SY33-8554-1
File No. S370-37

Systems

DOS/VS System Serviceability Aids Logic

Program Numbers 5745-SC-PDA
5745-SC-UTL
5745-SC-TPE
5745-SC-ERP

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. A change to the text or to an illustration is indicated by a vertical bar to the left of the change. 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 edition documents DOS/VS support of the following new systems and devices:

IBM System/370 Model 115
IBM System/370 Model 155-II
IBM System/370 Model 158
IBM 3203/5203
IBM 3340
IBM 3540

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, Netherlands. Comments become the property of IBM.

PREFACE

This PLM (Program Logic Manual) is a detailed guide to the serviceability aid programs provided with the DOS/VS system, supplementing various supervisor and transient functions. It documents internal logic and data flow through descriptive text, flowcharts, and figures.

For the complete logic of the DOS control and service operations, this manual is to be used with the following six companion PLMs.

- DOS/VS Error Recovery and Recording Transients, SY33-8552
- DOS/VS IPL and Job Control, SY33-8555
- DOS/VS Librarian, SY33-8557
- DOS/VS Linkage Editor, SY33-8556
- DOS/VS Supervisor Logic, SY33-8551
- DOS/VS Logical Transients, SY33-8553.

Prerequisite to the effective use of the seven PLMs are the following publications.

- IBM System/370 Principles of Operation, GA22-7000
- Introduction to DOS/VS, GC33-5370
- DOS/VS System Management Guide, GC33-5371
- IBM System/360 Disk and Tape Operating Systems, Assembler Language, GC24-3414.

Publications related in subject matter to the seven system control PLMs are:

- DOS/VS Serviceability Aids and Debugging Procedures, GC33-5380
- DOS/VS System Generation, GC33-5377
- DOS/VS Messages, GC33-5379
- DOS/VS Supervisor and I/O Macros, GC33-5373
- DOS/VS Operating Procedures, GC33-5378
- DOS/VS Data Management Guide, GC33-5372.

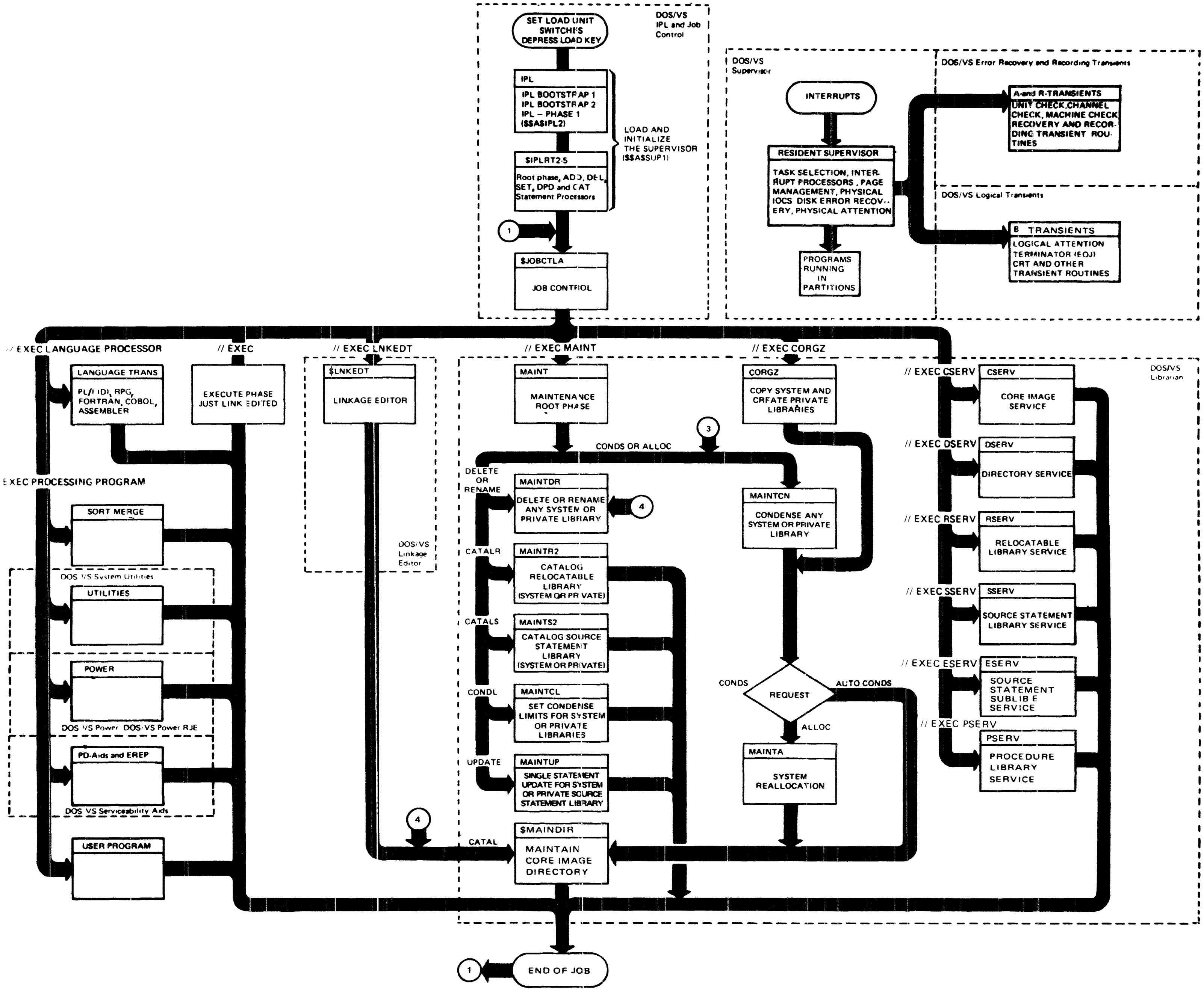
Titles and abstracts of other related publications are listed in the IBM System/360 and System/370 Bibliography, GA22-6822.

This manual consists of nine major sections:

- PDAIDS (Problem Determination Aids)
- SDAID (System Debugging Aid)
- DUMPGEN (Dump Generator)
- PDSDM (Page Data Set Dump)
- LSERV (Label Information Cylinder Display)
- ESTVUT (Error Statistics by Tape Volume Utility)
- EREP (Environmental Recording, Editing, and Printing)
- Printer support programs
- Appendixes containing various cross-reference lists, sample printouts, and supervisor tables.

The flowchart symbols used in this manual conform to the flowcharting standards of the American National Standards Institute, Inc. Numerals, such as C0, identify the program or general level flowcharts. The detailed flowcharts are identified by letters AA through ZZ. See Appendix D.

FIGURES


SET LOAD UNIT SWITCHES DEPRESS LOAD KEY
DOS/VS IPL and Job Control
IPL
IPL BOOTSTRAP 1
IPL BOOTSTRAP 2
IPL – PHASE 1 ($$A$IPL2)
$IPLRT2-5
Root phase, ADD, DEL, SET, DPD and CAT Statement Processors
LOAD AND INITIALIZE THE SUPERVISOR ($$A$SUP1)
1
$JOBCTLA
JOB CONTROL
DOS/VS Supervisor
INTERRUPTS
RESIDENT SUPERVISOR
TASK SELECTION, INTERRUPT PROCESSORS, PAGE MANAGEMENT, PHYSICAL IOCS DISK ERROR RECOVERY, PHYSICAL ATTENTION
PROGRAMS RUNNING IN PARTITIONS
DOS/VS Error Recovery and Recording Transients
A-and R-TRANSIENTS
UNIT CHECK, CHANNEL CHECK, MACHINE CHECK RECOVERY AND RECORDING TRANSIENT ROUTINES
DOS/VS Logical Transients
B TRANSIENTS
LOGICAL ATTENTION TERMINATOR (EOJ) CRT AND OTHER TRANSIENT ROUTINES
// EXEC LANGUAGE PROCESSOR
LANGUAGE TRANS
PL/I (D), RPG, FORTRAN, COBOL, ASSEMBLER
// EXEC
EXECUTE PHASE JUST LINK EDITED
// EXEC LNKEDT
$LNKEDT
LINKAGE EDITOR
DOS/VS Linkage Editor
// EXEC MAINT
MAINT
MAINTENANCE ROOT PHASE
// EXEC CORGZ
CORGZ
COPY SYSTEM AND CREATE PRIVATE LIBRARIES
DOS/VS Librarian
3
CONDS OR ALLOC
DELETE OR RENAME
MAINTDR
DELETE OR RENAME ANY SYSTEM OR PRIVATE LIBRARY
4
CATALR
MAINTR2
CATALOG RELOCATABLE LIBRARY (SYSTEM OR PRIVATE)
CATALS
MAINTS2
CATALOG SOURCE STATEMENT LIBRARY (SYSTEM OR PRIVATE)
CONDL
MAINTCL
SET CONDENSE LIMITS FOR SYSTEM OR PRIVATE LIBRARIES
UPDATE
MAINTUP
SINGLE STATEMENT UPDATE FOR SYSTEM OR PRIVATE SOURCE STATEMENT LIBRARY
CATAL
$MAINDIR
MAINTAIN CORE IMAGE DIRECTORY
MAINTCN
CONDENSE ANY SYSTEM OR PRIVATE LIBRARY
CONDS
REQUEST
AUTO CONDS
ALLOC
MAINTA
SYSTEM REALLOCATION
// EXEC CSERV
CSERV
CORE IMAGE SERVICE
// EXEC DSERV
DSERV
DIRECTORY SERVICE
// EXEC RSERV
RSERV
RELOCATABLE LIBRARY SERVICE
// EXEC SSERV
SSERV
SOURCE STATEMENT LIBRARY SERVICE
// EXEC ESERV
ESERV
SOURCE STATEMENT SUBLIBE SERVICE
// EXEC PSERV
PSERV
PROCEDURE LIBRARY SERVICE
EXEC PROCESSING PROGRAM
SORT MERGE
DOS/VS System Utilities
UTILITIES
POWER
DOS/VS Power DOS/VS Power RJE
PD-Aids and EREP
DOS/VS Serviceability Aids
USER PROGRAM
END OF JOB

INTRODUCTION

PDAIDs (Problem Determination Aids) provide the option to trace specified events during the operation of a program. Tracing may record the events of the problem program, the supervisor, or both. PDAIDs also provide the possibility to dump transient areas.

INPUT/OUTPUT TRACE: The I/O (Input/Output) trace records I/O device activity. The data recorded by the I/O trace function may be for all or a selected group of I/O devices. When an I/O interrupt occurs, the data recorded is:

- I/O old PSW
- CSW.

When an SIO instruction is issued by the DOS/VS supervisor, the data recorded is:

- Condition code
- Device address
- CCB address
- CSW.

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. When a fetch or a load is issued that causes an SVC 1, 2, or 4, the data recorded is:

- Phase or transient name
- SVC number
- Location of the SVC
- Partition identifier
- Load address of the phase
- Entry address of the phase.

At times, SVCs 5, 6, 11, and 14 branch directly into the supervisor fetch or load routine. The fetch or load (SVCs 1, 2, and 4) is recorded; however, the calling address and the SVC values for SVCs 5, 6, 11, and 14 are not indicated in the actual fetch or load trace record.

GENERALIZED SUPERVISOR CALL TRACE: The GSVC (Generalized Supervisor Call) trace records SVC interrupts as they occur. All SVCs, or a selected group of SVCs, may be traced. The data recorded by the GSVC trace is:

- SVC old PSW
- Task identification
- Contents of register 0
- Last three bytes of register 1
- SVC number
- Partition identifier.

If PTO is included in the system, the SVCs issued when the physical transient area is busy are not traced.

QTAM TRACE: The QTAM trace records the input/output activity of QTAM in three areas:

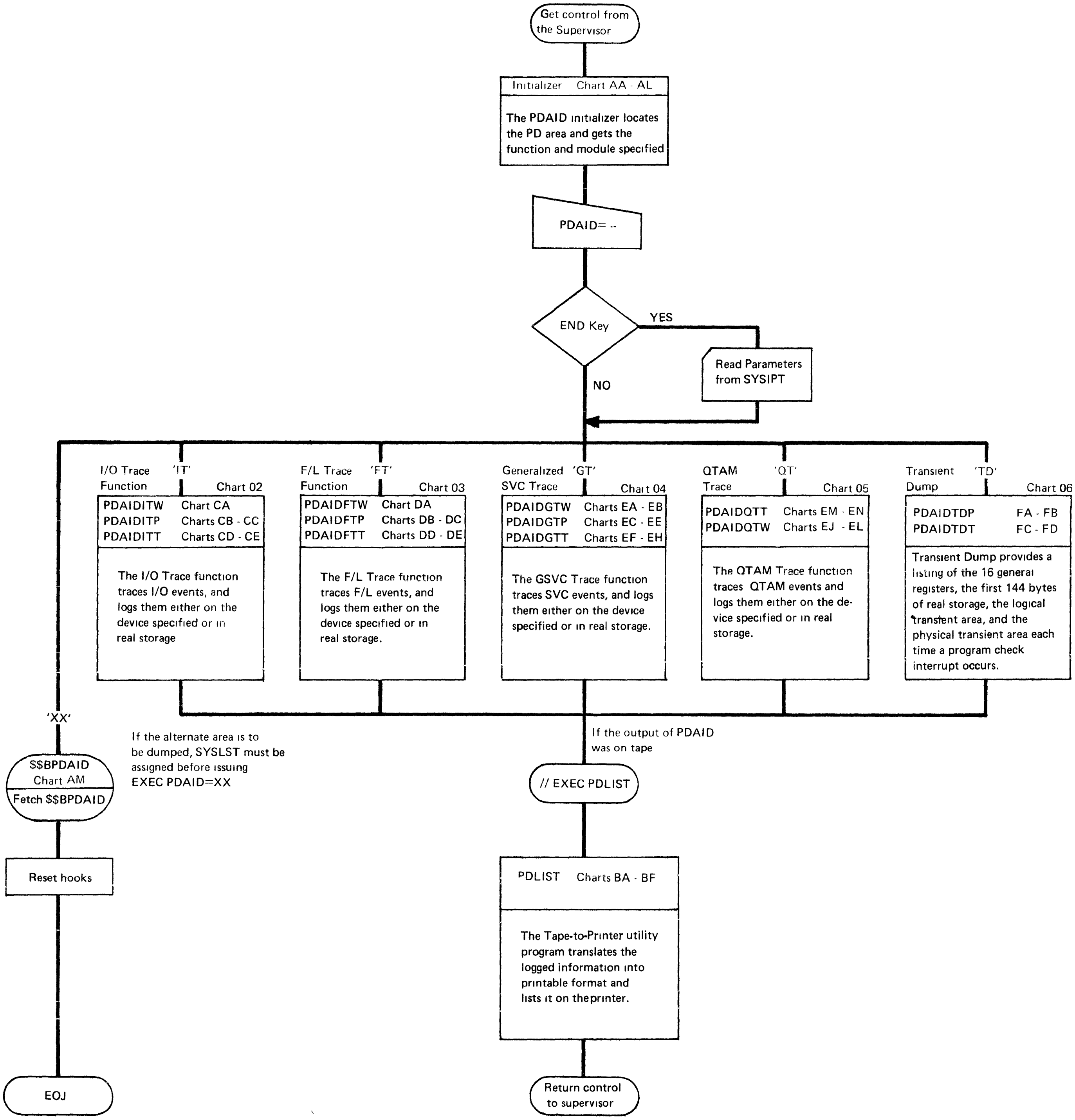
- SVCs 0 and 31
- A supervisor-issued SIO
- An I/O interrupt.

The I/O old PSW, the CSW, and the device address are recorded when an I/O interrupt occurs. The condition code, device address, CCB address, and CSW are recorded when a supervisor issues an SIO. The SVC old PSW, the SVC number, the contents of registers 0 and 1, and the partition identifier are recorded when an SVC 0 or 31 is issued.

TRANSIENT DUMP: The transient dump allows certain areas of the supervisor to be displayed when a program check occurs. These areas are:

- The 16 general registers used at the time of failure
- The first 144 bytes of main storage
- The physical transient area
- The logical transient area.

Chart 01. General Logic Flow of the PDAID Programs



System Consideration for PDAIDs

If the use of PDAIDs is desired, then the following must be performed prior to their execution:

- During system generation, specify PD=YES or PD≥1400 in the FOPT macro of the installation tailored supervisor. (See Figure 55 in Appendix B for supervisor storage allocations.)

  Note: Up to 10,240 may be specified to increase the size of the PD area for the core-wrap mode.

- Catalog the main phase (PDAID) to the core image library prior to its execution.

All the PDAID programs are self-relocating, and can be run in any partition. If data provided by PDAIDs is recorded on magnetic tape, the PDLIST program must be used to obtain a listing. See the DOS/VS System Generation manual listed in the Preface for supervisor specifications and cataloging procedures.

Initiation of PDAID

You can execute PDAIDs by using standard DOS/VS job control language from either SYSLOG or SYSIPT. The statement

// EXEC PDAID

causes the main phase, PDAID, to load at the address of the initiating partition. Control is given to PDAID, which issues the following message to the operator 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 transient dump. See Note 1.

XX Terminates the PDAID presently running. See Note 2.

END Indicates PDAID control statements are entered through SYSIPT.

Note 1: When IT, FT, GT, QT, or TD is specified, the operator must give additional PDAID control statements through SYSLOG.

Note 2: If a dump of the alternate area is desired, SYSLST must be assigned before XX is specified.

Figure 2 illustrates the PDAID control statements in the sequence in which they must be used.

- The END response is valid only for SYSLOG and cannot be used as a SYSIPT operand.

- 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.

- GO terminates the PDAID control input, and the default is taken for any PDAID options that are not specified. When you use SYSLOG, GO is a valid response (see Figure 2). When you use SYSIPT, GO should be the last parameter, and it has no operand associated with it.

Selection of an Output Device: 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).

Selection of Core-Wrap Mode: If an output device is not specified, core-wrap mode is assumed. The event trace tables (see Figures 3, 6, 7, and 12) are kept in the PD area. The number of events contained in this area depends on the size of that area generated at system generation time with the PD option of the FOPT macro. PD=1400 (or PD=YES) is the minimum that can be selected. Figure 1 shows the maximum number of entries for each trace type when PD=1400 (minimum PD area), and when PD=10,240 (maximum PD area).

If core-wrap mode is selected, an alternate area, which belongs to the supervisor, but resides in the main page pool, can be used for the trace tables (see Figures 3, 6, 7, and 12). The size of the alternate area is specified through the message/parameter AAA=nK. AAA= and OUTPUT DEVICE= are mutually exclusive; when one is specified, the other cannot be used. If the alternate save area is specified, the PD save area is not used. Neither can the PD save area be used when PDAID and SDAID are used concurrently.

Neither core wrap nor AAA are valid for transient dump.

Trace Type	Length of Entry	Maximum Number of Entries when PD = 1400	Maximum Number of Entries when PD = 10,240
I/O	18	42	544
F/L	22	42	453
GSVC	18	50	552
QTAM	21	35	465

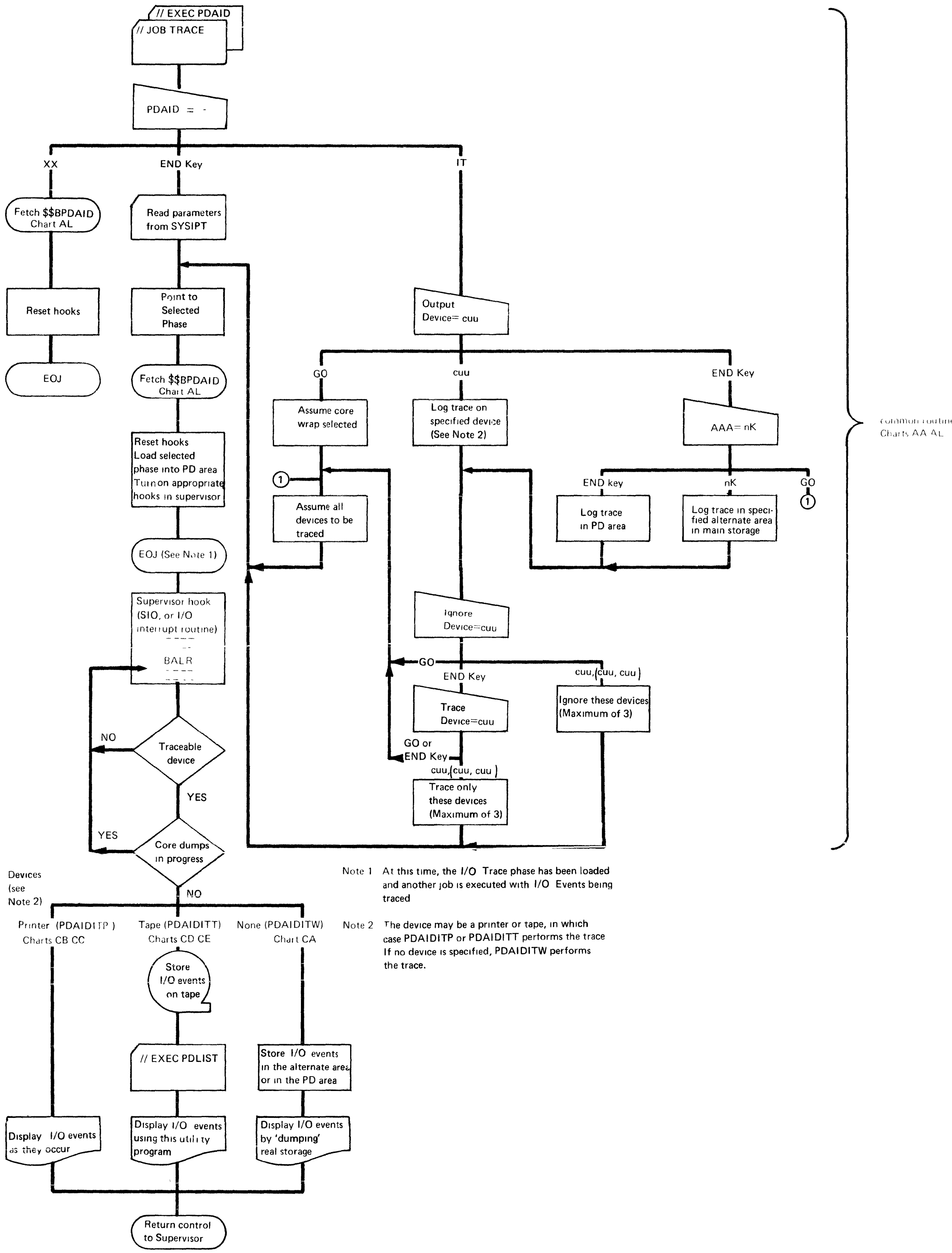
Figure 1. Maximum Number of PD Area Entries

SYSLOG Message / SYSIPT Parameter	SYSLOG Response / SYSIPT Operand	Meaning	Default
PDAID =	{ FT GT IT QT TD XX END Key }	FT Fetch/Load Trace GT GSVC Trace IT I/O Trace QT QTAM Trace TD Transient dump XX Terminate present PDAID function END Key Additional PDAID control input through SYSIPT	None, the function continues.
OUTPUT DEVICE = (Note 1)	{ cuu X'cuu' END Key GO }	Specify the hexadecimal channel and unit number of either a magnetic tape unit or a printer for the output device of the PDAID. A printer is invalid for QTAM Trace. A device must be specified for transient dump.	Core - Wrap mode; none for transient dump
AAA= (Notes 1, 3, and 4)	[{ nK END Key GO }]	Specify the size of an alternate area for Core - Wrap Mode. Minimum is 2K, maximum 99K.	Core - Wrap Mode using PD Save Area.
TRACE PARTITION= (Valid only for Fetch/Load, GSVC and QTAM trace)	[{ SP BG F4 F3 F2 F1 END Key GO }]	SP Supervisor BG Background F4 Foreground 4 F3 Foreground 3 F2 Foreground 2 F1 Foreground 1 For the QTAM trace only SVCs 0 and 31 are recorded per partition.	Trace all partitions and the supervisor.
IGNORE DEVICE= (Notes 2 and 3)	[{ cuu X'cuu' END Key GO }]	Specify the hexadecimal channel and unit number of the device to be ignored by the I/O or QTAM trace. A maximum of 3 may be specified.	Trace all devices.
TRACE DEVICE= (Notes 2 and 3)	[{ cuu X'cuu' END Key GO }]	Specify the hexadecimal channel and unit number of the device to be traced by the I/O or QTAM trace. A maximum of 3 may be specified.	Trace all devices.
IGNORE SVC= (Notes 2 and 3)	[{ nn END Key GO }]	Specify the hexadecimal SVC number to be ignored by the GSVC trace. A maximum of 6 may be specified.	Trace all SVC s.
TRACE SVC= (Notes 2 and 3)	[{ nn END Key GO }]	Specify the hexadecimal SVC number to be traced by the GSVC trace. A maximum of 6 may be specified.	Trace all SVC s.
GO (Valid SYSIPT parameter)	GO (Valid SYSLOG response)	GO terminates the PDAID control input and uses the default for those options that are not specified.	None.

Note 1. OUTPUT DEVICE and AAA are mutually exclusive. Neither can be shared with any other program.
Note 2. TRACE and IGNORE are mutually exclusive.
Note 3. Not valid for transient dump.
Note 4. When PDAID is used concurrently with SDAID the alternate PD save area cannot be used.

Figure 2. PDAID Control Statements

Chart 02. Logic Flow of the I/C Trace Functicn



Note 1 At this time, the I/O Trace phase has been loaded and another job is executed with I/O Events being traced

Note 2 The device may be a printer or tape, in which case PDAIDITP or PDAIDITT performs the trace If no device is specified, PDAIDITW performs the trace.

I/O TRACE FUNCTION

The I/O trace function (Chart 02) provides trace tables for input/output devices. (See Figures 3 and 4.)

I/O trace allows the I/O activity of programs run under DOS/VS to be traced. Tracing consists of:

- Recording the device address, the I/O old PSW, and the CSW when an I/O interruption occurs and
- Recording the SIO prefix, the condition code, the device address, the CCB address, and the CSW (when the CSW is stored in response to an SIO instruction issued by the DOS/VS supervisor).

Either of these is referred to as an _I/O event_. The events may be preserved in a rotating buffer (see Figure 6) in storage (first entry overwritten when the area is full, etc.), printed, or recorded on tape. When a tape is used, it must be processed by the PDLIST utility program to provide readable output data.

If the program is running in virtual mode, the real CCB and CSW addresses are translated to virtual addresses before printing.

Sample for I/O interrupt:

Device address	I/O old PSW	CSW
2	8	8

Sample trace for a supervisor issued SIO:

SIO bit		Dev. Addr.	CCB Address	CSW
2	2	2	4	8

You can detect the SIO issued by the supervisor (CSW stored condition) by checking the PSW portion of an entry in the storage dump and determining if the system mask byte contains an X'00'. If the system mask byte is not X'00', the I/O event was an interrupt.

Figure 3. I/O Trace Table Entry

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 (see Figure 2) are specified by the initializer keywords IGNORE DEVICE= or TRACE DEVICE=. All I/O activity is traced if one of these option keywords is not specified. The two keywords are mutually exclusive. When either is specified, the other becomes invalid.

The three 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.

Data Collection

I/O trace resides in the PD area and performs the actual tracing of I/O events. The first entry to the phase causes some initialization to occur before the I/O event is acted upon. At each entry, the phase tests the logical transient area for a dump transient; normally, it does not trace any I/O activity when a dump is in progress. (If it is necessary to trace I/O events during a dump routine, the exit branch following the compare instruction labeled DUMPCHK should be altered to a NOP.) This prevents a dump either from overflowing the trace table when core wrap is used (see Figure 6), or from causing excess output in output mode.

If no dump is in progress, the device address is matched against either IGN or TRC entries to determine if the event should be entered into the trace tables.

Note: It is not necessary to ignore the PD output device. I/O events from this device are handled internally by the module. Events from the PD output device are traced if the I/O activity originates outside the PD module (that is, the device is being shared).

If the event is not to be traced, control is returned to the supervisor routine from which the trace was entered. If the event involves a device to be traced, an entry is made in the trace table. At this point, the core-wrap returns to the supervisor.

Output

The I/O output phase tests for a full table before returning to the supervisor, and attempts to output the table when it contains enough entries. If the output device cannot be accessed, control is returned to the supervisor, and output is retried at each subsequent entry to the trace phase. Limited overflow buffers are available in each module:

Type Output	Full Table	Overflow Entries	Maximum Capacity
Printer (PDAIDITP)	3 Entries	5 Entries	8 Entries
Tape (PDAIDITT)	13 Entries	7 Entries	20 Entries

When the I/O output device must share a selector channel or a control unit, the overflow capacity can be exceeded, and I/O events can be lost. The trace phase tests for such losses, and indicates such with an * when they occur. (It is recommended that output options be selected to avoid such losses, and that one or more of the most active devices be ignored if an overflow occurs.) Figure 4 illustrates output after it has been printed.

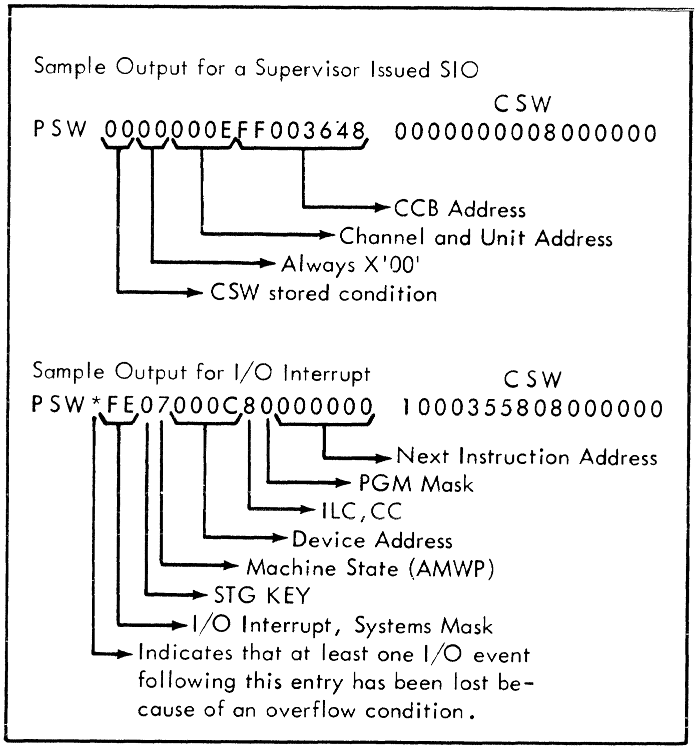


Figure 4. Sample Output for I/O Trace

The maximum number of entries which can be stored in the PD area is shown in Figure 1. When the area is full, the oldest entry is overlaid by each new entry. (Figure 6 illustrates the method of updating the trace table for core-wrap mode).

PDAIDITW (CORE-WRAP MODE)

PDAIDITW preserves a fixed number of I/O events in the PD save area, or in the alternate save area of the supervisor (when AAA= is specified).

Since PDAIDITW executes in the PD area, the actual save area - when the PD save area is used - is equal to the PD area (generated during system generation) minus the length of PDAIDITW.

To retrieve the trace tables, the PD area must be dumped. This can be done by using the PDAREA operand of the DUMP command. Of course, any dump that includes the supervisor will also include the PD area.

If the alternative area is used, it is dumped automatically on the device assigned to SYSLST, when the PDAID program is terminated.

PDAIDITW sets up the following pointers (Figure 6).

- SLOT1 -- address of the beginning of the save area.
- 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.)
- WRAPADR -- address of the end of the save area.

Trace Type	Address of SLOT1 is at
Input/output	PDAREA+X'194'
Fetch/load	PDAREA+X'E6'
GSVC	PDAREA+X'114'
QTAM	PDAREA+X'1BC'

Figure 5. Trace Entry Locations in the PD Save Area

PDAIDITP (PRINTER OUTPUT)

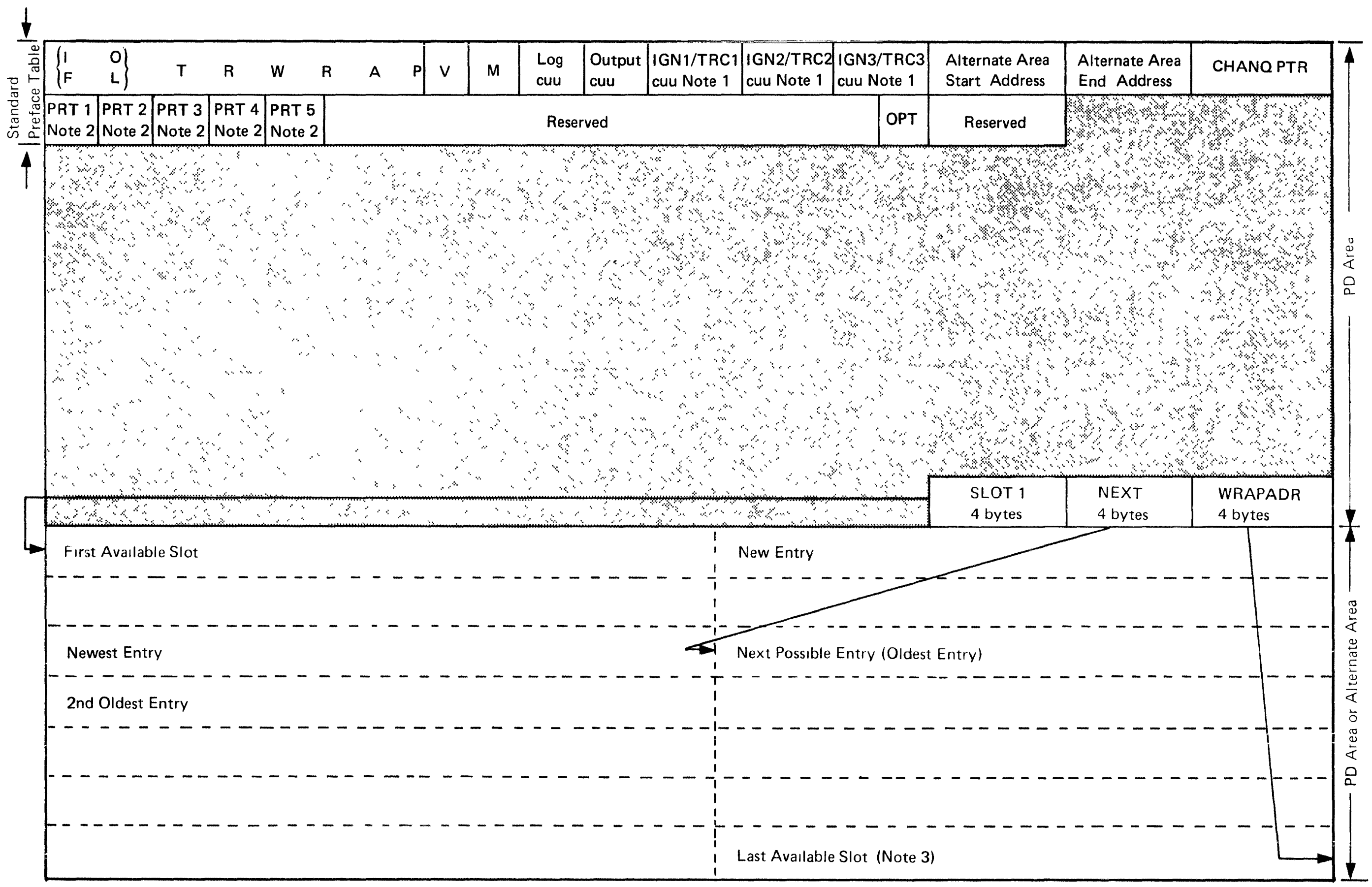
PDAIDITP is selected when a printer is specified as the output device. It collects three I/O events, then formats and prints them, using a line printer.

If the printer cannot be accessed, control is returned to the supervisor. Subsequent events are preserved by 'pushing up' the table slot, and entering each new event at the bottom (Figure 7). When an unreported event is 'pushed out' of the top, an asterisk is set into the I/O area to indicate the overflow. Printing is attempted at each entry to the phase, until successful. If the printer is not ready, or indicates errors, the message 4C24A is issued on SYSLOG and the system waits for an END response when the device is made ready.

PDAIDITT (TAPE OUTPUT)

PDAIDITT collects and writes on an unlabeled tape the I/O events that occur during execution of the problem program. The events are written on tape in core image (unprintable) format. PDAIDITT requires a tape-to-printer operation using the PDLIST utility program to retrieve the traced events. The tape must be processed with the tape drive temporarily assigned as SYS005 and SYSLST assigned to a printer to obtain readable listings of the traced events.

I/O events are collected in an area that may contain a maximum of 20 entries. An attempt is made to write the entire area as a single record when 13 events are present in the table. If the channel is busy on the attempted output, control is returned to the supervisor SIO routine or interrupt handler routine. When the next I/O event is received, it is entered in one of the five overflow entries, and another attempt to output is made. Thus, the records on tape contain between 13 and 20 I/O events per block. The PDLIST utility program takes this into account and prints only the valid I/O events.



Note 1: Used by Input/Output and QTAM Trace only.

Note 2: These bytes are used by Fetch/Load, GSVC, and QTAM Trace functions.

Note 3: When "Last Available Slot" is filled, "NEXT" is reset to "SLOT 1", and the table is overlaid with new entries.

The layout of the PD Area is shown in Figure 16.

Figure 6. Entering New Events in the Trace Table for I/C, F/L, GSVC, and QTAM Core-Wrap Modes

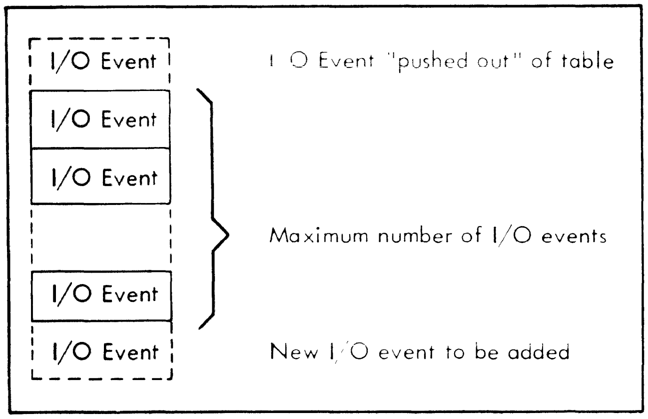


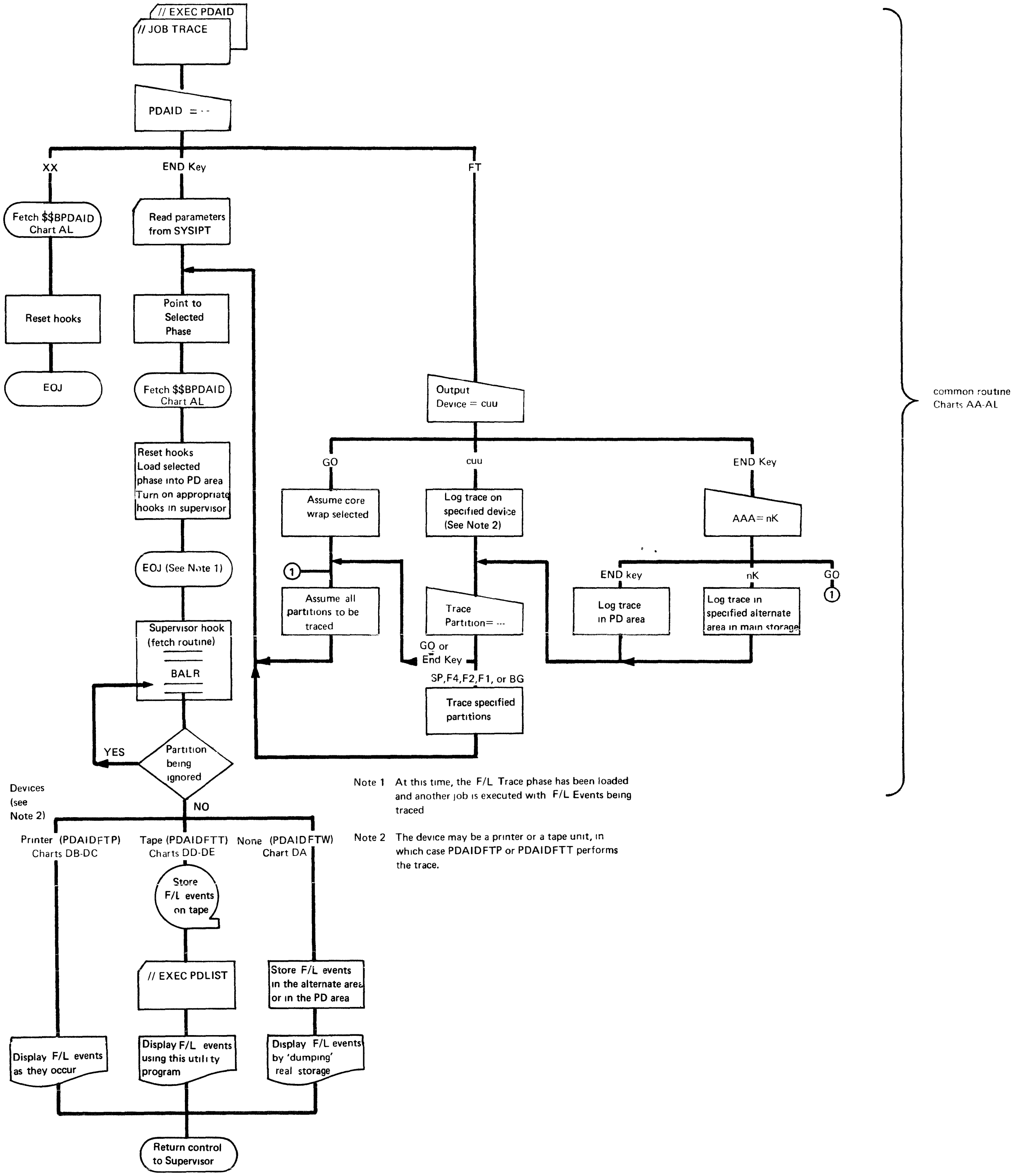
Figure 7. Entering New I/O Events in the Trace Table

Because 20 is the maximum number of I/O events that can be recorded, an overflow will occur when the 21st I/O event is entered in the table. The tape module recognizes this condition by checking the 20th entry location in the table for an unreported I/O event. If one is found, a flag is set in the preceding (19th) entry location. As the 20th entry is 'pushed out' (see Figure 7) of the table, the 19th entry replaces the 20th, and thus the flag is available in the last entry.

The PDLIST utility program checks for this flag when printing, and sets an * , indicating that an overflow has occurred and an I/O event(s) was lost. The * indicator is printed in the entry that precedes the missed I/O event(s).

PDAIDITT makes no provision to handle error conditions during output. If the tape unit is busy, an immediate return to the supervisor is made. If the tape unit is not busy, but a CSW stored condition is indicated or the tape unit is inoperable, the message 4C24A is issued on SYSLOG and the system waits for an END response when the device is ready.

Chart 03. Logic Flow of the Fetch/Load Trace Function



Note 1 At this time, the F/L Trace phase has been loaded and another job is executed with F/L Events being traced

Note 2 The device may be a printer or a tape unit, in which case PDAIDFTP or PDAIDFTT performs the trace.

FETCH/LOAD TRACE FUNCTION

The F/L (fetch/load) trace (Chart 03) allows tracing the order in which phases and transients are under control of DOS/VS. See Figure 8 for the format of F/L trace entries. Tracing consists of recording (for SVC 1, SVC 2, and SVC 4):

- The name of the phase or transient being called
- The supervisor call number
- The location of the supervisor call
- The partition identifier
- The load address of the phase
- The entry address of the phase.

Note: 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.

Each collection of data is referred to as an F/L event. The events may be preserved in a rotating buffer (first entry overwritten when the area is full), or may be printed or recorded on tape. When a tape is used, the tape must be processed by the PDLIST utility program to provide readable output data.

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 (PDAIDFTW, PDAIDFTP, or PDAIDFTT) may pertain to these incomplete SVCs.

Tracing Options

The F/L trace:

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

Trace limiting options are specified by the initializer keyword TRACE PARTITION= (see Figure 2). These options are useful only when the user runs several partitions at once, and does not wish to trace all of them. Normally, only one partition would be operating at a given time, and the default (trace all partitions) would allow both the single partition and the supervisor to be traced.

Data Collection

F/L trace phases reside in the PD area, and perform the actual tracing of F/L events. All events are recorded after the phase is physically loaded into storage. The first entry to the phase causes some initialization to occur before the F/L event is acted upon.

The value in the partition identification key (PIK) is matched against the (partition) entries to determine whether or not the event should be entered into the tables. To conserve storage, the phase tests for partitions to ignore, rather than partitions to trace. PDAID accepts parameters and converts them to ignore parameters for the F/L trace modules. For example, if F1, F2, F3, F4, and SP is specified, the initializer converts this information to an 'ignore BG' parameter for the F/L trace phase.

If the event is to be traced, an entry is made in the trace table. If not, control returns to the supervisor routine from which the trace was entered.

Output

F/L phases test for a full table before returning to the supervisor, and attempt to output the table when it contains enough entries. If the output device cannot be accessed, control returns to the supervisor, and output is retried at each subsequent entry to the trace function. Limited overflow buffers are available in each phase as shown in the following table:

Type Output	Full Table	Overflow Entries	Maximum Capacity
Printer (PDAIDFTP)	2 Entries	4 Entries	6 Entries
Tape (PDAIDFTT)	9 Entries	6 Entries	15 Entries

Phase Name	SVC	Calling Address	Load Address	Entry* Address	Partition ID
8	1	3	4	4	2

*If a phase not found condition is detected, the load and entry addresses are set to X'FFFFFF'.

Figure 8. F/L Trace Table Entry

PDAIDFTW (CORE-WRAP MODE)

PDAIDFTW preserves a fixed number of F/L events in the PD save area, or in the alternate save area of the supervisor (when AAA= is specified).

Since PDAIDFTW executes in the PD area, the actual save area - when the PD save area is used - is equal to the PD area (generated during system generation) minus the length of PDAIDFTW.

The maximum number of entries that can be stored in the PD area is shown in Figure 1. When the PD area is full, the oldest entry is overlaid by each new entry. (Figure 6 illustrates the method for updating the trace table for core-wrap mode.)

To retrieve the trace tables, the PD area must be dumped. This can be done by using the PDAREA operand of the DUMP command. Of course any dump that includes the supervisor will also include the PD area.

If the alternate area is used, it is dumped automatically on the device assigned to SYSLST, when the PDAID program is terminated.

PDAIDFTW sets up the following pointers:

- SLOT1 -- address of the beginning of the save area.
- 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.)
- WRAPADR -- address of the end of the save area.

Note: The location of SLOT1 is shown in Figure 5.

PDAIDFTP (PRINTER OUTPUT)

PDAIDFTP is selected when a printer is specified as the output device. It collects two F/L events, formats them for output, and prints them on a line printer. See Figure 9 for sample output.

If the printer cannot be accessed, control returns to the supervisor. If the printer is off-line or is not ready, the message 4C24A is issued on SYSLOG and the system waits for an END response when the printer is made ready.

When two entries have been saved, they are formatted for output, and the save area is cleared. Two more entries may be saved before output is achieved, without an overflow condition occurring. However, when the I/O area and save area are full and an entry must be made, the oldest entry in the save area is lost, and an * indicator is set in the I/O area. Thus, an * in the output indicates that the next chronological entry (or entries) was lost due to an overflow.

PDAIDFTT collects and writes, on an unlabeled tape, the F/L events that occur during execution of a job stream. The events are written on tape in core image format. Output from PDAIDFTT is formatted into printable characters by the PDLIST utility program. The input tape drive must be temporarily assigned as SYS005 before execution of PDLIST, and SYSLST must be assigned to a printer.

F/L events are collected in a save area inside the PD area. When nine entries have been made in the save area, the phase attempts to output. If the tape drive cannot be accessed, control returns to the supervisor. If the tape drive is not ready, or indicates errors, the message 4C24A is issued on SYSLOG and the system waits for an END response when the device is made ready. At each entry to the phase, the top slot in the save area is checked for an entry. If an entry is present, the save area is full, and an overflow occurs when the current entry is saved. If an overflow occurs, a flag is set in the oldest entry remaining after the current entry is saved. The PDLIST utility program checks for this flag when printing, and sets an * indicator when the flag is found, signifying that at least one F/L event was lost due to an overflow.

Because PDAIDFTT attempts to output when it receives the ninth entry and has a buffer of six entry slots, there are between 9 and 15 entries per 330-byte block on the tape. The PDLIST utility program therefore checks for a valid entry before formatting.

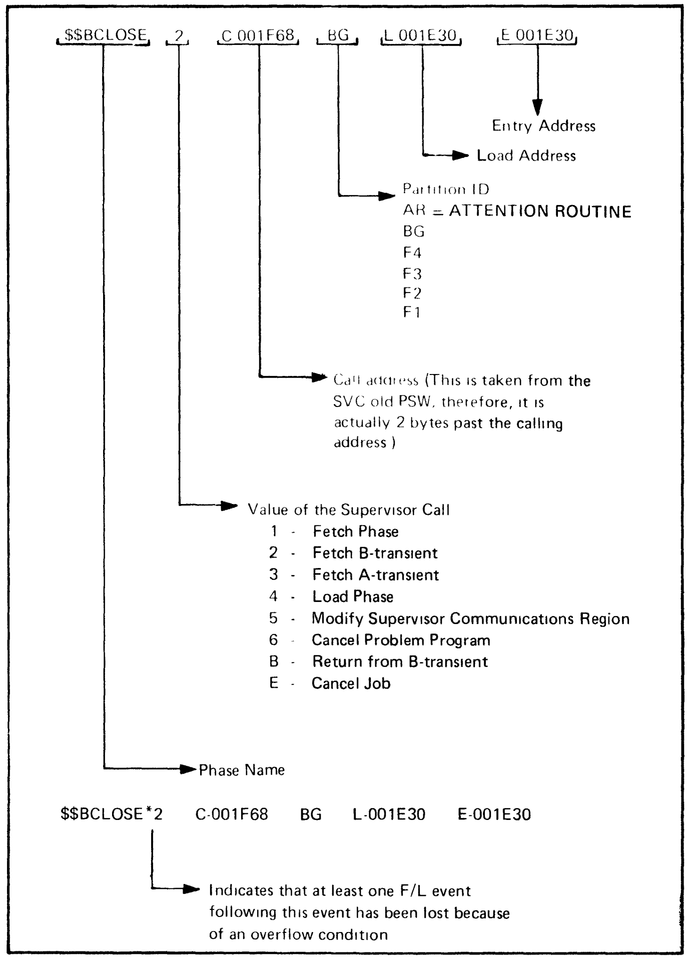
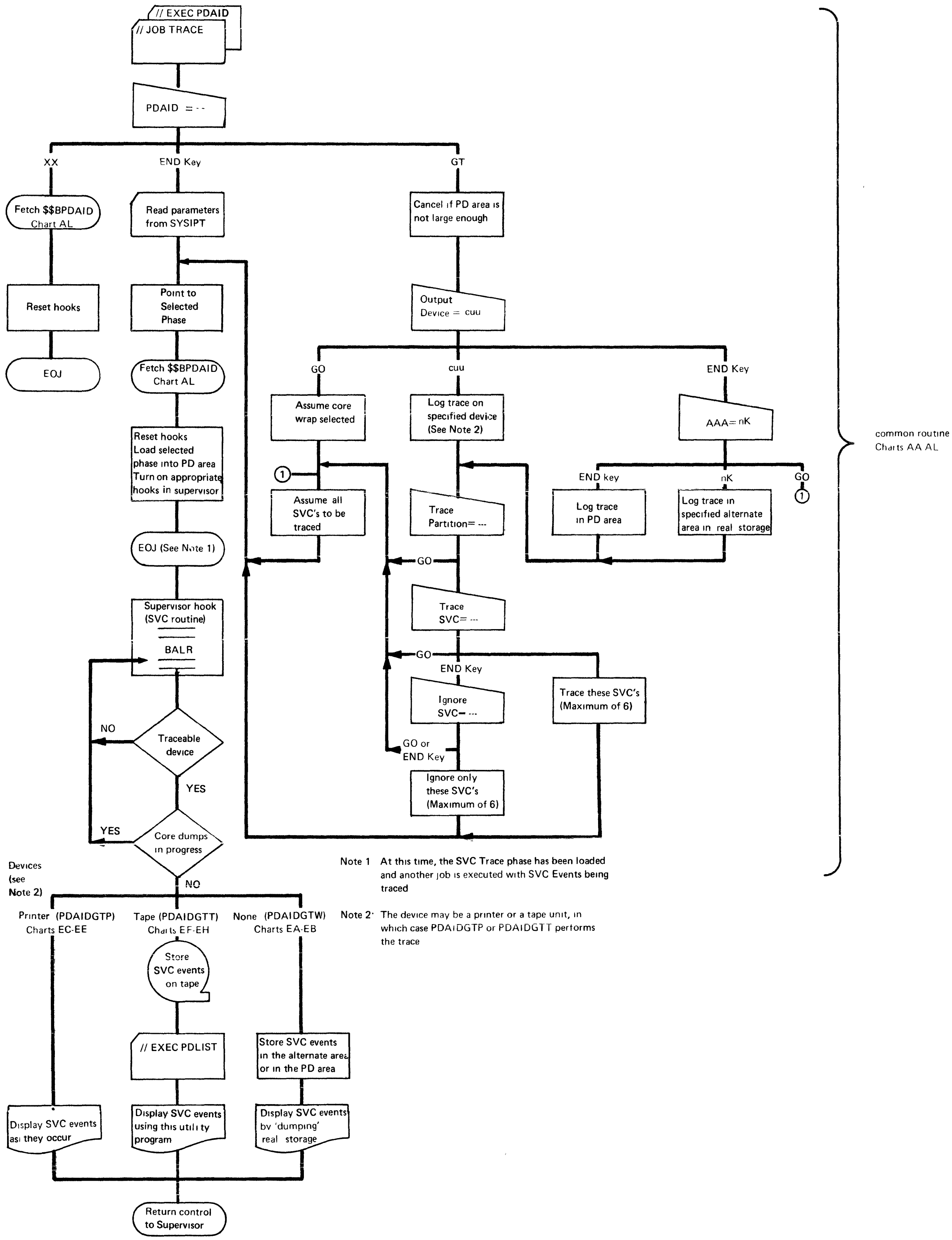


Figure 9. Sample Output for F/L Trace

Chart 04. Logic Flow of the GSVC Trace Function



Note 1 At this time, the SVC Trace phase has been loaded and another job is executed with SVC Events being traced

Note 2 The device may be a printer or a tape unit, in which case PDAIDGTP or PDAIDGTT performs the trace

GSVC TRACE FUNCTION

The GSVC trace function (Chart 04) builds a trace table as SVC interrupts occur. The trace table (Figure 10) consists of:

- The SVC old PSW
- The contents of register 0
- The last three bytes of register 1
- The SVC number
- The partition identifier.

The trace table entries may be stored in a rotating buffer in core (core-wrap mode), or output to a printer or tape unit. When tape output is used, the tape must be processed by the PDLIST utility program to provide readable output data.

SVC old PSW	Reg. 0	SVC no.	Last 3 bytes reg. 1	Partition ID
8	4	1	3	2

Figure 10. GSVC Trace Table Entry

Tracing Options

The GSVC function provides the following options:

- Trace all SVCs
- Selectively trace up to six SVCs
- Selectively eliminate up to six SVCs and trace all others
- Trace in all partitions
- Selectively trace up to five partitions.

Trace limiting options (see Figure 2) 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 either is specified, the other becomes invalid.

The six SVC limiting options are invoked by specifying the SVCs to be traced or ignored. 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.

Note: If PTO=YES in the FOPT macro then SVCs issued when the physical transient area is busy are not traced.

Data Collection

GSVC trace resides in the PD area and performs the actual tracing of SVCs. The first entry to the phase causes some initialization to occur before the SVC is acted upon.

The value in the program interrupt key (PIK) is matched against the PRT (partition) entries to determine whether or not the event should be entered into the tables. To conserve storage, the phases test for partitions to ignore, rather than partitions to trace. The initializer program (PDAID) accepts TRACE PARTITION= parameters and converts them to ignore parameters for the SVC trace phases. For example, if F1, F2, F3, F4, and SP is specified, the initializer converts this information to an 'ignore BG' parameter for the SVC trace phase.

If the event is to be traced, an entry is made in the trace table. If not, control returns to the supervisor routine from which the trace was entered.

Output

If the SVC event was caused by an SVC being issued, the phases store the necessary information in the table but do not attempt to move data to the I/O area. If, however, the SVC event was caused by an SIO or I/O interrupt, the phases first store the information in the table and then test for a full table. If the table is full, the phases attempt an output before returning control to the supervisor.

Type Output	Full Table	Overflow Entries	Maximum Capacity
Printer	2 Entries	4 Entries	6 Entries
Tape	9 Entries	5 Entries	14 Entries

When the output device must share a selector channel or a control unit, the trace tables overflow capacity can be exceeded and SVCs can be lost. The trace phase tests for such losses and indicates

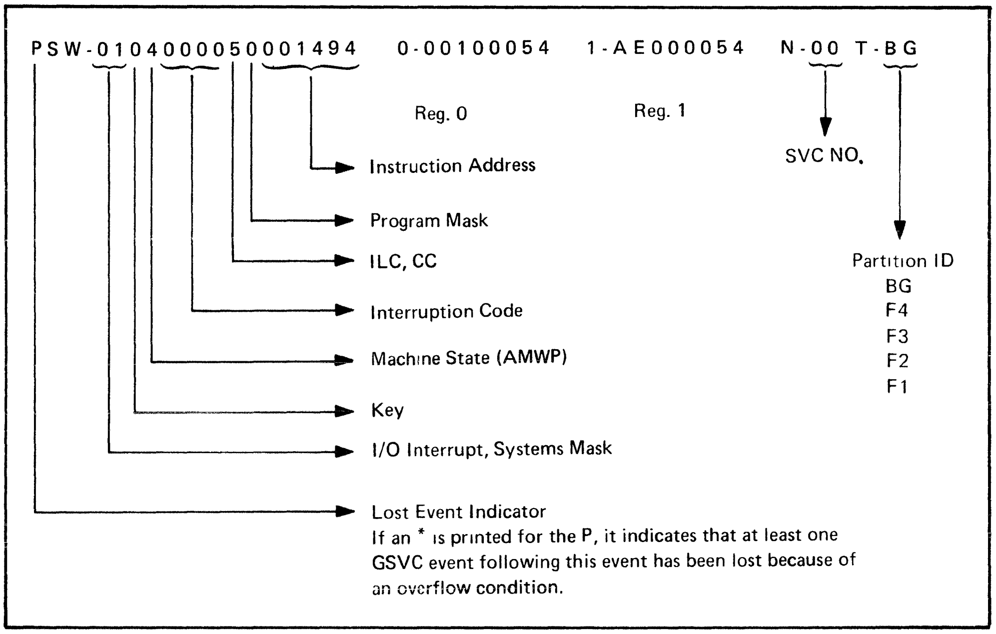


Figure 11. Sample Output for GSVC Trace

them with an * when they occur. (It is recommended that output options be selected to avoid such losses, and that one or more of the most active devices be ignored if an overflow occurs.) Figure 11 illustrates output after it has been printed.

PDAIDGTW (CORE-WRAP MODE)

PDAIDGTW preserves a fixed number of I/O events in the PD save area, or in the alternate save area of the supervisor (when AAA= is specified).

Since PDAIDGTW executes in the PD area, the actual save area - when the PD save area is used - is equal to the PD area (generated during system generation) minus the length of PDAIDGTW.

The maximum number of entries that can be stored in the PD area is shown in Figure 1. When the PD area is full, the oldest entry is overlaid by each new entry. (Figure 6 illustrates the method for updating the trace table for core-wrap mode.)

To retrieve the trace tables, the PD area must be dumped. This can be done by using the PDAREA operand of the DUMP command. Of course any dump that includes the supervisor will also include the PD area.

If the alternate area is used, it is dumped automatically on the device assigned to SYSLST, when the PDAID program is terminated.

PDAIDGTW sets up the following pointers:

- SLOT1 -- address of the beginning of the save area.
- 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, the entry should be ignored.)
- WRAPADR -- address of the end of the save area.

<u>Note</u>: The location of SLOT1 is shown in Figure 5.

PDAIDGTP (PRINTER OUTPUT)

PDAIDGTP is selected when a printer is specified as the output device. It collects the two GSVC events, formats them for output, and prints them using a line printer.

If the printer cannot be accessed, control returns to the supervisor. If the printer is not ready or indicates errors, the message 4C24A is printed on SYSLOG and the system waits for an END response when the printer is made ready.

When two entries have been saved, they are formatted for output and the save area is cleared. Four more entries may be saved before output is achieved, without an overflow condition occurring. However, when the I/O area and save area are full and an additional entry must be made, this entry causes the save area to overflow (Figure 12). The oldest entry in the save area is lost, and an * indicator is set in the I/O area. Thus, an * in the output indicates that the next chronological entry (or entries) was lost due to an overflow. Entries are double-spaced and printed two per line.

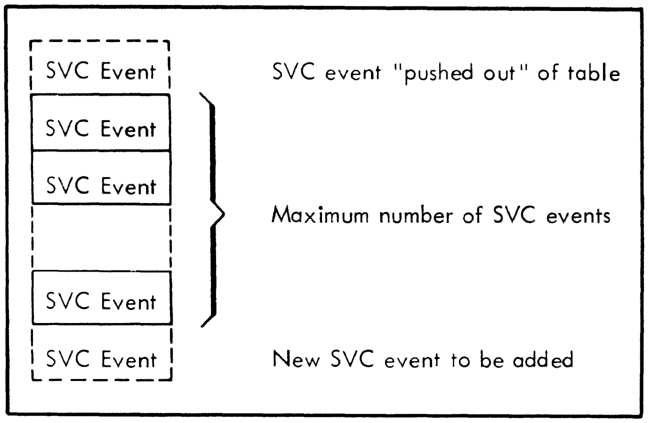


Figure 12. Entering New SVC Events in the Trace Table for Output Devices

PDAIDGTT (TAPE OUTPUT)

PDAIDGTT collects and writes on an unlabeled tape the SVC events that occur during execution of the problem program. The events are written on tape in core image (unprintable) format. PDAIDGTT requires a tape-to-printer operation using the PDLIST utility program to retrieve the traced events. The tape must be processed with the tape drive temporarily assigned as SYS005 and SYSLST assigned to a printer to obtain readable listings of the traced events.

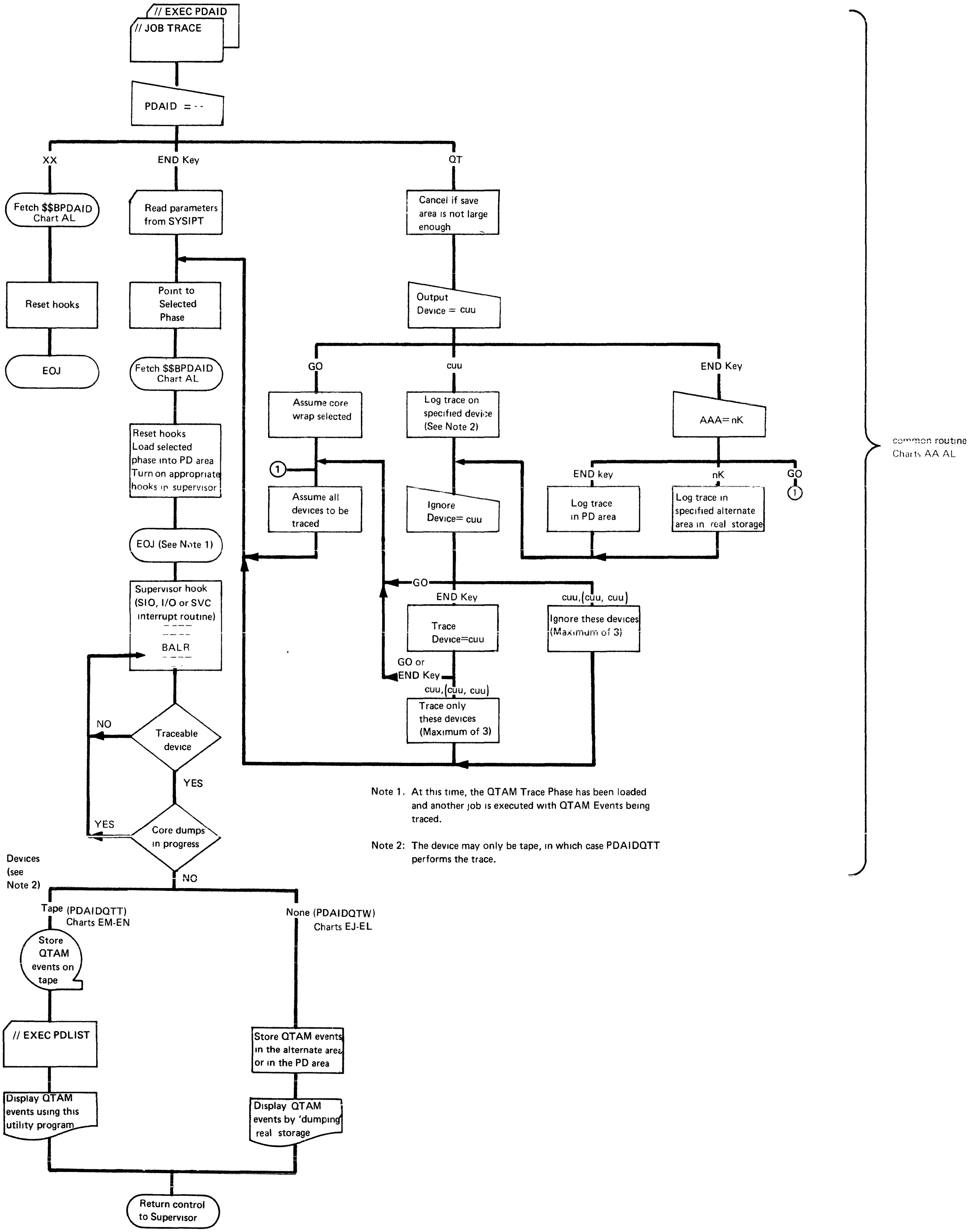
SVC events are collected in an area that may contain a maximum of 14 entries. An attempt is made to write the entire area as a single record when 9 events are present in the table. If the channel is busy on the attempted output, control is returned to the supervisor SIO routine or interrupt handler routine. When the next SVC event is received, it is entered in one of the five overflow entries, and another attempt to output is made. Thus, the records on tape contain between 9 and 14 SVC events per block. The PDLIST utility program takes this into account and prints only the valid SVC events.

Because 14 is the maximum number of SVC events that can be recorded, an overflow will occur when the 15th SVC event is entered in the table. PDAIDGTT recognizes this condition by checking the 14th entry location in the table for an unreported SVC event. If one is found, a flag is set in the preceding (13th) entry location. As the 14th entry is 'pushed out' of the table, (see Figure 12) the 13th entry replaces the 14th, and thus the flag is available in the last entry.

The PDLIST utility program checks for the flag when printing, and sets an * indicator, indicating that an overflow has occurred and an SVC event(s) was lost. The * indicator is printed in the entry that precedes the missed SVC event(s).

PDAIDGTT makes no provision to handle error conditions during output. If the tape unit is busy, an immediate return to the supervisor is made. If the tape unit is not busy, but a CSW stored condition is indicated or the tape unit is inoperable, the message 4C24A is issued on SYSLOG and the system waits for an END response when the device is made ready.

Chart 05. Logic Flow of the QTAM Trace Function



Note 1. At this time, the QTAM Trace Phase has been loaded and another job is executed with QTAM Events being traced.

Note 2: The device may only be tape, in which case PDAIDQTT performs the trace.

QTAM TRACE FUNCTION

Description and Operation

The QTAM trace function builds a trace table as interrupts occur. There are three types of trace events, each having a prefix that defines the event type (Figure 13).

A V-type event (Figure 13) is created when an SVC interrupt occurs and consists of:

- V-prefix
- SVC number
- SVC old PSW
- Contents of register 0
- Contents of register 1
- Partition identifier.

An S-type event (Figure 13) is created when an SIO interrupt occurs and consists of:

- S-prefix
- Condition code
- Device address
- CCB address
- CSW.

An I-type event (Figure 13) is created when an I/O interrupt occurs and consists of:

- I-prefix
- Device address
- I/O old PSW
- CSW.

The trace events may be stored in a rotating buffer in core (core wrap) or sent to a tape unit. When tape mode is used, the tape must be processed by the PDLIST utility program to provide readable output data.

SVC Interrupt:

V	SVC No	SVC OLD PSW	Reg.0	Reg.1	Part id
1	2	8	4	4	2

SIO Interrupt:

S	CC	Device Address	CCB Address	CSW	Not used
1	1	2	4	8	5

I/O Interrupt:

I	Device address	I/O old PSW	CSW	Not used
1	2	8	8	2

Figure 13. QTAM Trace Table Entries

Tracing Options

The QTAM trace function provides the following options:

- Trace all SVC 0 and 31, SIO, and I/O interrupts
- Selectively 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 any three devices
- Limit the trace by partition.

Trace-limiting options (see Figure 2) 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 if one of these options is not specified. They are mutually exclusive: when either is specified, the other becomes invalid.

The partition limiting options are specified by the initializer keyword TRACE PARTITION=. This is useful only if the user runs several partitions at the same time, but does not wish to trace all of them.

Data Collection

The QTAM trace phases reside in the PD area and perform the actual tracing of QTAM events. The first QTAM event causes some initialization to occur before it is handled. At each entry, the trace tests the logical transient area for a dump transient; normally, it does not trace any interrupt activity when a dump is in progress. (If it is necessary to trace QTAM events during a dump routine, the exit branch following the compare instruction should be altered to a NOP.) This prevents the dump from either causing the trace table to overflow when core wrap is used (see Figure 3), or from causing excess output.

If no dump is in progress, the device address is matched against either IGN or TRC entries to determine if the event should be entered into the trace tables.

If the event is not to be traced, control is returned to the supervisor routine from which the trace was entered. If the event involves a device to be traced, an entry is made in the trace table. At this point, the core-wrap module returns control to the supervisor.

Output

The QTAM output phases test for a full table before returning to the supervisor, and attempt to output the table when it contains enough entries. If the output device cannot be accessed, control is returned to the supervisor, and output is retried at each subsequent entry to the trace module. Limited overflow buffers are available in each module:

Type Output	Full Table	Overflow Entries	Maximum Capacity
Tape	8 Entries	2 Entries	10 Entries

When the output device must share a selector channel or a control unit, the overflow capacity can be exceeded, and QTAM events can be lost. The trace tests for such losses, and indicates such with an * when they occur. (It is recommended that output options be selected to avoid such losses, and that one or more of the most active devices be ignored if an overflow occurs.) Figure 14 illustrates output after it has been printed.

PDAIDQTW (CORE-WRAP MODE)

PDAIDQTW preserves a fixed number of events in the PD save area, or in the alternate save area of the supervisor (when AAA= is specified).

Since PDAIDQTW executes in the PD area, the actual save area - when the PD save area is used - is equal to the PD area (generated during system generation) minus the length of PDAIDQTW.

The number of entries that can be stored in the PD area is shown in Figure 1. When the PD area is full, the oldest entry is overlaid by each new entry. (Figure 6 illustrates the method for updating the trace table for core-wrap mode.)

To retrieve the trace tables, the PD area must be dumped. This can be done by using the PDAREA operand of the DUMP command. Of course any dump that includes the supervisor will also include the PD area.

If the alternate area is used, it is dumped automatically on the device assigned to SYSLST, when the PDAID program is terminated.

PDAIDQTW sets up the following pointers.

- SLOT 1 -- address of the beginning of the save area.
- NEXT -- address of the next available slot in the save area. (Because NEXT is filled with unchecked new data, it may contain either the oldest entry in the table, or the SVC number and calling address from the partition not being traced. If the latter is the case, the entry should be ignored.)
- WRAPADR -- address of the end of the save area.

Note: The location of SLOT1 is shown in Figure 5.

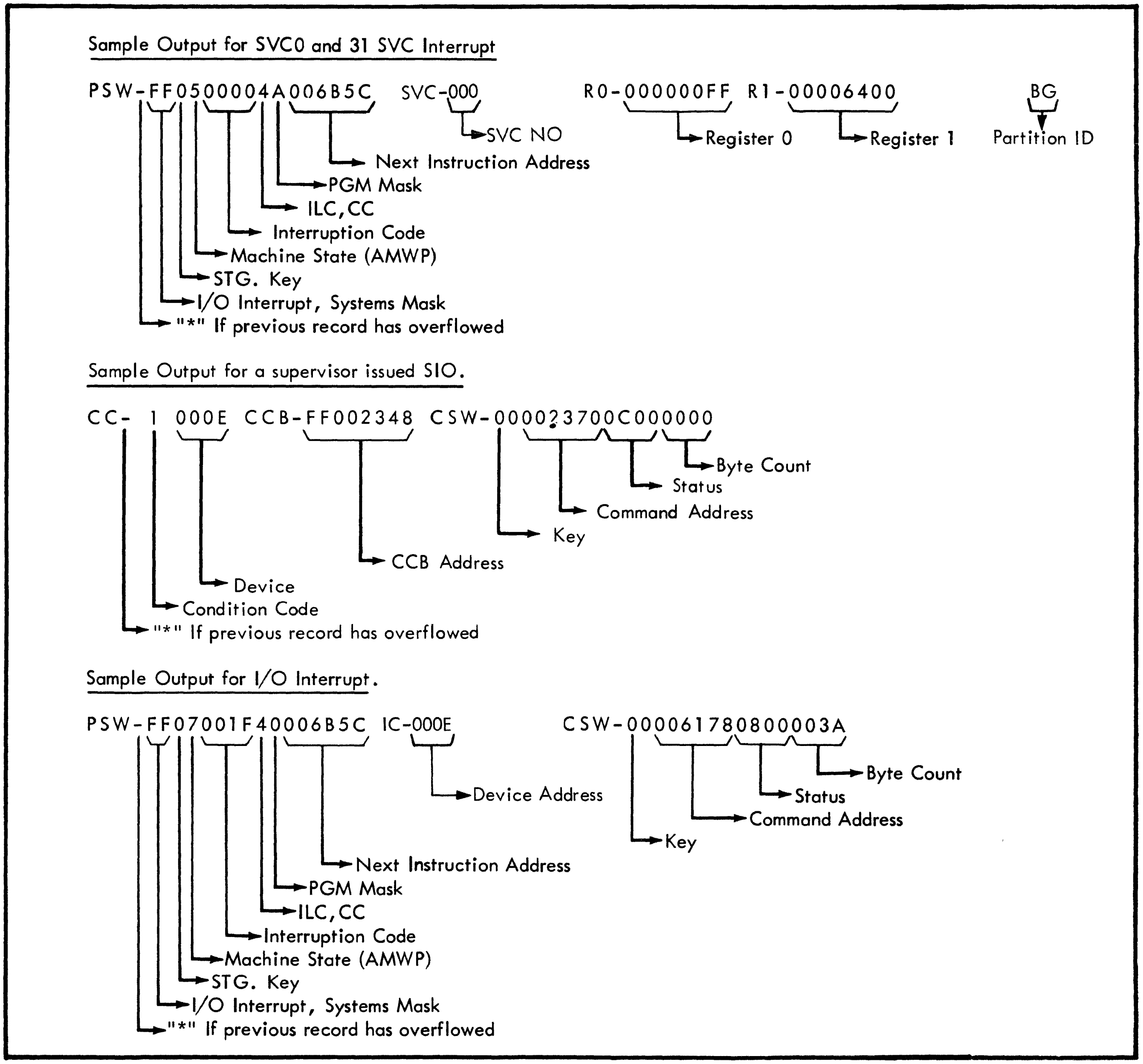


Figure 14. Sample Output for QTAM Trace

PDAIDQTT (TAPE OUTPUT)

PDAIDQTT collects and writes on an unlabeled tape the QTAM events that occur during execution of the problem program. The events are written on tape in core image (unprintable) format. Thus, this requires a tape-to-printer operation using the PDLIST utility program to retrieve the traced events. The tape must be processed with the tape drive temporarily assigned as SYS005, and SYSLST assigned to a printer to obtain readable listings of the traced events.

QTAM events are collected in an area that may contain a maximum of 10 entries. An attempt is made to write the entire area as a single record when 8 events are present in the table. If the channel is busy on the attempted output, control is returned to the supervisor SIO routine or interrupt handler routine. When the next QTAM event is received, it is entered in one of the two overflow entries, and another attempt to output is made. Thus, the records on tape contain between 8 and 10 QTAM events per block. PDLIST utility program takes this into account and prints only the valid QTAM events.

Because 10 is the maximum number of QTAM events that can be recorded, an overflow occurs when the 11th QTAM event is entered in the table. PDAIDQTT recognizes this condition by checking the 10th entry location in the table for an unreported QTAM event. If one is found, a flag is set in the preceding (9th) entry location. As the 10th entry is "pushed out" of the table (Figure 15), the 9th entry replaces the 10th, and thus the flag is available in the last entry. The PDLIST utility program checks for this flag when printing, and sets an * indicator, noting that an overflow has occurred and a QTAM event(s) was lost. The * indicator is printed in the entry that precedes the missed QTAM event(s).

PDAIDQTT makes no provision to handle error conditions during output. If the tape unit is busy, an immediate return to the supervisor is made. If the tape unit is not busy, but a CSW stored condition is indicated or the tape unit is inoperable, the message 4C24A is issued on SYSLOG and the system waits for an END response when the device is made ready.

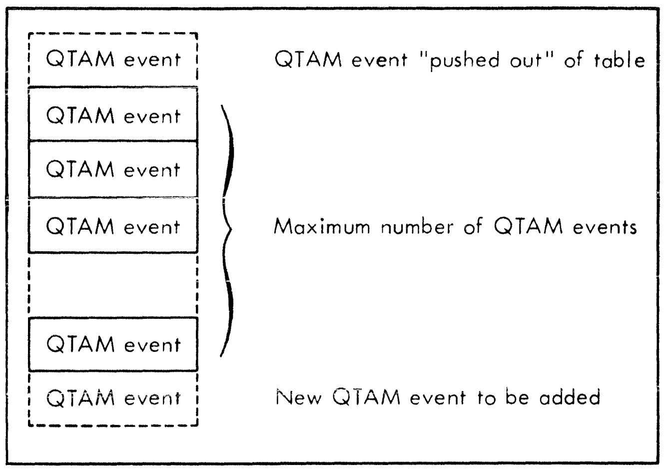
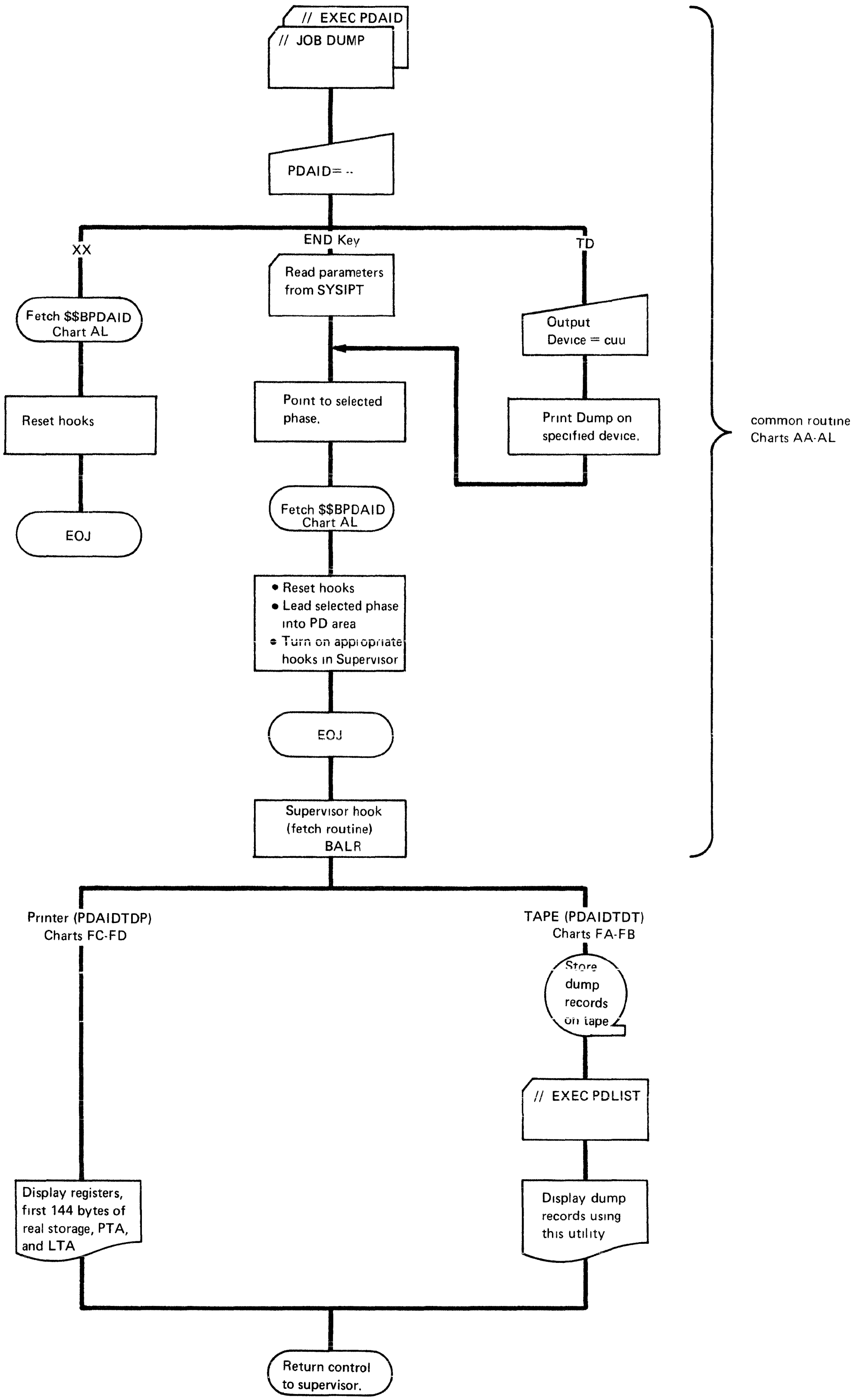


Figure 15. Entering New QTAM Events in the Trace Table for Tape Output Devices

Chart 06. Logic Flow of Transient Dump



TRANSIENT DUMP

Transient dump (Chart 06) allows certain areas of the supervisor to be displayed when a program check occurs. These areas are displayed before they are altered by any supervisor routines. The areas displayed are:

- The sixteen general registers being used at the time of failure
- The first 144 bytes of real storage
- The PTA (physical transient area)
- The LTA (logical transient area).

The output device of the transient dump program must be either a printer or a magnetic tape unit. If a tape is used, the tape must be processed by the PDLIST utility program to provide readable output data.

Tracing Options

None.

Data Collection

Transient dump phases reside in the PD area. The transient areas are displayed before they can be altered by any supervisor routine; the registers and first 144 bytes of real storage are restored to their values at the time of failure before they are displayed.

Output

Transient dump takes control of the system when a program check occurs. If the output device fails, message

4C24A NO I/O TO OD

is issued. The exit from this message is either to the output routine if the failure is corrected, or to the supervisor.

PDAIDTDP

PDAIDTDP is selected when a printer is specified as the output device. Control is obtained from the supervisor and all areas are printed. If the channel program for the printer cannot be activated, control returns to the supervisor. If the printer is off-line or not ready, the message 4C24A is issued on SYSLOG and the system waits for an END key response when the printer is made ready.

Each area is blocked, formatted for output, and printed. There is no loss of data due to overflow since the I/O new PSW is altered so that control is always returned to PDAIDTDP until all data is printed.

PDAIDTDT

PDAIDTDT collects and writes, on an unlabeled tape, the contents of the general registers, the first 144 bytes of real storage, the PTA, and the LTA. Control is obtained from the supervisor and the records containing all areas are written on tape. Output from PDAIDTDT is formatted into printable characters by the PDLIST utility program.

If the channel program for the magnetic tape unit cannot be activated, control returns to the supervisor. If the tape unit is off-line or not ready, the message 4C24A is issued on SYSLOG and the system waits for an END key response when the tape unit is made ready.

Each area is blocked, formatted for output, and written. There is no loss of data due to overflow since the I/O new PSW is altered so that control is always returned to PDAIDTDT until all data is written.

	Byte								
PDAREA	0	PDAREAND				PDINTRHK			
	8	PDSIOHK				PDTRANHK			
	16	PDFLTHK1				PDFLTHK2			
	24	Reserved				Register 9			
	32	Register 10				Register 11			
	40	Phase Name							
	48	VER	MOD	LCG cuu		Output cuu		IGN1/TRC1 cuu	
	56	IGN2/TRC2 cuu		IGN3/TRC3 cuu		Alternate Area Start			
	64	Alternate Area End				CHANQ PTR			
	72	PRT1	PRT2	PRT3	PRT4	PRT5	Reserved		
	80	Reserved							
	88	Reserved			OPT	Reserved			
	96	Entries							

Figure 16. Standard Preface in the PD Area (Part 1 cf 2)

Displacement	Label	Description
0-3	PDAREAND	Address of end of PD Area
4-7	PDINTRHK	Address of interrupt trace hook *
8-11	PDSIOHK	Address of START I/O hook *
12-15	PDTRANHK	Address of transient dump hook *
16-19	PDFLTHK1	Address of F/L trace hook 1 *
20-23	PDFLTHK2	Address of F/L trace hook 2 *
24-27		Reserved
28-39		Save area for registers 9, 10, and 11
40-47	Phase Name	Phase being run
48	VER	Version number in hexadecimal
49	MOD	Modification level in hexadecimal
50-51	LOG	Address of system log device
52-53	Output	Address of output device
54-55 56-57 58-59	IGN1/TRC1 IGN2/TRC2 IGN3/TRC3	Address(es) of devices to ignore or trace
60-63	Alternate Area Start	Start address of alternate area
64-67	Alternate Area End	Ending address of alternate area
68-71	CHANQ PTR	Address of channel queue pointer for output device
72-76	PRT1-PRT5	Partition(s) to be ignored **
77-90		Reserved
91	OPT	Option byte X'00' = TRC device X'80' = IGN device
92-95		Reserved
96-n		Entries

Note: The address of the PD Area is found at SYSCOM + X'48'

* 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.

** 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 16. Standard Preface in the PD Area (Part 2 of 2)

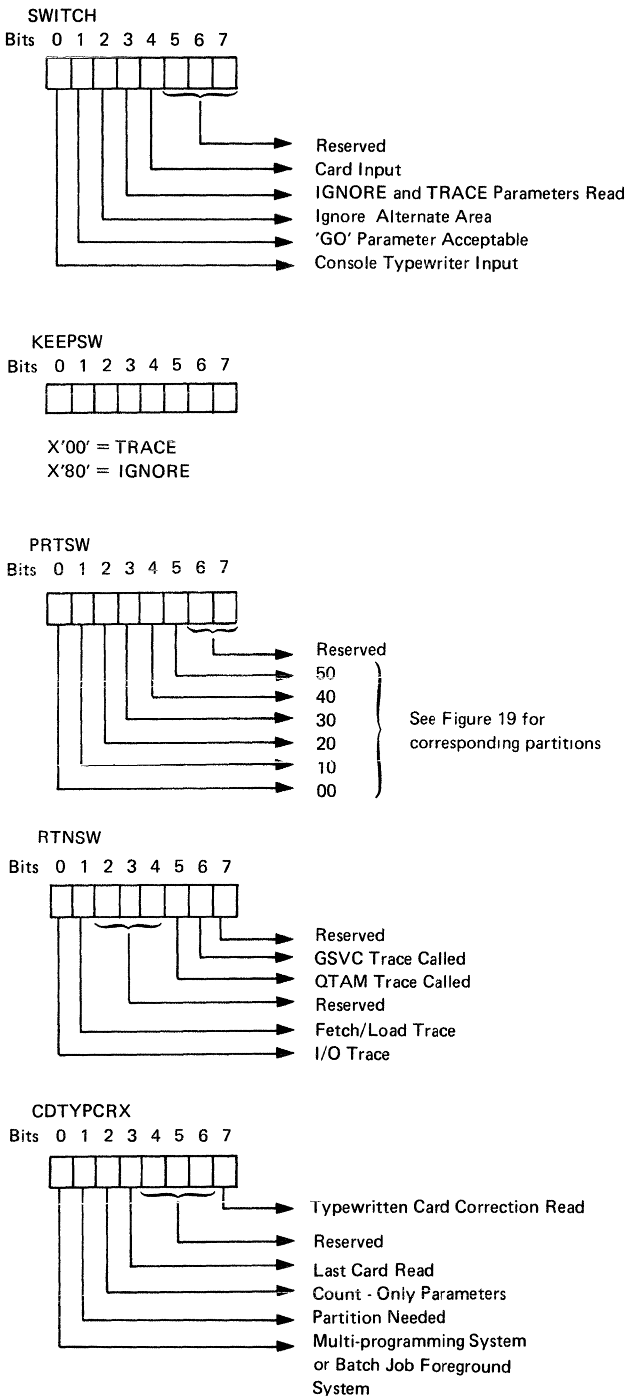


Figure 17. Switches Used by PDAID

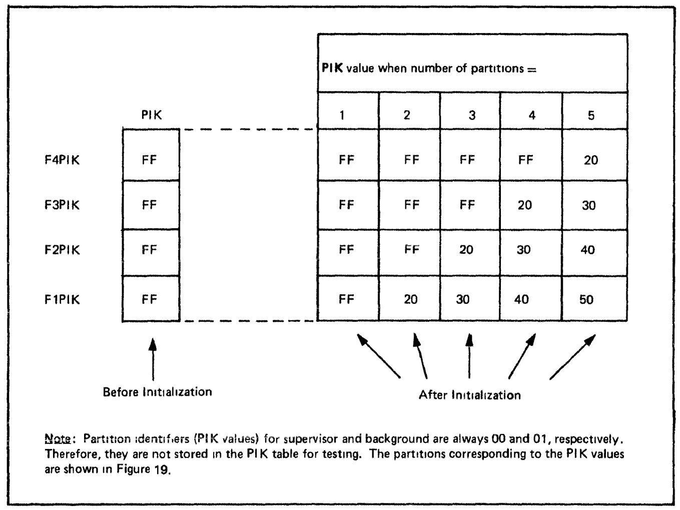
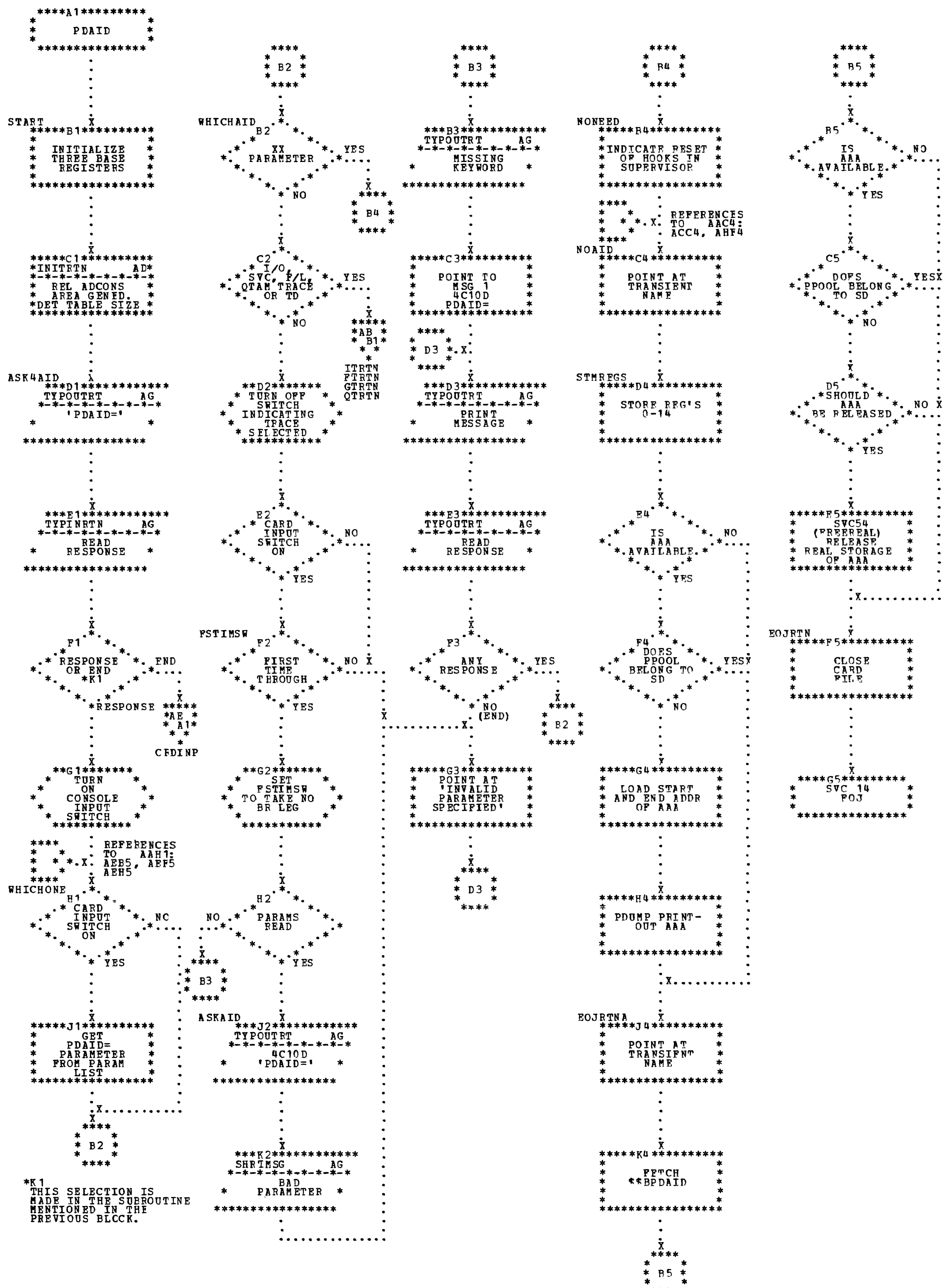


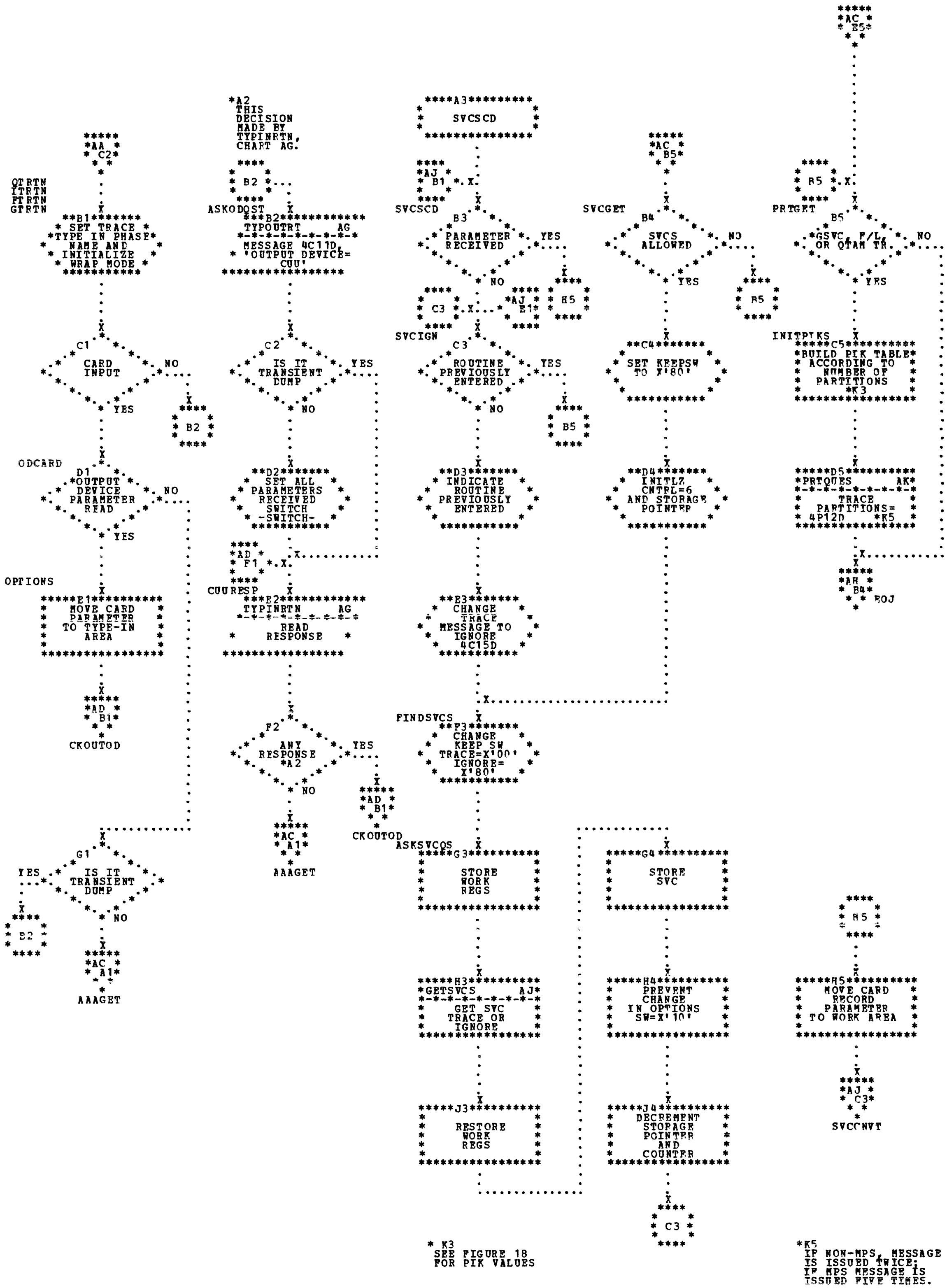
Figure 18. PIK Table in PDAID Phase

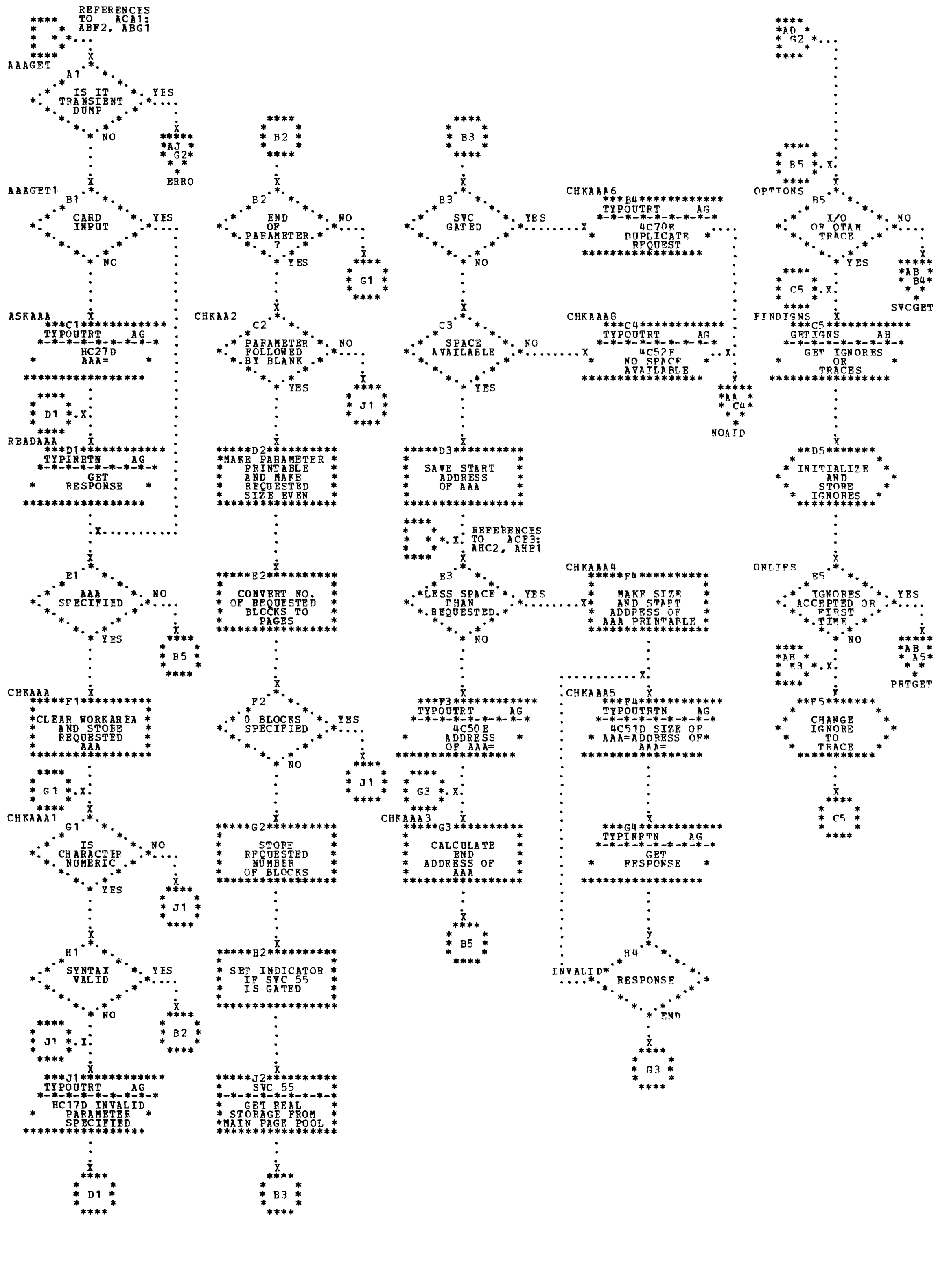
PIK Value	Partitions ignored when number of partitions = 1	2	3	4	5
50	-	-	-	-	F1
40	-	-	-	F1	F2
30	-	-	F1	F2	F3
20	-	F1	F2	F3	F4
10	BG	BG	BG	BG	BG
00	SP	SP	SP	SP	SP

Figure 19. Partition Identifiers

Chart AA. PDAID Initializer: Determining the Function




AC E5
A2 THIS DECISION MADE BY TYPINRTN, CHART AG.
A3 SVCSCD
AA C2
QTRTN ITRTN PTRTN GTRTN
B1 SET TRACE TYPE IN PHASE NAME AND INITIALIZE WRAP MODE
ASKODOST B2 TYPOUTRT AG MESSAGE 4C11D 'OUTPUT DEVICE= CUU'
SVCSCD B3 PARAMETER RECEIVED
SVCGET B4 SVCS ALLOWED
PRTGET B5 GSVC, F/L, OR QTAM TR.
C1 CARD INPUT
C2 IS IT TRANSIENT DUMP
SVCIGN C3 ROUTINE PREVIOUSLY ENTERED
C4 SET KEEPSW TO X'80'
INITPIKS C5 BUILD PIK TABLE ACCORDING TO NUMBER OF PARTITIONS K3
ODCARD D1 OUTPUT DEVICE PARAMETER READ
D2 SET ALL PARAMETERS RECEIVED SWITCH -SWITCH-
D3 INDICATE ROUTINE PREVIOUSLY ENTERED
D4 INITLZ CNTRL=6 AND STORAGE POINTER
D5 PRTQUES AK TRACE PARTITIONS= 4P12D K5
AH B4 EOJ
OPTIONS E1 MOVE CARD PARAMETER TO TYPE-IN AREA
CUURESP E2 TYPINRTN AG READ RESPONSE
E3 CHANGE TRACE MESSAGE TO IGNORE 4C15D
AD B1 CKOUTOD
F2 ANY RESPONSE A2
FINDSVCS F3 CHANGE KEEP SW TRACE=X'00' IGNORE= X'80'
AC A1 AAAGET
G1 IS IT TRANSIENT DUMP
ASKSVCOS G3 STORE WORK REGS
G4 STORE SVC
H3 GETSVCS AJ GET SVC TRACE OR IGNORE
H4 PREVENT CHANGE IN OPTIONS SW=X'10'
H5 MOVE CARD RECORD PARAMETER TO WORK AREA
AJ C3 SVCCNVT
J3 RESTORE WORK REGS
J4 DECREMENT STORAGE POINTER AND COUNTER
K3 SEE FIGURE 18 FOR PIK VALUES
K5 IF NON-MPS, MESSAGE IS ISSUED TWICE; IF MPS MESSAGE IS ISSUED FIVE TIMES.



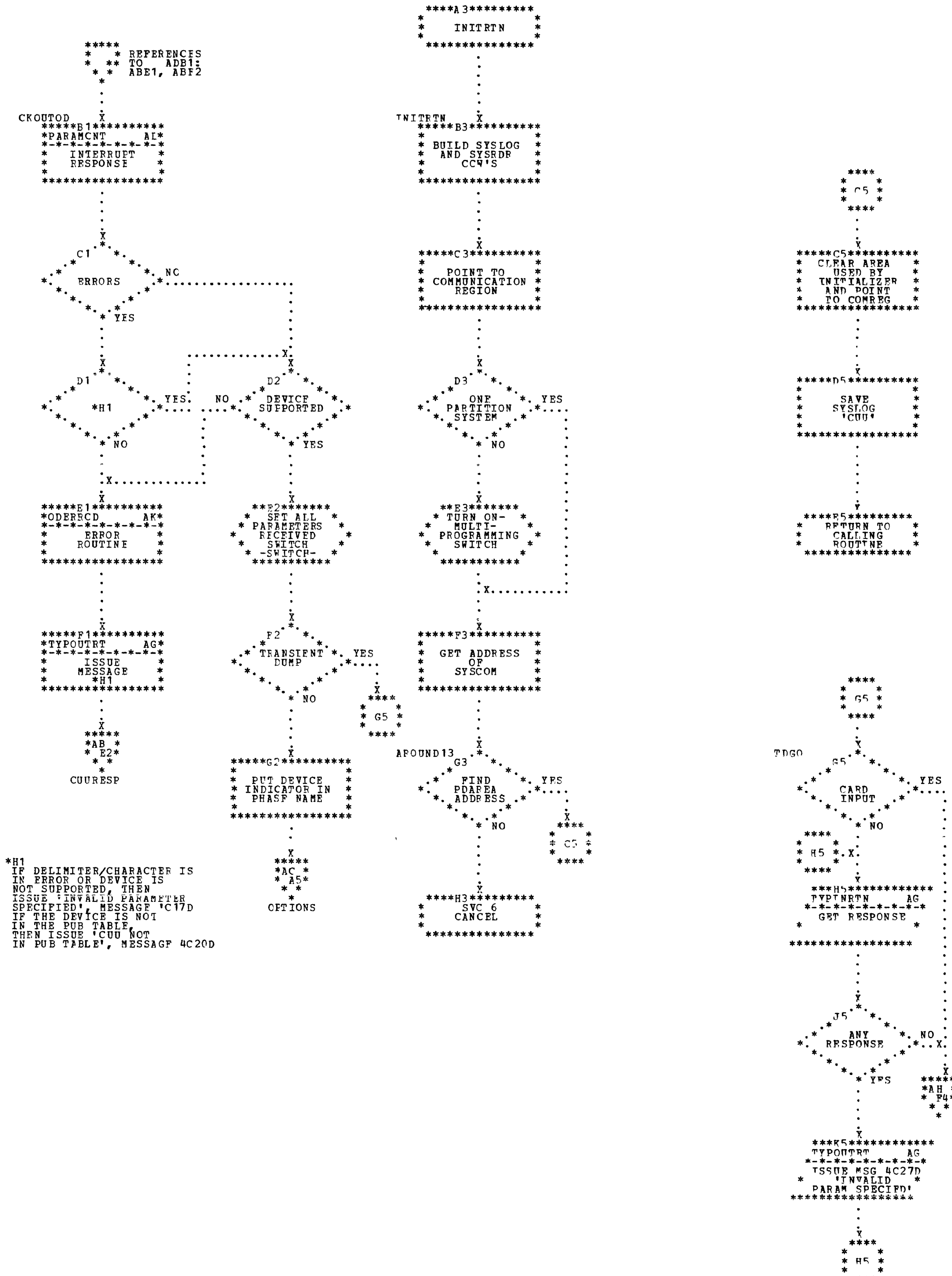
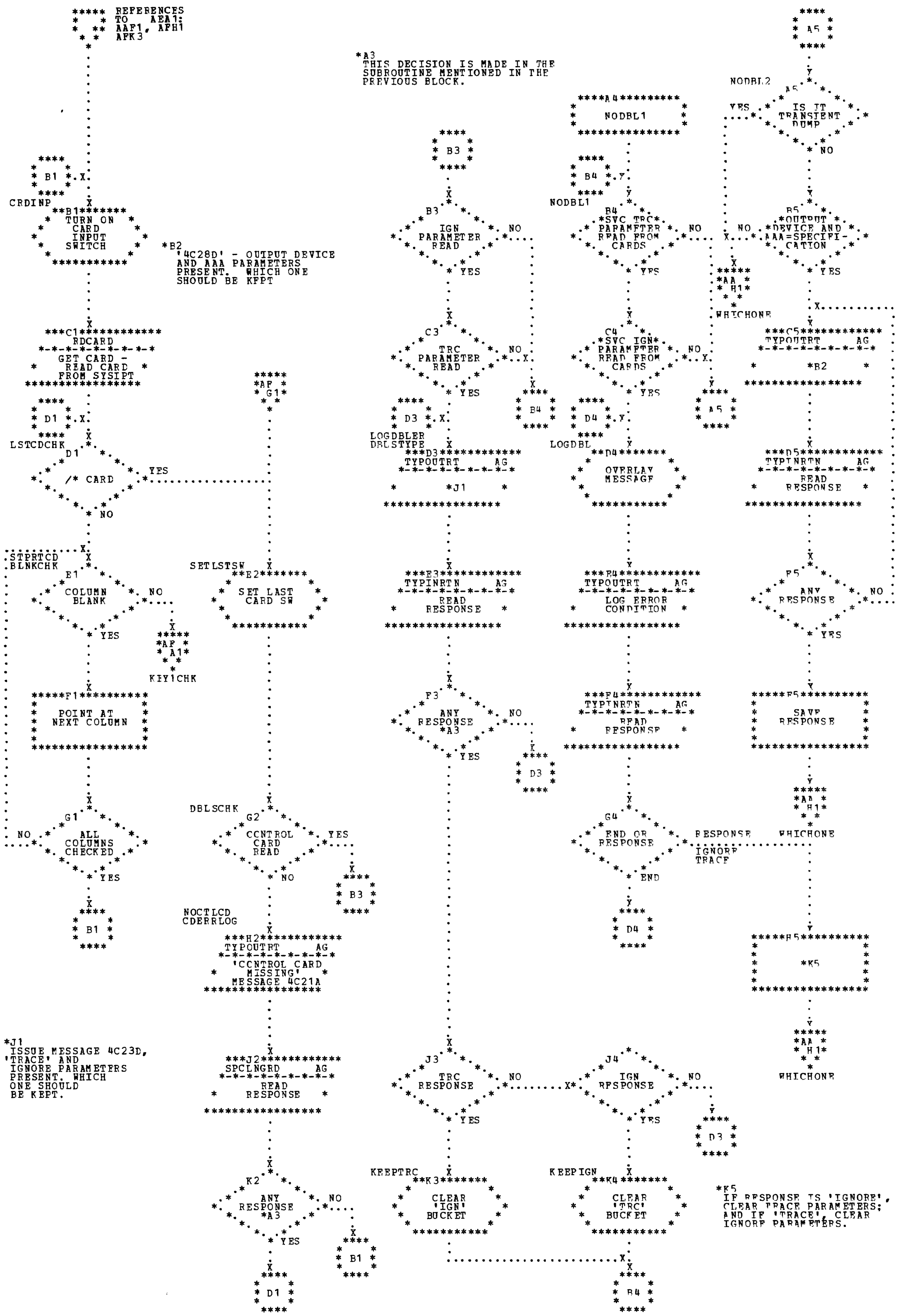

REFERENCES TO ADB1: ABE1, ABF2
CKOUTOD
B1 PARAMCNT AL INTERRUPT RESPONSE
C1 ERRORS
NO
YES
D1 *H1
YES
NO
E1 ODERRCD AK ERROR ROUTINE
F1 TYPOUTRT AG ISSUE MESSAGE *H1
AB E2
CUURESP
D2 DEVICE SUPPORTED
E2 SET ALL PARAMETERS RECEIVED SWITCH -SWITCH-
F2 TRANSIENT DUMP
G5
G2 PUT DEVICE INDICATOR IN PHASE NAME
AC A5
OPTIONS
A3 INITRTN
INITRTN
B3 BUILD SYSLOG AND SYSRDR CCW'S
C3 POINT TO COMMUNICATION REGION
D3 ONE PARTITION SYSTEM
E3 TURN ON- MULTI- PROGRAMMING SWITCH
F3 GET ADDRESS OF SYSCOM
AROUND13
G3 FIND PDAREA ADDRESS
C5
H3 SVC 6 CANCEL
C5
C5 CLEAR AREA USED BY INITIALIZER AND POINT TO COMREG
D5 SAVE SYSLOG 'CUU'
E5 RETURN TO CALLING ROUTINE
G5
TDGO
G5 CARD INPUT
H5
H5 TYPINRTN AG GET RESPONSE
J5 ANY RESPONSE
AH F4
K5 TYPOUTRT AG ISSUE MSG 4C27D 'INVALID PARAM SPECIFD'
H5
*H1
IF DELIMITER/CHARACTER IS IN ERROR OR DEVICE IS NOT SUPPORTED, THEN ISSUE 'INVALID PARAMETER SPECIFIED', MESSAGE 4C17D
IF THE DEVICE IS NOT IN THE PUB TABLE, THEN ISSUE 'CUU NOT IN PUB TABLE', MESSAGE 4C20D

Chart AE. PDAID Initializer: Card Input Routine



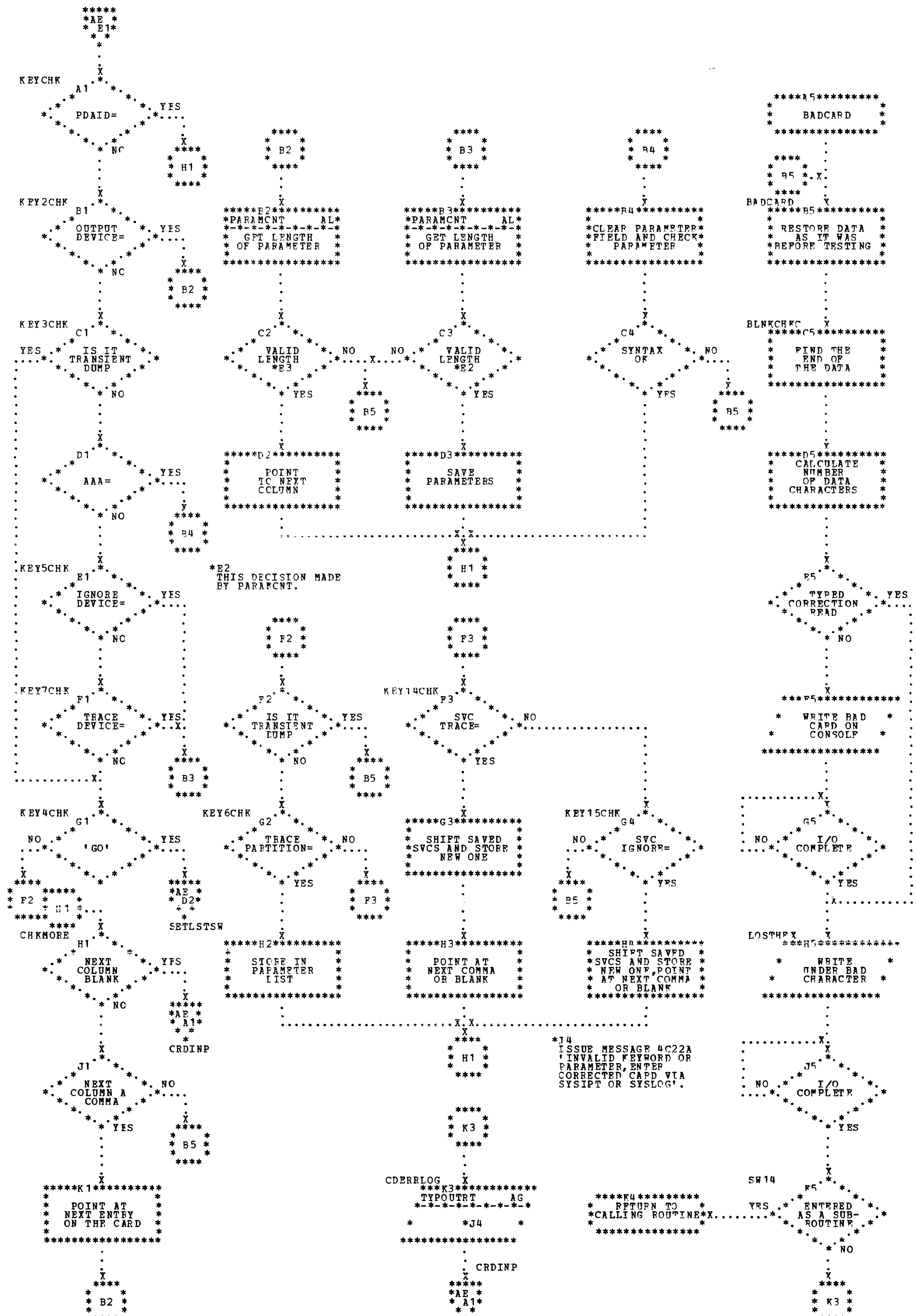

KEYCHK
A1 PDAID=
KEY2CHK
B1 OUTPUT DEVICE=
KEY3CHK
C1 IS IT TRANSIENT DUMP
D1 AAA=
KEY5CHK
E1 IGNORE DEVICE=
KEY7CHK
F1 TRACE DEVICE=
KEY4CHK
G1 'GO'
SETLSTSW
CHKMORE
H1 NEXT COLUMN BLANK
CRDINP
J1 NEXT COLUMN A COMMA
K1 POINT AT NEXT ENTRY ON THE CARD
B2 PARAMCNT AL GET LENGTH OF PARAMETER
C2 VALID LENGTH *E3
D2 POINT TO NEXT COLUMN
*E2 THIS DECISION MADE BY PARAMCNT.
F2 IS IT TRANSIENT DUMP
KEY6CHK
G2 TRACE PARTITION=
H2 STORE IN PARAMETER LIST
B3 PARAMCNT AL GET LENGTH OF PARAMETER
C3 VALID LENGTH *E2
D3 SAVE PARAMETERS
KEY14CHK
F3 SVC TRACE=
G3 SHIFT SAVED SVCS AND STORE NEW ONE
H3 POINT AT NEXT COMMA OR BLANK
CDERRLOG
K3 TYPOUTRT AG *J4
CRDINP
B4 CLEAR PARAMETER FIELD AND CHECK PARAMETER
C4 SYNTAX OK
KEY15CHK
G4 SVC IGNORE=
H4 SHIFT SAVED SVCS AND STORE NEW ONE,POINT AT NEXT COMMA OR BLANK
*J4 ISSUE MESSAGE 4C22A 'INVALID KEYWORD OR PARAMETER,ENTER CORRECTED CARD VIA SYSIPT OR SYSLOG'.
K4 RETURN TO CALLING ROUTINE
A5 BADCARD
BADCARD
B5 RESTORE DATA AS IT WAS BEFORE TESTING
BLNKCHEC
C5 FIND THE END OF THE DATA
D5 CALCULATE NUMBER OF DATA CHARACTERS
E5 TYPED CORRECTION READ
F5 WRITE BAD CARD ON CONSOLE
G5 I/O COMPLETE
LOSTHEX
H5 WRITE UNDER BAD CHARACTER
J5 I/O COMPLETE
SW14
K5 ENTERED AS A SUB-ROUTINE

Chart AG. PDAID Initializer: Console Input/Output Routines

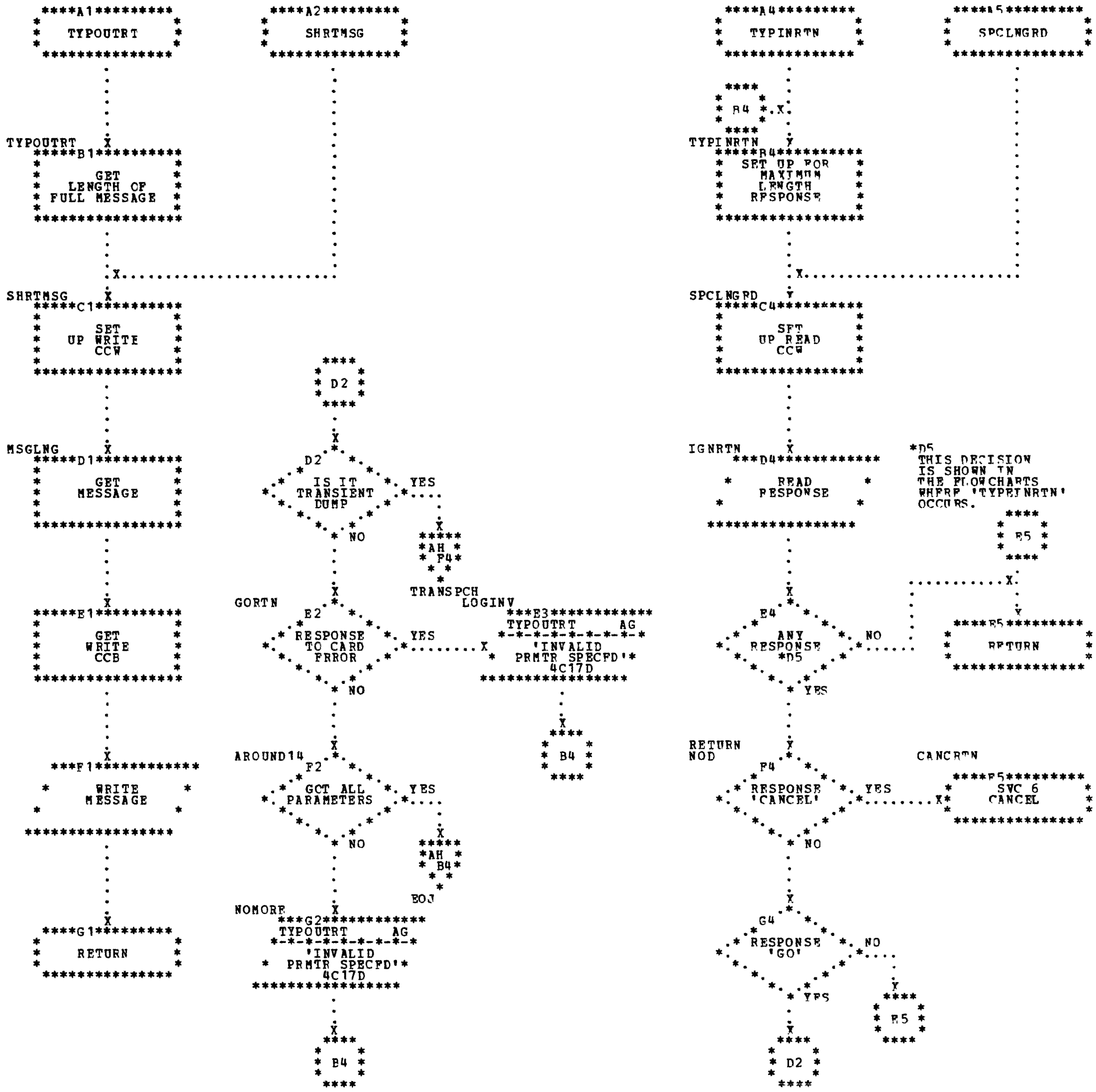


Chart AH. PDAID Initializer: Get IGN/TRC Parameters EOJ Routine; Check Display Limits (Part 1 of 2)

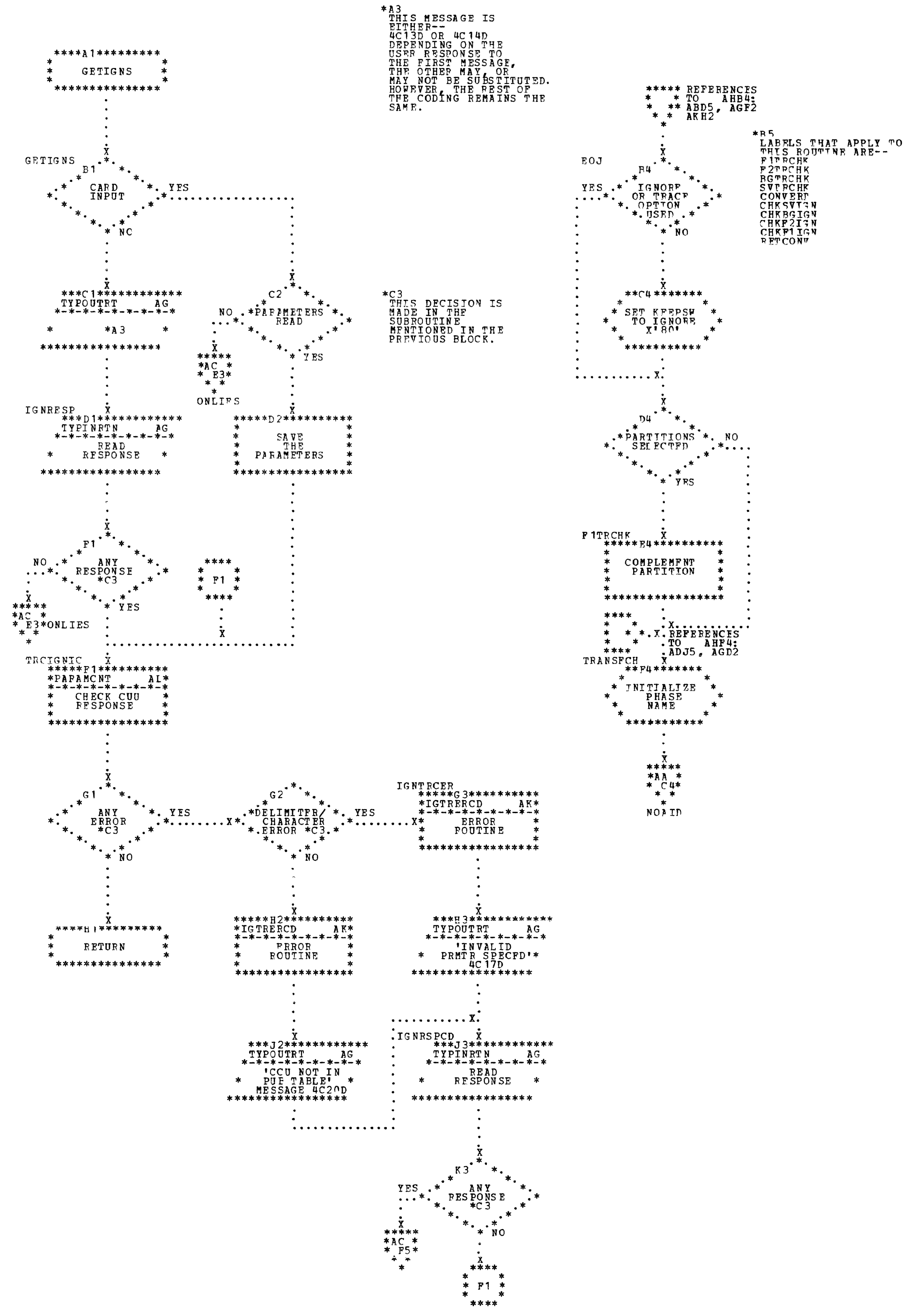
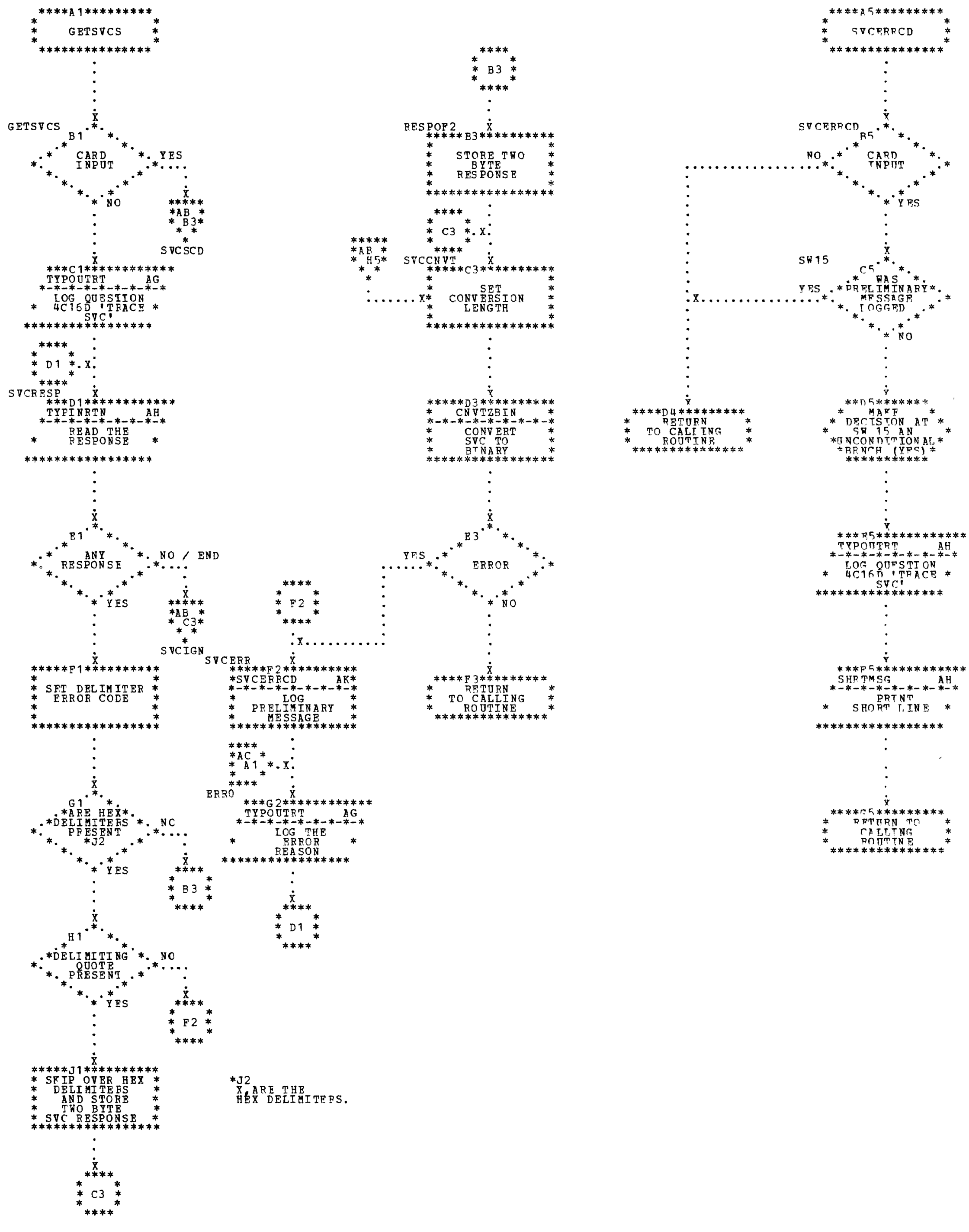


Chart AJ. PDAID Initializer: Get IGN/TRC Parameters ECJ Routine; Check Display Limits (Part 2 of 2)



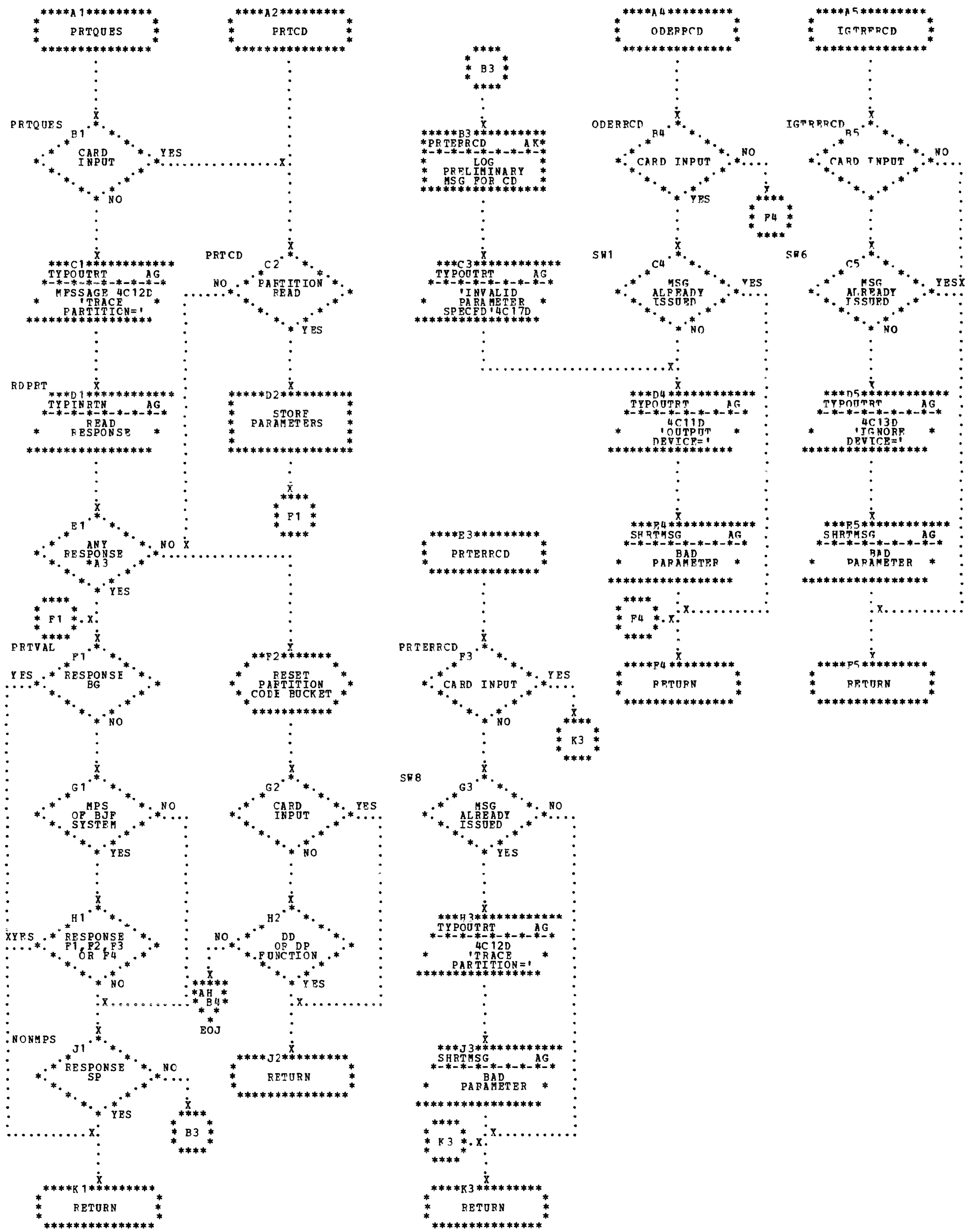

PRTQUES
PRTCD
ODERRCD
IGTRFRCD
B3
PRTERRCD
LOG PRELIMINARY MSG FOR CD
CARD INPUT
YES
NO
PARTITION READ
TYPOUTRT AG
MESSAGE 4C12D 'TRACE PARTITION='
INVALID PARAMETER SPECFD'4C17D
MSG ALREADY ISSUED
RDPRT
TYPINRTN AG
READ RESPONSE
STORE PARAMETERS
4C11D 'OUTPUT DEVICE='
4C13D 'IGNORE DEVICE='
ANY RESPONSE *A3
F1
SHRTMSG AG
BAD PARAMETER
P4
PRTVAL
RESPONSE BG
RESET PARTITION CODE BUCKET
RETURN
K3
MPS OF BJF SYSTEM
SW8
SW1
SW6
DD OF DP FUNCTION
RESPONSE F1,F2,F3 OR F4
AH B4
EOJ
NONMPS
RESPONSE SP
4C12D 'TRACE PARTITION='

Chart AL. PDAID Initializer: Check Parameters, Devices, and Convert to Binary

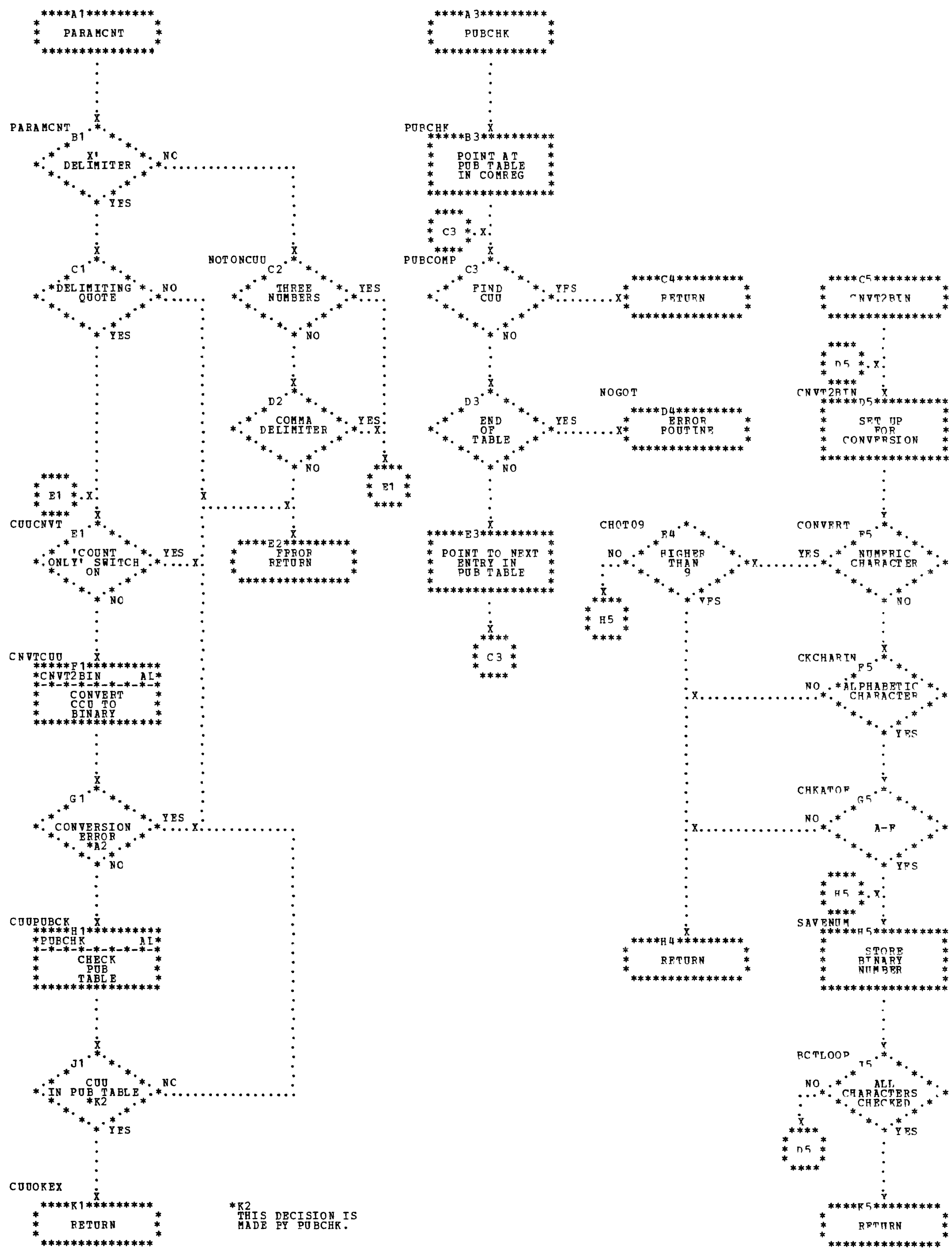
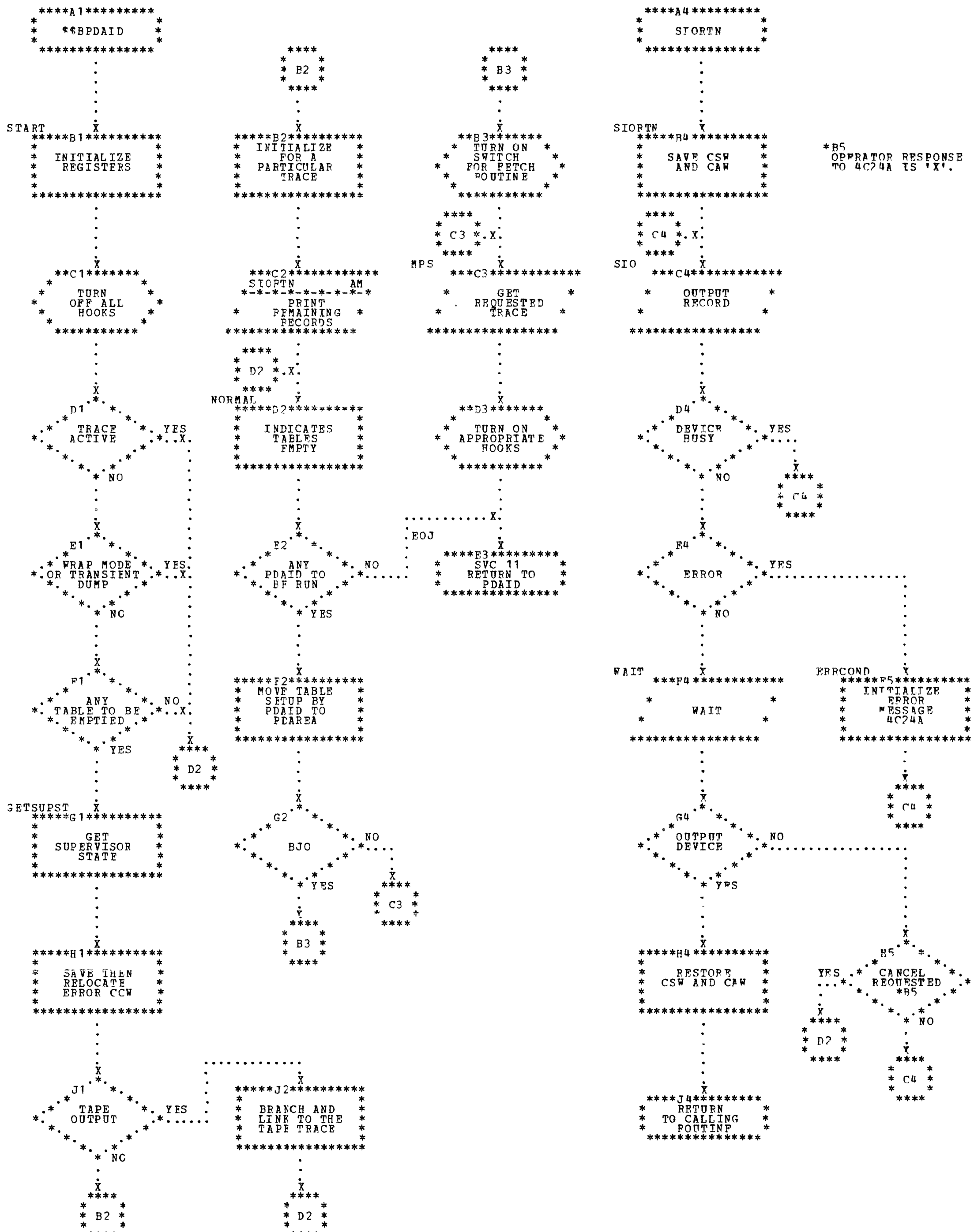
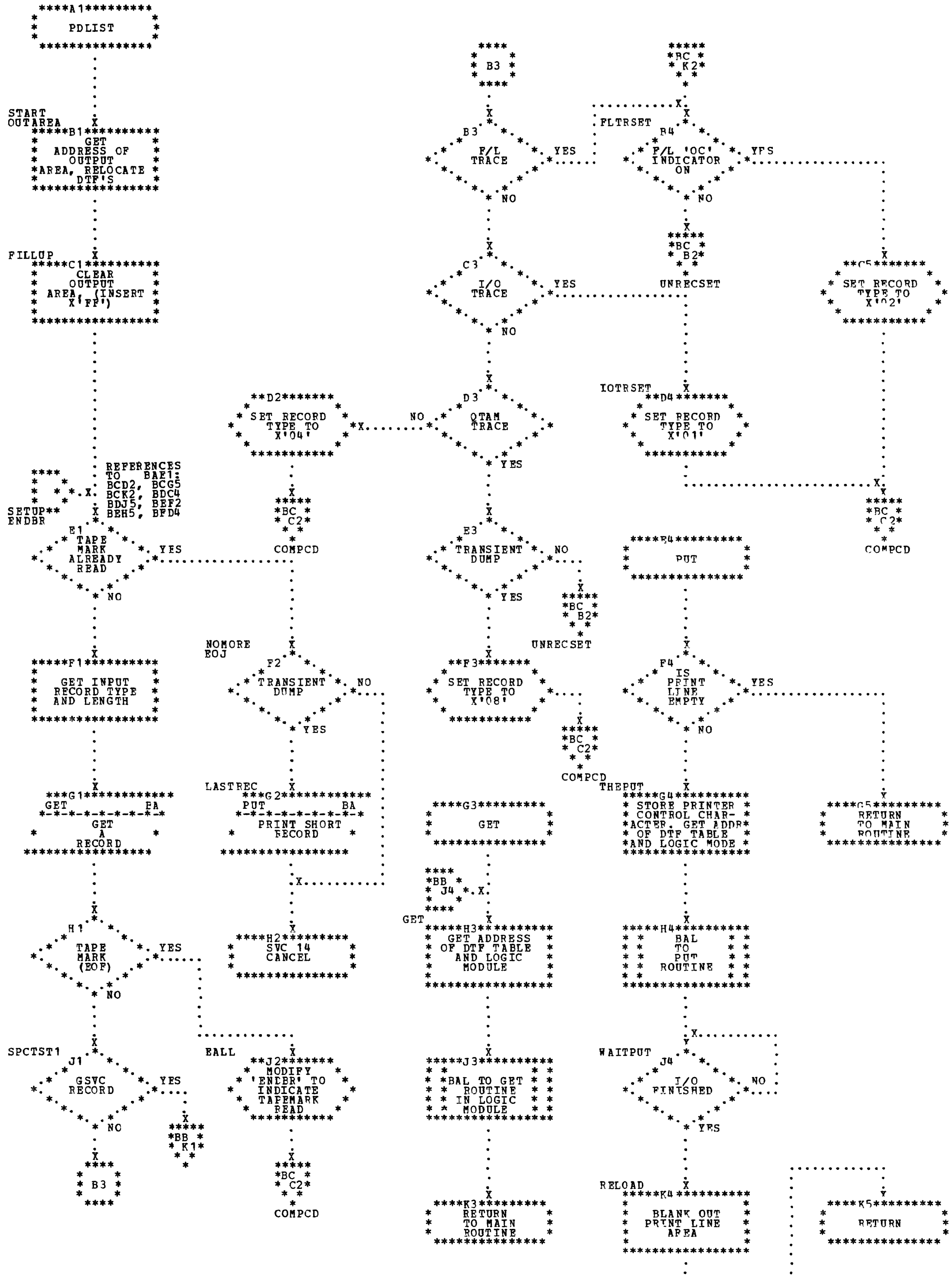
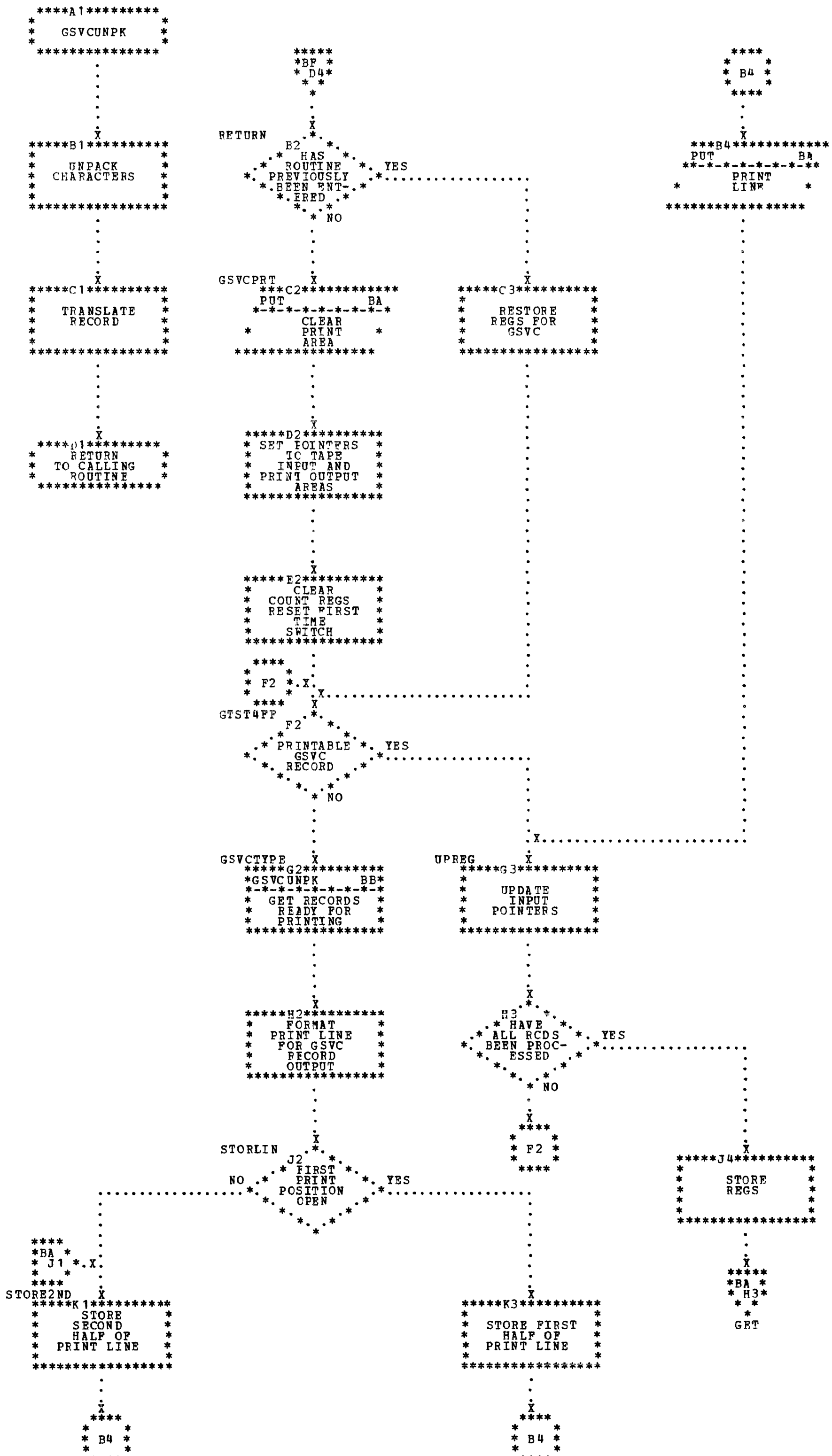
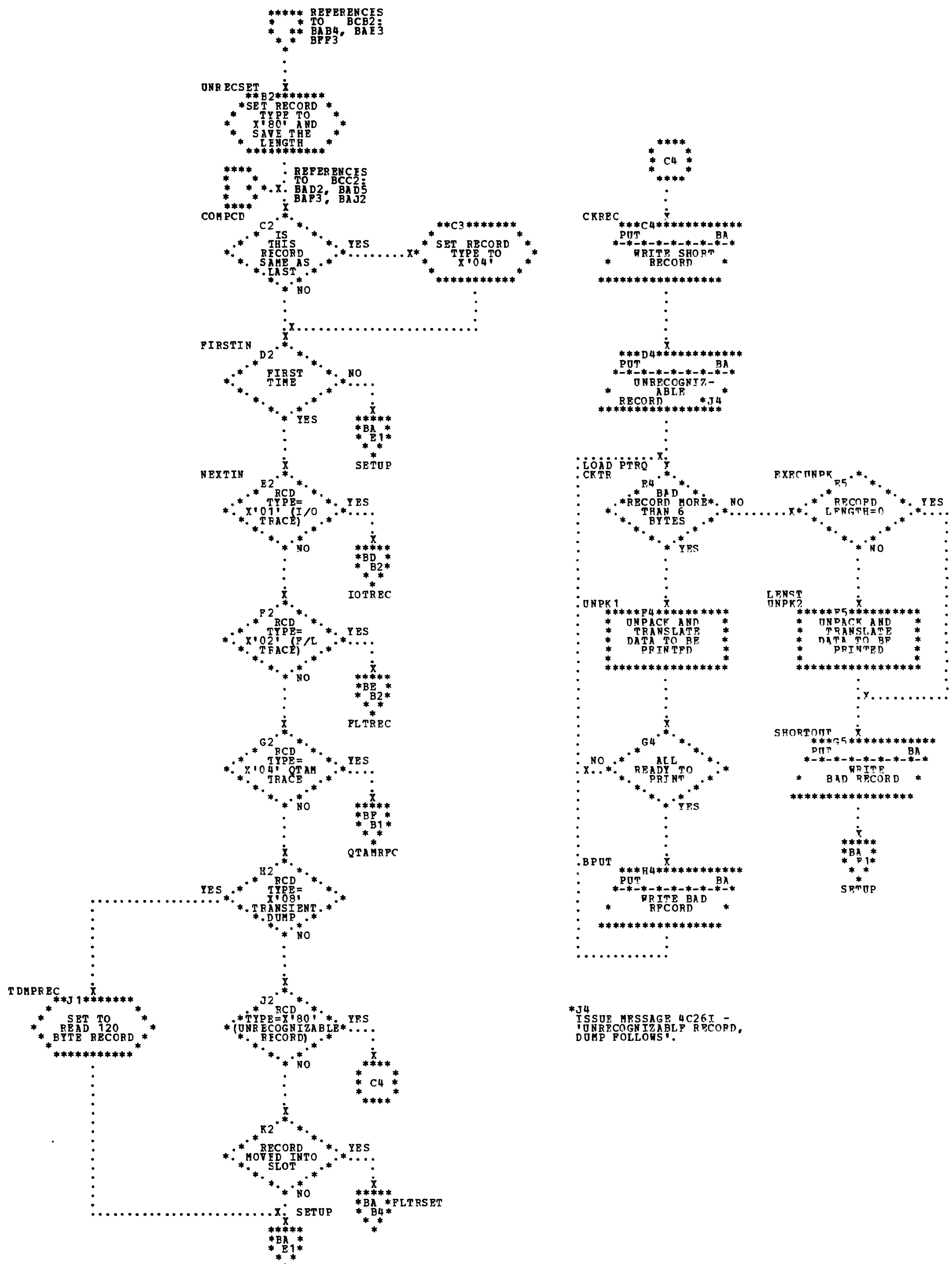


Chart AM. $$BPDAID - PDAID Transient Routine




A1
PDLIST
START
OUTAREA
B1
GET ADDRESS OF OUTPUT AREA, RELOCATE DTF'S
FILLUP
C1
CLEAR OUTPUT AREA, (INSERT X'FF')
REFERENCES TO BAE1: BCD2, BCG5 BCK2, BDC4 BDJ5, BEF2 BEH5, BFD4
SETUP
ENDBR
E1
TAPE MARK ALREADY READ
YES
NO
F1
GET INPUT RECORD TYPE AND LENGTH
G1
GET BA
GET A RECORD
H1
TAPE MARK (EOF)
YES
NO
SPCTST1
J1
GSVC RECORD
YES
NO
BB K1
B3
D2
SET RECORD TYPE TO X'04'
BC C2
COMPCD
NOMORE
EOJ
F2
TRANSIENT DUMP
NO
YES
LASTREC
G2
PUT BA
PRINT SHORT RECORD
H2
SVC 14 CANCEL
BALL
J2
MODIFY 'ENDBR' TO INDICATE TAPEMARK READ
BC C2
COMPCD
B3
B3
F/L TRACE
YES
NO
C3
I/O TRACE
YES
NO
D3
QTAM TRACE
NO
YES
E3
TRANSIENT DUMP
NO
YES
BC B2
UNRECSET
F3
SET RECORD TYPE TO X'08'
BC C2
COMPCD
G3
GET
BB J4
GET
H3
GET ADDRESS OF DTF TABLE AND LOGIC MODULE
J3
BAL TO GET ROUTINE IN LOGIC MODULE
K3
RETURN TO MAIN ROUTINE
BC K2
FLTRSET
B4
F/L 'OC' INDICATOR ON
YES
NO
BC B2
UNRECSET
IOTRSET
D4
SET RECORD TYPE TO X'01'
E4
PUT
F4
IS PRINT LINE EMPTY
YES
NO
THEPUT
G4
STORE PRINTER CONTROL CHARACTER. GET ADDR OF DTF TABLE AND LOGIC MODE
H4
BAL TO PUT ROUTINE
WAITPUT
J4
I/O FINISHED
NO
YES
RELOAD
K4
BLANK OUT PRINT LINE AREA
C5
SET RECORD TYPE TO X'02'
BC C2
COMPCD
G5
RETURN TO MAIN ROUTINE
K5
RETURN


A1 GSVCUNPK
B1 UNPACK CHARACTERS
C1 TRANSLATE RECORD
D1 RETURN TO CALLING ROUTINE
BB D4
RETURN B2 HAS ROUTINE PREVIOUSLY BEEN ENT-ERED
YES
NO
GSVCPRT C2 PUT BA CLEAR PRINT AREA
C3 RESTORE REGS FOR GSVC
D2 SET POINTERS TO TAPE INPUT AND PRINT OUTPUT AREAS
E2 CLEAR COUNT REGS RESET FIRST TIME SWITCH
F2
GTST4FF F2 PRINTABLE GSVC RECORD
GSVCTYPE G2 GSVCUNPK BB GET RECORDS READY FOR PRINTING
H2 FORMAT PRINT LINE FOR GSVC RECORD OUTPUT
STORLIN J2 FIRST PRINT POSITION OPEN
BA J1
STORE2ND K1 STORE SECOND HALF OF PRINT LINE
B4
UPREG G3 UPDATE INPUT POINTERS
H3 HAVE ALL RCDS BEEN PROC-ESSED
F2
K3 STORE FIRST HALF OF PRINT LINE
B4
B4
B4 PUT BA PRINT LINE
J4 STORE REGS
BA H3
GET


REFERENCES TO BCB2: BAB4, BAE3 BFF3
UNRECSET B2 SET RECORD TYPE TO X'80' AND SAVE THE LENGTH
REFERENCES TO BCC2: BAD2, BAD5 BAF3, BAJ2
COMPCD C2 IS THIS RECORD SAME AS LAST
YES
NO
C3 SET RECORD TYPE TO X'04'
FIRSTIN D2 FIRST TIME
NO
YES
BA E1 SETUP
NEXTIN E2 RCD TYPE= X'01' (I/O TRACE)
BD B2 IOTREC
F2 RCD TYPE= X'02' (F/L TRACE)
BE B2 FLTREC
G2 RCD TYPE= X'04' QTAM TRACE
BF B1 QTAMREC
H2 RCD TYPE= X'08' TRANSIENT DUMP
TDMPREC J1 SET TO READ 120 BYTE RECORD
J2 RCD TYPE=X'80' (UNRECOGNIZABLE RECORD)
C4
K2 RECORD MOVED INTO SLOT
BA B4 FLTRSET
SETUP
BA E1
C4
CKREC C4 PUT BA WRITE SHORT RECORD
D4 PUT BA UNRECOGNIZ- ABLE RECORD *J4
LOAD PTRQ CKTR
E4 BAD RECORD MORE THAN 6 BYTES
EXECUNPK E5 RECORD LENGTH=0
UNPK1 F4 UNPACK AND TRANSLATE DATA TO BE PRINTED
LENST UNPK2 F5 UNPACK AND TRANSLATE DATA TO BE PRINTED
G4 ALL READY TO PRINT
SHORTOUT G5 PUT BA WRITE BAD RECORD
BA F1 SETUP
BPUT H4 PUT BA WRITE BAD RECORD
*J4 ISSUE MESSAGE 4C26I - 'UNRECOGNIZABLE RECORD, DUMP FOLLOWS'.

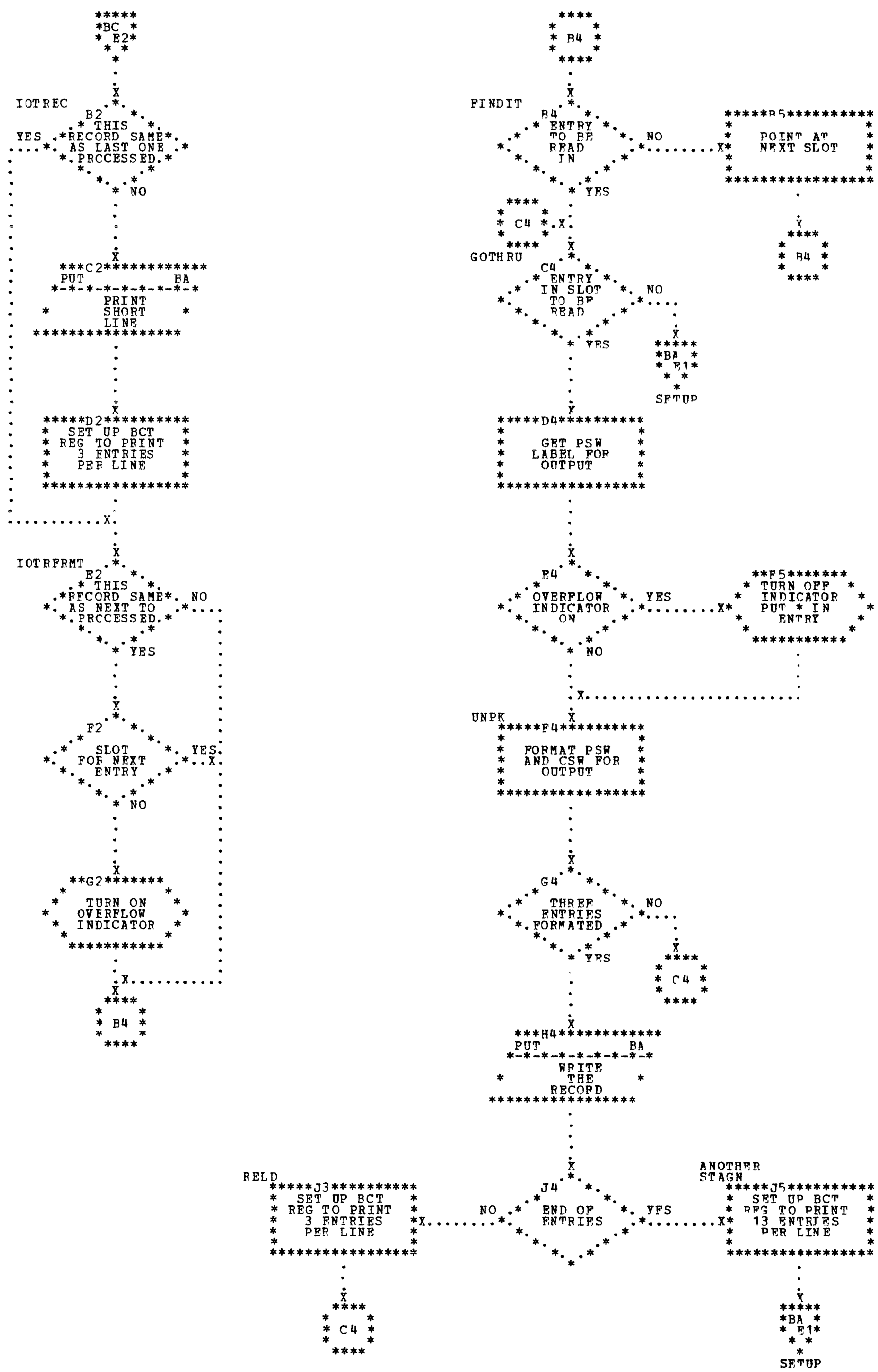
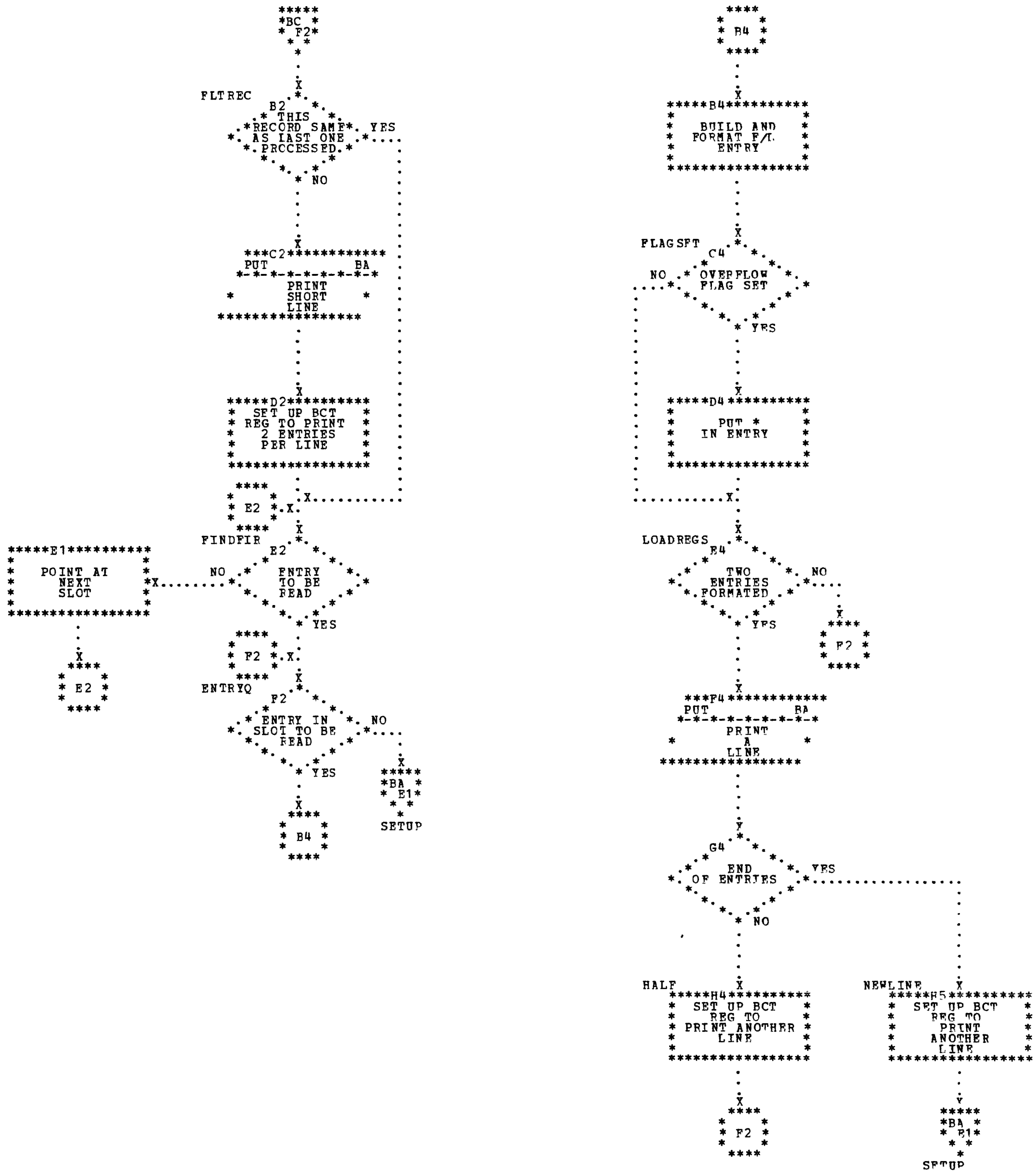
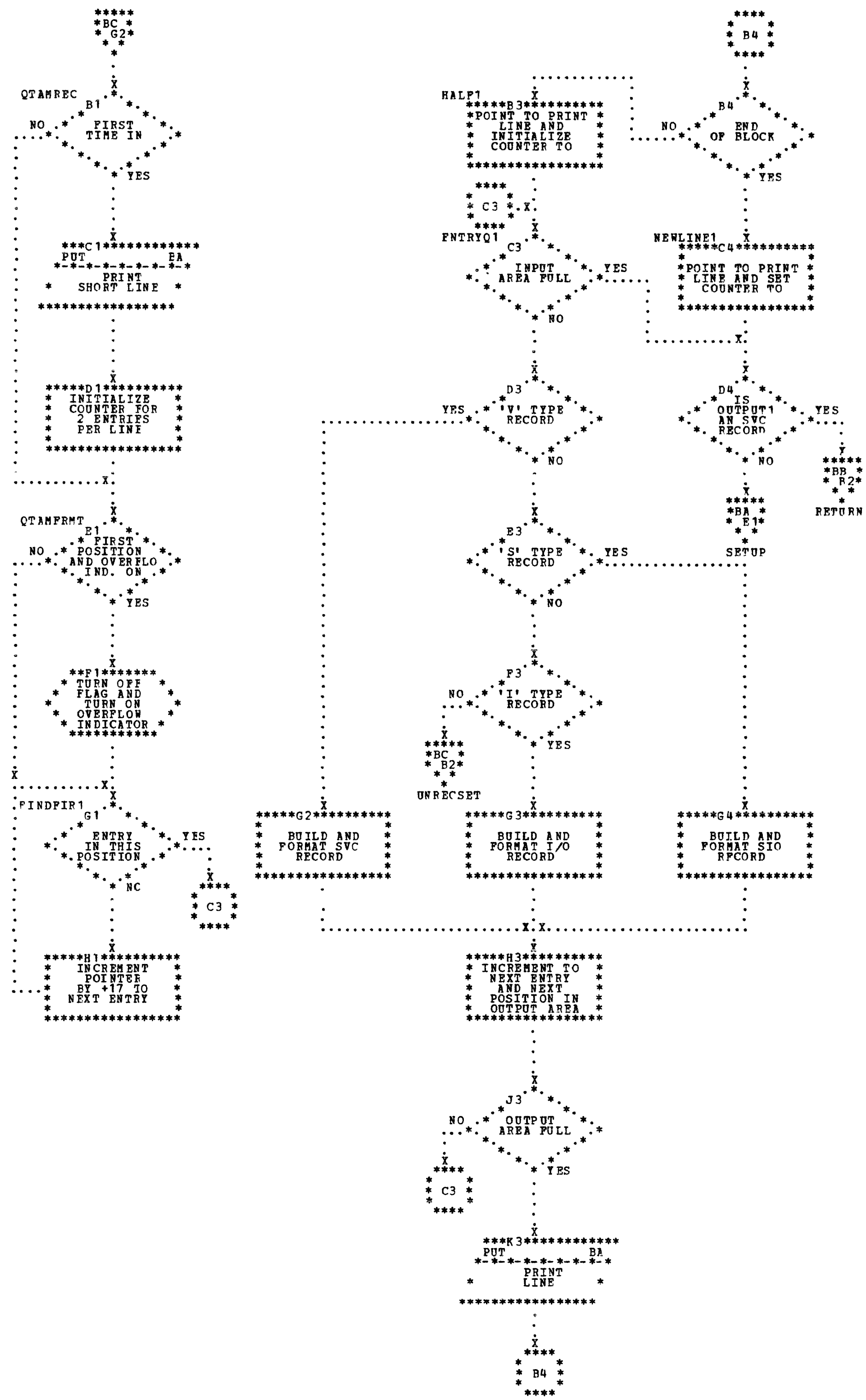

BC B2
IOTREC
B2 THIS RECORD SAME AS LAST ONE PROCESSED
YES
NO
C2 PUT BA PRINT SHORT LINE
D2 SET UP BCT REG TO PRINT 3 ENTRIES PER LINE
IOTRFRMT
E2 THIS RECORD SAME AS NEXT TO BE PROCESSED
NO
YES
F2 SLOT FOR NEXT ENTRY
YES
NO
G2 TURN ON OVERFLOW INDICATOR
B4
B4
FINDIT
B4 ENTRY TO BE READ IN
NO
YES
B5 POINT AT NEXT SLOT
B4
C4
GOTHRU
C4 ENTRY IN SLOT TO BE READ
NO
YES
BA B1 SETUP
D4 GET PSW LABEL FOR OUTPUT
E4 OVERFLOW INDICATOR ON
YES
NO
E5 TURN OFF INDICATOR PUT * IN ENTRY
UNPK
F4 FORMAT PSW AND CSW FOR OUTPUT
G4 THREE ENTRIES FORMATED
NO
YES
C4
H4 PUT BA WRITE THE RECORD
RELD
J3 SET UP BCT REG TO PRINT 3 ENTRIES PER LINE
NO
J4 END OF ENTRIES
YES
ANOTHER STAGN
J5 SET UP BCT REG TO PRINT 13 ENTRIES PER LINE
C4
BA B1 SETUP

Chart BE. PDLIST - F/L Trace Records




BC G2
QTAMREC B1
FIRST TIME IN
NO
YES
C1
PUT BA
PRINT SHORT LINE
D1
INITIALIZE COUNTER FOR 2 ENTRIES PER LINE
QTAMFRMT E1
FIRST POSITION AND OVERFLO IND. ON
F1
TURN OFF FLAG AND TURN ON OVERFLOW INDICATOR
FINDFIR1 G1
ENTRY IN THIS POSITION
H1
INCREMENT POINTER BY +17 TO NEXT ENTRY
C3
HALF1 B3
POINT TO PRINT LINE AND INITIALIZE COUNTER TO
ENTRYQ1 C3
INPUT AREA FULL
D3
'V' TYPE RECORD
E3
'S' TYPE RECORD
F3
'I' TYPE RECORD
BC B2
UNRECSET
G2
BUILD AND FORMAT SVC RECORD
G3
BUILD AND FORMAT I/O RECORD
H3
INCREMENT TO NEXT ENTRY AND NEXT POSITION IN OUTPUT AREA
J3
OUTPUT AREA FULL
K3
PUT BA
PRINT LINE
B4
END OF BLOCK
NEWLINE1 C4
POINT TO PRINT LINE AND SET COUNTER TO
D4
IS OUTPUT1 AN SVC RECORD
BB R2
RETURN
BA E1
SETUP
G4
BUILD AND FORMAT SIO RECORD

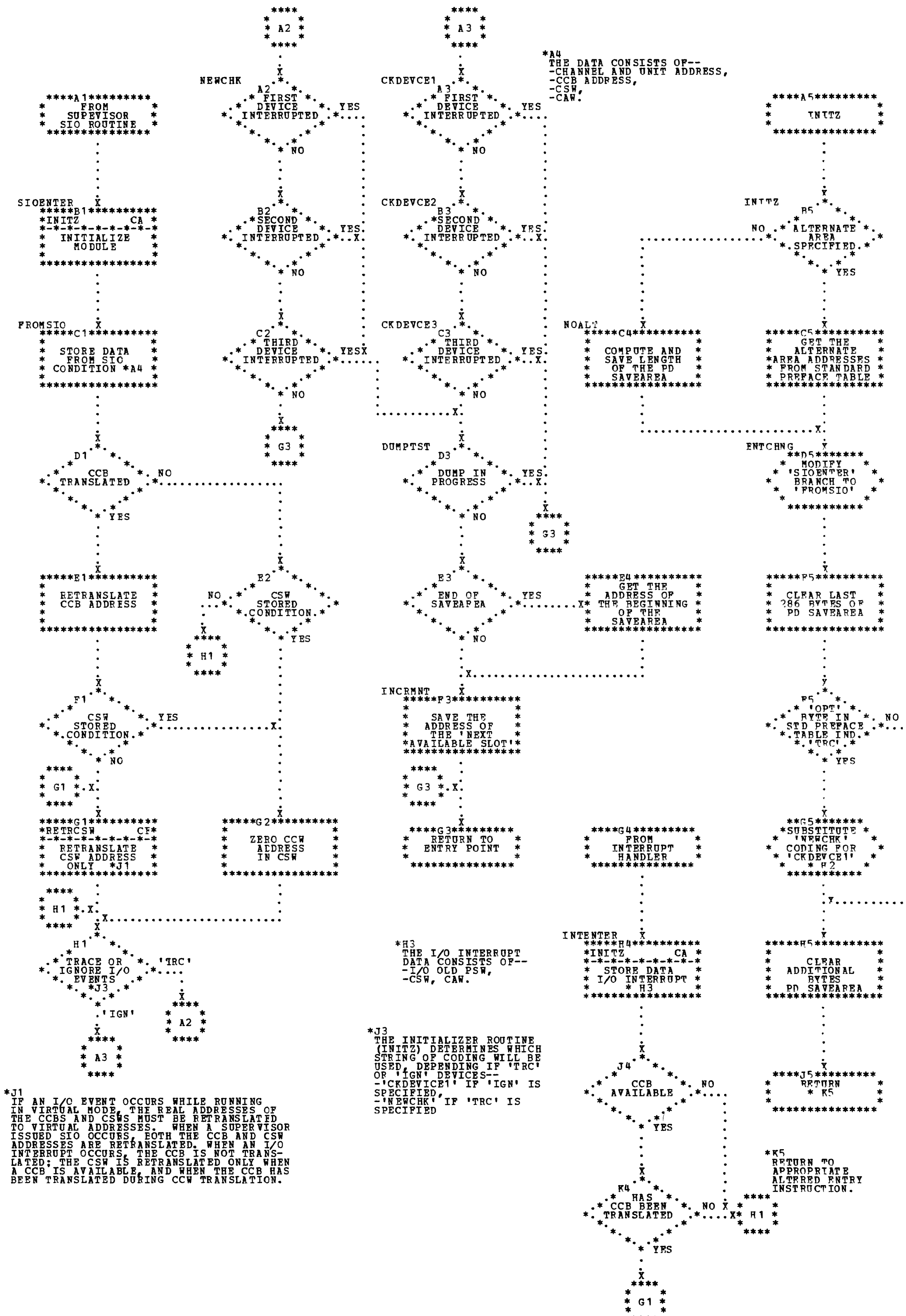
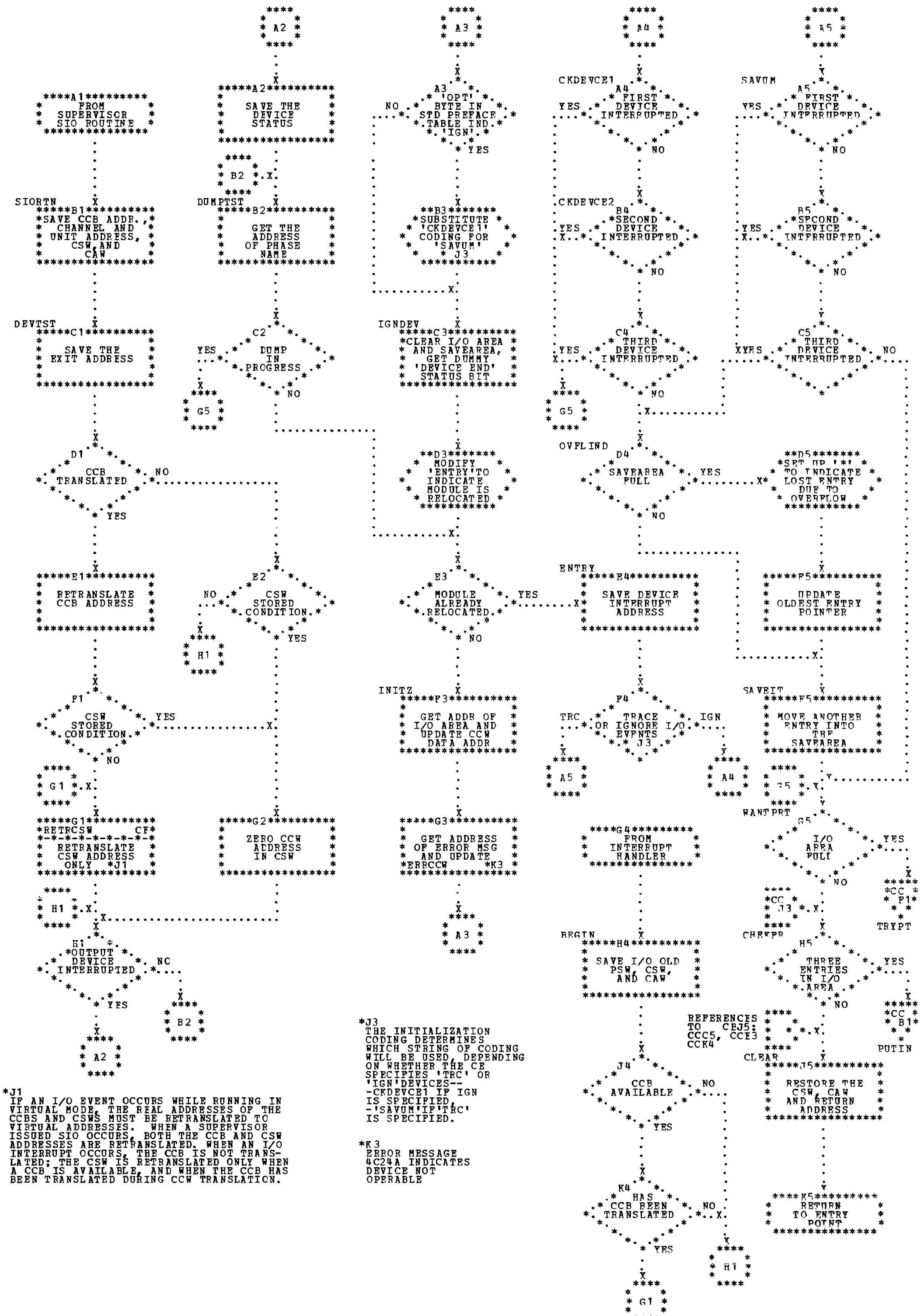

A2
A3
*A4
THE DATA CONSISTS OF--
-CHANNEL AND UNIT ADDRESS,
-CCB ADDRESS,
-CSW,
-CAW.
A1 FROM SUPERVISOR SIO ROUTINE
NEWCHK A2 FIRST DEVICE INTERRUPTED YES NO
CKDEVCE1 A3 FIRST DEVICE INTERRUPTED YES NO
A5 INITZ
SIOENTER B1 INITZ CA INITIALIZE MODULE
B2 SECOND DEVICE INTERRUPTED YES NO
CKDEVCE2 B3 SECOND DEVICE INTERRUPTED YES NO
INITZ B5 ALTERNATE AREA SPECIFIED NO YES
FROMSIO C1 STORE DATA FROM SIO CONDITION *A4
C2 THIRD DEVICE INTERRUPTED YES NO
CKDEVCE3 C3 THIRD DEVICE INTERRUPTED YES NO
NOALT C4 COMPUTE AND SAVE LENGTH OF THE PD SAVEAREA
C5 GET THE ALTERNATE AREA ADDRESSES FROM STANDARD PREFACE TABLE
G3
D1 CCB TRANSLATED NO YES
DUMPTST D3 DUMP IN PROGRESS YES NO
G3
ENTCHNG D5 MODIFY 'SIOENTER' BRANCH TO 'FROMSIO'
E1 RETRANSLATE CCB ADDRESS
E2 CSW STORED CONDITION NO YES
H1
E3 END OF SAVEAREA YES NO
E4 GET THE ADDRESS OF THE BEGINNING OF THE SAVEAREA
E5 CLEAR LAST 286 BYTES OF PD SAVEAREA
F1 CSW STORED CONDITION YES NO
INCRMNT F3 SAVE THE ADDRESS OF THE 'NEXT AVAILABLE SLOT'
F5 'OPT' BYTE IN STD PREFACE TABLE IND. 'TRC' NO YES
G1
G3
G1 RETRCSW CF RETRANSLATE CSW ADDRESS ONLY *J1
G2 ZERO CCW ADDRESS IN CSW
G3 RETURN TO ENTRY POINT
G4 FROM INTERRUPT HANDLER
G5 SUBSTITUTE 'NEWCHK' CODING FOR 'CKDEVCE1' *H2
H1
H1 TRACE OR IGNORE I/O EVENTS *J3 'TRC' 'IGN'
A2
A3
*H3
THE I/O INTERRUPT DATA CONSISTS OF--
-I/O OLD PSW,
-CSW, CAW.
INTENTER H4 INITZ CA STORE DATA I/O INTERRUPT *H3
H5 CLEAR ADDITIONAL BYTES PD SAVEAREA
*J3
THE INITIALIZER ROUTINE (INITZ) DETERMINES WHICH STRING OF CODING WILL BE USED, DEPENDING IF 'TRC' OR 'IGN' DEVICES--
-'CKDEVICE1' IF 'IGN' IS SPECIFIED,
-'NEWCHK' IF 'TRC' IS SPECIFIED
J4 CCB AVAILABLE NO YES
J5 RETURN *K5
*J1
IF AN I/O EVENT OCCURS WHILE RUNNING IN VIRTUAL MODE, THE REAL ADDRESSES OF THE CCBS AND CSWS MUST BE RETRANSLATED TO VIRTUAL ADDRESSES. WHEN A SUPERVISOR ISSUED SIO OCCURS, BOTH THE CCB AND CSW ADDRESSES ARE RETRANSLATED. WHEN AN I/O INTERRUPT OCCURS, THE CCB IS NOT TRANSLATED; THE CSW IS RETRANSLATED ONLY WHEN A CCB IS AVAILABLE, AND WHEN THE CCB HAS BEEN TRANSLATED DURING CCW TRANSLATION.
*K5
RETURN TO APPROPRIATE ALTERED ENTRY INSTRUCTION.
K4 HAS CCB BEEN TRANSLATED NO YES
H1
G1

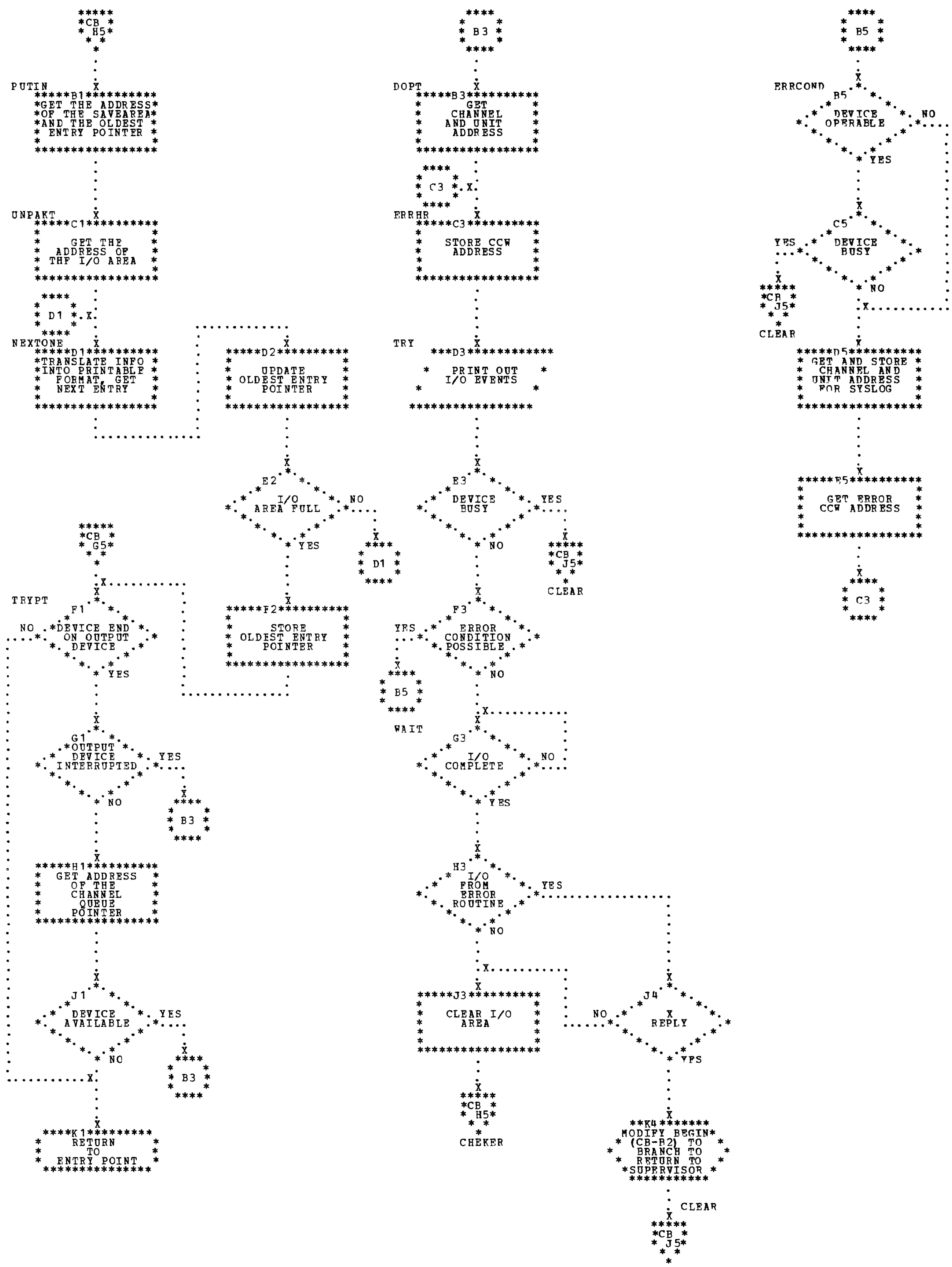
Chart CB. I/O Trace: Print Mode (PDAIDITP) (Part 1 of 2)



*J1
IF AN I/O EVENT OCCURS WHILE RUNNING IN VIRTUAL MODE, THE REAL ADDRESSES OF THE CCBS AND CSWS MUST BE RETRANSLATED TO VIRTUAL ADDRESSES. WHEN A SUPERVISOR ISSUED SIO OCCURS, BOTH THE CCB AND CSW ADDRESSES ARE RETRANSLATED. WHEN AN I/O INTERRUPT OCCURS, THE CCB IS NOT TRANSLATED; THE CSW IS RETRANSLATED ONLY WHEN A CCB IS AVAILABLE, AND WHEN THE CCB HAS BEEN TRANSLATED DURING CCW TRANSLATION.

*J3
THE INITIALIZATION CODING DETERMINES WHICH STRING OF CODING WILL BE USED, DEPENDING ON WHETHER THE CE SPECIFIES 'TRC' OR 'IGN' DEVICES--
-CKDEVCE1 IF IGN IS SPECIFIED,
-'SAVUM' IF 'TRC' IS SPECIFIED.

*K3
ERROR MESSAGE 4C24A INDICATES DEVICE NOT OPERABLE


CB H5
PUTIN
B1
GET THE ADDRESS OF THE SAVEAREA AND THE OLDEST ENTRY POINTER
UNPAKT
C1
GET THE ADDRESS OF THE I/O AREA
D1
NEXTONE
D1
TRANSLATE INFO INTO PRINTABLE FORMAT, GET NEXT ENTRY
D2
UPDATE OLDEST ENTRY POINTER
E2
I/O AREA FULL
NO
YES
D1
F2
STORE OLDEST ENTRY POINTER
CB G5
TRYPT
F1
DEVICE END ON OUTPUT DEVICE
NO
YES
G1
OUTPUT DEVICE INTERRUPTED
YES
NO
B3
H1
GET ADDRESS OF THE CHANNEL QUEUE POINTER
J1
DEVICE AVAILABLE
YES
NO
B3
K1
RETURN TO ENTRY POINT
B3
DOPT
B3
GET CHANNEL AND UNIT ADDRESS
C3
ERRHR
C3
STORE CCW ADDRESS
TRY
D3
PRINT OUT I/O EVENTS
E3
DEVICE BUSY
YES
NO
CB J5
CLEAR
F3
ERROR CONDITION POSSIBLE
YES
NO
B5
WAIT
G3
I/O COMPLETE
NO
YES
H3
I/O FROM ERROR ROUTINE
YES
NO
J3
CLEAR I/O AREA
CB H5
CHEKER
J4
REPLY
NO
YES
K4
MODIFY BEGIN (CB-B2) TO BRANCH TO RETURN TO SUPERVISOR
CLEAR
CB J5
B5
ERRCOND
B5
DEVICE OPERABLE
NO
YES
C5
DEVICE BUSY
YES
NO
CB J5
CLEAR
D5
GET AND STORE CHANNEL AND UNIT ADDRESS FOR SYSLOG
E5
GET ERROR CCW ADDRESS
C3

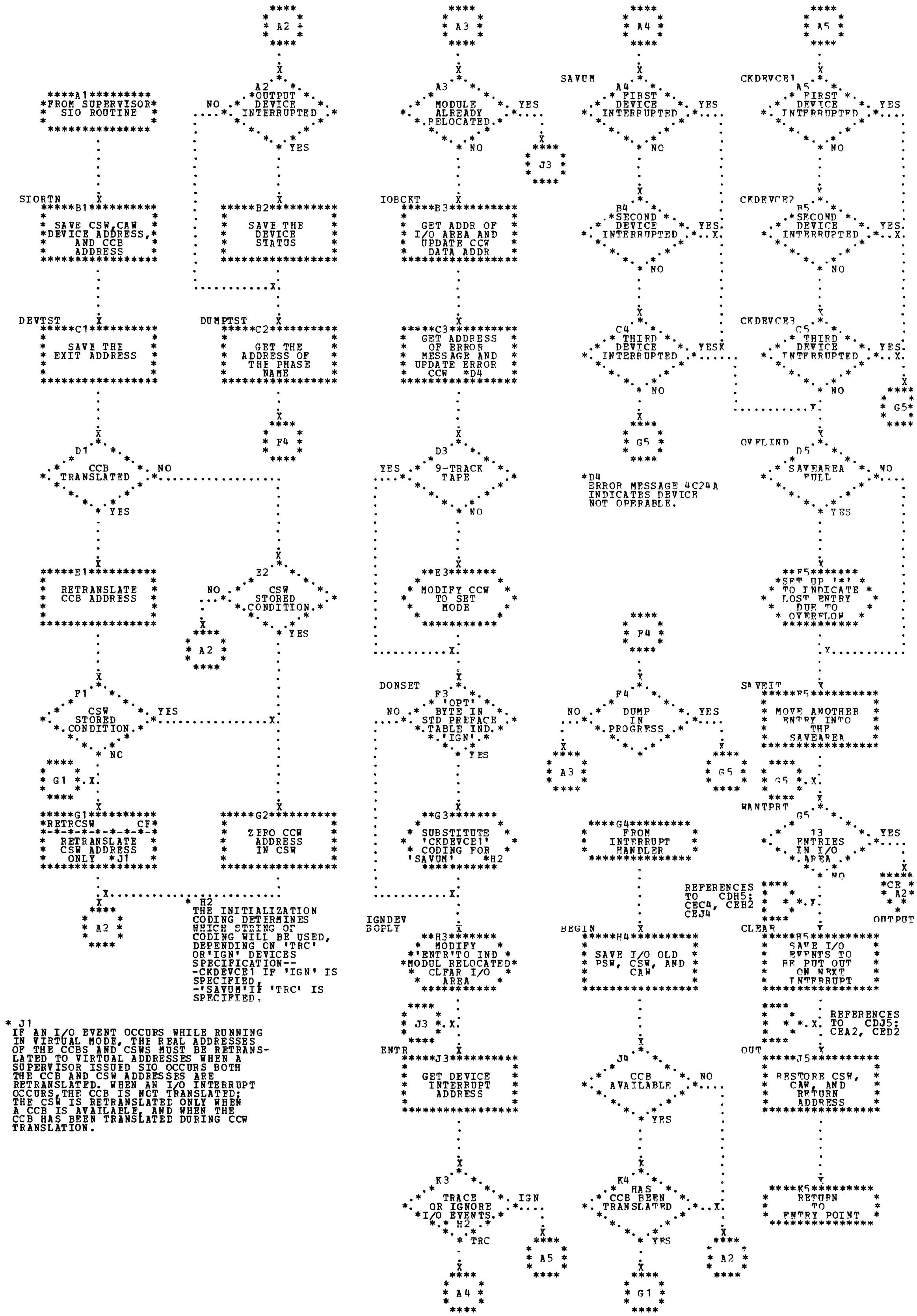

A1 FROM SUPERVISOR SIO ROUTINE
SIORTN B1 SAVE CSW,CAW DEVICE ADDRESS, AND CCB ADDRESS
DEVTST C1 SAVE THE EXIT ADDRESS
D1 CCB TRANSLATED NO YES
E1 RETRANSLATE CCB ADDRESS
F1 CSW STORED CONDITION YES NO
G1 RETRCSW CF RETRANSLATE CSW ADDRESS ONLY *J1
A2
A2 OUTPUT DEVICE INTERRUPTED NO YES
B2 SAVE THE DEVICE STATUS
DUMPTST C2 GET THE ADDRESS OF THE PHASE NAME
F4
E2 CSW STORED CONDITION NO YES
A2
G2 ZERO CCW ADDRESS IN CSW
*H2 THE INITIALIZATION CODING DETERMINES WHICH STRING OF CODING WILL BE USED, DEPENDING ON 'TRC' OR'IGN' DEVICES SPECIFICATION-- -CKDEVCE1 IF 'IGN' IS SPECIFIED. -'SAVUM'IF 'TRC' IS SPECIFIED.
* J1 IF AN I/O EVENT OCCURS WHILE RUNNING IN VIRTUAL MODE, THE REAL ADDRESSES OF THE CCBS AND CSWS MUST BE RETRANSLATED TO VIRTUAL ADDRESSES WHEN A SUPERVISOR ISSUED SIO OCCURS BOTH THE CCB AND CSW ADDRESSES ARE RETRANSLATED. WHEN AN I/O INTERRUPT OCCURS,THE CCB IS NOT TRANSLATED; THE CSW IS RETRANSLATED ONLY WHEN A CCB IS AVAILABLE, AND WHEN THE CCB HAS BEEN TRANSLATED DURING CCW TRANSLATION.
A3
A3 MODULE ALREADY RELOCATED YES NO
J3
IOBCKT B3 GET ADDR OF I/O AREA AND UPDATE CCW DATA ADDR
C3 GET ADDRESS OF ERROR MESSAGE AND UPDATE ERROR CCW *D4
D3 9-TRACK TAPE YES NO
E3 MODIFY CCW TO SET MODE
DONSET F3 'OPT' BYTE IN STD PREFACE TABLE IND. 'IGN' NO YES
G3 SUBSTITUTE 'CKDEVCE1' CODING FOR 'SAVUM' *H2
IGNDEV BOPLT H3 MODIFY 'ENTR'TO IND MODUL RELOCATED CLEAR I/O AREA
J3
ENTR J3 GET DEVICE INTERRUPT ADDRESS
K3 TRACE OR IGNORE I/O EVENTS *H2 IGN TRC
A5
A4
A4
SAVUM A4 FIRST DEVICE INTERRUPTED YES NO
B4 SECOND DEVICE INTERRUPTED YES NO
C4 THIRD DEVICE INTERRUPTED YES NO
G5
*D4 ERROR MESSAGE 4C24A INDICATES DEVICE NOT OPERABLE.
F4
F4 DUMP IN PROGRESS NO YES
A3
G5
G4 FROM INTERRUPT HANDLER
REFERENCES TO CDH5; CEC4, CEH2 CEJ4
BEGIN H4 SAVE I/O OLD PSW, CSW, AND CAW
J4 CCB AVAILABLE NO YES
K4 HAS CCB BEEN TRANSLATED YES
G1
A2
A5
CKDEVCE1 A5 FIRST DEVICE INTERRUPTED YES NO
CKDEVCE2 B5 SECOND DEVICE INTERRUPTED YES NO
CKDEVCE3 C5 THIRD DEVICE INTERRUPTED YES NO
G5
OVFLIND D5 SAVEAREA FULL NO YES
E5 SET UP '*' TO INDICATE LOST ENTRY DUE TO OVERFLOW
SAVEIT F5 MOVE ANOTHER ENTRY INTO THE SAVEAREA
G5
WANTPRT G5 13 ENTRIES IN I/O AREA YES NO
CE A2 OUTPUT
CLEAR H5 SAVE I/O EVENTS TO BE PUT OUT ON NEXT INTERRUPT
REFERENCES TO CDJ5; CEA2, CED2
OUT J5 RESTORE CSW, CAW, AND RETURN ADDRESS
K5 RETURN TO ENTRY POINT

Chart CE. I/O Trace: Tape Mode (PDAIDITT) (Part 2 of 2)

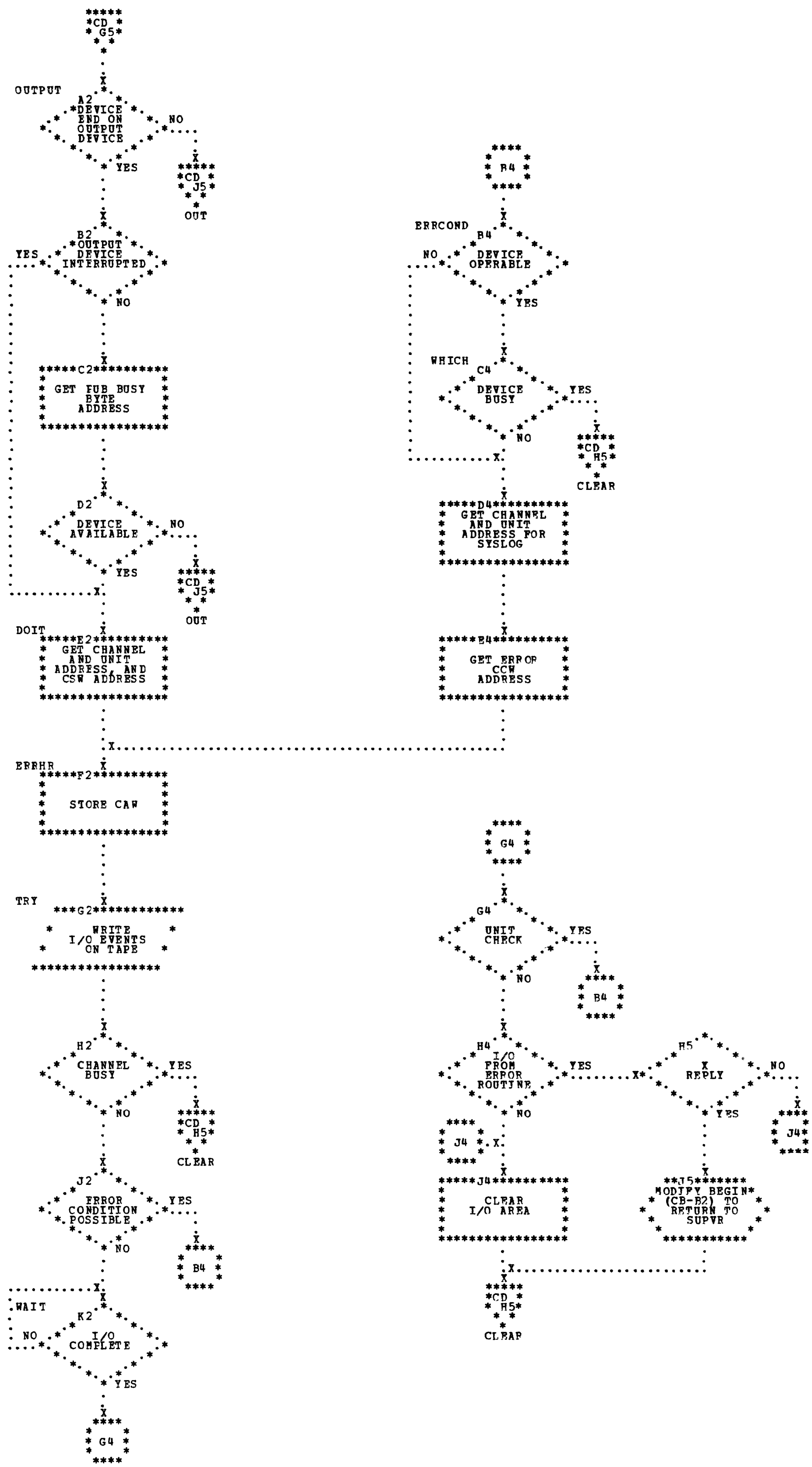


Chart CF. I/O Trace: CSW Retranslate Routine

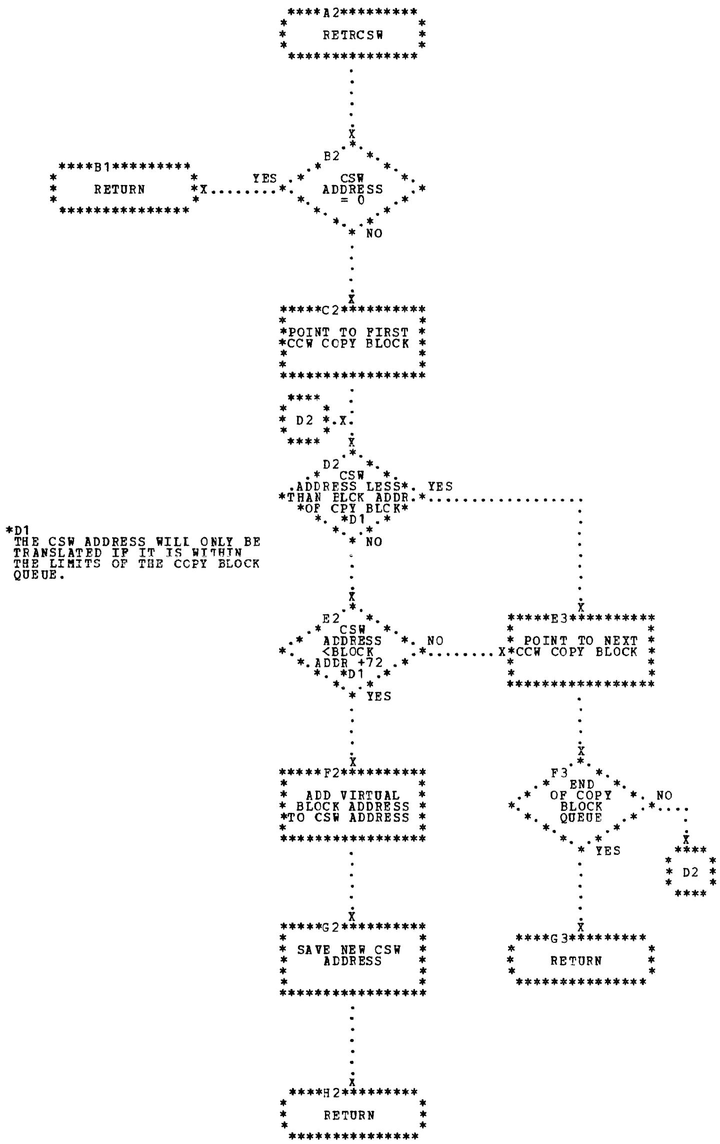
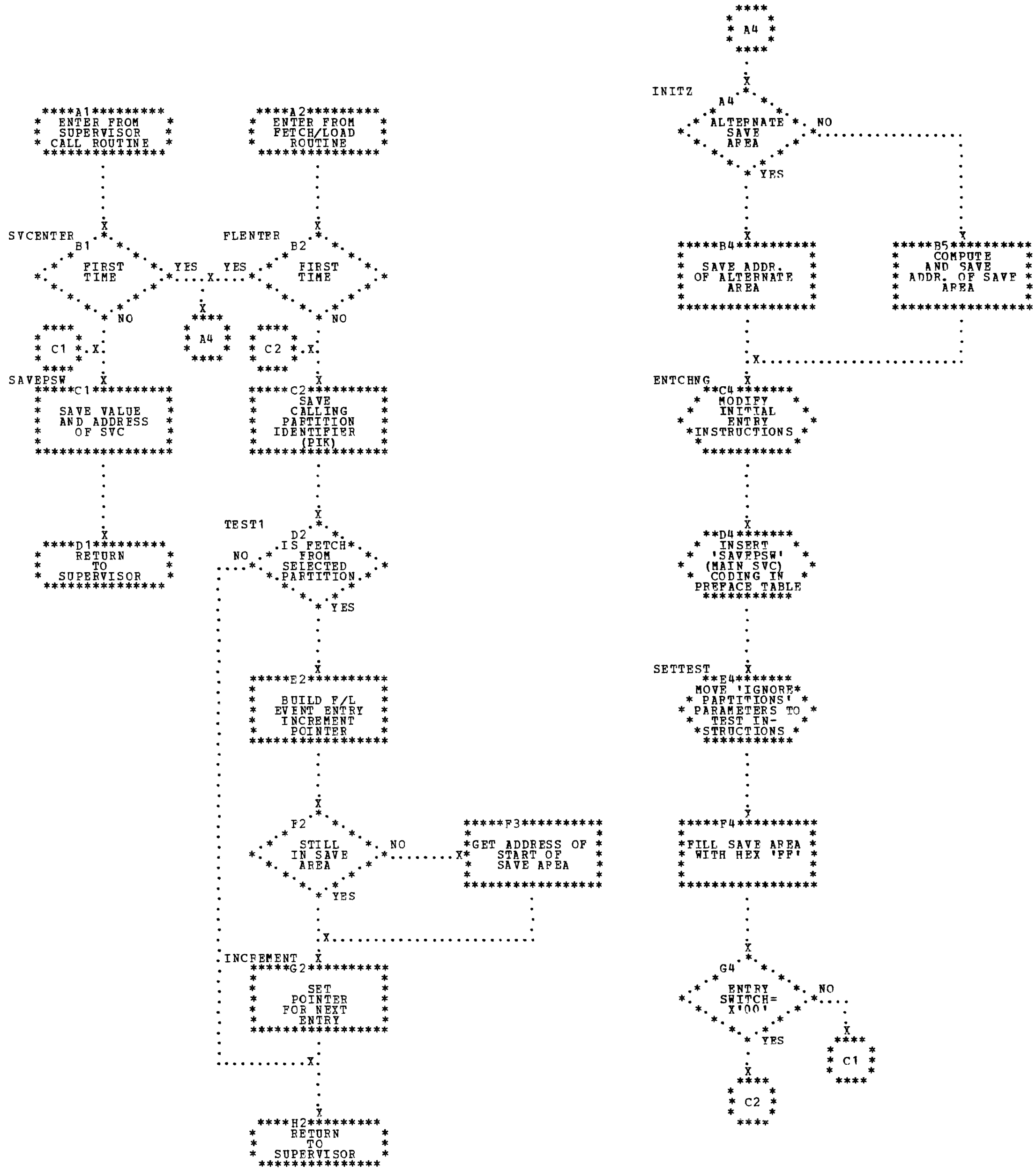
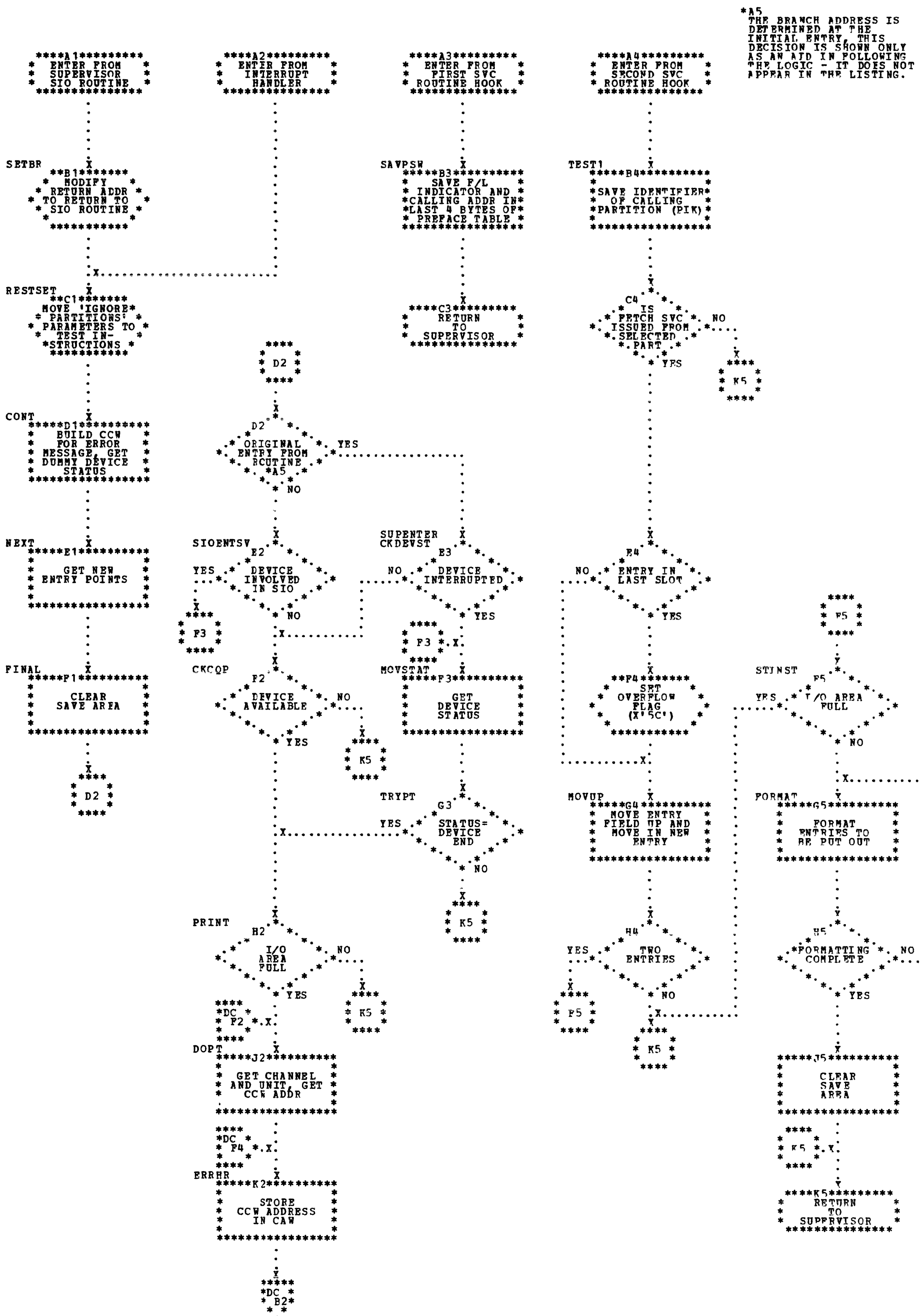


Chart DA. F/L Trace: Core-Wrap Mode (PDAIDFTW)





*A5 THE BRANCH ADDRESS IS DETERMINED AT THE INITIAL ENTRY. THIS DECISION IS SHOWN ONLY AS AN AID IN FOLLOWING THE LOGIC - IT DOES NOT APPEAR IN THE LISTING.
A1 ENTER FROM SUPERVISOR SIO ROUTINE
A2 ENTER FROM INTERRUPT HANDLER
A3 ENTER FROM FIRST SVC ROUTINE HOOK
A4 ENTER FROM SECOND SVC ROUTINE HOOK
SETBR B1 MODIFY RETURN ADDR TO RETURN TO SIO ROUTINE
SAVPSW B3 SAVE F/L INDICATOR AND CALLING ADDR IN LAST 4 BYTES OF PREFACE TABLE
TEST1 B4 SAVE IDENTIFIER OF CALLING PARTITION (PIK)
RESTSET C1 MOVE 'IGNORE PARTITIONS' PARAMETERS TO TEST INSTRUCTIONS
C3 RETURN TO SUPERVISOR
C4 IS FETCH SVC ISSUED FROM SELECTED PART. NO YES
K5
D2
D2 ORIGINAL ENTRY FROM ROUTINE *A5 YES NO
CONT D1 BUILD CCW FOR ERROR MESSAGE, GET DUMMY DEVICE STATUS
NEXT E1 GET NEW ENTRY POINTS
SIOENTSV E2 DEVICE INVOLVED IN SIO YES NO
F3
SUPENTER CKDEVST E3 DEVICE INTERRUPTED NO YES
E4 ENTRY IN LAST SLOT NO YES
F5
FINAL F1 CLEAR SAVE AREA
D2
CKCQP F2 DEVICE AVAILABLE NO YES
K5
F3
MOVSTAT F3 GET DEVICE STATUS
F4 SET OVERFLOW FLAG (X'5C')
STINST F5 I/O AREA FULL YES NO
TRYPT G3 STATUS= DEVICE END YES NO
K5
MOVUP G4 MOVE ENTRY FIELD UP AND MOVE IN NEW ENTRY
FORMAT G5 FORMAT ENTRIES TO BE PUT OUT
PRINT H2 I/O AREA FULL NO YES
K5
DC F2
H4 TWO ENTRIES YES NO
F5
K5
H5 FORMATTING COMPLETE NO YES
DOPT J2 GET CHANNEL AND UNIT, GET CCW ADDR
J5 CLEAR SAVE AREA
DC F4
K5
ERRHR K2 STORE CCW ADDRESS IN CAW
K5 RETURN TO SUPERVISOR
DC B2

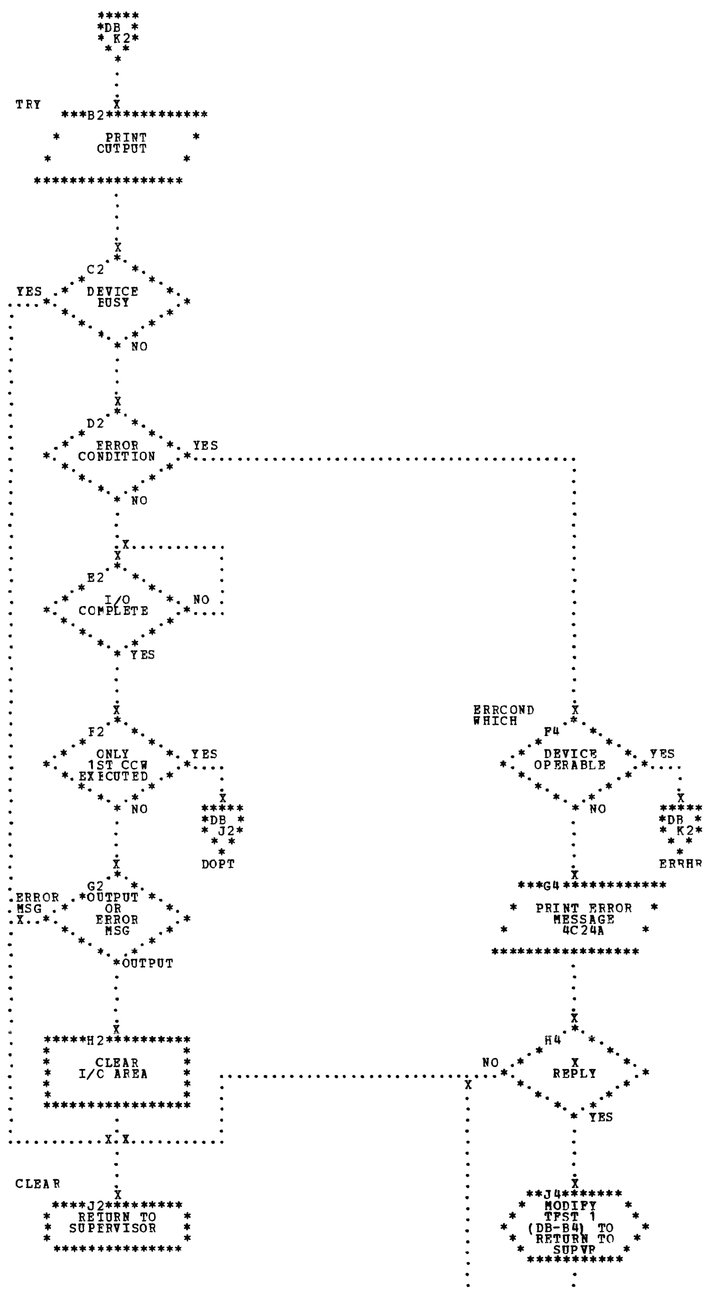
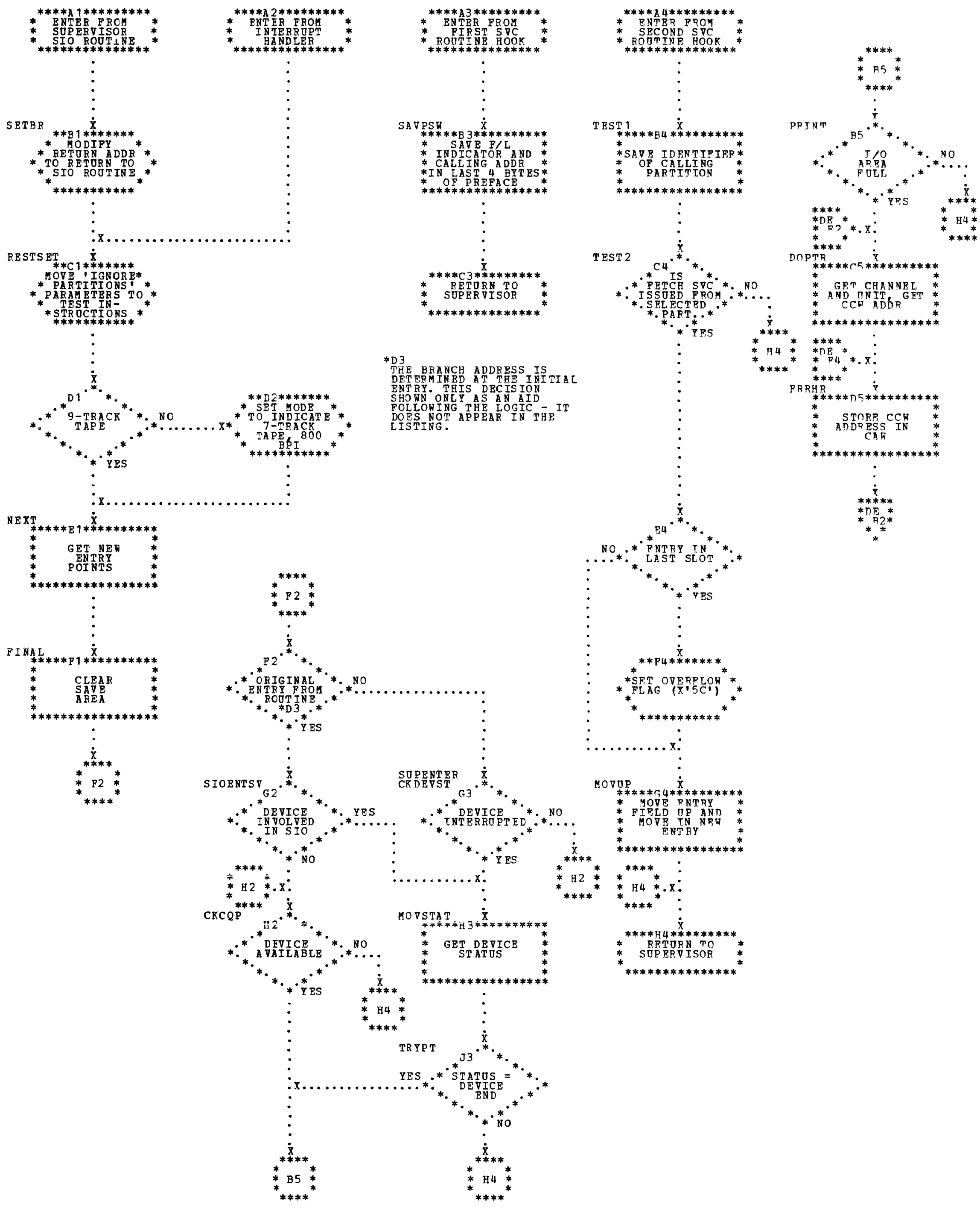
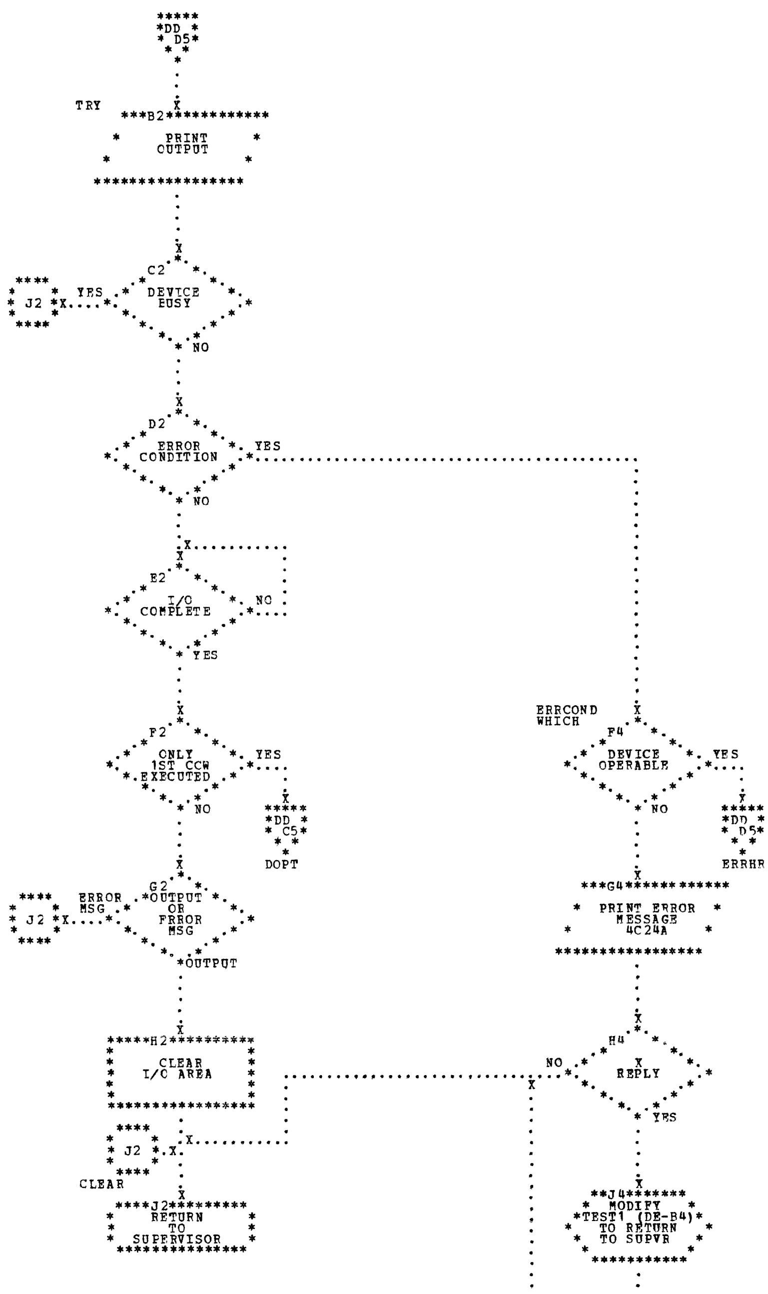

DB K2
TRY
B2
PRINT OUTPUT
C2
DEVICE BUSY
YES
NO
D2
ERROR CONDITION
YES
NO
E2
I/O COMPLETE
NO
YES
F2
ONLY 1ST CCW EXECUTED
YES
NO
DB J2
DOPT
G2
OUTPUT OR ERROR MSG
ERROR MSG
OUTPUT
H2
CLEAR I/O AREA
CLEAR
J2
RETURN TO SUPERVISOR
ERRCOND WHICH
F4
DEVICE OPERABLE
YES
NO
DB K2
ERRHR
G4
PRINT ERROR MESSAGE 4C24A
H4
REPLY
NO
YES
J4
MODIFY TPST 1 (DB-B4) TO RETURN TO SUPVR

Chart DD. F/L Trace: Tape Mode (PDAIDFTT) (Part 1 of 2)




DD D5
TRY
B2
PRINT OUTPUT
C2
DEVICE BUSY
YES
J2
NO
D2
ERROR CONDITION
YES
NO
E2
I/O COMPLETE
NO
YES
F2
ONLY 1ST CCW EXECUTED
YES
NO
DD C5
DOPT
G2
OUTPUT OR ERROR MSG
ERROR MSG
J2
OUTPUT
H2
CLEAR I/O AREA
J2
CLEAR
J2
RETURN TO SUPERVISOR
ERRCOND WHICH
F4
DEVICE OPERABLE
YES
NO
DD D5
ERRHR
G4
PRINT ERROR MESSAGE 4C24A
H4
REPLY
NO
YES
J4
MODIFY TEST1 (DE-B4) TO RETURN TO SUPVR

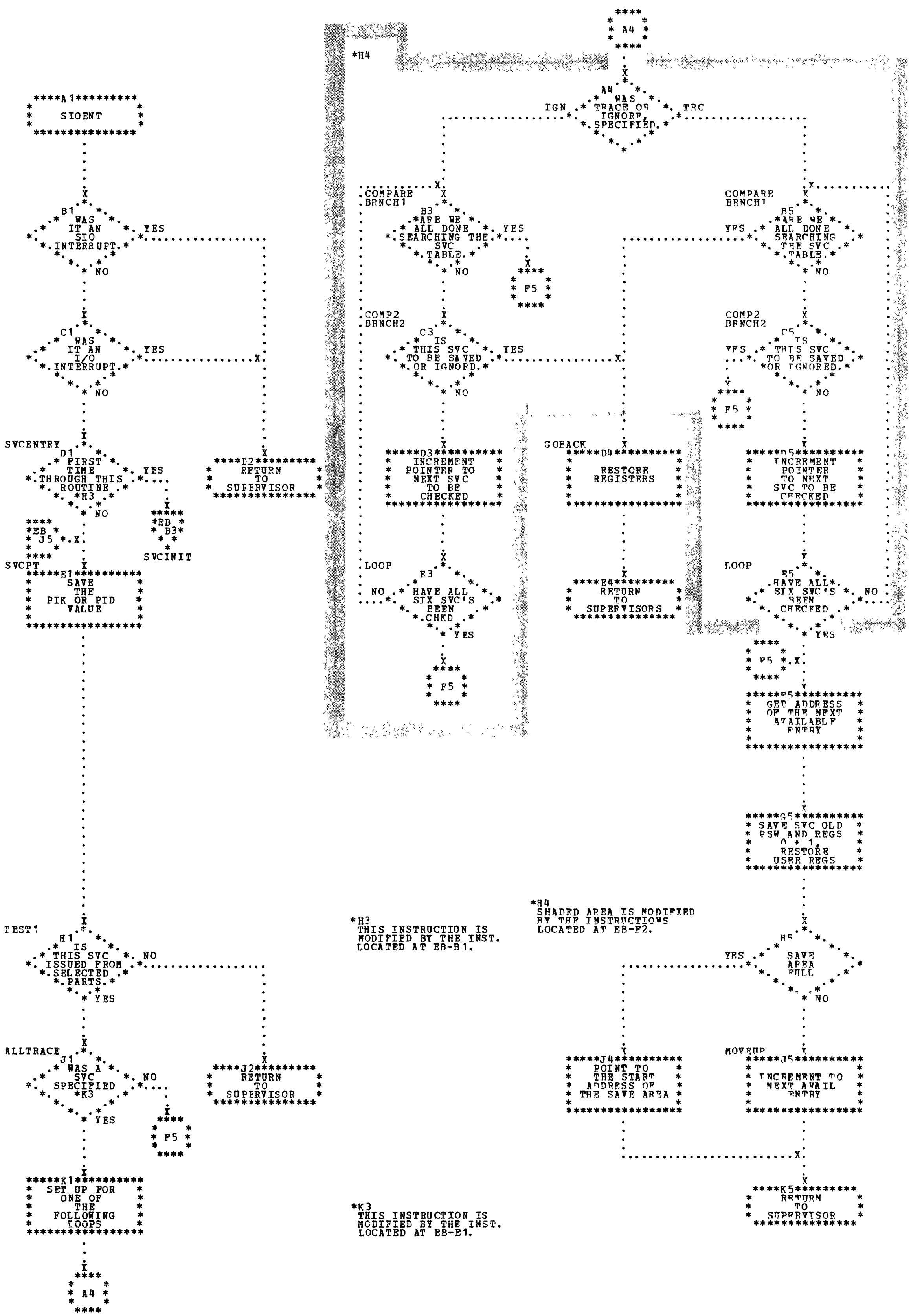

SIOENT
WAS IT AN SIO INTERRUPT
WAS IT AN I/O INTERRUPT
SVCENTRY
FIRST TIME THROUGH THIS ROUTINE
SVCPT
SAVE THE PIK OR PID VALUE
SVCINIT
RETURN TO SUPERVISOR
TEST1
IS THIS SVC ISSUED FROM SELECTED PARTS.
ALLTRACE
WAS A SVC SPECIFIED
SET UP FOR ONE OF THE FOLLOWING LOOPS
WAS TRACE OR IGNORE SPECIFIED
COMPARE BRNCH1
ARE WE ALL DONE SEARCHING THE SVC TABLE.
COMP2 BRNCH2
IS THIS SVC TO BE SAVED OR IGNORD.
INCREMENT POINTER TO NEXT SVC TO BE CHECKED
LOOP
HAVE ALL SIX SVC'S BEEN CHKD
GOBACK
RESTORE REGISTERS
RETURN TO SUPERVISORS
GET ADDRESS OF THE NEXT AVAILABLE ENTRY
SAVE SVC OLD PSW AND REGS 0 + 1, RESTORE USER REGS
SAVE AREA FULL
POINT TO THE START ADDRESS OF THE SAVE AREA
MOVEUP
INCREMENT TO NEXT AVAIL ENTRY
*H3
THIS INSTRUCTION IS MODIFIED BY THE INST. LOCATED AT EB-B1.
*H4
SHADED AREA IS MODIFIED BY THE INSTRUCTIONS LOCATED AT EB-F2.
*K3
THIS INSTRUCTION IS MODIFIED BY THE INST. LOCATED AT EB-E1.

Chart EB. GSVC Trace: Core-Wrap Mode (PDAIDGTW) (Part 2 of 2)

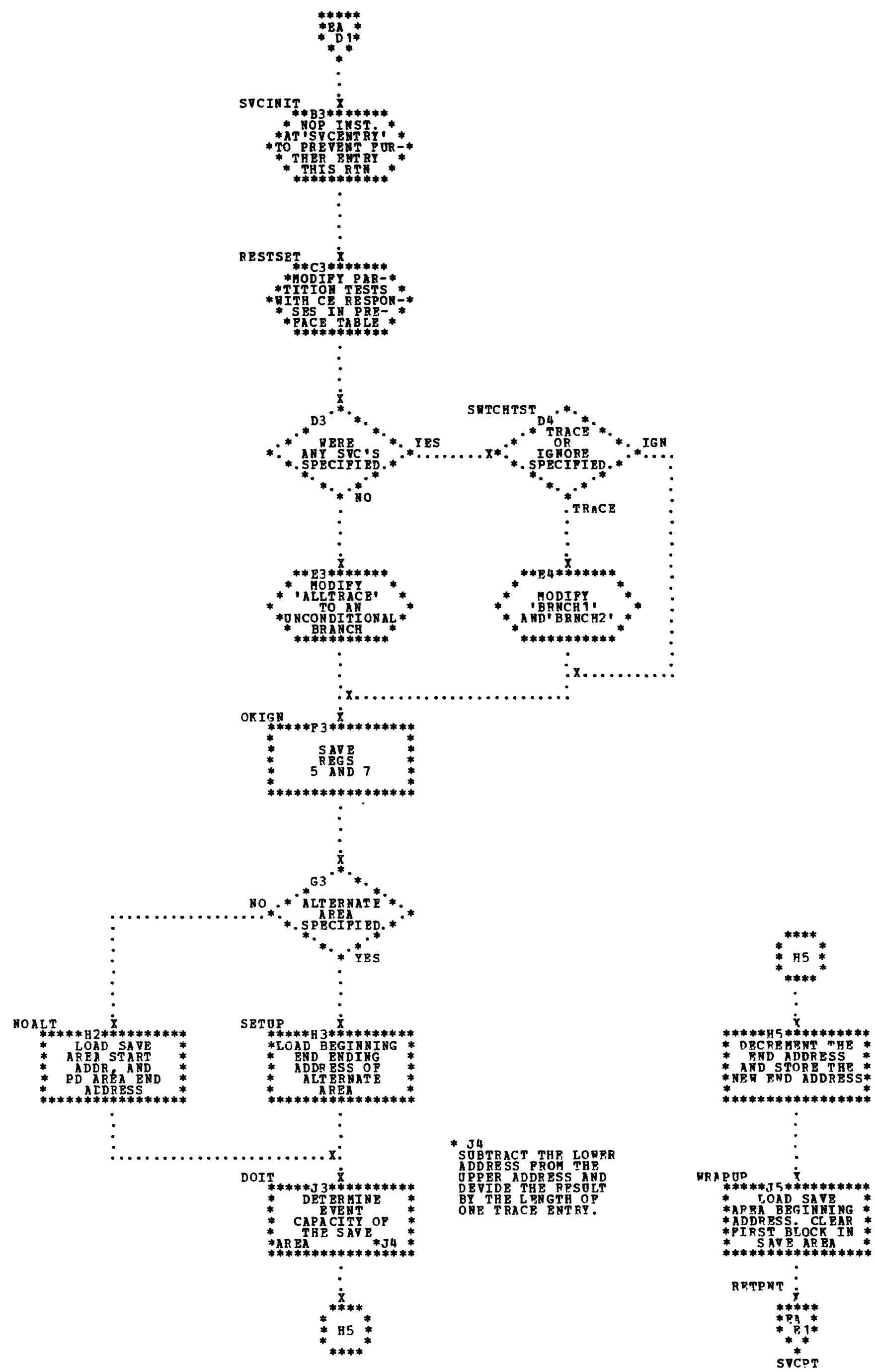


Chart EC. GSVC Trace: Print Mode (PDAIDGTP) (Part 1 of 3)

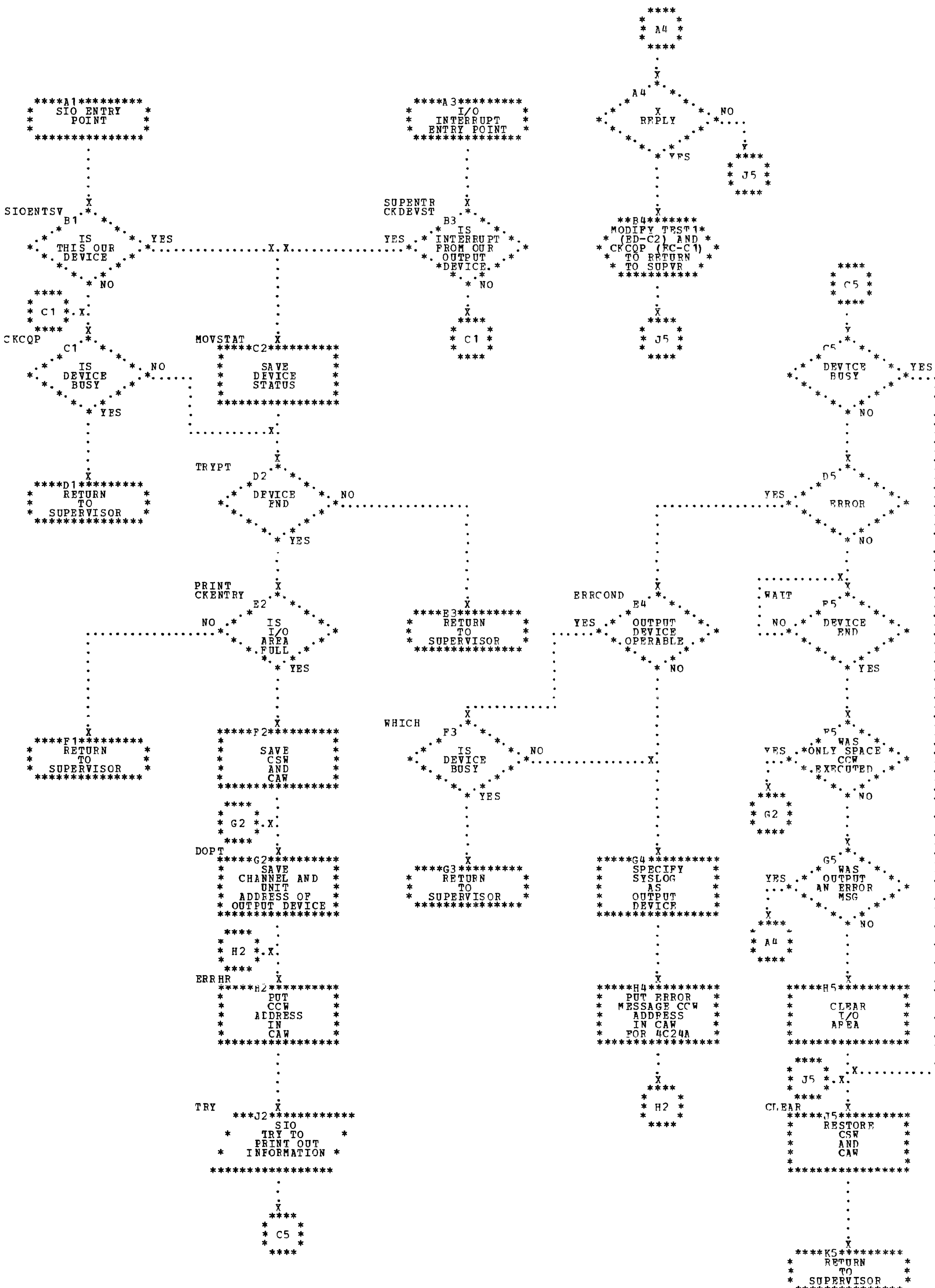
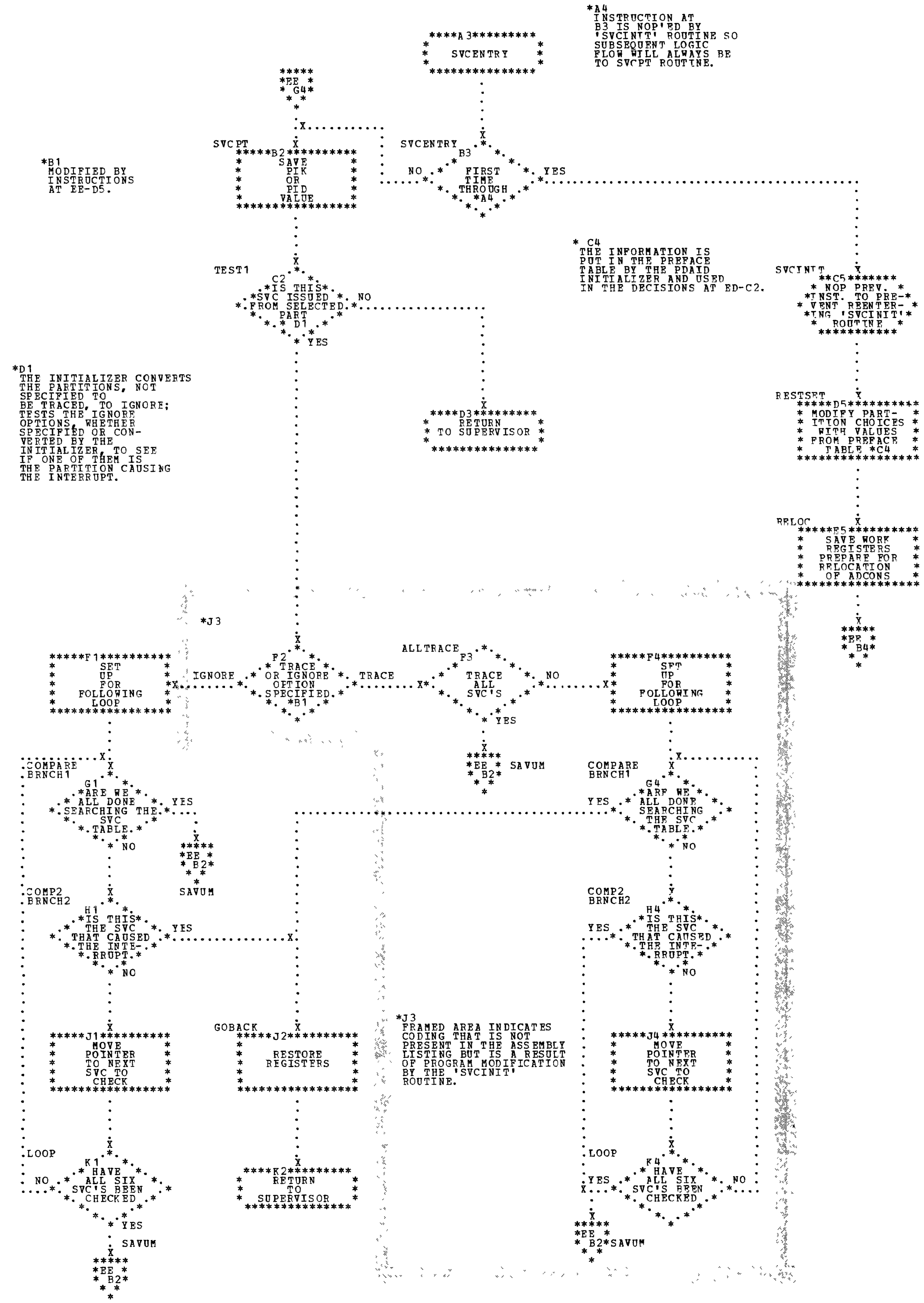
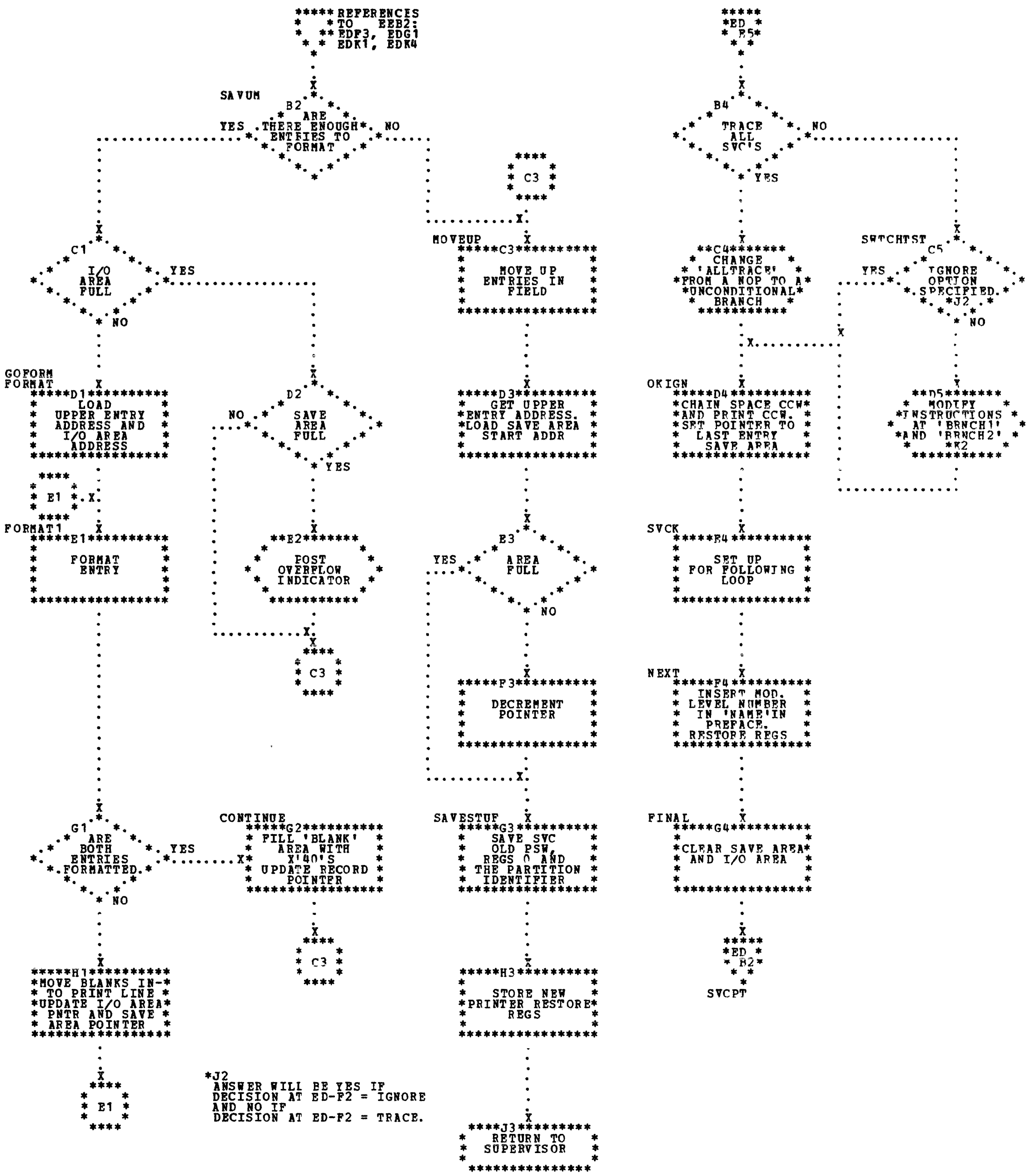


Chart ED. GSVC Trace: Print Mcde (PDAIDGTP) (Part 2 cf 3)



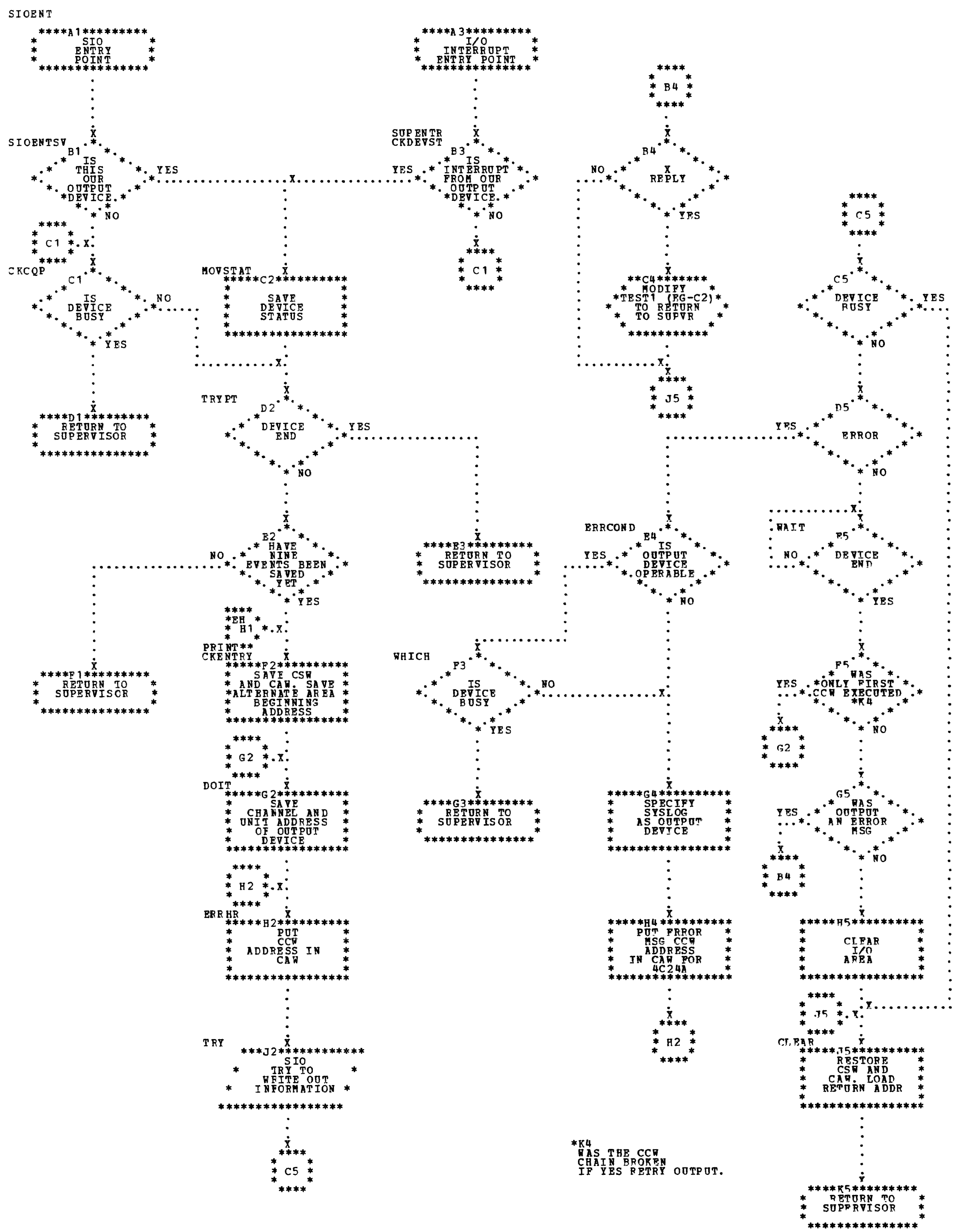


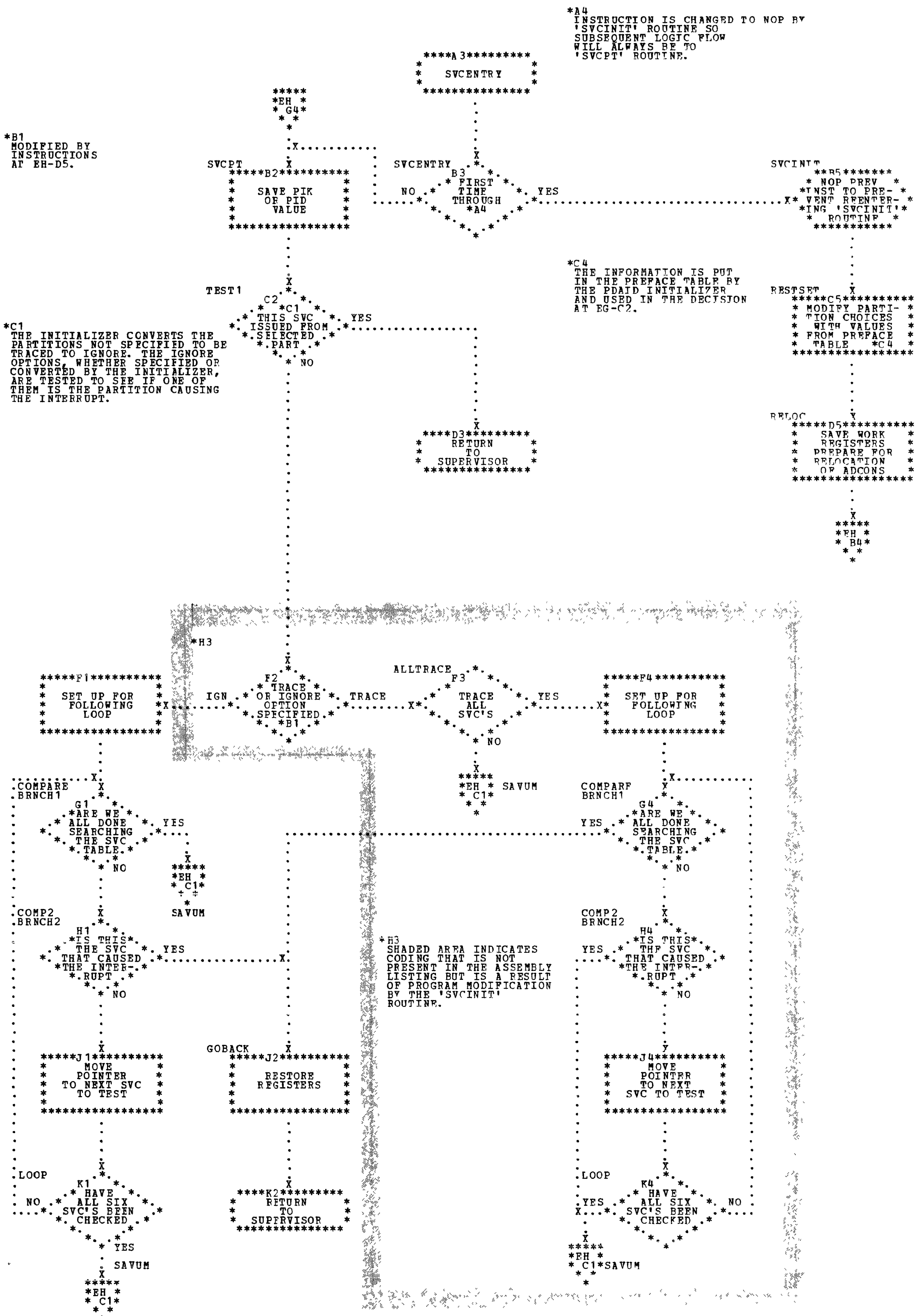
*J2
ANSWER WILL BE YES IF DECISION AT ED-F2 = IGNORE AND NO IF DECISION AT ED-F2 = TRACE.

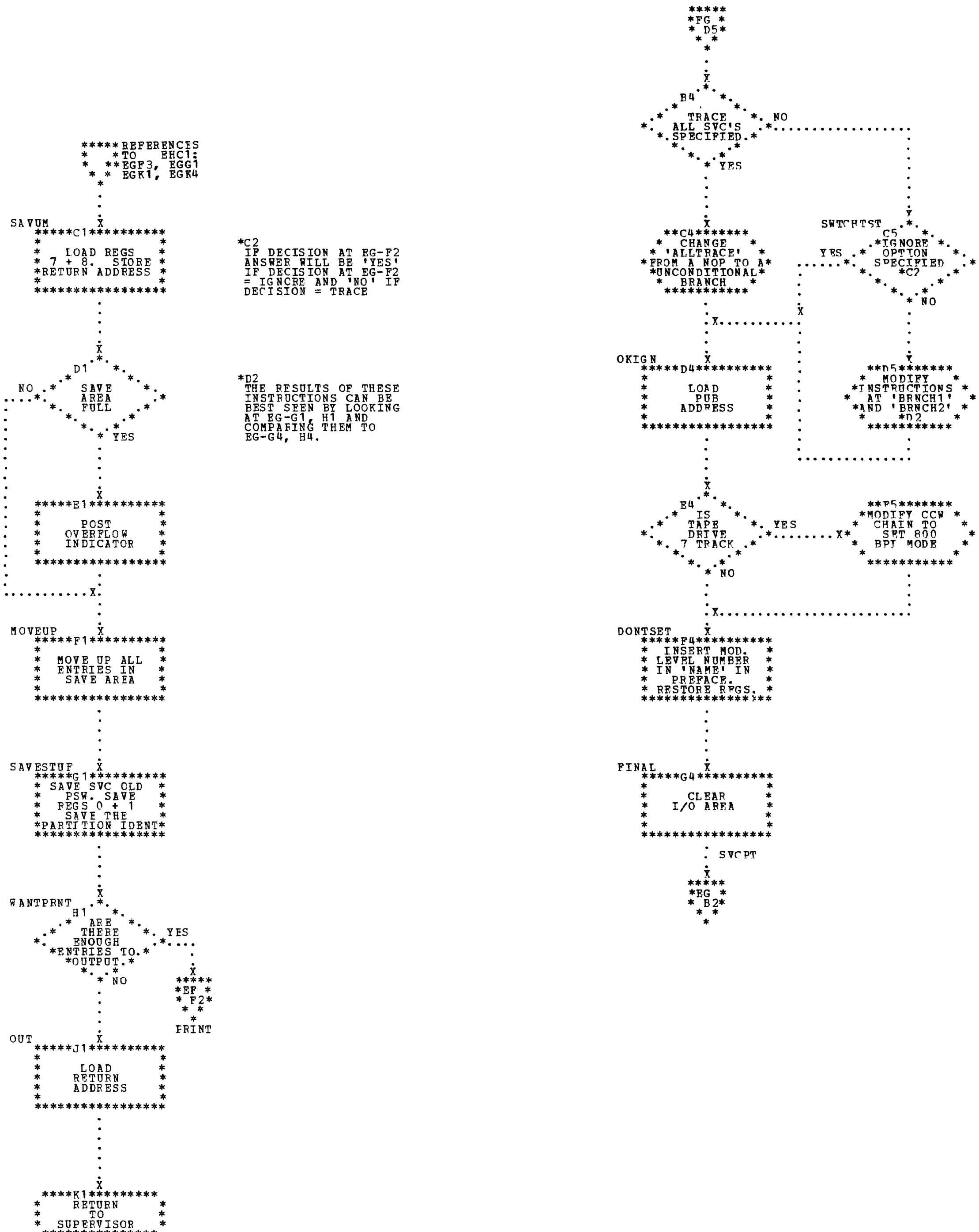
*K1
THE FIRST TIME THROUGH, THE POINTER IS POINTING AT THE DECISION AT ED-C2.

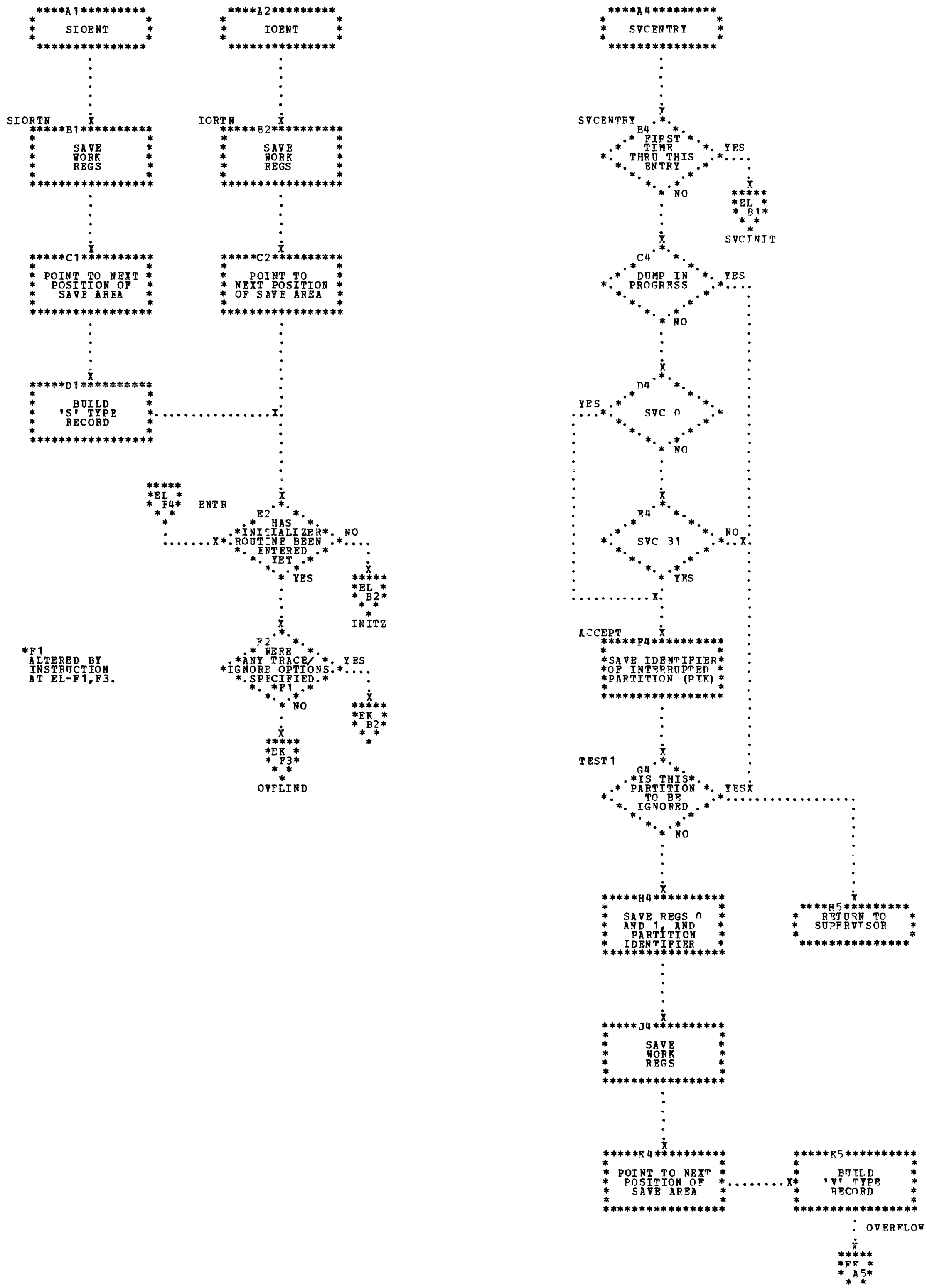
*K2
THE RESULTS OF THESE INSTRUCTIONS CAN BE BEST SEEN BY LOOKING AT ED-G1,H1, AND COMPARING THEM TO ED-G4,H4.

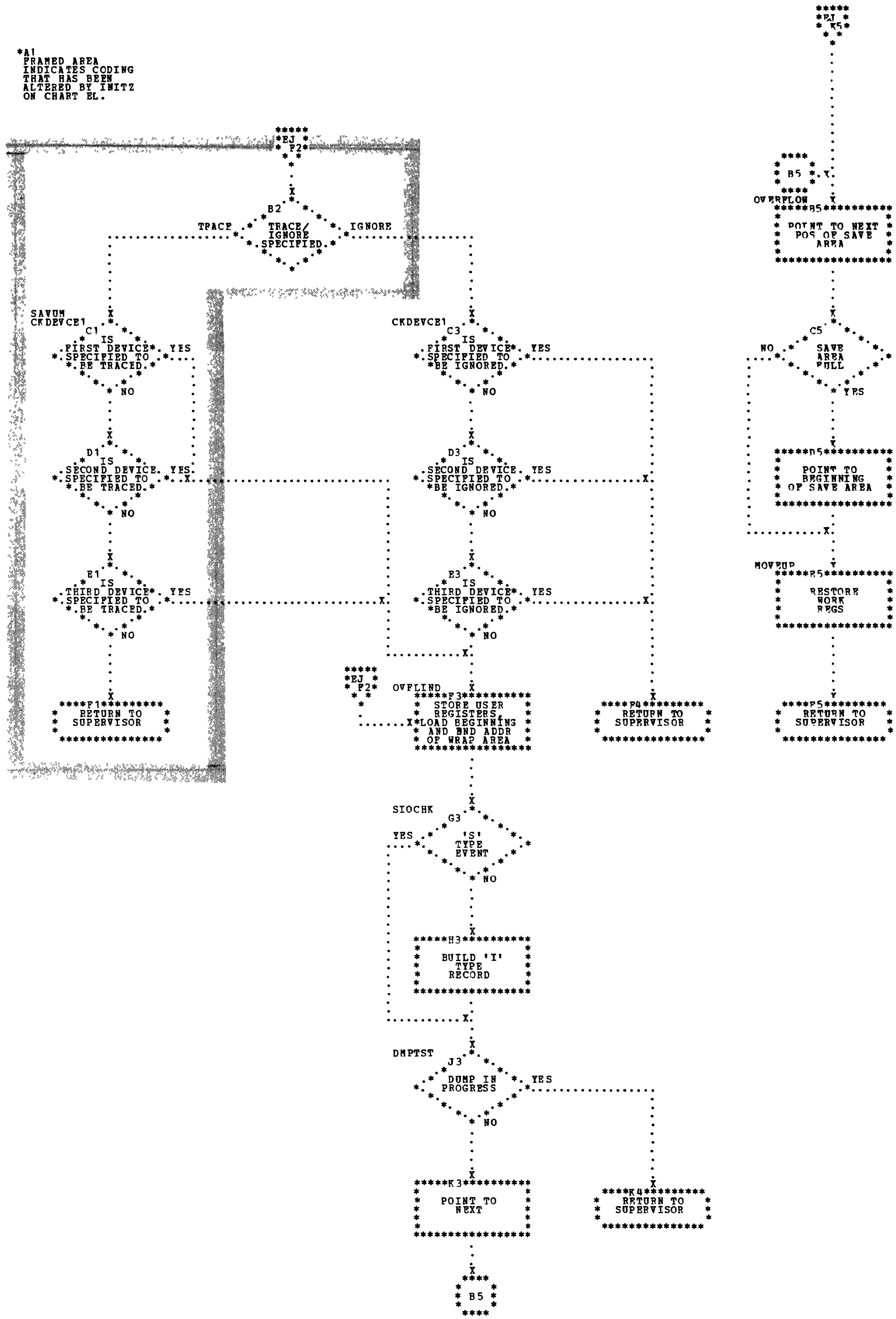
Chart EF. GSVC Trace: Tape Mode (PDAIDGTT) (Part 1 of 3)

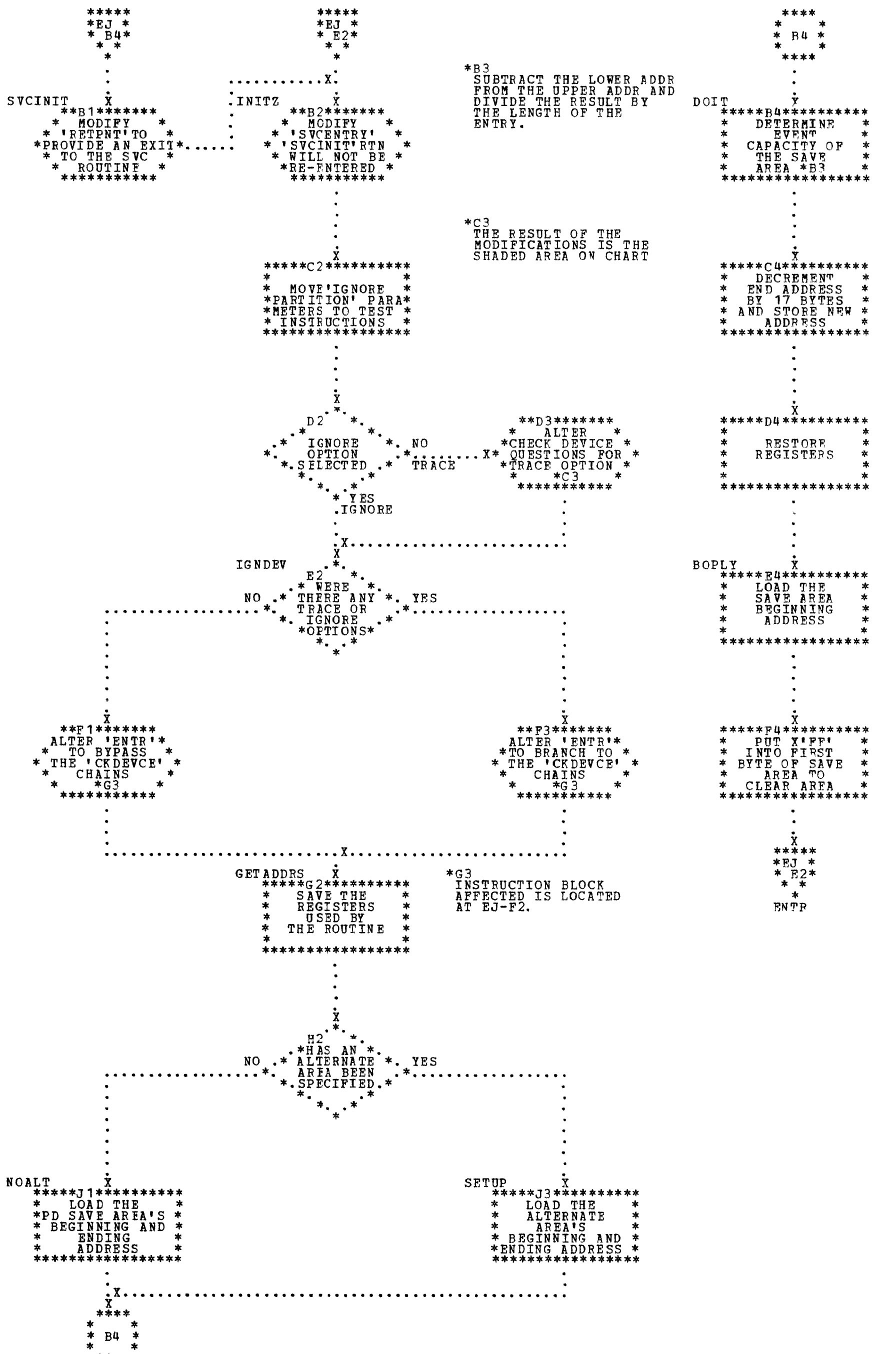



*A4
INSTRUCTION IS CHANGED TO NOP BY 'SVCINIT' ROUTINE SO SUBSEQUENT LOGIC FLOW WILL ALWAYS BE TO 'SVCPT' ROUTINE.
A3 SVCENTRY
EH G4
*B1
MODIFIED BY INSTRUCTIONS AT EH-D5.
SVCPT
B2 SAVE PIK OR PID VALUE
SVCENTRY
B3 FIRST TIME THROUGH *A4
NO
YES
SVCINIT
B5 NOP PREV INST TO PRE- VENT REENTER- ING 'SVCINIT' ROUTINE
*C4
THE INFORMATION IS PUT IN THE PREFACE TABLE BY THE PDAID INITIALIZER AND USED IN THE DECISION AT EG-C2.
TEST1
C2 *C1 THIS SVC ISSUED FROM SELECTED PART
YES
NO
*C1
THE INITIALIZER CONVERTS THE PARTITIONS NOT SPECIFIED TO BE TRACED TO IGNORE. THE IGNORE OPTIONS, WHETHER SPECIFIED OR CONVERTED BY THE INITIALIZER, ARE TESTED TO SEE IF ONE OF THEM IS THE PARTITION CAUSING THE INTERRUPT.
RESTSET
C5 MODIFY PARTI- TION CHOICES WITH VALUES FROM PREFACE TABLE *C4
D3 RETURN TO SUPERVISOR
RELOC
D5 SAVE WORK REGISTERS PREPARE FOR RELOCATION OF ADCONS
EH B4
*H3
F1 SET UP FOR FOLLOWING LOOP
IGN
F2 TRACE OR IGNORE OPTION SPECIFIED *B1
TRACE
ALLTRACE
F3 TRACE ALL SVC'S
YES
NO
F4 SET UP FOR FOLLOWING LOOP
EH C1 SAVUM
COMPARE BRNCH1
G1 ARE WE ALL DONE SEARCHING THE SVC TABLE
YES
NO
EH C1 SAVUM
COMPARE BRNCH1
G4 ARE WE ALL DONE SEARCHING THE SVC TABLE
YES
NO
COMP2 BRNCH2
H1 IS THIS THE SVC THAT CAUSED THE INTER- RUPT
YES
NO
*H3
SHADED AREA INDICATES CODING THAT IS NOT PRESENT IN THE ASSEMBLY LISTING BUT IS A RESULT OF PROGRAM MODIFICATION BY THE 'SVCINIT' ROUTINE.
COMP2 BRNCH2
H4 IS THIS THE SVC THAT CAUSED THE INTER- RUPT
YES
NO
J1 MOVE POINTER TO NEXT SVC TO TEST
GOBACK
J2 RESTORE REGISTERS
J4 MOVE POINTER TO NEXT SVC TO TEST
LOOP
K1 HAVE ALL SIX SVC'S BEEN CHECKED
NO
YES
K2 RETURN TO SUPERVISOR
LOOP
K4 HAVE ALL SIX SVC'S BEEN CHECKED
YES
NO
SAVUM
EH C1
EH C1 SAVUM

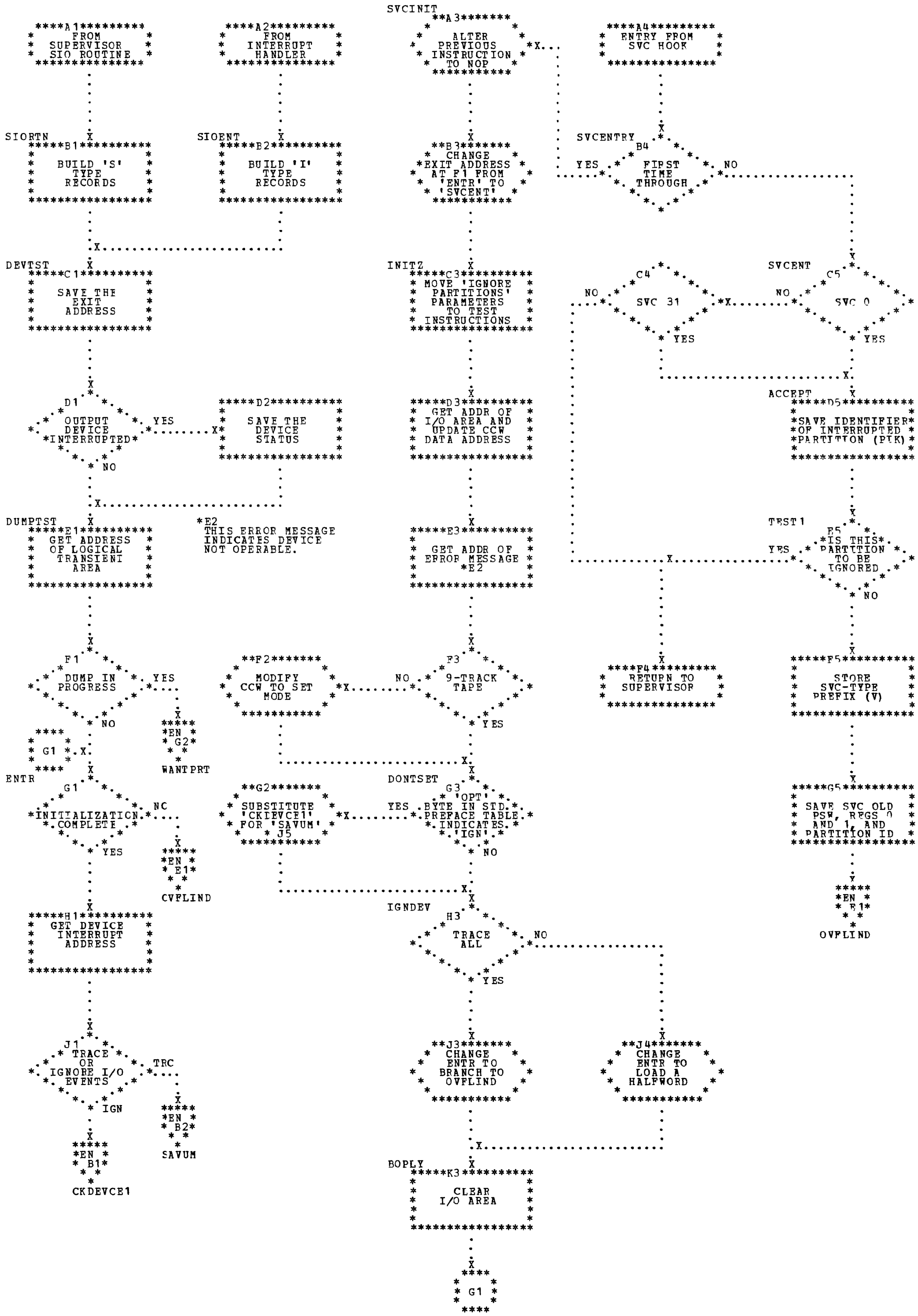

REFERENCES TO EHC1: EGF3, EGG1 EGK1, EGK4
SAVUM
C1
LOAD REGS 7 + 8. STORE RETURN ADDRESS
*C2 IF DECISION AT EG-F2 ANSWER WILL BE 'YES' IF DECISION AT EG-F2 = IGNCRE AND 'NO' IF DECISION = TRACE
D1
SAVE AREA FULL
NO
YES
*D2 THE RESULTS OF THESE INSTRUCTIONS CAN BE BEST SEEN BY LOOKING AT EG-G1, H1 AND COMPARING THEM TO EG-G4, H4.
E1
POST OVERFLOW INDICATOR
MOVEUP
F1
MOVE UP ALL ENTRIES IN SAVE AREA
SAVESTUF
G1
SAVE SVC OLD PSW. SAVE REGS 0 + 1 SAVE THE PARTITION IDENT
WANTPRNT
H1
ARE THERE ENOUGH ENTRIES TO OUTPUT
YES
NO
EF F2
PRINT
OUT
J1
LOAD RETURN ADDRESS
K1
RETURN TO SUPERVISOR
FG D5
B4
TRACE ALL SVC'S SPECIFIED
NO
YES
C4
CHANGE 'ALLTRACE' FROM A NOP TO A UNCONDITIONAL BRANCH
SWTCHTST
C5
IGNORE OPTION SPECIFIED *C2
YES
NO
OKIGN
D4
LOAD PUB ADDRESS
D5
MODIFY INSTRUCTIONS AT 'BRNCH1' AND 'BRNCH2' *D2
E4
IS TAPE DRIVE 7 TRACK
YES
NO
E5
MODIFY CCW CHAIN TO SET 800 BPI MODE
DONTSET
F4
INSERT MOD. LEVEL NUMBER IN 'NAME' IN PREFACE. RESTORE REGS.
FINAL
G4
CLEAR I/O AREA
SVCPT
EG B2


A1 SIOENT
A2 IOENT
A4 SVCENTRY
SIORTN B1 SAVE WORK REGS
IORTN B2 SAVE WORK REGS
SVCENTRY B4 FIRST TIME THRU THIS ENTRY
YES
NO
EL B1 SVCINIT
C1 POINT TO NEXT POSITION OF SAVE AREA
C2 POINT TO NEXT POSITION OF SAVE AREA
C4 DUMP IN PROGRESS
D1 BUILD 'S' TYPE RECORD
D4 SVC 0
EL F4 ENTR
E2 HAS INITIALIZER ROUTINE BEEN ENTERED YET
EL B2 INITZ
E4 SVC 31
*F1 ALTERED BY INSTRUCTION AT EL-F1,F3.
F2 WERE ANY TRACE/ IGNORE OPTIONS SPECIFIED *F1
EK B2
EK F3 OVFLIND
ACCEPT F4 SAVE IDENTIFIER OF INTERRUPTED PARTITION (PIK)
TEST1 G4 IS THIS PARTITION TO BE IGNORED
H4 SAVE REGS 0 AND 1, AND PARTITION IDENTIFIER
H5 RETURN TO SUPERVISOR
J4 SAVE WORK REGS
K4 POINT TO NEXT POSITION OF SAVE AREA
K5 BUILD 'V' TYPE RECORD
OVERFLOW
EK A5


*A1
FRAMED AREA INDICATES CODING THAT HAS BEEN ALTERED BY INITZ ON CHART EL.
EJ F2
B2
TRACE/ IGNORE SPECIFIED
TRACE
IGNORE
SAVUM CKDEVCE1
C1
IS FIRST DEVICE SPECIFIED TO BE TRACED
YES
NO
D1
IS SECOND DEVICE SPECIFIED TO BE TRACED
E1
IS THIRD DEVICE SPECIFIED TO BE TRACED
F1
RETURN TO SUPERVISOR
CKDEVCE1
C3
IS FIRST DEVICE SPECIFIED TO BE IGNORED
D3
IS SECOND DEVICE SPECIFIED TO BE IGNORED
E3
IS THIRD DEVICE SPECIFIED TO BE IGNORED
EJ F2
OVFLIND
F3
STORE USER REGISTERS LOAD BEGINNING AND END ADDR OF WRAP AREA
F4
RETURN TO SUPERVISOR
SIOCHK
G3
'S' TYPE EVENT
H3
BUILD 'I' TYPE RECORD
DMPTST
J3
DUMP IN PROGRESS
K3
POINT TO NEXT
K4
RETURN TO SUPERVISOR
B5
EJ K5
OVERFLOW
B5
POINT TO NEXT POS OF SAVE AREA
C5
SAVE AREA FULL
D5
POINT TO BEGINNING OF SAVE AREA
MOVEUP
E5
RESTORE WORK REGS
F5
RETURN TO SUPERVISOR


EJ B4
EJ E2
SVCINIT
B1 MODIFY 'RETPNT' TO PROVIDE AN EXIT TO THE SVC ROUTINE
INITZ
B2 MODIFY 'SVCENTRY' 'SVCINIT' RTN WILL NOT BE RE-ENTERED
*B3 SUBTRACT THE LOWER ADDR FROM THE UPPER ADDR AND DIVIDE THE RESULT BY THE LENGTH OF THE ENTRY.
B4
DOIT
B4 DETERMINE EVENT CAPACITY OF THE SAVE AREA *B3
C2 MOVE 'IGNORE PARTITION' PARAMETERS TO TEST INSTRUCTIONS
*C3 THE RESULT OF THE MODIFICATIONS IS THE SHADED AREA ON CHART
C4 DECREMENT END ADDRESS BY 17 BYTES AND STORE NEW ADDRESS
D2 IGNORE OPTION SELECTED
NO TRACE
YES IGNORE
D3 ALTER CHECK DEVICE QUESTIONS FOR TRACE OPTION *C3
D4 RESTORE REGISTERS
IGNDEV
E2 WERE THERE ANY TRACE OR IGNORE OPTIONS
NO
YES
BOPLY
E4 LOAD THE SAVE AREA BEGINNING ADDRESS
F1 ALTER 'ENTR' TO BYPASS THE 'CKDEVCE' CHAINS *G3
F3 ALTER 'ENTR' TO BRANCH TO THE 'CKDEVCE' CHAINS *G3
F4 PUT X'FF' INTO FIRST BYTE OF SAVE AREA TO CLEAR AREA
EJ E2
ENTR
GETADDRS
G2 SAVE THE REGISTERS USED BY THE ROUTINE
*G3 INSTRUCTION BLOCK AFFECTED IS LOCATED AT EJ-F2.
H2 HAS AN ALTERNATE AREA BEEN SPECIFIED
NO
YES
NOALT
J1 LOAD THE PD SAVE AREA'S BEGINNING AND ENDING ADDRESS
SETUP
J3 LOAD THE ALTERNATE AREA'S BEGINNING AND ENDING ADDRESS
B4

rt EM. QTAM Trace: Tape Output Phase (PDAIDQTT) (Part 1 of 2)



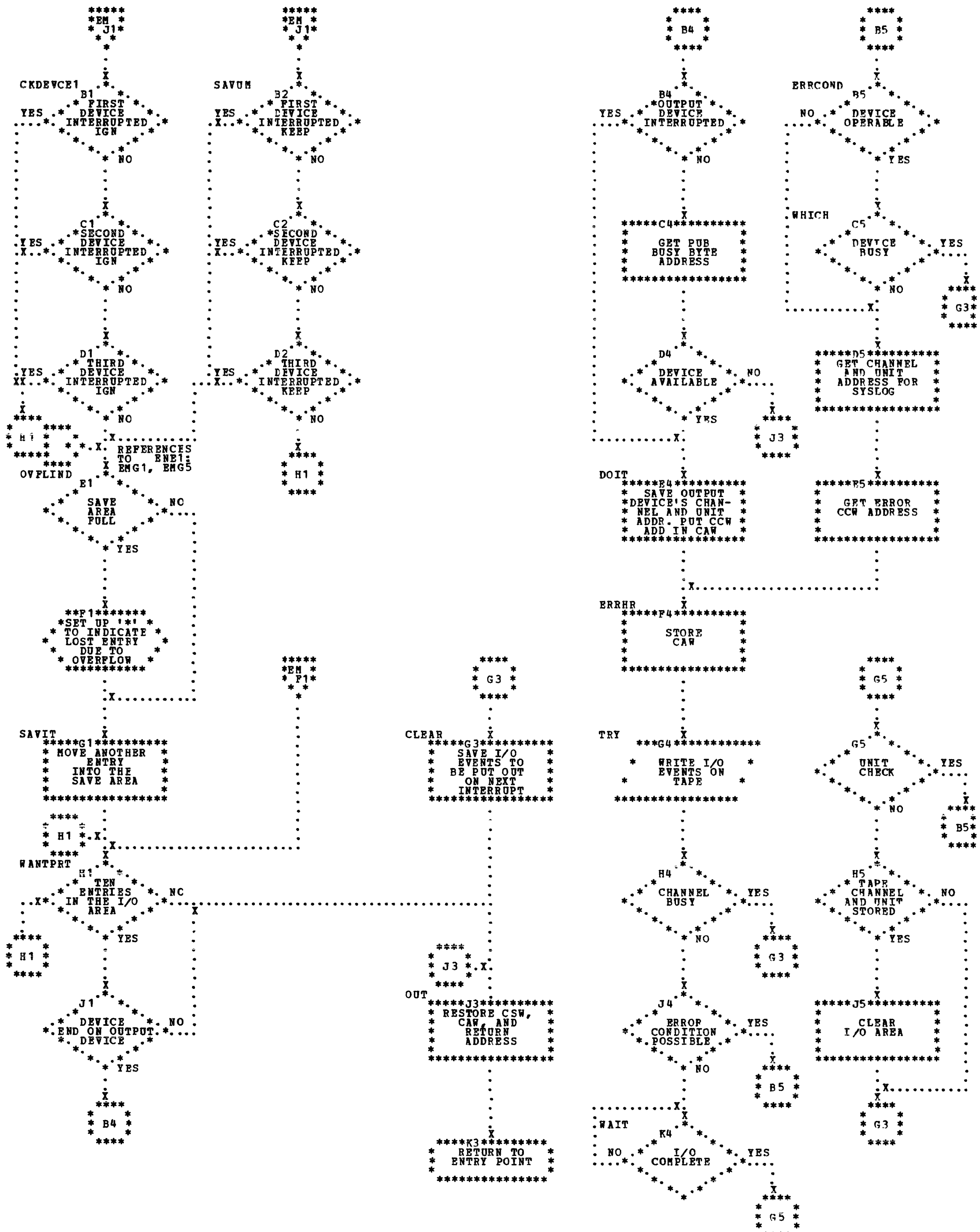
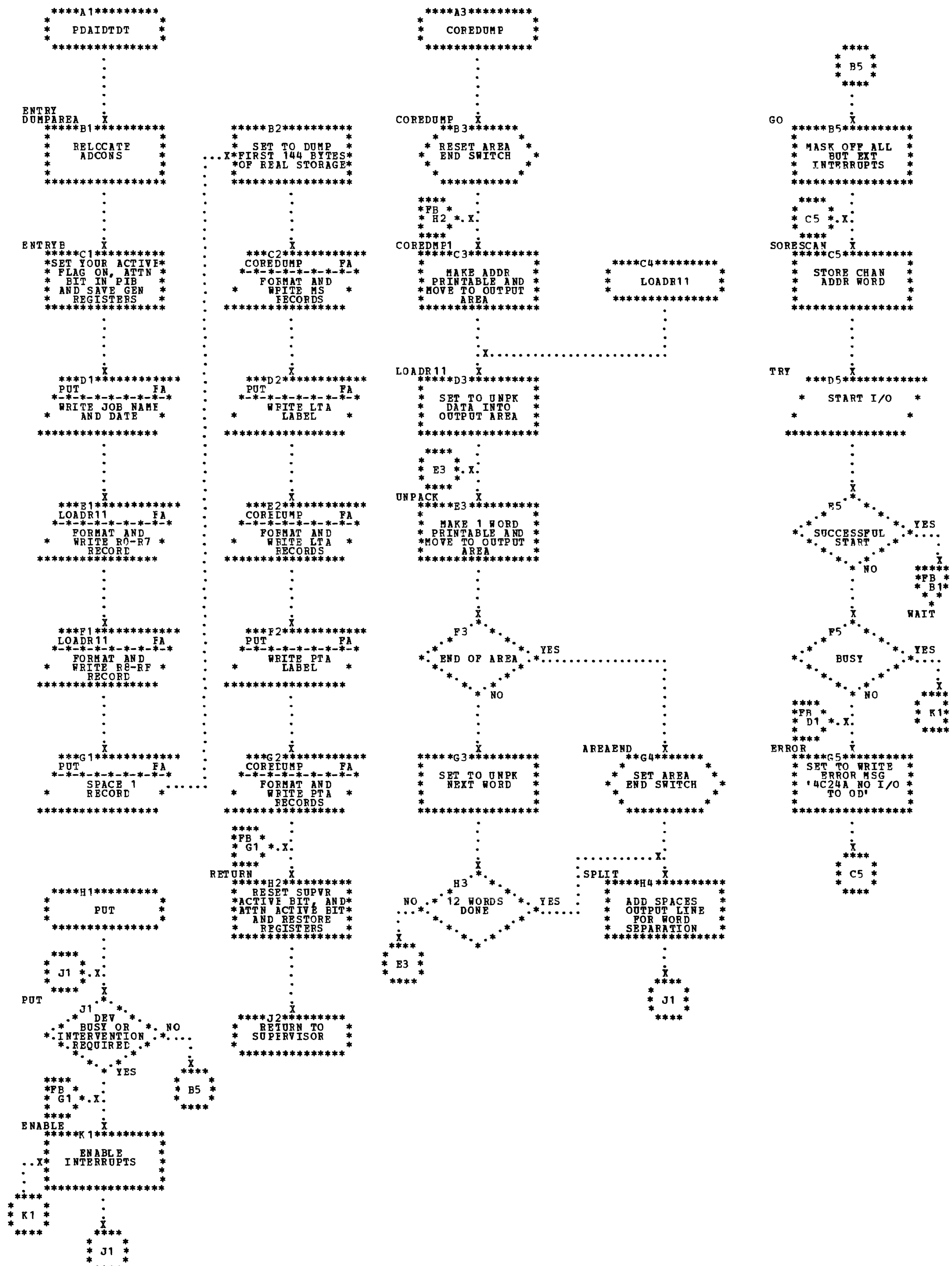

EN J1
CKDEVCE1 B1 FIRST DEVICE INTERRUPTED IGN
YES
NO
C1 SECOND DEVICE INTERRUPTED IGN
D1 THIRD DEVICE INTERRUPTED IGN
H1
REFERENCES TO ENE1: ENG1, ENG5
OVFLIND E1 SAVE AREA FULL
F1 SET UP '*' TO INDICATE LOST ENTRY DUE TO OVERFLOW
SAVIT G1 MOVE ANOTHER ENTRY INTO THE SAVE AREA
WANTPRT H1 TEN ENTRIES IN THE I/O AREA
J1 DEVICE END ON OUTPUT DEVICE
B4
SAVUM B2 FIRST DEVICE INTERRUPTED KEEP
C2 SECOND DEVICE INTERRUPTED KEEP
D2 THIRD DEVICE INTERRUPTED KEEP
EN F1
G3
CLEAR G3 SAVE I/O EVENTS TO BE PUT OUT ON NEXT INTERRUPT
J3
OUT J3 RESTORE CSW, CAW, AND RETURN ADDRESS
K3 RETURN TO ENTRY POINT
B4 OUTPUT DEVICE INTERRUPTED
C4 GET PUB BUSY BYTE ADDRESS
D4 DEVICE AVAILABLE
DOIT E4 SAVE OUTPUT DEVICE'S CHANNEL AND UNIT ADDR. PUT CCW ADD IN CAW
ERRHR F4 STORE CAW
TRY G4 WRITE I/O EVENTS ON TAPE
H4 CHANNEL BUSY
J4 ERROR CONDITION POSSIBLE
WAIT K4 I/O COMPLETE
G5
B5
ERRCOND B5 DEVICE OPERABLE
WHICH C5 DEVICE BUSY
D5 GET CHANNEL AND UNIT ADDRESS FOR SYSLOG
E5 GET ERROR CCW ADDRESS
G5 UNIT CHECK
H5 TAPE CHANNEL AND UNIT STORED
J5 CLEAR I/O AREA

Chart FA. Transient Dump: Tape Mode (PDAIDTDT) (Part 1 of 2)



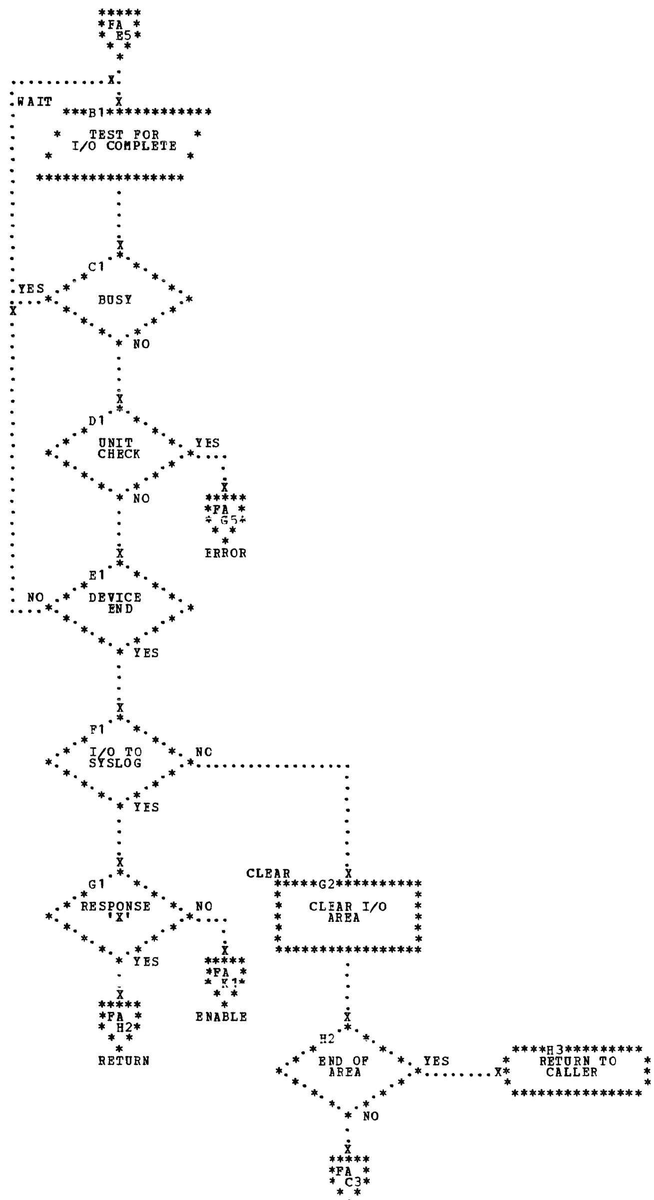
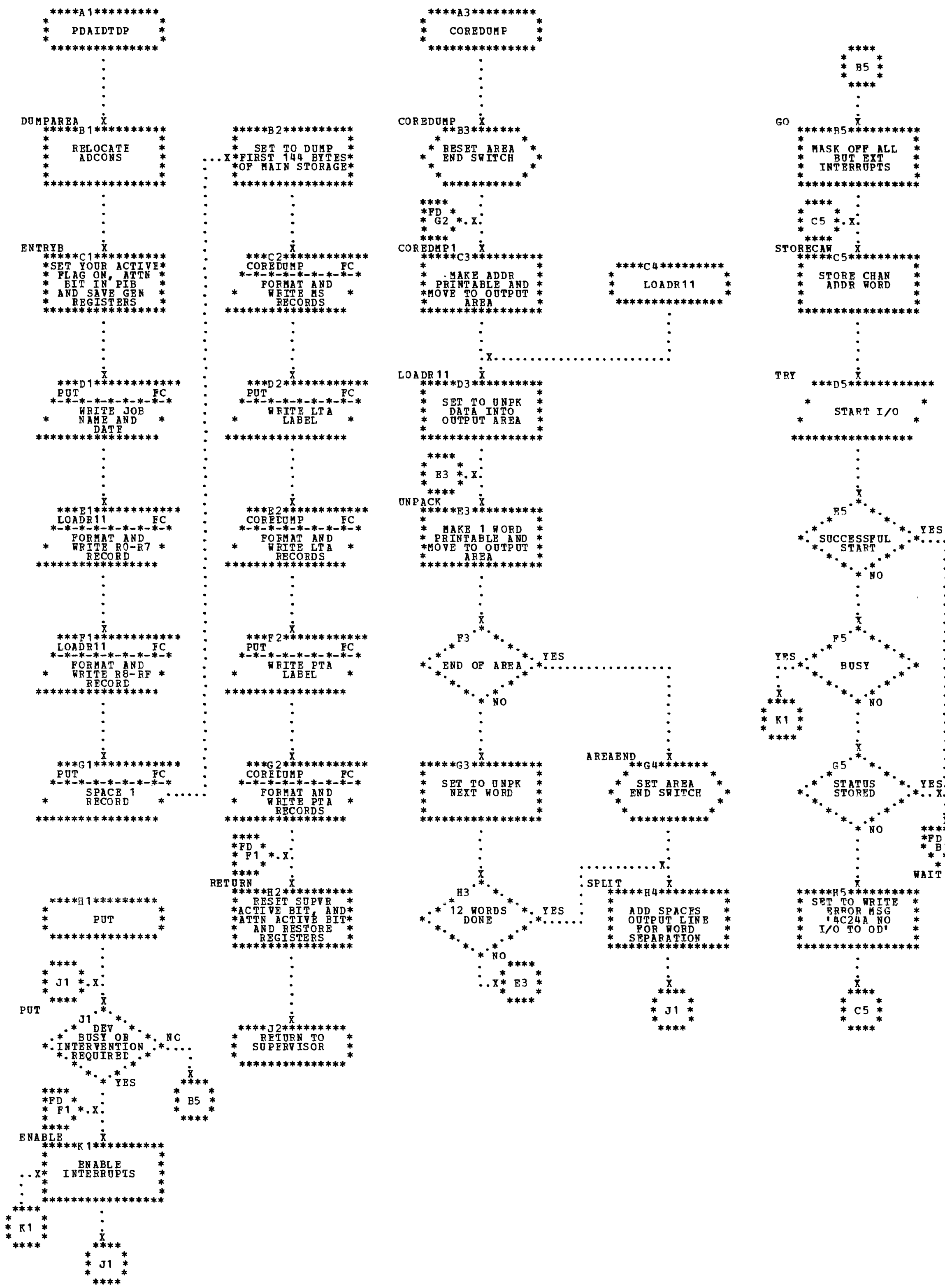
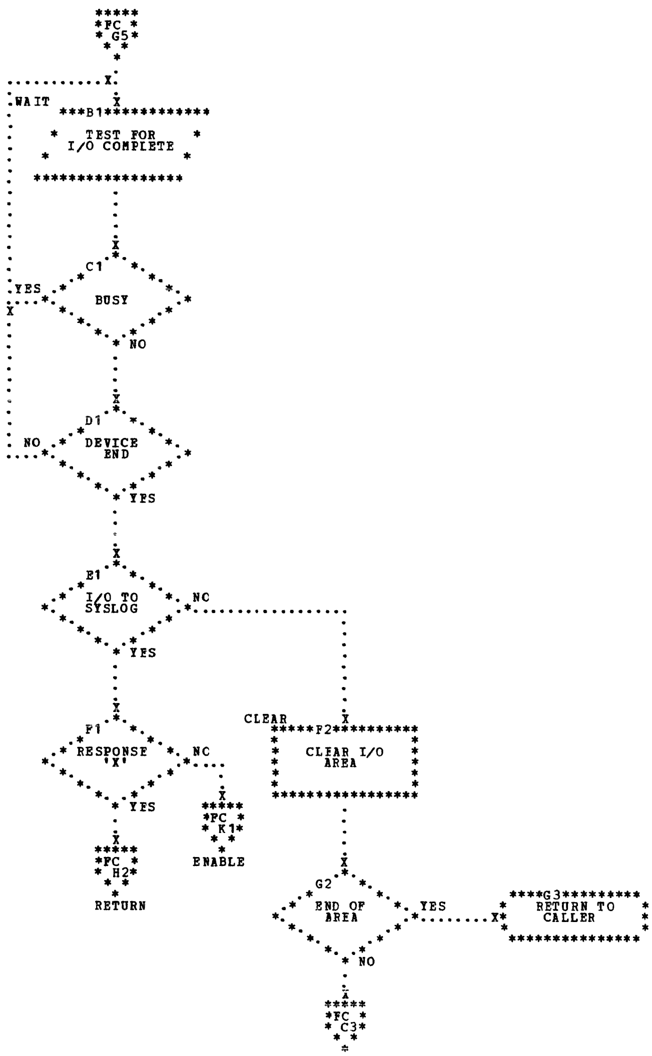

FA B5
WAIT
B1
TEST FOR I/O COMPLETE
C1
BUSY
YES
NO
D1
UNIT CHECK
YES
NO
FA G5
ERROR
E1
DEVICE END
NO
YES
F1
I/O TO SYSLOG
NO
YES
G1
RESPONSE 'X'
NO
YES
FA E1
ENABLE
FA H2
RETURN
CLEAR
G2
CLEAR I/O AREA
H2
END OF AREA
YES
NO
H3
RETURN TO CALLER
FA C3

Chart FC. Transient Dump: Printer Mode (PDAIDTDP) (Part 1 of 2)




FC G5
WAIT
B1
TEST FOR I/O COMPLETE
C1
BUSY
YES
NO
D1
DEVICE END
NO
YES
E1
I/O TO SYSLOG
NC
YES
F1
RESPONSE 'X'
NC
YES
FC H2
RETURN
FC K1
ENABLE
CLEAR
F2
CLEAR I/O AREA
G2
END OF AREA
YES
NO
G3
RETURN TO CALLER
FC C3

INTRODUCTION

SDAID provides software debugging facilities:

- For tracing the occurrence of user-selected program or monitor call events
- For dumping parts or the whole real storage at the time one or more of the selected events occurs.

SDAID tracing and dumping facilities are provided by the utilization of the two following System/370 features:

- Program Event Recording (PER)
- Monitor Call Event.

The occurrence of a designated program event or monitor call event is signaled to the program by a program interruption if the CPU is enabled for this condition. Control registers 9, 10, 11 and PSW bit 1 (PER bit) are dedicated to control program events interruptions.

The SDAID program can be executed in any real or virtual partition and is initiated by entering // EXEC SDAID via SYSLOG or SYSRDR. An operator/system dialog follows, beginning with the message

4C55D GIVE SPACE FOR SDAID =

The messages issued by SDAID are listed in Appendix A of this manual.

More information on the usage of the SDAID program can be found in Section 2-B, Part 2 of _DOS/VS Serviceability Aids and Debugging Procedures_, GC33-5380.

To get control as soon as a program interruption occurs, the tracing and dumping routines must reside permanently in real storage.

SDAID uses the two following DOS/VS inner functions to get the number of page frames from the page pool necessary to accommodate the tracing and dumping routines and to return them to the page pool when SDAID is no longer desired:

- GETREAL Get real storage
- FREEREAL Free real storage.

The tracing and dumping routines operate with events occurring both in the DOS/VS supervisor resident or transient area and in the problem program area.

SDAID uses the following DOS/VS inner functions to temporarily fix all pages in which fields must be initialized for subsequent tracing and dumping and to free them after initialization:

- TFIX Temporary fix a page frame
- TFREE Free a previously temporarily fixed page frame.

This chapter provides information to familiarize the reader with the external characteristics of the two System/370 features and the four DOS/VS inner functions. It will also facilitate understanding of SDAID internal program logic.

PROGRAM EVENT RECORDING (PER)

The Program Event Recording feature permits the program to be alerted of the following events:

- Successful execution of a branch instruction
- Successful fetching of an instruction between predefined limits of a logical storage area
- Alteration of the contents of storage between predefined limits of a logical storage area
- Alteration of the contents of one or more designated general registers.

The information concerning a program event is provided to the program by means of a program interruption, whose cause is identified in the interruption code.

CONTROL REGISTER ALLOCATION

The information for controlling program event recording resides in control registers 9, 10, and 11 (see Figure 20).

CR 9 | PER control bits | | PER general register mask |
0 8 16 31

PER control bits	Bit 0	Successful branch
	Bit 1	Successful instruction fetch
	Bit 2	Storage alteration
	Bit 3	General register alteration
	Bit 4-7	Unassigned

Bits 0-3	0	corresponding events ignored
	1	corresponding event will cause a program interruption

PER general register mask The 16 bits (16-31) correspond one for one with the 16 general registers (0-15) in ascending order.

Bits 16-31	0	corresponding register alteration is ignored
	1	corresponding register alteration will cause a program interruption

CR 10 | | PER logical start address |

CR 11 | | PER logical end address |

Bits 8-31 of control registers 10 and 11 designate the start and end address of the logical storage area for instruction fetch and storage alteration program event recording.

Figure 20. PER Control Information in Control Registers 9, 10, and 11

OPERATION

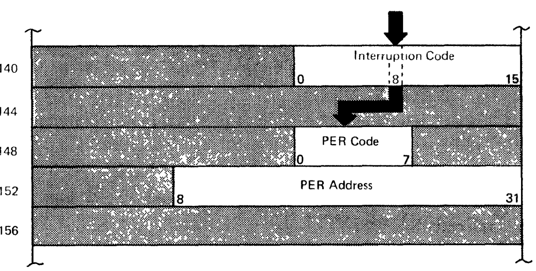
Operation of program event recording is controlled by PSW bit 1 (PER bit).

PSW Bit 1 = 0 : Program events are ignored

= 1 : Program events are permitted subject to the setting of PER control bits in control register 9.

When a designated program event occurs, with the CPU enabled for this condition, a program interruption will occur. The events causing the interruption are identified by the PER code in the program interruption code extension. When one or more bits in the PER code are on, bit 8 of the interruption code in the program interruption code extension is set to one. That is to say, bit 8 of the interruption code indicates whether or not PER action is required (see Figure 21).

The location of the instruction associated with the event is identified by the PER address in the program interruption extension in low real storage.



Program Interruption code extension in low core

PER Code	Bit 0	Successful branch
	Bit 1	Successful instruction fetch
	Bit 2	General register alteration
	Bit 3	Storage alteration
	Bit 4-7	Unassigned

Bit 0-3	0 = corresponding event did not occur
	1 = corresponding event occurred

Figure 21. Layout of Low Real Storage Used for Checking Whether or Not a Specific Event Occurred

MONITOR CALL EVENT

The monitor call event is defined by the execution of a monitor call (MC) instruction, the format of which is:

```
   MC D1 (B1), I2 (SI)

┌─────────┬─────────┬───────┬───────────┐
│   AF    │   I2    │  B1   │    D1     │
└─────────┴─────────┴───────┴───────────┘
 0       7 8      15 16   19 20        31
```

Bits 12-15 of the monitor call instruction identify one of the sixteen possible monitor call classes and are set up from the immediate operand value (I_2).

The first operand field (B1) + D1 is used to identify the monitor call code for the specified class.

CONTROL REGISTER ALLOCATION

The information for controlling monitor call events is contained in control register 8.

```
     ┌─────────────┬──────────────────────────┐
CR 8 │             │Monitor Call Class Mask   │
     └─────────────┴──────────────────────────┘
      0             16                      31
```

Monitor call class mask:	The 16 bits (16-31) correspond one for one with the 16 possible monitor call classes (0-15) in ascending order.
Bits 16-31 = 0 :	Execution of a monitor call instruction for the specified class will be treated as a NOP instruction.
= 1 :	Execution of a monitor call instruction for the specified class will cause a program interruption.

OPERATION

When a designated monitor call instruction is executed, with the monitor call class mask bit enabled for the specified class, a program interruption will occur.

The monitor call class (I2 field) causing the interruption is identified by the monitor class field in the program interruption code extension.

Bit 9 of the interruption code, in the program interruption code extension, is set to one.

The monitor code ((B1) + D1 field) associated with the instruction is identified in the monitor code field in the program interruption code extension.

The program interruption code extension is shown in Figure 22.

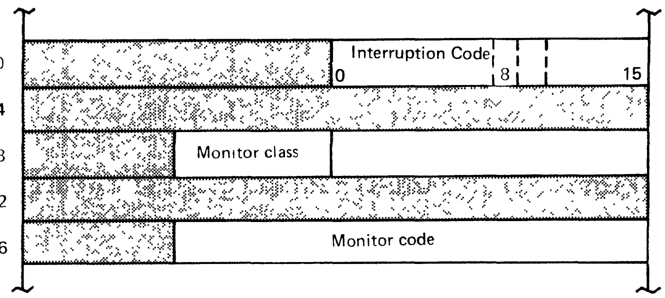


Figure 22. Program Interruption Code Extension

GETREAL - FREEREAL

Access to DOS/VS GETREAL and FREEREAL functions is provided by the two following SVCs:

SVC 55	Allocates page frames from the main page pool at the end of real storage for the PDAID alternate area or for the SDAID area.
SVC 54	Gives back to the page pool all the page frames allocated by a preceding SVC 55.

Figure 23 shows the parameters passed and returned by SVC 55 and SVC 54.

TFIX - TFREE

Access to DOS/VS TFIX and TFREE functions is provided by an offset table in low real storage.

TFIX	Attaches a page frame to a specified page and temporarily fixes that page frame so that it can no longer be swapped out.
TFREE	Frees a temporarily fixed page frame.

Figure 24 shows the parameters passed by TFIX and TFREE.

SVC	Macro	Parameter passed		Parameter returned	
		R0	R1	R0	R1
54	--	--	--	--	--
55	--	Number of page frames asked.	--	Number of page frames allocated.	Beginning address of allocated area. Zero if duplicate request.

Figure 23. Parameters Passed and Returned by SVCs 55 and 54.

Function	SVC	Macro	Parameter passed	Parameter returned
TFIX	-	-	R2 - Any logical address within the page to be fixed	-
TFREE	-		R3 - No. of page frame to be freed	-

Figure 24. Parameters Passed by TFIX and TFREE

GENERAL DESCRIPTION

The SDAID tracing/dumping facilities are performed by the SDAID event handling routines (phase SDEHR). The SDEHR phase takes control as soon as a program check occurs and operates in supervisor state with a key of zero. For this reason it must permanently reside in real storage as long as the tracing/dumping facilities are desired.

To meet these requirements a number of page frames, which can be specified by the user and cannot be less than three (6K bytes), is taken from the page pool at the end of real storage. These pages are then assigned to an area called SDAREA in which the SDEHR phase resides along with a core wrap recording buffer (SDBUF).

The set of SDAID routines can be divided into three parts distinguished by the time of their execution (see Figure 25).

- Initialization
- Operation
- Termination.

This section gives a general description of these three parts. In addition, it describes the inter-phase communication and the layout of the SD area in the supervisor area.

Part of SDAID	Phase	Module	CSECT	State	Key	Execution Time	Execution Area	Function
INITIALIZATION	SDAID	IJBSDAD1	SDAID1 PRTMOD CSLMOD	P	Partition	Initialization	Partition	Get real storage for SDAREA LIOCS for SYSLST LIOCS for SYSLOG
	SDPAR	IJBSDPAR	SDPAR	P	Partition	Initialization	Partition	User option acceptance routine
	SDAID2	IJBSDAD1 IJBSDAD2	SDAID2 SDEHR	P	Partition	Initialization	Partition	Print out selected options This phase will be moved to SDAREA
	$$BSDAID			P/SV	0	Initialization	L T A	Move SDEHR to SDAREA. Activate PSWs and Control Registers to perform selected options
OPERATION	SDEHR	IJBSDAD2	SDEHR	SV	0	Operation	SDAREA	Analyze program check and perform tracing/dumping
TERMINATION	ENDSD	IJBENDSD	ENDSD	P	Partition	Termination	Partition	Call $$BATTN3
	$$BATTN3			P/SV	0	Termination	L T A	Printout SDBUF Return SDAREA to page pool Deactivate PSWs and control registers

Figure 25. Summary of SDAID Program

COMMUNICATION BETWEEN PHASES

Communication between phases takes place by means of a communication region referred to as parameter list (MONPARM). This parameter list is divided into two sections. The first section (see Figure 26) contains parameters to control the desired tracing and dumping facilities of the operation phase SDEHR. It is transferred to SDAREA at the end of SDEHR.

The second section (see Figure 27) contains parameters to control the initialization process of the operation phase SDEHR.

MONWRAP	Start address of core wrap buffer		(SDBUF)
MONEWRAP	End address of core wrap buffer		(SDBUF)
MONWPNT	Next available entry in core wrap buffer		(SDBUF)
MONPRTR Printer address 0cuu		Reserved	
MONTE (1)	Reserved	MONCHK (1)	Reserved
MONENQ (1)	Reserved	MONHDL (1)	Reserved
MONHALT (2)	Reserved	NONDD (3)	Reserved
MONOUTCL Output class		Reserved	
MONDSTRT	Reserved		
MONDEND	Reserved		

(1) MONTE Translation exception event switch (Interruption code X'11')

MONCHK Program check event switch (interruption code X'01' to X'12' except X'11')

MONENQ Page fault enqueue event switch (Monitor Call class 5, code X'000010')

MONHDL Page fault handling event switch (Monitor Call class 5, code X'000011')

Switch X'00' Event is ignored
X'FF' Event is enabled - recording takes place

(2) MONHALT Stop on event switch

Switch X'00' Resume processing after event recording
X'FF' Enter wait state after event recording

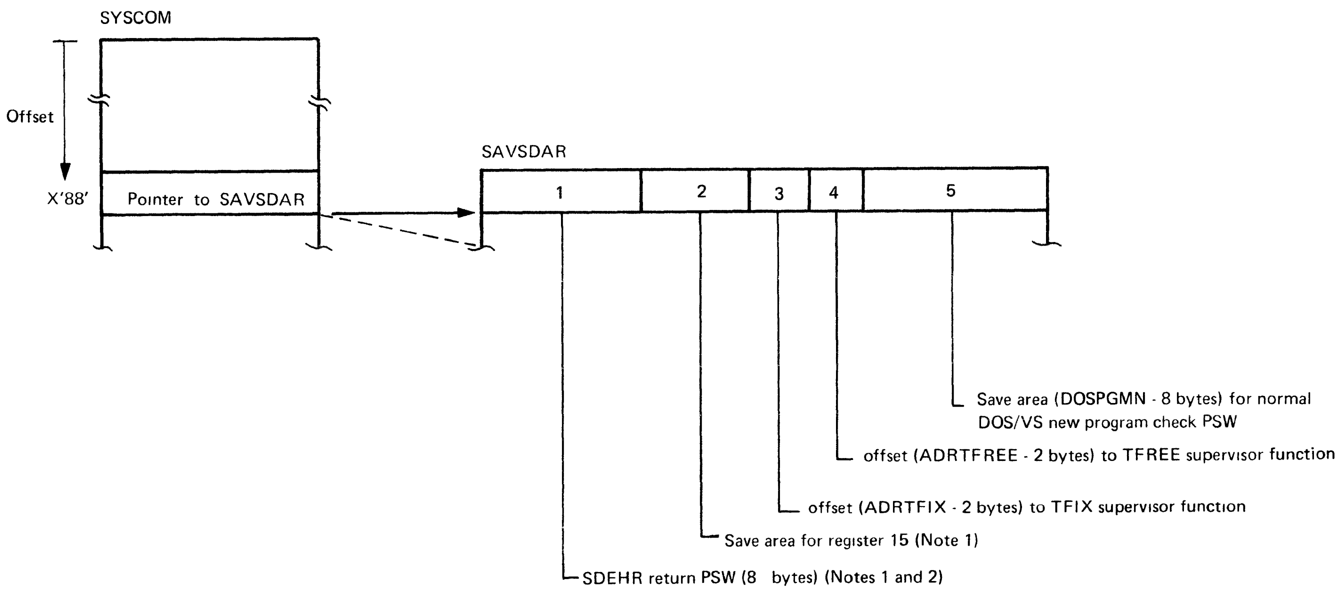
(3) NONDD Non-destroying dump switch.
This switch is only tested if MONHALT switch ON. It is set by the operator when system enters wait state and is reset to X'00' by the program after real storage dump is ended.

Switch X'00' No real storage dump
X'FF' Dump whole real storage

Figure 26. Section 1 of Parameter List (MCNPARM)

MONCR08	Control Register 8		
MONCR09	Control Register 9		
MONCR10	Control Register 10		
MONCR11	Control Register 11		
MONREAL	Start address of SDAREA		
MONRELAD	Start address of SDEHR CSECT in phase SDAID2		
MONDIFF	Length of SDEHR CSECT	Reserved	
MONCPU	CPU identification		
MONDUB	Address in SAVSDAR of return PSW	MONDUBX address in SAVSDAR for R15 Save	
	Reserved	MONMPER X'40'	Reserved
MONLDUB	Addresses in SDEHR CSECT where to move the address of the return PSW from MONDUB field (See SDAID area in supervisor)		
MONLDUBX	Addresses in SDEHR CSECT where to move the address of the save area for R15 from MONDUBX field. (See SDAID area in supervisor)		
NPSDOSAD	Address in SDEHR where to move address of DOSPGMN (SAVSDAR field 5)		
MONFIX	These are addresses in phase SDAID2. They are used by $$BSDAID to temporarily fix the pages within which they lie.	Start address of 1st Section of Parameter list	
		End address of 1st Section of Parameter list	
		Start address of SDEHR CSECT	
		End address of SDEHR CSECT	
		Reserved	

Figure 27. Section 2 of Parameter List (MONPARM)



Note 1: Fields 1 and 2 may be inverted (if SAVSDAR is not aligned on a doubleword boundary) to allow PSW load to return from SDEHR routines.

Note 2: If there is an interruption due to a monitor call for a class different from 0, 1, and 2, or due to a PER event without another program check interruption, this field contains the old program check PSW at interruption time. Otherwise the contents of field 5 are moved into this field.

Figure 28. Layout and Contents of SAVSDAR

SDAID AREA IN SUPERVISOR

An area (SAVSDAR) is provided in the supervisor area for SDAID (see Figure 28). It is mainly used for saving, but contains also a table that gives the offsets to TFIX and TFREE supervisor functions.

INITIALIZATION ROUTINES

A real partition of at least 12K or any virtual partition is needed to initalize SDAREA.

The following functions are performed by the three initialization phases and the initialization transient (see also Figure 29):

SDAID — Get page frames from the page pool and allocate them to SDAREA.

SDPAR — Initialize an operator-to-system communication to set up the operator-selected tracing and dumping options in the parameter list (MONPARM).

SDAID2 — Print on SYSLST the selected or default tracing/dumping options and give control to the initialization transient. Upon return from the transient, post the successful initialization and release the partition.

$$BSDAID — Initialize the event handling routines (SDEHR) and move them to the SDAREA. Initialize PSWs and control registers to perform selected tracing/dumping options.

The storage layout during initialization is shown in Figure 30.

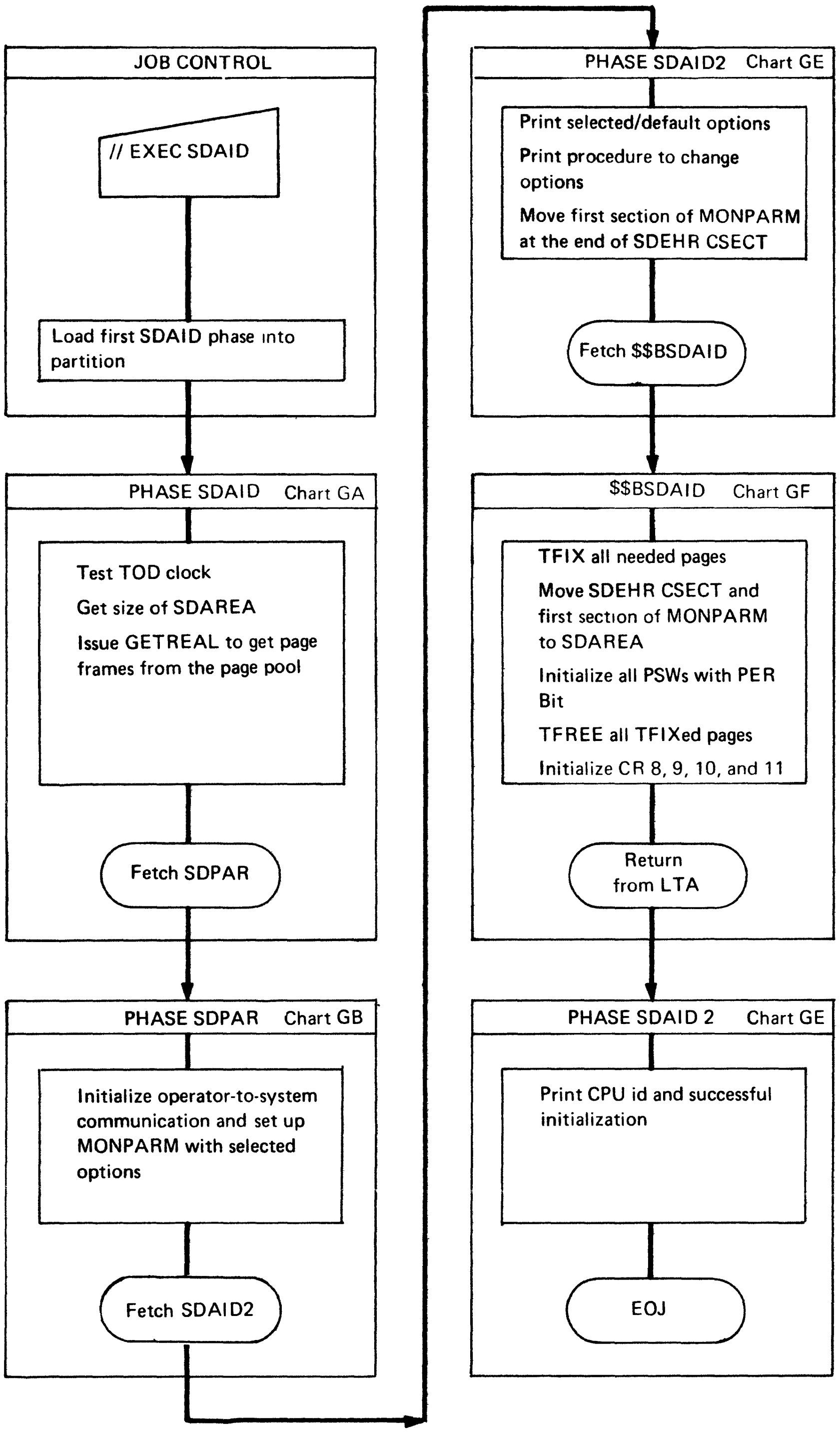


Figure 29. General Logic Flow of SDAID Initialization Routines

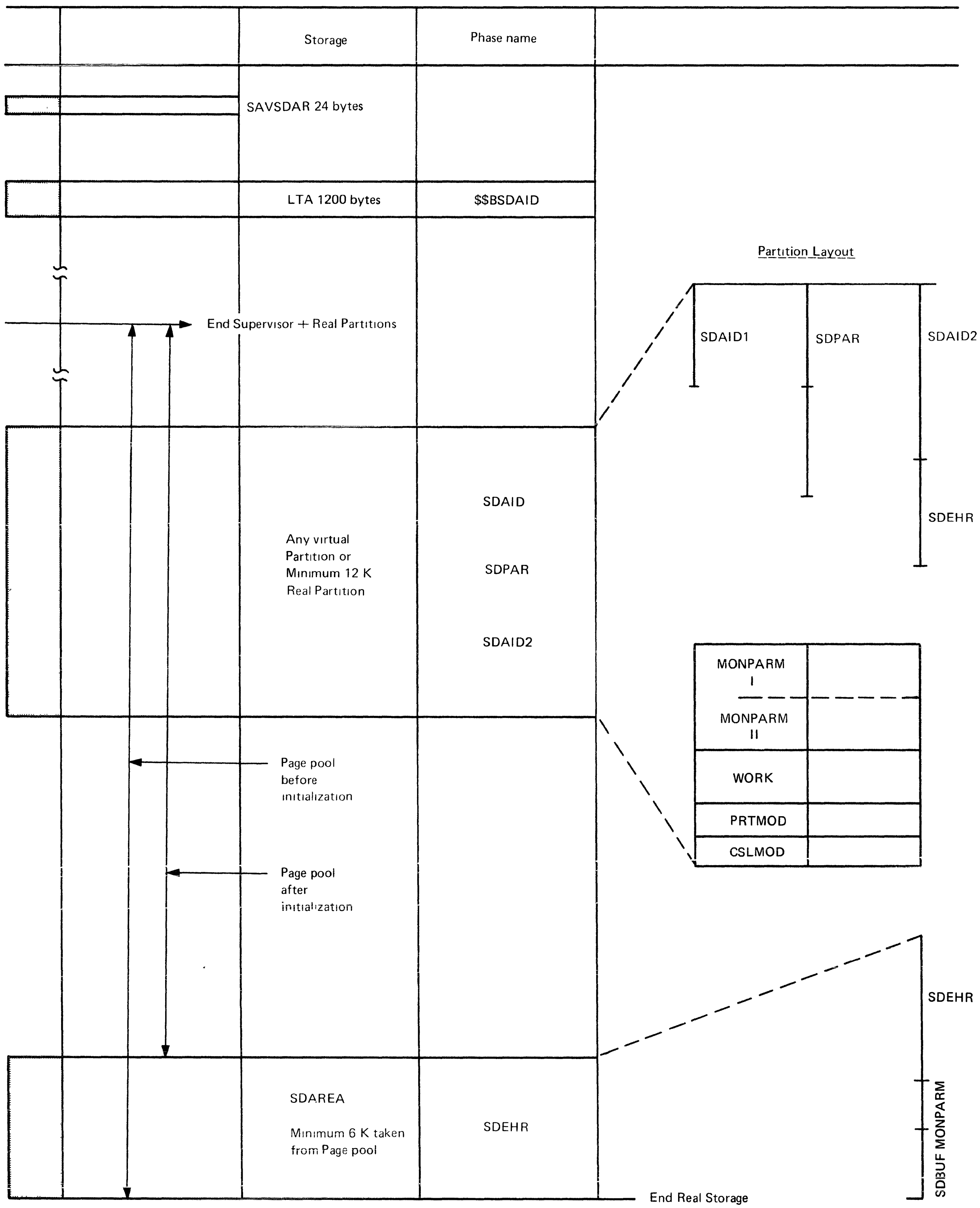


Figure 30. Storage Layout During Initialization

Figure 31 is a tabular summary of the logic flow of phase SDAID (compare Chart GA).

	Entry from job control
	OPEN SYSLST and SYSLOG
	Store TOD clock to determine status of the clock. If TOD clock in not-set state, MSG 4C53I is displayed If TOD clock in not-operational state, MSG 4C54I is displayed.
A	MSG 4C55D is displayed to ask the size of real storage to be allocated to SDAREA. A read operation is initiated to get the reply. If End is pressed, force a default value of 6K and go to B. If syntax is incorrect, MSG 4C17D is displayed. Go to A.
	The storage size is converted to binary and rounded up if necessary to an even integer. If zero K are asked, MSG 4C17D is displayed. Go to A. If less than 6K are asked, force a default value of 6K.
B	The storage size is divided by two to get the number of page frames to be removed from the page pool.
	GET REAL (SVC 55) is issued. If duplicate request, MSG 4C70E is issued and job is terminated. If less than 3 page frames are obtained, MSG 4C56I is issued, the allocated page frames are freed by issuing FREEREAL (SVC 54), and the job is terminated.
	Exit by fetching phase SDPAR.

Figure 31. Logic Flow of Phase SDAID

PHASE SDPAR

Processing of SDPAR routines is under control of the parameter description table included in this phase.

For each option, which the operator can select to set up the designed tracing and dumping facilities, the parameter description table contains one keyword entry.

For each keyword displayed the operator can either choose the default values for this option by pressing the END key or alter the default values by entering, depending on the keyword, one or more parameters separated by commas.

For each parameter entered by the operator for a specified keyword, the parameter description table contains, depending on the parameter, a subset of one or more alternatives.

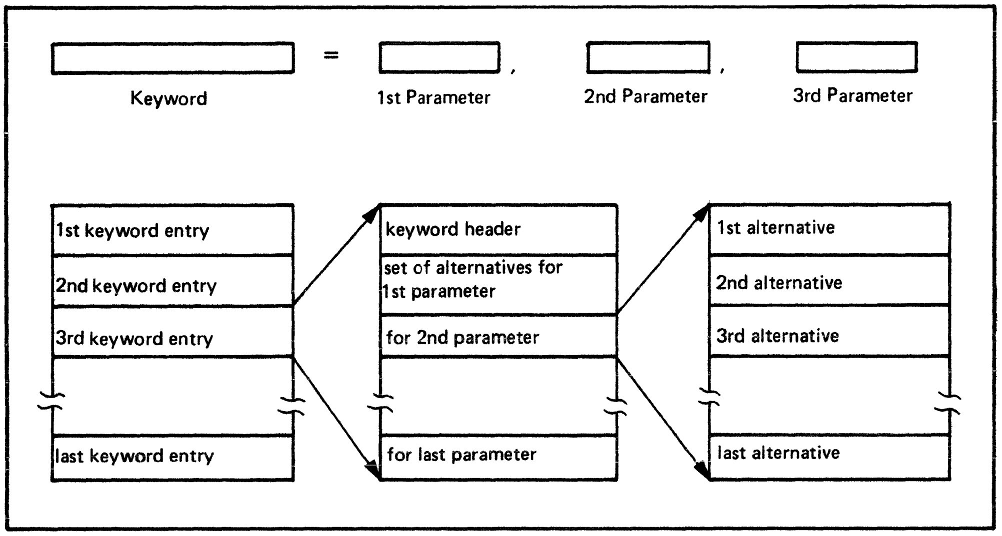
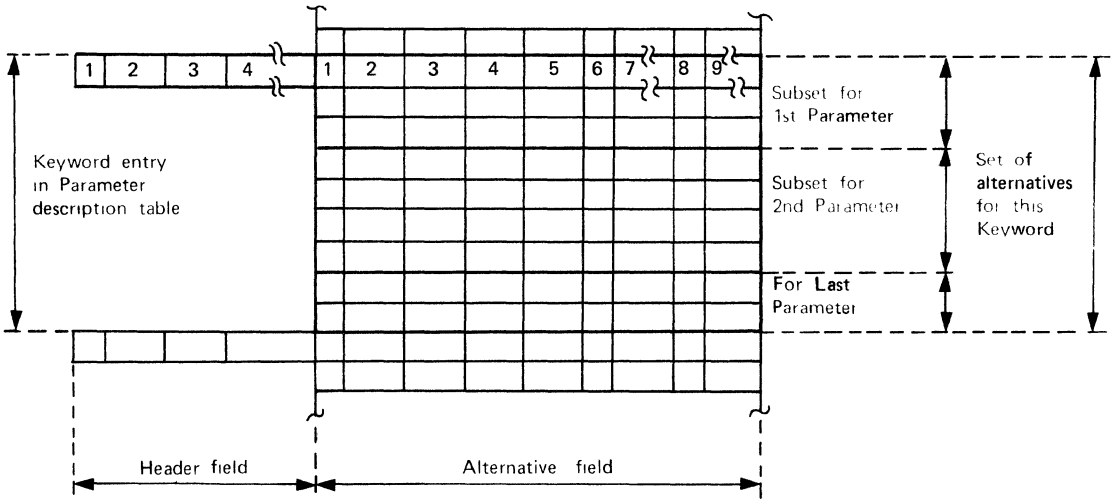


Figure 32. Logical Structure of the Parameter Description Table

The parameter description table thus consists of several keyword entries. Each keyword entry is composed of one header field followed by a set of one or more alternatives fields. The set of alternatives fields is divided into as many subsets of alternatives as there are possible parameters in the reply for the specified keyword. This logical structure of the parameter description table is shown in Figure 32. The physical structure of this table is shown and explained in Figure 33.



Field	Length	Header field
1	1	Length of keyword in field 4
2	2	Offset to the next keyword entry - X'FFFF' if last keyword entry
3	2	Reserved
4	Given by 1	Keyword to be displayed

		Alternative field
1	1	Type of reply (1)
2	2	Offset to special check routine if type of reply = VAR. Else X'0000'
3	2	Offset to next alternative for this parameter. X'FFFF' if last alternative
4	2	Offset to next subset of alternative for this keyword. X'0000' if last subset
5	2	Offset to the field to be set up in parameter list
6	1	Length of the information to be set up in parameter list (2)
7	Given by 6	Information to be moved, ORed, or ANDed in parameter list (2)
8	1	Length of predefined parameter if type of reply = FIX for this alternative (2)
9	Given by 8	Predefined reply for this alternative (2)

(1)	Type of reply	VAR	X'80'	
		FIX	X'08'	Field 7 will be moved in parameter list
			X'04'	Field 7 will be ORed in parameter list
			X'02'	Field 7 will be ANDed in parameter list

(2) This field is only available if type of reply = FIX

Figure 33. Physical Structure of the Parameter Description Table

Figure 34 is a tabular summary of the logic flow of phase SDPAR (compare flowcharts GB-GD).

	MAIN LINE
	Entry from phase SDAID.
	Get first keyword from the Parameter Description Table.
A	Link to TYPOUTRT to display this keyword.
	Link to TYPINRTN to read in the operator's reply. If return code indicates that end key was depressed, go to C.
B	Link to TSTBYTRT to test the parameter in input area. If return code indicates syntax error in reply, display MSG 4C17D and go to A.
	Link to CHKNSWRT to process this parameter. If return code indicates that an error was found in the parameter, display MSG 4C17D and go to A. If return code indicates that there is another parameter in input area, go to B. If return code indicates that this parameter was 'GO', go to D.
C	If last keyword from Parameter Description Table just displayed, go to D.
	Get next keyword in Parameter Description Table. Go to A.
D	Exit by Fetch SDAID2.

Figure 34. Logic Flow of Phase SDPAR (Part 1 of 3)

	TSTBYTRT Subroutine
	If the first character of the parameter in input area is a blank or a comma, go to B.
A	Test next character of the parameter. If X'00', indicate that this parameter is the last in input area and go to C. If X'6B' (comma), indicate there is another parameter in input area and go to C. If maximum length of reply not reached, go to A. If maximum length reached and next character is X'00', indicate last parameter in input area and go to C.
B	Indicate syntax error in return code.
C	Return to caller.

Figure 34. Logic Flow of Phase SDPAR (Part 2 of 3)

	CHKNSWRT subroutine
	If first parameter for this keyword take header offset to point to first alternative of the first subset for this keyword.
A	Test reply type for this alternative (field 1). If VAR, check if parameter is correct by branching to the special check routine. If YES, move information to MONPAR and go to C. If NO, go to B. If FIX, compare parameter with predefined value in field 9 of this alternative. If both match and parameter not 'GO', MOVE, OR, or AND information in MONPAR and go to C. If both match and parameter is 'GO', indicate the 'GO' condition in return code and go to D.
B	Test if last alternative for this subset. If YES, indicate parameter error in return code and go to D. If NO, point to next alternative for this subset and go to A.
C	Test if another parameter in input area. If YES, test if another subset for this keyword. If YES, point to first alternative of next subset, indicate another parameter in input area and go to D. If NO, indicate parameter error in return code.
D	Return to caller.

Figure 34. Logic Flow of Phase SDPAR (Part 3 of 3)

Figure 35 is a tabular summary of the logic flow of phase SDAID2 (compare flowchart GE).

	SDAID2
	Entry from phase SDPAR.
	Print selected and default tracing and dumping options. Print the procedure to change the options dynamically. Move first section of the parameter List to the end of the SDEHR CSECT.
	Fetch initialization transient $$BSDAID.
	Print the CPU identification. Print message 4C57E. Close SYSLST and SYSLOG.
	Exit to Job Control.

Figure 35. Logic Flow of Phase SDAID2

TRANSIENT $$BSDAID

Processing of $$BSDAID is under control of the three tables described below. The first two tables are built by $$BSDAID, the last one is partially built by $$BSDAID.

- Control List Table

  This table has 15 entries. Each entry corresponds one to one to the PIB of the 15 possibly running tasks or subtasks. Each entry contains the address of the corresponding PIB if the task is active, or zero if it is inactive. This table is used to build the other two tables.

- Fix List Table
  This table has 21 entries. Each entry contains, after initialization, the logical address within a page to be temporarily fixed.

  If SDAID2 runs in a virtual partition, the first five entries are used to accommodate the five addresses from the parameter list field MONFIX (see Figure 27). The sixth entry contains the address of the PSW in the partition save area. If SDAID2 runs in real mode, the first six entries of the Fix List table remain zero.

  Each of the 15 remaining entries contains the address of the PSW in the save area of each task or subtask when they run in virtual mode, or zero if they run in real mode. The entry that corresponds to the SDAID2 partition is always zero.

- PER List Table
  This table has 56 entries. Each entry contains, after initialization, the address of a PSW in which the PER bit must be set to 1.

  The first 16 entries overlap with the last 16 entries of the Fix List table. Entry 17 contains the address of the PSW in the LTA save area. If SDAID2 runs in real mode, entry 18 contains the address of the PSW in the partition save area.

  Each of the next 15 entries (entries 34 through 48) contains the address of the PSW in the system save area of each task or subtask if active.

  Entry 49 contains the address in SAVSDAR where the normal DOS/VS new program PSW has been saved.

  The last seven entries contain the addresses of all DOS/VS new and old PSWs in low real storage in which the PER bit must be set.

Figure 36 is a tabular summary of the logic flow of the transient $$BSDAID (compare flowchart GF)

$$BSDAID
Entry from phase SDAID2. Save DOS/VS normal new program PSW in field 5 of SAVSDAR. Seize the system and force a program check to get control in supervisor state. Build Control List table. Build Fix List table. Build PER List table. Temporarily fix all required pages using Fix List table. Initialize the SDEHR CSECT with the following addresses: The address of the return PSW in SAVSDAR. The address of the register 15 save area in SAVSDAR. The address of the normal DOS/VS program PSW in SAVSDAR. Move the SDEHR CSECT to SDAREA. Initialize all PSWs with PER bit using the PER List table. Load control registers 8, 9, 10, and 11 to perform specified tracing/dumping functions. Get the CPU identification and store it in MCNPARM. Free all temporarily fixed pages using the Fix List table. Initialize the new program interruption PSW with the address of the entry point of SDEHR in SDAREA. Note: SDAID tracing/dumping functions are active from now on. Release the system. Exit to caller.

Figure 36. Logic Flow of Transient $$BSDAID

EVENT HANDLING ROUTINES

The event handling routines (phase SDEHR) perform all the tracing and dumping functions of the SDAID program. They reside permanently in SDAREA where they are moved by the initialization routines. Phase SDEHR gets control as soon as a program interruption occurs and operates in supervisor state with a key of zero and disabled for I/O interrupts, external interrupts, and program check interrupts due to PER events.

Two ways of recording may be selected:

- Core mode (output class 00)
  The recording of all specified events takes place in a core wrap buffer (SDBUF). The number of events which can be kept in this buffer depends on the size of storage allocated to SDAREA during initialization. If the output mode is automatic, the buffer is printed out on the specified printer as soon as it is full. Otherwise the buffer is overlaid by the newest one.

- Print mode (output classes 01-09)
  All specified events are recorded on the specified printer at the time the event occurs. The amount of information given to the user depends on the output class selected.

To perform the I/O operations an integrated printer IOCS takes control of the specified printer so that it is possible to trace events along with customer's output on the same printer.

Phase SDEHR uses (1) the 24-byte SAVSDAR in the supervisor and (2) a minimum of 6K at the high end of real storage. The latter area (called SDAREA) is organized as follows:

1. The phase SDEHR

2. The parameter list MCNPARM (see section 2)

3. The core-wrap buffer SDBUF

The set of event handling routines can be divided into the five groups shown in Figure 37. The logic flow from groups of routines and from individual routines to one another is illustrated in Figure 38.

SDEHR Event Handling Routines		
Group	Name	Function
Event Handler Routines	DISPATCHER	Analyze the interruption code and transfer control to the appropriate routines
	PEREVENT	Process PER events
	MCEVENT	Process MC events, page handling and page enqueue
	DATEVENT	Process page translation exception event (code X'11')
	PGMEVENT	Process program check events (codes X'01' to X'12' except X'11')
	STOPONEV	Receive control if Stop-on-event switch (MONHALT) is on
Recording Routines	FASTRECX	Record the event in the core wrap buffer (SDBUF)
	PRINTHD	Print heading information
	PRINTHDMC	Print heading information for page enqueue and handling
	PRINTIT	Print output as specified by output class option
	PRINTBUF	Print the core wrap buffer (SDBUF)
Dumping and Editing Routines	DUMPSUB	Edit and print storage
	SPECDUMP	Edit and print storage
	EDITXBYT	Translate one or more bytes from HEX to EBCDIC
	EDITREGX	Translate one or more groups of four bytes from HEX to EBCDIC
	TRANSTD	Translate TOD clock
	INSTxxx	Translate six bytes instruction from HEX to EBCDIC
Printer IOCS Routines	MONOPEN	Take control of the specified printer
	MONPUT	Issue SIO and test condition code
	MONCLOSE	Return printer to DOS/VS
	CSWxxxx	Analyze the CSW
	NOTOPO	Process not operational status
	ERRSIO	Issue sense SIO or SIO in case of an error condition
Miscellaneous Routines	TESTLIM	Test whether event between logical storage address limits
	SAVEXT	Save previous external conditions
	RESTEXT	Restore previous external conditions
	BLANKRTN	Get a work area

Figure 37. Summary of Routines Contained in SDEHR

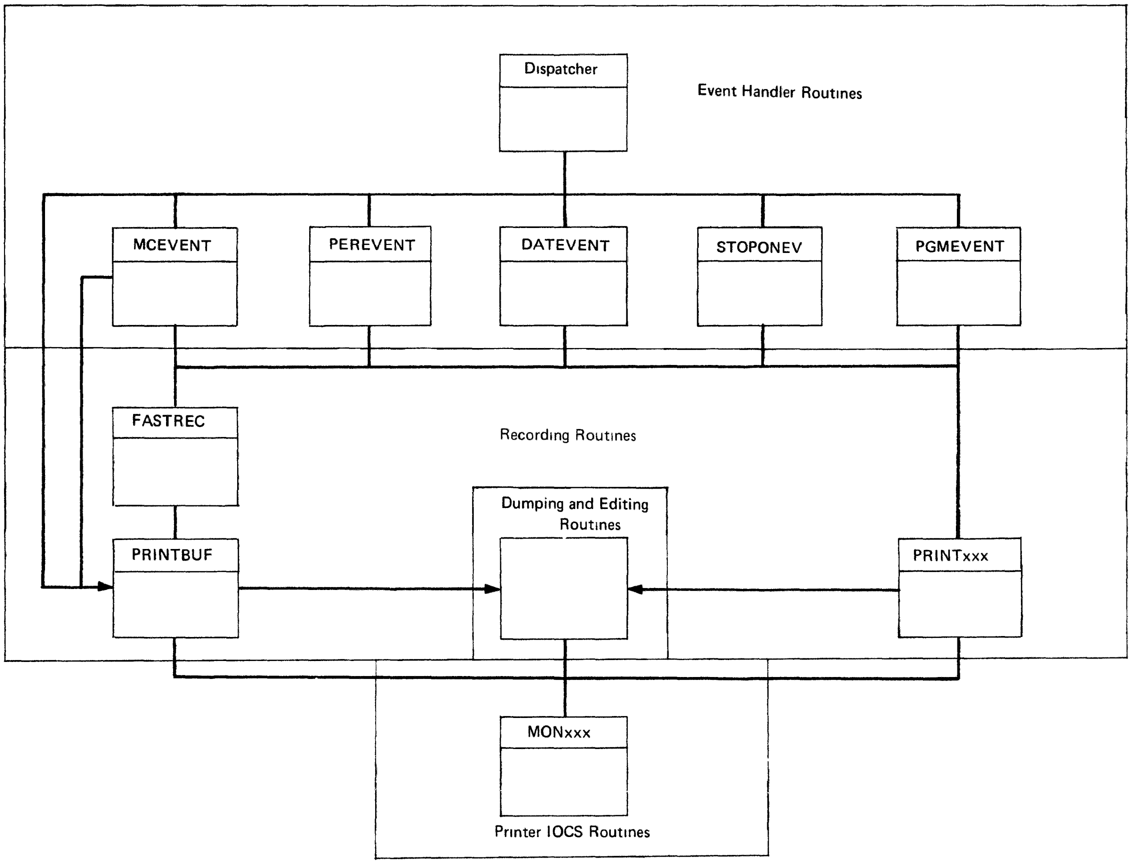


Figure 38. Flow of Logic Between Routines and Groups of Routines

CORE WRAP BUFFER (SDBUF)

When output class equals FASTREC, the recording of the specified events takes place in the core wrap buffer (SDBUF). The size of this buffer depends on the amount of storage allocated to SDAREA during initialization. Each entry in SDBUF is 64 bytes long. The information stored by the recording routine is shown in Figure 39.

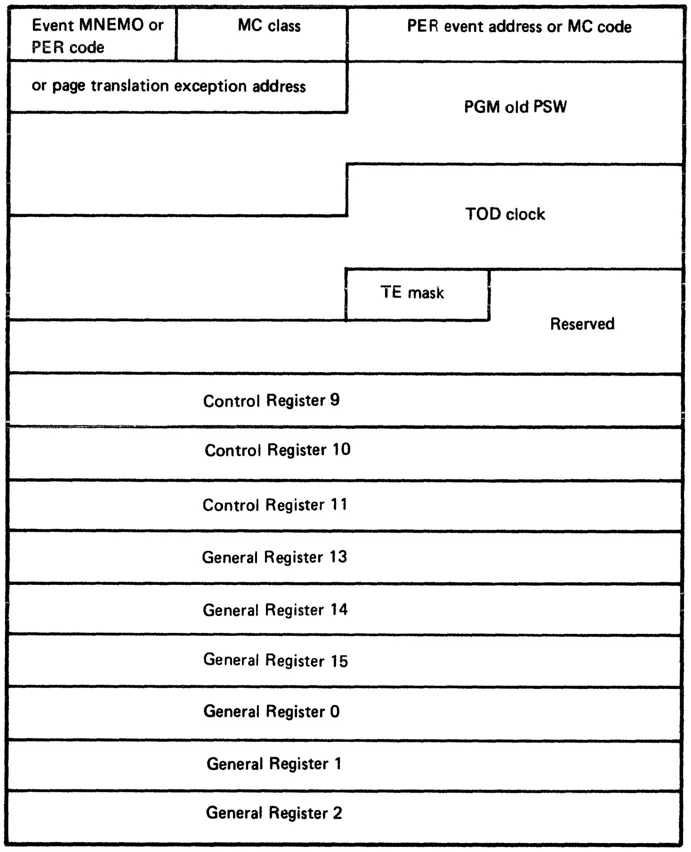


Figure 39. Format of Core-Wrap Buffer (SDBUF)

OPERATION

At initialization time, MONWRAP in the parameter list contains the SDBUF start address, MONEWRAP contains the SDBUF end address, and MONWPNT points to the first entry in SDBUF.

Each time recording is performed, MONWPNT is updated with the address of the next entry. When the last entry is filled, the SDBUF is printed out if recording mode automatic is specified, and MONWPNT is updated to point again to the first entry. Otherwise MONWPNT is updated to point to the first entry so that the oldest event is overlaid by the new one.

The logic flow of the individual routines of phase SDEHR is described in tabular form in the remainder of this section.

EVENT HANDLING ROUTINES

Figure 40 is a tabular summary of the logic flow of the Event Handling Routines.

DISPATCHER	
	Entry from program interruption new PSW.
	Store register 15 in field 2 of SAVSDAR.
	Replace the instruction to store register 15 by a branch instruction to get control at point G in case a program interruption occurs during SDEHR processing.
	Save machine check new PSW address and replace it to get control at point F in case a machine check interruption occurs during SDEHR processing.
	Move old program PSW to field 1 of SAVSDAR.
	If PER event (X'80'), give control to PEREVENT routine.
A	If MC event (X'40'), give control to MCEVENT routine.
B	If TE event (X'11'), give control to DATEVENT routine.
C	If program check event (X'01' through X'12' except X'11') and if program check event enabled (switch MONCHK in MONPARM), give control to PGMEVENT routine.
D	If stop on event enabled (switch MONHALT in MONPARM), give control to STOPONEV routine.
E	If interrupted by program check (X'01' through X'12'), save old program PSW and interruption code and move normal DOS/VS new program PSW from field 5 to field 1 of SAVSDAR.
	Restore machine check new PSW address.
	Replace at the beginning of this routine the branch instruction by the previous store register 15 instruction.
	Exit either to point of interruption or to DOS/VS normal program check routines.
F	If soft machine check, give control to DOS/VS normal machine check handler routines.
G	If program check or machine check occurred during execution of specified move or compare instructions in the SDEHR editing and dumping subroutines, give control to the interrupted subroutine.
H	Enter hard wait state.
Note: The labels A through H used in the description of the DISPATCHER are used to indicate return points in the description of the following event handling routines.	

Figure 40. Logic Flow of the Event Handling Routines (Part 1 of 6)

```
PEREVENT

     Entry from DISPATCHER if PER bit on in interruption code.

     Link to TESTLIM subroutine.

     If event is within limits then:
          If output class is FASTREC, link to FASTREC1 subroutine and go to A.
          If output class different from FASTREC, format and print all outputs
          implemented for the specified output class.

 A   Remove PER bit in interruption code.

     Exit to DISPATCHER point A.
```

Figure 40. Logic Flow of the Event Handling Routines (Part 2 of 6)

```
MCEVENT

     Entry from DISPATCHER if MC bit on in interruption code.

     If MC class 0, 1 or 2 (reserved to DOS/VS), exit to DISPATCHER point C.

     If MC class 5 then:

          If MC code 11 then:
               If page enqueue enabled (swith MCNENQ in MONPARM), format and print all
               outputs implemented for this event.
               Go to A.
          If MC code 12 then:
               If page handling event enabled (switch MCNHDL in MONPARM), format and
               print all outputs implemented for this event.
               Go to A.
          If MC code 20 then:
               Link to PRINTBUF subroutine.
               Go to A.

     If output class is FASTREC, link to FASTREC2 subroutine and go to A.

     If output class different from FASTREC, format and print all outputs implemented
     for the specified output class.

 A   Remove MC bit in interruption code.

     Exit to DISPATCHER point B.
```

Figure 40. Logic Flow of the Event Handling Routines (Part 3 of 6)

	DATEVENT
	Entry from the DISPATCHER if interruption code is X'11'.

	If TE event enabled (switch MONTE in MONPARM), then:
	    If event within specified limits then:
	        If output class is FASTREC, link to FASTREC3 subroutine and go to A.
	        If output class different from FASTREC, format and print all outputs implemented for the specified output class.

A	Exit to DISPATCHER point C.

Figure 40. Logic Flow of the Event Handling Routines (Part 4 of 6)

	PGMEVENT
	Entry from DISPATCHER if program check event enabled.
	Save current output class and force output class SUPVISOR.
	Format and print all outputs implemented for SUPVISOR output class.
	Restore current output class.
	Exit to DISPATCHER point D.

Figure 40. Logic Flow of the Event Handling Routines (Part 5 of 6)

	STOPONEV
	Entry from DISPATCHER if stop on event enabled.
	Link to SAVEXT subroutine.
	Initialize new external PSW address to get control at point A.
	Enter wait state.
A	External interrupt occurred.
	Link to RESTEXT subroutine.

	If non-destroying-dump required (switch NONDD in MONPARM), then:
	    Save current output class and force output class DUMPREAL.
	    Format and print all outputs implemented for output class DUMPREAL.
	    Restore current output class.
	    Disable non-destroying-dump.

	Exit to DISPATCHER point E.

Figure 40. Logic Flow of the Event Handling Routines (Part 6 of 6)

RECORDING ROUTINES

Figure 41 is a tabular summary of the logic flow of the Recording Routines.

FASTRECx Subroutine
If entry is from PEREVENT routine, move PER code to the SDBUF entry. If entry is from MCEVENT routine, move MC mnemo and MC class to the SDBUF entry. If entry is from DATEVENT routine, move translation exception mnemo to the SDBUF entry. Move all implementation specified information to the SDBUF entry. Update MONWPNT in MONPARM to point to the next SDBUF entry. If last entry was just used then: Update MONWPNT to point to the first SDBUF entry. If recording mode is AUTOMATIC, link to PRINTBUF subroutine. Return to caller.

Figure 41. Logic Flow of Recording Routines (Part 1 of 5)

PRINTHD Subroutine
Link to MONOPEN to get control of the printer. Link to MONPUT to space one line. Link to BLANKRTN to get a work area. Link to TRANSTD to edit TOD clock in this work area. Link to EDIT8BYT to edit the old PSW in this work area. Return to caller.

Figure 41. Logic Flow of Recording Routines (Part 2 of 5)

PRINTHDMC Subroutine
Link to MONOPEN to get control of the printer. Link to MONPUT to space one line. Link to BLANKRTN to get a work area. Move common text for page enqueue and page handling event to this work area. Take page queue entry address. Return to caller.

Figure 41. Logic Flow of Recording Routines (Part 3 of 5)

PRINTIT Subroutine	
	Link to MONPUT to print the heading line.
	If output class is FASTREC or PSW, go to B.
	Edit and print general registers 0 through 15.
	If output class is GPR, go to B.
	Edit and print control registers 0 through 15. Dump low storage from address 0 to 11F.
	If output class is LOCORE, go to A.
	If output class is COMREG then: Link to SPECDUMP to dump COMRG and SYSCOM. Go to A.
	If output class is PAGETAB then: Link to SPECDUMP to dump segment table, page table and page frame table. Go to A.
	If output class is SUPVISOR then: Link to DUMPSUB to dump the entire supervisor. Go to A.
	If output class is DUMPREAL then: Link to DUMPSUB to dump the entire real storage.
A	Link to MONPUT to space one line.
B	Link to MONCLOSE to return the printer to DOS/VS.
	Return to the caller.

Figure 41. Logic Flow of Recording Routines (Part 4 of 5)

PRINTBUF Subroutine	
	Link to PRINTHDMC to get control of the printer.
	Print buffer output header lines.
	If SDBUF has wrapped, initialize to start printing with the entry pointed to by MONWPNT in MONPARM.
	If the SDBUF has not wrapped, initialize to start printing with the first entry of SDBUF.
	If no event is recorded in this entry, go to B.
A	Format and print the information contained in this SDBUF entry.
	Point to next entry in SDBUF.
	If end of SDBUF is reached then: Initialize to point to the first entry of SDBUF. If this entry is already printed, go to C.
	Go to A.
B	Indicate no events are present in SDBUF.
C	Link to MONCLOSE to return the printer to DOS/VS.
	Return to caller.

Figure 41. Logic Flow of Recording Routines (Part 5 of 5)

DUMPING AND EDITING SUBROUTINES

The SDEHR routines run with the DAT feature off, and the storage area limits to be dumped or edited by the following subroutines are not checked for validity.

The program check due to addressing exception which may occur while executing predefined move or compare instructions are checked by the DISPATCHER, and control is returned to the appropriate subroutine entry point.

Furthermore, the machine check which may occur while executing these same predefined move or compare instructions are checked by the DISPATCHER, and control is returned to the appropriate subroutine entry point if a soft machine check occured.

In these cases, the subroutine returns a character 'X' for each storage location outside real storage.

Figure 42 is a tabular summary of Dumping and Editing subroutines.

DUMPSUB Subroutine	
	Round the start address to the lower X'20' boundary.
A	Link to BLANKRTN subroutine to get a work area.
	Link to EDIT4BYT subroutine to edit the start address of that line in the work area.
	Link to EDITREG8 subroutine to edit the following 32 bytes in the work area. Notes: - The pointer to process the next 32 bytes is updated by this subroutine. - If a program check occurs in this subroutine every byte outside real storage will be replaced by the character 'X'.
	If a program check has occurred while processing EDITREG8, go to B.
	If translation is desired, move the preceding 32 bytes into the work area and replace unprintable characters with 'X'.
B	Link to MONPUT to print the line.
	If a program check has occurred in this subroutine, go to E.
C	Compare the 32 bytes just printed with next pointed 32 bytes. Note : - If a program check with addressing exception occurs while executing the previous compare operation, control is given to G by the DISPATCHER.
D	If comparison is equal then: If '...SAME...' line not already printed then: Link to BLANKRTN to get a work area. Link to MONPUT to print '...SAME...' line. If number of requested lines already printed, go to F. Update pointer to process next 32 bytes. Go to C.
E	If a program check has occurred while processing EDITREG8, go to F.
	If number of requested lines not already printed, go to A.
F	Return to caller.
G	Restore old PSW and interruption code.
	Go to A.

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 1 of 6)

SPECDUMP Subroutine	
	Link to EDIT4BYT Subroutine to edit the start address of the area to be dumped.
	Link to MONPUT to print it.
	Enable translation in DUMPSUB.
	Link to DUMPSUB to dump the number of requested lines.
	Return to caller.

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 2 of 6)

TRANSTD Subroutine	
	Get the binary clock value in microseconds.
	Divide to convert to seconds.
	If quotient is zero, clear workfield 1 and go to A.
	Convert quotient to decimal in workfield 1.
	Shift workfield 1 right to obtain microseconds.
A	Convert remainder to decimal in workfield 2.
	Add workfield 2 to workfield 1.
	Link to EDIT8BYT subroutine to edit the result.
	Return to caller.

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 3 of 6)

	INSTxxx Subroutine
	If entry is at label INSTEDPS, substract old PSW ILC from the instruction address.
	Save interruption code.
	Point to the first byte of the instruction.
	If DAT feature is off in the old program PSW, transform LRA instruction to LA instruction.
A	Load Real Address or Load Address of the byte pointed to.
	If the address is invalid, move two 'Y' characters to the edit field and go to B.
	If a segment translation exception occurred, move two 'X' characters to the edit field and go to B.
	Link to EDIT1BYT to edit this byte.
	Point to the next byte.
	If not the last byte, go to A.
B	Restore LA instruction to LRA instruction.
	Restore interruption code.
	Return to caller.

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 4 of 6)

	EDITREGx Subroutine
	If entry is at label EDITREG8, initialize to edit 8 fullwords.
	If entry is at labels EDITREG5 or EDITREG, initialization is done by the caller.
A	Link to EDIT4BYT subroutine to edit one fullword.
	Update pointers.
	If more fullwords are to be edited, go to A.
	Return to caller.

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 5 of 6)

EDITXBYT Subroutine	
	If entry is at label EDIT8BYT, go to B.
	If entry is at label EDIT4BYT, initialize to return 8 bytes to the caller.
	If entry is at label EDIT3BYT, initialize to return 6 bytes to the caller.
	If entry is at label EDIT2BYT, initialize to return 4 bytes to the caller.
	Save interruption code.
A	Move 4 bytes of user data to an inner field.
	Note : - If a program check with addressing exception occurs while executing the previous operation, control is given to E by the DISPATCHER.
	Unpack and translate the 4 bytes.
	Go to D.
B	Initialize to return 16 bytes to the caller.
C	Move 8 bytes of user data to an inner field.
	Note: - If a program check with addressing exception occurs while executing the previous operation, control is given to F by the DISPATCHER.
	Unpack and translate the 8 bytes.
D	Move 17 'X' characters behind the previously translated bytes.
	Move to the user the number of requested bytes.
	Restore subroutine to return 2 bytes.
	Return to caller.
E	Restore old PSW and interruption code.
	Reduce length of the move instruction by one.
	If length equal zero, go to D.
	Go to reexecute the interrupted instruction. (A or C).

Figure 42. Logic Flow of Editing and Dumping Subroutine (Part 6 of 6)

PRINTER IOCS

When the output class is other than FASTREC, tracing and dumping takes place on the specified printer at the time the event occurs. SDEHR does not use the DOS/VS channel scheduler to perform its I/O operations; it has its own printer IOCS. At the time SDEHR has to perform output, the specified printer may be executing an operation previously initiated either by the system or by the user.

Before taking control of the printer the MONOPEN routine tests the status of the printer:

- If the specified channel, subchannel, and device are available, the SDEHR printer IOCS gets control of the printer immediately.
- If the channel is still working in burst mode, the MONOPEN routine enters a hard wait state.

- If the subchannel is still working with the device, MONOPEN enters a TIO loop until the channel-end condition is stored.
- If the device is still working, the TIO loop is entered until the DEVICE-END condition is stored
- In both preceding cases, the interruption conditions are saved to be simulated at the time all the desired output is performed.
- If the status received at channel-end or device-end time indicates abnormal conditions, the CSW and, if a unit check occurred, the sense byte are printed to make the user aware that the output may be incorrect.

Before returning control of the printer, the MONCLOSE routine tests whether interruption conditions have to be simulated:

- If a channel-end condition is to be simulated, the last user-executed CCW is replaced by a write-no-space CCW. The operation is initiated and the user CCW is restored before control of the printer is returned.

- If a device-end condition is to be simulated, a write-no-space operation is initiated. A TIO loop is entered until the channel-end interruption occurs and control of the printer is returned.

	MONOPEN Subroutine
	Save CAW, CSW and I/O interruption code extension.
A	Issue Test Channel. If working in burst mode, go to A. If interruption pending, go to D.
B	Issue Test I/O. If not operational, link to NOTOPO and go to A. If busy, go to A. If available, go to E.
C	Link to CSWOPEN to process CSW conditions and go to A.
D	Issue Test I/O. If CSW stored, go to C. If not operational, link to NOTOPO and go to A. Move low storage hard wait message. Enter hard wait state.
E	Indicate IOCS is OK. Return to caller.

Figure 43. Logic Flow of Printer IOCS Subroutines (Part 1 of 6)

	MONPUT Subroutine
	Initialize CAW and CCW data address.
A	Issue Test I/O.
	If CSW stored, link to CSWSTORD subroutine and go to A. If busy, go to A. If not operational, link to NOTCPC and go to A.
	Issue Start I/O.
	If not operational, link to NOTCPC and go to A. If busy, go to A. If device started, go to B. Link to CSWSTORD. If abnormal status bits, go to A.
B	Return to caller.

Figure 43. Logic Flow of Printer IOCS Subroutines (Part 2 of 6)

	NOTOPO Subroutine
	Link to SAVEXT subroutine.
	Move low storage message.
	Initialize new external PSW address to get control at A.
	Enter wait state.
A	External interruption has occurred.
	Link to RESTEXT subroutine.
	Return to caller.

Figure 43. Logic Flow of Printer IOCS Subrcutines (Part 3 of 6)

ERRSIO Subroutine	
	Initialize CAW.
A	Issue Start I/O. If busy, go to A. If CSW stored, gc tc C. If not operational, link to NOTOPO and go to A.
B	Issue Test I/O. If busy, go to B. If not operational, link to NOTCPC and gc tc A. If device status indicates conditions other than busy, channel end, device end or unit exception, go tc A. Set first time switch on. If channel status indicates conditicns cther than incorrect length, gc to A.
	Go to D.
C	If device status indicates unit check, gc tc A.
	If first time switch is on, link to NOTOFO and gc to A.
D	Set first time switch off.
	Return to caller.

Figure 43. Logic Flow of Printer IOCS Subroutines (Part 4 of 6)

CSWxxxx Subroutine	
	If entry at label CSWSTORD, go to B.
	Save CSW.
A	If the device status indicates Unit Exception, go to C.
B	Test the device status. If abnormal (see note 1), go to C. If Unit Exception, go to J. If Channel End or Device End, go to G.
	Go to J.
C	Save the CAW.
	Link to EDIT8BYT subroutine to edit the CSW.
	If unit check bit not on in device status, go to E.
	Link to ERRSIO subroutine to get sense byte.
	Test sense byte. If intervention required bit on, link to NOTOPO and go to H. If Channel 9 bit is on, go to D.
	If entry at CSWSTORD label, go to H.
D	Link to EDIT4BYT subroutine to edit sense information.
E	If entry at CSWOPEN label, go to F.
	If channel status abnormal (see note 2), go to DISPATCHER at point H.
F	Link to ERRSIO to print CSW and sense information.
	Restore CAW.
	If device or channel status are abnormal (see note 1 and 2), go to J.
G	Return to caller at return point 4. This indicates normal status conditions.
H	Restore CAW.
J	Return to caller. This indicates abnormal status conditions.

Note 1: Device status is abnormal if it indicates conditions other than Busy, Channel End, Device End or Unit Exception.

Note 2: Channel status is abnormal if it indicates one of the following conditions: PCI, IL, program check or Protection Check.

Figure 43. Logic Flow of IOCS Subroutines (Part 5 of 6)

	MONCLOSE Subroutine

A	Issue Test I/O.
	
	    If not operational, link to NOTOPO and go to A.
	    If busy, go to A.
	    If CSW stored, link to CSWSTORD and go to A.
	
	If no user interruption was killed, go to E.
	
	Get last command address from user saved CSW and subtract 8.
	
	If last command address was zero, initialize the CAW with the inner simulated CCW address and go to B.
	
	Initialize CAW with the last user CCW address.
	
	Save user CCW and replace it with inner simulated CCW.

B	Issue Start I/O.
	    If not operational, link to NOTOPO and go to B.
	    If busy or CSW stored, go to B.
	
	If only a device end condition was removed, go to D.

C	Issue Test I/O.
	    If not operational, link to NOTOPO and go to B.
	    If busy go to C.
	
	If Channel End bit not on in device status, go to B.
	
	If device status indicates Device End or Unit Check, go to B.
	
	If any channel status present, go to B.

D	Restore user CCW if saved.
E	Restore CAW, CSW and I/O interruption code extension. Return to caller.

Figure 43. Logic Flow of IOCS Subroutines (Part 6 of 6)

MISCELLANEOUS

Figure 44 is a tabular summary of the logic flow of the Miscellaneous Routines.

	TESTLIM Subroutine
	Get lower and upper limits from control registers 10 and 11.
	Reduce address of event to absolute zero.
	Reduce address of upper limit to absolute zero.
	If event is between limits, add 4 to caller return point.
	Return to caller.
	SAVEXR Subroutine
	Save the following information: Old and new external PSWs. External interruption code extention. Low storage bytes 0 through 3. Current value of control register 0.
	Initialize control register 0 to enable only key external interruptions.
	Initialize new external PSW to give control to SDEHR. (PER bit must be off).
	Return to caller.
	RESTEXT Subroutine
	Restore the following information saved by SAVEXT subroutine: Old and new external PSWs. External interruption code extention. Low storage bytes 0 through 3. Current value of control register 0.
	Return to caller.
	BLANKRTN Subroutine
	Get address of work area one.
	If work area one was just used, return address of work area two to the user and go to A.
	Return address of work area one to the user.
A	Clear the returned work area.
	Return to caller.

Figure 44. Logic Flow of Miscellaneous Routines

TERMINATION ROUTINES

The SDAID tracing and dumping functions are terminated either by issuing the ENDSD attention command or by executing the phase ENDSD in a virtual or real partition.

The Termination phase and the termination transient perform the following functions:

ENDSD	Call $$BATTN3 termination transient and release the partition upon return.
$$BATTN3	Print the SDBUF buffer. Switch off in control registers 8, 9, 10, and 11 all fields set up for the tracing/dumping options. Switch the PER bit off in all PSWs and release the page frames allocated to SDAREA.

The general logic flow of the SDAID termination routines is shown in Figure 45.

During termination, the SDAID program uses the following storage areas:

- SAVSDAR in the supervisor
- The logical transient area (LTA) for $$BATTN3
- A virtual or real partition (only if termination is via // EXEC ENDSD)
- SDAREA at the high end of real storage

Figures 46 and 47 show the logic flow of phase ENDSD and of the transient $$BATTN3, respectively, in tabular form.

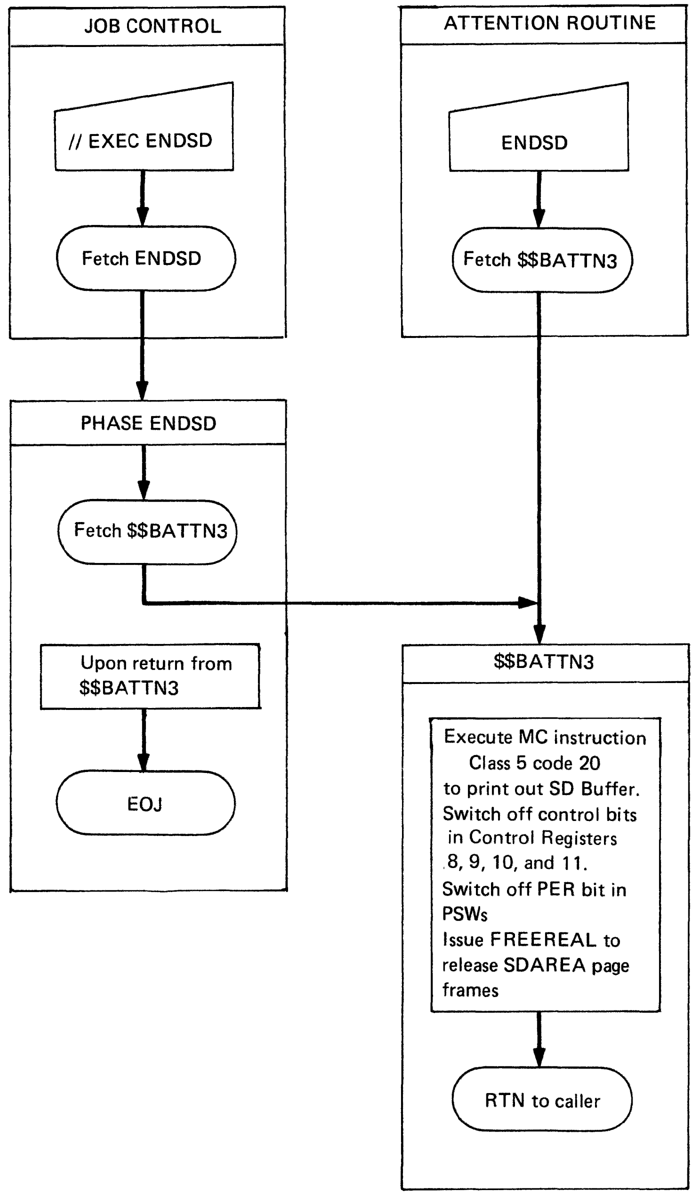


Figure 45. General Logic Flow of Termination Routines

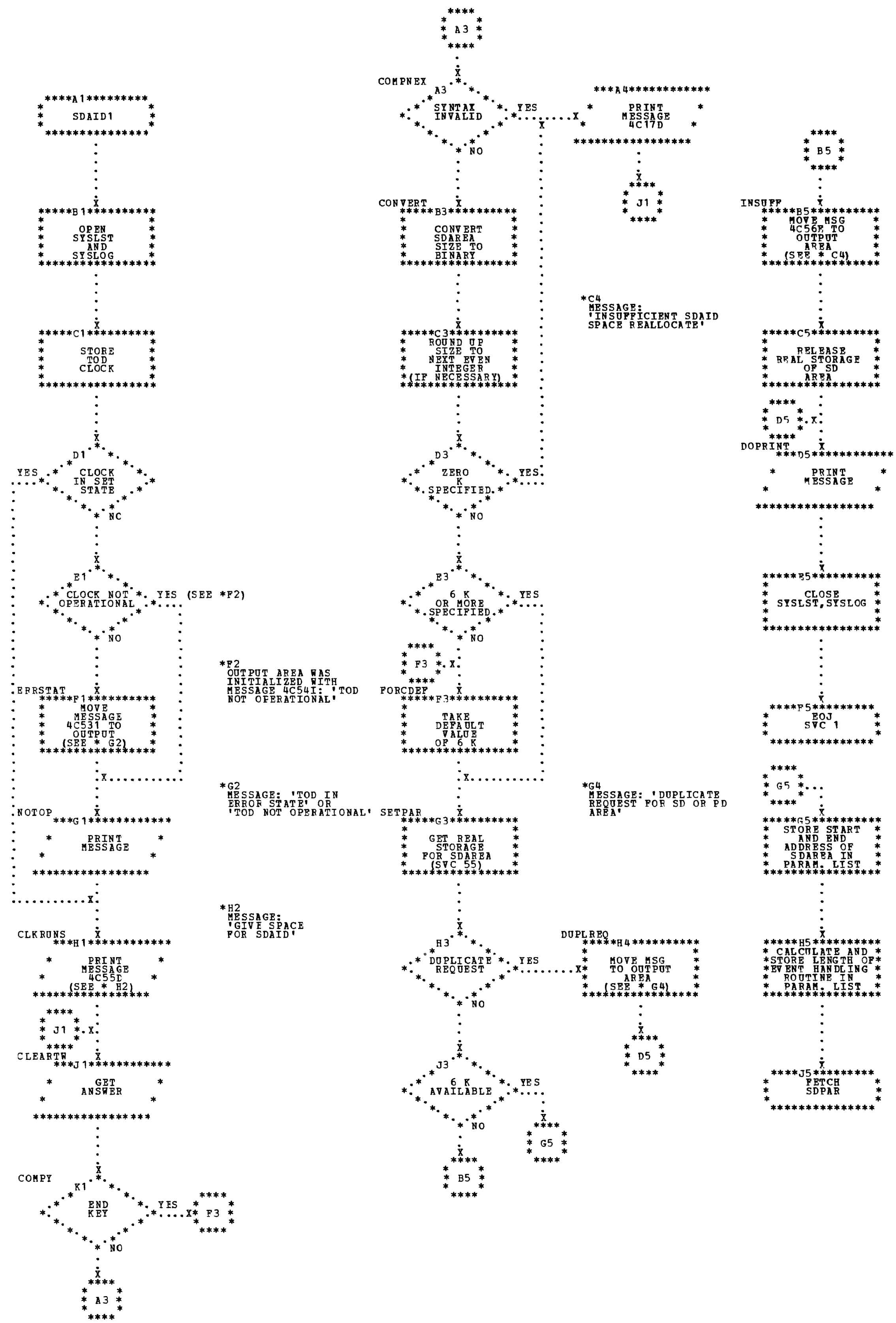
	ENDSD
	Entry from Job Control.
	Indicate to transient $$BATTN3 that the caller is phase ENDSD.
	Fetch $$BATTN3.
	Exit to Job Control.

Figure 46. Logic Flow of Phase ENDSD

	$$BATTN3
	Entry from the attention routine or from phase ENDSD.
	Issue Monitor Call (class 5, code 20) to print the core wrap buffer.
	If SDAID is not active, go to A.
	Seize the system and force a program check to get control in supervisor state.
	Switch off all SDEHR control bits in control registers 8, 9, 10 and 11.
	Restore normal DOS/VS new program check PSW and switch off the PER bit.
	Build Control List table. (see section 3)
	Build Per List table. (see section 3)
	Switch off the PER bit in all PSWs using the PER List table.
	Release the system.
	Issue FREEREAL to return to the page pool all page frames allocated to SDAREA.
A	Exit to the attention routine or the phase ENDSD.

Figure 47. Logic Flow of Transient $$BATTN3

Chart GA. SDAID1 - Get Page Frames and Allocate Them to SDAREA
Refer to Figure 29.



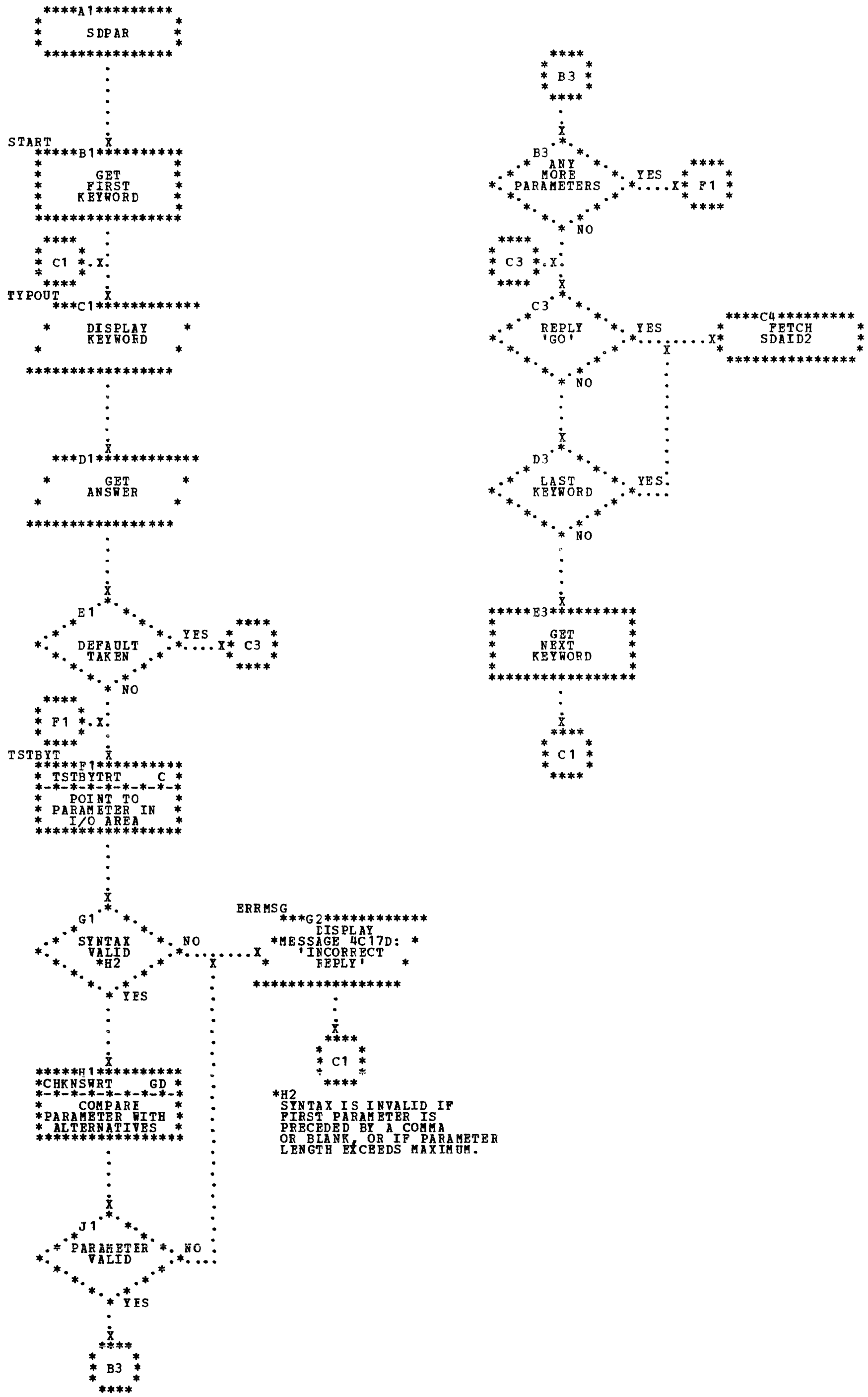

A1
SDPAR
START
B1
GET FIRST KEYWORD
C1
TYPOUT
C1
DISPLAY KEYWORD
D1
GET ANSWER
E1
DEFAULT TAKEN
YES
C3
NO
F1
TSTBYT
F1
TSTBYTRT C
POINT TO PARAMETER IN I/O AREA
G1
SYNTAX VALID *H2
NO
YES
ERRMSG
G2
DISPLAY *MESSAGE 4C17D: 'INCORRECT REPLY'
C1
*H2
SYNTAX IS INVALID IF FIRST PARAMETER IS PRECEDED BY A COMMA OR BLANK, OR IF PARAMETER LENGTH EXCEEDS MAXIMUM.
H1
CHKNSWRT GD
COMPARE PARAMETER WITH ALTERNATIVES
J1
PARAMETER VALID
NO
YES
B3
B3
B3
ANY MORE PARAMETERS
YES
F1
NO
C3
C3
REPLY 'GO'
YES
C4
FETCH SDAID2
NO
D3
LAST KEYWORD
YES
NO
E3
GET NEXT KEYWORD
C1

Chart GC. SDPAR - Initialize an Operator-to-System Communication (Part 2 of 3)
Refer to Figure 29.

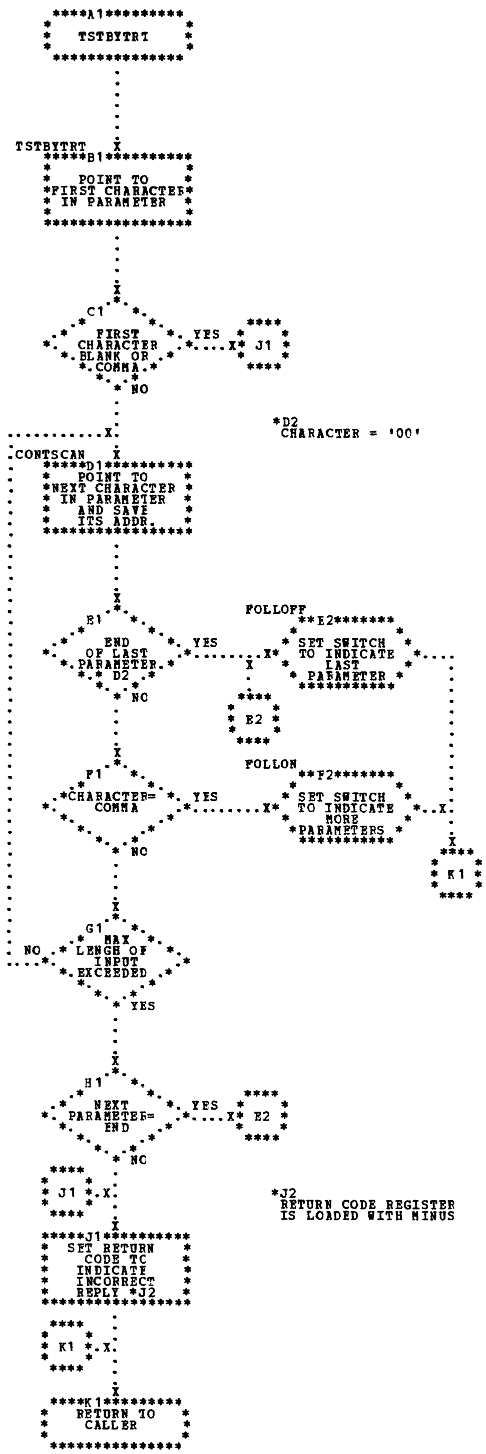


Chart GD. SDPAR - Initalize an Operator-to-System Communication (Part 3 of 3)
Refer to Figure 29.

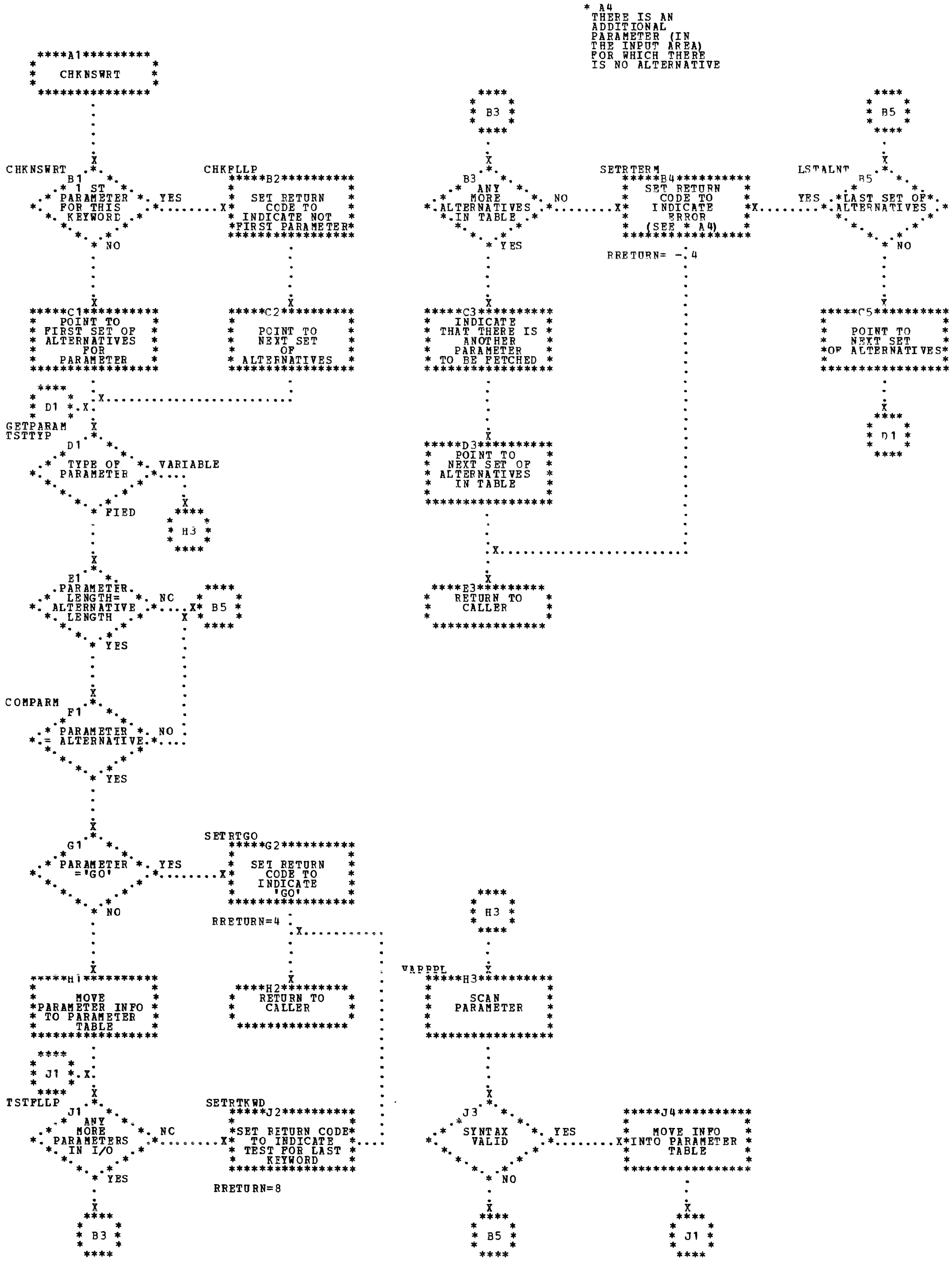
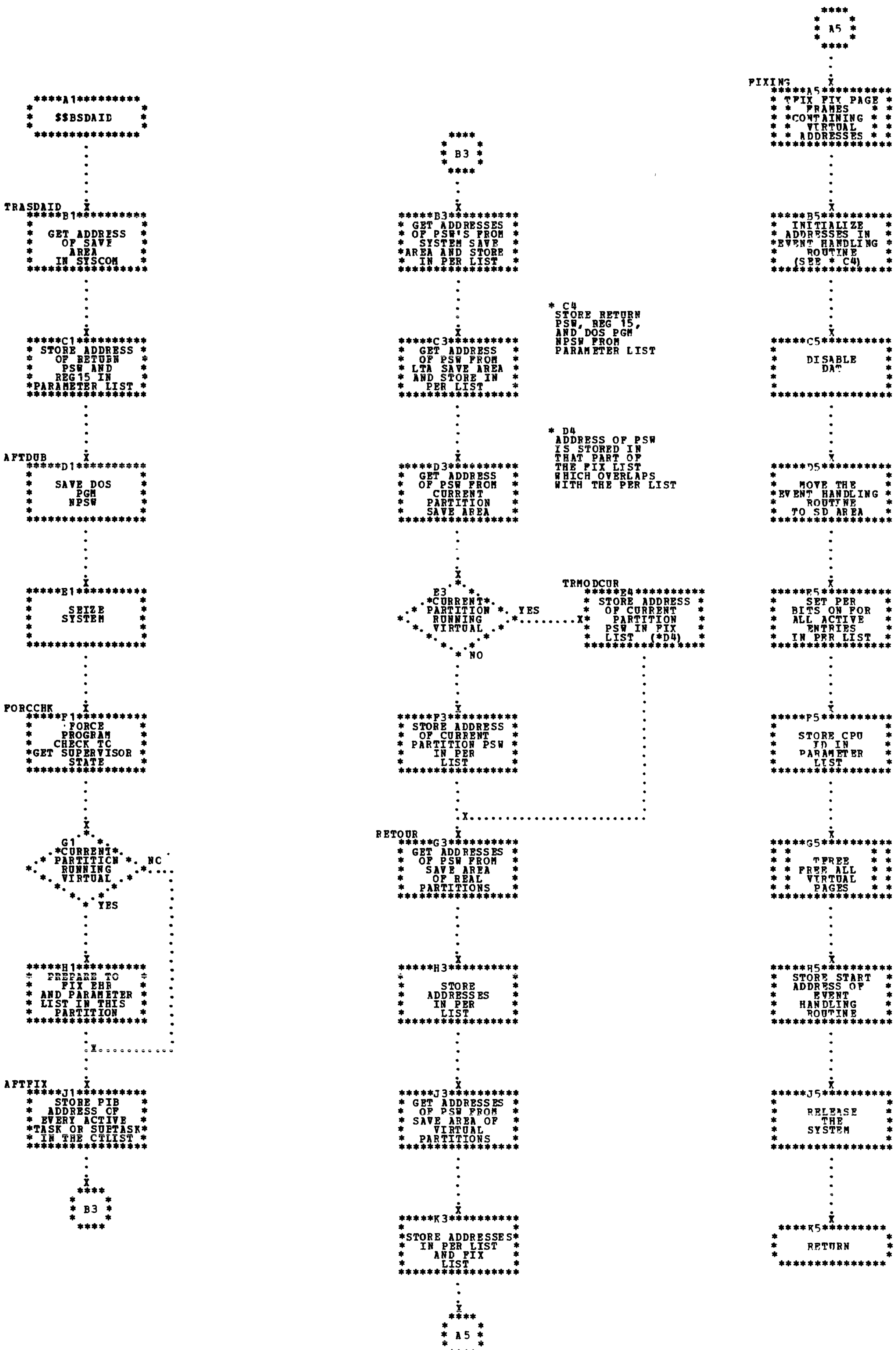
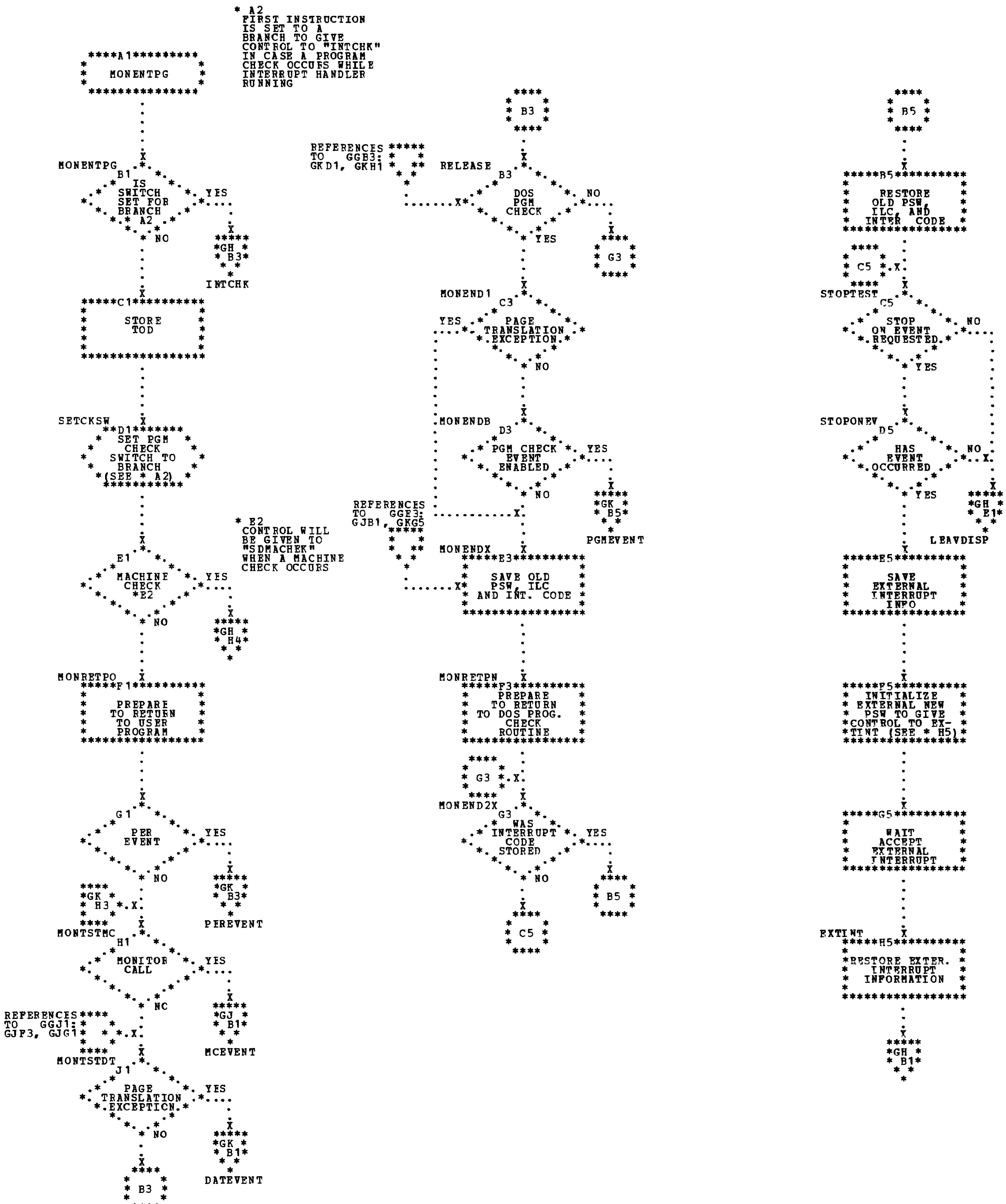


Chart GE. SDAID2 - Print Selected or Default Tracing/Dumping Options
Refer to Figure 29.

```
  ****A1*********
 *               *
 *    SDAID2     *
 *               *
  ***************
        .
        .
        .
        .
        .
        X
   ***B1************
  * PRINT DEFAULT *
     OPTIONS AND
 *     OPTIONS      *
      SPECIFIED
*****************
        .
        .
        .
        .
        X
 *****C1**********
 *  STORE START   *
 *AND END ADDRESS*
 *OF SD CORE-WRAP*
 *    BUFFER IN   *
 *PARAMETER LIST *
 *****************
        .
        .
        .
        .
        .
        X
   ***D1************
        PRINT
  *   PROCEDURES    *
         FOR
 *     CHANGING     *
      PARAMETERS
*****************
        .
        .                   * E2
        .                    FIRST PART OF
        .                    PARAMETER LIST
        X                    INCLUDES PARAMETERS
 *****E1**********           UP TO MONEND THIS
 *MOVE FIRST PART*           PART OF THE LIST IS
 * OF PARAMETER  *           REQUIRED BY THE EVENT
 *LIST TO END OF *           HANDLING RCUTINES FOR
 *EVENT HANDLING *           COMMUNICATION BETWEEN
 * ROUTINE *(E2) *           THE SDAID MODULES
 *****************
        .
        .
        .
        .
        .
        X
 *****F1**********
 *$$BSDAID     GF*
 *-*-*-*-*-*-*-*-*
 *  MOVE AREAS   *
 *   TO SDAREA   *
 *               *
 *****************
        .
        .
        .
        .
        .
        X
   ***G1************
  *     PRINT       *
         CPU
 *       ID         *

*****************
        .
        .
        .                   * H2
        .                    MESSAGE:
        .                    "SDAID
        X                    SUCCESSFULLY
   ***H1************         INITIATED"
        PRINT
  *    MESSAGE      *
        4C57E
 *                  *
      (SEE *H2)
*****************
        .
        .
        .
        .
        .
        X
 *****J1**********
 * *           * *
 * *   CLOSE   * *
 * *  SYSLOG   * *
 * *    AND    * *
 * *  SYSLST   * *
 *****************
        .
        .
        .
        .
        .
        .
        X
  ****K1*********
 *               *
 *      EOJ      *
 *               *
  ***************
```

Chart GF. $$BSDAID - Initialize Event Handling Routines, PSWs, and Control Registers
Refer to Figure 29.




A1
MONENTPG
* A2
FIRST INSTRUCTION IS SET TO A BRANCH TO GIVE CONTROL TO "INTCHK" IN CASE A PROGRAM CHECK OCCURS WHILE INTERRUPT HANDLER RUNNING
MONENTPG B1
IS SWITCH SET FOR BRANCH *A2
YES
NO
GH B3
INTCHK
C1
STORE TOD
SETCKSW D1
SET PGM CHECK SWITCH TO BRANCH (SEE * A2)
* E2
CONTROL WILL BE GIVEN TO "SDMACHEK" WHEN A MACHINE CHECK OCCURS
E1
MACHINE CHECK *E2
GH H4
MONRETPO F1
PREPARE TO RETURN TO USER PROGRAM
G1
PER EVENT
GK B3
PEREVENT
GK H3
MONTSTMC H1
MONITOR CALL
GJ B1
MCEVENT
REFERENCES TO GGJ1: GJF3, GJG1
MONTSTDT J1
PAGE TRANSLATION EXCEPTION
GK B1
DATEVENT
B3
B3
REFERENCES TO GGB3: GKD1, GKH1
RELEASE B3
DOS PGM CHECK
G3
MONEND1 C3
PAGE TRANSLATION EXCEPTION
MONENDB D3
PGM CHECK EVENT ENABLED
GK B5
PGMEVENT
REFERENCES TO GGE3: GJB1, GKG5
MONENDX E3
SAVE OLD PSW, ILC AND INT. CODE
MONRETPN F3
PREPARE TO RETURN TO DOS PROG. CHECK ROUTINE
G3
MONEND2X G3
WAS INTERRUPT CODE STORED
B5
C5
B5
B5
RESTORE OLD PSW, ILC, AND INTER CODE
C5
STOPTEST C5
STOP ON EVENT REQUESTED
STOPONEV D5
HAS EVENT OCCURRED
GH E1
LEAVDISP
E5
SAVE EXTERNAL INTERRUPT INFO
F5
INITIALIZE EXTERNAL NEW PSW TO GIVE CONTROL TO EX-TINT (SEE * H5)
G5
WAIT ACCEPT EXTERNAL INTERRUPT
EXTINT H5
RESTORE EXTER. INTERRUPT INFORMATION
GH B1

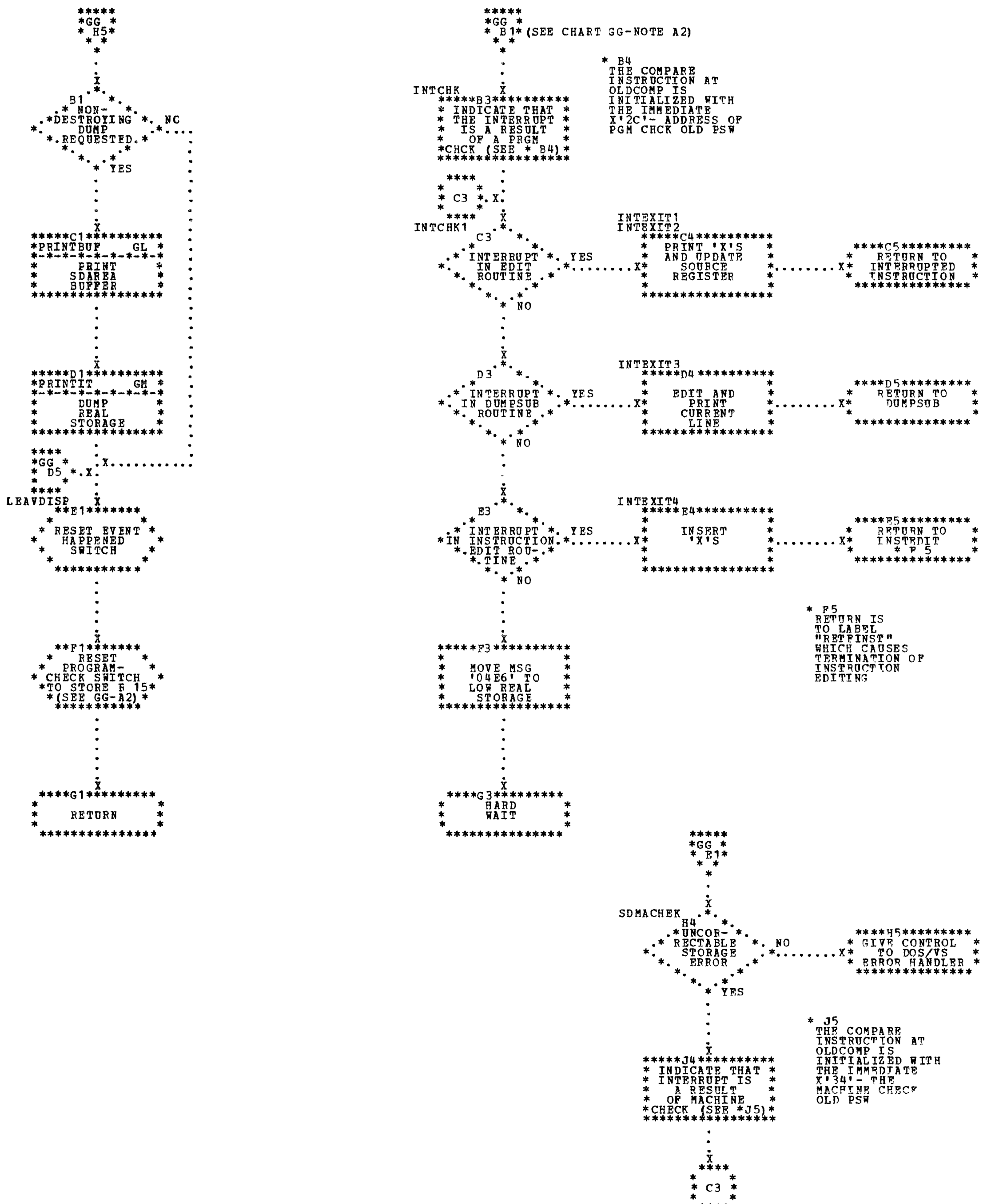
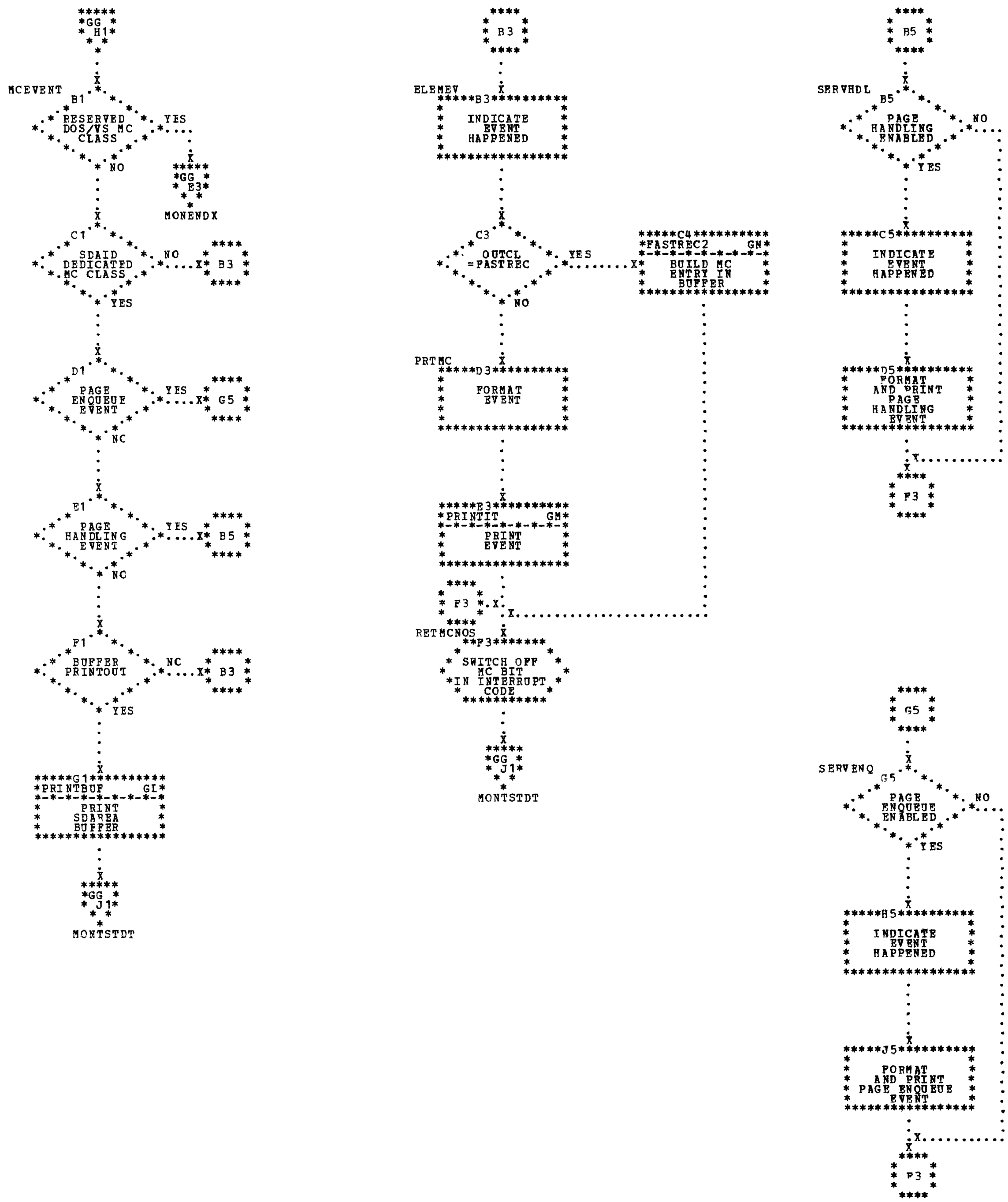

GG H5
B1 NON-DESTROYING DUMP REQUESTED
NO
YES
C1 PRINTBUF GL
PRINT SDAREA BUFFER
D1 PRINTIT GM
DUMP REAL STORAGE
GG D5
LEAVDISP
E1 RESET EVENT HAPPENED SWITCH
F1 RESET PROGRAM-CHECK SWITCH TO STORE F 15 (SEE GG-A2)
G1 RETURN
GG B1 (SEE CHART GG-NOTE A2)
INTCHK
B3 INDICATE THAT THE INTERRUPT IS A RESULT OF A PRGM CHCK (SEE * B4)
* B4 THE COMPARE INSTRUCTION AT OLDCOMP IS INITIALIZED WITH THE IMMEDIATE X'2C'- ADDRESS OF PGM CHCK OLD PSW
C3
INTCHK1
C3 INTERRUPT IN EDIT ROUTINE
YES
NO
INTEXIT1
INTEXIT2
C4 PRINT 'X'S AND UPDATE SOURCE REGISTER
C5 RETURN TO INTERRUPTED INSTRUCTION
D3 INTERRUPT IN DUMPSUB ROUTINE
YES
NO
INTEXIT3
D4 EDIT AND PRINT CURRENT LINE
D5 RETURN TO DUMPSUB
E3 INTERRUPT IN INSTRUCTION-EDIT ROUTINE
YES
NO
INTEXIT4
E4 INSERT 'X'S
E5 RETURN TO INSTEDIT * F 5
* F5 RETURN IS TO LABEL "RETFINST" WHICH CAUSES TERMINATION OF INSTRUCTION EDITING
F3 MOVE MSG '04E6' TO LOW REAL STORAGE
G3 HARD WAIT
GG E1
SDMACHEK
H4 UNCORRECTABLE STORAGE ERROR
NO
YES
H5 GIVE CONTROL TO DOS/VS ERROR HANDLER
J4 INDICATE THAT INTERRUPT IS A RESULT OF MACHINE CHECK (SEE *J5)
* J5 THE COMPARE INSTRUCTION AT OLDCOMP IS INITIALIZED WITH THE IMMEDIATE X'34'- THE MACHINE CHECK OLD PSW
C3

Chart GJ. MCEVENT



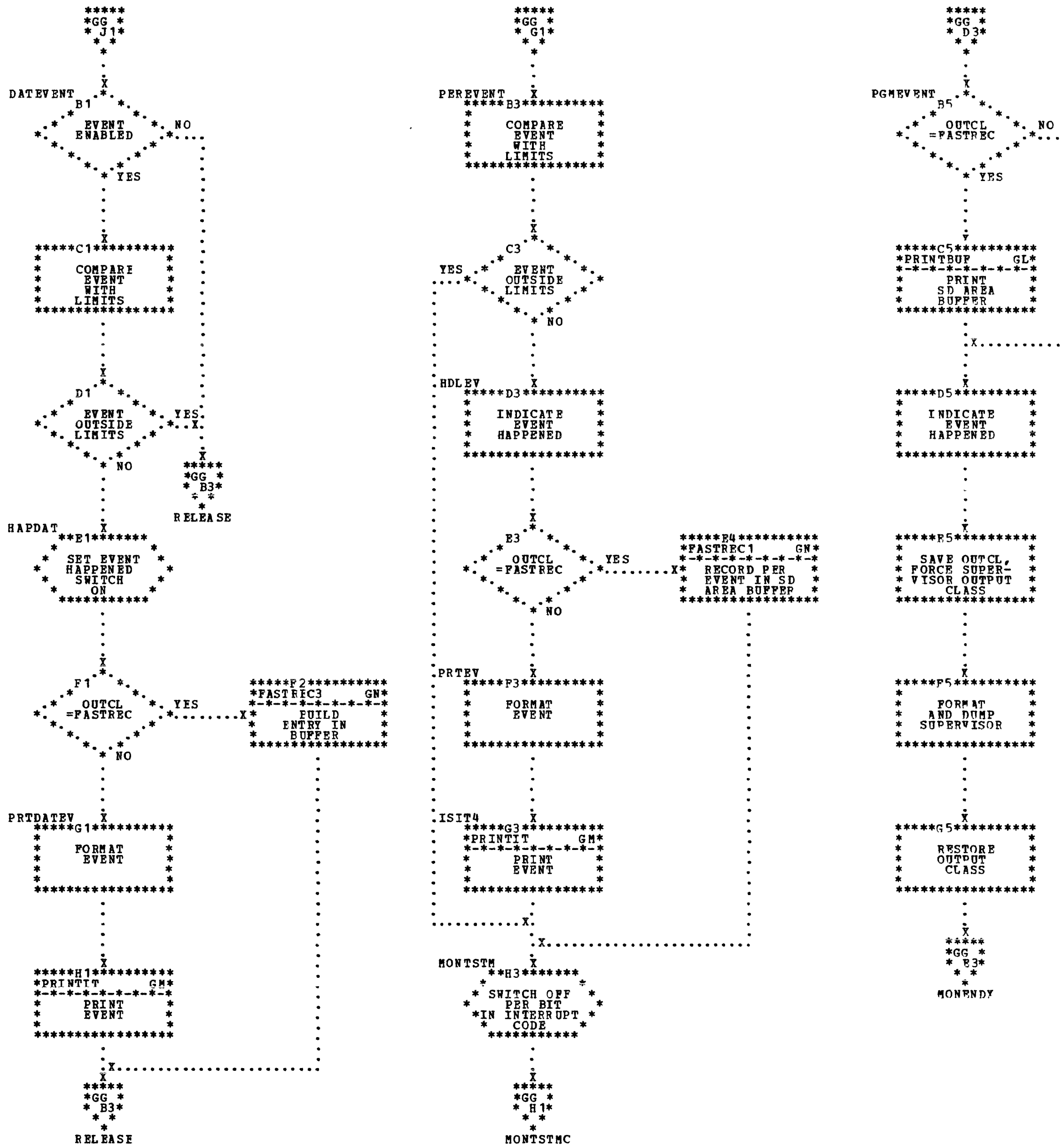
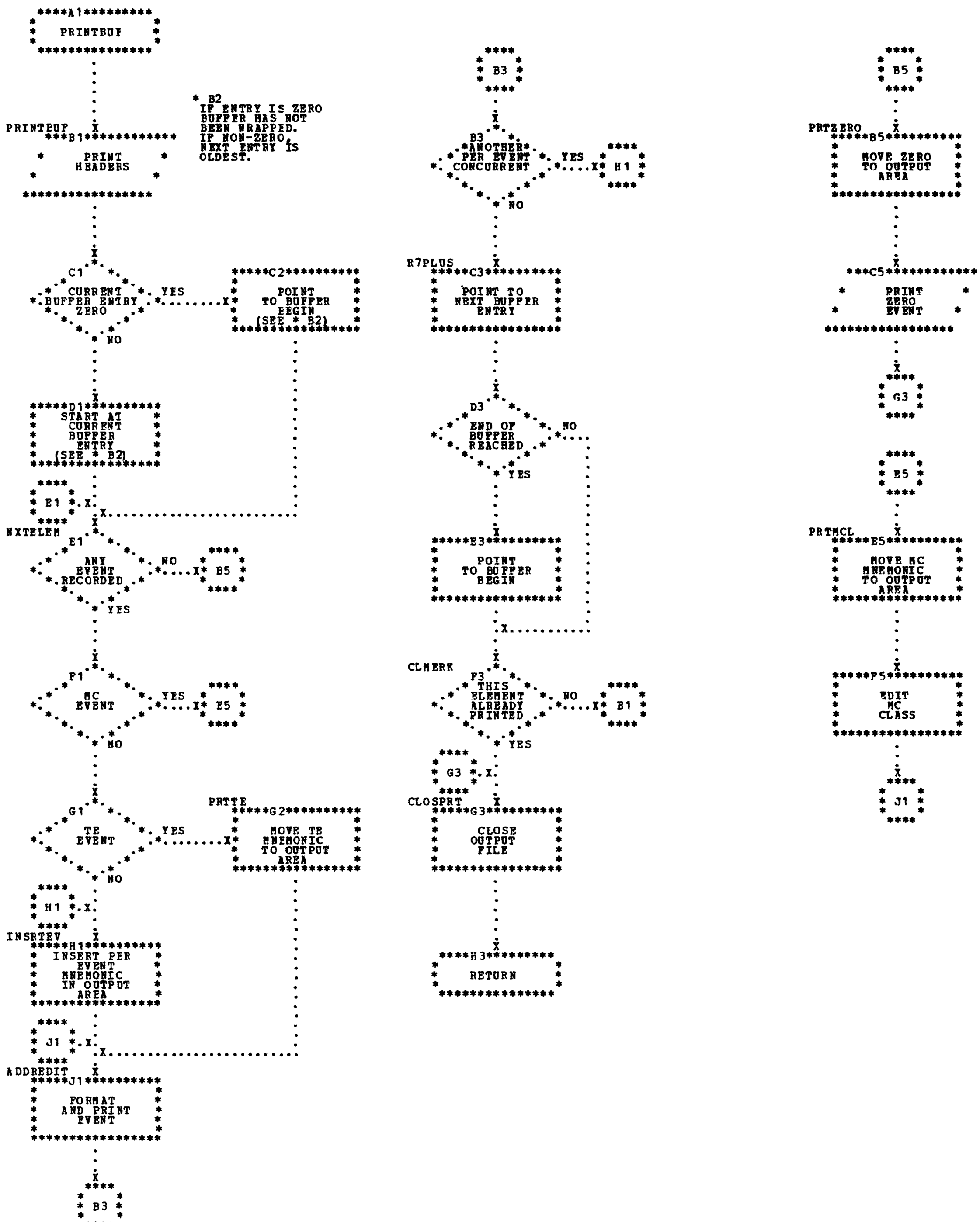
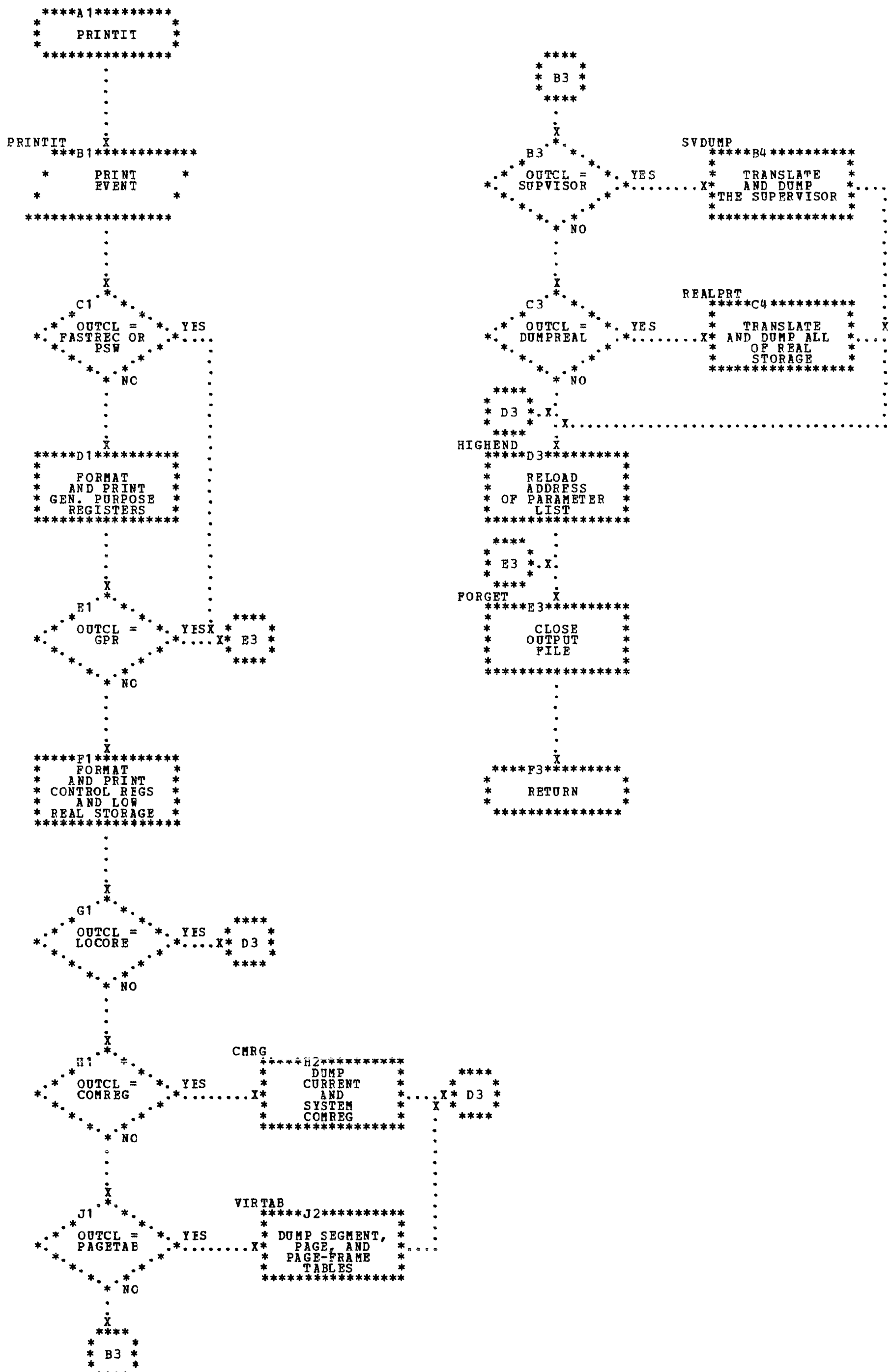

GG J1
DATEVENT B1
EVENT ENABLED
NO
YES
C1
COMPARE EVENT WITH LIMITS
D1
EVENT OUTSIDE LIMITS
YES
NO
GG B3
RELEASE
HAPDAT E1
SET EVENT HAPPENED SWITCH ON
F1
OUTCL =FASTREC
YES
NO
F2
FASTREC3 GN
BUILD ENTRY IN BUFFER
PRTDATEV G1
FORMAT EVENT
H1
PRINTIT GM
PRINT EVENT
GG B3
RELEASE
GG G1
PEREVENT B3
COMPARE EVENT WITH LIMITS
C3
EVENT OUTSIDE LIMITS
YES
NO
HDLEV D3
INDICATE EVENT HAPPENED
E3
OUTCL =FASTREC
YES
NO
E4
FASTREC1 GN
RECORD PER EVENT IN SD AREA BUFFER
PRTEV F3
FORMAT EVENT
ISIT4 G3
PRINTIT GM
PRINT EVENT
MONTSTM H3
SWITCH OFF PER BIT IN INTERRUPT CODE
GG H1
MONTSTMC
GG D3
PGMEVENT B5
OUTCL =FASTREC
NO
YES
C5
PRINTBUF GL
PRINT SD AREA BUFFER
D5
INDICATE EVENT HAPPENED
E5
SAVE OUTCL, FORCE SUPER- VISOR OUTPUT CLASS
F5
FORMAT AND DUMP SUPERVISOR
G5
RESTORE OUTPUT CLASS
GG E3
MONENDY

Chart GL. PRINTBUF




A1
PRINTIT
PRINTIT
B1
PRINT EVENT
C1
OUTCL = FASTREC OR PSW
YES
NO
D1
FORMAT AND PRINT GEN. PURPOSE REGISTERS
E1
OUTCL = GPR
YES
E3
NO
F1
FORMAT AND PRINT CONTROL REGS AND LOW REAL STORAGE
G1
OUTCL = LOCORE
YES
D3
NO
H1
OUTCL = COMREG
YES
CMRG
H2
DUMP CURRENT AND SYSTEM COMREG
D3
NO
J1
OUTCL = PAGETAB
YES
VIRTAB
J2
DUMP SEGMENT, PAGE, AND PAGE-FRAME TABLES
NO
B3
B3
B3
OUTCL = SUPVISOR
YES
SVDUMP
B4
TRANSLATE AND DUMP THE SUPERVISOR
NO
C3
OUTCL = DUMPREAL
YES
REALPRT
C4
TRANSLATE AND DUMP ALL OF REAL STORAGE
NO
D3
HIGHEND
D3
RELOAD ADDRESS OF PARAMETER LIST
E3
FORGET
E3
CLOSE OUTPUT FILE
F3
RETURN

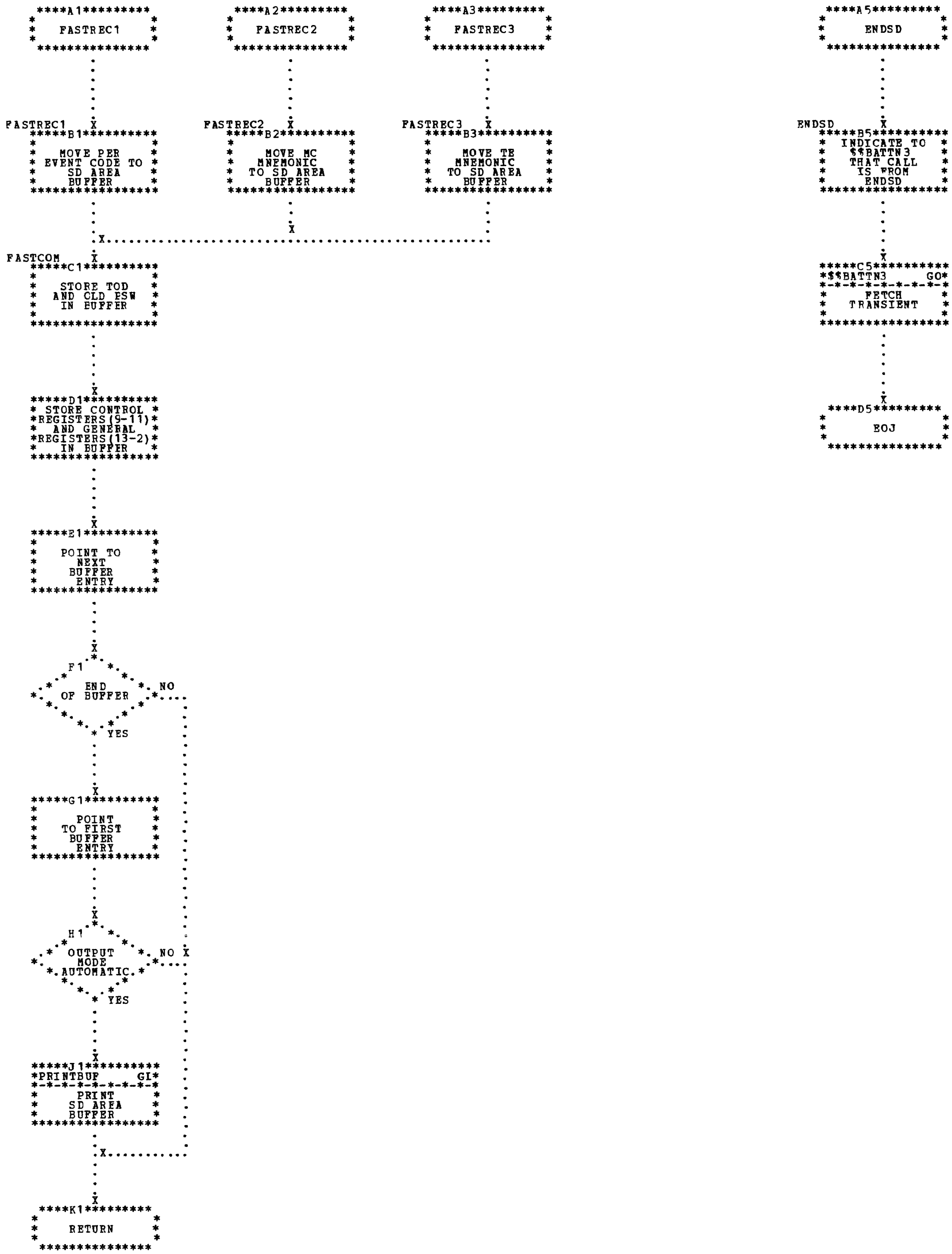
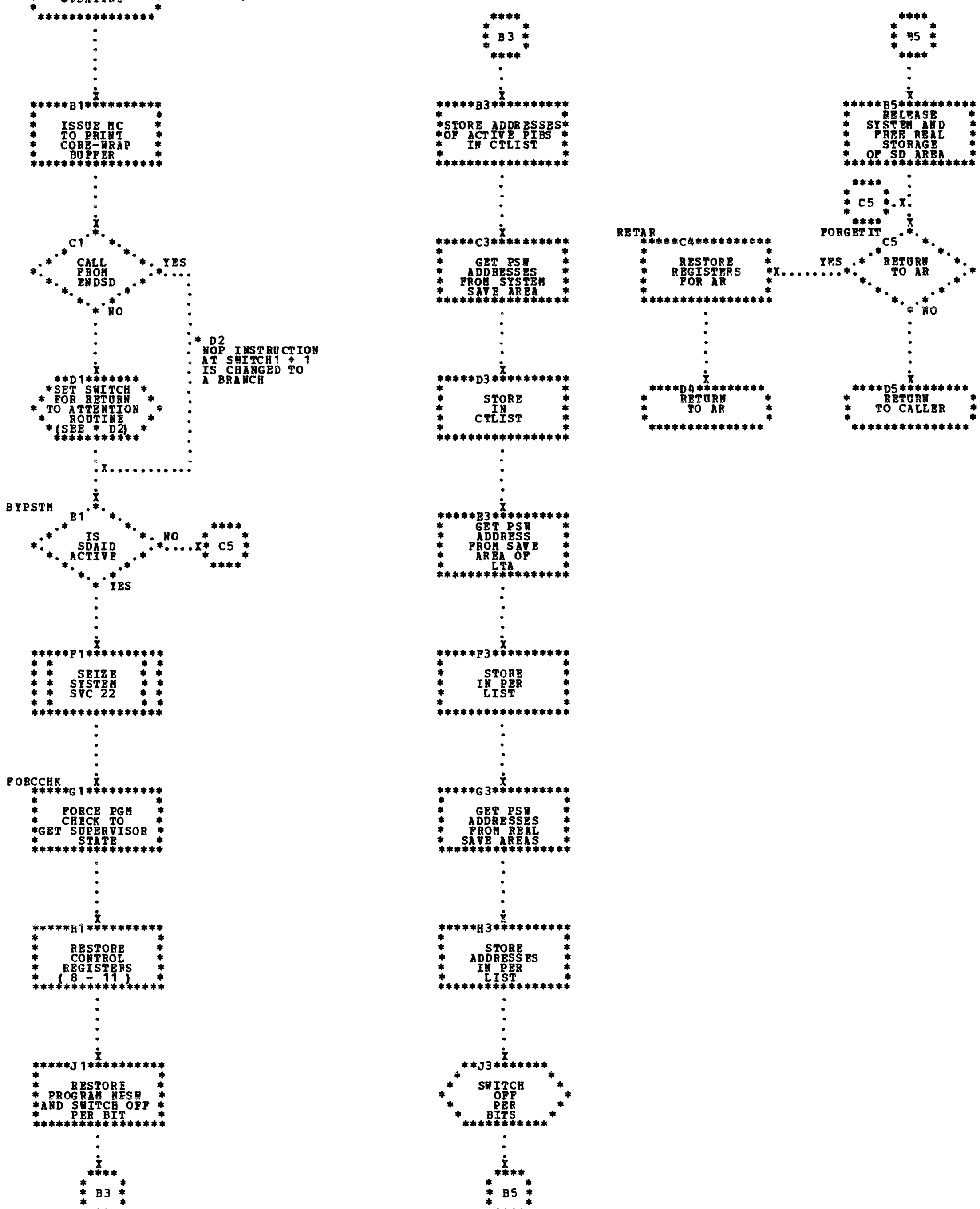

A1 FASTREC1
A2 FASTREC2
A3 FASTREC3
A5 ENDSD
FASTREC1 B1 MOVE PER EVENT CODE TO SD AREA BUFFER
FASTREC2 B2 MOVE MC MNEMONIC TO SD AREA BUFFER
FASTREC3 B3 MOVE TE MNEMONIC TO SD AREA BUFFER
ENDSD B5 INDICATE TO $$BATTN3 THAT CALL IS FROM ENDSD
FASTCOM C1 STORE TOD AND OLD PSW IN BUFFER
C5 $$BATTN3 GO FETCH TRANSIENT
D1 STORE CONTROL REGISTERS(9-11) AND GENERAL REGISTERS(13-2) IN BUFFER
D5 EOJ
E1 POINT TO NEXT BUFFER ENTRY
F1 END OF BUFFER
NO
YES
G1 POINT TO FIRST BUFFER ENTRY
H1 OUTPUT MODE AUTOMATIC
NO
YES
J1 PRINTBUF GI PRINT SD AREA BUFFER
K1 RETURN

Chart GP. $$BATTN3



DUMPGEN generates a translating stand-alone dump program (called REALDUMP), which can be tailored to individual system requirements. Using the DUMPGEN OPTN control statements you can generate a stand-alone dump, which can dump the contents of real storage (unformatted or formatted with DOS-VS supervisor tables), as well as the formatted contents of the boundary box and the page pool.

EXECUTING DUMPGEN
A self-relocating copy of DUMPGEN is provided in the relocatable library of the IBM-supplied system, and must be cataloged to the core image library before execution. It can be executed in any partition by the command:

// EXEC DUMPGEN

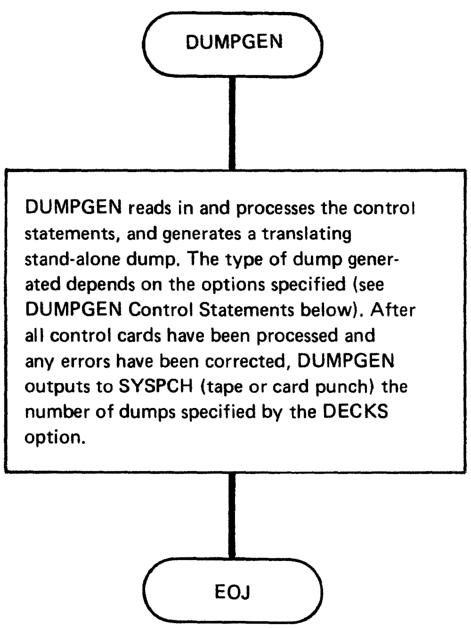


Chart 07. DUMPGEN (Dump Generator Program)

The entry point to DUMPGEN is BEGIN. The control statements (ASSGN and OPTN) are read from SYSIPT.

DUMPGEN Control Statements

ASSGN: The ASSGN control statement defines the output device for the stand-alone dump records. The format of the ASSGN statement is:

ASSGN SYSLST,X'cuu'

SYSLST is the only possible logical unit assignment.

X'cuu' must define the address of the SYSLST printer. If the ASSGN statement is omitted, X'00E' is assumed.

OPTN: The OPTN control statement is used to specify

- the type of printer-start control (manual or automatic),
- the number of card decks to be punched,
- the format of the dump,
- the type of tape output,
- a printout of the boundary box and page pool.

The format of the OPTN statement is:

OPTN operand

Figure 48 lists the operands for the OPTN statement. If any or all of the OPTN statements are omitted, the default operands, which are underlined, apply.

Operand	Description
INTR=YES	Produces a dump program that, when loaded, enters the WAIT state. Either press the interrupt button to print the output on X'00E' or first press the STOP button and then the START button of the printer to be used as output.
<u>INTR=NO</u>	Produces a dump program that prints the output on the SYSLST printer defined with the ASSGN statement or, if no ASSGN statement is specified, on X'00E'.
DECKS=nnnnnnnn	Specifies the number of REALDUMP card decks to be punched on SYSPCH. 1 to 99999999 may be specified. A blank card separates each deck produced.
<u>DECKS=1</u>	
FORMAT=YES	Produces a formatting dump that (if the BG communications region can be found) formats and displays the DOS/VS supervisor tables.
<u>FORMAT=NO</u>	A non-formatting dump is generated.
TAPEIPL=YES	If SYSPCH is assigned to a tape unit, the records of the stand-alone dump program are written on tape; the dump program can be IPLed directly from the tape.
<u>TAPEIPL=NO</u>	If SYSPCH is assigned to a tape, the stand-alone dump program is written on tape preceded by an ASA character.
PPOOL=YES	The formatted contents of the boundary box and the page pool are printed (in sequence of ascending virtual addresses).
<u>PPOOL=NO</u>	No further information is printed.

Figure 48. OPTN Statement Operands

The control statements may be specified in any order and number; however, the following rules apply:

1. The last statement processed overrides any previous statements with the same operation and operand. For example, if DECKS=2 is followed by DECKS=5, five card decks are punched.

2. Decimal operands may contain leading zeros.

3. The operation code must be preceded by one or more blanks.

4. Only one operation and only one operand per control statement is allowed.

5. One or more blanks must follow the operand, if comments are desired.

6. The parameters (YES, NO, and nnnnnnnn) must follow the equal sign without intervening blanks.

7. Column 71 must be left blank.

DUMPGEN MESSAGES

The DUMPGEN-to-operator error message routines make it possible to:

- Cancel the job if SYSLOG is not a console printer keyboard or a display operator console

- Reissue the message if operator response is ALTERNATE CANCEL

- Process an operator response of END as IGNORE

- Cancel the job if operator response is CANCEL

- Ignore the control card in question when operator response is IGNORE.

If none of the preceding operator responses is issued, DUMPGEN assumes a correction has been made and processes it. See Appendix A for a cross-reference list of message numbers and flowcharts.

REALDUMP (STAND-ALONE DUMP)

REALDUMP is the translating stand-alone dump generated by DUMPGEN. REALDUMP may be either formatting or non-formatting, depending on the FORMAT operand specified in the OPTN statement.

If FORMAT=NO was specified, the output from REALDUMP is a printout of:

- The general registers
- Formatted low real storage
- Real storage
- Page status information
- The control registers.

If FORMAT=YES was specified (and the BG communications region can be located), the DOS/VS supervisor tables are formatted and printed, following the dump of real storage.

For an example of the formatted REALDUMP output, refer to DOS/VS Serviceability Aids and Debugging Procedures, GC33-5380, Appendix G.

EXECUTING REALDUMP

The stand-alone dump is loaded directly into the system (using the normal IPL procedure), destroying the contents of bytes 0-23 (the IPL card) and the 210 bytes occupied by the dump program. These bytes are located in the program check area of the supervisor. The address of this area is contained in the last four bytes of the Program New PSW.

If information from any of these areas is needed, it must be manually displayed on the console, using the DM (display memory) command, before loading the dump program.

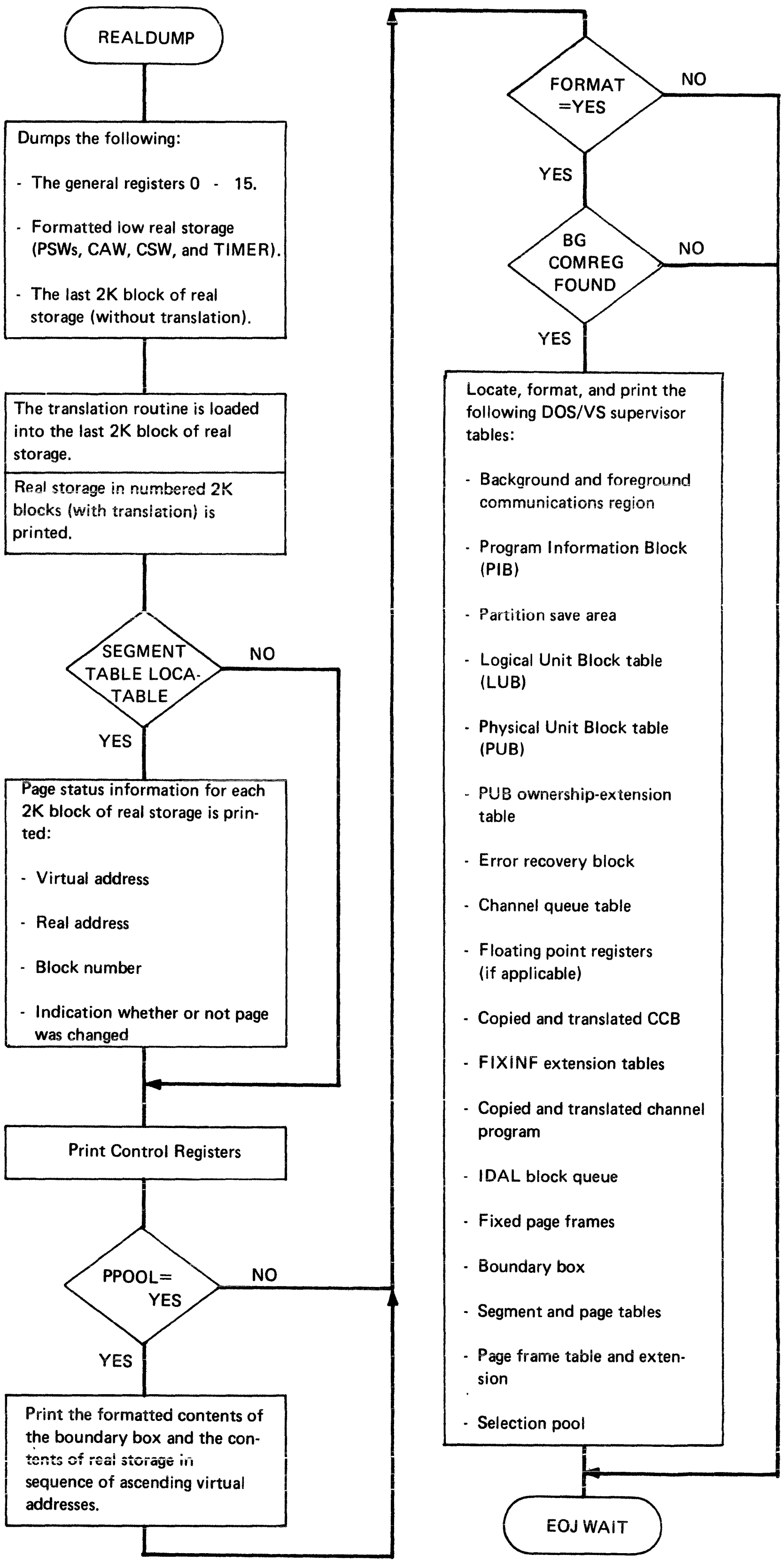
REALDUMP OUTPUT

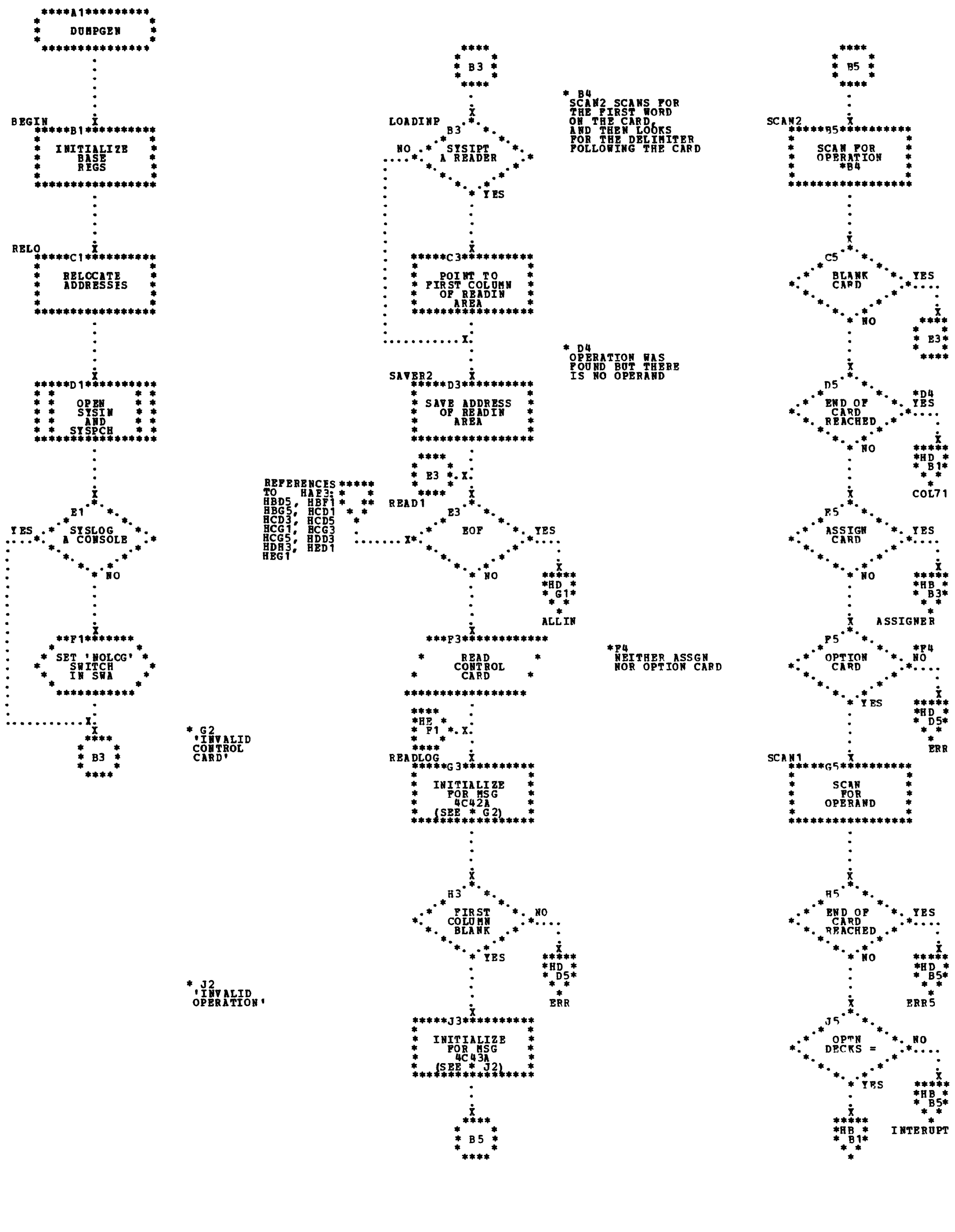
Each line of the output listing contains a maximum of eight fullwords. If the remaining portion of any line or group of lines is identical to the first word to be printed, the first word and the word "SAME" are printed. Printing is then suspended until a word with different characters is encountered.

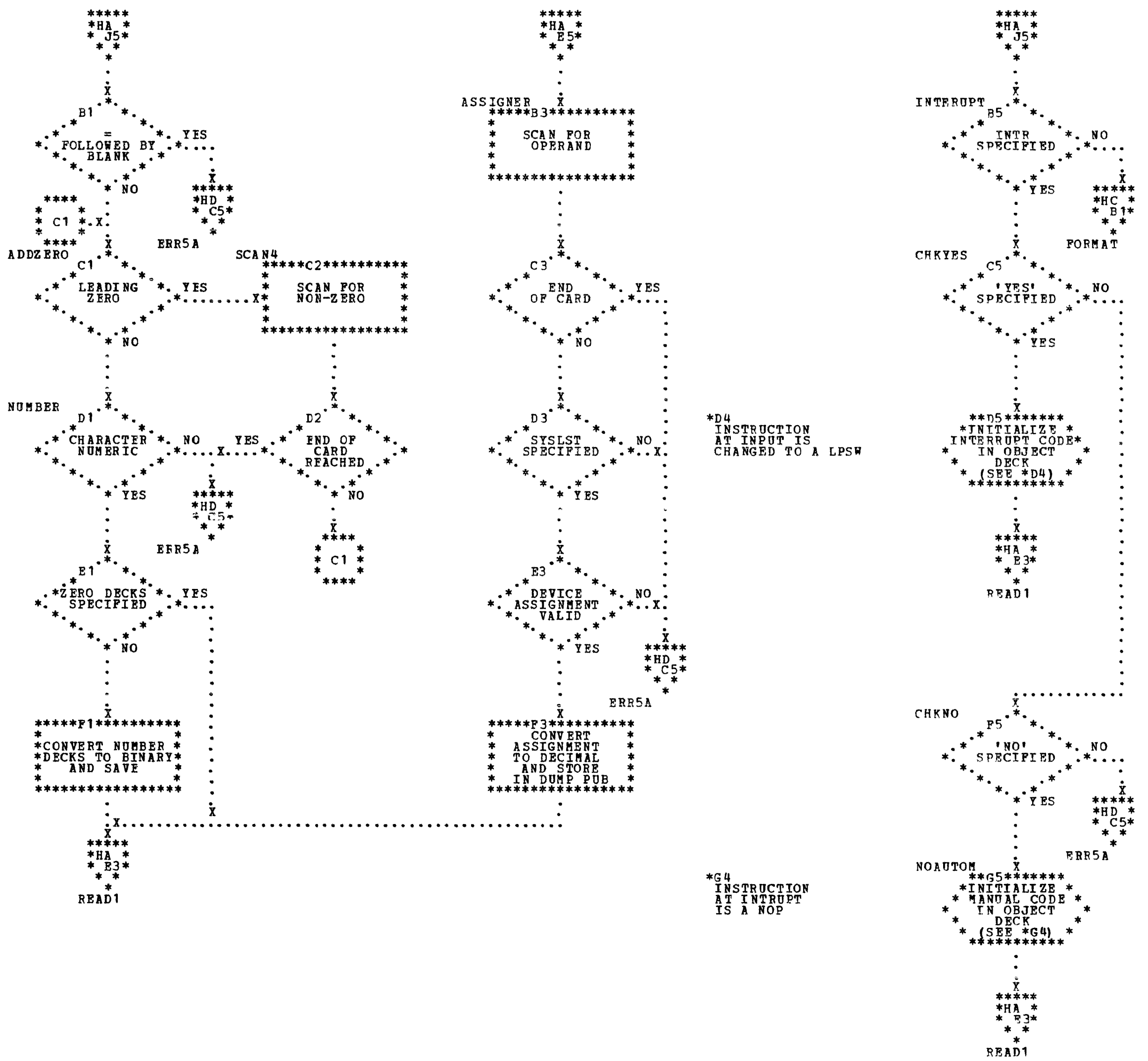
If a line is identical to the line previously printed, the word SAME is printed and printing is suspended until a line with different contents is found.

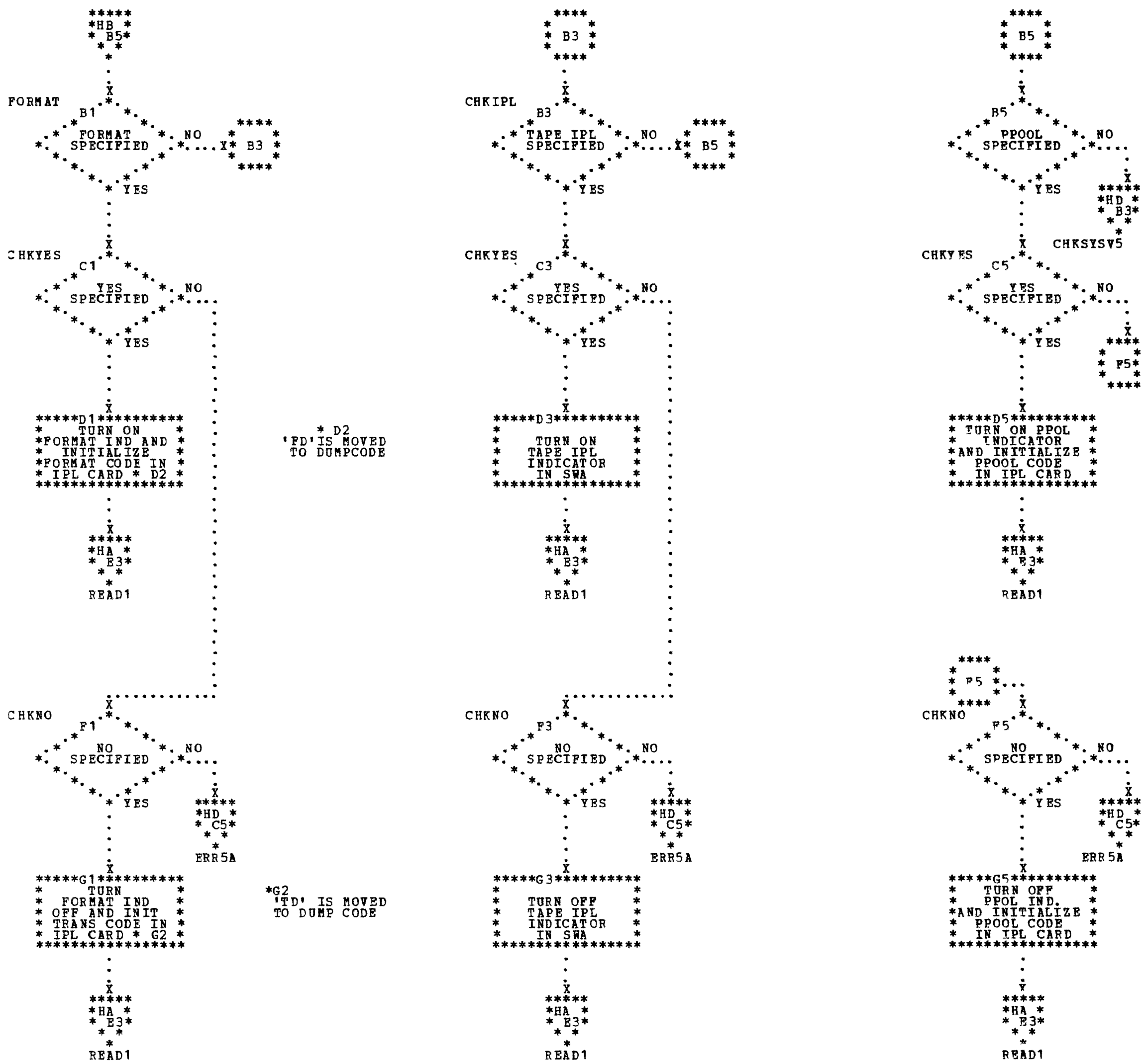
The first and last line of every 2K block are printed, regardless of their contents.

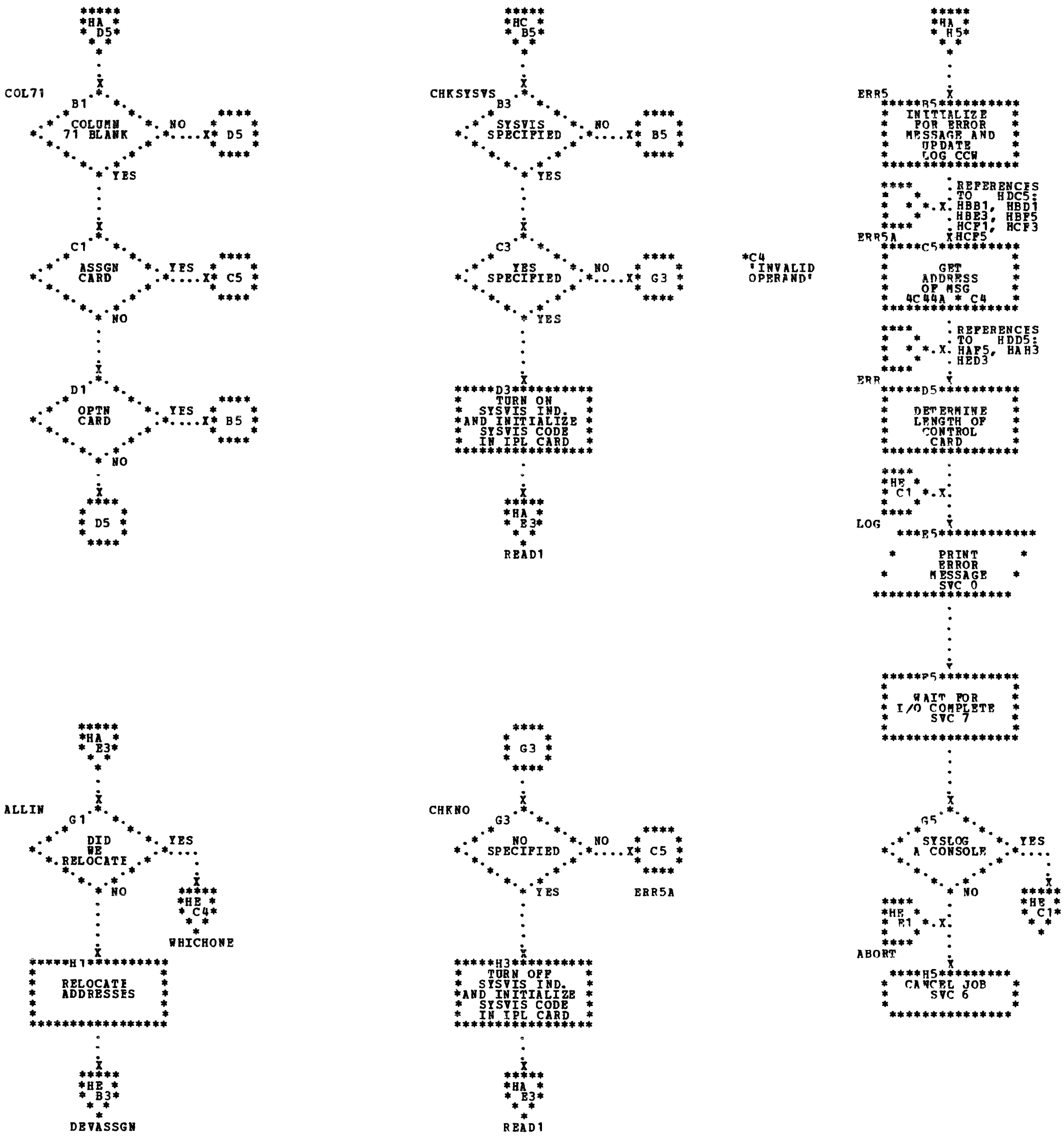
Chart 08. Logic Flow of the REALDUMP Program

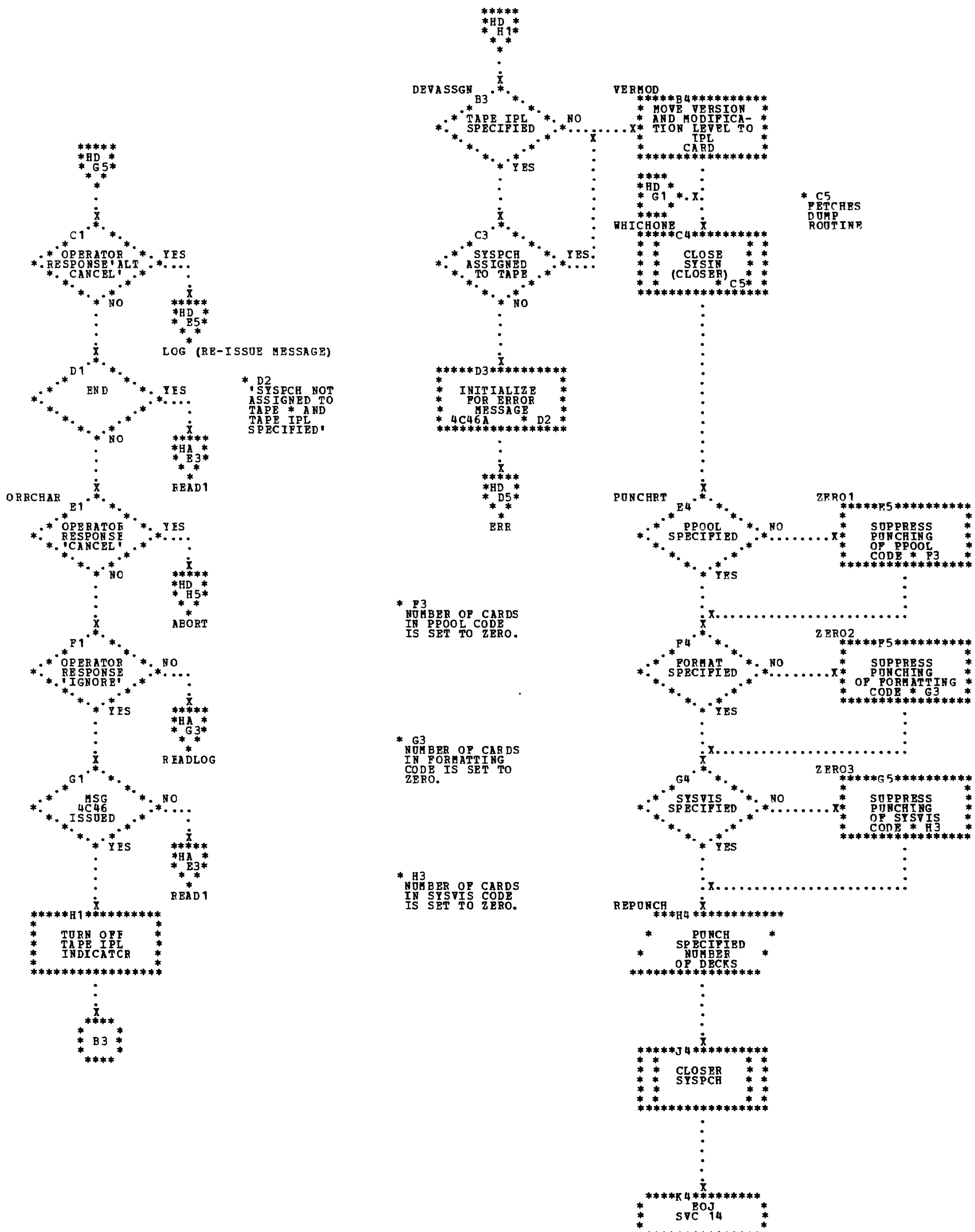


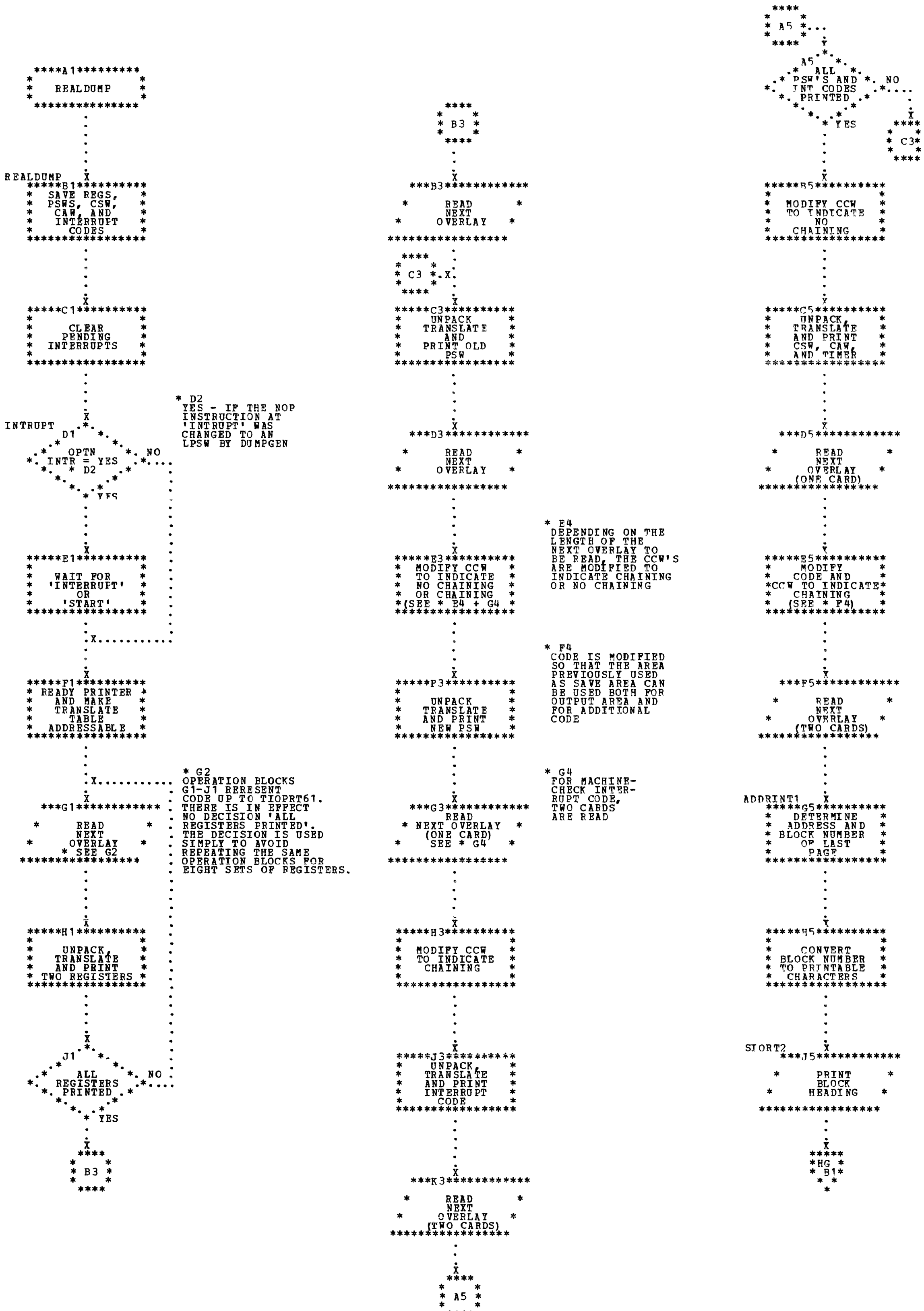

A1 DUMPGEN
BEGIN B1 INITIALIZE BASE REGS
RELO C1 RELOCATE ADDRESSES
D1 OPEN SYSIN AND SYSPCH
E1 SYSLOG A CONSOLE YES NO
F1 SET 'NOLOG' SWITCH IN SWA
B3
* G2 'INVALID CONTROL CARD'
* J2 'INVALID OPERATION'
REFERENCES TO HAE3: HBD5, HBF1 HBG5, HCD1 HCD3, HCD5 HCG1, HCG3 HCG5, HDD3 HDH3, HED1 HEG1
B3
LOADINP B3 SYSIPT A READER NO YES
C3 POINT TO FIRST COLUMN OF READIN AREA
SAVER2 D3 SAVE ADDRESS OF READIN AREA
B3
READ1 E3 EOF YES NO
HD G1 ALLIN
F3 READ CONTROL CARD
HE F1
READLOG G3 INITIALIZE FOR MSG 4C42A (SEE * G2)
H3 FIRST COLUMN BLANK NO YES
HD D5 ERR
J3 INITIALIZE FOR MSG 4C43A (SEE * J2)
B5
* B4 SCAN2 SCANS FOR THE FIRST WORD ON THE CARD, AND THEN LOOKS FOR THE DELIMITER FOLLOWING THE CARD
* D4 OPERATION WAS FOUND BUT THERE IS NO OPERAND
*F4 NEITHER ASSGN NOR OPTION CARD
B5
SCAN2 B5 SCAN FOR OPERATION *B4
C5 BLANK CARD YES NO
E3
D5 END OF CARD REACHED *D4 YES NO
HD B1 COL71
E5 ASSIGN CARD YES NO
HB B3 ASSIGNER
F5 OPTION CARD *F4 NO YES
HD D5 ERR
SCAN1 G5 SCAN FOR OPERAND
H5 END OF CARD REACHED YES NO
HD B5 ERR5
J5 OPTN DECKS = NO YES
HB B5 INTERUPT
HB B1

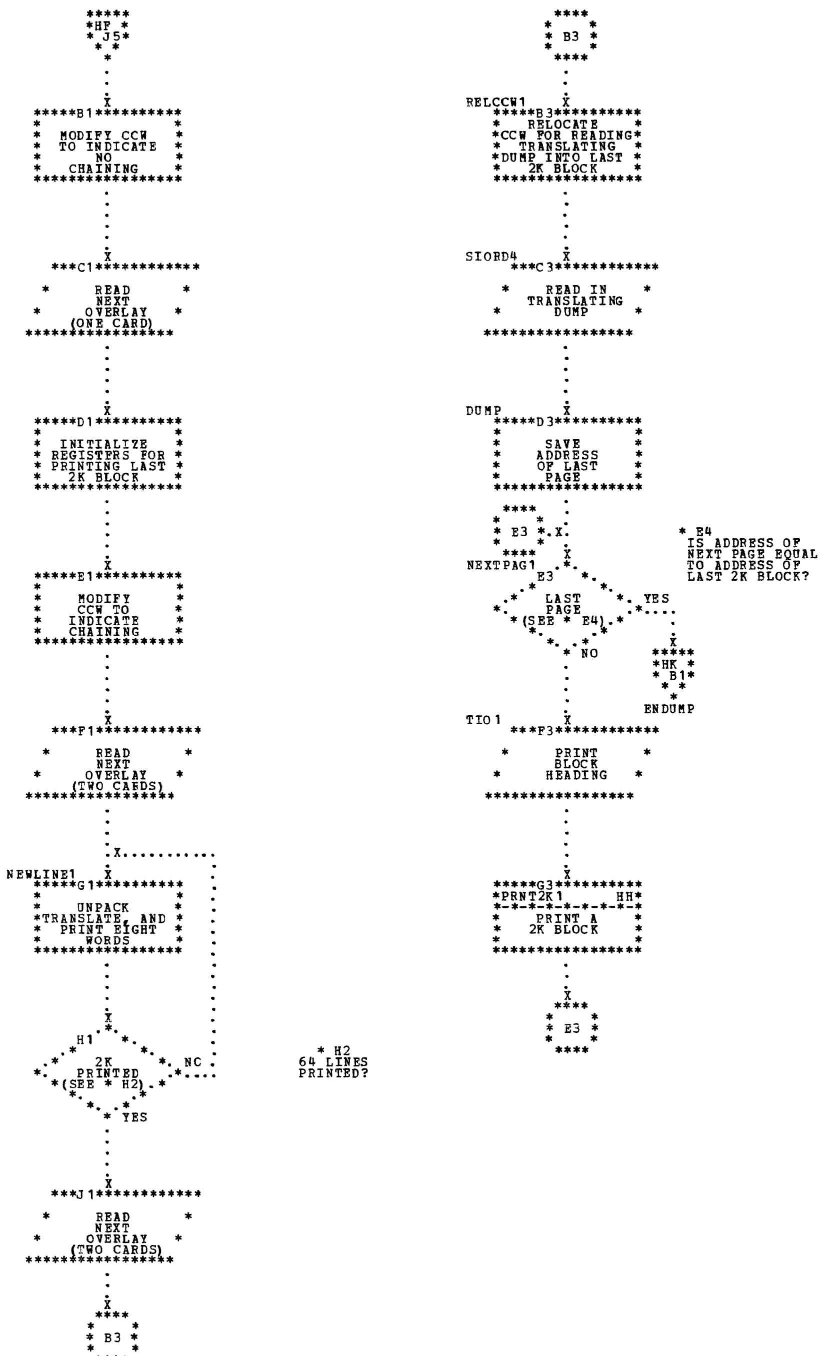

HA J5
B1 = FOLLOWED BY BLANK
YES
NO
HD C5
ERR5A
C1
ADDZERO
C1 LEADING ZERO
YES
NO
SCAN4
C2 SCAN FOR NON-ZERO
NUMBER
D1 CHARACTER NUMERIC
NO
YES
HD C5
ERR5A
D2 END OF CARD REACHED
YES
NO
C1
E1 ZERO DECKS SPECIFIED
YES
NO
F1 CONVERT NUMBER DECKS TO BINARY AND SAVE
HA E3
READ1
HA E5
ASSIGNER
B3 SCAN FOR OPERAND
C3 END OF CARD
YES
NO
D3 SYSLST SPECIFIED
NO
YES
E3 DEVICE ASSIGNMENT VALID
NO
YES
HD C5
ERR5A
F3 CONVERT ASSIGNMENT TO DECIMAL AND STORE IN DUMP PUB
*D4 INSTRUCTION AT INPUT IS CHANGED TO A LPSW
*G4 INSTRUCTION AT INTRUPT IS A NOP
HA J5
INTRRUPT
B5 INTR SPECIFIED
NO
YES
HC B1
FORMAT
CHKYES
C5 'YES' SPECIFIED
NO
YES
D5 INITIALIZE INTERRUPT CODE IN OBJECT DECK (SEE *D4)
HA E3
READ1
CHKNO
F5 'NO' SPECIFIED
NO
YES
HD C5
ERR5A
NOAUTOM
G5 INITIALIZE MANUAL CODE IN OBJECT DECK (SEE *G4)
HA E3
READ1

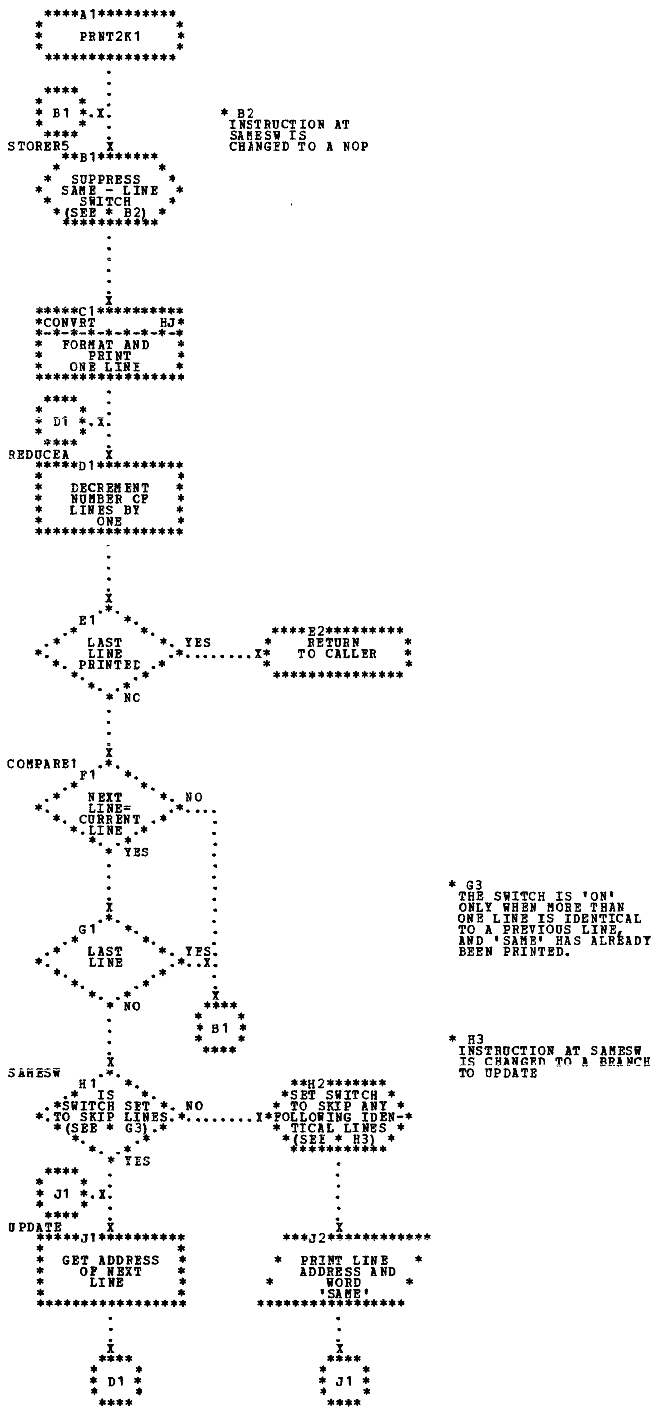

HB B5
FORMAT
B1
FORMAT SPECIFIED
NO
B3
YES
CHKYES
C1
YES SPECIFIED
NO
YES
D1
TURN ON FORMAT IND AND INITIALIZE FORMAT CODE IN IPL CARD * D2
* D2 'FD' IS MOVED TO DUMPCODE
HA E3
READ1
CHKNO
F1
NO SPECIFIED
NO
YES
HD C5
ERR5A
G1
TURN FORMAT IND OFF AND INIT TRANS CODE IN IPL CARD * G2
*G2 'TD' IS MOVED TO DUMP CODE
HA E3
READ1
B3
CHKIPL
B3
TAPE IPL SPECIFIED
NO
B5
YES
CHKYES
C3
YES SPECIFIED
NO
YES
D3
TURN ON TAPE IPL INDICATOR IN SWA
HA B3
READ1
CHKNO
F3
NO SPECIFIED
NO
YES
HD C5
ERR5A
G3
TURN OFF TAPE IPL INDICATOR IN SWA
HA E3
READ1
B5
B5
PPOOL SPECIFIED
NO
YES
HD B3
CHKSYSV5
CHKYES
C5
YES SPECIFIED
NO
YES
F5
D5
TURN ON PPOL INDICATOR AND INITIALIZE PPOOL CODE IN IPL CARD
HA E3
READ1
F5
CHKNO
F5
NO SPECIFIED
NO
YES
HD C5
ERR5A
G5
TURN OFF PPOL IND. AND INITIALIZE PPOOL CODE IN IPL CARD
HA E3
READ1

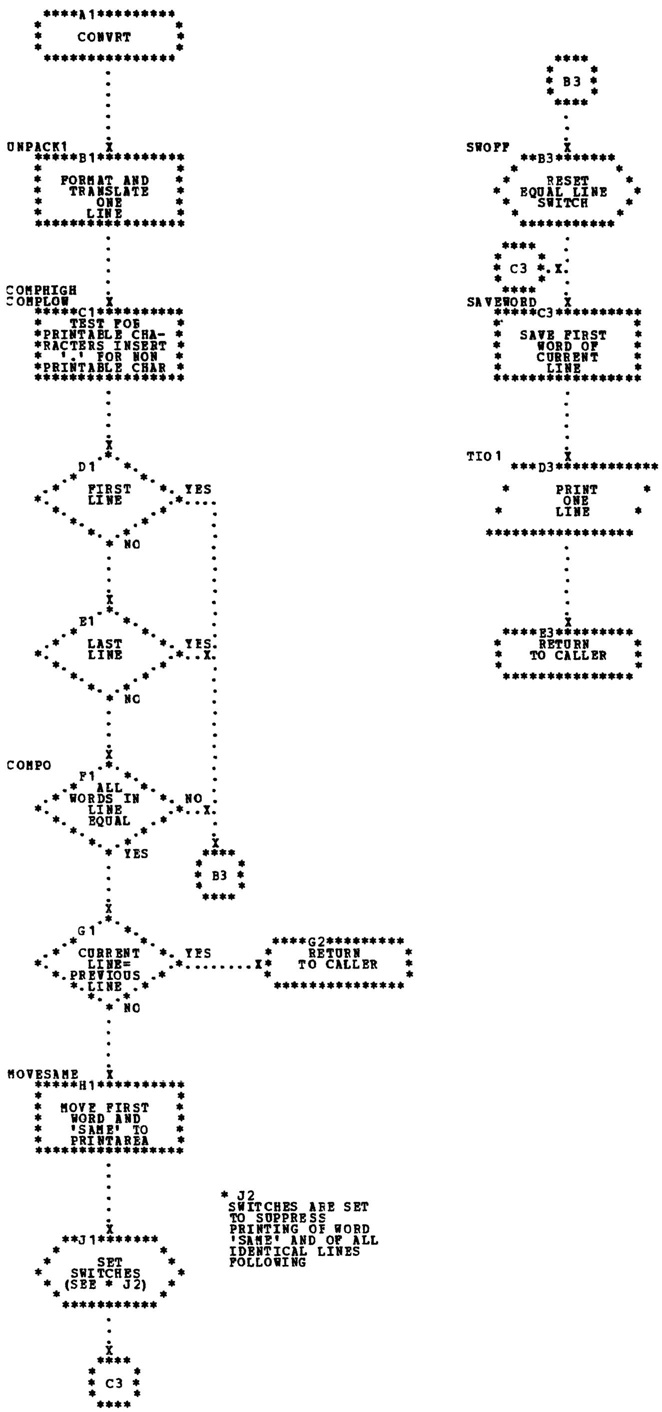

HA D5
COL71
B1
COLUMN 71 BLANK
NO
D5
YES
C1
ASSGN CARD
YES
C5
NO
D1
OPTN CARD
YES
B5
NO
D5
HC B5
CHKSYSVS
B3
SYSVIS SPECIFIED
NO
B5
YES
C3
YES SPECIFIED
NO
G3
YES
D3
TURN ON SYSVIS IND. AND INITIALIZE SYSVIS CODE IN IPL CARD
HA B3
READ1
*C4 'INVALID OPERAND'
HA H5
ERR5
B5
INITIALIZE FOR ERROR MESSAGE AND UPDATE LOG CCW
REFERENCES TO HDC5: HBB1, HBD1 HBE3, HBF5 HCF1, HCF3 HCF5
ERR5A
C5
GET ADDRESS OF MSG 4C44A * C4
REFERENCES TO HDD5: HAF5, HAH3 HED3
ERR
D5
DETERMINE LENGTH OF CONTROL CARD
HE C1
LOG
E5
PRINT ERROR MESSAGE SVC 0
F5
WAIT FOR I/O COMPLETE SVC 7
G5
SYSLOG A CONSOLE
YES
HE C1
NO
HE E1
ABORT
H5
CANCEL JOB SVC 6
HA E3
ALLIN
G1
DID WE RELOCATE
YES
HE C4
WHICHONE
NO
H1
RELOCATE ADDRESSES
HE B3
DEVASSGN
G3
CHKNO
G3
NO SPECIFIED
NO
C5
ERR5A
YES
H3
TURN OFF SYSVIS IND. AND INITIALIZE SYSVIS CODE IN IPL CARD
HA B3
READ1

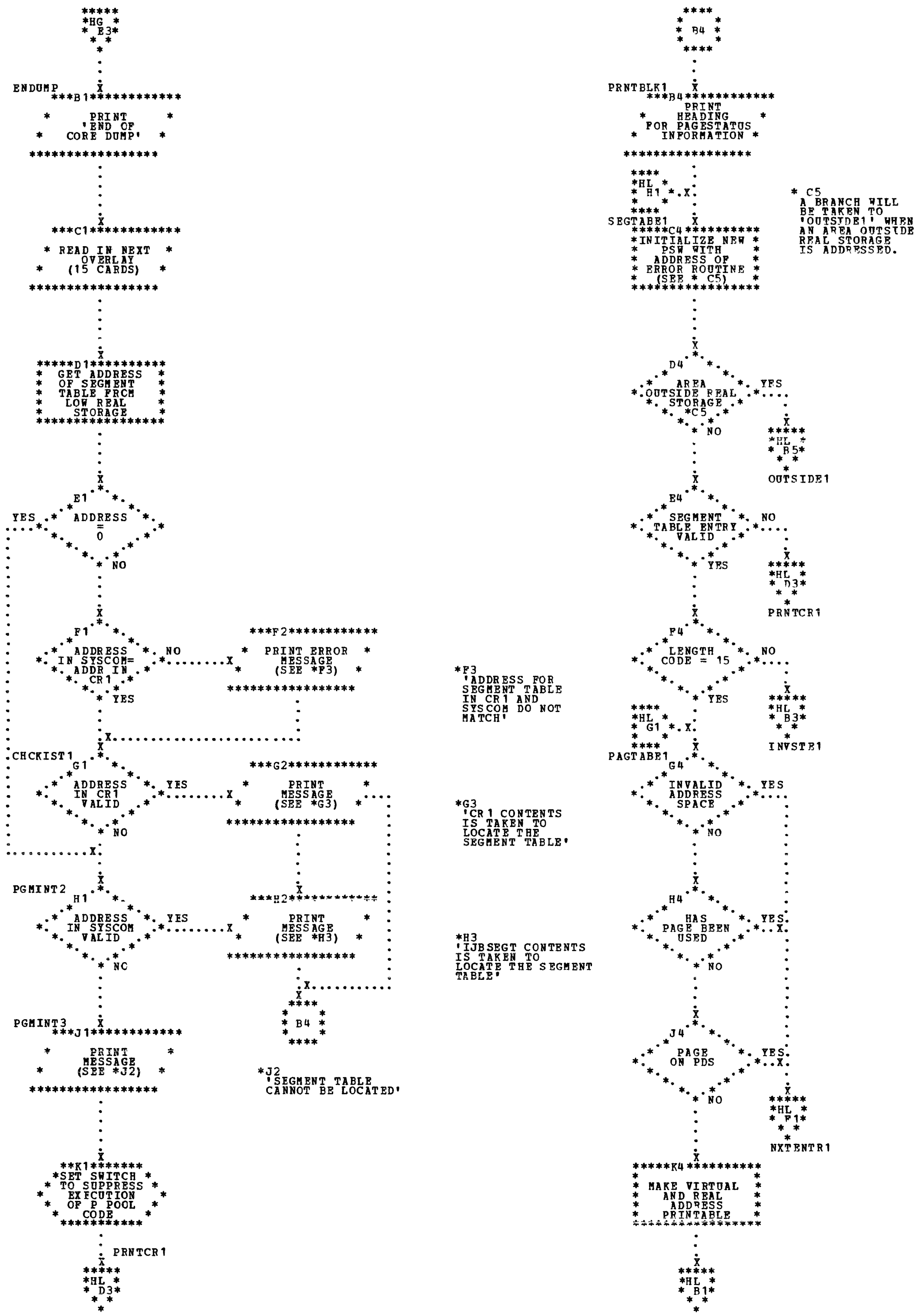

HD H1
DEVASSGN B3
TAPE IPL SPECIFIED
NO
YES
VERMOD B4
MOVE VERSION AND MODIFICATION LEVEL TO IPL CARD
HD G1
C5 FETCHES DUMP ROUTINE
C3
SYSPCH ASSIGNED TO TAPE
YES
NO
WHICHONE C4
CLOSE SYSIN (CLOSER) C5
D3
INITIALIZE FOR ERROR MESSAGE 4C46A D2
HD D5
ERR
HD G5
C1
OPERATOR RESPONSE 'ALT CANCEL'
YES
NO
HD E5
LOG (RE-ISSUE MESSAGE)
D1
END
YES
NO
D2 'SYSPCH NOT ASSIGNED TO TAPE * AND TAPE IPL SPECIFIED'
HA E3
READ1
ORRCHAR E1
OPERATOR RESPONSE 'CANCEL'
YES
NO
HD H5
ABORT
F1
OPERATOR RESPONSE 'IGNORE'
NO
YES
HA G3
READLOG
G1
MSG 4C46 ISSUED
NO
YES
HA E3
READ1
H1
TURN OFF TAPE IPL INDICATOR
B3
F3 NUMBER OF CARDS IN PPOOL CODE IS SET TO ZERO.
G3 NUMBER OF CARDS IN FORMATTING CODE IS SET TO ZERO.
H3 NUMBER OF CARDS IN SYSVIS CODE IS SET TO ZERO.
PUNCHRT E4
PPOOL SPECIFIED
NO
YES
ZERO1 E5
SUPPRESS PUNCHING OF PPOOL CODE F3
F4
FORMAT SPECIFIED
NO
YES
ZERO2 F5
SUPPRESS PUNCHING OF FORMATTING CODE G3
G4
SYSVIS SPECIFIED
NO
YES
ZERO3 G5
SUPPRESS PUNCHING OF SYSVIS CODE H3
REPUNCH H4
PUNCH SPECIFIED NUMBER OF DECKS
J4
CLOSER SYSPCH
K4
EOJ SVC 14

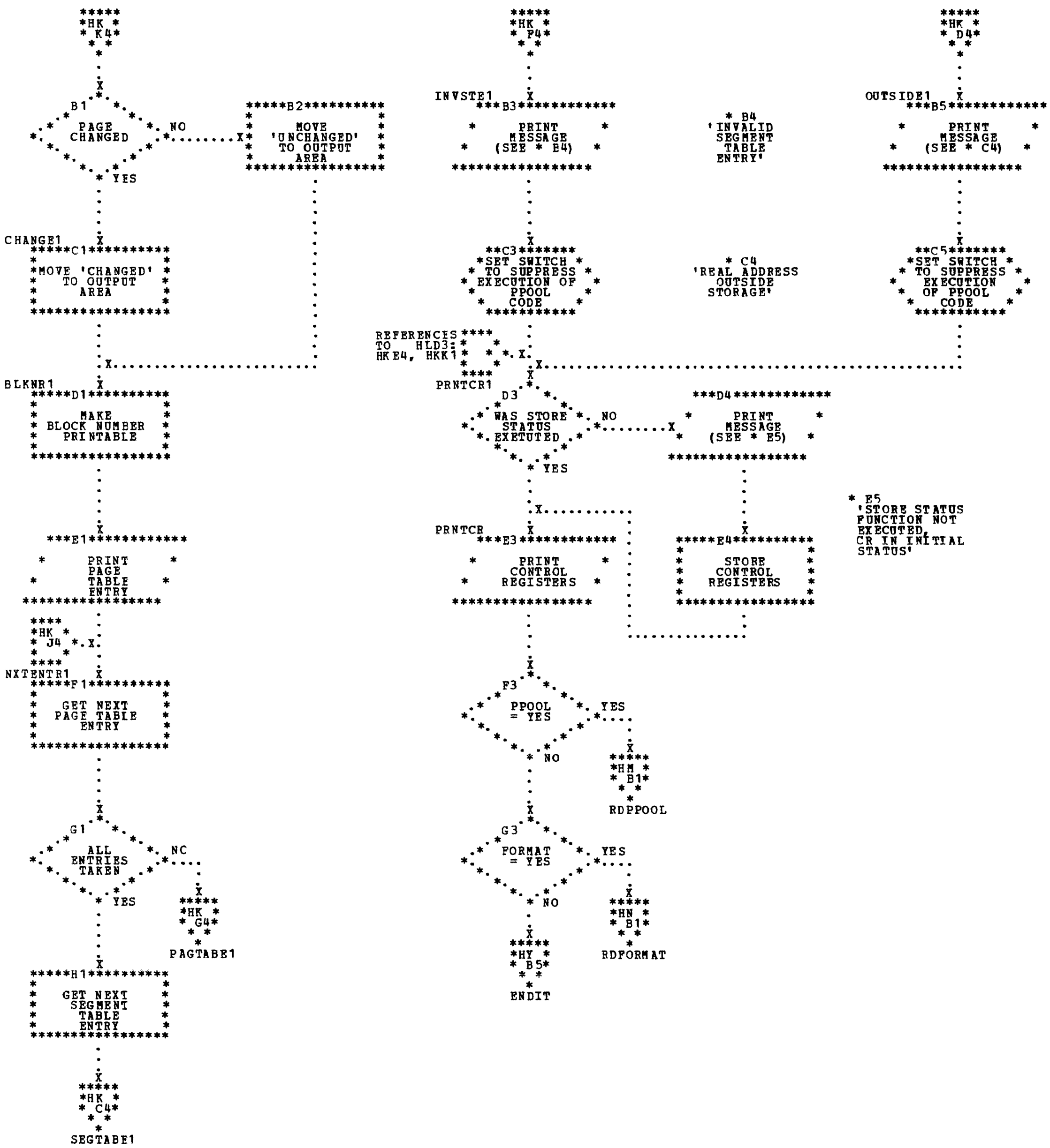

A1
REALDUMP
REALDUMP
B1
SAVE REGS, PSWS, CSW, CAW, AND INTERRUPT CODES
C1
CLEAR PENDING INTERRUPTS
INTRUPT
D1
OPTN INTR = YES * D2
NO
YES
* D2
YES - IF THE NOP INSTRUCTION AT 'INTRUPT' WAS CHANGED TO AN LPSW BY DUMPGEN
E1
WAIT FOR 'INTERRUPT' OR 'START'
F1
READY PRINTER AND MAKE TRANSLATE TABLE ADDRESSABLE
G1
READ NEXT OVERLAY * SEE G2
* G2
OPERATION BLOCKS G1-J1 RERESENT CODE UP TO TIOPRT61. THERE IS IN EFFECT NO DECISION 'ALL REGISTERS PRINTED'. THE DECISION IS USED SIMPLY TO AVOID REPEATING THE SAME OPERATION BLOCKS FOR EIGHT SETS OF REGISTERS.
H1
UNPACK, TRANSLATE AND PRINT TWO REGISTERS
J1
ALL REGISTERS PRINTED
NO
YES
B3
B3
READ NEXT OVERLAY
C3
C3
UNPACK TRANSLATE AND PRINT OLD PSW
D3
READ NEXT OVERLAY
E3
MODIFY CCW TO INDICATE NO CHAINING OR CHAINING (SEE * E4 + G4
* E4
DEPENDING ON THE LENGTH OF THE NEXT OVERLAY TO BE READ, THE CCW'S ARE MODIFIED TO INDICATE CHAINING OR NO CHAINING
F3
UNPACK TRANSLATE AND PRINT NEW PSW
* F4
CODE IS MODIFIED SO THAT THE AREA PREVIOUSLY USED AS SAVE AREA CAN BE USED BOTH FOR OUTPUT AREA AND FOR ADDITIONAL CODE
G3
READ NEXT OVERLAY (ONE CARD) SEE * G4
* G4
FOR MACHINE-CHECK INTER-RUPT CODE, TWO CARDS ARE READ
H3
MODIFY CCW TO INDICATE CHAINING
J3
UNPACK, TRANSLATE AND PRINT INTERRUPT CODE
K3
READ NEXT OVERLAY (TWO CARDS)
A5
A5
A5
ALL PSW'S AND INT CODES PRINTED
NO
YES
C3
B5
MODIFY CCW TO INDICATE NO CHAINING
C5
UNPACK, TRANSLATE AND PRINT CSW, CAW, AND TIMER
D5
READ NEXT OVERLAY (ONE CARD)
E5
MODIFY CODE AND CCW TO INDICATE CHAINING (SEE * F4)
F5
READ NEXT OVERLAY (TWO CARDS)
ADDRINT1
G5
DETERMINE ADDRESS AND BLOCK NUMBER OF LAST PAGE
H5
CONVERT BLOCK NUMBER TO PRINTABLE CHARACTERS
STORT2
J5
PRINT BLOCK HEADING
HG B1

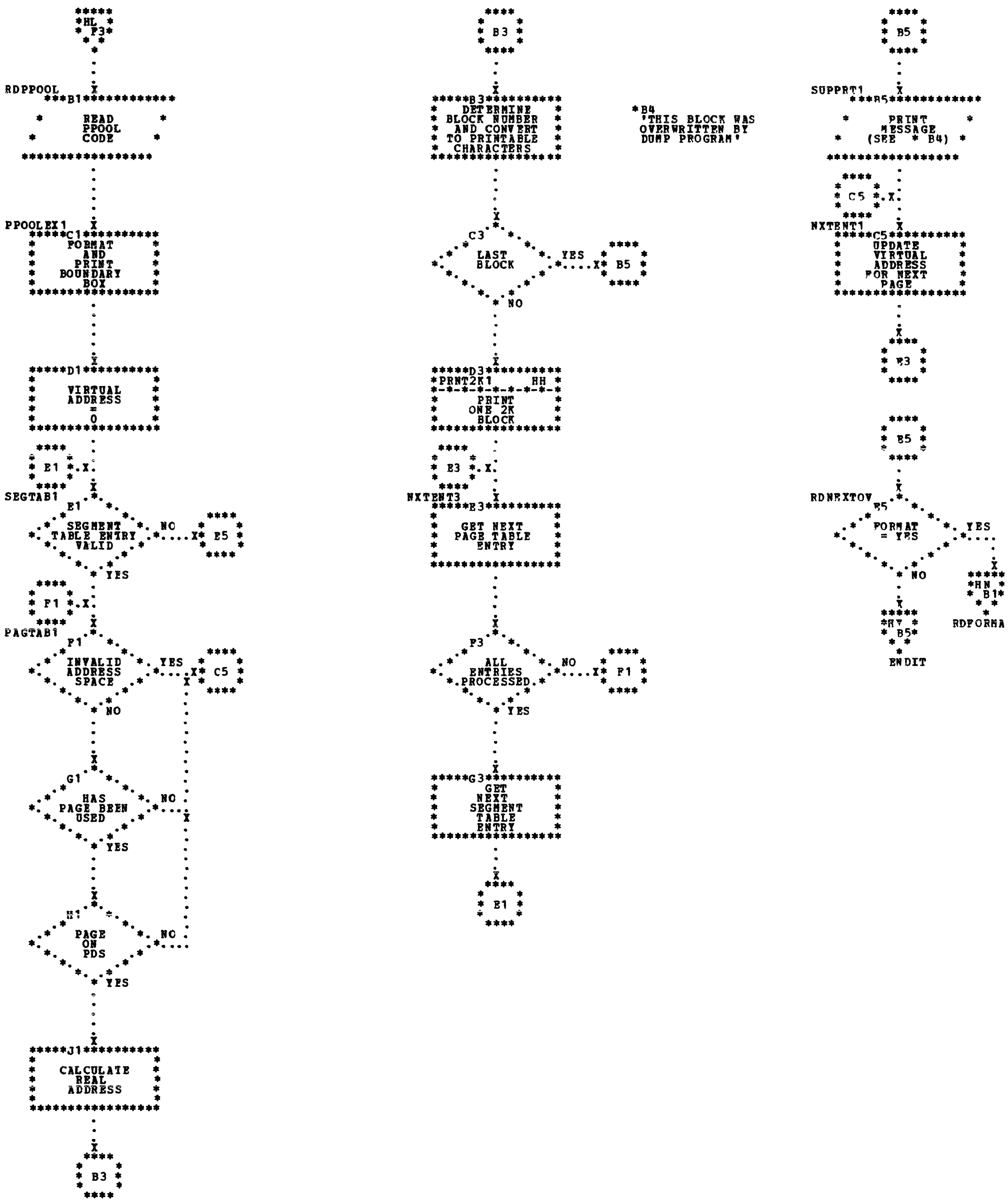

HF J5
B1
MODIFY CCW TO INDICATE NO CHAINING
C1
READ NEXT OVERLAY (ONE CARD)
D1
INITIALIZE REGISTERS FOR PRINTING LAST 2K BLOCK
E1
MODIFY CCW TO INDICATE CHAINING
F1
READ NEXT OVERLAY (TWO CARDS)
NEWLINE1
G1
UNPACK TRANSLATE, AND PRINT EIGHT WORDS
H1
2K PRINTED (SEE * H2)
NO
YES
* H2
64 LINES PRINTED?
J1
READ NEXT OVERLAY (TWO CARDS)
B3
B3
RELCCW1
B3
RELOCATE CCW FOR READING TRANSLATING DUMP INTO LAST 2K BLOCK
SIORD4
C3
READ IN TRANSLATING DUMP
DUMP
D3
SAVE ADDRESS OF LAST PAGE
E3
NEXTPAG1
E3
LAST PAGE (SEE * E4)
YES
NO
* E4
IS ADDRESS OF NEXT PAGE EQUAL TO ADDRESS OF LAST 2K BLOCK?
HK B1
ENDUMP
TIO1
F3
PRINT BLOCK HEADING
G3
PRNT2K1 HH
PRINT A 2K BLOCK
B3

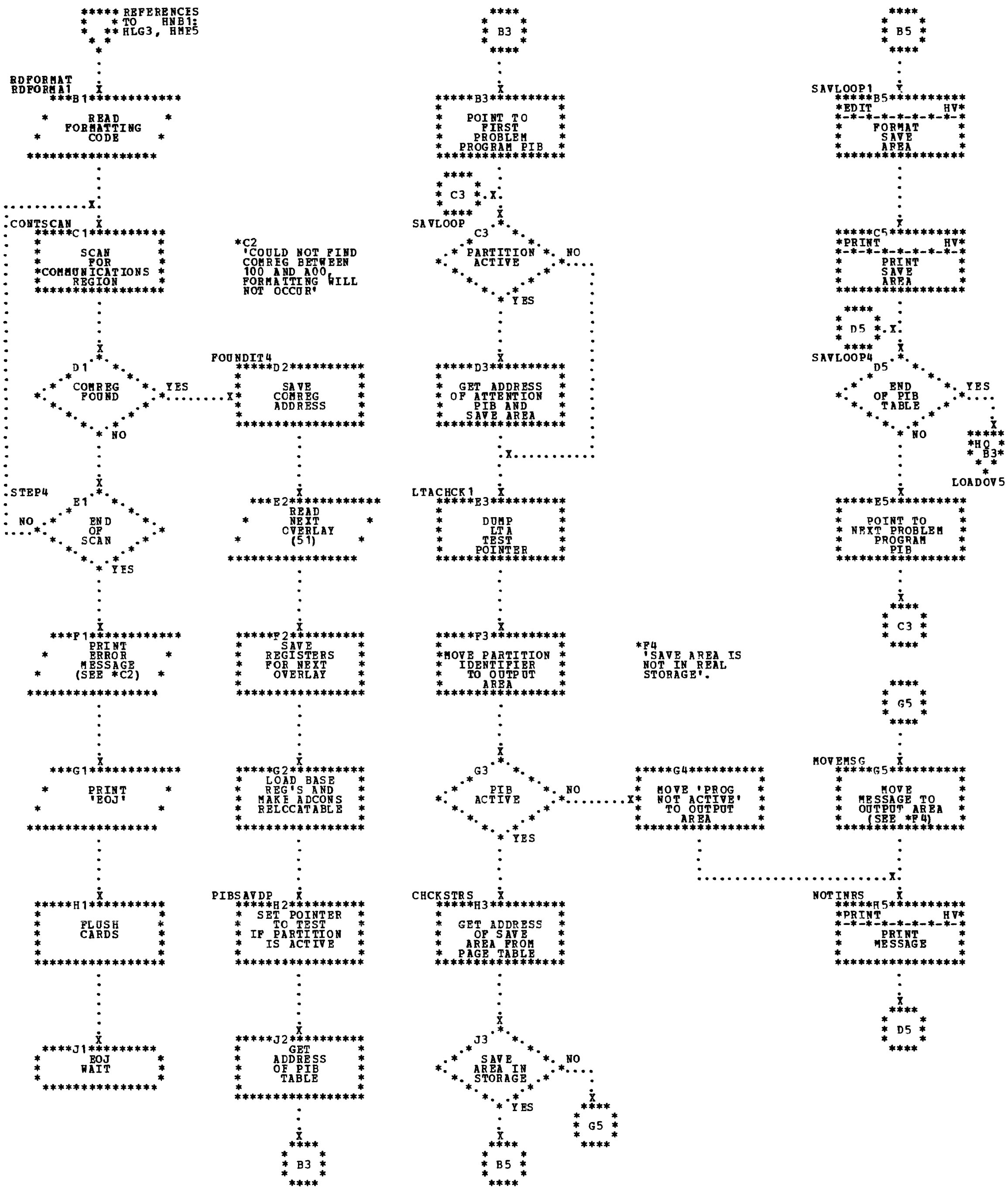

A1
PRNT2K1
B1
STORER5
B1
SUPPRESS SAME - LINE SWITCH (SEE * B2)
* B2
INSTRUCTION AT SAMESW IS CHANGED TO A NOP
C1
CONVRT HJ
FORMAT AND PRINT ONE LINE
D1
REDUCEA
D1
DECREMENT NUMBER OF LINES BY ONE
E1
LAST LINE PRINTED
YES
E2
RETURN TO CALLER
NO
COMPARE1
F1
NEXT LINE= CURRENT LINE
NO
YES
G1
LAST LINE
YES
B1
NO
* G3
THE SWITCH IS 'ON' ONLY WHEN MORE THAN ONE LINE IS IDENTICAL TO A PREVIOUS LINE, AND 'SAME' HAS ALREADY BEEN PRINTED.
SAMESW
H1
IS SWITCH SET TO SKIP LINES (SEE * G3)
NO
H2
SET SWITCH TO SKIP ANY FOLLOWING IDENTICAL LINES (SEE * H3)
* H3
INSTRUCTION AT SAMESW IS CHANGED TO A BRANCH TO UPDATE
YES
J1
UPDATE
J1
GET ADDRESS OF NEXT LINE
J2
PRINT LINE ADDRESS AND WORD 'SAME'
D1
J1


A1
CONVRT
UNPACK1
B1
FORMAT AND TRANSLATE ONE LINE
COMPHIGH
COMPLOW
C1
TEST FOR PRINTABLE CHA- RACTERS INSERT '.' FOR NON PRINTABLE CHAR
D1
FIRST LINE
YES
NO
E1
LAST LINE
YES
NO
COMPO
F1
ALL WORDS IN LINE EQUAL
NO
YES
B3
G1
CURRENT LINE= PREVIOUS LINE
YES
NO
G2
RETURN TO CALLER
MOVESAME
H1
MOVE FIRST WORD AND 'SAME' TO PRINTAREA
J1
SET SWITCHES (SEE * J2)
C3
* J2
SWITCHES ARE SET TO SUPPRESS PRINTING OF WORD 'SAME' AND OF ALL IDENTICAL LINES FOLLOWING
B3
SWOFF
B3
RESET EQUAL LINE SWITCH
C3
SAVEWORD
C3
SAVE FIRST WORD OF CURRENT LINE
TIO1
D3
PRINT ONE LINE
E3
RETURN TO CALLER


HG E3
ENDUMP B1
PRINT 'END OF CORE DUMP'
C1
READ IN NEXT OVERLAY (15 CARDS)
D1
GET ADDRESS OF SEGMENT TABLE FROM LOW REAL STORAGE
E1
ADDRESS = 0
YES
NO
F1
ADDRESS IN SYSCOM= ADDR IN CR1
NO
YES
F2
PRINT ERROR MESSAGE (SEE *F3)
*F3
'ADDRESS FOR SEGMENT TABLE IN CR1 AND SYSCOM DO NOT MATCH'
CHCKIST1 G1
ADDRESS IN CR1 VALID
YES
NO
G2
PRINT MESSAGE (SEE *G3)
*G3
'CR1 CONTENTS IS TAKEN TO LOCATE THE SEGMENT TABLE'
PGMINT2 H1
ADDRESS IN SYSCOM VALID
YES
NC
H2
PRINT MESSAGE (SEE *H3)
*H3
'IJBSEGT CONTENTS IS TAKEN TO LOCATE THE SEGMENT TABLE'
B4
PGMINT3 J1
PRINT MESSAGE (SEE *J2)
*J2
'SEGMENT TABLE CANNOT BE LOCATED'
K1
SET SWITCH TO SUPPRESS EXECUTION OF P POOL CODE
PRNTCR1
HL D3
B4
PRNTBLK1 B4
PRINT HEADING FOR PAGESTATUS INFORMATION
HL H1
SEGTABE1 C4
INITIALIZE NEW PSW WITH ADDRESS OF ERROR ROUTINE (SEE * C5)
* C5
A BRANCH WILL BE TAKEN TO 'OUTSIDE1' WHEN AN AREA OUTSIDE REAL STORAGE IS ADDRESSED.
D4
AREA OUTSIDE REAL STORAGE *C5
YES
NO
HL B5
OUTSIDE1
E4
SEGMENT TABLE ENTRY VALID
NO
YES
HL D3
PRNTCR1
F4
LENGTH CODE = 15
NO
YES
HL B3
INVSTE1
HL G1
PAGTABE1 G4
INVALID ADDRESS SPACE
YES
NO
H4
HAS PAGE BEEN USED
YES
NO
J4
PAGE ON PDS
YES
NO
HL F1
NXTENTR1
K4
MAKE VIRTUAL AND REAL ADDRESS PRINTABLE
HL B1


HK K4
B1 PAGE CHANGED
NO
YES
B2 MOVE 'UNCHANGED' TO OUTPUT AREA
CHANGE1
C1 MOVE 'CHANGED' TO OUTPUT AREA
BLKNR1
D1 MAKE BLOCK NUMBER PRINTABLE
E1 PRINT PAGE TABLE ENTRY
HK J4
NXTENTR1
F1 GET NEXT PAGE TABLE ENTRY
G1 ALL ENTRIES TAKEN
NO
YES
HK G4
PAGTABE1
H1 GET NEXT SEGMENT TABLE ENTRY
HK C4
SEGTABE1
HK P4
INVSTE1
B3 PRINT MESSAGE (SEE * B4)
* B4 'INVALID SEGMENT TABLE ENTRY'
C3 SET SWITCH TO SUPPRESS EXECUTION OF PPOOL CODE
* C4 'REAL ADDRESS OUTSIDE STORAGE'
HK D4
OUTSIDE1
B5 PRINT MESSAGE (SEE * C4)
C5 SET SWITCH TO SUPPRESS EXECUTION OF PPOOL CODE
REFERENCES TO HLD3: HKE4, HKK1
PRNTCR1
D3 WAS STORE STATUS EXETUTED
NO
YES
D4 PRINT MESSAGE (SEE * E5)
* E5 'STORE STATUS FUNCTION NOT EXECUTED, CR IN INITIAL STATUS'
PRNTCR
E3 PRINT CONTROL REGISTERS
E4 STORE CONTROL REGISTERS
F3 PPOOL = YES
YES
NO
HM B1
RDPPOOL
G3 FORMAT = YES
YES
NO
HN B1
RDFORMAT
HY B5
ENDIT


HL F3
RDPPOOL B1
READ PPOOL CODE
PPOOLEX1 C1
FORMAT AND PRINT BOUNDARY BOX
D1
VIRTUAL ADDRESS = 0
E1
SEGTAB1 E1
SEGMENT TABLE ENTRY VALID
NO
E5
YES
F1
PAGTAB1 F1
INVALID ADDRESS SPACE
YES
C5
NO
G1
HAS PAGE BEEN USED
NO
YES
H1
PAGE ON PDS
NO
YES
J1
CALCULATE REAL ADDRESS
B3
B3
B3
DETERMINE BLOCK NUMBER AND CONVERT TO PRINTABLE CHARACTERS
B4
'THIS BLOCK WAS OVERWRITTEN BY DUMP PROGRAM'
C3
LAST BLOCK
YES
B5
NO
D3
PRNT2K1 HH
PRINT ONE 2K BLOCK
E3
NXTENT3 E3
GET NEXT PAGE TABLE ENTRY
F3
ALL ENTRIES PROCESSED
NO
F1
YES
G3
GET NEXT SEGMENT TABLE ENTRY
E1
B5
SUPPRT1 B5
PRINT MESSAGE (SEE B4)
C5
NXTENT1 C5
UPDATE VIRTUAL ADDRESS FOR NEXT PAGE
B3
B5
RDNEXTOV E5
FORMAT = YES
YES
NO
HN B1
RDFORMA
HT B5
ENDIT


REFERENCES TO HNB1: HLG3, HNF5
RDFORMAT RDFORMA1
B1 READ FORMATTING CODE
CONTSCAN
C1 SCAN FOR COMMUNICATIONS REGION
*C2 'COULD NOT FIND COMREG BETWEEN 100 AND A00, FORMATTING WILL NOT OCCUR'
D1 COMREG FOUND
YES
NO
FOUNDIT4
D2 SAVE COMREG ADDRESS
STEP4
E1 END OF SCAN
NO
YES
E2 READ NEXT OVERLAY (51)
F1 PRINT ERROR MESSAGE (SEE *C2)
F2 SAVE REGISTERS FOR NEXT OVERLAY
G1 PRINT 'EOJ'
G2 LOAD BASE REG'S AND MAKE ADCONS RELOCATABLE
H1 FLUSH CARDS
PIBSAVDP
H2 SET POINTER TO TEST IF PARTITION IS ACTIVE
J1 EOJ WAIT
J2 GET ADDRESS OF PIB TABLE
B3
B3 POINT TO FIRST PROBLEM PROGRAM PIB
C3
SAVLOOP
C3 PARTITION ACTIVE
NO
YES
D3 GET ADDRESS OF ATTENTION PIB AND SAVE AREA
LTACHCK1
E3 DUMP LTA TEST POINTER
F3 MOVE PARTITION IDENTIFIER TO OUTPUT AREA
*F4 'SAVE AREA IS NOT IN REAL STORAGE'.
G3 PIB ACTIVE
NO
YES
G4 MOVE 'PROG NOT ACTIVE' TO OUTPUT AREA
CHCKSTRS
H3 GET ADDRESS OF SAVE AREA FROM PAGE TABLE
J3 SAVE AREA IN STORAGE
NO
YES
G5
B5
B5
SAVLOOP1
B5 EDIT HV FORMAT SAVE AREA
C5 PRINT HV PRINT SAVE AREA
D5
SAVLOOP4
D5 END OF PIB TABLE
YES
NO
HO B3
LOADOV5
E5 POINT TO NEXT PROBLEM PROGRAM PIB
C3
G5
MOVEMSG
G5 MOVE MESSAGE TO OUTPUT AREA (SEE *F4)
NOTINRS
H5 PRINT HV PRINT MESSAGE
D5

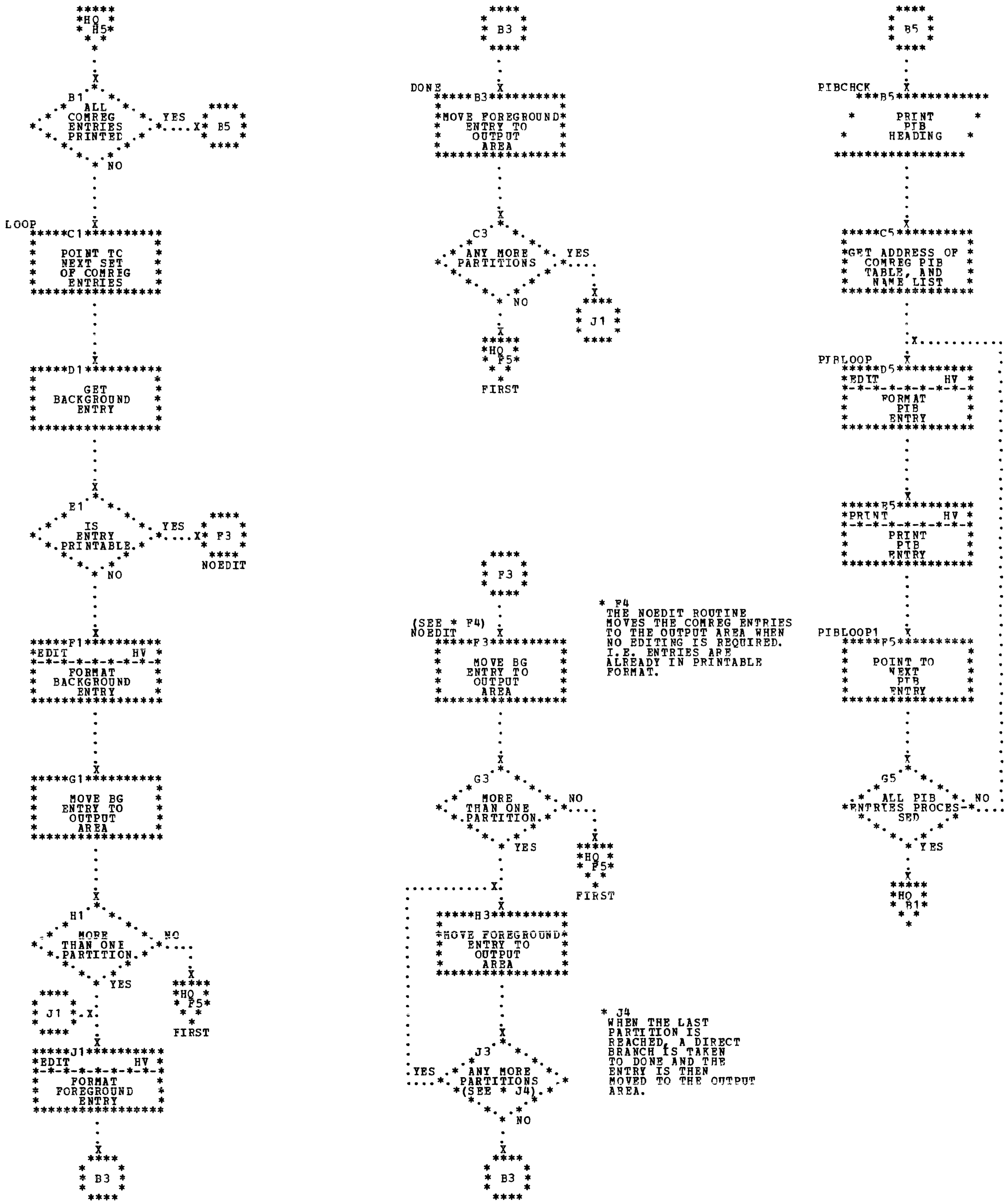
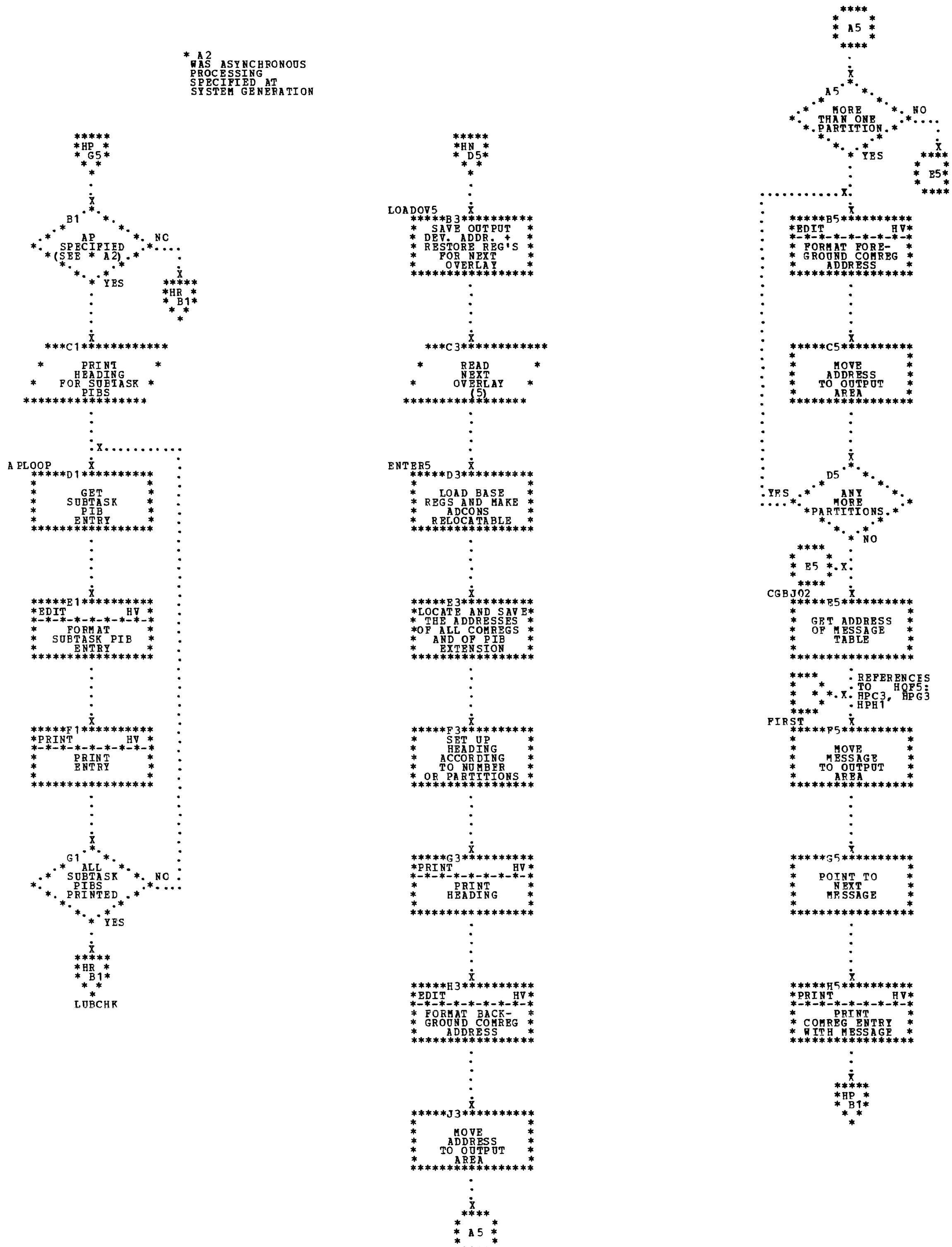
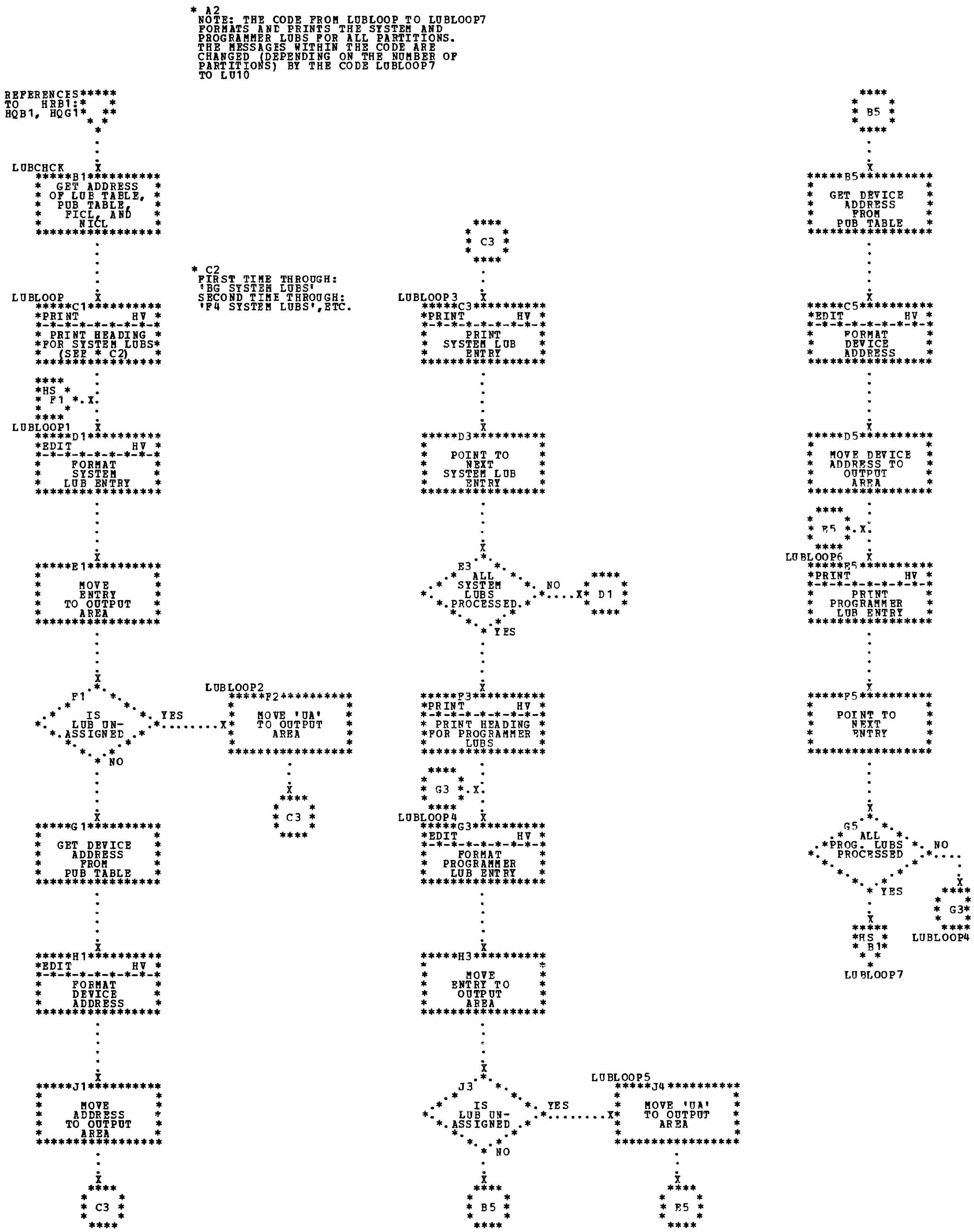
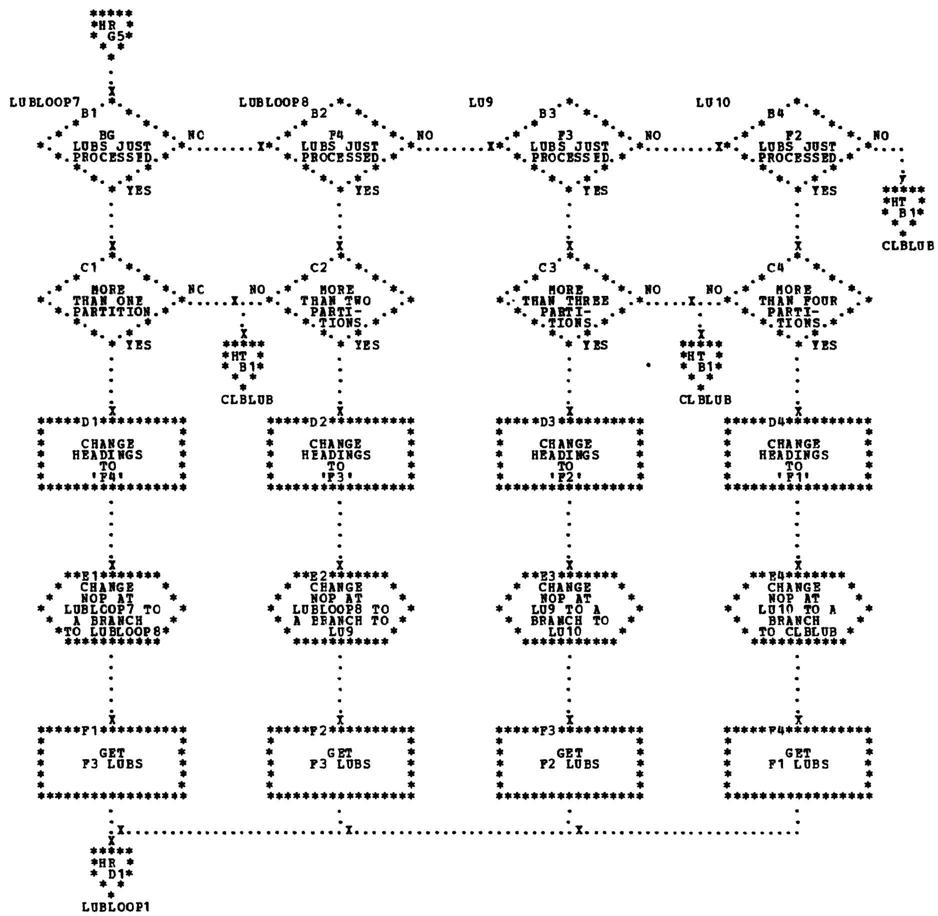

HQ H5
B1 ALL COMREG ENTRIES PRINTED YES B5 NO
LOOP C1 POINT TO NEXT SET OF COMREG ENTRIES
D1 GET BACKGROUND ENTRY
E1 IS ENTRY PRINTABLE YES F3 NOEDIT NO
F1 EDIT HV FORMAT BACKGROUND ENTRY
G1 MOVE BG ENTRY TO OUTPUT AREA
H1 MORE THAN ONE PARTITION NO HQ F5 FIRST YES
J1
J1 EDIT HV FORMAT FOREGROUND ENTRY
B3
B3
DONE B3 MOVE FOREGROUND ENTRY TO OUTPUT AREA
C3 ANY MORE PARTITIONS YES J1 NO
HQ F5 FIRST
F3
(SEE * F4) NOEDIT F3 MOVE BG ENTRY TO OUTPUT AREA
* F4
THE NOEDIT ROUTINE MOVES THE COMREG ENTRIES TO THE OUTPUT AREA WHEN NO EDITING IS REQUIRED. I.E. ENTRIES ARE ALREADY IN PRINTABLE FORMAT.
G3 MORE THAN ONE PARTITION NO HQ F5 FIRST YES
H3 MOVE FOREGROUND ENTRY TO OUTPUT AREA
J3 YES ANY MORE PARTITIONS (SEE * J4) NO
* J4
WHEN THE LAST PARTITION IS REACHED, A DIRECT BRANCH IS TAKEN TO DONE AND THE ENTRY IS THEN MOVED TO THE OUTPUT AREA.
B3
B5
PIBCHCK B5 PRINT PIB HEADING
C5 GET ADDRESS OF COMREG PIB TABLE, AND NAME LIST
PIBLOOP D5 EDIT HV FORMAT PIB ENTRY
E5 PRINT HV PRINT PIB ENTRY
PIBLOOP1 F5 POINT TO NEXT PIB ENTRY
G5 ALL PIB ENTRIES PROCESSED NO YES
HQ B1

Chart HQ. REALDUMP (Part 10 of 18)





* A2
NOTE: THE CODE FROM LUBLOOP TO LUBLOOP7 FORMATS AND PRINTS THE SYSTEM AND PROGRAMMER LUBS FOR ALL PARTITIONS. THE MESSAGES WITHIN THE CODE ARE CHANGED (DEPENDING ON THE NUMBER OF PARTITIONS) BY THE CODE LUBLOOP7 TO LU10
REFERENCES TO HRB1: HQB1, HQG1
LUBCHCK
B1
GET ADDRESS OF LUB TABLE, PUB TABLE, FICL, AND NICL
* C2
FIRST TIME THROUGH: 'BG SYSTEM LUBS' SECOND TIME THROUGH: 'F4 SYSTEM LUBS',ETC.
LUBLOOP
C1
PRINT HV
PRINT HEADING FOR SYSTEM LUBS (SEE * C2)
HS F1
LUBLOOP1
D1
EDIT HV
FORMAT SYSTEM LUB ENTRY
E1
MOVE ENTRY TO OUTPUT AREA
F1
IS LUB UN-ASSIGNED
YES
NO
LUBLOOP2
F2
MOVE 'UA' TO OUTPUT AREA
C3
G1
GET DEVICE ADDRESS FROM PUB TABLE
H1
EDIT HV
FORMAT DEVICE ADDRESS
J1
MOVE ADDRESS TO OUTPUT AREA
C3
C3
LUBLOOP3
C3
PRINT HV
PRINT SYSTEM LUB ENTRY
D3
POINT TO NEXT SYSTEM LUB ENTRY
E3
ALL SYSTEM LUBS PROCESSED
NO
D1
YES
F3
PRINT HV
PRINT HEADING FOR PROGRAMMER LUBS
G3
LUBLOOP4
G3
EDIT HV
FORMAT PROGRAMMER LUB ENTRY
H3
MOVE ENTRY TO OUTPUT AREA
J3
IS LUB UN-ASSIGNED
YES
NO
LUBLOOP5
J4
MOVE 'UA' TO OUTPUT AREA
B5
E5
B5
B5
GET DEVICE ADDRESS FROM PUB TABLE
C5
EDIT HV
FORMAT DEVICE ADDRESS
D5
MOVE DEVICE ADDRESS TO OUTPUT AREA
E5
LUBLOOP6
E5
PRINT HV
PRINT PROGRAMMER LUB ENTRY
F5
POINT TO NEXT ENTRY
G5
ALL PROG. LUBS PROCESSED
NO
G3
LUBLOOP4
YES
HS B1
LUBLOOP7



* NOTE
THE DECISIONS AT B1, B2, B3, AND B4 CONSIST OF A NOP AS THE FIRST INSTRUCTION IN EACH BLOCK OF CODE. WHEN THE CODE IS EXECUTED, THE NOP IS CHANGED TO A BRANCH TO THE NEXT SET OF CODE TO BE EXECUTED, THAT IS, THE CODE FOR THE NEXT PARTITION.

REFERENCES TO HTB1; HSB4, HSC2 HSC4

CLBLUB B1 — PRINT HV — PRINT HEADING 'PCIL LUBS'

C1 — GET ADDRESS OF BG PRIVATE -CORE- IMAGE LIBRARY LUB

D1 — SET PARTITION COUNTER

E1

LISTCLB E1 — EDIT HV — FORMAT LUB ENTRY

F1 — MOVE LUB ENTRY AND 'UA' TO OUTPUT AREA

G1 — IS LUB UN-ASSIGNED — YES — C3 — NO

H1 — GET PHYSICAL UNIT ADDRESS (CUU) FROM PUB TABLE

J1 — EDIT HV — FORMAT PHYSICAL UNIT ADDRESS

B3

B3

B3 — MOVE ADDRESS TO OUTPUT AREA

C3

NEXTCLB C3 — PRINT HV — PRINT LUB

D3 — POINT TO NEXT LUB

E3 — ALL LUBS PROCESSED — NO — E1 — YES

PUBCHCK F3 — PRINT HV — PRINT HEADINGS FOR PUB TABLE

G3 — GET ADDRESS OF SYSTEM COMREG AND PUB OWNERSHIP TABLE

PUBLOOP H3 — EDIT HV — FORMAT PUB ENTRY

PUBLOOP2 J3 — PRINT HV — PRINT PUB ENTRY AND PUB OWNERSHIP INFO

K3 — ALL PUBS PROCESSED — NO — YES — B5

B5

B5 — PRINT HV — PRINT HEADING 'ERROR RECOVERY BLOCK'

C5 — GET ADDRESS OF ERROR RECOVERY BLOCK AND MESSAGE TABLE

D5 — PRINT HV — PRINT PHASE NAME AND MESSA-GE:'FETCH NAME'

ERBLOOP E5 — GET NEXT ENTRY AND NEXT MESSAGE

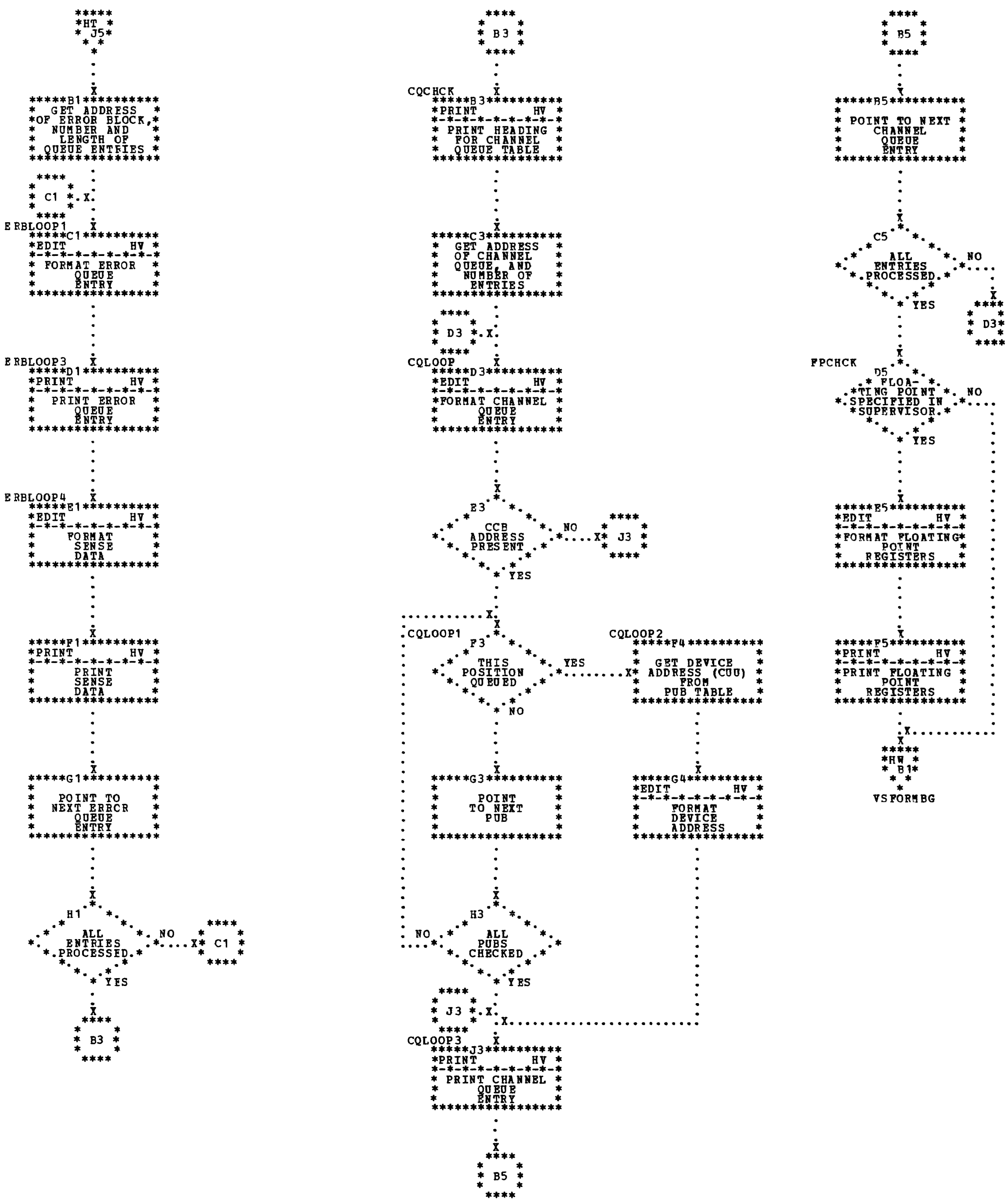
F5 — EDIT HV — FORMAT ENTRY

G5 — PRINT HV — PRINT ENTRY WITH MESSAGE

H5 — ALL ENTRIES PROCESSED — NO — YES

J5 — PRINT HV — PRINT HEADING FOR ERROR QUEUE ENTRIES

HU B1


HT J5
B1
GET ADDRESS OF ERROR BLOCK, NUMBER AND LENGTH OF QUEUE ENTRIES
C1
ERBLOOP1
C1
EDIT HV
FORMAT ERROR QUEUE ENTRY
ERBLOOP3
D1
PRINT HV
PRINT ERROR QUEUE ENTRY
ERBLOOP4
E1
EDIT HV
FORMAT SENSE DATA
F1
PRINT HV
PRINT SENSE DATA
G1
POINT TO NEXT ERROR QUEUE ENTRY
H1
ALL ENTRIES PROCESSED
NO
C1
YES
B3
B3
CQCHCK
B3
PRINT HV
PRINT HEADING FOR CHANNEL QUEUE TABLE
C3
GET ADDRESS OF CHANNEL QUEUE, AND NUMBER OF ENTRIES
D3
CQLOOP
D3
EDIT HV
FORMAT CHANNEL QUEUE ENTRY
E3
CCB ADDRESS PRESENT
NO
J3
YES
CQLOOP1
F3
THIS POSITION QUEUED
YES
NO
CQLOOP2
F4
GET DEVICE ADDRESS (CUU) FROM PUB TABLE
G3
POINT TO NEXT PUB
G4
EDIT HV
FORMAT DEVICE ADDRESS
H3
ALL PUBS CHECKED
NO
YES
J3
CQLOOP3
J3
PRINT HV
PRINT CHANNEL QUEUE ENTRY
B5
B5
B5
POINT TO NEXT CHANNEL QUEUE ENTRY
C5
ALL ENTRIES PROCESSED
NO
D3
YES
FPCHCK
D5
FLOATING POINT SPECIFIED IN SUPERVISOR
NO
YES
E5
EDIT HV
FORMAT FLOATING POINT REGISTERS
F5
PRINT HV
PRINT FLOATING POINT REGISTERS
HW B1
VSFORMBG

```
        ****A1*********
       *               *
       *     PRINT     *
       *               *
        ***************
               :
               :
PRINT          X
       *****B1**********
       *               *
       *  GET ADDRESS  *
       *    OF CCW     *
       *   AND STORE   *
       *   IT IN CSW   *
       *****************
               :
               :
               X
         ***C1*************
        *     PRINT      *
        *      ONE       *
       *       LINE     *
      *****************
               :
               :
               X
       *****D1**********
       *               *
       *     CLEAR     *
       *    OUTPUT     *
       *     AREA      *
       *               *
       *****************
               :
               :
               X
        ****E1*********
       *    RETURN     *
       *   TO CALLER   *
       *               *
        ***************


        ****F1*********
       *               *
       *     EDIT      *
       *               *
        ***************
               :
               :
EDIT           X
       *****G1**********
       *               *
       *    UNPACK     *
       *      AND      *
       *   TRANSLATE   *
       *     ENTRY     *
       *****************
               :
               :
               X
        ****H1*********
       *    RETURN     *
       *   TO CALLER   *
       *               *
        ***************
```

HU F5 → VSFORMBG B1: GET ADDRESS OF CCB COPY BLOCK QUEUE

B2 MESSAGE: 'CCB, CCW, IDAL AND FIXINF TABLES NOT AVAILABLE'

VSCCB C1: IS CCB QUEUE EMPTY? YES → VSCCB4 C2: PRINT MESSAGE (SEE * B2) → HX E5 VSBBOX; NO →

D1: PRINT HEADING: 'COPIED AND TRANSLATED CCB'

HX D5 → VSCCB1 E1: FORMAT CCB QUEUE ADDRESS

F1: PRINT ADDRESSES WITH TEXT (SEE *F2)

F2 TEXT: 'ADDR.OF CCB COPY BLOCK QUEUE'

G1: FORMAT ENTIRE CCB QUEUE

H1 → VSCCB2 H1: MOVE ONE QUEUE ENTRY WITH TEXT TO OUTPUT AREA

J1: PRINT QUEUE ENTRY → B3

B3 → B3: POINT TO NEXT QUEUE ENTRY AND NEXT TEXT

C3: ALL ENTRIES PRINTED? NO → H1; YES →

D3 → VSCCB3 D3: GET ADDRESS OF FIXINF EXTENSION

E3: ANY FIXINF AVAILABLE? NO → J3; YES →

F3: MAKE ENTIRE QUEUE PRINTABLE

G3: PRINT QUEUE

H3: ANY MORE FIXINF EXTENSIONS? YES → D3; NO →

J3 → VSCCW J3: GET ADDRESS OF CCB COPY BLOCK QUEUE → B5

B4 HEADINGS FOR COPIED AND TRANSLATED CHANNEL PROGRAM

B5 → VSCCW1 B5: PRINT HEADINGS (SEE * B4)

HX E1 → VSCCW2 C5: FORMAT VIRTUAL ADDRESS OF CCW

D5: FORMAT CCW QUEUE ADDRESS

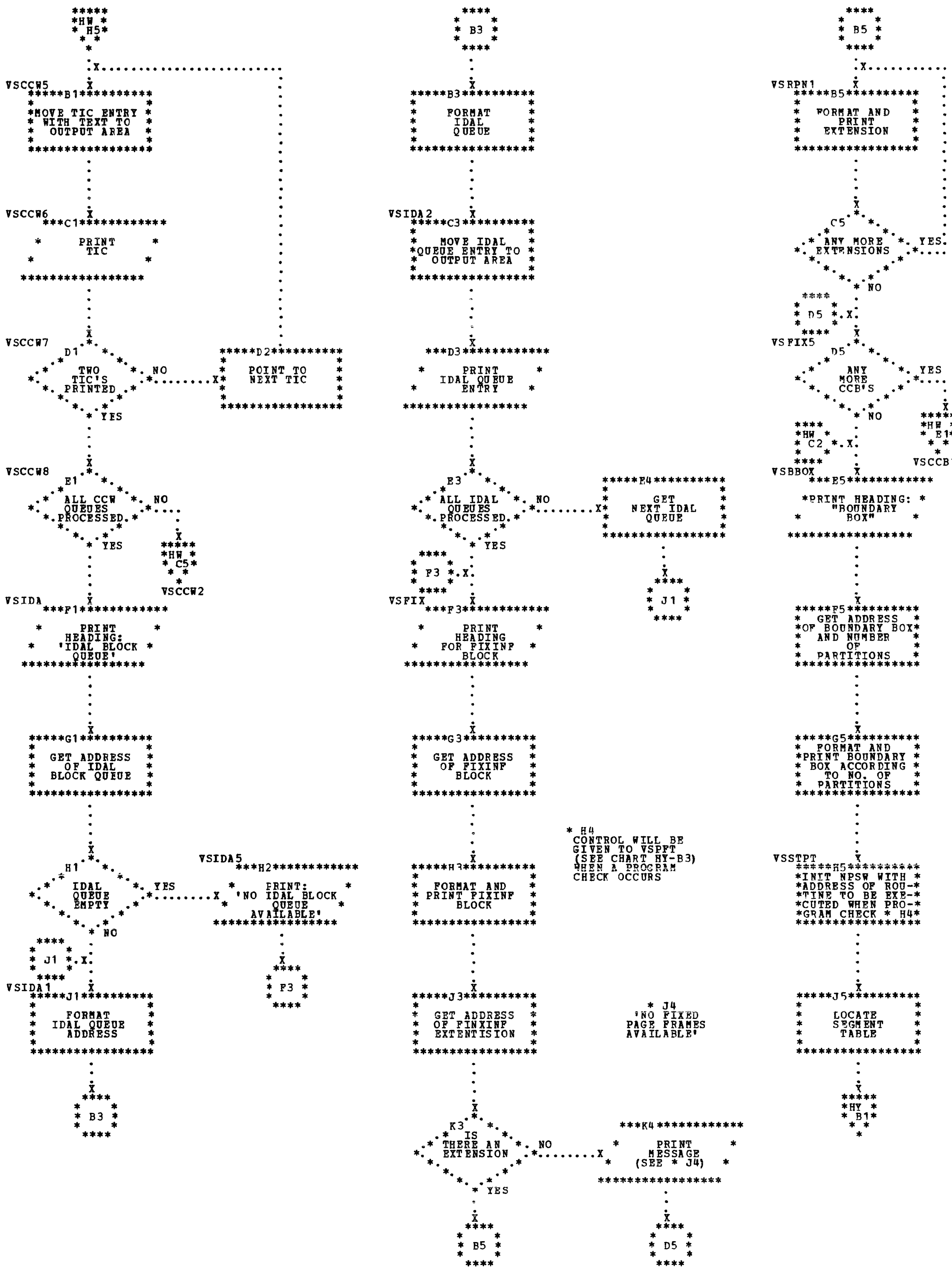
E5: FORMAT CCW QUEUE

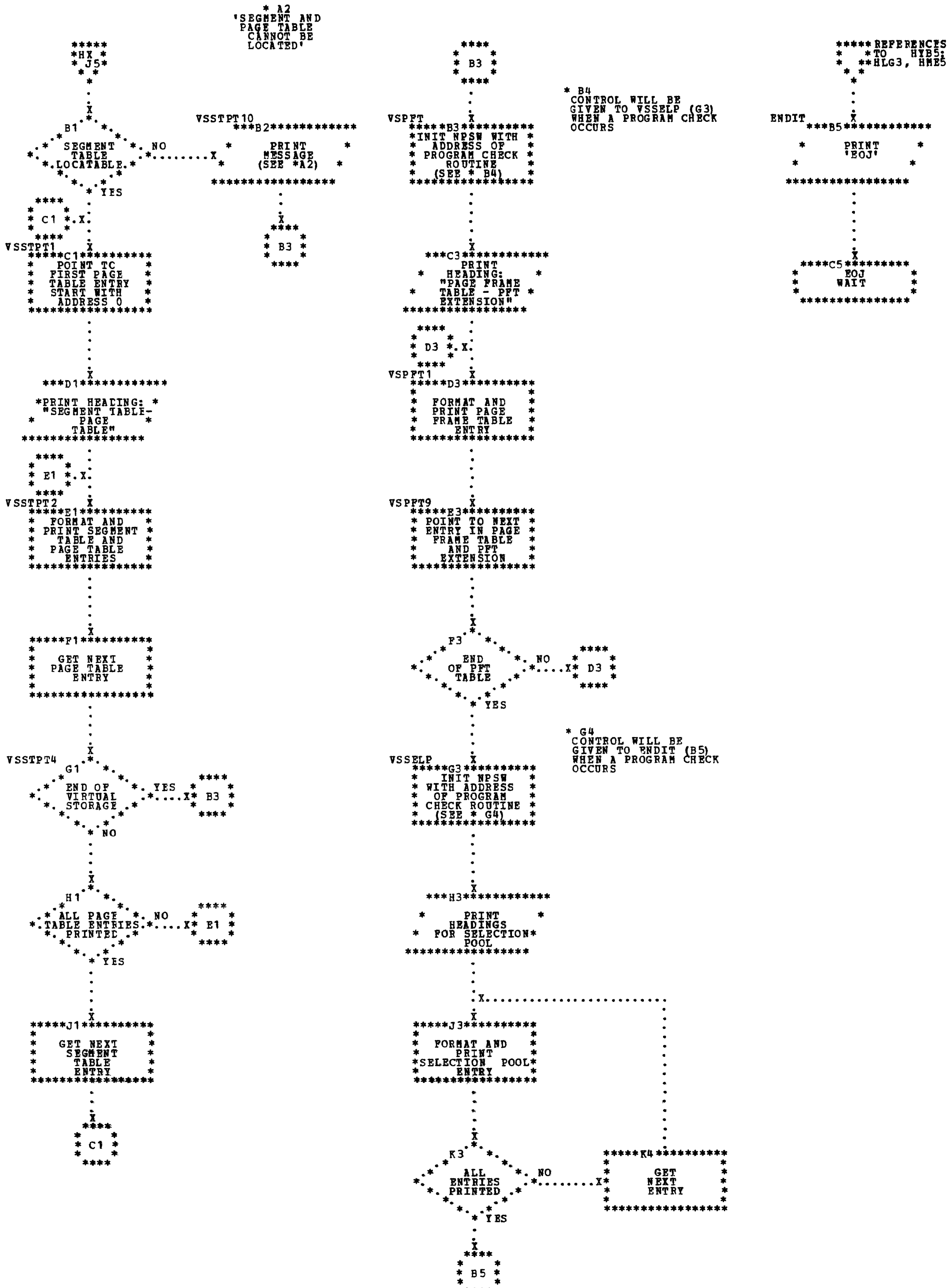
F5 → VSCCW3 F5: MOVE CCW WITH TEXT TO OUTPUT AREA

VSCCW4 G5: PRINT CCW

H5: ALL CCW'S PRINTED? YES → HX B1 VSCCW5; NO →

J5: POINT TO NEXT CCW → F5


HW H5
VSCCW5
B1
MOVE TIC ENTRY WITH TEXT TO OUTPUT AREA
VSCCW6
C1
PRINT TIC
VSCCW7
D1
TWO TIC'S PRINTED
NO
YES
D2
POINT TO NEXT TIC
VSCCW8
E1
ALL CCW QUEUES PROCESSED
NO
YES
HW C5
VSCCW2
VSIDA
F1
PRINT HEADING: 'IDAL BLOCK QUEUE'
G1
GET ADDRESS OF IDAL BLOCK QUEUE
H1
IDAL QUEUE EMPTY
YES
NO
VSIDA5
H2
PRINT: 'NO IDAL BLOCK QUEUE AVAILABLE'
F3
J1
VSIDA1
J1
FORMAT IDAL QUEUE ADDRESS
B3
B3
B3
FORMAT IDAL QUEUE
VSIDA2
C3
MOVE IDAL QUEUE ENTRY TO OUTPUT AREA
D3
PRINT IDAL QUEUE ENTRY
E3
ALL IDAL QUEUES PROCESSED
NO
YES
E4
GET NEXT IDAL QUEUE
J1
F3
VSFIX
F3
PRINT HEADING FOR FIXINF BLOCK
G3
GET ADDRESS OF FIXINF BLOCK
H3
FORMAT AND PRINT FIXINF BLOCK
* H4
CONTROL WILL BE GIVEN TO VSPPT (SEE CHART HY-B3) WHEN A PROGRAM CHECK OCCURS
J3
GET ADDRESS OF FINXINF EXTENTISION
* J4
'NO FIXED PAGE FRAMES AVAILABLE'
K3
IS THERE AN EXTENSION
NO
YES
K4
PRINT MESSAGE (SEE * J4)
B5
D5
B5
VSRPN1
B5
FORMAT AND PRINT EXTENSION
C5
ANY MORE EXTENSIONS
YES
NO
D5
VSFIX5
D5
ANY MORE CCB'S
YES
NO
HW E1
VSCCB1
HW C2
VSBBOX
E5
PRINT HEADING: "BOUNDARY BOX"
F5
GET ADDRESS OF BOUNDARY BOX AND NUMBER OF PARTITIONS
G5
FORMAT AND PRINT BOUNDARY BOX ACCORDING TO NO. OF PARTITIONS
VSSTPT
H5
INIT NPSW WITH ADDRESS OF ROUTINE TO BE EXECUTED WHEN PROGRAM CHECK * H4
J5
LOCATE SEGMENT TABLE
HY B1


HY J5
* A2 'SEGMENT AND PAGE TABLE CANNOT BE LOCATED'
B3
* B4 CONTROL WILL BE GIVEN TO VSSELP (G3) WHEN A PROGRAM CHECK OCCURS
REFERENCES TO HYB5: HLG3, HME5
B1 SEGMENT TABLE LOCATABLE
NO
YES
VSSTPT10 B2 PRINT MESSAGE (SEE *A2)
VSPFT B3 INIT NPSW WITH ADDRESS OF PROGRAM CHECK ROUTINE (SEE * B4)
ENDIT B5 PRINT 'EOJ'
C1
VSSTPT1 C1 POINT TO FIRST PAGE TABLE ENTRY START WITH ADDRESS 0
B3
C3 PRINT HEADING: "PAGE FRAME TABLE - PFT EXTENSION"
C5 EOJ WAIT
D1 PRINT HEADING: "SEGMENT TABLE-PAGE TABLE"
D3
VSPFT1 D3 FORMAT AND PRINT PAGE FRAME TABLE ENTRY
E1
VSSTPT2 E1 FORMAT AND PRINT SEGMENT TABLE AND PAGE TABLE ENTRIES
VSPFT9 E3 POINT TO NEXT ENTRY IN PAGE FRAME TABLE AND PFT EXTENSION
F1 GET NEXT PAGE TABLE ENTRY
F3 END OF PFT TABLE
NO
D3
YES
* G4 CONTROL WILL BE GIVEN TO ENDIT (B5) WHEN A PROGRAM CHECK OCCURS
VSSTPT4 G1 END OF VIRTUAL STORAGE
YES
B3
NO
VSSELP G3 INIT NPSW WITH ADDRESS OF PROGRAM CHECK ROUTINE (SEE * G4)
H1 ALL PAGE TABLE ENTRIES PRINTED
NO
E1
YES
H3 PRINT HEADINGS FOR SELECTION POOL
J1 GET NEXT SEGMENT TABLE ENTRY
J3 FORMAT AND PRINT SELECTION POOL ENTRY
C1
K3 ALL ENTRIES PRINTED
NO
K4 GET NEXT ENTRY
YES
B5

FUNCTION

PDSDM is a program that dumps the page data set (SYSVIS). Depending on the parameters submitted on SYSLOG or SYSIPT, PDSDM dumps the contents of

- The entire page data set
- A specified virtual partition
- One or more pages of the page data set.

The dump is from

- SYSVIS to SYSLST or
- SYSVIS to SYS001 or
- SYS001 to SYSLST.

The page data set (SYSVIS) is always on disk, and is accessed by assigning SYS000 to the disk drive containing SYSVIS.

SYS001 is used to temporarily store the page data set dump. It must be assigned to disk or tape.

SYSLST may be assigned to a printer, a tape, or a disk drive.

The first parameter given to PDSDM specifies the logical unit (SYSLOG or SYSIPT) from which further parameters will be read. These parameters may be:

- Parameters that specify the logical units from and to which the selected part(s) are to be dumped.
- Parameters that specify the part(s) of the page data set to be dumped.

The output format is described under the heading 'Dump Page Data Set'.

This program is initiated by the job control statement // EXEC PDSDM. Refer to Section 2-C-6 of Serviceability Aids and Debugging Procedures, GC33-5380 for the proper label information.

FLOW OF CONTROL

The flow of control in PDSDM is as follows:

1. Initialize PDSDM
2. Accept parameters (Part 1), specifying dump input and output devices
3. Open files
4. Accept parameters (Part 2), specifying the part(s) of SYSVIS to be dumped
5. Dump page data set
6. Close files.

INITIALIZE PDSDM (CHART JA)

This part

- Loads the base registers
- Saves the contents of the boundary box, the number of partition supported, and the end address of real storage in the field F1STREC
- Opens SYSLOG file.

ACCEPT PARAMETERS - PART 1 (CHARTS JA AND JB), SPECIFYING DUMP INPUT AND OUTPUT DEVICES

This routine checks the parameters that specify the logical units from and to which the selected parts are to be dumped.

Input	Action Taken
IPT	Determines SYSIPT as input device for further parameters.
LOG	Determines SYSLOG as input device for further parameters.
Press END or ENTER on SYSLOG after LOG, or enter /* or /& on SYSIPT after // EXEC PDSDM	Dumps entire contents of page data set on SYS001 and terminates job.
TO=SYSLST	Selects SYSVIS as input and SYSLST as output device for dump.
TO=SYS001	Selects SYSVIS as input and SYS001 as output device for dump.

TO=SYSLST,T	Selects SYS001 as input and SYSLST as output device for dump.

The logical units selected for input and output are indicated in the field SWITCH.

In case of error, the incorrect parameter is marked with * and a corresponding message is issued on SYSLOG. The invalid parameter may be corrected via SYSLOG.

OPEN FILES (CHARTS JB AND JJ)

The first part checks which input and output devices are assigned for the dump, tests for correct device assignments, and opens the corresponding files if the assignments are correct.

Selected Output Device	Action Taken
SYSLST	Opens SYSLST file.
SYS001	Tests if SYS001 is assigned to disk or tape. If yes, opens SYS001 file.

Selected Input Device	Action Taken
SYSVIS	Tests if SYS000 is assigned to disk. If yes, opens SYS000.
SYS001	Tests if SYS001 is assigned to disk or tape. If yes, opens SYS001.

In case of error, a corresponding message is issued on SYSLOG.

The second part tests if the dump is from

- SYSVIS to SYS001 or
- SYS001 to SYSLST or
- SYSVIS to SYSLST

and takes the following action:

Dump From	Action Taken
SYSVIS to SYS001	Writes the following information as first record on SYS001: internal file ID, number of partitions supported, end address of real storage, and contents of the boundary box.
SYS001 to SYSLST	Reads first record from SYS001 with above described contents and saves it in the field F1STREC.
SYSVIS to SYSLST	None.

ACCEPT PARAMETERS - PART 2 (CHARTS JC, JD, AND JG) SPECIFYING THE PART(S) OF SYSVIS TO BE DUMPED

This routine tests the parameters that specify the part(s) of the page data set to be dumped. The parameters tested and the corresponding actions taken are:

Input	Action Taken
(address,address)	Checks whether start and end addresses of selected pages are valid, translates them into record numbers and saves them in the table PARAM.
partition ID (F1,F2,F3,F4, or BG)	Checks for partition(s) specified, gets corresponding start and end addresses, translates them into record numbers, and saves them in the table PARAM.
address	Checks if address is valid, translates it into two equal record numbers and saves them in the table PARAM.
EOJ on SYSLOG	Terminates job.
IPT on SYSLOG	Switches from SYSLOG TO SYSIPT to read further parameters.
Press END or ENTER on SYSLOG	If this key is pressed after the message GIVE PARAMETERS has been issued, the contents of the entire page data set are dumped to the output device selected previously.

/* or /& indicates the end of input to PDSDM, if read from SYSIPT. If this card is encountered immediately after the card containing TO=, the entire page data set is dumped to the output device specified, and the job is terminated.

When all parameters have been checked and processed, control is passed to the dump page data set routine PRINT. After completion of the dump, control is returned for further parameter processing.

In case of error, the invalid parameter is marked with *, and a corresponding message is issued on SYSLOG. Reading of input is switched to SYSLOG, and new parameters may be entered, or the correct parameters up to the invalid one may be processed.

DUMP PAGE DATA SET (CHARTS JE, JF, AND JH)

This routine dumps the part(s) selected from the page data set to the output device specified.

The format of the dump depends on the output device specified:

Output Device	Dump Format
SYS001	Blocks of 2K.
SYSLST	Each block has a block number, starting with 0 from the first page of the page data set. Printed on SYSLST, each line contains 32 bytes of the page data set in hexadecimal format, followed by the translation in printable characters. Identical words and lines are printed only once followed by the word '--SAME--', and blocks containing only X'00' are not printed. Instead, the message 'BLOCK x (begin address - end address) CONTAINS ZEROS' is issued, or, if more than one block contains zeros, the message 'BLOCKS x-y (begin address - end address) CONTAIN ZEROS' is issued.

When all records indicated in the table PARAM have been processed, control is returned to the parameter acceptance routine.

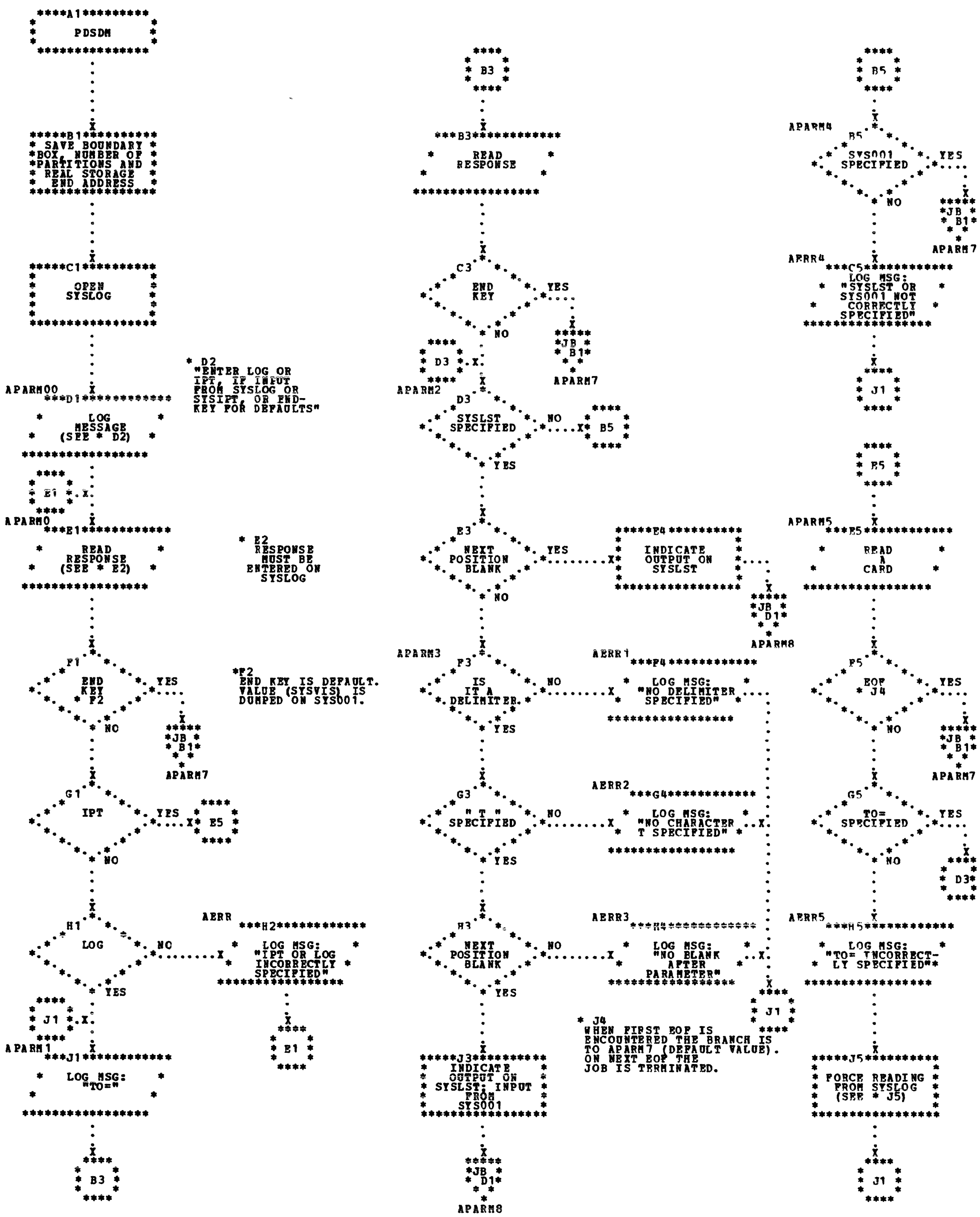
CLOSE FILES (CHART JJ)

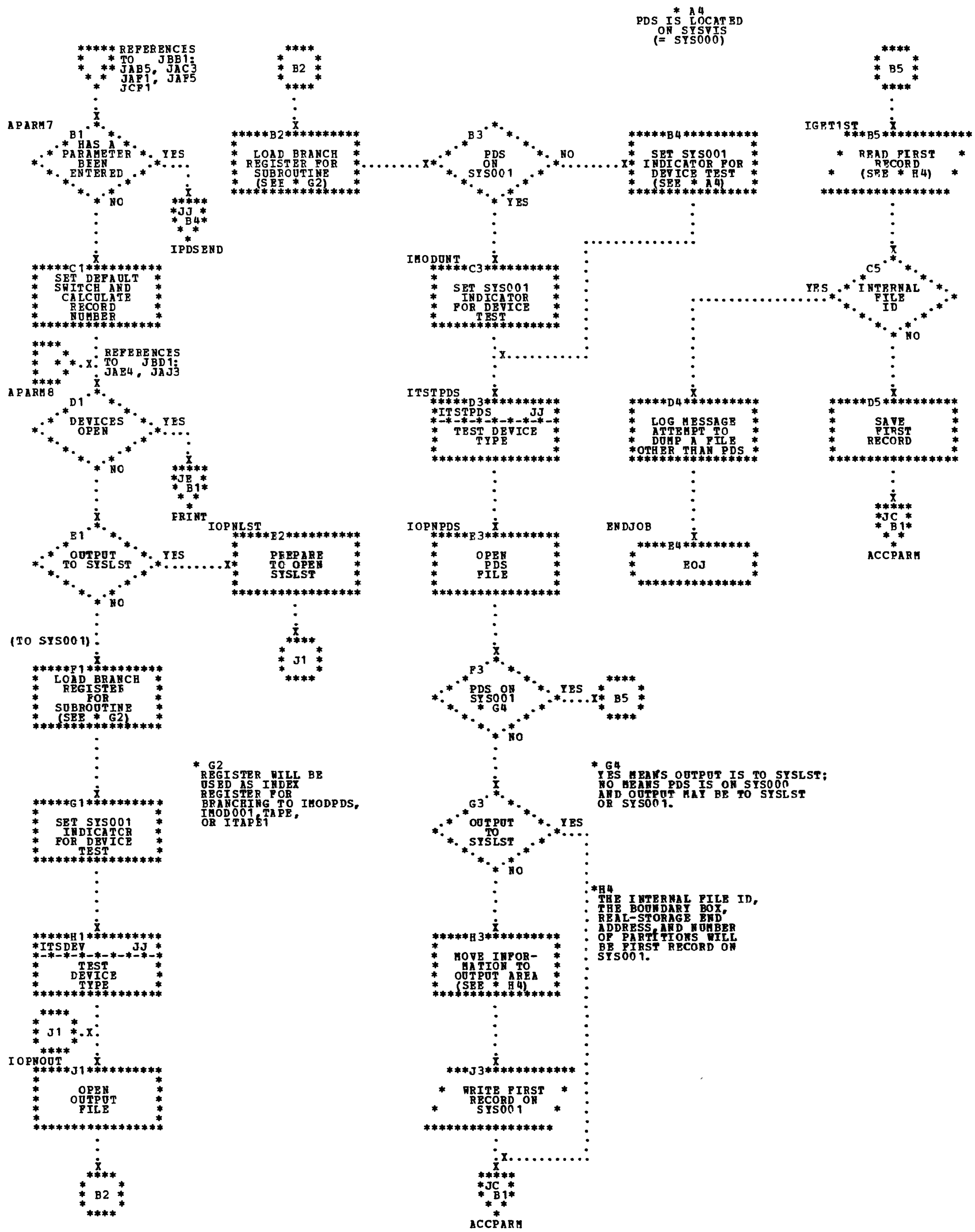
This part closes

- The input and output files used for the page data set dump
- The SYSLOG file
- The SYSIPT file, if it was opened.

In addition, any cards remaining in the reader are flushed. Finally, the program PDSDM is terminated.

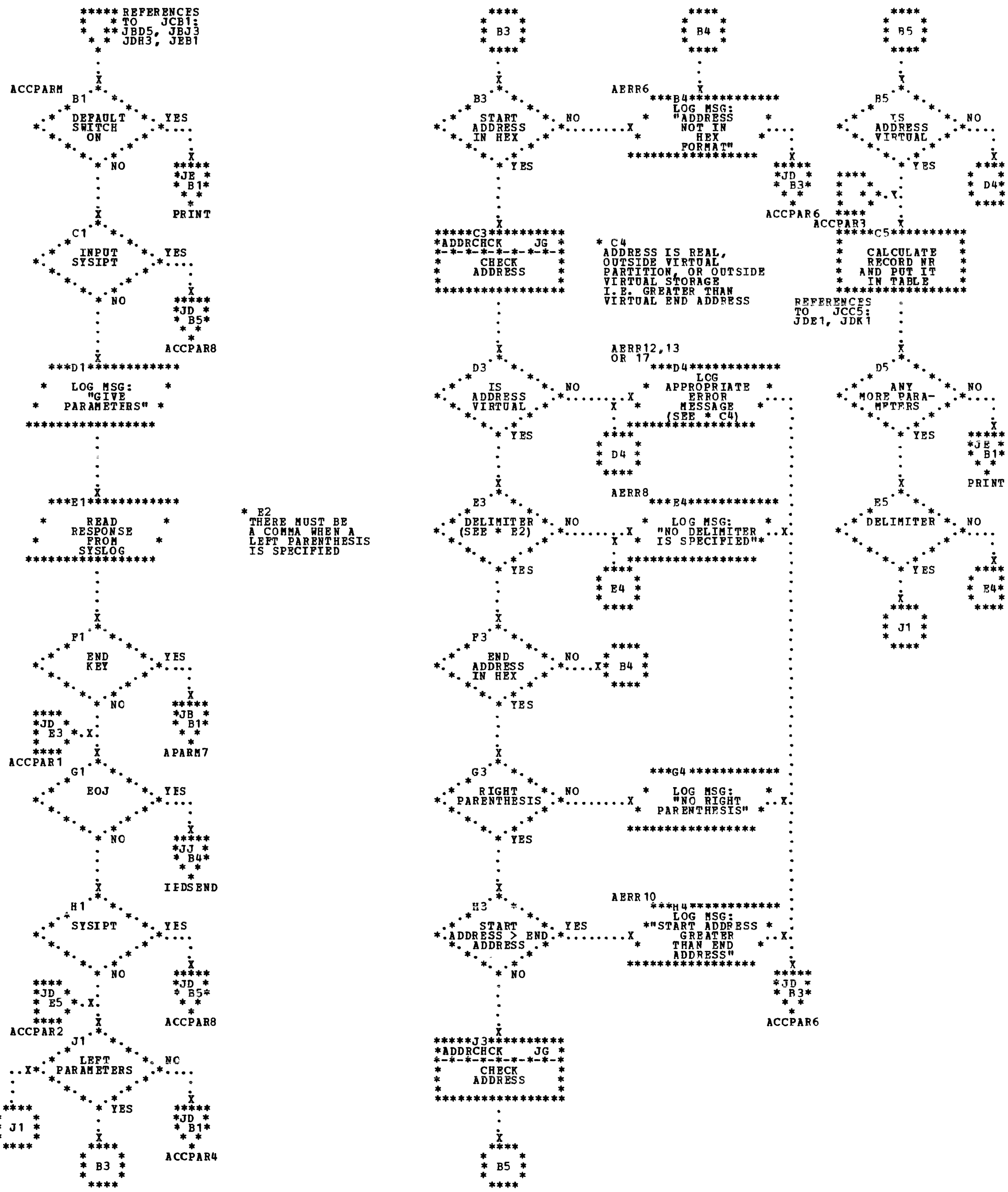
Chart JA. PDSDM - Parameter Acceptance (Part 1 of 4)

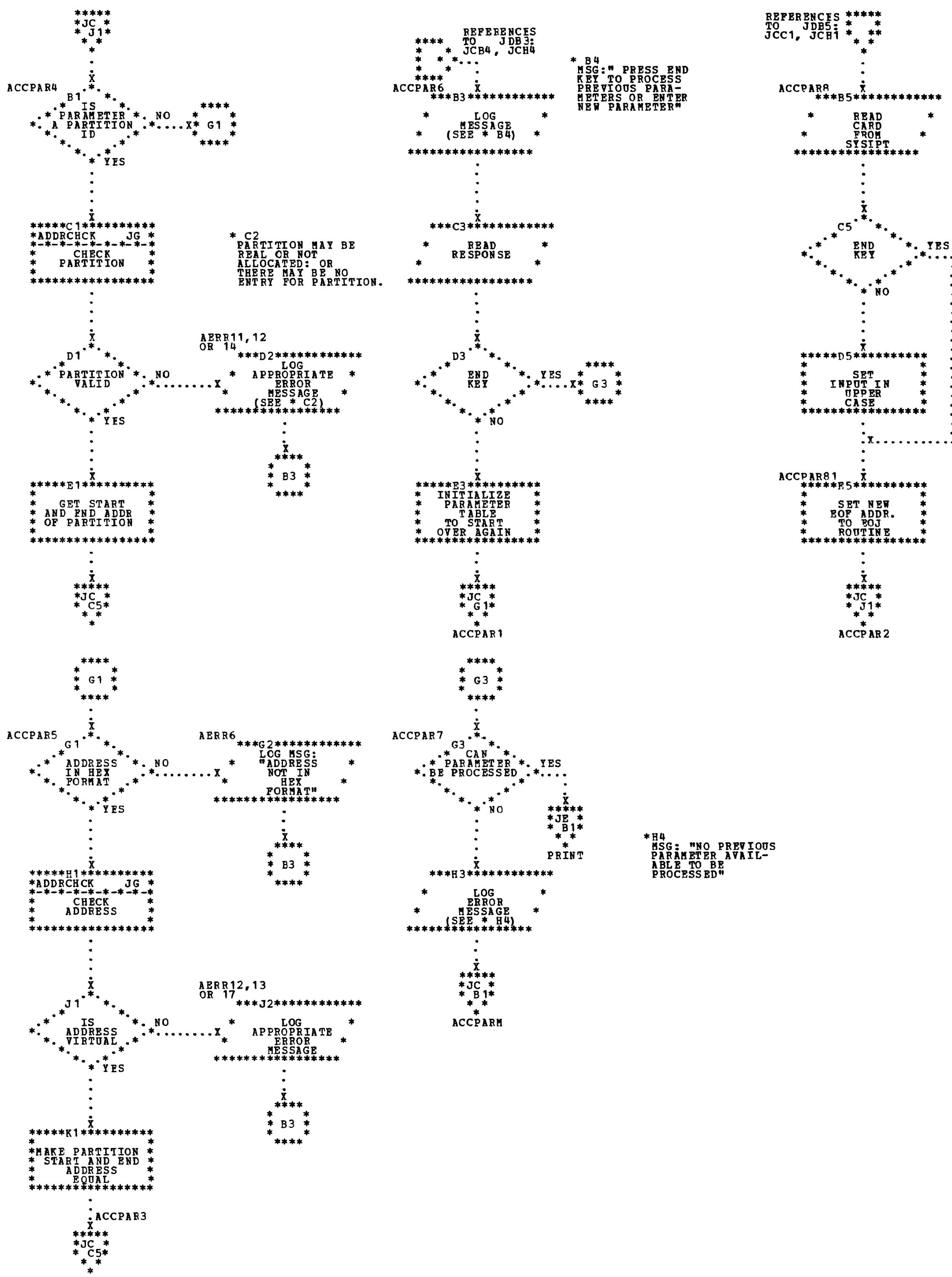


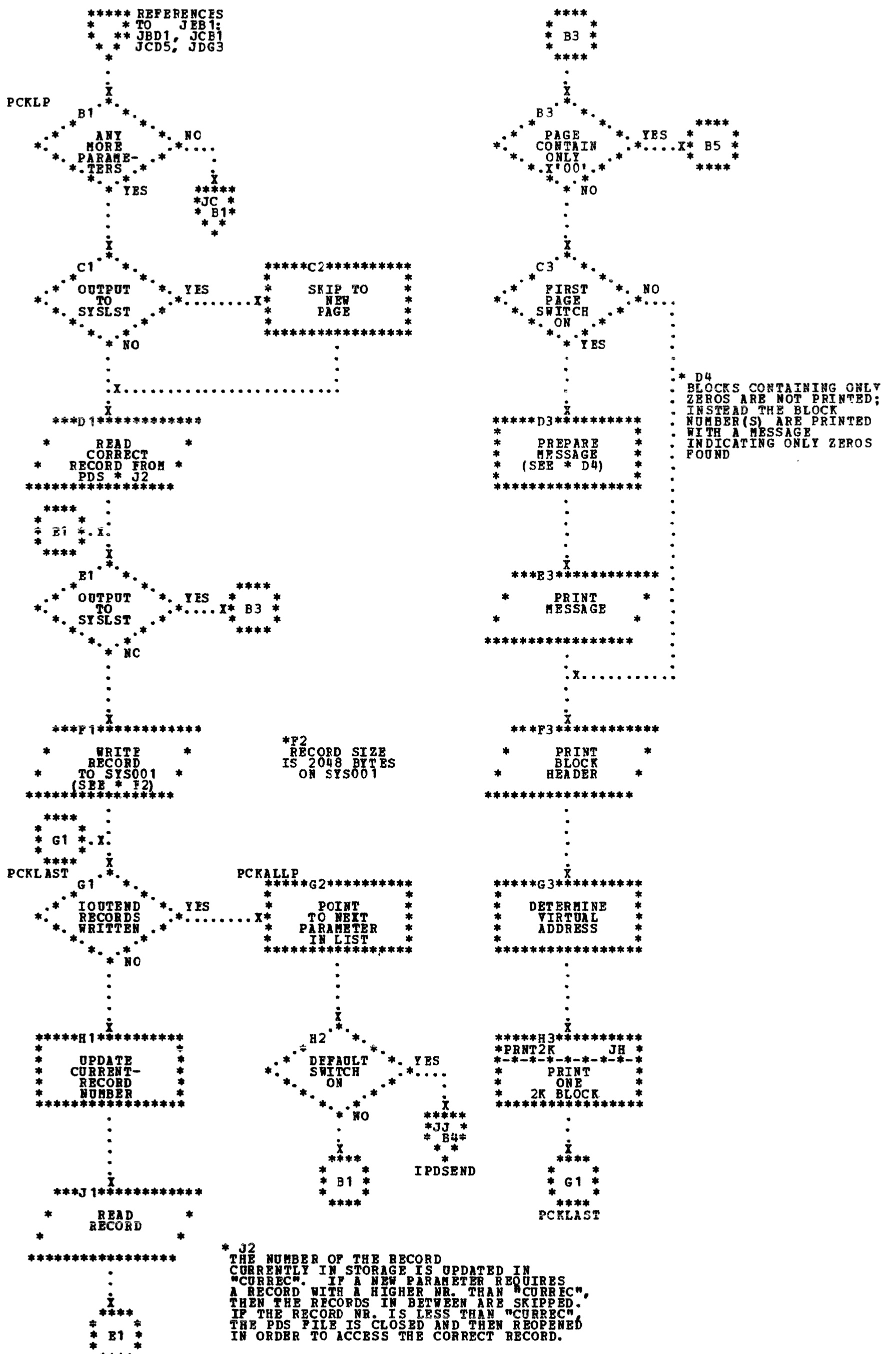


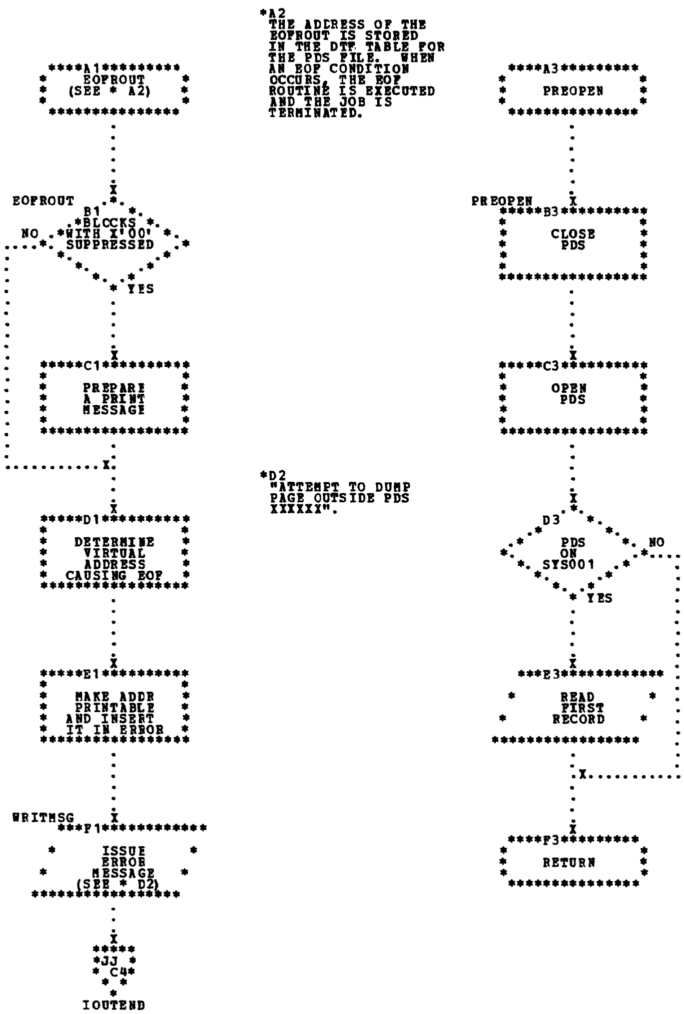
REFERENCES TO JBB1: JAB5, JAC3 JAF1, JAF5 JCF1
APARM7
B1 HAS A PARAMETER BEEN ENTERED
YES
NO
JJ B4
IPDSEND
C1 SET DEFAULT SWITCH AND CALCULATE RECORD NUMBER
REFERENCES TO JBD1: JAE4, JAJ3
APARM8
D1 DEVICES OPEN
YES
NO
JE B1
PRINT
E1 OUTPUT TO SYSLST
YES
NO
IOPNLST
E2 PREPARE TO OPEN SYSLST
J1
(TO SYS001)
F1 LOAD BRANCH REGISTER FOR SUBROUTINE (SEE * G2)
G1 SET SYS001 INDICATOR FOR DEVICE TEST
H1 ITSDEV JJ TEST DEVICE TYPE
J1
IOPNOUT
J1 OPEN OUTPUT FILE
B2
B2 LOAD BRANCH REGISTER FOR SUBROUTINE (SEE * G2)
* G2 REGISTER WILL BE USED AS INDEX REGISTER FOR BRANCHING TO IMODPDS, IMOD001, TAPE, OR ITAPE1
B3 PDS ON SYS001
NO
YES
IMODUNT
C3 SET SYS001 INDICATOR FOR DEVICE TEST
ITSTPDS
D3 ITSTPDS JJ TEST DEVICE TYPE
IOPNPDS
E3 OPEN PDS FILE
F3 PDS ON SYS001 * G4
YES
B5
NO
* G4 YES MEANS OUTPUT IS TO SYSLST; NO MEANS PDS IS ON SYS000 AND OUTPUT MAY BE TO SYSLST OR SYS001.
G3 OUTPUT TO SYSLST
YES
NO
H3 MOVE INFORMATION TO OUTPUT AREA (SEE * H4)
*H4 THE INTERNAL FILE ID, THE BOUNDARY BOX, REAL-STORAGE END ADDRESS, AND NUMBER OF PARTITIONS WILL BE FIRST RECORD ON SYS001.
J3 WRITE FIRST RECORD ON SYS001
JC B1
ACCPARM
* A4 PDS IS LOCATED ON SYSVIS (= SYS000)
B4 SET SYS001 INDICATOR FOR DEVICE TEST (SEE * A4)
D4 LOG MESSAGE ATTEMPT TO DUMP A FILE OTHER THAN PDS
ENDJOB
E4 EOJ
B5
IGET1ST
B5 READ FIRST RECORD (SEE * H4)
C5 INTERNAL FILE ID
YES
NO
D5 SAVE FIRST RECORD
JC B1
ACCPARM

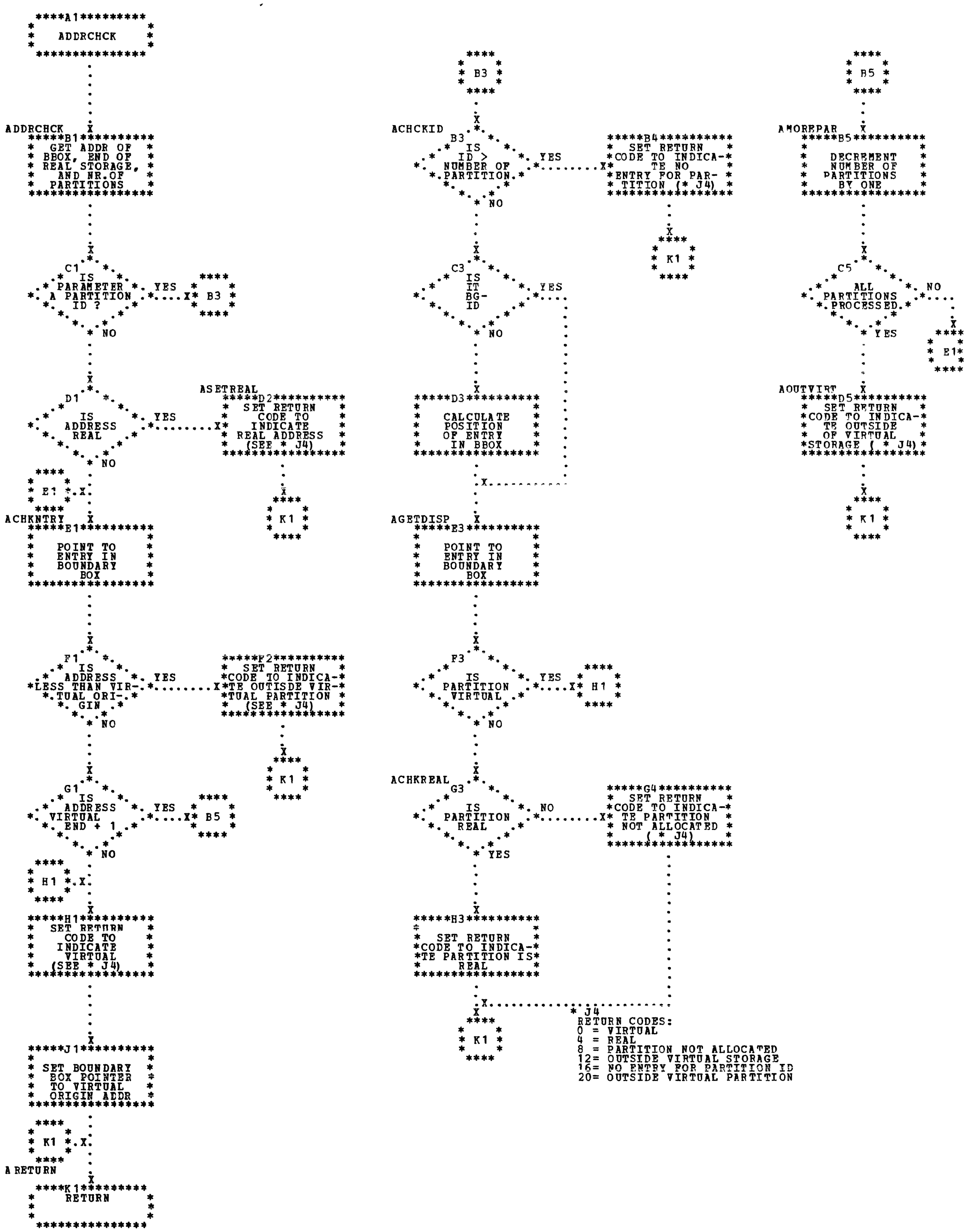
Chart JC. PDSDM - Parameter Acceptance (Part 3 of 4)



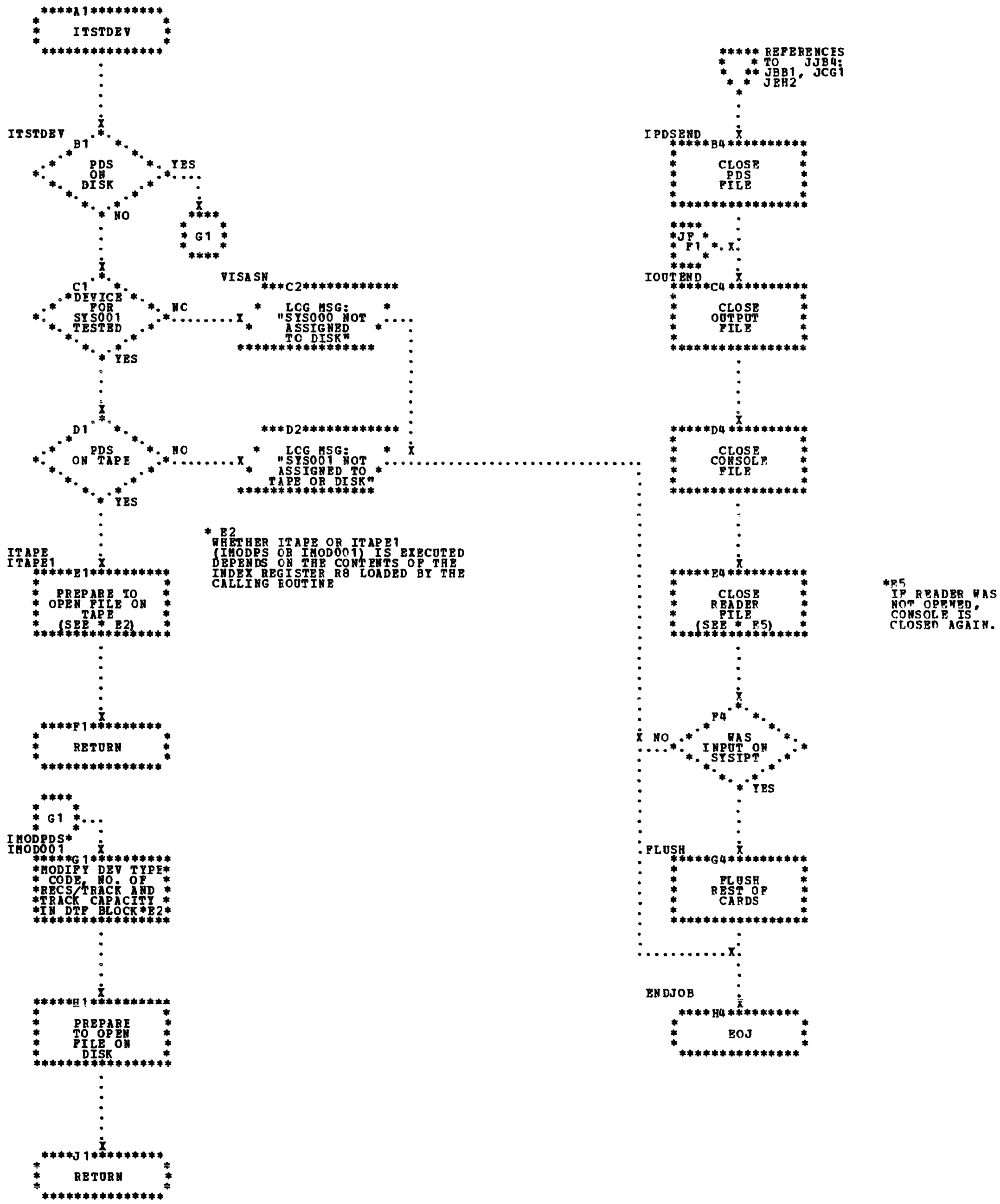

JC J1
ACCPAR4
B1 IS PARAMETER A PARTITION ID
NO
G1
YES
C1 ADDRCHCK JG
CHECK PARTITION
* C2
PARTITION MAY BE REAL OR NOT ALLOCATED; OR THERE MAY BE NO ENTRY FOR PARTITION.
D1 PARTITION VALID
NO
AERR11,12 OR 14
D2 LOG APPROPRIATE ERROR MESSAGE (SEE * C2)
B3
YES
E1 GET START AND END ADDR OF PARTITION
JC C5
G1
ACCPAR5
G1 ADDRESS IN HEX FORMAT
NO
AERR6
G2 LOG MSG: "ADDRESS NOT IN HEX FORMAT"
B3
YES
H1 ADDRCHCK JG
CHECK ADDRESS
J1 IS ADDRESS VIRTUAL
NO
AERR12,13 OR 17
J2 LOG APPROPRIATE ERROR MESSAGE
B3
YES
K1 MAKE PARTITION START AND END ADDRESS EQUAL
ACCPAR3
JC C5
REFERENCES TO JDB3: JCB4, JCH4
ACCPAR6
B3 LOG MESSAGE (SEE * B4)
* B4
MSG:" PRESS END KEY TO PROCESS PREVIOUS PARAMETERS OR ENTER NEW PARAMETER"
C3 READ RESPONSE
D3 END KEY
YES
G3
NO
E3 INITIALIZE PARAMETER TABLE TO START OVER AGAIN
JC G1
ACCPAR1
G3
ACCPAR7
G3 CAN PARAMETER BE PROCESSED
YES
JE B1
PRINT
NO
H3 LOG ERROR MESSAGE (SEE * H4)
*H4
MSG: "NO PREVIOUS PARAMETER AVAILABLE TO BE PROCESSED"
JC B1
ACCPARM
REFERENCES TO JDB5: JCC1, JCB1
ACCPAR8
B5 READ CARD FROM SYSIPT
C5 END KEY
YES
NO
D5 SET INPUT IN UPPER CASE
ACCPAR81
E5 SET NEW EOF ADDR. TO EOJ ROUTINE
JC J1
ACCPAR2


REFERENCES TO JEB1: JBD1, JCB1 JCD5, JDG3
PCKLP
B1 ANY MORE PARAME- TERS
NO
YES
JC B1
C1 OUTPUT TO SYSLST
YES
NO
C2 SKIP TO NEW PAGE
D1 READ CORRECT RECORD FROM PDS * J2
E1
E1 OUTPUT TO SYSLST
YES
NO
B3
F1 WRITE RECORD TO SYS001 (SEE * F2)
*F2 RECORD SIZE IS 2048 BYTES ON SYS001
G1
PCKLAST
G1 IOUTEND RECORDS WRITTEN
YES
NO
PCKALLP
G2 POINT TO NEXT PARAMETER IN LIST
H1 UPDATE CURRENT- RECORD NUMBER
H2 DEFAULT SWITCH ON
YES
NO
JJ B4
IPDSEND
B1
J1 READ RECORD
E1
* J2 THE NUMBER OF THE RECORD CURRENTLY IN STORAGE IS UPDATED IN "CURREC". IF A NEW PARAMETER REQUIRES A RECORD WITH A HIGHER NR. THAN "CURREC", THEN THE RECORDS IN BETWEEN ARE SKIPPED. IF THE RECORD NR. IS LESS THAN "CURREC", THE PDS FILE IS CLOSED AND THEN REOPENED IN ORDER TO ACCESS THE CORRECT RECORD.
B3
B3 PAGE CONTAIN ONLY X'00'
YES
NO
B5
C3 FIRST PAGE SWITCH ON
NO
YES
* D4 BLOCKS CONTAINING ONLY ZEROS ARE NOT PRINTED; INSTEAD THE BLOCK NUMBER(S) ARE PRINTED WITH A MESSAGE INDICATING ONLY ZEROS FOUND
D3 PREPARE MESSAGE (SEE * D4)
E3 PRINT MESSAGE
F3 PRINT BLOCK HEADER
G3 DETERMINE VIRTUAL ADDRESS
H3 PRNT2K JH PRINT ONE 2K BLOCK
G1
PCKLAST
B5
B5 FIRST PAGE SWITCH ON
YES
NO
G1
C5 SET FIRST PAGE SWITCH ON
D5 SAVE RECORD NUMBER
G1


A1
EOFROUT (SEE * A2)
*A2
THE ADDRESS OF THE EOFROUT IS STORED IN THE DTF TABLE FOR THE PDS FILE. WHEN AN EOF CONDITION OCCURS, THE EOF ROUTINE IS EXECUTED AND THE JOB IS TERMINATED.
A3
PREOPEN
EOFROUT
B1
BLOCKS WITH X'00' SUPPRESSED
NO
YES
PREOPEN
B3
CLOSE PDS
C1
PREPARE A PRINT MESSAGE
C3
OPEN PDS
D1
DETERMINE VIRTUAL ADDRESS CAUSING EOF
*D2
"ATTEMPT TO DUMP PAGE OUTSIDE PDS XXXXXX".
D3
PDS ON SYS001
NO
YES
E1
MAKE ADDR PRINTABLE AND INSERT IT IN ERROR
E3
READ FIRST RECORD
WRITMSG
F1
ISSUE ERROR MESSAGE (SEE * D2)
F3
RETURN
JJ C4
IOUTEND


A1 ADDRCHCK
ADDRCHCK
B1 GET ADDR OF BBOX, END OF REAL STORAGE, AND NR.OF PARTITIONS
C1 IS PARAMETER A PARTITION ID ?
YES B3
NO
D1 IS ADDRESS REAL
YES
ASETREAL
D2 SET RETURN CODE TO INDICATE REAL ADDRESS (SEE * J4)
K1
NO
E1
ACHKNTRY
E1 POINT TO ENTRY IN BOUNDARY BOX
F1 IS ADDRESS LESS THAN VIR-TUAL ORI-GIN
YES
F2 SET RETURN CODE TO INDICA-TE OUTISDE VIR-TUAL PARTITION (SEE * J4)
K1
NO
G1 IS ADDRESS VIRTUAL END + 1
YES B5
NO
H1
H1 SET RETURN CODE TO INDICATE VIRTUAL (SEE * J4)
J1 SET BOUNDARY BOX POINTER TO VIRTUAL ORIGIN ADDR
K1
ARETURN
K1 RETURN
B3
ACHCKID
B3 IS ID > NUMBER OF PARTITION.
YES
B4 SET RETURN CODE TO INDICA-TE NO ENTRY FOR PAR-TITION (* J4)
K1
NO
C3 IS IT BG-ID
YES
NO
D3 CALCULATE POSITION OF ENTRY IN BBOX
AGETDISP
E3 POINT TO ENTRY IN BOUNDARY BOX
F3 IS PARTITION VIRTUAL
YES H1
NO
ACHKREAL
G3 IS PARTITION REAL
NO
G4 SET RETURN CODE TO INDICA-TE PARTITION NOT ALLOCATED (* J4)
YES
H3 SET RETURN CODE TO INDICA-TE PARTITION IS REAL
K1
* J4
RETURN CODES:
0 = VIRTUAL
4 = REAL
8 = PARTITION NOT ALLOCATED
12= OUTSIDE VIRTUAL STORAGE
16= NO ENTRY FOR PARTITION ID
20= OUTSIDE VIRTUAL PARTITION
B5
AMOREPAR
B5 DECREMENT NUMBER OF PARTITIONS BY ONE
C5 ALL PARTITIONS PROCESSED.
NO E1
YES
AOUTVIRT
D5 SET RETURN CODE TO INDICA-TE OUTSIDE OF VIRTUAL STORAGE (* J4)
K1

```
     ****A1*********
     *             *
     *    PRNT2K   *
     *             *
     ***************
            :
     ****   :
    *    *  :                        * B2
    * B1 *.X:                         INSTRUCTION AT PSAMESW
    *    *  :                         IS CHANGED TO A "NOP"
     ****   :
 PSTORE14   X
       **B1*******
      *           *
     *  SUPPRESS   *
    *  SAME-LINE    *
     *   SWITCH    *
      *(SEE * B2) *
       ***********
            :
            :
            X
     *****C1**********
     *               *
     *    FORMAT     *
     *  AND PRINT    *
     *     ONE       *
     *     LINE      *
     *****************
     ****   :
    *    *  :
    * D1 *.X:
    *    *  :
     ****   :
 PREDUCE    X
     *****D1**********
     *               *
     *   DECREMENT   *
     *   NUMBER OF   *
     *   LINES BY    *
     *      ONE      *
     *****************
            :
            X
           .*.
      E1 .*   *.
       .*  LAST  *.  YES         ****E2*********
     *.    LINE    .*.........X*               *
       *. PRINTED .*           *    RETURN     *
         *.     .*             *               *
           *. .*               ***************
            * NO
            :
            X
 PCOMPARE  .*.
      F1 .*   *.
       .*  NEXT  *.  NO        ****
     *.   LINE =   .*.....X*    *
       *. CURRENT .*       X * B1 *
         *.LINE .*         :  *    *
           *. .*           :   ****
            * YES          :
            :              :
            X              :
           .*.             :                     * G3
      G1 .*   *.           :                      THE SWITCH IS 'ON' ONLY
       .* IS NEXT *.  YES  :                      WHEN MORE THAN ONE LINE
     *.  LINE THE  .*......                       IS IDENTICAL TO A
       *.  LAST   .*                              PREVIOUS LINE AND 'SAME'
         *.LINE .*                                HAS ALREADY BEEN PRINTED
           *. .*
            * NO
            :                                    * H3
            X                                     INSTRUCTION AT PSAMESW
 PSAMESW   .*.                                    IS CHANGED TO A BRANCH
      H1 .*   *.                                  TO PUPDATE
       .*   IS    *.         **H2*******
     .*   SWITCH    *.  NO   *SET SWITCH *
     *. SET TO SKIP .*......X*POLLOWING IDEN-*
       *.  LINES  .*        * TO SKIP ANY *
         *( *G3).*          * TICAL LINES *
           *. .*             * (SEE *H3) *
            * YES             ***********
     ****   :                      :
    *    *  :                      :
    * J1 *.X:                      :
    *    *  :                      :
     ****   :                      :
 PUPDATE    X                      X
     *****J1**********     ***J2************
     *               *     *               *
     *  GET ADDRESS  *     *  PRINT LINE   *
     *    OF NEXT    *     *    ADDRESS    *
     *     LINE      *     *   AND WORD    *
     *               *     *    'SAME'     *
     *****************     *****************
            :                      :
            X                      X
           ****                   ****
          *    *                 *    *
          * D1 *                 * J1 *
          *    *                 *    *
           ****                   ****
```


A1
ITSTDEV
ITSTDEV
B1
PDS ON DISK
YES
NO
G1
C1
DEVICE FOR SYS001 TESTED
NO
YES
VISASN
C2
LOG MSG: "SYS000 NOT ASSIGNED TO DISK"
D1
PDS ON TAPE
NO
YES
D2
LOG MSG: "SYS001 NOT ASSIGNED TO TAPE OR DISK"
* E2
WHETHER ITAPE OR ITAPE1 (IMODPS OR IMOD001) IS EXECUTED DEPENDS ON THE CONTENTS OF THE INDEX REGISTER R8 LOADED BY THE CALLING ROUTINE
ITAPE
ITAPE1
E1
PREPARE TO OPEN FILE ON TAPE (SEE * E2)
F1
RETURN
G1
IMODPDS*
IMOD001
G1
MODIFY DEV TYPE CODE, NO. OF RECS/TRACK AND TRACK CAPACITY IN DTF BLOCK
E2
H1
PREPARE TO OPEN FILE ON DISK
J1
RETURN
REFERENCES TO JJB4: JBB1, JCG1 JBH2
IPDSEND
B4
CLOSE PDS FILE
JF F1
IOUTEND
C4
CLOSE OUTPUT FILE
D4
CLOSE CONSOLE FILE
E4
CLOSE READER FILE (SEE * E5)
*E5
IF READER WAS NOT OPENED, CONSOLE IS CLOSED AGAIN.
F4
WAS INPUT ON SYSIPT
NO
YES
FLUSH
G4
FLUSH REST OF CARDS
ENDJOB
H4
EOJ

This program displays on SYSLST the contents of the label information cylinder of SYSRES.

The layout of the label information cylinder depends on:

- SYSRES device type
- The number of partitions defined at system generation time.

The different possible combinations are described in Figure 49.

The label information contained in the different label information cylinder areas are selected from the following job control file definition statement cards:

- For disk label . DLBL, EXTENT
  . VOL, DLAB, XTENT

- For tape label . TLBL
  . VOL, TPLAB

These cards are read from SYSRDR and the information is formatted and written on SYSRES by job control.

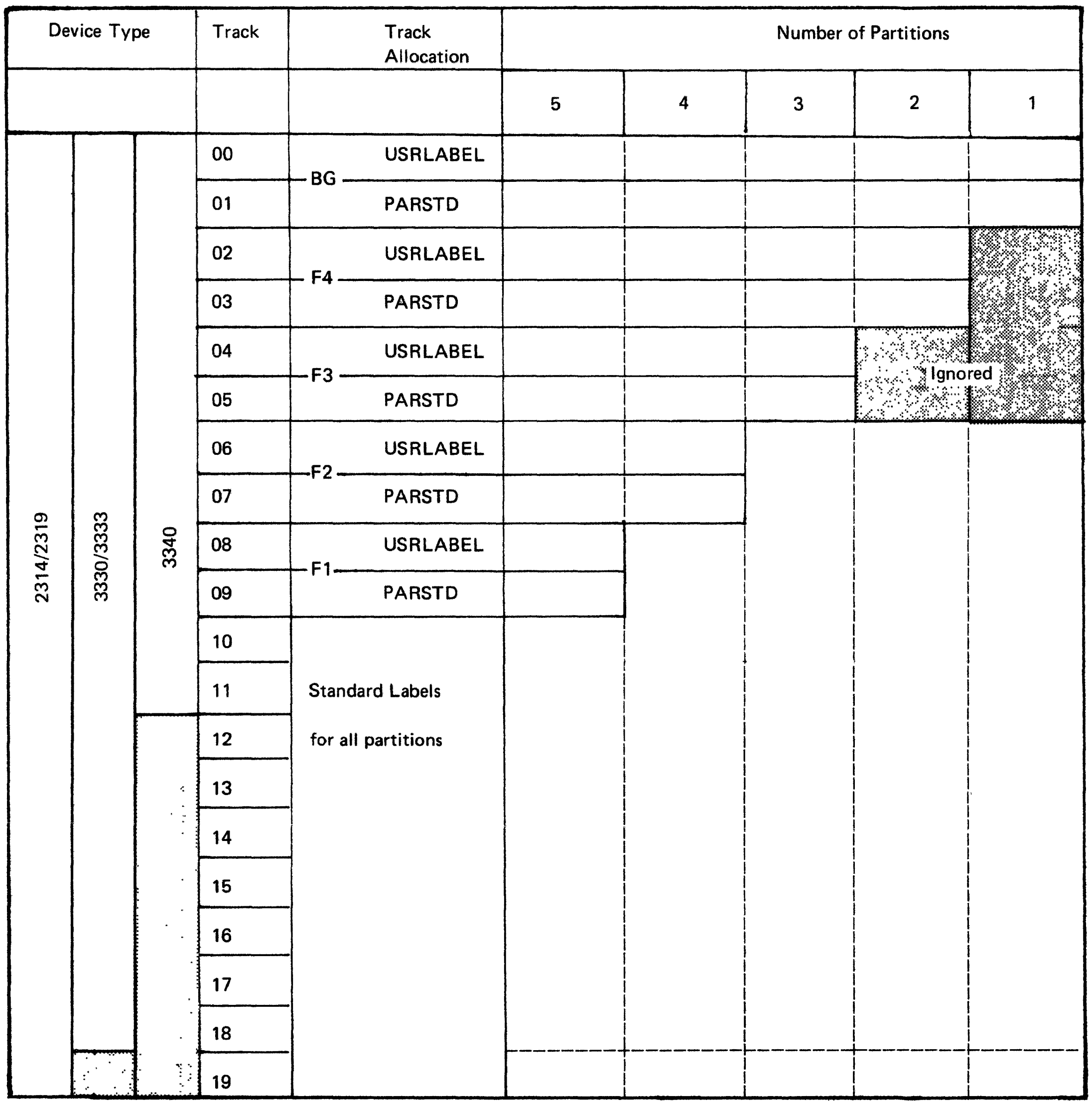


Figure 49. Label Information Cylinder Layout

The format of the label information records is described in Figure 50.

The area in which the records are written depends on the user-specified job control OPTION statement.

LSERV is executed by the use of the following job control statement from either SYSLOG or SYSRDR:

// EXEC LSERV

If LSERV runs real, the partition must have at least 8K bytes of real storage allocated.

LSERV assumes that the label information cylinder on SYSRES is formatted as described in Figure 49 and that the formats of the label information records correspond to those described in Figure 50.

If a defined area contains no label information records or only data-secured file label information records, a message is printed indicating that no records are present in this area.

Otherwise the label information records (except for data-secured files) are formatted and printed on SYSLST. Figure 51 shows a sample output of LSERV.

TAPE

	Field		Length
Key	1	File Name	7
	2	Not Used	1
Data	1	Same as Key Field	8
	2	Not Used	1
	3	File Name	7
	4	Not Used	1
	5	File Identifier	17
	6	File Serial Number	6
	7	Volume Sequence Number	4
	8	File Sequence Number	4
	9	Generation Number	4
	10	Creation Date	2
	11	Expiration Date	6
	12	File Security Number	1
	13	Block Count	6
	14	System Code	13
	15	Flag Bytes	2

DISK

	Field		Length
Key	1	File Name	7
	2	Not Used	1
Data	1	Same as Key Field	8
	2	DLBL-Extent Indicator	1
	3	File Name	7
	4	DA-IS Switch	1
	5	File Indentifier	44
	6	File Serial Number	6
	7	Volume Sequence Number	2
	8	Creation Date	3
	9	Expiration Date	3
	10	Retention Period	2
	11	Open Code	1
	12	System Code	13
	13	Volume Serial Number	6
	14	Extent Type	1
	15	Extent Sequence Number	1
	16	Extent Lower Limit	4
	17	Extent Upper Limit	4
	18	Logical Unit Address	2
	19	2321 Lower Cell	1
	20	2321 Upper Cell	1
		Another extent if present for DA or IS files	

Note: For sequential disk files, a complete block is repeated for each additional extent. For direct-access or ISAM files, only fields 14 through 20 are repeated for each extent.

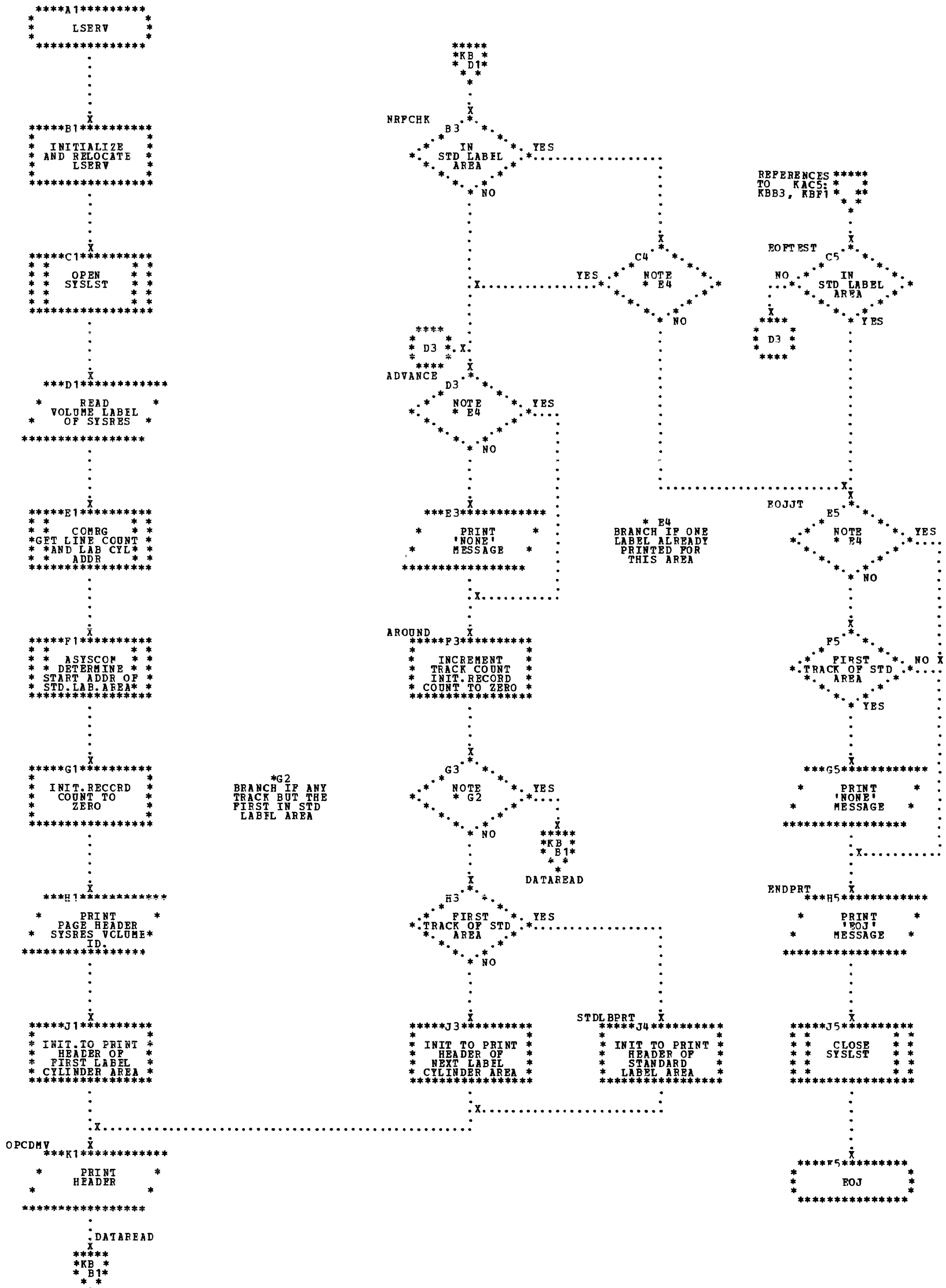
Figure 50. Format of the Label Information Records

```
DOS/VS LABEL CYLINDER DISPLAY

SYSRES VOLUME SERIAL NUMBER - DRV3.0

BG      USER LABELS (TEMPORARY PER PARTITION) TRACK 0
        NONE

BG      PARTITION STANDARD LABELS (PERMANENT) TRACK 1
        NONE

F4      USER LABELS (TEMPORARY PER PARTITION) TRACK 2
        NONE

F4      PARTITION STANDARD LABELS (PERMANENT) TRACK 3
        NONE

F3      USER LABELS (TEMPORARY PER PARTITION) TRACK 4
        NONE

F3      PARTITION STANDARD LABELS (PERMANENT) TRACK 5
        NONE

F2      USER LABELS (TEMPORARY PER PARTITION) TRACK 6
        NONE

F2      PARTITION STANDARD LABELS (PERMANENT) TRACK 7
        NONE

F1      USER LABELS (TEMPORARY PER PARTITION) TRACK 8
        NONE

F1      PARTITION STANDARD LABELS (PERMANENT) TRACK 9
        NONE
```

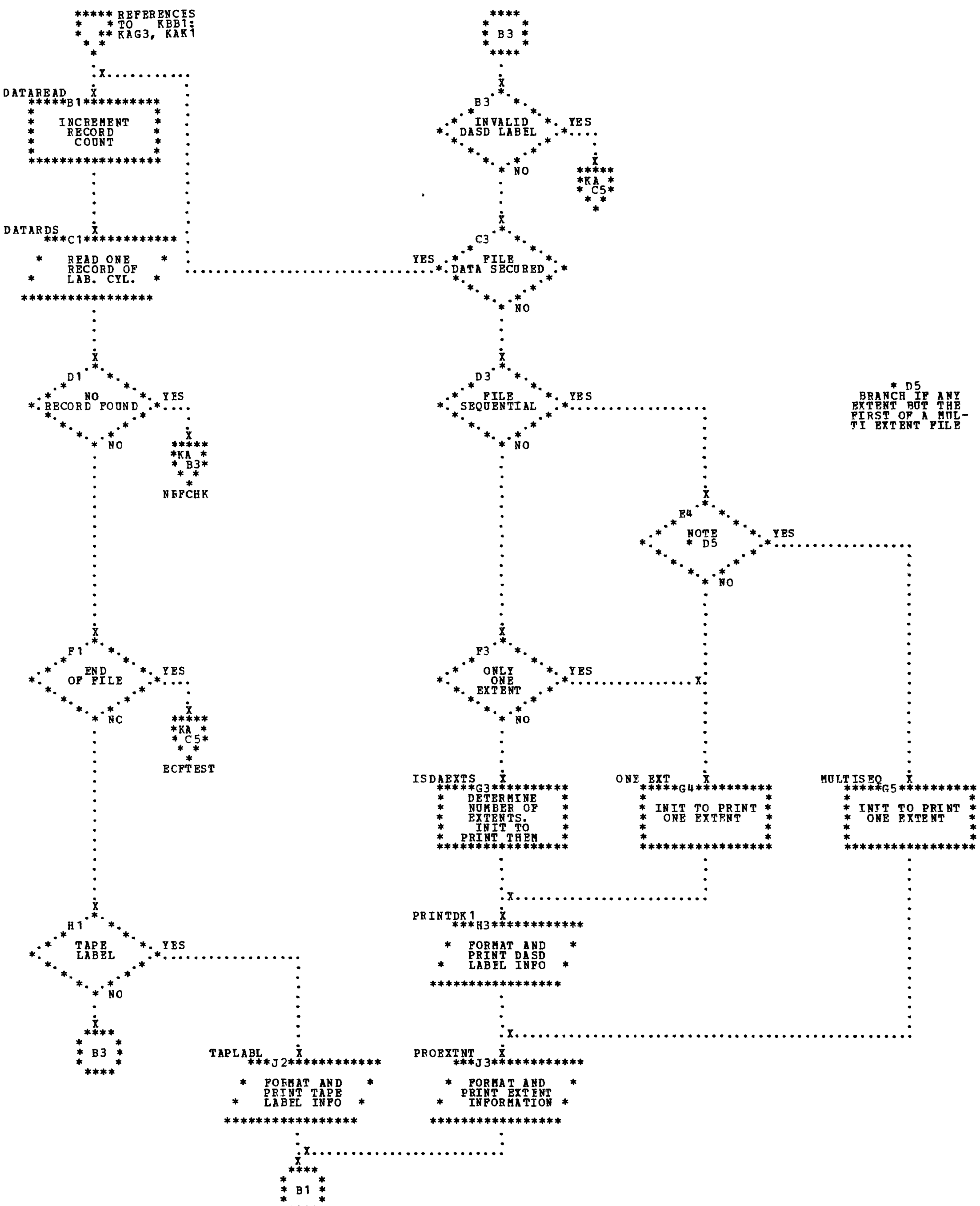
Figure 51. Sample LSERV Output (Part 1 of 2)

```
                          DOS/VS LABEL CYLINDER DISPLAY

STANDARD LABELS (ALL PARTITICNS-PERMANENT)

   IJSYSRS
         FILE IDENTIFIER                                DOS.SYSRES.FILE
         FILE SERIAL NUMBER                             DRV3.0
         VOLUME SEQUENCE NUMBER                         01
         CREATION DATE                                  OMITTED
         EXPIRATION DATE                                99/365
         FILE TYPE                                      SEQUENTIAL

    EXTENT INFORMATICN
         EXTENT SEQUENCE NUMBER                         01
         EXTENT TYPE                                    1 (PRIME DATA)
         RELATIVE (TO ZERO) START ADDRESS IN TRACKS     0001
         NUMBER OF TRACKS                               0739
         SYMBOLIC UNIT                                  SYSRES CCB FORMAT 0006
         VOLUME SERIAL NUMBER                           DRV3.0

   IJSYSCL
         FILE IDENTIFIER                                DOS.SYSCLB.FILE
         FILE SERIAL NUMBER                             DRV3.0
         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                 037
                                                        HEAD                      00
         EXTENT UPPER LIMIT                             CYLINDER                 068
                                                        HEAD                      19
         SYMBOLIC UNIT                                  SYSCLB CCB FORMAT 000B
         VOLUME SERIAL NUMBER                           DRV3.0

END OF LABEL CYLINDER DISPLAY
```

Figure 51. Sample LSERV Output (Part 2 of 2)

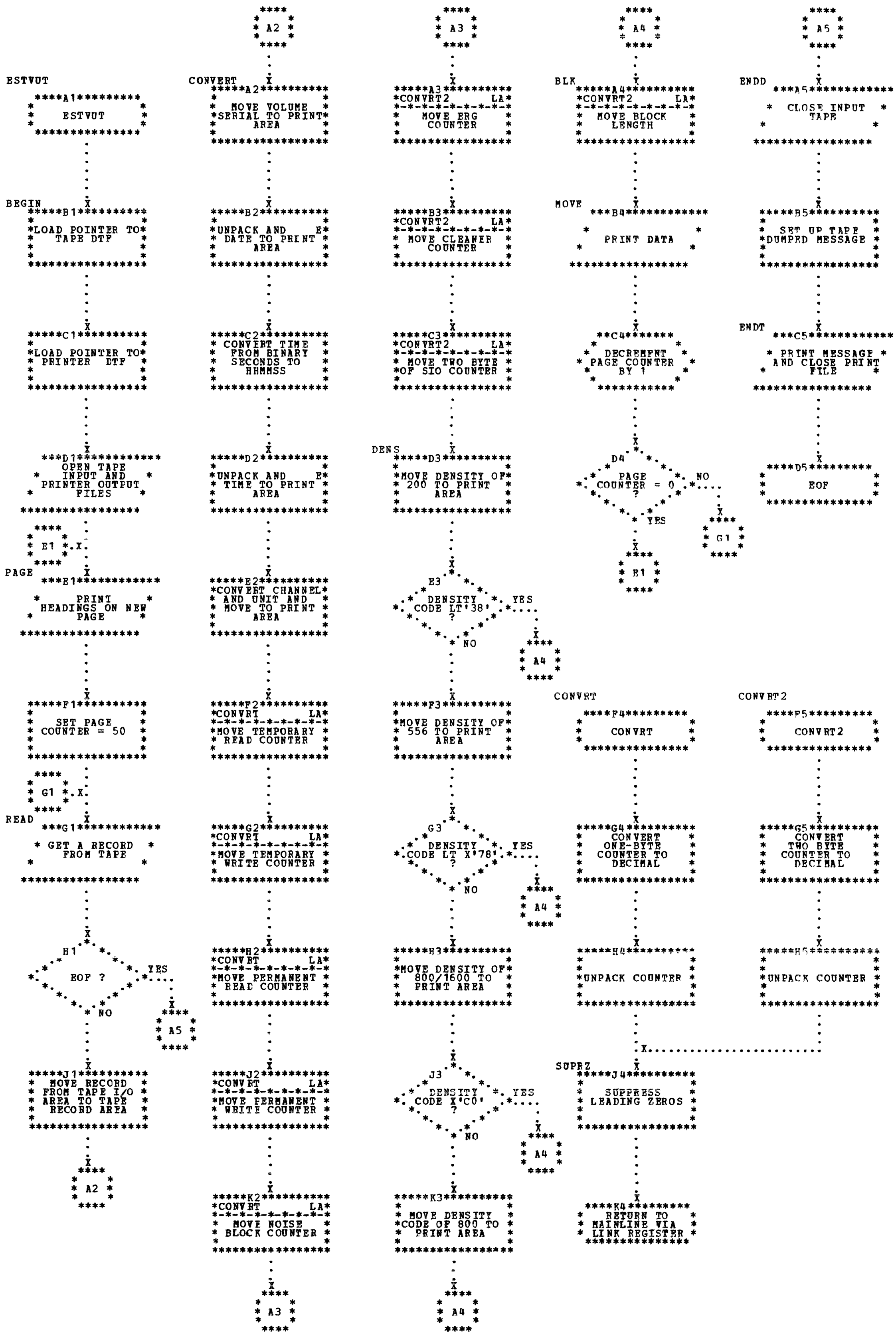


A1 LSERV
B1 INITIALIZE AND RELOCATE LSERV
C1 OPEN SYSLST
D1 READ VOLUME LABEL OF SYSRES
E1 COMRG *GET LINE COUNT *AND LAB CYL* ADDR
F1 ASYSCOM * DETERMINE * START ADDR OF * STD.LAB.AREA*
G1 INIT.RECORD COUNT TO ZERO
*G2 BRANCH IF ANY TRACK BUT THE FIRST IN STD LABEL AREA
H1 PRINT PAGE HEADER SYSRES VOLUME* ID.
J1 INIT.TO PRINT HEADER OF FIRST LABEL CYLINDER AREA
OPCDMV
K1 PRINT HEADER
DATAREAD
KB B1
KB D1
NRFCHK
B3 IN STD LABEL AREA
YES
NO
C4 NOTE * E4
D3
ADVANCE
D3 NOTE * E4
E3 PRINT 'NONE' MESSAGE
* E4 BRANCH IF ONE LABEL ALREADY PRINTED FOR THIS AREA
AROUND
F3 INCREMENT TRACK COUNT INIT.RECORD COUNT TO ZERO
G3 NOTE * G2
KB B1
DATAREAD
H3 FIRST TRACK OF STD AREA
J3 INIT TO PRINT HEADER OF NEXT LABEL CYLINDER AREA
STDLBPRT
J4 INIT TO PRINT HEADER OF STANDARD LABEL AREA
REFERENCES TO KAC5: KBB3, KBF1
EOFTEST
C5 IN STD LABEL AREA
D3
EOJJT
E5 NOTE * E4
F5 FIRST TRACK OF STD AREA
G5 PRINT 'NONE' MESSAGE
ENDPRT
H5 PRINT 'EOJ' MESSAGE
J5 CLOSE SYSLST
K5 EOJ


REFERENCES TO KBB1: KAG3, KAK1
DATAREAD B1
INCREMENT RECORD COUNT
DATARDS C1
READ ONE RECORD OF LAB. CYL.
D1 NO RECORD FOUND
YES
KA B3
NEFCHK
NO
F1 END OF FILE
YES
KA C5
EOFTEST
NO
H1 TAPE LABEL
YES
NO
B3
TAPLABL J2
FORMAT AND PRINT TAPE LABEL INFO
B1
B3
B3 INVALID DASD LABEL
YES
KA C5
NO
C3 FILE DATA SECURED
YES
NO
D3 FILE SEQUENTIAL
YES
NO
D5 BRANCH IF ANY EXTENT BUT THE FIRST OF A MULTI EXTENT FILE
E4 NOTE D5
YES
NO
F3 ONLY ONE EXTENT
YES
NO
ISDAEXTS G3
DETERMINE NUMBER OF EXTENTS. INIT TO PRINT THEM
ONE EXT G4
INIT TO PRINT ONE EXTENT
MULTISEQ G5
INIT TO PRINT ONE EXTENT
PRINTDK1 H3
FORMAT AND PRINT DASD LABEL INFO
PROEXTNT J3
FORMAT AND PRINT EXTENT INFORMATION

ESTVUT is a utility program used to list (on SYSLST) tape error statistics by volume from the TES history tape created by the EREP (Environmental Recording, Editing, and Printing) program. ESTVUT can be executed from either a card reader or from SYSLOG. SYS005 must be assigned to the magnetic tape unit on which the TES history tape file is mounted. SYSLOG must be assigned to a 3210, 3215, or equivalent device.

Chart LA. ESTVUT - Error Statistics by Tape Volume Utility Program



EREP provides retrieval support for system environmental records created by RMSR. These records include error data for a CPU, channels, and attached I/O devices, as well as data about system events such as volume dismount or IPL. For the format of these records refer to the DOS/VS Error Recovery and Recording Transients PLM listed in the Preface.

EREP provides six types of error statistics retrieval. Error statistics may be edited and printed, selectively edited and printed by record type, summarized by record type, used to create a continuing history of system events on a History or RDE tape, used to create a history of magnetic tape volume reliability on a TES history tape, or used to create a system reliability summary.

Detailed operating information, including options available and execution procedures, can be found in Section 2-F-3 of DOS/VS Serviceability Aids and Debugging Procedures, GC33-5380.

EREP Requirements

EREP is executed as a problem program in either virtual or real mode, using standard job control language. It requires at least 10K bytes for execution. EREP also requires the following device assignments:

- SYSREC or SYS009 (depending on the option to be executed) -- Error statistics input to EREP
- SYSLOG -- Communications and/or control information input
- SYSLST -- As output
- SYS009 (depending on option to be executed) -- As output
- SYSIPT (optional) -- Control information input

EREP will use available partition space in excess of 10K bytes for all summary phases except 2715 and RDE. The extra address space, if executing in real mode, significantly improves performance during summarization.

EREP Processing

In order to perform all the functions of the current EREP program, many phases are required that execute in an overlay structure. Figure 52 lists all EREP phases and shows the relative position of each within the user partition.

As long as EREP is active within a partition, it will contain at least two of these phases: EREP, which is always resident, and one of the control phases.

In order to perform certain functions, the EREP phase, a control phase, a processing phase, and possibly a table phase may be required.

Processing by Option

Charts 9-18 show the structure and order of execution of EREP phases during the processing of an EREP option. They include, where helpful, a general explanation of the function of each phase, the type of phase, (control, table, etc.), and its relative position in the partition. Actual displacements into the partition may be obtained from Figure 53.

If you wish to examine the phases involved in the execution of an EREP option, start at:

Chart 9 -- EDIT or SELECT

Chart 14 -- SUM (except TES and RDE)

Chart 15 -- TES SUM

Chart 16 -- TES (except SUM)

Chart 17 -- HIST

Chart 18 -- RDESUM

Examination of these charts gives you a clear understanding of what EREP does before you refer to the functional charts for a particular phase.

EREP Phase Functions

Charts MA-NL describe the function of EREP phases. Included are the input to the phase, the output of the phase, and the logical steps performed during execution of that phase. Associated with each step is the label, or labels, of the subroutine(s) required to perform that logical step. Most EREP table phases contain little or no executable code. The description of the operation of these phases has not been included in this manual. Some self-explanatory examples of the output of some of these phases have, however, been included for your convenience (see Appendix C).

Phase Interaction

EREP control phases perform several general functions. They open required devices, communicate with the operator, examine the applicable options, call the required processing phases or other control phases, and provide common subroutines for the called processing phases.

EREP processing phases read SYSREC or History/RDE records (via subroutines in a control phase), examine the record type, call the applicable table phase if the record is of the correct type, and pass control to a list processor (usually in the control phase), to process the macros in the called table.

EREP table phases contain a series of macros, and some executable code, that controls the execution of subroutines to edit and write records in the applicable format. These macros may also point to DCs for headings and descriptors for the records.

If the description of an environmental event requires more than one SYSREC or History/RDE record, the EREP table phase may also cause these additional records to be read and may call other table phases to fully process the event description.

Information Passing

There are two methods used to pass information between EREP phases.

The most widely used is a communication region. The EREP communication region is part of the EREP phase and is always resident in the EREP partition. This contains (after EREPMNTR has read and checked all available EREP control statements) all options, suboptions, and parameters to be processed, the EREPMNTR return address, a description of SYSREC, and areas to store information about individual error records.

The EREPEDIT common region is used as a communication link between EREPEDIT and the associated processing phases. It contains disk search information, a branch table used by the subroutines, and a table of pointers used by both EREPEDIT and the associated processing phases.

The second method to pass information (used during summarization, selective retrieval, and 2715 edit and print) is by loading an overlay phase around data gathered by the overlaid phase. Data area description is accomplished by DSECTs within the EREP phases.

EREP Displacements

EREP is a self-relocating program that can execute in either a real or virtual partition. If EREP runs in a real partition, at least 10K bytes of real address area must be allocated to the partition. If additional 8K blocks of real address area are allocated when summarizing (except for 2715 and RDE records), the summarizing will be speeded up. This is true for real mode execution of EREP; for virtual mode it is not possible to give relative performance estimates.

All displacements in this chart are relative to the partition start address in an MPS system when tape labels are not being used. When tape labels are used, add 80 bytes (X'50') to all displacements.

EREP Phase	Displacement (Dec)	Displacement (Hex)
EREP	+0000	+0000
EREPMNTR	+0200	+00C8
EREPEDIT	+0200	+00C8
EREPHIST	+0200	+00C8
EREPSMCP	+0200	+00C8
EREPEDCP	+0200	+00C8
EREPSMES	+0200	+00C8
EREPEDES	+0200	+00C8
EREPTES	+0200	+00C8
EREPRDE	+0200	+00C8
EREPESTR	+4088	+0FF8
EREPESWK	+4088	+0FF8
EREPMCAR	+4376	+1118
EREPCCH	+4376	+1118
EREPUNIT	+4376	+1118
EREPIPL	+4376	+1118
EREP2715	+4376	+1118
EREPRETV	+4376	+1118
EREPESPT	+4376	+1118
EREPSMTD*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMDA*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMUR*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMTP*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMCR*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMDK*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMCU*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMT1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMD1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMU1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPSMTR	+5448	+1548
EREPEDTD*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDDA*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDUR*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDTP*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDCR*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDDK	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDCU*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDT1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDX1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDD1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDU1*	+5448+(n-1)8192	+1548+(n-1)2000
EREPEDTR	+5448	+1548
EREPSHRT	+6424	+1918
EREPSHRU	+6424	+1918
EREP2400	+6424	+1918
EREP3420	+6424	+1918
EREP3410	+6424	+1918
EREPUCCM	+6424	+1918
EREPDOLD	+6424	+1918
EREP3330	+6424	+1918
EREP333X	+6424	+1918
EREP333Y	+6424	+1918
EREP333Z	+6424	+1918
EREP3340	+6424	+1918
EREP334X	+6424	+1918
EREP334Y	+6424	+1918
EREP334Z	+6424	+1918
EREPTPE1	+6424	+1918
EREPUOLD	+6424	+1918
EREP3211	+6424	+1918
EREP3500	+6424	+1918
EREP3540	+6424	+1918
EREPEOD	+6424	+1918
EREPADPT	+4482	+1182
EREPASTA	+4482	+1182
EREPSPCL	+4482	+1182

EREP Phase	Displacement (dec)	Displacement (Hex)
EREPCCHC	+6424	+1918
EREPC145	+6424	+1918
EREP145A	+6424	+1918
EREP145B	+6424	+1918
EREP145C	+6424	+1918
EREPC135	+6424	+1918
EREPM145	+6424	+1918
EREP145D	+6424	+1918
EREPMCRC	+6424	+1918

* If there is sufficient partition space, two or more of these phases may be loaded simultaneously. n indicates the load position relative to the beginning of the partition (first, second, etc.). The actual load address, relative to the partition start address, is obtained by substituting the load position for n and solving the expression.

EREP Edit or Selective Retrieval Phases →

	Processing Phase →	EREPMCAR					EREPCCH							EREPUNIT																				EREP2715			EREPIPL
DEVICE TYPES	Table Phase →	MCRC	M145	145D	C135	C125	CCHC	C145	145A	145B	145C	C135	C125	SHRT	SHRU	2400	3420	3410	UCCM	DOLD	UOLD	TPE1	3330	333X	333Y	333Z	3340	334X	334Y	334Z	3500	3211	3540	ADPT	SPCL	ASTA	EOD
CPU	M/115	X				X																															
	M/125	X				X																															X
	M/135	X			X																																X
	M/145	X	X	X																																	X
	M/115-II, 158	X																																			X
	MPX Channel						X	X		X		X	X																								
	SEL Channel						X	X	X			X																									
	BMPX Channel						X	X	X			X																									
	IFA						X	X			X	X	X																								
Printer	1403/1403U													X	X				X		X																
	1443													X	X				X		X																
	3211													X	X				X																		
Reader /Punch	1442N1/2596													X	X				X		X																
	2501/2520/2540/3881													X	X				X		X																
	3505/3525																		X																		
Tape	3410/3411													X				X	X																		
	3420													X			X	X	X																		
	2400													X		X			X																		
Disk	2314/2311/2321													X	X				X	X																	
	3330																		X				X	X	X	X											
	3340																		X								X	X	X	X							
Disk-ette	3540													X	X				X														X				
TP	2701/2702/2703													X					X			X															
	2715																		X															X	X	X	
MICR OCR	1255/1259/1419													X	X				X		X																
	1287/1288													X	X				X		X																
	3886													X	X				X		X																
	Unsupported													X	X				X		X																

Figure 52. EREP Table Phases Used for Device Processing (Part 1 of 2)

Figure 52. EREP Table Phases Used for Device Processing (Part 2 of 2)

EREP Summary Phases →

Device Types	Processing Phase →	EREPSMCP											EREPEDCP											
	Table Phase →	SMTR	SMDA	SMTD	SMUR	SMTP	SMCU	SMCR	SMDK	SMD1	SMT1	SMU1	EDTR	EDDA	EDTD	EDUR	EDTP	EDCU	EDCR	EDDK	EDD1	EDT1	EDX1	EDU1
CPU	M/145						X											X						
	MPX Channel						X											X						
	SEL Channel						X											X						
	BMPX Channel						X											X						
	IFA						X											X						
Printer	1403/1403U				X											X								
	1443				X											X								
	3211											X												X
Reader /Punch	1442N1/2596				X											X								
	2501/2520/2540/3881				X											X								
	3505/3525											X												X
Tape	3410/3411										X												X	
	3420										X											X		
	2400			X											X									
Disk	2314		X											X										
	3330		X							X				X							X			
	3340		X							X				X							X			
Disk-ette	3540								X											X				
TP	2701/2702/2703					X											X							
	2715	X											X											
MICR OCR	1255/1259/1419							X											X					
	3886							X											X					

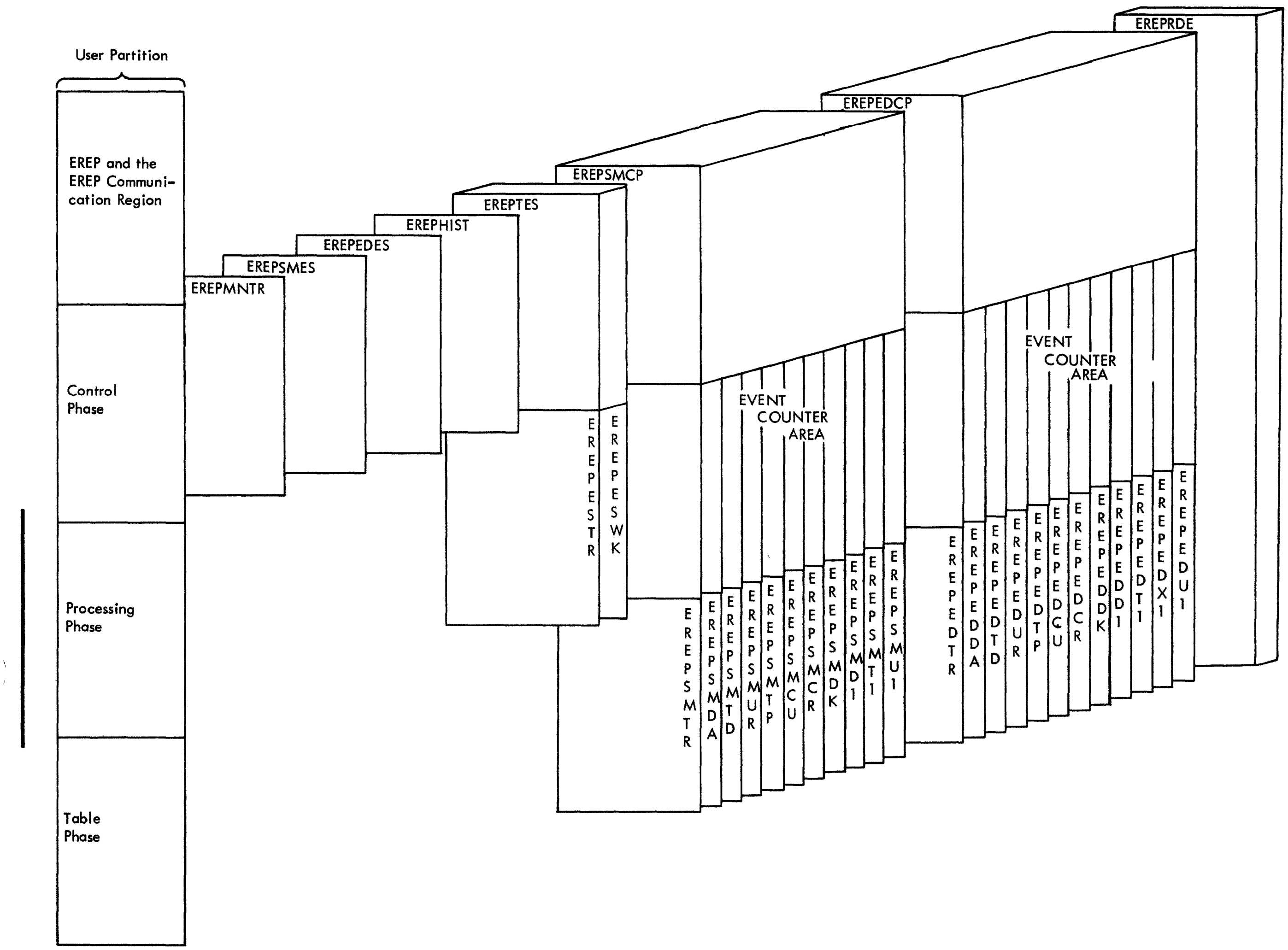


Figure 53. Relative Displacement of EREP Phases (Part 1 of 2)

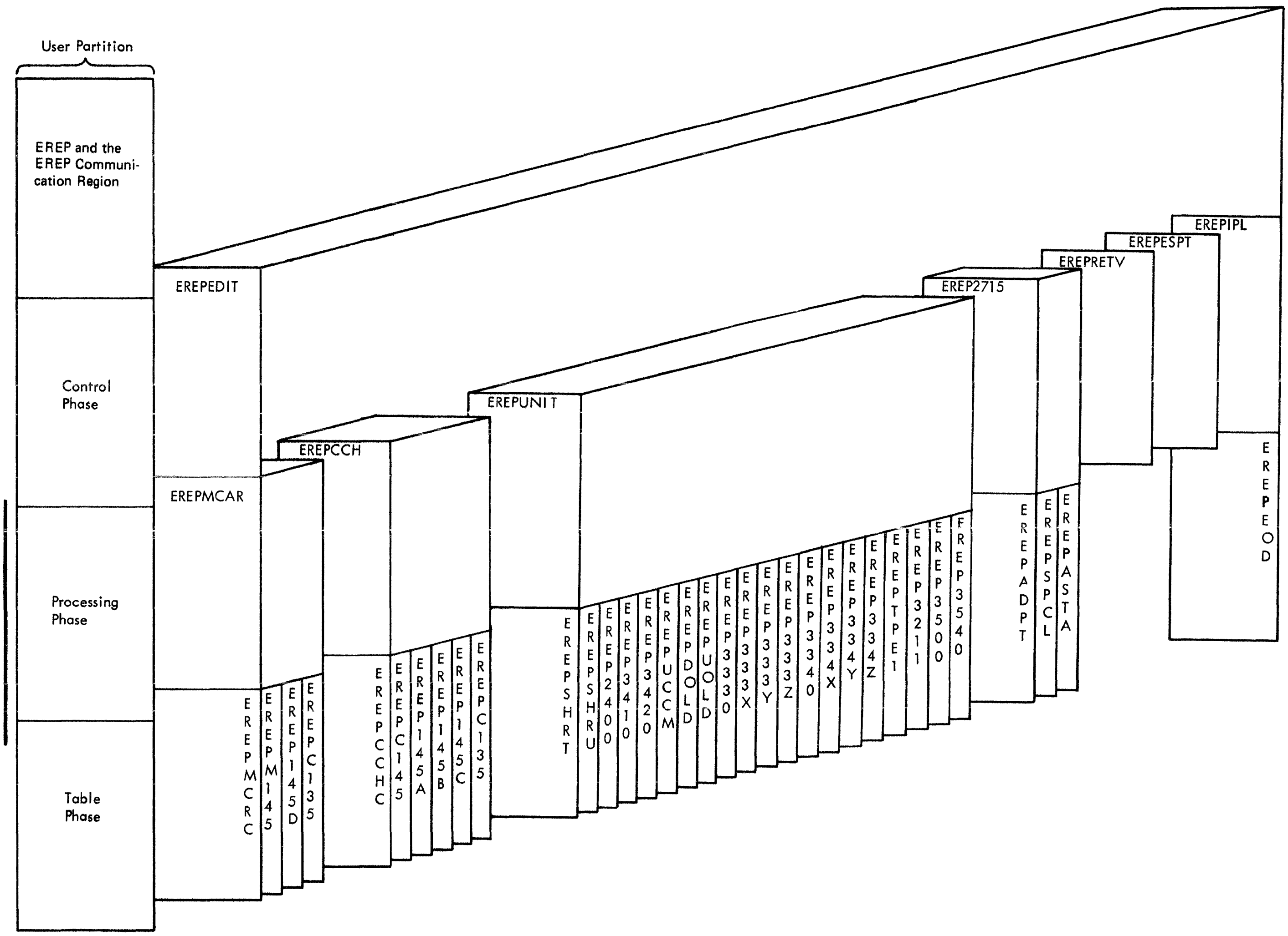


Figure 53. Relative Displacement of EREP Phases (Part 2 of 2)

Chart 09. Options EDIT and SELECT: Structure

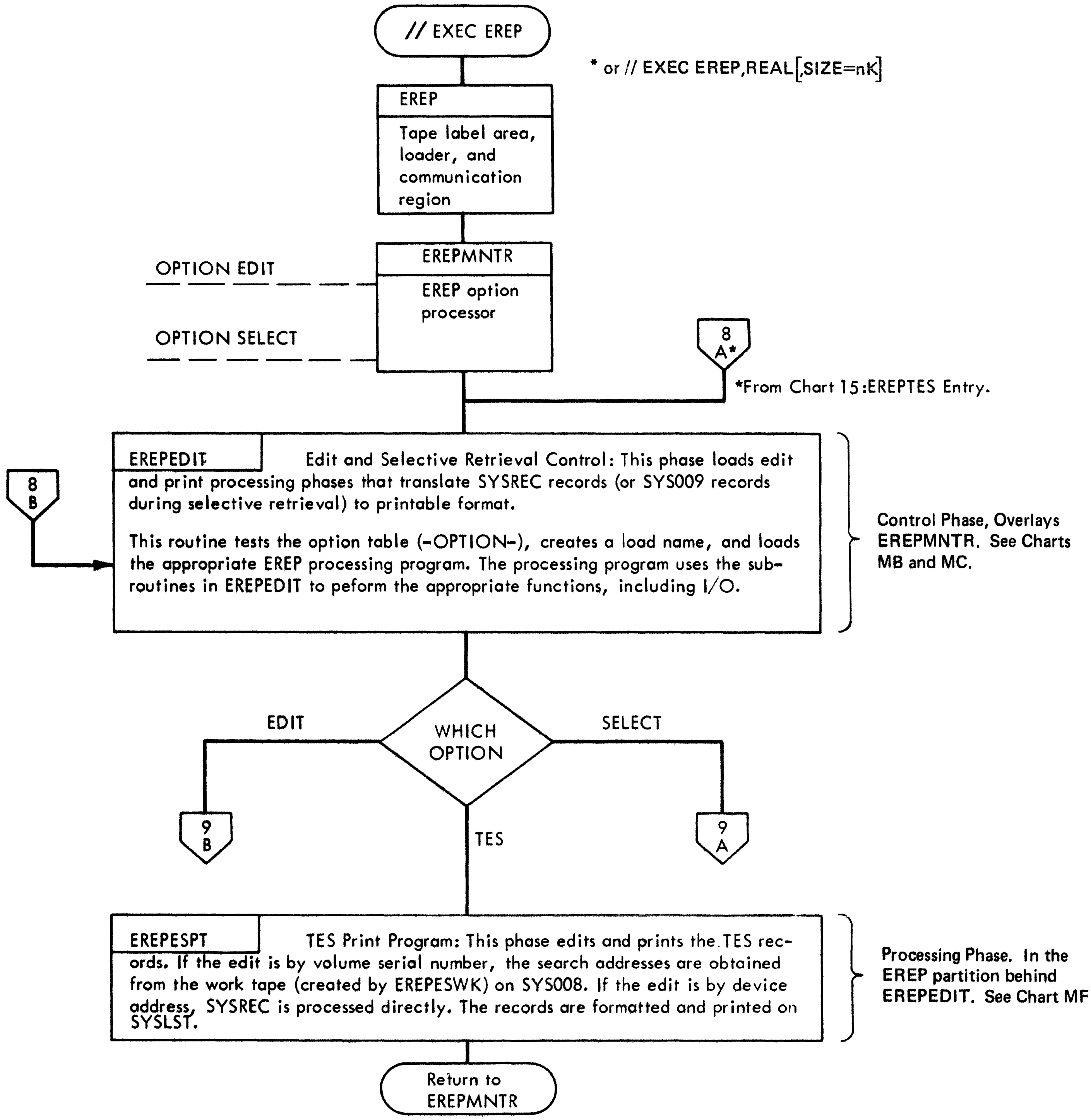


Chart 10. Selective Retrieval and Edit Structure

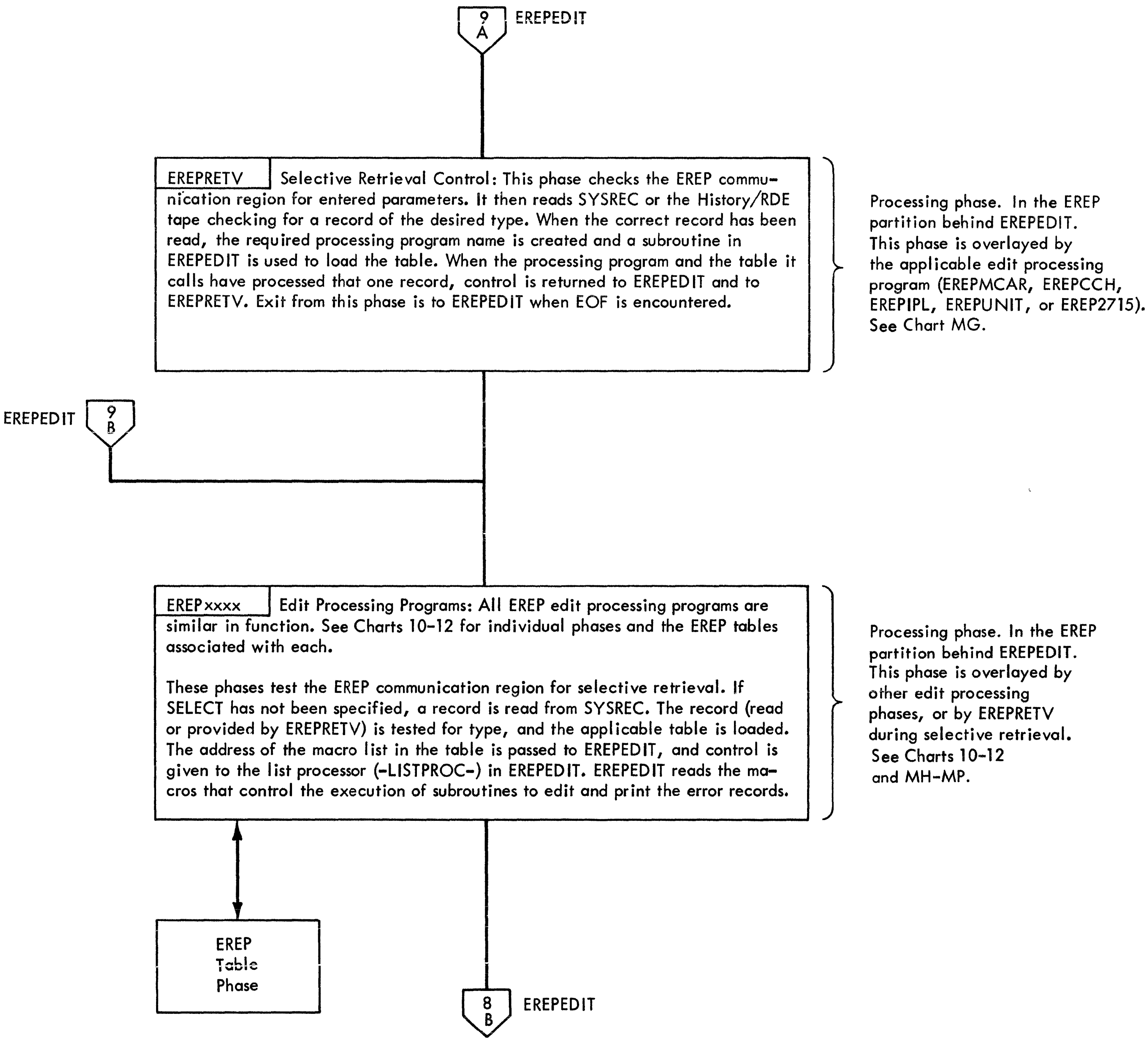


Chart 11. Edit Processor and Table Phase Structure (Part 1 of 3)
See Charts HH and HJ

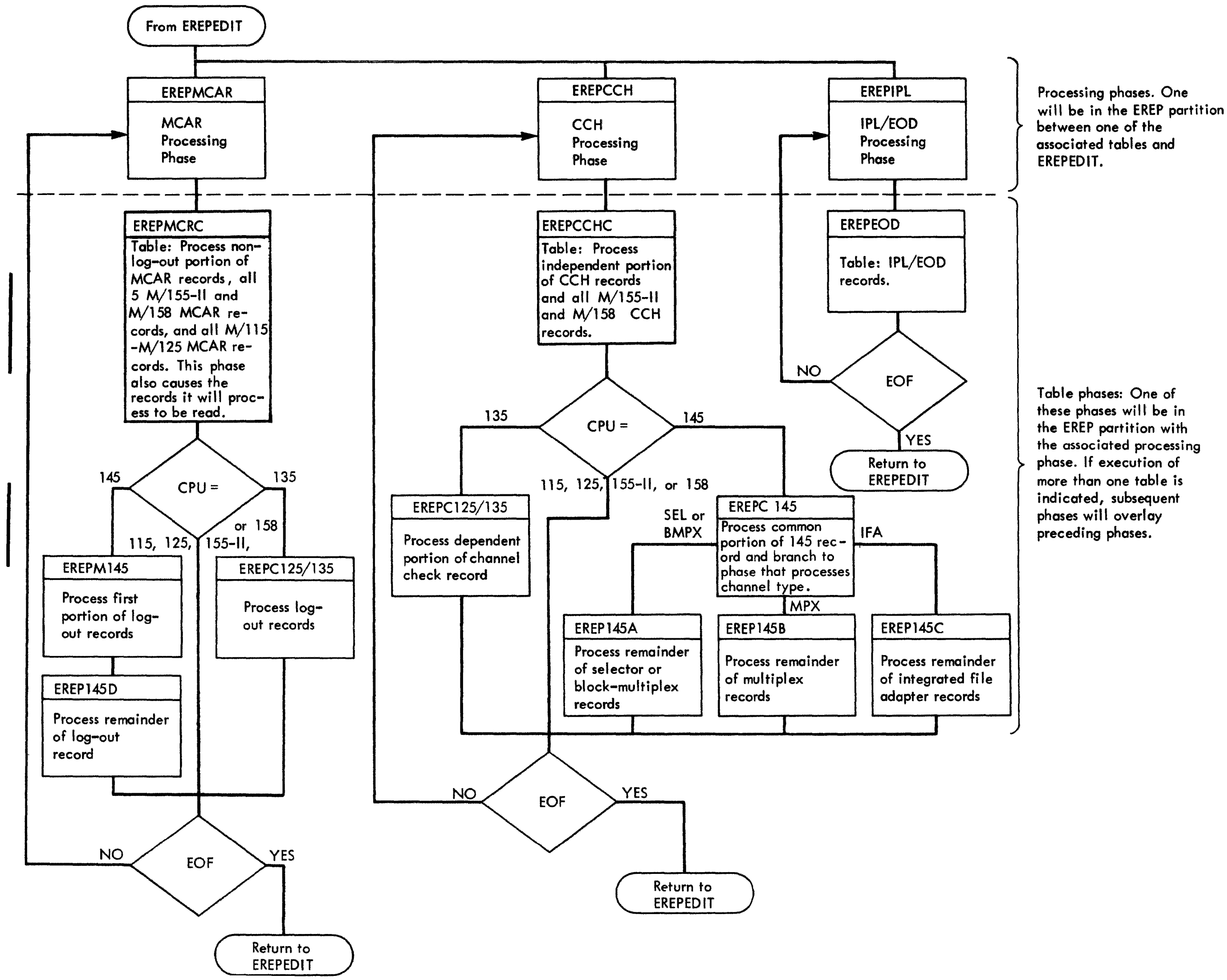


Chart 12. Edit Processor and Table Phase Structure (Part 2 of 3)
See Chart HK

Processing phase. In the EREP partition during processing of unit check records.

Table phases. One of these phases will be in the EREP partition with EREPUNIT. If execution of more than one phase is indicated, subsequent phases will overlay the preceding phase.

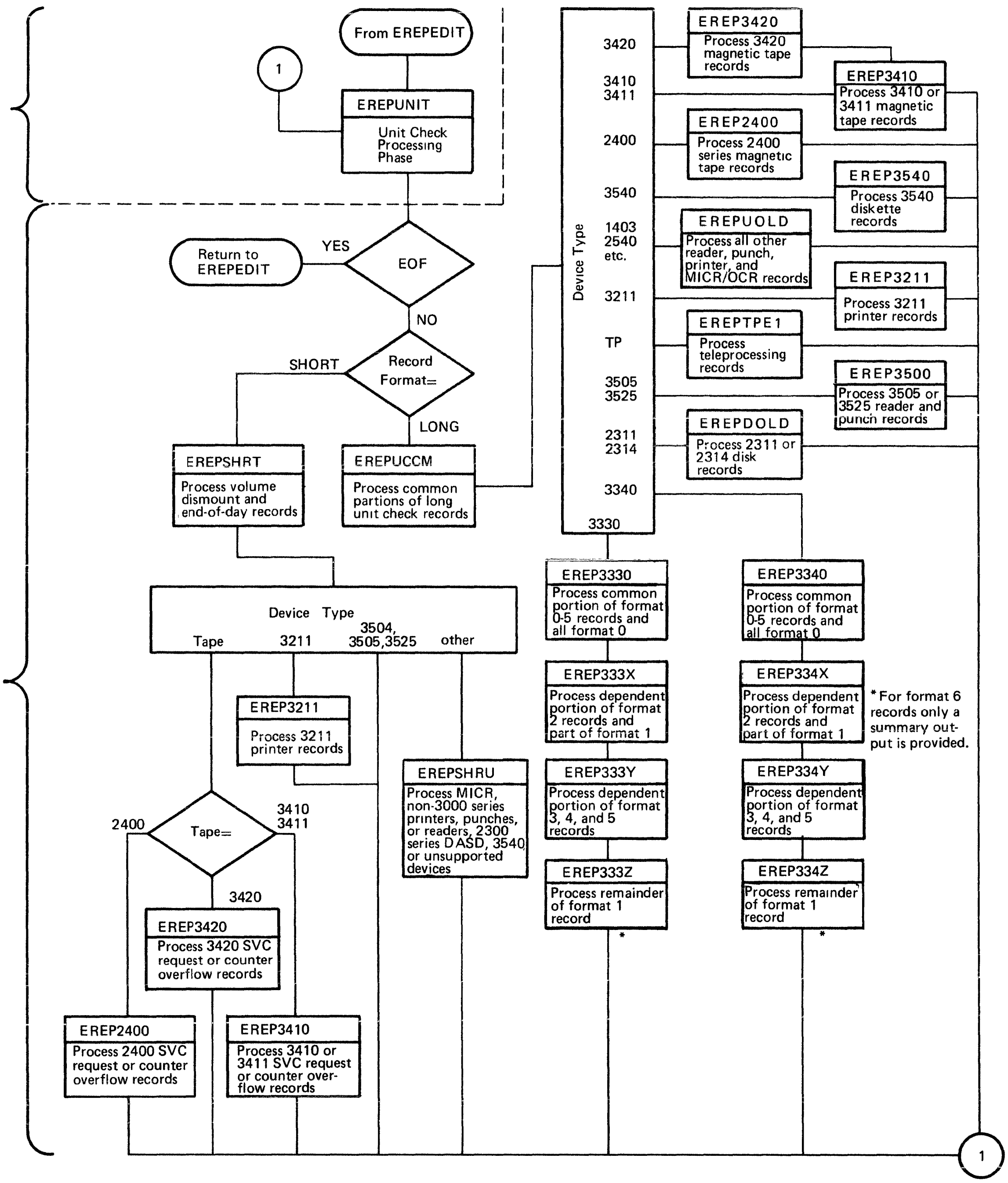


Chart 13. Edit Processor and Table Phase Structure (Part 3 of 3)
See Charts HL - HP

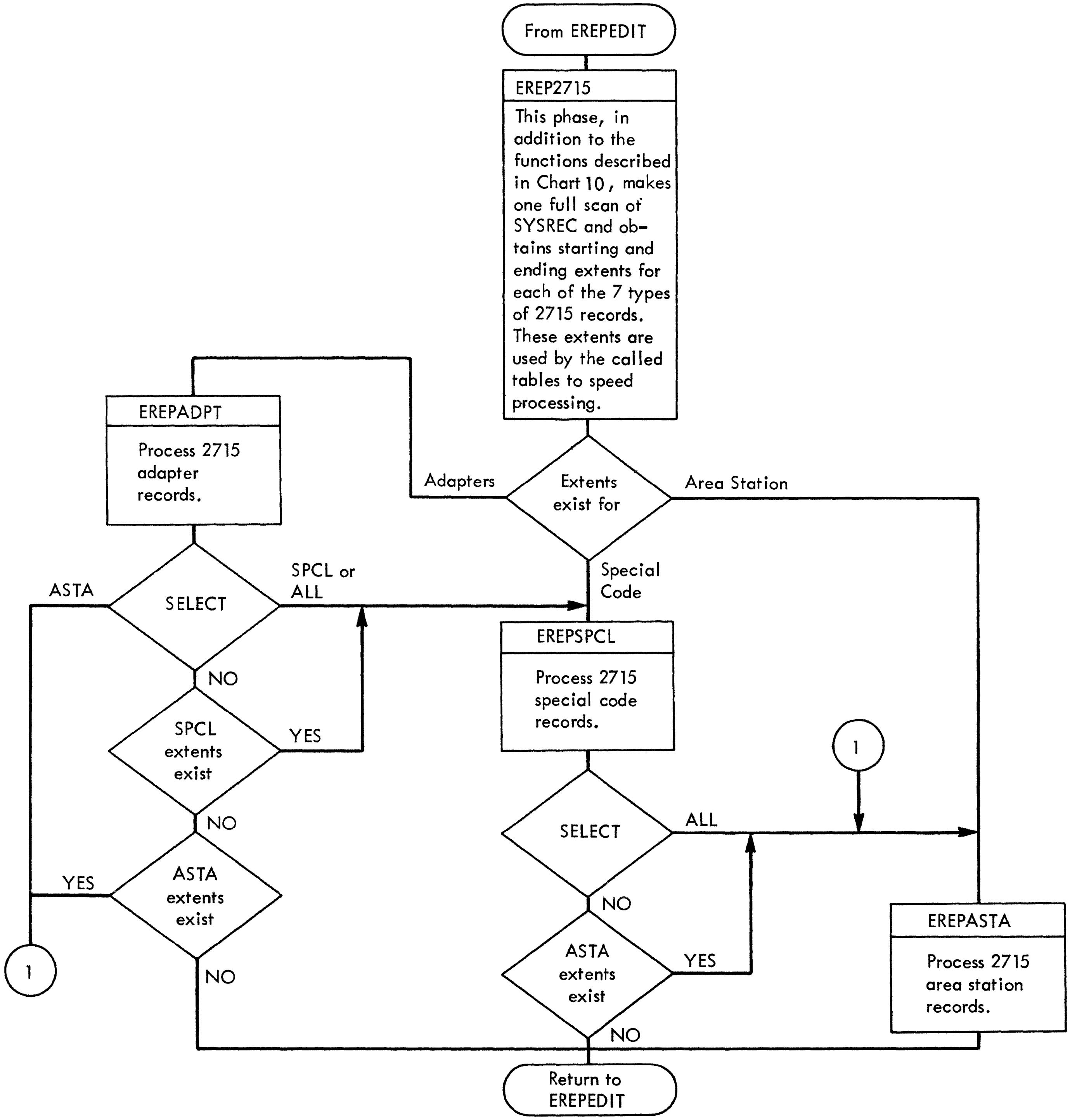


Chart 14. Summary Phases (Non-TES)

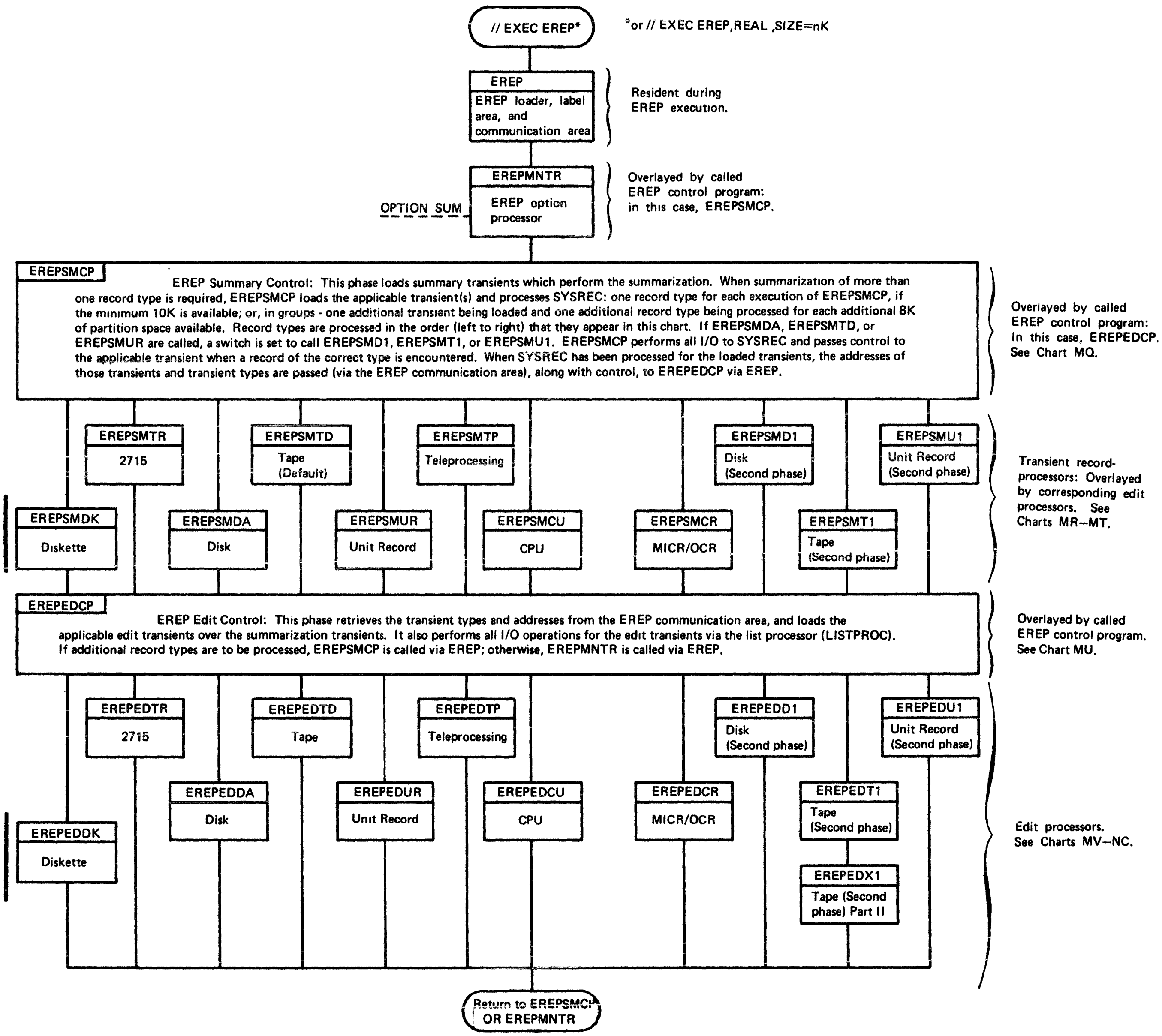


Chart 15. Option SUM: Structure for TES

// EXEC EREP*

*or // EXEC EREP,REAL[,SIZE=nK]

EREP

Tape label area, loader, and communication region.

EREPMNTR

EREP option processor.

OPTION TES, SUM

EREPSMES

TES Summary Phase: This phase checks the available partition space, creates a list of error counters by volume serial number for each day recording has taken place, or by device address (channel and unit). As each record is read, from either SYSREC or the TES history tape, it is placed into the error counter list or added to an existing record. If the error counters overflow the existing space, the last counter ID is saved in the EREP communication region and -SUMSWTH2- is set so that this phase is called to process the remaining records.

Control Program. Overlays EREPMNTR. See Chart ND.

EREPEDES

TES Summary Edit Phase: This phase edits and prints the list of error counters created by EREPSMES. The number of counters is calculated and the type of summary is obtained by testing -SUMSWTH2-. This phase processes a group of macros that control the execution of subroutines, which edit and print the error counters.

Control Program. Overlays EREPSMES. See Chart NE.

Return to EREPMNTR

Chart 16. TES: Print and Tape Output

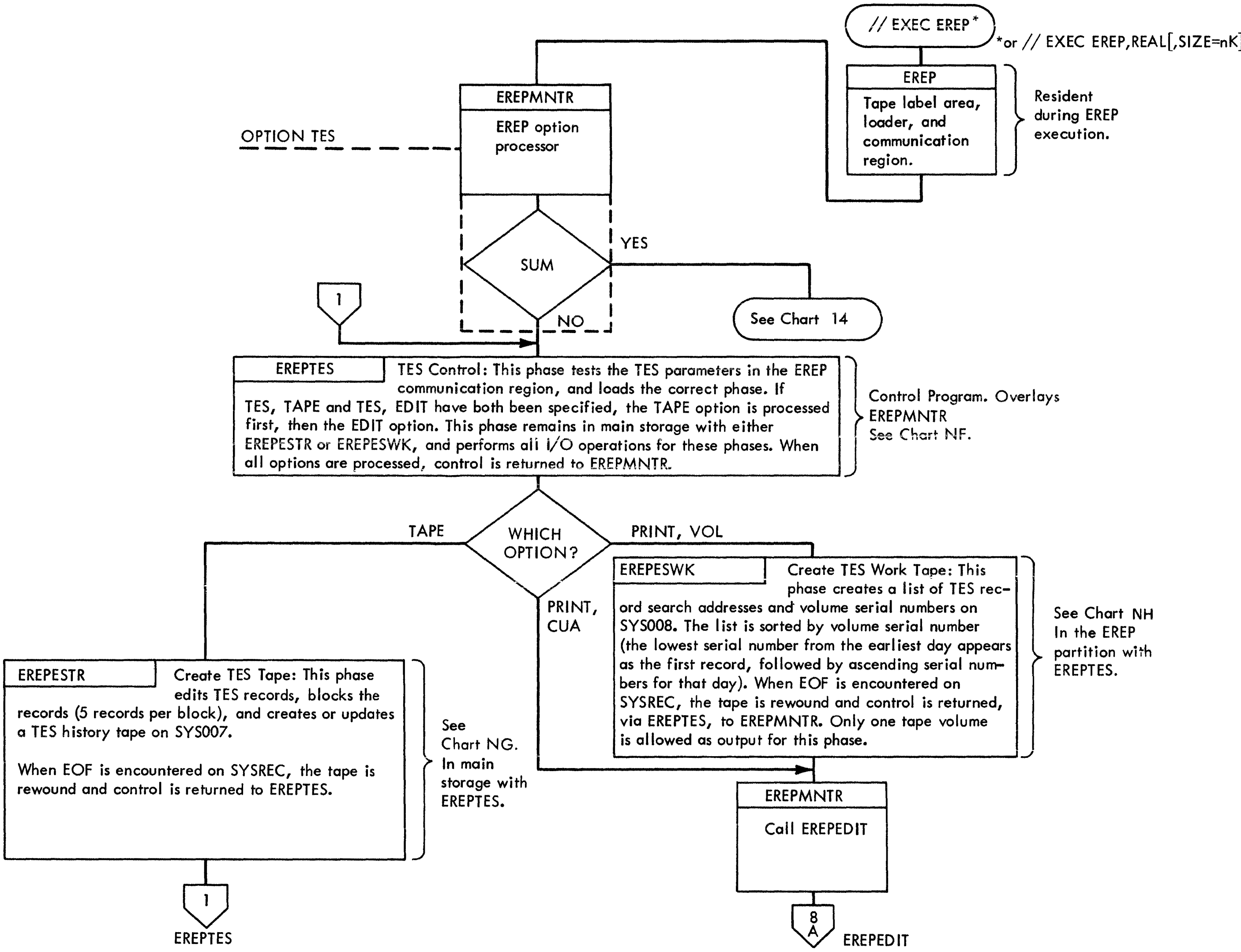


Chart 17. Option HIST: Structure

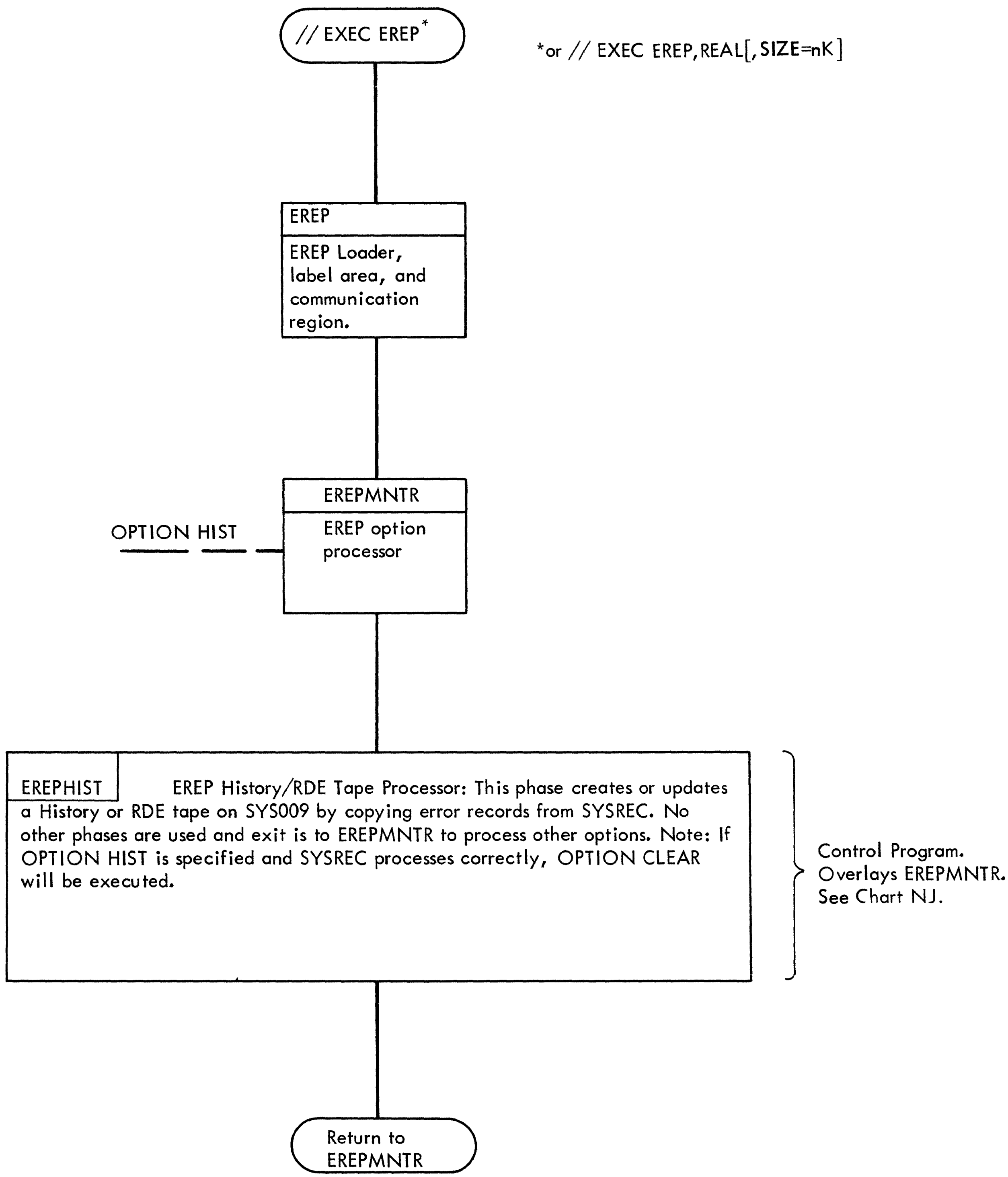


Chart 18. Option RDESUM: Structure

// EXEC EREP*

*or // EXEC EREP,REAL[,SIZE=nK]

EREP

Tape label area, leader, and communication region

EREPMNTR

OPTION RDESUM

EREP option proces processor

EREPRDE

RDE Summary Program : This phase summarizes, edits, and prints information about system activity, as previously recorded on the RDE tape, for a period of 1 to 30 days.
The types of system activity recorded include:

- The date, time, reason, and subsystem responsibility for each IPL during the specified period.
- The starting and ending date of the report.
- The average run time between IPL and EOD (or IPL and IPL if the ROD command was not issued to create an EOD record) for the specified period.
- The number of IPL records that occur in the specified cluster interval.
- The subsystem responsibility and number of times a subsystem caused possible system degradation or system failure.

Input for this phase is the RDE tape mounted on SYS009 and output is to SYSLST. The OPTION card may be entered from SYSLOG or SYSIPT, but control information is entered on SYSLOG in response to prompter messages.

Control processing phase. Overlays EREPMNTR. See charts NK and NL.

Return to EREPMNTR

Chart MA. EREP: Function

*or // EXEC EREP,REAL[,SIZE=nK]

// EXEC EREP*

Displacement from partition start addr.	Label List	
		Label Area: 80 Bytes - Reserved for use by LIOCS during history or RDE tape processing
	EREPSTRT	EREP Loader: 28 Bytes -
	LOADER	1. Set register 0 to -COMEND-. [Entry for loading EREP control program.]
	LOADER2	2. Set the load name address in register 1. [Entry for loading any phase except control program.]
		3. SVC 4, LOAD.
	LOADNAME	EREP
	EREPNAME	MNTR [This label is overlaid to call subsequent phases.]
		EREP Communication Region: 168 Bytes
108	OPTNRETN	4 Bytes [Return address to EREPMNTR.]
112	TRSAVE	4 Bytes [Save area for 2715 address.]
116	TRACKS	2 Bytes [SYSREC cylinder/address.]
118	RFONE	5 Bytes [First SYSREC record address.]
123	RFCURR	5 Bytes [Current SYSREC record address.]
128	QUEUE	16 Bytes [EREP option queue.]
		1 Byte [Reserved.]
145	SUBSWTH	4 Bytes [Option parameter switches.]
149	COMREGN	3 Bytes [General communications.]
152	SUMREGN	[Describes the 15-byte summary area.]
152	SUMPARM	2 Bytes [Summary parameter switches.]
154	SUMSWTH	3 Bytes [Summary options.]
157	SUMSWTH2	[Second byte of summary options.]
157	SUMCPUID	5 Bytes [Summary CPU ID.]
162	SUMSWBYT	1 Byte [Summary Switches]
163	SUMTPCUA	2 Bytes [TP Line done.]
		2 Bytes [Reserved.]
167	SELREGN	[Describes the 52-byte selective retrieval area.]
167	SELPARM	3 Bytes [Select parameter switches.]
170	SELSWTH	2 Bytes [Record type switches.]
172	SELCMRG	[Describes the 47-byte selection area.]
172	SEL2715	1 Byte [2715 switches.]
173	COMDATE	6 Bytes [Date.]
179	COMTIME	4 Bytes [Time.]
183	COMJOB	8 Bytes [Job name.]
191	COMPROG	8 Bytes [Program name.]
199	COMCUA	2 Bytes [CUA.]
201	COMTERM	8 Bytes [Terminal name.]
209	COMCPU	3 Bytes [CPU serial number.]
212	COMDEV	1 Byte [Device type.]
213	COMVOL	6 Bytes [Volume serial number.]
		57 Bytes [Reserved.]
276	COMEND	[Control program load address.]

EREP

SVC 4
Load Phase at
-COMEND-

INPUT

SYSLOG:
Response to message
3E11D

SYSLOG or SYSIPT:
EREP options

PROCESS

1. Establish base registers, relocate CCBs and CCWs, and OPENR SYSLST.
2. Query operator for option statement source. If an invalid source is specified, issue error message and reissue option source message.
3. If N (None) is the response, go to step 18 .
4. Set to read from the option source device (SYSLOG for console input, SYSIPT for card input). (SYSLOG Only)
5. Read option input and check syntax by scanning -OPTNTBLE-. If EOF, go to step 18 .
6. Check for suboptions and post them in the appropriate tables in the EREP communication region.
7. If errors are detected in steps 5 or 6 , issue error message.
8. Post the option in -QUEUE- in the EREP communications region and set default parameters for previous option.
9. Read statement. If option, go to step 6 . If parameter, check syntax in subroutines. If EOF and first execution, go to step 18 . If EOF, subsequent execution, and parameters entered, go to step 25 . If EOF, subsequent execution, and no parameters entered, go to step 22a .

Subroutines:

10. Check TIME parameter.
10a. Check names (job, program, terminal) or volume serial number.
11. Check DATE parameter.
11a. Check device address or CPU serial number.
12. Check DEVICE parameter.
12a. Check FORMAT parameter.
13. Check TYPE or GROUP parameter.
14. Check CPU parameter for SUM.
15. Check 2715 parameter.

MC

OUTPUT

3E11D ENTER OPTION SOURCE, C=CARD, S=CONSOLE, N=NONE

3E25I INVALID RESPONSE

3E14A ENTER OPTION

Set suboption switches at -SUBSWTH-, set -COMREGN-, set -SUMPARM- and -SUMSWTH-, or set -SELPARM- and -SELSWTH-.

3E12I INVALID OPTION

Build option table at -QUEUE- and set parameters.

LABEL LIST	
LABEL	STEP
START	1
RELOC	
GIVSRCE	2
	3
INITLOG	4
READOPTN	5
CHKKEY	
SCANBLK	
CHEKMAIN	
SUBCHEK	6
	7
SETRESET	8
PARMCHK	9
TIME	10
PARMCHAR	10a
DATE	11
PARMHEX	11a
DEVICE	12
FORMAT	12a
TYPE	13
CPU	14
PARM2715	15

MB

INPUT

The SYSREC header label.

PROCESS

16. If errors are detected in step 9-15, issue error message.
17. Go to step 9.
18. Set default options if no options were specified, queue the EDIT option if only CLEAR was specified, or queue the CLEAR option if HIST and not CLEAR was specified.
19. Sort options in priority order (CLEAR last in -QUEUE-).
20. If SYSREC is to be used, suppress error recording, OPENR SYSREC, and read the header record to obtain SYSREC extents.
21. If an I/O error occurs or the header record is invalid, issue an error message and go to step 28. Otherwise, go to step 25.
22. If input from SYSLOG and SELECT or SUM, go to step 9. If TES, process any suboptions.

22a. Get next option from -QUEUE-.

23. If there are no more options, go to step 28.
24. If CLEAR is to be executed, go to step 27.
25. Set the return address in the EREP communications region to step 22. Set the phase name to be called in the loader and issue the option in progress message.
26. Branch to the loader at -LOADER-.
27. Rewrite the SYSREC header label. If an I/O error occurs, issue an error message.
28. Enable recording on SYSREC, then issue SVC14, EOJ.

OUTPUT

3E041 INVALID PARAMETER

Set default options in -QUEUE-.

3E261 INVALID SYSREC FILE HEADER RECORD

3E271 EDITING IN PROGRESS

3E281 HISTORY IN PROGRESS

3E291 CLEAR IN PROGRESS

3E011 SUMMARY IN PROGRESS

3E021 TES IN PROGRESS

3E061 RETRIEVAL IN PROGRESS

3E261 INVALID SYSREC FILE HEADER RECORD

LABEL LIST

LABEL	STEP
PARMCHK	16
	17
DEFAULT	18
SORTER	19
OPENCKER	20
OPENFILE	
PREPROC	21
OPTION	22
OPTION1	23
	24
OPTION2	25
OPTION3	
OPTION4	26
CLRFILE	27
EOJ	28

Chart MD. EREPEDIT: Function

INPUT

The contents of -QUEUE- in the EREP communication region

PROCESS

1. Establish base registers, relocate CCBs and CCWs, set for return to EREPMNTR, set SYSREC extents, and OPENR SYSLST.
2. If a TES edit is to be done, set to load EREPESPT.
3. If selective retrieval is to be done, check for tape input and OPENR SYS009, if present, then set to load EREPRETV.
4. If EDIT is to be performed, check for extents associated with the next record type to be processed (in the table at KEYTBL). If the extents exist, or if this is the initial entry to EREPEDIT, create the phasename of the next or first transient.
5. Load the transient, and branch to the transient entry point.

Subroutines: Are used to format SYSREC records. Their order of execution is controlled by the table called by the transient. See Chart ME for the subroutines.

6. When processing is complete, any mounted tape is rewound and unloaded, and control is returned to EREPMNTR via EREP.

OUTPUT

3E10I I/O ERROR ON RECORDER FILE AT cchhr

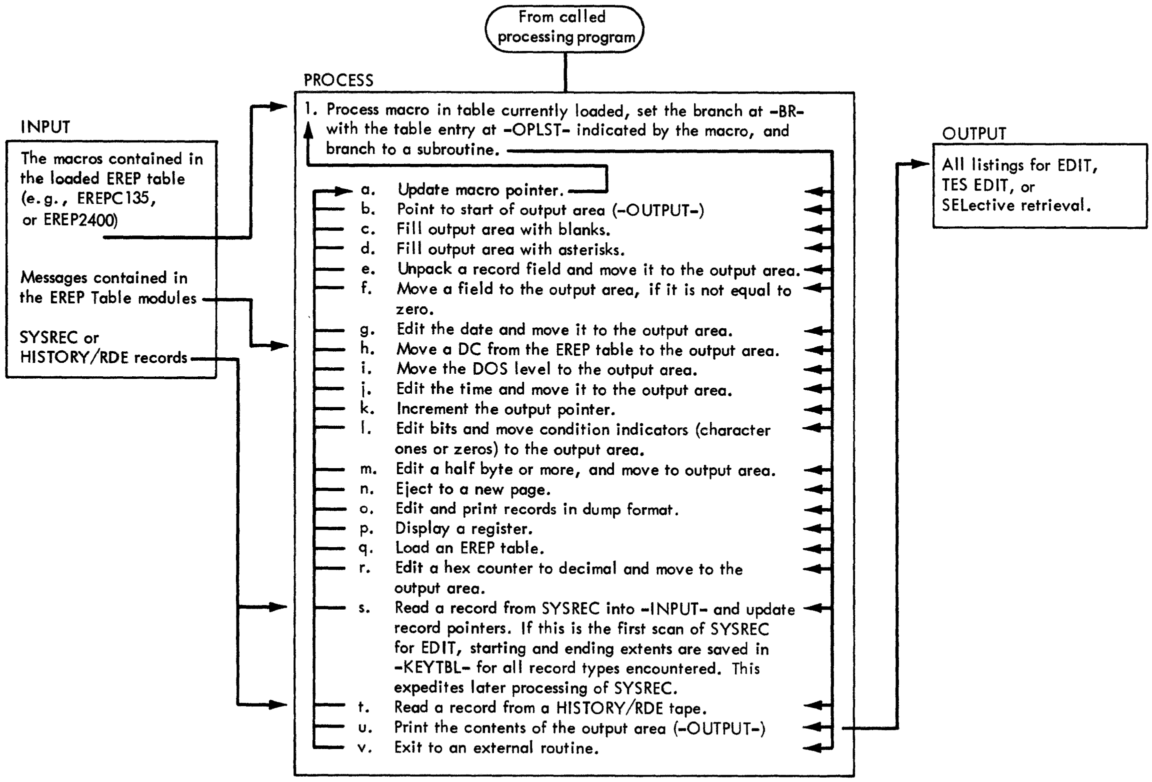
3E22I INCOMPLETE DATA RECORDING AT TIME OF ERROR

3E1I I/O ERROR ON HISTORY/RDE FILE

LABEL LIST

LABEL	STEP
BEGIN	1
INITIAL	2
	3
	4
EDIT	
LOADPHAS	5
LOADOPTN	6

Chart ME. EREPEDIT: Subroutines



LABEL LIST	
LABEL	STEP
LISTPRO	1
LSTRTN	a.
BACK	b.
BLANKS	c.
ASTRK	d.
MOVEHEX	e.
NONZERO	f.
DATE	g.
MOVECOM	h.
DOSLEVEL	i.
TIME	j.
INDEX	k.
EDITBITS	l.
UNPKHEX	m.
CNTROL	n.
DUMP	o.
MOVREG	p.
LOAD	q.
MVCCTR	r.
READLOG	s.
READ	
READISK	
EXNTPROC	
READLOG	t.
READ	
TAPEREAD	
PRINT	u.
EXIT	v.

Chart MF. EREPESPT: Function

INPUT

TES record addresses on SYS008 (created by EREPESWK)

TES records from SYSREC

PROCESS

1. Establish base registers, relocate CCBs and CCWs, and print page heading.
2. Read a block from SYS008 (TES record addresses).
3. Check for end of addresses on SYS009. If there are no more addresses, rewind/unload the tape and return to EREPMNTR.
4. Put search address in disk CCW, and read a TES record from SYSREC.
5. Format the TES record into the output area, and add the proper subheadings.
6. Print the TES record and perform any necessary page skipping and heading operations.
7. Point to the next TES search address, and return to step 4.

OUTPUT

Formatted TES records

LABEL LIST

LABEL	STEP
INITPG	1
BEGIN	
RDTPE	2
RDDISK	3
WIND	
CONTINUE	4
FORMAT	5
DVCK	
LASTLST	
MODN	
LSTFILD	6
SKPG	
RESET	7

Chart MG. EREPRETV: Function

INPUT

Selective retrieval parameters from the EREP communication region.

SYSREC or History tape records.

PROCESS

1. Establish addressability and check for first or subsequent execution. If subsequent execution, go to step 3.

2. If first execution, set to search for the specified parameters by setting EREPRETV subroutine addresses in -PARMSUBS- in the EREPEDIT communications region (-EDITCOM-).

3. Read a record from SYSREC or the History tape using the -READ- subroutine in EREPEDIT. If EOV or EOF is encountered, return to -EDIT- in EREPEDIT.

4. Check if this is a data set (multiple records) describing one error or check if an I/O error occurred. If so, return to step 3.

5. Form processing program name for type of record read, and check if this record is to be processed by BALs using addresses created in step 2. If not, return to step 3.

Subroutines:

6. Check date or time of record.

7. Check CPU used at time of failure.

8. Check program used at time of failure.

9. Check job name at time of failure.

10. Check volume ID at time of failure.

11. Check device type or terminal type.

12. Check device address.

13. Check for a 2715 miscellaneous data record.

14. Load the applicable processing program.

OUTPUT

Table of subroutine addresses in EREPEDIT at -PARMSUBS-.

LABEL LIST

LABEL	STEP
IJBERETV	1
BUILDTBL	
SHIFTSET	2
SELECT	3
SELINCP	4
RTRVLP2	5
PDATE	6
PTIME	
PCPU	7
PROGRAM	8
PJOB	9
VOLID	10
PDVTYP	11
PTERM	
PCUA	12
P2715	13
GETPHAS	14

Chart MH. EREPMCAR and EREPCCH: Function

INPUT

Error records from SYSREC.

PROCESS

1. Establish base registers and set up table area.
2. If selective retrieval is being done, go to step 5.
3. Set the return address to step 8.
4. Read a record using the -READ- subroutine in EREPEDIT. If EOF, return to -EDIT- in EREPEDIT.
5. If this is not an MCAR or CCH record, and the first (if multiple) records define the error, or an I/O error occurred, go to step 4.
6. Skip to a new page and test if the correct table is in main storage. If not, load the correct table; either EREPMCRC or EREPCCHC.
7. Load the list pointer and go to -LISTPROC- in EREPEDIT.
8. If the EOF switch is on, return to -EDIT- in EREPEDIT.
9. If an incomplete data set was encountered during processing with one of the called tables, go to step 5. Otherwise, go to step 4.

OUTPUT

The address of the formatting list in the called table.

LABEL LIST		
EREPMCAR		EREPCCH
LABEL	STEP	LABEL
IJBEMCAR	1	IJBECCH
	2	
	3	
MCARREAD	4	CCHREAD
	5	
MCARPART	6	CCHPART
MCARNDX	7	CCHNDX
MCARRTRN	8	CCHRTRN
	9	

Chart MJ. EREPIPL: Function

INPUT

Error records from SYSREC or History/RDE tape

PROCESS

1. Establish base registers and set up table area.
2. Load EREPEOD (IPL/EOD Table).
3. If selective edit, go to step 7.
4. Set return from table to step 9.
5. Read a record from SYSREC. If EOF, go to -EDIT- in EREPEDIT.
6. If the record is not IPL or EOD, or if an I/O error occurred, go to step 5.
7. If this is the first record to be edited, the page is full, or this is selective retrieval, skip to a new page.
8. Load the list pointer and go to -LISTPROC- in EREPEDIT.
9. If EOF is encountered, go to -EDIT- in EREPEDIT. Otherwise, go to step 5.

OUTPUT

The address of the formatting list in EREPEOD.

LABEL LIST	
LABEL	STEP
IJBEIPL	1
	2
	3
	4
IPLREAD	5
	6
IPLPAGE	7
IPLNDX	8
IPLRTRN	9

INPUT

Error records from SYSREC.

A particular error record from EREPRETV.

A 3420 error record

* Return to this routine from an EREP table is to the label -UNCKRTRN- which then branches to steps 3, 4, or 11 depending on the status of processing.

PROCESS

1. Establish base registers and table area. *Note
2. If selective retrieval is being done, go to step 10.

2a. Set phase load addresses in phase load table to use available main storage.

3. Read a record from SYSREC. If EOV, go to step 11.
4. If this is not a unit check or 3211 MDR record, or the first of multiple records that define the error, return to step 3.
5. Search -DEVTABLE- for supported device type records and pick up group code.
6. If the record is for the device group being processed, go to step 9.
7. If this is not the first scan of SYSREC, go to step 3.
8. Set the search parameter (high or low extent) of the applicable device group in -EXTNTBLE- and go to step 3.
9. If the record does not contain the same channel/unit address as the one being processed, go to step 3.
10. Test the phase load table to see if the needed phase is in main storage. If not, load the phase in the first free area, pointed to by the phase load table, or overlay the lowest priority phase in main storage. Set the list pointer and branch to -LISTPROC- in EREPEDIT.
11. If there are more channel/units to process, initialize to process next address and go to step 3.
12. If there are more device groups to process, initialize to process the next group and go to step 3.
13. Return to EREPEDIT at -EDIT-.
14. If the entry is to process TU-TCU serial information for 3420 tape units, branch to -LISTPROC- after completion.

OUTPUT

The address of the formatting list in EREP3211, EREPSHRT, or EREPUCCM

Edited serial numbers for TU and TCU

LABEL LIST

LABEL	STEP
IJBEUNIT	1
UNCKRTRN	
IJBEUNIT	2
UNCKREAD	3
UNCKSKIP	4
SEARCH	5
EXNTPROC	6
GROUPCK	
BRPROC	7
SETEXTNT	8
BRPROC20	9
RCDPRO	10
EOFRTN	11
NXTGRP	12
LOOP	13

Chart ML. EREP2715: Function

INPUT

Error records from SYSREC.

A particular 2715 error record from EREPRETV

PROCESS

1. Establish base registers, initialize pointers and switches, and the high/low extent table at -EXTENTS-.
2. If this is selective retrieval, go to step 7.
3. Read a record from SYSREC. If EOF encountered, go to step 7.
4. If the record is not 2715, an unrecoverable I/O error occurred, or this is a padded record at the end of a data set, go to step 3.
5. Test the type of 2715 record; set the low extent if it is the first one encountered, or the high extent. The extents are in a list at -EXTENT-.
6. Test to see what portion of the record has been processed. If it has all been processed, go to step 3; if not, go to step 5.
7. If this is selective retrieval, load the appropriate table. If not, EREPADPT is loaded first.

OUTPUT

SYSREC starting and ending addresses of the 7 types of 2715 error records in main storage.

LABEL LIST

LABEL	STEP
START	1
HERE	
HILOSET	
	2
SETXTENT	3
	4
ADPTDISK	5
AREASTAT	
ADPT2740	
ADPT2790	
ADPTMPX	
ADPTBSC	
SPECODE	
NEXTRCRD	6
TRANLOAD	7
LOADIT	

Chart MM. EREPADPT: Function

INPUT

Error records from SYSREC.

A particular error record from EREPRETV.

PROCESS

1. Establish base registers and initialize pointers and switches.
2. If selective retrieval was specified, go to step 5.
3. Read a record from SYSREC.
4. If the record is not for 2715, return to step 3.
5. Loop through the record looking for one type of adapter record on each pass of SYSREC (1. Disk adpt., 2. 2790 adpt., 3. MPX adpt., 4. 2740 adpt., 5. BSC adpt.). When all records of all types are processed, go to step 7.
6. Edit and print the error record for the adapter type currently being processed.
7. If there are special code records to be processed, load EREPSPCL.
8. If there are area station records to be processed, load EREPASTA.
9. Exit to EREPEDIT.

OUTPUT

Adapter error records by type from SYSREC:
1. Disk adapter records,
2. 2790, 3. MPX,
4. 2740, 5. BSC.

LABEL LIST	
LABEL	STEP
START	1
INIT	
	2
RECREAD	3
RDERRCK	4
DSKERROR	5
CTRINCR	
CK2790	
CKMPX	
CK2740	
CKBSC	
NEWPG	6
ADPTHD	
EDITMORE	7
	8
	9

Chart MN. EREPSPCL: Function

INPUT

Error records from SYSREC.

A particular error record from EREPRETV.

PROCESS

1. Establish base registers.
2. Set search extents and clear work area.
3. If selective retrieval, go to step 6.
4. Read a record from SYSREC. If EOF is encountered and there are more records to read, go to step 2; otherwise, go to step 9.
5. If it is not a 2715 record, go to step 4.
6. Loop through 2715 record looking for a valid special code record with a valid area station number. If there are no valid records, or all 4-byte error records are done, go to step 4.
7. Sort and print the 2715 special records on SYSLST.
8. Go to step 6.
9. If there are area station records to be processed, load EREPASTA; otherwise, branch to EREPEDIT.

OUTPUT

Special code 2715 error records, sorted by area station addr., channel/unit addr., and request code, on SYSREC.

LABEL LIST

LABEL	STEP
START	1
READREC	2
	3
READRCRD	4
CK2715ID	5
NEXTRCRD	6
CKDONE	7
PRINTIT	
COMPAS	
COMCUA	
PRNTDATA	
NEXTRC	8
CKASTAT	9

Chart MP. EREPASTA: Function

INPUT

Error records from SYSREC

A particular error record from EREPRETV

PROCESS

1. Establish base registers.
2. Set search addresses, clear work area. If this is selective retrieval, go to step 5.
3. Read a record from SYSREC. If EOF is encountered and there are more area stations to process, go to step 2; otherwise, go to step 8.
4. If this is not a 2715 record, return to step 3.
5. Loop through 2715 record looking for unprocessed area station records. If there are no valid area station records or all records have been processed, go to step 3.
6. Sort and print the area station records on SYSLST.
7. Go to step 5.
8. Branch to EREPEDIT.

OUTPUT

Listing of 2715 area station records, sorted by area station and device address, on SYSLST.

LABEL LIST

LABEL	STEP
START	1
READRCRD	2
RDREC	3
CK2715ID	4
NEXTRCRD	5
SAVEDATA	6
PRINTOUT	
PRNTDATA	
NEXTRC	7
EXIT	8

Chart MQ. EREPSMCP: Function

INPUT

The parameters in the EREP communications region (-SUMREGN-)

SYSREC records

PROCESS

1. Establish base register, relocate CCBs and CCWs, and set EREP to call EREPEDCP.

2. Test -SUMSWTH- for options. If 2715 is to be summarized, load EREPSMTR and branch to the transient when a record is encountered.* If 2715 is not specified or has been processed, load as many options as will fit in the partition.

3. Read a record and test for a valid record type (unit check, TP, misc., MCAR, or CCH). If none of these, read another record.

4. If the record read was unit check, test -DVCTABLE- for a valid device type. If the type is invalid, return to step 3.

5. If the transient, associated with the record just read, is in core, branch to that routine via -COUNT-. Return from the transient is to this routine, then to step 3.

6. When the last SYSREC record has been processed, branch to EREP to load EREPEDCP.

OUTPUT

3E101 I/O ERROR ON RECORDER FILE AT cchhr

LABEL LIST	
LABEL	STEP
SUMSTART	1
OPTNNODA	2
CLEARCTR	
OPTNFND	
SCAN	3
UPHDCYL	
ERROR	
DVCSCAN	4
COUNT	5
EDITLOAD	6

*Note: All 2715 addresses are processed, with multiple executions of EREPSMCP and EREPEDCP, if necessary, before other options are processed. 2715 summary will always use 10K, even if more room is available.

Chart MR. EREPSMTR: Function

INPUT

2715 SYSREC records provided by EREPSMCP.

PROCESS

1. Establish base registers and set lowest record to be processed.

2. Save channel/unit address, 2715 ID, area station and device address. Note: If all records on SYSREC have been processed, EREPSMCP will not return to this phase.

3. If this record has a valid error code, go to step 6.

4. Point to next error record. If all records from the current SYSREC read have been processed, branch to EREPSMCP at -SCAN- to read next record.

5. Go to step 2.

6. Check if this record is processed. If so, go to step 4.

7. Sort the current record into the list of 2715 records that starts at EREPSMCP end.

8. If a record already exists in the list with the same ID, add the error counters from the new record to the totals in the existing error record.

9. If the list space is exceeded, set -SUMSWTH- to request another execution of EREPSMTR and save the last ID combination in -TRSAVE- in the EREP communication region.

10. Return to step 4.

OUTPUT

List of summarized 2715 records in main storage. (In the 3400 bytes preceding this phase.)

Set -SUMSWTH- and -TRSAVE- to indicate additional scan of SYSREC required to process all 2715 records.

LABEL LIST	
LABEL	STEP
START	1
BUILDID	2
	3
UPDRCTPR	4
	5
STRTADDR	6
GPACTIVE	7
UPDATCTR	8
SETNLCOM	9
SETSCNSW	
	10

Chart MS. EREPSM - TD, DA, UR, CR, DK, CU, D1, T1, and U1: Function

INPUT

SYSREC records for the devices supported by this phase.

PROCESS

1. Set base register.
2. Get device type and set the list pointer.
3. Set device ID, date, and counter space.
4. If there is no space left to add additional counters, exit to EREPSMCP to read the next record.
5. Test for an ID match in the counter list. If there is a match, update the existing counter. If not, add this entry to the end of the list.
6. Go read another record via -SCAN- in EREPSMCP.

OUTPUT

List of summarized error records, by channel/unit address, in main storage. (This list is in the 3400 bytes preceding this phase.)

LABEL LIST

LABEL	STEP
SMxxSTRT*	1
DVCSCAN	2
DVCSUPRT	3
CTRLOC	4
	5
UPDTLOOP	
UPDTFWRD	
UPDTWRTN	6

* xx is the same as the last two letters of the phasename.

Chart MT. EREPSMTP: Function

INPUT

Non-2715 teleprocessing records from SYSREC via -SCAN- in EREPSMCP

PROCESS

1. Establish base registers, build record ID, and point to date.
2. If this record was processed on a previous pass, return to -SCAN- in EREPSMCP to read another record.
3. Set date.
4. Scan the counter list for an entry with the same ID. If there is, update the counters and return to -SCAN- in EREPSMCP to read another record.
5. If the counter list does not contain a match for the record read, sort the new record into the list.
6. If the counter list overflows, set -SUMSWTH- to recall EREPSMTP to process the remaining records.
7. Branch to -SCAN- in EREPSMCP.

OUTPUT

Sorted list of summarized non-2715 teleprocessing records in main storage. (This list is in the 3400 bytes preceding this phase.)

LABEL LIST

LABEL	STEP
SMTPSTRT	1
	2
DATE00	3
SCAN00	4
CNT05	
UPDT	
UPDTHWNO	
UPDTFWRD	
VUPDT	
SCAN00	5
SCAN20	
MOVE	
CNT00	
UPDT	
UPDTHWNO	
UPDTFWRD	
VUPDT	6

Chart MU. EREPEDCP: Function

INPUT

The load addresses left in the EREP partition by EREPSMCP.

The accumulated error records left in the summary processing phase areas of the EREP partition.

PROCESS

1. Establish base registers and OPENR SYSLST.
2. Point to all load addresses that previously contained called summary processing phases.
3. Pick up first (next) address to load edit phase. If there are no remaining addresses, set to return to EREPSMCP to process additional options or to EREPMNTR if all options have been processed.
4. Set return, from the transient, to step 3, then load and branch to an edit phase.

Subroutines: Used by the called processing phase

5. Process a macro in the called processing phase, set the branch at -BR- with the table entry at -OPLST- indicated by the macro, and branch to a subroutine.
6. Update the macro pointer.
7. Point to start of output area (-OUTPUT-).
8. Fill output area with blanks.
9. Fill output area with asterisks.
10. Move a DC from the processing phase to the output area.
11. Print the contents of the output area.
12. Exit to an external routine.
13. Increment output pointer.
14. Eject to a new page.
15. Edit a hexadecimal counter to decimal and put in output area.

OUTPUT

All summary listings.

LABEL LIST	
LABEL	STEP
EDCPSTRT	1
EDIT05	2
EDIT10	3
EDITL00	4
LISTPRO	5
LSTRTN	6
BACK	7
BLANKS	8
ASTRK	9
MOVECOM	10
PRINT	11
PRROUT	
EXIT	12
INDEX	13
CNTROL	14
NEWPAGE	
MVCCTR	15

Chart MV. EREPEDTR: Function

INPUT

List of 2715 area station counters left in the EREP partition (the 3400 bytes preceding EREPEDTR) by EREPSMTR.

PROCESS

1. Establish base registers, save return address, get number of error counters, and point to counter list.
2. If next counter group is not active, go to step 7.
3. If the area station is F0 or F1, print error total counter only.
4. If the area station number is higher than F1 (an invalid address), update counter and go to step 2.
5. Edit and print the 9 area station error counters. If this is a new page, also print page headings.
6. If all counters have not been printed, return to step 2.
7. Exit to EREPSMCP.

OUTPUT

Error counters, by area station, on SYSLST.

LABEL LIST	
LABEL	STEP
START	1
NXTBLOCK	2
	3
	4
BALCVDEC	5
CNVTODEC	
PRINTIT	
PRT	
INCRCGPT	6
EXIT	7

Chart MW. EREPEDDA and EREPEDTD: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding EREPEDDA or EREPEDTD) left by the applicable summary phase.

PROCESS

1. Establish addressability and set exit address.
2. Point to device table and counter area.
3. Scan device table for active entry. If none, exit to EREPEDCP.
4. Set the edit bit in the device table and point to the counter table.
5. Unpack the device address and date into the header.
6. Point to the LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

LABEL LIST	
LABEL	STEP
EDxxSTRT*	1
DASD00	2
DASD05	3
DASD10	
DAED00	4
	5
	6

* xx is DA for EREPEDDA or TD for EREPEDTD.

Chart MX. EREPEDCR, EREPEDDK, and EREPEDUR: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by the applicable summary phase.

PROCESS

1. Establish addressability and set exit address.
2. Point to device table and counter area.
3. Unpack date into page heading.
4. Scan device table for active entry. If none, exit to EREPEDCP.
5. Set edit bit in device table, point to counter group for this device, and unpack device address.
6. If list is complete, go to step 2.
7. Scan for device type, then set device type and message for use during printing.
8. Point to the LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

LABEL LIST	
LABEL	STEP
EDxxSTRT*	1
DASD00	2
	3
DASD05	4
DASD10	
DAED00	5
DAED05	6
	7
DAED10	
DAED15	
DAED20	8

*xx is CR for EREPEDCR, DK for EREPEDDK, or UR for EREPEDUR.

Chart MY. EREPEDTP: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by EREPSMTP.

PROCESS

1. Establish addressability and set exit address.
2. Set file dates for output. If none, print no-record message and exit to EREPEDCP.
3. Set date in page heading and set bit to indicate terminal done.
4. Scan for active device entry. If none, exit to EREPSMCP.
5. Unpack device address for output; indicate counter group edited.
6. Point to correct LIST table, then exit to -LISTPROC- in EREPEDCP to process counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

LABEL LIST	
LABEL	STEP
EDTPSTRT	1
	2
	3
SCAN00	4
EXIT00	
EDIT00	5
EDIT03	6
EDIT05	

Chart MZ. EREPEDCU: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by EREPSMCU.

PROCESS

1. Establish addressability and set exit address.
2. Point to device table and counter area.
3. Unpack date into page heading.
4. If there are MCAR records, point to the correct LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.
5. Set to process CCH records. If there are no active entries, exit to EREPEDCP.
6. Unpack channel address and set edit bit in counter.
7. Point to correct LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.

OUTPUT

Error totals, by channel for CCH records, on SYSLST.

LABEL LIST

LABEL	STEP
EDCUSTRT	1
	2
	3
	4
CCH00	
CCH10	5
EXIT	
CCH20	6
CCH30	7

Chart NA. EREPEDD1: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by EREPSMD1.

PROCESS

1. Establish addressability and set exit address.
2. Point to device table and correct counter area.
3. Unpack date into page heading.
4. Scan for active device entries. If none, exit to EREPEDCP.
5. Unpack device address and calculate read errors/mega byte read.
6. Point to the LIST table, then exit to -LISTPROC- in EREPEDCP to process the device counters.
7. Scan for errors by volume ID on the device being processed. If none, go to step 4.
8. Calculate read errors/mega byte read.
9. Point to the LIST table, then exit to -LISTPROC- in EREPEDCP to process the volume counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

Error totals by volume ID to the limit of the available counters, then error totals for all other volumes.

*

LABEL LIST	
LABEL	STEP
EDD1STRT	1
DVC00	2
	3
DVC05	4
EXIT	
EDIT00	5
EDIT10	6
EDITV00	7
EDITV05	8
EDITV15	
EDITV07	9
EDITV25	

* For a 3330 the 'device address' is actually the physical address. This means that:
260 = drive A
263 = drive D
etc.

Chart NB. EREPEDT1 and EREPEDX1: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by EREPSMT1.

PROCESS

1. Establish addressability and set exit address.
2. Point to device table and counter area.
3. Unpack date into page heading.
4. Scan device table for active entry. If none, exit to EREPEDX1 from EREPEDT1, or EREPEDCP from EREPEDX1.
5. Set edit bit in device table and point to counter group for this device.
6. Unpack device address and set device type.
7. Point to the LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

LABEL LIST	
LABEL	STEP
EDxxSTRT*	1
DATE00	2
	3
SCAN00	4
EXIT	
EDIT00	5
	6
	7

* xx is T1 for EREPEDT1 or X1 for EREPEDX1.

Chart NC. EREPEDU1: Function

INPUT

List of error counters in the EREP partition (the 3400 bytes preceding this phase) left by EREPSMU1.

PROCESS

1. Establish addressability and set exit address.
2. Unpack date into page heading.
3. Point to device table and correct counter area.
4. Scan for active entries. If none, exit to EREPEDCP.
5. Test for valid device types (3211, 3505, 3525, or 3525). If the device is none of these, go to step 3.
6. Point to correct LIST table, then exit to -LISTPROC- in EREPEDCP to process the counters.

OUTPUT

Error totals by device address (channel and unit) on SYSLST.

LABEL LIST	
LABEL	STEP
EDU1STRT	1
	2
SCAN00	3
SCAN05	4
EXIT00	
EDIT00	5
EDIT10	6
EDIT15	
EDIT20	
EDIT25	

Chart ND. EREPSMES: Function

INPUT

The input source and sort type from -SUMSWTH2- in the EREP communication region.

TES records from SYSREC or SYS009 (history tape).

PROCESS

1. Establish base registers, relocate CCBs and CCWs, and establish input type (if first pass and tape input, OPENR SYS009).
2. Set to utilize all available partition space for statistics accumulators.
3. Read a record from the input device (SYSREC or SYS009).
4. Check for a valid record type. If not, return to step 3.
5. If summary is by volume serial number by day and a new date is encountered, exit to load EREPEDES. If EOF or EOV is encountered, load EREPEDES.
6. Either add the record to an existing counter or place the record, as a new counter, in its proper place. If overflow occurs in the summary work area, save last counter ID and set -SUMSWBYT- to get second execution of EREPSMES.
7. Return to step 3.

OUTPUT

In-EREP-partition counters containing TES summaries by volume serial number by day, or by channel and unit address.

LABEL LIST

LABEL	STEP
SMESSTRT	1
RLDMINUS	
INIT00	
INIT20	
INIT10	2
READ	3
DISKREAD	
TAPEREAD	
READADDR	4
READ00	5
TAPE10	
DISK10	
SCAN	6
UPDTLOOP	

Chart NE. EREPEDES: Function

INPUT

Counter group type from -SUMSWTH2- in the EREP communication region.

In-EREP-partition counters containing TES summaries.

PROCESS

1. Establish base registers and OPENR SYSLST.
2. Calculate number of counters.
3. Point to the first/next counter. When all counters have been edited, exit to EREPSMES, if more TES records are to be processed, or to EREPMNTR when all TES records have been processed.
4. Determine counter type (cuu or vol. ser. no.) by testing -SUMSWTH2- and set to print that type of record.
5. Perform necessary page skipping and header printing.
6. Edit and print a record using the applicable subroutines; then return to step 3.

Subroutines:

7. Set the branch at -BR- with the table entry at -OPLST- indicated by a LIST macro, and branch to a subroutine.
8. Update the macro pointer.
9. Point to start of output area (-OUTPUT-).
10. Fill output area with blanks.
11. Fill output area with asterisks.
12. Move a DC from the processing phase to the output area.
13. Print the contents of the output area.
14. Exit to an external routine.
15. Increment output pointer.
16. Eject to a new page.
17. Edit a hexadecimal counter to decimal and put in output area.

OUTPUT

Formatted TES records on SYSLST.

LABEL LIST

LABEL	STEP
EDESSTRT	1
	2
SCAN	3
EXIT00	
EDIT	4
EDIT10	5
EDIT25	
EDIT05	6
EDIT20	
LISTPROC	7
LSTRTN	8
BACK	9
BLANKS	10
ASTRK	11
MOVECOM	12
PRINT	13
PRROUT	
EXIT	14
INDEX	15
CNTROL	16
NEWPAGE	
MVCCTR	17

Chart NF. EREPTES: Function

INPUT

The contents of -SUBSWTH- in the EREP communication region.

Responses of END/ENTER or CANCEL to TES messages

PROCESS

1. Establish base registers, set transient (EREPESTR or EREPESWK) load address to partition +4088, and relocate CCBs and CCWs.

2. Check -SUBSWTH- for the option type; if it is TAPE, load EREPESTR; if it is PRINT, VOL, load EREPESWK.

3. When all options are processed, -EREPNAME- is changed to MNTR and a branch is taken to -LOADER-.

Subroutines: Used by the called transient.

4. Read record from SYSREC, update cylinder and head addresses, and check for I/O errors.

5. Write TES tape records and check for I/O errors.

6. Write disk address records for use by PRINT, VOL, and check for I/O errors.

7. Print all TES messages.

8. Check operator response to those messages.

OUTPUT

3E07D UNEXPIRED TAPE FILE ON SYS009

3E15A TAPE FULL, MOUNT NEW TAPE

3E31A WRONG TAPE, MOUNT CORRECT TAPE

3E251 INVALID RESPONSE.

LABEL LIST

LABEL	STEP
BEGIN	1
INITIAL	
RELOC	
OPTCHK	2
CKEDIT	
EXIT	3
READREC	4
RECUP	
WRITE	5
LABLROUT	6
MESSAGE	7
CHKRESP	8
NEXTOPT	

Chart NG. EREPESTR: Function

INPUT

SYSREC records read by EREPTES.

PROCESS

1. Set base register and issue mount message.
2. Test -SUBSWTH- for a new tape. If it is, go to step 4.
3. OPENR the update tape (SYS007), check for valid trailer labels, then position the tape to write records over the trailer labels. Go to step 6.
4. OPENR the new tape (SYS007) and write the header labels.
5. Test if this is a continuation; if so, go to step 7.
6. Read a disk record via -READREC- in EREPTES.
7. Test if record is other than TES; if so, return to step 6.
8. Format the TES record and write it on SYS007 via -WRITE- in EREPTES, if 5 records (a block) have been read.
9. Test for EOF on SYSREC or EOV on SYS007. If neither, go to step 6. If EOV on SYS007, go to step 11.
10. Write records in output area via -WRITE- in EREPTES.
11. Write trailer labels and rewind/unload SYS007. If this was not EOF on SYSREC, return to step 4.
12. Branch to -OPTNCHK- in EREPTES to check for additional options.

OUTPUT

3E09A MOUNT TAPE FOR TES PROCESSING (issued via -MESSAGE- in EREPESTV).

Messages 3E15A, 3E31A, and 3E251, via -MESSAGE- in EREPTES, as applicable.

LABEL LIST

LABEL	STEP
START	1
INITAPE	2
GOODTAPE	3
WRONGTP	
NEWTAPE	4
SET	5
ONCE	6
CHK	7
FORMAT	8
	9
RECUP*	
ENDH	10
TRAILERS	11
NOTEND	12

*Routine is in EREPTES

INPUT

SYSREC records

PROCESS

1. Establish base registers, relocate CCBs and CCWs, and issue mount message.
2. Check for an expired tape. If the tape is not expired, issue message and rewind/unload tape. If a new tape is mounted, start the checking procedure again. If the original tape is readied, use it.
3. Read a record from SYSREC using subroutines in EREPTES.
4. If EOF is encountered, and (1) no TES records have been read, turn off the PRINT, VOL option and return to EREPMNTR, or (2) if TES records have been read, write the contents of the ESWK work area on tape.* If EOV on SYS008 is encountered, print msg. then exit to EREPMNTR.
5. If a record is read and it is not a TES record (if a X'50', X'52', or X'53' is not at record + 52), return to step 3.
6. Sort the TES record into -TABLE- and return to step 3.

* If all TES record addresses have not been written on the work tape, the highest Vol serial number and the last seek/search address are saved, and the sort is restarted (return to step 3) looking for additional records.

OUTPUT

3E08A MOUNT SCRATCH TAPE ON SYS008.

3E07D UNEXPIRED TAPE FILE ON SYS008.

TES WORK TAPE

3E20I LAST VOLUME SERIAL PROCESSED IS xxxxxx.

LABEL LIST

LABEL	STEP
INITIAL	1
BEGIN	
CHKDATE	2
MSG	
ONCE	3
READ	
END	4
OPTOFF	
LOOP	
ERROR4	
CHK	5
GOOD	6
FRSTCHK	
FOLLOW	
COMPARE	
SCAN	
INSERT	
MOVE	

Chart NJ. EREPHIST: Function

INPUT

END/ENTER in response to any message.

If CANCEL is entered in response to any message.

LABEL LIST
CHKRESP

PROCESS

1. Set the base register and relocate the CCBs and CCWs.
2. Issue the mount message.
3. Test -SUBSWTH- for a new tape. If it is, go to step 5.
4. OPENR the update tape (SYS009), check for valid trailer labels, then position the tape to write records over the trailer labels. Go to step 6.
5. OPENR the new tape (SYS009) and write the header label.
6. Set the search address for SYSREC, then read a record.
7. Update search address, then write the record on SYS009.
8. Test for EOF on SYSREC or EOV on SYS009. If neither, return to step 6.
9. Write trailer labels.
10. If this is EOV on SYS009 and all SYSREC has not been processed, issue tape full message and return to step 5.
11. If this is EOF on SYSREC and a second tape is to be processed, issue mount second tape message and return to step 3.
12. If this is EOF on SYSREC and all tapes have been processed, set -EREPNAME- in the loader to MNTR, then branch to -LOADER-.
13. Reset CLEAR option switch at -COMREGN- and go to step 12.

OUTPUT

3E18A MOUNT HISTORY/RDE TAPE.

3E31A WRONG TAPE, MOUNT CORRECT TAPE.

3E251 INVALID RESPONSE.

LABEL LIST
MESSAGE

3E15A TAPE FULL, MOUNT NEW TAPE.

3E30A MOUNT SECOND HISTORY/RDE TAPE.

LABEL LIST

LABEL	STEP
BEGIN	1
RELOC	
INITAPE	2
TAPEIO	3
	4
GOODTAPE	
WRONGTP	
NEWTAPE	5
HEADER2	
PROCESS	6
READ	
COMPUTE	7
WRITE	
	8
TRAILERS	9
CONTINUE	10
NOTEND	
ERROR4	
HISTCK	11
WAIT	
EXIT	12
EXIT1	13

Chart NK. EREPRDE: Function (Part 1 of 2)

INPUT

Responses to HISTORY/RDE mount message and enter parameters message

IPL, EOD, MCH, CCH, and unit check records from the RDE tape on SYS009

PROCESS

1. Set base registers, get SYSLST line count, relocate ADCONS, and OPENR SYSLST.
2. Issue mount RDE tape message and read reply. If reply is cancel, go to step 19
3. Write enter parameters message and read reply. If all parameters are defaulted, go to step 5
4. Check for valid parameters. If invalid parameter entered, write error message and return to step 3
5. Set defaults for unentered parameters, print parameters being used on SYSLST, and pass parameters to processing portion of RDESUM program.
6. OPENR SYS009 and clear tables overlayed by initialization code.
7. Read a tape record.
8. Convert the record date to seconds, relative to 1960, and check that is within the specified limits. If not, return to step 7
9. If 16 sequence errors occur, go to step 19
10. Test the key to determine the type of record. If the record is not IPL, EOD, MCH, CCH or unit check return to step 7
11. If the record is IPL, print any necessary headings, print IPL records, and update cluster table if clustering requested.
12. If the cluster table space is exceeded, issue an error message on SYSLST.

OUTPUT

3E32A MOUNT TAPE FOR RDE SUMMARY

3E33A ENTER RDE SUMMARY PARAMETERS

3E34I INVALID START DATE

3E35I INVALID END DATE

3E36I INVALID CLUSTER INTERVAL

3E42I START DATE-YYDDD, END DATE-YYDDD, CLUSTER INTERVAL-M

3E40I END DATE TOO HIGH, LAST RECORD'S DATE-YYDDD

3E41I START DATE TOO HIGH, LAST RECORD'S DATE-YYDDD

3E37I EXCEEDED 16 SEQUENCE ERRORS

IPL RECORDS ON SYSLST

3E43I WARNING-IPL TABLE SPACE EXEEDED

LABEL LIST

LABEL	STEP
BEGIN	1
INIT00	
RDLOG00	2
	3
RDLOG15	4
RDLOG85	5
STARTRUN	6
ZEROTAB	
READ	7
CONVDATE	8
CONVTIME	9
CMPTIME	9
INCRSEQ	
RCTYP	10
IPL	11
GENPRINT	
RDEPHZ	
LOOKDESC	
LOOKSUB	
CONVDEV	

Chart NL. EREPRDE: Function (Part 2 of 2)

PROCESS

13. If the record is MCH, CCH, or OBR, determine if it is a System Incident or System Recovery Incident and update recording table.

14. If the record is EOD, calculate run time (IPL to EOD).

15. If end date is exceeded or EOF is encountered, print end date of report, calculate and print average run time (IPL to EOD), and print IPLs within clustering interval. If no IPLs processed, issue message.

16. If the number of IPL cluster intervals calculated exceeds the available table space, issue an error message.

17. Edit and sort SI/SRI table entries, then print number of SI/SRI for the CPU, real storage, channels, and subsystems.

18. If the number of subsystems encountered exceeds the available table space, issue an error message on SYSLST.

19. CLOSER SYS009 and rewind the tape, then CLOSER SYSLST and return to EREPMNTR via EREP.

OUTPUT

Average run time, report end date, and IPLs within specified cluster interval.

3E381 NO IPL RECORDS PROCESSED

3E431 WARNING-SEQUENCE TABLE SPACE EXCEEDED

System Incidents and System Recovery Incidents by Subsystem.

3E431 WARNING-SYSTEM/SUBSYSTEM TABLE SPACE EXCEEDED.

LABEL LIST

LABEL	STEP
MCH	13
CCH	
OBR	
EOD	14
WRAPUP	15
GRUPHD	
P2EJ1	17
PHASE2ND	19

3203, 3211, AND 5203 PRINTER SUPPORT PROGRAMS

Two programs are supplied to load UCB (Universal Character Set Buffer) and FCB (Forms Control Buffer) of the 3203, 3211, and 5203 printers. These programs are:

- $$BUFLDR (with subphases $$BUFLD1 and $$BUFLD2), which loads the UCB and FCB with standard control information during IPL
- SYSBUFLD, which can be used to load the UCB and FCB with the required control information between jobs or job steps any time after IPL.

Note: For a 5203 printer without the universal character set feature, only the FCB is loaded.

$$BUFLDR is executed as part of the IPL procedure. If 3203, 3211, or 5203 printers are attached to the system, $$BUFLDR together with the corresponding standard buffer load phases must be available in the core image library. The standard buffer load phases for the individual printers are shown in the following table:

PRINTER	To load FCB	To load UCB
3203	$$BFCB3	$$BUCB3
3211	$$BFCB	$$BUCB
5203	$$BFCB5	$$BUCB5*
*Not required for a 5203 without UCS feature.		

$$BUFLDR is loaded by phase $IPLRT4. It scans the PUB table for 3203, 3211, and 5203 printer entries. When it finds an entry, it loads the UCB and calls phase $$BUFLD1. This phase loads the FCB of the printer and returns control to $$BUFLDR. $$BUFLDR then returns to scanning the PUB table for further printer entries. When all buffers have been loaded, $$BUFLDR returns to $IPLRT4.

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

2245 PRINTER SUPPORT PROGRAMS

One program, SYSBUFLD, is supplied to load the FCB (Forms Control Buffer) of the 2245 printer. SYSBUFLD is designed to execute as a job step within the user job stream. The EREP listing (printer output) for the 2245 printer is the same as that for unsupported devices.

Chart 19. Printer Buffer Load

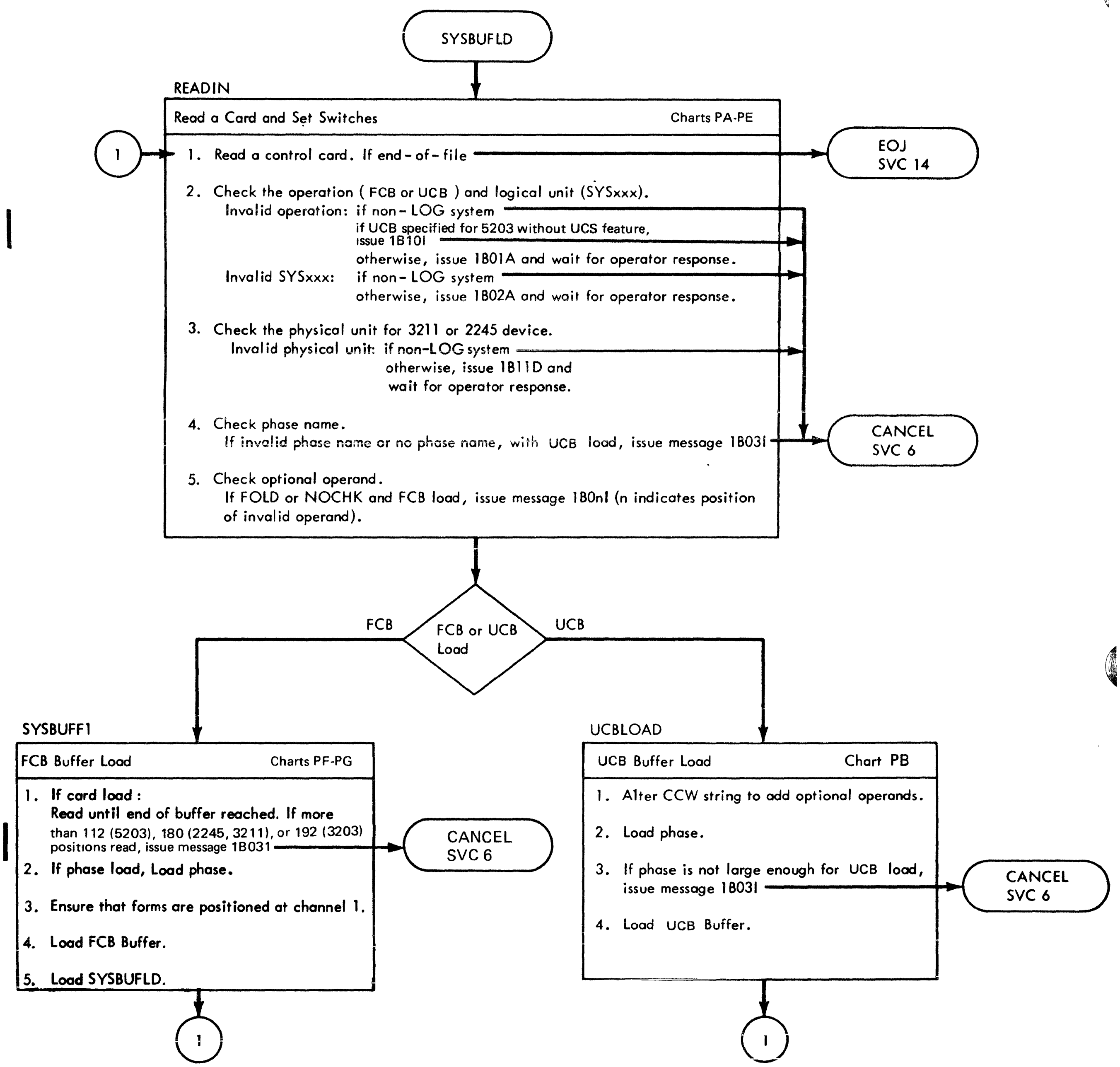


Chart 20. 3203, 3211, and 5203 Printer IPL Buffer Load Programs

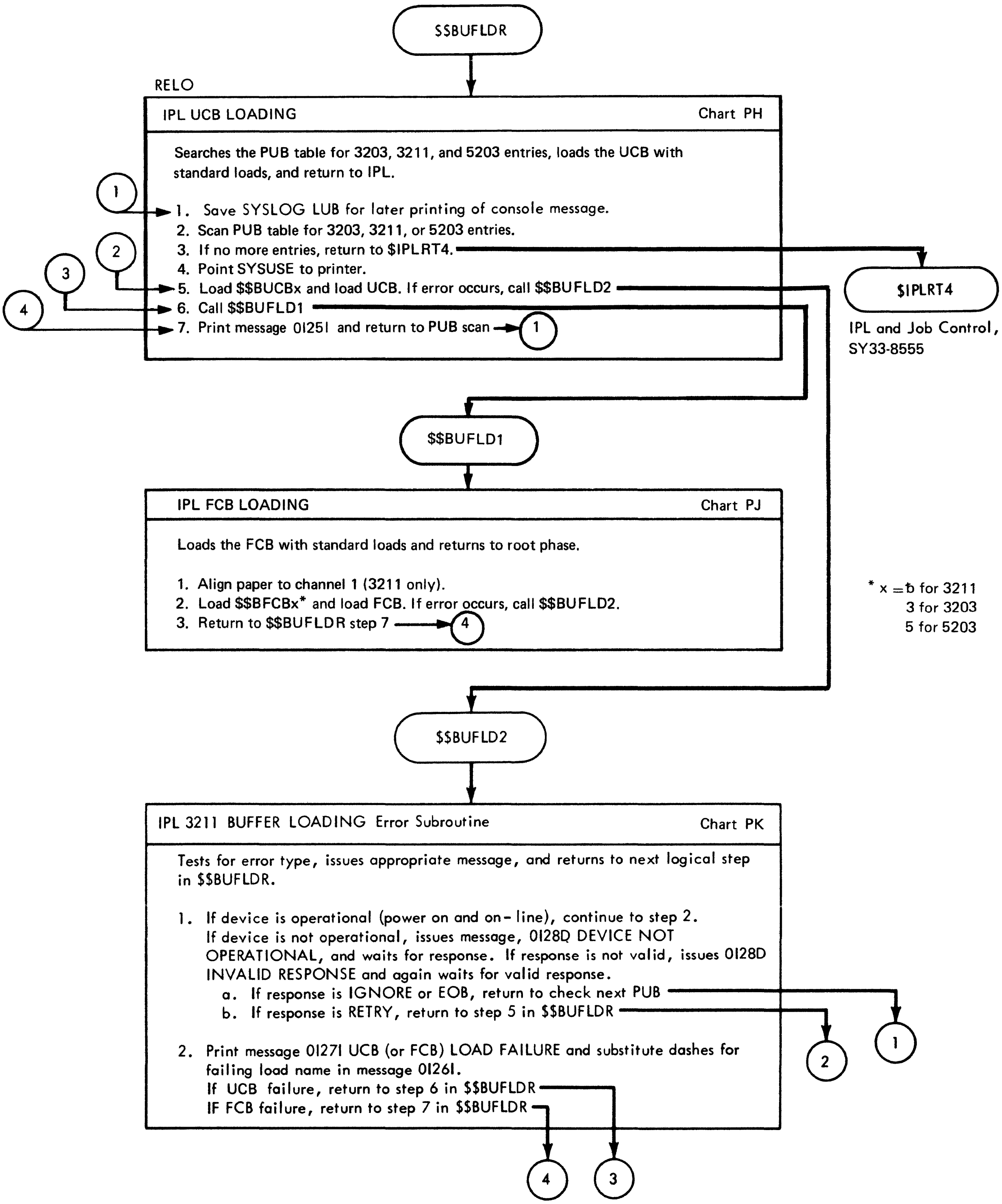


Chart PA. SYSBUFLD - Read Card and Print on SYSLOG and SYSLST
Refer to Chart 19.

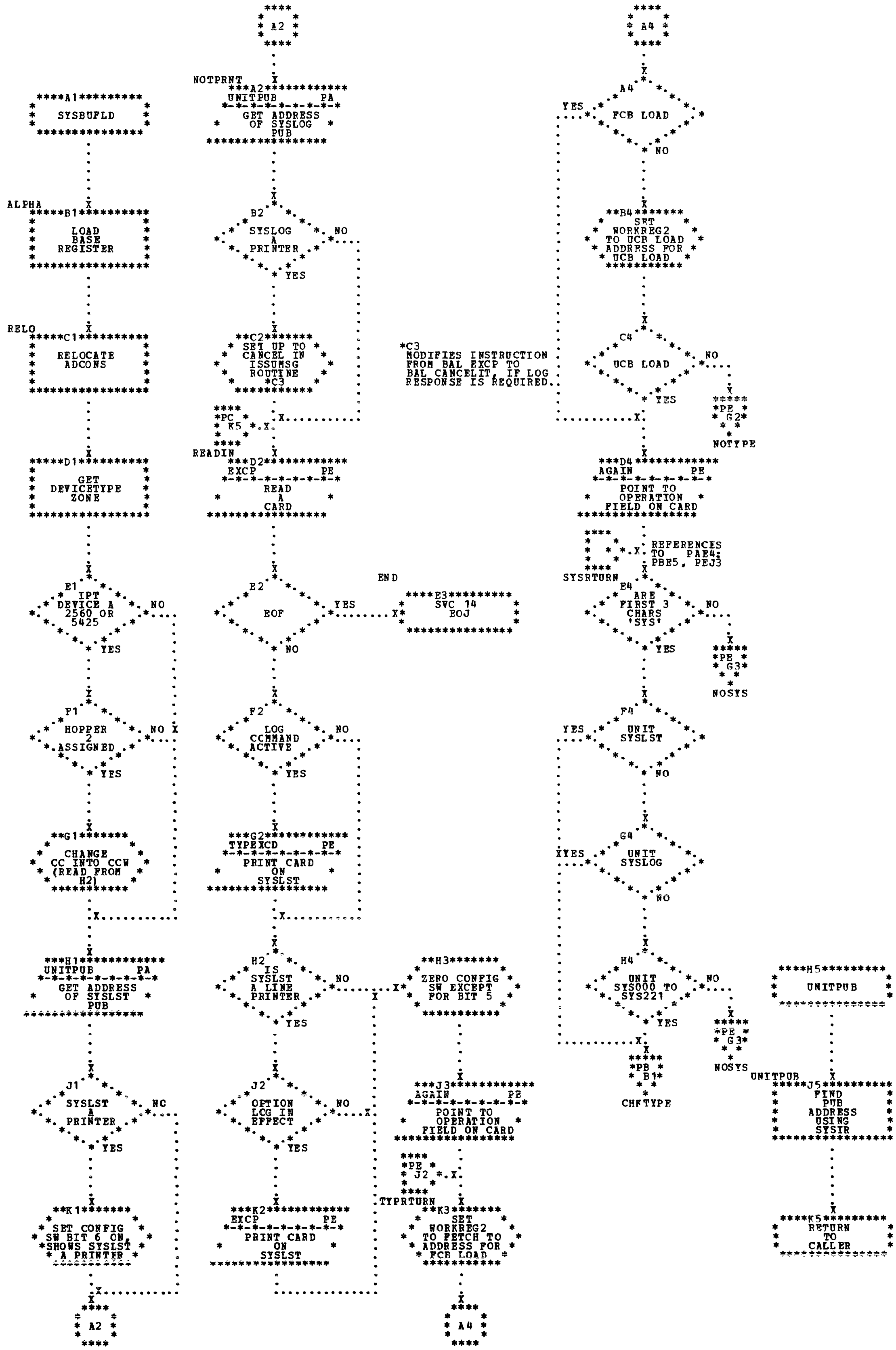


Chart PB. SYSBUFLD - Check for Valid Device and Command
Refer to Chart 19.

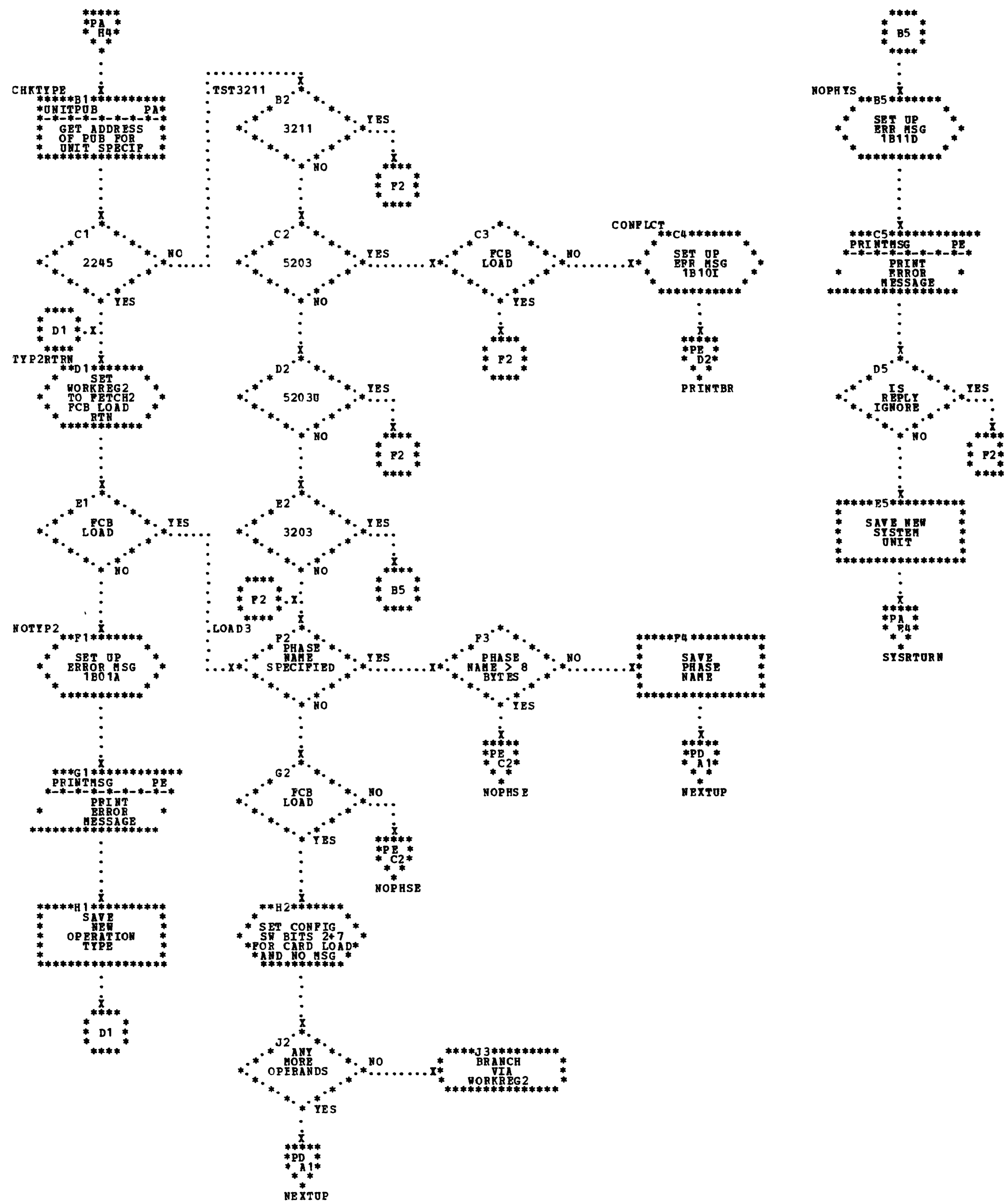


Chart PC. SYSBUFLD - Load UCB
Refer to Chart 19.

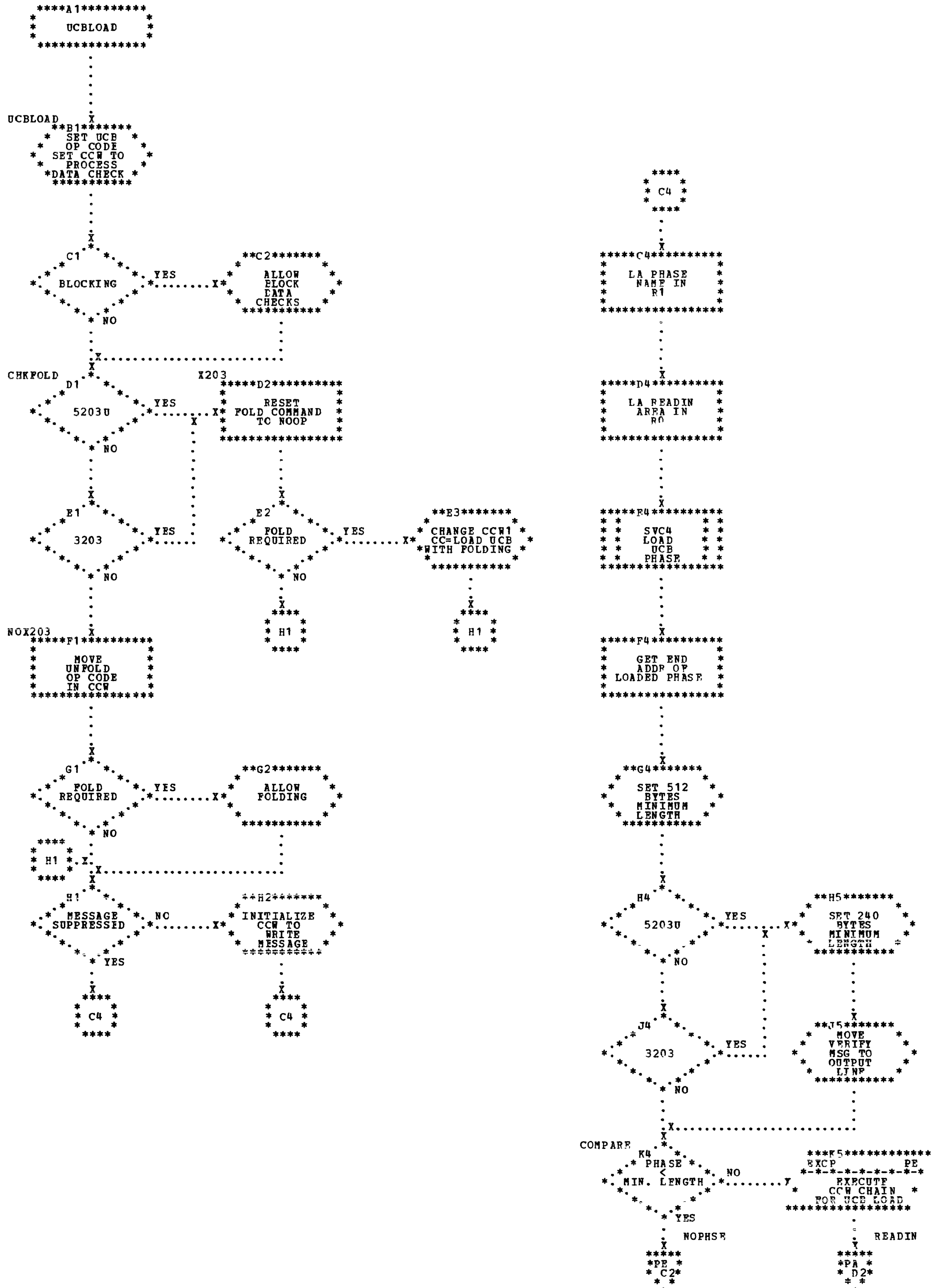


Chart PD. SYSBUFLD - Check Cptional Operands
Refer to Chart 19.

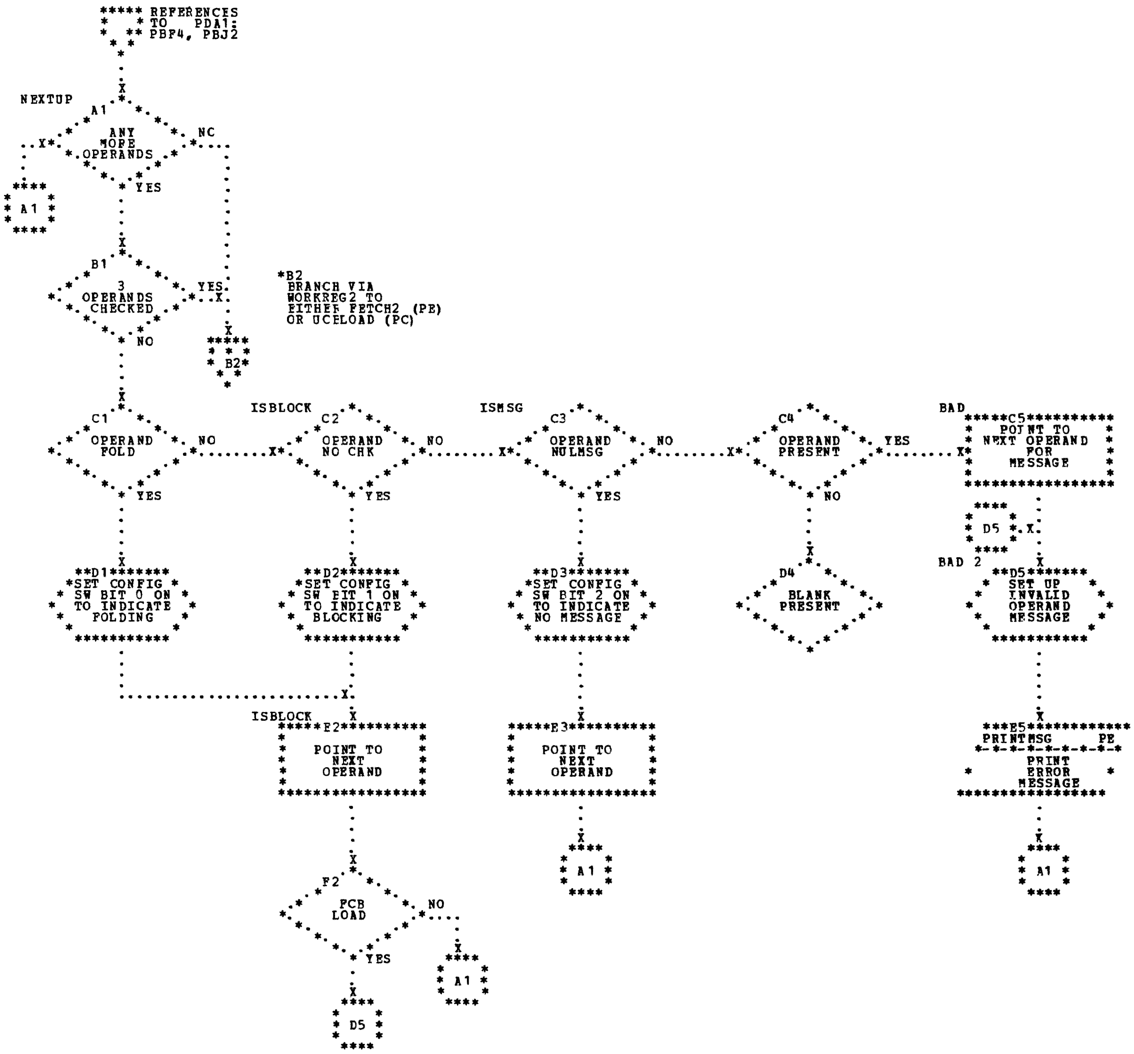


Chart PE. SYSBUFF1 - Miscellaneous Subroutines
Refer to Chart 19.

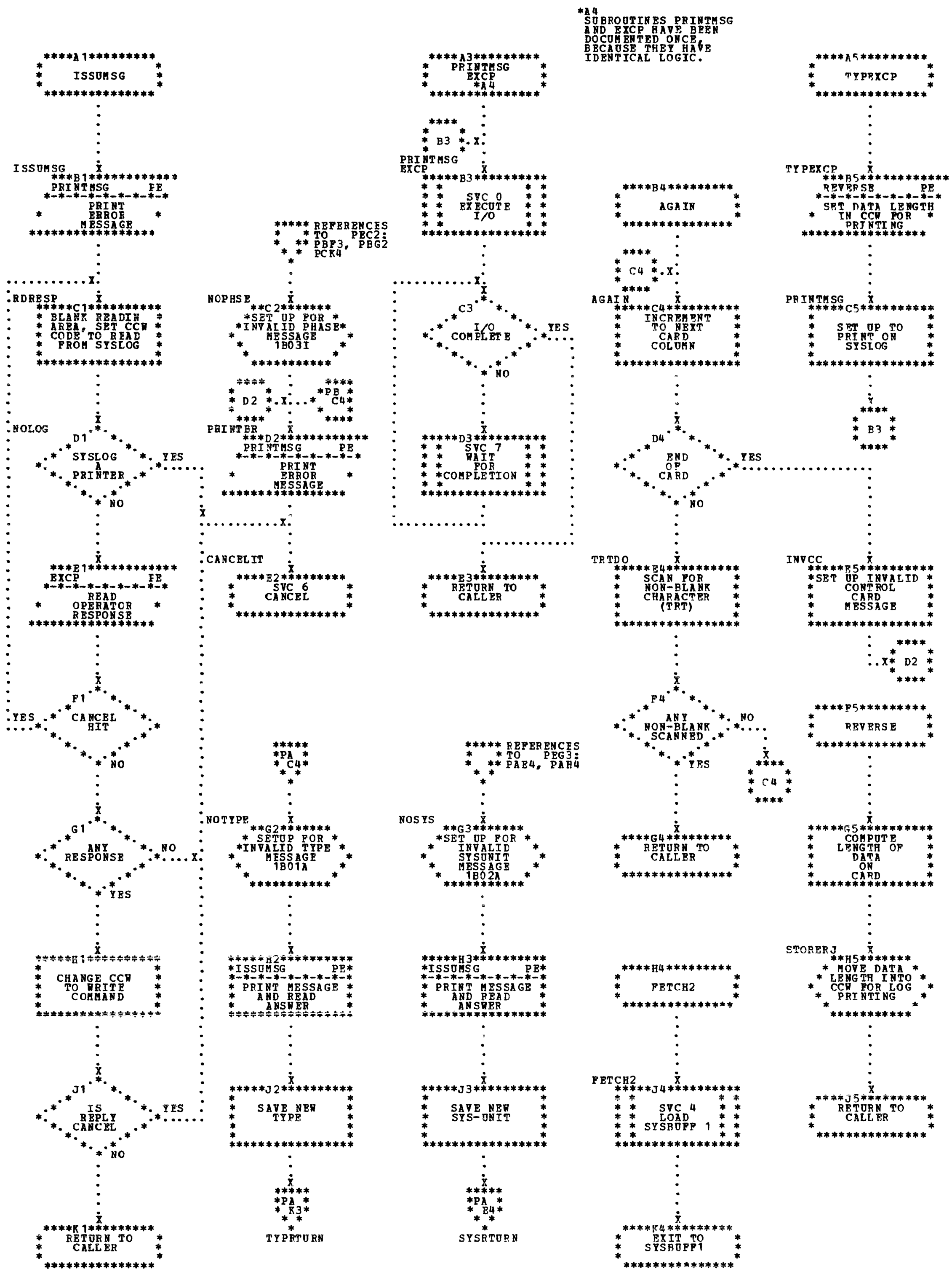


Chart PF. SYSBUFF1 - FCB Buffer Load (Part 1 of 2)
Refer to Chart 19.

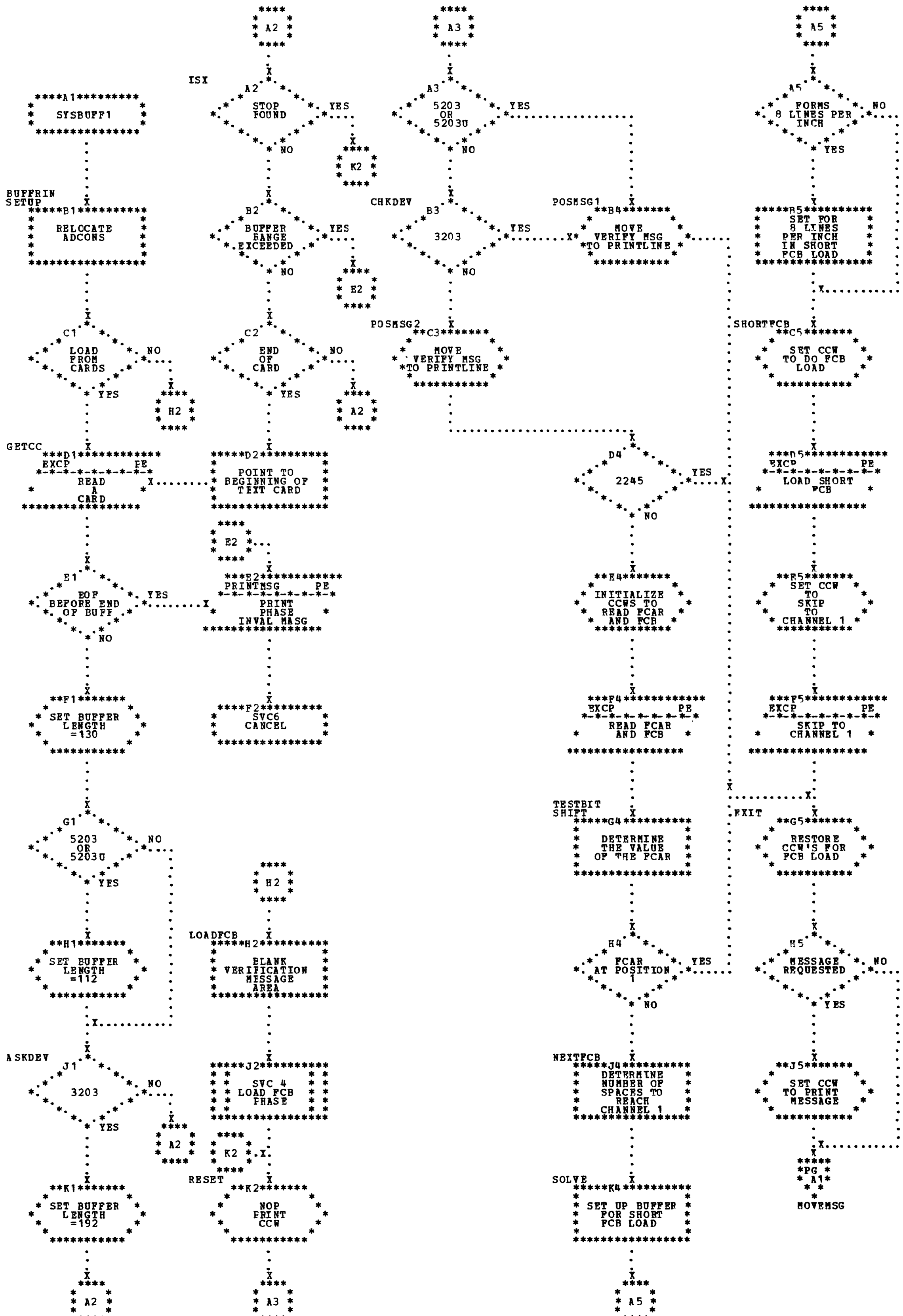


Chart PG. SYSBUFF1 - FCB Buffer Load (Part 2 of 2)
Refer to Chart 19.

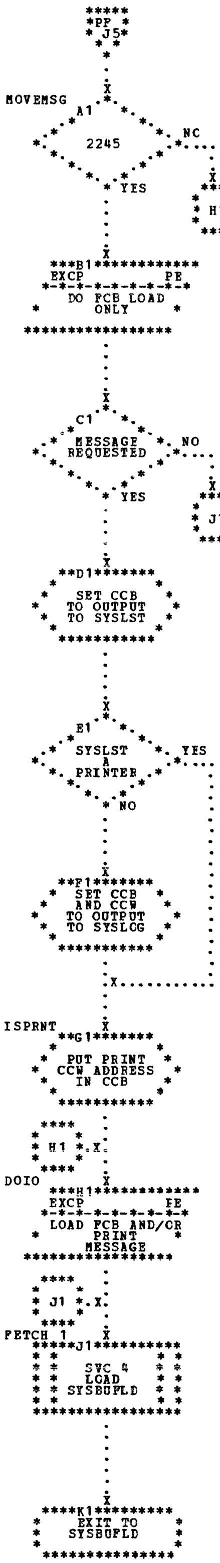


Chart PH. Load UCB
Refer to Chart 20.

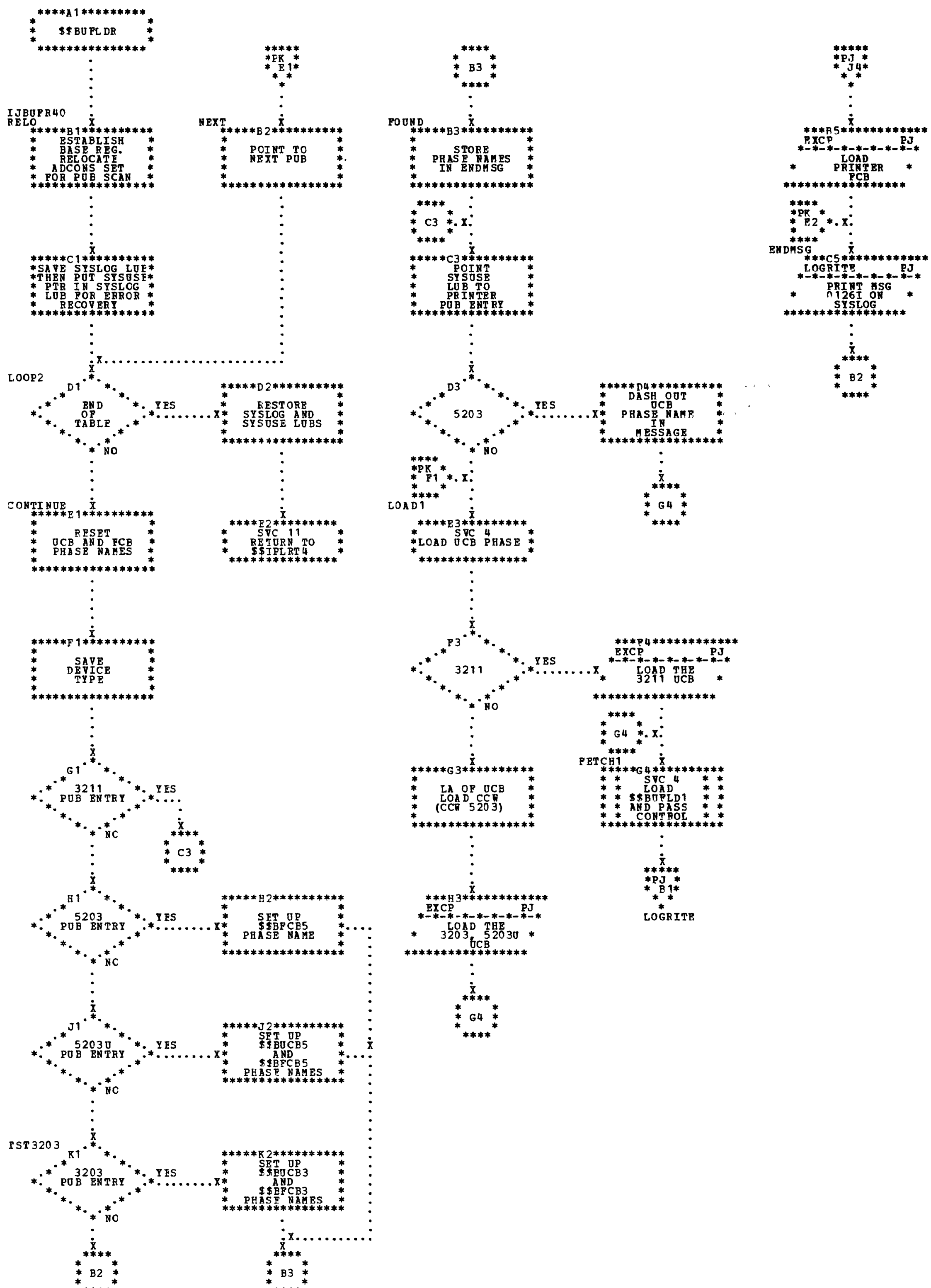


Chart PJ. Print Messages on SYSLOG and Load FCB
Refer to Chart 20.

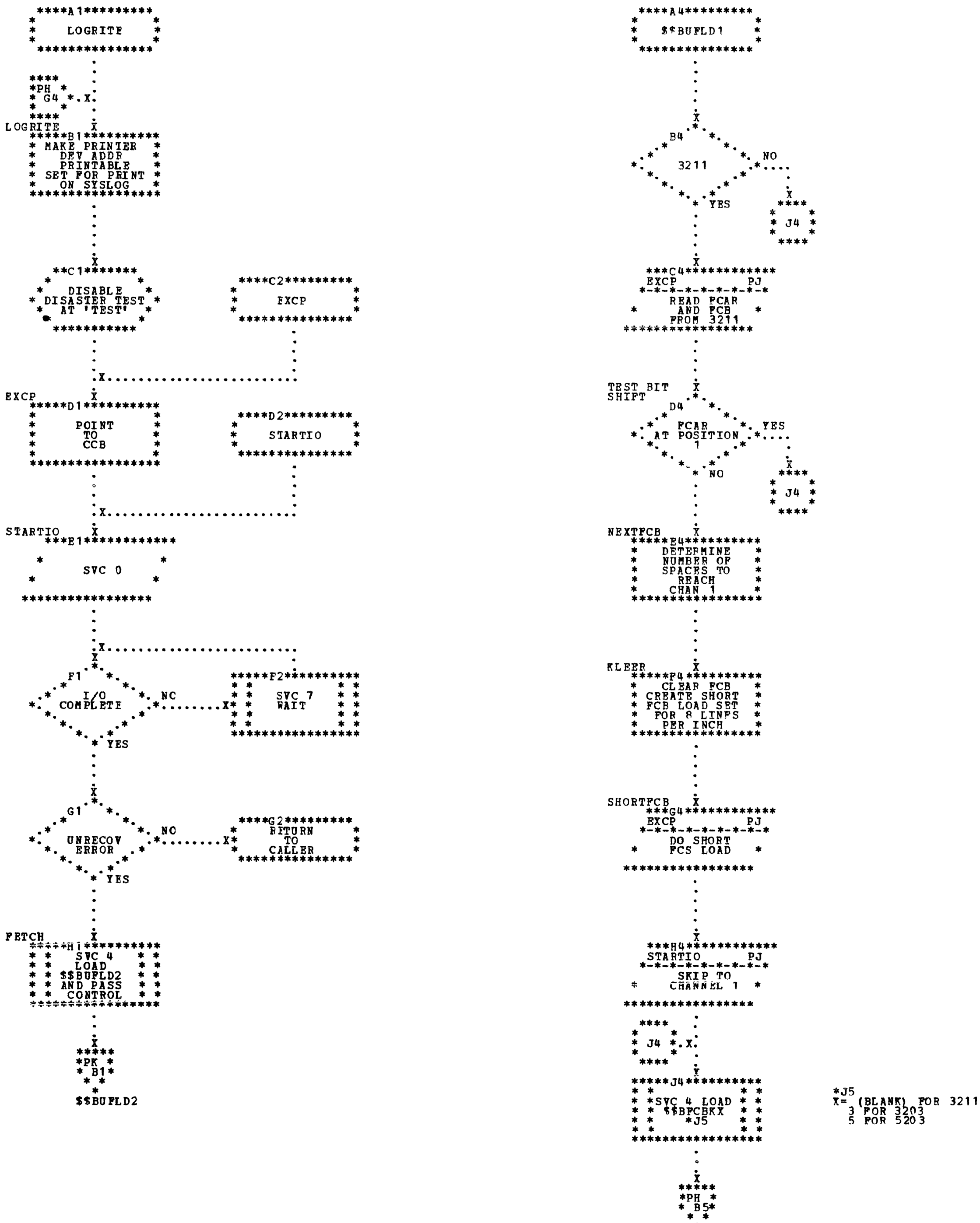
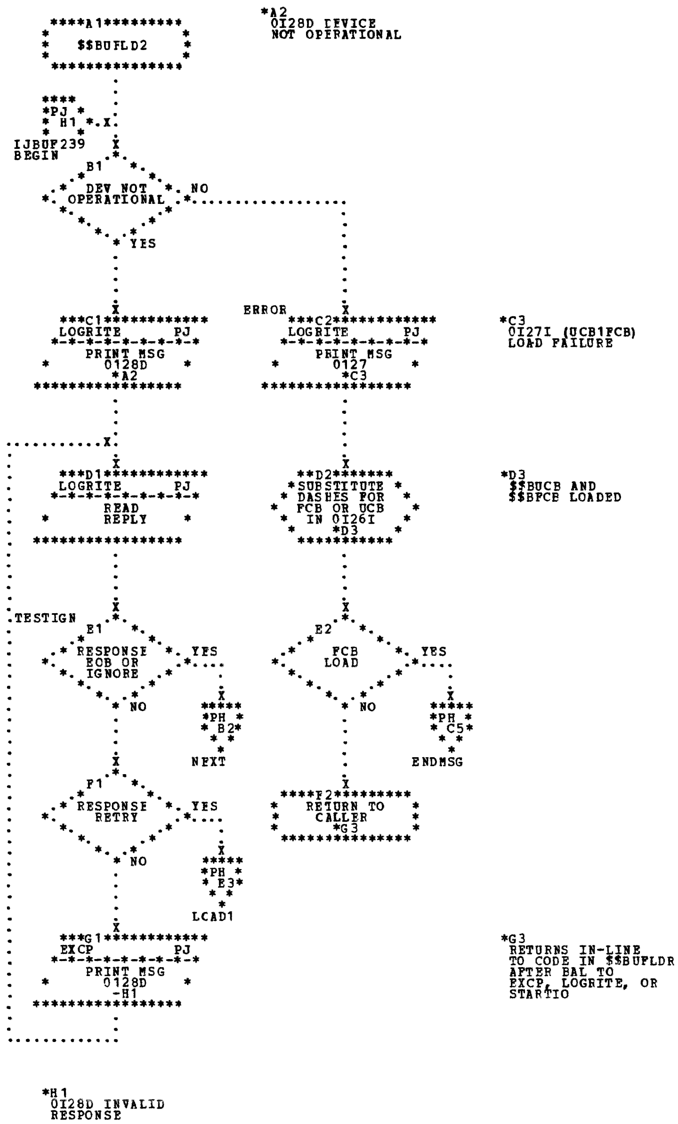


Chart PK. $$BUFLD2 - 3211 Buffer Load Error Subroutine
Refer to Chart 20.



PDAIDS ERROR MESSAGES

MESSAGE	CHART	PHASE
4C10	AA	PDAID
4C11	AB,AK	PDAID
4C12	AK	PDAID
4C13	AH,AK	PDAID
4C14	AH	PDAID
4C15	AJ	PDAID
4C16	AJ	PDAID
4C17	AC	PDAID
4C17	AD	PDAID
4C17	AG	PDAID
4C17	AH	PDAID
4C17	AK	PDAID
4C20	AD,AH	PDAID
4C21	AE	PDAID
4C22	AF	PDAID
4C23	AE	PDAID
4C24	AM	$$BPDAID
4C24	CB	PDAIDITP
4C24	CD	PDAIDITT
4C24	DC	PDAIDFTT
4C24	DE	PDAIDFTT
4C24	EC	PDAIDGTP
4C24	EF	PDAIDGTT
4C24	FA	PDAIDTDT
4C24	FC	PDAIDTDT
4C26	BC	PDLIST
4C27	AC	PDAID
4C28	AE	PDAID
4C50	AC	PDAID
4C51	AC	PDAID
4C52	AC	PDAID

SDAID ERROR MESSAGES

MESSAGE	CHART	PHASE
04E6	GH	SDEHR
4C17	GA	SDAID1
4C17	GB	SDPAR
4C53	GA	SDAID1
4C54	GA	SDAID1
4C55	GA	SDAID1
4C56	GA	SDAID1
4C57	GE	SDAID2
4C58	GB	SDPAR
4C59	GB	SDPAR
4C60	GB	SDPAR
4C61	GB	SDPAR
4C62	GB	SDPAR
4C63	GB	SDPAR
4C64	GB	SDPAR
4C65	GB	SDPAR
4C66	GB	SDPAR
4C67	GB	SDPAR
4C68	GB	SDPAR
4C69	GB	SDPAR

DUMPGEN ERROR MESSAGES

MESSAGE	CHART	PHASE
4C42	HA	DUMPGEN
4C43	HA	DUMPGEN
4C44	HD	DUMPGEN
4C46	HE	DUMPGEN

PDSDM ERROR MESSAGES

MESSAGE	CHART	PHASE
0V20	JA	PDSDM
0V21	JC	PDSDM
0V22	JD	PDSDM
0V23	JA	PDSDM
0V24	JA	PDSDM
0V25	JA	PDSDM
0V26	JA	PDSDM
0V27	JA	PDSDM
0V28	JA	PDSDM
0V29	JA	PDSDM
0V30	JC,JD	PDSDM
0V31	JC	PDSDM
0V32	JC,JD	PDSDM
0V33	JC	PDSDM
0V34	JD	PDSDM
0V35	JC,JD	PDSDM
0V36	JC,JD	PDSDM
0V37	JD	PDSDM
0V39	JD	PDSDM
0V40	JC,JD	PDSDM
0V41	JF	PDSDM
0V42	JJ	PDSDM
0V43	JJ	PDSDM
0V44	JB	PDSDM

EREP ERROR MESSAGES

MESSAGE	CHART	PHASE
3E01	MC	EREPMNTR
3E02	MC	EREPMNTR
3E04	MC	EREPMNTR
3E06	MC	EREPMNTR
3E07	NF	EREPTES
3E07	NH	EREPESWK
3E08	NH	EREPESWK
3E09	NG	EREPESTR

3E10	MD	EREPEDIT
3E10	MQ	EREPSMCP
3E11	MB	EREPMNTR
3E12	MB	EREPMNTR
3E14	MB	EREPMNTR
3E15	NF	EREPTES
3E15	NG	EREPESTR
3E15	NJ	EREPHIST
3E18	NJ	EREPHIST
3E20	NH	EREPESWK
3E21	MD	EREPEDIT
3E22	MD	EREPEDIT
3E25	MB	EREPMNTR
3E25	NF	EREPTES
3E25	NG	EREPESTR
3E25	NJ	EREPHIST
3E26	MC	EREPMNTR
3E27	MC	EREPMNTR
3E28	MC	EREPMNTR
3E29	MC	EREPMNTR
3E30	NJ	EREPHIST
3E31	NF	EREPTES
3E31	NG	EREPESTR
3E31	NJ	EREPHIST
3E32	NK	EREPRDE
3E33	NK	EREPRDE

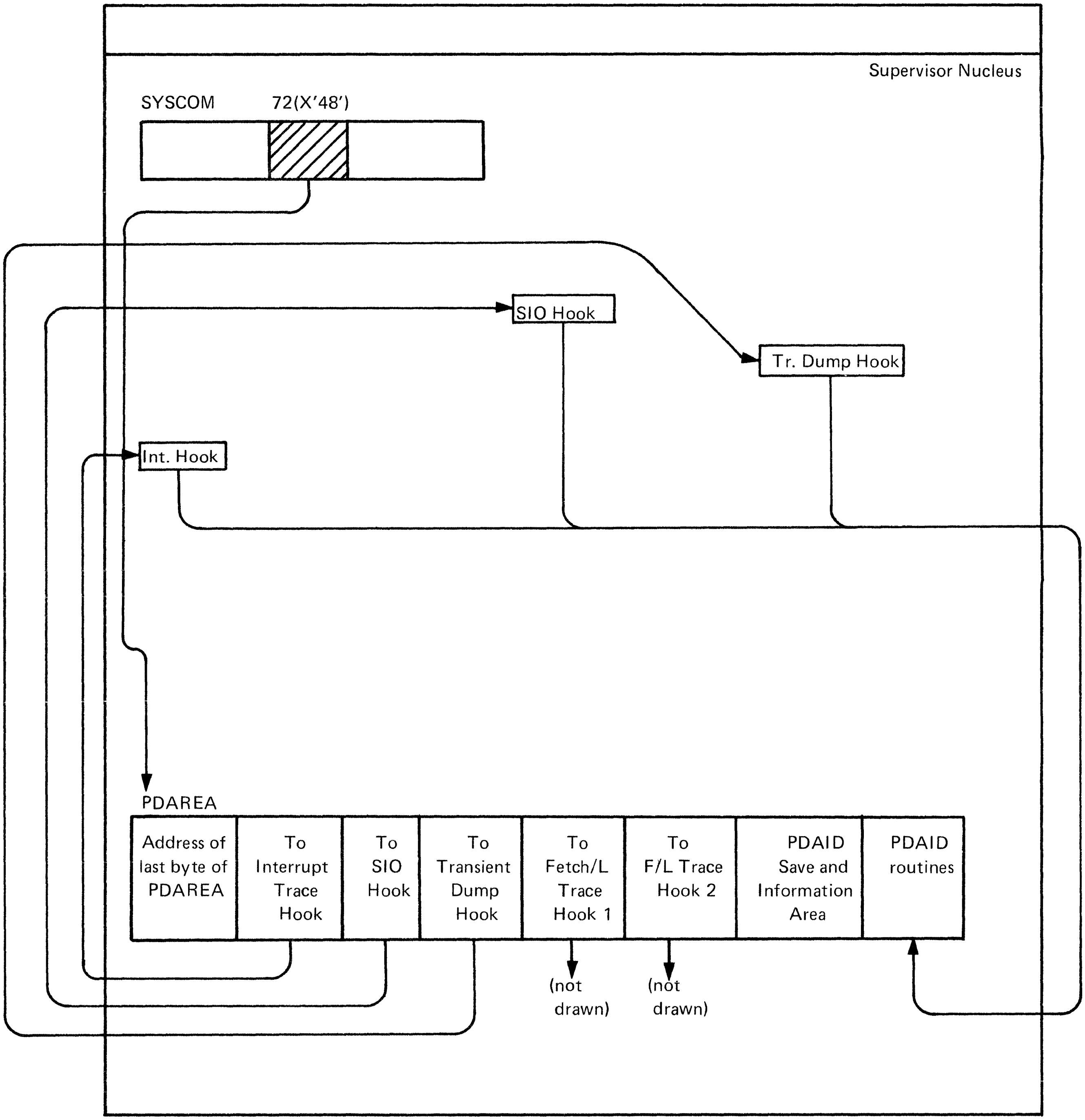
3E34	NK	EREPRDE
3E35	NK	EREPRDE
3E36	NK	EREPRDE
3E37	NK	EREPRDE
3E38	NL	EREPRDE
3E40	NK	EREPRDE
3E41	NK	EREPRDE
3E42	NK	EREPRDE
3E43	NK	EREPRDE
3E43	NL	EREPRDE

PRINTER SUPPORT ERROR MESSAGES

MESSAGE	CHART	PHASE
0I26	PH	$$BFLDR
0I28	PK	$$BFLD2
1B01	PE	SYSBUFLD
1B01	PB	SYSBUFLD
1B02	PE	SYSBUFLD
1B03	PE	SYSBUFLD
1B10	PB	SYSBUFLD
1B11	PB	SYSBUFLD

	Base Registers
HARDWARE/SOFTWARE INTERFACE (PSWs, Logout Areas, etc.)	R0 R11 R12 R13
TABLES AND CONSTANTS, that <u>must</u> be below 4K. Contained in FOPT macro (System Communications Region, Background Communication Region, etc.)	
CANCEL, GENERAL ENTRY, TASK SELECTION; these routines <u>must</u> be below 4K. Contained in FOPT macro. CHANNEL SCHEDULER, SVC ROUTINES AND AP ROUTINES; need <u>not</u> be below a certain boundary.	
Some CONSTANTS that <u>must</u> be below 16K.	
SAVE AREAS, OPTION TABLES, PIB TABLES, I/O TABLES, SYSTEM TASK BLOCKS, FOREGROUND COMMUNICATION REGIONS, etc., having Y-type address pointers in low storage. <u>Must</u> be below 32K.	
I/O INTERRUPT HANDLER, DISK ERP	R9, R10
EXTERNAL INTERRUPT HANDLER	R14
TABLES, SAVE AREAS, TRANSIENT AREAS, having A-type address pointers in low storage (System Task Save Area, Boundary Box, PTA, LTA, etc.)	* Up to this point addressability should be possible with Y-type address constants
FETCH ROUTINE	R6
CCW TRANSLATION ROUTINES	R9
PAGE MANAGEMENT ROUTINES	R14, R15, R13
VSAM ROUTINES	R14
MACHINE CHECK HANDLER, CHANNEL CHECK HANDLER, RAS MONITOR, RTA	R15
PAGE MANAGEMENT TABLES, CCW TRANSLATION COPY BLOCKS	

Figure 54. Supervisor Storage Allocation



Note: The PDAREA is present when PD=YES or PD=n was specified on the FOPT macro. The minimum size is 1400 bytes.

Figure 55. Accessing PDAID Routines

SVC Dec	SVC Hex	Macro Supported	Function
*optional			
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 another 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 communications region
13	D		Set switches in partition communications region
14	E	EOJ	Terminate 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 interrupt
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 command)
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 user's PSW
23*	17		Load phase header. Phase load address is stored at user's address

Figure 56. DOS/VS Supervisor Calls (Part 1 of 4)

SVC Dec	SVC Hex	Macro Supported	Function
*optional			
24*	18	SETIME	Set timer interval and provide supervisor with linkage to user's TECB, if any
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 internal macro COMRG
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		Reserved
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

Figure 56. DOS/VS Supervisor Calls (Part 2 of 4)

SVC Dec	Hex	Macro Supported	Function
*optional			
47*	2F	WAITF	Provide support fcr multiple wait macro WAITF fcr MICR type devices
48*	30		Fetch a CRT transient
49	31		Reserved
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	Release page frames to selection pool
55	37	GETREAL	Provide interface between SDAID and PDAID initialization rcutine and page management rcutine, to create the PDAID alternate area or the SDAID buffer 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 stcrage fcr real jcbs
59	3B		Provide interface between ECJ and the supervisor. Initialize specified page table entries
60	3C	GETADR	Provide virtual address of location within I/O areas for ERP and CRT routines
61*	3D	GETVIS	Get stcrage in virtual partiticn
62*	3E	FREEVIS	Free stcrage in virtual partition
63	3F	USE	Use a resource
64	40	RELEASE	Release a rescurce
65*	41	CDLOAD	Lcad VSAM or CI phase
66	42	RUNMODE	Return mode in which prcgram is running
67*	43	PFIX	Fix page(s) in real stcrage
68*	44	PFREE	Free page(s) in real stcrage
69*	45	REALAD	Return real address ccrrespcnding to a given virtual address
70*	46	VIRTAD	Return virtual address correspcnding to a given real address
71*	47	SETPFA	Establish or terminate the linkage between the supervisor and a user page-fault appendage routine

Figure 56. DOS/VS Supervisor Calls (Part 3 of 4)

SVC Dec	SVC Hex	Macro Supported	Function
*optional			
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 an RMSR I/O error
77	4D	TRANSCSW	Returns the virtual address of a copied CCW
78-84			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

Figure 56. DOS/VS Supervisor Calls (Part 4 of 4)

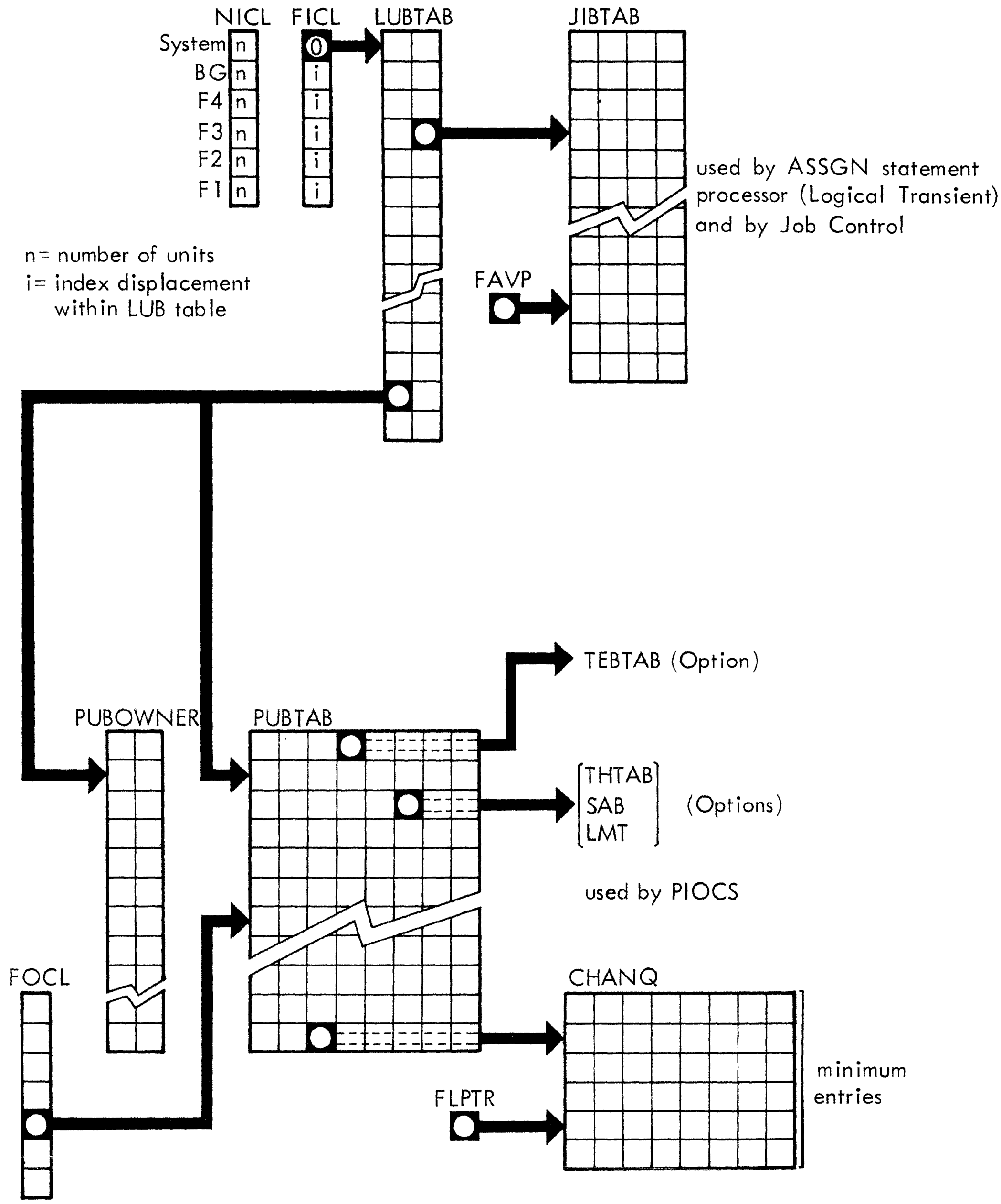


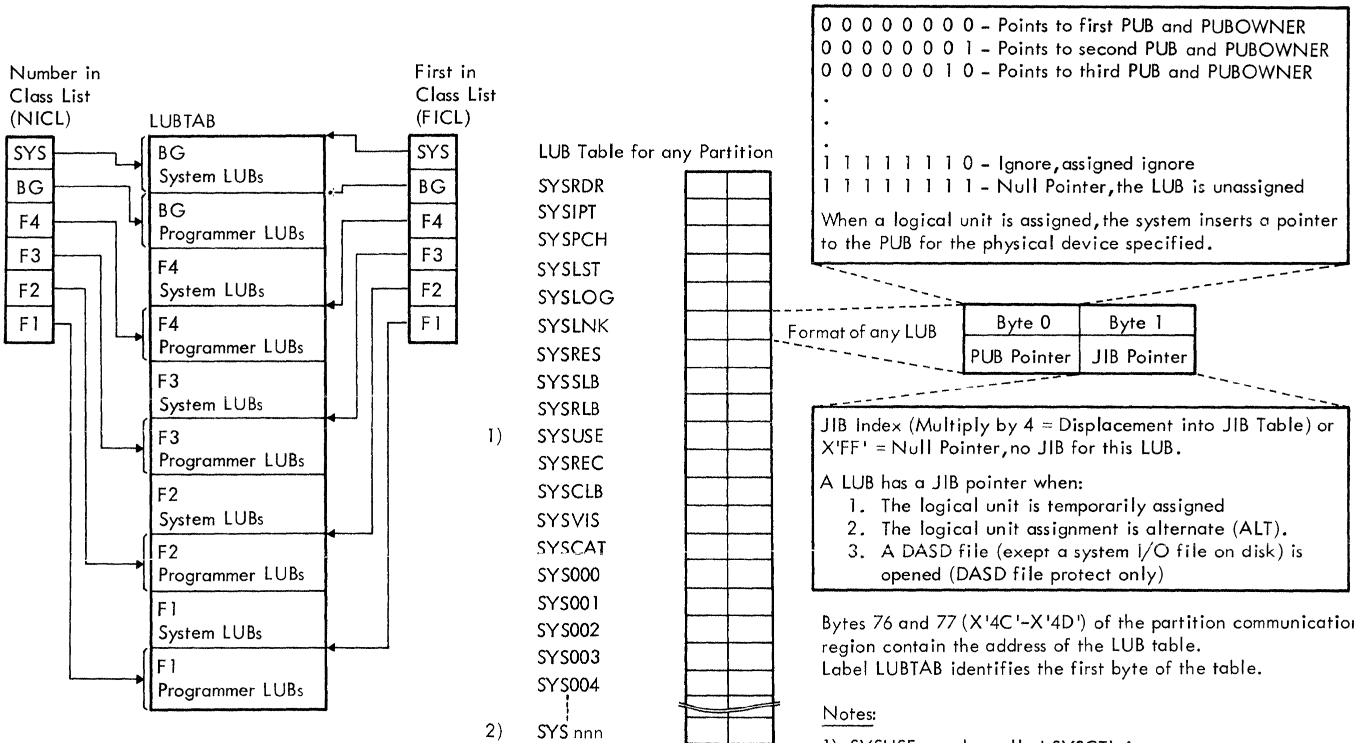
Figure 57. I/O Table Interrelationship (Part 1 of 3)

KEY	EXPLANATION
NICL (Number in Class)	Byte 0 contains the number of system class LUBs. The remaining bytes contain the number of programmer class LUBs for each partition. The total number of bytes is one more than the number of partitions supported.
FICL (First in Class)	Byte 0 points to the first system class LUB table (LUBTAB). This is always the first entry in the LUB table. The remaining bytes point to the first programmer class LUBs in the LUB table partition areas. The total number of bytes is one more than the number of partitions supported.
LUBTAB (Logical Unit Block Table)	Byte 0 of each entry is an index pointer to an entry in the PUB table (PUBTAB) and to an entry in the PUB ownership table (PUBOWNER), or contains X'FF' if no logical unit is assigned. Byte 1 points to an entry in the JIB table (JIBTAB) or contains X'FF'.
PUBTAB (Physical Unit Block Table)	Bytes 0 and 1 of each entry contain the channel and unit address of the physical device. Byte 2 points to the entry in the channel queue (CHANQ) table or contains X'FF'. Byte 3 is a retry counter or, if the unit is a tape cartridge reader and the TEB=n parameter was included in the FOPT macro, contains a pointer to the Tape Error Block table (TEBTAB) entry for the device. Byte 4 contains the device type code. Byte 5 is an index pointer to the entry in the track-hold table, the seek-address block table, or the mode table (Model 125 only) when one of these options is active for the device. Otherwise this byte contains X'FF'. Byte 6 contains the channel scheduler flags and byte 7 the job control flags.
FOCL (First on Channel List)	Byte 0 points to the first PUBTAB entry for a device on channel 0. Byte 1 points to the first PUBTAB entry for a device on channel 1, and so on. X'FF' indicates that the associated channel is not supported.
PUBOWNER	Byte 0 of each entry is reserved. Byte 1 identifies the partition that owns the corresponding PUB.
FAVP (First Available Pointer)	This one-byte pointer to the next available entry in the JIB table (JIBTAB) is used by the ASSGN statement processor and by job control.
JIBTAB (Job Information Block Table)	Bytes 0 and 1 of each entry contain extent or LUB information used by job control and the ASSGN statement processor. Byte 2 is a flag byte. Byte 3 is a chain byte.

Figure 57. I/O Table Interrelationship (Part 2 of 3)

KEY	EXPLANATION
CHANQ (Channel Queue table)	Byte 0 in each entry points to the next entry in the queue for the same device (or the next free entry if in the free list), or it contains X'FF' if the entry is the last in a chain. Bytes 1, 2, and 3 contain the CCB address. Byte 4, if I/O was requested by a user program, contains the Partition Identification Key (PIK). If a system task requested I/O, then the zone field is all zeros and the numeric field contains the ID of the specific system task. Byte 5 contains a displacement index pointing to the LUBTAB entry related to the I/O request (absolute LUBTAB index). Byte 6 contains the relative LUBTAB index for system LUBs, or X'FF' for programmer LUBs. Byte 7 contains the displacement index of the PIBTAB entry for the task requesting I/O (TIK), or X'FF' if the channel queue entry is free.
FLPTR (Free List Pointer)	This one-byte pointer contains the displacement index of the next free entry in the channel queue table (CHANQ).
TEBTAB (Tape Error Block Table)	One entry is built for each tape cartridge reader at supervisor generation time if the FOPT macro parameter TEB=n is included.
THTAB (Track Hold Table)	This table is built at supervisor generation time if the TRKHLD=n parameter is included in the FOPT macro. Byte 0 in each entry points to the next entry in the chain of requests for a track to be held on a specific DASD (or the next free entry if in the free list), or it contains X'FF' if the entry is the last in a chain. Bytes 1, 2, and 3 contain the CCB address. Bytes 4 through 9 contain the disk address (BBCCHH) of the held track. Byte 10 contains the key of the owning task, or all zeros when the entry is free. Byte 11 is a flag and counter byte: bit 0 is turned on when a task requests a track already held by another task, and the value in the low-order half-byte is incremented by one each time a task requests a hold on a track that it already holds itself. Note: when multiple holds by one task are effective, the value in the lower-order half-byte is one less than the actual number of holds.
SAB (Seek Address Block)	This table is built at supervisor generation time if the SKSEP=n or YES parameter is included in the FOPT macro. Bytes 0 through 3 of each entry contain the current disk address (BCCH) for the device. Byte 4 contains X'FF' or points to the THTAB entry.
LMT (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.

Figure 57. I/O Table Interrelationship (Part 3 of 3)

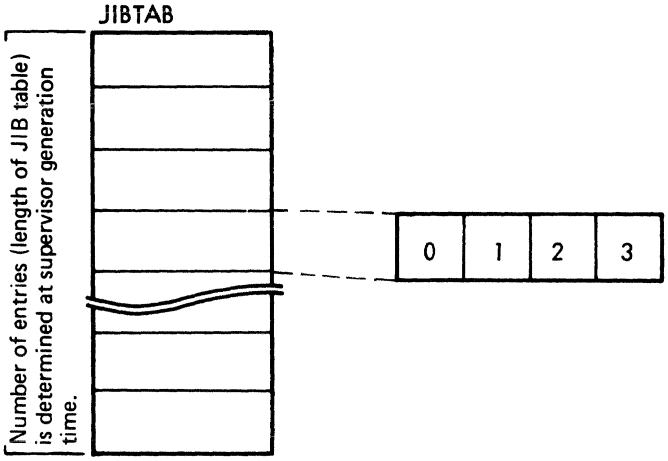


Bytes 76 and 77 (X'4C'-X'4D') of the partition communication region contain the address of the LUB table.
Label LUBTAB identifies the first byte of the table.

Notes:

1) SYSUSE may be called SYSCTL in error recovery messages.
2) Not more than 225 LUBs can be generated into the system. As there are 14 systems LUBs for each partition the maximum number of programmer LUBs in the system is: 241 if NPARTS=1, 227 if NPARTS=2, 213 if NPARTS=3, 199 if NPARTS=4, or 185 if NPARTS=5.
3) Null entries are generated at supervisor generation time. Standard assignments can then be made to PUBs at supervisor generation time or during IPL.

Figure 58. Logical Unit Block Table (LUBTAB)



Byte(s)	Description	
0-1 (Contents depends on the bit setting in byte 2)	Bit Setting Byte 2	Contents
	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 (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 or 3340: Cylinder lower limit, or Cylinder upper limit (One cylinder number uses two bytes) (Note 2) (Note 3)
2	Meaning if bit = 1: Bit 0: Stored standard assignment 1: Alternate assignment 2: 2311/2314/2319 extent 3: 2321/3330/3340 extent 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. 5: Cataloged procedures processing 6/7: Reserved	
3	Chain byte Contains the displacement index of the next JIB. X'FF' defines the end of the chain.	

Figure 59. Job Information Block (JIB) Table

Note 1: Only when file-protect on DASD.

Note 2: Two JIBs 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 type JIB contains the subcell number times 10 plus the strip number in binary.

Note 3: Extent information is supplied by the program initiator and logical IOCS open transient routines. The supervisor can then perform the file protect function for the specified file limits. File protection does not include supervisor and transient originated I/O.

Bytes 68-69 (X'44' - X'45') of the partition communication region contain the address of the JIB table entry. Label JIBTAB identifies the first byte of the table.

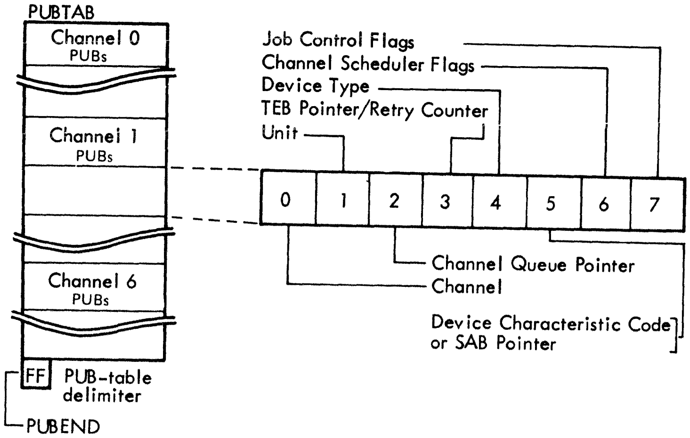


Figure 60. Physical Unit Block (PUB) Table

Bytes:

0: Channel number. (Hex 0-6, FF=NULL)

1: I/O device unit number

2: Hex 0, 1, 2, ... points to the first channel queue entry for this device

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 an ERP retry counter.

4: Device type code

5: SS of the MODE=parameter in the DVCGEN macro for tape unit.

For ICA line (Model 115, or 125), this byte contains the displacement index of the entry in the Line Mode Table (LMT). The address of the LMT is in 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 3704/3705 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

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 or overrunable device on Byte MPX channel
7: 1 = 7-track tape unit

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)
5: 1 = DASD device with disconnect command chaining feature
6-7: B'11' (both on) = Headqueue in progress*
B'01' = Headqueue requested*

*No I/O is started on a PUB or copied PUB that has both these switches on. If only bit 7 is on I/O can be started after seek separation.

Notes: A null entry is generated at supervisor generation time for each device to be supported by the supervisor. Then standard physical unit assignments are made to the PUB table. Physical unit assignments can also be made during IPL. 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.

Bytes 64-65 (X'40' - X'41') of the partition communication region contain the address of the PUB Table entry. Label PUBTAB identifies the first byte of the table.

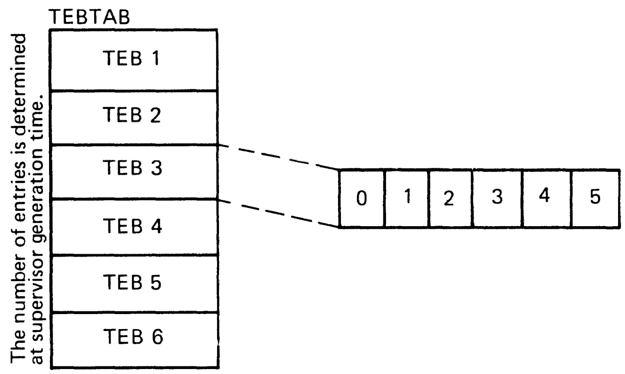


Figure 61. Tape Error Block (TEB) Table

Bytes:

0: Error recovery retry count.

1: Permanent read data check error count.

2: Number of times the read data check error routine is entered.

3: Number of times the write data check error routine is entered.

4: Write skip (erase gap) count.

5: Noise record count.

One TEB is generated for each 2495 Tape Cartridge Reader unit if the FOPT macro contains the TEB=n parameter. Job Control resets each TEB at normal or abnormal End-of-Job. An unused TEB contains HEX 'FF0000000000'. A TEB is referenced from byte 3 of a Tape Cartridge unit PUB.

Bytes 70-71 (X'46' - X'47') of the partition communications region(s) contain the address of the TEB Table entry. Label TEBTAB identifies the first byte of the table.

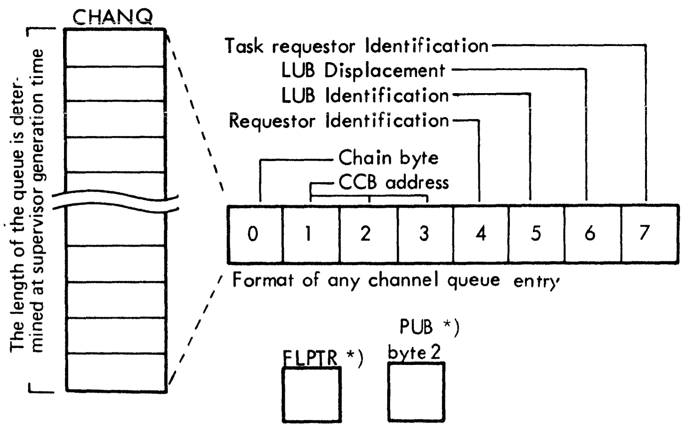


Figure 62. Channel Queue Table (CHANQ) and Entry

Bytes:

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.

1-3: Contains the CCB address for the specified device.

4: Contains a code identifying the task making the I/O request. This 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
4 for CRT
5 for ERP
6 for PAGEIN

nk=FF for unused channel queue entries.

If the I/O request is from a subtask, this byte does not identify this subtask, but the partition.

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.

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 Supervisor.)

7: Contains the displacement within the Pib table for 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.)

*)Notes: FLPTR: The free list pointer contains the displacement within the channel queue of the first entry in the free list or X'FF' when the channel queue is full.

Byte 36 (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.

Bytes 37-39 (X'25' - X'27') of the System Communication Region (SYSCOM) contain the address of the Channel Queue table entry. Label CHANQ identifies the first byte of the table.

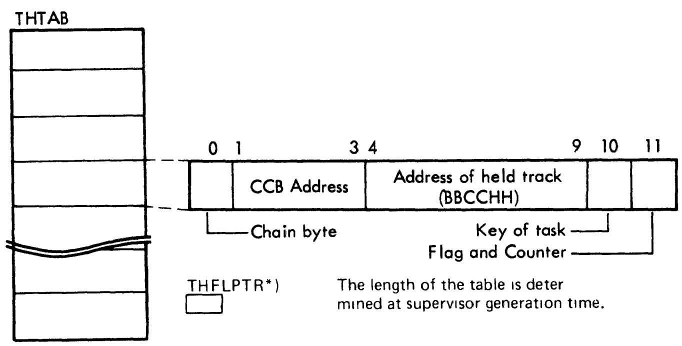


Figure 63. Track-Hold Table (THTAB)

Bytes:

0: Initially, pointer to next sequential entry or X'FF' (table delimiter) in last entry. After requests have been issued, this byte points to the entry for the next request for a track on the same device, or contains X'FF' (in entry for the last request), or it is a pointer in the free list chain.

1-3: Address of CCB associated with the task requesting the hold.

4-9: Disk address of the track being held (in the form BBCCHH).

10: Key of the task owning the track.

11: Bit 0=1: Indicates a task is waiting for this track

1-3: Unused

4-7: Counter of number of holds on the track. (The number of holds is one more than this value.)

*) Note:

THFLPTR: The Track-Hold Free List Pointer (1 byte) contains the displacement within the Track-Hold Table of the first entry in the free list or X'FF' when the Track-Hold Table is full.

Bytes 77-79 (X'4D' - X'4F') of the System Communication Region (SYSCOM) contain the address of the Track-Hold Table. Label THTAB identifies the first byte of the table.

Byte 76 (X'4C') of the System Communication Region (SYSCOM) contains the address of the Track-Hold Free List Pointer. Label THFLPTR identifies the location of the pointer.

The half-word at THTAB-2 contains the total number of 12-byte entries comprising the Track-Hold Table.

Displacement (Hex / Dec)

0	4	8	0A	0C	10	14	18
0	4	8	10	12	16	20	24
Address of Error Block	Address of Attention Exit	Address of Operator Option Cancel Exit	Address of Operator Request Cancel Exit	Address of SYSRES PUB	Address of Fetch Routine	Address of I/O Interr. Routine	Address of Ext. Interr. Routine
xxxx	xxxx	xx	xx	xxxx	xxxx	xxxx	xxxx

1C	20	24	25	28	2A	2C	2E	30	34
28	32	36	37	40	42	44	46	48	52
Address of Logical Transient Area	Address of 1st byte of Problem Program Area	Free List Pointer	Address of Channel Queue	Number of Channel Queue Entries	Length of One Error Queue Entry	Number of Partitions	Not used	Address of Channel Buckets	Address of CRT Table
xxxx	xxxx	x	xxx	xx	xx	xx	xx	xxxx	xxxx

38	3C	40	44	46	48	4C	4D	50
56	60	64	68	70	72	76	77	80
Address of SAB Table	Address of Channel Control Table	Flags and Switches (See expansion) *	System Task Selection Control Field *	Address of Task Selection	Address of PD Area	TH Free List Pointer	Address of TH Table	Address of Timer Request Table
xxxx	xxxx	xxxx	xx	xx	xxxx	x	xxx	xxxx

54	58	5A	5C	60	64	68	6C	70
84	88	90	92	96	100	104	108	112
Address of AB Table	Key of Task owning LTA (LIK)	Key of Task running (TIK)	Address of POWER Table	Reserved	Address of RF Table	Address of EU ECB Table	Address of OLTEP bucket	Address of RAS Linkage Area
xxxx	xx	xx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx

74	78	7C	80	84	88	8C	90
116	120	124	128	132	136	140	144
Address of ASCII Translate Table	Address of PUB Ownership Table	Address of Job Accounting Common Table	Base Address of Page Management Routine	Base Address of Channel Program Translation Routine	Address of SDAID Comm. Area	Address of Line Mode Table	Address of VSAM Communication Area
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx

94	98	9C	A0	A1	A2	A3	A4	A5	A6	A7
148	152	156	160	161	162	163	164	165	166	167
Address of PTA	Address of first System Task Block	Address of Task Block of Active System Task	1 byte for Alignment	Pointer to RAS Task Block	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')
xxxx	xxxx	xxxx	x	x	x	x	x	x	xxxxxxxxxx	xxxxxxxxx

B0	B4	B8	BC	BE	C0	CB	CC	CE
176	180	184	188	190	192	203	204	206
Not used	Address of MVCFLD	Not used	Not used	Not used	Repositioning Information for MFCM ERP	Number of Error Queue Entries	Length of PUB Table in bytes	Number of Active Partitions
xxxx	xxxx	xxxx	xx	xx	xxxxxxxxxxx	x	xx	xx

D0	D4	D8	DC	E0	E4	E8	EC
208	212	216	220	224	228	232	236
Address of Segment Table	Address of Page Frame Table	Address of Page Frame Table Extension	Address of Boundary Box	Address of DPD Table	Reserved	Address of VIRTAD Routine	Address of End of Real Storage
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx

F0	F4	F5	F8
240	244	245	248
Address of Fetch Table	SVA Flag (see expansion)	Address of SVA	Address of System GETVIS AREA
xxxx	x	xxx	xxxx

*See next page for further explanation.

Figure 64. System Communications Region (SYSCOM) (Part 1 of 2)

Expansion of SYSCOM Flag Bytes

Byte Dec	Byte Hex	Description
64	40	Reserved for RMS in Model 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'06' 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'06' POWER initialized
		X'02' GETREAL for SDAID or PDAID in progress
		X'01' Fetch for system task in progress (used by PDAID)
67	43	Reserved
244	F4	SVA Flag
		X'00' 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' Reserved
		X'02' Reserved
		X'01' Reserved

Layout of System Task Selection Control Field

Byte Dec	Byte 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

Note: The address of SYSCOM can be found at fixed location X'80' - X'83'.

Figure 64. System Communications Region (SYSCOM) (Part 2 of 2)

nnCOMREG

Displacement Hex	0	8	0A	0C	17	18	20	24	28	2C
Displacement Dec	0	8	10	12	23	24	32	36	40	44
Field	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
Length	xxxxxxxx	xx	xx	xxxxxxxxxxx	x	xxxxxxxx	xxxx	xxxx	xxxx	xx

Displacement Hex	2E	30	34	35	36	37	38	39	3A	3B	3C	3E
Displacement Dec	46	48	52	53	54	55	56	57	58	59	60	62
Field	PIK	End of Virtual Storage Address	Machine Configur. Byte	System Configur. Byte 1	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
Length	xx	xxxx	x	x	x	x	x	x	x	x	xx	xx
					Job Control Switches							

Displacement Hex	40	42	44	46	48	4A	4C	4E	4F	58	5A	5C
Displacement Dec	64	66	68	70	72	74	76	78	79	88	90	92
Field	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 1st Part of PIB Table	ID Number of last Checkpoint or DASDFP Indicator
Length	xx	xx	xx	xx	xx	xx	xx	x	xxxxxxxxx	xx	xx	xx

Displacement Hex	5E	60	62	64	66	68	6A	6C	6E
Displacement Dec	94	96	98	100	102	104	106	108	110
Field	Job Zone in Minutes	Address of Disk Information Block (DIB)	Reserved	Address of PC Option Table less 8 bytes	Address of IT Option Table less 8 bytes	Address of OC Option Table less 8 bytes	Key of Program with Timer Support	Reserved	Logical Transient Key (LTK)
Length	xx	xx	xx	xx	xx	xx	xx	xx	xx

Displacement Hex	70	74	78	7C	7E	80	84	86	87
Displacement Dec	112	116	120	124	126	128	132	134	135
Field	Address of SYSPARM	Address of J.A. Partition Table	Address of TOD clock Common Area	Address of 2d part of PIB Table	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
Length	xxxx	xxxx	xxxx	xx	xx	xxxx	xx	x	x

Displacement Hex	88	8C	8D	8E	8F	97	98	9F
Displacement Dec	136	140	141	142	143	151	152	159
Field	Pointer to Option Table in SYSCOM	Standard Options	Temporary Options	Disk Configuration	Catalog Procedure Name	Switch for Catalog Procedure	JCL Statement Name	81 bytes SYSIN Indicator
Length	xxxx	x	x	x	xxxxxxxx	x	xxxxxxx	x

Note: A communication region exists from each partition supported by the system.
The address of the communication region of the active partition is in fixed loc. X'14'-X'17'.

Figure 65. Partition Communications Region (Part 1 of 6)

Key to Communication Region Displacement

Key	Description of Use
0	MM/DD/YY or DD/MM/YY either set permanently 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, date convention bit 0.)
8	Address of the problem program area.
10	Address of the beginning of the problem program area (see above). Y(EOSSP) equals Y(PPBEG)
12	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	User program switch indicator.
24	Job name set by the job control program from information found in the job statement
32	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	Address of the uppermost byte of the last phase of the problem program fetched or loaded. Not filled in when phase is in SVA.
40	Highest ending real storage address of all the phases having the same first four characters as 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 this phase 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	Length of the problem program label area.
46	Partition Identification Key (PIK). The low-order byte identifies the active partition. Only significant for BG communication region.
48	End address of virtual storage.
52	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 = Physical transient overlap option 0 = No physical transient overlap option 4: Always set to indicate standard timer feature 5: 1 = Channel switching device 0 = No channel switching device 6: 1 = Burst mode on multiplex channel support 0 = No burst mode on multiplex channel support 7: Indicates MCH/CCH in system
53	System Configuration byte. Bit 0: 1 = DDMMYY (Date convention bit set at generation time by STDJC) 0 = MMDDYY (Date convention bit set at generation time by STDJC) 1: 1 = Two or more partitions 0 = One partition only

Figure 65. Partition Communications Region (Part 2 of 6)

Key	Description of Use
	53 System Configuration byte (Cont'd.) 2: 1 = DASD file-protect supported 0 = No file-protect support for DASD 3: 1 = DASD SYSIN - SYSOUT 0 = No DASD SYSIN - SYSOUT 4: 1 = Teleprocessing 0 = No teleprocessing 5: 1 = Two or more partitions 0 = One partition only 6: 1 = Asynchronous processing 0 = No asynchronous processing 7: 1 = Track Hold 0 = No Track Hold
54	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: LISTX 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	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 set 1: DUMP option 1 = yes, dump registers and storage on SYSLST 2: 1 = partition in wait state, because a 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: ASCII option 1 = yes, ASCII supported
56	Job Control byte. Bit 0: 1 = Job Accounting Interface (JA) 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 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

Figure 65. Partition Communications Region (Part 3 of 6)

Key	Description of Use
57	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 Load list 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 Core Image Library in program (interface between $MAINDIR and Supervisor) 7: 1 = Check automatic condense limits and End-of-Job (interface between Librarian and Job Control)
58	Language processor control byte. This is a set of switches used to specify nonstandard 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	Job duration indicator byte.
	
	Bit 0: 1 = Job in progress
	0 = No job in progress
	1: 1 = Dump on an abnormal End-of-Job condition
	0 = No dump on abnormal EOJ
	2: 1 = Pause at EOJ step (Set by attention routine for Job Control)
	0 = No pause at EOJ (Set by attention routine for Job Control)
	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

60	Binary disk address of the volume label area (label cylinder).
62	As illustrated (for detailed figures see <u>DOS/VS Supervisor Logic</u>, SY33-8551).
76	As illustrated (for detailed figures see <u>DOS/VS Supervisor Logic</u>, SY33-8551).
78	Set to the value nn specified in the LINES=nn parameter of the STDJC macro.
79	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	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	Address of the first part of the Program Information Block (PIB) table.

Figure 65. Partition Communications Region (Part 4 of 6)

Key	Description of Use
92	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 0: 1 = 3330 file protection 1: 1 = 3340 file protection
94	Job zone for Time-of-Day. If ZONE=EAST, value is positive; if ZONE=WEST, value is negative.
96	Address of disk I/O position data. This is the starting address of the Disk Information Block (DIB) table for the partition.
98	Reserved.
100	PC option table (zero if not specified).
102	IT option table (zero if not specified).
104	OC option table (zero if not specified).
106	X'0010' if interval timer support. X'0000' if no interval timer support.
108	Reserved.
110	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 SVC 2 routine sets the LTK. The SVC 11 routine resets the LTK. (Only significant in BG communication regions.)
112	Address of SYSPARM field.
116	Address of Job Accounting partition table.
120	Address of the Time-of-Day Clock common area.
124	Address of second part of Program Information Block (PIB) table.
126	Address of PDTABB, table of DTF addresses for MICR support.
128	Address of QTAM vector table (IJLQTTAD).
132	Address of background communications region.
134	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 End 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

Figure 65. Partition Communications Region (Part 5 of 6)

Key	Description of Use
135	System Configuration byte 2 and RMSR Open Flag byte. Bit 0: 1 = PCIL supported 1: 1 = TOD-clock supported 2: 1 = PFIX macro supported 3: 1 = Fetch $$BOPEN by $JOBCTLJ 4: 1 = Fetch $$BOPEN by $JOBCTLD 5: 1 = Fetch $$BOPEN by $JOBCTLJ for WTM 6: 1 = QTAM supported 7: Reserved
136	Pointer to Option table in SYSCOM Reserved for compatibility reasons.
140	Standard Job Control Option byte. Bit 0: 1 = EDECK Standard Option 1: 1 = ALIGN Standard Option 2: ACANCEL Standard 3-7: Not used
141	Temporary Job Control Option byte. Bit 0: 1 = EDECK Temporary Option 1: 1 = ALIGN Temporary Option 2: 1 = ACANCEL temporary 3-7: Not used
142	Disk Configuration byte. Bit 0-4: Not used 5: 1 = 3340 supported 6: 1 = 3330 supported 7: Always 1; indicates 2311 and 2314/2319 supported
143	Cataloged Procedure Name.
151	Interface byte for Cataloged 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 Supeviscr
152	JCL statement name for Cataloged Procedure.
159	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 65. Partition Communications Region (Part 6 of 6)

PIBTAB

Attention PIB
Background PIB
FG 4 PIB
FG 3 PIB
FG 2 PIB
FG 1 PIB
Subtask PIB
Subtask PIB

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 or Zero (Note 1) (Note 2)	X'07' PIB Assign Flag (See D)	BG User LUB Index	Number of BG Program LUBs	Not used

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 LUBs	Flag Byte (See E)

Notes:

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.

2. When LTA is active for Logical Attention, bytes 9-11 are zero and bytes 5-7 contain the LTA save area address.

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:

Number of Partitions	Number of Subtasks
2	13
3	12
4	11
5	10

Bytes 90-91 (X'5A' - X'5B') of the partition communication region(s) contain the address of the PIB table. Label PIBTAB identifies the first byte of the table.

A <u>Flag Byte (First byte in PIB)</u>

The following flags are always used:

X'71' = Program is waiting for SVC 58
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 free console buffer table entry (used only when CBF=n)
X'80' = Program is not active
X'81' = Program is SVC 2-bound (waiting for the LTA to be released)
X'82' = Program is SVC 7-bound (waiting for an I/O interruption)
X'83' = Program is ready to run
X'85' = Program is SVC 5-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

Figure 66. Program Information Block (PIB) Table (Part 1 of 2)

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 SVC 35-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 SVC 38-bound
X'53' = Program is SVC 41/42-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

The following flag is only used when CBF=n:

X'7D' = Program is waiting for free console buffer table entry.

The following flag is only used when TP=QTAM:

X'8B' = Task in QTAM wait.

B <u>PIB DAT Flag</u>

X'01' = Return to reentrant supervisor routine
X'02' = Return to gated supervisor routine
X'04' = Move CCB at dispatching time
X'08' = Service delayed external interrupt
X'10' = Deactivation of this task is being delayed
X'20' = Reserved
X'40' = Task has seized the system
X'80' = Program is running in virtual mode

C <u>Gate Identifier</u>

X'71' = Gating of SVC 58 required
X'53' = Gating of SVC 41/42 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 <u>PIB Assign Flag</u>

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 <u>Program Program PIB Flag</u> (Last byte in PIB)

Bit 0: 1 = Batched job in foreground (always on when tested)
1: 1 = Cancel in LTA and device not assigned
2: 1 = /& on SYSIN if DASD
3: 1 = Partition in stopped state
4: 1 = Fetch EOJ monitor
5: 1 = Task is canceled
6: 1 = Subtask(s) attached
7: 1 = in AB routine

F <u>Attention PIB Switch Byte</u>

Bit 0: Reserved
1: 1 = Fetch Physical Attention Transient $$ABERRZ
2: 1 = Delay cancellation (fetch $$ABERZ1)
3: 1 = Emergency cancel request
4: Reserved
5: 1 = Command available (DOC)
6: 1 = Fetch Logical Attention Routine ($$BATTNA)
7: 1 = External Interrupt request

Figure 66. Program Information Block (PIB) Table (Part 2 of 2)

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
2400T9	9-track Magnetic Tape units	50	
2400T7	7-track Magnetic Tape units	50	
3410T9	9-track 3410 Magnetic Tape units	53	Magnetic Tape
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	
2520B1	2520B1 Card Read Punch	31	
2560	2560 Multifunction Card Machine		Card Read Punches
2596	2596 Card Read Punch	30	
3525RP	3525 Card Punch (with optional read feature)	32	
5425	Multifunction Card Unit	34	
2501	2501 Card Reader	10	
2540R	2540 Card Reader	11	Card Readers
3504	3504 Card Reader	12	
3505	3505 Card Reader	12	
2540P	2540 Card Punch	21	
2520B2	2520B2 Card Punch	20	
1442N2	1442N2 Card Punch	22	Card Punches
2520B3	2520B3 Card Punch	20	
3525P	3525 Card Punch	23	
1403	1403 Printer	40	
1403U	1403 Printer with UCS feature	42	
1443	1443 Printer	41	
2260 (local)	1053 Printer with 2548 Control Unit. MODE operand must be entered as X'01'	C0	
3203	3203 Printer	4A	
3211	3211 Printer	43	Printers
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 multiplexer channel. MODE operand must be entered as X'01'	B0	
5203	5203 Printer	4C	
5203U	5203 Printer with UCS feature	4D	
1050A	3210, 3215 Console Printer Keyboards	00	Printer Keyboards
125D	Model 115/125 Integrated Display Operator Console	B2	
125DP	Model 115/125 Integrated Display Operator Console with 5213 Console Printer attached	B2	Display Operator Consoles

Figure 67. Device Type Codes (Part 1 of 2)

Card Code	Actual IBM Device	Device-Type X'nn'	Device Type
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 3333-1 Disk storage	63	
3340	3340 Disk Storage (General)	68	
3340	3340 Disk Storage with 3348 Model 35	69	
3340	3340 Disk Storage with 3348 Model 70	6A	
1419	1255 Magnetic Character Reader	72	
1419	1259 Magnetic Character Reader	72	MICR-Magnetic Ink
1419	1419 Magnetic Character Reader	72	Character Recognition
1419P	1419 Dual Address Adapter Primary Control Unit	73	devices
1419S	1419 Dual Address Adapter Secondary Control Unit	74	
2701	2701/2715 Data Adapter Unit	D0	Teleprocessing lines
A			A=SAD0 command when
B	2702 Transmission Control Unit	D1	B=SAD1 command enabling
2702 C			C=SAD2 command the line
D			D=SAD3 command
2703	2703 Transmission Control Unit	D2	
2703	Integrated Communication Adapter (Models 125/135)	D2	
2703	3704/3705 Communication 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	
1018	1018 Paper Tape Punch with 2826 Control Unit	79	Paper Tape Punch
1419	1270 Optical Reader/Sorter	79	
1419P	1275 Optical Reader/Sorter	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 Input/Output Unit	80	Diskette
2260	2260 Display Station	C0	
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	Display Station
7770	7770 Audio Response Unit	D3	Audio Response Unit

Figure 67. Device Type Codes (Part 2 of 2)

Density (Bytes per inch)	Parity	Convert Feature	Translate	SS Code *
200	odd	on	off	10
200	odd	off	off	30
200	odd	off	on	38
200	even	off	off	20
200	even	off	on	28
556	odd	on	off	50
556	odd	off	off	70
556	odd	off	on	78
556	even	off	off	60
556	even	off	on	68
800	odd	on	off	90
800	odd	off	off	B0
800	odd	off	on	B8
800	even	off	off	A0
800	even	off	on	A8
800	dual density nine-track			C8
1600	dual density nine-track			C0
6250	nine-track 3420			D0
* Refer to PUB Table (Figure 61), byte 5.				

Figure 68. Density Data

Cancel Code (hex)	Message Code	Descriptive Part of Message (or Condition)
10	----	Normal EOJ
11	0V07I	No channel program translation unsupported device
12	0V06I	Insufficient buffer space for channel program translation
13	0V05I	CCW with count greater than 32K
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	(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	Maintask Termination
1E	0S13I	Unknown ENQ Requestor
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 (invalid LUB code in CCB)
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	0V10I	I/O error on page data set
2B	0P84I	I/O error during fetch from private core image library
2C	0V09I	Illegal parameter passed by PHO routine
2D	0P88I	Program cannot be executed/ restarted due to failing storage block

Figure 69. Cancel Codes and Messages (Part 1 of 2)

Cancel Code (hex)	Message Code	Descriptive Part of Message (or condition)
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	Error Queue Overflow
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
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 set by $$ABERZ1 when no more I/O requests are pending and the cancel procedure therefore can be continued. Later it is tested for by $$BEOJ and subsequently reset.		

*If the CCB is unavailable, the logical unit is SYSxxx.

**This cancel code is not significant in case of a supervisor catalog or IPL failure, because the system is placed in a wait state without any further processing by the Terminator.

Figure 69. Cancel Codes and Messages (Part 2 of 2)

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	ACCT$JOB	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.

NPARTS=2

44-47		Address of F1 Job Accounting Table
48-51	ACCTSEAS	Control Field: prevents the accounting routine being loaded twice

NPARTS=3

44-47		Address of F2 Job Accounting Table
48-51		Address of F1 Job Accounting Table
52-57	ACCTSEAS	Control Field: prevents the accounting routine being loaded twice

*) These values are the same as the PIK values for the relevant tasks.

Figure 70. Job Accounting Interface Common Table (ACCTCOMN) (Part 1 of 2)

NPARTS=4

Displacement	Label	Description
44-47		Address of F3 Job Accounting Table
48-51		Address of F2 Job Acccunting Table
52-55		Address of F1 Job Acccunting Table
56-63	ACCTSEAS	Control Field: prevents the accounting rcutine being loaded twice

NPARTS=5

44-47		Address of F4 Job Accounting Table
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 loaded twice

Bytes 124-127 (X'7C' - X'7F') of the System Communication Region (SYSCOM) contain the address of the Job Accounting Interface Common Table. Label ACCTCOMN identifies the first byte of the table.

Figure 70. Job Accounting Interface Common Table (ACCTCOMN) (Part 2 of 2)

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
12-13	ACCTPART	ID of partition in charge (partition switch name)
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	ACCTCPUT	Counter for CPU time elapsed in a jobstep, counted in 300th 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

Figure 71. Job Accounting Interface Partition Table (ACCTxx) (Part 1 of 2)

Displacement	Label	Description
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 300th of a second
116-119	--------	All-bound time; system wait state time divided between running partitions, in 300th of a second
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

Notes:

1. 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, ACCTF1, for active partitions BG, F4, F3, F2, and F1, respectively.

2. The address of this table is in the partition Communication Region at displacement 116 (X'74').

Figure 71. Job Accounting Interface Partition Table (ACCTxx) (Part 2 of 2)

```
                                                    --- I/O DEVICE EDITING ---

                     TASK IDENTITY - EXAMPLE                                        DAY  YEAR          HH MM SS
RECORD TYPE - UNIT CHECK                                                    DATE - 135   73    TIME - 06 16 45

CPU MODEL 0155  SERIAL 510764
DOS/VS RELEASE LEVEL 29

FAILING CHANNEL/UNIT ADDRESS 0382      DEVICE TYPE 3420

              CC    DA     FL       CT                                                 K     CA     US  CS   CT
FAILING CCW  02  0038FE  CO  00  0050                                             CSW  00  003EF0  0E  08  0000

            --- UNIT STATUS ---                                             --- CHANNEL STATUS ---
ATTENTION         0        CHANNEL           1          PRGM-CTLD IRPT     0          CHAN DATA CHECK  1
STATUS MODIFIER  0        DEVICE END        1          INCORRECT LENGTH  0          CHAN CTL CHECK   0
CONTROL UNIT END 0        UNIT CHECK        1          PROGRAM CHECK      0          I/F CTL CHECK    0
BUSY              0        UNIT EXCEPTION   0          PROTECTION CHECK  0          CHAINING CHECK   0

NUMBER OF I/O RETRIES  00003
TCU SERIAL 0000    TU SERIAL 0000     VOLUME SERIAL 123456    MODE SET C3

BYTE 0        BYTE 1        BYTE 2        BYTE 3     0  BYTE 4        BYTE 5        BYTE 6        BYTE 7
CMND REJ  0   NOISE      1  TRK ERR 0 0   R/W VRC    0  ALU CHECK 0   NEW SUBSY 0   7 TRACK    0  LAMP FAIL 0
INTV REQ  0   TU STAT A 1   TRK ERR 1 0   MTE/LRCR   0  REJECT TU 0   NEW SUBSY 0   WRT HD CR 0   TP BOT LF 0
BUS CHK   1   TU STAT B 0   TRK ERR 2 0   SKEW       0  TAPE IND  0   WRT TM CK 0   DUAL DEN   0  TP BOT RH 0
EQUIP CK  0   7 TRACK    0  TRK ERR 3 0   EDC/CRC    0  WR TR VRC 0   PE ID BUR 0   NRZI       0  RESET KEY 0
DATA CK   0   LOAD PT    0  TRK ERR 4 0   VRC/ENV    0  U-PGM DET 0   ST RD CK  0   MODEL      0  DSE         0
OVERRUN   0   WRITE STA 0   TRK ERR 5 0   1600 BPI   0  LOOP W/R  0   PART RCD  0                 ERASE HD  0
WRD CNT   0   FL PROT    0  TRK ERR 6 0   BACKWARD   0  TU CK     0   EX POST A 0                 AIR PRESS 0
DT CNVT   0   NOT CAPAB 0   TRK ERR 7 0   C COMPARE 0   RPQ       0   RPQ        0                LOAD FAIL 0

BYTE 8        BYTE 9        BYTE 10       BYTE 11       BYTE 12       BYTE 17       BYTE 18       BYTE 19
IBG DROP  0   BIT 0      0  CND S REJ 0   B/BUS/LSR 0   B/BUS/LSR 0   2 CHAN SW 0   PRW CK     0  DE DR 7    0
FD THR CK 0   VEL CHG    0  SPARE      0  SPARE      0  SPARE      0  FEATURE    0  SPARE      0  DE DR 6    0
BIT 2      0  RESERVED   0  CTL S REJ 0   XFER/LOIC 0   XFER/LOIC 0   FEATURE    0  SPARE      0  DE DR 5    0
ERLY BOR  0   RESERVED   0  NO BOR     0  INST/HIIC 0   INST/HIIC 0   FEATURE    0  SPARE      0  DE DR 4    0
ERLY END  0   SPARE      0  WIM-EQ CK 0   U-PGM ERR 0   U-PGM ERR 0   EC OF CU   0  EC OF DRV 0   DR DE 3    0
SLOW BOR  0   SPARE      0  TACH FAIL 0   D BUS PTY 0   D BUS PTY 0   EC OF CU   0  EC OF DRV 0   DE DR 2    0
SLOW END  0   SPARE      0  SPARE      0  SPARE      0  SPARE      0  EC OF CU   0  EC OF DRV 0   DE DR 1    0
VEL TRY   0   CTL U RES 0   VEL CHK    0  AOC ALU1   0  BOC ALUZ  0   EC OF CU   0  EC OF DRV 0   DE DR 0    0

BYTE 20       BYTE 21       BYTE 22       BYTE 23       --- SDR AREA ---
DE DR F   0   LD BUTTON 0   FRU ALU1   0  FRU ALUZ   0  NOISE      000 WR TR VRC 00   ERLY BOR   00
DE DR E   0   LFT REEL   0                              R/W VRC    000 WRT TM CK 00   ERLY END   00
DE DR D   0   RMT REEL   0                              MTE/LRCR   000 ST RD CK  00   SLOW BOR   00
DE DR C   0   TAPE PRES 0                               EDC/CRC    000 PART RCD  00   SLOW END   00
DE DR B   0   REELS LOD 0                               VRC/ENV    000 EX POST A 00   VEL RTY    00
DE DR A   0   LD RWD     0                              OVERRUN    00  IBG DROP  00   VEL CHG    00
DE DR 9   0   LD COMPL   0                              SKEW CK    00  FD THR CK 00   BACKWARD   000
DE DR 8   0   LD CK      0                              C COMP     000 PE ID CK  0    BUS CHK    00
                                                                                    ALU CHECK 00
----- DEVICE DEPENDENT INFORMATION ------                                           TRK IN ERR      000000000
 BLOCK LENGTH    00000        PERM WRTS     003
 TEMP READ       002          NOISE BLKS    100
 TEMP WRT        005          ERASE GAPS    00000
 SIO COUNT       00003        CLEAN ACT     00003
 PERM RDS        001
```

Example Unit check record for 3420

DETAIL -- MAGNETIC TAPE ERROR STATISTICS BY VOLUME DATE 73/135

VOLUME SERIAL	DATE	TIME OF DAY	CUA	TU SERIAL	RD/ WRT	PERM READ	PERM WRT	TEMP READ	TEMP WRT	SIO COUNT	BLOCK LENGTH	PROGRAM ID	CPU ID	MOD NO	DENSITY
123456	73/135	06.16.45	0382	0000	R	001	003	002	005	00003	00000	EXAMPLE	0155		8/1600
123456	73/135	06.16.45	0383		R	006	002	001	004	00005	00000	TEST	0155	1	8/1600
983451	73/135	06.16.45	0382	0000	R	000	000	001	003	00007	00000		0155		8/1600
983451	73/135	06.16.45	0383		R	006	002	002	003	00005	00000	TEST2	0155	1	8/1600

Example of Error Statistics by Tape Volume Number.

DETAIL -- MAGNETIC TAPE ERROR STATISTICS BY TAPE UNIT DATE 73/135

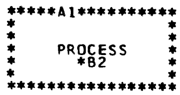
CUA	TU SERIAL	DATE	VOLUME SERIAL	TIME OF DAY	TEMP READ	TEMP WRT	SIO COUNT	DENSITY	NRZI NOISE	R/W VRC	WR TG VRC	LRC MTE	CRC EDC	ECC ENV	SKEW ERR	ERLY BOR	VEL CHG	TIE
0382	0000	73/135	123456	06.16.45	002	005	00003	8/1600	000	000	00	000	000	000	00	00	00	000
0382	0000	73/135	983451	06.16.45	001	003	00007	8/1600	214	064	05	213	193	212	12	15	01	003
0383		73/135	123456	06.16.45	001	004	00005	8/1600	000	000		000	000	000	00	00		000
0383		73/135	983451	06.16.45	002	003	00005	8/1600	000	000		000	000	000	00	00		000

Example of Error Statistics by Tape Unit.

```
                              --- I/O DEVICE EDITING ---

RECORD TYPE - VOLUME DISMOUNT                                            DAY  YEAR          HH MM SS
                                                                 DATE - 135   73   TIME - 06 16 45
CPU MODEL 0155   SERIAL 510764
DOS/VS RELEASE LEVEL 29

FAILING CHANNEL/UNIT ADDRESS  0382  DEVICE TYPE 3420

NUMBER OF SIOS FOR THIS VOLUME 00000007
VOLUME ID   983451
TAPE MODE C3

SDR COUNTERS                                                          --- DEVICE DEPENDENT DATA ---

         NOISE              214       R/W VRC              064  BLK LENGTH   00000     NOISE BLKS      080
         MTE/LRCR           213       EDC/CRC              193  TEMPY READ   001       TEMPY WRT       003
         VRC/ENV CK         212       OVERRUN              00   PERM READ    000       PERM WRT        000
         SKEW               12        C COMPARE            00   ERASE GAPS   00000     CLEANER ACTNS   00003
         WRITE TRIG VRC     05        PE ID BURST CHK      0    TRKS IN ERR  000000011
         WRITE TM CHK       00        START READ CHK       00
         PARTIAL RECORD     00        EXCESSIVE POSTAMBLE  15
         IBG DROP/WRITE     00        FEED THRU CHK        00
         END VEL CHK        15        EE READ BACK CHK     00
         SLOW BOR           00        SE READ BACK CHK     00
         VELOCITY RETR      03        VEL CHNG DURING WT   01
         BACKWARD           063       BUS OUT CHK          00
         ALU HARDWARE ER    00
```

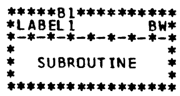
Example Volume Dismount for 3420.

DESCRIPTION

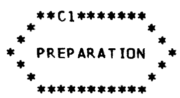


A group of program instructions that perform a processing function of the program. The label, if any, is shown above the block.

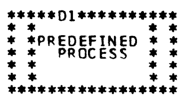
```
*B2
IF ANY ADDITIONAL EXPLANATION IS
REQUIRED, ITS LOCATION ON THE CHART
IS IDENTIFIED BY AN ASTERISK AND
THE BLOCK ID.
```



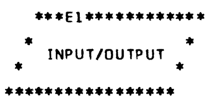
Description or title of a routine that is detailed on another flowchart. The starting label of the routine and the flowchart ID appear above the stripe.



An instruction, or group of instructions, that changes portion of a routine or initializes a routine for a given condition.

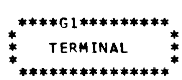


A group of operations not detailed in the flowcharts in this manual, such as user routines.

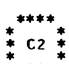


Any function of an input/output device or program, usually branching to an I/O routine to perform the function stated in the block.

Points where the program branches to alternate processing, based upon variable conditions such as program switch settings and test results.



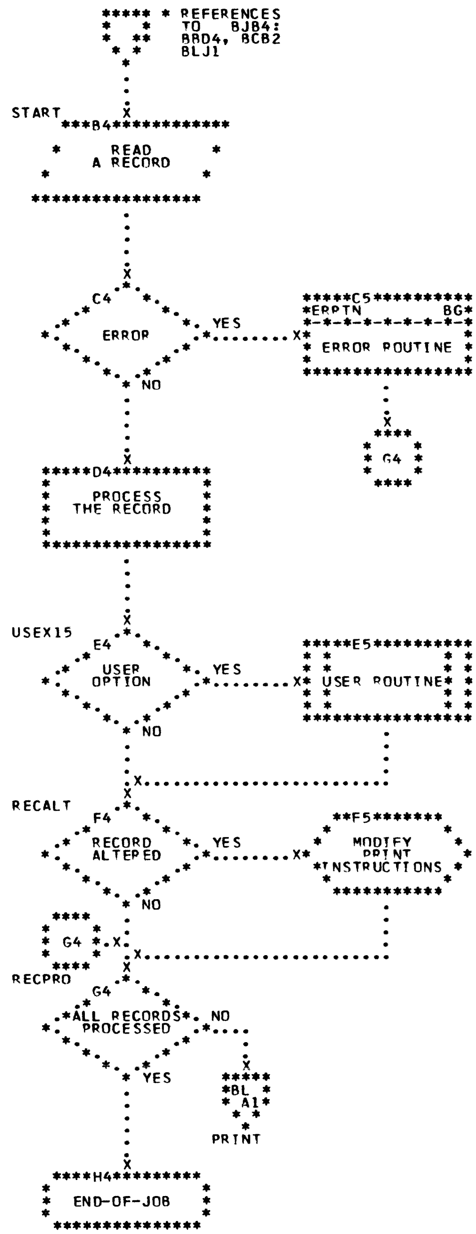
The beginning or end of a program or routine.



On-page connector. An entry from or an exit to another function on the same flowchart. The location in the connector identifies the block to which entry on a chart is made.

Off-page connector. An entry from, or exit to, a given point on another flowchart. The characters in the connector identify the chart and block to which or from which control is passed. The corresponding label, if any, is placed outside the outgoing connector. For multiple entries, an asterisk is placed in the connector and the locations from which control is passed are listed nearby.

EXAMPLE



Label	Phase	Location
AAAGET	PDAID	ACA1
AAAGET1	PDAID	ACB1
ABORT	DUMPGEN	HDH5
ACCEPT	PDAIDQTW	EJF4
ACCEPT	PDAIDQTT	EMD5
ACCPARM	PDSDM	JCB1
ACCPAR1	PDSDM	JCG1
ACCPAR2	PDSDM	JCJ1
ACCPAR3	PDSDM	JCC5
ACCPAR4	PDSDM	JDB1
ACCPAR5	PDSDM	JDG1
ACCPAR6	PDSDM	JDB3
ACCPAR7	PDSDM	JDG3
ACCPAR8	PDSDM	JDB5
ACCPAR81	PDSDM	JDE5
ACHCKID	PDSDM	JGB3
ACHKNTRY	PDSDM	JGE1
ACHKREAL	PDSDM	JGG3
ADDRCHCK	PDSDM	JGB1
ADDREDIT	SDEHR	GLJ1
ADDRINT1	REALDUMP	HFG5
ADDZERO	DUMPGEN	HBC1
ADPTESC	EREP2715	MLB4
ADPTDISK	EREP2715	MLA5
ADPTHD	EREPADPT	MMC2
ADPT2740	EREP2715	MLB2
ADPT2790	EREP2715	MLB3
ADVANCE	LSERV	KAD3
AERR	PDSDM	JAH2
AERR1	PDSDM	JAF4
AERR10	PDSDM	JCH4
AERR2	PDSDM	JAG4
AERR3	PDSDM	JAH4
AERR4	PDSDM	JAC5
AERR5	PDSDM	JAH5
AERR6	PDSDM	JCB4
AERR6	PDSDM	JDG2
AERR8	PDSDM	JCE4
AFTDUB	$$BSDAID	GFD1
AFTFIX	$$BSDAID	GFJ1
AGAIN	SYSBUFLD	PEC4
AGETDISP	PDSDM	JGE3
ALLIN	DUMPGEN	HDG1
ALLTRACE	PDAIDGTW	EAJ1
ALLTRACE	PDAIDGTP	EDF3
ALLTRACE	PDAIDGTT	EGF3
ALPHA	SYSBUFLD	PAB1
AMOREPAR	PDSDM	JGB5
AOUTVIRT	PDSDM	JGD5
APARM0	PDSDM	JAE1
APARM00	PDSDM	JAD1
APARM1	PDSDM	JAJ1
APARM2	PDSDM	JAD3
APARM3	PDSDM	JAF3
APARM4	PDSDM	JAB5
APARM5	PDSDM	JAE5
APARM7	PDSDM	JBB1
APARM8	PDSDM	JBD1
APLOOP	REALDUMP	HQD1
AREAEND	PDAIDTDT	FAG4
AREAEND	PDAIDTDP	FCG4

Label	Phase	Location
AREASTAT	EREP2715	MLB1
ARETURN	PDSDM	JGK1
AROUND	LSERV	KAF3
AROUND13	PDAID	ADG3
AROUND14	PDAID	AGF2
ASETREAL	PDSDM	JGD2
ASKAAA	PDAID	ACC1
ASKAID	PDAID	AAJ2
ASKDEV	SYSBUFF1	PFJ1
ASKODQST	PDAID	ABB2
ASKSVCQS	PDAID	ABG3
ASK4AID	PDAID	AAD1
ASSIGNER	DUMPGEN	HBB3
ASTRK	EREPEDIT	MEA5
ASTRK	EREPEDCP	MUB4
ASTRK	EREPEDES	NEC3
BACK	EREPEDIT	MEA3
BACK	EREPEDCP	MUB2
BACK	EREPEDES	NEC1
BAD	SYSBUFLD	PDC5
BAD 2	SYSBUFLD	PDD5
BADCARD	PDAID	AFB5
BALCVDEC	EREPEDTR	MVA3
BCTLOOP	PDAID	ALJ5
BEGIN	PDAIDITP	CBH4
BEGIN	PDAIDITT	CDH4
BEGIN	DUMPGEN	HAB1
BEGIN	ESTVUT	LAB1
BEGIN	EREPEDIT	MDA1
BEGIN	EREPESPT	MFA2
BEGIN	EREPTES	NFA1
BEGIN	EREPESWK	NHA2
BEGIN	EREPHIST	NJA1
BEGIN	EREPRDE	NKA1
BEGIN	$$BUFLD2	PKB1
BLANKS	EREPEDIT	MEA4
BLANKS	EREPEDCP	MUB3
BLANKS	EREPEDES	NEC2
BLK	ESTVUT	LAA4
BLKNR1	REALDUMP	HLD1
BLNKCHK	PDAID	AEE1
BLNKCHKC	PDAID	AFC5
BOPLY	PDAIDQTW	ELE4
BOPLY	PDAIDQTT	EMK3
BPUT	PDLIST	BCH4
BRNCH1	PDAIDGTW	EAB3
BRNCH1	PDAIDGTW	EAB5
BRNCH1	PDAIDGTP	EDG1
BRNCH1	PDAIDGTP	EDG4
BRNCH1	PDAIDGTT	EGG1
BRNCH1	PDAIDGTT	EGG4
BRNCH2	PDAIDGTW	EAC3
BRNCH2	PDAIDGTW	EAC5
BRNCH2	PDAIDGTP	EDH1
BRNCH2	PDAIDGTP	EDH4
BRNCH2	PDAIDGTT	EGH1
BRNCH2	PDAIDGTT	EGH4
BRPROC	EREPUNIT	MKB3
BRPROC20	EREPUNIT	MKB5
BUILDID	EREPSMTR	MRA2

Label	Phase	Location
BUILITBL	EREPRETV	MGA2
BYPSTM	$$BATTN3	GPE1
CANCELIT	SYSBUFLD	PEE2
CANCRTN	PDAID	AGF5
CCH	EREPRDE	NLA2
CCHNDX	EREPCCH	MHB3
CCHPART	EREPCCH	MHB1
CCHREAD	EREPCCH	MHA4
CCHRTRN	EREPCCH	MHB5
CCH00	EREPEDCU	MZA2
CCH10	EREPEDCU	MZA3
CCH20	EREPEDCU	MZA5
CCH30	EREPEDCU	MZB1
CDERRLOG	PDAID	AFK3
CGBJC2	REALDUMP	HQE5
CHANGE1	REALDUMP	HLC1
CHCKIST1	REALDUMP	HKG1
CHCKSTRS	REALDUMP	HNH3
CHEKER	PDAIDITP	CBH5
CHEKMAIN	EREPMNTR	MBB3
CHK	EREPESTR	NGB3
CHK	EREPESWK	NHC1
CHKAAA	PDAID	ACF1
CHKAAA1	PDAID	ACG1
CHKAAA3	PDAID	ACG3
CHKAAA4	PDAID	ACE4
CHKAAA5	PDAID	ACF4
CHKAAA6	PDAID	ACB4
CHKAAA8	PDAID	ACC4
CHKAA2	PDAID	ACC2
CHKATOF	PDAID	ALG5
CHKDATE	EREPESWK	NHA3
CHKDEV	SYSBUFF1	PFB3
CHKFILP	SDPAR	GDB2
CHKFOLD	UCBLOAD	PCD1
CHKIPL	DUMPGEN	HCB3
CHKKEY	EREPMNTR	MBB1
CHKMCRE	PDAID	AFH1
CHKNO	DUMPGEN	HBF5
CHKNO	DUMPGEN	HCF1
CHKNO	DUMPGEN	HCF3
CHKNO	DUMPGEN	HCF5
CHKNO	DUMPGEN	HDG3
CHKNSWRT	SDPAR	GDB1
CHKRESP	EREPTES	NFC2
CHKSYSVS	DUMPGEN	HDB3
CHKTYPE	SYSBUFLD	PBB1
CHKYES	DUMPGEN	HBC5
CHKYES	DUMPGEN	HCC1
CHKYES	DUMPGEN	HCC3
CHKYES	DUMPGEN	HCC5
CHOTC9	PDAID	ALE4
CKASTAT	EREPSPCL	MNC2
CKBSC	EREPADPT	MMB5
CKCHARIN	PDAID	ALF5
CKCQP	PDAIDFTP	DBF2
CKCQP	PDAIDFTT	DDH2
CKCQP	PDAIDGTP	ECC1
CKCQP	PDAIDGTT	EFC1
CKDEVCE1	PDAIDITW	CAA3
CKDEVCE1	PDAIDITP	CBA4
CKDEVCE1	PDAIDITT	CDA5
CKDEVCE1	PDAIDQTW	EKC1
CKDEVCE1	PDAIDQTW	EKC3
CKDEVCE1	PDAIDQTT	ENB1
CKDEVCE2	PDAIDITW	CAB3
CKDEVCE2	PDAIDITP	CBB4
CKDEVCE2	PDAIDITT	CDB5
CKDEVCE3	PDAIDITW	CAC3
CKDEVCE3	PDAIDITT	CDC5
CKDEVST	PDAIDFTP	DBE3
CKDEVST	PDAIDFTT	DDG3
CKDEVST	PDAIDGTP	ECB3
CKDEVST	PDAIDGTT	EFB3
CKDONE	EREPSPCL	MNB1
CKEDIT	EREPTES	NFA5
CKENTRY	PDAIDGTP	ECE2
CKENTRY	PDAIDGTT	EFF2
CKMPX	EREPADPT	MMB3
CKOUTOD	PDAID	ADB1
CKREC	PDLIST	BCC4
CKTR	PDLIST	BCE4
CK2715ID	EREPSPCL	MNA2
CK2715ID	EREPASTA	MPA4
CK2740	EREPADPT	MMB4
CK2790	EREPADPT	MMB2
CLBLUB	REALDUMP	HTB1
CLEAR	PDAIDITP	CBJ5
CLEAR	PDAIDITT	CDH5
CLEAR	PDAIDFTP	DCJ2
CLEAR	PDAIDFTT	DEJ2
CLEAR	PDAIDGTP	ECJ5
CLEAR	PDAIDGTT	EFJ5
CLEAR	PDAIDQTT	ENG3
CLEAR	PDAIDTDT	FBG2
CLEAR	PDAIDTDP	FDF2
CLEARCTR	EREPSMCP	MQA3
CLEARTW	SDAID1	GAJ1
CLKRUNS	SDAID1	GAH1
CLMERK	SDEHR	GLF3
CLOSPRT	SDEHR	GLG3
CLRFILE	EREPMNTR	MCC2
CMPTIME	EREPRDE	NKC1
CMRG	$$BSDAID	GMH2
CNTROL	EREPEDIT	MEC5
CNTROL	EREPEDCP	MUC5
CNTROL	EREPEDES	NED4
CNT00	EREPSMTP	MTC1
CNT05	EREPSMTP	MTA4
CNVTCUU	PDAID	ALF1
CNVTODEC	EREPEDTR	MVA4
CNVT2BIN	PDAID	ALD5
COL71	DUMPGEN	HDB1
COMCPU	EREP	MAG2
COMCUA	EREP	MAF5
COMCUA	EREPSPCL	MNB4
COMDATE	EREP	MAF1
COMDEV	EREP	MAG3
COMEND	EREP	MAG5
COMJOB	EREP	MAF3
COMPARE	EREPESWK	NHC5
COMPARE	UCBLOAD	PCK4
COMPARE1	REALDUMP	HHF1
COMPARM	SDPAR	GDF1
COMPAS	EREPSPCL	MNB3
COMPCD	PDLIST	BCC2
COMPLOW	REALDUMP	HJC1
COMPNEX	SDAID1	GAA3
COMPO	REALDUMP	HJF1
COMPROG	EREP	MAF4
COMPUTE	EREPHIST	NJC1

Label	Phase	Location
COMPY	SDAID1	GAK1
COMRGN	EREP	MAC3
COMTERM	EREP	MAG1
COMTIME	EREP	MAF2
COMVCL	EREP	MAG4
CONFLCT	SYSBUFLD	PBC4
CONT	PDAIDFTP	DBD1
CONTINUE	PDAIDGTP	EEG2
CONTINUE	EREPESPT	MFB1
CONTINUE	EREPHIST	NJC4
CONTINUE	$$BUFLDR	PHE1
CONTSCAN	SDPAR	GCD1
CONTSCAN	REALDUMP	HNC1
CONVDATE	EREPRDE	NKB4
CONVDEV	EREPRDE	NKD4
CONVERT	PDAID	ALE5
CONVERT	SDAID1	GAB3
CONVERT	ESTVUT	LAA2
CONVRT	ESTVUT	LAF4
CONVRT2	ESTVUT	LAF5
CONVTIME	EREPRDE	NKB5
COREDMP1	PDAIDTDT	FAC3
COREDMP1	PDAIDTDP	FCC3
COREDUMP	PDAIDTDT	FAB3
COREDUMP	PDAIDTDP	FCB3
COUNT	EREPSMCP	MQB4
CPTNRETN	EREP	MAB1
CPU	EREPMNTR	MBD4
CQCHCK	REALDUMP	HUB3
CQLOOP	REALDUMP	HUD3
CQLOOP1	REALDUMP	HUF3
CQLOOP2	REALDUMP	HUF4
CQLOOP3	REALDUMP	HUJ3
CRDINP	PDAID	AEB1
CTRINCR	EREPADPT	MMB1
CTRLOC	EREPSM**	MSA4
CUUCNVT	PDAID	ALE1
CUUOKEX	PDAID	ALK1
CUUPUBCK	PDAID	ALH1
CUURESP	PDAID	ABE2
DAED00	EREPED**	MWA5
DAED00	EREPED**	MXA5
DAED05	EREPED**	MXB1
DAED10	EREPED**	MXB2
DAED15	EREPED**	MXB3
DAED20	EREPED**	MXB5
DASD00	EREPED**	MWA2
DASD00	EREPED**	MXA2
DASD05	EREPED**	MWA3
DASD05	EREPED**	MXA3
DASD10	EREPED**	MWA4
DASD10	EREPED**	MXA4
DATARDS	LSERV	KBC1
DATAREAD	LSERV	KBB1
DATE	EREPMNTR	MBC4
DATE	EREPEDIT	MEB3
DATEVENT	SDERR	GKB1
DATE00	EREPSMTP	MTA2
DATE00	EREPED**	NBA2
DBLSCHK	PDAID	AEG2
DBLSTYPE	PDAID	AED3
DEFAULT	EREPMNTR	MCA2
DENS	ESTVUT	LAD3
DEVASSGN	DUMPGEN	HEB3
DEVICE	EREPMNTR	MBD1

Label	Phase	Location
DEVTST	PDAIDITP	CBC1
DEVTST	PDAIDITT	CDC1
DEVTST	PDAIDQTT	EMC1
DISKREAD	EREPSMES	NDB2
DISK10	EREPSMES	NDC2
DMPTST	PDAIDQTW	EKJ3
DOIO	SYSBUFF1	PGH1
DOIT	PDAIDITT	CEE2
DOIT	PDAIDGTW	EBJ3
DOIT	PDAIDGTT	EFG2
DOIT	PDAIDQTW	ELB4
DOIT	PDAIDQTT	ENE4
DONE	REALDUMP	HPB3
DONSET	PDAIDITT	CDF3
DONTSET	PDAIDGTT	EHF4
DONTSET	PDAIDQTT	EMG3
DOPRINT	SDAID1	GAD5
DOPT	PDAIDITP	CCB3
DOPT	PDAIDFTP	DBJ2
DOPT	PDAIDGTP	ECG2
DOPTR	PDAIDFTT	DDC5
DOSLEVEL	EREPEDIT	MEB5
DSKERROR	EREPADPT	MMA5
DUMP	REALDUMP	HGD3
DUMP	EREPEDIT	MED1
DUMPAREA	PDAIDTDT	FAB1
DUMPAREA	PDAIDTDP	FCB1
DUMPTST	PDAIDITW	CAD3
DUMPTST	PDAIDITP	CBB2
DUMPTST	PDAIDITT	CDC2
DUMPTST	PDAIDQTT	EME1
DUPLREQ	SDAID1	GAH4
DVCK	EREPESPT	MFB3
DVCSCAN	EREPSMCP	MQB3
DVCSCAN	EREPSM**	MSA2
DVCSUPRT	EREPSM**	MSA3
DVC00	EREPEDD1	NAA2
DVC05	EREPEDD1	NAA3
EALL	PDLIST	BAJ2
ED**STRT	EREPED**	MWA1
ED**STRT	EREPED**	MXA1
ED**STRT	EREPED**	NBA1
EDCPSTRT	EREPEDCP	MUA1
EDCUSTRT	EREPEDCU	MZA1
EDD1STRT	EREPEDD1	NAA1
EDESSTRT	EREPEDES	NEA1
EDIT	REALDUMP	HVG1
EDIT	EREPEDIT	MDA3
EDIT	EREPEDES	NEA4
EDITBITS	EREPEDIT	MEC3
EDITLOAD	EREPSMCP	MQB5
EDITLOO	EREPEDCP	MUA4
EDITMORE	EREPADPT	MMC3
EDITV00	EREPEDD1	NAB2
EDITV05	EREPEDD1	NAB3
EDITV07	EREPEDD1	NAB5
EDITV15	EREPEDD1	NAB4
EDITV25	EREPEDD1	NAC1
EDIT00	EREPEDTP	MYA4
EDIT00	EREPEDD1	NAA5
EDIT00	EREPED**	NBA5
EDIT00	EREPEDU1	NCA5
EDIT03	EREPEDTP	MYA5
EDIT05	EREPEDCP	MUA2
EDIT05	EREPEDTP	MYB1

Label	Phase	Location
EDIT05	EREPEDES	NEB2
EDIT10	EREPEDCP	MUA3
EDIT10	EREPEDD1	NAB1
EDIT10	EREPEDU1	NCB1
EDIT10	EREPEDES	NEA5
EDIT15	EREPEDU1	NCB2
EDIT20	EREPEDU1	NCB3
EDIT20	EREPEDES	NEB3
EDIT25	EREPEDU1	NCB4
EDIT25	EREPEDES	NEB1
EDTPSTRT	EREPEDTP	MYA1
EDU1STRT	EREPEDU1	NCA1
ELEMEV	SDEHR	GJB3
ENABLE	PDAIDTDT	FAK1
ENABLE	PDAIDTDP	FCK1
END	EREPESWK	NHB2
END	SYSBUFLD	PAE3
ENDBR	PDLIST	BAE1
ENDD	ESTVUT	LAA5
ENDH	EREPESTR	NGC1
ENDIT	REALDUMP	HYB5
ENDJOB	PDSDM	JBE4
ENDJOB	PDSDM	JJH4
ENDMSG	$$BUFLDR	PHC5
ENDPRT	LSERV	KAH5
ENDSD	$$BSDAID	GNB5
ENDT	ESTVUT	LAC5
ENDUMP	REALDUMP	HKB1
ENTCHNG	PDAIDITW	CAD5
ENTCHNG	PDAIDFTW	DAC4
ENTER5	REALDUMP	HQD3
ENTR	PDAIDITT	CDJ3
ENTR	PDAIDQTW	EJE2
ENTR	PDAIDQTT	EMG1
ENTRY	PDAIDITP	CBE4
ENTRYB	PDAIDTDT	FAC1
ENTRYB	PDAIDTDP	FCC1
ENTRYQ	PDLIST	BEF2
ENTRYQ1	PDLIST	BFC3
EOD	EREPRDE	NLA4
EOFRCUT	PDSDM	JFB1
EOFRIN	EREPUNIT	MKC2
EOFTEST	LSERV	KAC5
EOJ	PDAID	AHB4
EOJ	$$BPDAID	AME3
EOJ	PDLIST	BAF2
EOJ	EREPMNTR	MCC3
EOJJT	LSERV	KAE5
EOJRTN	PDAID	AAF5
EOJRTNA	PDAID	AAJ4
ERBLOOP	REALDUMP	HTE5
ERBLOOP1	REALDUMP	HUC1
ERBLOOP3	REALDUMP	HUD1
ERBLOOP4	REALDUMP	HUE1
EREPNAME	EREP	MAA5
EREPSTRT	EREP	MAA1
ERR	DUMPGEN	HDD5
ERRCOND	$$BPDAID	AMF5
ERRCOND	PDAIDITP	CCB5
ERRCOND	PDAIDITT	CEB4
ERRCOND	PDAIDGTP	ECE4
ERRCOND	PDAIDGTT	EFE4
ERRCOND	PDAIDQTT	ENB5
ERRHR	PDAIDITP	CCC3
ERRHR	PDAIDITT	CEF2
ERRHR	PDAIDFTP	DBK2
ERRHR	PDAIDFTT	DDD5
ERRHR	PDAIDGTP	ECH2
ERRHR	PDAIDGTT	EFH2
ERRHR	PDAIDQTT	ENF4
ERRMSG	SDPAR	GBG2
ERROR	PDAIDTDT	FAG5
ERROR	EREPSMCP	MQB2
ERROR	$$BUFLD2	PKC2
ERROR4	EREPESWK	NHB5
ERROR4	EREPHIST	NJC5
ERRSTAT	SDAID1	GAF1
ERR0	PDAID	AJG2
ERR5	DUMPGEN	HDB5
ERR5A	DUMPGEN	HDC5
ESTVUT	ESTVUT	LAA1
EXCP	SYSBUFLD	PEB3
EXCP	$$BUFLD1	PJD1
EXECUNPK	PDLIST	BCE5
EXIT	EREPEDIT	MEF3
EXIT	EREPASTA	MPB5
EXIT	EREPEDCP	MUC3
EXIT	EREPEDTR	MVB3
EXIT	EREPEDCU	MZA4
EXIT	EREPEDD1	NAA4
EXIT	EREPED**	NBA4
EXIT	EREPEDES	NED2
EXIT	EREPIES	NFB1
EXIT	EREPHIST	NJD4
EXIT	SYSBUFF1	PFG5
EXIT00	EREPEDTP	MYA3
EXIT00	EREPEDU1	NCA4
EXIT00	EREPEDES	NEA3
EXIT1	EREPHIST	NJD5
EXNTPROC	EREPEDIT	MEE3
EXNTPROC	EREPUNIT	MKB1
EXTINT	SDEHR	GGH5
FASTCOM	$$BSDAID	GNC1
FASTREC1	$$BSDAID	GNE1
FASTREC2	$$BSDAID	GNB2
FASTREC3	$$BSDAID	GNB3
FETCH	$$BUFLD1	PJH1
FETCH 1	SYSBUFF1	PGJ1
FETCH1	$$BUFLDR	PHG4
FETCH2	SYSBUFLD	PEJ4
FILLUP	PDLIST	BAC1
FINAL	PDAIDFTP	DBF1
FINAL	PDAIDFTT	DDF1
FINAL	PDAIDGTP	EEG4
FINAL	PDAIDGTT	EHG4
FINDFIR	PDLIST	BEE2
FINDFIR1	PDLIST	BFG1
FINDIGNS	PDAID	ACC5
FINDIT	PDLIST	BDB4
FINDSVCS	PDAID	ABF3
FIRST	REALDUMP	HQF5
FIRSTIN	PDLIST	BCD2
FIXING	$$BSDAID	GFA5
FLAGSET	PDLIST	BEC4
FLENTER	PDAIDFTW	DAB2
FLTREC	PDLIST	BEB2
FLTRSET	PDLIST	BAB4
FLUSH	PDSDM	JJG4
FOLLOFF	SDPAR	GCE2
FOLLON	SDPAR	GCF2
FOLLOW	EREPESWK	NHC4

Label	Phase	Location
FORCCHK	$$BSDAID	GFF1
FORCCHK	$$BATTN3	GPG1
FORCDEF	SDAID1	GAF3
FORGET	$$BSDAID	GME3
FORGETIT	$$BATTN3	GPC5
FORMAT	PDAIDFTP	DBG5
FORMAT	PDAIDGTP	EED1
FORMAT	DUMPGEN	HCB1
FORMAT	EREPMNTR	MBD2
FORMAT	EREPESPT	MFB2
FORMAT	EREPESTR	NGB4
FORMAT1	PDAIDGTP	EEE1
FOUND	$$BUFLDR	PHB3
FOUNDIT4	REALDUMP	HND2
FPCHCK	REALDUMP	HUD5
FROMSIO	PDAIDITW	CAC1
FRSTCHK	EREPESWK	NHC3
FSTIMSW	PDAID	AAF2
F1TRCHK	PDAID	AHE4
GENPRINT	EREPRDE	NKC5
GET	PDLIST	BAH3
GETADDRS	PDAIDQTW	ELG2
GETCC	SYSBUFF1	PFD1
GETIGNS	PDAID	AHB1
GETPHAS	EREPRETV	MGD2
GETSUPST	$$BPDAID	AMG1
GETSVCS	PDAID	AJB1
GIVSRCE	EREPMNTR	MBA3
GO	PDAIDTDT	FAB5
GO	PDAIDTDP	FCB5
GOBACK	PDAIDGTW	EAD4
GOBACK	PDAIDGTP	EDJ2
GOBACK	PDAIDGTT	EGJ2
GOOD	EREPESWK	NHC2
GOODTAPE	EREPESTR	NGA3
GOODTAPE	EREPHIST	NJA5
GORTN	PDAID	AGE2
GOTHRU	PDLIST	BDC4
GPACTIVE	EREPSMTR	MRA5
GROUPCK	EREPUNIT	MKB2
GRUPHD	EREPRDE	NLB1
GSVCPRT	PDLIST	BBC2
GSVCTYPE	PDLIST	BBG2
GTST4FF	PDLIST	BBF2
HALF	PDLIST	BEH4
HALF1	PDLIST	BFB3
HAPDAT	SDEHR	GKE1
HDLEV	SDEHR	GKD3
HEADER2	EREPHIST	NJB3
HERE	EREP2715	MLA2
HIGHEND	$$BSDAID	GMD3
HILOSET	EREP2715	MLA3
HISTCK	EREPHIST	NJD2
IGET1ST	PDSDM	JBB5
IGNDEV	PDAIDITP	CBC3
IGNDEV	PDAIDQTW	ELE2
IGNDEV	PDAIDQTT	EMH3
IGNRESP	PDAID	AHD1
IGNRSPCD	PDAID	AHJ3
IGNRTN	PDAID	AGD4
IGNTRCER	PDAID	AHG3
IGTRERCD	PDAID	AKB5
IJBECCH	EREPCCH	MHA2
IJBEIPL	EREPIPL	MJA1
IJBEMCAR	EREPCCH	MHA1
IJBEUNIT	EREPUNIT	MKA1
IJERETV	EREPRETV	MGA1
IMODUNT	PDSDM	JBC3
IMOD001	PDSDM	JJG1
INCRCGPT	EREPEDTR	MVB2
INCREMENT	PDAIDFTW	DAG2
INCRMNT	PDAIDITW	CAF3
INCRSEQ	EREPRDE	NKC2
INDEX	EREPEDIT	MEC2
INDEX	EREPEDCP	MUC4
INDEX	EREPEDES	NED3
INIT	EREPADPT	MMA2
INITAPE	EREPESTR	NGA2
INITAPE	EREPHIST	NJA3
INITIAL	EREPEDIT	MDA2
INITIAL	EREPTES	NFA2
INITIAL	EREPESWK	NHA1
INITLOG	EREPMNTR	MBA4
INITPG	EREPESPT	MFA1
INITPIKS	PDAID	ABC5
INITRTN	PDAID	ADB3
INITZ	PDAIDITW	CAB5
INITZ	PDAIDITP	CBF3
INITZ	PDAIDFTW	DAA4
INITZ	PDAIDQTW	ELB2
INITZ	PDAIDQTT	EMC3
INIT00	EREPSMES	NDA3
INIT00	EREPRDE	NKA2
INIT10	EREPSMES	NDA5
INIT20	EREPSMES	NDA4
INSERT	EREPESWK	NHD2
INSRTEV	SDEHR	GLH1
INSUFF	SDAID1	GAB5
INTCHK	SDEHR	GHB3
INTCHK1	SDEHR	GHC3
INTENTER	PDAIDITW	CAH4
INTERUPT	DUMPGEN	HBB5
INTEXIT2	SDEHR	GHC4
INTEXIT3	SDEHR	GHD4
INTEXIT4	SDEHR	GHE4
INTRUPT	REALDUMP	HFD1
INVCC	SYSBUFLD	PEE5
INVSTE1	REALDUMP	HLB3
IOBCKT	PDAIDITT	CDB3
IOPNLST	PDSDM	JBE2
IOPNOUT	PDSDM	JBJ1
IOPNPDS	PDSDM	JBE3
IORTN	PDAIDQTW	EJB2
IOTREC	PDLIST	BDB2
IOTRFRMT	PDLIST	BDE2
IOTRSET	PDLIST	BAD4
IOUTEND	PDSDM	JJC4
IPDSEND	PDSDM	JJB4
IPL	EREPRDE	NKC4
IPLNDX	EREPIPL	MJA4
IPLPAGE	EREPIPL	MJA3
IPLREAD	EREPIPL	MJA2
IPLRTRN	EREPIPL	MJA5
ISBLOCK	SYSBUFLD	PDC2
ISBLOCK	SYSBUFLD	PDE2
ISDAEXTS	LSERV	KBG3
ISIT4	SDEHR	GKG3
ISMSG	SYSBUFLD	PDC3
ISPRNT	SYSBUFF1	PGG1

Label	Phase	Location
ISSUMSG	SYSBUFLD	PEB1
ISX	SYSBUFF1	PFA2
ITAPE1	PDSDM	JJE1
ITSTDEV	PDSDM	JJB1
ITSTPDS	PDSDM	JBD3
KEEPIGN	PDAID	AEK4
KEEPIRC	PDAID	AEK3
KEYCHK	PDAID	AFA1
KEY14CHK	PDAID	AFF3
KEY15CHK	PDAID	AFG4
KEY2CHK	PDAID	AFB1
KEY3CHK	PDAID	AFC1
KEY4CHK	PDAID	AFG1
KEY5CHK	PDAID	AFE1
KEY6CHK	PDAID	AFG2
KEY7CHK	PDAID	AFF1
KLEER	$$BUFLD1	PJF4
LABLROUT	EREPTES	NFB5
LASTLST	EREPESPT	MFB4
LASTREC	PDLIST	BAG2
LEAVDISP	SDEHR	GHE1
LISTCLB	REALDUMP	HTE1
LISTPRO	EREPEDIT	MEA1
LISTPRO	EREPEDCP	MUA5
LISTPROC	EREPEDES	NEB4
LOAD	EREPEDIT	MED3
LOADER	EREP	MAA2
LOADER2	EREP	MAA3
LOADFCB	SYSBUFF1	PFH2
LOADINP	DUMPGEN	HAB3
LOADIT	EREP2715	MLC3
LOADNAME	EREP	MAA4
LOADOPTN	EREPEDIT	MDA5
LOADOV5	REALDUMP	HQB3
LOADPHAS	EREPEDIT	MDA4
LOADREGS	PDLIST	BEE4
LOADR11	PDAIDTDT	FAD3
LOADR11	PDAIDTDP	FCD3
LOAD1	$$BUFLDR	PHE3
LOAD3	SYSBUFLD	PBF2
LOG	DUMPGEN	HDE5
LOGDEL	PDAID	AED4
LOGINV	PDAID	AGE3
LOGRITE	$$BUFLD1	PJB1
LOOKDESC	EREPRDE	NKD2
LOOKSUB	EREPRDE	NKD3
LOOP	PDAIDGTW	EAE3
LOOP	PDAIDGTW	EAE5
LOOP	PDAIDGTP	EDK1
LOOP	PDAIDGTP	EDK4
LOOP	PDAIDGTT	EGK1
LOOP	PDAIDGTT	EGK4
LOOP	REALDUMP	HPC1
LOOP	EREPUNIT	MKC4
LOOP	EREPESWK	NHB4
LOOP2	$$BUFLDR	PHD1
LOSTHEX	PDAID	AFH5
LSTALNT	SDPAR	GDB5
LSTCDCHK	PDAID	AED1
LSTFILD	EREPESPT	MFC1
LSTRTN	EREPEDIT	MEA2
LSTRTN	EREPEDCP	MUB1
LSTRTN	EREPEDES	NEB5
LTACHCK1	REALDUMP	HNE3
LUBCHCK	REALDUMP	HRB1
LUBLOOP	REALDUMP	HRC1
LUBLOOP1	REALDUMP	HRD1
LUBLOOP2	REALDUMP	HRF2
LUBLOOP3	REALDUMP	HRC3
LUBLOOP4	REALDUMP	HRG3
LUBLOOP5	REALDUMP	HRJ4
LUBLOOP6	REALDUMP	HRE5
LUBLOOP7	REALDUMP	HSB1
LUBLOOP8	REALDUMP	HSB2
LU10	REALDUMP	HSB4
LU9	REALDUMP	HSB3
MCARNDX	EREPCCH	MHB2
MCARPART	EREPCCH	MHA5
MCARREAD	EREPCCH	MHA3
MCARRTRN	EREPCCH	MHB4
MCEVENT	SDEHR	GJB1
MCH	EREPRDE	NLA1
MESSAGE	EREPTES	NFC1
MODN	EREPESPT	MFB5
MONENDB	SDEHR	GGD3
MONENDX	SDEHR	GGE3
MONEND1	SDEHR	GGC3
MONEND2X	SDEHR	GGG3
MONENTPG	SDEHR	GGB1
MONRETPN	SDEHR	GGF3
MONRETPO	SDEHR	GGF1
MONTSTDT	SDEHR	GGJ1
MONTSTM	SDEHR	GKH3
MONTSTMC	SDEHR	GGH1
MOVE	ESTVUT	LAB4
MOVE	EREPSMTP	MTB5
MOVE	EREPESWK	NHD3
MOVECOM	EREPEDIT	MEB4
MOVECOM	EREPEDCP	MUB5
MOVECOM	EREPEDES	NEC4
MOVEHEX	EREPEDIT	MEB1
MOVEMSG	REALDUMP	HNG5
MOVEMSG	SYSBUFF1	PGA1
MOVESAME	REALDUMP	HJH1
MOVEUP	PDAIDGTW	EAJ5
MOVEUP	PDAIDGTP	EEC3
MOVEUP	PDAIDGTT	EHF1
MOVEUP	PDAIDQTW	EKE5
MOVREG	EREPEDIT	MED2
MOVSTAT	PDAIDFTP	DBF3
MOVSTAT	PDAIDFTT	DDH3
MOVSTAT	PDAIDGTP	ECC2
MOVSTAT	PDAIDGTT	EFC2
MOVUP	PDAIDFTP	DBG4
MOVUP	PDAIDFTT	DDG4
MPS	$$BPDAID	AMC3
MSG	EREPESWK	NHA4
MSGLNG	PDAID	AGD1
MULTISEQ	LSERV	KBG5
MVCCTR	EREPEDIT	MED4
MVCCTR	EREPEDCP	MUD2
MVCCTR	EREPEDES	NEE1
NEWCHK	PDAIDITW	CAA2
NEWLINE	PDLIST	BEH5
NEWLINE1	PDLIST	BFC4
NEWLINE1	REALDUMP	HGG1
NEWPAGE	EREPEDCP	MUD1
NEWPAGE	EREPEDES	NED5

Label	Phase	Location
NEWPG	EREPADPT	MMC1
NEWTAPE	EREPESTR	NGA5
NEWTAPE	EREPHIST	NJB2
NEXT	PDAIDFTP	DBE1
NEXT	PDAIDFTT	DDE1
NEXT	PDAIDGTP	EEF4
NEXT	$$BUFLDR	PHB2
NEXTCLB	REALDUMP	HTC3
NEXTFCB	SYSBUFF1	PFJ4
NEXTFCB	$$BUFLD1	PJE4
NEXTIN	PDLIST	BCE2
NEXTONE	PDAIDITP	CCD1
NEXTOPT	EREPTES	NFC3
NEXTPAG1	REALDUMP	HGE3
NEXTRC	EREPSPCL	MNC1
NEXTRC	EREPASTA	MPB4
NEXTRCRD	EREP2715	MLC1
NEXTRCRD	EREPSPCL	MNA5
NEXTRCRD	EREPASTA	MPA5
NEXTUP	SYSBUFLD	PDA1
NOAID	PDAID	AAC4
NOALT	PDAIDITW	CAC4
NOALT	PDAIDGTW	EBH2
NOALT	PDAIDQTW	ELJ1
NOAUTOM	DUMPGEN	HBG5
NOD	PDAID	AGF4
NODBL1	PDAID	AEB4
NODBL2	PDAID	AEA5
NOEDIT	REALDUMP	HPF3
NOGOT	PDAID	ALD4
NOLOG	SYSBUFLD	PED1
NOMORE	PDAID	AGG2
NONEED	PDAID	AAB4
NONMES	PDAID	AKJ1
NONZERO	EREPEDIT	MEB2
NOPHSE	SYSBUFLD	PEC2
NOPHYS	SYSBUFLD	PBB5
NORMAL	$$BPDAID	AMD2
NOSYS	SYSBUFLD	PEG3
NOTEND	EREPESTR	NGC3
NOTEND	EREPHIST	NJD1
NOTINRS	REALDUMP	HNH5
NOTONCUU	PDAID	ALC2
NOTOP	SDAID1	GAG1
NOTPRNT	SYSBUFLD	PAA2
NOTYPE	SYSBUFLD	PEG2
NOTYP2	SYSBUFLD	PBF1
NOX203	UCBLOAD	PCF1
NRFCHK	LSERV	KAB3
NUMBER	DUMPGEN	HBD1
NXTBLOCK	EREPEDTR	MVA2
NXTELEM	SDEHR	GLE1
NXTENTR1	REALDUMP	HLF1
NXTENT1	REALDUMP	HMC5
NXTENT3	REALDUMP	HME3
NXTGRP	EREPUNIT	MKC3
OBR	EREPRDE	NLA3
ODCARD	PDAID	ABD1
ODERRCD	PDAID	AKB4
OKIGN	PDAIDGTW	EBF3
OKIGN	PDAIDGTP	EED4
OKIGN	PDAIDGTT	EHD4
ONCE	EREPESTR	NGB2
ONCE	EREPESWK	NHA5
ONE EXT	LSERV	KBG4
ONLIES	PDAID	ACE5
OPCDMV	LSERV	KAK1
OPENCKER	EREPMNTR	MCA4
OPENFILE	EREPMNTR	MCA5
OPTCHK	EREPTES	NFA4
OPTION	EREPMNTR	MCB2
OPTIONS	PDAID	ACB5
OPTION1	EREPMNTR	MCB3
OPTION2	EREPMNTR	MCB4
OPTION3	EREPMNTR	MCB5
OPTION4	EREPMNTR	MCC1
OPTNFND	EREPSMCP	MQA4
OPTNNODA	EREPSMCP	MQA2
OPTOFF	EREPESWK	NHB3
OR 14	PDSDM	JDD2
OR 17	PDSDM	JCD4
OR 17	PDSDM	JDJ2
ORRCHAR	DUMPGEN	HEE1
OUT	PDAIDITT	CDJ5
OUT	PDAIDGTT	EHJ1
OUT	PDAIDQTT	ENJ3
OUTAREA	PDLIST	BAB1
OUTPUT	PDAIDITT	CEA2
OUTSIDE1	REALDUMP	HLB5
OVERFLOW	PDAIDQTW	EKE5
OVFLIND	PDAIDITP	CBD4
OVFLIND	PDAIDITT	CDD5
OVFLIND	PDAIDQTW	EKF3
OVFLIND	PDAIDQTT	ENE1
PAGE	ESTVUT	LAE1
PAGTABE1	REALDUMP	HKG4
PAGTAB1	REALDUMP	HMF1
PARAMCNT	PDAID	ALB1
PARMCHAR	EREPMNTR	MBC3
PARMCHK	EREPMNTR	MBC1
PARMCHK	EREPMNTR	MCA1
PARMHEX	EREPMNTR	MBC5
PARM2715	EREPMNTR	MBD5
PCKALLP	PDSDM	JEG2
PCKLAST	PDSDM	JEG1
PCKLP	PDSDM	JEB1
PCOMPARE	PDSDM	JHF1
PCPU	EREPRETV	MGB4
PCUA	EREPRETV	MGC5
PDATE	EREPRETV	MGB2
PDVTYP	EREPRETV	MGC3
PEREVENT	SDEHR	GKB3
PGMEVENT	SDEHR	GKB5
PGMINT2	REALDUMP	HKH1
PGMINT3	REALDUMP	HKJ1
PHASE2ND	EREPRDE	NLB3
PIBCHCK	REALDUMP	HPE5
PIBLOOP	REALDUMP	HPD5
PIBLOOP1	REALDUMP	HPF5
PIBSAVDP	REALDUMP	HNH2
PJOB	EREPRETV	MGC1
POSMSG1	SYSBUFF1	PFB4
POSMSG2	SYSBUFF1	PFC3
PPOOLEX1	REALDUMP	HMC1
PREDUCE	PDSDM	JHD1
PREOPEN	PDSDM	JFB3
PREPROC	EREPMNTR	MCB1
PRINT	PDAIDFTP	DBH2
PRINT	PDAIDFTT	DDB5
PRINT	REALDUMP	HVB1

Label	Phase	Location
PRINT	EREPEDIT	MEF2
PRINT	EREPEDCP	MUC1
PRINT	EREPEDES	NEC5
PRINTBR	SYSBUFLD	PED2
PRINTBUF	SDEHR	GLB1
PRINTDK1	LSERV	KBH3
PRINTIT	$$BSDAID	GMB1
PRINTIT	EREPSPCL	MNB2
PRINTIT	EREPEDTR	MVA5
PRINTMSG	SYSBUFLD	PEC5
PRINTOUT	EREPASTA	MPB2
PRNTBLK1	REALDUMP	HKB4
PRNTCR	REALDUMP	HLE3
PRNTCR1	REALDUMP	HLD3
PRNTDATA	EREPSPCL	MNB5
PRNTDATA	EREPASTA	MPB3
PROCESS	EREPHIST	NJB4
PROEXTNT	LSERV	KBJ3
PROGRAM	EREPRETV	MGB5
PRROUT	EREPEDCP	MUC2
PRROUT	EREPEDES	NED1
PRT	EREPEDTR	MVB1
PRTCD	PDAID	AKC2
PRTDATEV	SDEHR	GKG1
PRTERRCD	PDAID	AKF3
PRTEV	SDEHR	GKF3
PRTGET	PDAID	ABB5
PRTMC	SDEHR	GJD3
PRTMCL	SDEHR	GLE5
PRTQUES	PDAID	AKB1
PRTTE	SDEHR	GLG2
PRTVAL	PDAID	AKF1
PRTZERO	SDEHR	GLB5
PSAMESW	PDSDM	JHH1
PSTORE14	PDSDM	JHB1
PTERM	EREPRETV	MGC4
PTIME	EREPRETV	MGB3
PUBCHCK	REALDUMP	HTF3
PUBCHK	PDAID	ALB3
PUBCCMP	PDAID	ALC3
PUBLOOP	REALDUMP	HTH3
PUBLOOP2	REALDUMP	HTJ3
PUNCHRT	DUMPGEN	HEE4
PUPDATE	PDSDM	JHJ1
PUT	PDAIDTDT	FAJ1
PUT	PDAIDTDP	FCJ1
PUTIN	PDAIDITP	CCB1
P2EJ1	EREPRDE	NLB2
P2715	EREPRETV	MGD1
QTAMFRMT	PDLIST	BFE1
QTAMREC	PDLIST	BFB1
QUEUE	EREP	MAC1
RCDPRO	EREPUNIT	MKC1
RCTYP	EREPRDE	NKC3
RDDISK	EREPESPT	MFA4
RDEPHZ	EREPRDE	NKD1
RDERRCK	EREPADPT	MMA4
RDFORMA1	REALDUMP	HNB1
RDLOG00	EREPRDE	NKA3
RDLOG15	EREPRDE	NKA4
RDLOG85	EREPRDE	NKA5
RDNEXTOV	REALDUMP	HME5
RDPPCOL	REALDUMP	HMB1
RDPRT	PDAID	AKD1

Label	Phase	Location
RDREC	EREPASTA	MPA3
RDRESP	SYSBUFLD	PEC1
RDTPE	EREPESPT	MFA3
READ	ESTVUT	LAG1
READ	EREPEDIT	MEE1
READ	EREPEDIT	MEE5
READ	EREPSMES	NDB1
READ	EREPESWK	NHB1
READ	EREPHIST	NJB5
READ	EREPRDE	NKB3
READAAA	PDAID	ACD1
READADDR	EREPSMES	NDB4
READIN	SYSBUFLD	PAD2
READISK	EREPEDIT	MEE2
READLOG	DUMPGEN	HAG3
READLOG	EREPEDIT	MED5
READLOG	EREPEDIT	MEE4
READOPTN	EREPMNTR	MBA5
READRCRD	EREPSPCL	MNA4
READRCRD	EREPASTA	MPA2
READREC	EREPSPCL	MNA3
READREC	EREPTES	NFB2
READ00	EREPSMES	NDE5
READ1	DUMPGEN	HAE3
REALDUMP	REALDUMP	HFB1
REALPRT	$$BSDAID	GMC4
RECREAD	EREPADPT	MMA3
RECUP	EREPTES	NFB3
RECUP	EREPESTR	NGB5
REDUCEA	REALDUMP	HHD1
RELCCW1	REALDUMP	HGB3
RELD	PDLIST	BDJ3
RELEASE	SDEHR	GGB3
RELO	DUMPGEN	HAC1
RELO	SYSBUFLD	PAC1
RELO	$$BUFLDR	PHB1
RELOAD	PDLIST	BAK4
RELOC	PDAIDGTP	EDE5
RELOC	PDAIDGTT	EGD5
RELOC	EREPMNTR	MBA2
RELOC	EREPTES	NFA3
RELOC	EREPHIST	NJA2
REPUNCH	DUMPGEN	HEH4
RESET	EREPESPT	MFC3
RESET	SYSBUFF1	PFK2
RESPOF2	PDAID	AJB3
RESTSET	PDAIDFTP	DBC1
RESTSET	PDAIDFTT	DDC1
RESTSET	PDAIDGTW	EBC3
RESTSET	PDAIDGTP	EDD5
RESTSET	PDAIDGTT	EGC5
RETAR	$$BATTN3	GPC4
RETMCNOS	SDEHR	GJF3
RETOUR	$$BSDAID	GFG3
RETURN	PDLIST	BBB2
RETURN	PDAIDTDT	FAH2
RETURN	PDAIDTDP	FCH2
RFCURR	EREP	MAB5
RFONE	EREP	MAB4
RLDMINUS	EREPSMES	NDA2
RTRVLP2	EREPRETV	MGB1
R7PLUS	SDEHR	GLC3
SAMESW	REALDUMP	HHH1
SAVEDATA	EREPASTA	MPE1
SAVEIT	PDAIDITP	CBF5

Label	Phase	Location
SAVEIT	PDAIDITT	CDF5
SAVENUM	PDAID	ALH5
SAVEPSW	PDAIDFTW	DAC1
SAVER2	DUMPGEN	HAD3
SAVESTUF	PDAIDGTP	EEG3
SAVESTUF	PDAIDGTT	EHG1
SAVEWORD	REALDUMP	HJC3
SAVIT	PDAIDQTT	ENG1
SAVLOOP	REALDUMP	HNC3
SAVLOOP1	REALDUMP	HNB5
SAVLOOP4	REALDUMP	HND5
SAVPSW	PDAIDFTP	DBB3
SAVPSW	PDAIDFTT	DDB3
SAVUM	PDAIDITP	CBA5
SAVUM	PDAIDITT	CDA4
SAVUM	PDAIDGTP	EEB2
SAVUM	PDAIDGTT	EHC1
SAVUM	PDAIDQTT	ENB2
SCAN	EREPSMCP	MQA5
SCAN	EREPSMES	NDC3
SCAN	EREPEDES	NEA2
SCAN	EREPESWK	NHD1
SCANBLK	EREPMNTR	MBB2
SCAN00	EREPSMTP	MTA3
SCAN00	EREPEDTP	MYA2
SCAN00	EREPED**	NBA3
SCAN00	EREPEDU1	NCA2
SCAN05	EREPEDU1	NCA3
SCAN1	DUMPGEN	HAG5
SCAN2	DUMPGEN	HAB5
SCAN20	EREPSMTP	MTB4
SCAN4	DUMPGEN	HBC2
SDMACHEK	SDEHR	GHH4
SEARCH	EREPUNIT	MKA5
SEGTABE1	REALDUMP	HKC4
SEGTAB1	REALDUMP	HME1
SELCMRG	EREP	MAE4
SELECT	EREPRETV	MGA4
SELINCP	EREPRETV	MGA5
SELPARM	EREP	MAE2
SELREGN	EREP	MAE1
SELSWTH	EREP	MAE3
SEL2715	EREP	MAE5
SERVENQ	SDEHR	GJG5
SERVHDL	SDEHR	GJB5
SET	EREPESTR	NGB1
SETBR	PDAIDFTP	DBB1
SETBR	PDAIDFTT	DDB1
SETCKSW	SDEHR	GGD1
SETEXTNT	EREPUNIT	MKB4
SETLSTSW	PDAID	AEE2
SETNLCOM	EREPSMTR	MRB2
SETPAR	SDAID1	GAG3
SETRESET	EREPMNTR	MBB5
SETRTERM	SDPAR	GDB4
SETRTGO	SDPAR	GDG2
SETRTKWD	SDPAR	GDJ2
SETSCNSW	EREPSMTR	MRB3
SETTEST	PDAIDFTW	DAE4
SETUP	PDAIDGTW	EBH3
SETUP	PDAIDQTW	ELJ3
SETUP	SYSBUFF1	PFB1
SETXTENT	EREP2715	MLA4
SHIFT	SYSBUFF1	PFG4
SHIFT	$$BUFLD1	PJD4
SHIFTSET	EREPRETV	MGA3
SHORTFCB	SYSBUFF1	PFC5
SHORTFCB	$$BUFLD1	PJG4
SHORTOUT	PDLIST	BCG5
SHRTMSG	PDAID	AGC1
SIO	$$BPDAID	AMC4
SIOCHK	PDAIDQTW	EKG3
SIOENT	PDAIDGTT	EFA1
SIOENT	PDAIDQTT	EMB2
SIOENTER	PDAIDITW	CAB1
SIOENTSV	PDAIDFTP	DBE2
SIOENTSV	PDAIDFTT	DDG2
SIOENTSV	PDAIDGTP	ECB1
SIOENTSV	PDAIDGTT	EFB1
SIORD4	REALDUMP	HGC3
SIORTN	$$BPDAID	AMB4
SIORTN	PDAIDITP	CBB1
SIORTN	PDAIDITT	CDB1
SIORTN	PDAIDQTW	EJB1
SIORTN	PDAIDQTT	EMB1
SIORT2	REALDUMP	HFJ5
SKPG	EREPESPT	MFC2
SM**STRT	EREPSM**	MSA1
SMESSTRT	EREPSMES	NDA1
SMTPSTRT	EREPSMTP	MTA1
SOLVE	SYSBUFF1	PFK4
SORESCAN	PDAIDTDT	FAC5
SORTER	EREPMNTR	MCA3
SPCLNGRD	PDAID	AGC4
SPCTST1	PDLIST	BAJ1
SPECODE	EREP2715	MLB5
SPLIT	PDAIDTDT	FAH4
SPLIT	PDAIDTDP	FCH4
STAGN	PDLIST	BDJ5
START	PDAID	AAB1
START	$$BPDAID	AMB1
START	SDPAR	GBB1
START	EREPMNTR	MBA1
START	EREP2715	MLA1
START	EREPADPT	MMA1
START	EREPSPCL	MNA1
START	EREPASTA	MPA1
START	EREPSMTR	MRA1
START	EREPEDTR	MVA1
START	EREPESTR	NGA1
STARTIO	$$BUFLD1	PJE1
STARTRUN	EREPRDE	NKB1
STDLBPRT	LSERV	KAJ4
STEP4	REALDUMP	HNE1
STINST	PDAIDFTP	DBF5
STMREGS	PDAID	AAD4
STOPONEV	SDEHR	GGD5
STOPTEST	SDEHR	GGC5
STORECAW	PDAIDTDP	FCC5
STORERJ	SYSBUFLD	PEH5
STORER5	REALDUMP	HHB1
STORE2ND	PDLIST	BBK1
STORLIN	PDLIST	BBJ2
STRTADDR	EREPSMTR	MRA4
SUBCHEK	EREPMNTR	MBB4
SUBSWTH	EREP	MAC2
SUMCPUID	EREP	MAD3
SUMPARM	EREP	MAC5
SUMREGN	EREP	MAC4
SUMSTART	EREPSMCP	MQA1
SUMSWBYT	EREP	MAD4
SUMSWTH	EREP	MAD1

Label	Phase	Location
SUMSWTH2	EREP	MAD2
SUMTPCUA	EREP	MAD5
SUPPRT1	REALDUMP	HMB5
SUPRZ	ESTVUT	LAJ4
SVCCNVT	PDAID	AJC3
SVCENT	PDAIDQTT	EMC5
SVCENTER	PDAIDFTW	DAB1
SVCENTRY	PDAIDGTW	EAD1
SVCENTRY	PDAIDGTP	EDB3
SVCENTRY	PDAIDGTT	EGB3
SVCENTRY	PDAIDQTW	EJB4
SVCENTRY	PDAIDQTT	EMB4
SVCERR	PDAID	AJF2
SVCERRCD	PDAID	AJB5
SVCGET	PDAID	ABB4
SVCIGN	PDAID	ABC3
SVCINIT	PDAIDGTW	EBB3
SVCINIT	PDAIDGTP	EDC5
SVCINIT	PDAIDGTT	EGB5
SVCINIT	PDAIDQTW	ELB1
SVCINIT	PDAIDQTT	EMA3
SVCK	PDAIDGTP	EEE4
SVCPT	PDAIDGTW	EAE1
SVCPT	PDAIDGTP	EDB2
SVCPT	PDAIDGTT	EGB2
SVCRESP	PDAID	AJD1
SVCSCD	PDAID	ABB3
SVDUMP	$$BSDAID	GMB4
SWOFF	REALDUMP	HJB3
SWTCHTST	PDAIDGTW	EBD4
SWTCHTST	PDAIDGTP	EEC5
SWTCHTST	PDAIDGTT	EHC5
SW1	PDAID	AKC4
SW14	PDAID	AFK5
SW15	PDAID	AJC5
SW6	PDAID	AKC5
SW8	PDAID	AKG3
SYSRTURN	SYSBUFLD	PAE4
TAPEIO	EREPHIST	NJA4
TAPEREAD	EREPEDIT	MEF1
TAPEREAD	EREPSMES	NDB3
TAPE10	EREPSMES	NDC1
TAPLABL	LSERV	KBJ2
TDGO	PDAID	ADG5
TDMPREC	PDLIST	BCJ1
TESTIGN	$$BUFLD2	PKE1
TEST1	PDAIDFTW	DAD2
TEST1	PDAIDFTP	DBB4
TEST1	PDAIDFTT	DDB4
TEST1	PDAIDGTW	EAH1
TEST1	PDAIDGTP	EDC2
TEST1	PDAIDGTT	EGC2
TEST1	PDAIDQTW	EJG4
TEST1	PDAIDQTT	EME5
TEST2	PDAIDFTT	DDC4
THEPUT	PDLIST	BAG4
TIME	EREPMNTR	MBC2
TIME	EREPEDIT	MEC1
TIO1	REALDUMP	HGF3
TIO1	REALDUMP	HJD3
TRACKS	EREP	MAB3
TRAILERS	EREPESTR	NGC2
TRAILERS	EREPHIST	NJC3
TRANLOAD	EREP2715	MLC2
TRANSFCH	PDAID	AHF4
TRASDAID	$$BSDAID	GFB1
TRCIGNIC	PDAID	AHF1
TRMODCUR	$$BSDAID	GFE4
TRSAVE	EREP	MAB2
TRTDO	SYSBUFLD	PEE4
TRY	PDAIDITP	CCD3
TRY	PDAIDITT	CEG2
TRY	PDAIDFTP	DCB2
TRY	PDAIDFTT	DEB2
TRY	PDAIDGTP	ECJ2
TRY	PDAIDGTT	EFJ2
TRY	PDAIDQTT	ENG4
TRY	PDAIDTDT	FAD5
TRY	PDAIDTDP	FCD5
TRYPT	PDAIDITP	CCF1
TRYPT	PDAIDFTP	DBG3
TRYPT	PDAIDFTT	DDJ3
TRYPT	PDAIDGTP	ECD2
TRYPT	PDAIDGTT	EFD2
TSTBYT	SDPAR	GBF1
TSTBYTRT	SDPAR	GCB1
TSTFLLP	SDPAR	GDJ1
TSTTYP	SDPAR	GDD1
TST3203	$$BUFLDR	PHK1
TST3211	SYSBUFLD	PBB2
TYPE	EREPMNTR	MBD3
TYPEXCP	SYSBUFLD	PEB5
TYPINRTN	PDAID	AGE4
TYPOUT	SDPAR	GBC1
TYPOUTRT	PDAID	AGB1
TYPRTURN	SYSBUFLD	PAK3
TYP2RTRN	SYSBUFLD	PBD1
UCBLOAD	UCBLOAD	PCB1
UNCKREAD	EREPUNIT	MKA3
UNCKRTRN	EREPUNIT	MKA2
UNCKSKIP	EREPUNIT	MKA4
UNITPUB	SYSBUFLD	PAJ5
UNPACK	PDAIDTDT	FAE3
UNPACK	PDAIDTDP	FCE3
UNPACK1	REALDUMP	HJB1
UNPAKT	PDAIDITP	CCC1
UNPK	PDLIST	BDF4
UNPKHEX	EREPEDIT	MEC4
UNPK1	PDLIST	BCF4
UNRECSET	PDLIST	BCB2
UPDATCTR	EREPSMTR	MRB1
UPDATE	REALDUMP	HHJ1
UPDRCTPR	EREPSMTR	MRA3
UPDT	EREPSMTP	MTA5
UPDTFWRD	EREPSM**	MSB1
UPDTFWRD	EREPSMTP	MTB2
UPDTHWNO	EREPSMTP	MTB1
UPDTLOOP	EREPSM**	MSA5
UPDTLOOP	EREPSMES	NDC4
UPDTWRTN	EREPSM**	MSB2
UPHDCYL	EREPSMCP	MQB1
UPREG	PDLIST	BBG3
VARRPL	SDPAR	GDH3
VERMOD	DUMPGEN	HEB4
VIRTAB	$$BSDAID	GMJ2
VISASN	PDSDM	JJC2
VOLID	EREPRETV	MGC2
VSBBOX	REALDUMP	HXE5
VSCCB	REALDUMP	HWC1

Label	Phase	Location
VSCCB1	REALDUMP	HWE1
VSCCB2	REALDUMP	HWH1
VSCCB3	REALDUMP	HWD3
VSCCB4	REALDUMP	HWC2
VSCCW	REALDUMP	HWJ3
VSCCW1	REALDUMP	HWB5
VSCCW2	REALDUMP	HWC5
VSCCW3	REALDUMP	HWF5
VSCCW4	REALDUMP	HWG5
VSCCW5	REALDUMP	HXB1
VSCCW6	REALDUMP	HXC1
VSCCW7	REALDUMP	HXD1
VSCCW8	REALDUMP	HXE1
VSFIX	REALDUMP	HXF3
VSFIX5	REALDUMP	HXD5
VSFORMBG	REALDUMP	HWB1
VSIDA	REALDUMP	HXF1
VSIDA1	REALDUMP	HXJ1
VSIDA2	REALDUMP	HXC3
VSIDA5	REALDUMP	HXH2
VSPFT	REALDUMP	HYB3
VSPFT1	REALDUMP	HYD3
VSPFT9	REALDUMP	HYE3
VSRPN1	REALDUMP	HXB5
VSSELP	REALDUMP	HYG3
VSSTPT	REALDUMP	HXH5
VSSTPT1	REALDUMP	HYC1
VSSTPT10	REALDUMP	HYB2
VSSTPT2	REALDUMP	HYE1
VSSTPT4	REALDUMP	HYG1
VUPDTV	EREPSMTP	MTB3
WAIT	$$BPDAID	AMF4
WAIT	PDAIDITP	CCG3
WAIT	PDAIDITT	CEK2
WAIT	PDAIDGTP	ECE5
WAIT	PDAIDGTT	EFE5
WAIT	PDAIDQTT	ENK4
WAIT	PDAIDTDT	FBB1
WAIT	PDAIDTDP	FDB1
WAIT	EREPHIST	NJD3
WAITPUT	PDLIST	BAJ4
WANTPRNT	PDAIDGTT	EHH1
WANTPRT	PDAIDITP	CBG5
WANTPRT	PDAIDITT	CDG5
WANTPRT	PDAIDQTT	ENH1
WHICH	PDAIDFTP	DCF4
WHICH	PDAIDFTT	DEF4
WHICH	PDAIDGTP	ECF3
WHICH	PDAIDGTT	EFF3
WHICH	PDAIDQTT	ENC5
WHICHAID	PDAID	AAB2
WHICHONE	PDAID	AAH1
WHICHONE	DUMPGEN	HEC4
WIND	EREPESPT	MFA5
WRAPUP	PDAIDGTW	EBJ5
WRAPUP	EREPRDE	NLA5
WRITE	EREPTES	NFB4
WRITE	EREPHIST	NJC2
WRITMSG	PDSDM	JFF1
WRONGTAPE	EREPESTR	NGA4
WRONGTP	EREPHIST	NJB1
X203	UCBLOAD	PCD2
ZEROTAB	EREPRDE	NKB2
ZERO1	DUMPGEN	HEE5
ZERO2	DUMPGEN	HEF5
ZERO3	DUMPGEN	HEG5

SY33-8554-1

DOS/VS System Serviceability Aids (S370-37) Printed in U.S.A. SY33-8554-1

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)

DOS/VS System Serviceability Aids

READER'S COMMENT FORM

SY33-8554-1

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.

Name

Address

What is your occupation?

How did you use this manual?

As a reference source

As a classroom text

As a self-study text

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)