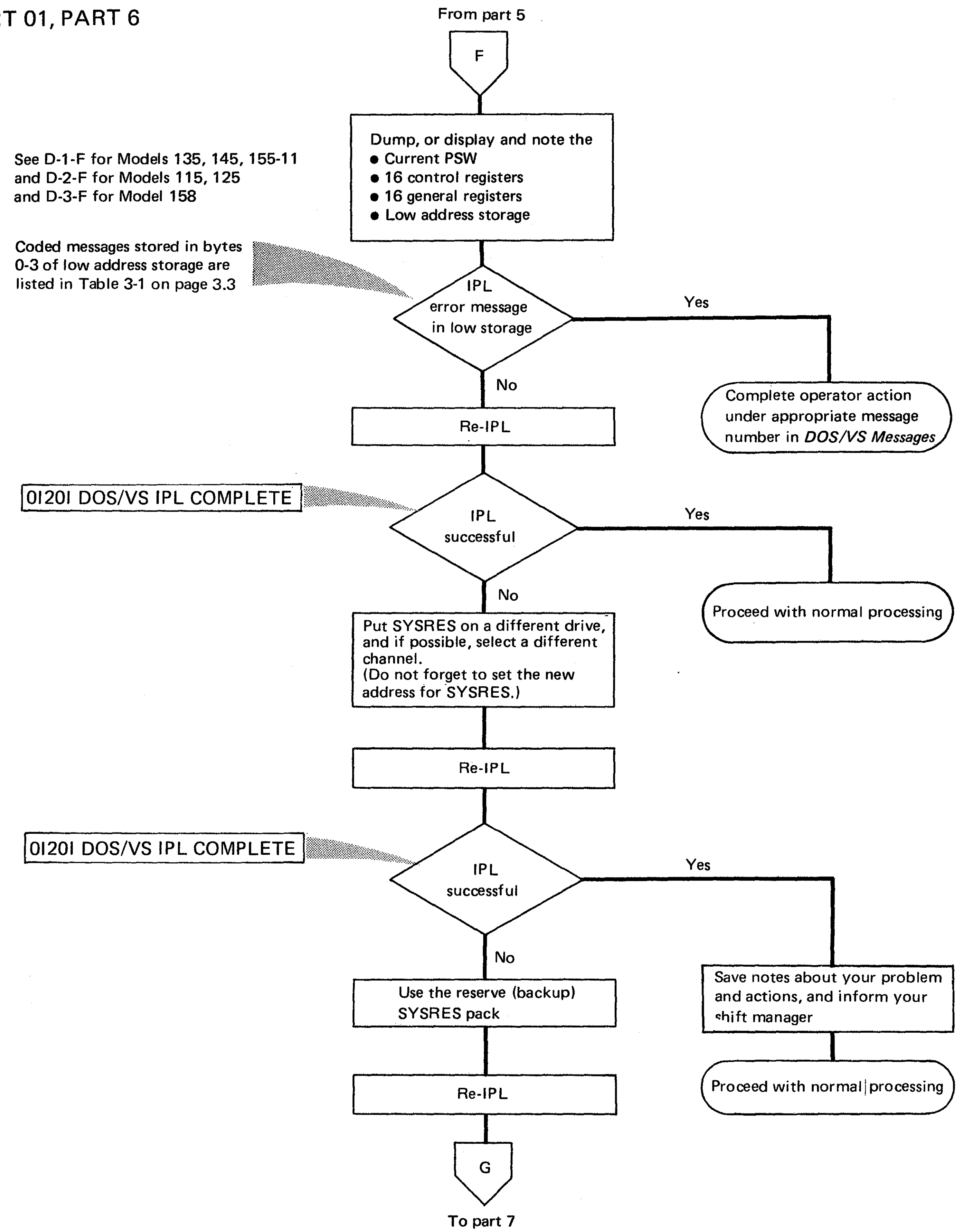
CHART 01, PART 4

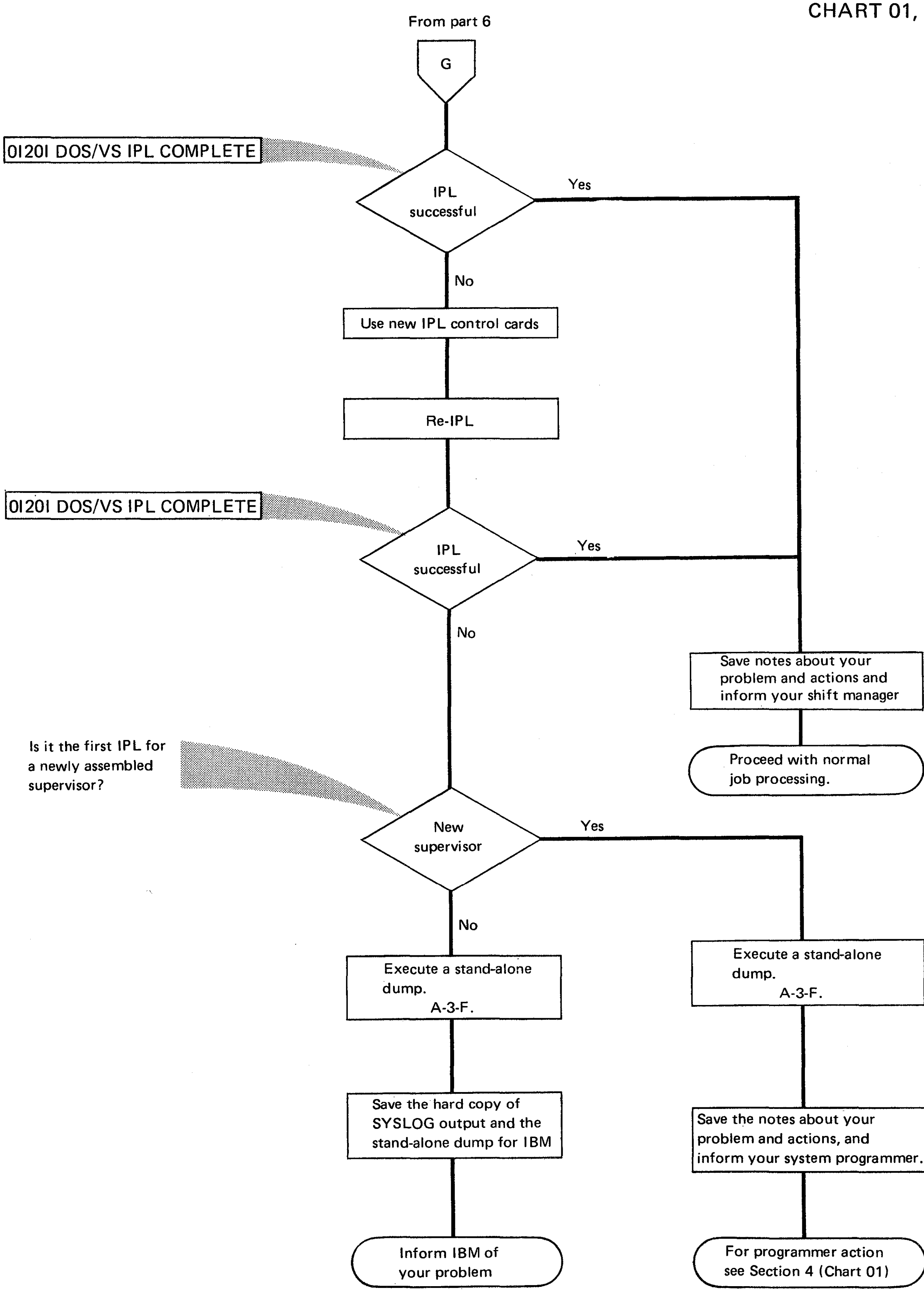



From part 1
A
IPL control cards read
No
Yes
Dump, or display and note the
● Current PSW
● 16 control registers
● 16 general registers
● Low address storage
See D-1-F for Models 135, 145, 155-11
and D-2-F for Models 115, 125
and D-3-F for Model 158
F
To part 6
The first character of a message originated by IPL is a 0
IPL message in bytes 0 - 3
Yes
Complete operator action under appropriate message number in *DOS/VS Messages*
No
Put SYSRES on a different drive, and if possible, select a different channel.
(Do not forget to set the new address for SYSRES.)
01201 DOS/VS IPL COMPLETE
IPL successful
Yes
Proceed with normal job processing
No
Dump, or display and note the
● Current PSW
● 16 control registers
● 16 general registers
● Low address storage
See D-1-F for Models 135, 145, 155-11
and D-2-F for Models 115, 125
and D-3-F for Model 158
H
To part 8

Error during IPL

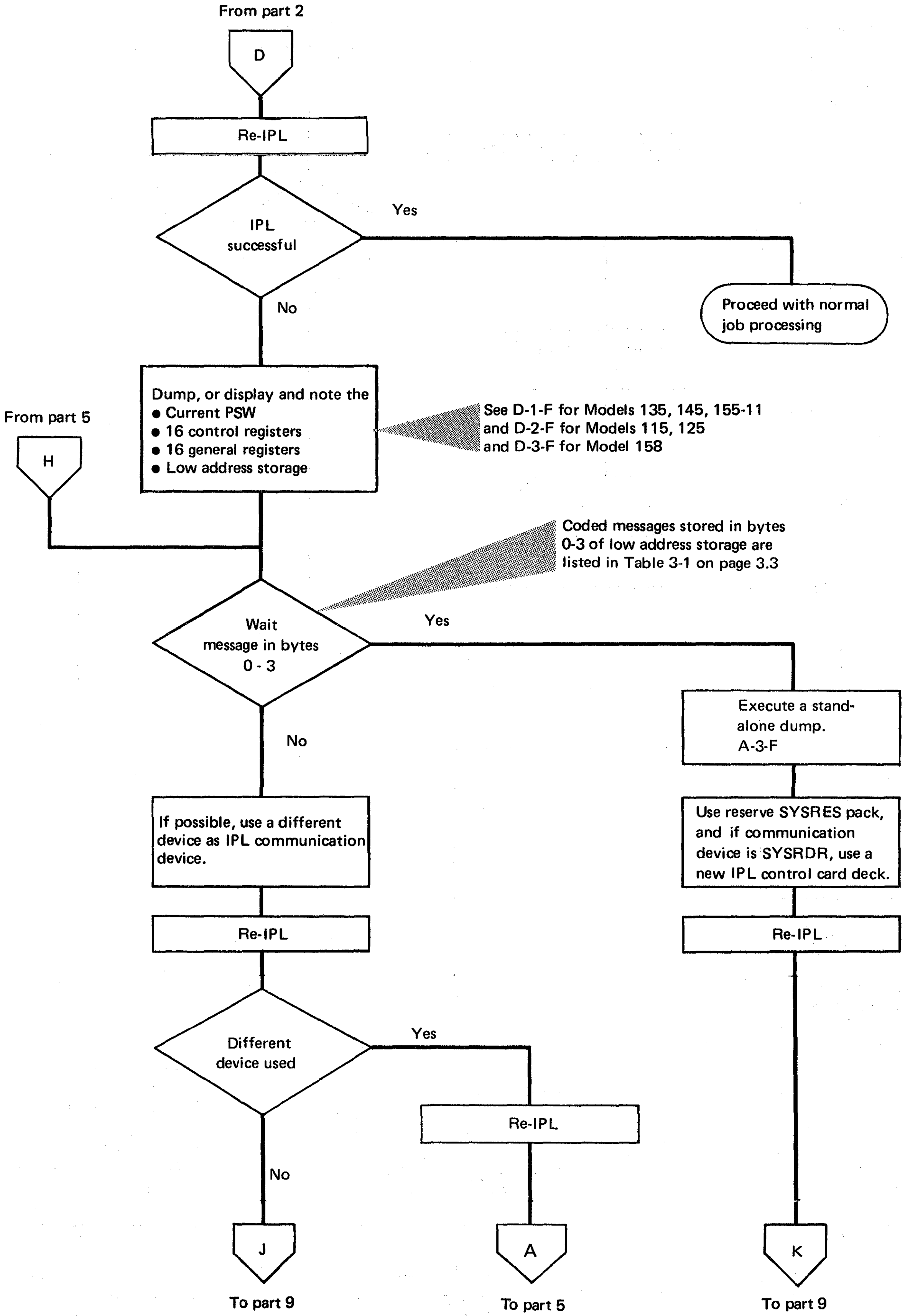
CHART 01, PART 6

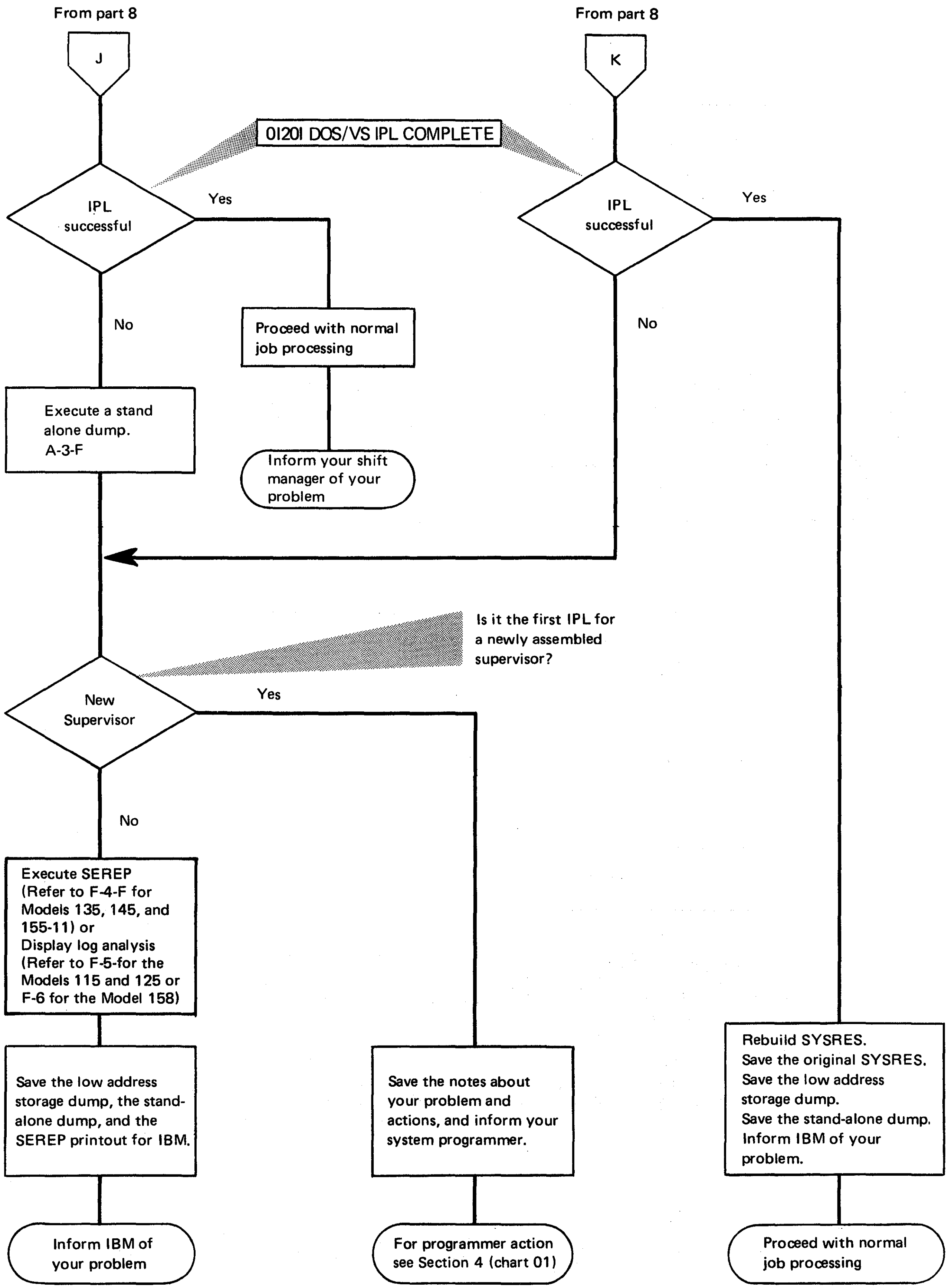



From part 6
G
0120I DOS/VS IPL COMPLETE
IPL successful
Yes
No
Use new IPL control cards
Re-IPL
0120I DOS/VS IPL COMPLETE
IPL successful
Yes
No
Save notes about your problem and actions and inform your shift manager
Proceed with normal job processing.
Is it the first IPL for a newly assembled supervisor?
New supervisor
Yes
No
Execute a stand-alone dump. A-3-F.
Execute a stand-alone dump. A-3-F.
Save the hard copy of SYSLOG output and the stand-alone dump for IBM
Save the notes about your problem and actions, and inform your system programmer.
Inform IBM of your problem
For programmer action see Section 4 (Chart 01)

Error during IPL

CHART 01, PART 8




From part 8
J
From part 8
K
0120I DOS/VS IPL COMPLETE
IPL successful
Yes
No
Proceed with normal job processing
Inform your shift manager of your problem
Execute a stand alone dump. A-3-F
IPL successful
Yes
No
Is it the first IPL for a newly assembled supervisor?
New Supervisor
Yes
No
Execute SEREP (Refer to F-4-F for Models 135, 145, and 155-11) or Display log analysis (Refer to F-5-for the Models 115 and 125 or F-6 for the Model 158)
Save the low address storage dump, the stand-alone dump, and the SEREP printout for IBM.
Inform IBM of your problem
Save the notes about your problem and actions, and inform your system programmer.
For programmer action see Section 4 (chart 01)
Rebuild SYSRES. Save the original SYSRES. Save the low address storage dump. Save the stand-alone dump. Inform IBM of your problem.
Proceed with normal job processing

NOTES FOR CHART 02

Note 1

Recognizing a wait state

Any of the following observations confirm that the system is in a Wait State:

- *WAIT indicator remains on, or on the Models 115 and 125 the word WAIT remains displayed on the video display unit.*
- *SYS indicator remains off (Not applicable to the Models 115 and 125).*
- *No I/O device activity occurs.*
- *One or more SYSTEM CHECK indicators on.*
- *A HARD MACHINE CHECK message is issued on SYSLOG or a coded "wait state" message may be contained in bytes 0-3 of low address storage or in GR II (X'B')*

Note 2

Recognizing a loop

One or more of the following occurrences may indicate that a job/program is in an unintended loop:

- *A steady glow in the light of the system control panel with the SYS indicator on. For the Models 115 and 125, the word WAIT may flicker on the video display unit. (This depends on the size and nature of the loop.)*
- *A rhythmic pattern in the lights of the system control panel, or, for the Models 115 and 125, the word WAIT may flicker on the video display unit.*
- *A pointless recurrence of I/O activity.*
- *A job/program that does not change status for a long time. This may result, for example, in an absence of I/O activity with both SYS and WAIT indicators on.*

A note to the operator: When a loop is recognized, first try to contact the programmer before beginning any debugging procedures. If this is not possible, follow the instructions in chart 04.

Note 3

Recognizing incorrect output

Incorrect output during system operation may be recognized by any one of the following:

A. Duplicate output

Output of identical data or more output than expected on:

- *line printer*
- *console printer*
- *card punch*
- *video display unit.*

B. Invalid or unidentified output

Printed (or displayed) output that is obviously incorrect on:

- *line printer*
- *console printer*
- *video display unit.*

C. Lack of output

No output when there should be, or less output than expected on:

- *line printer*
- *console printer*
- *card punch*
- *video display unit.*

Note 4

Job/program canceled by system

The system's canceling of job is normally caused by a Program Check Interrupt that is recognized by a message, for example:

```
BG 0S031 PROGRAM CHECK INTERRUPTION – HEX LOCATION 0610F8 –
          CONDITION CODE 2 – SPECIFICATION EXCEPTION
   0S001 JOB NO NAME CANCELED
```

The program is automatically canceled by the supervisor and depending on the use of the job control statement, // OPTION DUMP, a dump of the partition and supervisor is executed.

Check 1
Message on SYSLOG

Did you understand the message and respond to it correctly? If no message was printed, go to check 2.

- Yes → Save SYSLOG output, and go to check 2.
- No → Contact your shift manager before rerunning or restarting the job/program

Check 2
Output from job/program (See note 3)

Is there unscheduled output?
Is scheduled output not as expected?

- Yes → Save the output, and go to check 3
- No → Make a note, and go to check 3.

Check 3
Input/output devices.

Are the assigned devices set up correctly? (e.g. bits per inch)

- No → Rerun or restart job/program.

Have any of the assigned devices a malfunction condition? (Some examples of device malfunctions are listed in Section 1.)

- Yes → Try to use a different device or consult the operating guide for the device, and inform your IBM customer engineer.
- No → Go to check 4.

Check 4
Input/output media

Are you using correct SYSRES pack?
input files/volumes?
work files/volumes?
Input/output devices?
teleprocessing lines?

- No → Contact your shift manager before rerunning or restarting the job/program
- Yes → Go to check 5.

Check 5
Program change during run

Have you altered the contents of any real or virtual address area since the job/program began running?

- Yes → Compare your notes of the alterations with the programmer's instructions. If there is a difference, try to contact the programmer before rerunning the job.
- No → Go to check 6.

Check 6
Programmer's instructions

Do you understand the programmer's instructions?
(If in doubt, try to contact the programmer.)

- No → Contact your shift manager before rerunning or restarting the job/program
- Yes → Go to check 7.

Check 7
Operating procedures

Are there any recent changes in the system operating procedures, and if so, do they affect this job?
(If in doubt, contact your shift manager or senior operator.)

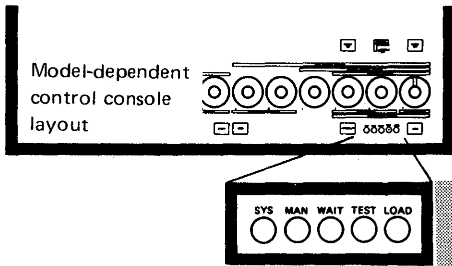
- Yes → Contact your shift manager before rerunning or restarting the job/program
- No → Go to check 8.

Check 8
Categorize the malfunction

- System in a WAIT STATE? (See note 1) → Turn to chart 03
- LOOP? (See note 2) → Turn to chart 04
- Obviously incorrect output? (See note 3) → Turn to chart 05
- Job/program canceled by system? (See note 4) → Turn to chart 06

Wait State

CHART 03, PART 1



Start

WAIT indicator ON
SYS indicator OFF

TEST indicator ON

Yes:

The TEST indicator is on when any of the following switches are not in the process or normal position.

1. RATE
2. CHECK CONTROL
3. DIAGNOSTIC/CONSOLE FILE CONTROL
4. ADDRESS COMPARE CONTROL

No:

MAN indicator ON

Yes:

The MANUAL indicator is on when the CPU clock is stopped or the system is in a soft-stop state. All pending interrupts are handled. Manual store/display operations are possible only when the MANUAL indicator is on.

No:

Device error recovery message printed on a line printer

Yes:

If SYSLOG is inoperative, a device error recovery message may have been printed on the line printer assigned as SYSLST.

Example of a device error recovery message

0P08A INTERV REQ SYSRDR=00C

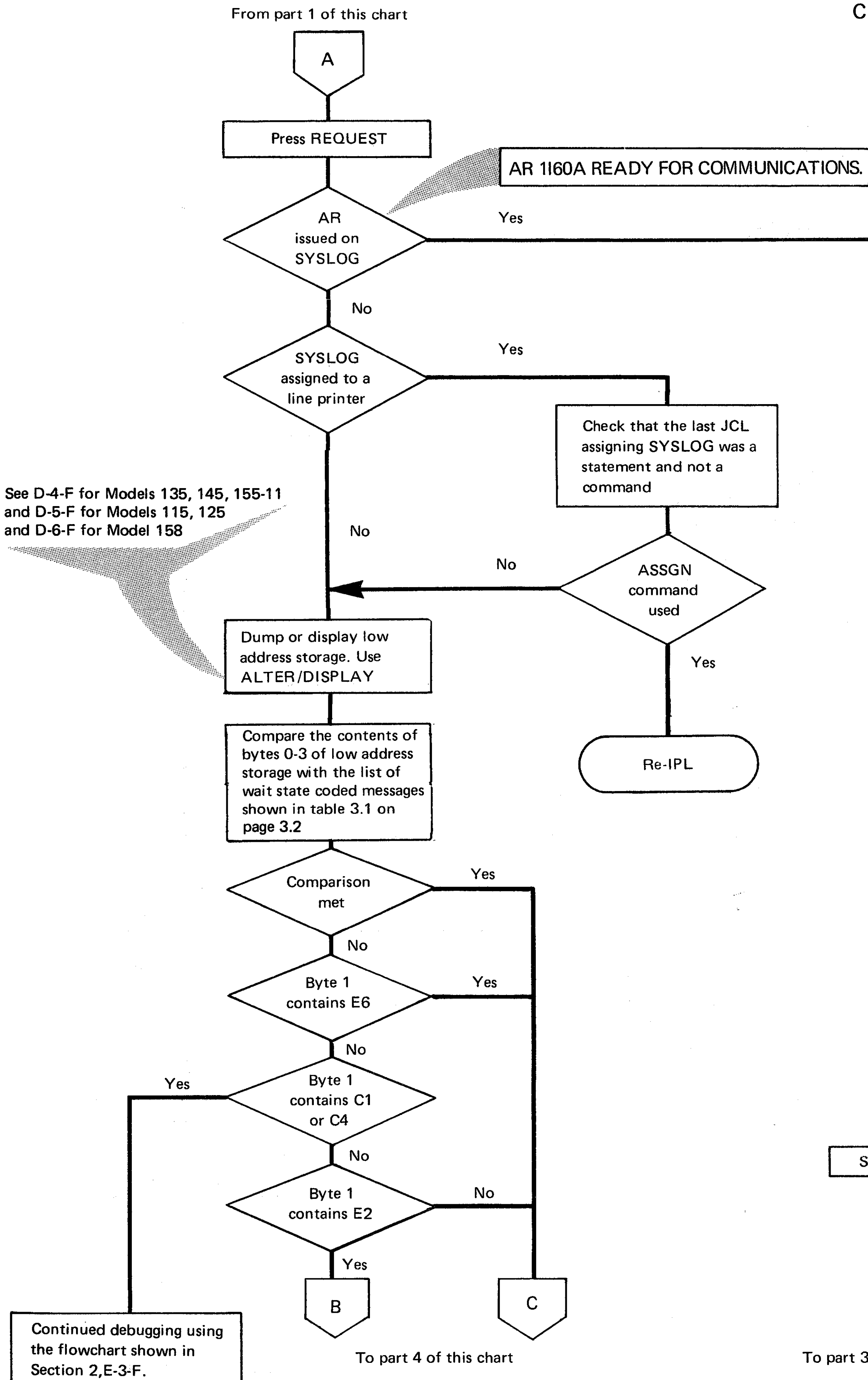
Consult *DOS/VS Messages* manual

No:

Inform your shift leader or manager of the condition, and request time for hands-on debugging to gather system information.

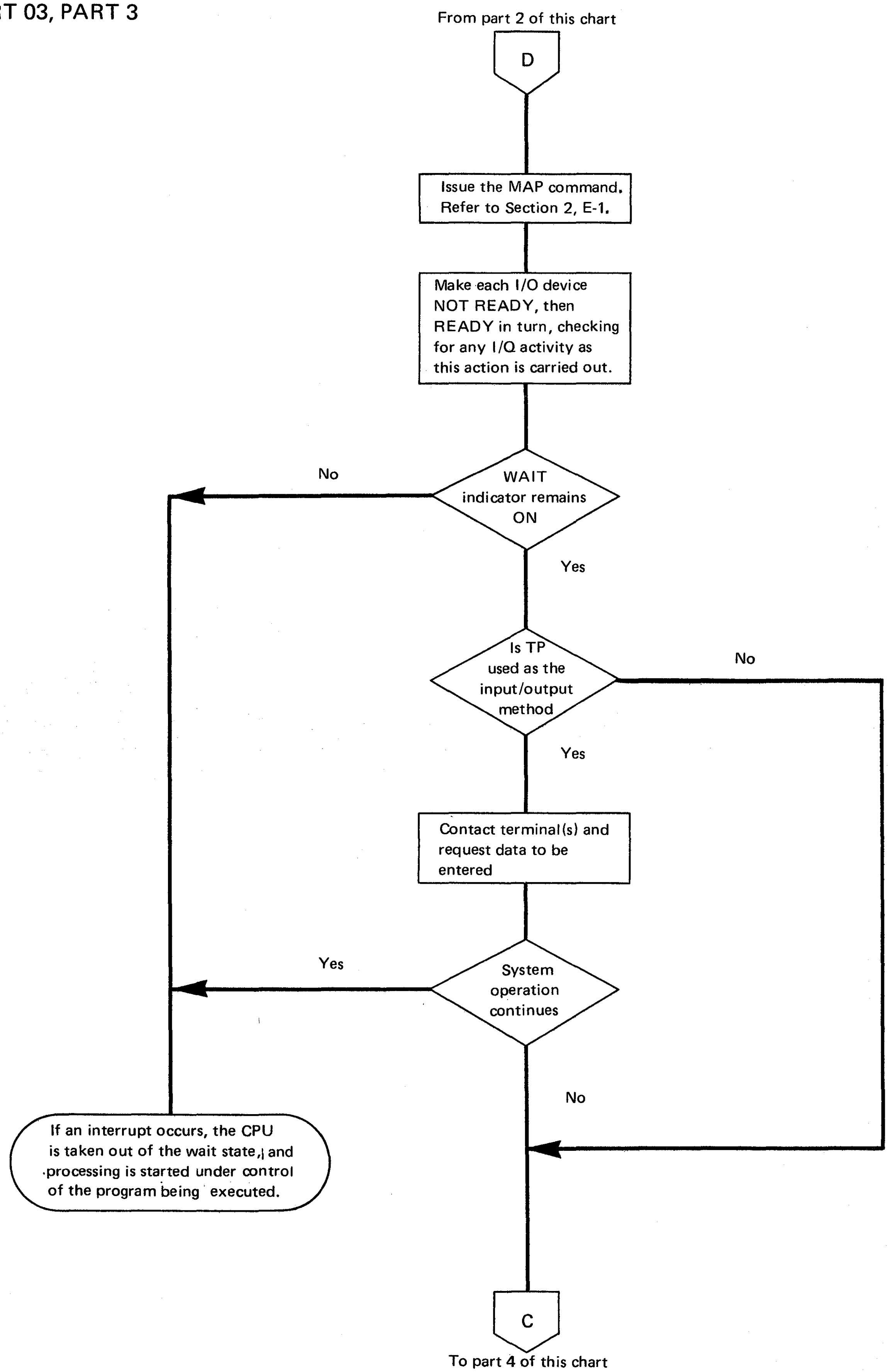
A

To part 2 of this chart

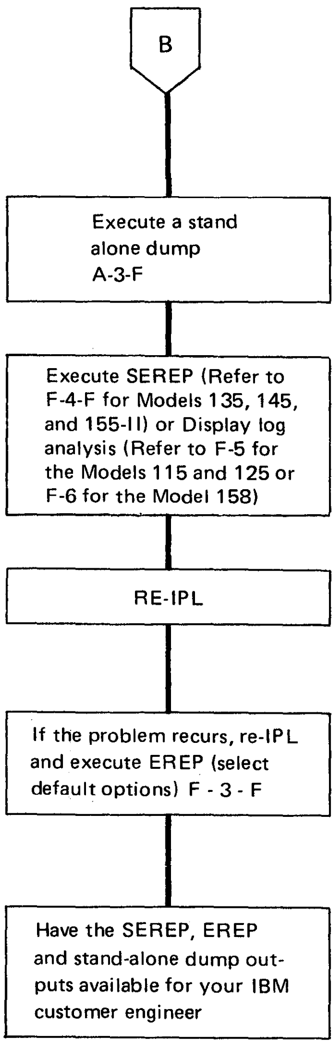

From part 1 of this chart
A
Press REQUEST
AR 1160A READY FOR COMMUNICATIONS.
AR issued on SYSLOG
Yes
No
SYSLOG assigned to a line printer
Yes
Check that the last JCL assigning SYSLOG was a statement and not a command
See D-4-F for Models 135, 145, 155-11 and D-5-F for Models 115, 125 and D-6-F for Model 158
No
No
ASSGN command used
Yes
Re-IPL
Dump or display low address storage. Use ALTER/DISPLAY
Compare the contents of bytes 0-3 of low address storage with the list of wait state coded messages shown in table 3.1 on page 3.2
Comparison met
Yes
No
Byte 1 contains E6
Yes
No
Byte 1 contains C1 or C4
Yes
No
Byte 1 contains E2
No
Yes
B
C
D
SOFT WAIT
Continued debugging using the flowchart shown in Section 2,E-3-F.
To part 4 of this chart
To part 3 of this chart

Wait State

CHART 03, PART 3



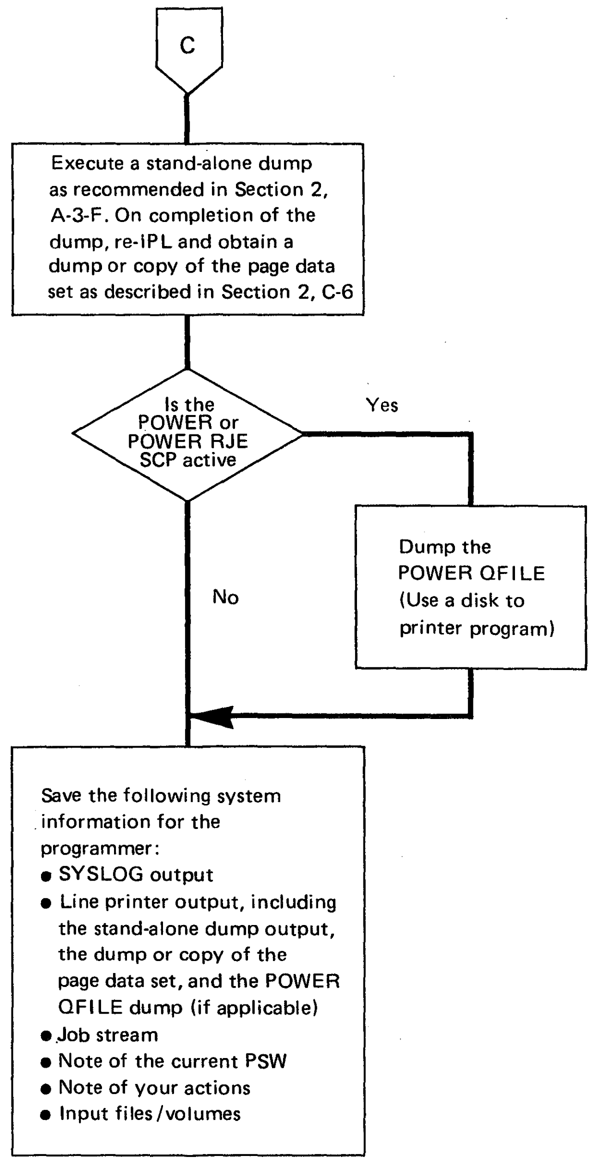
From part 2 of this chart



HARDWARE ERROR

Note: Certain unusual hardware and software failures can cause the system to halt processing with both the system light and the wait light on continuously. This indicates that the current PSW has its wait bit set on but the CPU is processing microprogram instructions. If possible, the system should be left in this state until a customer engineer has arrived. A stand-alone dump can show the I/O operations in process.

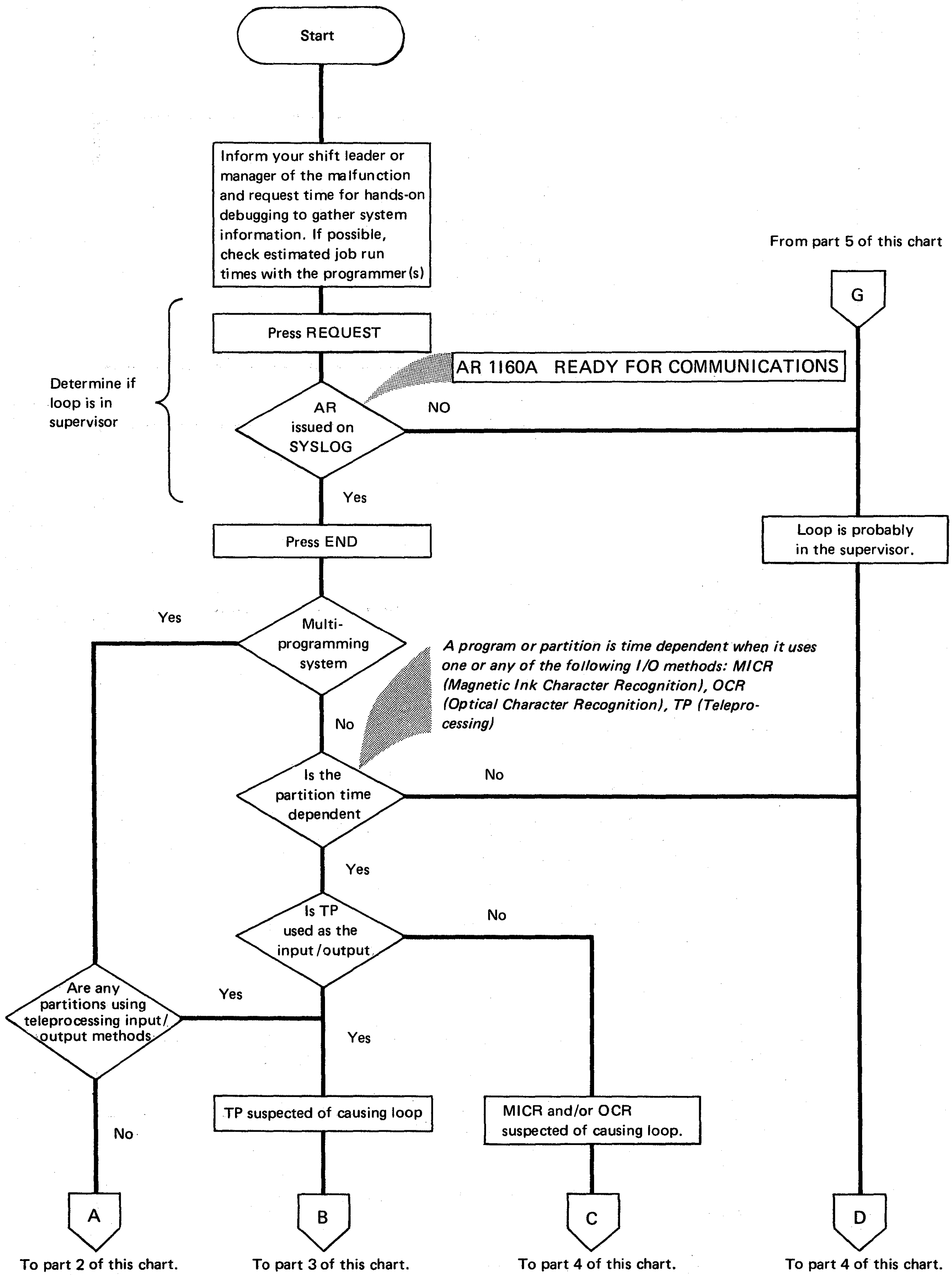
From part 2 and 3 of this chart.

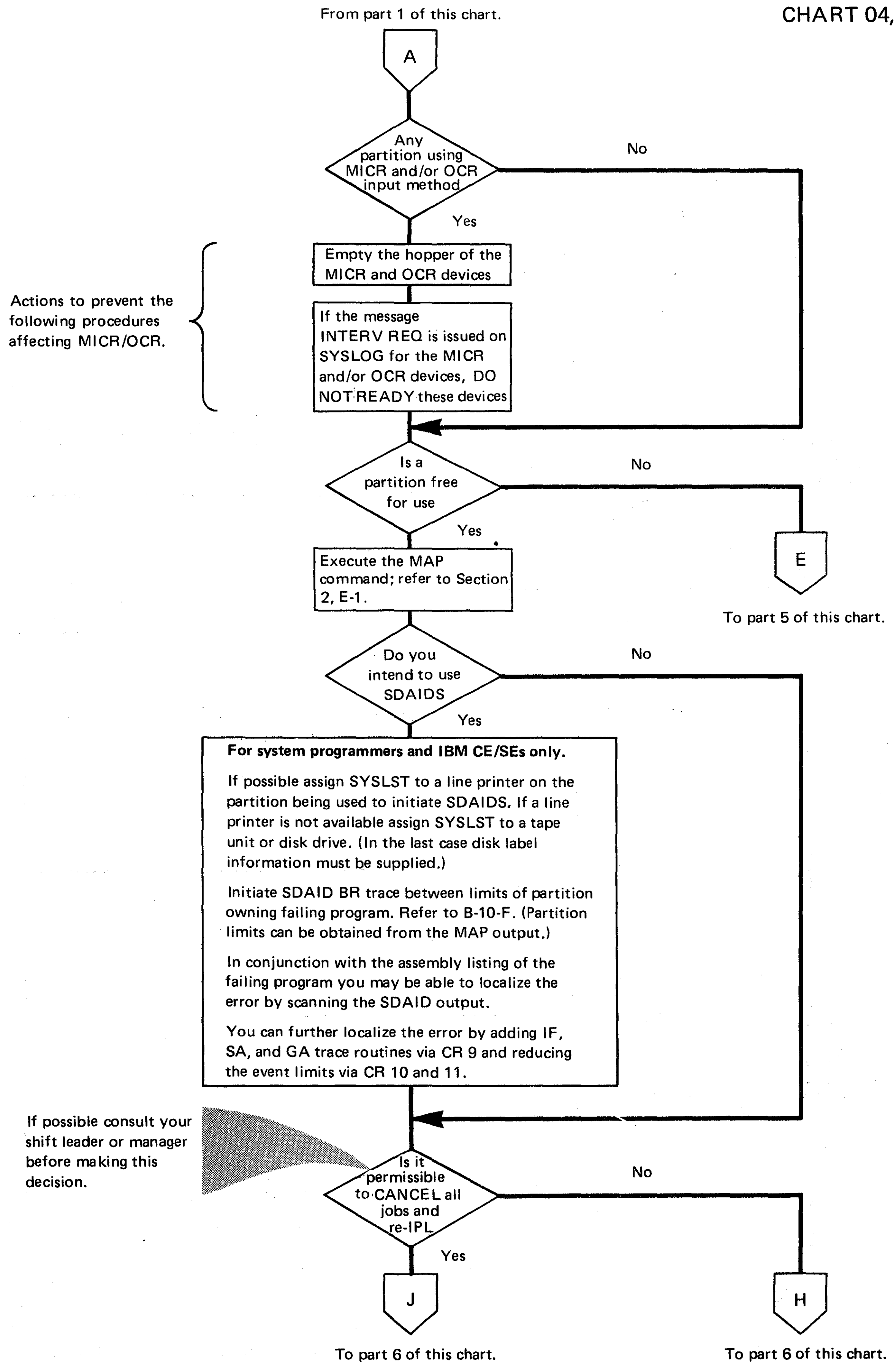


OPERATIONAL OR PROGRAMMING ERROR

Unintended Loop

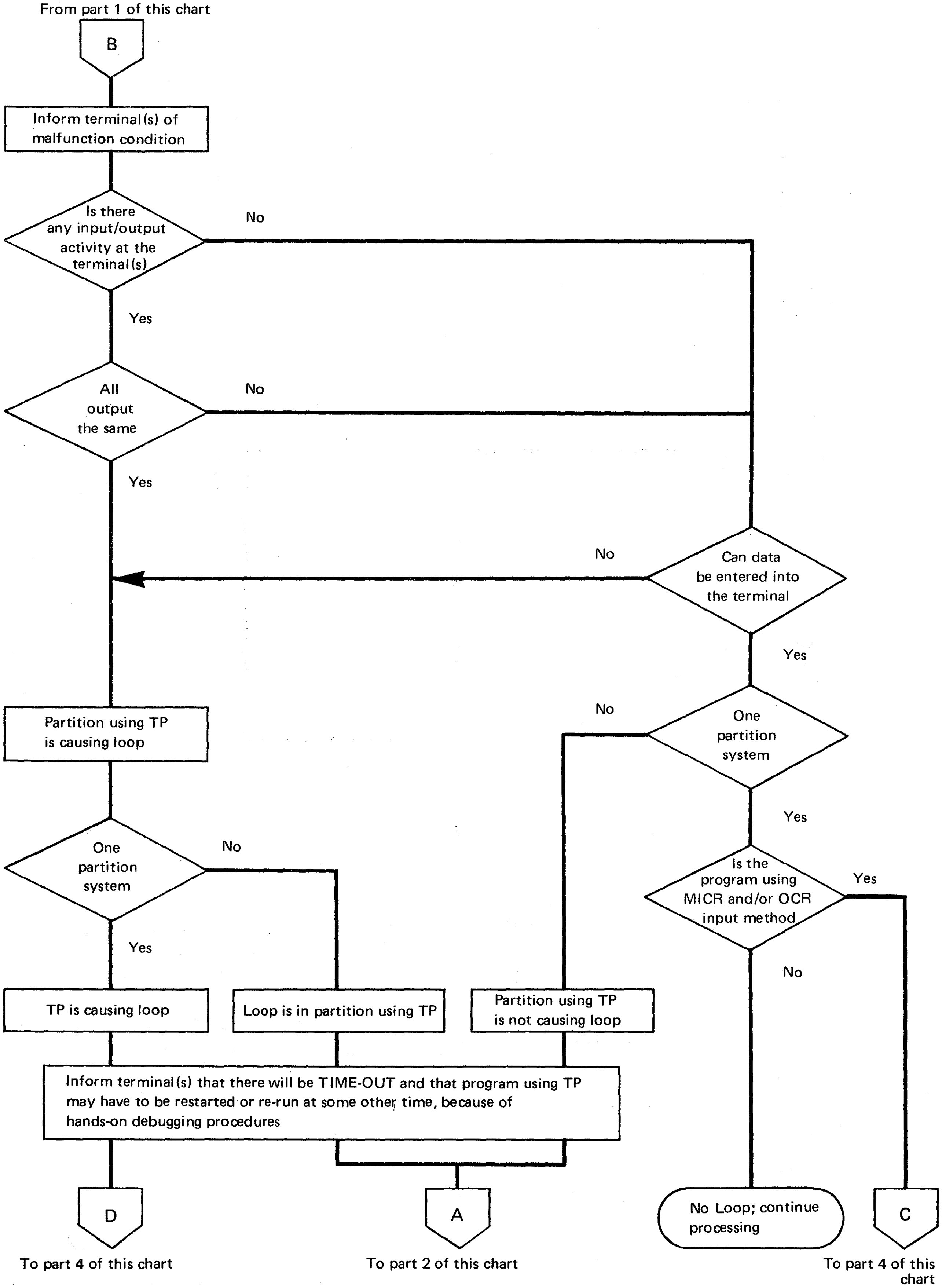
CHART 04, PART 1

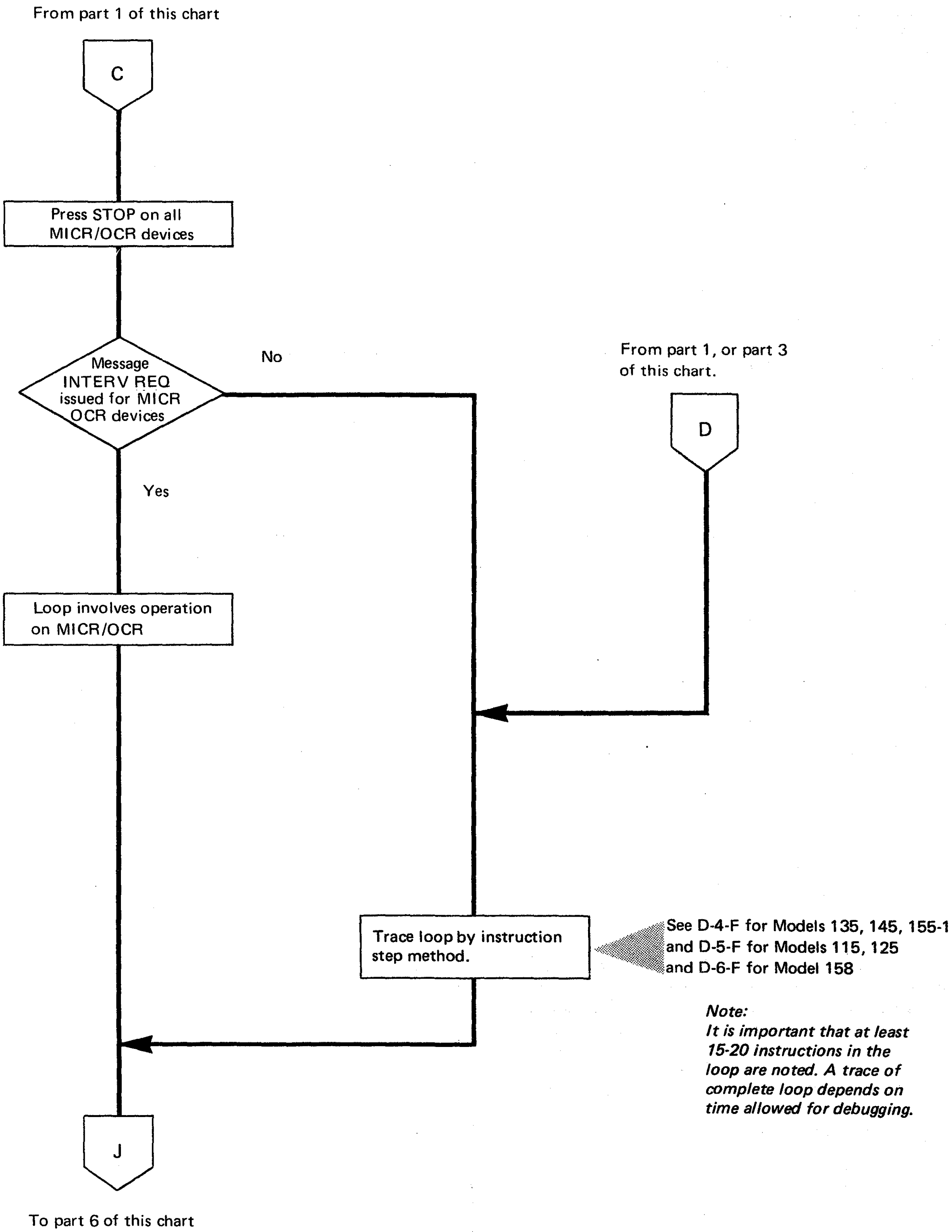



From part 1 of this chart.
A
Any partition using MICR and/or OCR input method
No
Yes
Actions to prevent the following procedures affecting MICR/OCR.
Empty the hopper of the MICR and OCR devices
If the message INTERV REQ is issued on SYSLOG for the MICR and/or OCR devices, DO NOT READY these devices
Is a partition free for use
No
Yes
E
To part 5 of this chart.
Execute the MAP command; refer to Section 2, E-1.
Do you intend to use SDAIDS
No
Yes
For system programmers and IBM CE/SEs only.
If possible assign SYSLST to a line printer on the partition being used to initiate SDAIDS. If a line printer is not available assign SYSLST to a tape unit or disk drive. (In the last case disk label information must be supplied.)
Initiate SDAID BR trace between limits of partition owning failing program. Refer to B-10-F. (Partition limits can be obtained from the MAP output.)
In conjunction with the assembly listing of the failing program you may be able to localize the error by scanning the SDAID output.
You can further localize the error by adding IF, SA, and GA trace routines via CR 9 and reducing the event limits via CR 10 and 11.
If possible consult your shift leader or manager before making this decision.
Is it permissible to CANCEL all jobs and re-IPL
No
Yes
J
H
To part 6 of this chart.
To part 6 of this chart.

Unintended Loop

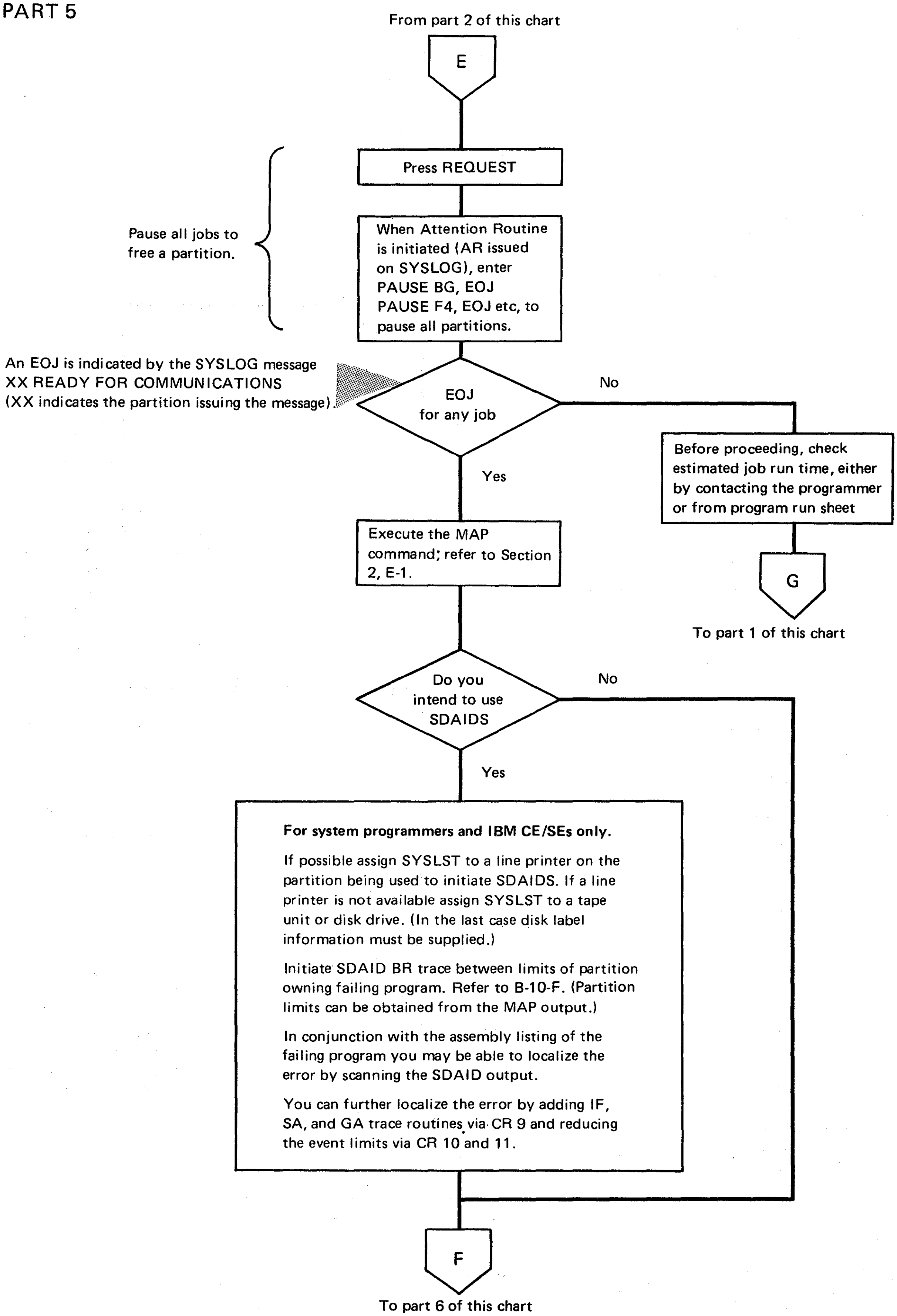
CHART 04, PART 3

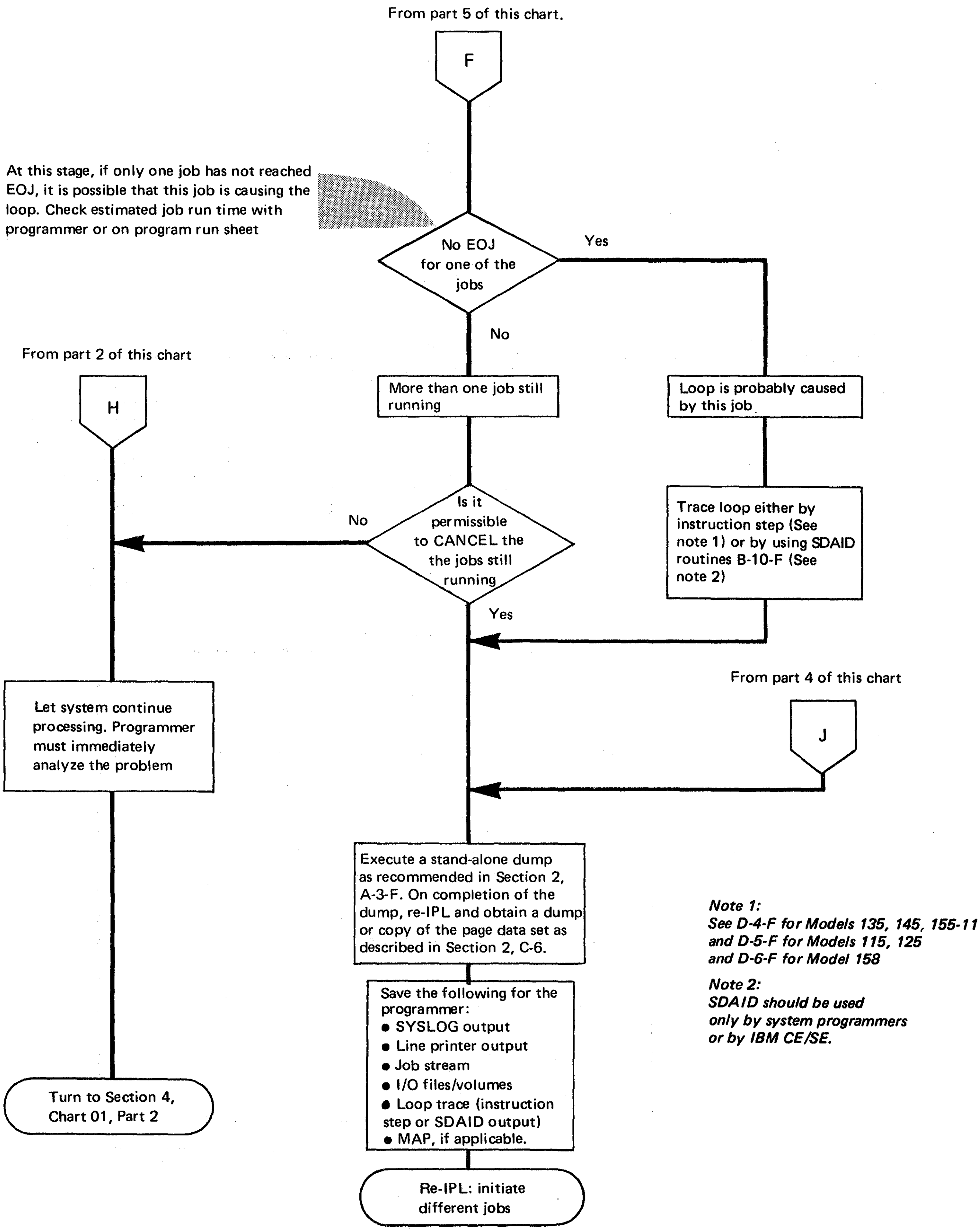



From part 1 of this chart
C
Press STOP on all MICR/OCR devices
Message INTERV REQ issued for MICR OCR devices
No
Yes
From part 1, or part 3 of this chart.
D
Loop involves operation on MICR/OCR
Trace loop by instruction step method.
See D-4-F for Models 135, 145, 155-11 and D-5-F for Models 115, 125 and D-6-F for Model 158
Note:
It is important that at least 15-20 instructions in the loop are noted. A trace of complete loop depends on time allowed for debugging.
J
To part 6 of this chart

Unintended Loop

CHART 04, PART 5




From part 5 of this chart.
F
At this stage, if only one job has not reached EOJ, it is possible that this job is causing the loop. Check estimated job run time with programmer or on program run sheet
No EOJ for one of the jobs
Yes
No
Loop is probably caused by this job
More than one job still running
From part 2 of this chart
H
Is it permissible to CANCEL the the jobs still running
No
Yes
Trace loop either by instruction step (See note 1) or by using SDAID routines B-10-F (See note 2)
Let system continue processing. Programmer must immediately analyze the problem
From part 4 of this chart
J
Execute a stand-alone dump as recommended in Section 2, A-3-F. On completion of the dump, re-IPL and obtain a dump or copy of the page data set as described in Section 2, C-6.
Save the following for the programmer:
• SYSLOG output
• Line printer output
• Job stream
• I/O files/volumes
• Loop trace (instruction step or SDAID output)
• MAP, if applicable.
Turn to Section 4, Chart 01, Part 2
Re-IPL: initiate different jobs
Note 1:
See D-4-F for Models 135, 145, 155-11 and D-5-F for Models 115, 125 and D-6-F for Model 158
Note 2:
SDAID should be used only by system programmers or by IBM CE/SE.

Obviously Incorrect Output

CHART 05

Start

The identity of the partition using the device:
This can be deduced from information given on the job run sheet, or the device ASSGN statements and commands issued on SYSLOG and SYSLST.
An alternative method is to make the device producing incorrect output NOT READY. This will cause a device error recovery message to be issued on SYSLOG.

Determine the partition using the device from the message printed on SYSLOG as a result of this action.

$\boxed{\text{BG}}$ 0P08A INTERV REQ SYS008 = $\boxed{383}$

Partition identity *I/O Device Address*

If possible, contact the programmer to establish the expected output, and identify which job or partition is using the device that is producing the incorrect output

Make the device producing incorrect output NOT READY

Press REQUEST and wait for message 0P60D, then enter IGNORE

AR 1160A READY FOR COMMUNICATIONS.

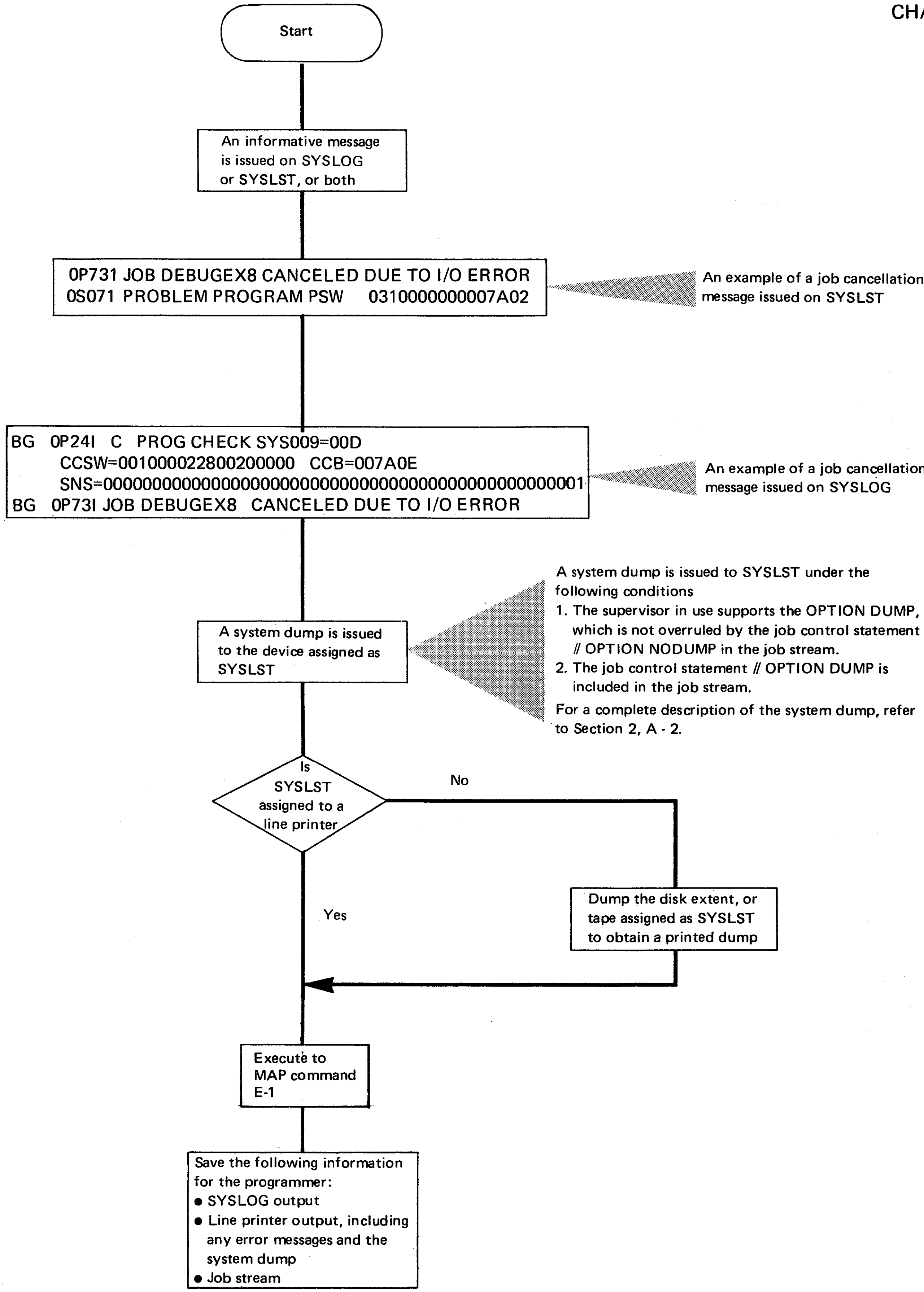
If SYSLST for the partition to be dumped is not assigned to a line printer, use the line printer assigned as SYSLST on another partition.
For example, DUMP F2S(BG).
Note: This may produce dump output from the partition being dumped, interspersed with output from the program using the line printer.

Respond to the AR message by issuing the DUMP command to dump the partition and the supervisor, for example, DUMP F2S
SYSLST must be assigned to a line printer.
Refer to Section 2, A-1

A System dump may be issued on the device assigned to SYSLST for the partition being canceled.
(The System dump is described in Section 2, A-2.)

Cancel the job in the partition producing the incorrect output

Save the following information for the programmer(s).
- SYSLOG output
- Line printer output, including all dump output
- Job stream
- I/O files/volumes


Start
An informative message is issued on SYSLOG or SYSLST, or both
0P731 JOB DEBUGEX8 CANCELED DUE TO I/O ERROR
0S071 PROBLEM PROGRAM PSW 0310000000007A02
An example of a job cancellation message issued on SYSLST
BG 0P24I C PROG CHECK SYS009=00D
CCSW=001000022800200000 CCB=007A0E
SNS=000000000000000000000000000000000000000000000001
BG 0P73I JOB DEBUGEX8 CANCELED DUE TO I/O ERROR
An example of a job cancellation message issued on SYSLOG
A system dump is issued to the device assigned as SYSLST
A system dump is issued to SYSLST under the following conditions
1. The supervisor in use supports the OPTION DUMP, which is not overruled by the job control statement // OPTION NODUMP in the job stream.
2. The job control statement // OPTION DUMP is included in the job stream.
For a complete description of the system dump, refer to Section 2, A - 2.
Is SYSLST assigned to a line printer
No
Dump the disk extent, or tape assigned as SYSLST to obtain a printed dump
Yes
Execute to MAP command E-1
Save the following information for the programmer:
• SYSLOG output
• Line printer output, including any error messages and the system dump
• Job stream

For your notes

DEBUGGING FOR SYSTEM PROGRAMMERS

DEBUGGING PROCEDURES FOR SYSTEM PROGRAMMERS

How to use

The choice of serviceability aids and methods of off-line program debugging and of analyzing each programming error rests with the programmer. The flowcharts in this section, however, will help the programmer to choose the method best suited to the type of error. For efficient analysis of dumps, program output, and printouts, an understanding of DOS/VS information blocks and supervisor interface tables is required. This section describes how your programs, referred to as user programs, interface with the IBM System Control Programs (SCPs). It also illustrates the allocations of storage, program and supervisor save areas, and details the information contained in the interface tables useful for program debugging.

The debugging of user programs written in a high-level language or for use with teleprocessing are not discussed. However, the serviceability aids described in Section 2 and the operator procedures of Section 3 should be used initially to gather information from the system. Having obtained the information, the procedures in this Section can then be used inconjunction with the debugging procedures described in the applicable high level language or teleprocessing component manuals.

This Section is divided into two parts. The first part consists of checklists in the form of flowcharts that should be used as a guide during offline program debugging. They help in selecting a method of analysis, and if required, help in the choice of the serviceability aid for further error isolation.

The second part of this section consists of a general description of the DOS/VS supervisor tables, information blocks, and save areas, together with other system information useful for offline debugging. More details about these tables and areas can be found in the IBM publication *DOS/VS Supervisor Logic.*

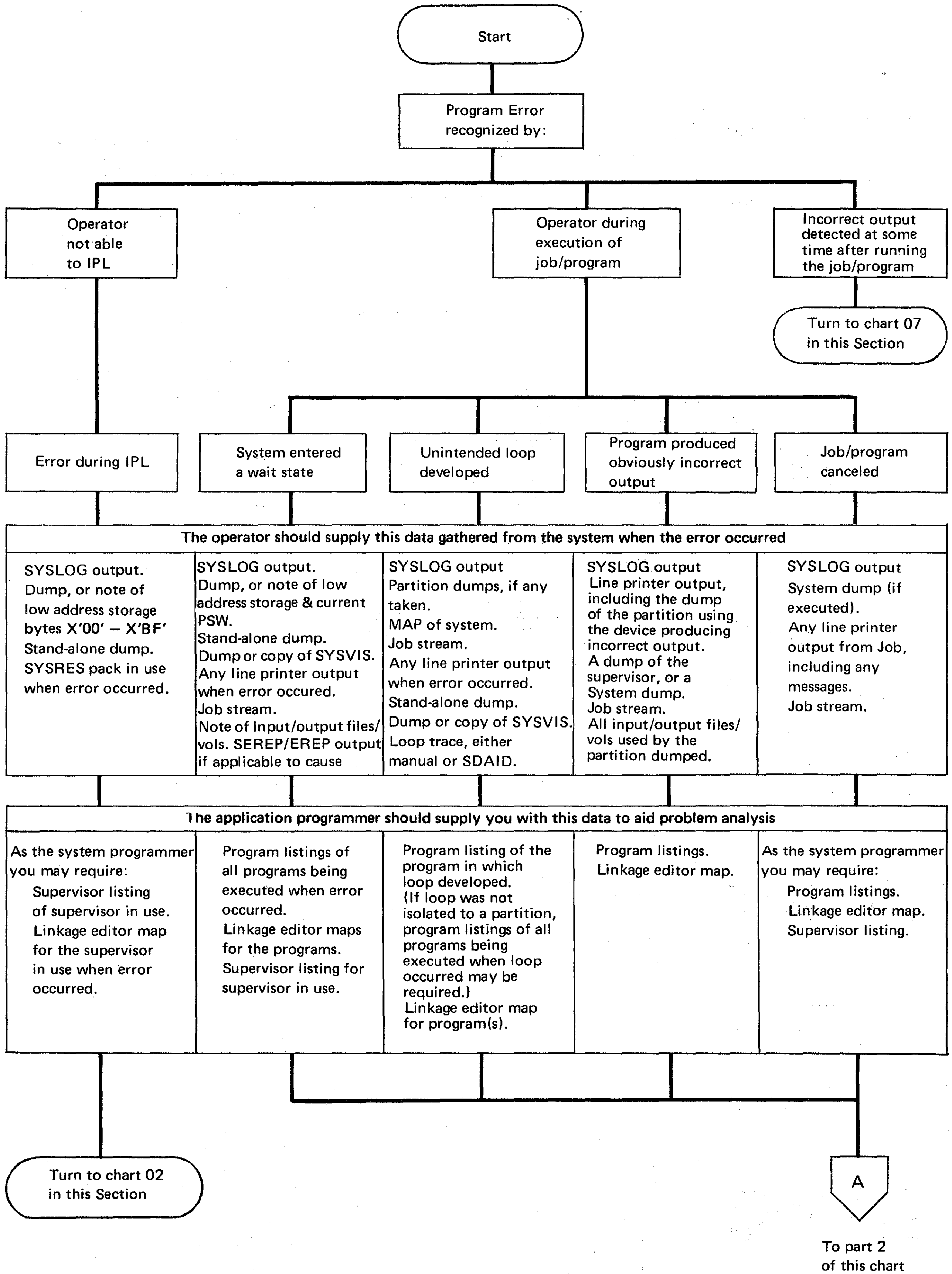
Note: It is assumed that the programmer using this section is familiar with DOS/VS multiprogramming, asynchronous processing (multitasking), relocating load, virtual storage, and data management techniques. These techniques are described in the DOS/VS System Management Guide.

CONTENTS

Flowcharts for offline debugging

Initial Checks

CHART 01, PART 1 OF 2



From part 1 of this chart

A

Check 1 — Error messages
→ Look for messages on console printer log sheet, or line printer output associated with the job in error.
→ Consult "Programmer Action" listed in *DOS/VS Messages.* If solution to problem not found, return to check 2.

Check 2 — Program execution
→ Has the program ever run successfully?
- No → Go to check 9
- Yes → Go to check 3 and continue through this checklist. It is recommended to make a copy of this chart and make notes (for future reference) in this column of any differences from previous execution of the program.
At this stage there is no need for further action, unless the solution to the problem becomes obvious.

Check 3 — Program, or module (sub-routine) coding
- Has there been a reassembly of the program, or any module used by it, since the previous run?
- Has the program or any module used by it been recataloged since previous run? (For example, a change in LOAD or ENTRY addresses.)
- Has any coding been changed by REP or SOURCE cards?
- Has an IBM programming temporary fix (PTF) been applied to the program, or any of its modules?

Check 4 — Input media
- Has there been any change in the location of data on any of the input files? (For example, the area containing the data on a disk pack has been moved to another address either on the same pack, or on a different pack.)
- Has there been a change in any input media? (For example, input data was previously on cards, but has since been transfered to disk pack.)

Check 5 — Input/output device and type
→ Was a different physical input/output device type assigned? (For example, in the previous run, I/O device was a 3330 disk drive, but is now a tape unit.)

Check 6 — DOS/VS level of release
- Did the program run previously under the same supervisor DOS/VS release?
- Was the same supervisor used for both runs?
- Have there been any changes to the supervisor, since the previous run? (For example, re-system generation)

Check 7 — Job control statements
→ Has there been any real change in job control since previous run? (For example, a change in the // EXEC parameter to REAL from default.)

Check 8 — System hardware configuration
→ Was the program executed on a different system configuration from the previous run? (For example, a different model of CPU.)

Check 9 — Categorize error

Program error recognized by:

WAIT state	Turn to chart 03
LOOP	Turn to chart 06
Obvious incorrect output	Turn ot chart 08
Job/program canceled	Turn to chart 09

Intentionally Blank

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Supervisor listing	If the IPL procedure can not be completed and the supervisor in use has never been used on the system, check the supervisor assembly listing for flagged statements. Also note in the listing the options supported by the supervisor. Check that the hardware is present in the system for options supported by the supervisor that require hardware support. Also check that the supervisor is generated to support hardware options that are present on the system.	If the cause of the error can not be isolated, inform IBM and have the following information available: ● The supervisor listing ● The stand-alone dump output, including the dump of the low address storage ● SYSVIS dump, or copy of the PDS ● The SYSRES pack used when the error occurred ● Any notes about actions taken during error analysis both on and offline.

Hard Wait Message Codes

SUPPORT FOR CHART 03

BYTE 0	BYTE 1	BYTE 2	BYTE 3	EXPLANATION
MCH/CCH/IPL Hard Wait Codes placed in low address storage				
X'C1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable machine check.
X'C2'	X'E2'(2)	Not used	Not used	Irrecoverable channel failure during RMS fetch.
X'C3'	X'E2'(2)	A, I, S(1)	Not used	Channel failure on SYSLOG when RMS message scheduled.
X'C4'	X'E2'(2)	A, I, S(1)	Not used	No ECSW stored
X'C5'	X'E2'(2)	A, I, S(1)	Not used	Channel failure: ERPBs exhausted.
X'C6'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; two channels damaged or a damaged channel situation occurred while RMS was executing an I/O operation.
X'C7'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system reset was presented by a channel.
X'C8'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system codes in ECSW are invalid.
X'C9'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; channel address invalid.
X'D1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable channel failure on SYSVIS.
X'07'	X'E6'	Channel	Unit or X'00'	IPL I/O error or equipment malfunction; condition code 2 during STIDC instruction. Channel and unit indicate whether device in error is SYSRES or communication device. When byte 3 = X'00', byte 2 indicates the channel for which STIDC instruction was issued. Re-IPL system.
SDAID Hard Wait Code				
X'61'	X'E6'(3)	Channel	Unit	Another device is running in burst mode on same channel as SDAID output device. Re-IPL system.
SDAID Soft Wait Code				
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'(3)	X'00'	X'00'	SDAID Stop on Event. Press EXTERNAL INTERRUPT key to continue operations.
The following Hard Wait Codes are placed in general register 11 X'B' as well as in low address storage.				
X'00'	X'00'	X'0F'	X'FF'	Program Check in Supervisor.
X'00'	X'00'	X'0F'	X'FE'	I/O error during fetch from System CIL.
X'00'	X'00'	X'0F'	X'FD'	Channel Failure if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
X'00'	X'00'	X'0F'	X'FC'	Machine Check if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
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'FA'	Translation Specification Exception
X'00'	X'00'	X'0F'	X'F9'	Error on Paging I/O.
X'00'	X'00'	X'0F'	X'F8'	CRT phase not found.
X'00'	X'00'	X'0F'	X'F7'	No copy blocks available for BTAM appendage I/O request.
X'00'	X'00'	X'0F'	X'F6'	$MAINDR canceled during system CIL update. If this occurs, the system CIL is only partially updated and must be restored before use. This hard wait condition can also occur if the FETCH QUEUE BIT (FCHQ) is set in the linkage control byte in the partition communication region owned by the terminating partition.
Device Error Recovery Soft Wait Codes placed in low address storage.				
X'08' to	X'C1' or	Channel	Unit	Error recovery messages. Refer to OP messages in *DOS/VS Messages.*

Notes: 1. A (X'C1') = SYSREC recording unsuccessful.
I (X'C9') = SYSREC recording incomplete.
S (X'E2') = SYSREC recording successful.
2. S (X'E2') = Run SEREP.
3. SDAID wait states are identified by X'EEEE' in the address part of the wait PSW.

Refer to Section 2-E-3 for a list of IPL error message codes and a more detailed description of wait states.

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Eliminate components	Was the POWER or POWER RJE system control program active when the malfunction occured?	Yes: Re-run the program(s) without initiating POWER. (Run the same job(s) under direct control of DOS/VS.) When running under POWER RJE you will have to submit the job to the central station and request it to be run without POWER RJE. If the identical error occurs go to check 3. If the error does not occur you may suspect the POWER SCP of causing the error. Before calling IBM turn to check 14 of chart 08, part 5.
		No: Go to check 2.
Check 2 WAIT STATE occurs when programs run under DOS/VS without POWER	HARD WAIT STATE	Proceed to check 3.
	SOFT WAIT STATE (You may require an interview with the operator to verify if this type of wait occurred, and request information about system occurrence and messages printed prior to the wait state.)	Turn to chart 05.
Check 3 Analyze message in low address storage	Use the dump, or notes made by the operator, of low address storage and current PSW, and the list shown on the opposite page. Also, refer to section 2-E-3. Analyze the message in bytes 0–3 to isolate a possible cause of the error.	If GR 11 contains a hard wait code (see table opposite) or a hard wait code exists in bytes 2 and 3 of low address storage, proceed to check 4. If there is no recognizable message turn to chart 04.
Check 4 Locate old PSW	Use the stand-alone output to locate: ● the old PSW applicable to the message. ● the instruction length code (Refer to section 2-E-2.) ● the interrupt code. The reason for the error may be that a transient used by the program is not cataloged in the system core image library. Execute DSERV. (Refer to section 2-C-2.)	Calculate the failing instruction address by subtracting the length code from the instruction address part of the PSW. Analyze the cause of the program check using the interrupt code. If you cannot isolate the error with this information proceed to check 5.
Check 5 Locate failing instruction address	Use the stand-alone dump output and supervisor listing to locate where in the supervisor the program check occurred (which routine). Analyze the failing instruction and compare the code of the instruction in the dump with the instruction in the listing.	Make certain that user routines have not altered the supervisor, and that the supervisor assembly is free of errors. Before calling for IBM programming support, execute the program with PDAID F/L trace on all partitions to trace phases used. (Refer to section 2-B-2.) Also, use the SDAID stop on event program check and obtain a non-destroying dump when the stop occurs. (Refer to section 2-B-10.) If the error cannot be isolated after re-analysis of the problem using the output obtained from the re-run, inform IBM. Have the following information available: ● SYSLOG output ● Supervisor listing ● Program listing ● Linkage Editor maps ● Stand-alone dump ● Dump or copy of SYSVIS ● PDAID and SDAID output ● Job stream ● DSERV output if one was obtained.

Hard Wait, no Message in Low Address Storage

CHART 04, PART 1 OF 1

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Scan the stand-alone dump output	A. Look for unexpected or unreasonable values in the following areas in the dump output: ● PSWs in low address area (machine check old, program old, input/output old, SVC old, external old), and note interruption codes for the associated PSWs. ● General purpose registers. ● Control registers (see appendix C). ● CSW and CAW (refer to section 2-E-2). B. If the dump produces a formatted output, scan the LUB, PUB, CHANQ, PIB, and ERROR Q. (Refer to chapter 3 in this section for a description of these supervisor tables.) C. Locate SYSCOM (address of SYSCOM is found at location X'80-83' of low address storage). D. Locate the logical transient area. The address of the LTA is found at X'1C' of SYSCOM. Convert the first eight bytes of the LTA to characters. This is the name of the transient in the LTA at the time the dump was taken. If it is a user-written transient, obtain a listing of the transient and check for the use of SVC 22 (Seize System). When the system is seized, no interrupts can occur until a second SVC 22 is issued to release the system. If the transient is supplied by IBM, inform IBM and have the following available: Supervisor Listing, Stand-alone dump, Jobstream, Program Listing, Linkage Editor maps, and SYSLOG output.	● If the error cannot be isolated, re-run the program with SDAID IF trace function. Use FASTREC output class and specify as large an area as possible for SDAID (refer to section 2-B-10). Proceed to check 2.
Check 2 Analyze the SDAID output	Use the SDAID IF trace output to determine where the LPSW (load program status word) was given. The HARD WAIT is caused by a LPSW instruction that can be issued only when the system is in the supervisor state. For example, LPSW can be issued by the DOS/VS supervisor or any IBM routines that run in supervisor state.	Compare the last few instruction addresses in the IF trace output with, for example, the supervisor listing, or if the instruction addresses are not in the supervisor area, the listing of the IBM routine which was running in supervisor state. Determine the routine in which the LPSW is issued. Inform IBM and have the following information available: ● SYSLOG output ● List of the IBM routine in which the LPSW instruction is issued, for example, the supervisor listing ● Stand-alone dump output ● SYSVIS dump or a copy of the PDS on tape or disk ● SDAID IF trace output.

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Establish task(s) waiting for input/output interrupts	Use the stand-alone dump output to locate the PIB. Make a note of the flag byte and address of the partition save area for each entry in the PIB. (How to locate the PIB and analyze information contained in an entry is described in Chapter 7 in this Section.)	Proceed to check 2.
Check 2 Determine status of the partitions	A. Determine the status of each partition by analyzing the flag byte. For example, a flag byte containing X'82' indicates that the task is waiting for an input/output interrupt. B. Locate the partition save area of each partition and make a note of the return PSW in each of the save areas. (Refer to chapter 11 in this Section.)	Use the instruction address part of each return PSW to locate the instruction (in the dump) which will be executed when the partition regains control. If the sequence of instructions in the dump starting at this instruction is TM, BO, SVC 7, proceed to check 3. Otherwise go to check 4.
Check 3 Analyze the contents of general purpose register 1	Determine if the contents of general purpose register 1 in the partition save area is pointing to a CCB. An example of a valid CCB is shown in Chapter 5 in this Section.	If GR 1 is not pointing to a valid CCB proceed to check 4. If the contents of GR1 is pointing to a valid CCB, analyze the contents of bytes 2, 3, and 12 of the CCB. (Refer to Chapter 5 in this Section for a description of the format and contents of the CCB). If the CCB is posted, check the seek address and unit check bits in the CCB. If you are using user-written unit check procedures, check that the sense data is properly tested. If the CCB is not posted, the device using this CCB is waiting for an input/output interrupt. Go to check 5.
Check 4 Analyze the program or programs running when the wait state occurred.	Use the program listing and linkage editor map* for each of the programs that were running, and, in conjunction with the instruction address part of the return PSWs in the associated save areas, locate the instruction in each of the programs.	Determine the routine that was being executed in each of the programs running when the wait state occurred. Check the instruction coding in each routine for the correct use of macros, for example, the use of ENQ, DEQ, and WAITM macros. (A task may be waiting for a particular resource, or there may be a programmed WAIT for a timer control block.) If the track hold facility is supported, check that the maximum number of tracks specified in the TRKHLD parameter is not exceeded. Also ensure that one task is not waiting for tracks held by another task. For details about these macros and their format refer to the IBM publication *DOS/VS Supervisor and I/O Macros,* and for examples of the use of these macros and multitasking, refer to the *DOS/VS System Management Guide.* If the error can not be isolated, proceed to Check 5.

**If the program has been link-edited using the DOS/VS relocating loader option, the relocation factor must be calculated based on the start address of the partition used for the program when it failed.*

CHART 05, PART 2 OF 2

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 5 Analyze the last START I/O	Inspect the channel bucket and analyze the contents to obtain information about the last input/output operation started on each channel. (Refer to Chapter 3 in this Section.)	Proceed to check 6.
Check 6 Determine the I/O device involved	A. Locate the associated PUB entry to determine the device address. (Refer to Chapter 3 in this Section.) B. Lo cate the associated LUB entry, (via the JIB if necessary), to determine the logical unit involved. (Refer to Chapter 3 in this Section.)	Proceed to check 7.
Check 7 Establish if there are other requests for the same device or if the CHANQ is full	● Locate the channel queue, and check if any other input/output requests are queued for the same device. ● Check the contents of FLPTR, if it is X'FF' the CHANQ is full. This may be an indication that the CHANQ is too small for the type of programs in use. (Refer to Chapter 3 in this Section.)	Proceed to check 8.
Check 8 Inspect the ERROR QUEUE	Locate the error recovery block and check if there are any entries in the error queue for the device. From the error queue entry you can obtain the CSW, CCB address, and Message Code. Use the message code to locate the meaning and possible cause of the message by referring to the *DOS/VS Messages* manual. (Refer to Chapter 3 in this Section.)	If the error can not be isolated, inform IBM of the problem, and have the following information available: ● SYSLOG output ● Stand-alone dump ● SYSVIS dump or copy of the PDS on tape or disk ● Job stream ● Description of system environment and conditions under which the program is executed. Otherwise make corrections to your program as necessary. (This may require a new assembly of the supervisor, or re-running the program using a supervisor that can accommodate the required input/output operations.)

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Approximate location of the loop	Use the trace of the loop (SDAID BR trace output or manual trace) and any notes made by the operator to inspect the path made by the loop. Also, use the linkage editor map and stand-alone dump to check if the loop addresses are within a module used by the program.	If the loop is entirely within the supervisor area or in an IBM module used by the program or in an IBM-written program, inform IBM and have the following system information available: ● SYSLOG output ● Any output from the program ● Job stream ● Trace of the loop ● Stand-alone dump ● SYSVIS dump, or copy of the PDS ● Supervisor listing. If the loop is not in the supervisor, or only part of it is, proceed to check 2.
Check 2 Eliminate components	● Is the loop in a user written module?	No: Loop is caused by the program. Proceed to check 3.
	Yes: ● Are any other programs using the same module?	No: Suspect the module of causing the loop. Proceed to check 3.
	Yes: ● Did these programs that use this common module generate unintended loops?	No: Suspect the module of causing the loop. Proceed to check 3. Yes: Loop is very likely generated by the module. Proceed to check 3.
Check 3 Preparation for problem analysis	A. If the loop is in the program, use the program listing, the trace of the loop, stand-alone dump, and program logic charts for analysis. B. If the module is suspected of causing the loop, use the listing and logic charts of the module for analysis.	Proceed to check 4.
	Also, check if any changes have been applied to the module since the last time the program ran successfully.	Remove the changes and re-run the program. Otherwise, proceed to check 4.
Check 4 Program and problem analysis	A. Check program listing and logic flow for: ● Count locations that can never reach their limits. ● Counters that are not updated. ● Counters that are never initiated. ● Count locations, the contents of which are not periodically or properly tested. ● Check that I/O operations are able to reach their completion, and are not prevented from doing so by program logic. B. Check the counter locations in the stand-alone dump. They may have been overwritten, in which case use the program listing to isolate the point in the program where the locations are prepared. Note: Examine the channel programs if I/O operations are directly responsible for preparing count locations.	If error cannot be isolated, re-run the program using SDAID BR and/or IF trace functions to obtain a complete trace of program and module logic flow during program execution. (Refer to Section 2-B-10.) Also, use PDAID F/L trace to check phases used by the program. (Refer to Section 2-B-4.) Re-analyze the problem. If the error can not be isolated, inform IBM. Have the following system information available: ● SDAID output ● SYSLOG output ● Job stream ● Program listing ● Listing of module used (if module suspected) ● Trace of loop ● Stand-alone dump ● SYSVIS dump or copy of PDS on tape or disk ● RDAID output.

Incorrect Output not immediately detected

CHART 07, PART 1 OF 1

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Gather any information pertaining to program	A. Ascertain System Environment (Supervisor, DOS/VS release number, I/O devices used, modules used, etc.) B. Obtain SYSLOG output from the job, and if possible the input job stream and input files. (This depends on many circumstances, such as time involved, storage of output from previous runs, original input files already overwritten for other jobs etc.) C. Program listing and linkage editor map.	Turn to check 1 of chart 06 and analyze the problem using all available information. Try to simulate the same conditions for a re-run of the program. Use chart 06 for error isolation.
	D. Ascertain the status of the machine at the time the job ran. For example, find out if an I/O unit in use for the program developed an error and analyze its effects on the program.	Re-run program; execute a system dump or stand-alone dump when incorrect output is detected.

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Type of program producing the incorrect output	Userwritten?	Proceed to check 2.
	IBM-supplied program? Assembly language High level language compiler (RPG II, FORTRAN, etc.) IBM library program (DSERV, SORT/MERGE) etc.	Inform IBM and have the following information available: ● All output from the program ● SYSLOG output ● Job stream ● Input files ● Supervisor generation notes and level of release ● Listing of the support for the failing program ● Notes about system type and model, and any notes about system status and input/output units used for the program.
Check 2 Type of incorrect output	Use any output from the program and messages printed during program execution to determine the type of incorrect output according to the following categories: ● Duplicate (identical lines printed). ● Lack of (missing data printed). ● Too much or unexpected (more data printed than expected).	Use the program listing and analyze the coding, checking for: ● Unnecessary program loops in input/output routines ● Routines not being entered or completed ● Overlaid input/output areas. If necessary, re-run the program using SDAID IF, SA, and GA trace functions. Specify event limits that will adequately cover routines that prepare output areas. (Refer to section 2-B-10.) Re-analyze the output from the re-run. If the error can not be isolated proceed to check 3.
Check 3 System condition during program execution	Were there any abnormal interruptions in computer operations during execution of the program?	Yes: Analyze the possible effects of any interruption, and re-run the program ensuring no interruptions.
	No: Were any abnormalities observed in system actions or response to messages by the operator during program execution?	Yes / No: Re-run the program and note any unexpected system actions. Ensure the correct response is given to messages issued. (Refer to the *DOS/VS Messages* manual). Use any notes made to analyze program logic. If necessary, re-run the program using with the SDAID IF trace to aid analysis of program logic. Check the input used for the program. Proceed to check 4.

Incorrect Output detected during Program execution

CHART 08, PART 2 OF 5

Check	*Method of Analysis*		*Recommendations to Aid Further Problem Analysis*
Check 4 Environment	Was the program operating under POWER or POWER RJE?	Yes	Turn to part 5 of this chart, check 13.
		No	Proceed to check 5.
Check 5 Job stream	Check that the correct job stream was used for the program. This applies also to job streams cataloged on the procedure library. (Use PSERV to display the procedure library; refer to section 2-C-3.)		If error isolated, make corrections as necessary and re-run the program. If error can not be isolated, proceed to check 6.
Check 6 Input used	A. Isolate suspected areas on the input files by examining the output from the program. For example, a missing record in the output indicates the approximate area on the input file that may be in error.		A. Obtain a dump of suspected areas on the input files used. For VSAM files use the Access Method Services utility program PRINT command. (Refer to section 2-C). B. Check input/output file macro definitions in the program listing against characteristics of data on the input files. DTF (LIOCS), CCB (PIOCS) ACB, RPL, and GENCB (VSAM).
	B. Check the labels and volume ID and extent information. Use LSERV and LVTOC, or for VSAM files use LISTCAT Access Method Services utility commands.		Compare the output of the LSERV, LVTOC, and LISTCAT programs with DLBL/EXTENT, VOL ID, data in the job stream. If no errors can be found, proceed to check 7.

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 7 Dump of partition	Did the operator obtain a dump of the partition containing the failing program? For example, a stand-alone dump, a system dump, or a dump obtained by using the DUMP command.	Yes: Inspect input/output areas in the dump, also inspect CCBs and channel programs. (Refer to chapter 5 in this section.) If no errors are found proceed to check 8, or if satisfied with the information that can be obtained from the dump go to check 10.
		No: Proceed to check 8.
Check 8 Obtain a non-destroying dump	Use the program listing and the information already obtained from the previous analysis of the problem to isolate the approximate area of the failure. For example, a specific input/output area, or a specific work area.	A. Re-run the program with SDAID IF and/or SA trace function. Specify a stop on event and obtain non-destroying dumps at specific instructions, or specific storage alterations. (Refer to section 2-B-10.) B. Also, as a further aid for problem analysis, use the PDAID F/L trace to trace phases used by the program. Proceed to check 9.
Check 9 Analyze the SDAID dump output	A. Inspect input/output areas, CCBs, and channel programs in the non-destroying dump output. B. Compare input/output areas as defined in the DTFs, CCB and CCW macros, and ACB, RPL and GENCB macros in the program listing, with the areas as printed in the dump. C. Check the sequence, names, and load/entry addresses of the phases used by the program, and compare them with the linkage editor map addresses. D. Use the program listing and dump output to inspect values in registers and storage locations used for intermediate results.	If no errors can be found and the cause of the error can not be isolated, proceed to check 10.

Incorrect Output detected during Program execution

CHART 08, PART 4 OF 5

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 10 Unusual conditions generated by the program and its input	Check that the program logic and counters can accommodate unusual conditions and circumstances of input. For example, data changes (year), or extraordinary changes in amounts and quantities.	If the error can not be isolated, proceed to check 11.
Check 11 Preparation of the input files	Check the input used for this program. If necessary, analyze the program that prepared the input files, starting at check 1 of this chart.	If the input is correct, and the program that generated the input is not at fault, proceed to check 12.
Check 12 Gather more system information	Re-run the program, using, if possible, newly prepared input data that is known to be correct. Try to simulate or run under identical conditions to those in which the failure occurred.	A. Re-run the program with the PDAID F/L trace, to trace phases used by the program. (Refer to Section 2-B-4.) B. If you have not carried out the recommendations listed in checks 8 and 9 of this chart, initiate the SDAID IF, and/or SA trace functions. Use specific addresses for the event limits that are related to the problem, based on your previous analysis. (Refer to Section 2-B-10.) C. Before re-running the program, re-assemble it using the PDUMP macro to dump all input/output areas before and after every input/output operation. (Refer to Section 2-A-5.) If the program is using VSAM files, re-assemble the program and insert the TESTCB or SHOWCB macros, before and after each OPEN, GET and PUT. (Refer to Chapter 4 in this Section.) If, after re-analysis of the problem using the output from the re-run, the error can not be isolated, inform IBM and have the following information available: ● All program output, whatever it is ● All dump output ● SDAID/PDAID output ● SYSLOG output ● Job stream ● Any notes pertaining to the problem ● Notes about system environment and condition.

Check | *Method of Analysis* | *Recommendations to Aid Further Problem Analysis*

Check 13

Eliminate components used.

- Without initiating a POWER partition or running under POWER RJE re-run the program producing the incorrect output.

If program still generates incorrect output, analyze the problem starting at check 1 of chart 06, part 1. Otherwise suspect the POWER or POWER RJE partition and proceed to check 14

- Are other programs generating incorrect output when running under POWER or POWER RJE

No

Yes: Proceed to check 14

Check 14

Programs falling which run under control of POWER or POWER RJE

Inspect the contents of general purpose register 7 printed at the beginning of a system dump or stand- alone dump executed just before the malfunction occurred. This register should contain the address of the TIB of the active POWER or POWER RJE routine just before the dump was executed.

If the program is run under POWER consult Appendix K, or if run under POWER RJE Appendix L.
By using the system information and Appendix K or L (as applicable), you may be able to isolate the cause of the error. In any case it is recommended that you inform IBM of the problem and have the following information available:

- SYSLOG output
- Stand-alone dump
- A dump, or copy of SYSVIS
- Dump of the QFILE
- Supervisor assembly listing
- POWER SCP assembly listing
- Job stream
- Any notes made during system operation when the malfunction occurred.

You may re-run the program under POWER or POWER RJE with the following aids.
A. Use PDAID I/O or QTAM trace functions to trace I/O operations on all units used by the program producing incorrect output, and by the partition owning POWER/POWER RJE. (Refer to section 2-B-4).
B. Use SDAID BR function on the partition owning POWER to trace the logic flow during program execution (Refer to section 2-B-10).
C. As an option, re-run the program using PDAID F/L trace on the partition owning POWER to trace phases used. (Refer to section 2-B-4).

Intentionally Blank

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Type of cancellation	Program Check.	Proceed to check 2.
	Illegal SVC.	Turn to chart 10.
	Other (for example, operator cancellation).	Turn to chart 11.
Check 2 Area of failing instruction (Level 1)	Is the program check in the SVA? There are two methods to establish this: 1. From the MAP command output check if the instruction address shown in the Program Check message is within the SVA. (The MAP command should have been issued immediately after the Program Check occurred) 2. If a System Dump was executed when the Program Check occurred and the Program Check was in the SVA, a dump of the SVA (or part of it) will be preceded by the following message printed in the dump output: PART OF SVA WHICH CAUSES ERROR	Yes: If a System Dump was executed, you may be able to isolate the error by using the information already available to you. If this is the case, go to check 13 on part 4 of this chart. If a System Dump was *not* executed, it will be necessary to re-run the job and include the // OPTION DUMP statement in the job stream before going to check 13 on part 4 of this chart. Also initiate the SDAID dump on event Program Check (refer to Section 2-B-10) and if your system suports PDAID, initiate a FETCH/LOAD trace to trace the phase being loaded in the SVA. (Refer to Section 2-B-4).
		No: Proceed to check 3
Check 3 Area of failing instruction (Level 2)	Use the load and entry addresses of phases and modules used by the program as indicated on the linkage editor map,* and the error message to locate in which area the program check occurred:	
	● Program check in user program area	Proceed to check 4.
	● Program check within the supervisor area (The address of the end of the supervisor is contained in bytes X'20'–X'23' of SYSCOM, and the address of SYSCOM is contained in bytes X'80'-X'83' of the low address storage.)	Turn to chart 12.
	● Program check within the partition containing the POWER or POWER RJE system control programs	Turn to chart 13.
Check 4 Location of failing instruction	Use the linkage editor map as in check 2 above, to determine whether: ● the program check is within the user-written program, ● or in a user-written module used by the program. Note: If the program check is within an IBM-supplied module used by the program, and no system dump was executed when the program check occurred, re-run the program with the // OPTION DUMP statement in the job stream. (Refer to Section 2-A-2) Otherwise inform IBM and have the following information available: ● SYSLOG output ● Job stream ● System dump output ● Program listing and linkage editor map.	Proceed to check 5.

**If the program has been link-edited using the DOS/VS relocating loader option, the relocation factor must be calculated based on the start address of the partition used for the program when it failed.*

Program canceled by Program Check

CHART 09, PART 2 OF 4

Check	*Method of Analysis*		*Recommendations to Aid Further Problem Analysis*
Check 5 Analyze failing instruction and type of program check	Can the cause of the failure be isolated using the program listing, error messages, and job stream?	Yes	Make correction as necessary and re-run the program.
		No	Proceed to check 5.
Check 6 Dump of partition available?	Was a system dump executed when the program check occurred? (System dump is described in Section 2-A-2).	Yes	Proceed to check 7.
		No	Re-run the program with // OPTION DUMP statement in the job stream. Also, use the SDAID stop-on-event program check and obtain a non-destroying dump. (Refer to Section 2-B-10.) When you have obtained the dumps continue problem analysis from check 7 of this chart.
Check 7 Analyze dump of partition (Level 1)	Use error messages, linkage editor map*, and program listing to locate failing instruction in the Dump. (An example is shown in Section 2-E-3.)		A. If the instruction and code appears correct and has a valid operation code and data field, proceed to check 8. B. If the program check message indicates an error in input/output operation in your program, go to check 10; otherwise proceed to check 8.
Check 8 Analyze dump of partition (Level 2)	A. Locate the partition save area and inspect the contents of general purpose registers. (The location and format of partition save areas are described in Chapter 11 of this Section.)		Look for unreasonable or totally unexpected values in the general purpose registers. If you find any that are connected to the failing instruction, locate the code in the program listing responsible for preparing the register(s). (General register usage is described in Chapter 9 of this Section.) If the cause of the error can not be isolated, proceed to check 9.
	B. Use program listing, and linkage editor map* to locate data areas in the dump used by the failing instruction.		Depending on the type of program check indicated by the error message, inspect data for validity, duplication, missing information, and sequence errors. If the cause of the error cannot be isolated, proceed to check 9.

**If the program has been link-edited using the DOS/VS relocating loader option, the relocation factor must be calculated based on the start address of the partition used for the program when it failed.*

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 9 Analyze the general register and data preparation instructions	Inspect the program logic and instructions responsible for preparing the general registers and data areas used by the failing instruction.	If an input/output operation is involved, proceed to check 10; otherwise, go to check 11.
Check 10 Analyze the input/output operation	Locate and inspect command control blocks (CCBs) and channel programs in the system dump as described in Chapter 5 in this Section. *Note: to locate and analyze CCBs and channel programs for programs using VSAM files, consult the description given in Chapter 4 in this Section.*	If it is an input operation, and this input is used to prepare the general registers or data fields used by the failing instruction, you may suspect the program that produced the input. Proceed to check 11, or first analyze the program that provided the input to this program by following the recommendations given in chart 08 (incorrect output).
Check 11 Prepare a program re-run to gather more detailed system information	A. Use SDAID SA and/or GA trace functions if you suspect that data of general register alteration is the cause of the error. Specify address limits on the suspected areas calculated from the start of the partition in which the program is re-run. Refer to Section 2-B-10.) *Note: the SA trace can not be used to trace areas altered directly by input/output operation.* If the logic flow of the program during execution is in doubt, use the SDAID BR trace function on the partition in which the program is re-run.	Proceed to check 12.
	B. If the program is running in virtual mode or using user written channel program translation routines, use SDAID stop-on-event program check and obtain a non-destroying dump. (See note) (Refer to Section 2-B-10.) C. If the program is running in virtual mode and using the IBM-supplied channel program translation routine, check that your program is not violating the rules for channel program translation support.	Check the general rules applicable to channel program translation routines. (Refer to Chapter 13 in this Section.)

Note: The SDAID non-destroying dump enables you to analyze the CCB/CCW copy blocks and the CCW/TCB in the supervisor area. Note that these blocks may be overwritten by the system dump when analyzing the output from a system dump.

Program canceled by Program Check

CHART 09, PART 4 OF 4

Check 12

Re-analyze the output from the program re-run

→

Use all the information obtained from the re-run and re-analyze the problem. If the error cannot be isolated, you may either inform IBM or re-assemble the program, inserting one or more of the following assembly macros in your program. Then re-run the program.

- PRINT GEN (to obtain an expansion of all macros in used by the program).
- PDUMP (to obtain the dumps of selected areas of storage, such as input/output areas, during program execution. Refer to Section 2-A-5)
- DUMP or JDUMP (to obtain a dump of partition and supervisor at a point in the program before the program check occurs. Refer to Section 2-A-5.)

→

If you choose to inform IBM, have the following information available:

- SYSLOG output
- Job stream
- SDAID output
- Program listing and linkage editor map.

→

Re-analyze the problem using output obtained from the re-run. Examine the expansions of macros used by the program and check the DTF macros used for file definition with the program listing.
If the error can not be isolated, inform IBM and have all information obtained from program re-runs, plus any notes made of your previous analysis.

Check 13

Analyze dump of the SVA

→

A. If it is known which program or phase in the SVA caused the program check, use error messages and program listing to locate the failing instruction in the dump. (An example is shown in Section 2-A-2 of this manual).

B. If the name of the phase in the SVA that caused the program check is not known, use the PDAID FETCH/LOAD trace output to locate the last phase used by the program.
The load address of the phase should be the start address of the SVA printed in the dump.

→

Compare the hex code of the failing instruction given in the assembly listing with the hex code in the dump output.
If the instruction and code appears to be correct and has valid operation code and data fields specified, continue problem analysis from check 8 on part 2 of this chart.
If the program check message indicates an error in I/O operations, continue problem analysis from check 10 on part 3 of this chart.

Intentionally Blank

The complete text for message 0S04I is:

ILLEGAL SVC – HEX LOCATION nnnnnn – SVC CODE nn

where nn is in hexadecimal notation.

This message results from the following causes:

1. When nn is 02: The phase name given does not start with $$B, or

   For LIOCS, macros called in invalid sequence. As a result, on SVC 8 is issued after an SVC 2 before an SVC 9 has been issued to free the transient area, or

   For other conditions, the user specified a temporary exit (SVC 8) for a logical transient. In the temporary exit routine, another routine is called (by an SVC 2) before an SVC 9 is issued to free the transient area.

2. When nn is 05: The 'to' range specified in the MVCOM macro is invalid.

3. When nn is 0A, 12, 13, or 18: The supervisor was generated without the timer option.

4. When nn is 0B: The call was not given by a logical transient routine.

5. When nn is 16, 17, or 1A: The caller did not have a PSW key of zero. This is applicable only in a multiprogramming system.

6. When nn is 23: More than 16 holds have been issued for the same track.

7. When nn is 24: Free a non-DASD or a track that is not held.

8. When nn is 26: A subtask issued attach, or the save area is not on a doubleword boundary.

9. When nn is 27: A main task issued detach without SAVE = parameter, or

   A main task issued detach, but the ID of the subtask in the save area passed is not valid, or

   A main task attempts to detach on already terminating subtask.

10. When nn is 29: A DEQ is issued by a task that did not ENQ the resource. (This is valid in an AB routine.)

11. When nn is 2A: A subtask (without an ECB = parameter) has issued an ENQ macro, or

    A subtask has issued an ENQ macro to a resource that has not been dequeued by another task that has been terminated, or

    A task has issued two ENQ macros to the same resource without an intervening DEQ.

12. When nn is 2D: Emulator execution was attempted, but the EU parameter of the SUPVR macro was omitted or incorrectly specified during system generation.

13. When nn is 32: For LIOCS:

    a. An imperative macro (such as WRITE or PUT) was issued to a module that does not contain the requested function, or

    b. A PUT was issued for an ISAM retrieve module without a preceding GET, or

    c. An invalid ASA first character for the printer was used, or

    d. A wrong length record indication occurred while processing 1287 documents when RECFORM=UNDEF, or

    e. The 1287 program erroneously contained a CCW(s) with the SLI flag bit 'OFF', or

14. When nn is any other value: The supervisor function requested by the operand of the SVC is not defined for the supervisor being used.

Causes for Message 0S04I (Cancel Code X'21')

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Supervisor support for the SVC code printed in the error message	Use the supervisor listing to check if the option used by the SVC is supported. For example, SVC X'43' must be supported by a supervisor generated with PFIX = YES.	If the option is not supported, avoid the use of the SVC by the program, or re-assemble the supervisor, or re-run the program using a supervisor that supports the option.
	Compare the SVC code printed with the list of SVCs given in Chapter 6 of this Section.	If the SVC code printed in the error message is not listed, go to check 3. Use the figure opposite to isolate possible causes of error if the error message is 0S04I. If the option is supported and no reason can be established for the error, proceed to check 2.
Check 2 Location of SVCs issued by the program	Use the program listing and linkage editor map* to locate where the SVC is issued. Note: To ensure that all SVCs issued by the program are checked, use a program listing that shows the expansions of the macros used. (If such a listing is not available, you will have to re-assemble the program with the PRINT GEN assembly control statement.)	Check the sequence of SVCs issued and macros used to ensure that the program is written to enable correct use and sequence of SVCs. If there is any doubt about the sequence of SVCs issued by the program, re-run the program with the PDAID GSVC trace. (Refer to Section 2-B-4.) If the error can not be isolated, proceed to check 3.
Check 3 Dump of partition	Was a System dump taken when the program check occurred? (System dump is described in Section 2-A-2.)	No: Re-run the program with the // OPTION DUMP statement in the job stream, or initiate the SDAID dump on event program check (refer to Section 2, B-10-F). Yes: Proceed to check 4.
Check 4 Analyze the dump	Use the program listing, error message, linkage editor map*, and system dump to locate the SVC instruction in the dump.	Compare the hexadecimal code for the SVC in the dump with the SVC instruction in the program listing. If there are differences, analyze the program to isolate the point where the correct SVC instruction is overwritten. As a further aid in this analysis, you may need to re-run the program with the SDAID SA trace. Use the STOP ON EVENT option and obtain a non-destroying dump at the point suspected of causing the overwriting. If there are no differences and the error can not be isolated, inform IBM and have the following ingormation available: • Supervisor listing • Program listing, including the macro expansions • Linkage editor map • System dump • SYSLOG output • Job stream.

* *If the program has been link edited using the DOS/VS relocating loader option, the relocation factor must be calculated based on the start address of the partition used for the program when it failed.*

Program canceled for other reasons

CHART 11, PART 1 OF 1

Check | *Method of Analysis* | *Recommendations to Aid Further Problem Analysis*

Check 1

Reason for program cancelation

Did the operator cancel the program?

Yes

Inspect messages printed on SYSLOG and SYSLST prior to the cancelation. If necessary, interview the operator to establish system activity during program execution. (It is possible that incorrect response to messages issued during program execution caused an unusual condition to occur.)
Proceed to check 2.

No

Examine the program listing and check for the use of DUMP or CANCEL macros left in the program from testing phases (program testing).
Remove these macros and proceed to check 2.

Check 2

Re-run the program

Re-run the program. If possible, be present during the re-run to observe system activity. Follow the recommendations given in *DOS/VS Messages* manual, and ensure the correct response is given to messages issued during program execution.
Ensure the use of the // OPTION DUMP statement in the job stream, to obtain a system dump when the program is canceled. (Refer to Section 2-A-2.)

If the identical error recurs, it may be due to the incorrect use by the program of an IBM-supplied routine.
Inform IBM and have the following information available:

- SYSLOG output
- SYSLST output
- System dump output
- Job stream
- Program listing and linkage editor map.

Check	*Method of Analysis*		*Recommendations to Aid Further Problem Analysis*
Check 1 Approximate the area where the program check occurred	Is the program check within the logical transient area? (Message 0S08I followed by message 0S11I will inform you where the instruction address, (the program check occurred) is located.	No	Go to check 8.
		Yes	Proceed to check 2.
Check 2 Dump available?	Was a system dump executed at the occurrence of the program check? (Refer to Section 2-A-2.)	No	Re-run the program with the //OPTION DUMP statement in the job stream. Also, initiate the PDAID transient dump program, described in Section 2-A-4, or use the SDAID stop-on-event program check function and obtain a non-destroying dump. (Refer to Section 2-B-10.) After the re-run, go to check 4.
		Yes	Proceed to check 3.
Check 3 Analyze the system dump to determine the name of the logical transient	Use the system dump to locate the partition save area. (Refer to Chapter 12 in this Section.) Locate the return PSW in the save area and determine the failing instruction address. Locate this address in the dump and go two bytes to the left. Is the contents of these two bytes X'0A02' (SVC 2)?		Re-run the program and initiate the PDAID transient dump program as described in Section 2-A-4, or use the SDAID stop-on-event program check function and obtain a non-destroying dump. (Refer to Section 2-B-10.) (Also include the //OPTION DUMP statement in the job stream to obtain a system dump.) Proceed to check 4.
			Locate the general purpose register 1 in the save area. This consists an address. Locate this address in the dump and convert the first 8 bytes, starting from this address, to characters. This is normally the name of the logical transient fetched by the program. (For OPEN or CLOSE register 1 may point to the DFT). Note: The cause of the program check need not necessarily be in this transient, since a logical transient may need to fetch another transient to complete a service. The program check could therefore be caused by incorrect parameters being passed to the transient.

Program canceled by Program Check in Supervisor

CHART 12, PART 2 OF 3

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 4 Determine the name of the logical transient fetched	Use the PDAID transient dump output to determine the name by converting the first 8 bytes of the dump to characters. (Refer to the example of a transient dump output given in Section 2-A-4.)	Proceed to check 5.
	Use the SDAID non-destroying dump output to locate Byte X'1C' of SYSCOM which contains the address of the logical transient area. (The address of SYSCOM is contained in bytes X'80-83' of the low address storage. (Refer to Section 2-E-2.) The first 8 bytes of the LTA, when converted to characters, give the name of the logical transient fetched.	Proceed to check 5.
Check 5 Type of logical transient program	Is the logical transient user-written?	No: Use the problem program listing and the system to determine the routine that was being executed when the program check occurred. Then determine which macro in the routine was responsible for fetching the logical transient. For example, the OPEN macro will cause the logical transient $$BOPEN to be fetched. If the error can not be isolated, inform IBM and have the following information available: • SYSLOG output • PDAID/SDAID output (if available) • Job stream • Supervisor listing • Program and transient assembly listings • Linkage editor maps.
		Yes: Proceed to check 6.
Check 6 PDAID or SDAID output available?	To complete problem analysis you will require the PDAID transient dump output, or a non-destroying dump output from the SDAID program.	If you have already re-run the program and obtained one or both of these dumps, proceed to check 7. Otherwise, re-run the program, following the recommendations given in check 2 before proceeding to check 7.

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*

Check 7

Analyze the dump of the logical transient area

→ Use the PDAID/SDAID output and the assembly listing of the logical transient fetched to locate the failing instruction address in the logical transient area.

A Locate the general registers at the top of the dump.

→ Look for unreasonable or totally unexpected values in the registers. If you find any that are connected with the failing instructions locate the code in the assembly listing that is responsible for preparing the register(s). If the contents of the general registers appear to be correct proceed to check 8. (General Register usage is described in Chapter 9 of this Section).

B Locate data preparation areas that are used by the failing instruction in the dump. Check the instructions in the assembly listing of the Logical Transient that is responsible for preparing data areas for use by the failing instruction.

→ Depending on the error message, inspect data used by the failing instruction for validity, duplication, and missing information.

If the error can not be isolated, inform IBM and have the following information available:

- SYSLOG output
- PDAID/SDAID output
- Job stream
- Supervisor listing
- Program and transient assembly listings
- Linkage editor maps.

Check 8

Isolate area in the supervisor causing the program check

→ Use the error message and assembly listing of the supervisor to determine in which routine in the supervisor the program check occurs.

→ Before informing IBM of the problem, re-run the program with the SDAID stop-on-event program check function and obtain a non-destroying dump when the program check occurs. (Refer to Section 2-B-10.)

Have the following information available.

- SYSLOG output
- Job stream
- SDAID output
- Supervisor listing
- Program listing and linkage editor map.

Program canceled by Program Check in POWER or POWER RJE

CHART 13, PART 1 OF 2

Check	*Method of Analysis*		*Recommendations to Aid Further Problem Analysis*
Check 1 Job flow	Were there any interruptions in normal system operations during program execution? For example, input/output device problems, necessitating a change in I/O devices assigned, or waiting times, or changes to jobs using POWER/POWER RJE during job execution.	Yes	Re-run the program ensuring no interruptions.
	No ↓ Have any PTFs (IBM supplied program temporary fixes) been applied to the POWER/ POWER RJE program since the program last ran successfully?	Yes	If possible have the PTF removed and re-run the program.
	No ↓ Has an identical program check occurred before when running this program under POWER/POWER RJE?	Yes	Go to check 3.
		No	Proceed to check 2.
Check 2 Dump of the POWER/POWER RJE partition	Was a system dump executed when the program check occurred? (Refer to Section 2-A-2.)	Yes	If the program is running under POWER, analyze the dump using the illustrations shown in Appendix K. If the program is running under POWER RJE, analyze the dump using the illustrations shown in Appendix L. If the error can not be isolated, go to check 4.
		No	Re-run the program with the //OPTION DUMP statement in the job stream. Also initiate the SDAID stop on event program check and obtain a non destroying dump when the program check occurs. (Refer to Section 2-B-10). If the error can not be isolated go to check 4. If the program is running under POWER, analyze the dump using the illustrations shown in Appendix K. If the program is running under POWER RJE, analyze the dump using the illustrations shown in Appendix L. If the error can not be isolated go to check 4.

Check	*Method of Analysis*		*Recommendations to aid further Analysis.*
Check 3 Program check occurred before	Was a note of system activity and environment made when the identical program check occurred in any previous run of the program?	Yes	Compare note made with the conditions and environment under which this program check occurred again. Proceed to check 4.
		No	Make note of the system environment and I/O units involved. Also note the I/O files in use. Proceed to check 4.
Check 4 Gather more system information	Re-run the program with the PDAID F/L trace to trace phases used. Also use the DIAG parameter to trace SVCs issued by the POWER or POWER RJE SCP. (Refer to Appendix J).		Re-analyze the problem using the output from the re-run and information obtained from your previous analysis of the problem. If the error can not be isolated, inform IBM and have the following information available: o Program output o SYSLOG output o Job stream o Supervisor listing o POWER or POWER RJE assembly listing listing and linkage editor map o Any notes made about system environment o I/O units used and system condition o PDAID output o POWER DIAG SVC trace output.

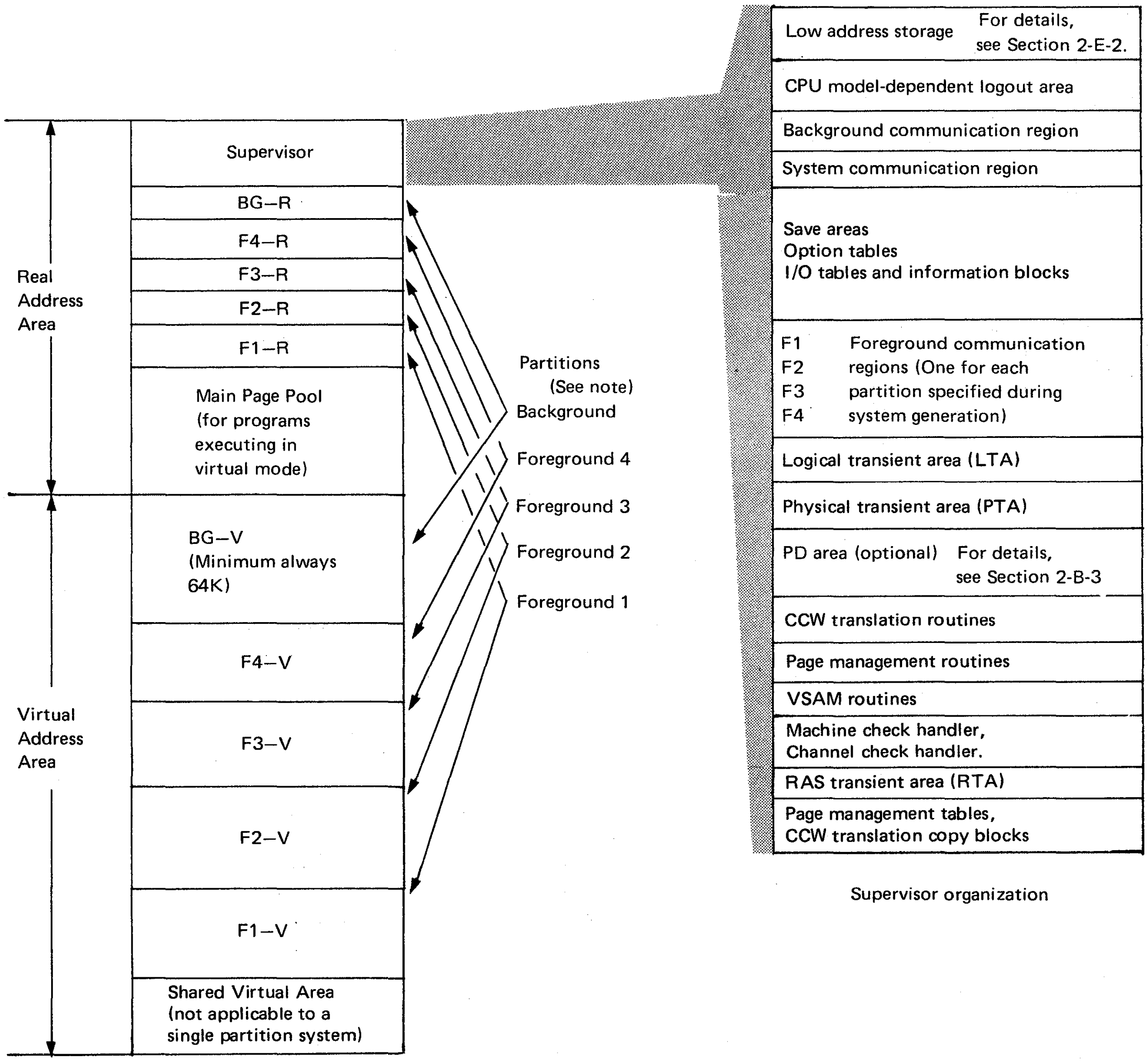
CONTENTS

Note: Contents and addresses shown in the illustrations are subject to change and are shown only as an aid to offline debugging of DOS/VS release 28.
IBM will not be responsible for any system malfunction resulting from a change made by the user to any contents or addresses of the tables and blocks described.

GENERAL ORGANIZATION OF VIRTUAL STORAGE

The figure below illustrates the general organization of virtual storage.

The supervisor is loaded in the real address area beginning at virtual address byte 0. Virtual storage can extend up to 16 million bytes. The figure also shows the general organization of the supervisor. Each area within the supervisor is described in more detail in this Section, except for the low address area and the PD area which are described in Section 2.



Note: Up to five partitions may be specified: one background and four foreground. Each partition consists of the pair "real partition – virtual partition".
The SVA and each virtual or real partition must be a multiple of 2K. It may also, however, be ØK, except the SVA and BG virtual which must be at least 64K.
To be active a foreground partition must have a virtual partition of at least 64K.

Figure 4.1 The organization of Virtual Storage.
The organization of the supervisor area is also illustrated and parts of it are described in this Section.

COMMUNICATION REGIONS

Partition Communication Regions, (Comregs)

In a multiprogramming system individual communication regions are defined for each partition. The communication region (comreg) belonging to the active partition is an area that serves as an initial pointer to other supervisor tables and areas. The comreg also contains pointers to user program tables and areas. The MVCOM and COMRG macro instructions enable access to information contained in these regions. Fields in the comreg are addressed relative to the first byte. The communication regions are located within the supervisor and their format is described in Figure 4.2 and Figure 4.3, parts 1 through 6, explain the contents of each field.

Locating the partition communication regions

After IPL, low address storage bytes X'14–17' contain the address of the comreg used by the active partition.

Note: The contents of these bytes will not be valid after executing the stand-alone dump program. Therefore, it is important for the operator to dump, or display and note, the contents of low address storage before executing the stand-alone dump. Locate bytes X'7C' and '7D' in the active comreg. This is the address of PIB2, also referred to as the PIB (program Information Block) Extension. The first two bytes of an entry in the PIB2 contain the address of its associated comreg. (Refer to Chapter 7 in this Section for a detailed description of the PIB2.)

Example A below shows a dump of low address storage. Bytes X'14–17' contain the address 04A0. This address has then been located in a stand-alone dump output as shown in example B. The address of PIB2 is indicated in this example. Example C shows the PIB2 from which the addresses of all the partition comregs are found.

Example A

```
                                                              Address of active comreg
DM 000000
00000000  00000000  00000000  00000000  00000000  000004A0  40000000  00000000
470D0000  00040C3C  440C0000  000013AA  00000000  00000000  470F2000
                                                                       (Ex 80 KT)
```

Example B

```
07B7E0   E2C9C4C5 40E2E3D6 D9C1C7C5 40E4D5C3 C8C1D5C7 C5C4C3C8 C1D5C7C5 C4404004
                                                                  Stand-Alone dump output
         Address of active comreg taken                Address of active comreg
BLOCK 000  from dump of low address storage              overwritten by the dump program
000000   0900A619  executed before taking the  000 00000000 4B4B4B4B 4B4B4B4B
000020   470D0000                              000 00000000 470F2000 0000090C
000040   00078038  stand-alone dump!           900 01AAAB17 00080000 0000A67E  ..C............
000060   440C0000                             0000 0007B3AA 00080000 0000A67E  ...............0
000080   00000540 00000000 00020007 00040005  10043010 00010000 00000000 000001E0
0000A0   00000000 00000000 20000060 000002C0  00000000 00000100 6000000C 00000000
0000C0   00000000 --SAME--                                     Bytes X'7C'-
0004A0   F1F261F0 F661F7F3 70007000 00000000  00000000 00000000 D5D640D5 C1D4C540  ......NO NAME
0004C0   00060FFF 00043067 00000000 00000010  000BFFFF FD7FCED3 1000CE50 002E40F0  X'7D'
0004E0   41044296 42974389 3F003F06 3F0C38F1  F2F0F6F7 F3F3F4F0 00003CD4 0000803C  Address of PIB2  673340...M....
000500   46300000 3DCC3E4C 3EBC0010 00000000  00008090 00007118 00003864 3BD45A30  ............M..
                                                                               (Ex 81 KT)
```

Example C

```
003B60   27000000 40000001 03003B...                   60000... ........  ....XY....
003B80                         3BD4 Address of  B C2C4  Address of F4 comreg  ..$$BDUMP .....
003BA0   Address of BG comreg       PIB2        B 0000
003BC0                                                 04A00000 00020008 00000000
003BE0   00000000 04A00000 00020007 00000000  00100000 45900036 00000000 00000000  } PIB 2
003C00   00200000 44F00062 00000000 00000000  00300000 4450008E 00000C00 00000000
003C20   00400000 43B000BA 00000000 00000000  00500000
                                                Address of F2 comreg
Address of F3 comreg  16 byte PIB2 entry belonging to F1                            ......AR.......Y
         3200C2C7 80040000 00006458 600E2880 8000C6F4 80007000 00006518  ......BG........  ......F4........ (Ex 82 KT)
```

FnCOMREG*

Displacement hexadecimal	0	8	0A	0C	17	18	20	24	28	2C
Displacement decimal	0	8	10	12	23	24	32	36	40	44
	Date	Address of PPBEG	Address of EOSSP	Problem Program Use	UPSI Byte	Job Name	Highest Storage Address of the Partition	End Address of Last Phase Fetched or Loaded	Address of Uppermost Byte of Phase with Highest Ending Address	Label Area Length
	XXXXXXXX	XX	XX	XXXXXXXX	X	XXXXXXXX	XXXX	XXXX	XXXX	XX

Displacement hexadecimal	2E	30	34	35	36	37	38	39	3A	3B	3C	3E
Displacement decimal	46	48	52	53	54	55	56	57	58	59	60	62
	PIK	End of Virtual Storage Address	Machine Confg. Byte	System Confg. Byte	Standard Language Translator I/O Options	Dump Log RELLDR and ASCII Options	Job Control Byte	Linkage Control Byte	Language Translator Control Byte	Job Duration Indicator Byte	Disk Address of Label Cylinder	Address of FOCL
	XX	XXXX	X	X	X	X	X	X	X	X	XX	XX
						Job Control Switches						

Displacement hexadecimal	40	42	44	46	48	4A	4C	4E	4F	58	5A	5C
Displacement decimal	64	66	68	70	72	74	76	78	79	88	90	92
	Address of PUBTAB	Address of FAVP	Address of JIBTAB	Address of TEBTAB	Address of FICL	Address of NICL	Address of LUBTAB	Line Count for SYSLST	System date	LIOCS Comm. Bytes	Address of PIB Table	ID Number of Last Checkpoint or DASDFP Indicator
	XX	XX	XX	XX	XX	XX	XX	X	XXXXXXXX	XX	XX	XX

Displacement hexadecimal	5E	60	62	64	66	68	6A	6C	6E
Displacement decimal	94	96	98	100	102	104	106	108	110
	Job Zone in Minutes	Address of Disk Information Block (DIB)	Reserved	Address of PC Option Table less 8 Bytes	Address of IT Option Table	Address of OC Option Table less 8 bytes	Key of Program with Timer Support	Reserved	Logical Transient Key
	XX	XX	XX	XX	XX	XX	XX	XX	XX

Displacement hexadecimal	70	74	78	7C	7E	80	84	86	87
Displacement decimal	112	116	120	124	126	128	132	134	135
	Address of SYSPARM	Address of JA Partition Table	Address of TOD-clock Common Area	Address of PIB Extension (PIB2)	Address of MICR DTF Table (PDTABB)	Address of QTAM Vector Table	Address of BG Comm. Region	Option Indicator	System Configuration, Byte 2, and RMSR Open Flag Byte
	XXXX	XXXX	XXXX	XX	XX	XXXX	XX	X	X

	88	8C	8D	8E	8F	97	98	9F
	136	140	141	142	143	151	152	159
	Reserved for compatibility reasons	Standard Job Control Options	Temporary Job Control Options	Disk Configuration	Catalog Procedure Name	Switch for Catalog Procedure	JCL Statement Name	81 byte sysin indicator
	XXXX	X	X	X	XXXXXXXX	X	XXXXXXX	X

Figure 4.2 Format and contents of any partition communications

*The address of the communications region is in fixed location X'14' – X'17'

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	Displacement (Hex)	Meaning
0	00	MM/DD/YY or DD/MM/YY either set permanently at IPL time or temporary by the job control date statement, or updated every time a GETIME macro is issued when time-of-day support is provided. Format controlled by BGCOMREG +53. (System Configuration Byte, data convention bit 0)
8	08	Address of the problem program area. (PPBEG)
10	0A	Address of the beginning of the problem program area. The 16 lower order bits of Y (EOSSP) equals Y (PPBEG)
12	0C	User area. If seek separation option is specified, bytes 12 and 13 are used at IPL time for the address of the seek address block.
23	17	Use program switch indicator. (UPSI byte)
24	18	Job name set by the job control program from information found in the job statement.
32	20	Address of the uppermost byte available to the problem program, that is either the address of the uppermost byte of the partition as determined during processing of the ALLOC or ALLOCR macro or statement, or the end address of the area specified by the SIZE parameter in the EXEC statement.
36	24	Address of the uppermost byte of the last phase of the problem program fetched or loaded. Not filled in when the phase is in the SVA.
40	28	Highest ending main-storage address of the last phase among all the phases having the same first four characters as the operand on the EXEC statement. For the phase $LNKEDT this field is not filled in. The address value may be incorrect if the program loads any of these phases above or below its link-edited origin address. If the EXEC statement has no operand, job control places in this location the highest ending address of all programs just link-edited.
44	2C	Length of the problem program label area.
46	2E	The low order byte identifies the partition (see Appendix B), and equals the displacement from the start of the PIB to the start of the PIB of the partition (without AP). The PIK from BGCOMREG changes during system operation and contains the PIK of the active partition (whichever one is active). The PIK in the FnCOMREG remains unchanged.
48	30	End address of virtual storage.
52	34	Machine Configuration Byte (Values set at supervisor generation time) Bit 0: Always set to indicate standard storage protect 1: 1=Decimal feature (always set) 2: 1=Floating point feature 0=No floating point feature 3: 1=Always set to indicate Physical transient overlap 4: Always set to indicate standard timer feature 5: 1=Channel switching device 0=No channel switching device 6: 1=Burst made on multiplex channel support 0=No burst mode on multiplex channel support 7: Indicates MCH/CCH in system.

Figure 4.3. Key to Communication Region displacement, part 1 of 6

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	(Hex)	Meaning
53	35	System Configuration byte Bit 0: 1=DDMMYY / 0=MMDDYY (Date convention bit set at generation time by STDJC) 1: 1=Two or more partitions 0=One partition only supported 2: 1=DASD file-protect supported 0=No file-protect support for DASD 3: 1=DASD SYSIN-SYSFIL 0=No DASD SYSIN-SYSFIL 4: 1=Teleprocessing 0=No teleprocessing 5: 1=Two or more partitions 0=One partition only supported 6: 1=Asynchronous processing 0=No asynchronous processing 7: 1=Track Hold 0=No Track Hold.
54	36	This byte contains the standard language translator I/O options (set by STDJC macro). Bit 0: DECK option 1= yes, output object modules on SYSPCH 1: LIST option 1= yes, output source module listings and diagnostics on SYSLST 2: LIST X option 1= yes, output hexadecimal object module listings on SYSLST (compilers only) 3: SYM option 1= yes, output symbol tables on SYSLST/SYSPCH 4: XREF option 1= yes, output symbolic cross-reference list on SYSLST 5: ERRS option 1= yes, output diagnostics on SYSLST (compilers only) 6: CHARSET option 1= 48, input on SYSIPT is 48 or 60 character set 7: Reserved.
55	37	This byte contains the standard supervisor options for abnormal EOJ, Relocating Loader and Control statement display and the indicator for the presence of the ASCII-EBCDIC and EBCDIC-ASCII translation tables. Bit 0: Always on 1: DUMP option 1= yes, dump registers and storage on SYSLST 2: 1=partition in wait state, because volume is to be mounted 3: LOG option 1= yes, list all control statements on SYSLST 4: 1=dummy device search in progress; do not enter ERP 5: Not used 6: Relocating Load option 1= yes, Relocating Loader supported 7: ASII option 1= yes, ASCII supported.
56	38	Job Control byte Bit 0: 1= Job accounting Interface (JA) is not supported 0= Job accounting Interface (JA) is supported 1: 1= Return to caller on LIOCS disk open failure 0= Do not return to caller on LIOCS disk open failure 2: 1= Job control input from SYSRDR 0= Job control input from SYSLOG

Figure 4.3. Key to Communication Region displacement, part 2 of 6

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	(Hex)	
56	38	Job Control byte (. . . . Cont'd) Bit 3: 1= Job control output on SYSLOG 0= Job control output not on SYSLOG 4: 1= Cancel job 0= Do not cancel job 5: 1= Pause at end-of-job step 0= No pause at end-of-job step 6: 1= SYSLOG is not a console printer-keyboard or DOC 0= SYSLOG is a console printer-keyboard or DOC 7: 1= SYSLOG is assigned to the same device as SYSLST 0= SYSLOG is not assigned to the same device as SYSLST.
57	39	Linkage control byte Bit 0: 1= SYSLNK open for output 0= SYSLNK not open for output 1: 1= Update of Second Level Directory and RAS loadlist in progress. (Interface between $MAINDIR and supervisor). 2: 1= Allow EXEC 0= Suppress EXEC 3: 1= Catalog linkage editor output 0= Do not catalog linkage editor output 4: 1= Supervisor has been updated 0= Supervisor has not been updated 5: Reserved 6: 1= Update of system CIL in progress. (Interface between $MAINDIR and supervisor). 7: 1= Check automatic condense limits and EOJ. (Interface between librarian and job control).
58	3A	Language processor control byte. This is a set of switches used to specify non-standard language translator options. The switches within the byte are controlled by job control OPTION statements and when set to 1, override standard options. The format of this byte is identical to the standard option byte (displacement 54) with one exception: Bit 7 in this byte is used to indicate to LIOCS that the rewind and unload option has been specified.

59	3B	Job duration indicator byte
		Bit 0: 1= Job in progress
		0= Job not in progress
		1: 1= Dump on an abnormal end-of-job condition
		0= No dump on abnormal EOJ
		2: 1= Pause at EOJ step
		0= No pause at EOJ
		3: 1= Job control output on SYSLST
		0= Output not on SYSLST
		4: 1= Job is being run out of sequence with a temporary assignment for SYSRDR
		0= Conditions for 1-setting not met
		5: 1= PCIL is being condensed
		0= PCIL is not being condensed
		6: 1= //DATE statement processed for current job
		0= No //DATE statement processed for current job
		7: 1= Batch command just issued
		0= Condition for 1-setting did not occur.

Figure 4.3. Key to Communication Region displacement, part 3 of 6

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	Displacement (Hex)	Meaning
60	3C	Binary disk address of the volume label area (label cylinder).
62 76	3E 4C	Addresses of FOCL, PUB, FAVP, JIB, TEB, FICL, NICL and LUB (See Figure 4.2)
78	4E	Set to the value nn specified in the LINES= nn parameter of the STDJC macro.
79	4F	The format of the system date contained within this field is determined by the IPL program from information supplied in the date convention bit (displacement 53). Bytes 85-87 contain the day count.
88	58	Bytes reserved for use by LIOCS. Transient dump programs insert a key to indicate to the LIOCS end-of-volume routine, $$BCMT07, that it was called by a B-transient.
90	5A	Address of the first part of the program information block (PIB) table.
92	5C	ID number of the last checkpoint. Byte 92 is also the temporary indicator of file protected DASD. Bits 0-6 correspond to channels 0-6. A bit ON means DASDFP for that channel. Bit 7 indicates 2321 DASDFP support. Byte 93 is used at IPL time by PIOCS — Bit Ø: 1 = 3330 file protection Bit 1: 1 = 3340 file protection
94	5E	Job zone for Time of Day. If ZONE=EAST, value is positive; if ZONE=WEST, value is negative.
96	60	Address of the disk information block (DIB) table for the partition.
98	62	Reserved.
100	64	Address for PC, IT, and OC option tables.
104	68	
106	6A	Key of the program that has internal timer support. The key is the same as the PIK for the timer supported partition. If multiple partitions all have timer support it is initially X'0010' but may be changed to the PIK of another partition by the TIMER command. It is copied into all partition communications regions. If no partition has interval timer support, these bytes contain X'0000'.
108	6C	Reserved.
110	6E	Logical Transient Key (LTK) contains the same value as the PIK (PID) (Displacement 46) when the logical transient is requested. When the transient area is not in use, LTK is equal to zero. The SVC2 routine sets the LTK. (See Appendix B for a description of the LTK). The SVC11 routine resets the LTK, (only significant in BG communication reg.)
112	70	Address of SYSPARM field.
116	74	Address of Job Accounting partition table.
120	78	Address of the Time of Day Clock common area.
124	7C	Address of second part of program information block (PIB) table.
126	7E	Address of PDTABB, table of DTF addresses for MICR support.
128	80	Address of QTAM vector table (IJLQTTAD).
132	84	Address of background communications region.

Figure 4.3. Key to Communication Region displacement, part 4 of 6

PARTITION COMMUNICATION REGION (. . . Cont'd)

Displacement (Dec)	(Hex)	Meaning
134	86	Option Indicator byte Bit 0: Reserved 1: 1= EU interface active 0= EU interface inactive 2: 1= Teleprocessing request 0= No teleprocessing request 3: 1= Supervisor support for tape 0= Supervisor does not support tape 4: Reserved 5: 1= RETAIN support generated 0= RETAIN support not generated 6: 1= Linkage to Channel En d Appendage Routine allowed 0= Linkage to Channel End Appendage Routine not allowed 7: 1= GETVIS function has been initiated 0= GETVIS function has not been initiated.
135	87	System Configuration byte 2 and RMSR Open Flag byte Bit 0: 1= PCIL supported 0= PCIL not supported 1: TOD supported 2: 1= PFIX macro supported 0= PFIX macro not supported 3: 1= Fetch $$OPEN by $JOBCTLJ 4: 1= Fetch $$OPEN by $JOBCTLD 5: 1= Fetch $$OPEN by $JOBCTLJ for WTM 6: 1= QTAM supported 0= QTAM not supported 7: Reserved (3–6: RMSR OPEN flags)
136	8B	Pointer to Option table in SYSCOM. Reserved for compatibility reasons.
140	8C	Standard Job Control Option byte Bit 0: 1= EDECK Standard Option 1: 1= ALIGN Standard Option Bit 2: 1 =ACANCEL standard 3-7: Not used
141	8D	Temporary Job Control Option byte Bit 0: 1= EDECK Temporary Option 1: 1= ALIGN Temporary Option. Bit 2: 1 =ACANCEL temporary 3-7: Not used
142	8E	Disk Configuration byte Bit 0-4: Not used 5: 1 = 3340 supported 0 = 3340 not supported 6: 1= 3330 supported 0= 3330 not supported 7: Always 1; indicates 2311 and 2314/2319 supported.
143	8F	Catalogued Procedure Name.

Figure 4.3. Key to Communication Region displacement, part 5 of 6

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	(Hex)	Meaning
151	97	Interface byte for Catalogued Procedures Bit 0: 1= Procedure being executed 1: 1= Overwrite processing 2: 1= Procedure with data 3: 1= Overwrite request for Job Control 4: 1= Insert request for Job Control 5: 1= Procedure end 6: 1= SYSLOG procedure 7: 1= Overwrite request for Supervisor.
152	98	JCL statement name for Catalogued Procedure.
159	9F	SYSIN 81 bytes indicator Bit 0: 1= Permanent 81 bytes on SYSRDR 1: 1= Permanent 81 bytes on SYSIPT 2: 1= Temporary 81 bytes on SYSRDR 3: 1= Temporary 81 bytes on SYSIPT 4-6: Not used 7: 1=Allow /&for MAINT CATALS.

Figure 4.3. **Key to Communication Region displacement, part 6 of 6**

COMMUNICATION REGIONS

System communication Region (SYSCOM)

This table is located in the supervisor, immediately after the background partition communication region. It contains partition-independent pointers and addresses of tables used by the system control program (SCP). The contents of SYSCOM is listed in Figure 4.4 parts 1 and 2, displacements are given in hexadecimal from the first byte of SYSCOM

Locating SYSCOM

Bytes X‘80 – 83’ of low address storage contain the address of SYSCOM.

Displacement Hex	Displacement Dec.	Contents
00	00	Address of Error Block
04	04	Address of Attention Exit
08	08	Address of Operator Option Cancel Exit
0A	10	Address of Operator Request Cancel Exit
0C	12	Address of SYSRES PUB
10	16	Address of Fetch Routine
14	20	Address of I/O Interrupt Routine
18	24	Address of External Interrupt Routine
1C	28	Address of Logical Transient Area
20	32	Address of First Byte of Problem Program Area
24	36	Free List Poi nter
25	37	Address of Channel Queue
28	40	Number of Channel Queue Entries
2A	42	Length of One Error Queue Entry
2C	44	Number of Partitions
2E	46	Not Used
30	48	Address of Channel Buckets
34	52	Address of CRT Table
38	56	Address of Seek Address Block Table
3C	60	Address of Channel Control Table
40	64	Flags and Switched (see expansions)
44	68	System Task Selection Control Field
46	70	Address of Task Selection
48	72	Address of PD Area
4C	76	Address of Track Hold Table
50	80	Address of Timer Request Table
54	84	Address of AB Table
58	88	Key of Task Owning LTA
5A	90	Key of Task Running
5C	92	Address of POWER Table
60	96	Reserved
64	100	Address of RF Table
68	104	Address of EU ECB Table
6C	108	Address of OLTEP Bucket
70	112	Address of RAS Linkage Area
74	116	Address of ASCII Translate Table
78	120	Address of PUB Owner-ship Table
7C	124	Address of Job Account-ing Common Table
80	128	Base Address of Page Management Routine
84	132	Base Address of Channel Program Translation Routine
88	136	Address of SDAID Save Area
8C	140	Address of Line Mode Table
90	144	Address of VSAM Comm Area
94	148	Address of PTA
98	152	Address of First System Task Block
9C	156	Address of Task Block of active System Task
A0	160	Align-ment Byte
A1	161	Pointer to RAS Task Block

Figure 4.4 Format and contents of SYSCOM part 1 of 2

Hex	A2	A3	A4	A5	A6	A7	B0	B4	B8
Dec	162	163	164	165	166	167	176	180	184
	Pointer to PMGR Task Block	Pointer to SUPVR Task Block	Pointer to CRT Task Block	Pointer to ERP Task Block	Pointer to PAGEIN Task block	Reserved 9 X'00'	Not Used	Address of MVCFLD	Not Used

Hex	BC	BE	C0	CB	CC	CE	D0	D4	D8
Dec	188	190	192	203	204	206	208	212	216
	Not Used	Not Used	Repositioning information for MFCM ERP	Number of Error Queue Entries	Length of PUB Table in Bytes	Number of Active Partitions	Address of Segment Table	Address of Page Frame Table	Address of Page Frame Table Extension

Hex	DC	E0	E4	E8	EC	F0	F4	F5	F8
Dec	220	224	228	232	236	240	244	245	248
	Address of Boundary Box	Address of DPD Table	Reserved	Address of VIRTAD Routine	Address of End of Real Storage (Fullword)	Address of Fetch table	SVA Flag (See expansion)	Address of SVA	Address of System GETVIS area

Figure 4.4 Format and contents of SYSCOM, part 2 of 2

Dec	Hex	Description	
64	40	Reserved for RMS support on the Models 115 and 125	
		X'80'	RMSR for channel attached devices, tapes and TP devices
		X'40'	Full RMS support (MCAR/CCH and RMSR)
		X'20'	MCAR/CCH support
65	41	X'80'	Initial selection of ErP
		X'40'	Reserved
		X'20'	Timer interrupt pending
		X'10'	MICR Stacker-select active
		X'08'	Invalid address during fetch
		X'04'	SIO routine entered after interrupt
		X'02'	Reserved
		X'01'	IPL in progress
66	42	X'80'	Initial RAS request
		X'40'	RAS WAIT request outstanding
		X'20'	RAS IPL in progress
		X'10'	Reserved
		X'08'	POWER supported
		X'04'	POWER initialized
		X'02'	GETREAL for SDAID in progress
		X'01'	Fetch for system task in progress (used by PDAID's)
67	43	Reserved	

LAYOUT OF SYSTEM TASK SELECTION CONTROL FIELD

244	F4	SVA Flag	
		X'80'	Do not test for warm start copy of SVA
		X'40'	SDL active
		X'20'	No 'Set SVA' or 'Set SDL. allowed
		X'10'	Build of SDL in progress
		X'08'	SDL overflow
		X'04' X'02' X'01'	Reserved

Dec	Hex	Description	
68	44	Always zero	
69	45	SELECT byte:	
		X'00'	No system task active
		X'01'	RAS active
		X'02'	PMGR active
		X'03'	SUPVR active
		X'04'	CRT active
		X'05'	ERP active
		X'06'	PAGEIN active

Figure 4.5 SYSCOM Expansion flag bytes.

I/O TABLES AND INFORMATION BLOCKS

The I/O tables and information blocks in the supervisor establish the interface between a specific task and the hardware channels. For example, for every logical unit name (SYSXXX) used, there must be a LUB (Logical Unit Block) entry. For every physical device used by the system, there must be a PUB (Physical Unit Block) entry. Then, when an I/O request occurs, an entry is made, via LUB and PUB, in the CHANQ (Channel Queue). An I/O request queued in the CHANQ contains an address that points to the CCB (Command Control Block) that contains the address of the channel program.

During the I/O device operation, entries are made in the ERRQ (Error Queue) in the Error Recovery Block, if a hardware failure occurs on the device.

Figure 4.6 (opposite) is an overall picture that illustrates the connections between the tables, (It does not represent their actual position, size, or relation as they are stored or printed in a dump output.) The initial pointers for all the tables and blocks are stored either in SYSCOM or in the active partition comreg.

Where a connection exists between tables, for example, the PUB and the CHANQ, pointers are contained in the table entries concerned. For example, the Figure shows a one-byte pointer in the PUB entry number 00 that relates this PUB entry to CHANQ entry number 00.

Figures 4.7 through 4.16 show the format of the tables and describe their contents.

A formatted stand-alone dump output prints the contents of the tables in a more readily understood format, and an example of such an output is shown in appendix G. The stand-alone dump program is described in Section 2-A of this manual.

I/O TABLES AND INFORMATION BLOCKS

The LUB table

This table is built up during system generation by the IOTAB supervisor generation macro, according to the BGPGR and FnPGR parameters (where n is the partition number). The table has one entry for each logical unit required for the system. Each entry is two bytes long and entries are grouped into two classes:
- System LUBs
- Programmer LUBs

There are always 14 system LUBs for each partition on the system.
By examining the contents of this table you can see the logical units that:
- Are unassigned or assigned (and, if assigned, to which entry in the PUB table)
- Have a temporary assignment or an alternate assignment, or indicate that a DASD file is opened.

How to locate:

Bytes X'4C'–X'4D' in the partition communication regions contain the address of the first entry in this table. Label LUBTAB in the supervisor listing identifies the address of the first byte of this table.
The number of LUB entries for system logical units in the BG System LUB and the number of LUB entries for programmer logical units in each programmer LUB is stored in the NICL information block.

NICL, (Number in Class List)

Byte 0 of this information block contains the number of System LUB entries (for DOS/VS, always 14, X'0E'). Byte 1 contains the number of programmer LUBs for the BG partition, and the remaining bytes contain the number of programmer LUBs for each foreground partition in the system (one byte per partition).
The total number of bytes in the NICL is equal to the number of partitions in the system plus one.

How to locate

Bytes X'4A' – X'4B' of the partition comregs contain the address of the first entry in this information block. Label NICL in the supervisor listing identifies the address of the first byte of this information block.
A pointer to the first entry in the LUB table and a pointer to the first LUB entry for the programmer LUBs for each partition is stored in the FICL information block.

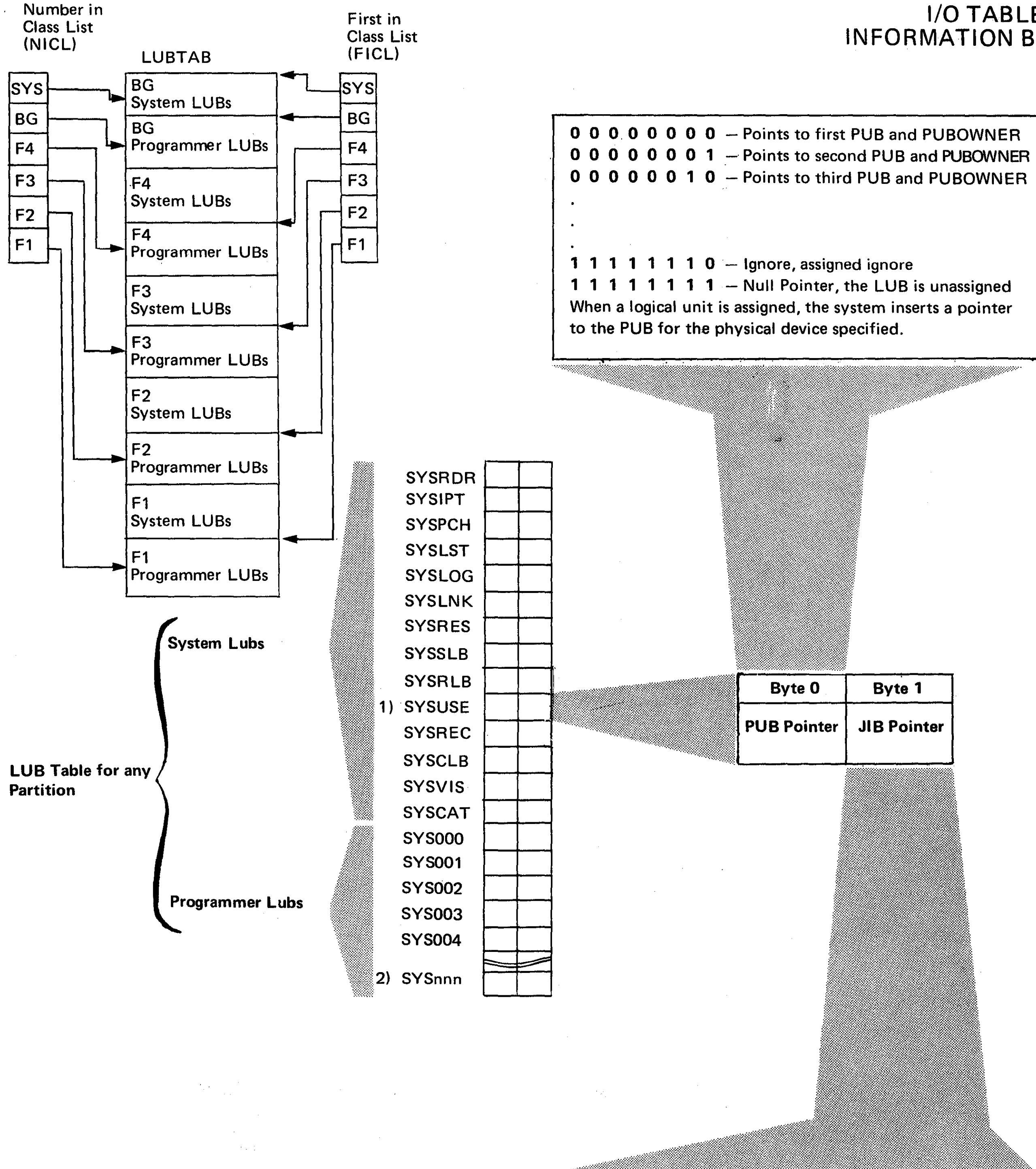
FICL, (First In Class List)

Each byte of this information block points as a displacement index to the beginning of a LUB sector.
Byte 0 to the first LUB entry, and the remaining bytes to the first LUB entries for each programmer LUB of each partition. The total number of bytes in the FICL is equal to the number of partitions in the system plus one.

How to locate:

Bytes X'48' – X'49' of the partition comregs contain the address of the first entry in this information block. Label FICL in the supervisor listing identifies the address of the first byte of this information block.

Figure 4.7 (opposite) shows the format and contents of the LUB table, and expands one entry in order to explain its contents. The figure also shows the relationship between the LUB, NICL, and FICL.



1) *SYSUSE may be called SYSCTL in error recovery messages.*

2) Since there are 14 system LUBs for each partition, the maximum number of Programmer LUBs is as follows:
When number of partitions (NPARTS) = 1:241
When number of partitions (NPARTS) = 2:227
When number of partitions (NPARTS) = 3:213
When number of partitions (NPARTS) = 4:199
When number of partitions (NPARTS) = 5:185

Figure 4.7. The LUB table.
The figure illustrates the format and contents of one entry and shows its relationship to the NICL and FICL information blocks.

I/O TABLES AND INFORMATION BLOCKS

The PUB table

This table is built up during system generation by the IOTAB supervisor generation macro and each DVCGEN macro fills one PUB entry in the PUB table.

By examining the contents of this table you can see both the physical address of each I/O device attached to the system and which devices are queued in the CHANQ. In conjunction with the contents of the LUB and JIB, you can ascertain the status of an I/O request for any logical unit.

The number of bytes in the PUB table (its size) is determined during system generation, although the operator can ADD or DELETE I/O devices during IPL. The PUB is divided into seven parts, each part containing the I/O devices attached to one of the seven channels. The first entry in the PUB belongs to the I/O device with the highest priority on channel 0. A pointer to the first PUB entry for each channel on the system is stored in the FOCL information block.

How to locate:

Bytes X'40' – X'41' of the partition communication regions contain the address of the first entry in this table. Label PUBTAB in the supervisor listing identifies the address of the first byte of this table.

The figure below shows the format and describes the contents of an entry in the PUB. Figure 4.9 (opposite) details a PUB entry to bit level.

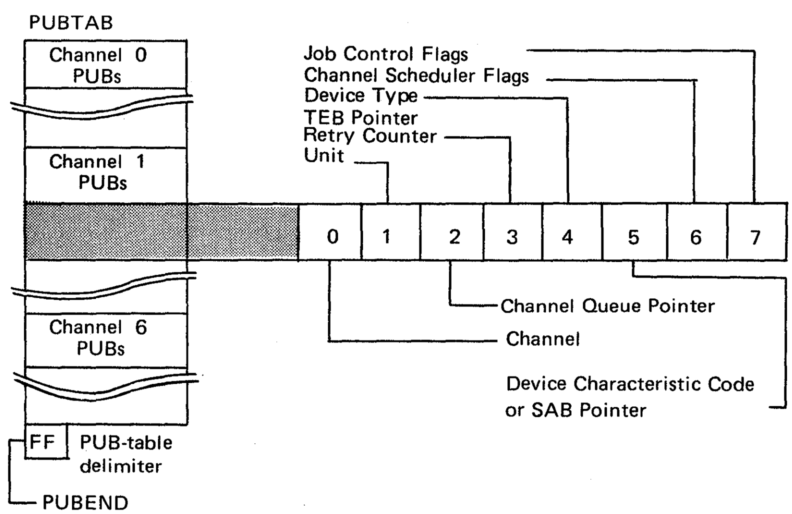


Figure 4.8. [redacted]

PUB

FOCL (First on Channel List)

Byte 0 of this information block points as a displacement index to the first PUB entry for the I/O device attached to channel 0, and byte 1 points to the first PUB entry for channel 1. The remaining five bytes point to the first entries in the PUB belonging to channels 2 to 6. X'FF' indicates that the associated channel is not supported on the system.

How to Locate:

Bytes X'3E' – X'3F' of the partition communication regions contain the address of the first entry in this information block. Label FOCL in the supervisor listing identifies the address of the first byte of this information block.

PUB

Byte 0: Channel number. (Hex 0-6, FF = NULL)

Byte 1: I/O device unit number

Byte 2: Hex 0, 1, 2, ... points to the first channel queue entry for this device.

Byte 3: If device is a 2495 Tape Cartridge Reader and TEBs are specified, this byte is a TEB pointer (Hex 1, 2, 3, ...). Otherwise, this byte is a ERP retry counter.

Byte 4: Device type code. See Figure 4.10, parts 1 through 3.

Byte 5: SS of the MODE= parameter in the DVCGEN macro for tape unit (See Section 2). For the Models 115 and 125 ICA line, this byte contains the displacement index of the entry in the Line Mode Table (LMT). The address of the LMT is contained in bytes X'8C' – X'8F' of SYSCOM.

For DASD with seek separation, this byte is used as the SAB Pointer. With Track Hold but not seek separation supported, this byte contains a pointer to the Track Hold Table entry or X'FF' (with both SKSEP and TRKHLD specified, the track hold pointer is found in the SAB entry).

For MICR type devices, this byte indicates which external interrupt line is in use.

For a 3705 Communications Controller, this byte contains the type number of the Channel Adapter.

For MFCM or MFCU

Bit 0: 1= Repositioning required
1: 0= SYSPCH temporarily assigned to hopper 1
1= SYSPCH temporarily assigned to hopper 2
2: 0= SYSIPT temporarily assigned to hopper 1
1= SYSIPT temporarily assigned to hopper 2
3: 0= SYSRDR temporarily assigned to hopper 1
1= SYSRDR temporarily assigned to hopper 2
5: 0= SYSPCH permanently assigned to hopper 1
1= SYSPCH permanently assigned to hopper 2
6: 0= SYSIPT permanently assigned to hopper 1
1= SYSIPT permanently assigned to hopper 2
7: 0= SYSRDR permanently assigned to hopper 1
1= SYSRDR permanently assigned to hopper 2.

Byte 6: Channel Scheduler Flags

Bit 0: 1= Device busy
1: 1= Switchable device
2: 1= EOJ for SYSRDR or SYSIPT
3: 1= I/O error queued for recovery
4: 1= Operator intervention required
5: 1= Device End posting required
6: 1= Burst mode or overrunnable device on byte MPX channel
7: 1= 7-track tape unit.

Byte 7: Job Control Flags

Bit 0-4: Standard MODE assignment for 7-track tape (all ones if not tape, all zeros if device is down)

Bit 5: 1 = DASD device with disconnect command chaining feature

Bit 6-7: B'11' (both on)= Headqueue in progress
B'01' = Headqueue requested.

Note: A null is generated for each device to be supported by the supervisor. Standard physical unit assignments are made to the PUB table at supervisor generation time. PUBs are ordered by channel and priority within a channel.

An entry in the PUB Ownership Table is associated with each entry in the PUB Table, if the supervisor has been generated to support multiprogramming.

Figure 4.9. Explanation of the contents of an entry in the PUB table

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
2400T9	9-track Magnetic Tape Tape units	50	Magnetic Tape devices
2400T7	7-track Magnetic Tape units	50	
3410T9	9-track 3410 Magnetic Tape units	53	
3410T7	7-track 3410 Magnetic Tape units	53	
3420T9	9-track 3420 Magnetic Tape units	52	
3420T7	7-track 3420 Magnetic Tape units	52	
2495TC	2495 Tape Cartridge Reader	51	Tape Cartridge Reader
1442N1	1442N1 Card Read Punch	30	Card Read Punches
2520B1	2520B1 Card Read Punch	31	
2560	2560 Multifunction Card machine	33	
2596	2596 Card Read Punch	30	
3525RP	3525 Card Punch (with optional read feature)	32	
5425	5425 Multifunction Card Unit	34	
2501	2501 Card Reader	10	Card Readers
2540R	2540 Card Reader	11	
3504	3504 Card Reader	12	
3505	3505 Card Reader	12	
1540P	2540 Card Punch	21	Card Punches
2520B2	2520B2 Card Punch	20	
1442N2	1442N2 Card Punch	22	
2520B3	2520B3 Card Punch	20	
3525P	3525 Card Punch	23	
1403	1403 Printer	40	Printers
1403U	1403 Printer with UCS feature	42	
1443	1443 Printer	41	
2260(local)	1053 Printer with 2848 Control Unit. MODE operand must be entered as X'01'	C0	
3203	3203 Printer	4A	
3211	3211 Printer	43	
3277 (local 3270)	3284 or 3286 Printer with 3272 Control Unit. MODE operand must be entered as X'01'	B0	
3277B (local 3270)	3284 or 3286 Printer with 3272 Control Unit, attached in burst mode to a multi-plexer channel. MODE operand must be entered as X'01'	B0	
5203	5203 Printer	4C	
52030	5203 Printer with UCS feature	4D	

Figure 4.10 **Device Type Codes, part 1 of 3**

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
1050A	3210, 3215 Console Printer Keyboards	00	Printer-Keyboards
125D	Models 115 and 125 Integrated Video Display Unit	B2	
125DP	Models 115 and 125 Integrated Video Display Unit with 5213 Console Printer attached	B2	Video Display Unit
UNSP	Unsupported device	FF	Unsupported. No burst mode on multiplexer channel
UNSPB	Unsupported device	FF	Unsupported with burst mode on multiplexer channel
2311	2311 Disk Storage device	60	
2314	2314 Direct Access Storage Facility	62	
2314	2319 Disk Storage Facility	62	DASD
2321	2321 Data Cell Drive	61	
3330	3330-1, 3330-2, or 333-1 Disk Storage	63	
3340	3340 Disk Storage (general)	68	
3340	3340 Disk Storage with 3348 Mod 35	69	
3340	3340 Disk Storage with 3348 Mod 70	6A	
1419	1255 Magnetic Character Reader	72	
1419	1259 Magnetic Character Reader	72	MICR- Magnetic Ink Character Recognition devices
1419	1419 Magnetic Character Reader	72	
1419P	1419 Dual Address Adapter Primary Control Unit	73	
1419S	1419 Dual Address Adapter Secondary Contr. Unit	74	
2701	2701/2715 Data Adapter Unit	D0	Teleprocessing lines
2702 A B C D	2702 Transmission Control Unit	D1	A= SAD0 comm'd B= SAD1 comm'd C= SAD2 comm'd D= SAD3 comm'd when enabling the line
2703	2703 Transmission Control Unit	D2	
2703	Integrated Communications Adapter (Models 125 and 135)	D2	
2703	3705 Communications Controller in Emulation Mode	D2	
2955	2955 Data Adapter Unit	D7	Data Link for RETAIN
1017	1017 Paper Tape Reader with 2826 Control Unit	78	Paper Tape Readers
2671	2671 Paper Tape Reader	70	

Figure 4.10 Device Type Codes, part 2 of 3

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
1018	1018 Paper Tape Punch with 2826 Control Unit	79	Paper Tape Punch
1419 1419P	1270 Optical Reader/ Sorter 1275 Optical Reader/ Sorter	72 73	Optical Readers
1287 1288 3881 3886 3540	1287 Optical Reader 1288 Optical Page Reader *3881 Optical Mark Reader 3886 Optical Character Reader 3540 Diskette I/O unit	77 77 11 7C	Optical Readers DISKETTE
2260 3277 (local 3270) 3277B (local 3270)	2260 Display Station 3277 Display Station; MODE operand need not be entered 3277 Display Station; attached in burst mode to a multiplexer channel. MODE operand need not be entered	C0 B0 B0	Display Stations
7770	7770 Audio Response Unit	D3	Audio Response unit

**Note: The logical unit name SYSIN cannot be assigned to a 3881*

Figure 4.10 Device Type Codes, part 3 of 3

PUBOWNER (PUB ownership)

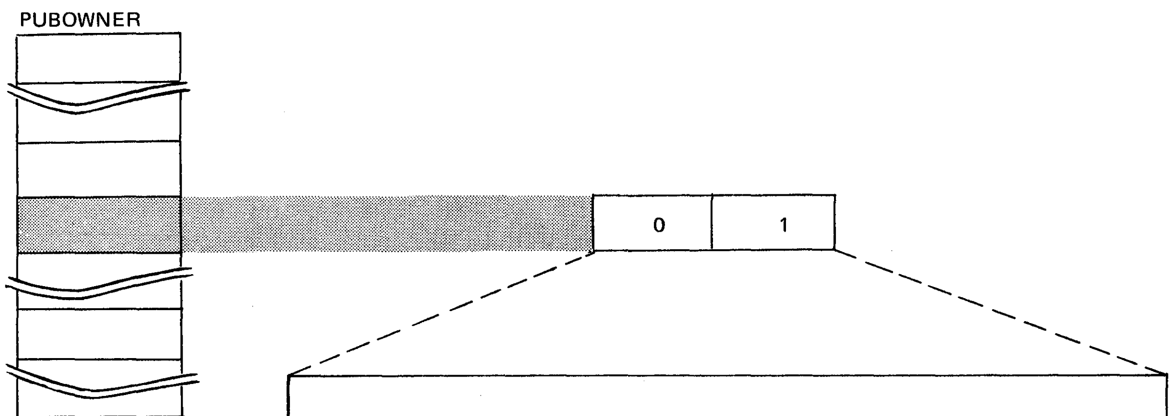
entries

An area in the supervisor is always reserved for this table. The number of entires is equal to the number of entries in the PUB, and each entry is two bytes long.

By examining the contents of this table in conjunction with the associated entry in the PUB, you can identify the partition using a particular I/O device, for example, when conflicting assignments are thought to be the cause of a system malfunction.

How to locate:

Bytes X‘78’ – X‘7B’ of SYSCOM contain the address of the first entry in this table. Label PUBOWNER in the supervisor listing identifies the address of the first byte of this table. The Figure below shows the format and describes the contents of an entry in the PUBOWNER.



Byte 0: Bit 0 Reserved
1 1 = waiting for volume to be mounted
2-7 Reserved

Byte 1: Identifies the partition that owns the PUB according to following table:

Flag	Partition owning PUB if number of partition is:			
	2	3	4	5
X'01'	BG	BG	BG	BG
X'02'	F1	F2	F3	F4
X'04'		F1	F2	F3
X'08'			F1	F2
X'10'				F1

Notes: The number of entries in the PUB Ownership table is equal to the number of entries in the PUB table. Associated with each PUB entry is an entry in the PUB Ownership table.

Figure 4.11 Contents of an entry in the PUBOWNER.

The relationship between the PUB, the PUBOWNER, and the FOCL is shown in Figure 4.6 at the beginning of this Chapter.

I/O TABLES AND INFORMATION BLOCKS

The JIB (Job Information Block)

An area in the supervisor is reserved for this information block during system generation by the JIB parameter of the IOTAB macro. This information block records any changes to the standard or permanent assignments made by the // ASSGN job control statement. Extent information is also recorded in the JIB when the supervisor supports the DASDFP feature.

By examining the contents of an entry in the JIB and its associated LUB, PUB, and PUBOWNER entries you can identify the logical units that are temporarily assigned, the address of the I/O device, and the partition using the device. Useful information can also be obtained from the JIB about DASD extents (DASDFP only), for example, when it is not certain why the message INVALID SEEK ADDRESS is printed during the execution of a particular job.

How to locate:

Bytes X'44' – X'45' of the partition communication regions contain the address of the first entry in this information block. Label JIBTAB in the supervisor listing identifies the address of the first byte of this information block.

Entries in the JIB are made:

- when a temporary assignment is made
- by alternate tape assignments
- by DASD extent information (when the file protect feature is supported by the supervisor.)

The next available JIB entry is recorded in the FAVP.

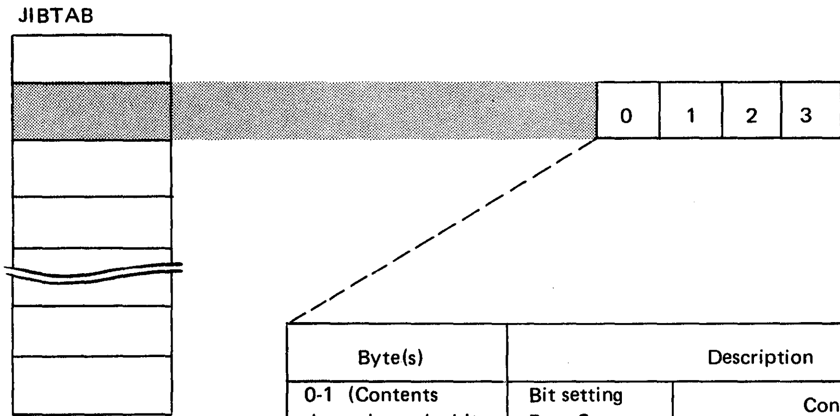
FAVP (First Available Pointer)

This is a one-byte pointer to the next available JIB entry. It contains a hexadecimal displacement from the first entry in the JIB. If it contains X'FF', no more entries in the JIB are available.

How to locate:

Bytes X'42' – X'43' of the partition communication regions contain the address of this pointer. Label FAVP in the supervisor listing identifies the address of this one-byte pointer.

Figure 4.12 (opposite) illustrates the format and contents of a JIB entry. Its relationship to the LUB and PUB is indicated in Figure 4.6.



Number of entries (length of JIB table) is determined at supervisor generation time

Byte(s)	Description	
	Bit setting Byte 2	Contents
0-1 (Contents depends on the bit setting in byte 2)	Bit 0= 1 Stored standard assignment	LUB entry of stored standard assignment. (PUB and JIB pointer)
	Bit 1= 1 Alternate assignment	Byte 0: PUB pointer Byte 1: X'00'
	Bit 2= 1 2311/2314/ 2319 Extent	Byte 0: Cylinder lower limit Byte 1: Cylinder upper limit See Note 1
	Bit 3= 1 2321/3330/3340 Extent	For 2321: Lower limit (Cell or combined subcell and strip), or Upper limit (Cell or combined subcell and strip) For 3330/ 3340: Cylinder lower limit, or Cylinder upper limit (One cylinder number of the 3330/3340 uses two bytes) See Notes 1 and 2
2	Meaning if bit = 1 : Bit 0: Stored standard assignment Bit 1: Alternate assignment Bit 2: 2311/2314/2319 extent Bit 3: 2321/3330/3340 extent Bit 4: The alternate assignment indicated in bit 1 is permanent. This bit is also on when one of the extent indicators (bit 2 and bit 3) is on. Bit 5: Catalogued procedures processing Bit 6/7: Reserved.	
3	Chain byte Contains the displacement index of the next JIB. X'FF' defines the end of the chain.	

Note 1: *Only when file-protect on DASD.*

Note 2: *Two JIB's are required for a 2321/3330/3340 extent; one for lower limit and one for upper limit. The lower limit defining JIB must be chained to the upper limit defining JIB.*
For 2321, byte 1 of this JIB contains the subcell number times 10 plus the strip number in binary.

Figure 4.12 Explanation of the contents of an entry in the JIB.

CHANQ (Channel Queue)

The area in the supervisor reserved for this table is determined during system generation by the CHANQ parameter of the IOTAB macro.

This table is used by the supervisor to schedule I/O operations. An entry is made in the channel queue whenever a request is made for an I/O operation, and the entry remains in the queue until the operation is completed. Thus, at any point in time, the queue will consist of entries for I/O operations in progress and I/O operations waiting for initiation. Whenever an I/O event completes, the queue is examined to see if an operation is waiting for the device, and if so, the operation is initiated.

Each entry made in this table occupies an eight-byte field. Entries are pointed to by a CHANQ POINTER contained in byte 2 of any PUB entry owning a device waiting for an I/O operation to complete.

By examining the contents of this table together with the contents of the PUB table you can determine the following:

- Whether a particular I/O device is waiting for an I/O operation to be completed.
- The reason for an uncompleted operation.
- How many I/O requests have been made for a particular device (by looking at the CHAIN byte).
- The CCB (Command Control Block) address and, therefore, the channel program and I/O area used by a particular device. (The CCB and channel program are described in Chapter 5 in this Section.)
- The identity of the task that requests an I/O operation for a particular device.
- Whether the channel queue is completely occupied (probably causing a soft wait state).

How to locate:

Bytes X'25' – X'27' of SYSCOM contain the address of the first entry in this table. Label CHANQ in the supervisor listing identifies the address of the first byte of this table.

The number of channel queue entries occupied at any given point in time depends on the I/O activity in the system. A one-byte pointer (FLPTR) points to the next eight-byte field in this table that is free for use.

FLPTR (Free List Pointer)

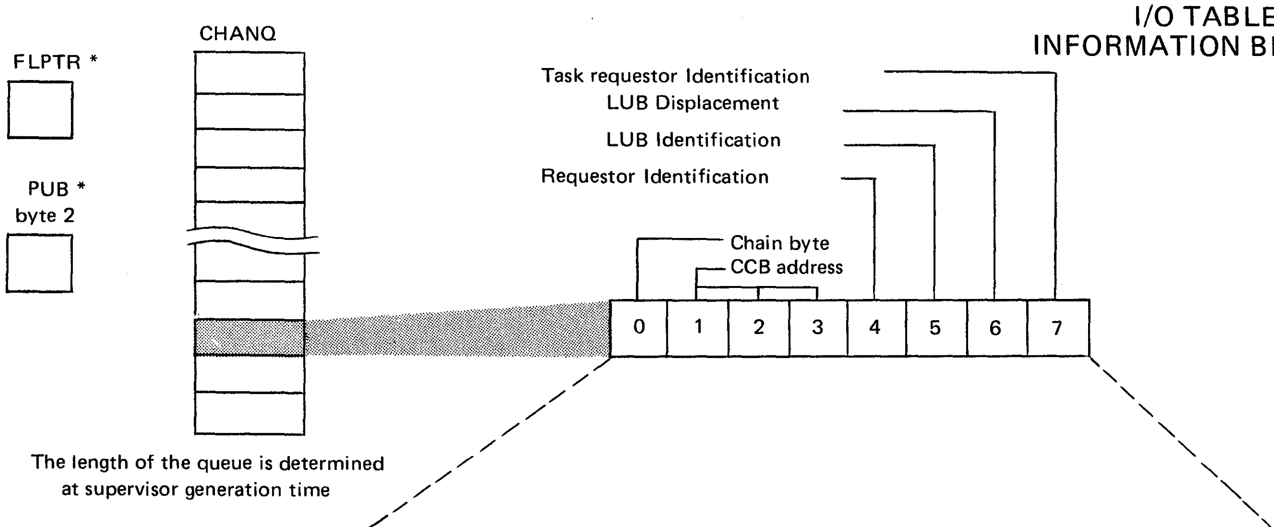
This one-byte pointer contains the hex displacement from the beginning of the channel queue table to the next available CHANQ entry. When the channel queue is full, it contains X'FF'.

How to locate:

Byte X'24' of SYSCOM contains the address of this information byte. Label FLPTR in the supervisor listing identifies the address of the first byte of this information byte.

For a detailed description of the operation of the CHANQ and FLPTR refer to the *DOS/VS Supervisor Logic* manual.

Figure 4.13 (opposite) shows the format and describes the contents of the CHANQ table, and Figure 4.6 illustrates the relationship between the PUB, CHANQ, and FLPTR.



Byte 0:	Contains the displacement within the channel queue of the next entry in the free list or in the list for a specific device, or X'FF' when it is the last entry in the free list or a device list.
Bytes 1-3:	Contains the CCB address for the specified device.
Byte 4:	Contains a code identifying the task making the I/O request. The one-byte entry indicates to which task the CCB belongs, and is in the form X'nk' where: n = user storage protection key. (Attention or system task = 0, BG task = 1, FG tasks = 2-5 depending on the number of partitions.) k = 0 for Attention and all user tasks. 1 for RAS 2 for PMGR 3 for SUPVR nK = FF for any unused channel queue entries. 4 for CRT 5 for ERP 6 for PAGEIN If the I/O request is from a subtask, this byte does not identify the subtask, but the partition. See Appendix B for an explanation of the REQID.
Byte 5:	Contains pointer (displacement index) to the entire LUB table identifying the logical unit making the I/O request. This is doubled to get the actual displacement into the full LUB table.
Byte 6:	Contains X'FF' if the LUB is a programmer class, or the displacement within the partition LUB if it is a system class. (Not used by a non-MPS Supervisior.)
Byte 7:	Contains the displacement within the PIB table of the PIB of the task requesting I/O, or X'FF' when the channel queue entry is in the free list. (Not used by a non-MPS Supervisor.) See Appendix B for an explanation of the TKREQID.

**Notes:* *FLPTR: The free list pointer contains the displacement within the channel queue of the first entry in the free list of X'FF' when the channel queue is full. Byte X'24' of the System Communication Region (SYSCOM) contains the address of the Free List Pointer.*
Label FLPTR identifies the location of the pointer (1 byte).

PUB byte 2: The PUB channel queue pointer contains the displacement within the channel queue of the first entry for a specific device.

Figure 4.13. Explanation of the contents of an entry in CHANQ.

I/O TABLES AND INFORMATION BLOCKS

Channel Control Table.

This table contains a code identifying the channel types attached to the system. There is one entry for each channel attached, and each entry is two bytes long.

No system generation macro is required to reserve an area in the supervisor for this table; information is entered into it by the STORE CHANNEL ID instruction during IPL.

How to locate:

Bytes X'3C' – X'3F' of SYSCOM contain the address of the first entry in this table. Label CHNTAB in the supervisor listing identifies the address of the first byte of this table.

Figure 4.14 (opposite) lists the meaning of the code contained in byte 0 of this table; byte 1 is always zero.

Channel bucket

This information block is always generated in a supervisor. Each channel attached to the system owns a 24-byte field in this information block, which records the contents of the I/O registers (general registers 1, 2, 3, and 4) and a pointer to the PIB (Program Information Block) for the last I/O started on each channel.

Its size, or the number of bytes reserved for this information block, is always sufficient to allow a 24-byte field for each of the 7 channels, whether attached to the system or not.

By examining the contents of this block, information relating to the last I/O started on any attached channel can be obtained.

Similar information can be obtained by examining the contents of the PUB, CHANQ, and FOCL, but the channel bucket formats the information and, in addition, contains a pointer to the PIB. Information in the PIB allows more details about the task issuing the last START I/O instruction to be obtained. (The PIB is described in chapter 8 in this Section.)

How to locate:

Bytes X'30' – X'33' of SYSCOM contain the address of the first entry in this information block. Label REGSAV in the supervisor listing identifies the address of the first byte in this information block.

Figure 4.15 (opposite) shows the format and contents of an entry made in the channel bucket for a system.

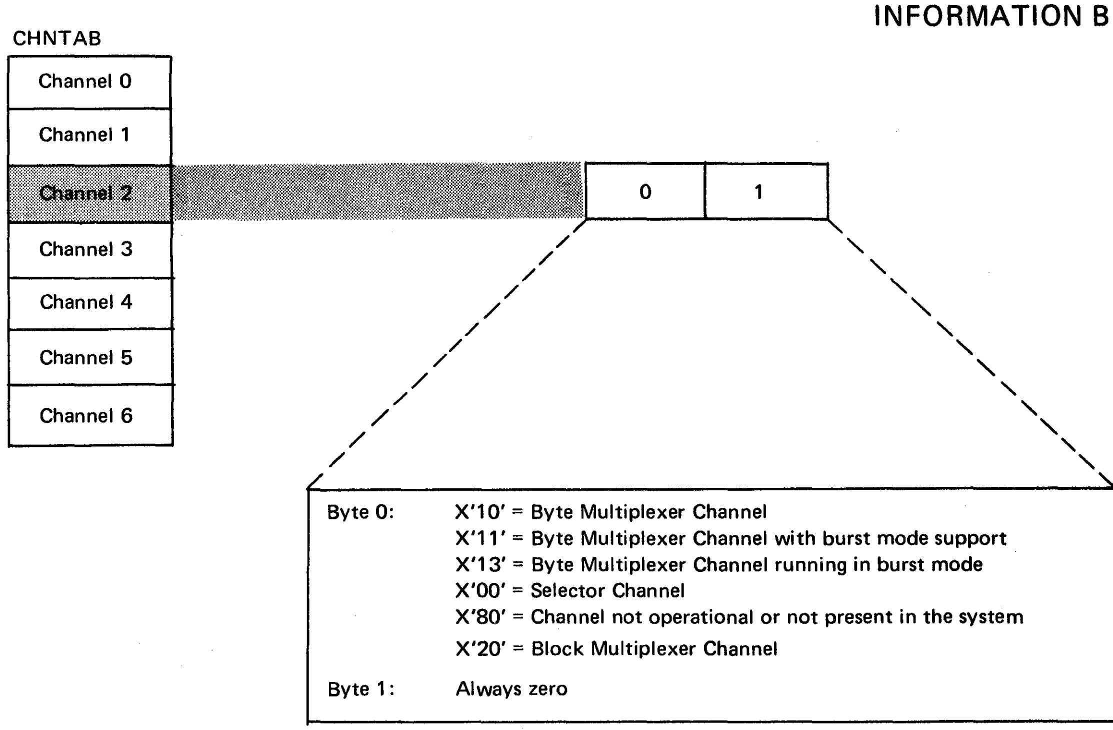
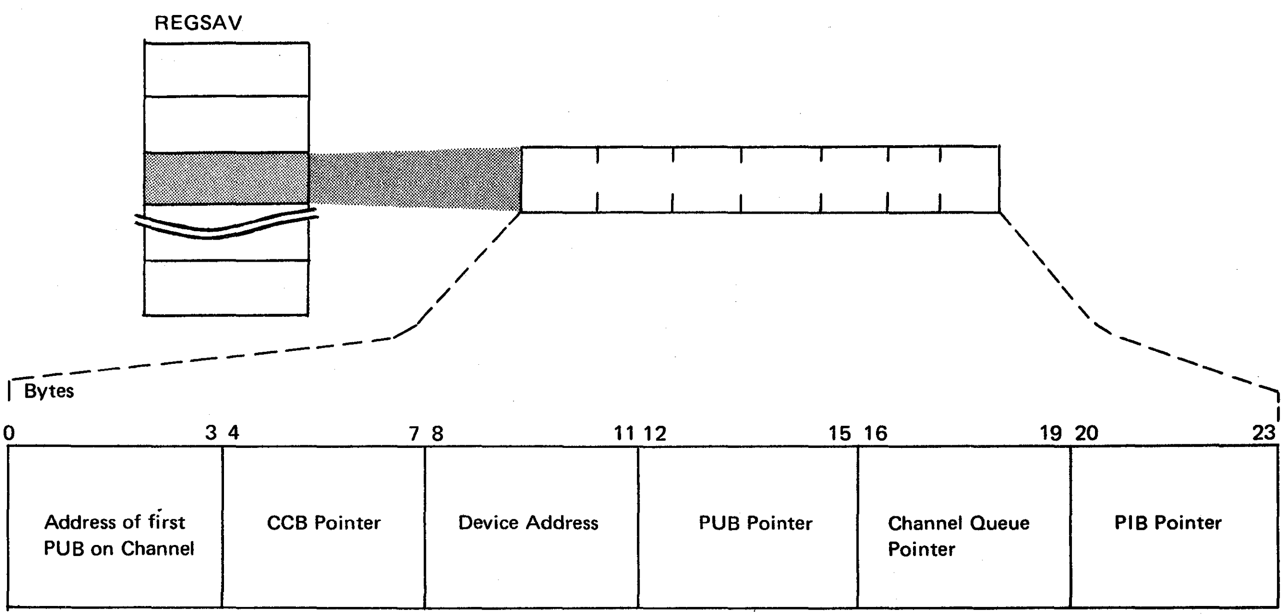


Figure 4.14. Explanation of the contents of the Channel Control Table



Notes: *1 A channel bucket contains information related to the last I/O started on the channel.*

2 The number of channel buckets in a system equals the number of I/O channels in the system.

Figure 4.15 Contents of the Channel Bucket

I/O TABLES AND INFORMATION BLOCKS

Error Recovery Block and Error Queue

Real storage area is reserved in the supervisor for the error recovery block during system generation by the ERRQ parameter of the FOPT macro.

The block is used by error queue entries that are built up by the supervisor in the event of an I/O device error during program operation.

Data recorded in an error queue entry is used by both the ERP (Error Recovery Procedure) and RMSR (Recovery Management Support Recorder) routines.

Each error queue entry is 44 bytes long (hex 2C), and the number of entries determined by the ERRQ parameter can be between 3 and 25 for a supervisor not supporting multiprogramming, or between 5 and 25 for a supervisor supporting multiprogramming.

On the occurrence of an I/O device error that can not be corrected by hardware or software error recovery, a message is printed on SYSLOG. The message may require operator response or action, and contains data recorded in the error queue. An example of this type of message is:

```
BG  0P47A        UNX  INTERV  SYS003=2A1
             CCSW=021000B49002000000  CCB=00B440
             SNS=402000040240241000000000000892B1614020102001A0010
```

If no message can be printed because of the severity of the error, for example, a hard wait state, data recorded in the error queue should be analyzed in a dump output.

By examining the contents of the error queue the following information can be obtained about any I/O device error recorded in the queue:

- The status of the I/O device and the last CCW issued.
- The active entries, if any (X'01' in byte 10).
- The address of the associated PUB entry, from which the device address can be found.
- The message code. (This code may refer to a DOS/VS message. For example, code 08 refers to device error recovery message 0P08A. The reason for the error and possible solutions are listed in *DOS/VS Messages*.)
- The address of the associated CCB, from which the address of the channel program and I/O area used in the operation can be located.

How to locate:

Bytes X'00' – '04' of SYSCOM contain the address of the first byte in this information block. Label ERBLOC in the supervisor listing identifies the address of the first byte of this information block.

Figure 4.16 (opposite) illustrates the format and describes the contents of an error queue entry.

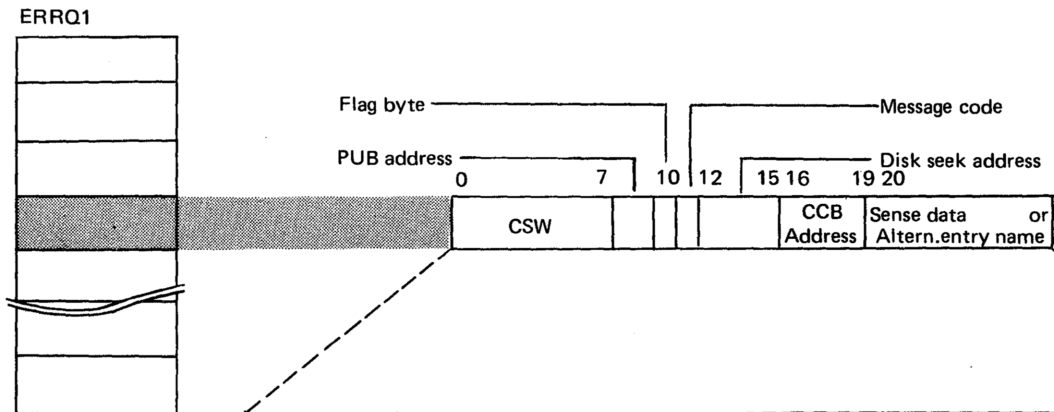
ERBLOC

0 7	8 9	10 11	12 13	14 15	First error queue entry
Name of phase to be fetched (always begins $$ABERRX)	Branch address to retry the the failing operation	Branch address to ignore the error	Branch address to cancel the task	Return address for A-transient	

ERRQ1

The A-transient loads one of these address in register 14 before branching to the return address (see bytes 14-15).

ERROR QUEUE TABLE



Bytes 0- 7:	CSW
Bytes 8- 9:	Address of PUB for device in error.
Byte 10 :	Flag byte: Bit 0: 1= No record found on DASD 1: 1= Intervention required 2: 1= Passback (Set by device ERP) 3: 1= Allow ignore 4: Not used 5: 1= Allow retry 6: Not used 7: 1= Active entry.
Byte 11 :	Message code: may refer to a device error recovery message generated by physical IOCS. (See *DOS/VS Messages* manual.) or This location may contain one of the following: X'E2' = The error is recoverable. X'AE' = A record is to be recorded on the system recorder file for SVC44 or a BTAM appendage routine, and a Physical Transient is to be fetched (last two characters of phasename are in bytes 20-21).
Bytes 12-15:	Disk seek address
Bytes 16-19:	Address of CCB
Bytes 20-43:	Sense data: The number of sense bytes generated depends on the options specified; the minimum is 24 bytes. or Alternate entry name: If byte 11 contains X'AE', bytes 20-21 contain the last two characters of the phase name of the Physical Transient to be fetched for SVC 44 (A3) or BTAM (A5). X'AF' in byte 22 indicates that the I/O area associated with an alternate entry has been fixed temporarily

Figure 4.16. Explanation of the contents of the Error Block and an entry in the Error Queue

I/O CONTROL SYSTEM

The data management facilities of DOS/VS are provided for by a group of routines collectively referred to as input/output control system (IOCS). A distinction is made between two types of routines:

1. Physical IOCS (PIOCS). The physical unit I/O routines included in the supervisor.
2. Logical IOCS (LIOCS). The logical unit I/O routines linked with the user's problem program.

Physical IOCS

Physical IOCS controls the actual transfer of data between the external medium and real storage. It performs the functions of initiating the execution of channel commands and handling associated I/O interrupts. Physical IOCS consists of the following routines:

- Start I/O routine
- I/O interrupt routine
- Channel scheduler
- Device error routines.

Logical IOCS

Logical IOCS performs the functions a user needs to locate and address a logical record for processing. A logical record is one unit of information in a file of like units, such as one employee's record in a master payroll file, one part number in an inventory file, or one customer account record in an account file. One or many logical records may be included within one physical record, such as a physical tape record (gap-to-gap). The term logical IOCS refers to the routines that perform the following functions:

- Blocking and deblocking records
- Switching between I/O areas when two areas are specified for a file
- Handling end-of-file and end-of-volume conditions
- Translating American National Standard Code for Information Interchange (ASCII) into Extended Binary Coded Decimal Interchange Code (EBCDIC) on input, and EBCDIC into ASCII on output
- Checking and writing labels.

A user's problem program normally uses LIOCS for file processing (this applies also to programs using POWER and VSAM files). LIOCS uses PIOCS to perform the data transfers. Figure 4.17 (opposite) illustrates the relationship between LIOCS and PIOCS using the GET macro instruction in a user program.

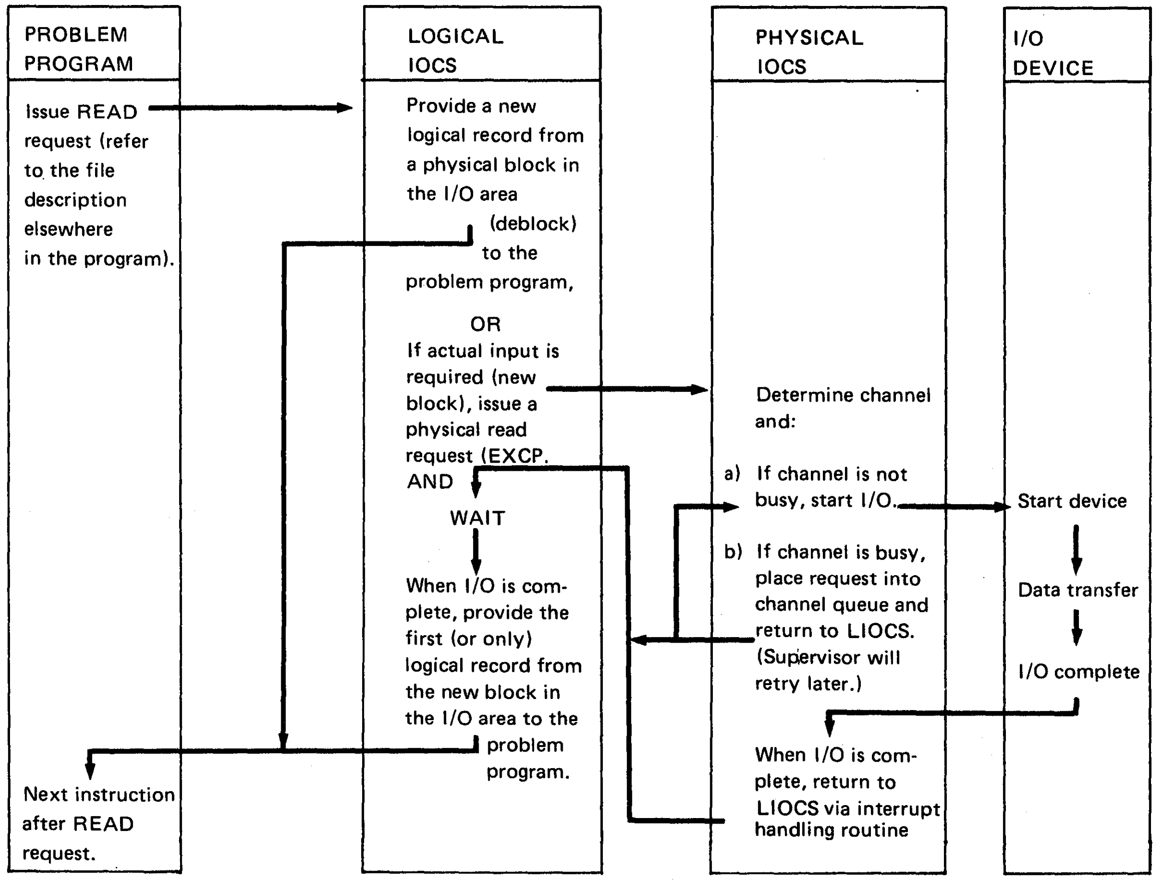


Figure 4.17 Example of LIOCS and PIOCS interrelationship.

Explanation of Figure 4.17:

Logical IOCS makes a request to physical IOCS to start an I/O operation by means of the EXCP macro instruction. From information in the CCB, physical IOCS determines the channel for which the request was made and places the request on a queue for that device. If the channel(s) or device is not busy, the I/O is started and control returns to the problem program. If the channel is busy, control returns to the supervisor task selection routines, but the I/O request waits in the channel queue. When the request reaches the top of the channel queue, the I/O is started.

Control returns to the program requesting the I/O unless there was an error condition detected on the START I/O (SIO) instruction. The problem program normally continues processing until it requires that the requested I/O operation be complete (either the information being read into real storage is needed, or the output area must be freed on an output operation). At this time, the WAIT macro causes the now waiting task to be removed from task selection until the proper interrupt is processed for this device by the supervisor.

I/O CONTROL SYSTEM

Using PIOCS and LIOCS Macro Instructions.

By use of macro instructions you can create, access and maintain files at both physical and logical IOCS levels. Through these macro instructions, the user can communicate with the pre-written routines and tailor them to his needs.

As part of most user programs, LIOCS provides an interface between user programs, LIOCS provides an interface between user's file processing routines and PIOCS. (AII COBOL, FORTRAN, RPG II, PL10PT and PL/I(D) programs use LIOCS; most assembler programs use LIOCS.)

Using PIOCS

Using PIOCS requires a detailed knowledge of device control and system operation. A channel program using the CCW assembler instruction must be written in conjunction with three macro instructions provided to communicate with PIOCS.

CCB: This macro instruction generates a command control block. (Refer to Chapter 5 in this Section for a description of the CCB.)

EXCP: This macro instruction is converted to an SVC0 to request execution of the channel program. It supplies the location of the corresponding CCB to the supervisor.

WAIT: This macro instruction generates an SVC 7 which tests CCB byte 2 bit 0 (traffic bit) to determine when an I/O operation is complete. If the operation is not complete, the supervisor gets control until PIOCS within the supervisor sets the traffic bit to indicate completion of the operation. The WAIT macro should always be used for each I/O operation.

The example below shows part of an assembly program listing using the EXCP WAIT and CCB macros. A full description of these MACRO instructions can be found in *DOS/VS Supervisor and I/O Macros.*

```
003B00 47F0 B75C       03F5E         261          B      SKIP
                                     262 ****************************************************************************
                                     263 *BUG GENERATOR ROUTINE*
                                     264 ****************************************************************************
003B04 5010 C61E             04E20   265 BUG      ST     1,R1SAVE            SAVE CONTENTS OF RETURN REGISTER
                                     266          EXCP   TCCB1
                                     270          WAIT   TCCB1
                                     276          EXCP   TCCB2
                                     280          WAIT   TCCB2
003B30 9640 C83F       05041         286          OI     OPRESP1,X'40'
003B34 95D3 C83F       05041         287          CLI    OPRESP1,C'L'
```

(Ex 83 RR)

```
004E8C 40404040                      1347 TOT      DC     CL4' '
004E90 4040404040404040              1348 TOWN     DC     CL10' '
                                     1349 TCCB1    CCB    SYSLOG,TCCW1,X'0400'
                                     1360 TCCB2    CCB    SYSLOG,TCCW2,X'0400'
004EBA 000000000000
004EC0 09004F106000001C              1371 TCCW1    CCW    09,MSSG1,X'60',28
004EC8 09004F2C60000020              1372          CCW    09,MSSG2,X'60',32
004ED0 09004F4C6000002F              1373          CCW    09,MSSG3,X'60',47
004ED8 09004F786000001F              1374          CCW    09,MSSG4,X'60',31
004EE0 09004F9A60000021              1375          CCW    09,MSSG5,X'60',33
004EE8 09004FBB60000021              1376          CCW    09,MSSG6,X'60',33
004EF0 09004FDC60000020              1377          CCW    09,MSSG7,X'60',32
004EF8 09004FFC60000020              1378          CCW    09,MSSG8,X'60',32
004F00 0900501C20000025              1379          CCW    09,MSSG9,X'20',37
004F08 0A00504120000001              1380 TCCW2    CCW    10,OPRESP1,X'20',1
                                     1381 ****************************************************************************
                                     1382 *MESSAGE CONSTANTS*
```

(Ex 84 RR)

Using LIOCS

Logical IOCS requires a minimum knowledge of the hardware I/O devices and is easily implemented within the problem program by the coding of macros. This system is also used by most of the high-level languages to control I/O operations.

Two types of macro instructions are available to communicate with LIOCS.

- Imperative Macros

These macros order an action to be performed. For example, the macro GET commands LIOCS to place the next record in the user's problem program area.

- Declarative Macros

These macros supply information about the file and about types of processing the I/O routine will have to perform for the user.

Imperative Macros

The problem programmer issues imperative logical IOCS macro instructions to initiate such functions as opening a file, making records available for processing, writing records that have been processed, and controlling physical device operations. A full list can be found in *DOS/VS LIOCS Vol 1.*

Declarative macros DTF (Define the File) Macros

For each imperative macro issued by the problem program, the assembler program generates an in-line expansion that links the instruction to the DTF table (and consequently, the logic module) for the specified file. As an operand, the imperative macro instruction must always contain the filename in the DTF macro describing the file.

For VSAM files, the DTF macro is replaced by the ACB, EXLST and RPL macros to describe a file.

Whenever logical IOCS imperative macro instructions are used in a problem program to control the transfer of records in a file, that file must be defined by a declarative DTF macro instruction. The DTF macro instruction describes (through various parameters specified by the problem programmer) the characteristics of the logical file, indicates the type of processing for the file, and specifies the virtual storage areas and routines, Figure 4.19 summarizes the various DTF table types supported by DOS/VS. Detailed descriptions of the logical IOCS file definition (DTF) macros and their parameters are described in *Supervisor and I/O Macros.*

When one of these DTF macro instructions is encountered at assembly time, the assembler builds a DTF table tailored to the DTF parameters. The table contains:

- Device CCB
- A V-type address constant used by the Linkage Editor to resolve the linkage to the logic module with this DTF
- Logic indicators; that is, one I/O area, two I/O areas, device type, etc.
- Addresses of all of the areas (except work areas) and control functions used by this device.

Regardless of the method of assembling logic modules and DTF tables (that is, together with the main program or separately), a symbolic linkage results between the DTF table and the logic module. The Linkage Editor resolves this linkage at edit time.

Byte	Bits	Function
0–15 (0–F)		CCB.
16 (10)		X'08' indicates DTF relocated by OPENR.
17–19 (11–13)		Address of logic module.
20 (14)		DTF type (X'10')
21 (15)	0	1 = No rewind.
	1	1 = Unload rewind.
	2	1 = Workfile.
	3	1 = Read backward.
	4	1 = Write.
	5	1 = PCINTW.
	6	1 = Force checking of read or write.
	7	1 = Forward space before next operation.
22–23 (16-17)		Not used.
24-25 (18–19)		Record length.
26–27 (1A–1B)		Maximum BLKSIZE.
28 (1C)		Read op code.
29–31 (1D–1F)		EOF address.
32–39 (20–27)		CCW.
40–43 (28–2B)		Block count, initialized 00000000 for read forward, 00400000 for read backward.
44 (2C)	0	1 = Error routine.
	1	1 = Ignore.
	2	1 = Read next record switch.
	3	1 = Record fixed unblocked.
	4	Not used.
	5	Not used.
	6	Not used.
	7	Not used.
45–47 (2D–2F)		Address of error routine.

Note: Numbers in parentheses are displacements in hexadecimal notation.

Figure 4.18. The format of the DTF table generated by a DTFMT declarative macro for a DTFMT workfile.

An example of an assembly program listing is shown in Figure 4.22 that shows the expansion of a DTFMT macro. The macro expansion was obtained by the use of the assembly control statement PRINT GEN (a useful aid to use when in doubt). Figure 4.23 shows how this same DTFMT is printed in a system dump. The table of Figure 4.19 (opposite) lists all the DTF codes and relates them to their specific files.

(Byte 20) of DTF Table	DTF	Description
X'00'	DTFCD	Combined files
X'01'	DTFPT	Paper tape files
X'02'	DTFCD	Reader and 3881 Optical Mark Reader files
X'03'	DTFCN	Console
X'04'	DTFCD	Punch files
X'08'	DTFPR	Printer files
X'09'	DTFOR	Optical Reader files except 3881 files
X'0A'	DTFOR	Optical Reader files (HEADER=YES)
X'0B'	DTFMR	Magnetic Ink Character Recognition (MICR) and Optical Reader/Sorter files
X'0C'	DTFDR	3886 Optical Character Reader files
X'10'	DTFMT	Magnetic tape workfiles
	DTFCP	Magnetic tape workfiles (compiler). (Note 1)
X'11'	DTFMT	Nonstandard or unlabeled tape files
X'12'	DTFMT	Standard labeled, output tape files
	DTFPH	Standard labeled, output tape files (physical IOCS)
X'13'	DTFMT	Standard labeled, input tape files (read backward)
X'14'	DTFMT	Standard labeled, input tape files (read forward)
X'1A'	DTFDU	Diskette I/O Unit files
X'20'	DTFSD	Sequential DASD workfiles and data files
	DTFCP	DASD workfiles (compiler)
X'21'	DTFPH	Sequential DASD files, MOUNTED=SINGLE (physical IOCS)
X'22'	DTFDA	Direct access files
X'23'	DTFPH	Direct access files, MOUNTED=ALL (physical IOCS)
X'24'	DTFIS	Indexed sequential, LOAD file
X'25'	DTFIS	Indexed sequential, ADD file
X'26'	DTFIS	Indexed sequential, RETRVE file
X'27'	DTFIS	Indexed sequential, ADDRTR file
X'28'	ACB	Access Method Control Block for VSAM files
X'30'	DTFCP	Compiler file for DOS Version 1 (Note 1)
X'31'	DTFCP	Compiler file for DOS Versions 2 and 3
X'32'	DTFCP	Compiler file for DOS Versions 2 and 3 (Note 2)
X'33'	DTFDI	Device independent system unit files
X'40'	DTFBT	Basic Telecommunications Access Method (BTAM) file (Notes 3 and 4)
X'50'	DTFQT	Queued Telecommunications Access Method (QTAM) file (Notes 3 and 4)
X'60' through X'67'		Reserved

Note 1: *DTF type is X'30' except for tape or DASD assigned to units SYS000 to SYSnnn. In this case, the DTFCP open phases change the DTF type to X'10' for tape workfiles, or X'20' for DASD workfiles.*

Note 2: *DTF type is X'32' except for DASD assigned to units SYS000 to SYSnnn. In this case, the DTFCP open phases change the DTF type to X'20' for DASD workfiles.*

Note 3: *The following control unit codes are ORed into the low-order 4 bits of the DTF type code.*

Control Unit	*Code*
7770	*1*
2848	*3*
2701	*4*
2702	*5*
2703	*6*

Note 4: *The DTF tables for BTAM and QTAM files are not documented in this manual. They are documented in the respective publications DOS/VS BTAM Logic and DOS/VS QTAM Logic.*

Note 5: *VSAM differs from other DOS/VS access methods in that it does not use a DTF. The declarative macro equivalent for VSAM is the ACB. (Access Control Block).*

Figure 4.19. DTF type codes.

MOD (Module Generation) Macros

To speed up assembly time and save storage, LIOCS uses another type of declarative macro instruction. Called logic module generation macros, declaratives of the form xxMOD describe the type of processing that the I/O routines will have to do for a particular file. A module is generated that handles only what the user has specified.

The module generation macros generate the data handling logic modules. These modules contain generalized routines needed to perform the functions of the logical IOCS imperative macros. The generalized routines in the logic modules are altered and made more specific through various parameters (specified by the problem programmer) included in the xxMOD macro statements. It is possible, therefore, to generate many variations of a particular type of logic module, each specifically suited to the need of the problem programmer.

At assembly time, the assembler produces an EXTRN (External Symbol) card for every V-type constant (or EXTRN statement) in the user program. The assembler expansion of the DTF statement produces an EXTRN card with the name of the logic module needed to support the parameters that were specified in the DTF macro. The IBM-generated module names indicate the type of file and the support that each is capable of supplying for the DTF. Refer to Figure 4.20 for a breakdown of these names. Because of the descriptive nature of the IBM standard names, the programmer should be careful when specifying his own names for the logic modules to avoid overriding the IBM standard names. At the time this program is link-edited, the linkage editor resolves these EXTRN symbols (AUTOLINK). If the program is not to be executed immediately, option CATAL causes the linkage editor catalogs the program into the core image library.

I	H J K L	xxxxxx
DATA MANAGEMENT	IJB	System Service and System Control
	IJC	Card logic
	IJD	Printer logic
	IJE	Paper tape logic
	IJF	Magnetic tape logic
	IJG	Sequential DASD logic
	IJH	Indexed Sequential DASD logic
	IJI	Direct Access DASD logic
	IJJ	Device independent logic
	IJL	Teleprocessing routines
DATA MANAGEMENT	IJM	Optical Reader logic
	IJN	Diskette I/O Unit logic
	IJU	Magnetic Readers logic
	IJW	Utilities
	IJZ	OLTEP
	IKQ	VSAM

Figure 4.20. A list of module names and their prefixes.

Reentrant modules

A re-entrant module is a logic module that can be used asynchronously, or shared by more than one file. The RDONLY (read only) parameter implies that the generated logic module is never modified in any way, regardless of the processing requirements of any file(s) using the module. To provide this feature, unique save areas external to the logic module are established, one for each task using the module. Each save area must be 72 bytes and doubleword aligned. Before a logic module is entered or an imperative macro is issued to the file, the task must provide the address of its unique save area in register 13.

Interrelationships of the DTF and Module Macro Instructions

The DTFCD, DTFDA, DTFDI, DTFDU, DTFMR, DTFMT, DTFOR, DTFPR, DTFPT, and DTFSD declarative macros are similar in that each of them generates a DTF table that references an IOCS logic module. The first 16 bytes of each table have the same format, that is, a command control block (CCB) and bytes 17 to 19 contain a logic module address. The length of each table depends on the particular device and file type. Figure 4.19 lists the DTF device codes. To accomplish the linkage between the DTF table and the logic module, the assembler generates a V-type address constant in the DTF of a named CSECT in the logic module. To resolve this linkage, the linkage symbols (module names) must be identical. The Figure below shows the relationship of the program, the DTF, and the logic module. The assumed parameters have generated a request for a MTMOD named IJFFBCWZ. Based on this name, the linkage editor was able to locate the module. The GET statement generated coding to load the address of the DTF table into register 1. This gives the program access to the MTMOD address, and the program branches to the required routine within the module.

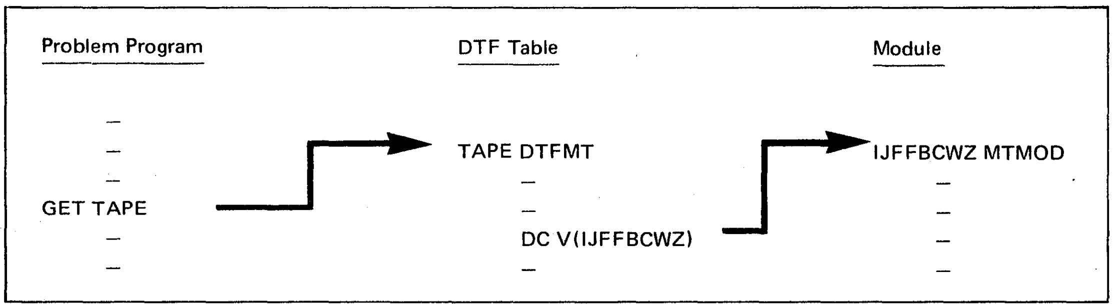


Figure 4.21 The relationship between imperative and declarative macros.
See also Figure 4.24

```
                                                                                              PAGE   15

   LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                   DOS/VS ASSEMBLER REL 29.0 20.02 73-10-05
                                    664 TAPEOUT  DTFMT                                                  C
                                                       BLKSIZE=800,                                     C
                                                       DEVADDR=SYS003,                                  C
                                                       FILABL=NO,                                       C
                                                       IOAREA1=RITETAPE,                                C
                                                       RECFORM=FIXBLK,                                  C
                                                       TYPEFLE=OUTPUT,                                  C
                                                       WORKA=YES,                                       C
                                                       RECSIZE=80
                                    665+* MAGNETIC TAPE IOCS - DTFMT - 5745-SC-TAP - REL. 28.0                  02250028
 003FA8                             666+          DC   0D;0;                                            3-8 29400025
 003FA8 000080000000                667+TAPEOUT   DC   X;000080000000;      CCB                             30300025
 003FAE 01                          668+          DC   AL1(1)               LOGICAL UNIT CLASS              30400025
 003FAF 03                          669+          DC   AL1(3)               LOGICAL UNIT                    30500025
 003FB0 00003FE0                    670+          DC   AL4(IJF10060)        CCW ADDRESS                 3-8 31000025
 C03FB4 00000000                    671+          DC   4X;00;               CCB-ST BYTE,CSW CCW ADDRESS 3-8 31100025
 003FB8 00                          672+          DC   AL1(0)                                           3-8 31800025
 003FB9 000000                      673+          DC   VL3(IJFFZZWZ)        ADDRESS OF LOGIC MODULE     3-8 38500025
 003FBC 11                          674+          DC   X;11;                DTF TYPE                        39500025
 003FBD 50                          675+          DC   AL1(80)              LOGICAL IOCS SWITCHES           41300025
 003FBE E3C1D7C5D6E4E340            676+          DC   CL8;TAPEOUT;                                         41800025
 003FC6 01                          677+          DC   X;01;                                                42300025
 003FC7 60                          678+          DC   AL1(96)          SWITCHES FOR OPEN               3-8 45200025
 C03FC8 00                          679+          DC   AL1(0)               SWITCH ONE FOR OPEN AND CLOSE   47200025
 003FC9 000000                      680+          DC   AL3(0)               USER LABEL ROUTINE              47700025
 003FCC 00                          681+          DC   AL1(0)         SWITCH FOR OPEN AND CLOSE             48600025
 003FCD 003FCD                      682+          DC   AL3(*)                                               49000025
 C03FD0 00000000                    683+          DC   F;0;                 BLOCKCOUNT                      49100025
 003FD4 86BC F018           00018   684+          BXH  11,12,24(15)         DEBLOCKING FORWARD              49700025
 003FD8 41EE 0001           00001   685+          LA   14,1(14)             INCREASE BLOCKCOUNT BY ONE      49800025
 003FDC 4700 0000     00000         686+          NOP  0(0)                 LOAD USER IOREG                 50600025
 C03FE0 010047A120000320            687+IJF10060       CCW   X;01;,RITETAPE,X;20;,800                       55600025
 003FE8 000047A1                    688+          DC   A(RITETAPE)          ONE IOAREA                 3-10 56160026
 003FEC 000047A1                    689+IJF20060       DC    A(RITETAPE)    DEBLOCKER 1                3-10 52050026
 003FF0 00000050                    690+          DC   F;80;                DEBLOCKER 2                3-10 52100026
 003FF4 00004AC0                    691+          DC   A(RITETAPE+800-1)    DEBLOCKER 3                3-10 52150026
 003FF8 0320                        692+          DC   Y(800)               BLOCKSIZE                       52400025
 003FFA 031F                        693+          DC   Y(800-1)             BLOCKSIZE-1                     52600025
 C03FFC 004F                        694+          DC   Y(80-1)              RECSIZE-1                   3-8 52900025
                                    695           MTMOD WORKA=YES
                                    696+*  MAGNETIC TAPE IOCS - MTMOD - 5745-SC-TAP - REL. 29.0          fDL29ZCN 00280029
 000000                             697+IJFFDTF   DSECT                                                     03160025
 000000 0000000000000000            698+IJFFNM    DC   4F;0;                CCB                             03200025
                            00003   699+IJFFCB2   EQU  IJFFNM+3             COMMUMICATION BYTE 2            03240025
 000010 00000000                    700+          DC   A(0)                 ADDRESS OF LOGIC MODULE         03480025
 000014 10                          701+          DC   X;10;                DTF TYPE                        03520025
 000015 00                          702+IJFFSWI   DC   X;00;                LOGICAL IOCS SWITCHES           03560025
 000016 C6D5C1D4C5404040            703+          DC   CL8;FNAME   ;                                        03600025 (Ex 85)
```

Figure 4.22. **An example of an assembly listing showing the expansion of a DTFMT macro instruction.**

The program was assembled using the PRINT GEN assembler control statement

```
            040760  D20C8977 CE3CD202 B983C683 5810CDAE   58F10010 45EF000C 5810CDAE 4100000B   K.....K...F.....  .1..............
            040780  58F10010 05EFD283 B936B935 D20CB93B   CE49D203 B948C686 D206B959 CE56D203   .1....K.....K...  ..K...F.K.....K.
  DTFMT     0407A0  B960C68A 5810CDAE 58F10010 45EF000C   47F0B75C 4110CD86 4500B746 00040878   .-F......1......  .0.*............
   for      0407C0  0A020700 4110CD7E 4500B756 00040878   0A0247F0 B24C5810 CDAE4100 008B58E1   ................  ...0...........1
  TAPEOUT   0407E0  001005EF 5840CDBA 47F0B24C 5810CDAE   4100008B 58F10010 05EF[illegible] B73A0700   ................  .......1........
            040800  4110CD86 4500B792 00040878 0A020700   4110CD86 4500[illegible] [illegible] 0A020A0E   ................  ................
            040820  00008000 0C000103 00040858 00040860   00041ED8 1150E3C1 D7C5D6E4 E3400160   ..............-   ...Q.TAPEOUT .-
            040840  00000000 00040845 00000000 86BCF018   41EE0001 47000000 01041019 20000320   ...............0.  ................
            040860  00041019 00041019 00000050 00041338   0320031F 004F0000 00008000 0C000103   ...........&....  ................
            040880  00040880 000408B8 00041ED8 11D8E3C1   D7C5C9D5 40400270 00000000 2C0407EC   ...........Q.QTA  PEIN  ..........
            0408A0  0000000F 86BCF018 41EE0001 47000000   02040CF9 00000320 00040CF9 00040E39   ......0.........  ...9.......9....
            0408C0  00000050 00041018 0320031F 004F0000   47FF001C 47FF001C 0000C000 0D000000   ...&............  ................
            0408E0  000408F8 00040900 00042230 02000202   00040960 0004021A 02040960 20000050   ...8............  ...-.......-...& (Ex 86 KT)

                                                                                            DTF
                                                                                            file name
```

Figure 4.23. **An example showing how the same DTFMT is printed in a dump.**

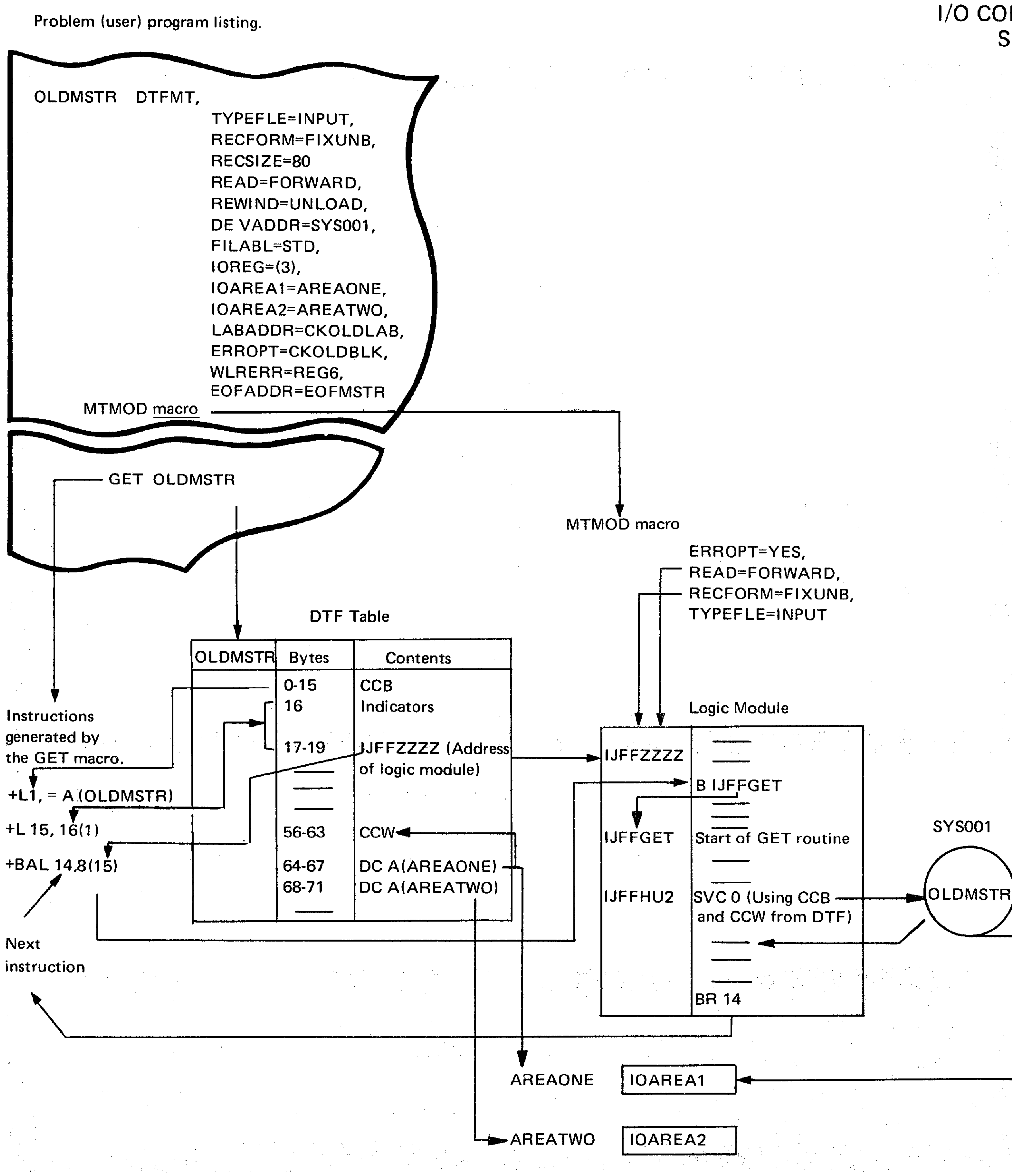


Figure 4.24. **A summary of the relationships between an imperative macro, a declarative macro, and a module generation macro specified in a program.**

The GET imperative macro is used in this illustration, which also shows the linkage between the generated DTF table and the logic module.

VSAM (Virtual Storage Access Method) I/O

VSAM IOCS differs from that of other DOS/VS access methods as follows:

- VSAM declarative macros are ACB, EXLST, and RPL instead of DTF and xxMOD.
- VSAM routines are dynamically loaded into virtual storage when a VSAM file is opened. They are not assembled or link edited with the user's program.

Declarative Macros

The VSAM declarative macros are ACB, which creates an Access-Method Control Block; EXLST, which creates an Exit List; and RPL, which creates a Request Parameter List. The Access-Method Control Block (ACB) is like a DTF in that it defines the file to be processed. Opening a VSAM file involves opening the ACB for that file. The Request Parameter List (RPL) defines the parameters necessary for a particular execution of a request (imperative) macro. It contains some of the information, such as address of the user's work area, located in the DTF in other access methods. The Exit List (EXLST) contains the addresses of optional user exit routines. Up to four exit routines can be specified--one for handling end-of-file, one for handling logical errors, one for handling I/O errors, and one to allow user processing during VSAM I/O operations.

Codes indicating errors resulting from execution of imperative macros are set in registers or in the ACB or RPL as described below.

Imperative Macros

The user's program issues imperative macros to open or close a file and to retrieve, add, delete, or update records. It can also issue imperative macros to generate, modify, display, or test the control blocks created by the declarative macros. When control is returned to the user's program after execution of an imperative macro, a "return code" is set in the low-order byte of register 15. The return code indicates the results of the macro execution. If an error or certain other exceptional conditions occur, an "error code" will be set in the ACB, the RPL, or in register 0, depending on the macro. Figure xx summarizes the return codes and error codes issued by the imperative macros and user exit routines which can be used. More information on the return codes and user exits as well as a complete list of the error codes and their meanings is in the VSAM chapter of *DOS/VS Supervisor and I/O Macros.*

An ACB, EXLST, or RPL can be created dynamically, during program execution, by using the GENCB macro. The fields in these control blocks can be modified during program execution by using the MODCB macro. Refer to *DOS/VS Supervisor and I/O Macros* for information on how to write the GENCB and MODCB macros.

RPL Debugging Hints

If the RPL hold byte, 35(23), is set to X'FF', the error occurred while the request was being executed by VSAM. Check the type of request byte, 29(1D), to determine what request was active. If the request was a POINT, GET, or PUT, check the following parameters in the RPL (of GENCB for RPL) in your program to ensure that they are valid:

Macro	Check these RPL Parameters
POINT	ARG and KEYLEN
GET	AREA and AREALEN IF OPTCD=DIR or OPTCD=SKP, also check ARG and KEYLEN
PUT	AREA and RECLEN

If the RPL parameters are specified correctly or if the type of request is other than POINT, GET, or PUT, the error is probably in VSAM itself. Save the dump, console log, and program listing and contact your IBM programming support representative.

Note: MODCB, SHOWCB, and TESTCB macros also set the RPL Hold byte. If this byte was set by one of those macros, the type of request byte will have no meaning.

Imperative Macro	Return Code found in	If Return Code no 0, Error Code found in	Method of Inspecting Error Code	Exits Taken if User Exit Routines are Supplied
OPEN CLOSE TCLOSE	Register 15	ACB	Code ERROR parameter in TESTCB** or SHOWCB	SYNAD EXIT FOR I/O errors in CLOSE; code FDBK in TESTCB** or SHOWCB
GET PUT POINT ERASE ENDREQ	Register 15	RPL	Code FDBK parameter in TESTCB** or SHOWCB	LERAD exit for logical errors* (Return Code is X'08') SYNAD exit for I/O errors (Return Code is X'0C')
GENCB SHOWCB TESTCB MODCB	Register 15	Register 0	Inspect Register 0	None

* End-of-file is indicated by an error code of X'04'. The EODAD exit is taken if an EODAD routine is supplied. Otherwise, the LERAD exit is taken.

** A user routine can be supplied to receive control if NSAM is unable to test for the condition specified because of an error occurring during execution of the TESTCB macro. The routine is pointed to by the ERET parameter of TESTCB.

Figure 4.25. Summary of Error Checking for VSAM Imperative Macros

Error Detection with VSAM Macros

When an error occurs, the user can either attempt to correct it in his program, close the file, or terminate the job. These actions can be taken in the SYNAD and LERAD exit routines or in-line in the program. If the user wants to evaluate the error condition after the job has finished, he should write the error code in a field in his program (for locating it in a dump output) or in a message. He obtains the error code from the ACB or the RPL by issuing either a TESTCB or a SHOWCB macro.

The SHOWCB macro is used to display the contents of an ACB, EXLST, or RPL in a work area specified by the user. The contents include the fields coded for each block by the user. An ACB also includes fields, such as number of levels in the index, read from the file's catalog entry when the file is opened. The TESTCB macro is used to test the value of a field or a combination of fields in the ACB, EXLST, or RPL. See the *Supervisor and I/O Macros* publication for information on how to write the SHOWCB and TESTCB macros.

The SHOWCB and TESTCB macros can also be used when a task terminates abnormally (such as through a program check). The VSAM macro can be included in a routine called by the STXIT macro instruction.

VSAM Control Blocks

When a VSAM file is opened, VSAM uses the information supplied by the user in the ACB, EXLST, and RPL along with information in the file's catalog entry to construct VSAM control blocks. The contents of the EXLST, RPL and ACB control blocks are shown in Figures 4.25.B, C and D. Additional control blocks internal to VSAM are pointed to by the ACB and RPL and are described in *DOS/VS LIOCS Logic, Volume 4: VSAM.*

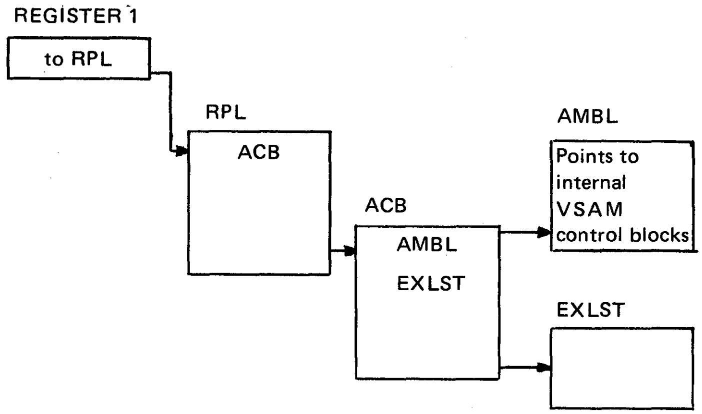


Figure 4.25.A. Relationship of VSAM Control Blocks

Displacement dec (hex)	Length in bytes	Description and/or Contents
0(0)	1	Control block identifier – X'81'
1(1)	1	EXLST active byte. Set to X'FF' when file is open.
2(2)	2	Length of EXLST. The length will be 10 bytes, 15 bytes, or 20 bytes.
4(4)	1	Reserved
5(5)	1	EODAD entry flags: X'80' – Entry present X'40' – Entry active X'20' – Entry to be dynamically loaded
6(6)	4	Address of the user's EODAD exit routine. Set from the EODAD parameter of the EXLST macro.
10(A)	1	SYNAD entry flags: X'80' – Entry present X'40' – Entry active X'20' – Entry to be dynamically loaded
11(B)	4	Address of the user's SYNAD exit routine. Set from the SYNAD parameter of the EXLST macro.
15(F)	1	LERAD entry flags: X'80' – Entry present X'40' – Entry active X'20' – Entry to be dynamically loaded
16(10)	4	Address of the user's LERAD exit routine. Set from LERAD parameter of the EXLST macro.
20 (14)	1	EXCPAD entry flags: X'80' – Entry present E'40' – Entry active X'20' – Entry to be dynamically loaded
21 (15)	4	Address of the user's EXCPAD exit routine. Set from EXCPAD parameter of the EXLST macro.

Figure 4.25.B. Explanation of the contents of the EXIT LIST (EXLST)

Displacement dec (hex)	Length in bytes	Description and/or Contents
0(0)	1	Control block identifier – X'00'
1(1)	1	X'00'
2(2)	2	Length of RPL
4(4)	4	Relative Byte Address (RBA) of last record process or DD field indicator.
8(8)	4	Address of the key or RBA which is the search argument. Set from ARG parameter or RPL or GENCB macro.
12(C)	4	Address of user's work area which contains the logical record. Set from AREA parameter of RPL or GENCB macro.
16(10)	4	Length of the last logical record processed. Set from the RECLEN parameter of RPL or GENCB macro and modified (for variable length records) by the user for a PUT macro and by VSAM for a GET macro.
20(14)	4	Length of the user work area. Set from the AREALEN parameter of RPL or GENCB macro.
24(18)	4	Address of the ACB
28(1C)	1	X'01'
29(1D)	1	Type of current request: X'00'– POINT macro X'04'– GET macro X'08'– ERASE macro X'0C'– PUT macro (update) X'10'– PUT macro (insert) X'18'– Internal VSAM request X'IC'– ENDREQ macro X'20' to X'24'– internal VSAM requests
30(1E)	2	Key length
32(20)	1	First byte of option codes (set from OPTCD parameter of RPL of GENCB macro): X'80'– Keyed access X'40'– Addressed access X'20'– Sequential processing X'10'– Direct processing X'04'– Skip sequential processing X'02'– Control interval access X'01'– Update
33(21)	1	Second byte of option codes: X'80'– Search key greater than or equal X'40'– Generic key request X'20'– Note string position X'10'– No update X'08'– Locate mode X'04'– User buffer processing
34(22)	1	RPL available byte. If set to X'FF'' the RPL is not available for a request. If set to X'00', the RPL is available for a request.
35(23)	1	RPL hold byte. If set to X'FF', the RPL is active (a request has been started but has not been completed). If set to X'00', the RPL is not active (no requests are in process).
36(24)	1	X'FF'
37(25)	1	Return code (also set in Register 15). See Supervisor and I/O Macros for a description of return codes.
38(26)	1	Reserved
39(27)	1	Error code (describes the type of error detected by VSAM). See Supervisor and I/O Macros for a description of the error codes.
40(28)	4	Address of the Placeholder (PLH). This is an internal VSAM control block and is described in DOS/VS LIOCS Logic Volume 4: VSAM.
44(2C)	8	Reserved

Figure 4.25.C. Explanation of the contents of the REQUEST PARAMETER LIST (RPL)

Displacement dec (hex)	Length in bytes	Description and/or Contents
0(0)	1	Control block identifier – X'A0'
1(1)	1	ACB active byte. Set to X'FF' when ACB is open.
2(2)	2	Length of ACB
4(4)	4	Address of the Access Method Block List (AMBL). This is an internal VSAM control block and points to other internal VSAM control blocks. They are described in *DOS/VS LIOCS Logic Volume 4: VSAM.*
8(8)	4	Address of the VSAM load module (VSAM routines).
12(C)	2	Reserved
14(E)	2	Number of data buffers
16(10)	2	Number of index buffers
18(12)	1	MACRF first byte: X'80' – Keyed access X'40' – Addressed access X'20' – Control interval access X'10' – Sequential processing X'08' – Direct processing X'04' – GET macro X'02' – PUT macro X'01' – User buffers
19(13)	1	MACRF second byte: X'20' – Skip sequential processing
20(14)	1	DOS/VS DTF identifier – X'28'
21(15)	1	Open/Close flags X'10' – ACB is open X'01' – ACB will accept keyed as well as RBA requests Other Open/Close flags are internal to VSAM
22(16)	1	X'01'
23(17)	1	Error code. This code describes errors that occur during Open Close, or TCLOSE. The codes are described in *Supervisor and I/O Macros.*
24(18)	4	Size of VSAM's buffer pool
28(1C)	8	Filename of ACB. Set from label field of ACB macro or DDNAME parameter of ACB or GENCB macro.
36(24)	4	Address of password. Set from PASSWD parameter of ACB or GENCB macro.
40(28)	4	Reserved
44(2C)	4	Address of VSAM's buffer pool
48(30)	4	Address of Exit List (EXLST)
52(34)	4	Reserved

Figure 4.25.D. Explanation of the contents of the Access-Method Control Block (ACB)

Note: ACB. The ACB macro produces an Access Method Control Block (ACB) for a VSAM file. The control block identifies the key-sequenced file and its index or the entry-sequenced file that is to be processed, and indicates the types of requests that are to be made. The ACB is similar to a DTF in that it identifies the file to be processed. However, most information about the file, such as key length and record format, is specified in the Access Method Services' (AMS) DEFINE command. Information supplied in this command resides in the VSAM catalog and is read into storage when the ACB is opened.

Command Control Block (CCB)

This information block is generated in the problem program during assembly or during program operation, depending on the methods of I/O control employed by the program. As described in Chapter 4 the CCB is generated as the first 16 bytes of a DTF when the program is using LIOCS. When using PIOCS, the CCB macro generates the CCB.

The CCB establishes communication between the problem program and physical IOCS. The CCB is 16 bytes in length with eight major fields, and does not have to be aligned on a doubleword boundary. Eight optional bytes are generated if the user requests that a sense operation be performed on the occurrence of an I/O error. Data transferred from the device to real storage during a sense operation provides information concerning unusual conditions detected in the last operation and the status of the device. All data in the CCB is in the hexadecimal format.

By examining the contents of the CCB in a dump, the following information can be obtained about the associated I/O operation:

- Whether the operation was completed (by inspecting the traffic bit and device-end bit)
- Status of the channel and device to which the I/O command was issued
- The logical unit involved in the operation
- Whether the CCB is in a real or a virtual partition
- The address of the channel program (the first CCW in a CCW string)
- The address of the next CCW to be executed in the channel program (Subtracting eight from this address gives the address of the last CCW used.)
- The residual count associated with the last CCW.

This count taken from the channel status word (CSW), is stored by PIOCS when the pointer to this CCB is removed from the channel queue. The residual count, in conjunction with the original count specified in the last CCW used, indicates the number of bytes transferred to or from the area designated by the CCW. When an input operation is terminated, the difference between the original count in the CCW and the residual count in the CSW is equal to the number of bytes transferred to storage. For an output operation, the difference is equal to the number of bytes transferred to the I/O device.

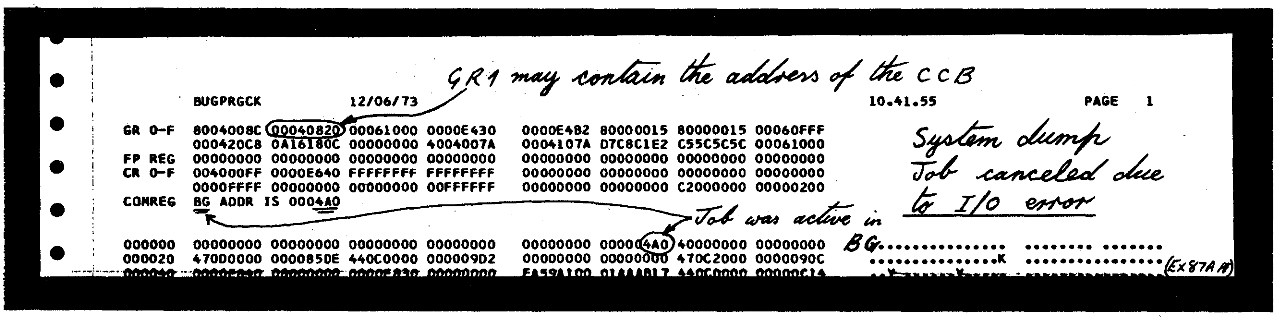
CCB AND THE CHANNEL PROGRAM

How to locate

1. For programs using LIOCS, locate the address of the associated DTF in the program listing. Then use the linkage editor map to locate the DTF in the dump. The first 16 bytes of the DTF is the CCB.

2. For programs using PIOCS, locate the address of the CCB macro in the program listing and use the linkage editor map to locate the CCB in the dump.

3. If the interrupt code in the PSW stored in the partition save area is 00 or 07 (SVC 0 or SVC 7), the contents of general register 1 may contain the address of the CCB. To confirm whether the address in register 1 is that of a CCB, inspect the first few bytes starting from that address. (It is not difficult to recognize a valid CCB in a dump. See the example below.)



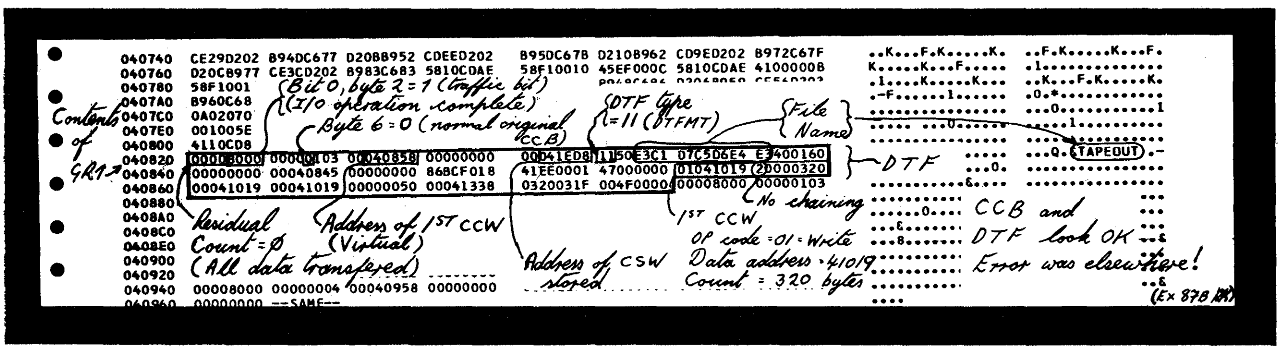


Figure 4.26 The pointer and CCB in a dump

Figure 4.27 parts 1, 2 and 3 illustrate the format and contents of the information contained in any CCB.

Count	Transmission information	CSW Status Bits	Type Code	Reserved for logical IOCS	CCW Address	Reserved for 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 Used for residual Count.			
2-3 Transmitting information between Physical IOCS & Problem Program	Byte 2		Set on by:
	Bit 0:	Traffic Bit (Wait)/(Note 5)	PIOCS*
	Bit 1:	End of File(/* or /&); 3211-UCSB Parity Check (Line Complete). (Note 2)	PIOCS
	Bit 2:	Irrecoverable I/O error	PIOCS
	Bit 3:	Accept Irrecoverable I/O error	Pr. Pr.**
	Bit 4:	5425 not ready, or return DASD Data Checks, 2671 errors, 3540 Diskette Data Checks, or 1017/1018 errors to the user; indicate action-type message for Video Display Unit	
	Bit 5:	Post at Device End (Note 5)	Pr. Pr.
	Bit 6:	Return Tape Read Data Check; 1018 or 2560 Data Check 2520, 2540, 2560, 3881 or 5425 Equipment Check Accept 3504, 3505 or 3525 Perm. Error, DA SD-Data Checks on Read or Verify Command on 3211 or 2245 Passback Requested. (Notes 3, 6 and 8)	
	Bit 7:	User Error Routine	Pr. Pr.
	Byte 3		Set on by:
	Bit 0:	DASD-Data Check in Count Area; Permanent Error for 3330, 3340; MICR-SCU Not Operational; 1287/1288-Data Check; 3211-Print Check/Equipment Check; 3540 Special Record Transferred.	PIOCS
	Bit 1:	DASD-Track Overrun; MICR-Intervention required; 1287-Keyboard Correction in Journal Tape Mode; 1017-Broken Tape; 3211-Print Quality/Equipment Check.	PIOCS
	Bit 2:	DASD-End of Cylinder; MICR-(Note 4); 1287/1288-Hopper Empty in Document Mode. 3211/2245-Line Position Error. (Note 7)	
	Bit 3:	2520, 3881-Equipment Check; 2560, 3203, 5203, 5425 Data Check/Equipment Check; Tape-Read Data Check; DASD-Any Data Check. 1287-Equipment Check; 1017/1018-Data Check; 3211-Print Check/Data Check; 3504, 3505, 3525 Perm. Error, (Note 8); 3540 Diskette Data Check.	PIOCS
	Bit 4:	Non-Recovery Questionable Condition: Card-Unusual Command Sequence; DASD- No Record Found; 1287/1288- Document Jam or Torn Tape; 3211-UCSB Parity Check (Command retry); 5425 not ready	PIOCS
	Bit 5:	No Record Found Condition (Retry on 2311, 2314, 2319, 3330 or 3340)	Pr. Pr.
	Bit 6:	Carriage Channel 9 Overflow or Verify Error for DASD; 1287-Document Mode-Late Stacker Select; 1288-End of Page.	PIOCS
	Bit 7:	Command Chaining, Retry from the next CCW to be executed	Pr. Pr

*Physical IOCS **Problem Program

Figure 4.27 Explanation of the contents of the CCB, part 1 of 3

CCB AND THE CHANNEL PROGRAM

Count	Transmission information	CSW Status Bits	Type Code	Reserved for logical IOCS	CCW Address	Reserved for 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	
4-5 CSW Status Bits	Byte 4 (Note 1)	Byte 5
	Bit 0(32) Attention 1(33) Status Modifier 2(34) Control Unit End 3(35) Busy 4(36) Channel End 5(37) Device End 6(38) Unit Check 7(39) Unit Exception	Bit 0(40) Program Controlled Interruption 1(41) Incorrect Length 2(42) Program Check 3(43) Protection Check 4(44) Channel Data Check 5(45) Channel Control Check 6(46) Interf. Control Check 7(47) Chaining Check
6-7 Type Code	Byte 6	
	X'0u' Original CCB (Bytes 9-11 and 13-15 contain virtual addresses X'2u' Translated CCB (Byte 9-11 contain real address, bytes 13-11 virtual address) X'4u' BTAM request original CCB (Bytes 9-11 and 13-15 contain virtual addresses) X'6u' BTAM request translated CCB (Bytes 9-11 contain real address, bytes 13-15 virtual address) X'8u' User-translated CCB in virtual partition (Bytes 9-11 and 13-15 contain real addresses) Note: Any one of the above incremented by X'10' (bit 3 on) indicates automatic switching to the beginning of the next cylinder at End of Cylinder condition. u: 0= The address in byte 7 refers to a System Logical Unit. 1= The address in byte 7 refers to a Programmer Logical Unit.	
	Byte 7	
	Hexadecimal representation of SYSnnn: SYSRDR = 00 SYSIPT = 01 SYSPCH = 02 SYSLST = 03 SYSLOG = 04 SYSLNK = 05 SYSRES = 06 SYSSLB = 07 SYSRLB = 08 SYSUSE = 09	SYSREC = 0A SYSCLB = 0B SYSVIS = 0C SYSCAT = 0D SYS000 = 00 SYS001 = 01 SYS002 = 02 SYSnnn (Note 9)

Figure 4.27. Explanation of the contents of the CCB, part 2 of 3

Count	Transmission information	CSW Status Bits	Type Code	Reserved for logical IOCS	CCW Address	Reserved for 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
8 Reserved for Logical IOCS	Buffer Offset ASCII Input Tapes X'00'--X'63' ASCII Output Tapes Fixed X'00' Variable X'00' or X'04' Undefined X'00'
9-11 CCW Address	Virtual or real address of CCW associated with this CCB depending on byte 6: Real address if byte 6= X'2u', X'6u', or X'8u'; Virtual address if byte 6= X'0u', or X'4u'.
12 Reserved for Physical IOCS	X'80' CCB being used by ERP X'40' Channel Appendage Routine present for TP device, VSAM or POWER X'20' Sense Information desired X'10' Message writer X'08' EU Tape Error X'04' OLTEP Appendage available X'02' Tape ERP Read Opposite Recovery X'01' Seek Separation
13-15 CCW Address in CSW	Virtual Address of CCW pointed to by CSW at Channel End (if byte 6= X'8u', it is the real address) or address of the Channel End Appendage Routine for TP devices, VSAM or POWER.
16-23 Optional Sense CCW	8 bytes appended to the CCB when Sense Information is desired.

Note 1: Bytes 4 and 5 contain the status bytes of the Channel Status Word (Bits 32-47). If byte 2, bit 5 is on and device end results as a separate interrupt, device end will be OR-ed into CCB byte 4.

Note 2: Indicates /* or/& statement on SYSRDR or SYSIPT. Byte 4, bit 7 (unit exception) is also on.

Note 3: DASD data checks on count not returned.

Note 4: For 1255/1259/1270/1275/1419, disengage. For 1275/1419D, I/O Error is external interrupt routine (Channel data check or bus-out check).

Note 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. Also see Note 1.

Note 6: 1018 ERP does not support the Error Correction Function.

Note 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.

Note 8: For 3504, 3505, 3525 input or output files using ERROPT, byte 3-bit 3 is set on if a permanent error occurs. Byte 2-bit 6 is set on to allow you to accept permanent errors.

Note 9: SYSnnn= 255–(Number of partitions x 14).

Figure 4.27. Explanation of the contents of the CCB, part 3 of 3.

CCB AND THE CHANNEL PROGRAM

The channel program

A channel program consists of one or more CCWs (channel command words). The channel program is generated during assembly or during program operation, depending on the method of I/O control employed by the program. A CCW specifies the command, the storage area to be used for the I/O operation, and the action to be followed when the operation is completed. When a program is running in a virtual partition, the CCWs are copied into the real address area.

Translation from the virtual I/O area addresses in the CCW to real addresses is accomplished either by the supervisor channel program translation routines.

The contents of CCWs should be inspected when the cause of a system malfunction leads you to suspect I/O operation errors. For example, parts of a program being overwritten and causing invalid instructions, or unexpected information in your program I/O data area will probably cause a program check and generate incorrect output from your program.

By examining the contents of the channel program the following information can be obtained:

- Validity of the operation code and of the sequence of CCWs used. If either of these is invalid, an informatory message is normally printed on SYSLOG to help you to determine the cause of the error.
  (Consult the component manual for the I/O device for the valid codes and sequence of use.)
- Data address in the last CCW used. Translated channel programs are destroyed in a system dump by the channel programs required for the DUMP and by channel programs started for other partitions. However, they can be located in a stand-alone dump, an example of which is shown in Appendix G. (Refer to chapter 12 in this section for methods of translating real addresses to virtual and vice-versa.)
- Count in the CCW. This must be a byte count of one or more for any I/O operation not involving magnetic tape units. (For a transfer in channel (TIC) command, the count may be zero.)
- When working with wrong length records or variable length records, the suppress length indicator should be set to 1 to prevent an error condition.

How to locate:

1. Bits 8-31 of the CSW (Channel Status Word) stored in location X'40' of low address storage contain the address of the next CCW to be executed. Subtract eight from this address to obtain the address of the last CCW used. (Refer to Section 2-E-2 for details of low address storage.)
   Caution: The data stored in low address storage may be overwritten by the dump program. If this is thought to be the case, use the method described below.
2. Bytes 9-11 of the CCB associated with the channel program contain the addresses of the first CCW in the channel program. Bytes 13-15 of the same CCB contain the address of the next CCW. Subtract eight from this address to obtain the address of the last CCW used.
   The figure below shows the format and contents of any CCW.

CCW

Command	Data Address	Flags	00	TP code	Count
0	8	32	38	40	48 — 63

FIELD	DESCRIPTION
Command Code	Bits 0-7: Specify the operation to be performed. Consult device component manual
Data Address	Bits 8-31: Specify the location of a byte in main storage. It is the first location referred to in the area designated by the CCW.
Flags	Bits 32-36: Specify the flag bits used in conjunction with the CCW. Bit 32– Chain-Data (CD) causes the address portion of the next CCW to be used with the current CCW. Bit 33– Chain-Command (CC) causes the command code and data address of the next CCW to be used. The chain data flag (bit 32) takes precedence over this flag. Bit 34– Suppress Length Indication (SLI) causes a possible incorrect length indication to be suppressed. The chain data flag (bit 32) takes precedence over this flag. Bit 35-- Skip (SKIP) suppresses the transfer of information to real storage. Bit 36– Program Control Interruption (PCI) causes the channel to generate an interrupt when the CCW is fetched. Bit 37– IDAL bit. Set to 1 if I/O area crosses page boundary, that is, if the I/O area is not confined to one page frame in real storage.
Reserved	Bits 38-39: (Must contain zeros)*
TP	Bits 40-47:
Count	Bits 48-63: Specify the number of bytes in the operation

*The transfer in channel command (TIC) is the one exception to this statement.

Figure 4.28. Explanation of contents of the CCW

SUPERVISOR CALLS

A problem program running in any partition fields control to the supervisor by issuing a supervisor call instruction. The SVC instruction contains a code that indicates its purpose. For example, SVC 0 requests the supervisor to execute the channel program. Some SVCs are optional and cause program cancellation if the supervisor does not support the option requested.

A complete list of DOS/VS SVC codes with the associated macro instructions that generate the SVC is shown in Figure 4.29 parts 1, 2 and 3.

A detailed description of the SVCs can be found in *DOS/VS Supervisor Logic.*

SVC Dec	SVC Hex	Macro supported	Function
0	0	EXCP	Execute Channel Program
1	1	FETCH	Fetch any phase
2	2		Fetch a logical transient (B-transient)
3	3		Force dequeue
4	4	LOAD	Load any phase
5	5	MVCOM	Modify supervisor communication region (if issued by MVCOM macro) Fetch anyother physical transient (if issued by a physical transient)
6	6	CANCEL	Cancel a problem program or task
7	7	WAIT	Wait for a CCB or TECB
8	8		Transfer control to the problem program from a logical transient (B-transient)
9	9	LBRET	Return to a logical transient (B-transient) from the problem program after an SVC 8
10*	A	SETIME	Set timer interval
11	B		Return from a logical transient (B-transient)
12	C		Reset switches in partition communication region.
13	D		Set switches in partition communication region.
14	E	EOJ	Cancel job and go to job control for end of job step
15	F	SYSIO	Headqueue and execute channel program
16*	10	STXIT(PC)	Provide supervisor with linkage to user's PC routine for program check interrupts
17*	11	EXIT(PC)	Return from user's PC routine
18*	12	STXIT(IT)	Provide supervisor with linkage to user's IT routine for interval timer interrupts
19*	13	EXIT(IT)	Return from user's IT routine
20*	14	STXIT(OC)	Provide supervisor with linkage to user's OC routine for external or attention interrupts (operator comm.)
21*	15	EXIT(OC)	Return from user's OC routine
22	16	SEIZE	Seize/release system; enable/disable for external and I/O interrupts; set key in users PSW
23*	17		Load phase header. Phase load address is stored at user's address
24*	18	SETIME	Set timer interval and provide supervisor with linkage to user's TECB, if any

* optional

Figure 4.29 Supervisor Calls (Part 1 of 3)

SVC Dec	SVC Hex	Macro Supported	Function
25*	19		Issue HALT I/O on a teleprocessing device, or HALT I/O on any device if issued by OLTEP. With multiprogramming dequeue an unstarted OLTEP I/O request to a shared device
26*	1A		Validate address limits
27*	1B		Special HIO on teleprocessing devices
28*	1C	EXIT(MR)	Return from user's stacker select routine (MICR type devices only)
29*	1D	WAITM	Provide support for multiple wait macro WAITM
30*	1E	QWAIT	Wait for a QTAM element
31*	1F	QPOST	Post a QTAM element
32	20		Reserved
33	21		Reserved for COMRG macro
34	22	GETIME	Provides Time-of-Day and updates the DATE field
35*	23	HOLD	Hold a track for use by the requesting task only
36*	24	FREE	Free a track held by the task issuing the FREE
37*	25	STXIT(AB)	Provide supervisor with linkage to user's AB routine for abnormal termination of a task
38*	26	ATTACH	Initialize a subtask and establish its priority
39*	27	DETACH	Perform normal termination of a subtask. It includes calling the FREE routine to free any tracks held by the subtask
40*	28	POST	Inform the system of the termination of an event and ready any waiting tasks
41*	29	DEQ	Inform the system that a previously enqueued resource is now available
42	2A	ENQ	Prevent tasks from simultaneous manipulation of a shared data area (resource)
43	2B		SDR SVC
44*	2C		Provide supervisor support for external creation of unit check records by specific request
45*	2D		Provide emulator interface
46*	2E		Provide OLTEP with the facility to operate in supervisory state
47*	2F	WAITF	Provide support for multiple wait macro for MICR type devices
48*	30		Fetch a CRT transient
49	31		Reserved

* optional

Figure 4.29 Supervisor Calls (Part 2 of 3)

SVC Dec	SVC Hex	Macro supported	Function
50	32		Reserved for LIOCS error recovery
51	33		Return phase header
52*	34	TTIMER	Return the remaining time interval, or cancel a time interval
53	35		Reserved
54	36	FREEREAL	
55	37	GETREAL	Provide interf. between SDAID and PDAID initialization routine and page management routine, to create the PDAID alternate area or the SD area
56*	38	GETPUB-FREEPUB	Occupy or free PUB of the device used by POWER
57*	39		Make POWER-supported partition dispatchable
58	3A		Provide interface between job control and the supervisor. Get real storage for real jobs
59	3B		Provide interface between EOJ and the supervisor. Reset the storage key for virtual jobs
60	3C	GETADR	Provide virtual address of location within I/O areas for ERP and CRT routines
61*	3D	GETVIS	Get storage in virtual partition
62*	3E	FREEVIS	Free storage in virtual partition
63	3F	USE	Use a resource
64	40	RELEASE	Release a resource
65*	41	CDLOAD	Load VSAM or core image phase
66	42	RUNMODE	Return mode in which program is running
67*	43	PFIX	Fix page(s) in real storage
68*	44	PFREE	Free page(s) in real storage
69*	45	REALAD	Return real address corresponding to a given virtual address
70*	46	VIRTAD	Return virtual address corresponding to a given real address
71*	47	SETPFA	Establish or terminate the linkage between the supervisor and a user page-fault appendage routine
72*	48	GETCBUF-FREECBUF	Get or free copy buffer for IDAL or tape ERP
73*	49	SETAPP	Allow linkage to channel and appendage routines
74*	4A		Fix page(s) in real storage for restart
75	4B		Reserved
76	4C		Initiate recording of a RMS I/O error
77	4D	TRANSCSW	Returns the virtual address of a copied CCW
78 thru 84			Reserved
85	55	RELPAG	Release contents of one or more pages
86	56	FCEPGOUT	Force a page-out for one or more pages
87	57	PAGEIN	Page in one or more pages

* optional

Figure 4.29 Supervisor Calls (Part 3 of 3)

PIB AND PIB2

The PIB (program Information Block)

Real storage area is reserved in the supervisor for this information block by the MPS multiprogramming and/or NPARTS and AP (Asynchronous Processing) parameters of the SUPVR supervisor generation macro. Each entry in this block is 16 bytes and contains status information about the program and, if AP is supported about the subtasks running in each partition supported by the supervisor.

The first entry is reserved for the attention routine, this entry is called the Attention PIB (AR PIB).

Other entries in the PIB belong to the problem programs and subtasks. The sum of all subtasks and problem program entries may not exceed 15. The maximum number of entries, including the attention PIB and AP (subtask) PIBs, is 16.

For a supervisor that is not generated to support more than one partition there is only one 16-byte entry, which is shared by the attention routine and the problem program.

By examining the data recorded in the appropriate PIB entry, the status and location of programs running in any partition can be established. Some of the more important data to be looked at in the PIB during the first analysis of a dump output are:

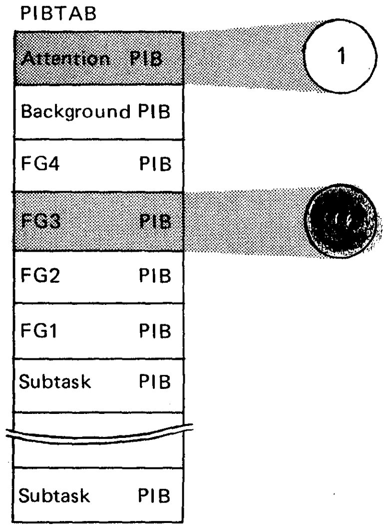
- Byte 0, from which you can determine whether the program is waiting for
  - The LTA (Logical Transient Area), X'81'
  - The PTA (Physical Transient Area), X'85'
  - An I/O interrupt, X'82'
  - A page to be paged in, X'87'
  - A page to be paged in with QTAM active, X'8F'
- Byte 4, X'80', which indicates that the job or task is running in virtual mode
- The address of the program save area
- The address of the system save area.

Figure 4.30 (opposite) shows the format and describes the contents of an entry in the PIB.

How to locate:

Bytes X'5A' – X'5B' of the partition comregs contain the address of the first entry in this information block. Label PIBTAB in the supervisor listing identifies the address of the first byte of this information block.

Appendix G shows an example of locating the PIB in a dump output.



1

Format of Attention PIB

0	1	2 3	4	5 6 7	8	9 10 11	12	13	14	15	
Flag Byte (See A)	Cancel Code	SYSLOG ID (AR)	always zero	Inactive = zero Active = Address of LTA save area (Note 2)	Switch Byte (See F)	Address of save area of zero (Note 1) (Note 2)	X'07' PIB assign flag (See D)	BG user LUB index	Number of BG program LUB's	Not used	

Note 1: *a) When LTA is inactive= LTA save area address.*
b) When LTA is active for Problem Programs, this address is exchanged with that in the Problem Program PIB.

Note: *When LTA is active for Logical Attention, bytes 9-11 are zero and bytes 5-7 contain the LTA save area address.*

Legend: A, B, C, D, E, F refer to next part of this figure, which describes the meaning of each bit in a PIB entry

Format of any Probl. Program or Subtask PIB

0	1	2 3	4	5 6 7	8	9 10 11	12	13	14	15	
Flag Byte (See A)	Cancel Code	SYSLOG ID	DAT flag (See B)	Address of Problem Program save area or LTA save area (Note 3)	Gate ID (See C)	Address of system save area	PIB assign flag (See D)	User LUB index	Number of Program LUB's	Flag Byte (See E)	

Note 3: *When the Logical Transient Area is active the save area address in the Problem Program PIB is exchanged with that in the Attention PIB.*

The number of Problem Program PIBs generated depends on the number of partitions specified during system generation.
Subtask PIBs are generated only if AP= YES has been specified during system generation.
The number of subtask PIBs generated depends on the number of partitions, that is: →

No. of partitions	No. of subtasks
2	13
3	12
4	11
5	10

Figure 4.30 Explanation of the contents of an entry in the PIB, part 1 of 3.

PIB AND PIB2

A Flag Byte (First byte in PIB)

The following flags are always used:

X'71' = Program is waiting for SVC58
X'73' = Program is waiting because system is seized
X'75' = Program is waiting for copy block
X'77' = Program is waiting for TFREE
X'79' = Program is waiting for channel queue entry
X'7B' = Program is waiting for CCW translation
X'7D' = Program is waiting for a free Console Buffer
X'80' = Program is not active
X'81' = Program is SVC2-bound (waiting for the LTA to be released)
X'82' = Program is SVC7-bound (waiting for an I/O interruption)
X'83' = Program is ready to run
X'85' = Program is SVC5-bound (waiting for the PTA to be released)
X'86' = Initial selection of RAS (used only for RAS PIB flag)
X'87' = Program is set to common bound condition

The following flags are used only if NPARTS = 1. X'61' through X'69' are used by the load leveller to deactivate a partition. The partition to which a flag refers depends on NPARTS as follows:

	NPARTS –			
	2	3	4	5
X'61' refers to	BG	BG	BG	BG
X'63' refers to	F1	F2	F3	F4
X'65' refers to	–	F1	F2	F3
X'67' refers to	–	–	F1	F2
X'69' refers to	–	–	–	F1

X'6B' = Program is SVC35-bound
X'6D' = Program is waiting for next freed page frame
X'6F' = Program is IDRA-bound

The following flags are only used if AP= YES:

X'51' = Program is SVC38-bound
X'53' = Program is SVC41/42-bound

The following codes are only used if AP= YES and PFIX= YES.
The codes are used by the PFIX routines to set a partition PFIX bound.
The partition to which a flag refers depends on NPARTS as follows:

	NPARTS =			
	2	3	4	5
X'47' refers to	BG	BG	BG	BG
X'49' refers to	F1	F2	F3	F4
X'4B' refers to	–	F1	F2	F3
X'4D' refers to	–	–	F1	F2
X'4F' refers to	–	–	–	F1

The following codes are used only if AP= YES and VSAM= YES.
The codes are used by the VSAM routines to set a partition PFIX bound.
The partition to which a flag refers depends on NPARTS as follows:

	NPARTS =			
	2	3	4	5
X'3D' refers to	BG	BG	BG	BG
X'3F' refers to	F1	F2	F3	F4
X'41' refers to	–	F1	F2	F3
X'43' refers to	–	–	F1	F2
X'45' refers to	–	–	–	F1

Figure 4.30 Explanation of the contents of an entry in the PIB, part 2 of 3.

B PIB DAT Flag

X'01' = Return to re-entrant supervisor routine
X'02' = Return to gated supervisor routine
X'04' = Move CCB at dispatching time
X'08' = Service delayed external interrupt
X'10' = Task is temporarily deactivated
X'20' = Reserved
X'40' = Task has seized the system
X'80' = Program is running in virtual mode

C Gate Identifier

X'71' = Gating of SVC58 required
X'53' = Gating of SVC41/42 required

The flags are only used if the PIB DAT Flag is X'03', that is, the first two flags are on (See B).

D PIB Assign Flag

X'80' = SYSRES DASD file protect inhibited (allow write operation on SYSRES)
X'40' = Channel appendage exit allowed (BTAM)
X'20' = Cancel in progress (used in terminator function)
X'10' = Cancel control (set on a foreground cancel)
X'08' = Hold foreground assignments
X'07' = Attention PIB

E Problem Program PIB Flag (Last byte in PIB)

Bit 0: 1= Batched job in foreground (always on when tested)
Bit 1: 1= Cancel in LTA and device not assigned
Bit 2: 1= /& on SYSIN if DASD
Bit 3: 1= Partition in stopped state
Bit 4: 1= Fetch EOJ monitor
Bit 5: 1= Task is canceled
Bit 6: 1= Subtask(s) attached
Bit 7: 1= in AB routine

F Attention PIB Switch Byte

Bit 0: Reserved
Bit 1: Reserved
Bit 2: 1= Delay cancelation
Bit 3: 1= Emergency cancel request
Bit 4: 1= Detach Logical Attention Routine ($$BATTNA)
Bit 5: Reserved
Bit 6: 1= Fetch Logical Attention Routine ($$BATTNA)
Bit 7: 1= External Interrupt request

Figure 4.30. Explanation of the contents of an entry in the PIB, part 3 of 3

PIB AND PIB2

PIB2 (Program Information Block Extension)

As the name of this block implies, it is an extension of the PIB and is of identical size, being generated with the PIB during system generation. Data recorded in each 16-bytes entry supplements the data recorded in the PIB.

By examining the contents of bytes 0 and 1 of the appropriate problem program PIB2 entry, the address of the associated partition communication region can be established.

How to locate:

Bytes X'7C' – X'7D' of the partition comregs contain the address of the first entry in this information block. Label PIB2TAB in the supervisor listing identifies the address of the first byte of this information block.

Appendix G shows an example of locating the PIB2 in a dump.

PIB2TAB
Attention PIB
Background PIB
FG4 PIB
FG3 PIB
FG2 PIB
FG1 PIB
Subtask PIB
Subtask PIB

Format of any PIB extension entry

0 1	2 3	4 5 6 7	8 9 10 11	12 13	14	15
16-bit Address of communication region of partition (Note 1)	System LUB index	Interrupt Information (See table A below)	Address of termination ECB if any, otherwise zeros	Program Interrupt Key (PIK)	unused	Flag Byte (See B below)

Note 1: *Always BG communication region in Attention—and Background PIB extension. Appropriate communication region in other PIB extensions when a multiprogram system has been generated. To place this address in a register, the instruction ICM should be used.*
For each PIB Table entry, an entry exists in the PIB Table Extension.

A

Type of interruption	Contents of PIB Extension Bytes			
	4	5	6	7
SVC	00	ILC*	Interruption Code	
PC	00	ILC*	Interruption Code	
I/O	00	00	I/O Address	

* ILC (Instruction Length Code) is in bits 5 and 6; other bits are zeros.

B

Byte 14	Byte 15
Not used	Bit 0: 1= POWER runs in this partition Bit 1: 1= This partition is supported by POWER, if POWER is active 2: 1= Task owns CRT 3-7: Not used

Figure 4.31. Explanation of the contents of an entry in the PIB2

CANCEL CODES

Byte 1 of the PIB contains a cancel code that is stored by the supervisor in the event of program cancellation. Normally a message is printed on SYSLOG and/or SYSLST that informs the operator about the reason for the cancellation, for example:

BG 0S04I ILLEGAL SVC – HEX LOCATION 007884 – SVC CODE 14

The cancel code (stored in byte 1 of the associated partition PIB) should be examined also in the event of a system malfunction such as a LOOP or WAIT STATE that prevents the system from issuing an error message.

Figure 4.32 (below) shows a list of all the cancel codes and their message prefixes.

All these cancel codes cancel the program, task, or subtask when they occur. If multitasking is being used and a main task is canceled, all of the subtasks attached are detached and canceled as a result of the main task being canceled. If a dump option was specified at system generation time or by job control, the contents of the supervisor and the partition in which the cancel condition occurred is printed on SYSLST.

Cancel Code (hex)	Message Code	Descriptive part of Message or Condition
10	- - - - -	Normal EOJ
11	0V07I	No channel program translation for unsupported device
12	0V06I	Insufficient buffer space for channel program translation
13	0V05I	CCW with count greater than 32 K
14	0V04I	Page pool too small
15	0V02I	Page fault in disabled program
16	0V01I	Page fault in MICR stacker select or PHO routine
17	0S02I	Program request (Same as 23 but causes dump because subtasks were attached when maintask issued CANCEL macro)
18	- - - - -	Eliminates cancel message when maintask issues DUMP macro with subtasks attached
19	0P74I	I/O operator option
1A	0P73I	I/O error
1B	0P82I	Channel failure
1C	0S14I	CANCEL ALL macro
1D	0S12I	Main task termination
1E	0S13I	Unknown ENQ requestor

Figure 4.32. DOS/VS Cancel Codes and Messages, part 1 of 2.

Cancel Code (hex)	Message Code	Descriptive part of Message or Condition
1F	0P81I	CPU failure
20	0S03I or 0S11I	Program check
21	0S04I or 0S09I	Illegal SVC
22	0S05I or 0S06I	Phase not found
23	0S02I	Program request
24	0S01I	Operator intervention
25	0P77I	Invalid address
26*	0P71I	SYSxxx not assigned (unassigned LUB code)
27	0P70I	Undefined logical unit
28	-----	QTAM cancel in progress
29	0S15I	No relocating loader support (Fetch or load request for relocatable phase while supervisor does not support relocating load)
2A	0P84I	I/O error during fetch (irrecoverable I/O error during fetch
2B	0V10I	I/O error on page data set
2C	0V09I	Illegal parameter passed by PHO routine
2D	0P88I	Program cannot be executed/restarted due to failing storage block
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	0P75I	I/O error queue overflow (error queue over-flow)
32	0P76I	Invalid DASD address
33	0P79I	No long seek (disk)
34		Reserved
35	0P85I	Job control open failure
36	0V08I	Page fault in I/O appendage routine
37		Reserved
38	0V11I	Wrong privately translated CCW
39		Reserved
FF	0P78I	Unrecognized cancel code
	0P83A**	Supervisor catalog failure
	0P87A**	IPL failure

Figure 4.32. DOS/VS Cancel Codes and Messages, part 2 of 2.

* If the CCB is not available, the logical unit is SYSxxx.

** The cancel code is not significant in case of a supervisor catalog or IPL failure, because the system is placed in the wait state without any further processing by the Terminator.

Note: In addition to recognizing the cancel codes above, the Terminator also recognizes the same codes with the X'80' bit on (cancel occurred in LTA). The X'80' bit is tested by $$BEOJ and subsequently reset.

GENERAL PURPOSE REGISTER USAGE

The following paragraphs describe the general usage of registers 0, 1, 13, 14, and 15 by IOCS, but the description is not meant to be all-inclusive.

Registers 0 and 1: Logical IOCS macros, the supervisor macros, and other IBM-supplied macros use these registers to pass parameters. Therefore, these registers may be used without restriction only for immediate computations, where the contents of the register are no longer needed after the computation. If you use them, however, you must either save their contents yourself (and reload them later) or finish with them before IOCS uses these registers.

Register 13: Control program subroutines, including logical IOCS, use this register as a pointer to a 72-byte doubleword-aligned save area. When using the CALL, SAVE, or RETURN macros, you can set the address of the save area at the beginning of each program phase, and leave it unchanged thereafter. However, when sharing a reenterable (read only) logic module among tasks, each time that module is entered by another task, register 13 must contain the address of another 72-byte save area to be used by that logic module.

Registers 14 and 15: Logical IOCS uses these two registers for linkage. Register 14 contains the return address (to the program) from DTF routines, called programs, and your subroutines. Register 15 contains the entry point into these routines, and is used as a base register by the OPEN (R), CLOSE (R), and certain DTF macros. IOCS does not save the contents of these registers before using them. If you use these registers you either save their contents yourself (and reload them later) or finish with them before IOCS uses them.

Registers for Your Use

Registers 2-12 are available for general usage. There are, however, a few restrictions.

The assembler instruction for translate and test (TRT) makes special use of register 2. It is your responsibility to save the contents of this register before executing the TRT instruction if register 2 contains valuable information (such as pointers or counters) for later use in your program. After the TRT instruction has been executed, you can then restore the contents of register 2 from the save area.

If an ISMOD logic module precedes a USING statement or follows your program, the use of registers 2-12 remains unrestricted even at assembly time. However, if the ISMOD logic module lies within the problem program, you should issue the same USING statement (which was issued before the logic module) directly following the logic module. This action is necessary because the ISMOD logic module uses registers 1, 2, and 3 as base registers, and the ISMOD CORDATA logic module uses registers 1, 2, 3, and 5 as base registers. Each time either module is assembled, these registers are dropped.

Register usage by JOB ACCOUNTING

(The Job Accounting option is discussed in Chapter 13 of this Section)

The system passes registers 11-15 to the user's I/O routine ($JOBACCT). These registers contain the following information:

- Register 11: Length of the job accounting table. Each table may vary in length according to the number of SIO counts specified at system generation time.
- Register 12: Base register for $JOBACCT (this eliminates the need for the user to load the base register)
- Register 13: Address of the user save area
- Register 14: Link register ($JOBACCT must exit via BR 14 to return to job control)
- Register 15: Address of the partition's job accounting table.

Because some of the job step information is cleared in the step-to-step transition, job control calls $JOBACCT at the end of each step. If $JOBACCT does not save or accumulate this information, it is lost.

Register usage by POWER

Register 2: linkage register for POWER subroutines; also used as a base for the Data Unit Table.
Register 4: address of the Device Control Table.
Register 6: base register for readers, writers, and the operator communications task command processor phases.
Register 7: Address of the TIB of the task currently controlling the CPU
Register 8, 9, 10 and 13: Base registers for the resident POWER code
Register 14: I/O bugger address
Register 15: Pointer to disk address for QFILE and Data File I/O

Register usage by POWER RJE

Register 1: Pointer to RJE task DECB (DSKIO or GETBUF etc)
Register 2: Subroutine linkage
Register 4: Base register for BTAM Interface Module
Register 6: Base register for RJE overlay phase
Register 7: RJE task TIB address
Register 8:
Register 9:
Register 10: FGPSPOOL CSECT base registers
Register 13:
Register 11: Subroutine linkage
Register 14: I/O Buffer address
Register 15: Pointer to disk address for QFILE and DATAFILE I/O or linkage register for BTAM
Register 5:
Register 12: Work registers

GENERAL PURPOSE REGISTER USAGE

Linkage Registers

To standardize branching and linking, registers are assigned specific roles (Figure 4.33). Registers 0, 1, 13, 14, and 15, are known as the linkage registers. Before a branch to another routine, the calling program is responsible for the following calling sequence:

1. Loading register 13 with the address of a register save area in the program that the called program is to use
2. Loading register 14 with the address to which the called program will return control
3. Loading register 15 with the address from which the called program will take control
4. Loading registers 0 and 1 with parameters, or loading register 1 with the address of a parameter list. (Although permissible, it is not normal to load register 0 with parameters).

Register Number	Register Name	Contents
0	Parameter register	Parameters to be passed to the called program.
1	Parameter register or Parameter list register	Parameters to be passed to the called program. Address of a parameter list to be passed to either the control program or a user's subprogram.
13	Save area register	Address of the register save area to be used by the called program.
14	Return register	Address of the location in the calling program to which control should be returned after execution of the called program.
15	Entry point register	Address of the entry point in the called program.

Figure 4.33 Linkage Registers

After execution of the calling sequence, the following should occur as a result of called program execution:

1. The contents of registers 2 through 14, and the program mask are unchanged.
2. The contents of registers 0, 1 and 15, the contents of the floating point registers, and the condition mode may be changed.
3. The parameter list addresses contain the results obtained by the execution of the called program.

TABLES REQUIRED BY USER EXIT ROUTINES

When support is provided during system generation for user exit routines (other than VSAM exit routines), an area is reserved in the supervisor for one or more of the following tables:

- Interval timer (IT)
- Abnormal termination (AB)
- Page fault handling overlap (PHO)
- Program check (PC)
- Operator communication (OC).

Entries in the table are generated from the STXIT macro issued by the problem program, and the number of entries depends on the number of partitions for which the system has been generated.

The number of entries for the PC and AB tables is increased by the number of subtasks allowed on a system generated for use with multitasking.

TABLES REQUIRED BY USER EXIT ROUTINES

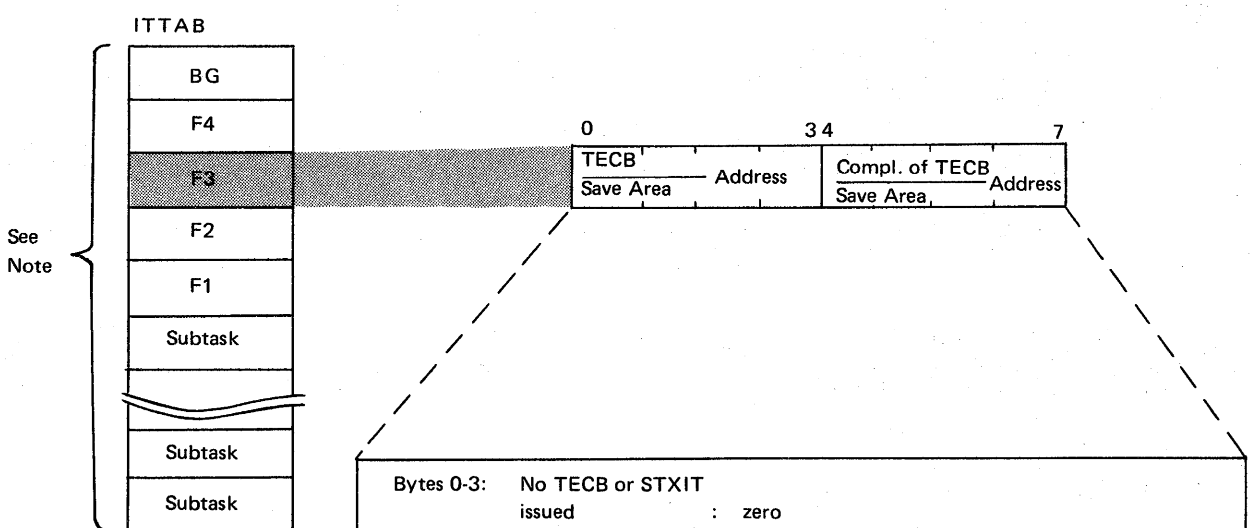
Interval Timer Support (IT)

This parameter generates programming support for the hardware interval timer feature, which is used to time-stamp the system. It enables a problem program to set a time interval (via the SETIME macro).

By using the STXIT, EXIT, and TECB macros, a specific routine within the problem program or task is entered when this time interval elapses.

How to locate the IT option table

Bytes X‘66’–X‘67’ of the partition communication regions contain the address of the IT Option Table. Label ITTAB identifies the first byte of the table.



Bytes 0-3:	No TECB or STXIT issued	:	zero
	TECB issued	:	Address of the timer event control block
	STXIT issued	:	Address of user interval timer routine
	STXIT issued and user routine is already in use	:	Complement of the user interval timer routine
Bytes 4-7:	No TECB or STXIT issued	:	zero
	TECB issued	:	Complement of the TECB address
	STXIT issued	:	Address of the user save area

Note: One table entry is built for each partition and an IT Request table is also built.

With multiple timer and asynchronous processing supported, the table always comprises 15 entries; the subtask entries occupy the higher address locations in the table.

Figure 4.34. Explanation of the contents of the IT option table.

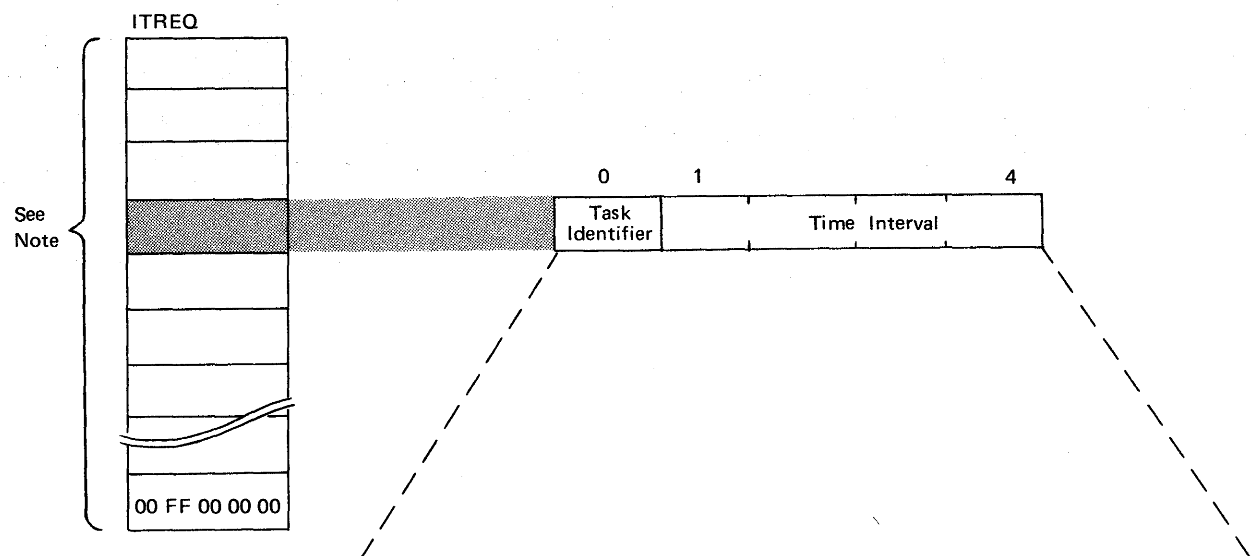
Interval Timer Request Table

This table is generated only for systems supporting the interval timer option (IT= YES). It is used in conjunction with the IT option table described in Figure 4.34.

The number of entries is one more than the number of partitions supported, but with multiple timer and asynchronous processing supported, the table always comprises 16 entries.

How to locate the IT request table

Bytes X'50'–X'53' of the System Communication region (SYSCOM) contain the address of the IT Request Table. Label ITREQ identifies the first byte of the table.



Byte 0 :	No SETIME issued	:	X'00', the last entry is always X'00'.
	SETIME issued	:	PIK or TIK (X'10–X'F0') of the program or task that issued the SETIME macro.
Bytes 1-4:	No SETIME issued	:	Highest time interval possible (X'FF000000'). The last entry is always X'FF000000'.
	SETIME issued	:	Time interval that will elapse between the occurrence of the currently pending interrupt in SYSTIMER and the occurrence of the interrupt itself. The entries are in ascending order of magnitude, that is, the smallest interval appears first. The interval corresponding to the currently pending interrupt is in SYSTIMER (address X'50') and the key of the respective program or task is in TWTIMS.

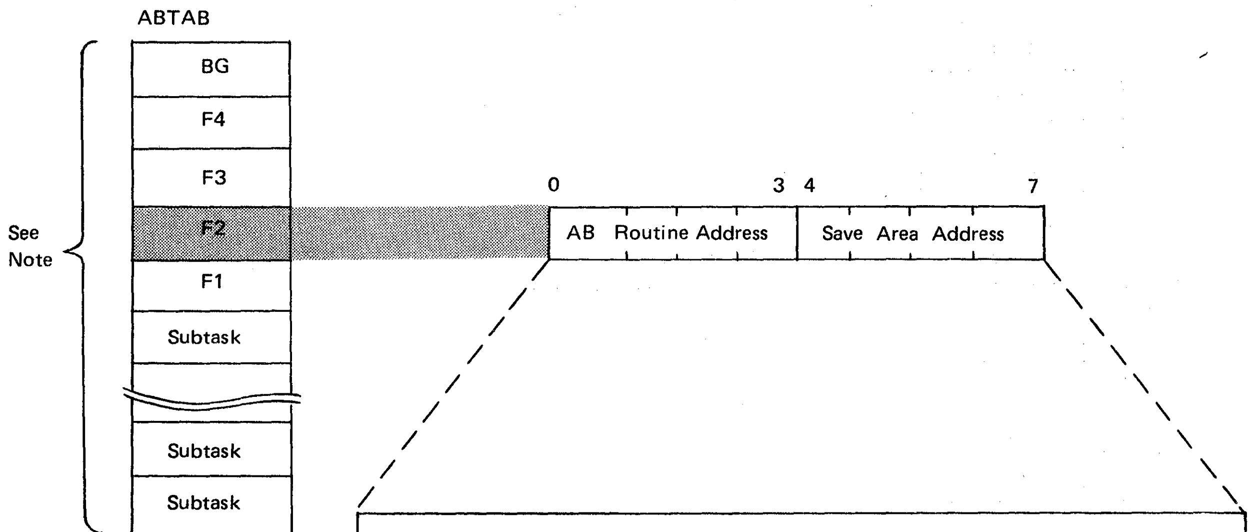
Figure 4.35. Explanation of the contents of the IT option request table.

Abnormal termination support (AB)

Abnormal termination exits are available for main tasks and/or subtasks, allowing you to gain control before an abnormal condition removes the task from the system. For example, in the abnormal termination routine, you can close your files. This function is provided by the AB operand of the STXIT macro. See *Supervisor and I/O Macros* for detailed information on the format and use of the STXIT macro.

How to locate:

Bytes X'54'–X'57' of the System Communication region (SYSCOM) contain the address of the AB Option Table. Label ABTAB identifies the first byte of the table.



Bytes 0-3:	No STXIT issued	:	Zero.
	STXIT issued and rtnaddr parameter passed	:	Address of entry point of user's abnormal termination routine. If AP (Asynchronous Processing) is supported the maintask and subtasks may have the same or different AB routines. When a subtask is ATTACHed after a STXIT AB macro has been issued by the maintask, the subtask will receive the AB routine address 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.
Bytes 4-7:	No STXIT issued or no save area parameter passed	:	Zero.
	STXIT issued and save area parameter passed	:	Address of a 72-byte save area used by the supervisor to store the interrupt status information and the contents of the general registers. (See Chapter 12.)

Note: One table entry is generated for each partition supported. With asynchronous processing support, the table always comprises 15 entries; the subtask entries occupy the higher address locations in the table.

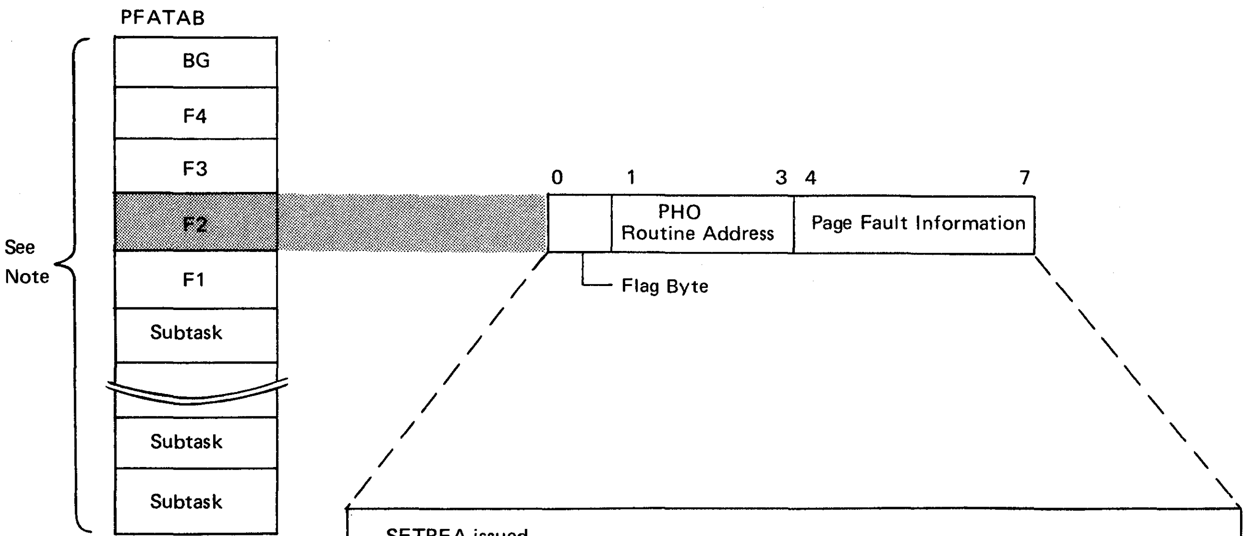
Figure 4.36. Explanation of the contents of the AB option table.

Page Fault Handling Overlap Support (PHO)

This option enables a user routine to continue processing during the time a page fault, occurring in the same task, is being handled, PHO=YES in the SUPVR supervisor generation macro reserves an area in the supervisor for the PHO option table. Entries are made in this table when the user program issues a SETPFA macro instruction. The SETPFA macro instruction is described in *DOS/VS Supervisor and I/O macros.* If asynchronous processing (AP) is not supported, one entry is generated in the table for each partition supported by the system (NPARTS). If AP is supported, 15 entries are generated.

How to locate:

Label PFATAB in the supervisor listing identifies the first byte of this table.



<u>SETPFA issued</u>

Byte 0 :	Bit 0 :	1= A page fault from the task is in the Page Queue Table.
	Bit 1 :	1= A page fault occuring in a supervisor service of the task is queued in the Page Queue Table.
	Bits 2-7:	Unused.
Bytes 1-3:	Address of the user's page fault appendage routine.	
Bytes 4-7:	Page fault information (for detailed layout see Page Queue Table).	
	This information is stored here only when a page fault occurs in a supervisor service working for the task, while another page fault is queued for the task in the Page Queue Table; otherwise, zeros.	

<u>NO SETPFA issued</u>

Bytes 0-7: Zeros.

Note: One table entry is generated for each partition supported. With asynchronous processing support, the table always comprises 15 entries; the subtask entries occupy the higher address locations in the table.

PFATAB is only built if PHO=YES was specified in the SUPVR macro at supervisor generation.

Figure 4.37. Explanation of the contents of an entry in the PHO option table.

TABLES REQUIRED BY USER EXIT ROUTINES

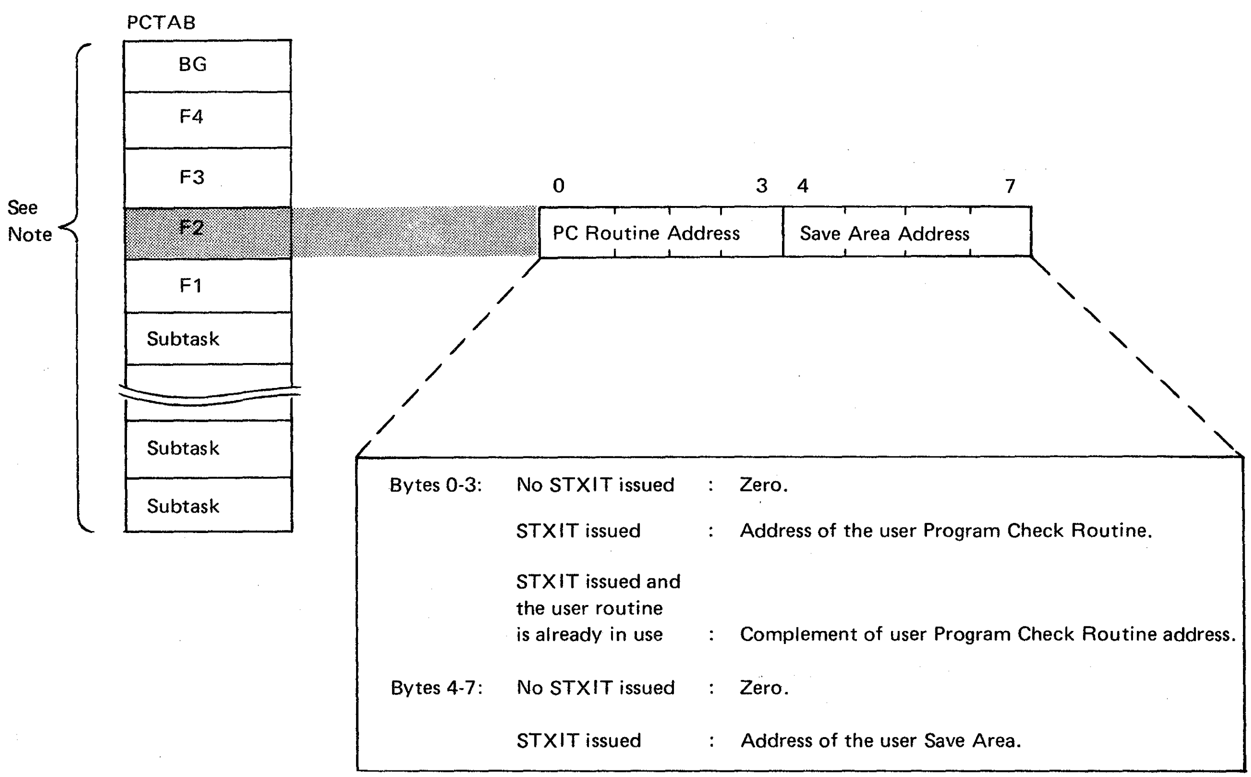
Program check support (PC)

Program check (PC) support generates a PC table within the supervisor (see the Figure below). The address of a user program check routine is placed in the table via the STXIT macro issued by the problem program. If the STXIT PC linkage is established and a program check occurs within this program, the supervisor gives control to the user's routine instead of canceling the job being run in this partition. The support is extremely advantageous when using LIOCS. (For example, files can be closed before job termination.) If a program check occurs in a routine being executed from the logical transient area (LTA), only the task associated with that routine is abnormally terminated.

In a multitasking environment each subtask and main task may have its own PC routine. A PC routine can be shared by more than one task within a partition. This can be done issuing a STXIT macro in each task with the same routine address but with separate save areas. To successfully share the same PC routine, it must be reenterable (capable of being used concurrently by two or more tasks).

How to locate:

Bytes X'64'–X'65' of the partition communication regions contain the address of the PC Option Table. Label PCTAB identifies the first byte of the table.



Note: In a supervisor without multiprogramming support, there is only one entry (BG) in each generated table. With multiprogramming support, there is one entry for each partition supported.
With asynchronous processing support, each generated table always comprises 15 entries; the subtask entries occupy the higher address locations in the table.

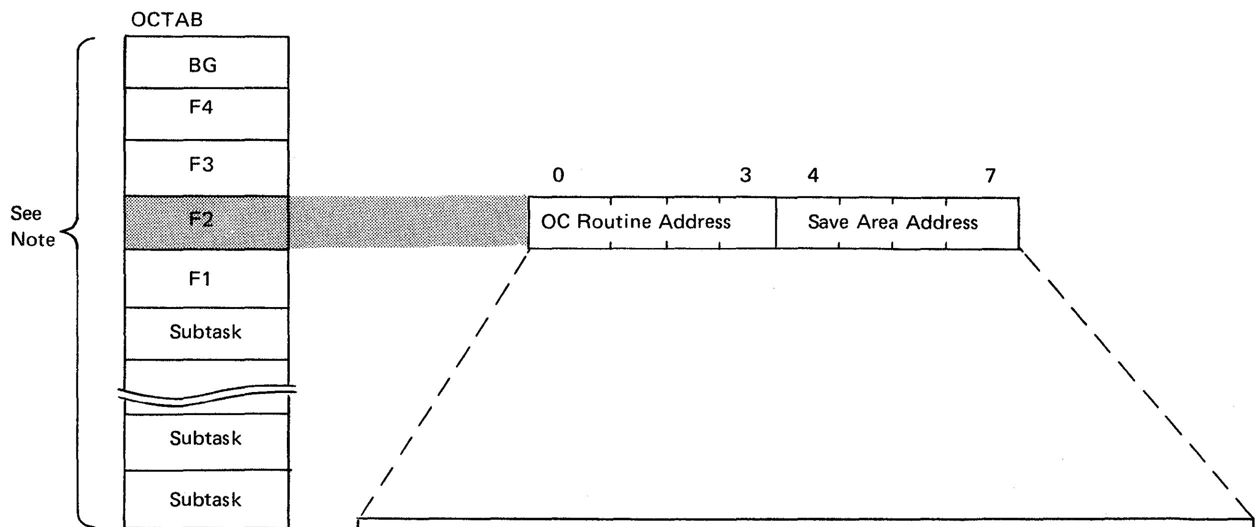
Figure 4.38. Explanation of the contents of an entry in the PC option table.

Operator communications support (OC)

Operator Communications (OC) refers to the processing of an external interrupt by a problem program. In a multitasking environment, only the main task can communicate via the OC linkage. By specifying OC=YES, a table (OC option table) is generated within the supervisor (see Figure below). When the problem program issues the STXIT macro, the address of its external interrupt routine is moved to the OC option table. The user's routine is terminated by issuing the EXIT macro. When OC=YES is specified, support is available to all partitions.

How to locate:

Bytes X'68'–X'69' of the partition communication regions contain the address of the OC Option Table. Label OCTAB identifies the first byte of the table.



Bytes 0-3:	No STXIT issued	:	Zero.
	STXIT issued	:	Address of the user Operator Communication Routine.
	STXIT issued and the user routine is already in use	:	Complement of the user Operator Communication Routine
Bytes 4-7:	No STXIT issued	:	Zero.
	STXIT issued	:	Address of the user Save Area.

Note: In a supervisor without multiprogramming support, there is only one entry (BG) in each generated table.
With multiprogramming support, there is one entry for each partition supported.
With asynchronous processing support, each generated table always comprises 15 entries; the subtask entries occupy the higher address locations in the table.

Figure 4.39. Explanation of the contents of an entry in the OC option table.

SAVE AREAS

Partition Save Areas and Label Save Areas

Each partition contains a save area for program name, old program status word, and registers.

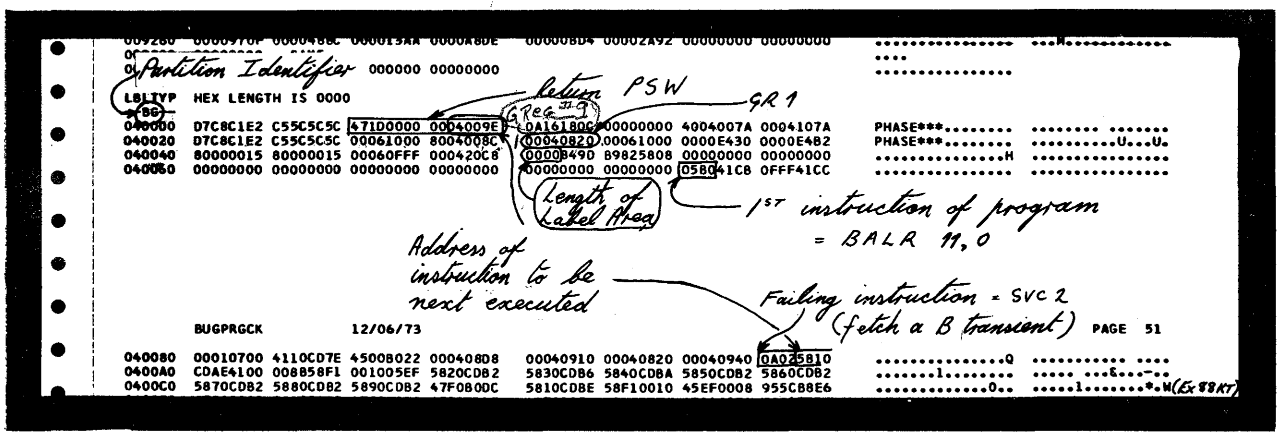
Following the partition save area, each partition contains a label area for label processing if the LBLTYP statement is used. Both areas are at the low end of the partition.

Save area length = 88 (dec) bytes or, if the floating point feature is supported, 120 (dec) bytes (FP=YES specified in the CONFG system generation macro).

Label area length is determined by the system according to the LBLTYP card specification:

- TAPE (standard tape labels) = 80 bytes
- NSD (nn) (nonsequential disk) = 84 bytes + 20 bytes per extent statement
- Omitted = 0.

Figure 4.41 (opposite) illustrates the location of the partition and label save areas in virtual storage for a system supporting multiprogramming. The figure below shows an example of a background program partition, save area as it is printed in a system dump. The programmers remarks on the figure indicate how the programmer used the save area during offline debugging.



```
Partition Identifier   000000 00000000

LBLTYP   HEX LENGTH IS 0000
BG
040000   D7C8C1E2 C55C5C5C 471D0000 0004009E   0A16180C 00000000 4004007A 0004107A   PHASE***........
040020   D7C8C1E2 C55C5C5C 00061000 8004008C   00040820 00061000 0000E430 0000E4B2   PHASE***........
040040   80000015 80000015 00060FFF 000420C8   0000B49D B9825808 00000000 00000000
040060   00000000 00000000 00000000 00000000   00000000 00000000 05B041CB 0FFF41CC

BUGPRGCK          12/06/73                                                           PAGE  51

040080   00010700 4110CD7E 45008022 000408D8   00040910 00040820 00040940 0A025810
0400A0   CDAE4100 008858F1 001005EF 5820CDB2   5830CDB6 5840CDBA 5850CDB2 5860CDB2
0400C0   5870CDB2 5880CDB2 5890CDB2 47F080DC   5810CDBE 58F10010 45EF0008 955CB8E6
```

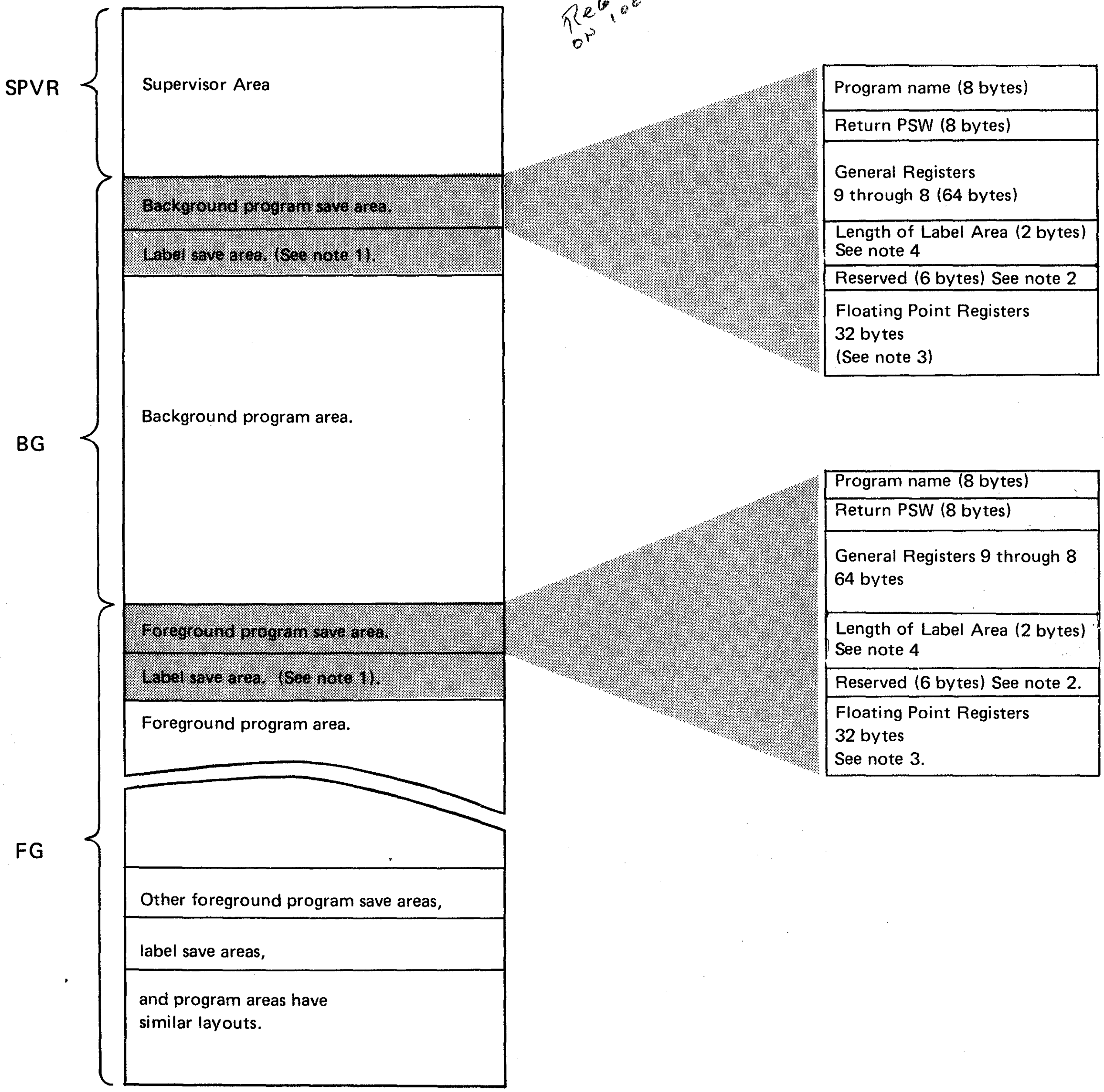

Figure 4.40. An example of a system dump output showing the partition save area. The programmer's remarks on the dump show areas of immediate interest during offline debugging.

How to locate:

The addresses of the partition save areas are stored in the problem program PIB, described in Chapter 7 in this Section.

DOS SUPVR PLACES INFO IN THE SAVE AREAS WHEN YOU PARTITION BLOWS, STARTS WITH SVC PAST THE INSTR CAUSING THE PROB.

REG 1 GOOD PLACE TO START ON 1OCS PROBLEMS



Note 1: *Length of the label area depends on the amount of storage specified by LBLTYP statement:*

- *A.* *For standard tape labels (any number)-80 Bytes*
- *B.* *For sequential DASD and DTFPH MOUNTED SINGLE-0 Bytes*
- *C.* *For DTFIS, DTFDA and DTFPH MOUNTED ALL-84 Bytes plus 20 Bytes per extent.*

Note 2: *Job start time, for time stamp, is stored in last 4 bytes of this area.*

Note 3: *Floating point register save area is required only when floating point feature is specified at supervisor generation.*

Note 4: *Only non-zero if a // LBLTYP statement read before the current job step. Otherwise reserved by the linkage editor, but not entered in these bytes.*

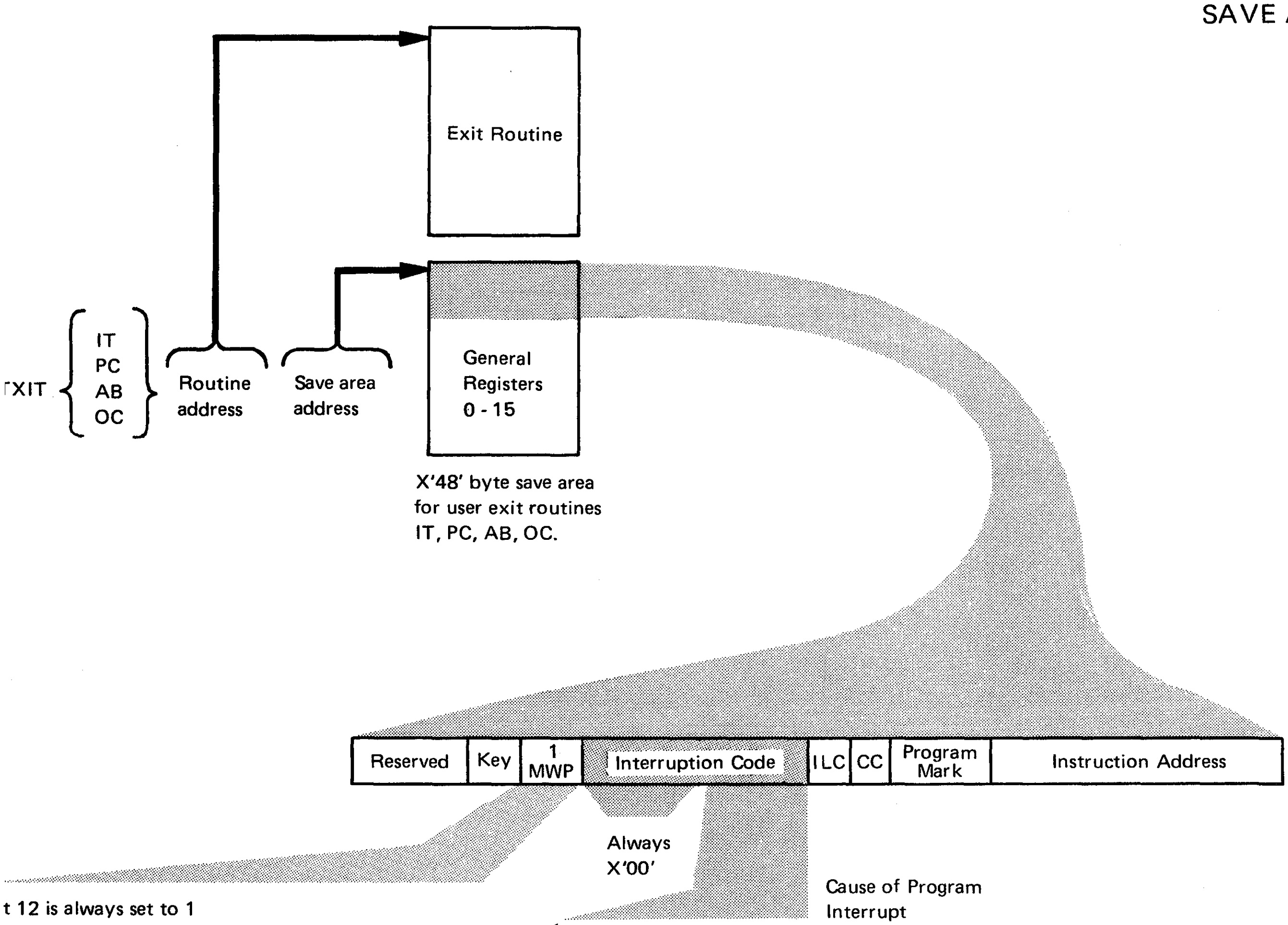
Figure 4.41. Organization of partition save and label save areas.

SAVE AREAS

ABSAVE area

In all abnormal termination conditions where an exit is taken to an abnormal termination routine specified and written by the user, the register values are stored in the ABSAVE save area before the appropriate error code is stored in the low-order byte of register 0. To have this value available when looking at a storage dump you should store (STC or ST) register 0 in another save area upon entry into the abnormal termination routine. You will find that the SVC code shown in the "0S04I ILLEGAL SVC– . . . " message along with the error codes in register 0 will be helpful in tracing program errors. Each user exit routine must have its own save area in order to preserve the contents of the 16 general registers and interrupt status information at the time the exit routine is entered. The address of the save area is specified in the STXIT macro and is contained in the appropriate option table.

Figure 4.42 (opposite) illustrates the format and contents of the Interrupt Status Information (the first 8 bytes of the save area). Details about the STXIT macro can be found in *DOS/VS Supervisor and I/0 macros,* and details about the option tables in the *DOS/VS Supervisor Logic.*



t 12 is always set to 1
it 13 if set to 1, enables machine check interrupt
it 14 if set to 1, indicates CPU wait state.
t 15 if set to 1, indicates CPU in problem state
if set to 0, indicates CPU in supervisor state

			Cause of Program Interrupt	
Interrupt codes for PC:	X'01'	00000001	Operation	
	X'02'	00000010	Privileged operation	
	X'03'	00000011	Execute	
	X'04'	00000100	Protection	
	X'05'	00000101	Addressing	
	X'06'	00000110	Specification	
	X'07'	00000111	Data	
	X'08'	00001000	Fixed point overflow	
	X'09'	00001001	Fixed point divide	
	X'0A'	00001010	Decimal overflow	
	X'0B'	00001011	Decimal divide	
	X'0C'	00001100	Exponent overflow	
	X'0D'	00001101	Exponent underflow	
	X'0E'	00001110	Significance	
	X'0F'	00001111	Floating-point divide	
	X'10'	00010000	Segment translation	
	X'11'	00010001	Page translation	
	X'12'	00010010	Translation specification exception	
	X'13'	00010011	Special operation exception	
	X'40'	01000000	Monitor call	
	X'8x'	1xxxxxxx	PER event, where x = code	for cause of program check. For example, a PER + a data exception = X'87' (10000111).
Interrupt code for IT:	X'80'	10000000	Timer	
Interrupt codes for OC:	X'40'	01000000	External	

Figure 4.42. The format and contents of the user exit routine save area and interrupt status information.

SAVE AREAS

System Save Areas (for system tasks)

There are occasions when task information must be saved by the page manager. For example, page faults may occur when supervisor services are executed under control of user PIBs. Because the user's partition save area is occupied during this time, an additional system save area for each user task is provided.

The information saves is contained in a 72-byte (dec) field, and includes the return PSW and 16 general purpose registers belonging to the interrupt supervisor task. The registers are stored in numerical sequence beginning with GR9. The save areas for all tasks (maximum of 15), are within the supervisor area. The address of each user task save area is recorded in the program information block. (Refer to Chapter 7 in this Section.)

Immediately following each save area is an area reserved for the CCW/TCB (Channel Command Word/Translation Control Block), the format and contents of which is described in Chapter 13 in this Section. The addresses of system (task) save areas are contained in the system communication region, refer to Chapter 2 in this Section.

Save Areas for the Job Accounting option

If the JALIOCS parameter is specified in the FOPT supervisor generation macro, two save areas are reserved in the supervisor.

$$\text{JALIOCS=} \left\{ \begin{array}{l} \text{NO} \\ \text{(s, 1)} \end{array} \right\}$$

NO indicates that no special LIOCS support is required. Specification of (s, 1) indicates that a user save area and a label area are to be reserved.

S is the decimal number of bytes to be reserved for the user save area (located in the supervisor). This save area may be used to save DTF information or for any other purpose desired by the user. The system does not access this area. (The address of the save area is available in register 13 when $JOBACCT is called.) The valid range of s is 0-1024, with a default of 16. 1 is the decimal number of bytes needed for a label area. This label area replaces the one normally used by LIOCS label processing. It is required when $JOBACCT uses LIOCS for such things as standard tape labels, DTFDA, and DTFPH with MOUNTED=ALL. The valid range of 1 is 0-224, with a default of zero. The value that is substituted for 1 is normally the number of bytes that would be allocated by a given parameter on the LBLTYP statement. See Figure 4.41 in this Section to determine the number of bytes allocated for any given LBLTYP statement.

If the JA parameter is specified and JALIOCS is not the job accounting interface is generated but no alternate label area is reserved (16 bytes are reserved for the save area). The routine $JOBACCT must then use a device or method that does not require LIOCS label programming. If the JA parameter is not specified, the JALIOCS parameter is ignored.

PAGE MANAGEMENT TABLES

The purpose of this chapter is to describe the tables that are used by the page management routines which may need to be inspected during program debugging. A knowledge of the concept of virtual storage is assumed.

The Segment Table

One segment table is generated within the supervisor area during system generation. Each entry in the segment table corresponds to one 64K segment of virtual storage.

How to locate:

The address of the first entry in the segment table is contained in bytes X'D0' to X'D3' of SYSCOM. Label STAB in the supervisor listing identifies the address of the first byte of the segment table. The address of the segment table is also contained in control register 1. Refer to the example shown in Figure 4.47.

The Page Table

One page table is generated for each segment of virtual storage during system generation. Each page table is 64 (decimal) bytes in length, and has 32 two-byte entries. Each entry corresponds to 2048 (decimal) bytes of virtual storage. As illustrated in Figure 4.46, the page tables occupy a consecutive area in the supervisor.

Initialization of the Page Table

After IPL, page table entries are initialized as follows:

- All page table entries belonging to the supervisor area (nucleus and transient areas):
  Bit 13 = 0
  Bit 15 = 0
  Bits 0-12 = the leftmost 13 bits of the address of the corresponding page frame.
- All page table entries for allocated real partitions:
  Bit 13 = 0
  Bit 15 = 1
  Bit 0 = 1
  Bits 8-11 = storage key of the partition.
- Page table entries belonging to virtual partitions:
  Bit 13 = 1
  Bit 15 = 1
  Bit 0 = 0
  Bits 8-11 = storage key of corresponding partition.
- All remaining page table entries:
  Bit 13 = 0
  Bit 15 = 1
  Bit 0 = 1
  Bits 1-12 = 0

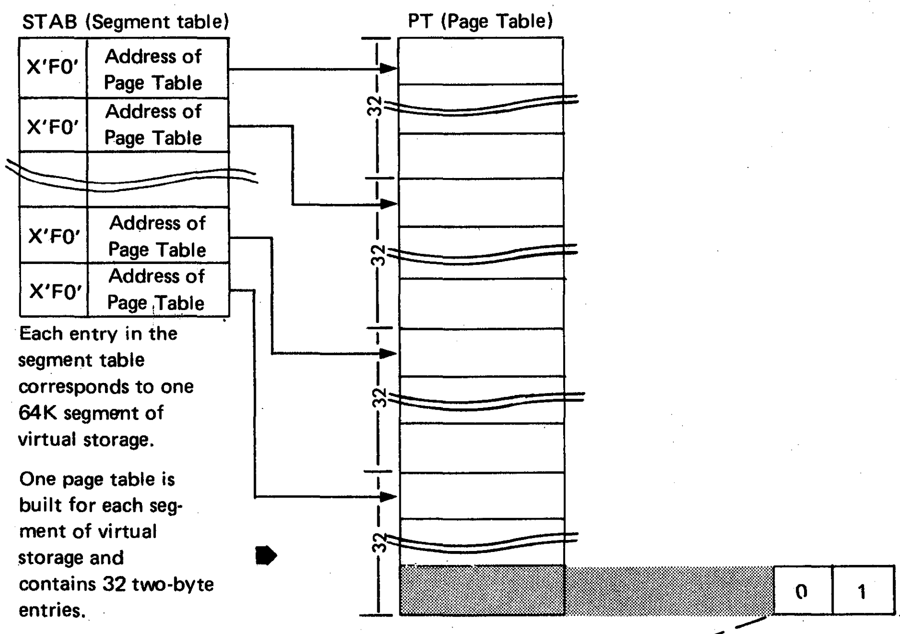
How to locate

The address of the page table belonging to the first 64K of virtual storage is contained in the first entry in the segment table.

The address of the page table belonging to subsequent segments of virtual storage are contained in the associated segment table. Refer to Figure 4.46 which illustrates this.

Appendix G shows an example of locating the segment table and page table in a dump.

Figure 4.43 (below) shows the format and contents of an entry in the segment table and an entry in the page table. The figure also illustrates the interconnection between these two tables.



Bytes 0 and 1 :	Bit 0	:	1= Address of any byte in the corresponding page is invalid, that is the page is not in real storage
	Bits 0-12	:	Leftmost 13 bits of address of page in real storage if bit 13=0.
	Bits 8-11	:	Storage key of corresponding partition if bit 13= 1.
	Bit 13	:	0 = Page is in real storage; 1 = Page is not in real storage.
	Bit 14	:	Always zero.
	Bit 15	:	0 = Page must be read from page data set, (a valid copy of the page is on SYSVIS); 1 = Page need not be read from page data set, (there is not a valid copy of the page on SYSVIS).

Note: Label STAB identifies the first byte of the table
Label PT identifies the first byte of the Page Table.

Figure 4.43. Explanation of the contents of an entry in the Page Table.

This figure also illustrates the relationship between the page table and the segment table.

PAGE MANAGEMENT TABLES

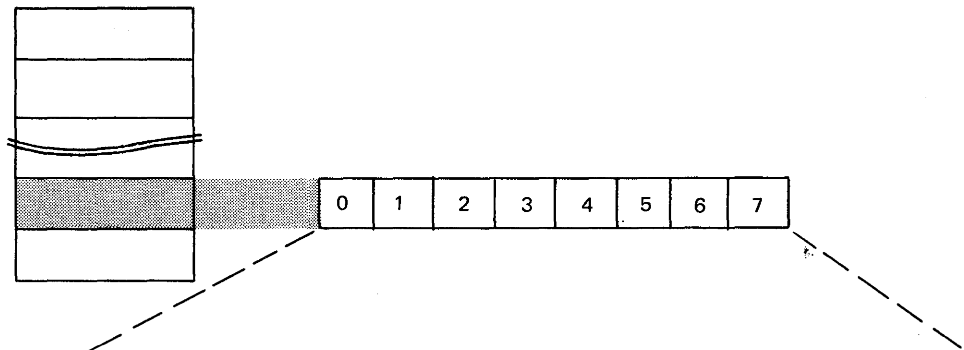
Page Frame Table

The page frame table is built at supervisor generation time and contains one 8-byte entry for each 2K block of real storage (page frame) as specified in the RSIZE parameter of the VSTAB macro.

How to locate:

Bytes X'D4'–X'D7' of the SYSCOM contain the address of the first entry in this table. Label PFT in the supervisor listing identifies the address of the first byte of this table.

The format and contents of an entry in the page frame table is shown below.



Bytes 0 and 1:

Bits 0-10 :	Counter for temporary fixes (tfix counter.)
Bit 11 :	Reserved for later use.
Bit 12 :	1= PFIX request (into another page frame) pending for the page.
Bit 13 :	If on, page frame is unusable (hardware failure). DRAP BIT
Bit 14 :	If 0, page frame belongs to selection pool SP BIT
Bit 15 :	1= PFIX (different page) or GETREAL request pending for the page frame.

Bytes 2 and 3:	Pointer to next page frame in queue or to queue header if last element in the queue (all page frames in selection pool are queued).
Bytes 4 and 5:	If the frame is occupied, the page number (The virtual address of the page divided by 2048); if unused, X'FFFF'.
Bytes 6 and 7:	Pointer to the previous frame in the queue or to queue header if first element in the queue (all page frames in the selection pool are queued).

Figure 4.44. Explanation of the contents of an entry in the Page Frame Table.

Page frame table extention (PFTX)

This table is a one-byte appendage to each entry in the page frame table. It serves as a counter for the number of times a page has been permanently fixed in a page frame, and is labeled PFIX counter.

How to locate

Bytes X'D8' – X'DB' of SYSCOM contain the address of the first entry in this table. Label PFTX in the supervisor listing identifies the address of the first byte of this table.

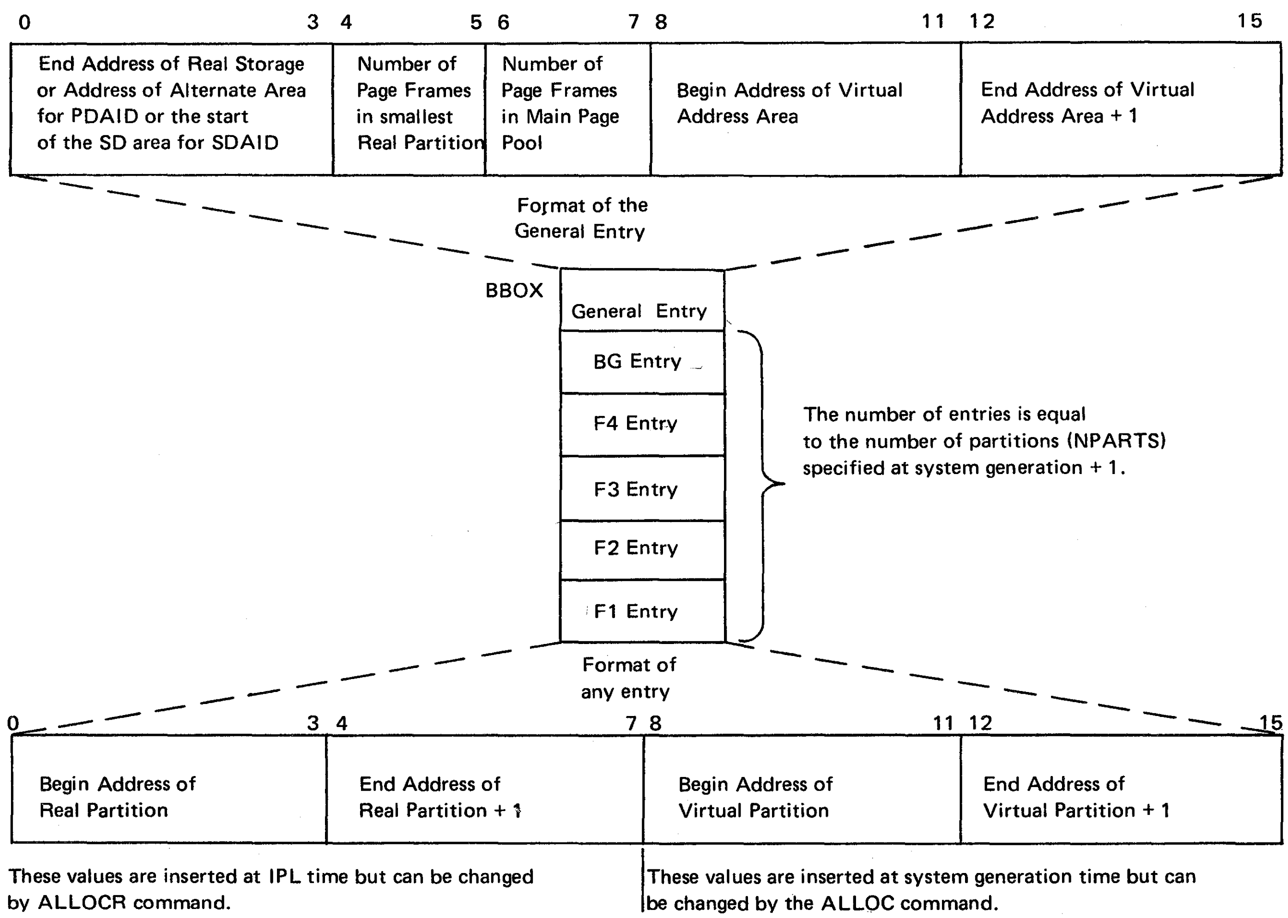
Boundary Box

This information block is generated in the supervisor during system generation. The area occupied by the boundary box is sufficient to contain up to six entries, depending on the number of partitions specified during system generation. The first entry contains information about virtual storage allocation, and the remaining entries contain information pertaining to each partition supported by the supervisor. If a partition is not supported by the supervisor, the beginning and end addresses are identical to those of the next partition.

How to locate:

Bytes X'DC' -- X'DF' of SYSCOM contain the address of the first entry in this information block. Label BBOX in the supervisor listing identifies the address of the first byte of this information block. Appendix G shows an example of locating the boundary box in a dump.

The format and contents of the boundary box is shown below:



***Note:** The begin and end address fields for a partition that is not allocated contain the begin and end addresses of the following partition.*

Figure 4.45. Explanation of the contents of the Boundary Box

PAGE MANAGEMENT TABLES

Converting virtual to real addresses and vice-versa

There are several methods of calculating real addresses from virtual and vice versa. One method is given below.
(The values assumed in the examples apply to the illustration opposite.)

A. Converting a virtual address to real:

1. Write the hexadecimal virtual address in binary. For example (assuming a virtual address of 1FA20),

   1FA20
   0001 1111 1010 0010 0000

2. Ignore the ten rightmost bits. For the example, this leaves
   0001 1111 10
3. If after step 2 the rightmost bit is a 1, change it to a 0; if it is a 0, leave it 0.
4. Convert the binary value obtained in step 3 to hexadecimal.
   For example,

   0001 1111 10
   7 E

5. Locate the address of the page table, contained in the first entry of the segment table, the address of which is contained in CR1. (For example shown opposite, this is 6A28,)
6. Add the address of the page table to the hexadecimal number obtained in step 4.
   For example,

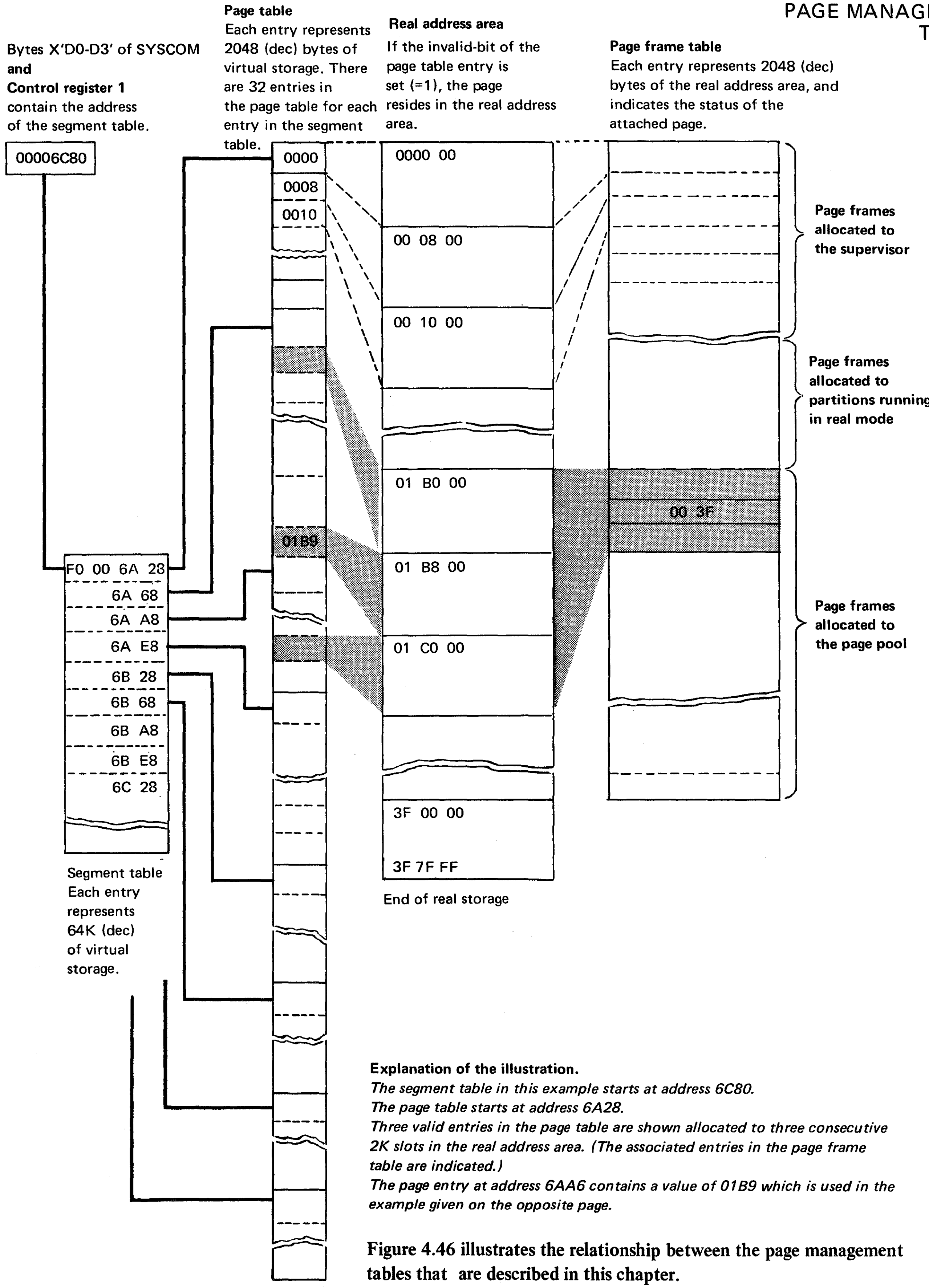
   + 6A28
   7E
   = 6AA6 *(This is the address of the entry in the page table associated with the virtual address to be converted to real.)*

7. Locate the page table entry in the dump.
8. Replace the right most bit of the contents of the page table entry by a 0.
   For example, as shown in the illustration opposite, the page table at address 6AA6 contains 01B9. The right most bit is a 1 (X'9' = 1001.)
9. After replacing the right most bit by a 0, convert the resulting four-bit binary string to hexadecimal.
   For the example, 1000 - X'8'.
   The value thus obtained in this example is 01B8.
10. Increase the value obtained in step 9 by attaching two 0s to the right.
    For example, 01B800 *(This number is the address in real storage of the lower limit of the page frame in which the real address is located.)*
11. Convert the eleven rightmost bits of the binary value obtained in step 1 to hexadecimal.
    For example,

    010 0010 0000
    2 2 0

12. Add the value obtained in step 11 to the number obtained in step 10.
    For example,

    + 01B800
    220
    = 01BA20 *(This is the real address.)*



Explanation of the illustration.

The segment table in this example starts at address 6C80.
The page table starts at address 6A28.
Three valid entries in the page table are shown allocated to three consecutive 2K slots in the real address area. (The associated entries in the page frame table are indicated.)
The page entry at address 6AA6 contains a value of 01B9 which is used in the example given on the opposite page.

Figure 4.46 illustrates the relationship between the page management tables that are described in this chapter.

PAGE MANAGEMENT TABLES

Converting a real address to virtual:

1. Write the hexadecimal real address in binary. For example,

   01BA20

   0000 0001 1011 1010 0010 0000

2. Ignore the eleven rightmost bits. Thus, for the example,

   0000 0001 1011 1 is the remaining binary number.

3. Add three 0s to the right of the binary number obtained in step 2. Thus

   0000 0001 1011 1000

4. Convert this binary value to hexadecimal. For example,

   0000 0001 1011 1000
   0 1 B 8

5. Add the number obtained in step 4 to the address of the page frame table. Bytes X'D4' – X'D7' of SYSCOM contain the address of the page frame table. (For the example, this is assumed to be 6100.) For example,

   $$\begin{array}{r} 6100 \\ +\ \ 1B8 \\ \hline =\ 62B8 \end{array}$$

6. Locate this address in the dump. (This is the address of the page frame table entry associated with the real address to be converted.)

7. Locate bytes 4 and 5 of this page frame table entry. (For the example, as illustrated in Figure 4.46, a value of 003F is assumed.)

8. Write this hexadecimal number in binary. Thus for the example,

   0000 0000 0011 1111

9. Ignore the leftmost three bits, and add three 0s to the right hand end of the resulting binary string. Thus,

   0000 0001 1111 1000
   0 1 F 8

11. Convert the eleven rightmost bits of the real address (as written in binary in step 1) to hexadecimal. For example,

    010 0010 0000
    2 2 0

12. Add the number obtained in step 10 to the number obtained in step 11. For example,

    $$\begin{array}{r} 01F800 \\ +\ \ \ \ 220 \\ \hline =\ 01FA20 \end{array}$$ *(This is the virtual address.)*

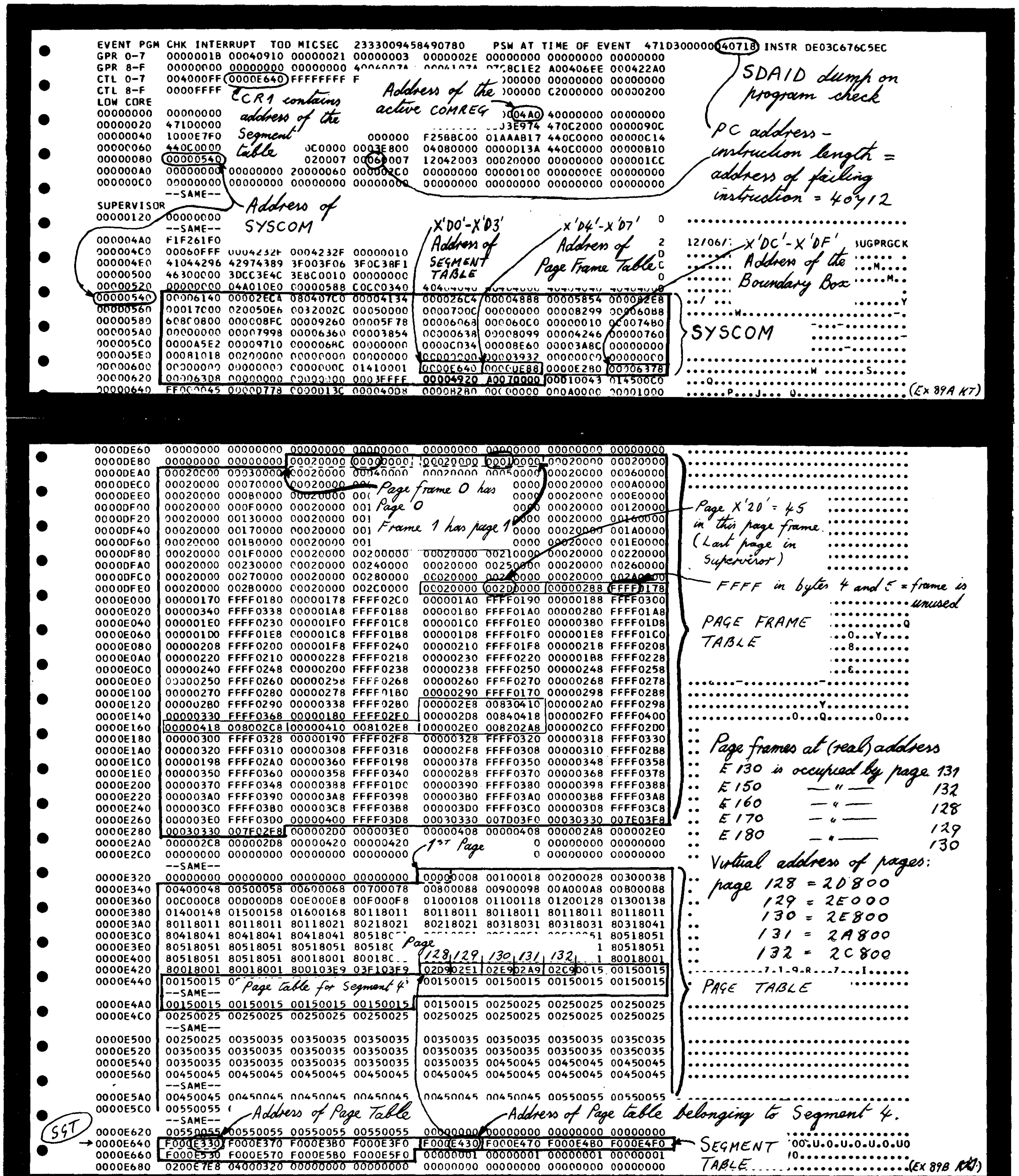


Figure 4.47. An example of an SDAID "dump on program check" showing how to locate the page management tables.

CHANNEL PROGRAM TRANSLATION

Channel program translation

This chapter describes the control blocks used by the DOS/VS channel program translation routines, which may require examination under certain circumstances of a system malfunction.

I/O operations and of virtual storage

For a full description of the channel program translation routine, refer to the *DOS/VS Supervisor Logic* manual.

General functions

Because the DAT (dynamic address translation) feature is not available for data and channel command words of I/O operations, software routines are required that perform the following functions for an I/O request from a virtual partition:

1. The CCB and, if applicable, the user sense CCW will be copied into a buffer. This buffer is called the CCB copy block and is maintained by the CCW-translation buffer management.
   If a second I/O operation from a virtual partition is requested, the copied and translated CCB is queued behind the first request in the CCB copy block queue. Label ACCBB in the supervisor listing points to the address of the CCB copy block queue. Displacement X'44' from this address contains the address of the second CCB copy block.

2. The complete channel program, consisting of one or more CCWs, will be copied into a buffer area called the CCW copy block. The copied channel program is logically equivalent to the original channel program, the data addresses being translated to real addresses. The copy process conserves the channel program structure, but TIC (transfer in channel) commands will be inserted in the copied channel program when there are more than seven CCWs in a channel program. Figure 4.51 shows a channel program having eleven CCWs; two copy blocks that are linked by a TIC command are therefore required. Figure 4.52 illustrates the format and contents of the CCW/TCB.

3. Addresses in the copied channel program that refer to an I/O area in a virtual partition are translated into the corresponding real addresses. If the I/O area is completed on one page, the real address will replace the virtual address in the copied channel program. If the I/O area occupies more than one page (crosses page boundaries), an IDAL (Indirect Data Address List) block is built up. The IDAL block contains the real address of the I/O area and the real page addresses of any pages occupied by the I/O area. The address of the IDAL will replace the virtual address in the copied CCW, and the IDAL bit (bit 37 in the CCW) will be set to 1. If the virtual channel program already uses the IDA feature, both the IDAL from the virtual partition copied and the virtual addresses will be replaced by the corresponding real addresses.
   Figure 4.48 illustrates the actions described in points 1, 2 and 3 above, and Figure 4.49 illustrates the relationship between the blocks described.

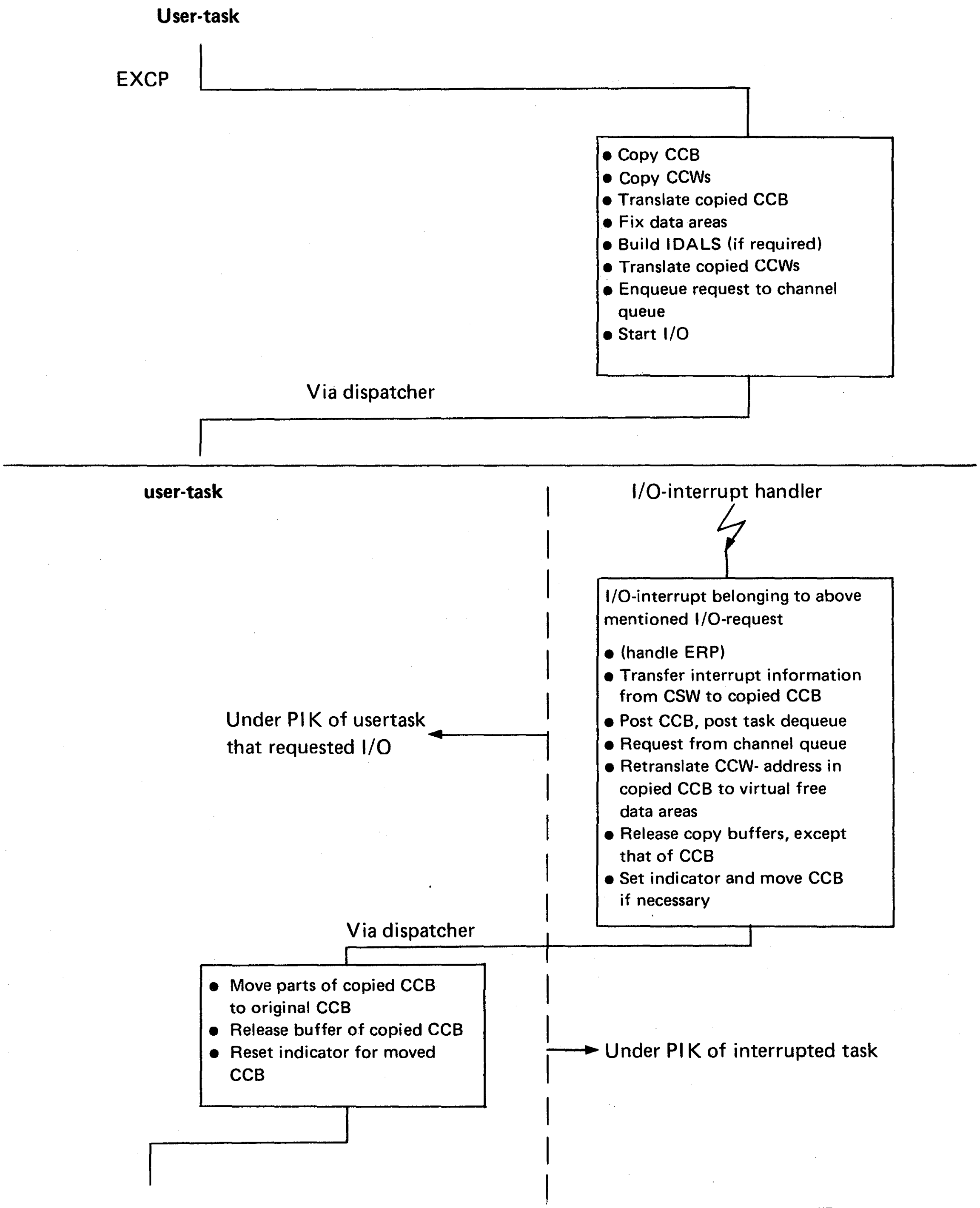


Figure 4.48. Illustrates the activity between user program and supervisor during the handling of an I/O request from a program running in virtual mode.

CHANNEL PROGRAM TRANSLATION

Additional functions performed by the channel program translation routine are:
4. A sense CCB (if applicable) is updated in the copied CCB.
5. All I/O areas required by the channel program must remain in real storage until the I/O operation is complete. For this reason all pages involved with an I/O operation are fixed in real storage.

After the above functions have been performed, the I/O request is handled as if it were a request from a real partition. The following supervisor activity then ensues:
1. The request is placed in the channel queue.
2. A START I/O is issued.
3. The corresponding interrupts are processed.
4. The ERPs are activated (error recovery procedures in case of I/O device errors).
5. Status information is posted in the copied CCB.
6. The request is removed from the channel queue.

After completion of an I/O request from a virtual partition, the channel program translation routine translates the real command address (from the CSW) to the corresponding virtual address, frees all fixed pages that were required by the request, transfer parts of the copied CCB to the virtual CCB, and releases all areas used by the buffers required by the channel program translation routine.

Figure 4.48 illustrates the complete operation described above under points 1 through 5.

IDAL block

The IDAL block is generated by the CCW translation routine if the I/O area specified in a CCW crosses page boundaries. The IDAL blocks are placed in the buffer area at the end of the supervisor together with associated CCB and CCW copy blocks. Each block contains real addresses of the data areas in real storage. Because each address is 4 bytes in length, an IDAL block can contain up to 18 addresses (also referred to as IDA words.) Each IDAL must be completely contained in one IDAL copy block. If more than one I/O request requires an IDAL, as many IDALs are placed in one IDAL copy block as will fit.

The figure opposite shows the relationships between the blocks.

Appendix G shows an example of locating a CCB copy block and CCW copy block in a stand-alone dump output. The CCB address in the channel queue is used as the initial pointer.

CCB copy block

CCB copy blocks are placed in a buffer area (specifically reserved for the channel program translation routine) at the end of the supervisor together with the associated CCW copy blocks and IDAL block (if required). Each CCW copy block consists of nine entries. The first seven entries are used to store copied CCWs, and the remaining two entries (16 bytes) contain pointers and end-of-buffer indicators. The format and contents of a CCW copy block is shown in Figure 4.50.

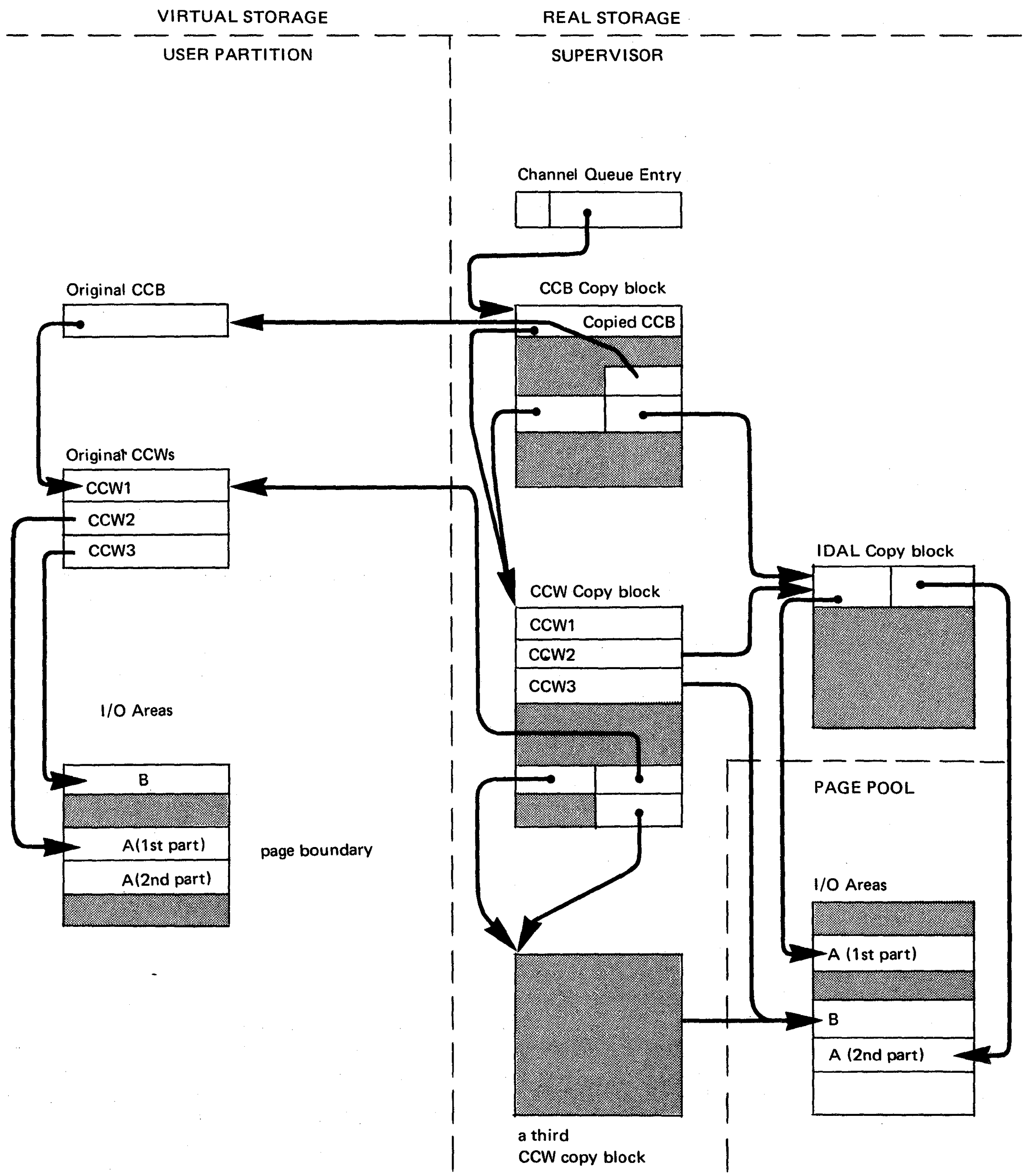


Figure 4.49. **Illustrates the relationship between an original channel program in a virtual partition and the copy blocks required by the channel program translation routines. The input/output areas in real storage are also shown.**

CHANNEL PROGRAM TRANSLATION

Legend: Square keys refer to the description below.

	0	1	2	3	4	5	6	7
0	CCBCNT		CCB COM1	CCB COM2	CCB STA1	CCB STA2	CCB CLS *	CCB LNO
8	CCBCCW Address of first CCW				CCBBY3	CCBCSWW		
16	CCBSENS Sense CCW if any							
24	CCBPIK User PIK		CCB FLAG **	Unused	CCBVA Virtual Address of CCB			
32	CCBACB Address of first CCW copy block in channel program				CCBICB Address of first IDAL block in channel program			
40	CCBXINF (Fix information; 24 bytes) Each bit in this field represents one page frame. If a bit is on, the associated page frame contains a page fixed for this I/O request. If more than 384K of real storage are available, the address in CCBXPTR will point to any additional field which contains bits for the page frames beyond 384K.							
48								
56								
64	CCBXPTR Address of additional Fix information				CCBNEXT Address of next CCB copy block			

* Set to X'21' (= copied CCB)

** Legend CCBFLAG: Bit 0: 1= Translation complete
1: 1= Pages fixed
2: Not used
3: 1= BTAM Second Time Request (I/O request from BTAM appendage)
4: Not used
5: Not used
6: Not used

Field	Description
	(16 bytes): Copied and updated CCB.
	(8 bytes): If a user sense CCW is available, the CCW will be copied into this area. If the sense I/O area crosses a page boundary, an IDAL will be generated and the address of the IDAL will replace the address in Sense CCW.
	(2 bytes): Contains the PIK-value of the virtual I/O requestor. This value will be used by the MOVECCB routine to identify the requestors CCBs.
	(4 bytes): Contains the virtual address of the original CCB.
	(4 bytes): Address of the first CCW copy block occupied by the real channel program.
	(4 bytes): Address of the first IDAL block of zero, if no IDAL is needed.
	(24 bytes = 128 bits): Contains the fix information for the I/O request (FIXINF). Each bit corresponds to a physical page frame. If a bit is one, the corresponding page is fixed for the current I/O request. The 128 bits are sufficient for a Relocate System with up to 384K bytes of real storage.
	(4 bytes): If real storage is greater than 384K, the FIXINF is logically continued in another copy block with 68 usable bytes corresponding to 1032K additional real memory. The address in H will point to the attached fix information field. For real storage not greater than 384K, the value of H will be zero.
	(4 bytes): Contains a chain pointer. All CCB copy blocks will be enqueued into a CCB Copy queue. CHAINPTR points to the next CCB copy block on the queue. If this copy block is the last one, CHAINPTR equals zero. The pointer ACCBB either will point to the first CCB copy block on this queue or is zero, if no CCB is copied.

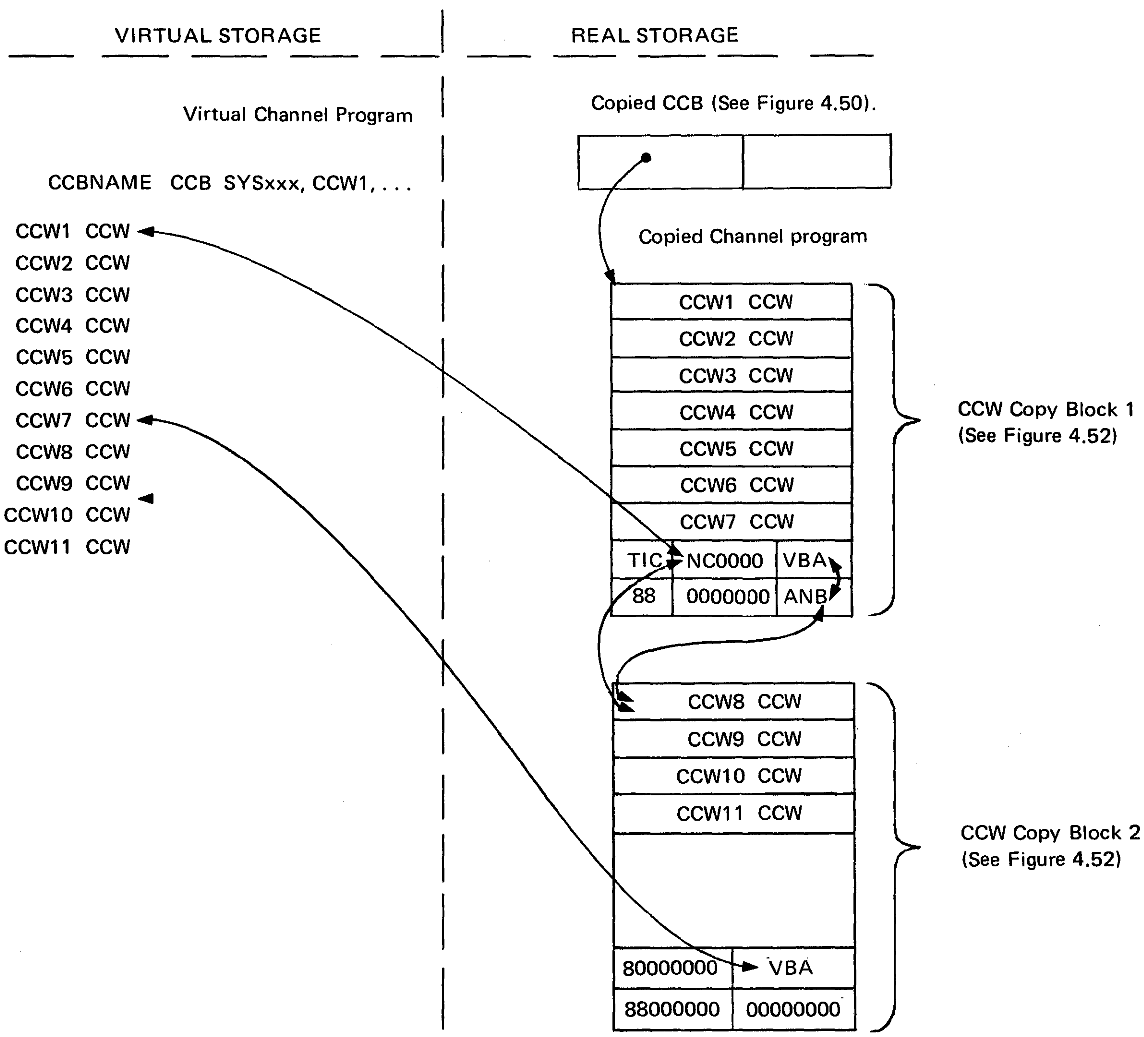
4.123 **Figure 4.50. Explanation of the contents of a CCB copy block.**

Channel Program without TIC or SEARCH Commands

The CCWs in a channel program without TIC or SEARCH commands are copied into sequential locations in the CCW copy buffer. If the program has more than seven CCWs, a TIC is inserted in the eighth copying position and is made to point to the first CCW in the next copy buffer. CCWs 8 through 14 are then copied in the next copy buffer. If there are more than 14 CCWs, the process is repeated until all CCWs are copied.

Refer to the *DOS/VS Supervisor Logic* manual for a full description of the CCW copy block when using TIC and SEARCH commands.

Figure 4.51 (below) shows the copying of CCWs for a channel program requiring two CCW copy buffers.



Legend: TIC = Transfer in channel command
ACCW8 = Address of CCW8
ANB = Address of next CCW Copy Block
VBA = Virtual Address of CCW1 (for Copy Block 1) and virtual address of CCW8 (for Copy Block 2).

Figure 4.51. A channel program requiring a TIC to be inserted in copying.

	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 the Copy block			
64	X'88'**	X'000000'			Address of next CCW Copy block in the chain			

* 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.

** 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. For example a SEARCH command to a disk. Bytes 65-67 will then point to the copy location of the second CCW following the status modifier CCW.

Figure 4.52. Format and contents of a CCW copy block

Translation Control Block

The routine CCWTRANS is called by the channel scheduler whenever a channel program must be copied and translated. Since a page fault may occur during CCWTRANS, the routine and its subroutines are reenterable and can therefore process several translation requests concurrently. In order to make CCWTRANS reenterable a translation control block (TCB) is built for each task to serve as a dynamic work and save area. Each TCB is located behind the special save area for its task and has the format shown in Figure 4.53.

How to locate:

To locate the TCB (associated with the partition/task), add X'50' to the address of the System Save Area (displacement X'09' of the appropriate PIB). Labels CCWTCB1 – CCWTCBn identify the first byte of the appropriate TCB.

Format of any TCB

0	1	2 3	4 7	8 11	12 15	16 19	20 23	24 27
Flag byte *	used by BTAM	TIK/PIK	Pointer to Status Modifier List	Pointer to Control Com'd List	Pointer to TIC line	Pointer to Copy Block End	Address of copied CCB (for cancel)	Number of free IDA words in IDAL block

28 47	48 51	52 107	108 111
Work Areas	Address of last TFIX request	Save Area (Registers 2-F)	Pointer to next used TCB

* Byte 0: bit 0 = 1 : data chaining specified
1 = 1 : Read/Sense command specified
2 = 1 : Read backward command specified
3 = 1 : Status modifier command with data chaining
4 = 1 : Status modifier command only
5 : Reserved
6 : Reserved
7 : Reserved

Note: One TCB is generated for each partition supported.
With asynchronous processing support, 15 TCBs are generated.

Figure 4.53. Explanation of the contents of the TCB.

CHANNEL PROGRAM TRANSLATION

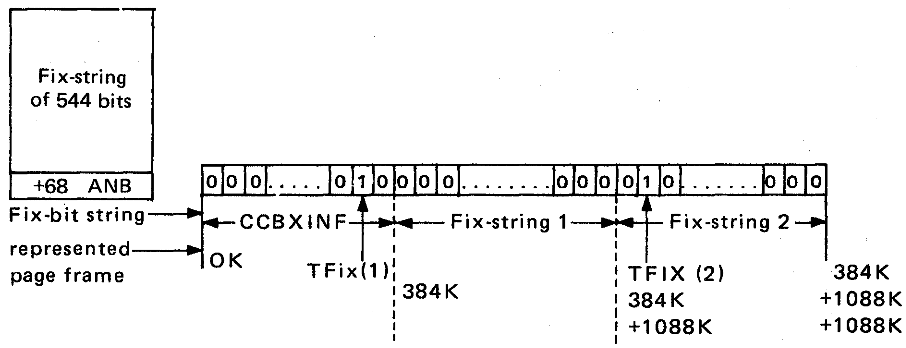
Fix Information Blocks

In order to keep track of which page frames have been TFIXed for a request, a bit string is used which has a bit for at least every page frame up to the highest one which is TFIXed. If no page is TFIXed in an address higher than 384K, then the bit string in CCBXINF is sufficient (192 bits = 384K). Whenever a page is TFIXed, the bit corresponding to its page frame is set to one. If a page is used more than once by a request, it is TFIXed only once.

If a page is TFIXed at a location beyond 384K, one or more additional bit strings must be added. This is done by enqueuing a copy block. Each copy block thus enqueued provides fix information for an additional 1088K of real storage. The additional blocks are queued with the first one being pointed to by the field CCBXPTR in the CCB copy block. Figure 4.54 shows how fix information is kept.

Fix-String:
bit-table where each bit belongs unequivocally to a page frame (for 1038K bytes); if a bit is on, the page frame belonging to this bit has been TFIXed for this I/O-request.

ANB:
–0 if Fix-block is last one in Fix-block queue.
–address of next Fix-block.



- *if for a specific page frame the Fix-bit is already on, noTFIX-request is transferred to the .page manager*

- *the TFIX-blocks are freed after I/O-request has been posted complete*

Figure 4.54 Fix information Bit String and Block

General Rules for channel program translation

The following rules apply to IBM-supplied channel program translation routine

1. Channel program translation is skipped:
   - for I/0 requests from programs running in real mode
   - for I/0 requests from system tasks (FETCH/LOAD has its own small CCW-translation.)
   - for I/0 requests for console when console buffering option is supported
2. Channel program translation is modified for BTAM running in virtual mode (modify CCW-chain from I/0 appendages).
3. The following components work via copied and translated CCW chains:
   - CRT
   - seek separation routine
   - ERPs
   - BTAM-ERPs
4. Channel program translation does not support:
   - self-modifying channel programs
   - start of I/0 requests from I/0 appendages for not translated channel programs (except BTAM)
   - time dependent I/0 requests (channel program may get longer after translation)
   - channel programs with CCWs whose count is 32K
   - channel programs with data chaining in connection with TIC-commands when the same CCW gets different command codes during execution of channel program.

CONTENTS

Appendix A

PD AREA TABLES

	Byte	Field	Description
PDAREA	0	PDAREAND	— Address of end of PD area
	4	PDINTRHK	— Address of interrupt trace hook*
	8	PDSIOHK	— Address of START I/O hook*
	12	PDTRANHK	— Address of transient dump hook*
	16	PDFLTHK1	— Address of F/L trace hook 1*
	20	PDFLTHK2	— Address of F/L trace hook 2*
	24		— Reserved
PDREGSAV	28		— Save area for registers 9, 10, and 11
	32		
	36		

**A hook is coding introduced at supervisor generation. The coding normally branches around itself. The initialization makes the branch instruction a NOP to allow a PDAID function to be performed.*

Figure A.1. The PD address table (Displacements are in decimal)

Byte				
0	Phase Name			
8	VER	MCD	Log CUU	
16	IGN2/TRC2 CUU	IGN3/TRC3 CUU	Alternate Area Start	
24	Alternate Area End		CHANC PTR	
32	PRT1	PRT2	PRT4	PRT5	Reserved
40	Reserved			
48	Reserved	OPT	Register 1Ø	

Displacement	Label	Description
0-7	Phase Name	Phase being run
8	VER	Version number in hex
9	MOD	Modification level in hex
10-11	LOG	Address of system log device
12-13	Output	Address of output device
14-15	IGN1/TRC1	Address(es) of devices to ignore or trace
16-17	IGN2/TRC2	
18-19	IGN3/TRC3	
20-23	Alternate Area Start	Start address of alternate area
24-27	Alternate Area End	Ending address of alternate area
28-31	CHANG PTR	Address of channel queue pointer for output device
32	PTR1	Partitions to be ignored (see note)
33	PTR2	
34	PTR3	
35	PTR4	
36	PTR5	
37-50	Reserved	
51	OPT	Option byte X'00' = TRC device X'80' = IGN device
52-55	Register 10	Save area for register 10 (used by GSVC trace only)

Note: The initializer inverts the logic. When the user specifies a partition(s) to be traced, PDAID enters the partition(s) to be ignored in the standard preface table.

Figure A.2. The PD Standard Preface Table (Displacements are in decimal)

PIK (Partition Identification Key)

During debugging, it may be necessary to locate and to be able to interpret the PIK value allocated to each partition.

The PIK of each partition is determined during system generation the PIK value being contained in a two byte field at displacement address X'2E' in the appropriate partition communication region. Byte 0 of this location always contains X'00', and byte 1 a hex number equal to the displacement from the start of the PIB to the start of the entry in the PIB belonging to the partition.

Foreground partition PIK

The PIK value for foreground partitions depends on how many partitions are specified by NPARTS parameter of SUPVR macro.

PIK value at displacement address X'2E' of foreground COMREGs, where byte 0=X'00' and byte 1= ;				
NPARTS=5	NPARTS=4	NPARTS=3	NPARTS=2	PIK value indicated
50	40	30	20	F1
40	30	20		F2
30	20			F3
20				F4

Note: The PIK values for foreground partitions do not change during system operation.

Figure B.1. PIK Values

Background partition PIK

The PIK value for the BG partition is always X'10'. However, unlike the values in the foreground communication regions, the value held in this address changes during system operation. It always contains the PIK value of the active partition.

Supervisor PIK

A separate PIK value is given to the attention routine. The value is X'00', and indicates that the supervisor is in control.

SYSLOG ID

For PD output, values will be AR, BG, F4, F3, F2, or F1, which identify the partition generating the trace entry.

TIK, LIK

TIK (Task Interrupt Key)

The halfword TIK at displacement X'5A' in the SCP Communications Region (SYSCOM) has a zero value in the high-order byte and a key value in the low-order byte. This key value is only significant when AP is supported. The key value in the TIK is the key of the program (task or subtask) that is being serviced. When an interruption occurs, the value of the TIK indicates to the supervisor which program (task or subtask) was interrupted.

The TIK is set by Task Selection in the General Exit Routine and equals the index displacement of the task's Program Information Block (PIB) within the PIB Table.

Depending on the number of partitions supported, the value of the TIK indicates which task was interrupted according to the following table:

TIK Value				Interrupted Task
NPARTS=2	NPARTS=3	NPARTS=4	NPARTS=5	
X'00'	X'00'	X'00'	X'00'	Attention
X'10'	X'10'	X'10'	X'10'	BG
			X'20'	F4
		X'20'	X'30'	F3
	X'20'	X'30'	X'40'	F2
X'20'	X'30'	X'40'	X'50'	F1
X'30' – X'F0'	X'40' – X'F0'	X'50' – X'F0'	X'60' – X'F0'	Subtasks*

**Asynchronous Processing option. The number of PIBs initially available for subtasks is 10, 11, 12, or 13, depending on the number of partitions (in an AP supervisor the total number of PIBs is always sixteen).*

LIK (Logical Transient Owner Identification Key)

The halfword LIK at displacement 88 in the SCP Communications Region (SYSCOM) is only significant if AP is supported and contains the same value as the TIK when the logical Transient Area (LTA) is in use. LIK therefore, identifies the owner of the LTA. When the LTA is free, the halfword LIK contains zeros. The SVC 2 routine sets the LIK, and the SVC 11 routine resets it to zero. If AP is not supported the LIK contains zeros.

LTK (Logical Transient Key)

The halfword LTK at displacement X'6E' in each partition's communications region has a zero value in the high-order byte and a key value in the low-order byte.

In a foreground communications region, the key value in the LTK is not significant. The LTK in the background communications region (BGCOMREG) 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 a task, therefore, the BGCOMREG has the same value in its LTK as in its PIK when the owning task is active. The SVC 2 routine sets the LTK, and the SVC 11 routine resets it to zero.

REQID (I/O Requestor's Partition or System Task ID)

The REQID is one-byte identifier in the Channel Queue (CHANQ) entry.

When a background or foreground program has requested I/O, the REQID has the same value as the key byte of the PIK for that task's partition. When the Attention Task has requested I/O, the REQID contains X'00'.

When the request for I/O is from a System Task, the REQID has one of the following values:

RAS – X'01'
PMGR – X'02'
SUPVR – X'03'
CRT – X'04'
ERP – X'05'
PAGEIN – X'06'

The REQID is set by the Channel Scheduler Routine.

Note that X'00' indicates that no system task is active.

TKREQID (I/O Requestor's Task Identification)

The TKREQID is a one-byte identifier in the Channel Queue (CHANQ) entry for a task that has requested I/O (see Figure 4.13). In an unused CHANQ entry the TKREQID contains X'FF'.

The TKREQID byte in an active CHANQ entry has the same value as the key byte of the TIK of the task that has requested I/O.

If AP is not supported it has the same value as the PIK of the task that requested the I/O.

TKREQID is set by the Channel Scheduler Routine and reset by the I/O Interrupt Handler.

Appendix C

CONTROL REGISTER ALLOCATION

Control Register			
0	SYSTEM CONTROL	TRANSL. CONTROL	EXTERNAL-INTERRUPTION MASKS
1	SEGM–TBL LENGTH	SEGMENT–TABLE–ORIGIN ADDRESS	
2	CHANNEL MASKS		
3			
4			
5			
6			
7			
8			MONITOR MASKS
9	PER EVENT MASKS		PER GR ALTERATION MASKS
10		PER STARTING ADDRESS	
11		PER ENDING ADDRESS	
12			
13			
14	ERROR–RECOVERY CONTROL & MASKS		
15		MCEL ADDRESS	

Figure C.1.

ESERV EXAMPLE JOB STREAMS

The following two examples show the two different features of the ESERV program: that of de-editing *without updating* an edited macro definition, and that of de-editing and *updating* an edited macro definition.

Sample Coding for De-editing without Updating a Macro Definition

```
// JOB NOUPDATE          (See note 1)
// EXEC ESERV            (See note 2)
PUNCH E.MAC1,MAC2        (See note 3)
/*
/&
```

Notes:
1. *Name of job is NOUPDATE.*
2. *Causes ESERV to de-edit the macro specified in the following PUNCH statement.*
3. *Causes the macros MAC1 and MAC2 to be punched out from the macro library (E)*

You could use the above coding to produce a de-edited source macro for possible future updates.

Sample Coding for De-editing and Updating a Macro Definition

The Procedure in the following example produces a de-edited, updated macro definition in source format, and edits and places the update macro definition in the macro library, using the MAINT program.

```
// JOB UPDATE
// EXEC ESERV                   Causes ESERV to de-edit the macro
                                specified in the following DSPCH statement.
GENEND                          Causes an END and /* statement to be
                                generated. These are necessary to allow
                                output from ESERV to be used immediately
                                as input to assembler program.
DSPCH E.MAC1                    Causes the macro definition MAC1 to be
                                punched and printed from the macro
                                library (E).
) COL 77,4                      For explanation see: "Verifying/
) VER 72+1,5                    Updating Statements from a Printout
                                of Macro Definition".
.PP9
) ADD 72+1
  AIF (&PCH NE 1400) D4
) DEL 102,103
) REP 245
JOYCE CLC 0 (4,REGG) ,BLANKS
) END
/*
// PAUSE                        Check list, move deck to reader.
// OPTION EDECK,NODECK          Causes the assembler to produce an edited
// EXEC ASSEMBLY                deck (EDECK): no object module
                                will be produced (NODECK).
      (deck produced by ESERV
       goes here)
/*
// PAUSE                        Move SYSPCH deck to reader.
// EXEC MAINT                   Causes MAINT to put edited macro
                                definition on macro library.
      (deck produced by
       assembler goes here)
/&
```

Appendix E

PROGRAM EVENT RECORDING

The purpose of the program-event-recording facility is to assist in debugging programs, for example, SDAIDS. It permits the program to monitor the following events:

- Successful execution of a branch instruction within the designated virtual storage limit
- Alteration of the contents of designated general registers
- Fetching of an instruction from designated storage locations
- Alteration of the contents of designated storage locations

The information for controlling program-event-recording resides in control registers 9, 10, and 11, and consists of the following fields:

Control register 9:

EVENT M.		GR ALTERATION M.
0	8	16 31

Control register 10 (X'A'):

	STARTING ADDRESS
0	8 31

Control register 11 (X'B'):

	ENDING ADDRESS
0	8 31

PER Event Mask: Bits 0-7 of control register 9 specify which events are monitored. The bits are assigned as follows:
Bit 0: Successful Branching
Bit 1: Instruction Fetching
Bit 2: Storage Alteration
Bit 3: General-Register Alteration
Bit 4: Unassigned
Bit 5: Unassigned
Bit 6: Unassigned
Bit 7: Unassigned
Bits 0-3, when ones, specify that the corresponding events are monitored. When a bit is zero, the event is not monitored.

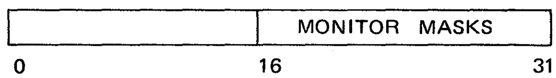
Appendix E

MONITOR CALL

The monitor call instruction provides the capability for passing control to a monitoring program such as the IBM supplied SDAID trace routines, when selected indicators are reached in the monitored program. The indicators are MONITOR CALL instructions implanted in the monitored program. When executed, these instructions cause a program interruption for monitoring to take place, provided that an interruption is allowed for the monitor class specified by the instruction. Along with the interruption, the monitor class number and a monitor code are stored for subsequent use by the monitoring program.

The instruction MONITOR CALL designates one of 16 monitoring classes together with a set of 16 monitor masks in a control register. One mask bit is associated with each class. The execution of the instruction causes a program interruption when the monitor-mask bit for the class specified in the instruction is one. The cause of the interruption is identified by setting bit 9 of the interruption code to one, and by the information placed at locations 148-149 and 156-159 of low address storage.

The monitor-mask bits are in bit positions 16-31 of control register 8.

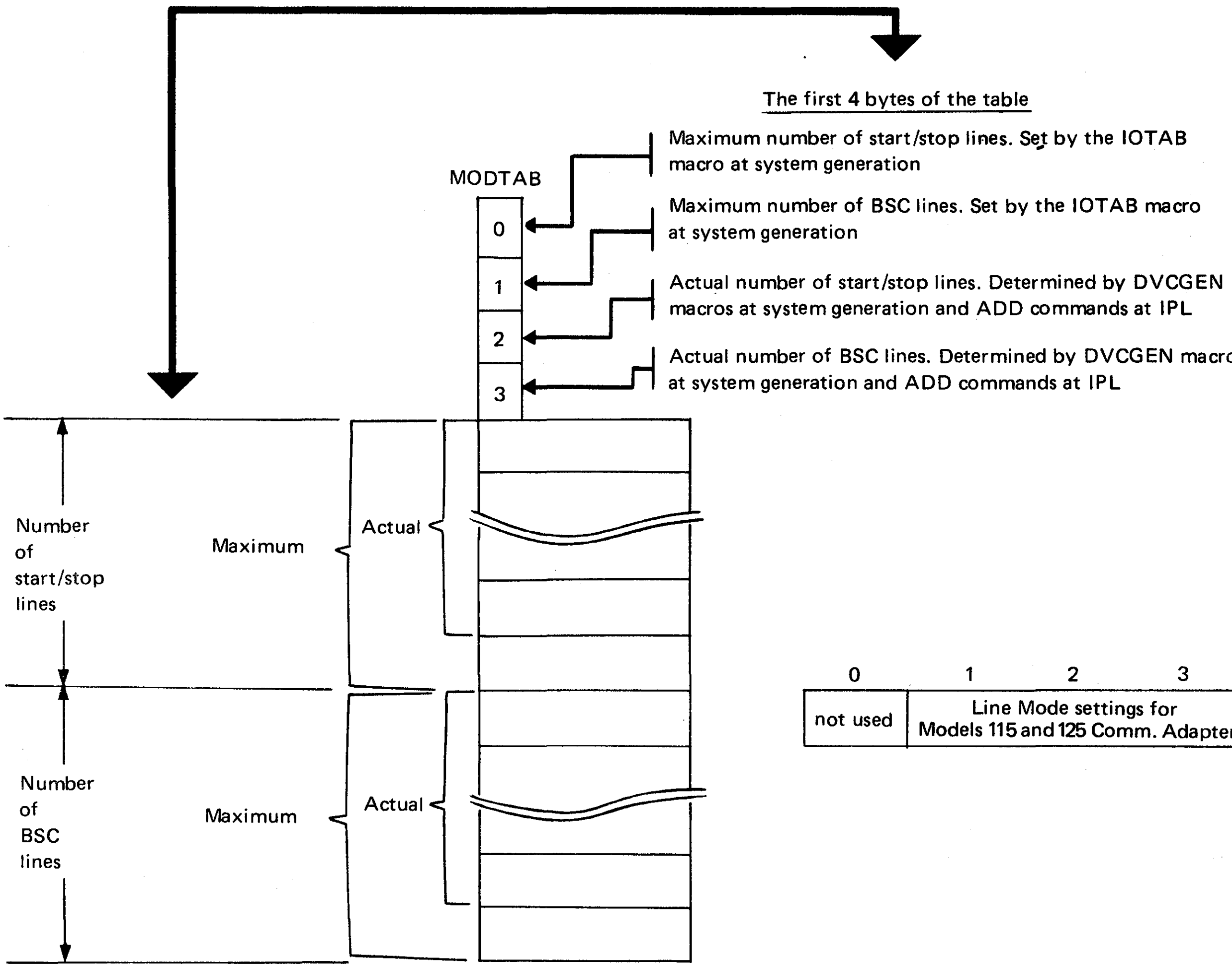


The mask bits, in ascending order of bit positions, correspond to monitor classes 0-15. Any number of monitor-mask bits may be on at any time; together they specify the classes of monitor events that are monitored at that time. The mask bits are initialized to zero.

Appendix F

LINE MODE TABLE

This table is built at supervisor generation time when the TP=BTAM, or QTAM parameter is included in the SUPVR macro, and MODEL=115 or 125. An entry is built for each device for which the DVGEN macro includes the MODE=X'ssss' or X'ssssss' parameter. Each entry contains the actual mode setting for the device.



0	1	2	3
not used	Line Mode settings for Models 115 and 125 Comm. Adapter		

Bytes 140-143 (X'8C'–X'8F') of the System Communication Region (SYSCOM) contain the address of the table.

Label MODTAB identifies the first byte of the table

Figure F.1. Line Mode Table (LMT)

EXAMPLE OF A STAND ALONE DUMP OUTPUT

This appendix shows an example of the output obtained from a formatted stand-alone dump as generated by the IBM program DUMPGEN with the parameters FORMAT=YES and PPOOL=NO.

Refer to Section 2-A-3 for a description of DUMPGEN and the stand-alone dump program.

In a system dump output, the supervisor area dumped is almost identical to that dumped by the formatted stand-alone dump, the only difference being that a system dump does not divide the dump into blocks of 2K storage areas. Refer to Section 2-A-2 for a description of the system dump.

The programmer's remarks on the dump example indicate how the various tables and information blocks are located by using addresses stored in the communication regions. The programmer has also indicated the meaning of several bytes on the dump, enabling a mental picture to be built up of the system status just before the dump program was executed.

Following the last but one 2K block of real storage (246 in the example shown), the page status information and contents of the control registers is printed. In the example shown of a formatted dump, the control registers are followed by the communication regions. (This does not include the system communication region.)

The remaining part of the example shows the order and format of the tables and information blocks printed in a formatted dump output.

The last block to be printed is the SELECTION POOL, the contents of which are explained in a note at the end of the example.

IMPORTANT NOTE
The location and addresses of the table and information blocks shown in this example apply only to the system that produced this example. The actual location of areas indicated depends on the system generation options specified, and the program running in your system just before the dump program is executed.

```
GR 0-1   00000000 00078000
GR 2-3   0000E2B0 900780D2
GR 4-5   0000000E 00000000
GR 6-7   0000E330 00000540
GR 8-9   00079800 0007A800
GR A-B   0007B800 0000E2A8
GR C-D   0000DE88 00000029
GR E-F   0007A935 00000000

EXT OLD 4B4B4B4B 4B4B4B4B
EXT NEW 00080000 0000A67E
EXT INT 00000000
SVC OLD 074D0000 00089B62
SVC NEW 040C0000 000008CC
SVC INT 00020007
PGM OLD 000C2000 0000A648
PGM NEW 000C0000 00079EC8
PGM INT 00040005
MCK OLD 00000000 00000000
MCK NEW 000C0000 0007B078
MCK INT 00000000 00000000
I/O OLD 070F2000 0000090C
I/O NEW 00080000 0000A67E
I/O INT 6000000C
CSW     0007B038 08000000
CAW     0007B030
TIMER   F4E3C400

BLOCK 246

07B000  50B01028 92021028 41F01028 50F00048 9C002000 47701010 9D002000 47701018
07B020  41BB0048 060107FE 0207AC88 20000048 0907B0D8 20000078 000A0000 00000000
07B040  5C5C5C40 C5D6D140 5C5C5C00 928B1030 9D004000 47701050 9C004000 9D004000
07B060  4730105C 924010D8 027610D9 10D89209 10309278 103707F3 92C10000 58100010
07B080  48201182 48401184 41301030 50300048 92111030 45301050 D22610D8 11BE4530
```

General registers
(The control registers are printed at end of dump of real storage, before formatted output.)

Stand-alone dump
FORMAT = YES
(3 partitions active)
Wait State during output to 3215 PR-KB

Last 2K block main storage
(That is the physical CPU storage)

(Ex 100 KT)

Figure G.1 part 1 of 20

Appendix G

EXAMPLE OF A STAND ALONE DUMP OUTPUT

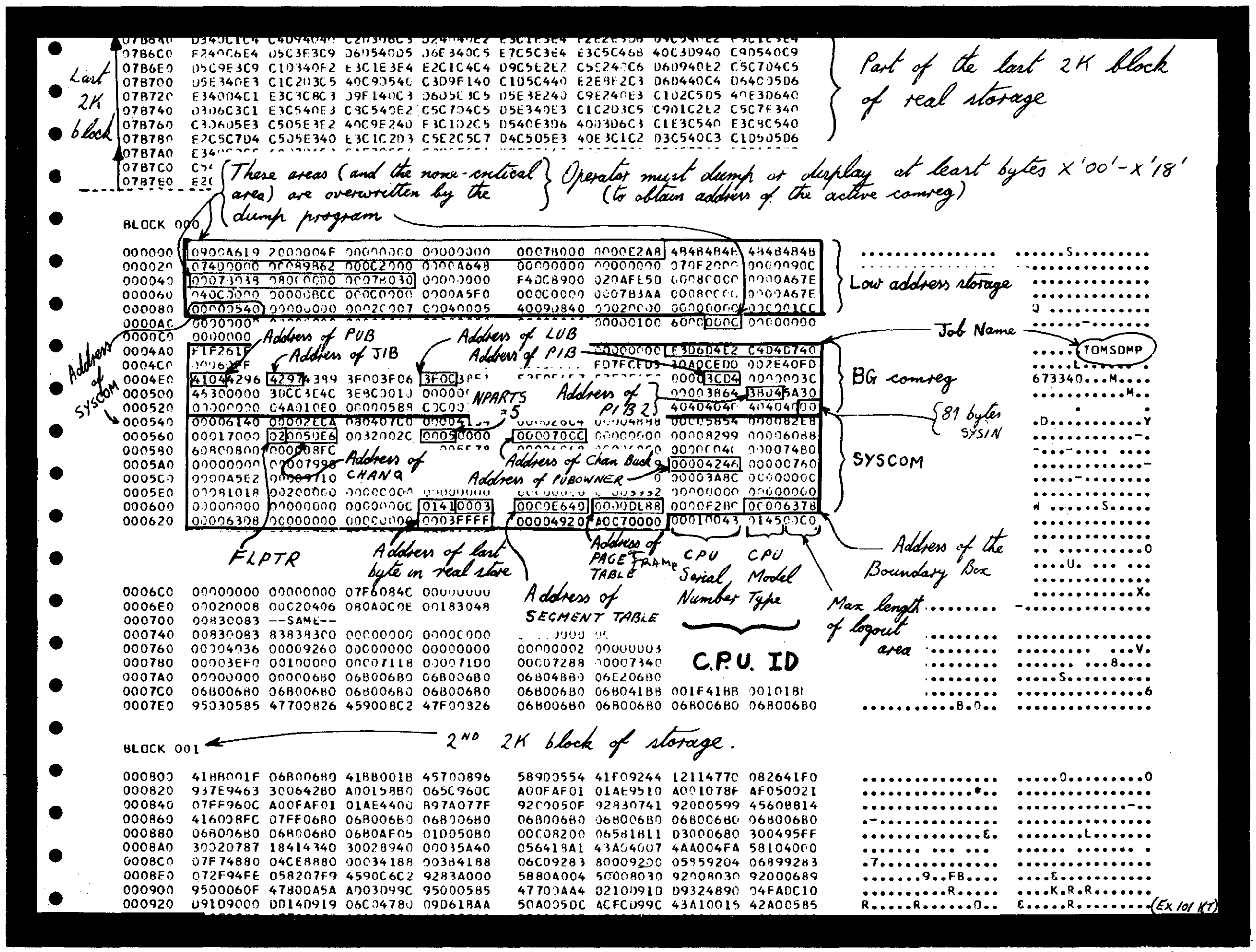


Figure G.1 part 2 of 20

```
003B40   1A0038D- ...                                 80000001 0800E830 80000001   ...0&........J8U  C..&.......Y.-...
003B60   ...6                                         60000004 0800E7E8 00000A01   ....-.......-....
003B80   ...0                                         C2C4E4D4 D7401700 000070A0   ................
003BA0   ...1                                         00000000 00000000 00000000   K......&...8....
003BC0   0000000                                      04A00000 00020008 00000000   ....
003BE0   00000000 04A00000 00020007 00000000          00100000 45900036 00020007 00000000   ....
003C00   00200000 44F00062 00000000 00000000          00300000 4450008E 00020007 00000000   ....  ......&.........
003C20   00400000 43B000BA 00000000 00000000          00500000 00000000 00000000 00000000   ....  ..............
003C40   00000000 --SAME--
003CC0   00000000 00000000 00000000 00000000          00000000 8000C1D9 00000000 000063E8   ....  .....AR......Y
003CE0   000E2880 8200C2C7 80040000 00006458          000E2880 8200C6F4 80061000 00006518   .....BG........  .....F4........
003D00   80441E80 8000C6F3 80007800 000065D8          00701E00 8300C6F2 80089000 00006698   .....F3......Q  .....F2........
003D20   009C1E80 8000C6F1 80008800 00006758          00C81E00 80000000 00000000 00006818   .....F1........  .H.............
003D40   00000000 80000000 00000000 000068D8          00000000 80000000 00000000 00006998
003D60   00000000 80000000 00000000 00006A58          00000000 80000000 00000000 00006B18
003D80   00000000 80000000 00000000 00006BD8          00000000 80000000 00000000 00006C98
003DA0   00000000 80000000 00000000 00006D58          00000000 80000000 00000000 00006E18
003DC0   00000000 80000000 00000000 00006ED8          00000000 00000000 00000000 00000000
003DE0   00000000 --SAME--
003F00   000E4470 9CC80E28 1E1E1E1E 01FF01FF          02FF03FF 00FF0702 06FF06FF 0604FFFF
003F20   07010600 07FFFFFF 07FF07FF 07FF1303          0BFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
003F40   FFFFFFFF --SAME--
003F60   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF          FFFFFFFF FFFFFFFF 0FFF0FFF 12FF0EFF
003F80   00FF07FF 06FF ...              07FFFFFF      07FFFFFF FFFF07FF 07FF07FF FFFFFFFF
003FA0   FFFFFFFF --S...
003FC0   FFFFFFFF FFFI ...              FFFFFFFF      FFFFFFFF FFFFFFFF 00FFFFFF 06FF08FF   ................  ..  ..............
003FE0   0BFFFFFF 07FI ...              FFFFFFFF      FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

BLOCK 008

004000   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF          FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
004020   FFFFFFFF FFFFFFFF 10FF10FF 14FF15FF          00FF07FF 06FF0606 0BFFFFFF 07FFFFFF   ................  ................
004040   07FFFFFF 07FF0708 07090707 07FFFFFF          FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   ................  ................
004060   FFFFFFFF --SAME--                                                                  ................  ................
004080   FFFFFFFF FFFFFFFF 00FFFFFF                   0BFFFFFF 07FFFFFF 07FFFFFF FFFFFFFF
0040A0   FFFFFFFF --SAME--
0040C0   FFFFFFFF FFFFFFFF FFFFFFFF                   FFFFFFFF FFFFFFFF 0130FF00 620000F8
0040E0   40D80000 0009FF00 000000F8 40E40000          0130FF00 620002F8 40F00000 000050B
004100   12FFFFFF 00090400 000080F8 000CFF00          110000F8 000DFF00 210000F8 000EFF00
004120   4200                                             0FF00 620002F8 0131FF00
004140   6201                                             4FF00 620402F8 0260FF00
004160   6305                                             1FF00 50C300C0 02A2FF00
004180   50C3                                           C0 0380FF00 50C302C0 0381FF00
0041A0   50C3                                           C0 FF000000 00000000 FF000000
0041C0   00000000 FF000000 00000000 FF000000          00000000 FF000000 00000000 FF000000
0041E0   --SAME--
004240   00000000 FF000000 00010001 00010000          00000008 001F0000 00000000 001F0000
004260   00000002 00020008 00000002 00010008          00080000 00000000 00000000 00000000
004280   00000000 00000000 00000000 00000000          00000000 0000052F 4828FFC6 C628FF01
0042A0   1428FF0B FF80FF49 6628FF00 00000A67          C428FF15 9C28FF15 9C28FF15 9C28FF00
0042C0   00000B00 00000C00 00000D00 00000E00          00000F00 00001000 00001100 00001200
0042E0   00001300 00001400 00001500 00001600          00001700 00001800 00001900 00001A00
004300   00001B00 00001C00 00001D00 00001E00          00001F00 00002000 00002100 00002200
004320   00002300 00002400 00002500 00002600          00002700 00002800 00002900 00002A00
004340   00002B00 00002C00 00002D00 00002E00          00002F00 00003000 00003100 00003200
004360   00003300 00003400 00003500 00003600          00003700 00003800 00003900 00003A00
004380   00003B00 0000FF00 06FF0000 000000FF          00000000 00FF0000 000000FF 00000000   ..........        ..............
0043A0   00FF0000 000000FF 00000000 00000000          F1F261F0 F661F7F3 70007000 00000000   ..........        12/06/73........
0043C0   00000000 00000000 D5D640D5 C1D4C540          00000000 00000000 00000000 00000050   NO NAME           ...............&
0043E0   000BFFFF FD7FCED3 0000CE50 002E40FD          41044296 42974389 3F003F06 3F0C38F1   L...&.. .         ...............1
004400   F2F0F6F7 F3F3F4F0 00003CD4 0000003C          48100000 3DCC3E4C 3E8C0010 00000050   O...M....         ...............&
004420   0000806C 000073F8 00003864 38D45A30          00000000 04A010E0 00000588 C0C00040   8.....M..         ................
004440   40404040 40404000 40404040 40404000          F1F261F0 F661F7F3 70007000 00000000   .                 12/06/73........
004460   00000000 00000000 C3C1D3C3 E2C9D440          000A1FFF 0008AFF6 0008D4DF 00000040   CALCSIM           ........6..M....
004480   000BFFFF FD7FCED3 30804ED0 002E40FD          41044296 42974389 3F003F06 3F0C38F1   L..+... .         ...............1
0044A0   F2F0F6F7 F3F3F4F0 00003CD4 0000003C          47980000 3DCC3E4C 3E8C0010 00000040   O...M....         ................
0044C0   00008075 00007340 00003864 38D45A30          00000000 04A010E0 00000588 C0C00040   ......M..         ..............
0044E0   40404040 40404000 40404040 40404000          F1F261F0 F661F7F3 70007000 00000000   .                 12/06/73........
004500   00000000 00000000 D5D640D5 C1D4C540          00000000 00000000 00000000 00000030   NO NAME           ................
004520   000BFFFF FD7FCED3 0000CE50 002E40FD          41044296 42974389 3F003F06 3F0C38F1   L...&.. .         ...............1
004540   F2F0F6F7 F3F3F4F0 00003CD4 0000003C          47200000 3DCC3E4C 3E8C0010 00000030   O...M....         ................
004560   0000807E 00007288 00003864 38D45A30          00000000 04A010E0 00000588 C0C00040   ......M..         ..............
004580   40404040 40404000 40404040 40404000          F1F261F0 F661F7F3 70007000 00000000   .                 12/06/73........
0045A0   00000000 00000000 D2C5D5E2 D3D6D6D7          00074FFF 00063C5B 00000000 00000020   KENSLOOP          .......$........
0045C0   000BFFFF FD7FCED3 3000CED2 002E40FD          41044296 42974389 3F003F06 3F0C38F1   L...K.. .         ...............1
0045E0   F2F0F6F7 F3F3F4F0 00003CD4 0000003C          46A80000 3DCC3E4C 3E8C0010 00000020   O...M....         ................
004600   00008087 000071D0 00003864 38D45A30          00000000 04A010E0 00000588 C0C00040   ......M..         ..............
004620   40404040 40404000 40404040 40404000          00000000 00000000 2F000000 00000000   .       .         ................
004640   00000000 00000000 00000000 00000000          00500000 00000000 00000019 00000000   .........         .&..............
004660   00000000 00000000 00510000 00000000          00000019 03E80000 00000000 00000000   .........         .....Y..........
004680   00790000 00000000 00000013 03E80000          00000000 00000000 00500000 00000000   ............Y..   .........&......
0046A0   00000019 00000000 00000000 000000FF          FF000000 00000000 00000000 00000000   ...............   ................
0046C0   00000000 00000000 00500000 00000000          00000019 00000000 00000000 00000000   .........&......  ................
0046E0   00510000 00000000 00000019 03E80000          00000000 00000000 00790000 00000000   ............Y..   ................
004700   00000013 03E80000 00000000 00000000          00500000 00000000 00000019 00000000   .....Y.........   .&..............
004720   00000000 000000FF FF000000 00000000          00000000 00000000 00000000 00000000   ...............   ................
004740   00500000 00000000 00000019 00000000          00000000 00000000 00510000 00000000   .&.............   ................
004760   00000019 03E80000 00000000 00000000          00790000 00000000 00000013 03E80000   .....Y.........   .............Y..
004780   00000000 00000000 00500000 00000000          00000019 00000000 00000000 000000FF   .........&.....   ................
0047A0   FF000000 00000000 00000000 00000000          00000000 00000000 00500000 00000000   ...............   .........&......
0047C0   00000019 00000000 00000000 00000000          00510000 00000000 00000019 03E80000   ...............   .............Y..
0047E0   00000000 00000000 00790000 00000000          00000013 03E80000 00000000 00000000   ...............   .....Y..........

BLOCK 009

004800   00500000 00000000 00000019 00000000          00000000 000000FF FF000000 00000000   .&.............   ................
```

- Address of BG comreg
- Entry in PIB2 belonging to BG
- Recognize the PIB by these
- PIB2
- PIB
- AP Subtask PIBs
- NICL
- FICL
- No 11 in LUB points to No 13 in PUB
- No 5 in LUB points to No 2 in JIB
- " 8 — " — 4 "
- " A — " — 1 "
- " B — " — 0 "
- " 11 — " — 3 "
- Non standard assignments
- LUB
- Device Busy
- FOCL
- Chan 0 points to PUB 0
- " 1 " " " 5
- " 2 " " " B
- " 3 " " " 12
- " 4, 5 and 6 are not supported
- FOCL
- Device at address 0009
- Device type = Printer Keyboard
- Queued to CHANQ entry 4
- PUB
- Start of PUB OWNER
- No 13 in PUB, I/O device address is 381
- FAVP, next free entry in JIB = 5
- JIB
- Job names
- F1 comreg
- F2 comreg
- F3 comreg
- F4 comreg
- (Ex 102 Kt)

Figure G.1 part 3 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
005D60  E5164620 E39247F0 E4729104 E23A4710  E54C9180 E23A4710 E53247F0 E4EA9120  V...T..OU...S...  V...S...V..OU...
005D80  00454780 E4724B60 06E25060 00484166  0008D300 0048E215 96407000 47F0E386  ....U..-.S&-....  ..L...S.. ...OT.
005DA0  96207003 92036000 47F0E386 D207E23C  00400203 0048E230 58190050 9C001000  ......-..OT.K.S.  . K...S....&....
005DC0  9D001000 4720E56C 4710E472 91020044  4710E472 50600048 D3000048 E215D207  ......V...U.....  ..U.&-..I...S.K.
005DE0  0040E23C 07F30100 76C8203A 04FF0800  7660203A 04FF0300 E7A04095 07FF0500
005E00  7798203A 04FF0700 E9081004 04100000  7730203A 04FF0B00 39A0033C 06FFFF00
005E20  E680203A 04200A00 7938203A 04FF0600  7868203A 04FF0900 78D0203A 04FF0C00
005E40  0000FF00 FFFF0D00 0000FF00 FFFF0E00  0000FF00 FFFF0F00 0000FF00 FFFF1000
005E60  0000FF00                                      FF1300 0000FF00 FFFF1400
005E80  0000FF00                                      FF1700 0000FF00 FFFF1800
005EA0  0000FF00                                      FF1B00 0000FF00 FFFF1C00
005EC0  0000FF00                                      FF1F00 0000FF00 FFFF2000
005EE0  0000FF00                                     FFFF2300 0000FF00 FFFF2400
005F00  0000FF00                                     FFFF2700 0000FF00 FFFF2800
005F20  0000FF00                                     FFFF2B00 0000FF00 FFFF2C00
005F40  0000FF00                                      FF2F00 0000FF00 FFFF3000
005F60  0000FF00 FFFF3100 0000FF00 FFFFFF00  0000FF00 FFFF0014 01000000 00000000  ................  ................
005F80  00000000 02000000 00000000 00000000  03000000 00000000 00000000 04000000  ................  ................
005FA0  00000000 00000000 05000000 00000000  00000000 06000000 00000000 00000000  ................  ................
005FC0  07000000 00000000 00000000 08000000  00000000 00000000 09000000 00000000  ................  ................
005FE0  00000000 0A000000 00000000 00000000  0B000000 00000000 00000000 0C000000  ................  ................

BLOCK 012

006000  00000000 00000000 0D000000 00000000  00000000 0E000000 00000000 00000000  ................  ................
006020  0F000000 00000000 00000000 10000000  00000000 00000000 11000000 00000000  ................  ................
006040                    000000 00000000  13000000 00000000 00000000 FF000000  ................  ................
006060                    FF0000 0000FF00                      000000 00FF0000  ................  ................
006080                    000000 FF000000                      0000FF 00000000  ................  ................
0060A0                    00FF00 000000FF                      000000 20002000  ................  ................
0060C0                    000000 00000000                      000000 00000000  ................  ................
0060E0                                                                           ....
006140  5B5BC1C2 C5D9D9C1 4DC24A4C 08062E6C  00000000 00000000 00000000 00000000  $$ABERRA B......  ................
006160  00000000 --SAME--
006300  00000000 00000000 1000E600 0A000000  410C0400 00000000 0000E7E8 10000000                    ........XY....
006320  00000000 00000000 00000000 00000000  00000000 4000E9C0 0E000005 00000400                    .. .Z.........
006340  007A0013 00000000 00200040 00000000  00000000 00000000 00000000 00000000                    ..............
006360  00000000 00000000 00000000 00000000  00000000 00000000 0003D800 00050026                    ........Q.....
006380  00040000 000C0000 00017000 0001E800  00040000 00061000 0001E800 00021000                    ........Y.....
0063A0  00061000 00075000 00021000 00023800  00075000 00089000 00023800 00027000                    &.............
0063C0  00089000 000A2000 00027000 0002A800  000A2000 000C0000 01310000 00000E10                    ..............
0063E0  00000E74 0100F1F1 F1F1F1F1 0000C95C  07001000 00008460 000085D8 000084CA  ......111111..1-  .......-...Q....
006400  000086B8 000084BA 00008534 8000837A  000082E8 00008318 00008308 00089D40  ................  ...Y..........
006420  00089000 FFFF7D18 00008340 00008660  00004450 00000000 00000000 00000000  ........... ...-  ...&............
006440  00000000 --SAME--                                                          ....
006460  040C0000 0000AC8E 00006458 00003CE4  00001000 00002000 00003000 0000A5E2  ..............U  ..............S
```

FLPTR pointed to 02, which is chained to 03 → 05 → 00 → 01 → 08 → 0A → 09 → 06 → 0B → 0C - - - > FF

CHANQ

PUB entry 00 was queued to CHANQ entry 04

CCB address = E908 chained to CHANQ entry 07

Phase name of ERP

Start of ERRQ

Error Recovery Block

Boundary Box

(EX 103 KT)

```
BLOCK 014

007000  00000000 00000000 00000000 00004104  FF00E908 00000009 00004104 03005E06  ................  ..Z.............
007020  00003CE4 0000412C 00000000 00000130  00004134 02005DF6 00003D14 0000415C  ..U.............  ........6.......*
007040  00000000 000002A2 0                         000000 00000381  ................  ....4...........
007060  0000419C 04005E06 0                         004104 00000000  ...........U....  ................
007080  00000000 00004104 0                 0004104 00000000 00000000 00004104  ................  ................
0070A0  00000000 00000000 00004104 00000000  00000000 00000000 07023D1C 60000006  ................  ..............-...
0070C0  1F0070C5 40100001 0800E6D0 00000000  07007000 40000006 1F0070DD 40000001  ...E .....W.....  .... ...... ....
```

Address of PIB

Address of 1ST PUB on Channel 0

CCB

PUB pointer

Channel Bucket

Channel Bucket for Channel Ø

(EX 104 KT)

```
0093C0  F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C3C1  D5C3C5D3 C9C7D5D6 D9C54040 40404040  23456789ABCDEFCA  NCELIGNORE
0093E0  40404000 C10940                     00000 00000000                      ..AR ........   ...............
009400  00000000 --SAME-                                                          ....
009480  00000000 00000000 000099FF 0000479C  0000168E 0000B44E 00000C56 00009F02  ................  ........+.......
0094A0  00000000 00004798 00003AB4 00001000  D7C4C1C9 C4C7E3E6 FFFF0009 FFFFFFFF  ................  PDAIDGTW........
0094C0  FFFFFFFF 0003D800 0003FFF0 3FDE       FF FFFFFFFF FFFFFFFF 0003FFF0  ......Q....0....  ...............0
0094E0  FFFFF                                20 42C0910B 58C00080 9500C045  ................  ................
009500  4770                                 0B 078A95FF 9108078A 95FF910B  ....                           ...
009520  078A                                 FF 60004780 90CAD500 8000008B  ....                           ...
009540  4780                                 70 9118D207 70000020 90017008  ....                           ...
009560  D2007                                01 70108002 41770012 5970911C  K....                          ...
009580  4740                                 10 00000001 00000000 0003D800  . ...                          .Q.
0095A0  0003E                                20 90839049 D2009089 904AD200  .....                          .K.
0095C0  908F                                 F0 909F47F0 917090CA 9180905B  ....K                          ..&
0095E0  47109                                5D 928090C7 95009070 4770917E  ....K                          ...
009600  D2019     91DA9057 912095FF 903C     50 90305850 90400203 90549040  K..............  ...#...-. K....
009620  18651876 18665060 910C4C70 910E       75 50509114 50509118 5070911C  .......-.......  ....##..##..#...
009640  98579120 47F091CC 41509120 5869                            00047F0  .....0...#......  .0......P......0
009660  906CA000 00000000 00000000 0000                                    0000  ................  ................
009680  00000000 --SAME--                                                  ....
0097E0  00000000 00000000 00000000 0000       00 00000000 00000000 00000000  ................  ................
```

START OF PD area

End Address of PD area

Name of active PDAID

Address of Start of Alternate Area

Address of End of Alternate Area

Start of Standard Preface Table

PDAID GSVC trace active in core-wrap output

Part of PD area

(EX 105 KT)

Figure G.1 part 4 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
00DE20  00045070 F6E447F0 FCFE9200 F7489200   058594DF 058247F0 08FC9529 F7484770
00DE40  FD109200 F7485870 F6E407F7 98         07E78000 00000000 00000000 00000000
00DE60  00000000 --SAME--
00DE80  00000000 00000000 00020000 00000000   00020000 00010000 00020000 00020000
00DEA0  00020000 00030000 00020000 00040000   00020000 00050000 00020000 00060000
00DEC0  00020000 00070000 00020000 00080000   00020000 00090000 00020000 000A0000
00DEE0  00020000 000B0000 00020000 000C0000   00020000 000D0000 00020000 000E0000
00DF00  00020000 000F0000 00020000 00100000   00020000 00110000 00020000 00120000
00DF20  00020000 00130000 00020000 00140000   00020000 00150000 00020000 00160000
00DF40  00020000 00170000 00020000 00180000   00020000 00190000 00020000 001A0000
00DF60  00020000 001B0000 00020000 001C0000   00020000 001D0000 00020000 001E0000
00DF80  00020000 001F0000 00020000 00200000   00020000 00210000 00020000 00220000
00DFA0  00020000 00230000 00020000 00240000   00020000 00250000 00020000 00260000
00DFC0  00020000 00270000 00020000 00280000   00020000 00290000 00020000 002A0000
00DFE0  00020000 002B0000 00020000 002C0000   00020000 002D0000 000001E0 FFFF0400

BLOCK 028

00E000  002201F8 00820268 00220260 00830410   00000208 00840240 000001F0 013B02A0
00E020  000002A0 00810268 00000210 FFFF01E0   00000250 00C501C0 000001E8 01360218
00E040  000001C0 00C70418 000001A8 00C601B8   002202A0 00C20190 00000418 00C301D8
00E060  000001D0 00C40208 000001A0 FFFF0170   00000268 013701B0 000001F8 01380190
00E080  00000200 013901F0 00000228 013A01F8   000001D8 00800188 000002B8 FFFF01A0
00E0A0  00000180 013102B0 00000248 01190298   00000290 01120200 00000258 01150250
00E0C0  00000410 01130290 00000188 01170248   00000240 01180220 00000230 011401A8
00E0E0  00000298 01160230 00000270 013F0410   00000198 013501E8 00000278 013E0260
00E100  00000280 013D0270 00000288 01410278   000002A8 01400280 00000238 013C0228
00E120  00000220 011A0258 00000190 00C80198   000002B0 01420288 00000218 014302A8
00E140  000002C0 FFFF0210 000002C8 FFFF02B8   000002D0 FFFF02C0 000002D8 FFFF02C8
00E160  000002E0 FFFF02D0 000002E8 FFFF02D8   000002F0 FFFF02E0 000002F8 FFFF02E8
00E180  00000300 FFFF(                                       00 00000318 FFFF0308
00E1A0  00000320 FFFF(                                       20 00000338 FFFF0328
00E1C0  00000340 FFFF(                                       40 00000358 FFFF0348
00E1E0  00000360 FFFF(                                       60 00000378 FFFF0368
00E200  00000380 FFFF(                                       80 00000398 FFFF0388
00E220  000003A0 FFFF0390 000003A8 FFFF0398   000003B0 FFFF03A0 000003B8 FFFF03A8
00E240  000003C0 FFFF03B0 000003C8 FFFF03B8   000003D0 FFFF03C0 00000400 FFFF03C8
00E260  000303E0 007B03D0 000303E8 007C03D0   000303F0 007D03D0 000303F8 007E03D0
00E280  00030400 007F03D0 00000170 000003D0   00000408 00000408 00000260 00000238
00E2A0  000001B8 000001D0 00000420 00000420   00000000 00000000 00000000 00000000
00E2C0  00000000 --SAME--
00E320  00000000 00000000 00000000 00000000   00000008 00100018 00200028 00300038
00E340  00400048 00500058 00600068 00700078   00800088 00900098 00A000A8 00B000B8
00E360  00C000C8 00D000D8 00E000E8 00F000F8   01000108 01100118 01200128 01300138
00E380  01400148 01500158 01600168 80118011   80118011 80118011 80118011 80118011
00E3A0  80118011 80118011 80118021 80218021
00E3C0                                 80518051
00E3E0
00E420  80018001 800103D9 03E103E9 03F103F9   02090199 01790181 01890015 00150015
00E440  00150015 --SAME--
00E4A0  00150015 00150015 00150015 00150015   00150015 01C901D1 01D901A9 01C101B9
00E4C0  02A10025 00250025 00250025 00250025   00250025 00250025 00250025 00250025
00E4E0  00250025 --SAME--
00E500  00250025 00350035 00350035 00350035   00350035 00350035 00350035 00350035
00E520  00350035 --SAME--
00E540  00350035                         0035   00350035 02290239 02510231 02590241
00E560  02490221                         0045
00E580  00450045                         0045
00E5A0  01F101F9                         0261
00E5C0  00550055
00E620  00550055 00550055 00550055 00550055   00000000 00000000 00000000 00000000
00E640  F000E330 F000E370 F000E3B0 F000E3F0   F000E430 F000E470 F000E4B0 F000E4F0
00E660  F000E530 F000E570 F000E5B0 F000E5F0   00000001 00000001 00000001 00000001
00E680  00000000 00002004 0000E830 00061518   00000000 00000000 0020C000 000614E8
00E6A0  0000E830 00000000 00000000 00000000   00000000 00000000 00000000 00000000
00E6C0  00000000 0000E908 07023D1C 60000006   31023D1E 60000005 0800E6D0 00000000
00E6E0  06023D50 600000A0 92023D38 20000005   00000000 00000000 00000000 00000000
00E700  80000000 00089CE0 88000000 0000E998   09018074 60000020 09018094 20000025
00E720  00000000 --SAME--
00E740  00000000 00000000 80000000 00041770   88C00000
00E760  00000000
00E780  00000000
00E7A0  00008000
00E7C0  0000E6C8
00E7E0  00000000 0000E6C8 09017F88 6000001C   09017FA4 60000020 09017FC4 6000002F

BLOCK 029

00E800  09                            12 60000021   09018033 60000021 09018054 60000020
00E820  88                            00 0000E710   0901C96E 6000000F 0101C8EE 60000003
00E840  0901C9BE 2000001C 00000000 00000000   00000000 00000000 00000000 00000000
00E860  00000000 00000000 80000000 00061500   88000000 00000000 1500E978 3000005A
00E880  00000000 --SAME--
00E8A0  00000000 00000000 00000000                              000E8C0
00E8C0  00008000 0C002003 0000E758                              0089D30
00E8E0  0000E758 00000000 00000000 00000000   00000000 00000000 00000000 00000000
00E900  00000000 0000E758 00000400 00002004   0000E7E8 00041780 00000000 00000000
00E920  0010C000 00041712 0000E7E8 0000E950   00000000 00000000 00000000 00000000
00E940  00000000 00000000 00000000 00000000   00017FF3 00018000 FF000000 00000000
00E960  00000000
00E980  00000000
00E9A0  31022898
```

Page Frame 0

PAGE 1

PAGE FRAME TABLE

X'2D' = 45

Last page in SUPVR

FFFF in bytes 4 and 5 indicate frame is unused

page frame at address E018 was occupied by page 315 (X'13B' = 315 dec)

All page frames unused

5 frames used by PDAID alternate area

5 Selection Pool Queues

Pages belonging to the supervisor

PAGE TABLE

Real addresses
20800 19800 17800 18000 18800

5 pages belonging to Segment 4 occupied by BGV in following order:
Block 53 at real 20800
– " – 51 – " – 19800
– " – 47 – " – 17800
– " – 48 – " – 18000
– " – 49 – " – 18800

Address of Page Table

Address of page table belonging to SEGMENT 4

SEG Table

SEGMENT TABLE

2ND CCW COPY BLOCK

Write Command

I/O area address

Address of next CCW COPY BLOCK for this I/O operation

chain bit set

COUNT

2ND CCW

3RD CCW in String

A CCW COPY BLOCK (1ST in a string of 2)

CCB pointer in CHANQ entry 4 pointed to E908

Pointer to first CCW (Channel Program)

Address of CCW COPY BLOCK

CCW address in CSW

A CCB COPY BLOCK

IDAL BLOCK

Address of Virtual CCB

Address of IDAL block

(Ex 106 KT)

Figure G.1 part 5 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
BLOCK 047

017800  4040D440 40404040 C1D4E240 40404040   40404040 40404040 4040C340 7E40D4C9    M     AMS             C . MI
017820  D5C4C5D9 40C2C5D3 C1D5C7D9 C9D1D248   40E5C5D9 C4C5D9C5 40D6D5E3 E6C9D2D2    NDER BELANGRIJK.  VERDERE ONTWIKK
017840  C5D3C9D5 C740E5C1 D540E2C9 E3E4C1E3   C9C540D4 D6C5E340 C1C6C7C5 E6C1C3C8    ELING VAN SITUAT  IE MOET AFGEWACH
017860  E340                                       )40 D5C9C5E3 40E5C5D9 D6D5E3D9    T WORDEN  D . .N  OG. NIET VERONTR
017880  E4E2                                       5D6 D9C4E340 C7C5C5D5 40C1D2E3    USTEND. VOORLOPI  G WORDT GEEN AKT
0178A0  C9C5                                       )40 40404040 40404040 40404040    IE ONDERNOMEN
0178C0  4040
017960  4040                                       :40 40404040 40404040 40404040      ** AFKORTINGEN   **
017980  4040
017A20  4040                                       )4D E5D6D6D9 D3D6D7C9 C75D40C1                       * .VOORLOPIG. A
017A40  C6C7                                       'C5 D5E3C9C5 4040D7D3 E3407E40    FGEWERKT  UGN .   URGENTIE  PLT .
017A60  D7D3                                       ,D9 C7C5D5E3 C9C5C3D6 C4C57A40    PLAATS              URGENTIECODE.
017A80  4040                                       )40 40404040 4040C140 7E40C8D6                              A . HO
017AA0  D6C7                                       ,C5 D3D3C9D1 D2C540C1 D2E3C9C5    OGST URGENT. ONM  IDDELLIJKE AKTIE
017AC0  40E5                                       )40 40404040 40404040 40404040     VEREIST
017AE0  4040                                       5C7 D9C9D1D2 4B40C1D2 E3C9C540               B . BE  LANGRIJK. AKTIE
017B00  C9D5                                       )40 40404040 40404040 40404040    IN NABIJE TOEKOM  ST
017B20  4040                                       )40 40404040 40000000 00000000                              .......
017B40  0000                                                                         ....
017E60  0000                                       )00 0000005C 00000000 0000000C    ...............   .......*.......
017E80  0000                                       )00 0000012C 8004033C 00000000    ...............   ...............
017EA0  0000                                                                         ....
017EE0  0000                                       )F6 404040F1 404040F5 404040F0    ........TOMS  12     6   1   5   0
017F00  4040                                       )00 04000000 00040004 17380004      2I  33AMSTERDA  M .............
017F20  1780                                       )00 00000000 09041788 6000001C    ...............   ...........-...
017F40  0904                                       'F3 6000001F 09041812 60000021    ....-......D-...  ...3-.......-...
017F60  0904.                                      874 60000020 09041894 20000025    ....-.......-...  ....-...........
017F80  0A0418B9 20000001 C2E4C740 C7C5D5C5   D9C1E3D6 D940D9D6 E4E3C9D5 C540C1C3    .......?BUG GENE  RATOR ROUTINE AC
017FA0  E3C9E5C5 40404040 E2C5D3C5 C3E340C1   40C2E4C7 40C2E840 C5D5E3C5 D9C9D5C7    TIVE     SELECT A  BUG BY ENTERING
017FC0  40D6D5C5 40404040 D6C640E3 C8C540C6   D6D3D3D6 E6C9D5C7 40D3C5E3 E3C5D9E2     ONE     OF THE F  OLLOWING LETTERS
017FE0  40C6D6D3 D3D6E6C5 C440C2E8 40C5D6C2   4B404040 404040D3 404040C5 D5E3C5D9     FOLLOWED BY EOB   .        L  .ENTER

BLOCK 048

018000  40C1D540 E4D5C5D5 C4C9D5C7 40D3D6D6   D7504040 4040C940 4040D7D9 D6C4E4C3     AN UNENDING LOO  P.     I  .PRODUC
```

PUB 00 device address 0009 device type 00 = PR-KB, was busy and queued to CHANQ entry 04.

Address of CCB COPY BLOCK (from CCB address in CHANQ) = E908

Address of 1ST CCW COPY BLOCK (from CCB COPY BLOCK) = E7E8

Address of I/O area (from 1ST CCW) = 17F88

Start of I/O area

1ST block of data to be transferred by 1ST CCW

(EX107KT)

```
03FDA0  00000008 9B620008 9A7C0708 9D24C6F2   074D2000 00089B58 00089A7C 00089D24    ..............F2  ...............
03FDC0  C6F2074D 00000008 9B620008 9A7C0708   9D24C6F2 074D2000 00089B58 00089A7C    F2..............  .F2.............
03FDE0  00089D24 C6F2074D 00000008 9B620008   9A7C0708 9D24C6F2 074D2000 00089B58    ....F2..........  .....F2.........
03FE00  00089A7C 00089D24 C6F2074D 00000008                                          ........F2......  ..........F2....
03FE20  00089B58 00089A7C 00089D24 C6F2074D                                          ............F2..  ..............F2
03FE40  074D2000 00089B58 00089A7C 00089D24                                          ................  F2..............
03FE60  9D24C6F2 074D2000 00089B58 00089A7C                                          ..F2............  ....F2..........
03FE80  9A7C0708 9D24C6F2 074D2000 00089B58                                          ......F2........  ........F2......
03FEA0  9B620008 9A7C0708 9D24C6F2 074D2000                                          ..........F2....  ............F2..
03FEC0  00000008 9B620008 9A7C0708 9D24C6F2                                          ..............F2  ................
03FEE0  C6F2074D 00000008 9B620008 9A7C0708                                          F2..............
03FF00  00089D24 C6F2074D 00000008 9B620008                                          ....F2..........
03FF20  00089A7C 00089D24 C6F2074D 00000008                                          ........F2......
03FF40  00089B58 00089A7C 00089D24 C6F2074D                                          ............F2..
03FF60  074D2000 00089B58 00089A7C 00089D24                                          ................
03FF80  9D24C6F2 074D2000 00089B58 00089A7C   00089D24 C6F2074D 00000008 9B620008    ..F2............
03FFA0  9A7C0708 9D24C6F2 074D2000 00089B58   00089A7C 00089D24 C6F2074D 00000008    ......F2........
03FFC0  9B620008 9A7C0708 9D24C6F2 074D2000   00089B58 00089A7C 00089D24 C6F20000    ..........F2....  ............F2..
03FFE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000    ................  ................

BLOCK 128

040000  00000000 00000000 0000                                          00 00000000    ...
040020  --SAME--
0407E0  00000000 00000000 0000                                          00 00000000    ...

BLOCK 129

040800  00000000 00000000 0000                                          00 00000000    ...
040820  --SAME--
040FE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000    ................  ................

BLOCK 130

041000  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000    ................  ................
041020  --SAME--
```

Last part of real storage in this example taken from the main page pool to be used an an alternate area by the PDAID GSVC trace

PDAID GSVC trace entries recognized by SYSLOG ID for F2

3FFFF (Last byte of real address area)

Start of the "space" between the end of real storage specified during system generation and the end of main storage (The CPU physical storage)

(EX108KT)

Figure G.1 part 6 of 20

```
079FE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................ ................

BLOCK 244

07A000  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................ ................
07A020  --SAME--
07A7E0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................ ................

BLOCK 245

07A800  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................ ................
07A820  --SAME--
07AFE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ................ ................

        END OF CORE DUMP
```

Last 2K block but one of CPU physical (main) storage

(EX 109 KT)

```
IJBSEGT CONTENTS IS TAKEN TO LOCATE THE SEGMENT TABLE
VIRT.ADDR  REAL ADDR  BLOCK  STATUS
  000000    000000    000    CHANGED
  000800    000800    001    CHANGED
  001000    001000    002    CHANGED
  001800    001800    003    UNCHANGED
  002000    002000    004    CHANGED
  002800    002800    005    CHANGED
  003000    003000    006    UNCHANGED
  003800    003800    007    CHANGED
  004000    004000    008    CHANGED
  004800    004800    009    CHANGED
  005000    005000    010    CHANGED
  005800    005800    011    CHANGED
  006000    006000    012    CHANGED
  006800    006800    013    UNCHANGED
  007000    00700r    014    CHANGED
  007800    007800    015    CHANGED
  008000    008000    016    CHANGED
  008800    008800    017    CHANGED
  009000    009000    018    CHANGED
  009800    009800    019    UNCHANCED
  00A000    00A000    020    CHANGED
  00A800    00A800    021    CHANGED
  00B000    00B000    022    CHANGED
  00B800    00B800    023    CHANGED
  00C000    00C000    024    CHANGED
  00C800    00C800    025    CHANGED
  00D000    00D000    026    CHANGED
  00D800    00D800    027    CHANGED
  00E000    00E000    028    CHANGED
  00E800    00E800    029    CHANGED
  00F000    00F000    030    CHANGED
  00F800    00F800    031    CHANGED

  010000    010000    032    CHANGED
  010800    010800    033    CHANGED
  011000    011000    034    CHANGED
  011800    011800    035    CHANGED
  012000    012000    036    CHANGED
  012800    012800    037    CHANGED
  013000    013000    038    CHANGED
  013800    013800    039    CHANGED
  014000    014000    040    CHANGED
  014800    014800    041    CHANGED
  015000    015000    042    CHANGED
  015800    015800    043    CHANGED
  016000    016000    044    CHANGED
  016800    016800    045    CHANGED

  03D800    03D800    123    CHANGED
  03E000    03E000    124    CHANGED
  03E800    03E800    125    CHANGED
  03F000    03F000    126    CHANGED
  03F800    03F800    127    CHANGED

  040000    020800    065    CHANGED
  040800    019800    051    CHANGED
  041000    017800    047    CHANGED
  041800    018000    048    CHANGED
  042000    018800    049    CHANGED

  061000    01C800    057    CHANGED
  061800    01D000    058    CHANGED
```

Segment 0

Segment 0, all 32 pages in real storage (for the supervisor)

Segment 1, only 14 pages in real storage from this segment (also for the supervisor)

No pages from Segments 2 and 3 were in real storage

Segment 4, 5 pages frames used by PDAID GSVC trace as an alternate area.

Segment 5, 5 page frames used by BG job running in virtual mode.

(EX 110 KT)

Figure G.1 part 7 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
09E800     027800     079    UNCHANGED
09F000     027000     078    UNCHANGED
09F800     026000     076    UNCHANGED
```

Part of Segment 9

```
0A0000     028800     081    CHANGED
0A0800     028000     080    CHANGED
0A1000     02A800     085    UNCHANGED
0A1800     028000     086    UNCHANGED
```

Segment 10

Address of last virtual page in real storage

Indicates that operator did not execute the STORE STATUS function

```
STORE STATUS FUNCTION NOT EXECUTED, CR IN INITIAL STATUS

CR 0-7   000000E0 00000000 FFFFFFFF FFFFFFFF    00000000 00000000 00000000 00000000
CR 8-F   00000000 00000000 00000000 00000000    00000000 00000000 C2000000 00000200
```

(Ex 111 KT)

*** COMMUNICATION REGION ***

HEX DISP	BG	F4	F3	F2	F1	
DISP	04A0	4590	44F0	4450	4380	COMMUNICATION REGION ADDRESS
00	12/06/73	12/06/73	12/06/73	12/06/73	12/06/73	DATE
08	7000	7000	7000	7000	7000	PPBEG ADDR
0A	7000	7000	7000	7000	7000	END OF STORAGE PROTECT
0C	0000	0000	0000	0000	0000	SEEK ADDRESS BLOCK,ONLY BG VLD
0E	0000000000	0000000000	0000000000	0000000000	0000000000	PROBLEM PROGRAM USERS
	00000000	00000000	00000000	00000000	00000000	AREA IN HEX
17	00	00	00	00	00	UPSI BYTE IN HEX
18	TOMSDMP	KENSLOOP	NO NAME	CALCSIM	NO NAME	JOB NAME
20	00060FFF	00074FFF	00000000	000A1FFF	00000000	UPPERMOST BYTE OF EACH PPA
24	0004232F	00063C5B	00000000	0008AFF6	00000000	END ADDR OF LAST FETCH OR LOAD
28	0004232F	00000000	00000000	0008D4DF	00000000	LARGEST PROBLEM PROGRAM PHASE
2C	0000	0000	0000	0000	0000	LENGTH OF PP LABEL AREA
2E	0040	0020	0030	0040	0050	PROGRAM IDENTIFICATION KEY
30	0008FFFF	0008FFFF	0008FFFF	0008FFFF	0008FFFF	END OF STORAGE ADDRESS
34	FD	FD	FD	FD	FD	MACHINE CONFIGURATION
35	7F	7F	7F	7F	7F	SYSTEM CONFIGURATION
36	CED330A0CED0	CED33000CED2	CED30000CE50	CED330804E00	CED30000CE50	JOB CONTROL SWITCHES
3C	002E	002E	002E	002E	002E	DISK ADDR OF LABEL CYLINDER
3E	40FD	40FD	40FD	40FD	40FD	ADDR OF FOCL
40	4104	4104	4104	4104	4104	ADDR OF PUB
	4296	4296	4296	4296	4296	ADDR OF FAVP
44	4297	4297	4297	4297	4297	ADDR OF JIB
46	4389	4389	4389	4389	4389	ADDR OF TEB
48	3F00	3F00	3F00	3F00	3F00	ADDR OF FICL
4A	3F06	3F06	3F06	3F06	3F06	ADDR OF NICL
4C	3F0C	3F0C	3F0C	3F0C	3F0C	ADDR OF LUB
4E	38	38	38	38	38	LINE COUNT FOR SYSLST
4F	120673340	120673340	120673340	120673340	120673340	SYSTEM DATE
58	0000	0000	0000	0000	0000	LIOCS COM BYTE
5A	3CD4	3CD4	3CD4	3CD4	3CD4	ADDR OF PIB TABLE
5C	0000	0000	0000	0000	0000	LAST CHECK POINT NO.
5E	003C	003C	003C	003C	003C	JOB ZONE IN MINUTES
60	4630	46A8	4720	4798	4810	ADDR OF DIB
62	0000	0000	0000	0000	0000	CURRENTLY NOT ASSIGNED
64	3DCC	3DCC	3DCC	3DCC	3DCC	ADDR OF PC OPTION TABLE
66	3E4C	3E4C	3E4C	3E4C	3E4C	ADDR OF IT OPTION TABLE
68	3EBC	3EBC	3EBC	3EBC	3EBC	ADDR OF OC OPTION TABLE
6A	0010	0010	0010	0010	0010	KEY OF PROGRAM WITH IT SUPPORT
6C	0000	0000	0000	0000	0000	CURRENTLY NOT ASSIGNED
6E	0000	0020	0030	0040	0050	LTK
70	00008090	00008087	0000807E	00008075	0000806C	SYSPARM
74	00007118	000071D0	00007288	00007340	000073F8	JOB ACCOUNTING
78	00003864	00003864	00003864	00003864	00003864	ADDR OF TOD COMMUNICATIONS AREA
7C	3BD4	3BD4	3BD4	3BD4	3BD4	ADDR OF PIB EXTENSION
7E	5A30	5A30	5A30	5A30	5A30	ADDR OF MICR DTF LABEL
80	00000000	00000000	00000000	00000000	00000000	ADDR OF QTAM VECTOR TABLE
84	04A0	04A0	04A0	04A0	04A0	ADDR OF BG COMREG
86	10E0	10E0	10E0	10E0	10E0	RESERVED
88	00000588	00000588	00000588	00000588	00000588	ADDR OF COMREG EXTENSION
8C	C0C0	C0C0	C0C0	C0C0	C0C0	RESERVED
8E	03	00	00	00	00	DISK CONFIGURATION BYTE

(Ex 112 KT)

Figure G.1 part 8 of 20

```
                        *** PROGRAM INFORMATION BLOCK ***

AR PIB  80  00  C1D9  00000000  000063E8  000E  2880
BG PIB  82  00  C2C7  80040000  00006458  000E  2880
F4 PIB  82  00  C6F4  80061000  00006518  8044  1E80
F3 PIB  80  00  C6F3  80007800  000065D8  0070  1E00
F2 PIB  83  00  C6F2  80089000  00006698  009C  1E80
F1 PIB  80  00  C6F1  80008800  00006758  00C8  1E00

                  AP SUBTASK PIBS

         80000000 00000000 00006818 00000000
         80000000 00000000 000068D8 00000000
         80000000 00000000 00006998 00000000
         80000000 00000000 00006A58 00000000
         80000000 00000000 00006B18 00000000
         80000000 00000000 00006BD8 00000000
         80000000 00000000 00006C98 00000000
         80000000 00000000 00006D58 00000000
         80000000 00000000 00006E18 00000000
         80000000 00000000 00006ED8 00000000


PARTITION SAVE AREA

BG    PSW=071D00000004038A
      REG  9-0  00000006 0000000A 4004007A 0004107A D7C8C1E2 900402D6 00041ED8 00040C59
      REG  1-8  00041712 00000021 00000061 00000010 00000001 00000001 00000001 00000000

F4    PSW=070D000000061CC6
      REG  9-0  80062A60 50061FFA 00061088 00062888 00063888 00004590 00000025 00000003
      REG  1-8  000614E8 00004104 00061106 00063A92 00000000 00004254 A0061A2E B0063012

F3    *** PROG NOT ACTIVE ***

F2    PSW=074D000000089858
      REG  9-0  00089E80 0009E447 40089F5A 4008AD9A 000892E0 80089EF0 00089AA0 00089A7C
      REG  1-8  00089D24 00089AC0 00089ECF 00089E80 00000005 A0089EDC 0008A157 00089978

F1    *** PROG NOT ACTIVE ***
```

3 partitions active

(EX 113 KT)

```
*** LOGICAL UNIT BLOCK TABLE ***

LUB LOGIC   PUB JIB
TAB UNIT    PTR PTR CUU

BG SYSTEM LUBS

00  SYSRDR 01  FF  00C
01  SYSIPT 01  FF  00C
02  SYSPCH 02  FF  00D
03  SYSLST 03  FF  00E
04  SYSLOG 00  FF  009
05  SYSLNK 07  02  131
06  SYSRES 06  FF  130
07  SYSSLB 06  FF  130
08  SYSRLB 06  04  130
09  SYSUSE FF  FF  UA
0A  SYSREC 07  01  131
0B  SYSCLB 06  00  130
0C  SYSVIS 07  FF  131
0D  SYSCAT FF  FF  UA

BG PROGRAMMER LUBS

0E  SYS000 07  FF  131
0F  SYS001 07  FF  131
10  SYS002 07  FF  131
11  SYS003 13  03  381
12  SYS004 0B  FF  260
13  SYS005 FF  FF  UA
14  SYS006 FF  FF  UA
15  SYS007 FF  FF  UA
16  SYS008 FF  FF  UA
17  SYS009 FF  FF  UA
18  SYS010 FF  FF  UA
19  SYS011 FF  FF  UA
1A  SYS012 FF  FF  UA
1B  SYS013 FF  FF  UA
1C  SYS014 FF  FF  UA
1D  SYS015 FF  FF  UA
```

(EX 114 KT)

Figure G.1 part 9 of 20

Appendix G

EXAMPLE OF A STAND ALONE DUMP OUTPUT

*** PHYSICAL UNIT BLOCK TABLE ***

POS	CHAN AND UNIT	CHAN QUE PTR	TEB PTR	DEV TYP	DEV CODE	CHAN SCHD FLGS	JOB CTL FLGS	DEV BUSY	SWIT CHAB LE	EOF SYSRDR SYSIPT	IOERR QUED RECOV	OPER INTV REQ	DEV END POST	BURST DV ON MPX	SEVEN TRACK TAPE	* PUB OWNER * ** SHIP ** * EXTENSION *
000	0009	04	00	00	00	80	F8	*								0000
001	000C	FF	00	11	00	00	F8									0001
002	000D	FF	00	21	00	00	F8									0001
003	000E	FF	00	42	00	00	F8									0001
004	001F	FF	00	00	00	00	F8									0000
005	0111	FF	00	80	F0	00	F8									0000
006	0130	FF	00	62	00	02	F8							*		0008
007	0131	FF	00	62	01	02	F8							*		001F
008	0132	FF	00	62	02	02	F8							*		0000
009	0133	FF	00	62	03	02	F8							*		0000
00A	0134	FF	00	62	04	02	F8							*		0000
00B	0260	FF	00	63	05	02	F8							*		001F
00C	0261	FF	00	63	06	02	F8							*		0000
00D	02A0	FF	00	50	C3	00	C0									0000
00E	02A1	FF	00	50	C3	00	C0									0002
00F	02A2	FF	00	50	C3	00	C0									0002
010	02A3	FF	00	50	C3	00	C0									0008
011	02A4	FF	00	50	C3	00	C0									0000
012	0380	FF	00	50	C3	02	C0							*		0002
013	0381	FF	00	50	C3	02	C0							*		0001
014	0382	FF	00	50	C3	02	C0							*		0008
015	0383	FF	00	50	C3	02	C0							*		0008

Ex115Kt

Figure G.1 part 10 of 20

```
*** ERROR RECOVERY BLOCK ***

  //ABERRA  FETCH NAME
  4DC2  RETRY EXIT ADDRESS
  4A4C  IGNORE EXIT ADDRESS
  0806  CANCEL EXIT ADDRESS
  2E6C  SUPERVISOR RETURN ADDR


      STORED CSW       PUB   FLAG  MSG * SEEK ADR    CCB ADDRESS  DEV
                       ADDR  BYTE  CODE                           ADDR

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  0000000000000000  0000    00    00    00000000  00000000     000
  SENSE DATA  000000000000000000000000000000000000000000000000

  1000E6D00A000000  410C    04    00    00000000  0000E7E8     00C
  SENSE DATA  100000000000000000000000000000000000000000000000

  4000E9C00E000005  0000    04    00    007A0013  00000000     000
  SENSE DATA  002000400000000000000000000000000000000000000000


* MESSAGE CODE IS SECOND AND THIRD BYTE OF DEVICE ERROR RECOVERY MESSAGES
  GENERATED BY PHYSICAL IOCS (EXAMPLE 0P08A INTERV REQ)




    *** CHANNEL QUEUE TABLE ***

POS CHAIN CCB ADDR       CUU
     PTR

 00  01     0076C8
 01  08     007660
 02  03     00E7A0
 03  05     007798
 04  07     00E908        009
 05  00     007730
 06  0B     0039A0
 07  FF     00E680
 08  0A     007938
 09  06     007868
 0A  09     0078D0
 0B  0C     000000
 0C  0D     000000
 0D  0E     000000
 0E  0F     000000
 0F  10     000000
 10  11     000000
 11  12     000000
 12  13     000000
 13  14     000000
```

(Ex116KT)

Figure G.1 part 11 of 20

Appendix G

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
2C  2D    000000
2D  2E    000000
2E  2F    000000
2F  30    000000
30  31    000000
31  FF    000000
```

Last part of CHANQ

```
*** FLOATING POINT REGISTERS ***

  REG 0  00.00000000000000
  REG 2  00.00000000000000
  REG 4  00.00000000000000
  REG 6  00.00000000000000
```

(EX117 KT)

Figure G.1 part 12 of 20

```
                    *** COPIED AND TRANSLATED CCB ***

00E680                                           ADDR.OF CCB COPY BLOCK QUEUE
0000                                             RESIDUAL COUNT
00                                               1ST COMMUNICATION BYTE
00                                               2ND COMMUNICATION BYTE
0000                                             STATUS BYTES FROM CSW
20                                               LUB CLASS
04                                               LUB NUMBER
0000E830                                         ADDR.OF CCW STRING
00                                               3RD COMMUNICATION BYTE
061518                                           CCW ADDRESS IN CSW
0000000000000000                                 USER SENSE CCW
0020                                             PIK VALUE
C0                                               FLAG BYTE
00                                               UNUSED
000614E8                                         ADDR.OF VIRTUAL CCB
0000E830                                         ADDR.OF CCW COPY.BLOCK
00000000                                         ADDR.OF IDAL
000000000000004000000000000000000000000000000000 PAGES FIXED
00000000                                         PTR TO FIXINF EXT
0000E908                                         ADDR.OF NEXT CCB
```

No FIXINF EXT BLOCKS (There would follow if present)

(Ex 118 KT)

```
              *** COPIED AND TRANSLATED CHANNEL PROGRAM ***

ADDR.OF CCW BLOCK        COM.CODE  DATA ADDR.  FLAGS  BYTE COUNT  VIRT.ADDR.OF 1ST CCW  ADDR.OF NEXT CCW BLOCK

00E830            CCW1      09      01C96E      60      000F         061500                000000
                  CCW2      01      01C8EE      60      0003
                  CCW3      09      01C9BE      20      001C
                  CCW4      00      000000      00      0000
                  CCW5      00      000000      00      0000
                  CCW6      00      000000      00      0000
                  CCW7      00      000000      00      0000
                   TIC      80      000000
                   TIC      88      000000
```

(Ex 119 KT)

```
                          *** IDAL BLOCK QUEUE ***

              NO IDAL BLOCK QUEUE AVAILABLE

                          *** FIXED PAGE FRAMES ***

0000057
```

(Ex 120 KT)

Figure G.1 part 13 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

*** COPIED AND TRANSLATED CCB ***

00E908	ADDR.OF CCB COPY BLOCK QUEUE
0000	RESIDUAL COUNT
04	1ST COMMUNICATION BYTE
00	2ND COMMUNICATION BYTE
0000	STATUS BYTES FROM CSW
20	LUB CLASS
04	LUB NUMBER
0000E7E8	ADDR.OF CCW STRING
00	3RD COMMUNICATION BYTE
041780	CCW ADDRESS IN CSW
0000000000000000	USER SENSE CCW
0010	PIK VALUE
C0	FLAG BYTE
00	UNUSED
00041712	ADDR.OF VIRTUAL CCB
0000E7E8	ADDR.OF CCW COPY.BLOCK
0000E950	ADDR.OF IDAL
00000000000180000000000000000000000000000000	PAGES FIXED
00000000	PTR TO FIXINF EXT
00000000	ADDR.OF NEXT CCB

(Ex121 KT)

*** COPIED AND TRANSLATED CHANNEL PROGRAM ***

ADDR.OF CCW BLOCK		COM.CODE	DATA ADDR.	FLAGS	BYTE COUNT	VIRT.ADDR.OF 1ST CCW	ADDR.OF NEXT CCW BLOCK
00E7E8	CCW1	09	017F88	60	001C	041738	00E710
	CCW2	09	017FA4	60	0020		
	CCW3	09	017FC4	60	002F		
	CCW4	09	00E950	64	001F		
	CCW5	09	018012	60	0021		
	CCW6	09	018033	60	0021		
	CCW7	09	018054	60	0020		
	TIC	88	00E710				
	TIC	88	000000				
00E710	CCW1	09	018074	60	0020	041770	000000
	CCW2	09	018094	20	0025		
	CCW3	00	000000	00	0000		
	CCW4	00	000000	00	0000		
	CCW5	00	000000	00	0000		
	CCW6	00	000000	00	0000		
	CCW7	00	000000	00	0000		
	TIC	80	000000				
	TIC	88	000000				

(Ex 122 KT)

*** IDAL BLOCK QUEUE ***

ADDR.OF IDAL BLOCK	IDAL ADDRESSES		
00E950	00017FF3	00018000	FF000000

*** FIXED PAGE FRAMES ***

0000047 0000048

(Ex123 KT)

Figure G.1 part 14 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

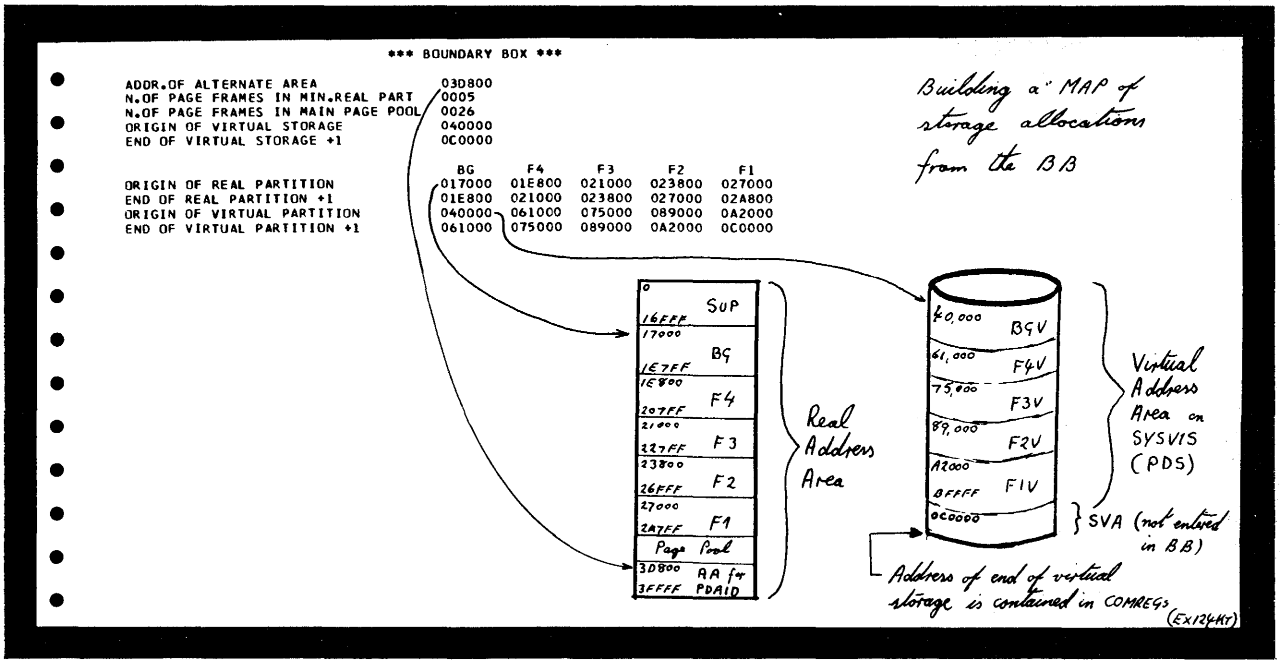


Figure G.1 part 15 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

*** SEGMENT TABLE ***

ADDR.OF ST	PT ENTRIES	PT ORIGIN ADDR
00E640	32	00E330

Segment 0

*** PAGE TABLE ***

SUPVR

VIRT.ADDR	REAL ADDR	KEY	STATUS
000000	000000	00	CHANGED
000800	000800	00	CHANGED
001000	001000	00	CHANGED
001800	001800	00	UNCHANGED
002000	002000	00	CHANGED
002800	002800	00	CHANGED
003000	003000	00	UNCHANGED
003800	003800	00	CHANGED
004000	004000	00	CHANGED
004800	004800	00	CHANGED
005000	005000	00	CHANGED
005800	005800	00	CHANGED
006000	006000	00	CHANGED
006800	006800	00	UNCHANGED
007000	007000	00	CHANGED
007800	007800	00	CHANGED
008000	008000	00	CHANGED
008800	008800	00	CHANGED
009000	009000	00	CHANGED
009800	009800	00	UNCHANGED
00A000	00A000	00	CHANGED
00A800	00A800	00	CHANGED
00B000	00B000	00	CHANGED
00B800	00B800	00	CHANGED
00C000	00C000	00	CHANGED
00C800	00C800	00	CHANGED
00D000	00D000	00	CHANGED
00D800	00D800	00	CHANGED
00E000	00E000	00	CHANGED
00E800	00E800	00	CHANGED
00F000	00F000	00	CHANGED
00F800	00F800	00	CHANGED

*** SEGMENT TABLE ***

ADDR.OF ST	PT ENTRIES	PT ORIGIN ADDR
00E644	32	00E370

Segment 1

*** PAGE TABLE ***

PIK for BG

BG

VIRT.ADDR	REAL ADDR	KEY	STATUS
010000	010000	00	CHANGED
010800	010800	00	CHANGED
011000	011000	00	CHANGED
011800	011800	00	CHANGED
012000	012000	00	CHANGED
012800	012800	00	CHANGED
013000	013000	00	CHANGED
013800	013800	00	CHANGED
014000	014000	00	CHANGED
014800	014800	00	CHANGED
015000	015000	00	CHANGED
015800	015800	00	CHANGED
016000	016000	00	CHANGED
016800	016800	00	CHANGED
017000		10	NOT ADDRESSABLE
017800		10	NOT ADDRESSABLE
018000		10	NOT ADDRESSABLE
018800		10	NOT ADDRESSABLE
019000		10	NOT ADDRESSABLE
019800		10	NOT ADDRESSABLE
01A000		10	NOT ADDRESSABLE
01A800		10	NOT ADDRESSABLE
01B000		10	NOT ADDRESSABLE
01B800		10	NOT ADDRESSABLE
01C000		10	NOT ADDRESSABLE
01C800		10	NOT ADDRESSABLE
01D000		10	NOT ADDRESSABLE
01D800		10	NOT ADDRESSABLE
01E000		10	NOT ADDRESSABLE
01E800		20	NOT ADDRESSABLE
01F000		20	NOT ADDRESSABLE
01F800		20	NOT ADDRESSABLE

(EX125K)

Figure G.1 part 16 of 20

Segment 3

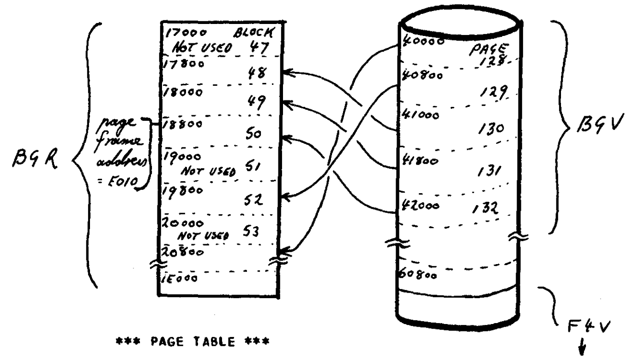
SUPVR

V=R

033000		00	NOT ADDRESSABLE
033800		00	NOT ADDRESSABLE
034000		00	NOT ADDRESSABLE
034800		00	NOT ADDRESSABLE
035000		00	NOT ADDRESSABLE
035800		00	NOT ADDRESSABLE
036000		00	NOT ADDRESSABLE
036800		00	NOT ADDRESSABLE
037000		00	NOT ADDRESSABLE
037800		00	NOT ADDRESSABLE
038000		00	NOT ADDRESSABLE
038800		00	NOT ADDRESSABLE
039000		00	NOT ADDRESSABLE
039800		00	NOT ADDRESSABLE
03A000		00	NOT ADDRESSABLE
03A800		00	NOT ADDRESSABLE
03B000		00	NOT ADDRESSABLE
03B800		00	NOT ADDRESSABLE
03C000		00	NOT ADDRESSABLE
03C800		00	NOT ADDRESSABLE
03D000		00	NOT ADDRESSABLE
03D800	03D800	00	CHANGED
03E000	03E000	00	CHANGED
03E800	03E800	00	CHANGED
03F000	03F000	00	CHANGED
03F800	03F800	00	CHANGED

Last part of real storage used by PDAID alternate area in this example.

Information obtained from the page table and page frame table enables you to build up a picture of pages and block allocations.



*** SEGMENT TABLE ***

ADDR.OF ST	PT ENTRIES	PT ORIGIN ADDR
00E650	32	00E430

Segment 4

*** PAGE TABLE ***

BGV

VIRT.ADDR	REAL ADDR	KEY	STATUS
040000	020800	10	CHANGED
040800	019800	10	CHANGED
041000	017800	10	CHANGED
041800	018000	10	CHANGED
042000	018800	10	CHANGED
042800		10	NOT USED
043000		10	NOT USED
043800		10	NOT USED
044000		10	NOT USED
044800		10	NOT USED
045000		10	NOT USED
045800		10	NOT USED
046000		10	NOT USED
046800		10	NOT USED
047000		10	NOT USED
047800		10	NOT USED
048000		10	NOT USED
048800		10	NOT USED
049000		10	NOT USED
049800		10	NOT USED
04A000		10	NOT USED
04A800		10	NOT USED
04B000		10	NOT USED
04B800		10	NOT USED
04C000		10	NOT USED
04C800		10	NOT USED
04D000		10	NOT USED
04D800		10	NOT USED
04E000		10	NOT USED
04E800		10	NOT USED
04F000		10	NOT USED
04F800		10	NOT USED

(EX126KT)

Figure G.1 part 17 of 20

EXAMPLE OF A STAND ALONE DUMP OUTPUT

Segment 10

*** SEGMENT TABLE ***

ADDR.OF ST	PT ENTRIES	PT ORIGIN ADDR
00E668	32	00E580

*** PAGE TABLE ***

P/K for F2

F2V

VIRT.ADDR	REAL ADDR	KEY	STATUS
0A0000	028800	40	CHANGED
0A0800	028000	40	CHANGED
0A1000	02A800	40	UNCHANGED
0A1800	028000	40	UNCHANGED
0A2000		50	NOT USED
0A2800		50	NOT USED
0A3000		50	NOT USED
0A3800		50	NOT USED
0A4000		50	NOT USED
0A4800		50	NOT USED
0A5000		50	NOT USED
0A5800		50	NOT USED
0A6000		50	NOT USED
0A6800		50	NOT USED
0A7000		50	NOT USED
0A7800		50	NOT USED
0A8000		50	NOT USED
0A8800		50	NOT USED
0A9000		50	NOT USED
0A9800		50	NOT USED
0AA000		50	NOT USED
0AA800		50	NOT USED
0AB000		50	NOT USED
0AB800		50	NOT USED
0AC000		50	NOT USED
0AC800		50	NOT USED
0AD000		50	NOT USED
0AD800		50	NOT USED
0AE000		50	NOT USED
0AE800		50	NOT USED
0AF000		50	NOT USED
0AF800		50	NOT USED

P/K for F1

F1V

Segment 11

*** SEGMENT TABLE ***

ADDR.OF ST	PT ENTRIES	PT ORIGIN ADDR
00E66C	32	00E5F0

*** PAGE TABLE ***

VIRT.ADDR	REAL ADDR	KEY	STATUS
0B0000		50	NOT USED
0B0800		50	NOT USED
0B1000		50	NOT USED
0B1800		50	NOT USED
0B2000		50	NOT USED
0B2800		50	NOT USED
0B3000		50	NOT USED
0B3800		50	NOT USED
0B4000		50	NOT USED
0B4800		50	NOT USED
0B5000		50	NOT USED
0B5800		50	NOT USED
0B6000		50	NOT USED
0B6800		50	NOT USED
0B7000		50	NOT USED
0B7800		50	NOT USED
0B8000		50	NOT USED
0B8800		50	NOT USED
0B9000		50	NOT USED
0B9800		50	NOT USED
0BA000		50	NOT USED
0BA800		50	NOT USED
0BB000		50	NOT USED
0BB800		50	NOT USED
0BC000		50	NOT USED
0BC800		50	NOT USED
0BD000		50	NOT USED
0BD800		50	NOT USED
0BE000		50	NOT USED
0BE800		50	NOT USED
0BF000		50	NOT USED
0BF800		50	NOT USED

EX127KT

Figure G.1 part 18 of 20

*** PAGE FRAME TABLE *** — ** PFT EXT **

ADDR.OF PFT	FIX CNT	NFF	DR	SP	NF	FORWARD-PTR	BACKWARD-PTR	PAGE NR	ADDR.OF PAGE	PFIX CNT
		---FLAGS---								
00DE88	00000	0	0	1	0			000	000000	000
00DE90	00000	0	0	1	0			001	000800	000
00DE98	00000	0	0	1	0			002	001000	000
00DEA0	00000	0	0	1	0			003	001800	000
00DEA8	00000	0	0	1	0			004	002000	000
00DEB0	00000	0	0	1	0			005	002800	000
00DEB8	00000	0	0	1	0			006	003000	000
00DEC0	00000	0	0	1	0			007	003800	000
00DEC8	00000	0	0	1	0			008	004000	000
00DED0	00000	0	0	1	0			009	004800	000
00DED8	00000	0	0	1	0			010	005000	000
00DEE0	00000	0	0	1	0			011	005800	000
00DEE8	00000	0	0	1	0			012	006000	000
00DEF0	00000	0	0	1	0			013	006800	000
00DEF8	00000	0	0	1	0			014	007000	000
00DF00	00000	0	0	1	0			015	007800	000
00DF08	00000	0	0	1	0			016	008000	000
00DF10	00000	0	0	1	0			017	008800	000
00DF18	00000	0	0	1	0			018	009000	000
00DF20	00000	0	0	1	0			019	009800	000
00DF28	00000	0	0	1	0			020	00A000	000
00DF30	00000	0	0	1	0			021	00A800	000
00DF38	00000	0	0	1	0			022	00B000	000
00DF40	00000	0	0	1	0			023	00B800	000
00DF48	00000	0	0	1	0			024	00C000	000
00DF50	00000	0	0	1	0			025	00C800	000
00DF58	00000	0	0	1	0			026	00D000	000
00DF60	00000	0	0	1	0			027	00D800	000
00DF68	00000	0	0	1	0			028	00E000	000
00DF70	00000	0	0	1	0			029	00E800	000
00DF78	00000	0	0	1	0			030	00F000	000
00DF80	00000	0	0	1	0			031	00F800	000
00DF88	00000	0	0	1	0			032	010000	000
00DF90	00000	0	0	1	0			033	010800	000
00DF98	00000	0	0	1	0			034	011000	000
00DFA0	00000	0	0	1	0			035	011800	000
00DFA8	00000	0	0	1	0			036	012000	000
00DFB0	00000	0	0	1	0			037	012800	000
00DFB8	00000	0	0	1	0			038	013000	000
00DFC0	00000	0	0	1	0			039	013800	000
00DFC8	00000	0	0	1	0			040	014000	000
00DFD0	00000	0	0	1	0			041	014800	000
00DFD8	00000	0	0	1	0			042	015000	000
00DFE0	00000	0	0	1	0			043	015800	000
00DFE8	00000	0	0	1	0			044	016000	000
00DFF0	00000	0	0	1	0			045	016800	000
00DFF8	00000	0	0	0	0	00E068	00E288	UNUSED		000
00E000	00001	0	0	1	0			130	041000	000
00E008	00001	0	0	1	0			131	041800	000
00E010	00000	0	0	0	0	00E090	00E0C8	132	042000	000
00E018	00000	0	0	0	0	00E078	00E128	315	09D800	000
00E020	00000	0	0	0	0	00E128	00E0F0	129	040800	000
00E028	00000	0	0	0	0	00E098	00E068	UNUSED		000
00E030	00000	0	0	0	0	00E0D8	00E048	197	062800	000
00E038	00000	0	0	0	0	00E070	00E0A0	310	098000	000
00E040	00000	0	0	0	0	00E048	00E2A0	199	063800	000

Page frame at address E010 was occupied by page 132 which has a virtual address of 42,000

*** PAGE FRAME TABLE *** — ** PFT EXT **

ADDR.OF PFT	FIX CNT	NFF	DR	SP	NF	FORWARD-PTR	BACKWARD-PTR	PAGE NR	ADDR.OF PAGE	PFIX CNT
		---FLAGS---								
00E048	00000	0	0	0	0	00E030	00E040	198	063000	000
00E050	00001	0	0	1	0			194	061000	000
00E058	00000	0	0	0	0	00E2A0	00E060	195	061800	000
00E060	00000	0	0	0	0	00E058	00E090	196	062000	000
00E068	00000	0	0	0	0	00E028	00DFF8	UNUSED		000
00E070	00000	0	0	0	0	00E0F0	00E038	311	098800	000
00E078	00000	0	0	0	0	00E080	00E018	312	09C000	000
00E080	00000	0	0	0	0	00E088	00E078	313	09C800	000
00E088	00000	0	0	0	0	00E0B0	00E080	314	09D000	000
00E090	00000	0	0	0	0	00E060	00E010	128	040000	000
00E098	00000	0	0	0	0	00E140	00E028	UNUSED		000
00E0A0	00000	0	0	0	0	00E038	00E138	305	098800	000
00E0A8	00000	0	0	0	0	00E0D0	00E120	281	08C800	000
00E0B0	00000	0	0	0	0	00E118	00E088	274	089000	000
00E0B8	00000	0	0	0	0	00E0E0	00E0D8	277	08A800	000
00E0C0	00000	0	0	0	0	00E298	00E118	275	089800	000
00E0C8	00000	0	0	0	0	00E010	00E0D0	279	08B800	000
00E0D0	00000	0	0	0	0	00E0C8	00E0A8	280	08C000	000
00E0D8	00000	0	0	0	0	00E0B8	00E030	276	08A000	000
00E0E0	00000	0	0	0	0	00E120	00E0B8	278	08B000	000
00E0E8	00000	0	0	0	0	00E0F8	00E298	319	09F800	000
00E0F0	00000	0	0	0	0	00E020	00E070	309	09A800	000
00E0F8	00000	0	0	0	0	00E100	00E0E8	318	09F000	000
00E100	00000	0	0	0	0	00E108	00E0F8	317	09E800	000
00E108	00000	0	0	0	0	00E110	00E100	321	0A0800	000
00E110	00000	0	0	0	0	00E130	00E108	320	0A0000	000
00E118	00000	0	0	0	0	00E0C0	00E0B0	316	09E000	000

EX12817

Figure G.1 part 19 of 20

00E1A8	00000	0	0	0	0	00E1B0	00E1A0	UNUSED		000
00E1B0	00000	0	0	0	0	00E1B8	00E1A8	UNUSED		000
00E1B8	00000	0	0	0	0	00E1C0	00E1B0	UNUSED		000
00E1C0	00000	0	0	0	0	00E1C8	00E1B8	UNUSED		000
00E1C8	00000	0	0	0	0	00E1D0	00E1C0	UNUSED		000
00E1D0	00000	0	0	0	0	00E1D8	00E1C8	UNUSED		000
00E1D8	00000	0	0	0	0	00E1E0	00E1D0	UNUSED		000
00E1E0	00000	0	0	0	0	00E1E8	00E1D8	UNUSED		000
00E1E8	00000	0	0	0	0	00E1F0	00E1E0	UNUSED		000
00E1F0	00000	0	0	0	0	00E1F8	00E1E8	UNUSED		000
00E1F8	00000	0	0	0	0	00E200	00E1F0	UNUSED		000
00E200	00000	0	0	0	0	00E208	00E1F8	UNUSED		000

These pages not in real storage

*** PAGE FRAME TABLE ***   ** PFT EXT **

ADDR.OF PFT	FIX CNT	---FLAGS--- NFF	DR	SP	NF	FORWARD-PTR	BACKWARD-PTR	PAGE NR	ADDR.OF PAGE	PFIX CNT
00E208	00000	0	0	0	0	00E210	00E200	UNUSED		000
00E210	00000	0	0	0	0	00E218	00E208	UNUSED		000
00E218	00000	0	0	0	0	00E220	00E210	UNUSED		000
00E220	00000	0	0	0	0	00E228	00E216	UNUSED		000
00E228	00000	0	0	0	0	00E230	00E220	UNUSED		000
00E230	00000	0	0	0	0	00E238	00E228	UNUSED		000
00E238	00000	0	0	0	0	00E240	00E230	UNUSED		000
00E240	00000	0	0	0	0	00E248	00E238	UNUSED		000
00E248	00000	0	0	0	0	00E250	00E240	UNUSED		000
00E250	00000	0	0	0	0	00E258	00E248	UNUSED		000
00E258	00000	0	0	0	0	00E288	00E250	UNUSED		000
00E260	00000	0	0	1	1			123	03D800	000
00E268	00000	0	0	1	1			124	03E000	000
00E270	00000	0	0	1	1			125	03E800	000
00E278	00000	0	0	1	1			126	03F000	000
00E280	00000	0	0	1	1			127	03F800	000

These pages not in real storage

NOTE

NF BIT ON-PAGE FRAME NOT TEMPORARILY FIXABLE
NFF BIT ON-PAGE NOT TEMPORARILY FIXABLE
DR BIT ON-FAILING STORAGE
SP BIT ON-PAGE FRAME NOT IN SELECTION POOL

Real addresses of these pages in real storage used by PDAIO alternate area (in this example)

All page frames in the selection pool are queued in one of 5 queues. From this table it is possible to establish the order in which page frames are selected.

Address of page frame to be used next in this queue

*** SELECTION POOL ***

ADDR.OF QUEUE	FORWARD PTR	BACKWARD PTR
00E288	00DFF8	00E258
00E290	00E290	00E290
00E298	00E0E8	00E0C0
00E2A0	00E040	00E058
00E2A8	00E2A8	00E2A8

5 Selection Pool Queues

Forward and Backward pointers are the same, this indicates that there are no page frames in these two queues

Figure G.1 part 20 of 20

TABLES USED BY JOB ACCOUNTING

Job Accounting Interface

The Job Accounting Interface provides job step and job information that you can use for charging system use, supervising system operation, planning new applications, etc.

The job accounting option is supported when JA=YES in the FOPT supervisor generation macro. When this option is supported, the following tables are generated:

- A job accounting interface partition table for each partition
- A job accounting common table.

Both tables are generated as part of the supervisor.

The interface table is part of the partition table and provides user access to the job accounting routines and information.

For each job step the following information is accumulated in this table:

- Job name
- User information
- Partition ID
- Cancel code
- Record type
- Date
- Job start time
- Phasename (from EXEC card)
- Highest address used (from communications region)
- CPU time
- Overhead time
- Stop time (at EOJ only)
- All bound time
- SIO count (optional).

Note: If the CPU is not equipped with a timer, time fields are zero.

To utilize this information, you must link-edit a routine to be relocatable by using the relocating loader option (or write a self-relocating routine) to store or print the desired portions of the table. This routine must be catalogued in the core image library under the name $JOBACCT.

How to locate

The address of the interface partition table is contained in bytes X'74'–X'77' of the partition communication region.

The address of the common table is contained in bytes X'7C'–X'7F' of SYSCOM.

Appendix H

TABLES USED BY JOB ACCOUNTING

Displacement	Label	Description
	(ACCTABLE)	
0 -3	ACCTWK1	Work area used in SIO update
4 -7	ACCTWK2	Work area used with ACCTWK1 in start/stop time routine
8 -11	ACCTSVPT	Job card pointer; address of job card field following jobname
12	ACCTPART	ID of partition in charge (partition switch name)
13	ACCTRES2	Reserved
14-15	ACCTLEN	Length of SIO area = 6n+1, where n = number of devices for this partition in SYSGEN option JA = n1, n2, n3, n4, n5)
16-21	ACCTLOAD	Label area instruction; moves JAI label area address to OPEN/CLOSE transients
22-23	ACCTRES3	Reserved
24-27	ACCTLADD	Address of alternate label area
28-31	ACCTPUT	Counter for CPU time elapsed in a jobstep, counted in 300ths of a second
32-35	ACCTOVHT	Counter for overhead time: time not charged to any partition
36-39	ACCTBNDT	Counter for all-bound time: system wait state time divided between running partitions
40-47	ACCTSVJN	Save area for job name during simulated EOJ
48-55	ACCTJBNM	Job name; taken from job card
56-71	ACCTUSRS	User information 16 bytes from job card
72-73	ACCTPTID	Partition ID: 'BG', 'F4', 'F3', 'F2' or 'F1' in EBCDIC format
74	ACCTCNCL	Cancel code; see Cancel Codes and Messages
75	ACCTYPER	Type of record: 'S' = job step, 'L' = last step of job
76-83	ACCTDATE	Date in format specified at SYSGEN (MM/DD/YY or DD/MM/YY)
84-87	ACCTSTRT	Start time of job, in packed decimal (DHHMMSSF; F = sign)
88-91	ACCTSTOP	Stop time of job, in same format as ACCTSTRT
92-95	ACCTRES	Reserved
96-103	ACCTEXEC	Phase name taken from execute card
104-107	ACCTHICR	End address of active program phase, from COMREG
108-111	ACCTIMES	CPU time elapsed in a job step counted in 300th of a second
112-115	----------------	Overhead time: elapsed time not charged to any partition, in 300ths of a second
116-119	----------------	All-bound time: system wait state time divided between running partitions, in 300ths of a sec.
120	ACCTSIOS	SIO tables: 6 bytes for each device specified by SYSGEN options, as follows: 2 bytes for device address (0cuu), 4 bytes for count of SIOs in current jobstep.
--------	----------------	Overflow byte: normally X'20', but is X'30' if more devices are used within a partition than specified by SYSGEN options

This part of the table is for user reference

Notes: DSECT ACCTABLE symbolically addresses the JAI Partition Tables with labels as shown. Each partition in which JAI is supported has its own JAI Partition Table, labeled ACCTBG, ACCTF4, ACCTF3, ACCTF2 and ACCTF1 for active partitions BG, F4, F3, F2, and F1 respectively.

Figure H-1. Explanation of the contents of the Job Accounting Interface partition table.

Displacement	Label	Description
	(ACCTCOMN)	
0-15	ACCTSVRG	Temporary register save area
16-17	ACCTSVRX	Save area for remainder of overhead counter times distributed by partition on exit
18-19	ACCTSVRE	Save area for remainder of all-bound counter times distributed by partitions on entry
20-23	ACCTPCNT	Count of partitions using the Job Accounting interface
24	ACCTSAID	Owner of physical transient area *)
25	ACCTFAID	Interrupted program *)
26	ACCTRAID	Active program *)
27	ACCTSWCH	Accounting switches: if bit = 1, true; if bit = 0, not true bit 0: cancel accounting · bit 4: IPL indicator bit 1: no active partitions · bit 5: not used bit 2: catalog in process · bit 6: not used bit 3: alternate label area · bit 7: not used
28-31	ACCTIME	Start time of current accounting interval, in complement format
32-33	ACCTRESC	Reserved
34-35	ACCTUSEP	Address of user save area (ACCTUSER)
36-37	ACCTUSEL	Length of user save area (Set with 1st operand of FOPT macro parameter JALIOCS)
38-39	ACCTSJOB	Job accounting partition indication
40-43	ACCTBLES	Address of BG Job Accounting Table

If multiprogramming is supported, this table is to be extended with one of the following fields (depending on the number of supported partitions), otherwise the table ends here.

Displacement	Label	Description	
44-47		Address of F1 Job Accounting Table	NPARTS = 2
48-51	ACCTSEAS	Control Field: prevents the accounting routine being active in more than one partition simultaneously	

Displacement	Label	Description	
44-47		Address of F2 Job Accounting Table	NPARTS = 3
48-51		Address of F1 Job Accounting Table	
52-57	ACCTSEAS	Control Field: prevents the accounting routine being active in more than one partition simultaneously	

Displacement	Label	Description	
44-47		Address of F3 Job Accounting Table	NPARTS = 4
48-51		Address of F2 Job Accounting Table	
52-55		Address of F1 Job Accounting Table	
56-63	ACCTSEAS	Control Field: prevents the accounting routine being active in more than one partition simultaneously	

Displacement	Label	Description	
44-47		Address of F4 Job Accounting Table	NPARTS = 5
48-51		Address of F3 Job Accounting Table	
52-55		Address of F2 Job Accounting Table	
56-59		Address of F1 Job Accounting Table	
60-69	ACCTSEAS	Control Field: prevents the accounting routine being active in more than one partition simultaneously	

*) These values are the same as the PIK values for the relevant tasks

Figure H-2. Explanation of the contents of the Job Accounting common table.

Appendix H

TABLES USED BY JOB ACCOUNTING

Programming considerations

The user program for processing the information entered by the supervisor in the Job Accounting Table must be cataloged and be self relocating with the name $JOBACCT in a core image library. For efficiency, an overlay structure should be avoided, and the length of the program should preferably not exceed one core image library block.

Because $JOBACCT is called in at the end of each job step, it should perform only data gathering and recording, but not data reduction and formatting if additional system overhead is to be held to a minimum. Overhead depends largely upon the efficiency of $JOBACCT. The optional SIO accounting (JA=n1, n2, n3) also causes additional overhead.

LIOCS uses registers 13-15. If $JOBACCT needs any of these registers after a LIOCS function has been performed, save and restore the desired registers (register 14 should always be saved when using LIOCS because it is necessary to return to job control via the instruction BR 14). Chapter 9 in this section describes the usage of the general registers by system control programming and job accounting.

If $JOBACCT uses LIOCS, it should save at least part of the DTF information (status switches, extent information, and pointers) in the user save area. If more than one DTF is used, information from each should be saved. The user save area may be used to save any type of information as well as to accumulate step to step statistics for end job accounting. This accumulation reduces the rate of scheduled output records caused by writing a step accounting record for each job step. The user save area is not accessed by system functions. Chapter 12 in this section describes the save areas and the system generation macro JALIOCS.

If an error causes $JOBACCT to be canceled, $JOBACCT is not called again until the system is re-IPLed. "JOB ACCT" appears in the cancel message, and the problem program name appear in the EOJ message. The STXIT option may be used to pass a message informing the operator that an error occured in $JOBACCT rather than in the problem program. (A description of tables used by user exit routines can be found in Section 4 of this manual, Chapter 10.) The job in that partition is terminated and normal processing continues with the next job.

Refer to *DOS/VS System Management Guide* for details on writing job accounting routines.

SERVICEABILITY AIDS FOR POWER AND POWER RJE

This appendix describes the serviceability aids provided by IBM to a system programmer who suspects an error in the POWER or the POWER RJE (system control program).

For a detailed description of the internal logic of the POWER and POWER RJE program refer to the POWER logic and POWER RJE logic manuals.

Eliminating the components

In the event of a suspected error in the POWER/POWER RJE program, the programs running under the control of POWER/POWER RJE should be executed without POWER/POWER RJE. That is, execute the programs under direct control of DOS/VS. In order to eliminate the POWER RJE component it may be necessary to submit your program to the central station and request it to be run without POWER RJE.

If the identical error occurs, it is not caused by the POWER/POWER RJE SCP, and the failing program must be debugged. If the error does not occur when running under direct control of DOS/VS, it is possible that an error exists in the POWER/POWER RJE SCP. Whichever is the case, the flowcharts in part 1 of Section 4 should be consulted. These charts indicate which serviceability aid to use and recommend a method for analyzing the system information obtained.

TRACE and DUMP Monitoring.

If the identical error occurs, it can not therefore be caused by the POWER/POWER RJE SCP, and the failing program must be debugged. If the error does not occur when running under direct control of DOS/VS, it is possible that an error exists in the POWER/POWER RJE SCP. Whichever is the case, the flowcharts in part 1 of Section 4 should be consulted. These charts indicate which serviceability aid to use and recommend a method for analyzing the system information obtained.

TRACE: Monitoring Tasks and SVCs.

Tasks and/or SVC zero (SVC 00) are monitored if the operator issues an appropriate Z command.

$$\text{Z TRACE,} \left\{ \begin{array}{l} \text{SVC [,TSK]} \\ \text{TSK} \end{array} \right\}$$

Trace information is collected in a log table at the end of phase FGPTYPZ3, which is loaded for execution into a program buffer. To locate the log table in a dump of virtual storage, scan the two translated dump columns for the constant LOGADDR=.

Figure J.1 shows the layout of the log table. The status of the trace can be displayed on the console by entering the command:

Z TRACE.LST

Figure J.1. The format and contents of the SVC and TSK log table.

Appendix J

SERVICEABILITY
AIDS FOR POWER
AND POWER RJE

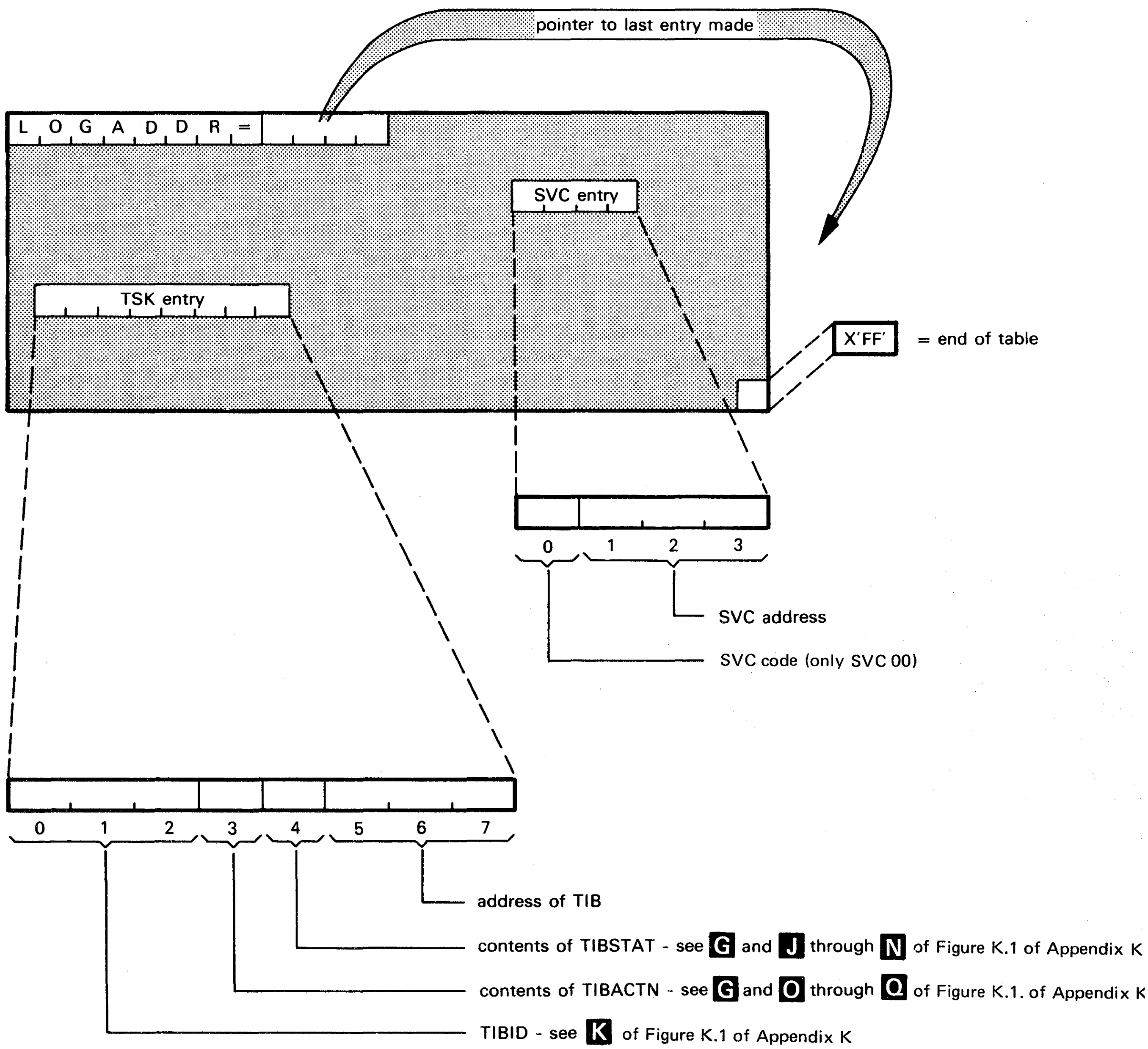


Figure J.1. **The format and contents of the SVC and TSK log table.**

DUMP: Monitoring TIBs, QFILE Records and DATAFIL Records

The operator command whose format is shown below causes pointers contained in execution processor TIBs, in QFILE records, or in DATAFIL records to be monitored. The format of the command is:

$$\text{Z DUMP,}\left\{\begin{array}{l}\text{TIB}\\ \text{QFL}\ \ [\text{,TIB}]\ \ [\text{,DFL}]\\ \text{DFL}\end{array}\right\}$$

When TIB is specified, POWER/POWER RJE monitors the first and last queue pointers for all POWER/POWER RJE queues. The TIB fields containing these pointers are:

- TIBRDRQF = pointer to the first JCT in the partition's reader queue
- TIBRDRQL = pointer to the last JCT in the partition's reader queue
- TIBPRTQF = pointer to the first JCT in the partition's print queue
- TIBPRTQL = pointer to the last JCT in the partition's print queue
- TIB PUNQF = pointers to the first JCT in the partition's punch queue
- TIBPUNQL = pointers to the last JCT in the partition's punch queue
- TIBPURGF = pointers to the first JCT in the partition's purge queue
- TIBPURGB = pointers to the last JCT in the partition's purge queue.

POWER/POWER RJE verifies that these pointers, which are in the format MCHR, are valid disk addresses within the QFILE extent.

When QFL is specified, POWER/POWER RJE monitors:
- in the QFILE master record, the first and last queue pointers for the reader, print, punch, and purge queues of all partitions
- in the JCT being processed, the forward and backward chaining pointers.

Refer to Appendix M for a description of the QFILE.

POWER/POWER RJE verifies that these pointers are valid disk addresses within the QFILE extent and that no pointer in the QFILE master record appears on more than one queue.

When DFL is specified, POWER/POWER RJE verifies that the DATAFIL record chain-pointer contains a valid disk address within the DATAFIL extent.

When POWER/POWER RJE finds an invalid disk address, it causes an appropriate message to be displayed on the console.

Contents of Registers

An analysis of the register contents at the time of a dump occasionally provides a clue to the cause of the dump. Refer to Chapter 9 in Section 4 for a description of general usage for POWER and POWER RJE.

The BIM (BTAM Interface Module) Simulator

This is a software feature of the POWER RJE that simulates the function of the BTAM module. The simulator replaces the BIM for one RJE task in simulation mode, and any TPIO request from the task is serviced by the simulator on local unit record devices. All initialization procedures are performed for the task as if it were not in simulation mode. Some of these procedures are reversed to accommodate the simulator. In particular, where the dynamic assignment is made for the line, the line is unassigned to permit the simulator to use the CCB in the Line Control Block for the card reader. The dynamic assignment is made in accordance with the specified card reader in the special start command. The print and punch functions are performed on the SYSLST and SYSPCH unit record devices assigned by job control during POWER initiation. By using the BIM simulator, execution of RJE task overlays may be divorced completely from any line activity, including BTAM execution.

Appendix K

ANALYZING A DUMP OF THE POWER PARTITION

A method of locating POWER control blocks and areas is shown in a series of illustrations in Figure K.1. part 1 through 9. Notes within these illustrations provide information that will help you in determining the status of the various POWER routines at the time the dump was taken in analyzing the contents of specific areas of bytes.

The reference table below is provided to help you to locate descriptions shown in the following nine illustrations.

Block or area to be located	Reference
The POWER partition (if it was active at the time the dump was taken)	A
POWER control blocks:	
GENSW (generation option table and switch)	D
parameter list table	D R
DATABLKS and PROGBLKS table	R
DCTs (device control tables)	F
TIBTAB	C E
TIB	E G-Q
3540 control block	S
Program buffers	S
Data buffers	H

A This area (described in Section 4 of this manual, Chapter 11) is neither used nor altered by the POWER program. If POWER was active immediately before the dump was taken, the area can be located as follows:

If the MAP command was executed and is available on the SYSLOG output, the name of the job that initiated POWER will be printed together with the partition identifier and upper and lower addresses for that partition. The lower address is the address of the partition save area of the POWER partition.
(Note that the address values printed by the MAP command are in decimal.)
An alternative method is described below:

1. Locate the BOUNDARY BOX in the dump as described in this manual. (Refer to Chapter 12 Section 4.)
2. Make a note of the upper and lower partition addresses contained in the BOUNDARY BOX.
3. Locate the PIB2 in the dump as described in this manual. (Refer to Chapter 7 Section 4.)
   The PIB2 entry belonging to the partition containing POWER will have bit 0 of byte 15 set to 1 (X'80'), and the PIB2 entry belonging to partitions supported by POWER will have bit 1 of byte 15 set to 1 (X'40')
4. From this information and the contents of the BOUNDARY BOX you can locate the address of the partition save area of the partition containing POWER and of the partition supported by POWER.

Note: If POWER was assembled with a unique name for the POWER generation macro, you can look up that name in the left of the translated dump columns. Otherwise the name is FGPSPOOL.

B The last eight bytes contain the name of the operator communication transient phase that has last been executing in the area. This name has the format

FGPTYPxx

where xx = phase identifiers.

Examples of phase names: FGPTYPS0 or FGPTYPL1.

The address of the name can be calculated by adding X'400' to the address of the last byte of the partition save area. (The partition save area is either X'58' or X'78' bytes in length; refer to Section 4 in this manual, Chapter 11 for a description.) Subtract 8 from the result.

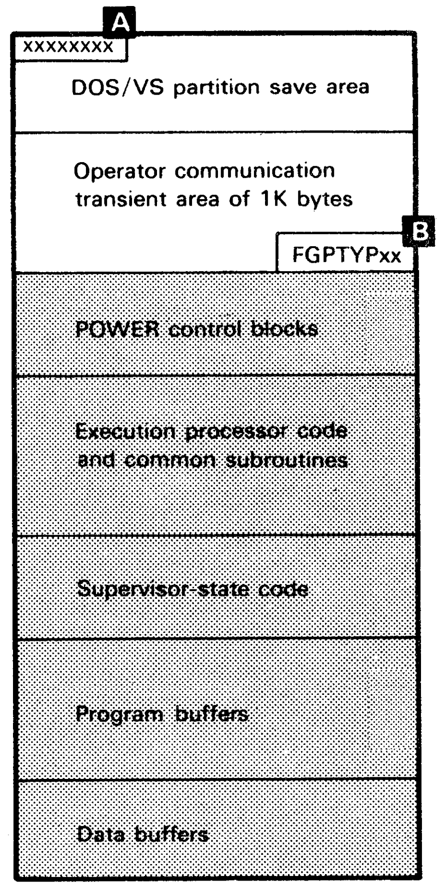
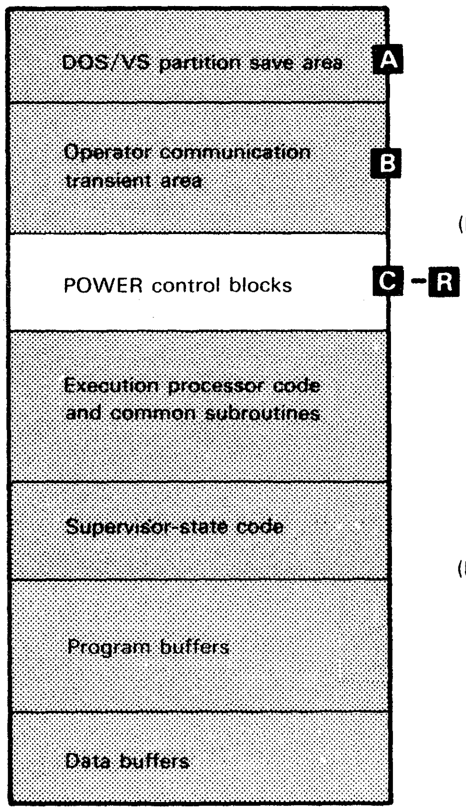
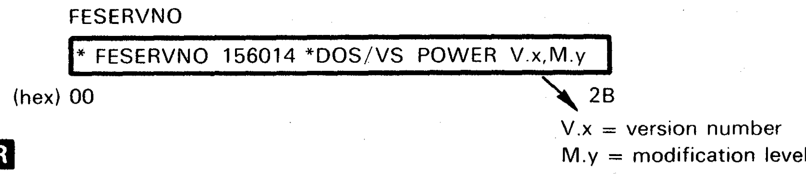


Figure K.1, part 1 of 9

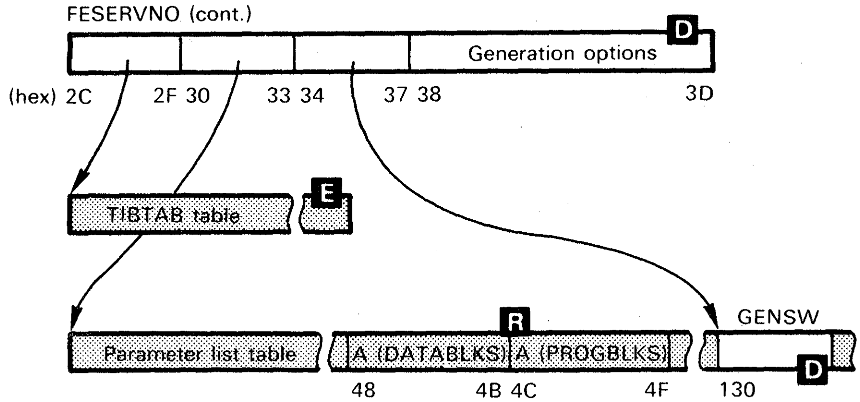
ANALYZING A DUMP OF THE POWER PARTITION



C The following helps you locate the POWER control blocks. For full details about the fields of the control blocks, see *Chapter 4. Data Areas*.



Locating this constant in the translated dump columns is a quick method of locating the beginning of the POWER control blocks area. FESERVNO (cont.)



D Analyze both the generation options and the generation option switch (GENSW) for the options specified.

Generation options

Byte	Meaning
38	number of tracks in a track group (TRACKGP)
39	number of DATAFIL disks allowed (NUMDDKS)
3A	maximum number of readers and writers that can be started at one time (MAXRW)
3B	maximum number of RJE tasks that can be started at one time (MAXRJS)
3C	maximum number of data buffers that the POWER program can manage (MAXBUFS)
3D	the number of CCB's that can be used for unit record or tape I/O (MAXCCB)

GENSW

Byte	Bit	Meaning if 1
131	0	autostart for a reader task
	1	autostart for a print writer task
	2	autostart for a punch writer task
	3	DIAG=YES
	4	RJE=YES
	5	reserved
	6	READER=YES
	7	READER=YES
130	0	TAPE=YES
	1	RDRCLOS=EXIT
	2	ACCOUNT=YES
	3	RJEND=YES
	4	JOBSEP=YES
	5-7	reserved

Figure K.1, part 2 of 9

E Examine the TIBTAB to determine which tasks are active and which are ready to receive control.

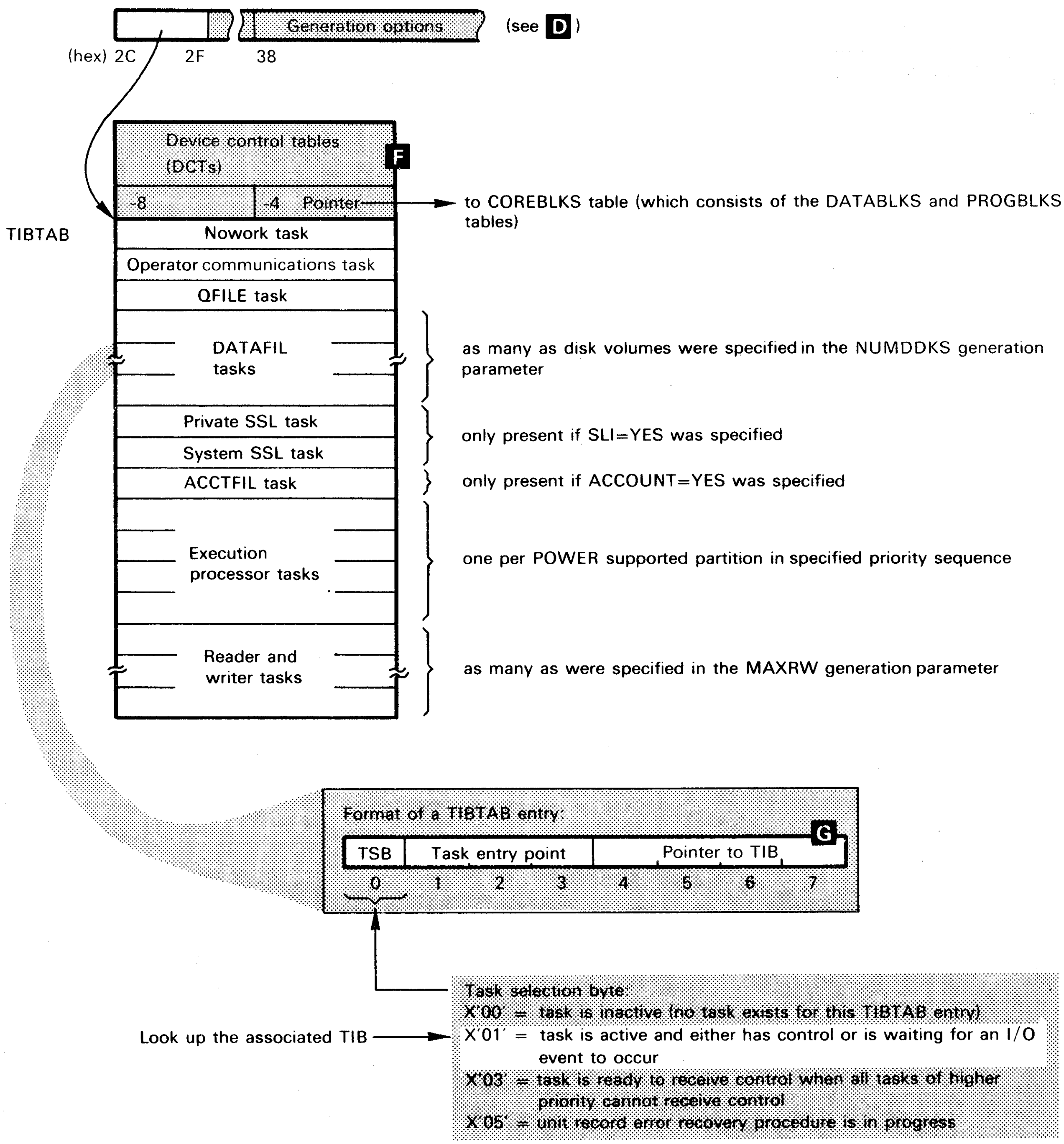
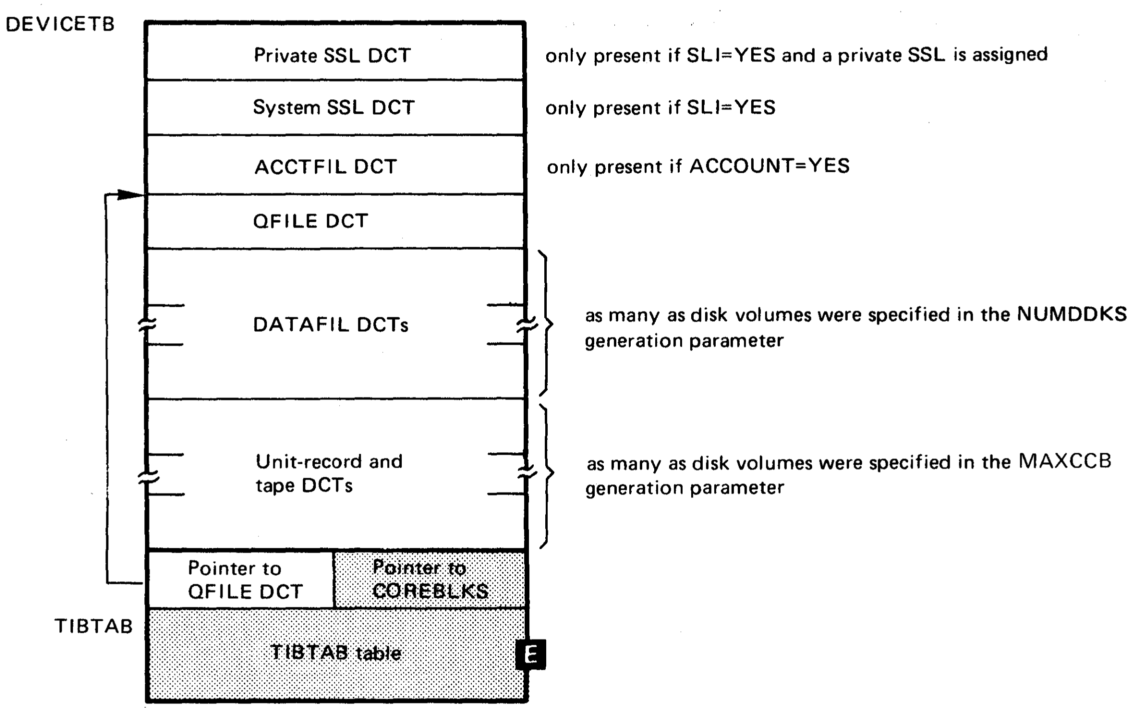


Figure K.1, part 3 of 9

ANALYZING A DUMP OF THE POWER PARTITION

F Locate DCTs and analyze access queues:



The following is an illustration of a DCT access queue with three TIBs waiting to do I/O to the same DCT:

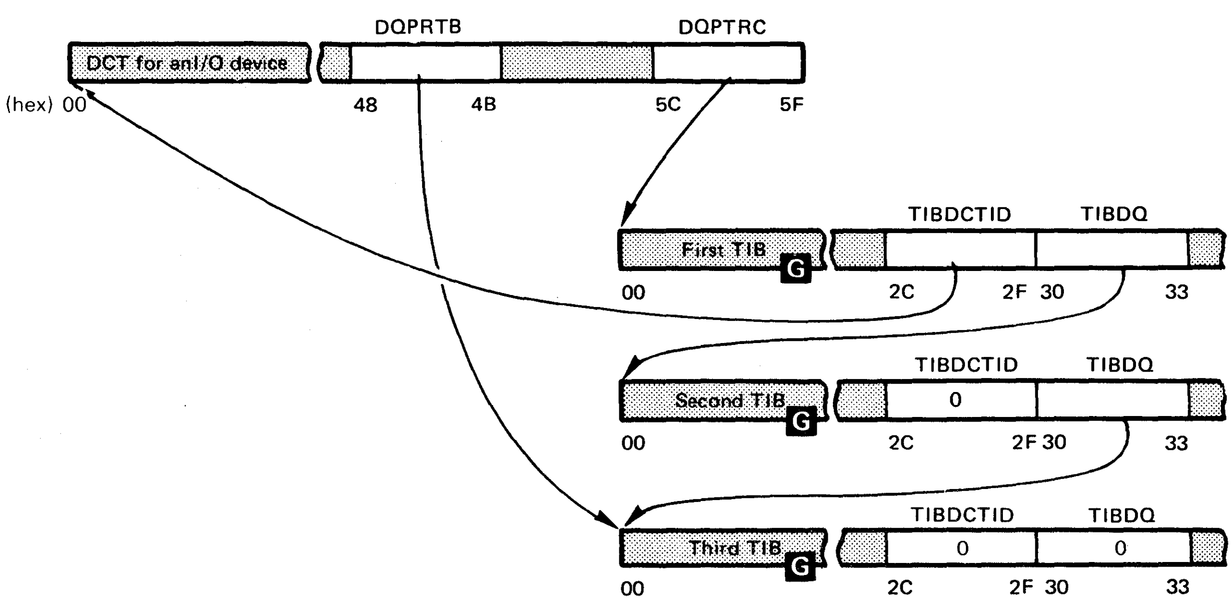
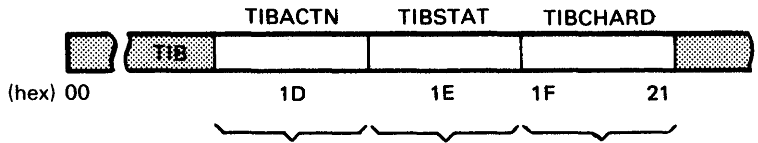


Figure K.1, part 4 of 9

G Except for the TIBs for reader and writer tasks, all TIBs are located in the POWER control blocks area of the POWER partition. The TIB for a reader or writer task is located in the program buffers (see **S**) allocated to the task.



TIBCHARD: Identifies the type of the associated task:

$OP = operator communication task

ƀpp = execution processor task for the indicated partition:see **H**

ppP = punch writer task for the pp partition
ppR = reader task for the pp partition
ppT = print writer task for the pp partition

XXP = partition-independent punch writer task
XXR = partition-independent reader task
XXT = partition-independent print writer task

where pp = partition identification: BG for background, F1 through F4 for foreground 1 through 4

TIBSTAT: Indicates the action performed by the task when the error occurred:

For an execution processor TIB: see **J**

For a reader TIB: see **K**

For a print writer TIB: see **L**

For a punch writer TIB: see **M**

For the operator communication TIB: see **N**

TIBACTN: Indicates why the task gave up the CPU or, if it has control of the CPU, that the last I/O performed was completed:

For an execution processor TIB: see **O**

For a reader TIB: see **P**

For a print writer TIB: see **P**

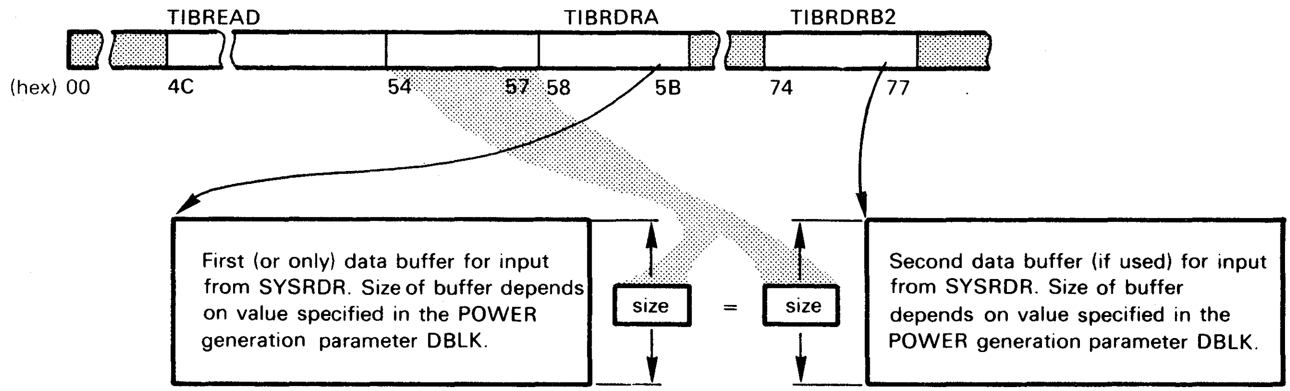
For a punch writer TIB: see **P**

For the operator communication TIB: see **Q**

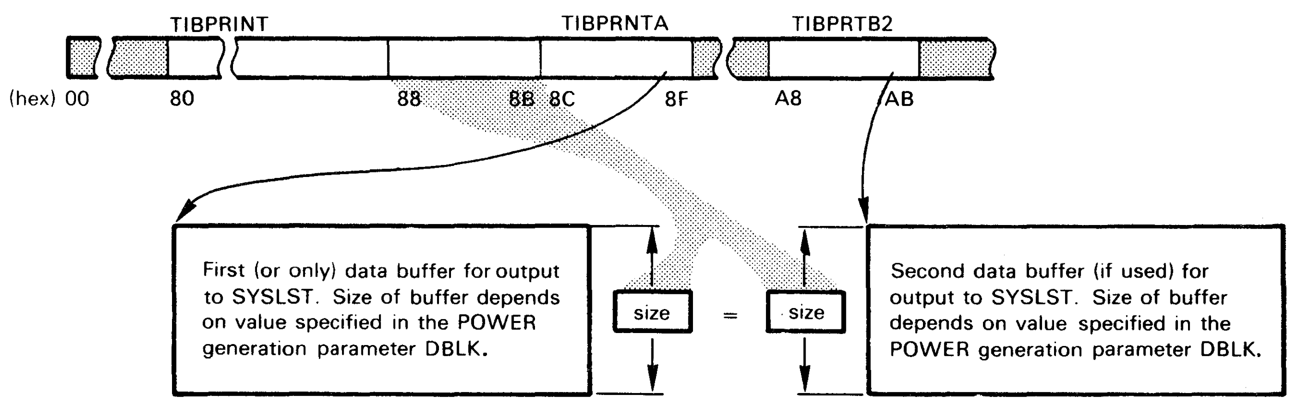
Figure K.1, part 5 of 9.

H Execution processor TIB. To locate an execution processor TIB see **E**

Input from SYSRDR:



Output to SYSLST:



Output to SYSPCH:

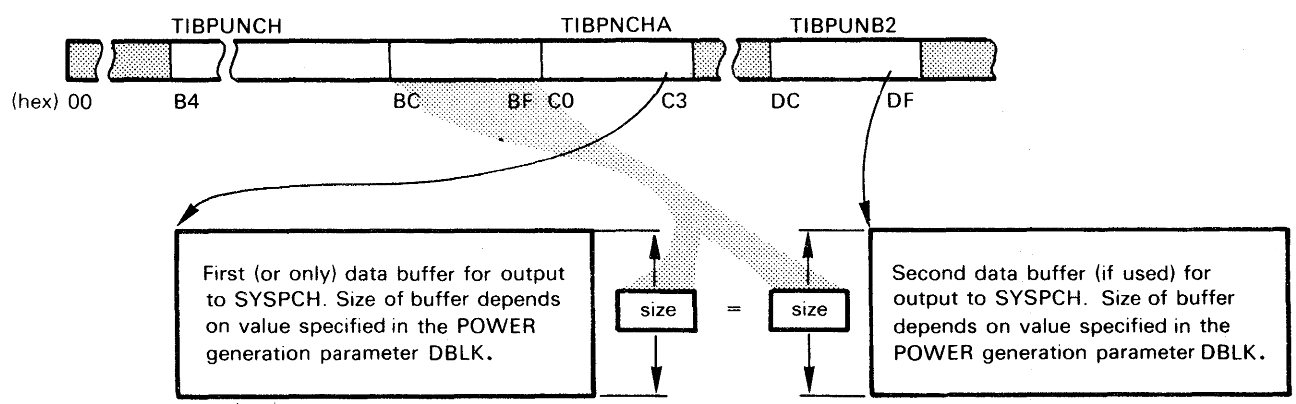


Figure K.1, part 6 of 9.

TIBSTAT of an execution processor TIB:

Bit	Meaning
Ø	1 = SYSLST output for the current job entry will not be intercepted.
1	reserved.
2	1 = SYSPCH output for the current job entry will not be intercepted.
3	1 = end-of-job processing for the current job entry takes place.
4	1 = printed output is being intercepted.
5	reserved.
6	1 = punched output is being intercepted.
7	1 = a job is being executed.

K TIBSTAT of a reader TIB:

Bit	Meaning
Ø	1 = task is going to be terminated.
1	reserved.
2	1 = Sequence of DISKETTE volumes is to be checked.
3	reserved.
4	1 = task is reading from a 3450.
5	1 = both a card reader and a 3540 are assigned.
6	reserved.
7	1 = task is to be cancelled.

L TIBSTAT of a print writer TIB:

Bit	Meaning
Ø-1	1Ø = a stop command has been issued for the task. Ø1 = a flush command for the current job has been issued for the task. 11 = a restart command for the current job has been issued for the task.
2	1 = a flush all command has been issued for the task.
3	1 = the writer has completed output for the current job.
4	1 = restart command directs the task to start from the beginning of the current job.
5	reserved.
6	1 = indicates to a tape writer that job separator pages are required.
7	1 = a cancel command has been issued for the task

Figure K.1, part 7 of 9.

M TIBSTAT of a punch writer TIB:

Bit	Meaning
Ø	1 = a stop command has been issued for the task.
1	1 = a flush command for the current job has been issued for the task.
2	1 = a flush all command has been issued for the task.
3	1 = the writer has completed output for the current job.
4-6	reserved.
7	1 = a cancel command has been issued for the task.

N TIBSTAT of the operator communications task TIB:

Bit	Meaning
0-6	Reserved.
7	POWER command is entered while a previous command is being executed.

O TIBACTN of an execution processor TIB:

X'ØØ' = disk I/O was in progress or just completed if the task was in control of the CPU.
X'Ø4' = the contents of a print or punch buffer are to be printed or punched, respectively.
X'Ø8' = a POWER EOJ card is to be processed.
X'ØC' = an input card is to be read.
X'1Ø' = a PRT card is to be processed.
X'14' = a PUN card is to be processed.
X'18' = a message is being written on the console.
X'1C' = either a JOB card (writer-only system) or an SLI card (reader system) is to be processed.
X'2Ø' = a card from the Source Statement library is to be processed.

P TIBACTN of a reader or writer TIB:

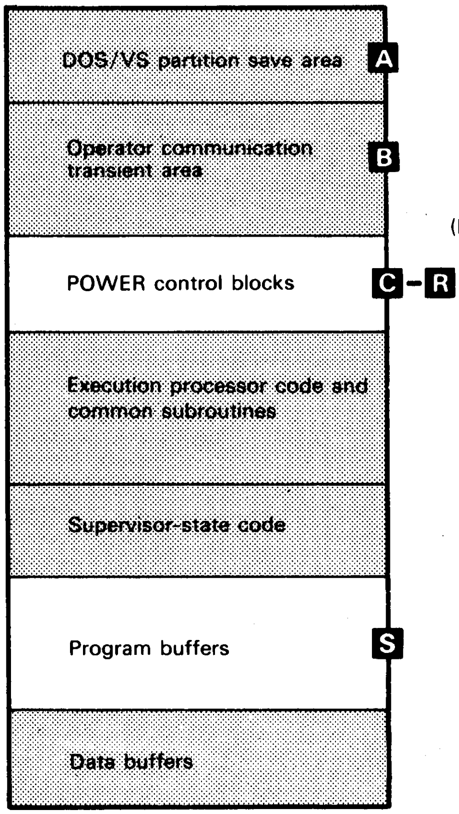
X'ØØ' = disk I/O was in progress or just completed if the task was in control of the CPU.
X'Ø4' = the task waits for work. This code is set by the execution processor for the same partition:
— for a reader task when all card input for a job is complete
— for a writer task when all print (or punch) output for a job is complete.
X'Ø8' = The task waits for completion of
— a read operation on the associated card reader if a reader task.
— a print operation on the associated printer if a print writer task.
— a punch operation on the associated card punch if a punch writer task.
X'18' — The task has issued a message to the console and waits for completion of the typing operation.

Q TIBACTN of the operator communication TIB:

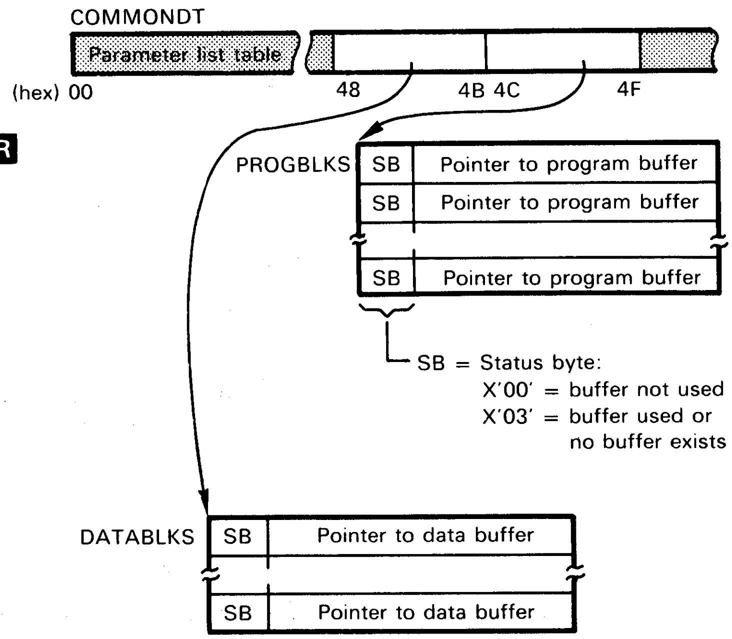
X'ØØ' = disk I/O was in progress or just completed if the task was in control of the CPU.
X'Ø4' = The task is inactive, that is, all commands have been processed.
X'08' = the task has received a command and has initiated processing of same.
X'1Ø' = I/O for the ACCTFIL is in progress.
X'18' = the task has initialized writing a message on the console and waits for completion of the typing operation.

Note: For POWER start up, the TIBACTN code has been assembled as X'Ø8' and the TSB of the associated TIBTAB entry has been assembled at X'Ø3'.

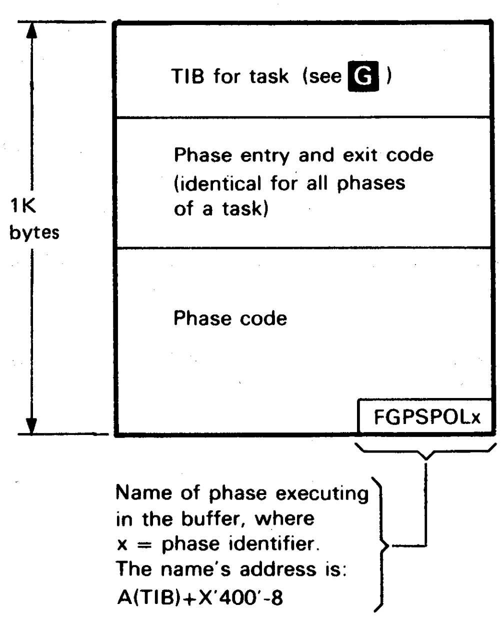
Figure K.1, part 8 of 9.



R Locate the program and data buffers by means of pointers in the parameter list table and in the PROGBLKS and DATABLKS tables:



S Program buffers are generated only for reader and writer tasks. The number of program buffers generated is equal to the number specified in the MAXRW generation parameter. The format of a program buffer as shown below in the illustration on the left. The illustration on the right shows that two program buffers are acquired by a reader task using the 3540 as input reader.



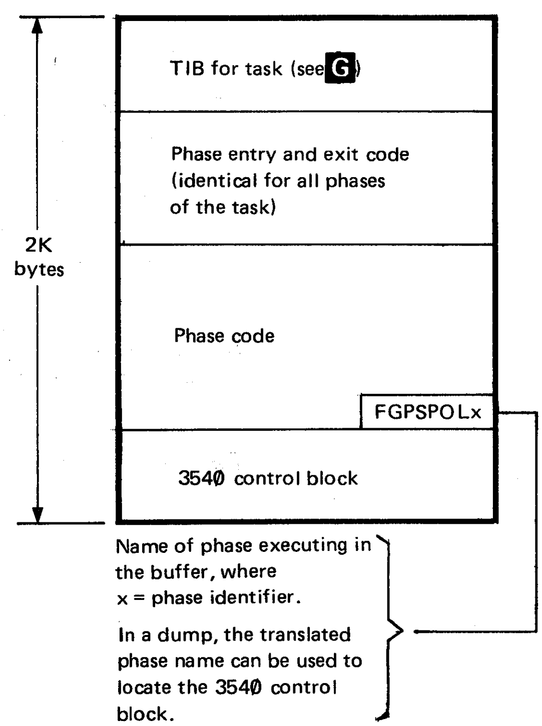


Figure K.1, part 9 of 9.

Appendix L

ANALYZING A DUMP OF THE POWER RJE PARTITION

The method of locating POWER/RJE control blocks and areas is shown in a series of illustrations in Figure L.1, parts 1 through 14. Notes within this figure provide information that will help you in determining the status a POWER/RJE task had at the time the dump was taken and in analyzing the contents of specific areas or bytes.

The table below assumes that you have successfully located the POWER partition either (1) by tracing its beginning from the pointer in the active partition to the partition's save area or (2) by locating in the translated dump column the name you have used for the generation macro at the time of POWER/RJE assembly.

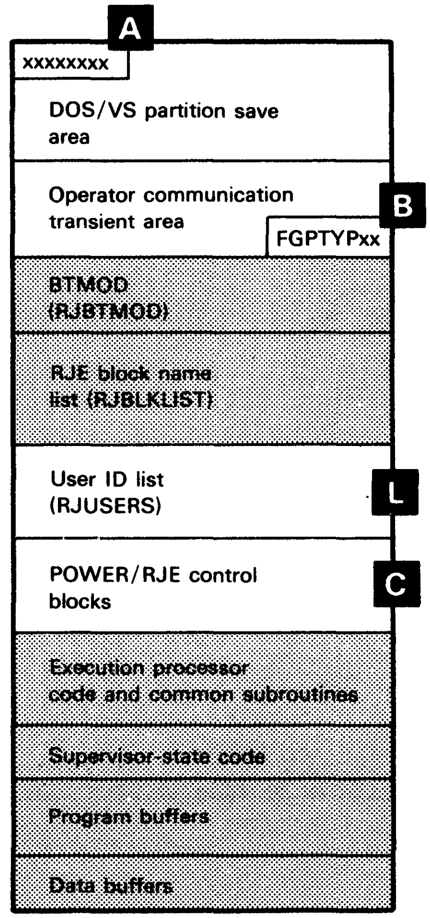
The table does not include the steps that must be taken in order to locate normal POWER control blocks, such as an execution processor TIB or a reader or writer TIB, nor does it include information about how to analyze these POWER control blocks. For this information refer to Appendix K or the DOS/VS POWER Program Logic Manual.

The reference table below is provided as a help in locating the information which you need to find a specific block or area in a dump. In the illustration, always look for the given reference to the left of text.

Block or area to be located	Reference to the Illustrations
CCB (in DTFBT)	M
DTFBT	M
DECB (data event control block)	J
Generation options table	D
Parameter list table	D
RJEBLK (RJE block)	H
RJBLKLST (RJE block name list)	K
RJE TIB (RJE task information block)	G
RJLIST (RJE active task list)	F
RJUSERS (RJE list of authorized users)	L
TIBTAB (task information block table)	D

ANALYZING A DUMP OF THE POWER RJE PARTITION

A If the POWER/RJE program was assembled with a unique name for the generation macro you can look up that name in the left of the translated dump columns.



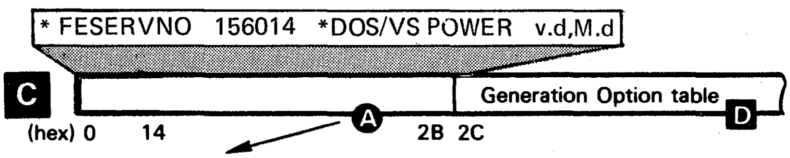
B A 1K area used for the execution of the operator communication transient phases. The last eight bytes contain the name of the phase that has been executing in the area. This name has the format

FGPTYPxx
where xx = phase identifiers

The address of the name is:

$$\text{A (partition save area)} + \left\{ \begin{matrix} X'78' \\ X'58' \end{matrix} \right\} + X'400'\text{-}8$$

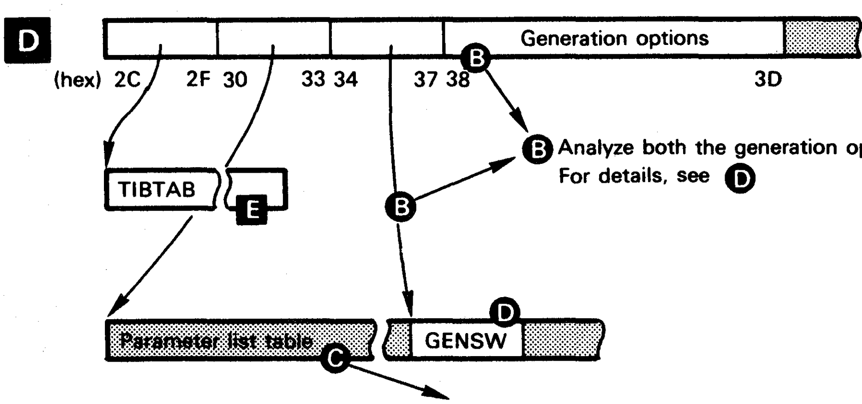
C



A V.d. = program version number
M.d. = program modification number

Locating this constant in the translated dump columns is a quick method of locating the beginning of the POWER/RJE control blocks area.

D



B Analyze both the generation option switch and the options in the table. For details, see **D**

C For full details about the fields of the table, see the "Data Area" section. For important switches, see **D**

Figure L.1, part 1 of 14

ANALYZING A DUMP
OF THE POWER RJE
PARTITION

D continued

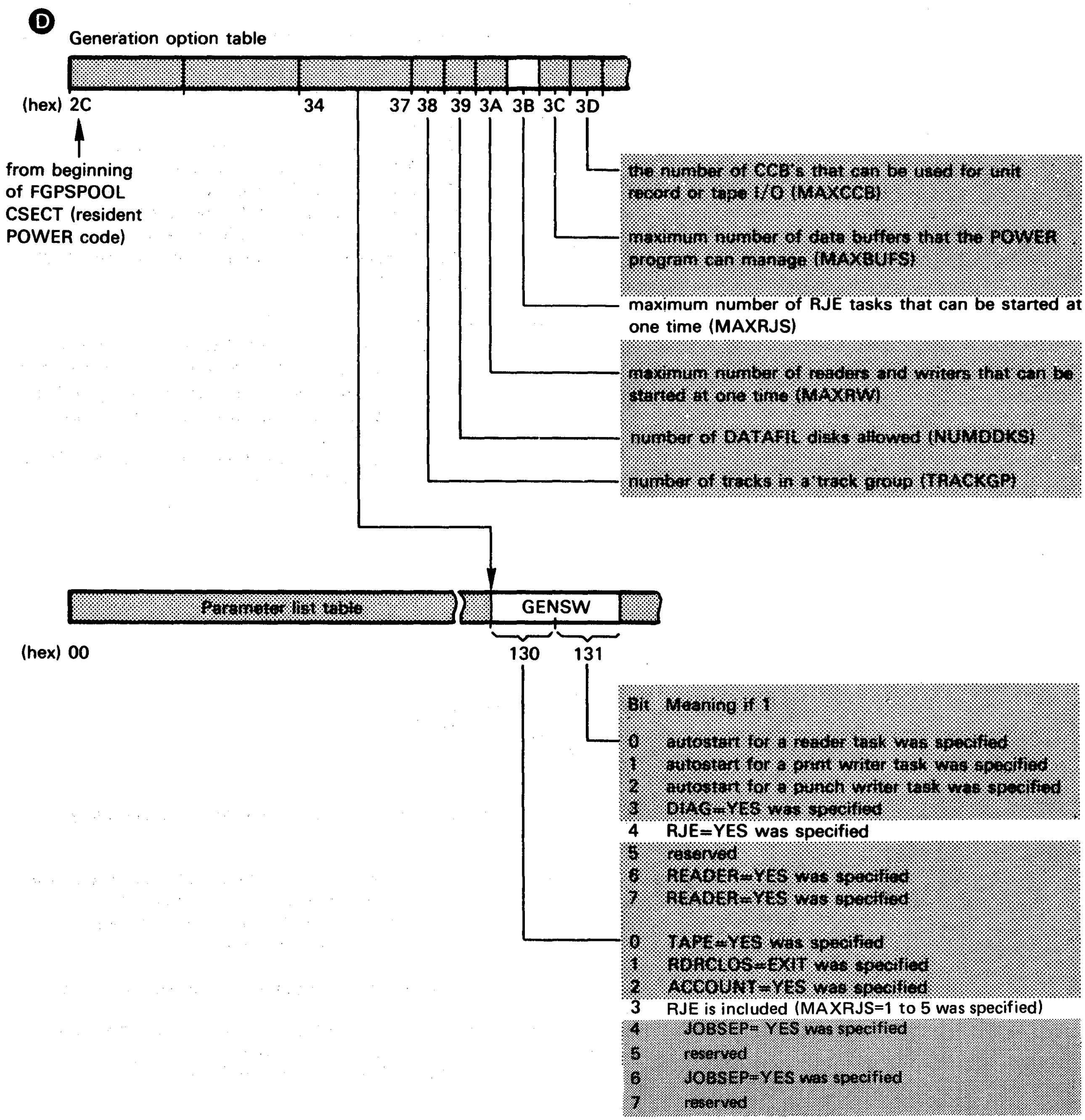
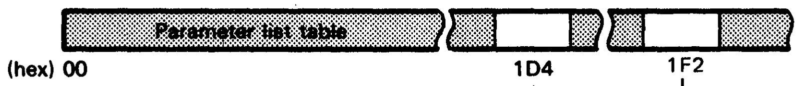


Figure L.1, part 2 of 14

D continued

D continued



TYPEIN+X'1E': indicates buffers obtained for RJE task initiation

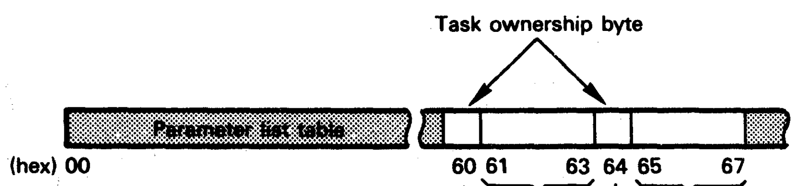
Bits Meaning if 1

0-2 reserved
3 BIM's turnaroud data buffer obtained
4 BIM program buffers obtained
5 task s data buffer obtained
6 RJEBLK program buffer obtained
7 RJE task program buffer obtained

Note: These switches are used only by FGPTYPS1 and FGPTYPS2

TYPEIN:
X'00' = phase FGPTYPB0 has finished executing
X'FF' = an additional track group must be obtained for RJE messages, and the tracks of that group must be initialized as free message records.

Note: This switch is used only by phases FGPTYPB0, FGTYPB1, and FGPTYPB2.



address of the BIM simulator if the preceding byte is ≠ X'00'

X'00' = the BIM simulator is not in storage and the next three bytes are insignificant
not X'00' = the BIM simulator is in storage (see Note below)

address of BIM if the preceding byte is ≠ X'00'

X'00' = the BIM is not in storage and the next three bytes are insignificant
not X'00' = the BIM is in storage

Note: Each time a task requests the services of the BIM (or BIM simulator), the contents of the task ownership byte are shifted to the left by one bit position and bit 7 of the byte is set to 1. Each time a task finished using the BIM (or BIM simulator), the contents of the task ownership byte are shifted to the right by one bit position and bit 0 of the byte is set to 0. A copy of the byte's bit configuration for a specific task is contained in the RJBLKLST entry for that task (see K).

Figure L.1, part 3 of 14

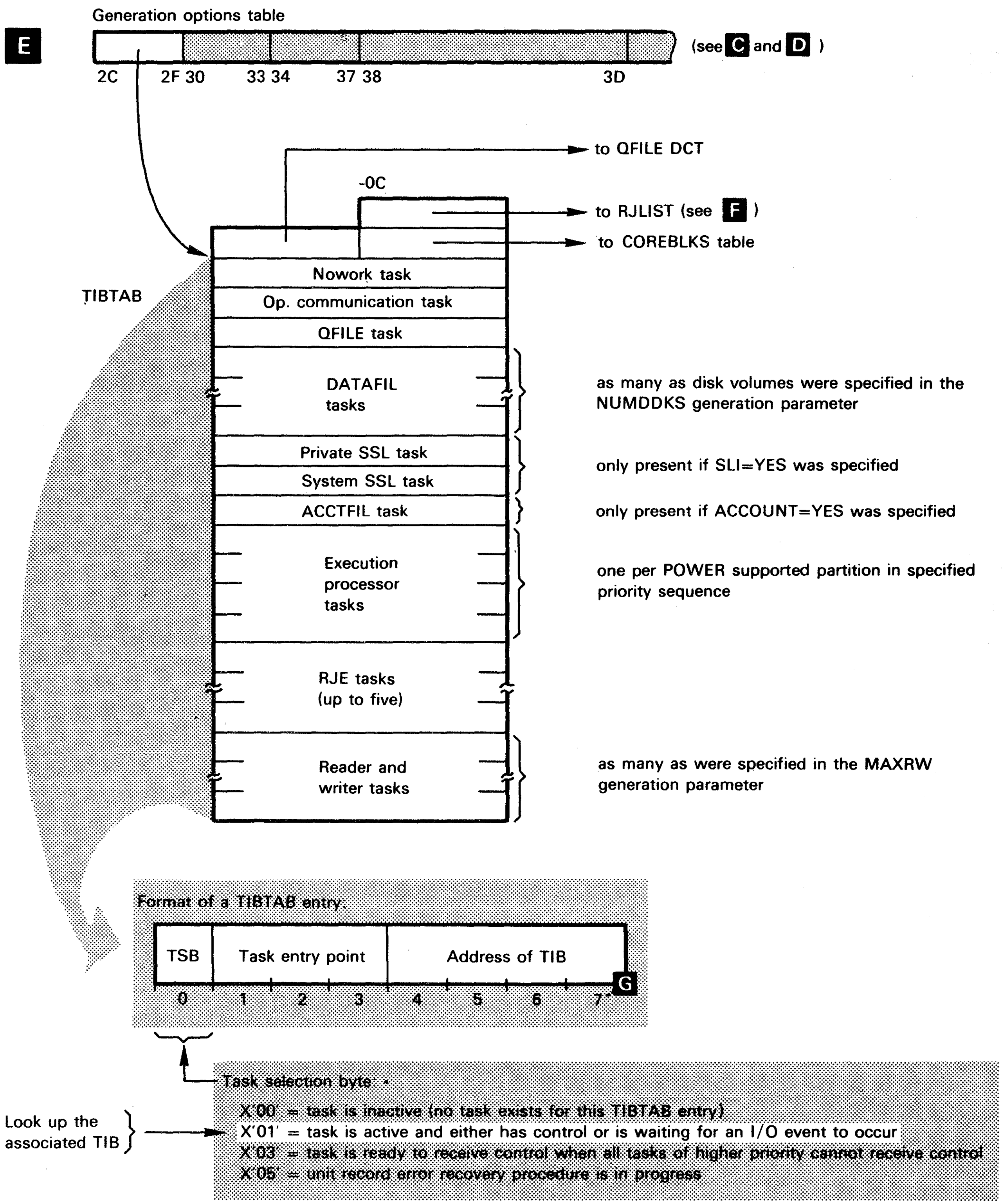


Figure L.1, part 4 of 14

F TIBTAB-0C (see **E**)

E only if the task is active; set to F'0' otherwise

eight bytes

RJLIST*

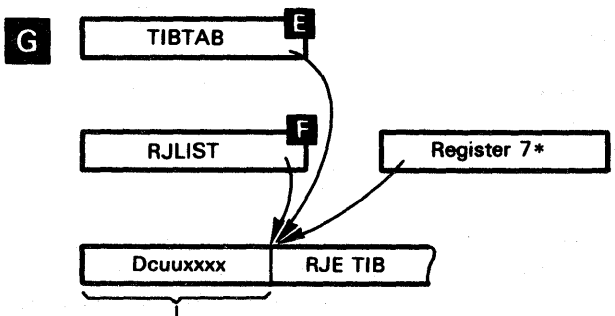
<table>
<tr><td colspan="2">Address of DECB **G**</td><td colspan="2">Address of TIB **E**</td></tr>
<tr><td colspan="2">Address of DECB</td><td colspan="2">Address of TIB</td></tr>
<tr><td colspan="2">Address of DECB</td><td colspan="2">Address of TIB</td></tr>
<tr><td colspan="2">Address of DECB</td><td colspan="2">Address of TIB</td></tr>
<tr><td colspan="2">Address of DECB</td><td colspan="2">Address of TIB</td></tr>
<tr><td>X'FF'</td><td>Reserved</td><td>n</td><td>Address of 2780 BIM</td></tr>
<tr><td>n</td><td>Address of BIM SIM</td><td colspan="2">Address of RJBLKLST</td></tr>
<tr><td colspan="2">Address of RJUSERS</td><td colspan="2">Address of RJBTMOD</td></tr>
</table>

X'FF' = indicates the end of the list

* contains as many entries as RJE tasks were specified in the MAXR

Figure L.1, part 5 of 14

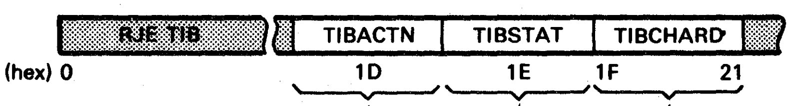
ANALYZING A DUMP OF THE POWER RJE PARTITION



* only if the associated task is active

name of RJE block that contains the RJE TIB (for further details about the RJE block, see **H**)

cuu = line address at the central station
xxxx = type of terminal used: 2770, 2780, or 3780



TIBCHARD: contains the characters RJE

TIBSTAT:

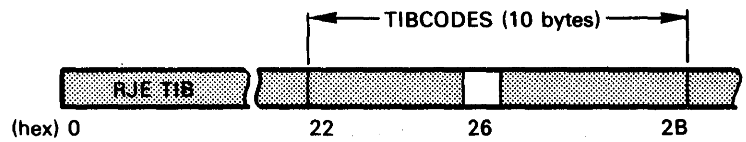
Bit	Meaning if 1
0	stop the task
1	flush the TP input and write a message to the work station operator
2	a TP input record is available (a TPIO READ cannot be issued)
3	(only if a 2780 is used) the first record in the data buffers has a space-immediate CCW and is preceded by a print-no-space CCW as the last record of the buffer used previosly
4-7	reserved

TIBACTN:

X'00' = disk I/O was in progress or just completed if the task was in control of the CPU
X'04' = the task waits for work
X'07' = a TP READ INITIAL has been issued
X'08' = a TIPO other than read initial has been issued
X'18' = the task has issued a message to the console and waits for completion of the typing operation

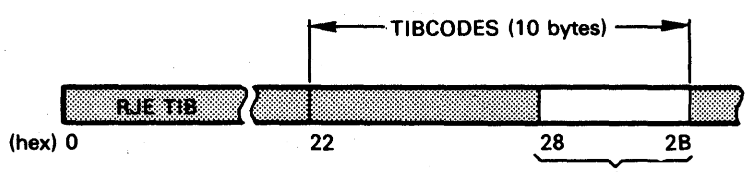
Figure L.1, part 6 of 14

G continued



TIBCODES+4: indicates the transfer of control between the OUTPUT work station command processor, the print overlays and the punch overlays:

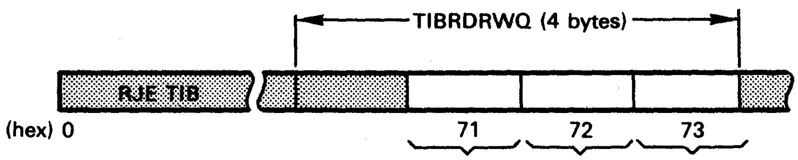
X'10' = print operations are in progress
X'14' = punch operations are in progress or to be processed for an ALLUSERS request
X'16' = a punch job is to be placed in the hold state
X'18' = punched output is available for an ALLUSERS request or a jobname request if the JCT userid field is ALLUSERS
X'1C' = the print overlays have transferred control directly to the punch overlays or punched output is available for userid for a jobname request
X'30' = the punch overlays have transferred control directly to the print overlays or printed output is available for userid for a jobname request
X'38' = printed output available for an ALLUSERS request or a jobname request if the JCT userid field is ALLUSERS
X'50' = a print job is to be placed in the hold state
X'60' = a print job is to be placed in local state for processing at a local output device
X'70' = a print job is to be placed in local state for processing at a local output device



TIBCODES+6: contains the address of the next TP input record or the data portion for TP output (this field is altered only by the BIM)

Figure L.1, part 7 of 14

G continued

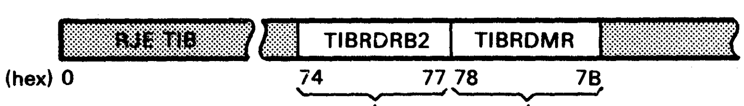


Bit Meaning if 1

Byte 73:
- 0-5 reserved
- 6 termination of reader overlay has been requested
- 7 reserved

Byte 72:
- 0-3 reserved
- 4 the first record read was a JOB statement
- 5 end-of-file on TP
- 6 request for a track group is to be forced
- 7 no track group found

Byte 71:
- 0-1 reserved
- 2 a DOS/VS job card was read
- 3 FGPPRJE4 was called by FGPPRJEY
- 4 an EOJ JECL statement is being moved to the data block
- 5 an input record is available (a TPIO READ cannot be issued)
- 6 the first record is in a disk buffer
- 7 a JOB JECL statement was read (JECL must control when the end of a job entry has been reached, not a DOS/VS /& statement).

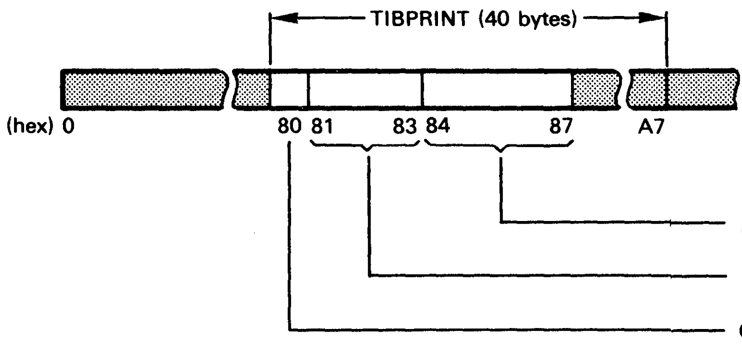


78-7B: pointer to next available position for a logical record in the current data block

74-77: pointer to DECB used by the associated task (see **J** for details)

Figure L.1, part 8 of 14

G continued



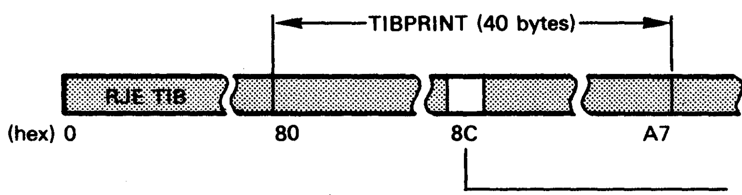
address of last BTAM I/O - used for retries

address of BIM being used

C'R' = the task is in read mode
C'W' = the task is in write mode

* X'00' = the switched line is disconnected
* X'02' = an EOT received (no line disconnect)
* X'04' = intervention is required at the work station
* X'40' = stop was requested when a TPIO macro was issued
* X'80' = an irrecoverable error has taken place
* X'81' = timeout has occurred at the work station and the user session is terminated

* settings that arise from TP exceptions when the task's TP exception routine is entered

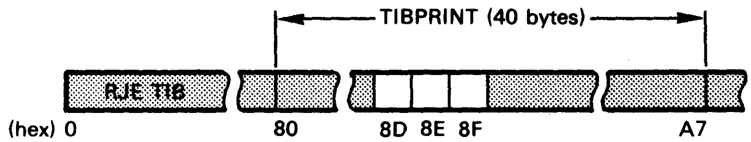


TIBPRINT+12: its bits are set by the TPCOMPL routine to indicate to the BIM that an exception has taken place:

Bits	Meaning if 1
0	TPCOMPL has been entered
1	an exception has taken place
2	stop was requested since last TPIO was issued
3-7	reserved

Figure L.1, part 9 of 14

G continued



TIBPRINT +15 indicates input and TP buffer status:

Bits Meaning if 1

0 last block from DATAFIL was used to fill the TP I/O area
1 input in TP I/O area was from a BTAM read
2 last print block is in TP I/O area
3 intervention required on print (EOT)
4 the end of a data block was reached when the TP area was last filled with input stored on disk
5 input in TP area is from the DATAFIL intermediate storage
6 input is being transferred to a data block during TP read-to-write line turnaround
7 input has been stored on the DATAFIL

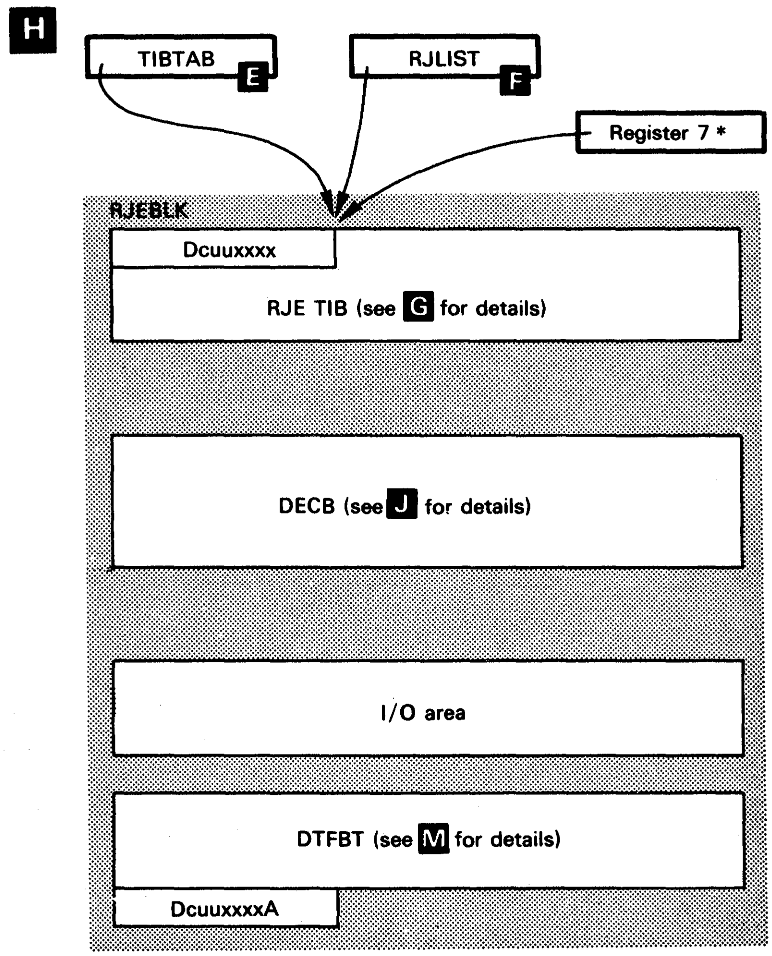
TIBPRINT+14 indicates the TPIO function requested when the BIM was last entered:

X'02' = read
X'04' = detach
X'05' = punch
X'06' = print
X'07' = flush the input and write to the work station operator
X'08' = write to the work station operator
X'09' = open the RJE task DTFBT
X'0A' = close the RJE task DTFBT

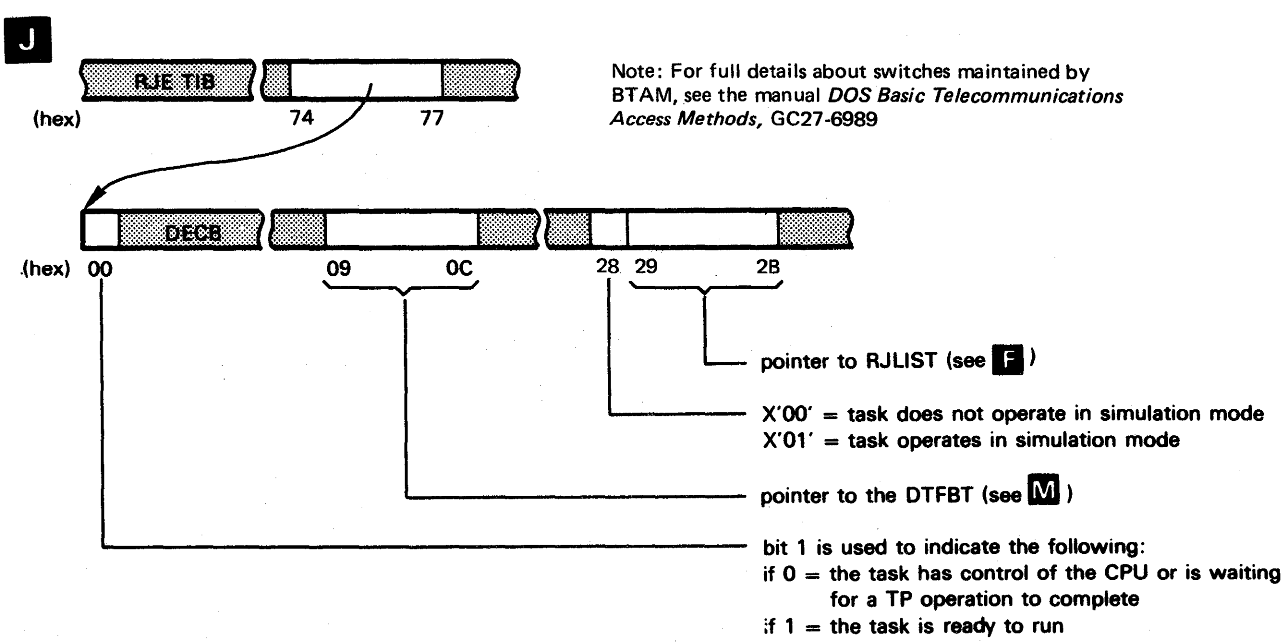
TIBPRINT+13 indicates line direction:

X'00' = line is not initialized or is in control mode
X'01' = CPU is receiving
X'02' = CPU is transmitting punch records
X'04' = CPU is transmitting print records
X'08' = CPU is transmitting a message

Figure L.1, part 10 of 14



* only if the associated task is active

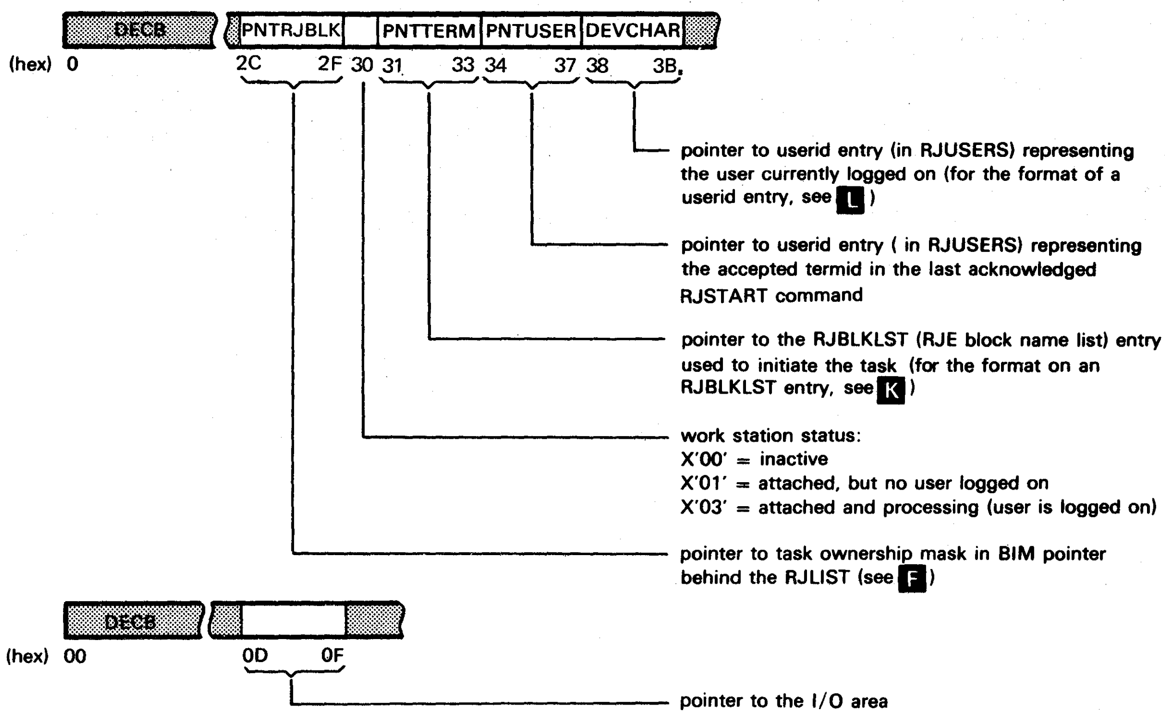


Note: For full details about switches maintained by BTAM, see the manual *DOS Basic Telecommunications Access Methods,* GC27-6989

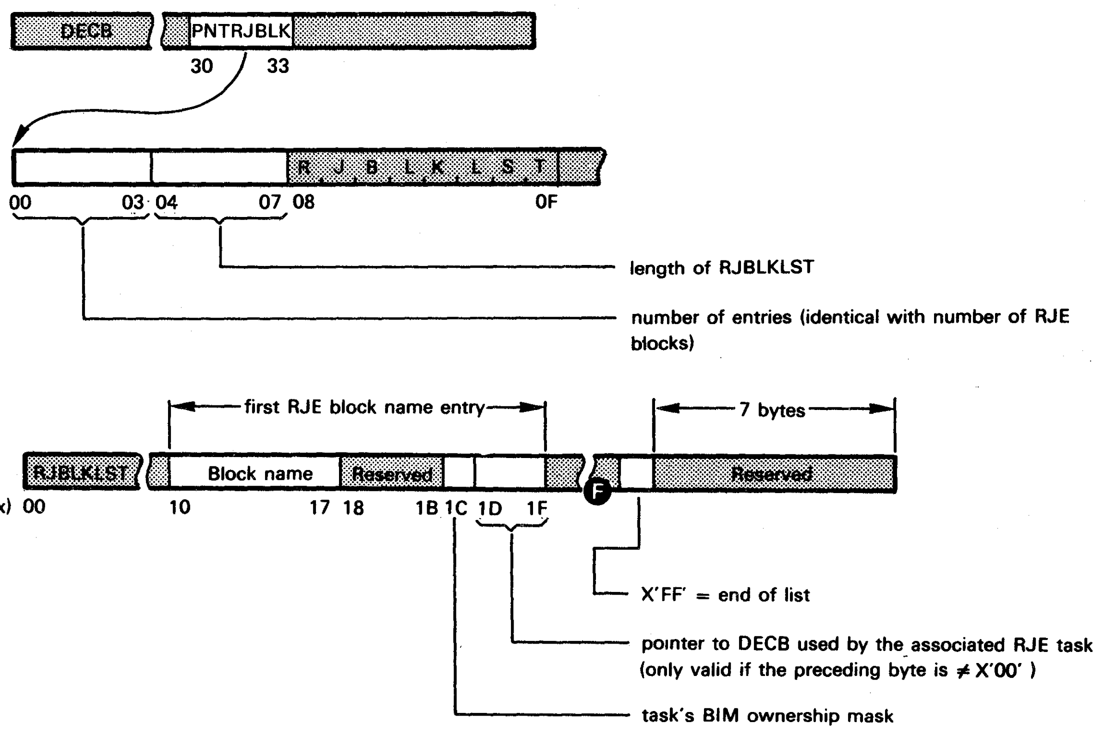
Figure L.1, part 11 of 14

ANALYZING A DUMP OF THE POWER RJE PARTITION

J continued



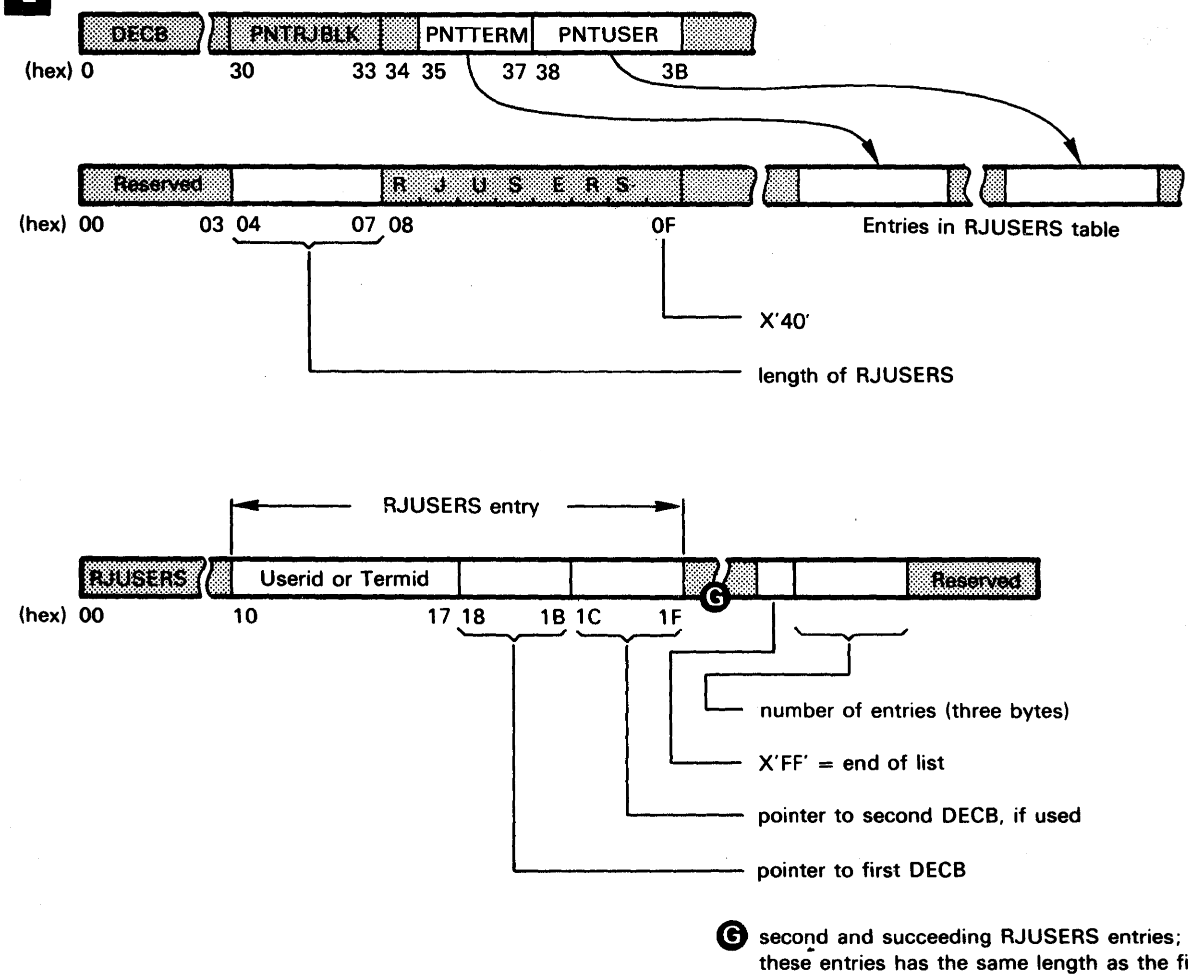
K



F second and succeeding RJE block name entries

Figure L.1, part 12 of 14

L



M

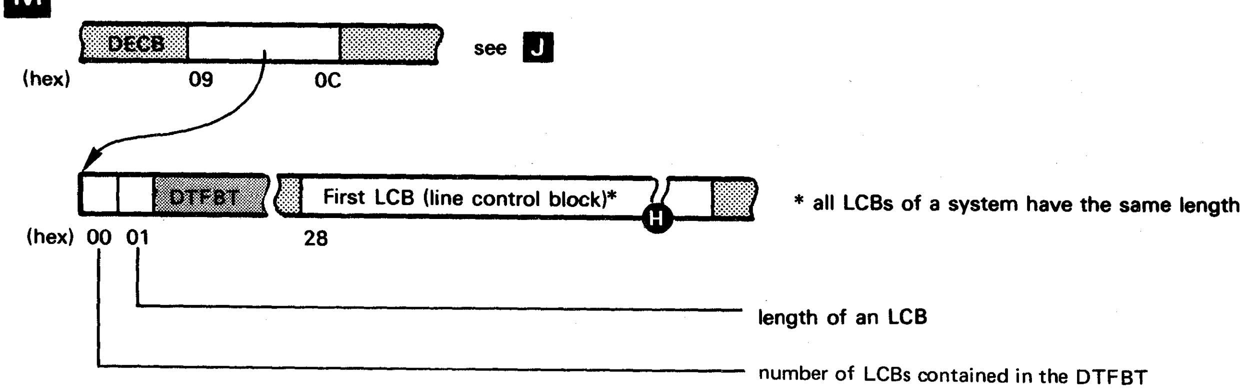
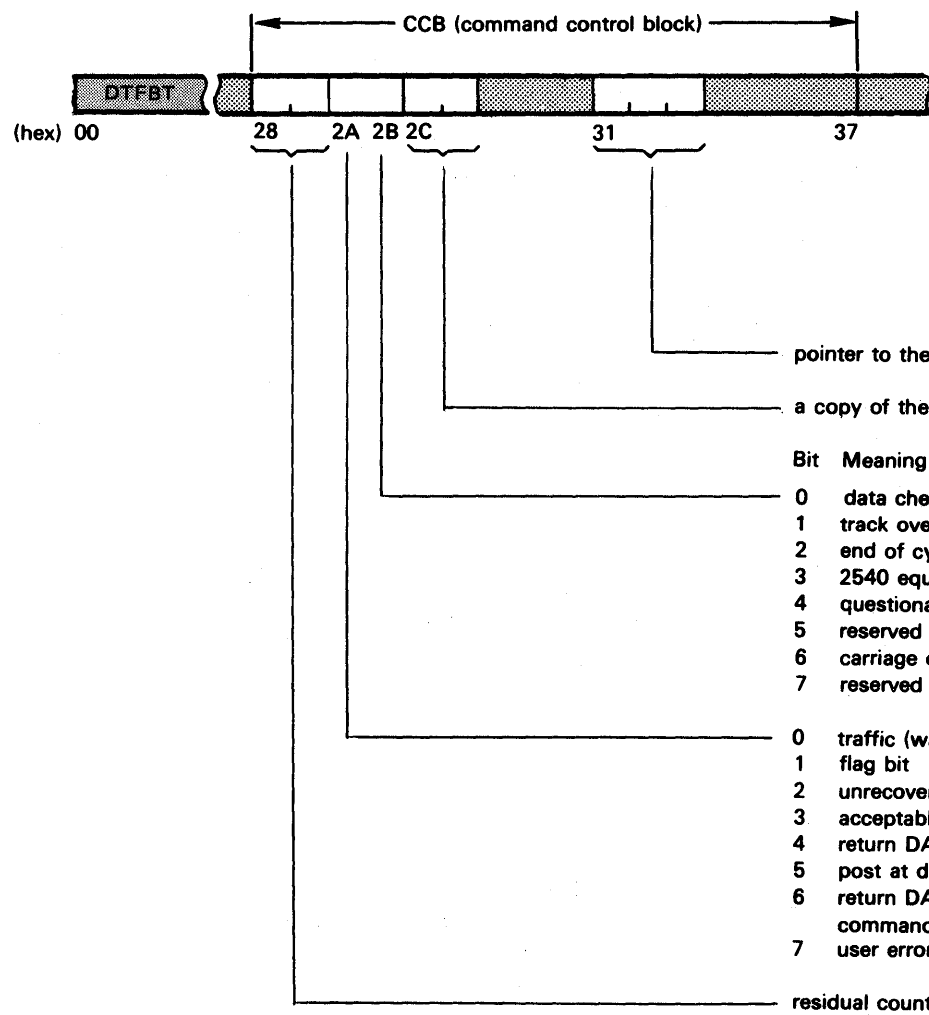


Figure L.1, part 13 of 14

M continued

H



pointer to the first CCW in the channel program

a copy of the CSW status bytes

Bit	Meaning if 1
0	data check in count area (DASD)
1	track overrun (DASD)
2	end of cylinder (DASD)
3	2540 equipment check
4	questionable condition
5	reserved
6	carriage channel 9 or verify error for DASD
7	reserved

0	traffic (wait) bit
1	flag bit
2	unrecoverable I/O error
3	acceptable unrecoverable I/O error
4	return DASD data checks to user
5	post at device end
6	return DASD data checks on count, read command, or verify command
7	user error routine exists

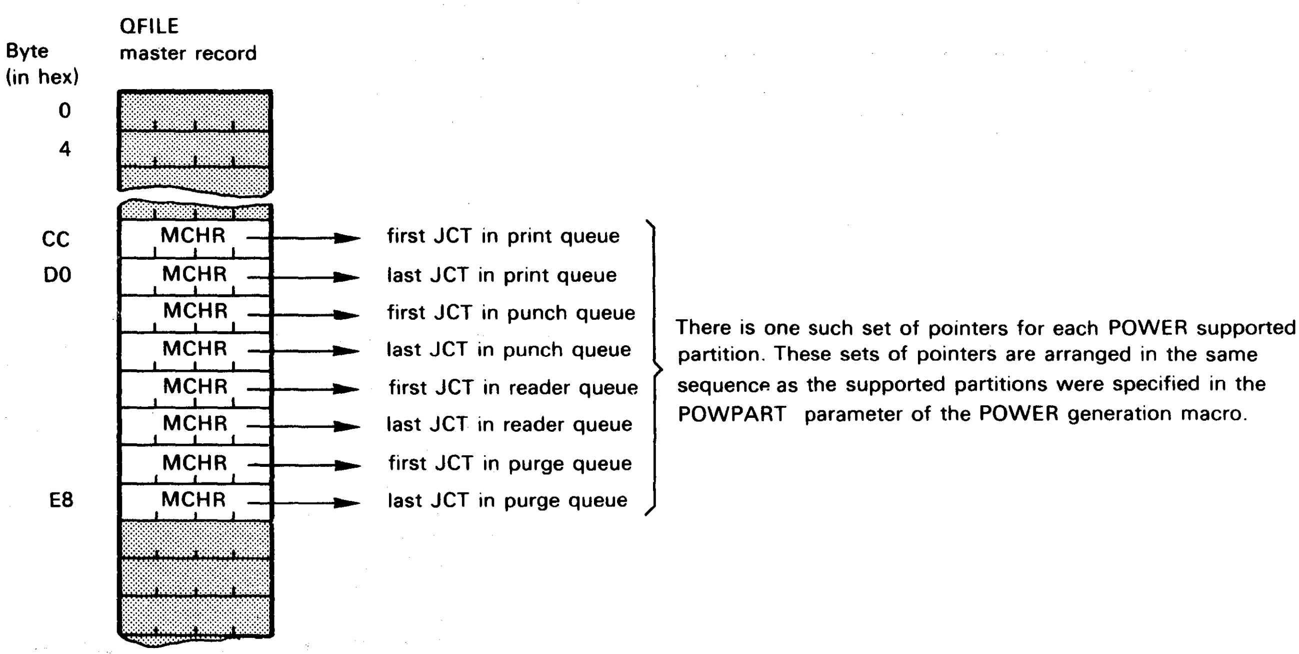
residual count from the CSW

Figure L.1, part 14 of 14

ANALYZING A DUMP OF THE POWER/POWER RJE QFILE

Analyzing a QFILE Dump

A dump of the POWER disk file QFILE can be very helpful if, for example, the POWER program failed because of a disk I/O error. Although analyzing the JCTs in the various queues may not isolate a problem, it may provide hints on how to define the problem still further. Figure M.1 parts 1 and 2 are provided to help you in analyzing a QFILE dump.



Note: For 3330/3333 MCHR means the following:

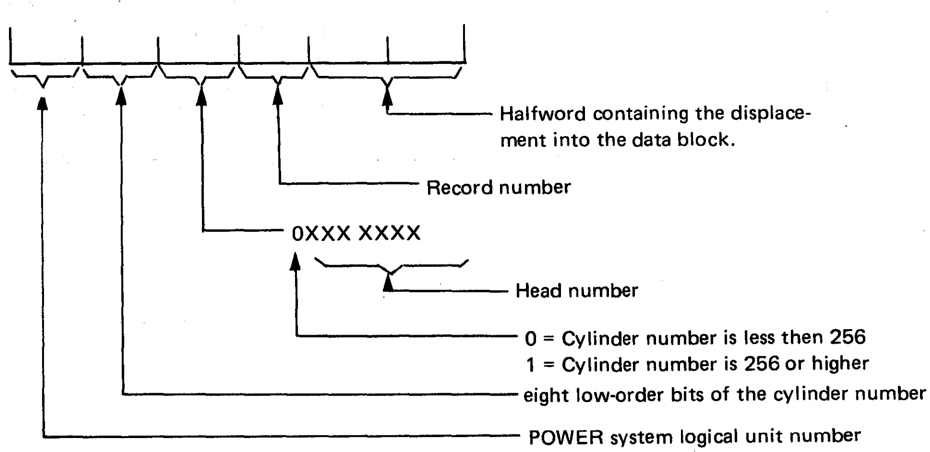
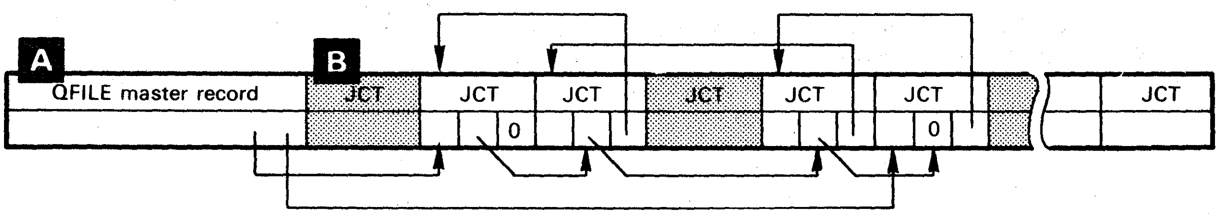


Figure M.1, part 1 of 2

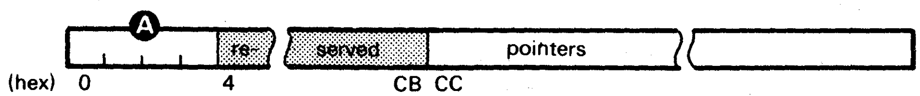
Appendix M

ANALYZING A DUMP OF THE POWER/POWER RJE QFILE

Structure of a hypothetical QFILE (only one queue is shown):

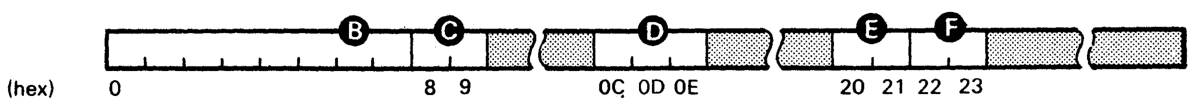


A Layout of a QFILE master record (for further details refer to Figure 5.3):



A If ACCOUNT=YES was specified for POWER generation, this field points to the next available ACCTFIL record; otherwise the field is set to zero.

B Layout of a JCT (for further details refer to Figure 4.4):



B Contains the DOS/VS or POWER job name; if no job name was specified, the field contains the character string AUTONAME.

C Contains the POWER assigned job number.

D Each of the bytes contains either C'D' or C'H':

D = dispatch (the job is ready for processing by the proper task)
H = hold (the job is in the hold state)

Byte 0C: indicates the job's status in the reader queue.
Byte 0D: indicates the job's status in the printer queue.
Byte 0E: indicates the job's status in the punch queue.

E Identifies the associated partition: BG, F1, F2, F3, or F4.

F Contains the class of the job's print or punch output.

Figure M.1, part 2 of 2

Appendix N

ONLINE TERMINAL TEST FOR THE IBM 2780/2770

The online terminal test for the IBM 2780 and 2770 is an optional service provided by BTAM. It is provided to ensure proper operation of the system, and it may be used in the diagnosis and correction of a terminal malfunction. BTAM recognizes Request-for-Test messages transmitted by the remote work station. When an RFT message is recognized, BTAM performs the requested test, which is usually transmission of a tewt message.

The BSC online test facility recognizes Request-for-Test message only if:
1. BSCTEST=YES is coded in the RJE/BTMOD macro.
2. TERMTST=YES is coded in the RJEBLK macro that assembles the DTFBT.
3. The operation is a Read Initial.
4. The application program issued a TWAIT with TERMTST=YES following the macro instruction that was executing when the RFT was received.
5. If a 2770 terminal:
   a. The ONLINE TEST button is pressed.
6. The RFT message is received without error.

Because BTAM only recognizes RFT messages on a Read Initial, the RFT card must be read within 28 seconds to be received without error. Failure to submit the card within 28 seconds requires the user to send an RJSTART and RJEND card. The RJE task must issue a Read Initial before any RFT cards will be accepted. The online test facility is described in the DOS/VS Basic Telecommunications Access ›d Method manual (GC27-6989).

The online test facility prints the results of the test on the computer console. Two messages are provided; one is used when BTAM is transmitting test messages (or RFT messages with X=0), the other when BTAM is receiving test messages. The content of these messages is:

Transmitter	*Receiver*
Line Address	Line Address
Number of Transmissions (Y)	Number of Transmissions
X Field	X Field
Time-outs	Time-outs
NAK's Received	Lost Data Occurrences
Terminal ID (multipoint)	Data Checks

The formats of these messages are:

for the transmitter:

4B70I ON-LINE TEST cuu xx yy TO NK TI

for the receiver:

4B71I ON-LINE TEST cuu xx yy TO LD DC

where:

4B70I identifies the messages as BSC online test results for the transmitter.

4B71I identifies the message as BSC online test results for the receiver.

cuu specifies the line in the form channel and unit.

xx specifies the test type. This is the X field from the RFT message.

yy specifies the number of transmissions. For the transmitter, this value is the value from the RFT message. For the receiver, this value is accumulated by the online test program. If online test was not successfully initiated, this field will contain zero.

TO specifies the number of time-out occurrences.

NK specifies the number of NAK's received by the transmitter.

TI for multipoint lines, specifies the terminal ID; for point-to-point lines, it is blank or specifies the component selection sequence received with the RFT.

LD specifies the number of occurrences of lost data.

DC specifies the number of occurrences of data check.

The following online test procedures may be used for switched and leased-line IBM 2780s.

2780 ONLINE TESTS CONTENTION – PRINT

Before operating these tests be sure that the customer's program has the online test features option in. Do not try to run these tests if online test in the customer's program is not available. 2780 setup:

1. Place the Operate/Test switch on the data set cable and the CE panel in the operate position.
2. Turn the Mode switch on the 2780 operator console to the TRANSMIT position.
3. Flip the Online Test Switch on the CE panel to the ON position.
4. Place the RFT (request for test message) card into the hopper.
5. Ready the printer.
6. Depress the Serial Reader Punch Start key to begin the test.

SOH % 01 PRINT TEST

The purpose of this test is to transmit a test message card to the CPU. The CPU then transmits the data back to the 2780 that was contained in cc 8-80 of the test message card. The CPU sends this message to the 2780 as many times as indicated by the punch in cc 5 and 6 of the test message card.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Indicates this card is an RFT (request for test
2	0–4–8	%	message) card.
3	0	0	Indicates the type of test requested.
4	1	1	
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	This is the message the CPU will send back
9	0–7–9	ESC	to the 2780. Columns 11–79 can contain any
10	0–1		non-control characters. The ETX is shown in
11			column 80, but it may be placed anywhere in
80	12–3–9	ETX	the RFT card to denote the end of the message.

This test should run to the 2780 without error indications. The data contained in cc 11–79 should print with single spacing the number of times indicated by cc 5 and 6.

Appendix N

ONLINE TERMINAL TEST FOR THE IBM 2780/2770

SOH % 12 PRINT TESTS

The purpose of this test is to request that a stored message from the CPU be sent to the 2780 terminal. The 12 designation indicates which transmission code the 2780 terminal has that is requesting the test (EBCDIC).

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for test
2	0–4–8	%	message) card.
3		1	These columns designate a request for a
4		2	stored message.
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	Ending sequence required by the CPU program.
9	12–3–9	ETX	

The following message will print in the 2780 without any errors. The message will be single-spaced and print the number of times indicated by cc 5 and 6 of the test message card. Each print line should look like this:

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

2780 ONLINE TESTS CONTENTION – PUNCH

Before running these tests be sure that the customer's program has online test option in. Do not try to run these tests if online test feature is not available. 2780 setup:

1. If the 2780 has the AutoTurnaround feature, depress the AutoTurnaround switch to place the 2780 in AutoTurnaround mode. If the 2780 does not have AutoTurnaround, install the following cards into the B gate:

Card Location	*6–Bit*	*EBCDIC*	*USASCII*
02B1B20	None	AJW (370643)	None
02B1C34	DGV (370378)	DHC (370372)	DGV (370378)
02B1C35	DGT (370380)	DGV (370378)	DGT (370380)
02B1C36	DGU (370379)	DGT (370380)	DGT (370380)
02B1C37	DGV (370379)	DGU (370379)	DGV (370379)
02B1C38	DHC (370372)	DHC (370372)	DHC (370372)
02B1C39	AJW (370643)	DHC (370372)	None

Note: *The cards to be installed are the special test SMS cards included in the 2780 shipping group. These cards must be removed before returning the 2780 to the customer.*

2. Place the Operate/Test switch on the 2780 data set cable and the CE panel into the operate position.
3. Turn the Mode switch on the 2780 operator console to the transmit position.
4. Flip the Online Test switch located on the CE panel to the ON position.
5. Place the RFT (request for test message) card and a deck of blank cards into the hopper.
6. Depress the Serial Reader Punch Start key to begin the test.

SOH % 01 PUNCH TEST

The purpose of this test is to transmit a test message card to the CPU. The CPU then transmits data to the 2780 that was contained in cc 8–80 of the test message card. The CPU will send this message as many times as indicated by the punching in cc 5 and 6 of the test message card. The 2780 automatically reverts to punch mode because AutoTurnaround was activated, or because the CE test SMS cards were installed as described in step 1.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for
2	0–4–8	%	test message) card.
3	0	0	Indicates the type of test requested.
4	1	1	
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	This is the message the CPU will send back to
9	0–7–9	ESC	the 2780. Any non-control characters can be
10	4	4	punched in cc 11–80.
11			
80			

This test should operate to the 2780 with no error indications. The 2780 should punch the data contained in cc 11–80 of the RFT card in cc 1–69 of the punched output card.

Appendix N

ONLINE TERMINAL
TEST FOR THE
IBM 2780/2770

SOH % 13 PUNCH TEST

The purpose of this test to request that a stored message from the CPU be sent to the 2780 terminal. The 13 designation indicates which transmission code is to be used. The 2780 automatically reverts to punch mode because the AutoTurnaround feature was activated or because the CE test SMS cards were installed as described in step 1.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for test
2	0–4–8	%	message) card.
3		1	These columns indicate a request for a
4		3	stored message.
5		Y	Indicates the number of times that a test
6		Y	is requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	Ending sequence required by the CPU.

The 2780 should punch the following data into cc 1–36 of the output cards as many times as indicated by cc 5 and 6 of the test message card. No errors should occur.

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Note: *Each punched output data card will be followed by a blank card. This is a normal indication caused by the way the 2780 reverts to AutoTurnaround mode.*

ONLINE TERMINAL TEST FOR THE IBM 2780/2770

2770 ONLINE TESTS CONTENTION – PRINT

Before operating these tests be sure that the customer's program has the online test features option in. Do not try to run these tests if online test in the customer's program is not available. 2770 setup:

1. Turn the "Transparency" switch on the 2772 console off.
2. Place the RFT (request for test message) card into the hopper.
3. Press Check Reset and Term Reset at the 2772 console.
4. Press START at the card reader.
5. Press the ON Line Test button at the 2772 console.

SOH % 01 PRINT TEST

The purpose of this test is to transmit a test message card to the CPU. The CPU then transmits the data back to the 2770 that was contained in cc 8–80 of the test message card. The CPU sends this message to the 2770 as many times as indicated by the punch in cc 5 and 6 of the test message card.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Indicates this card is a RFT (request for
2	0–4–8	%	test message) card.
3	0	0	Indicates the type of test requested.
4	1	1	
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	This is the message the CPU will send back
9	0–7–9	ESC	to the 2770. Columns 11–79 can contain any
10	0–1		non-control characters. The ETX is shown in
11			column 80, but it may be placed anywhere in
80	12–3–9	ETX	the RFT card to denote the end of the message.

This test should run to the 2770 without error indications. The data contained in cc 11–79 should print with single spacing the number of times indicated by cc 5 and 6.

SOH % 12 PRINT TESTS

The purpose of this test is to request that a stored message from the CPU be sent to the 2770 terminal. The 12 designation indicates which transmission code the 2770 terminal has that is requesting the test (EBCDIC).

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for
2	0–4–8	%	test message) card.
3		1	These columns designate a request for a stored
4		2	message.
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	Ending sequence required by the CPU program.
9	12–3–9	ETX	

The following message will print in the 2770 without any errors. The message will be single-spaced and print the number of times indicated by cc 5 and 6 of the test message card. Each print line should look like this:

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

2770 ONLINE TESTS CONTENTION – TRANSMIT

Before operating these tests, be sure that the customer's program has the online tests option in. Do not try to run these tests if ED 60 or Online feature is not available. 2770 setup:

1. Turn the "Transparency" switch on the 2772 console off.
2. Place the RFT (request for test message) card into the hopper.
3. Press Check Reset and Term Reset at the 2772 console.
4. Press START at the card reader.
5. Press the On Line Test button at the 2772 console.

SOH % 00 TRANSMIT TEST

The purpose of this test is to allow the 2770 to transmit a deck of test cards to the CPU. The CPU responds with a DLE sequence if no error occurred during transmission; the CPU responds with a NAK if a CRC or VRC error occurred. There is no response for any other type of error.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Indicates that this is an RFT (request
2	0–4–8	%	for test message) card.
3	0	0	Indicates the type of test required.
4	0	0	
5	0	0	For this test, 01 is the only allowable
6	1	1	number of messages to be sent (see note).
7	0	0	Denotes the contention terminal.
8	12–2–9	STX	Data to be sent as a test. Columns 9–79 can contain any non-control characters.
80	12–3–9	ETX	Last card only.

Note: *Columns 5 and 6 of the transmit test must be 01. In order to transmit more than one card, a deck of cards must be punched with the same control characters in cc 1–8 and cc 80 of the test message card. Any non-control characters can be punched in cc 9–79. Place this deck into the hopper and hold the start key depressed until a buffer is read. If the start key is not held depressed, only one card will be sent by the 2770.*

The test messages should be transmitted to the CPU and accepted by it without any errors. The audible alarm will sound for a short period after each card and will remain on at the end of this test.

Appendix N

ONLINE TERMINAL TEST FOR THE IBM 2780/2770

SOH % 15 OR 16 WEAK DIBIT TEST

The purpose of this test is to request a stored message from the CPU. A stored message transmits the worst-case conditions for the data set and communication lines. When the RFT (request for test message) card has been transmitter and the audible alarm sounds, the 2780 mode switch should be turned to the print or punch position and either unit made ready. The output message (weak dibit) will then be printed or punched depending upon the position of Mode switch.

Note: *The user has 6 seconds to ready the output unit after the Mode switch is turned to the print or punch position.*

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this as an RFT (request for test message)
2	0–4–8	%	card.
3		X	Identifies the type of stored message we are
4		X	requesting (see note).
5		Y	Indicates the number of times we are
6		Y	requesting the test (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	Ending sequence required by the CPU.
9	12–3–9	ETX	

Note: *Columns 3 and 4 should be punched according to the transmission code and line facilities of the terminal involved.*

Transmission Code	*Column 3*	*Column 4*
EBCDIC switched or IBM clock	1	5
EBCDIC leased line	1	6

The 2780 should print or punch the weak dibit message as many times as indicated by cc 5 and 6 on a test message card. No errors should occur during the test. If line errors occur during this test but not during any of the other online tests, the data set or communications line may be failing. The dibit characters may not print or may print as something else. Operating with no line check is the basic goal of this test.

2770 ONLINE TESTS CONTENTION – PUNCH

Before running these tests be sure that the customer's program has online test option in. Do not try to run these tests if online test feature is not available. 2770 setup:

1. Be sure POWER is on for the 545 card punch and that it is initialized.
2. Turn the "Transparency" switch on the 2772 console off.
3. Place the RFT (request for test message) card into the hopper.
4. Press Check Reset and Term Reset as the 2772 console.
5. Press START at the card reader.
6. Press the On Line Test button at the 2772 console.

SOH % 01 PUNCH TEST

The purpose of this test is to transmit a test message card to the CPU. The CPU then transmits data to the 2770 that was contained in cc 8–80 of the test message card. The CPU will send this message as many times as indicated by the punching in cc 5 and 6 of the test message card. The 2770 automatically reverts to punch mode.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for test
2	0–4–8	%	message) card.
3	0	0	Indicates the type of test requested.
4	1	1	
5		Y	Indicates the number of times the test is
6		Y	requested (01–99).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	This is the message the CPU will send back
9	0–7–9	ESC	to the 2770. Any non-control characters can
10	4	4	be punched in cc 11–80.
11			
80			

This test should operate to the 2770 with no error indications. The 2780 should punch the data contained in cc 11–80 of the RFT card in cc 1–69 of the punched output card.

Note: Each punched output data card will be followed by a blank card.

Appendix N

ONLINE TERMINAL TEST FOR THE IBM 2780/2770

SOH % 13 PUNCH TEST

The purpose of this test to request that a stored message from the CPU be sent to the 2770 terminal. The 13 designation indicates which transmission code is to be used. The 2770 automatically reverts to punch mode.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Identifies this card as an RFT (request for
2	0–4–8	%	test message) card.
3		1	These columns indicate a request for a
4		3	stored message.
5		Y	Indicates the number of times that a test
6		Y	is requested (01–99).
7	0	0	Denotes a contention terminal.
8	12-2-9	STX	Ending sequence required by the CPU.

The 2770 should punch the following data into cc 1–36 of the output cards as many times as indicated by cc 5 and 6 of the test message card. No errors should occur.

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

2780 ONLINE TESTS CONTENTION – TRANSMIT AND WEAK DIBIT

Before operating these tests, be sure that the customer's program has the online tests option in. Do not try to run these tests if ED 60 or Online feature is not available. 2780 setup:

1. Place the Operate/Test switch on the data set cable and on the 2780 CE panel to the operate position.
2. Turn the Mode switch on the 2780 console to the transmit position.
3. Flip the Online Test switch located on the CE panel into the ON position.
4. Place the RFT (request for test message) card or cards into the hopper.
5. Depress the Serial Reader Punch Start key to begin the test.

The following online test procedures may be used for switched and leased-line IBM 2770s.

SOH % 00 TRANSMIT TEST

The purpose of this test is to allow the 2780 to transmit a deck of test cards to the CPU. The CPU responds with a DLE sequence if no error occurred during transmission; the CPU responds with a NAK if a CRC or VRC error occurred. There is no response for any other type of error.

Format is as follows:

Col	*Punches*	*Character*	
1	12–1–9	SOH	Indicated that this is an RFT (request
2	0–4–8	%	for test message) card.
3	0	0	Indicates the type of test required.
4	0	0	
5	0	0	For this test, 01 is the only allowable
6	1	1	number of messages to be sent (see note).
7	0	0	Denotes a contention terminal.
8	12–2–9	STX	Data to be sent as a test. Columns 9–79 can contain any non-control characters.
80	12–3–9	ETX	Last card only.

Note: Columns 5 and 6 of the transmit test must be 01. In order to transmit more than one card, a deck of cards must be punched with the same control characters in cc 1–8 and cc 80 of the test message card. Any non-control characters can be punched in cc 9–79. Place this deck into the hopper and hold the start key depressed until a buffer is read. If the start key is not held depressed, only one card will be sent by the 2780.

The test messages should be transmitted to the CPU and accepted by it without any errors. The audible alarm will sound for a short period after each card and will remain on at the end of this test.

PROCESSING TAPE ERROR STATISTICS USING EREP

You can cause detailed or summarized tape statistics to be printed through the use of the various combinations of EREP options shown in Figure F-3-D in Section 2-F of this manual. The summarized format combines the individual recordings (for example, Unit Check, Volume Dismount, and End-of-Day records) either by volume serial number or by tape unit, and prints the summarized statistics. The detail format prints each recording in either volume serial number format or tape unit format. Whenever detail or summarized data is printed in volume serial number format, the data is printed in sequence by volume serial number.

Example 1: Print detail tape error statistics from SYSREC. The information is printed in the format of record 4 of the example printout below. Enter the following job control statements:

```
// EXEC EREP
  OPTION TES,NOTAPE,PRINT
/*
```

Example 2: Print the summarized tape error statistics from SYSREC only. The data is printed in the format of record 3 of the example printout below. Enter the following job control statements:

```
// EXEC EREP
  OPTION TES,NOTAPE,SUM
/*
```

Example 3: Print the detail tape error records and then print their summary by volume serial number. The data is printed in the format of records 1 and 3 of the example printout below. The following job control statements:

```
// EXEC EREP
  OPTION TES,NOTAPE,PRINT,SUM,SUMTAPE,VOL
/*
```

A work tape is required because the VOL option is specified.The work tape will contain a sequential list of all volume serial numbers along with a 5-byte disk address for each of these numbers. The message

3E08A MOUNT SCRATCH TAPE ON SYS008

is printed on SYSLOG. After the scratch tape is mounted the operator should respond END. If the operator chooses not to mount a work tape, he should respond CANCEL END. This causes the SUM and PRINT TES options to be canceled. Any other response results in the messages

3E25I INVALID RESPONSE
3E08A MOUNT SCRATCH TAPE ON SYS008

being printed on SYSLOG.

Example 4: Update the TES history tape on SYS007. Then a scratch tape is mounted on SYS008. The error records are edited and printed from SYSRES onto SYSLST in the detail volume serial number format (record 2 of the example printout below). The tape error records on the history tape are then summarized and printed on SYSLST in the summarized volume serial number format (record 1 of the example printout below). Enter the following job control statements:

```
// LBLTYP TAPE
// TLBL EREPNEW
// EXEC EREP
 OPTION HIST
 OPTION TES,PRINT,SUM,SUMTAPE,VOL
/*
```

First the TES history tape is updated: the message

3E09A MOUNT TES HISTORY TAPE ON SYS007

is printed on SYSLOG. After the TES history tape has been updated, the tape error data on SYSREC is edited. The message

3E08A MOUNT SCRATCH TAPE ON SYS008

is printed on SYSLOG. The tape data is printed on SYSLOG and then the message

3E18A MOUNT HISTORY/RDE TAPE

is printed on SYSLOG. The history tape is read and the tape error data is summarized by volume serial number. Finally, the history tape is updated and the SYSREC file is cleared.

```
RECORD 1              SUMMARY   MAGNETIC TAPE ERROR STATISTICS        XX/XXX

VOLUME         PERM PERM TEMP TEMP SIO NRZI           CPU MOD ERASE CLEANER
SERIAL  DATE   READ WRT   RD   WRT COUNT NOISE ID SERIAL NO GAP ACTION


RECORD 2              DETAIL MAGNETIC TAPE ERROR STATISTICS BY VOLUME DATE XX/XXX
VOLUME           TIME          TU  RD/  PERM PERM TEMP TEMP SIO BLOCK PROGRAM CPU MOD DENSITY
SERIAL DATE OF DAY CUA SERIAL WRT READ WRT RD WRT COUNT LENGTH ID ID NO


RECORD 3            SUMMARY MAGNETIC TAPE ERROR STATISTICS            XX/XXX
           TU      SIO TEMP TEMP PERM PERM NRZI EQUIP OVDR EARLY WR TM IBG FEED VEL PART SLOW EXC
CUA SERIAL DATE COUNT RD WRT RD WRT NOISE CK RUN END CHECK DROP THRU RTRY REC BOR PAMB


RECORD 4              DETAIL MAGNETIC TAPE ERROR STATISTICS BY TAPE UNIT DATE XX/XXX
          TU      VOLUME  TIME TEMP TEMP SIO DENSITY NRZI R/W WR TG LRC CRC ECC SKEW ERLY VEL
CUA SERIAL DATE SERIAL OF DAY RD WRT COUNT          NOISE VRC VRC MTE EDC ENV ERR BOR CHG TIE
```

An example of the EREP TES print formats.

Appendix P

EXAMPLES OF THE SUM OPTION OF EREP

Note: This option of EREP is only applicable to the Model 145

Example 1: The job control statements required for a summary of the SYSREC file by disk, tape, unit record, and TP groups are:

```
// EXEC EREP
 OPTION SUM
 GROUP=DISK,TAPE,UNITREC,TP
/*
```

Example 2: The control statements required for a summary of the SYSREC file by MICR/OCR, CPU, and 2715 hardware groups are:

```
// EXEC EREP
 OPTION SUM
 GROUP=MICR/OCR,CPU,2715
/*
```

The 2715 groups is summarized first

Example 3: job entered through SYSIPT requesting the RDE Summary Option

```
// JOB EXAMPLE
// ASSGN SYS009,X'283'
// TLBL EREPNEW
// LBLTYP TAPE
// EXEC EREP
   OPTION SELECT,TAPE
   DEVICE=2314
   CUA=0134
    OPTION RDESUM
   OPTION RDESUM
   OPTION EDIT
/*
/&
```

The RDE summary parameters will be requested on SYSLOG.

This glossary contains technical terms associated with the subject of this publication. A more general range of terms is contained in *IBM Date Processing Glossary, GC20-1699.*

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard Vocabulary for information Processing (copyright © 1970 by American National Standards Institute, Incorporated) which was prepared by subcommittee X3K5 on Terminology and Glossary of American National Standards Committee X3. These definitions are indicated by an asterisk.

A

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

asynchronous. without regular time relationship.

auxiliary storage. Data storage other than real storage; for example, storage on magnetic tape or disk. Synonymous with external storage, secondary storage.

B

basic control mode. When PSW bit 12 is 0, PSW format and system operation are compatible with standard System/360 operation. This is the basic control mode in which control registers 0, 8, and 14 are available to the system. Abbreviated to BC mode. See also "Extended Control Mode."

BTAM (basic telecommunications access method). A basic access method that permits a READ/WRITE communication with remote devices.

buffer. (1) A storage device in which data is assembled temporarily during data transfer. (2) During I/O operations, a portion of real storage into which data is read or from which data is written.

C

channel program. One or more Channel Command Words (CCWs) that control(s) a specific sequence of channel operations. Execution of the specific sequence is initiated by a single start I/O instruction.

channel program translation. In a channel program, replacement, by software, of virtual addresses with real addresses.

command control block (CCB). A 16-byte field required for each channel program executed by physical IOCS. This field is used for communication between physical IOCS and the problem program.

communication region. An area of the supervisor set aside for interprogram and intraprogram communication. It contains information useful to both the supervisor and the problem program. Abbreviated comreg. (Not to be confused with the COMRG macro instruction).

control program. A program that is designed to schedule and supervise the performance of data processing work by a computing system.

control registers. A set of registers used for operating system control of relocation, priority interruption, program event recording, error recovery, and masking operations.

core-wrap mode. The method of operation that records the events of a trace in the PD area or an alternate area (used by PDAIDS). It is the default process when no output device for a PDAID trace has been specified.

D

DTF (define the file) macro instruction. A macro instruction that describes the characteristics of a logical input/output file, indicates the type of processing for the file, and specifies the I/O areas and routines to process the file.

default value. The choice among exclusive alternatives made by the system when no explicit choice is specified by the user. A default value is indicated by underlining in tables listing parameters.

diskette. A flexible magnetic oxide coated disk, permanently enclosed in a semi-rigid protective plastic jacket approx, 8 inches square. During data processing operations the disk turns freely within the jacket. It is capable of storing 1898 128 character data records.

dump. (1) To print out the contents of all or part of virtual storage or of auxiliary storage (2) The data resulting from the process as in (1).

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

E

emulator (1) * A device or computer program that emulates. (2) The combination of programming techniques and special machine features that permits a given computing system to execute programs written for another system.

environmental recording, editing, and printing (EREP). A program that processes the data contained on the system recorder file.

error recovery procedures. Procedures designed to help isolate, and, when possible, to recover from hardware errors in equipment. The procedures are often used in conjunction with programs that record the statistics of machine malfunctions.

error volume analysis (EVA). With this option, the system issues a message to the operator when a number of temporary read or write errors (specified by the user at system generation time) has been exceeded on a currently accessed tape file.

event. An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as input/output.

extent. The physical locations on Input/Output devices occupied by or reserved for a particular volume.

extended control mode. When PSW bit 12 is set to 1, the PSW format is changed from that used for standard System/360 operation: the channel mask bits, instruction length code, and interruption code are removed, and additional mode and mask bits are included. This is the extended control mode, in which all control registers are available to the system for control of facilities that are particular to System/370. Abbreviated to EC mode. See also "Basic Control Mode."

F

fetch. (1) To bring a program phase into real storage from a core image library or from the page data set for immediate execution. (2) The routine that retrieves requested phases and loads them. (3) The name of a macro instruction (FETCH) used to transfer control to the system loader. (4) To transfer control to the system loader.

* **file.** A collection of related records treated as a unit. For example, one line of an invoice may form an item, a complete invoice may form a record, the complete set of such records may form a file, the collection of inventory control files may form a library, and the libraries used by an organization are known as its data bank.

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

F/L Trace (Fetch/Load Trace). A program that records information about phases and transients as they are called from a core image library.

G

GSVC Trace (Generalized Supervisor Calls Trace). A program that records SVC interrupts as they occur. All or a selected group of SVCs can be traced.

H

hard copy. A printed copy of machine output in a visually readable form, for example, a printed recording of the messages displayed on the System/370 Model 125 video display unit.

hard stop. A condition, usually caused by an error, in which the CPU is stopped and is not executing the microprogram.

* **hardware.** Physical equipment, as opposed to the computer program or method of use, for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

I

Input Job Stream. A sequence of job control statements entering the system, which may also include input data.

* **interface.** A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

interrupt. A break in the normal sequence of instruction execution. It causes an automatic transfer to a preset storage location where appropriate action is taken.

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

I/O area. An area (portion) of real storage into which data is read or from which data is written, the term buffer is often used in place of I/O area.

I/O Trace (Input/Output Trace). A program that records I/O device activity for all or a selected group of I/O devices.

IOCS (input/output control system). A group of macro instruction routines provided by IBM for handling the transfer of data between main storage and external storage devices.

irrecoverable error. A hardware error which cannot be recovered from by the normal hardware and retry procedures.

J

job. (1) * A specified group of tasks prescribed as a unit of work for a computer. By extension, a job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) A collection of related problem programs, identified in the input stream by a JOB statement followed by one or more EXEC statements.

L

linkage editor. A processing program that prepares the output of language translators for execution. It combines separately produced object or load modules; resolves symbolic cross references among them, and generates overlay structures on request; and produces executable code (a load module) that is ready to be fetched into virtual storage.

load. In programming, to enter instructions or data into storage or working registers. In DOS/VS, to bring a program phase from a core image library into virtual storage for execution.

logic module. The logical IOCS routine that provides an interface between a processing program and physical IOCS.

* **loop.** A sequence of instructions that is executed repeatedly until a terminal condition prevails.

LSERV (label cylinder display). A program that formats a listing of the label cylinder located on SYSRES.

M

machine check analysis and recovery. A feature that checks the severity of a CPU hardware failure and attempts to recover from the interrupt. Abbreviated MCAR.

machine check interrupt. The interrupt that occurs if the CPU fails to operate.

main page pool. The set of all page frames in real storage not assigned to the supervisor or one of the real partitions.

main storage. (1) The real address area of virtual storage. Contrast with auxiliary storage. (2) All program addressable storage from which instructions may be executed and from which data can be loaded directly into registers.

microprogram. A set of basic or elementary machine instructions that is loaded into control storage to control CPU operations.

* **module.** A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading, for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine.

multiplexer channel. A channel designed to operate with a number of I/O devices simultaneously on a byte basis. That is, several I/O devices can be transferring records over the multiplexer channel, time sharing it on a byte basis.

multiplexer mode. A means of transferring records to or from low-speed I/O devices on the multiplexer channel, by interleaving bytes of data. The multiplexer channel sustains simultaneous I/O operations on several subchannels. Bytes of data are interleaved and then routed to or from the selected I/O devices or to and from the desired locations in main storage. Multiplex mode is sometimes referred to as byte mode.

multiprogramming system. A system that controls more than one program simultaneously by interleaving their execution.

multitasking. The concurrent execution of one main task and one or more subtasks in the same position.

O

offline. (1) * Pertaining to equipment or devices not under control of the central processing unit. (2) Pertaining to program error diagnosis without using the computer system (offline program debugging).

* **online.** (1) Pertaining to equipment or devices under control of the central processing unit. (2) Pertaining to a user's ability to interact with a computer.

online test executive program (OLTEP). The control program of the online test system. OLTEP is the interface between the online test and the operating system.

operand. (1) * That which is operated upon. An operand is usually identified by an address part of an instruction. (2) Information entered with a command name to define the data on which the command processor operates and to control the execution of the command processor.

* **overflow.** (1) That portion of the result of an operation that exceeds the capacity of the intended unit of storage. (2) Pertaining to the generation of overflow as in (1).

P

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

page data set. An extent in auxiliary storage, in which pages are stored.

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

page frame. A 2K block of real storage that can contain a page.

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

processor. (1) * In hardware, a data processor. (2) * In software, a computer program that includes the compiling, assembling, translating, and related functions for a specific programming language. RPG II processor, FORTRAN processor. (3) Same as processing program.

Private Second Level Directory (PSLD). The Private Second Level Directory is a table, located in the Supervisor and containing the highest phasenames found on the corresponding directory tracks of the Private Core Image Library.

page pool. The set of all page frames that may contain pages of programs in virtual mode.

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

page translation exception. A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set. See also segment translation exception, translation specification exception.

paging. The process of transferring pages between real storage and the page data set.

* **parameter.** A variable that is given a constant value for a specific purpose or process.

physical IOCS. Macro instructions and supervisor routines (Channel Scheduler) that schedule and supervise the execution of channel programs. Physical IOCS controls the actual transfer of records between the external storage medium and real storage.

problem determination aids (PDAID). Programs that trace a specified event when it occurs during the operation of a program. The traces provided are: QTAM Trace, I/O Trace, F/L Trace, and GSVC Trace.

problem program. (1) The user's object program. It can be produced by any of the language translators. It consists of instructions and data necessary to solve the user's problem. (2) A general term for any routine that is executed in the data processing system's problem state; that is, any routine that does not contain privileged operations. (Contrasted with Supervisor.)

processing program. (1) A general term for any program that is not a control program. (2) Synonymous with problem program.

program event recording. A System/370 feature that enables a program to be alerted to specific events. Abbreviated PER.

Q

QTAM Trace. A program that records certain supervisor and I/O activities on tape or in core-wrap mode.

queue. (1) A waiting line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted in message switching system. (2) To arrange in, or from, a queue.

R

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

real address area. In DOS/VS, the area of virtual storage where virtual addresses are equal to real addresses.

real mode. In DOS/VS, the mode of a program that may not be paged.

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

real partition. In DOS/VS, a division of the real address area of virtual storage that may be allocated for programs that are not to be paged, or programs that contain pages that are to be fixed.

recovery management support. The facilities that gather information about hardware reliability and allow retry of operations that fail because of CPU, I/O device, or channel errors. Abbreviated to RMS.

reenterable. The attribute of a set of code that allows the same copy of the set of code to be used concurrently by two or more tasks.

reliability data extractor (RDE). A function that provides hardware reliability data that is analyzed by IBM.

relocatable library. A library of relocatable object modules and IOCS modules required by various compilers. It allows the user to keep frequently used modules available for combination with other modules without recompilation.

resource. Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the central processing unit, data files, and control and processing programs.

* **routine.** An ordered set of instructions that may have some general or frequent use.

S

Second Level Directory (SLD). The table, located in the Supervisor and containing the highest phase-names found on the corresponding directory tracks of the system core image.

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

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

segment translation exception. A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set. See also page translation exception, translation specification exception.

self-relocating. A programmed routine that is loaded at any doubleword boundary and can adjust its address values so as to be executed at that location.

self-relocating program. A program that is able to run in any area of storage by having an initialization routine to modify all address constants at object time.

selector channel. A channel designed to operate with only one I/O device at a time. Once the I/O device is selected, a complete record is transferred one byte at a time.

SEREP. A stand-alone environment recording, editing, and printing program that makes the data contained in an error logout area of real storage available for further analysis.

Shared Virtual Area (SVA): The last part of the virtual system address space that contains phases which are reenterable and relocatable and which can be shared between partitions.

soft stop. A condition in which the CPU has stopped processing but continues to handle any requested interruptions.

stand-alone dump. A program that displays the contents of the registers and all of real storage and that runs independently and is not controlled by DOS/VS.

subtask. A task in which control is initiated by a main task by means of a macro instruction that attaches it.

* **storage protection.** An arrangement for preventing access to storage for either reading, or writing, or both.

system generation. The process of tailoring the IBM supplied operating system to user requirements.

system debugging aids. A set of routines provided to trace specific program events by using the program event recording facilities. Abbreviated SDAIDS.

System Directory List (SDL). A list of highly used phases (either only in the system CIL or also in the SVA). This list is placed in the SVA.

system recorder file. The data file that is used to record hardware reliability data.

T

task. A unit of work for the central processing unit from the standpoint of the control program.

task selection. The supervisor mechanism for determining which program should gain control of CPU processing.

teleprocessing. The processing of data that is received from or sent to remote locations by way of telecommunication lines.

terminal. (1) * A point in a system or communication network at which data can either enter or leave. (2) Any device capable of sending and receiving information over a communication channel.

Terminating partition. This is a partition owning a program which is in the process of being terminated either because of a program cancel condition or because of EOJ.

trace. (1) To record a series of events as they occur. (2) The record of a series of events.

* **tracing routine.** A routine that provides a historical record of specified events in the execution of a program.

track hold. A function for protecting DASD tracks that are currently being processed. When track hold is specified in the DTF, a track that is being modified by a task in one partition cannot be concurrently accessed by a task or subtask in another partition.

transient area. An area in the supervisor used for temporary storage of transient routines, such as non-resident supervisor call or error-handling routines.

transient routines. These self-relocating routines are permanently stored on the system residence device and loaded (by the supervisor) into the transient area when needed for execution.

translation specification exception. A program interruption that occurs when a page table entry, segment table entry, or the control register pointing to the segment table contains information in an invalid format. See also page translation exception, segment translation exception.

U

user program. see problem program.

unrecoverable error. see irrecoverable error.

utility program. A program designed to perform a routine task, such as transcribing data from one storage device to another.

V

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

virtual address area. In DOS/VS, the area of virtual storage whose addresses are greater than the highest address of the real address area.

virtual mode. In DOS/VS, the mode of a program which may be paged.

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

virtual storage access method (VSAM). VSAM is an access method for direct or sequential processing of fixed and variable length records on direct access devices. The records in a VSAM file can be organized either in logical sequence by a key field (key sequence) or in the physical sequence in which they are written on the file (entry-sequence). A key sequenced file has an index, an entry-sequenced file does not.

volume. (1) That portion of a single unit of storage media which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and dismounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

VSAM access method services. A multifunction utility program that defines VSAM files and allocates space for them, converts indexed sequential files to key-sequenced files with indexes, facilitates data portability between operating systems, creates backup copies of files and indexes, helps make inaccessible files accessible, and lists file and catalog entries.

General

IBM System/360 and System/370 Bibliography, GA22-6822
IBM System/360 Advanced Function Bibliography, GC20-1763
Teleprocessing and Data Acquisition Bibliography, GA24-3089
A Data Processing Glossary, GC20-1699
Introduction to IBM Data Processing Systems, GC20-1684
IBM System/370 Principles of Operation, GA22-7000
A Guide to IBM System/370 Model 135, GC20-1738.

Note: These publications identify and describe all literature in, or related to, the Systems Library for the System/370. *The accumulative index lists the currently-available literature and should be consulted before you order a publication.*

Publication title	Form No.
Introduction to DOS/VS	GC33-5370
DOS/VS Data Management Guide	GC33-5372
DOS/VS System Management Guide	GC33-5371
DOS/VS Supervisor and I/O Macros	GC33-5373
DOS/VS Tape Labels	GC33-5374
DOS/VS DASD Labels	GC33-5375
DOS/VS System Control Statements	GC33-5376
DOS/VS System Generation	GC33-5377
DOS/VS Operating Procedures	GC33-5378
DOS/VS Messages Reference	GC33-5379
DOS/VS System Utilities	GC33-5381
DOS/VS Access Method Services	GC33-5382
Guide to the DOS/VS Assembler	GC33-4024
OS/VS and DOS/VS Assembler Language Guide	GC33-4010
DOS/VS OLTEP Reference	GC33-5383
DOS/VS Supervisor Logic	SY33-8551
DOS/VS Error Recovery and Recording Transient Logic	SY33-8552
DOS/VS Logical Transients Logic	SY33-8553
DOS/VS System Serviceability Aids Logic	SY33-8554
DOS/VS IPL and Job Control Logic	SY33-8555
DOS/VS LIOCS Logic Vol. 1 (Introduction and imperative macros)	SY33-8559
DOS/VS LIOCS Logic Vol. 2 (SAM)	SY33-8560
DOS/VS LIOCS Logic Vol. 3 (DAM and ISAM)	SY33-8561
DOS/VS LIOCS Logic Vol. 4 (VSAM)	SY33-8562
DOS/VS System Utilities Logic	SY33-8558
DOS/VS Linkage Editor Logic	SY33-8556
DOS/VS Librarian Logic	SY33-8557
DOS/VS Access Method Services Logic	SY33-8564
DOS/VS POWER Logic	SY33-8565
DOS/VS POWER/RJE Logic	SY33-8566
DOS/VS Assembler Logic	SY33-8567
DOS/VS OLTEP Logic	SY33-8568
DOS/VS BTAM Reference	GC27-6989
DOS/VS BTAM Logic	SY27-7251
QTAM Message Processing Program Services	GC27-6985
QTAM Message Control Program Guide	GC27-6986
DOS/VS QTAM Message Control Program Logic	SY27-7249
System/370, Model 115, Operating Procedures	GA33-1514
System/370, Model 125, Central Test Manual	4686240
System/370, Model 125, Operating Procedures	GA33-1509
System/370, Model 135, Operating Procedures	GC38-0005
System/370, Model 145, Operating Procedures	GC38-0015
System/370, Model 158, Operating Procedures	GC38-0025
System/370, Principles of Operation	GA22-7000

INDEX

C (continued)

D

INDEX D (continued)

D (continued)

E

INDEX E (continued)

F

G

G (continued)

H

INDEX **H (continued)**

J (continued)

L

M

INDEX **M (continued)**

P (continued)

Q

R

S

INDEX S (continued)

S (continued)

T

INDEX **T (continued)**

DOS/VS Serviceability Aids and Debugging Procedures

GC33-5380-1

READER'S COMMENT FORM

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

GC33-5380-1

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

CUT ALONG THIS LINE

DOS/VS Serviceability Aids and Debugging Procedures (S370-37) Printed in U.S.A. GC33-5380-1

Fold Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 U

Fold Fold

IBM®

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

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

Technical Newsletter

This Newsletter No.	GN33-8780
Date	June 30, 1974
Base Publication No.	GC33-5380-1,
File No.	S370-37
Previous Newsletters	None

DOS/VS Serviceability Aids
and Debugging Procedures

This Technical Newsletter, a part of Release 30 of the IBM Disk Operating System DOS/VS, provides replacement pages for your publication. These replacement pages remain in effect for subsequent DOS/VS releases unless specifically altered. Pages to be inserted and/or removed are:

Cover - 4	2.103, 2.104	4.39, 4.40	A.17, A.18
ix, x	2.111, 2.112	4.43, 4.44	A.37 - A.40
1.19, 1.20	2.169, 2.170	4.49 - 4.52	A.45, A.46
2.9, 2.10	2.213 - 2.216	4.63, 4.64	A.49, A.50
2.17, 2.18	2.233, 2.234	4.71, 4.72	G.1, G.2
2.23, 2.24	3.17, 3.18	4.72.1 - 4.72.4 (added)	G.5, G.6
2.29, 2.30	3.25 - 3.28	4.77, 4.78	
2.39, 2.40	4.5, 4.6	4.87, 4.88	
2.57, 2.58	4.13, 4.14		
2.85 - 2.94	4.31, 4.32		

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Programming support for Rotational Position Sensing a new feature of IBM disk storage devices and support of two new job control commands, LFCB and LUCB is documented in this edition. Minor corrections have been made throughout the manual.

Note: *Please insert this page in your publication to provide a record of changes.*

IBM Laboratory, Publications Dept., Uithoorn, The Netherlands

Printed in U.S.A.

GC33-5380-1
File No. S370-37

Systems

DOS/VS Serviceability Aids and Debugging Procedures

Release 30

IBM

Second Edition (November, 1973)

This edition, together with Technical Newsletter GN33-8780, applies to Version 5, Release 30, of the Disk Operating System/Virtual Storage, DOS/VS, and to all subsequent versions and relea releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/360 and System/370 Bibliography,* GA22-6822, for the editions that are applicable and current.

Note: *For the availability dates of features and programming support described in this manual, please contact your IBM representative or the IBM branch office serving your locality.*

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 to IBM Laboratory, Publications Department, P.O. Box 24, Uithoorn, The Netherlands. Comments become the property of IBM.

GC33-5380-1

DOS/VS Serviceability Aids and Debugging Procedures (S370-37) Printed in U.S.A. GC33-5380-1

IBM

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

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

PREFACE

THIS MANUAL . . .

. . . is intended to guide System/370 operators and programmers using DOS/VS in determining and isolating the cause of a system malfunction.

METHOD OF PRESENTATION

Serviceability aids and how to use them are described in this manual through extensive use of diagrams and examples. This enables fast retrieval of information and largely avoids the need to use other publications in order to analyze the dumps and printouts discussed.

Contents and addresses shown in the illustrations are subject to change and are shown only as an aid to offline debugging of DOS/VS release 29. IBM will not be responsible for any system malfunction resulting from a change made by the user of any contents or addresses of the tables and blocks described.

SUBJECTS COVERED

There are four major sections;

SECTION 1: Introduction, introduces the serviceability aids detailed in Section 2, and the debugging procedures described in Sections 3 and 4.

SECTION 2: Serviceability Aids, describes in detail the serviceability aids, showing in flowchart form how to use them, and recommending when to use them. Examples show how to analyze dumps and printouts in conjunction with the debugging procedures of Section 3 and 4.

SECTION 3: Debugging for Operators, consists of flowcharts that help the operator to isolate the cause of a system malfunction. The operator is instructed when to use the procedures of Section 2 to ensure that information is gathered from the system.

SECTION 4: Debugging for programmers, this section is divided into two parts:

Part 1 consists of checklists in flowchart form that recommend the method of analysis and choice of serviceability aids best suited to isolate the cause of a given type of system malfunction. An indication is made on the flowcharts when it is considered necessary to inform your IBM customer engineer when it is not possible to isolate the cause of an error. System information to be saved for the IBM CE is also listed at these points in the flowcharts.

Part 2 is a general description of the DOS/VS supervisor/problem program interface tables, information blocks and save areas. It shows how to locate these areas in a dump, and how to analyze the data during offline program debugging. Debugging aids for high level languages are described in publications dealing with the specific language.

PREREQUISITE KNOWLEDGE

Operators using this manual must be familiar with the following IBM publications:

DOS/VS Operating Procedures	*GC33 – 5378*
DOS/VS Messages	*GC33 – 5379*

Programmers using Section 4 must be familiar with the following IBM publications:

IBM System/370 Principles of Operation	*GA22 – 7000*
DOS/VS System Management Guide	*GC33 – 5371*

Other IBM publications referenced in this manual are listed in the bibliography at the back.

DOS/VS Serviceability Aids and Debugging Procedures

OVERVIEW OF CONTENTS

ABBREVIATIONS

HIR	Hardware Instruction Retry
ICA	Integrated Communications Adapter
ID	Identifier
IDAL	Indirect Data Address List
ILC	Instruction Length Code
IMPL	Initial Micro-Program Load
INT	Interrupt
INTVN	Intervention
INVAL	Invalid
I/O	Input/Output
IOCS	Input/Output Control System
IPL	Initial Program Load
IR	Individual Recording
IT	Interval Timer
JAI	Job Accounting Interface
JCC	Job Control Command
JCL	Job Control Language
JCS	Job Control Statement
JIB	Job Information Block
K	1024 Bytes (Dec)
KBD	Keyboard
LDL	Local Directory List
LIK	Logical Transient Owner Identification Key
LIOCS	Logical Input/Output Control System
LMT	Line Mode Table
LOC	Location
LTA	Logical Transient Area
LTK	Logical Transient Key
LUB	Logical Unit Block
MCAR	Machine Check Analysis and Recovery
MCI	Machine Check Interrupt
MCK	Machine Check
MDR	Miscellaneous Data Record
MFCM	Multifunction Card Machine
MICR	Magnetic Ink Character Reader
MPS	Multiprogramming System
MPX	Multiplexer
MSG	Message
NICL	Number in Class List
NSD	Non Sequential Disk
OC	Operator Communication
OCR	Optical Character Reader
OD	Output Device
OLTEP	Online Test Executive Program
OLTS	Online Test System
PART	Partition
PC	Program Check
PCI	Program Controlled Interrupt
PCIL	Private Core Image Library
PD	Problem Determination
PDAID	Problem Determination Aid
PDS	Page Data Set
PER	Program Event Recording
PF	Page Frame
PFT	Page Frame Table
PFTX	Page Frame Table Extension
PG	Page
PGM	Program
PHO	Page Fault Handling Overlap
PIB	Program Information Block
PIB2	Program Information Block Extension
PIK	Partition Identification Key
PIOCS	Physical Input/Output Control System
PMGR	Page Manager
POWER	Priority Output Writers, Execution Processors, and Readers
POWER RJE	Power Remote Job Entry
PP	Page Pool
PPBEG	Start of Problem Program Area

ABBREVIATIONS

**CE, SE, IBM CE/SE is the IBM representative*

PRT	Partition
PSLD	Private second level directory
PSW	Program Status Word
PT	Page Table
PTA	Physical Transient Area
PTF	Program Temporary Fix
PTR	Pointer
PUB	Physical Unit Block
QTAM	Queued Telecommunication Access Method
RAS	Reliability, Availability, and Serviceability
RDE	Reliability Data Extractor
REQID	I/O Requestor Partition or System Task Identity
REQD	Required
RF	Recorder File
RID	Routine Identifier
RLD	Relocation Dictionary
RMS	Recovery Management Support
RMSR	Recovery Management Support Recorder
RPG	Report Program Generator
RPS	Rotational Position Sensing
RTN	Routine
SAB	Seek Address Block
SCP	System Control Program
SCU	Secondary Control Unit
SDAID	System Debugging Aid
SDL	System Directory List
SE	System Engineer*
SEREP	Stand-Alone EREP
SIO	Start I/O
SLD	Second Level Directory
SLI	Suppress Length Indication
SPVR	Supervisor
SRI	System Recovery Incident
STAB	Segment Table
STMT	Statement
SVA	Shared Virtual Area
SVC	Supervisor Call
SYSCOM	System Communication Region
SYSREC	System Recorder File
SYSRES	System Residence Unit
SYSVIS	Page Data Set
TCB	Translation Control Block
TES	Tape Error Statistics
TIB	Task Information Block
TIC	Transfer in Channel
TIK	Task Interrupt Key
TKREQID	I/O Requestor's Task Identity
TOD	Time of Day
TP	Teleprocessing
TPER	Teleprocessing Error Record
TXT	Text
UCS	Universal Character Set
UCSB/UCB	Universal Character Set Buffer
UPSI	User Program Switch Indicator
VDU	Virtual Display Unit
VSAM	Virtual Storage Access Method
VS	Virtual Storage
VTOC	Volume Table of Contents
WTM	Write Tape Mark
X' '	Hexadecimal Value
YR	Year

STORAGE DUMPS

A dump is a program or an operation that prints the image, in hexadecimal format, of a selected area of virtual storage. This term is also used when an area of virtual storage is recorded or stored on magnetic tape or on a disk pack

Figure 1-4 illustrates the various type of dumps offered by IBM. Section 2-A of this manual describes how to execute the dump programs and operations, and discusses the meaning of dump output that is useful during offline program debugging.

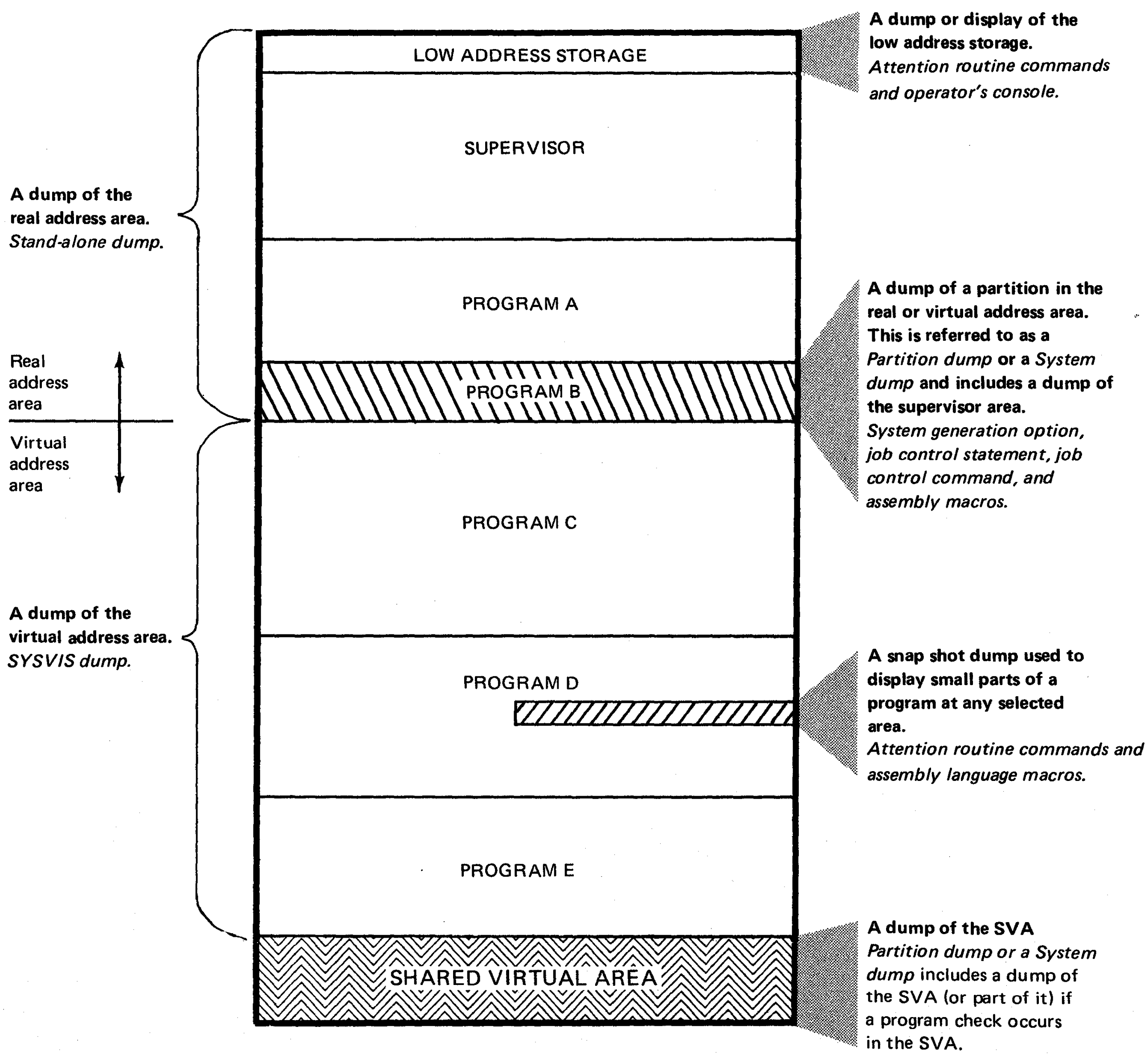


Figure 1-4. Storage dumps.
Various areas of storage can be dumped or displayed using the IBM dump programs and console aids.

(The dividing point between real and virtual address areas depends on the size of the hardware memory on your System/370).

LOOP TRACING

Three methods of tracing or recording the path of a loop are provided on the System/370:

1. By using the facilities provided by the operator's console, the operator can list the addresses of the instructions used by the loop.
2. By using the successful branch routine of the SDAIDS.
3. By using the instruction fetch trace of the SDAIDS.

All three methods are described in Section 2. The first method is useful to trace small loops during hands-on debugging. However, the amount of time that may be spent tracing a loop by this method depends on the answer to the following:

1. How important is it to system operation that the loop be fully traced?
2. How will the time spent tracing the loop affect system commitments?

Normally the operator is not in a position to answer these questions and if the programmer or the DP manager is not available, he can only take a short trace. The second and third methods can be used either during hands-on debugging, or during re-runs of the program generating the loop.

A note to operators

Before tracing a loop by using any of the above methods, you must consider their effects on time-dependent programs currently running in the system. Such programs are, for example, those using magnetic ink character recognition or teleprocessing equipment as input/output devices.

Guidelines on how to isolate an unintended loop and trace it are given in flowcharts in Sections 2 and 3.

Restrictions

1. If the sixteen bytes cross the boundary from a valid to an invalid address space (see the third restriction, below) only the bytes in the valid address space are displayed, and the following message is issued on SYSLOG:

   1148I XX BYTES COULD ONLY BE DISPLAYED

2. If the highest available virtual storage address is exceeded before sixteen bytes are printed, the command is terminated. However, the contents of those bytes that fall within the virtual address area are printed.

3. If the specified address is within an invalid address space the following message is issued on SYSLOG:

   1141D INVALID ADDRESS

   The definition of invalid address space is listed under item three of "Restrictions" in the description of the ALTER command.

When to use

This aid can be used during hands-on debugging, or an operator can be instructed to use it at specific addresses in a program.

For instance, loop count areas, small areas modified by loops, or parts of I/O areas can be dumped or displayed during program execution. The dump information will help during offline program debugging.

Dumps of, and changes to Real and Virtual Address Areas

OPERATOR COMMANDS (DUMP)

The DUMP Command

To activate the DUMP command press the REQUEST key and enter DUMP. The command allows the operator to display large areas of virtual storage on SYSLST. The SYSLST used may be assigned to any partition, but it must be a printer and it should not be in use by the partition. If the same printer is being used by the partition, the printed output will be a mixture of dump and partition output. If SYSLIST is assigned to a 3211 printer and the printer's indexing feature is being used, a certain number of characters may get lost on each line of the dump. To avoid this, the operator should load a new FCB (forms control buffer) image to disable the indexing feature before he issues the DUMP command. The new FCB image can be loaded either by issuing an LFCB command or by executing the SYSBUFLD program.

Operation	Operand
DUMP	[S BG Fn BGS FnS PDAREA xxxxxx, xxxxxx] [(BG) (Fna')] * Note 1

n,n' = 1,2,3,4

Note:

If the first operand is omitted, the general registers, control registers, and all storage that is currently used by programs, except that used by the supervisor (unless the operand BGS or FnS is specified), will be dumped. See note 2. The storage used consists of:

1. *Real storage not belonging to the page-pool*
2. *The virtual partitions in which a program is currently running.*

Description of the operands:

Operand	Meaning
S	Causes a dump of storage used and the supervisor area. See note 2
[BG Fn]	Causes a dump of the specified partition and its associated registers. If a real-mode program is running in the specified partition, the temporary real partition is dumped. If a virtual-mode program is executed in the specified partition, the whole virtual partition is dumped.
[BGS FnS]	Causes a dump of the same areas as described for the BG/Fn operand. However, the dump will include the supervisor area.
PDAREA	The PD area and the registers will be dumped (See Section 2, B-3 for details and a description of the PD area.)
XXXXXX, XXXXXX	Specifies the starting and ending address of virtual storage, with associated registers, that is to be printed. If the starting address is not on a fullword boundary, the address is rounded down to the first fullword boundary; if the ending address is not on a fullword boundary, the address is rounded up to the first fullword boundary. A minimum of one fullword is dumped, beginning at the start address.

Note 1:

BG F1 F2 F3 F4	*When any of these additional operands are specified, the area of virtual storage specified by the first operand is dumped on the SYSLST assigned to the partition specified by this operand. SYSLST must be a printer and should not be in use by its assigned partition. If the same printer is being used by the partition, the printed output will be a mixture of dump and partition output. (If this operand is not specified, the SYSLST printer assigned to BG is used. See note 2.)*

Control statements for the DUMPGEN operands

Control statements may be specified in any order; however, the following rules apply:

- All statements may be omitted, but if they are, DUMPGEN assigns printer X'00E', INTR=NO, FORMAT=NO, and PPOOL=NO options.
- Only one operation and only one operand per control statement is allowed.
- The last statement processed of a duplicate operation overrides all previous statements of the same operation with similar operands (if DECKS=2 is followed by DECKS=5, five decks are punched).
- The name field must be blank.
- Decimal operands may contain leading zeros.
- One of more blanks must follow the operand if comments are to be made.

Job stream example

The following example is a typical job used to create a stand-alone dump.

```
// JOB
// EXEC DUMPGEN
Col. 2
ASSGN SYSLST, X'00F'
OPTN FORMAT=YES
OPTN PPOOL=YES
OPTN DECKS=5
/*
/&
```

Note: If a 3221 is the only printer in your installation, the indexing feature should be used with great care; shifting the print line to the left or too far to the right causes loss of a certain number of characters on each line of the dump.

DUMPGEN messages

The functions of DUMPGEN-to-operator error message routines are:

- Cancel the job if SYSLOG is not a 3215/3210 or a System/370 Model 125/115 video display unit.
- Reissue the message if operator response is to press the CANCEL key
- Process an operator response of END/ENTER as IGNORE
- Cancel the job if operator response is CANCEL.
- Ignore the control card in question when the operator response is IGNORE.

If none of the preceding operator responses is issued, then DUMPGEN assumes that a correction has been made and processes it.

Dumps of, and changes to Real and Virtual Address Areas

DUMPGEN AND STAND-ALONE DUMP

Stand-alone Dump Program

This program is generated for your installation using the IBM program DUMPGEN.

DUMPGEN produces a dump program that is either punched into a card deck or stored on magnetic tape. When required, the dump program thus generated can be loaded into the system via the standard IPL procedure.

The stand-alone dump program that is generated by DUMPGEN provides either a conventional dump or a formatted dump, depending on the FORMAT option used in the DUMPGEN program.

Operation

During execution of the stand-alone dump program, a non-critical area in the supervisor is used to load the program. The LOAD ADDRESS of the non-critical area is punched (in decimal) in the first card of the stand-alone dump card deck punched by the DUMPGEN program. Because of this use of the non-critical area it is recommended to use the stand-alone program for a system using a supervisor that was used for the generation of that dump.

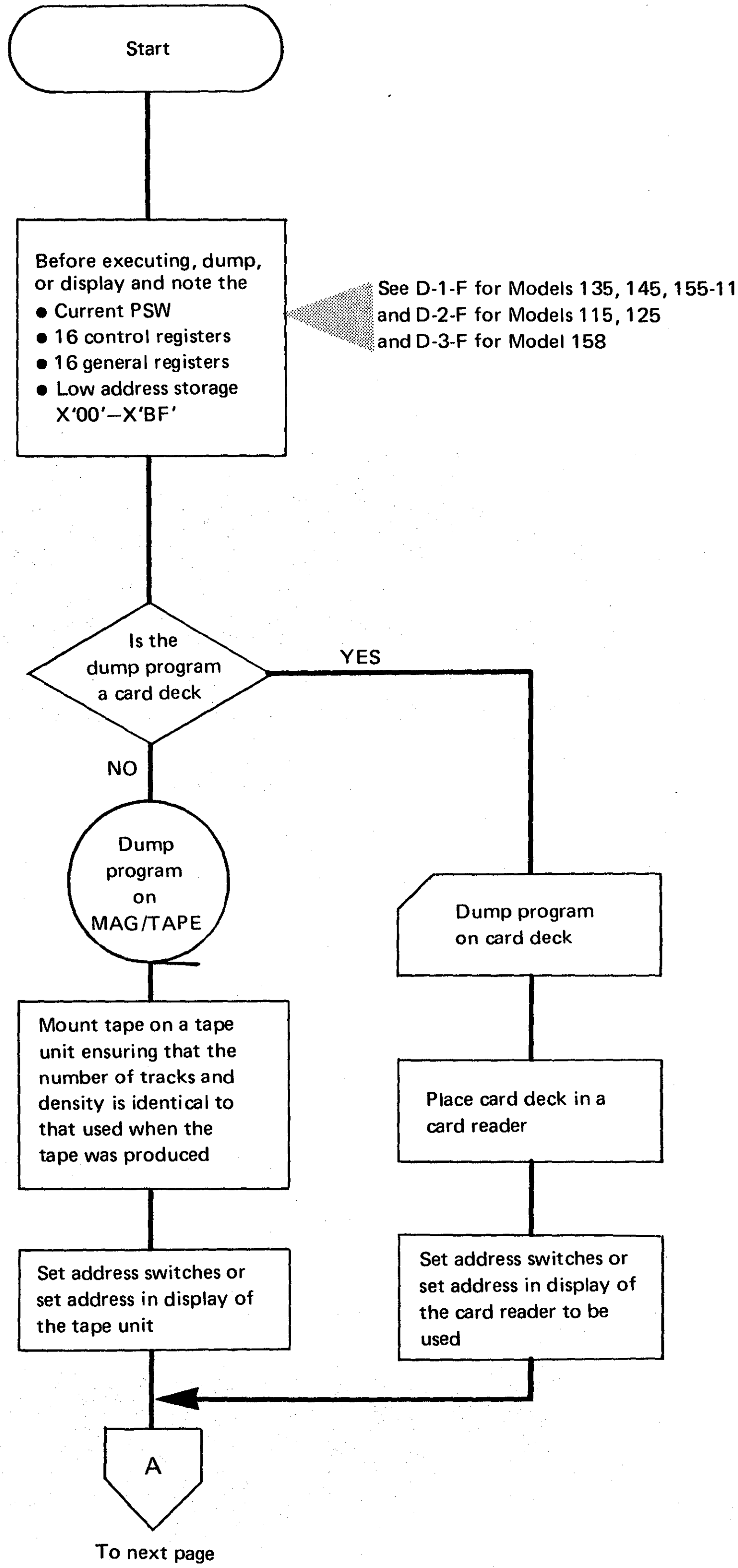
The conventional dump prints the contents of real storage locations, but does not dump the floating point registers. In addition to the areas dumped by the conventional dump, the formatted dump prints the DOS/VS interface tables in a more readable form.

For both types of dump the following is printed:

1. The contents of the general registers, the old and new PSWs, the interruption codes, CSW, CAW, and TIMER.
2. The contents of real storage in 2k blocks. Each block is preceded by a sequence number.
3. At the end of the real storage dump, page address and status information is printed that contains the following information for each page frame:
   - The virtual address
   - The real address of the associated page
   - The sequence number of the 2k block
   - Information that indicates whether the contents of the page frames has been changed.
4. The contents of the control registers are printed after page address and status information.
5. Depending on the options selected, the following then occurs:

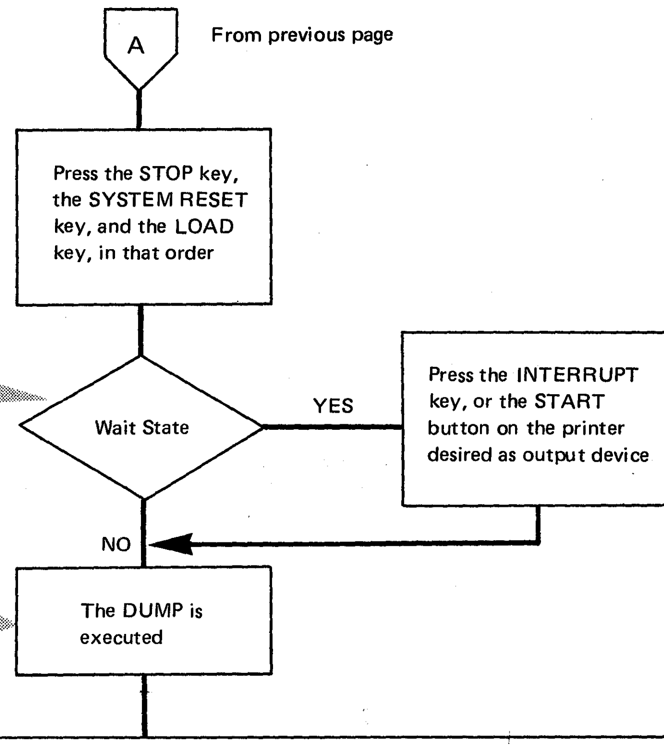
   If PPOOL=YES
   - The formatted contents of the boundary box is printed after the control registers.
   - The contents of real storage is printed in 2k blocks in sequence of ascending virtual addresses.

DUMPGEN AND STAND-ALONE DUMP



Dumps of, and changes to Real and Virtual Address Areas

DUMPGEN AND STAND-ALONE DUMP



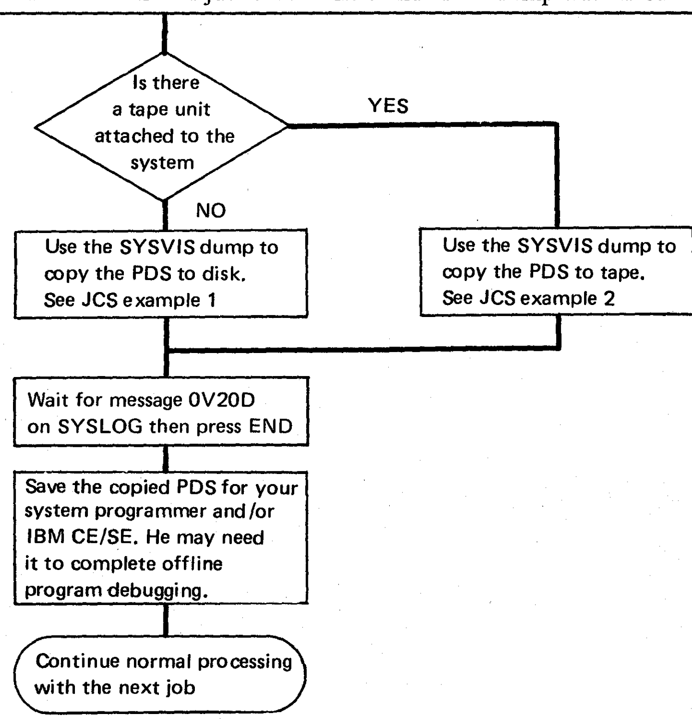
A wait state occurs if INTR YES is specified during generation of the dump program using DUMPGEN.

The contents of REAL storage is printed in blocks of 2K bytes on the line printer specified in the ASSGN statement of DUMPGEN or the line printer made ready after entering the WAIT STATE. Other information is also printed depending on the type of dump program generated by DUMPGEN.

The SYSVIS dump utility program is described under C-6 in this Section

The following procedure executes the SYSVIS dump utility program. This procedure copies the PDS to tape or disk which can then be dumped to SYSLIST if required during offline debugging. It is also possible to dump the PDS directly to SYSLST. To ensure that the contents of the PDS and the allocations of the virtual address area are the same as they were just prior to the execution of the stand-alone dump the following instructions must be adhered to:

1. Re-IPL using identical parameters for the DPD command as specified in the previous system IPL. **However you must specify N to the parameter TYPE=.**
2. Check for any previous ALLOC commands. You must specify identical virtual partition allocations as existed just before the stand-alone dump was executed.

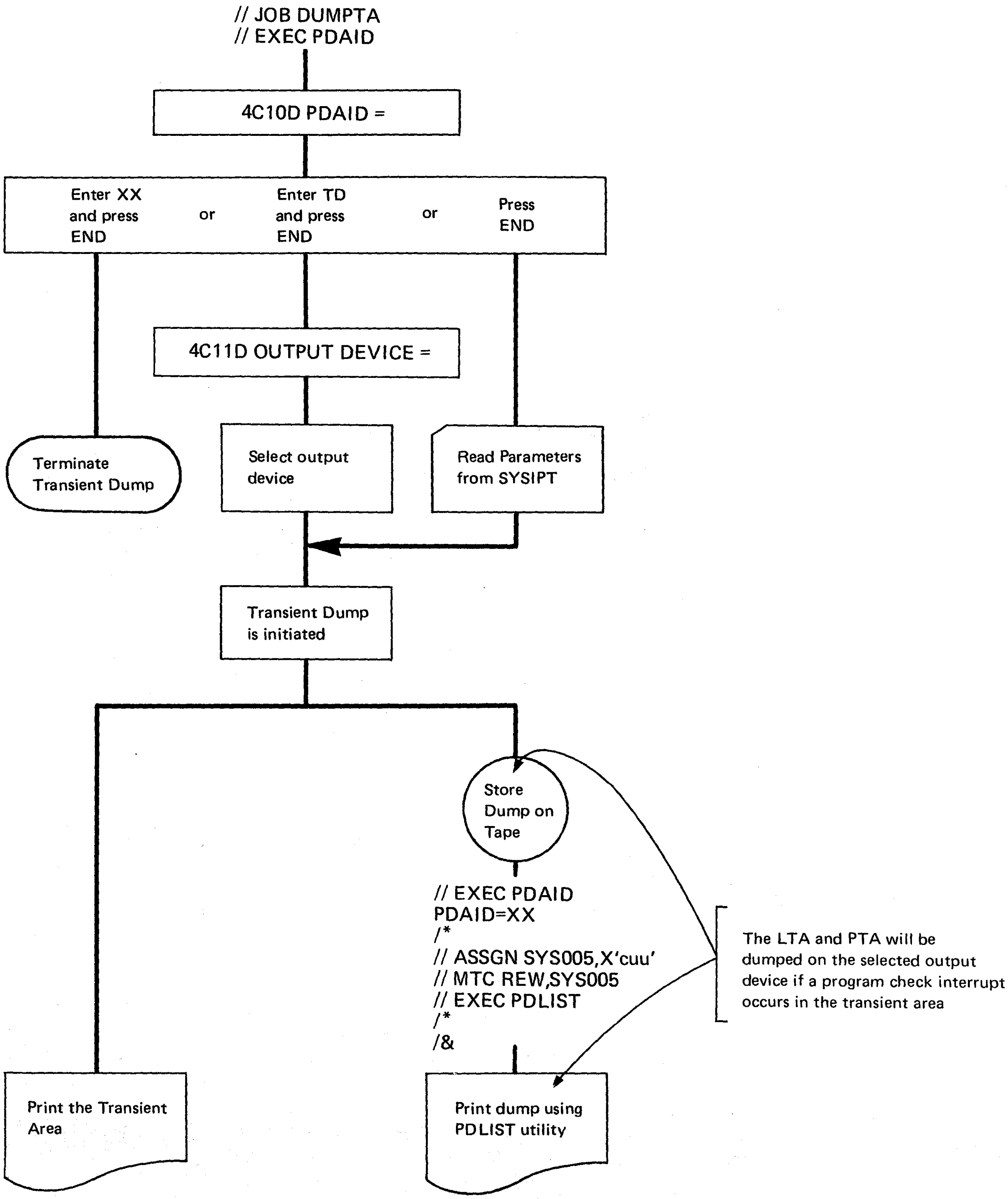


Example 1

```
// JOB DUMPPDS
// ASSGN SYS000,X'cuu'
// ASSGN SYS001,X'cuu'
// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// DLBL S01DISK,'BACKUP FOR PDS',
// EXTENT SYS001,111111,1,,0020,180
// EXEC PDSDM
/&
```

Example 2

```
// JOB DUMPPDS
// ASSGN SYS000,X'cuu'
// ASSGN SYS001,X'cuu'
// DLBL PDSDISK,'PAGE DATA SET'
// EXTENT SYS000
// TLBL S01TAPE,'BACKUP FOR PDS'
// EXEC PDSDM
/&
```


// JOB DUMPTA
// EXEC PDAID
4C10D PDAID =
Enter XX and press END
or
Enter TD and press END
or
Press END
4C11D OUTPUT DEVICE =
Terminate Transient Dump
Select output device
Read Parameters from SYSIPT
Transient Dump is initiated
Store Dump on Tape
// EXEC PDAID
PDAID=XX
/*
// ASSGN SYS005,X'cuu'
// MTC REW,SYS005
// EXEC PDLIST
/*
/&
The LTA and PTA will be dumped on the selected output device if a program check interrupt occurs in the transient area
Print the Transient Area
Print dump using PDLIST utility

A-4-F

Dumps of, and changes to Real and Virtual Address Areas

TRANSIENT DUMP

Job stream examples

The following two examples show job stream to initiate the transient dump program.

Example 1, via SYSIPT:

// JOB CARDINP7	
// EXEC PDAID	Calls for initializer.
PDAID=TD	Calls for transient dump function.
OUTPUT DEVICE=00E	Specifies printer output.
GO	Signals end of input.
/*	
/&	

Note: A dump is given on all program checks.

Example 2, via SYSLOG:

// JOB TYPINPT6	
// EXEC PDAID	Calls for initializer.
PDAID=	Console requests function.
TD and END key	Operator specifies transient dump function.
OUTPUT DEVICE=	Console requests output device.
00E and END key	Operator specified printer output.

An output device must be specified for the transient dump fiction. If this is a 3211 printer and the printer's indexing feature is used, it may occur that not the full length of every line of the dump is printed. This loss of characters can be avoided by disabling the indexing feature. This is brought by loading a new FCB image into the printer's FCB in one of the following ways;

Using the SYSBUFLD program.
This method is to be used when the transient dump is entered via SYSIPT. The job stream as shown in example 1, would than have to be as follows:

```
// JOB
// EXEC SYSBUFLD
   FCB   SYSxxx,phasename
/*
// JOB CARDINP7
// EXEC PDAID
   •
   •
```

Using the LFCB command

This method is to be used when the transient dump is initialized by means of the console printer keybord (SYSLOG). The job stream as shown in example 2 would then be as follows:

```
LFCB X'cuu',phasename
// JOB TYPINPT6
// EXEC PDAID
   •
   •
```

phasename = the name by which the FCB image is cataloged.

Two series of trace routines are provided on the System/370: PDAIDS and SDAIDS

These aids enable information to be obtained from the system at the time of a malfunction. They are aids for further error isolation, and are usually initiated during a rerun of a troublesome program after a first analysis of the problem. The type of trace to use for a particular problem depends on the result of the first analysis and how much more information is required to help in further isolation of the error.

This section is divided into two parts:

Part 1 describes the PDAIDS, and part 2 describes the SDAIDS.

PART 1 PDAIDS

General description

There are four trace routines that can produce printed output of certain events which occur during the execution of programs.

The trace routine will:

- Record I/O operations (I/O trace)
- Record the order in which phases and transients are called (Fetch/Load trace)
- Record the order in which supervisor calls (SVCs) are executed (Generalized SVC trace)
- Record the order in which either an SVC 0 or an SVC 31, and I/O interrupts occur. (QTAM trace).

On the occurrence of an event, an entry is generated which, by selection of the trace, can be recorded on magnetic tape, printed on a line printer, or preserved either in the PD area or, if specified, in an alternate area of real storage.

Caution

The effect on the operation of programs currently running in the system that are time dependent, for example, a program using MICR or teleprocessing as input/output, must be considered before using this serviceability aid.

Trace Routines

PDAIDS

System requirements

Before any PDAID function can be executed, the following requirements must be met:

- During the system generation, specify a minimum value of 1400 in the PD parameter of the FOPT macro. (The maximum value is 10,240).
- If data provided by the trace routines is recorded on magnetic tape, use the PDLIST program after tracing is complete to obtain a printout of the tape.

All PDAID modules are distributed by IBM in the core image library. They are self-relocating for initialization in any real or virtual partition (6K or greater) of a multiprogramming system.

Restrictions: More than one PDAID trace routine cannot operate concurrently. This also applies to the PDAID Transient Dump program described in Section 2-A-4. Therefore, more than one program rerun must be executed if more than one PDAID function is used to gather information about a failing program.

Using PDAID and SDAID concurrently: IF SDAID is active it must first be terminated before initiating a PDAID trace in core-wrap output mode in an alternate area.

Modes of output

Line printer: (not available as output mode for QTAM trace) Examples in this section show the trace outputs when the output device is a line printer. An asterisk on the print-out indicates that at least one event (trace entry) has been overwritten. This occurs when an overflow is caused in the trace table in the PD area (described in B-3) or in an alternate area. This may occur when the trace output device, or its control unit, or channel, is shared with other programs running simultaneously.

If the printer is not ready or has an error condition, message 4C24A NO I/O TO OD is printed on SYSLOG and the system waits for the END/ENTER key to be pressed after the printer is made READY.

Magnetic tape: This mode of output collects and writes on an unlabled tape the trace entries that occur during execution of a job stream.

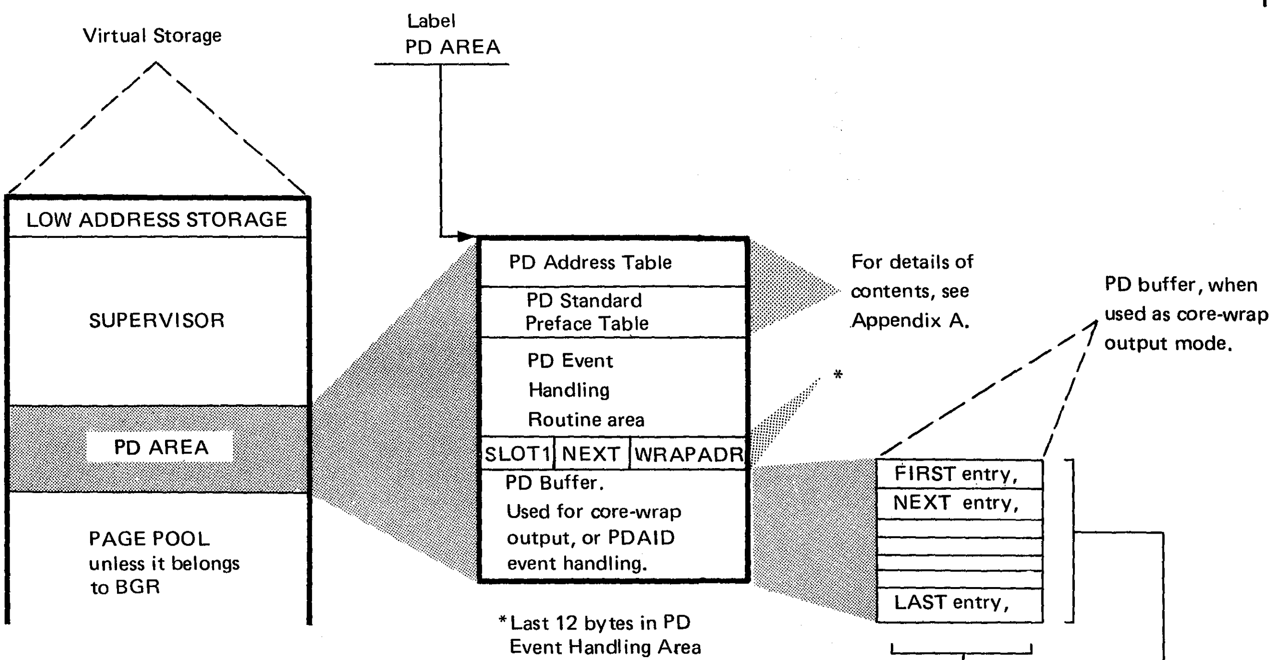
The events are written on tape in core image (unprintable) format.

The tape must be processed using the PDLIST utility. The tape unit must be assigned temporarily or permanently to SYS005 and SYSLST assigned temporarily or permanently to a line printer in order to obtain readable listings of the events traced. Examples in this section show the output format after using the PDLIST utility.

If the tape unit is not ready or has an error condition the message

4C24A NO I/O TO OD

is issued on SYSLOG and the system waits for the END/ENTER key to be pressed after the tape drive is made READY.



Note:
See Section 4 for details about the supervisor organization.

Trace type.	SLOT1 Contains address of FIRST entry in PD Buffer. It is found at:	NEXT Contains address of NEXT available entry in PD Buffer. It is found at:	WRAPADR Contains address of LAST entry in PD Buffer. It is found at:	Length of one entry in bytes;	Maximum number of entries in the minimum PD area of 1400 bytes (dec)
Input/output	PDAREA + 1B4	PDAREA + 1B8	PDAREA + 1BC	18 (dec)	52
Fetch/load	PDAREA + E8	PDAREA + EC	PDAREA + F0	22 (dec)	52
GSVC	PDAREA + X'114'	B = A + 4	C = A + 8	18 (dec)	61
QTAM	PDAREA + '1CØ'	B = A + 4	C = A + 8	21 (dec)	44

Notes:

1. *NEXT – address of the next available entry in the PD Buffer. The NEXT entry in the PD Buffer contains either the oldest entry in the table, or the most recent entry of a device, SVC, or partition being ignored (all entries are placed in the NEXT entry before they are checked for trace or ignore). If the latter is the case, ignore the entry.*

2. *When the LAST entry is filled, the address in SLOT1 is loaded into NEXT and the buffer is overwritten by new entries.*

Table B-3 Trace entry locations and lengths for core-wrap output mode in the PD area.

PDAIDS

Initiating the PDAID trace routines

You can initiate PDAID trace routines by using standard DOS/VS job control languages from either SYSLOG or SYSIPT. The statement

// EXEC PDAID

causes the main phase (PDAID) to be loaded at the address of the initiating partition. Control is given to the PDAID for further specifications to indicate the type of trace to be performed.

The options and control statements for the trace routines may be entered through SYSLOG or through the device assigned to SYSIPT.

If a card reader is used as SYSIPT, the card deck must be punched as follows:

Entries may be punched one-per-card, or as multiple entries (separated by commas) in a single card. An entry may not be split between two cards. All 80 columns of a card may be used, but a card is terminated either by the first blank following an entry, or by a GO entry. The last card must be followed by a /* CARD.

Note: If an incorrect parameter is read from SYSIPT, corrections are requested on SYSLOG.

When the initializing phase (PDAID) has been loaded, the following message is issued on SYSLOG:

4C10D PDAID=

The operator must respond to this message with one of the following:

IT Specifies an I/O Trace (See note 1.)
FT Specifies an F/L Trace (See note 1.)
GT Specifies a GSVC Trace (See note 1.)
QT Specifies a QTAM Trace (See note 1.)
TD Specifies the Transient Dump
(refer to A-4 in this Section)
XX Terminates the PDAID presently running
Pressing the END or ENTER key indicates that PDAID control statements are entered through SYSIPT (See note 2.)

Notes:

1. *When IT, FT, GT, or QT is specified, the operator must provide additional PDAID control statements through SYSLOG.*

2. *The END response is valid only for SYSLOG and cannot be used as a SYSIPT operand.*

3. *Multiple operands or operator responses to PDAID control statements for traces with a variable number of functions (such as ignoring SVCs) are not allowed. Repeat each parameter with each variable). Repeat each message until either the maximum number of variables is reached or an END response is given.*

4. *GO terminates the PDAID control input, and the default is taken for any PDAID options that are not specified. When you use SYSIPT, GO should be the last parameter, and it has no operand associated with it. A /* card must follow the GO operand.*

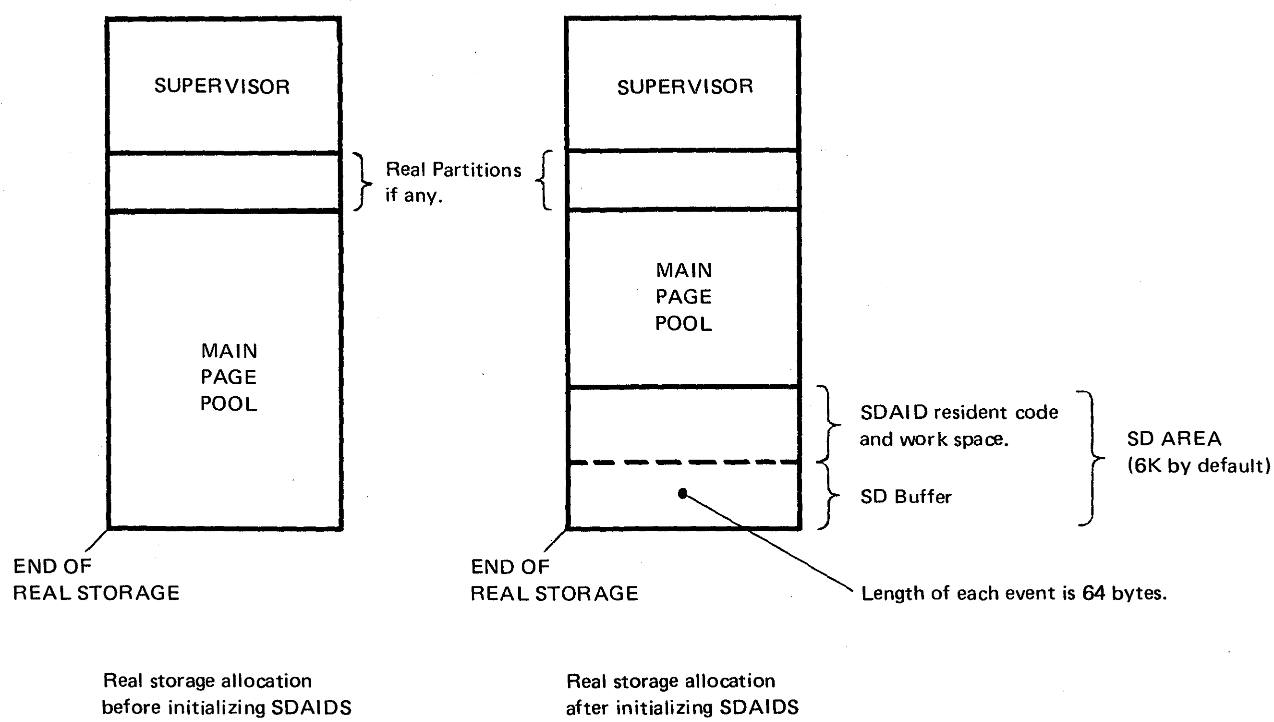
Dump on a program check: On the occurrence of a program check interrupt codes X'01' – X'0F', X'10', and X'12', the following information is dumped automatically:

- Event ID
- Program old PSW
- Time of day in microseconds
- Control registers
- General purpose registers
- Real storage from byte location 0 to the end of the supervisor area, and the contents of the SDAID buffer.

After this automatic non-destroying dump is executed, the DOS/VS program check handler routine will be entered.

When to use: If PDAIDS are not available on your system, the use of the SDAID dump on a program check is the only way to obtain a non-destroying dump of the supervisor transient area at the time of a program check interrupt.

The SD area



How to locate

The address of the beginning of the SD area is printed on the device assigned to SYSLST during initialization of SDAID. Refer to page 2.90 for an example of SDAID initializing output.

Trace Routines

SDAIDS

Initializing SDAID

SDAID may be initialized in any real or virtual partition by entering the following execute statement via SYSLOG or SYSRDR:

// EXEC SDAID

An operator/system dialog follows, beginning with the message:

4C55D GIVE SPACE FOR SDAID=

The operator may respond by pressing the END key (which gives a default value of 6K to SDAID), or he may specify a value nK, where n represents a multiple of 1024 bytes. The maximum value that may be specified is 999K. If an odd number is specified, the value is incremented to the next even number. The SDAID space is taken from the main page pool. If the main page pool is not large enough to accept the area specified, an area of the size PPOOL-16K is automatically taken,with a minimum of 6K. If the page pool is not large enough to accept the minimum(6K), the following message is printed on SYSLOG.

4C56E INSUFFICIENT SDAID SPACE, REALLOCATE

The MAP command should be issued before reallocating real partition areas in order to increase the size of the page pool before re-initializing SDAIDS.

The following message will be issued if this is a second request for SDAID space:

4C70E DUPLICATE REQUEST FOR PDAID AND/OR SDAID

This message is also issued if PDAID using the core wrap output mode in an alternate area is active in the main page pool and a request for SDAIDS is made. When the space allocated to SDAID is accepted, a message dialog follows that allows the operator to select one or more events to be traced and to specify between which address limits of real or virtual storage the events are to be traced. (Event limits do not apply to event PAGENQ, event PAGEHDL, and event PGMCHK.) The dialog also enables the selection of a line printer at a device address other than X'00E', which is the device address by default. However, the device must be a line printer.

An output class may also be specified (refer to Tables B-6-A and B-6-B in this chapter). A response of EOB (pressing the END key) to all SDAID messages will give default values.

When the SDAID message dialog is complete, the SDAID initializing outputs part 1 and 2 are issued to the device assigned as SYSLST. This need not be the same device on which SDAID trace output is printed. The SDAID trace output is printed immediately after the initializing output on the device at the address specified in the reply to message

4C58D OUTPUT DEVICE=

(Address X'00E' is taken as default.) After initialization, the partition used for the initialization is given back to the main page pool.

The table shown in Figure B-3 lists all SDAID messages in the order in which they are issued and describes the responses.

SDAID job entry examples are shown after the example of the SDAID initializing output. Operator flowcharts follow.

SDAID messages after initialization time

4C71I SDAID FOUND PRTR STATUS CSW SENSE

This message may be written out on the printer. It is accompanied by the CSW and SENSE information if applicable. It indicates that the previous printer operation which was started may not have been completed successfully.

Altering SDAID functions and/or address limits after initialization

When the SDAID is initialized, trace functions and events limits, where applicable, can be changed by altering the SDAID program parameters directly in storage. The contents of the parameters at the addresses printed on part 2 of the SDAID initializing output, and of control registers 8, 9, A, and B, must be altered to predetermined values. Their values are also printed in the initializing output.

To make SDAID parameter changes:

- Press the STOP key.
- Use the console ALTER/DISPLAY feature to alter the contents of the program parameters.
- Press the START key.

Note: When SDAID is terminated and later re-initialized, new SDAID parameters are printed in the SDAID initializing output.

Note: SDAID requires SYSLST for the initializing output. Therefore, if you intend to change SDAID parameters after initializing SDAIDS, you should ensure that the SYSLST device is a line printer on the partition used for SDAID initialization.

A note to programmers

SDAIDS are primarily designed to be initialized before re-running failing programs. If you, as the programmer, are debugging on the system (hands-on debugging), it is recommended that you initiate SDAIDS without specifying any events. (Press the END key as a response to all SDAID messages.) SDAID is then retained in the page pool ready to be activated. The failing programs can then be executed and SDAID events made active by entering event parameters directly into control registers 8, 9, 10, and 11. For example, altering the contents of the high-order byte of control register 9 (by the console ALTER/DISPLAY feature) enables you to activate any one or all of the events BR, IF, SA, and GA.

You can also specify which general registers are to be traced by entering values into the lower 2 bytes of control register 9. Control registers 10 and 11 contain, respectively, the start and end addresses for the event limits. The output of the MAP command will tell you the partition in which the failing programs reside.

From the MAP output you can also obtain the addresses of the upper and lower limit of the partition, which can then be used as the event limits for the SDAID trace. (Note that addresses printed by the MAP command are decimal.)

If you are unable to use the system for hands-on debugging, you as the programmer must specify clear instructions to the operator about the events to be traced and the event limits to be used.

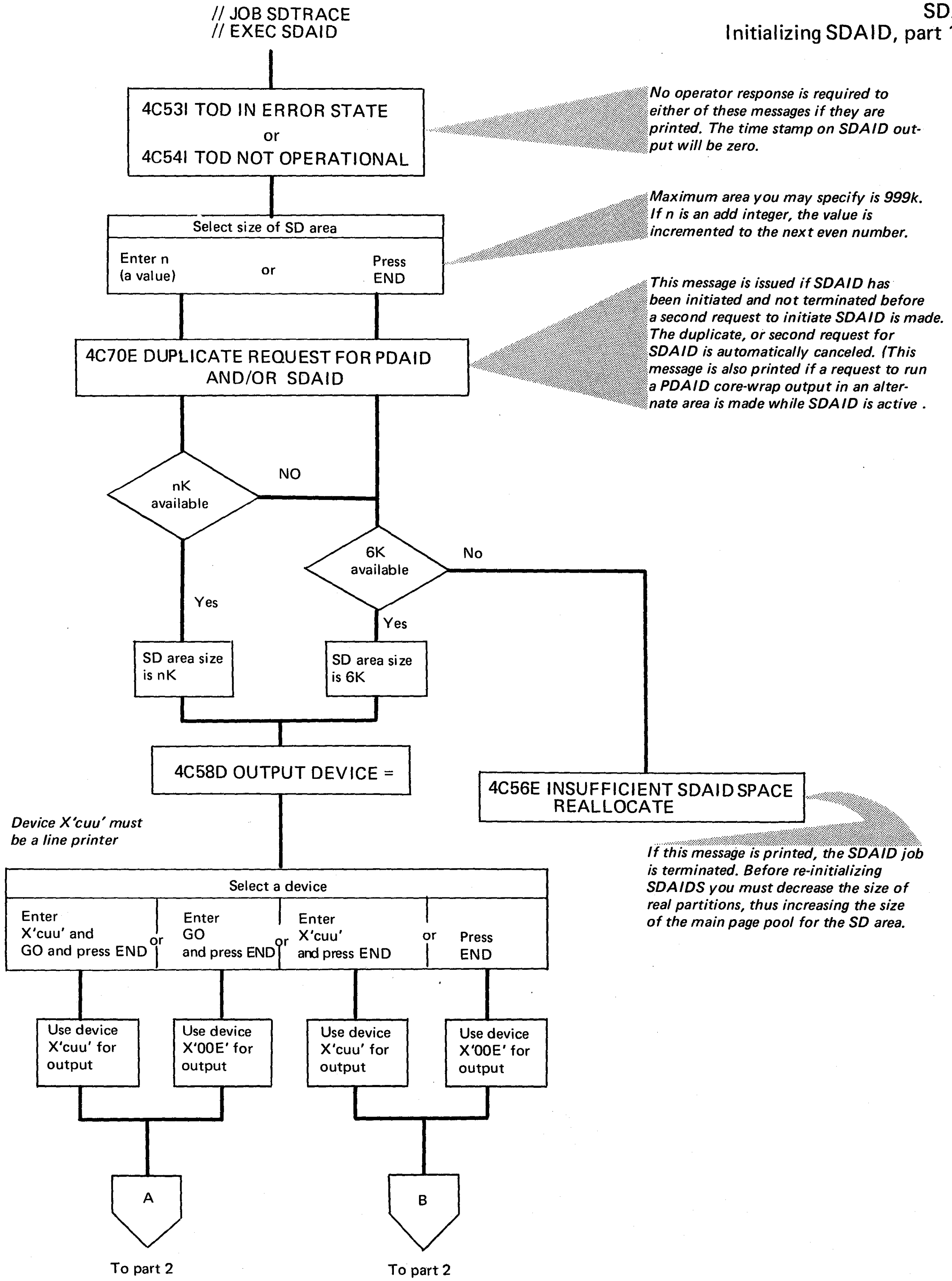
Trace Routines

SDAIDS

Message Code	Message issued on SYSLOG	Parameters entered by Operator	Remarks
4C58D	OUTPUT DEVICE=	X'00E' [,GO] X'cuu' [,GO] GO END/ENTER	
4C60D	STOP ON EVENT=	YES NO [,GO] END/ENTER	
4C61D	EVENT LIMITS=	X'000000',X'FFFFFF' [,GO] X'lllll' ,X'lllll' [,GO] ,X'hhhhhh' [,GO] ,GO END/ENTER	X'lllll',X'hhhhhh': Lower and upper limit of virtual storage to be traced with events BR, IF, SA and TE.
4C59D	OUTCL=	PSW GPR LOCORE COMREG PAGETAB SUPVISOR DUMPREAL [,GO] PDUMP ,lower and upper event limit [,GO] ,X'aaaaaa' ,upper event limit [,GO] ,X'bbbbbb' [,GO] ,GO ,GO FASTREC ,PGMCHK [,GO] ,AUTOMATIC [,GO] ,GO END/ENTER	Valid output classes for the events BR, IF, SA, GA, and TE. END/ENTER PGMCHK: Causes wrap around mode of internal buffer. It is written each time a PGMCHK event occurs. AUTOMATIC: If the internal buffers is full, it is written out.
4C62D	EVENT BR=	YES NO [,GO] END/ENTER	
4C63D	EVENT IF=	YES NO [,GO] END/ENTER	
4C64D	EVENT SA=	YES NO [,GO] END/ENTER	
4C65D	EVENT GA=	X'012...EF' [,GO] NO END/ENTER	Designate the general purpose registers to be traced. At least one must be specified.
4C66D	EVENT TE=	YES NO [,GO] END/ENTER	
4C67D	EVENT PGMCHK=	YES NO [,GO] END/ENTER	
4C68D	EVENT PAGENQ=	YES NO [,GO] END/ENTER	
4C69D	EVENT PAGEHDL=	YES NO END/ENTER	

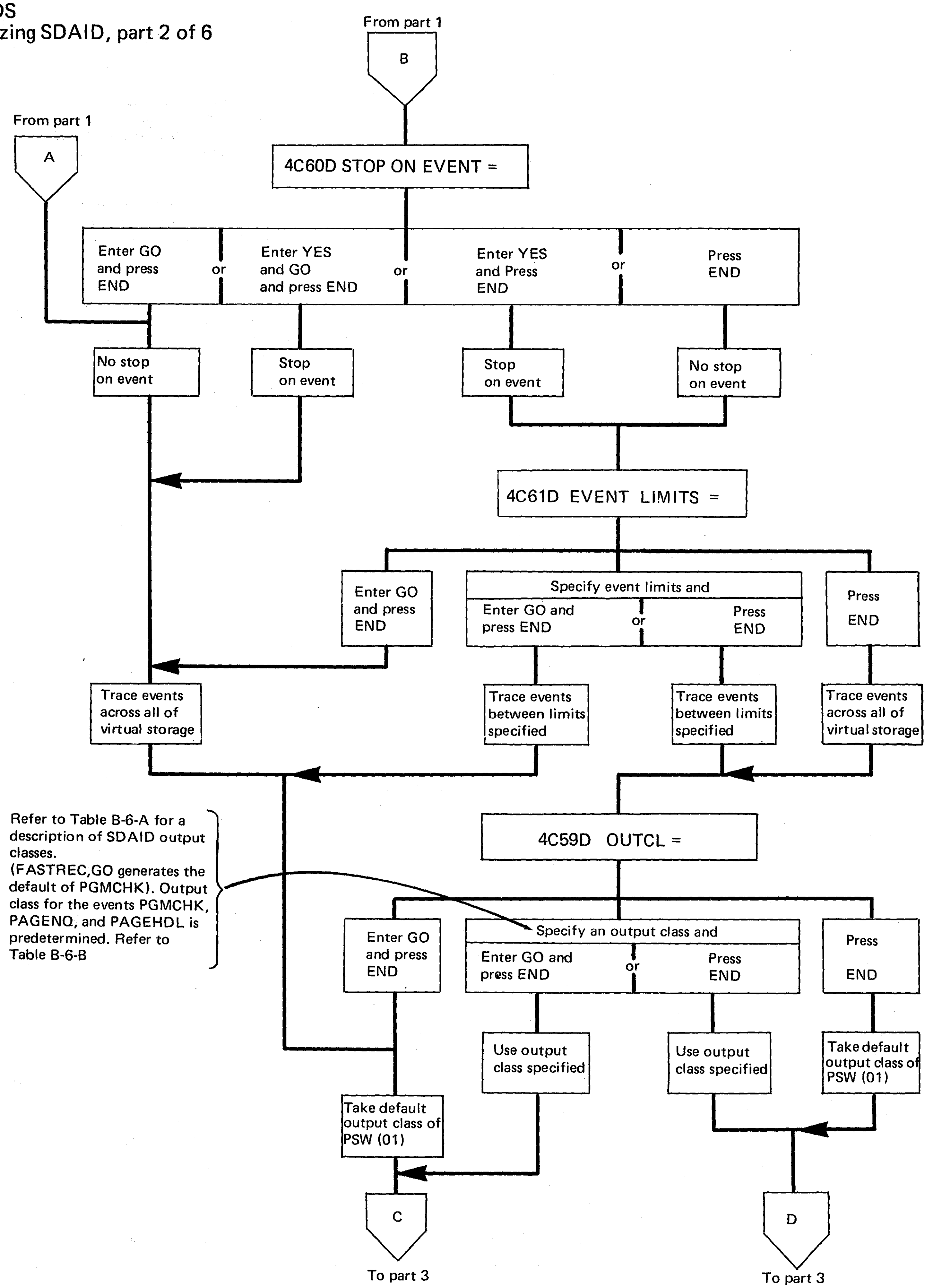
Note: Go cannot be entered as a first parameter. If it is, the dialogue is terminated ; defaults (underlined) are taken or the parameters are ignored by SDAID.

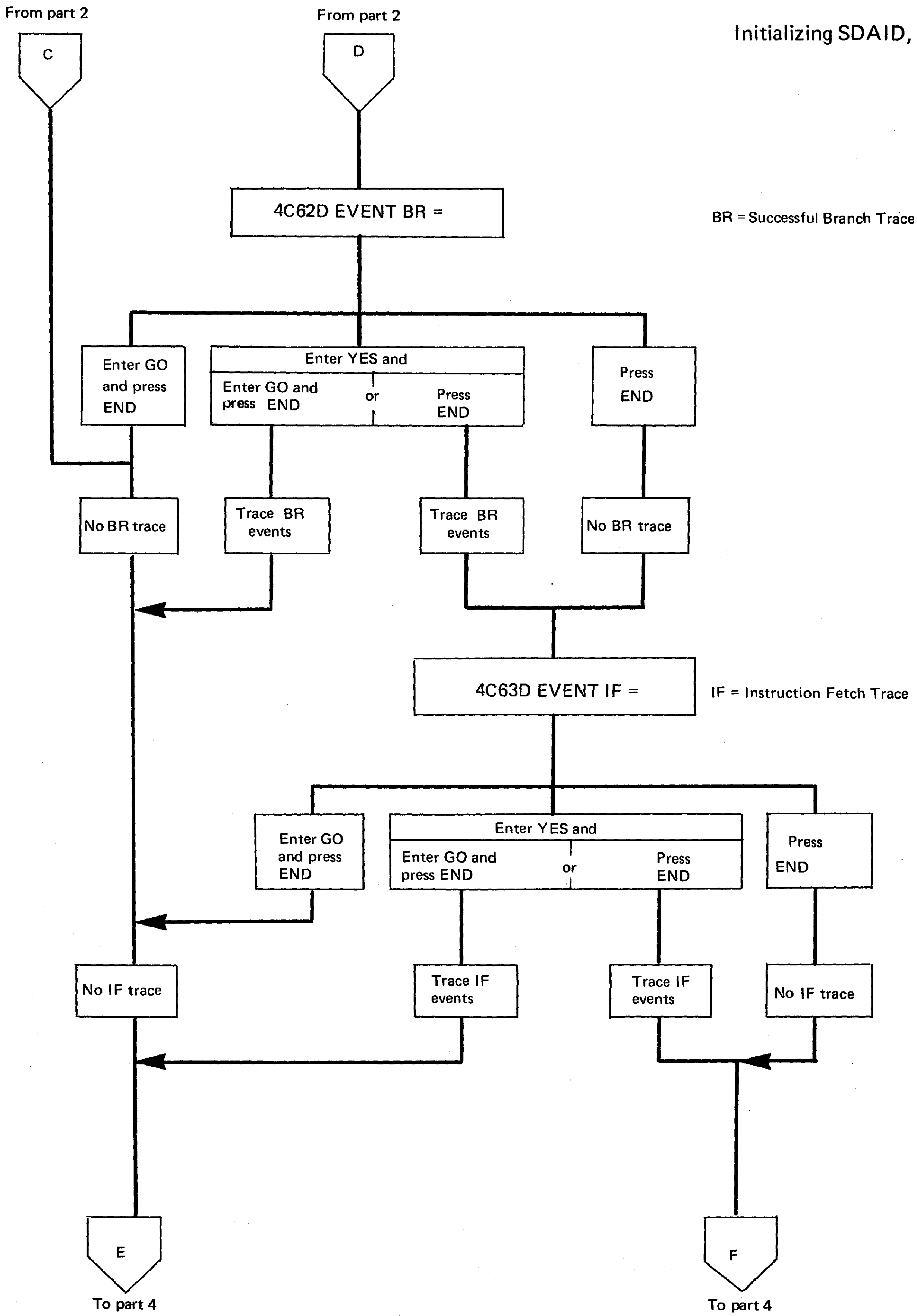
Table B-10. The parameters required to initialize SDAID event tracing.


// JOB SDTRACE
// EXEC SDAID
4C53I TOD IN ERROR STATE
or
4C54I TOD NOT OPERATIONAL
No operator response is required to either of these messages if they are printed. The time stamp on SDAID output will be zero.
Select size of SD area
Enter n (a value)
or
Press END
Maximum area you may specify is 999k. If n is an add integer, the value is incremented to the next even number.
4C70E DUPLICATE REQUEST FOR PDAID AND/OR SDAID
This message is issued if SDAID has been initiated and not terminated before a second request to initiate SDAID is made. The duplicate, or second request for SDAID is automatically canceled. (This message is also printed if a request to run a PDAID core-wrap output in an alternate area is made while SDAID is active .
nK available
NO
Yes
6K available
No
Yes
SD area size is nK
SD area size is 6K
4C58D OUTPUT DEVICE =
4C56E INSUFFICIENT SDAID SPACE REALLOCATE
If this message is printed, the SDAID job is terminated. Before re-initializing SDAIDS you must decrease the size of real partitions, thus increasing the size of the main page pool for the SD area.
Device X'cuu' must be a line printer
Select a device
Enter X'cuu' and GO and press END
or
Enter GO and press END
or
Enter X'cuu' and press END
or
Press END
Use device X'cuu' for output
Use device X'00E' for output
Use device X'cuu' for output
Use device X'00E' for output
A
B
To part 2
To part 2

Trace Routines

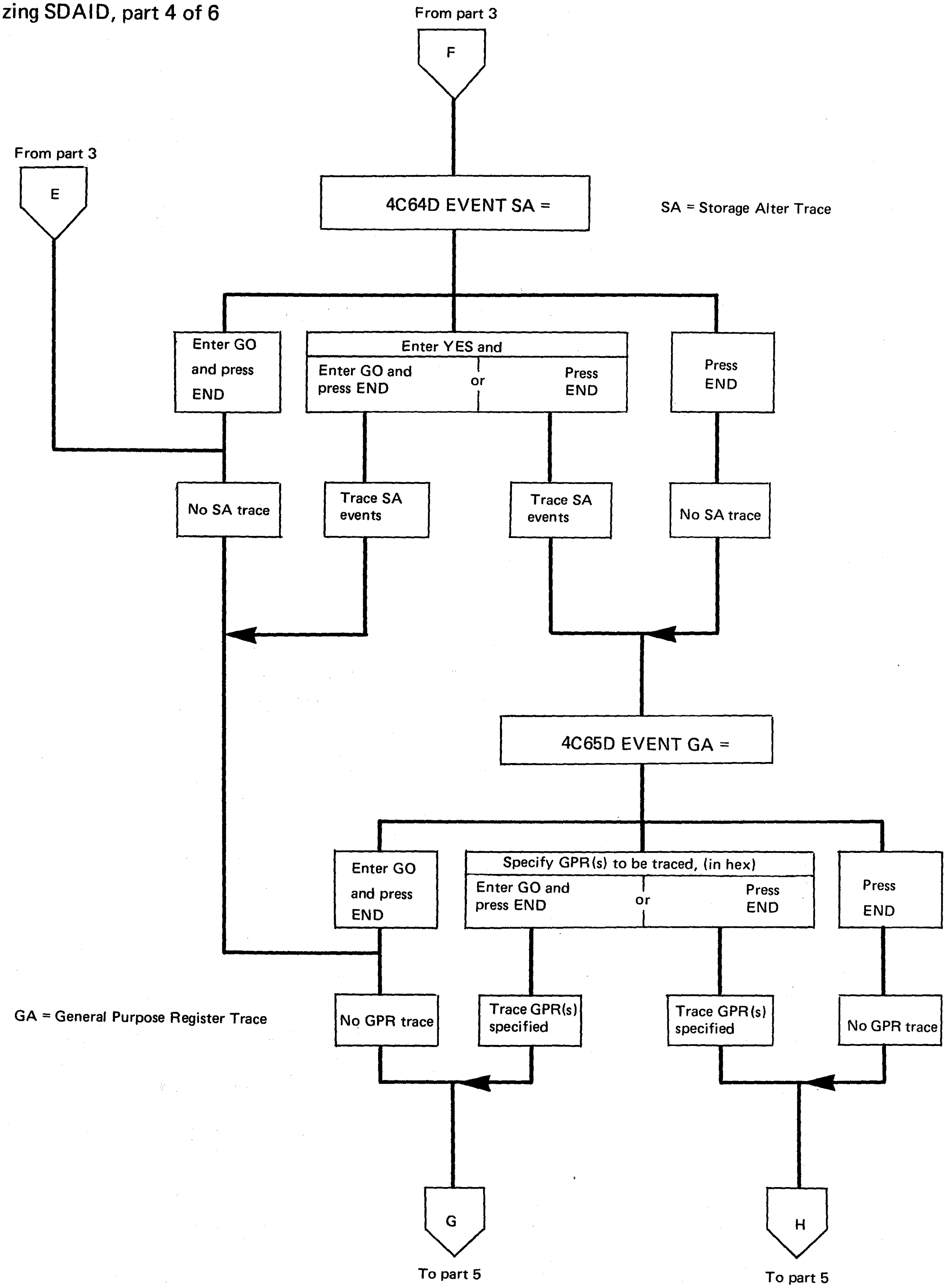
SDAIDS
Initializing SDAID, part 2 of 6

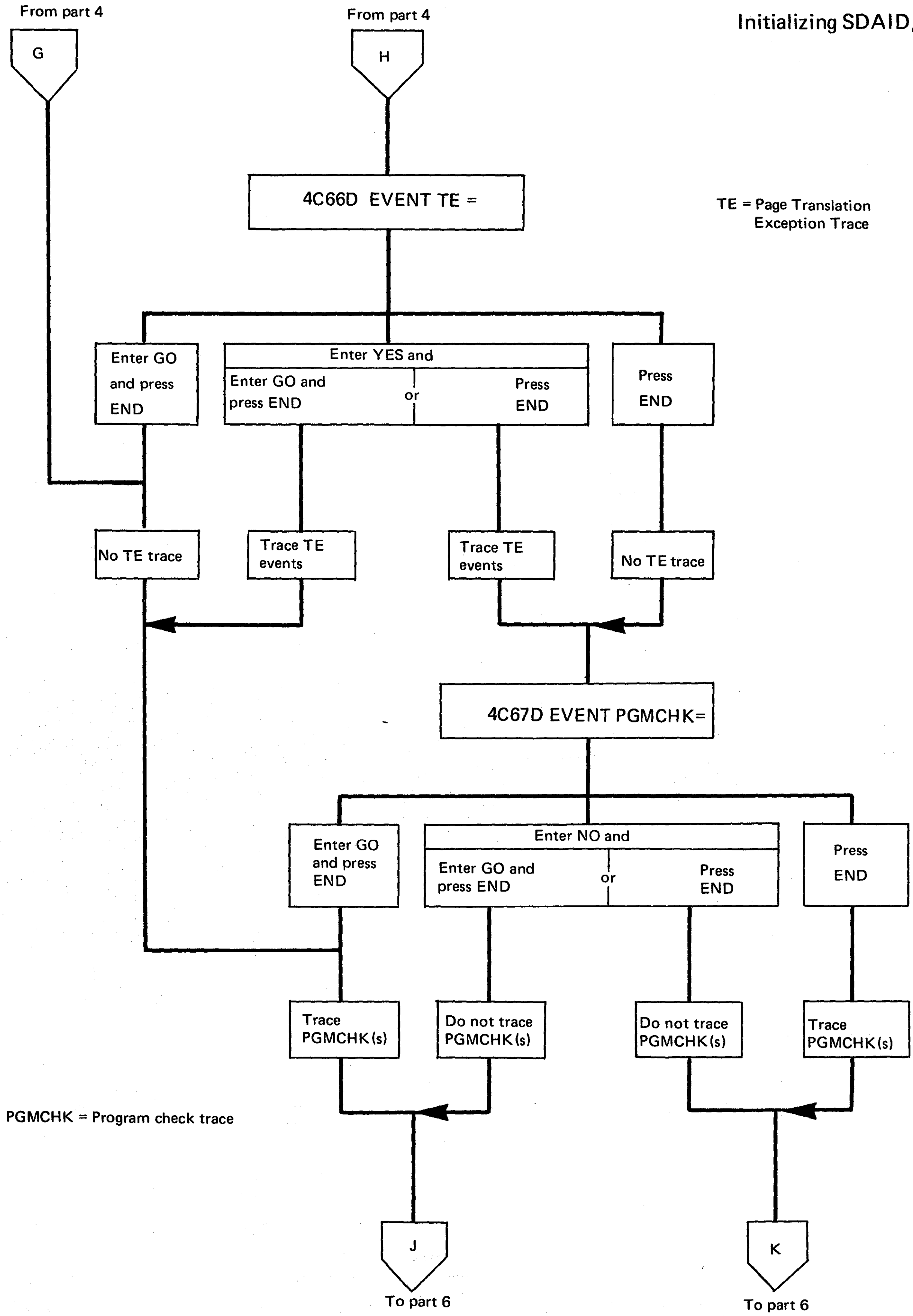



From part 2
C
From part 2
D
4C62D EVENT BR =
BR = Successful Branch Trace
Enter GO and press END
Enter YES and
Enter GO and press END
or
Press END
Press END
No BR trace
Trace BR events
Trace BR events
No BR trace
4C63D EVENT IF =
IF = Instruction Fetch Trace
Enter GO and press END
Enter YES and
Enter GO and press END
or
Press END
Press END
No IF trace
Trace IF events
Trace IF events
No IF trace
E
To part 4
F
To part 4

Trace Routines

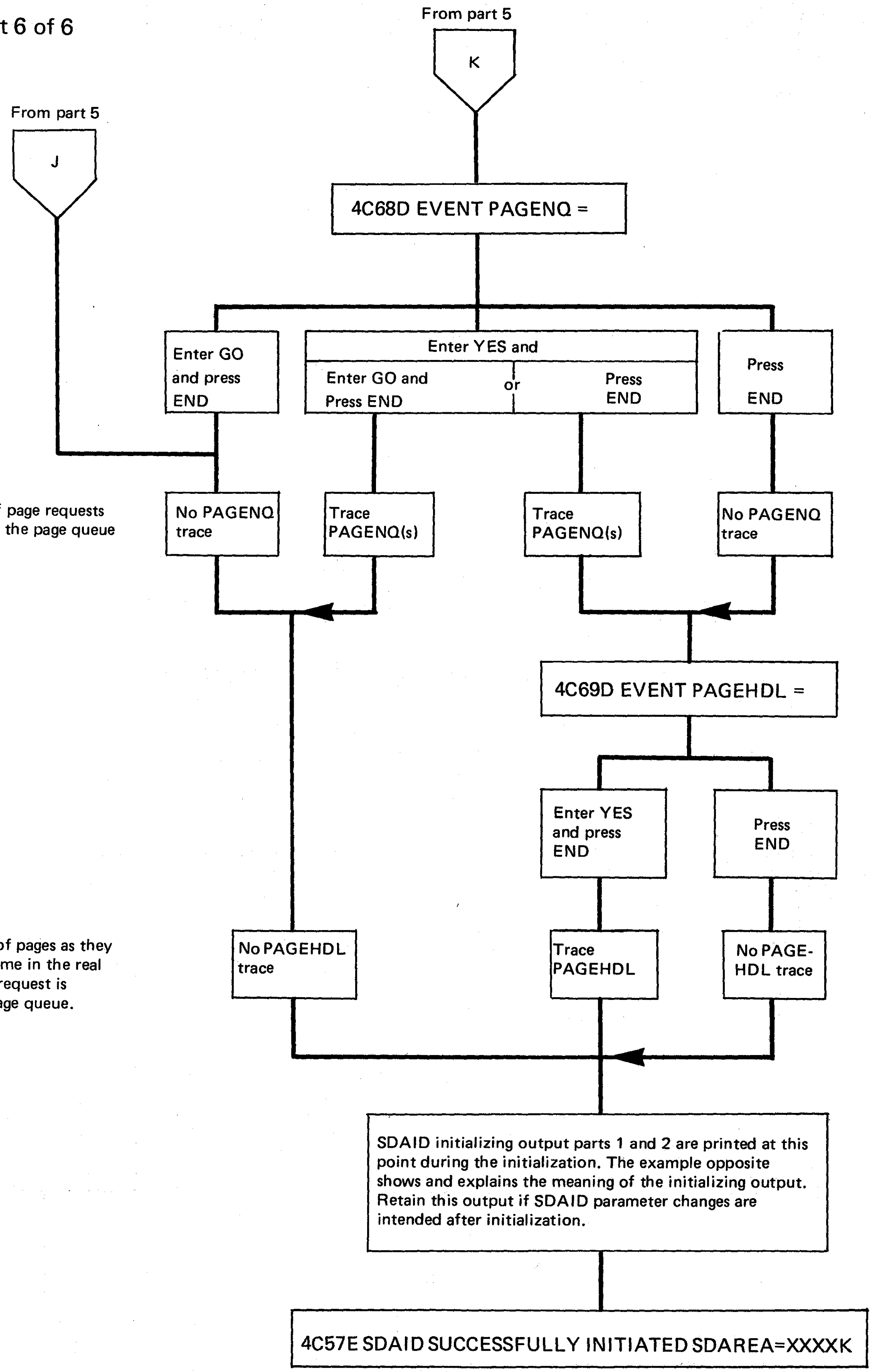
SDAIDS
Initializing SDAID, part 4 of 6




From part 4
G
From part 4
H
4C66D EVENT TE =
TE = Page Translation
Exception Trace
Enter GO
and press
END
Enter YES and
Enter GO and
press END
or
Press
END
Press
END
No TE trace
Trace TE
events
Trace TE
events
No TE trace
4C67D EVENT PGMCHK=
Enter GO
and press
END
Enter NO and
Enter GO and
press END
or
Press
END
Press
END
Trace
PGMCHK (s)
Do not trace
PGMCHK (s)
Do not trace
PGMCHK (s)
Trace
PGMCHK (s)
PGMCHK = Program check trace
J
To part 6
K
To part 6

Trace Routines

SDAIDS
Initializing SDAID, part 6 of 6



PAGENQ = Trace of page requests as they are placed in the page queue

PAGEHDL = Trace of pages as they are allocated to a frame in the real address area — page request is removed from the page queue.

The Label Information Cylinder

The label information cylinder contains standard and user label information for background and foreground partitions.

19 tracks on the 3330/3333, 12 on the 3340, or 20 on the 2314/2319 are allocated to the label information cylinder.

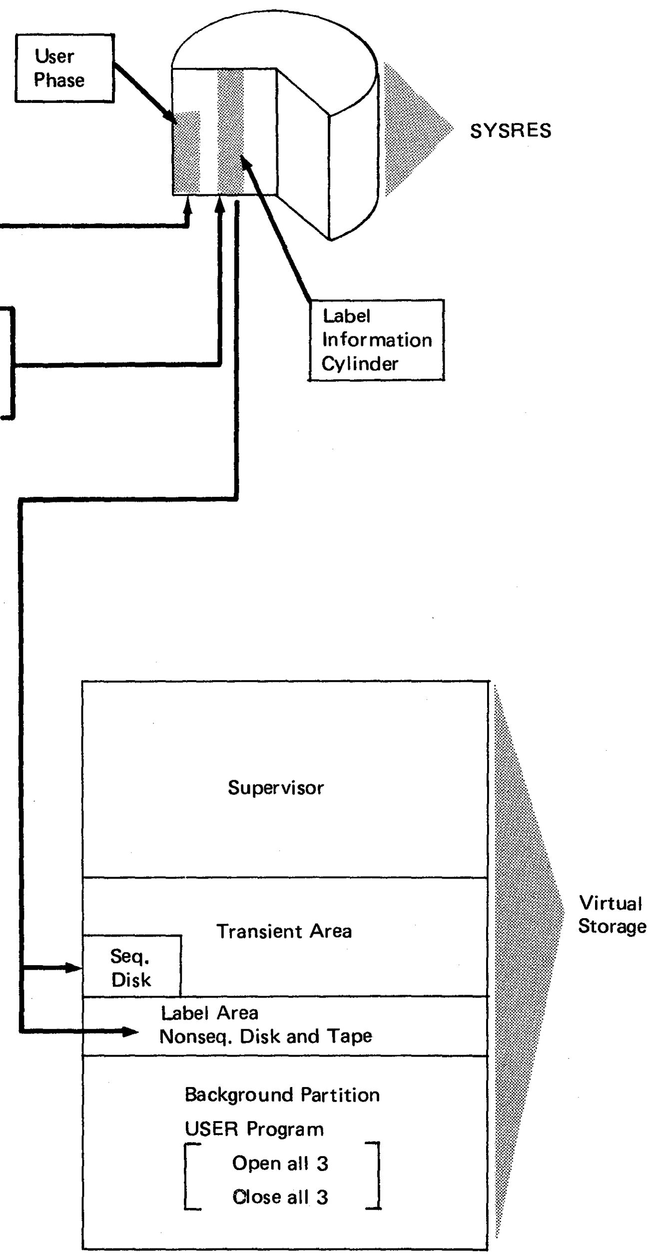
The purpose of the label cylinder is to enable label information to be placed in the label save areas during program execution.

Job control stores label information, which is contained in the input job stream, on the label cylinder.

During program execution, label information on the label cylinder is used by the OPEN and CLOSE routines.

SAMPLE JOB STREAM

```
     // JOB CATALOG USER PROGRAM
     // OPTION CATAL
      [Object module preceded ]
      [by PHASE and INCLUDE   ]
1    /*
     // LBLTYP NSD (01)
2    // EXEC LNKEDT
     /&
     // JOB EXECUTE USER PROGRAM
     // DLBL
     // EXTENT      File 1 (Seq. Disk)
3    // DLBL
     // EXTENT      File 2 (Nonseq. Disk)
     // TLBL        File 3 (Labeled Tape)
4    // ASSGN       File 1
     // ASSGN       File 2
     // ASSGN       File 3
5    // EXEC USER
     /&
```



Sequence of Events

1 LBLTYP reserves label area in virtual storage for longest label block

2 Link edit and catalog user phase to core image library

3 Write label information to label cylinder

4 Assign user files

5 Bring program into virtual storage and process OPEN/CLOSE
   a. Bring label storage data into virtual storage
   b. Bring file labels into virtual storage
   c. Compare input, create new output labels

An overview showing how DOS/VS uses the label information cylinder.

Library Display Programs and Utilities

LSERV

<table>
<tr><th rowspan="2">NO.</th><th rowspan="2" colspan="2">COMPONENT</th><th colspan="4">STARTING DISK ADDRESS</th><th rowspan="2">NUMBER OF TRACKS (Allocation)</th><th rowspan="2">R = REQUIRED
O = OPTIONAL</th></tr>
<tr><th>BB</th><th>CC</th><th>HH</th><th>R</th></tr>
<tr><td rowspan="4">1</td><td colspan="2">IPL Bootstrap Record 1 ($$A$IPL1)</td><td>00</td><td>00</td><td>00</td><td>1</td><td rowspan="4">1</td><td>R</td></tr>
<tr><td colspan="2">IPL Bootstrap Record 2 ($$A$IPL1)</td><td>00</td><td>00</td><td>00</td><td>2</td><td>R</td></tr>
<tr><td colspan="2">Volume Label</td><td>00</td><td>00</td><td>00</td><td>3</td><td>R</td></tr>
<tr><td colspan="2">User Volume Label</td><td>00</td><td>00</td><td>00</td><td>4</td><td>O</td></tr>
<tr><td rowspan="5">2</td><td rowspan="4">System Directory</td><td>Record 1</td><td>00</td><td>00</td><td>01</td><td>1</td><td rowspan="5">1</td><td>R</td></tr>
<tr><td>Record 2</td><td>00</td><td>00</td><td>01</td><td>2</td><td>R</td></tr>
<tr><td>Record 3</td><td>00</td><td>00</td><td>01</td><td>3</td><td>R</td></tr>
<tr><td>Record 4</td><td>00</td><td>00</td><td>01</td><td>4</td><td>R</td></tr>
<tr><td colspan="2">IPL Retrieval Program ($$A$IPL2)</td><td>00</td><td>00</td><td>01</td><td>5</td><td>R</td></tr>
<tr><td>3</td><td colspan="2">Core Image Library Directory</td><td>00</td><td>00</td><td>02</td><td>1</td><td>*</td><td>R</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2" colspan="2">Core Image Library</td><td rowspan="2">00</td><td colspan="2">End of CI Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">R</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">5</td><td rowspan="2" colspan="2">Relocatable Library Directory</td><td rowspan="2">00</td><td colspan="2">End of CI Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2" colspan="2">Relocatable Library</td><td rowspan="2">00</td><td colspan="2">End of RL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2" colspan="2">Source Statement Library Directory</td><td rowspan="2">00</td><td colspan="2">End of RL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">8</td><td rowspan="2" colspan="2">Source Statement Library</td><td rowspan="2">00</td><td colspan="2">End of SS Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">9</td><td rowspan="2" colspan="2">Procedure Library Directory</td><td rowspan="2">00</td><td colspan="2">End of SS Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
<tr><td rowspan="2">10</td><td rowspan="2" colspan="2">Procedure Library</td><td rowspan="2">00</td><td colspan="2">End of PL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y + 1</td></tr>
<tr><td rowspan="2">11</td><td rowspan="2" colspan="2">Label Information Cylinder</td><td rowspan="2">00</td><td colspan="2">End of P Library</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>Z + 1</td><td>00</td></tr>
</table>

*Allocation Dependent On User Requirements
X = Ending CC of the Preceeding Directory
Y = Ending HH of the Preceeding Directory
Z = Ending CC of the Preceeding Library

The location of the label information cylinder on the SYSRES extent

LIBRARY DISPLAY

Under certain circumstances knowing the contents of libraries can be helpful during program debugging. The library display programs DSERV, CSERV, PSERV, SSERV, and RSERV enable you to print an image of:

- Any library directory
- Any library
- Any program in any library
- Any phase in any library.

When using DSERV, a System Status Report is always printed before the specified directory. A private status report is also printed when private libraries are used with the system.

An example of a system status report is shown in two examples at the end of the section describing the Linkage Editor Map (E-4 of this manual).

When and how to use

Control statements required to execute the library display programs are shown in the next two tables.

The following list gives some examples of when to use the various library display programs:

1. The operator action given under the appropriate message in *DOS/VS Messages* indicate when to execute DSERV.

   For example, under the message:

   1C33A PROGRAM NOT FOUND

When error message instructions include a library display, enter cards that correspond to the library and type of display. Be sure to substitute the actual program module, book, sublibrary or phase name for the words phase 1, module 1, book 1, sublib 1, or prog 1.

Note: If you assign a private library and display that type of library, only the private library will be displayed. To obtain a display of the system library, the private library must be unassigned.

Additional information on the display program is found in *DOS/VS System Control Statements.*

Further recommendations as to when to use the library display programs are given after the two tables following.

Library Display Programs and Utilities

LIBRARY DISPLAY

Control cards required to execute the Library display programs

Unit	Element	Control Statements Required	

Core Image Library	Phase	// JOB jobname
		// EXEC CSERV
		DSPLY phase1 [,phase2,...]
		/*
		/&	

	Program	// JOB jobname
		// EXEC CSERV
		DSPLY prog1.ALL[,prog2.ALL,...]
		/*
		/&	

	Library	// JOB jobname
		// EXEC CSERV
		DSPLY ALL
		/*
		/&	

	Directory	Unsorted
		// JOB jobname
		// EXEC DSERV
		DSPLY CD
		/*
		/&
			Sorted
			// JOB jobname
			// EXEC DSERV
			DSPLYS CD
			/*
			/&

Relocatable Library	Module	// JOB jobname
		// EXEC RSERV
		DSPLY module1 [,module2,...]
		/*
		/&	

	Program	// JOB jobname
		// EXEC RSERV
		DSPLY prog1.ALL[,prog2.ALL,...]
		/*
		/&	

	Library	// JOB jobname
		// EXEC RSERV
		DSPLY ALL
		/*
		/&	

	Directory	Unsorted
		// JOB jobname
		// EXEC DSERV
		DSPLY RD
		/*
		/&
			Sorted
			// JOB jobname
			// EXEC DSERV
			DSPLYS RD
			/*
			/&

Table C-2 Library Display Control Cards (Part 1 or 2)

Note: To execute DSERV, SYSIN must be assigned to a card reader, a tape unit, or a disk drive. SYSLOG must be assigned to a 1052, 3210 or 3215 console printer, or for the Model 125 it must be assigned to the CRT.

MAP command

The MAP command produces on SYSLOG a map of virtual storage areas allocated to programs.

An example of the output produced on the console printer is shown below:

```
F2   map
F2   AREA      K-REAL  UPPER LIMIT  K-VIRT  UPPER LIMIT     NAME
F2
F2    SP         92K      94207               262144
F2    BG   V5A   30K     124927      292K     561151     KENSLOOP
F2    F4   I4    10K     135167       80K     643071
F2    F3   V3D   10K     145407       80K     724991
F2    F2   V2A   14K     159743      100K     827391     NO NAME
F2    F1   I1    82K     243711      120K     950271
F2    SVA                             96K    1048575
F2    VIS                             68K    1048575
F2    PP         18K     262143
```

	SP = Supervisor, **V** = Virtual, **PP** = Main Page Pool, **I** = Inactive, **SVA** = Shared Virtual Area, **R** = Real, **A** = Active, D=Deactivated 1, 2, 3, 4, 5 = Partition Priority (1 = highest priority), VIS = Amount of SVA reserved by GETVIS parameter of the VSTAB system generation macro
K-REAL	gives the number of bytes allocated to the real partition or the number of bytes of the main page pool. The size is given in multiples of 2K.
UPPER LIMIT	shows the highest storage addresses in decimal of the respective real partition, of the supervisor, and of the main page pool.
K-VIRT	specifies the number of bytes allocated to the respective virtual partition. The size is given in multiples of 2K. This field is blank for the supervisor and for the main page pool.
UPPER LIMIT	contains the highest storage address in decimal of the respective virtual partition. For the supervisor, this field specifies the start address of the virtual address area. This field is blank for the main page pool.
NAME	contains the name of the job which is currently executing in the corresponding partition. This field is blank for the supervisor, SVA, VIS, and for the main page pool. When the listing shows NO NAME for the background, or when the name field is blank for a foreground partition, no program is being executed in that area. However, the name field for any partition contains NO NAME when no job control statement or command was entered, but the program is active.

Note: If a program issues an SVC55, some page frames in the main page pool (PP) will also belong to that program. Therefore to calculate the total area in real storage occupied by that program, the MAP command should be issued before running it. The difference between the number of K in the PP before running the program, and the number of K during the execution of the program is the amount of K seized by the SVC55.
This also applies when PDAID output mode is an alternate area or the SDAID is initiated.
In this case, the area occupied by the PDAID or SDAID is printed during their initialization.

Note: The output does not indicate storage temporarily added to the page pool as a result of using the SIZE parameter of the EXEC statement.

JOB CONTROL COMMANDS AND STATEMENTS

OPTION

The OPTION statement specifies one or more of the Job Control options. The format of the OPTION statement is:

JCS Format

// OPTION option 1 (,option2,...)

The options that can appear in the operand field follow. Selected options can be in any order. Options are reset to the standards established at system generation time upon encountering a JOB or a /& statement.

DUMP — Causes a dump of the registers and main storage to be output on SYSLST, if assigned, in the case of an abnormal program end. (See A-2 in this section for a full description.)

NODUMP — Suppresses the DUMP option, if the latter was specified in the STDJC macro during system generation.

LOG — Causes the listing of columns 1-80 of all control statements on SYSLST. Control statements are not listed until a LOG option is encountered. Once a LOG option statement is read, logging continues from job-step to job-step until NOLOG option is encountered or until either the JOB or /& control statement is encountered.

NOLOG — Suppresses the listing of all valid control statements on SYSLST until a LOG option is encountered. If SYSLST is assigned, invalid statements and commands are listed.

LIST — Causes language translators to write the source module listing on SYSLST. In addition, it causes the Assembler to write the hexadecimal object module listing and causes the Assembler and the FORTRAN compiler to write a summary of all errors in the source program. All are written on SYSLST.

NOLIST — Suppresses the LIST option.

PAUSE

The PAUSE command causes a pause at the end of the current job step. The PAUSE Job Control statement causes a pause immediately after processing this statement. At the time, SYSLOG is unlocked for message input. END (on the Models 135 and 145) or ENTER (on the Model 125) causes processing to continue. The PAUSE statement or command always appear on SYSLOG. If a 3210 or 3215 or video console display unit is not available, the PAUSE statement or command is ignored.

RDESUM

The RDESUM option provides a summary of information about system operation during a specified 1 to 30 day period. This summary is created by searching the history/RDE tape, mounted on SYS009, for IPL, EOD, MCAR, CCH, and unit check records, after which these records are edited and printed on SYSLST. The information provided by the RDESUM option includes:

- The starting and ending dates of the report.
- The date, time, reason, and subsystem responsibility for each IPL.
- The average run time between IPL and EOD (or between two consecutive IPLs if the ROD command was not issued to create an EOD record) for the specified interval. If specified, the number of IPL records that occur in the cluster interval, (see note)
- The subsystem responsibility and number of times a subsystem caused a System Recovery Incident (a recoverable error that may cause system degradation) or a System Incident (an unrecoverable error that caused system failure).
- If the history/RDE tape contains no records within the specified dates, an error message is printed and the report is terminated.
- IPL records are not counted in the reports of sub-systems SI (System Recovery Incidents)
- If an IPL record with a reason code of UN, IE, IM, ME or DF is immediately preceeded on the tape by an SRI that occurred within 30 minutes of the IPL, the SRI may be reclassified as an SI. The SRI is reclassified if (1) the subsystem ID specified for the IPL is the same as the device type of the SRI, or (2) if the subsystem ID is unknown (00).
- Multiple SRIs on the same device are counted as a single SRI until there is a ten minute interval without an incident or an IPL record.
- If an SI occurs within ten minutes of the IPL record following an SI, the SI is counted as a multiple occurrence of the first SI regardless of the subsystem involved. Intervening SRIs are ignored.
- If 16 sequence errors occur on the history/RDE tape, RDESUM is terminated; if fewer than 16 sequence errors occur, the out-of-sequence records are ignored.

Note: Clustering is the process of searching for multiple IPL records that have occurred within a specified number of minutes. Clustering can be used to detect multiple false starts that may distort other information provided by RDESUM.

RDESUM is executed when the appropriate option card is encountered. The control information, including the start date for the report, the end date, the clustering interval if clustering is desired, and the company name, is entered once the EREPRDE phase is in main storage. The control information is entered in response to prompter messages.

RDESUM does not summarize across multiple volumes of a history/RDE file. If EOF is encountered on the input tape, RDESUM goes to EOJ and the report printed reflects the information available from the start date to the last record on the tape. There may be some inaccuracy in the average run time per IPL (because RDESUM does not know when the EOD or next IPL record will occur, it uses the time of the last error record to compute the IPL period), but no other information is lost.

RDESUM can be executed again for the next volume in the history/RDE file to obtain the remainder of the information for the desired reporting period. The previously specified period may be used on the subsequent volume because RDESUM starts with the first record on the tape if the specified start date is earlier than the date of the first record.

The following rules govern the method for summarizing RDE information:

- If the history/RDE tape does not contain information for a portion of the required time period, only those dates on the tape that fall within the time period are processed. The actual dates processed are reflected on the summary listing.
- If the starting date is defaulted, the first record on the tape is used to start the report. The report is stopped with the specified end date or, if that date is more than 30 days from the date of the first record processed, the thirtieth day processed.
- If the end date is defaulted, the report is stopped with the last date on the tape or, if that date is more than 30 days from the starting date, on the thirtieth day processed.

HIST or HIST with Operands

This option copies the data on the SYSREC file to the history/RDE tapes. All records on the tape(s) appear in chronological order. If an unrecoverable I/O error occurs while a record is being read from the SYSREC file, the record is ignored and processing continues with the next sequential record. If the data fills the complete tape, the message

3E15A TAPE FULL, MOUNT NEW TAPE

is printed on SYSLOG. The operator must mount a new tape and press END to continue processing, or he may respond with CANCEL and press END to cancel the HIST option.

The tape must be mounted on SYS009, which must be assigned to a tape drive before EREP is executed. The tape contains standard labels that are checked before the history/RDE tape is written. If the wrong tape is mounted, the message

3E31A WRONG TAPE, MOUNT CORRECT TAPE

is printed on SYSLOG, Mount the correct tape and press END to continue processing, or respond CANCEL END to cancel the HIST option. When the HIST option is specified, the CLEAR option is forced. The SYSREC file is cleared after the history/RDE tape has been created or updated, thus preventing redundant data from being transferred to the history/RDE tape the next time the HIST option is executed.

HIST,NEW,[,2] : This option causes EREP to create a history file on the tape unit assigned to SYS009. If 2 is also specified, a second history file is created on the same tape unit for RDE data. The data contained on both tapes is identical. The tape(s) contain the contents of the SYSREC file. The SYSREC file is cleared after all options have been executed.

HIST,UPNEW: This option causes the tape file mounted on SYS009 (either history or RDE) to be updated, after which a new tape file is created. If UPNEW is specified, TLBL information for creation and updating must be included in the job stream. The SYSREC file is cleared when all options have been executed.

TES or TES with Operands

The TES options provide for the editing and printing of the tape error records on SYSREC and the summarizing of tape data found on either SYSREC or the history file.

To enable this option to be used a work or scratch tape must be mounted on a tape unit assigned to SYS008. This option can also select tape error data from the SYSREC file and create a TES history tape with the same format as the previously supported ESTV tape file. All records on the tape appear in chronological order. If an unrecoverable I/O error occurs while reading a record from the SYSREC file, the record is ignored and processing continues with the next sequential record. If the data fills the complete tape, the message

3E15A TAPE FULL, MOUNT NEW TAPE

is printed on SYSLOG. The operator must mount a new tape and press END, or he may respond CANCEL END; the latter response causes tape updating to be discontinued, but TES records are still printed.

The tape must be mounted on SYS009, which must be assigned to a tape drive before EREP is executed. The tape contains standard labels that are checked before the history/RDE tape is written. If the wrong tape is mounted, the message

3E31A WRONG TAPE, MOUNT CORRECT TAPE

is printed on SYSLOG. Mount the correct tape and press END to continue processing, or respond CANCEL END to cancel the TES option. The history/RDE tape and TES history tape should be created or updated during the same EREP run. If the HIST option is specified without the TES option, the SYSREC File is cleared after HIST has been executed, and the TES data is lost. If you wish to maintain both these history tapes and the TES and HIST options are not specified together in one EREP run, the data on the TES history file may be redundant or lost.

TES,NEW: This causes EREP to create a TES history file on the tape unit assigned to SYS007. The tape file contains tape error data from the SYSREC file. The tape error data on the tape has the same record format as the previously supported ESTV tape file. Use ESTVUT utility program to print this tape file.
TES: EREP updates the TES history tape on SYS007.
TES,NOTAPE,PRINT: Causes the tape data on SYSREC to be edited and printed into SYSLST. Data is printed in the detail tape unit format.
TES,PRINT,NEW: A new TES history tape is created on SYS007, after which the tape error data on SYSREC is edited and printed on SYSLST. The data is printed in the detail tape unit format.
TES,PRINT: The TES history tape, which is mounted on SYS007, is updated. The tape error data on SYSREC is then edited and printed on SYSLST in the detail tape unit format.
TES,NOTAPE,SUM: The tape error data on SYSREC is summarized by tape drive.
TES,NOTAPE,PRINT,SUM: The tape error data on SYSREC is edited and printed on SYSLST in the detail tape unit format. Then the tape error data on SYSREC is summarized by channel and unit and printed on SYSLST.
TES,SUM,VOL: The TES history tape on SYS007 is updated. Afterwards the tape error data found on SYSREC is summarized by volume serial number.
TES,PRINT,VOL: The TES history tape mounted on SYS007 is updated. The tape error data on SYSREC is edited and printed on SYSLST in the detail volume serial number format. SYS008 is used as a work tape and the detail records are printed in sequence by volume serial number.

Four examples of processing tape error statistics using EREP are given in Appendix O.

EREP

EREP History Tapes

There are three types of EREP history tapes: the History tape, the RDE tape, and the TES history tape. The History and RDE tapes are created and updated from the SYSREC file and contain all the record types found on the SYSREC file. The TES history tape is also created from the SYSREC file, but contains only tape error records. If your installation has the History/RDE tapes and a TES history tape, you should create (or update) all the history tapes in the same run. If this procedure is not followed, the TES history tape may have redundant or missing data.

Retain the History and TES history tapes for those persons who work on problem determination. The History tape can be used as input for certain online test programs of OLTEP. (See the OLTEP manual.) The TES history tape can be printed with the ESTVUT utility program. Retain the RDE tape; it will be used by IBM.

History/RDE Tape

The History/RDE tape is created and updated using the EREP history option. This tape contains RDE data only if ERRLOG=RDE is specified at system generation. A magnetic tape unit assigned to SYS007 must be used for this function. EREPNEW must be the filename that is used when a tape is created, and EREPUP when a tape is updated (both TLBL cards must be included for UPNEW). When the tape becomes full or when a second tape must be mounted, the operator is notified via SYSLOG.

Note: If EREP is link-edited as a self-relocating program, an LBLTYP card is needed when EREP builds a history/RDE tape.

TES History Tape

The TES history tape is created and updated using the EREP TES options. A magnetic tape unit assigned to SYS007 must be used for this function. The filename of the tape file must be TAPEIN when the file is created and the file is updated.

Creating the History Tapes

You can create a history tape only if DOS/VS has recorded errors on SYSREC. The EREP program allows you to create or update the three types of history tapes. three types of history tapes.

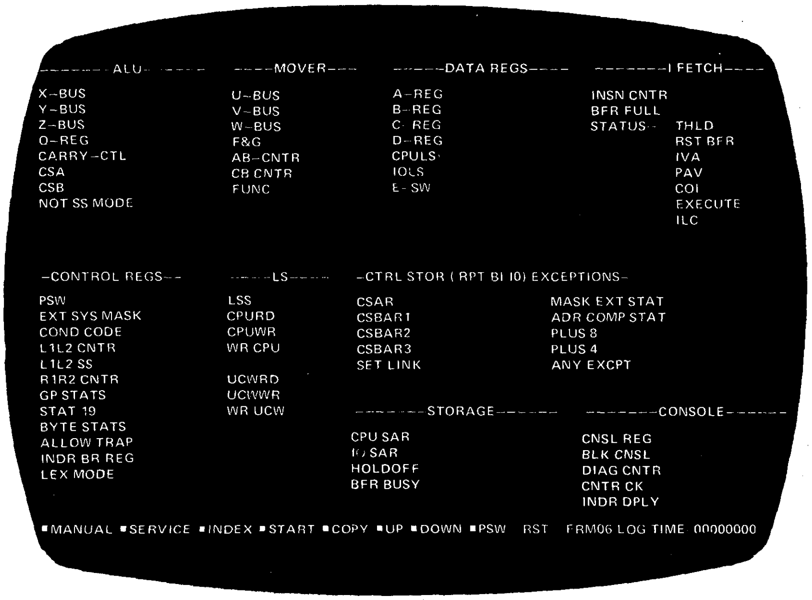
You can create the History/RDE tape by specifying OPTION HIST, NEW, and update it by specifying OPTION HIST.

If a System/370 RDE tape is to be processed, the message 3E16A is printed on SYSLOG after the History tape is written. This message instructs you to replace the History tape reel with the RDE tape reel and then respond to the message. A response of END will cause the RDE tape to be processed and response of CANCEL END will cancel only the HIST option. Any other response will cause the system to reissue message 3E16A.

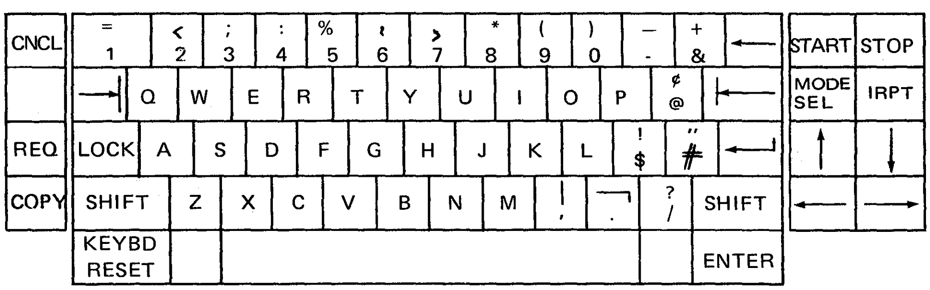
In addition, you can create a TES history tape, which contains only tape error records. If you want to maintain a TES history tape, create (or update) it in the same EREP run in which you create (or update) the History/RDE tape. You can create the TES history tape by specifying OPTION TES,TAPE,NEW, and update it by specifying OPTION TES, TAPE.

DISPLAY FRAMES (MODEL 158 ONLY)

How to use



5. Scan the frame for any characters that flash on and off beside an entry displayed. For example, Z-BUS 614250 indicates that the hardware failure is caused by a failure in the Z-BUS.
   If a hardware failure is indicated, press COPY to obtain a hard copy of that frame and make a note on the hard copy about the error. (Characters that indicate an error are not copied by the system.)
6. Press the key marked ↑ on the keyboard, as illustrated below.
   The next display frame will be displayed.



7. Repeat steps 5 and 6 until all display frames have been scanned and hard copies made of those containing information about the failure.
8. Press CANL to obtain the program frame.

When to use
1. After recognizing a hardware failure as shown in the example above.
2. On advice from your IBM CE.

F-6

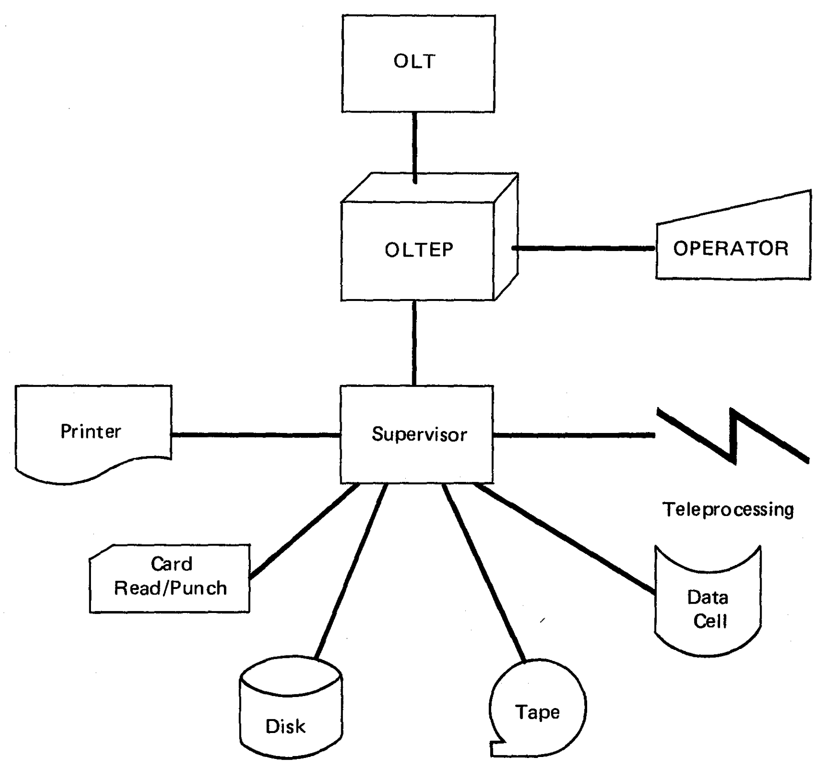
OLTEP

IBM provides a set of device test programs that run under control of DOS/VS. These test programs and the online test executive program form the online test system. The Online Test Executive program (OLTEP) is an interface between the system and the online test programs (OLTs) and communicates with the operator during the running of tests.

Some uses:
- Diagnosing I/O errors
- Verifying I/O device repairs and engineering changes
- Checking I/O devices.

Some features:
- Multiple device testing
- Data security
- Data protection
- No re-IPL time required
- Prompting
- ASCII data conversion
- Accessing of error recording information
- CDS Equate function.

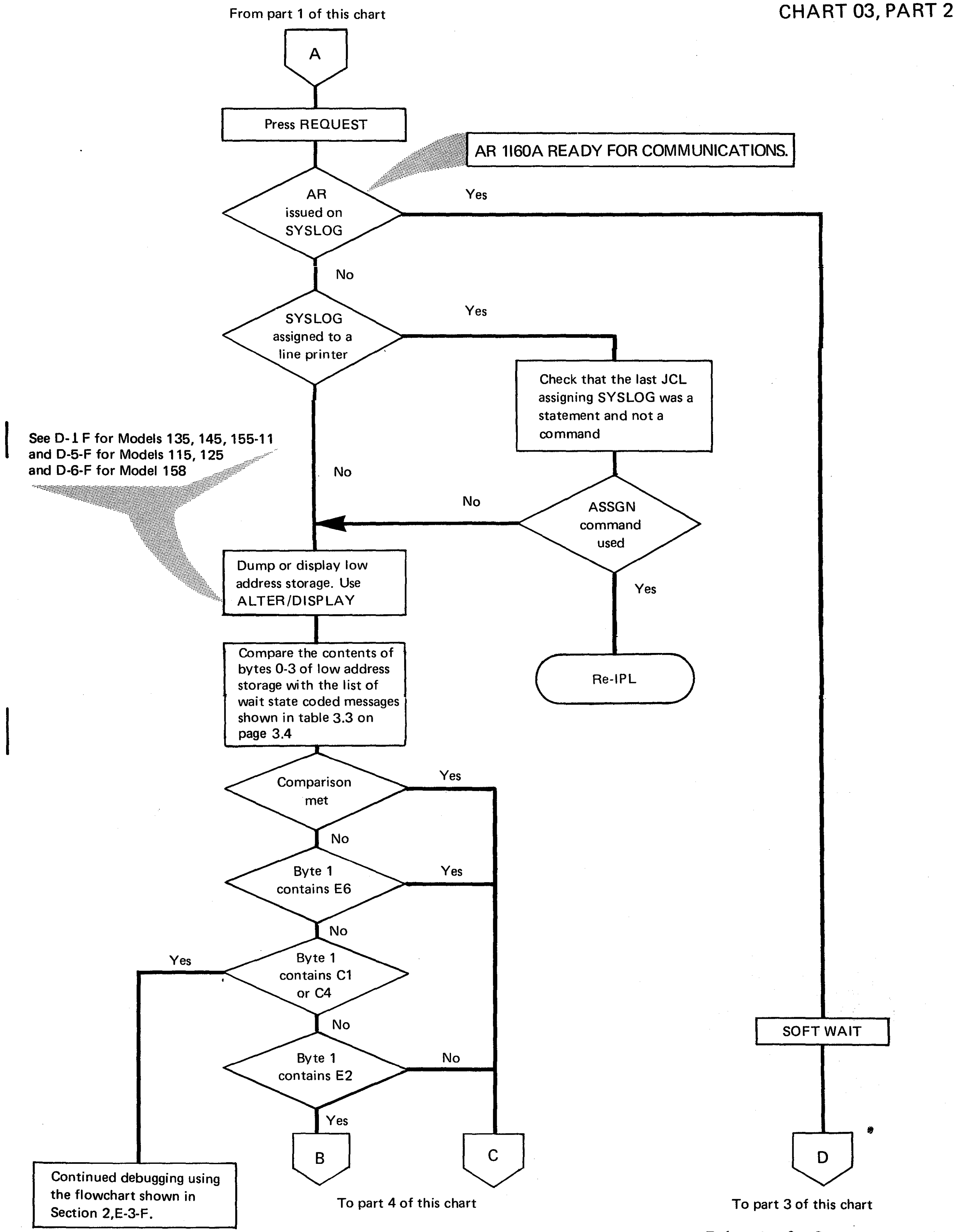
Note: Not applicable to the Models 115 and 125 using a supervisor that does not support the RMSR function.



OLTEP-System Relationship

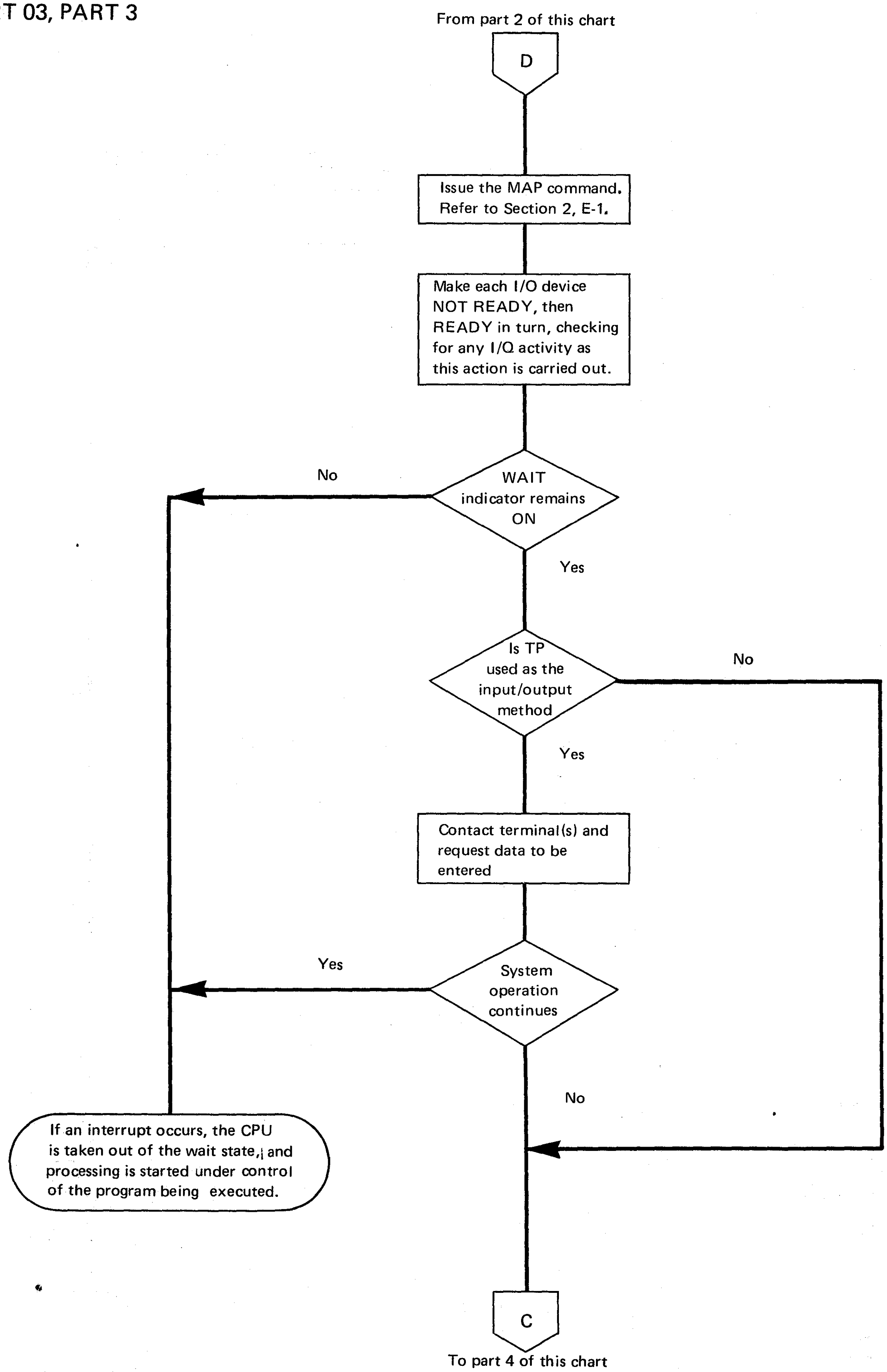
Wait State

CHART 03, PART 2



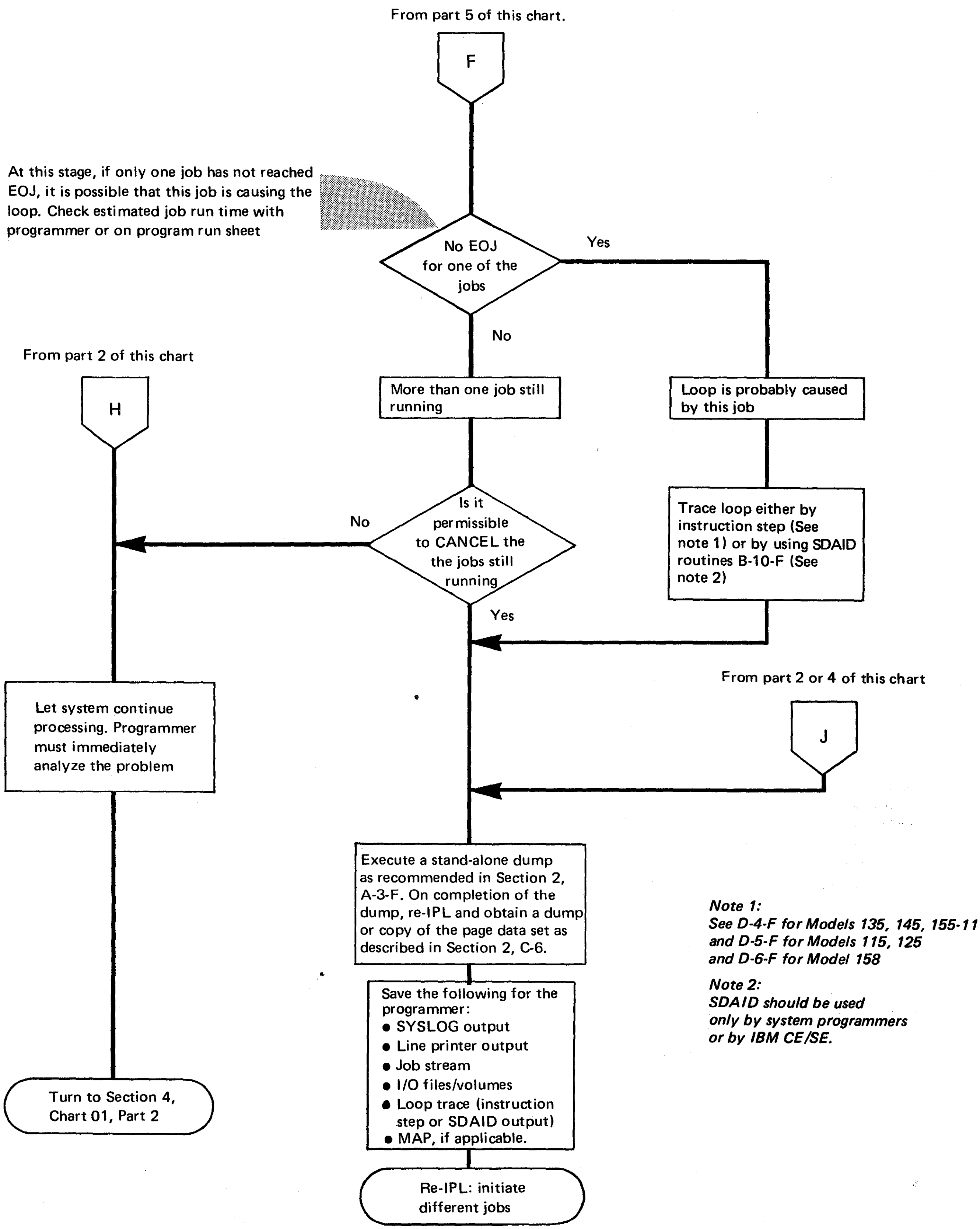
Wait State

CHART 03, PART 3



Unintended Loop

CHART 04, PART 6



Obviously Incorrect Output

CHART 05

Start

The identity of the partition using the device:
This can be deduced from information given on the job run sheet, or the device ASSGN statements and commands issued on SYSLOG and SYSLST.
An alternative method is to make the device producing incorrect output NOT READY. This will cause a device error recovery message to be issued on SYSLOG.

If possible, contact the programmer to establish the expected output, and identify which job or partition is using the device that is producing the incorrect output

Determine the partition using the device from the message printed on SYSLOG as a result of this action.

BG	0P08A INTERV REQ SYS008 =	383

Partition identity — *I/O Device Address*

Make the device producing incorrect output NOT READY

Press REQUEST and wait for message 0P60D, then enter IGNORE

AR 1I60A READY FOR COMMUNICATIONS.

If SYSLST for the partition to be dumped is not assigned to a line printer, use the line printer assigned as SYSLST on another partition.
For example, DUMP F2S(BG).
Note: This may produce dump output from the partition being dumped, interspersed with output from the program using the line printer.

Respond to the AR message by issuing the DUMP command to dump the partition and the supervisor, for example, DUMP F2S
SYSLST must be assigned to a line printer.
Refer to Section 2, A-1

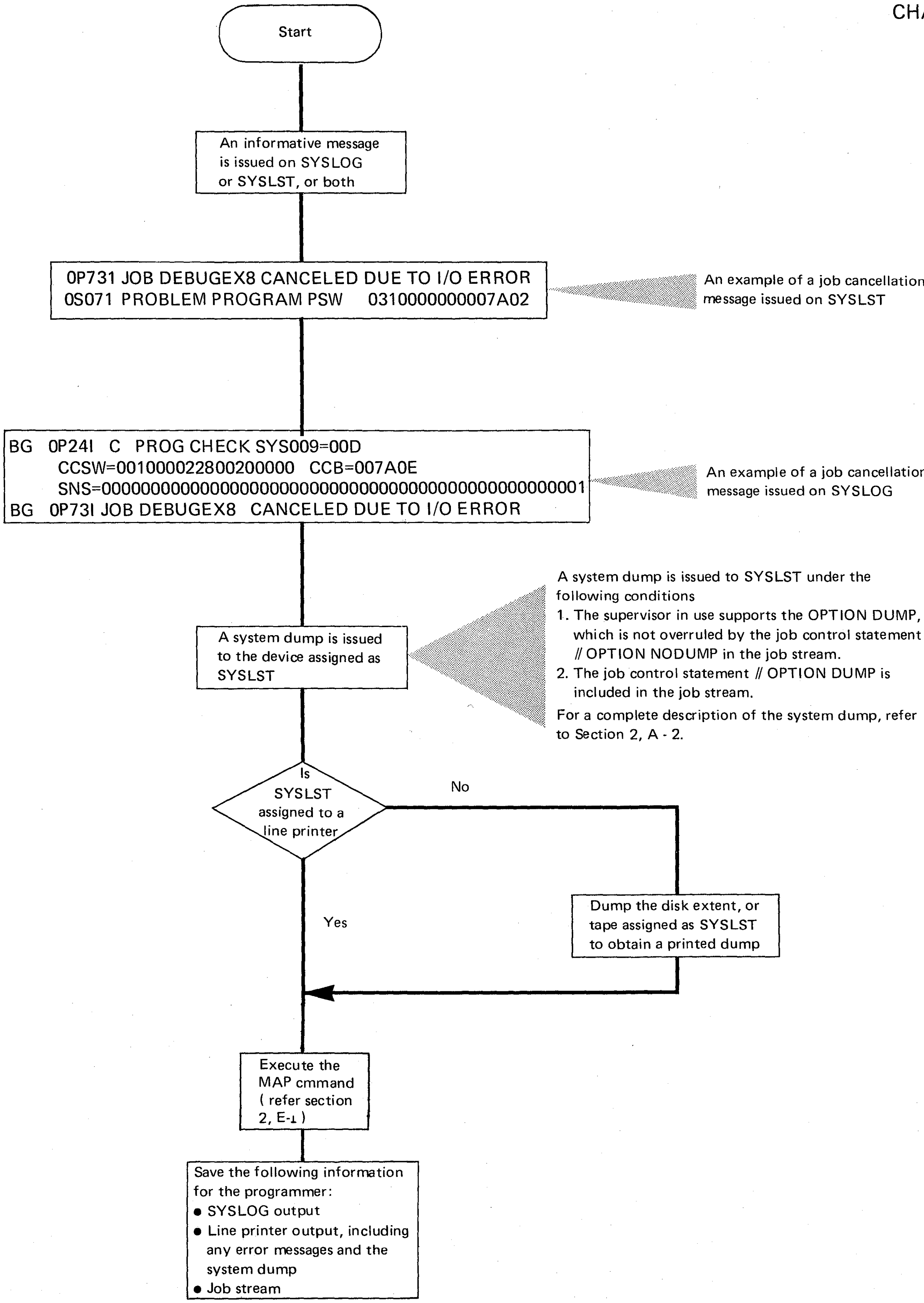
A System dump may be issued on the device assigned to SYSLST for the partition being canceled.
(The System dump is described in Section 2, A-2.)

Cancel the job in the partition producing the incorrect output

Save the following information for the programmer(s).
- SYSLOG output
- Line printer output, including all dump output
- Job stream
- I/O files/volumes

Program Canceled

CHART 06



For your notes

CHART 02, PART 1 OF 1

Check	*Method of Analysis*	*Recommendations to Aid Further Problem Analysis*
Check 1 Supervisor listing	If the IPL procedure can not be completed and the supervisor in use has never been used on the system, check the supervisor assembly listing for flagged statements. Also note in the listing the options supported by the supervisor. Check that the hardware is present in the system for options supported by the supervisor that require hardware support. Also check that the supervisor is generated to support hardware options that are present on the system.	If the cause of the error can not be isolated, inform IBM and have the following information available: ● The supervisor listing ● The stand-alone dump output, including the dump of the low address storage ● The SYSRES pack used when the error occurred ● Any notes about actions taken during error analysis both on and offline.

Hard Wait Message Codes

SUPPORT FOR CHART 03

BYTE 0	BYTE 1	BYTE 2	BYTE 3	EXPLANATION
MCH/CCH/IPL Hard Wait Codes placed in low address storage				
X'C1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable machine check.
X'C2'	X'E2'(2)	Not used	Not used	Irrecoverable channel failure during RMS fetch.
X'C3'	X'E2'(2)	A, I, S(1)	Not used	Channel failure on SYSLOG when RMS message scheduled.
X'C4'	X'E2'(2)	A, I, S(1)	Not used	No ECSW stored
X'C5'	X'E2'(2)	A, I, S(1)	Not used	Channel failure: ERPBs exhausted.
X'C6'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; two channels damaged or a damaged channel situation occurred while RMS was executing an I/O operation.
X'C7'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system reset was presented by a channel.
X'C8'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; system codes in ECSW are invalid.
X'C9'	X'E2'(2)	A, I, S(1)	Not used	Channel failure; channel address invalid.
X'D1'	X'E2'(2)	A, I, S(1)	Not used	Irrecoverable channel failure on SYSVIS.
X'07'	X'E6'	Channel	Unit or X'00'	IPL I/O error or equipment malfunction; condition code 2 during STIDC instruction. Channel and unit indicate whether device in error is SYSRES or communication device. When byte 3 = X'00', byte 2 indicates the channel for which STIDC instruction was issued. Re-IPL system.
SDAID Hard Wait Code				
X'61'	X'E6'(3)	Channel	Unit	Another device is running in burst mode on same channel as SDAID output device. Re-IPL system.
SDAID Soft Wait Code				
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'(3)	X'00'	X'00'	SDAID Stop on Event. Press EXTERNAL INTERRUPT key to continue operations.
The following Hard Wait Codes are placed in general register 11 X'B' as well as in low address storage.				
X'00'	X'00'	X'0F'	X'FF'	Program Check in Supervisor.
X'00'	X'00'	X'0F'	X'FE'	I/O error during fetch from System CIL.
X'00'	X'00'	X'0F'	X'FD'	Channel Failure if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
X'00'	X'00'	X'0F'	X'FC'	Machine Check if MCH=NO and RMS=NO is specified during system generation. (Models 115 and 125 only).
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'FA'	Translation Specification Exception
X'00'	X'00'	X'0F'	X'F9'	Error on Paging I/O.
X'00'	X'00'	X'0F'	X'F8'	CRT phase not found.
X'00'	X'00'	X'0F'	X'F7'	No copy blocks available for BTAM appendage I/O request
X'00'	X'00'	X'0F'	X'F6'	$MAINDR canceled during system CIL update. If this occurs, the system CIL is only partially updated and must be restored before use. This hard wait condition can also occur if the FETCH QUEUE BIT (FCHQ) is set in the linkage control byte in the partition communication region owned by the terminating partition.
Device Error Recovery Soft Wait Codes placed in low address storage.				
X'08' to	X'C1' or	Channel	Unit	Error recovery messages. Refer to OP messages in *DOS/VS Messages.*

Notes: 1. A (X'C1') = SYSREC recording unsuccessful.
I (X'C9') = SYSREC recording incomplete.
S (X'E2') = SYSREC recording successful.
2. S (X'E2') = Run SEREP.
3. SDAID wait states are identified by X'EEEE' in the address part of the wait PSW.

Refer to Section 2-E-3 for a list of IPL error message codes and a more detailed description of wait states.

Incorrect Output detected during Program execution

CHART 08, PART 1 OF 5

Check | *Method of Analysis* | *Recommendations to Aid Further Problem Analysis*

Check 1

Type of program producing the incorrect output

- Userwritten?
  - Proceed to check 2.
- IBM-supplied program?
  Assembly language
  High level language compiler (RPG II, FORTRAN, etc.)
  IBM library program (DSERV, SORT/MERGE) etc.
  - Inform IBM and have the following information available:
    - All output from the program
    - SYSLOG output
    - Job stream
    - Input files
    - Supervisor generation notes and level of release
    - Listing of the support for the failing program
    - Notes about system type and model, and any notes about system status and input/output units used for the program.

Check 2

Type of incorrect output

Use any output from the program and messages printed during program execution to determine the type of/incorrect output according to the following categories:

- Duplicate (identical lines printed).
- Lack of (missing data printed).
- Too much or unexpected (more data printed than expected).

Use the program listing and analyze the coding, checking for:

- Unnecessary program loops in input/output routines
- Routines not being entered or completed
- Overlaid input/output areas.

If necessary, re-run the program using SDAID IF, SA, and GA trace functions. Specify event limits that will adequately cover routines that prepare output areas. (Refer to section 2-B-10.)
Re-analyze the output from the re-run.
If the error can not be isolated proceed to check 3.

Check 3

System condition during program execution

Were there any abnormal interruptions in computer operations during execution of the program?

- Yes: Analyze the possible effects of any interruption, and re-run the program ensuring no interruptions.
- No: Were any abnormalities observed in system actions or response to messages by the operator during program execution?
  - Yes / No: Re-run the program and note any unexpected system actions. Ensure the correct response is given to messages issued. (Refer to the *DOS/VS Messages* manual).
    Use any notes made to analyze program logic.
    If necessary, re-run the program using with the SDAID IF trace to aid analysis of program logic.
    Check the input used for the program.

    Proceed to check 4.

Incorrect Output detected during Program execution

CHART 08, PART 2 OF 5

Check	*Method of Analysis*		*Recommendations to Aid Further Problem Analysis*
Check 4 Environment	Was the program operating under POWER or POWER RJE?	Yes	Turn to part 5 of this chart, check 13.
		No	Proceed to check 5.
Check 5 Job stream	Check that the correct job stream was used for the program. This applies also to job streams cataloged on the procedure library. (Use PSERV to display the procedure library; refer to section 2-C-3.)		If error isolated, make corrections as necessary and re-run the program. If error can not be isolated, proceed to check 6.
Check 6 Input used	A. Isolate suspected areas on the input files by examining the output from the program. For example, a missing record in the output indicates the approximate area on the input file that may be in error.		A. Obtain a dump of suspected areas on the input files used. For VSAM files use the Access Method Services utility program PRINT command. (Refer to section 2-C). B. Check input/output file macro definitions in the program listing against characteristics of data on the input files. DTF (LIOCS), CCB (PIOCS) ACB, RPL, and GENCB (VSAM).
	B. Check the labels and volume ID and extent information. Use LSERV and LVTOC, or for VSAM files use LISTCAT Access Method Services utility commands.		Compare the output of the LSERV, LVTOC, and LISTCAT programs with DLBL/EXTENT, VOL ID, data in the job stream. If no errors can be found, proceed to check 7.

Check	*Method of Analysis*	*Recommendations to aid further Analysis.*
Check 3 Program check occurred before	Was a note of system activity and environment made when the identical program check occurred in any previous run of the program?	Yes: Compare note made with the conditions and environment under which this program check occurred again. Proceed to check 4.
		No: Make note of the system environment and I/O units involved. Also note the I/O files in use. Proceed to check 4.
Check 4 Gather more system information	Re-run the program with the PDAID F/L trace to trace phases used. Also use the DIAG parameter to trace SVCs issued by the POWER or POWER RJE SCP. (Refer to Appendix J).	Re-analyze the problem using the output from the re-run and information obtained from your previous analysis of the problem. If the error can not be isolated, inform IBM and have the following information available: o Program output o SYSLOG output o Job stream o Supervisor listing o POWER or POWER RJE assembly listing listing and linkage editor map o Any notes made about system environment o I/O units used and system condition o PDAID output o POWER DIAG SVC trace output.

Section 4, Part 2

CONTENTS

Note: Contents and addresses shown in the illustrations are subject to change and are shown only as an aid to offline debugging of DOS/VS.
IBM will not be responsible for any system malfunction resulting from a change made by the user to any contents or addresses of the tables and blocks described.

PARTITION COMMUNICATION REGION (. . . . Cont'd)

Displacement (Dec)	(Hex)	Meaning
60	3C	Binary disk address of the volume label area (label cylinder).
62 76	3E 4C	Addresses of FOCL, PUB, FAVP, JIB, TEB, FICL, NICL and LUB (See Figure 4.2)
78	4E	Set to the value nn specified in the LINES= nn parameter of the STDJC macro.
79	4F	The format of the system date contained within this field is determined by the IPL program from information supplied in the date convention bit (displacement 53). Bytes 85-87 contain the day count.
88	58	Bytes reserved for use by LIOCS. Transient dump programs insert a key to indicate to the LIOCS end-of-volume routine, $$BCMT07, that it was called by a B-transient.
90	5A	Address of the first part of the program information block (PIB) table.
92	5C	ID number of the last checkpoint. Byte 92 is also the temporary indicator of file protected DASD. Bits 0-6 correspond to channels 0-6. A bit ON means DASDFP for that channel. Bit 7 indicates 2321 DASDFP support. Byte 93 is used at IPL time by PIOCS – Bit Ø: 1 = 3330 file protection Bit 1: 1 = 3340 file protection
94	5E	Job zone for Time of Day. If ZONE=EAST, value is positive; if ZONE=WEST, value is negative.
96	60	Address of the disk information block (DIB) table for the partition.
98	62	Reserved.
100	64	Address for PC, IT, and OC option tables.
104	68	
106	6A	Key of the program that has internal timer support. The key is the same as the PIK for the timer supported partition. If multiple partitions all have timer support it is initially X'0010' but may be changed to the PIK of another partition by the TIMER command. It is copied into all partition communications regions. If no partition has interval timer support, these bytes contain X'0000'.
108	6C	Reserved.
110	6E	Logical Transient Key (LTK) contains the same value as the PIK (PID) (Displacement 46) when the logical transient is requested. When the transient area is not in use, LTK is equal to zero. The SVC2 routine sets the LTK. (See Appendix B for a description of the LTK). The SVC11 routine resets the LTK, (only significant in BG communication reg.)
112	70	Address of SYSPARM field.
116	74	Address of Job Accounting partition table.
120	78	Address of the Time of Day Clock common area.
124	7C	Address of second part of program information block (PIB) table.
126	7E	Address of PDTABB, table of DTF addresses for MICR support.
128	80	Address of QTAM vector table (IJLQTTAD).
132	84	Address of background communications region.

Figure 4.3. Key to Communication Region displacement, part 4 of 6

PARTITION COMMUNICATION REGION (. . . Cont'd)

Displacement (Dec)	(Hex)	Meaning
134	86	Option Indicator byte Bit 0: Reserved 1: 1= EU interface active 0= EU interface inactive 2: 1= Teleprocessing request 0= No teleprocessing request 3: 1= Supervisor support for tape 0= Supervisor does not support tape 4: Reserved 5: 1= RETAIN support generated 0= RETAIN support not generated 6: 1= Linkage to Channel En d Appendage Routine allowed 0= Linkage to Channel End Appendage Routine not allowed 7: 1= GETVIS function has been initiated 0= GETVIS function has not been initiated.
135	87	System Configuration byte 2 and RMSR Open Flag byte Bit 0: 1= PCIL supported 0= PCIL not supported 1: TOD supported 2: 1= PFIX macro supported 0= PFIX macro not supported 3: 1= Fetch $$OPEN by $JOBCTLJ 4: 1= Fetch $$OPEN by $JOBCTLD 5: 1= Fetch $$OPEN by $JOBCTLJ for WTM 6: 1= QTAM supported (bits 3–6: RMSR OPEN flags) 0= QTAM not supported 7: 1= RPS supported 0= RPS not supported
136	88	Pointer to Option table in SYSCOM. Reserved for compatibility reasons.
140	8C	Standard Job Control Option byte Bit 0: 1= EDECK Standard Option 1: 1= ALIGN Standard Option 2-6: Not used 7: 1 = ACANCEL standard
141	8D	Temporary Job Control Option byte Bit 0: 1= EDECK Temporary Option 1: 1= ALIGN Temporary Option. 2-5: Not used 6: 1 = SUBLIB = DF Temporary Option 7: 1 = ACANCEL Temporary Option
142	8E	Disk Configuration byte Bit 0-4: Not used 5: 1 = 3340 supported 0 = 3340 not supported 6: 1= 3330 supported 0= 3330 not supported 7: Always 1; indicates 2311 and 2314/2319 supported.
143	8F	Catalogued Procedure Name.

Figure 4.3. **Key to Communication Region displacement, part 5 of 6**

Hex	A2	A3	A4	A5	A6	A7	B0	B4	B8
Dec	162	163	164	165	166	167	176	180	184
	Pointer to PMGR Task Block	Pointer to SUPVR Task Block	Pointer to CRT Task Block	Pointer to ERP Task Block	Pointer to PAGEIN Task block	Reserved 9 X'00'	Not Used	Address of MVCFLD	Not Used

Hex	BC	BE	C0	CB	CC	CE	D0	D4	D8
Dec	188	190	192	203	204	206	208	212	216
	Not Used	Not Used	Repositioning information for 2560/ 5425 ERP	Number of Error Queue Entries	Length of PUB Table in Bytes	Number of Active Partitions	Address of Segment Table	Address of Page Frame Table	Address of Page Frame Table Extension

Hex	DC	E0	E4	E8	EC	F0	F4	F5	F8	FC	100
Dec	220	224	228	232	236	240	244	245	248	252	256
	Address of Boundary Box	Address of DPD Table	Reserved	Address of VIRTAD Routine	Address of End of Real Storage (Fullword)	Address of Fetch table	SVA Flag (See expansion)	Address of SVA	Address of System GETVIS area	Address of LDL in SVA	Pointer to RPS Sector Calculation Routine

Figure 4.4 Format and contents of SYSCOM, part 2 of 2

Dec	Hex	Description
64	40	Reserved for RMS support on the Models 115 and 125
		X'80' RMSR for channel attached devices, tapes and TP devices
		X'40' Full RMS support (MCAR/CCH and RMSR)
		X'20' MCAR/CCH support
65	41	X'80' Initial selection of ErP
		X'40' Reserved
		X'20' Timer interrupt pending
		X'10' MICR Stacker-select active
		X'08' Invalid address during fetch
		X'04' SIO routine entered after interrupt
		X'02' Reserved
		X'01' IPL in progress
66	42	X'80' Initial RAS request
		X'40' RAS WAIT request outstanding
		X'20' RAS IPL in progress
		X'10' Reserved
		X'08' POWER supported
		X'04' POWER initialized
		X'02' GETREAL for SDAID in progress
		X'01' Fetch for system task in progress (used by PDAID's)
67	43	Reserved

LAYOUT OF SYSTEM TASK SELECTION CONTROL FIELD

244	F4	SVA Flag
		X'80' Do not test for warm start copy of SVA
		X'40' SDL active
		X'20' No 'Set SVA' or 'Set SDL. allowed
		X'10' Build of SDL in progress
		X'08' SDL overflow
		X'04', X'02', X'01' Reserved

Dec	Hex	Description
68	44	Always zero
69	45	SELECT byte:
		X'00' No system task active
		X'01' RAS active
		X'02' PMGR active
		X'03' SUPVR active
		X'04' CRT active
		X'05' ERP active
		X'06' PAGEIN active

Figure 4.5 SYSCOM Expansion flag bytes.

I/O TABLES AND INFORMATION BLOCKS

The I/O tables and information blocks in the supervisor establish the interface between a specific task and the hardware channels. For example, for every logical unit name (SYSXXX) used, there must be a LUB (Logical Unit Block) entry. For every physical device used by the system, there must be a PUB (Physical Unit Block) entry. Then, when an I/O request occurs, an entry is made, via LUB and PUB, in the CHANQ (Channel Queue). An I/O request queued in the CHANQ contains an address that points to the CCB (Command Control Block) that contains the address of the channel program.

During the I/O device operation, entries are made in the ERRQ (Error Queue) in the Error Recovery Block, if a hardware failure occurs on the device.

Figure 4.6 (opposite) is an overall picture that illustrates the connections between the tables, (It does not represent their actual position, size, or relation as they are stored or printed in a dump output.) The initial pointers for all the tables and blocks are stored either in SYSCOM or in the active partition comreg.

Where a connection exists between tables, for example, the PUB and the CHANQ, pointers are contained in the table entries concerned. For example, the Figure shows a one-byte pointer in the PUB entry number 00 that relates this PUB entry to CHANQ entry number 00.

Figures 4.7 through 4.16 show the format of the tables and describe their contents.

A formatted stand-alone dump output prints the contents of the tables in a more readily understood format, and an example of such an output is shown in appendix G. The stand-alone dump program is described in Section 2-A of this manual.

Byte 0: Channel number. (Hex 0-6, FF = NULL)

Byte 1: I/O device unit number

Byte 2: Hex 0, 1, 2, ... points to the first channel queue entry for this device.

Byte 3: If device is a 2495 Tape Cartridge Reader and TEBs are specified, this byte is a TEB pointer (Hex 1, 2, 3, ...). Otherwise, this byte is a ERP retry counter.

Byte 4: Device type code. See Figure 4.10, parts 1 through 3.

Byte 5: SS of the MODE= parameter in the DVCGEN macro for tape unit (See Section 2). For the Models 115 and 125 ICA line, this byte contains the displacement index of the entry in the Line Mode Table (LMT). The address of the LMT is contained in bytes X'8C' – X'8F' of SYSCOM.

For DASD with seek separation, this byte is used as the SAB Pointer. With Track Hold but not seek separation supported, this byte contains a pointer to the Track Hold Table entry or X'FF' (with both SKSEP and TRKHLD specified, the track hold pointer is found in the SAB entry).

For MICR type devices, this byte indicates which external interrupt line is in use.

For a 3705 Communications Controller, this byte contains the type number of the Channel Adapter.

For 2560 or 5425

Bit 0: 1= Repositioning required
1: 0= SYSPCH temporarily assigned to hopper 1
1= SYSPCH temporarily assigned to hopper 2
2: 0= SYSIPT temporarily assigned to hopper 1
1= SYSIPT temporarily assigned to hopper 2
3: 0= SYSRDR temporarily assigned to hopper 1
1= SYSRDR temporarily assigned to hopper 2
5: 0= SYSPCH permanently assigned to hopper 1
1= SYSPCH permanently assigned to hopper 2
6: 0= SYSIPT permanently assigned to hopper 1
1= SYSIPT permanently assigned to hopper 2
7: 0= SYSRDR permanently assigned to hopper 1
1= SYSRDR permanently assigned to hopper 2.

Byte 6: Channel Scheduler Flags

Bit 0: 1= Device busy
1: 1= Switchable device
2: 1= EOJ for SYSRDR or SYSIPT
3: 1= I/O error queued for recovery
4: 1= Operator intervention required
5: 1= Device End posting required
6: 1= Burst mode or overrunnable device on byte MPX channel
7: 1= 7-track tape unit.

Byte 7: Job Control Flags

Bit 0-4: Standard MODE assignment for 7-track tape (all ones if not tape, all zeros if device is down)

Bit 5: 1 = DASD device with Rotational Position Sensing (RPS) feature

Bit 6-7: B'11' (both on)= Headqueue in progress
B'01' = Headqueue requested.

Note: A null is generated for each device to be supported by the supervisor. Standard physical unit assignments are made to the PUB table at supervisor generation time.
PUBs are ordered by channel and priority within a channel.
An entry in the PUB Ownership Table is associated with each entry in the PUB Table, if the supervisor has been generated to support multiprogramming.

Figure 4.9. Explanation of the contents of an entry in the PUB table

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
2400T9	9-track Magnetic Tape Tape units	50	Magnetic Tape devices
2400T7	7-track Magnetic Tape units	50	
3410T9	9-track 3410 Magnetic Tape units	53	
3410T7	7-track 3410 Magnetic Tape units	53	
3420T9	9-track 3420 Magnetic Tape units	52	
3420T7	7-track 3420 Magnetic Tape units	52	
2495TC	2495 Tape Cartridge Reader	51	Tape Cartridge Reader
1442N1	1442N1 Card Read Punch	30	Card Read Punches
2520B1	2520B1 Card Read Punch	31	
2560	2560 Multifunction Card machine	33	
2596	2596 Card Read Punch	30	
3525RP	3525 Card Punch (with optional read feature)	32	
5425	5425 Multifunction Card Unit	34	
2501	2501 Card Reader	10	Card Readers
2540R	2540 Card Reader	11	
3504	3504 Card Reader	12	
3505	3505 Card Reader	12	
1540P	2540 Card Punch	21	Card Punches
2520B2	2520B2 Card Punch	20	
1442N2	1442N2 Card Punch	22	
2520B3	2520B3 Card Punch	20	
3525P	3525 Card Punch	23	
1403	1403 Printer	40	Printers
1403U	1403 Printer with UCS feature	42	
1443	1443 Printer	41	
2260(local)	1053 Printer with 2848 Control Unit. MODE operand must be entered as X'01'	C0	
3203	3203 Printer	4A	
3211	3211 Printer	43	
3277 (local 3270)	3284 or 3286 Printer with 3272 Control Unit. MODE operand must be entered as X'01'	B0	
3277B (local 3270)	3284 or 3286 Printer with 3272 Control Unit, attached in burst mode to a multi-plexer channel. MODE operand must be entered as X'01'	B0	
5203	5203 Printer	4C	
52030	5203 Printer with UCS feature	4D	

Figure 4.10 **Device Type Codes, part 1 of 3**

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
1050A	3210, 3215 Console Printer Keyboards	00	Printer-Keyboards
125D	Models 115 and 125 Integrated Video Display Unit	B2	Video Display Unit
125DP	Models 115 and 125 Integrated Video Display Unit with 5213 Console Printer attached	B2	
UNSP	Unsupported device	FF	Unsupported. No burst mode on multiplexer channel
UNSPB	Unsupported device	FF	Unsupported with burst mode on multiplexer channel
2311	2311 Disk Storage device	60	DASD
2314	2314 Direct Access Storage Facility	62	
2314	2319 Disk Storage Facility	62	
2321	2321 Data Cell Drive	61	
3330	3330-1, 3330-2, or 333-1 Disk Storage	63	
3340	3340 Disk Storage (general)	68	
3340	3340 Disk Storage with 3348 Mod 35	69	
3340	3340 Disk Storage with 3348 Mod 70	6A	
1419	1255 Magnetic Character Reader	72	MICR- Magnetic Ink Character Recognition devices
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 Contr. Unit	74	
2701	2701/2715 Data Adapter Unit	D0	Teleprocessing lines
2702 A B C D	2702 Transmission Control Unit	D1	A= SAD0 comm'd B= SAD1 comm'd C= SAD2 comm'd D= SAD3 comm'd when enabling the line
2703	2703 Transmission Control Unit	D2	
2703	Integrated Communications Adapter (Models 125 and 135)	D2	
2703	3705 Communications Controller in Emulation Mode	D2	
2955	2955 Data Adapter Unit	D7	Data Link for RETAIN
1017	1017 Paper Tape Reader with 2826 Control Unit	78	Paper Tape Readers
2671	2671 Paper Tape Reader	70	

Figure 4.10 Device Type Codes, part 2 of 3

Section 4, Chapter 3

I/O TABLES AND INFORMATION BLOCKS

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
1018	1018 Paper Tape Punch with 2826 Control Unit	79	Paper Tape Punch
1419	1270 Optical Reader/ Sorter	72	Optical Readers
1419P	1275 Optical Reader/ Sorter Primary Control Unit	73	
1419S	1275 Optical Reader/Sorter Secundary Control Unit	73	
1287	1287 Optical Reader	77	Optical Readers
1288	1288 Optical Page Reader	77	
3881	*3881 Optical Mark Reader	11	
3886	3886 Optical Character Reader	7C	
3540	3540 Diskette I/O unit		DISKETTE
2260	2260 Display Station	C0	Display Stations
3277 (local 3270)	3277 Display Station; MODE operand need not be entered	B0	
3277B (local 3270)	3277 Display Station; attached in burst mode to a multiplexer channel. MODE operand need not be entered	B0	
7770	7770 Audio Response Unit	D3	Audio Response unit

**Note:* *The logical unit name SYSIN cannot be assigned to a 3881*

Figure 4.10 Device Type Codes, part 3 of 3

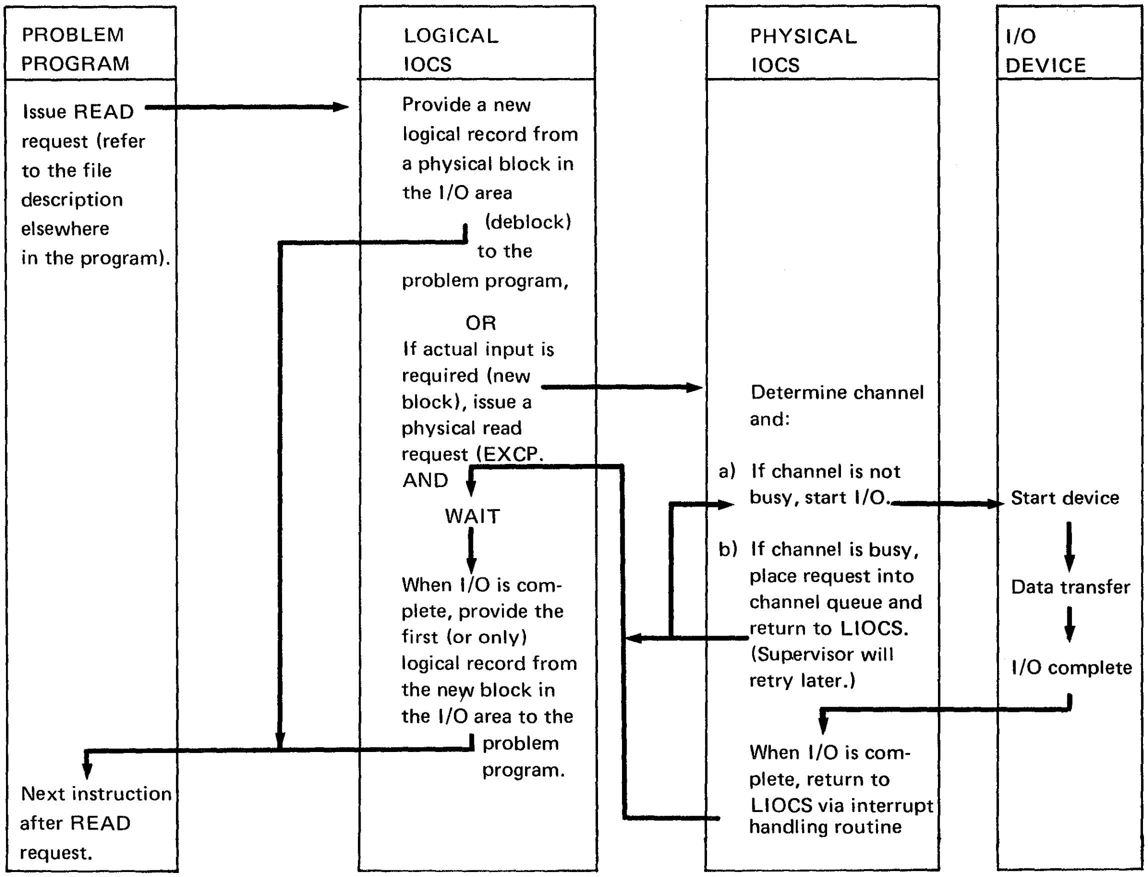


Figure 4.17 Example of LIOCS and PIOCS interrelationship.

Explanation of Figure 4.17:

Logical IOCS makes a request to physical IOCS to start an I/O operation by means of the EXCP macro instruction. From information in the CCB, physical IOCS determines the channel for which the request was made and places the request on a queue for that device. If the channel(s) or device is not busy, the I/O is started and control returns to the problem program. If the channel is busy, control returns to the supervisor task selection routines, but the I/O request waits in the channel queue. When the request reaches the top of the channel queue, the I/O is started.

Control returns to the program requesting the I/O unless there was an error condition detected on the START I/O (SIO) instruction. The problem program normally continues processing until it requires that the requested I/O operation be complete (either the information being read into real storage is needed, or the output area must be freed on an output operation). At this time, the WAIT macro causes the now waiting task to be removed from task selection until the proper interrupt is processed for this device by the supervisor.

Section 4, Chapter 4

I/O CONTROL SYSTEM

Using PIOCS and LIOCS Macro Instructions.

By use of macro instructions you can create, access and maintain files at both physical and logical IOCS levels. Through these macro instructions, the user can communicate with the pre-written routines and tailor them to his needs.

As part of most user programs, LIOCS provides an interface between user programs, LIOCS provides an interface between user's file processing routines and PIOCS. (All COBOL, FORTRAN, RPG II,PL 1OPTand PL/I(D) programs use LIOCS; most assembler programs use LIOCS.)

Using PIOCS

Using PIOCS requires a detailed knowledge of device control and system operation. A channel program using the CCW assembler instruction must be written in conjunction with three macro instructions provided to communicate with PIOCS.

CCB: This macro instruction generates a command control block. (Refer to Chapter 5 in this Section for a description of the CCB.)

EXCP: This macro instruction is converted to an SVC 0 to request execution of the channel program. It supplies the location of the corresponding CCB to the supervisor.

WAIT: This macro instruction generates an SVC 7 which tests CCB byte 2 bit 0 (traffic bit) to determine when an I/O operation is complete. If the operation is not complete, the supervisor gets control until PIOCS within the supervisor sets the traffic bit to indicate completion of the operation. The WAIT macro should always be used for each I/O operation.

A channel program written to make use of the RPS feature of a direct-access device must contain Set Sector commands and either Read Sector commands or the SECTVAL macro instruction.

SECTVAL: This macro instruction generates an SVC 75 to supply the sector in which the record is located.

For information on the format of Sector CCWs and on Rotation Position Sensing see the Appropriate reference manuel for the device.

The example below shows part of an assembly program listing using the EXCP WAIT and CCB macros. A full description of these MACRO instructions can be found in *DOS/VS Supervisor and I/O Macros.*

```
003800 47F0 B75C       03F5E        261           B     SKIP
                                    262 ***********************************************************************
                                    263 *BUG GENERATOR ROUTINE*
                                    264 ***********************************************************************
003804 5010 C61E             04E20  265 BUG       ST    1,R1SAVE           SAVE CONTENTS OF RETURN REGISTER
                                    266           EXCP  TCCB1
                                    270           WAIT  TCCB1
                                    276           EXCP  TCCB2
                                    280           WAIT  TCCB2
003830 9640 C83F       05041        286           OI    OPRESP1,X'40'
003834 95D3 C83F       05041        287           CLI   OPRESP1,C'L'
003838 4780 B266             03866  288           BC    8,LOOP
```

(Ex 83 RR)

```
004E8C 40404040                     1347 TOT      DC    CL4' '
004E90 4040404040404040             1348 TOWN     DC    CL10' '
                                    1349 TCCB1    CCB   SYSLOG,TCCW1,X'0400'
                                    1360 TCCB2    CCB   SYSLOG,TCCW2,X'0400'
004EBA 000000000000
004EC0 09004F106000001C             1371 TCCW1    CCW   09,MSSG1,X'60',28
004EC8 09004F2C60000020             1372          CCW   09,MSSG2,X'60',32
004ED0 09004F4C6000002F             1373          CCW   09,MSSG3,X'60',47
004ED8 09004F786000001F             1374          CCW   09,MSSG4,X'60',31
004EE0 09004F9A60000021             1375          CCW   09,MSSG5,X'60',33
004EE8 09004FB860000021             1376          CCW   09,MSSG6,X'60',33
004EF0 09004FDC60000020             1377          CCW   09,MSSG7,X'60',32
004EF8 09004FFC60000020             1378          CCW   09,MSSG8,X'60',32
004F00 0900501C20000025             1379          CCW   09,MSSG9,X'20',37
004F08 0A00504120000001             1380 TCCW2    CCW   10,OPRESP1,X'20',1
                                    1381 ***********************************************************************
                                    1382 *MESSAGE CONSTANTS*
```

(Ex 84 RR)

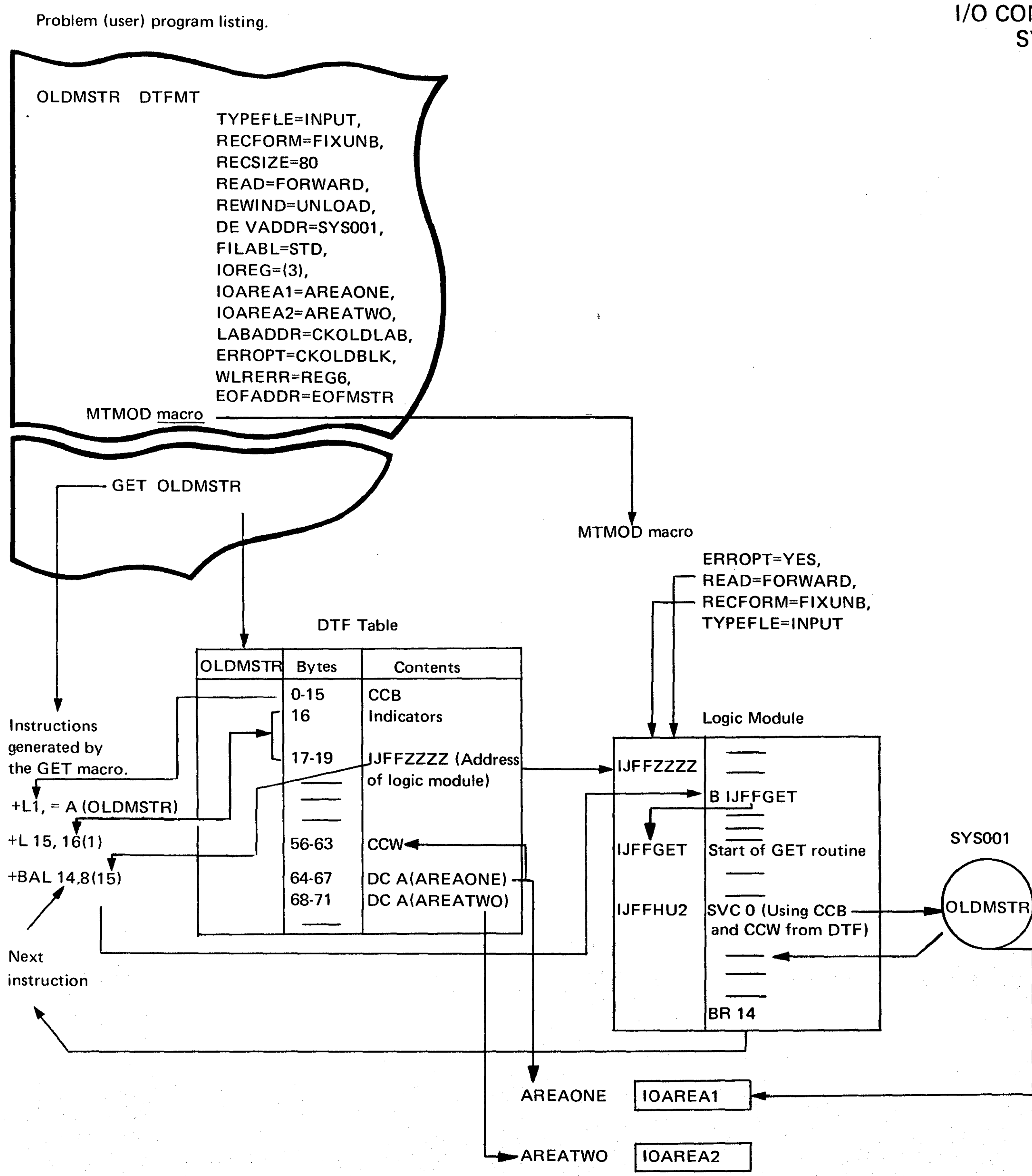


Figure 4.24. A summary of the relationships between an imperative macro, a declarative macro, and a module generation macro specified in a program.

The GET imperative macro is used in this illustration, which also shows the linkage between the generated DTF table and the logic module.

RPS (Rotational Position Sensing) Option

System Support

RPS support for devices attached to block multiplexer channels in full block multiplex mode (or their equivalent on Model 3115/3125 CPUs) is provided as an option in DOS/VS. The option is specifiable at the operating system and device level. System support is provided at system generation time by coding the FOPT macro with RPS=YES. The IBM 3330/3333 supports RPS as a standard feature. The IBM 3340 supports RPS when 3340R is specified in the DVCGEN macro. Please note that Block Multiplexing cannot be used with the 2311-1/3330, 2311-1/3340, and 2314/3340 Series Compatibility Features, if your CPU is a Model 3115 or a Model 3125.

Data Management Support

RPS support will be provided dynamically in Data Management when the operating system and the device support and all of the following conditions are met:

- One of the DASD access methods is being used, that is, the file is defined by one of the following DTFs: DTFSD, DTFDA, DTFIS, DTFDI, or DTFPH .
- There is room in the user's virtual storage to extend the DTF.
- An RPS version of the logic module necessary to process the DTF has been, or can be, loaded into the SVA.

At OPEN time, if it is determined that the system and the device both support RPS, a bit is set in the DTF tables (see Figure 4.24A). This bit will not be turned off until CLOSE time whether or not the other conditions for RPS support are met. Space is then obtained for the DTF extension. The amount is dependent on which access method is in use (see Figure 4.24A). If space is unavailable, the DTF will be opened without RPS support.

Determination as to which RPS logic module is required to process this DTF is made and the module is loaded into the SVA, unless it is already there in which case it is sharable across partitions. If the required logic module cannot be made available, OPEN releases the DTF extension space and the DTF is opened without RPS support.

If the space for the DTF extenstion is available and the RPS logic module is loaded into the SVA, OPEN sets another bit in the DTF table indicating that the data set will be processed in RPS mode (see Figure 4.24A).

The first section of the DTF extension contains the RPS channel program (so that the pointer to the extension is also the pointer to the RPS CCW chain). The extension also contains CCW build and work areas necessary to construct the RPS channel program, a sector value bucket, and register and address save areas (see Figure 4.24B for DTF extension format).

The addresses of the original channel program and logic module are saved in the extension while the address of the RPS channel program is put into bytes 9 - 11 of the DTF and the address of the RPS logic module is put into bytes 17 - 19. These pointers are restored at CLOSE time.

The original DTF is used for all fields except the channel program so that no mapping between the DTF and the extension is required. (see Figure 4.24C for an overview of the OPEN.)

No program recompiling or relink-editing is required, though there must be enough dynamically allocatable space in the user's partition for the RPS extensions. Since RPS gets this space via the GETVIS macro, the SIZE= parameter must be specified in the program's EXEC statement.

The DTF extension provides a register save area for the RPS logic modules since they are all reentrant and sharable between partitions. If the original non-RPS logic module is reentrant because it was coded read-only, the user-supplied save area will not be used. The RPS logic modules are supersets of functions needed to process the DTF.

System Component Support

System Components support RPS where there is a significant amount of DASD I/O. This support is provided by building RPS channel commands and then changing these to NO-OPs or TICs if the affected device or system does not support RPS.

Wherever the component uses LIOCS for its I/O, this optional support is provided through the data management support of RPS.
Where LIOCS is not used, the component logic interrogates the indicator set by OPEN for I/O using DTFPH or, when DTFPH is not used, the same PUB and COMREG indicators interrogated by OPEN.

The system components supporting RPS are:

- POWER
- Supervisor Fetch and Paging I/O
- Linkage Editor
- Job Control
- Librarian
- Checkpoint/Restart
- System Utilities

	DTFDA DTFPH*	DTFSD DTFPH*	DTFSD (work files)	DTFDI	DTFIS (All)
DTF offset: set at byte	32(20)	44(2C)	37(25)	42(2A)	65(41)
System supports RPS––––bit	1	1	1	7	4,7#
DTF has been extended––––bit	7	7	7	1	5
Length of DTF extension	512	256	256	256	384

* DTFPH has no logic module; therefore, the only RPS processing done is the setting of the System Support RPS bit.

\# Bit 4 on - prime data resides on an RPS device
Bit 7 on - index resides on an RPS device

Figure 4.24A. DTFs for RPS Support

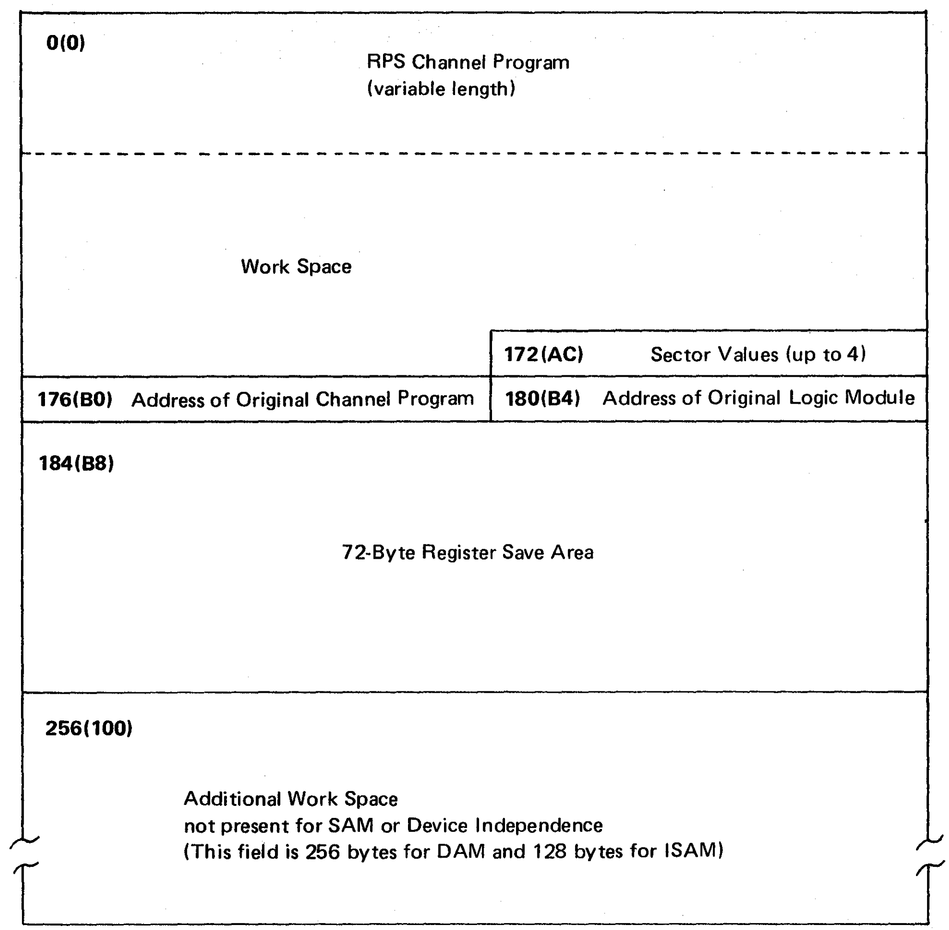


Figure 4.24B DTF Extension Work Area for RPS

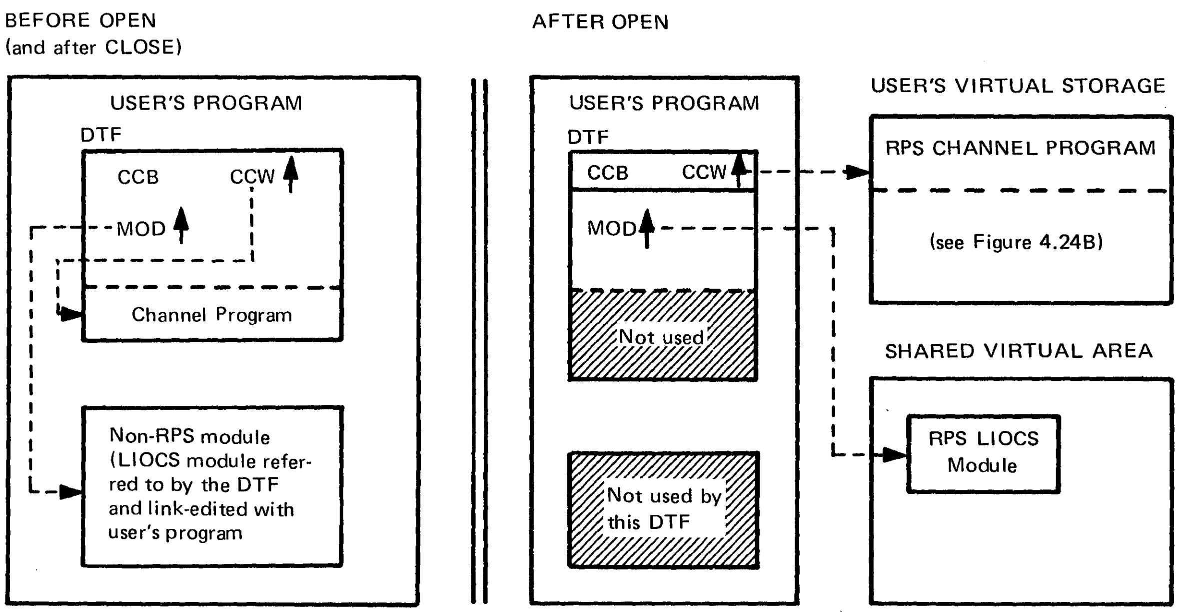


Figure 4.24C Effect of RPS Support on OPEN

Section 4, Chapter 4

I/O CONTROL SYSTEM

VSAM (Virtual Storage Access Method) I/O

VSAM IOCS differs from that of other DOS/VS access methods as follows:

- VSAM declarative macros are ACB, EXLST, and RPL instead of DTF and xxMOD.
- VSAM routines are dynamically loaded into virtual storage when a VSAM file is opened. They are not assembled or link edited with the user's program.

Declarative Macros

The VSAM declarative macros are ACB, which creates an Access-Method Control Block; EXLST, which creates an Exit List; and RPL, which creates a Request Parameter List. The Access-Method Control Block (ACB) is like a DTF in that it defines the file to be processed. Opening a VSAM file involves opening the ACB for that file. The Request Parameter List (RPL) defines the parameters necessary for a particular execution of a request (imperative) macro. It contains some of the information, such as address of the user's work area, located in the DTF in other access methods. The Exit List (EXLST) contains the addresses of optional user exit routines. Up to four exit routines can be specified--one for handling end-of-file, one for handling logical errors, one for handling I/O errors, and one to allow user processing during VSAM I/O operations.

Codes indicating errors resulting from execution of imperative macros are set in registers or in the ACB or RPL as described below.

Imperative Macros

The user's program issues imperative macros to open or close a file and to retrieve, add, delete, or update records. It can also issue imperative macros to generate, modify, display, or test the control blocks created by the declarative macros. When control is returned to the user's program after execution of an imperative macro, a "return code" is set in the low-order byte of register 15. The return code indicates the results of the macro execution. If an error or certain other exceptional conditions occur, an "error code" will be set in the ACB, the RPL, or in register 0, depending on the macro. Figure xx summarizes the return codes and error codes issued by the imperative macros and user exit routines which can be used. More information on the return codes and user exits as well as a complete list of the error codes and their meanings is in the VSAM chapter of *DOS/VS Supervisor and I/O Macros.*

An ACB, EXLST, or RPL can be created dynamically, during program execution, by using the GENCB macro. The fields in these control blocks can be modified during program execution by using the MODCB macro. Refer to *DOS/VS Supervisor and I/O Macros* for information on how to write the GENCB and MODCB macros.

RPL Debugging Hints

If the RPL hold byte, 35(23), is set to X'FF', the error occurred while the request was being executed by VSAM. Check the type of request byte, 29(1D), to determine what request was active. If the request was a POINT, GET, or PUT, check the following parameters in the RPL (of GENCB for RPL) in your program to ensure that they are valid:

Macro	Check these RPL Parameters
POINT	ARG and KEYLEN
GET	AREA and AREALEN IF OPTCD=DIR or OPTCD=SKP, also check ARG and KEYLEN
PUT	AREA and RECLEN

If the RPL parameters are specified correctly or if the type of request is other than POINT, GET, or PUT, the error is probably in VSAM itself. Save the dump, console log, and program listing and contact your IBM programming support representative.

Note: MODCB, SHOWCB, and TESTCB macros also set the RPL Hold byte. If this byte was set by one of those macros, the type of request byte will have no meaning.

CCB AND THE CHANNEL PROGRAM

Command Control Block (CCB)

This information block is generated in the problem program during assembly or during program operation, depending on the methods of I/O control employed by the program. As described in Chapter 4 the CCB is generated as the first 16 bytes of a DTF when the program is using LIOCS. When using PIOCS, the CCB macro generates the CCB.

The CCB establishes communication between the problem program and physical IOCS. The CCB is 16 bytes in length with eight major fields, and does not have to be aligned on a doubleword boundary. Eight optional bytes are generated if the user requests that a sense operation be performed on the occurrence of an I/O error. Data transferred from the device to real storage during a sense operation provides information concerning unusual conditions detected in the last operation and the status of the device. All data in the CCB is in the hexadecimal format.

By examining the contents of the CCB in a dump, the following information can be obtained about the associated I/O operation:

- Whether the operation was completed (by inspecting the traffic bit and device-end bit)
- Status of the channel and device to which the I/O command was issued
- The logical unit involved in the operation
- Whether the CCB is in a real or a virtual partition
- The address of the channel program (the first CCW in a CCW string)
- The address of the next CCW to be executed in the channel program (Subtracting eight from this address gives the address of the last CCW used.)
- The residual count associated with the last CCW.

Note: When all the following coditions have been met, bytes 9-11 will now be pointing to a non-RPS channel program in the DTF, but the one actually used has been released from the user's virtual save area:

- RPS was in effect.
- The data set has been closed.
- The CCB was generated as the first 16 bytes of a DTF in a program using LIOCS.

This count taken from the channel status word (CSW), is stored by PIOCS when the pointer to this CCB is removed from the channel queue. The residual count, in conjunction with the original count specified in the last CCW used, indicates the number of bytes transferred to or from the area designated by the CCW. When an input operation is terminated, the difference between the original count in the CCW and the residual count in the CSW is equal to the number of bytes transferred to storage. For an output operation, the difference is equal to the number of bytes transferred to the I/O device.

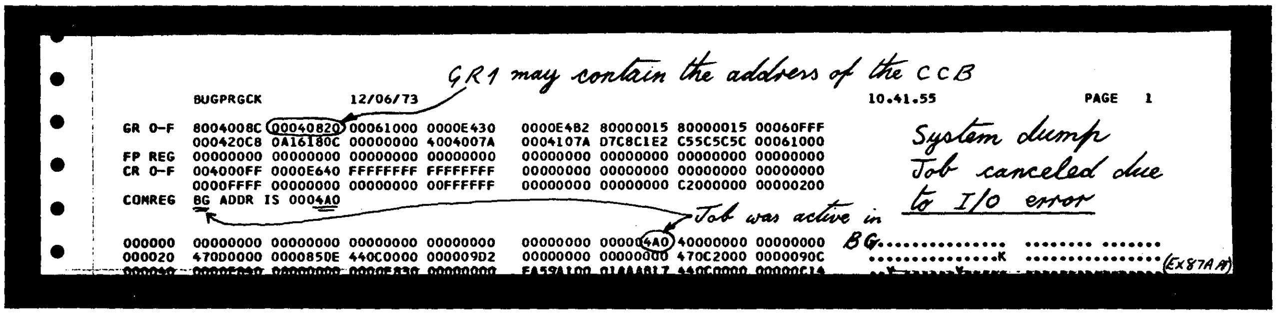
CCB AND THE CHANNEL PROGRAM

How to locate

1. For programs using LIOCS, locate the address of the associated DTF in the program listing. Then use the linkage editor map to locate the DTF in the dump. The first 16 bytes of the DTF is the CCB.

2. For programs using PIOCS, locate the address of the CCB macro in the program listing and use the linkage editor map to locate the CCB in the dump.

3. If the interrupt code in the PSW stored in the partition save area is 00 or 07 (SVC 0 or SVC 7), the contents of general register 1 may contain the address of the CCB. To confirm whether the address in register 1 is that of a CCB, inspect the first few bytes starting from that address. (It is not difficult to recognize a valid CCB in a dump. See the example below.)



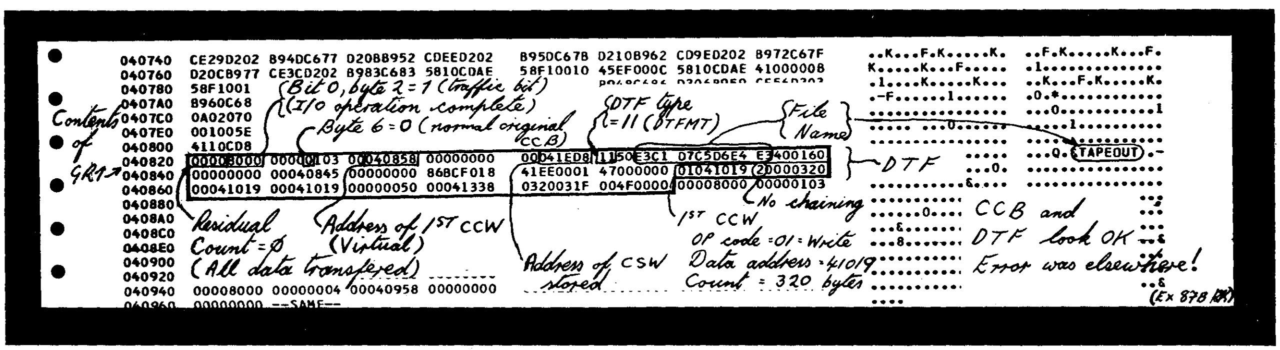


Figure 4.26 The pointer and CCB in a dump

Figure 4.27 parts 1, 2 and 3 illustrate the format and contents of the information contained in any CCB.

SVC Dec	SVC Hex	Macro supported	Function
50	32		Reserved for LIOCS error recovery
51	33		Return phase header
52*	34	TTIMER	Return the remaining time interval, or cancel a time interval
53	35		Reserved
54	36	FREEREAL	
55	37	GETREAL	Provide interf. between SDAID and PDAID initialization routine and page management routine, to create the PDAID alternate area or the SD area
56*	38	GETPUB-FREEPUB	Occupy or free PUB of the device used by POWER
57*	39		Make POWER-supported partition dispatchable
58	3A		Provide interface between job control and the supervisor. Get real storage for real jobs
59	3B		Provide interface between EOJ and the supervisor. Reset the storage key for virtual jobs
60	3C	GETADR	Provide virtual address of location within I/O areas for ERP and CRT routines
61*	3D	GETVIS	Get storage in virtual partition
62*	3E	FREEVIS	Free storage in virtual partition
63	3F	USE	Use a resource
64	40	RELEASE	Release a resource
65*	41	CDLOAD	Load VSAM or core image phase
66	42	RUNMODE	Return mode in which program is running
67*	43	PFIX	Fix page(s) in real storage
68*	44	PFREE	Free page(s) in real storage
69*	45	REALAD	Return real address corresponding to a given virtual address
70*	46	VIRTAD	Return virtual address corresponding to a given real address
71*	47	SETPFA	Establish or terminate the linkage between the supervisor and a user page-fault appendage routine
72*	48	GETCBUF-FREECBUF	Get or free copy buffer for IDAL or tape ERP
73*	49	SETAPP	Allow linkage to channel and appendage routines
74*	4A		Fix page(s) in real storage for restart
75	4B		SECTVAL Calculate value of sector for RPS
76	4C		Initiate RMS recording of an I/O error
77	4D	TRANSCSW	Returns the virtual address of a copied CCW
78 thru 84			Reserved
85	55	RELPAG	Release contents of one or more pages
86	56	FCEPGOUT	Force a page-out for one or more pages
87	57	PAGEIN	Page in one or more pages

* optional

Figure 4.29 Supervisor Calls (Part 3 of 3)

Section 4, Chapter 7

PIB AND PIB2

The PIB (program Information Block)

Real storage area is reserved in the supervisor for this information block by the MPS multiprogramming and/or NPARTS and AP (Asynchronous Processing) parameters of the SUPVR supervisor generation macro. Each entry in this block is 16 bytes and contains status information about the program and, if AP is supported about the subtasks running in each partition supported by the supervisor.

The first entry is reserved for the attention routine, this entry is called the Attention PIB (AR PIB).

Other entries in the PIB belong to the problem programs and subtasks. The sum of all subtasks and problem program entries may not exceed 15. The maximum number of entries, including the attention PIB and AP (subtask) PIBs, is 16.

For a supervisor that is not generated to support more than one partition there is only one 16-byte entry, which is shared by the attention routine and the problem program.

By examining the data recorded in the appropriate PIB entry, the status and location of programs running in any partition can be established. Some of the more important data to be looked at in the PIB during the first analysis of a dump output are:

- Byte 0, from which you can determine whether the program is waiting for
  - The LTA (Logical Transient Area), X'81'
  - The PTA (Physical Transient Area), X'85'
  - An I/O interrupt, X'82'
  - A page to be paged in, X'87'
  - A page to be paged in with QTAM active, X'8F'
- Byte 4, X'80', which indicates that the job or task is running in virtual mode
- The address of the program save area
- The address of the system save area.

Figure 4.30 (opposite) shows the format and describes the contents of an entry in the PIB.

How to locate:

Bytes X'5A' – X'5B' of the partition comregs contain the address of the first entry in this information block. Label PIBTAB in the supervisor listing identifies the address of the first byte of this information block.

Appendix G shows an example of locating the PIB in a dump output.

Appendix G

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
079FE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ............... ...............

BLOCK 244

07A000  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ............... ...............
07A020  --SAME--
07A7E0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ............... ...............

BLOCK 245 <-- Last 2K block but one of CPU physical (main) storage

07A800  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ............... ...............
07A820  --SAME--
07AFE0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   ............... ...............

        END OF CORE DUMP
```

(EX 109K)

```
IJBSEGT CONTENTS IS TAKEN TO LOCATE THE SEGMENT TABLE

VIRT.ADDR  REAL ADDR  BLOCK  STATUS

  000000    000000    000   CHANGED
  000800    000800    001   CHANGED
  001000    001000    002   CHANGED
  001800    001800    003   UNCHANGED
  002000    002000    004   CHANGED
  002800    002800    005   CHANGED
  003000    003000    006   UNCHANGED
  003800    003800    007   CHANGED
  004000    004000    008   CHANGED
  004800    004800    009   CHANGED
  005000    005000    010   CHANGED
  005800    005800    011   CHANGED
  006000    006000    012   CHANGED
  006800    006800    013   UNCHANGED
  007000    00700r    014   CHANGED
  007800    007800    015   CHANGED
  008000    008000    016   CHANGED
  008800    008800    017   CHANGED
  009000    009000    018   CHANGED
  009800    009800    019   UNCHANCED
  00A000    00A000    020   CHANGED
  00A800    00A800    021   CHANGED
  00B000    00B000    022   CHANGED
  00B800    00B800    023   CHANGED
  00C000    00C000    024   CHANGED
  00C800    00C800    025   CHANGED
  00D000    00D000    026   CHANGED
  00D800    00D800    027   CHANGED
  00E000    00E000    028   CHANGED
  00E800    00E800    029   CHANGED
  00F000    00F000    030   CHANGED
  00F800    00F800    031   CHANGED
```

Segment 0

Segment 0, all 32 pages in real storage (for the supervisor)

```
  010000    010000    032   CHANGED
  010800    010800    033   CHANGED
  011000    011000    034   CHANGED
  011800    011800    035   CHANGED
  012000    012000    036   CHANGED
  012800    012800    037   CHANGED
  013000    013000    038   CHANGED
  013800    013800    039   CHANGED
  014000    014000    040   CHANGED
  014800    014800    041   CHANGED
  015000    015000    042   CHANGED
  015800    015800    043   CHANGED
  016000    016000    044   CHANGED
  016800    016800    045   CHANGED
```

Segment 1, only 14 pages in real storage from this segment (also for the supervisor)

No pages from Segment 2 were in real storage

```
  03D800    03D800    123   CHANGED
  03E000    03E000    124   CHANGED
  03E800    03E800    125   CHANGED
  03F000    03F000    126   CHANGED
  03F800    03F800    127   CHANGED
```

Segment 3, 5 pages frames used by PDAID GSVC trace as an alternate area.

```
  040000    020800    065   CHANGED
  040800    019800    051   CHANGED
  041000    017800    047   CHANGED
  041800    018000    048   CHANGED
  042000    018800    049   CHANGED
```

Segment 4, 5 page frames used by BG job running in virtual mode.

No pages from segment 5

```
  061000    01C800    057   CHANGED
  061800    01D000    058   CHANGED
```

Segment 6

(EX 110K)

Figure G.1 part 7 of 20

Appendix G

EXAMPLE OF A STAND ALONE DUMP OUTPUT

```
09E800    027800    079   UNCHANGED  } Part of segment 9
09F000    027000    078   UNCHANGED
09F800    026000    076   UNCHANGED

0A0000    028800    081   CHANGED    } Segment 10
0A0800    028000    080   CHANGED
0A1000    02A800    085   UNCHANGED
0A1800    028000    086   UNCHANGED
Address of last virtual page in real storage
                                          Indicates that operator did not
                                          execute the STORE STATUS function
STORE STATUS FUNCTION NOT EXECUTED, CR IN INITIAL STATUS

CR 0-7  000000E0 00000000 FFFFFFFF FFFFFFFF   00000000 00000000 00000000 00000000
CR 8-F  00000000 00000000 00000000 00000000   00000000 00000000 C2000000 00000200
                                                                        (Ex 111 KT)
```

```
                              *** COMMUNICATION REGION ***
HEX   BG            F4            F3            F2            F1
DISP  04A0          4590          44F0          4450          4380          COMMUNICATION REGION ADDRESS
 00   12/06/73      12/06/73      12/06/73      12/06/73      12/06/73      DATE
 08   7000          7000          7000          7000          7000          PPBEG ADDR
 0A   7000          7000          7000          7000          7000          END OF STORAGE PROTECT
 0C   0000          0000          0000          0000          0000          SEEK ADDRESS BLOCK,ONLY BG VLD
 0E   0000000000    0000000000    0000000000    0000000000    0000000000    PROBLEM PROGRAM USERS
      00000000      00000000      00000000      00000000      00000000          AREA IN HEX
 17   00            00            00            00            00            UPSI BYTE IN HEX
 18   TOMSDMP       KENSLOOP      NO NAME       CALCSIM       NO NAME       JOB NAME
 20   00060FFF      00074FFF      00000000      000A1FFF      00000000      UPPERMOST BYTE OF EACH PPA
 24   0004232F      00063C5B      00000000      0008AFF6      00000000      END ADDR OF LAST FETCH OR LOAD
 28   0004232F      00000000      00000000      0008D4DF      00000000      LARGEST PROBLEM PROGRAM PHASE
 2C   0000          0000          0000          0000          0000          LENGTH OF PP LABEL AREA
 2E   0040          0020          0030          0040          0050          PROGRAM IDENTIFICATION KEY
 30   000BFFFF      000BFFFF      000BFFFF      000BFFFF      000BFFFF      END OF STORAGE ADDRESS
 34   FD            FD            FD            FD            FD            MACHINE CONFIGURATION
 35   7F            7F            7F            7F            7F            SYSTEM CONFIGURATION
 36   CED330A0CED0  CED33000CED2  CED30000CE50  CED330804ED0  CED30000CE50  JOB CONTROL SWITCHES
 3C   002E          002E          002E          002E          002E          DISK ADDR OF LABEL CYLINDER
 3E   40FD          40FD          40FD          40FD          40FD          ADDR OF FOCL
 40   4104          4104          4104          4104          4104          ADDR OF PUB
 _    4296          4296          4296          4296          4296          ADDR OF FAVP
 44   4297          4297          4297          4297          4297          ADDR OF JIB
 46   4389          4389          4389          4389          4389          ADDR OF TEB
 48   3F00          3F00          3F00          3F00          3F00          ADDR OF FICL
 4A   3F06          3F06          3F06          3F06          3F06          ADDR OF NICL
 4C   3F0C          3F0C          3F0C          3F0C          3F0C          ADDR OF LUB
 4E   38            38            38            38            38            LINE COUNT FOR SYSLST
 4F   120673340     120673340     120673340     120673340     120673340     SYSTEM DATE
 58   0000          0000          0000          0000          0000          LIOCS COM BYTE
 5A   3CD4          3CD4          3CD4          3CD4          3CD4          ADDR OF PIB TABLE
 5C   0000          0000          0000          0000          0000          LAST CHECK POINT NO.
 5E   003C          003C          003C          003C          003C          JOB ZONE IN MINUTES
 60   4630          46A8          4720          4798          4810          ADDR OF DIB
 62   0000          0000          0000          0000          0000          CURRENTLY NOT ASSIGNED
 64   3DCC          3DCC          3DCC          3DCC          3DCC          ADDR OF PC OPTION TABLE
 66   3E4C          3E4C          3E4C          3E4C          3E4C          ADDR OF IT OPTION TABLE
 68   3EBC          3EBC          3E8C          3E8C          3E8C          ADDR OF OC OPTION TABLE
 6A   0010          0010          0010          0010          0010          KEY OF PROGRAM WITH IT SUPPORT
 6C   0000          0000          0000          0000          0000          CURRENTLY NOT ASSIGNED
 6E   0000          0020          0030          0040          0050          LTK
 70   00008090      00008087      0000807E      00008075      0000806C      SYSPARM
 74   00007118      000071D0      00007288      00007340      000073F8      JOB ACCOUNTING
 78   00003864      00003864      00003864      00003864      00003864      ADDR OF TOD COMMUNICATIONS AREA
 7C   3BD4          3BD4          3BD4          3BD4          3BD4          ADDR OF PIB EXTENSION
 7E   5A30          5A30          5A30          5A30          5A30          ADDR OF MICR DTF LABEL
 80   00000000      00000000      00000000      00000000      00000000      ADDR OF QTAM VECTOR TABLE
 84   04A0          04A0          04A0          04A0          04A0          ADDR OF BG COMREG
 86   10E0          10E0          10E0          10E0          10E0          RESERVED
 88   00000588      00000588      00000588      00000588      00000588      ADDR OF COMREG EXTENSION
 8C   COCO          COCO          COCO          COCO          COCO          RESERVED
 8E   03            00            00            00            00            DISK CONFIGURATION BYTE
                                                                                          (Ex 112KT)
```

Figure G.1 part 8 of 20

DUMP: Monitoring TIBs, QFILE Records and DATAFIL Records

The operator command whose format is shown below causes pointers contained in execution processor TIBs, in QFILE records, or in DATAFIL records to be monitored. The format of the command is:

$$\text{Z DUMP,}\left\{\begin{array}{l}\text{TIB}\\ \text{QFL}\\ \text{DFL} \quad [\text{,TIB}] \quad [\text{,DFL}]\\ \text{END}\\ \text{LST}\end{array}\right\}$$

When TIB is specified, POWER/POWER RJE monitors the first and last queue pointers for all POWER/POWER RJE queues. The TIB fields containing these pointers are:

- TIBRDRQF = pointer to the first JCT in the partition's reader queue
- TIBRDRQL = pointer to the last JCT in the partition's reader queue
- TIBPRTQF = pointer to the first JCT in the partition's print queue
- TIBPRTQL = pointer to the last JCT in the partition's print queue
- TIB PUNQF = pointers to the first JCT in the partition's punch queue
- TIBPUNQL = pointers to the last JCT in the partition's punch queue
- TIBPURGF = pointers to the first JCT in the partition's purge queue
- TIBPURGB = pointers to the last JCT in the partition's purge queue.

POWER/POWER RJE verifies that these pointers, which are in the format MCHR, are valid disk addresses within the QFILE extent.

When QFL is specified, POWER/POWER RJE monitors:

- in the QFILE master record, the first and last queue pointers for the reader, print, punch, and purge queues of all partitions
- in the JCT being processed, the forward and backward chaining pointers.

Refer to Appendix M for a description of the QFILE.

POWER/POWER RJE verifies that these pointers are valid disk addresses within the QFILE extent and that no pointer in the QFILE master record appears on more than one queue.

When DFL is specified, POWER/POWER RJE verifies that the DATAFIL record chain-pointer contains a valid disk address within the DATAFIL extent.

When END is specified, the diagnostic operations are terminated.

When LST is specified, the current status of the diagnostic operation is displayed on SYSLOG. This option does not affect the status itself.

When POWER/POWER RJE finds an invalid disk address, it causes an appropriate message to be displayed on the console.

Contents of Registers

An analysis of the register contents at the time of a dump occasionally provides a clue to the cause of the dump. Refer to Chapter 9 in Section 4 for a description of general usage for POWER and POWER RJE.

The BIM (BTAM Interface Module) Simulator

This is a software feature of the POWER RJE that simulates the function of the BTAM module. The simulator replaces the BIM for one RJE task in simulation mode, and any TPIO request from the task is serviced by the simulator on local unit record devices. All initialization procedures are performed for the task as if it were not in simulation mode. Some of these procedures are reversed to accommodate the simulator. In particular, where the dynamic assignment is made for the line, the line is unassigned to permit the simulator to use the CCB in the Line Control Block for the card reader. The dynamic assignment is made in accordance with the specified card reader in the special start command. The print and punch functions are performed on the SYSLST and SYSPCH unit record devices assigned by job control during POWER initiation. By using the BIM simulator, execution of RJE task overlays may be divorced completely from any line activity, including BTAM execution.

Appendix K

ANALYZING A DUMP OF THE POWER PARTITION

A method of locating POWER control blocks and areas is shown in a series of illustrations in Figure K.1. part 1 through 9. Notes within these illustrations provide information that will help you in determining the status of the various POWER routines at the time the dump was taken in analyzing the contents of specific areas of bytes.

The reference table below is provided to help you to locate descriptions shown in the following nine illustrations.

Block or area to be located	Reference
The POWER partition (if it was active at the time the dump was taken)	A
POWER control blocks:	
GENSW (generation option table and switch)	D
parameter list table	D R
DATABLKS and PROGBLKS table	R
DCTs (device control tables)	F
TIBTAB	C E
TIB	E G-Q
3540 control block	S
Program buffers	S
Data buffers	H

ANALYZING A DUMP OF THE POWER PARTITION

A This area (described in Section 4 of this manual, Chapter 11) is neither used nor altered by the POWER program. If POWER was active immediately before the dump was taken, the area can be located as follows:

If the MAP command was executed and is available on the SYSLOG output, the name of the job that initiated POWER will be printed together with the partition identifier and upper and lower addresses for that partition. The lower address is the address of the partition save area of the POWER partition.
(Note that the address values printed by the MAP command are in decimal.)
An alternative method is described below:

1. Locate the BOUNDARY BOX in the dump as described in this manual. (Refer to Chapter 12 Section 4.)
2. Make a note of the upper and lower partition addresses contained in the BOUNDARY BOX.
3. Locate the PIB2 in the dump as described in this manual. (Refer to Chapter 7 Section 4.)
   The PIB2 entry belonging to the partition containing POWER will have bit 0 of byte 15 set to 1 (X'80'), and the PIB2 entry belonging to partitions supported by POWER will have bit 1 of byte 15 set to 1 (X'40')
4. From this information and the contents of the BOUNDARY BOX you can locate the address of the partition save area of the partition containing POWER and of the partition supported by POWER.

Note: If POWER was assembled with a unique name for the POWER generation macro, you can look up that name in the left of the translated dump columns. Otherwise the name is FGPSPOOL.

B The last eight bytes contain the name of the operator communication transient phase that has last been executing in the area. This name has the format

FGPTYPxx

where xx = phase identifiers.

Examples of phase names: FGPTYPS0 or FGPTYPL1.

The address of the name can be calculated by adding X'400' to the address of the last byte of the partition save area. (The partition save area is either X'58' or X'78' bytes in length; refer to Section 4 in this manual, Chapter 11 for a description.) Subtract 8 from the result.

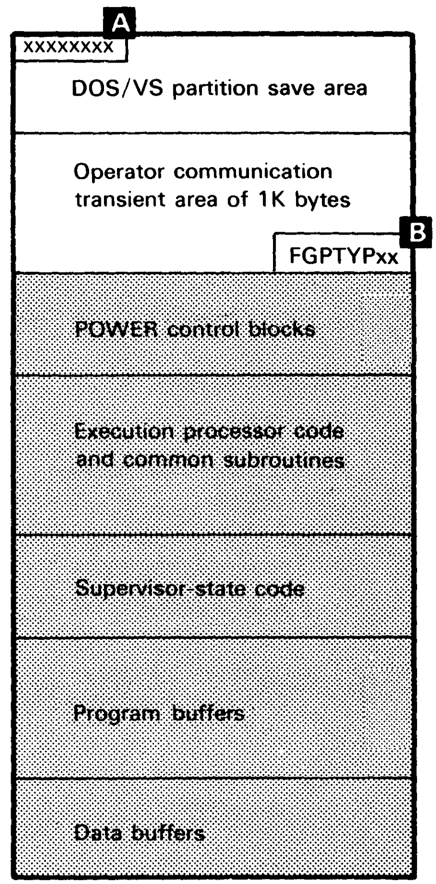
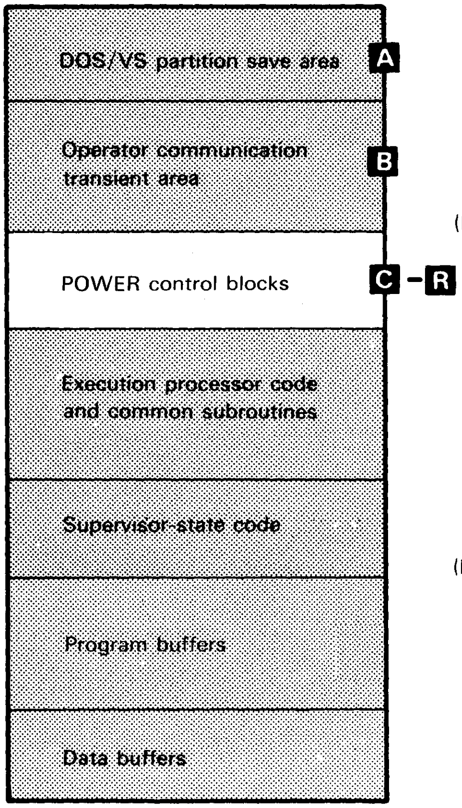


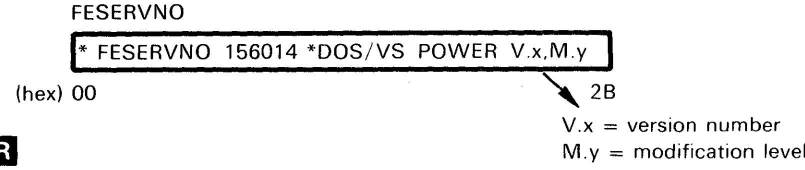
Figure K.1, part 1 of 9

Appendix K

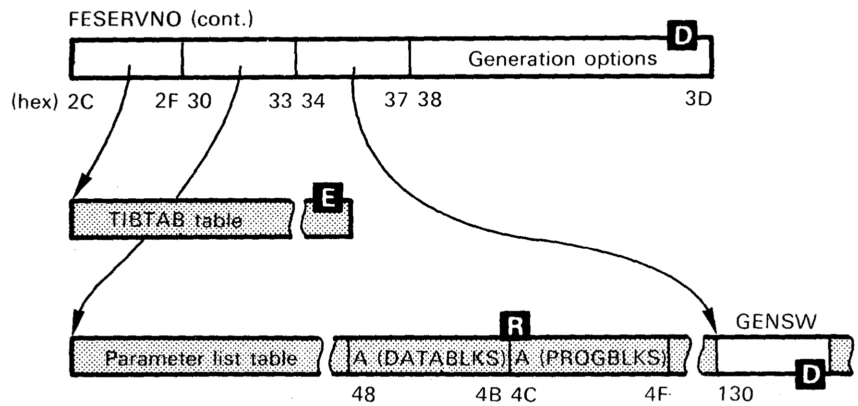
ANALYZING A DUMP OF THE POWER PARTITION



C The following helps you locate the POWER control blocks. For full details about the fields of the control blocks, see *Chapter 4. Data Areas*.



Locating this constant in the translated dump columns is a quick method of locating the beginning of the POWER control blocks area. FESERVNO (cont.)



D Analyze both the generation options and the generation option switch (GENSW) for the options specified.

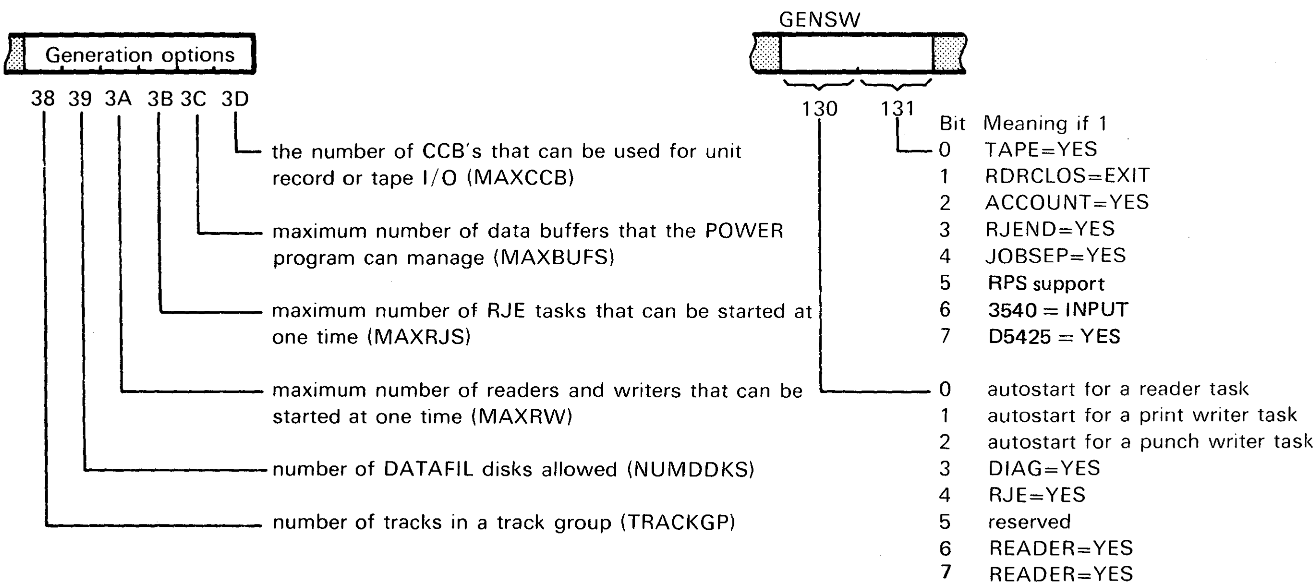


Figure K.1, part 2 of 9

J TIBSTAT of an execution processor TIB:

Bit	Meaning
Ø	1 = SYSLST output for the current job entry will not be intercepted.
1	reserved.
2	1 = SYSPCH output for the current job entry will not be intercepted.
3	1 = end-of-job processing for the current job entry takes place.
4	1 = printed output is being intercepted.
5	reserved.
6	1 = punched output is being intercepted.
7	1 = a job is being executed.

K TIBSTAT of a reader TIB:

Bit	Meaning
Ø	1 = task is going to be terminated.
1	reserved.
2	1 = Sequence of DISKETTE volumes is to be checked.
3	reserved.
4	1 = task is reading from a 3450.
5	1 = both a card reader and a 3540 are assigned.
6	reserved.
7	1 = task is to be cancelled.

L TIBSTAT of a print writer TIB:

Bit	Meaning
Ø-1	1Ø = a stop command has been issued for the task. Ø1 = a flush command for the current job has been issued for the task. 11 = a restart command for the current job has been issued for the task.
2	1 = a flush all command has been issued for the task.
3	1 = the writer has completed output for the current job.
4	1 = restart command directs the task to start from the beginning of the current job.
5	reserved.
6	1 = indicates to a tape writer that job separator pages are required.
7	1 = a cancel command has been issued for the task

Figure K.1, part 7 of 9.

M TIBSTAT of a punch writer TIB:

Bit	Meaning
0	1 = a stop command has been issued for the task.
1	1 = a flush command for the current job has been issued for the task.
2	1 = a flush all command has been issued for the task.
3	1 = the writer has completed output for the current job.
4-6	reserved.
7	1 = a cancel command has been issued for the task.

N TIBSTAT of the operator communications task TIB:

Bit	Meaning
0-6	Reserved.
7	POWER command is entered while a previous command is being executed.

O TIBACTN of an execution processor TIB:

X'00' = disk I/O was in progress or just completed if the task was in control of the CPU.

X'04' = the contents of a print or punch buffer are to be printed or punched, respectively.

X'08' = a POWER EOJ card is to be processed.

X'0C' = an input card is to be read.

X'10' = a PRT card is to be processed.

X'14' = a PUN card is to be processed.

X'18' = a message is being written on the console.

X'1C' = either a JOB card (writer-only system) or an SLI card (reader system) is to be /processed.

X'20' = a card from the Source Statement library is to be processed.

P TIBACTN of a reader or writer TIB:

X'00' = disk I/O was in progress or just completed if the task was in control of the CPU.

X'04' = the task waits for work. This code is set by the execution processor for the same partition:
- for a reader task when all card input for a job is complete
- for a writer task when all print (or punch) output for a job is complete.

X'08' = The task waits for completion of
- a read operation on the associated card reader if a reader task.
- a print operation on the associated printer if a print writer task.
- a punch operation on the associated card punch if a punch writer task.

X'18' — The task has issued a message to the console and waits for completion of the typing operation.

Q TIBACTN of the operator communication TIB:

X'00' = disk I/O was in progress or just completed if the task was in control of the CPU.

X'04' = The task is inactive, that is, all commands have been processed.

X'08' = the task has received a command and has initiated processing of same.

X'10' = I/O for the ACCTFIL is in progress.

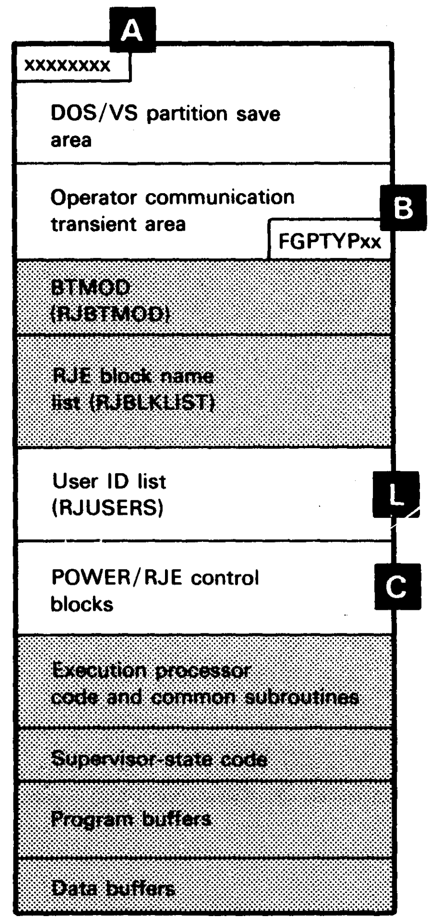
X'18' = the task has initialized writing a message on the console and waits for completion of the typing operation.

Note: For POWER start up, the TIBACTN code has been assembled as X'08' and the TSB of the associated TIBTAB entry has been assembled at X'03'.

Figure K.1, part 8 of 9.

ANALYZING A DUMP OF THE POWER RJE PARTITION

A If the POWER/RJE program was assembled with a unique name for the generation macro you can look up that name in the left of the translated dump columns.



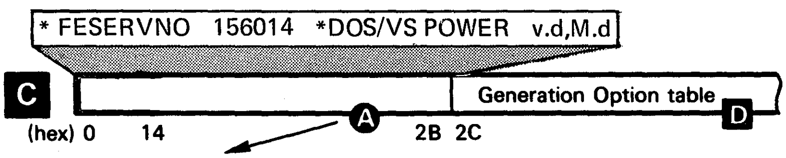
B A 1K area used for the execution of the operator communication transient phases. The last eight bytes contain the name of the phase that has been executing in the area. This name has the format

FGPTYPxx

where xx = phase identifiers

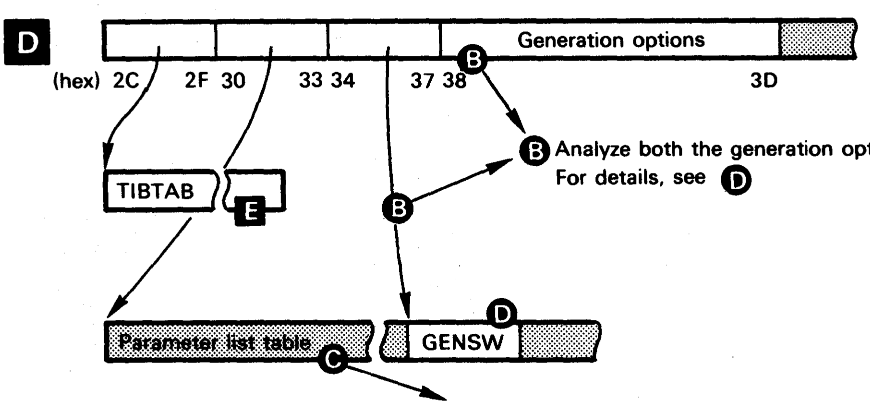
The address of the name is:

A (partition save area) + { X'78' / X'58' } + X'400'-8



A V.d. = program version number
M.d. = program modification number

Locating this constant in the translated dump columns is a quick method of locating the beginning of the POWER/RJE control blocks area.



B Analyze both the generation option switch and the options in the table. For details, see **D**

C For full details about the fields of the table, see the "Data Area" section. For important switches, see **D**

Figure L.1, part 1 of 14

Appendix L

ANALYZING A DUMP OF THE POWER RJE PARTITION

D continued

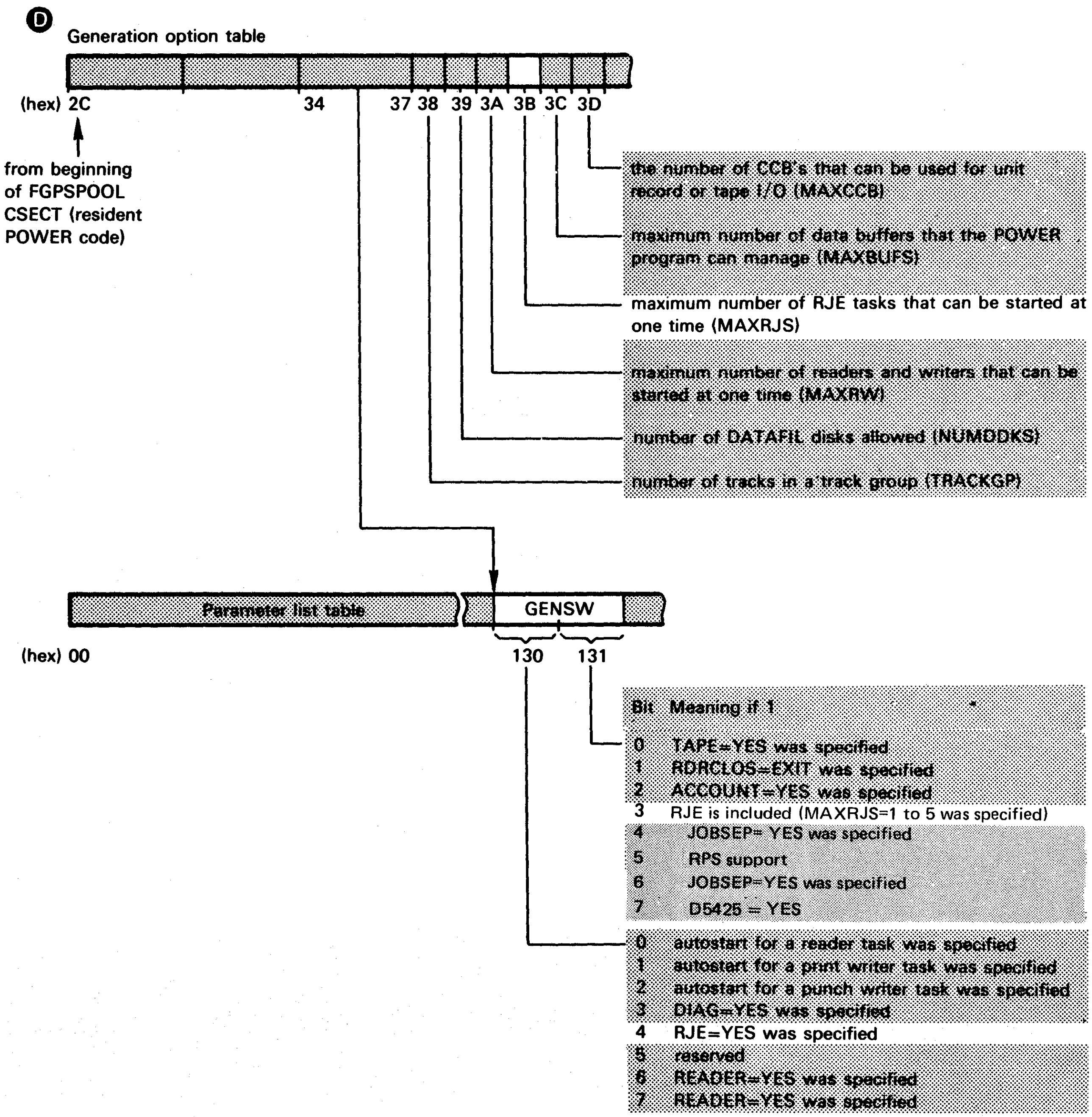


Figure L.1, part 2 of 14

GLOSSARY

This glossary contains technical terms associated with the subject of this publication. A more general range of terms is contained in *IBM Date Processing Glossary, GC20-1699.*

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard Vocabulary for information Processing (copyright © 1970 by American National Standards Institute, Incorporated) which was prepared by subcommittee X3K5 on Terminology and Glossary of American National Standards Committee X3. These definitions are indicated by an asterisk.

A

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

asynchronous. without regular time relationship.

auxiliary storage. Data storage other than real storage; for example, storage on magnetic tape or disk. Synonymous with external storage, secondary storage.

B

basic control mode. When PSW bit 12 is 0, PSW format and system operation are compatible with standard System/360 operation. This is the basic control mode in which control registers 0, 8, and 14 are available to the system. Abbreviated to BC mode. See also "Extended Control Mode."

BTAM (basic telecommunications access method). A basic access method that permits a READ/WRITE communication with remote devices.

buffer. (1) A storage device in which data is assembled temporarily during data transfer. (2) During I/O operations, a portion of real storage from which data is read or into which data is written.

C

channel program. One or more Channel Command Words (CCWs) that control(s) a specific sequence of channel operations. Execution of the specific sequence is initiated by a single start I/O instruction.

channel program translation. In a channel program, replacement, by software, of virtual addresses with real addresses.

command control block (CCB). A 16-byte field required for each channel program executed by physical IOCS. This field is used for communication between physical IOCS and the problem program.

communication region. An area of the supervisor set aside for interprogram and intraprogram communication. It contains information useful to both the supervisor and the problem program. Abbreviated comreg. (Not to be confused with the COMRG macro instruction).

control program. A program that is designed to schedule and supervise the performance of data processing work by a computing system.

control registers. A set of registers used for operating system control of relocation, priority interruption, program event recording, error recovery, and masking operations.

core-wrap mode. The method of operation that records the events of a trace in the PD area or an alternate area (used by PDAIDS). It is the default process when no output device for a PDAID trace has been specified.

D

DTF (define the file) macro instruction. A macro instruction that describes the characteristics of a logical input/output file, indicates the type of processing for the file, and specifies the I/O areas and routines to process the file.

default value. The choice among exclusive alternatives made by the system when no explicit choice is specified by the user. A default value is indicated by underlining in tables listing parameters.

diskette. A flexible magnetic oxide coated disk, permanently enclosed in a semi-rigid protective plastic jacket approx, 8 inches square. During data processing operations the disk turns freely within the jacket. It is capable of storing 1898 128 character data records.

dump. (1) To print out the contents of all or part of virtual storage or of auxiliary storage (2) The data resulting from the process as in (1).

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

E

emulator (1) * A device or computer program that emulates. (2) The combination of programming techniques and special machine features that permits a given computing system to execute programs written for another system.

GLOSSARY

environmental recording, editing, and printing (EREP). A program that processes the data contained on the system recorder file.

error recovery procedures. Procedures designed to help isolate, and, when possible, to recover from hardware errors in equipment. The procedures are often used in conjunction with programs that record the statistics of machine malfunctions.

error volume analysis (EVA). With this option, the system issues a message to the operator when a number of temporary read or write errors (specified by the user at system generation time) has been exceeded on a currently accessed tape file.

event. An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as input/output.

extent. The physical locations on Input/Output devices occupied by or reserved for a particular volume.

extended control mode. When PSW bit 12 is set to 1, the PSW format is changed from that used for standard System/360 operation: the channel mask bits, instruction length code, and interruption code are removed, and additional mode and mask bits are included. This is the extended control mode, in which all control registers are available to the system for control of facilities that are particular to System/370. Abbreviated to EC mode. See also "Basic Control Mode."

F

fetch. (1) To bring a program phase into real storage from a core image library or from the page data set for immediate execution. (2) The routine that retrieves requested phases and loads them. (3) The name of a macro instruction (FETCH) used to transfer control to the system loader. (4) To transfer control to the system loader.

* **file.** A collection of related records treated as a unit. For example, one line of an invoice may form an item, a complete invoice may form a record, the complete set of such records may form a file, the collection of inventory control files may form a library, and the libraries used by an organization are known as its data bank.

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

F/L Trace (Fetch/Load Trace). A program that records information about phases and transients as they are called from a core image library.

G

GSVC Trace (Generalized Supervisor Calls Trace). A program that records SVC interrupts as they occur. All or a selected group of SVCs can be traced.

H

hard copy. A printed copy of machine output in a visually readable form, for example, a printed recording of the messages displayed on the System/370 Model 125 video display unit.

hard stop. A condition, usually caused by an error, in which the CPU is stopped and is not executing the microprogram.

* **hardware.** Physical equipment, as opposed to the computer program or method of use, for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

I

Input Job Stream. A sequence of job control statements entering the system, which may also include input data.

* **interface.** A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

interrupt. A break in the normal sequence of instruction execution. It causes an automatic transfer to a preset storage location where appropriate action is taken.

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

I/O area. An area (portion) of real storage into which data is read or from which data is written, the term buffer is often used in place of I/O area.

I/O Trace (Input/Output Trace). A program that records I/O device activity for all or a selected group of I/O devices.

IOCS (input/output control system). A group of macro instruction routines provided by IBM for handling the transfer of data between main storage and external storage devices.

irrecoverable error. A hardware error which cannot be recovered from by the normal hardware and retry procedures.

J

job. (1) * A specified group of tasks prescribed as a unit of work for a computer. By extension, a job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) A collection of related problem programs, identified in the input stream by a JOB statement followed by one or more EXEC statements.

relocatable library. A library of relocatable object modules and IOCS modules required by various compilers. It allows the user to keep frequently used modules available for combination with other modules without recompilation.

resource. Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the central processing unit, data files, and control and processing programs.

Rotational Position Sensing. A standard feature of IBM 3330/3333 and an optional feature of IBM 3340 disk storage devices. It permits a device to disconnect from a block multiplexer channel (or its equivalent on Model 3115/3125 CPUs) during rotational positioning opera operations, thereby allowing the channel to service other devices on the channel during the positioning delay.

* **routine.** An ordered set of instructions that may have some general or frequent use.

S

Second Level Directory (SLD). The table, located in the Supervisor and containing the highest phase-names found on the corresponding directory tracks of the system core image.

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

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

segment translation exception. A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set. See also page translation exception, translation specification exception.

self-relocating. A programmed routine that is loaded at any doubleword boundary and can adjust its address values so as to be executed at that location.

self-relocating program. A program that is able to run in any area of storage by having an initialization routine to modify all address constants at object time.

selector channel. A channel designed to operate with only one I/O device at a time. Once the I/O device is selected, a complete record is transferred one byte at a time.

SEREP. A stand-alone environment recording, editing, and printing program that makes the data contained in an error logout area of real storage available for further analysis.

Shared Virtual Area (SVA): The last part of the virtual system address space that contains phases which are reenterable and relocatable and which can be shared between partitions.

soft stop. A condition in which the CPU has stopped processing but continues to handle any requested interruptions.

stand-alone dump. A program that displays the contents of the registers and all of real storage and that runs independently and is not controlled by DOS/VS.

subtask. A task in which control is initiated by a main task by means of a macro instruction that attaches it.

* **storage protection.** An arrangement for preventing access to storage for either reading, or writing, or both.

system generation. The process of tailoring the IBM supplied operating system to user requirements.

system debugging aids. A set of routines provided to trace specific program events by using the program event recording facilities. Abbreviated SDAIDS.

System Directory List (SDL). A list of highly used phases (either only in the system CIL or also in the SVA). This list is placed in the SVA.

system recorder file. The data file that is used to record hardware reliability data.

T

task. A unit of work for the central processing unit from the standpoint of the control program.

task selection. The supervisor mechanism for determining which program should gain control of CPU processing.

teleprocessing. The processing of data that is received from or sent to remote locations by way of telecommunication lines.

terminal. (1) * A point in a system or communication network at which data can either enter or leave. (2) Any device capable of sending and receiving information over a communication channel.

Terminating partition. This is a partition owning a program which is in the process of being terminated either because of a program cancel condition or because of EOJ.

trace. (1) To record a series of events as they occur. (2) The record of a series of events.

* **tracing routine.** A routine that provides a historical record of specified events in the execution of a program.

GLOSSARY

track hold. A function for protecting DASD tracks that are currently being processed. When track hold is specified in the DTF, a track that is being modified by a task in one partition cannot be concurrently accessed by a task or subtask in another partition.

transient area. An area in the supervisor used for temporary storage of transient routines, such as non-resident supervisor call or error-handling routines.

transient routines. These self-relocating routines are permanently stored on the system residence device and loaded (by the supervisor) into the transient area when needed for execution.

translation specification exception. A program interruption that occurs when a page table entry, segment table entry, or the control register pointing to the segment table contains information in an invalid format. See also page translation exception, segment translation exception.

U

user program. see problem program.

unrecoverable error. see irrecoverable error.

utility program. A program designed to perform a routine task, such as transcribing data from one storage device to another.

V

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

virtual address area. In DOS/VS, the area of virtual storage whose addresses are greater than the highest address of the real address area.

virtual mode. In DOS/VS, the mode of a program which may be paged.

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

virtual storage access method (VSAM). VSAM is an access method for direct or sequential processing of fixed and variable length records on direct access devices. The records in a VSAM file can be organized either in logical sequence by a key field (key sequence) or in the physical sequence in which they are written on the file (entry-sequence). A key sequenced file has an index, an entry-sequenced file does not.

volume. (1) That portion of a single unit of storage media which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and dismounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

VSAM access method services. A multifunction utility program that defines VSAM files and allocates space for them, converts indexed sequential files to key-sequenced files with indexes, facilitates data portability between operating systems, creates backup copies of files and indexes, helps make inaccessible files accessible, and lists file and catalog entries.

DOS/VS Serviceability Aids and Debugging Procedures

GC33-5380-1

READER'S COMMENT FORM

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

CUT ALONG THIS LINE

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 U

Fold

Fold

IBM

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

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