GY26-3741-3

Systems

DOS Emulator Logic (on IBM System/370 under OS)

Program Number 360C-EU-738 for OS/MFT or OS/MVT

Program Number 5744-AS1 for OS/VS1

IBM

Fourth Edition (July 1972)

This edition applies to Version 2, Level 1, of the IBM DOS Emulator program and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult the latest IBM System/360 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form is provided at the back of this publication for readers' comments. If this form has been removed, address comments to IBM Corporation, Programming Publications, Department 813 LGP, 1133 Westchester Avenue, White Plains, New York 10604.

PREFACE

This publication describes the internal logic of the DOS Emulator program, which runs under the control of the Operating System on IBM System/370 Models 135, 145, and 155. DOS emulation is a combination of the DOS Emulator program and the System/370 DOS Compatibility Feature.

The relationship of the program logic to the logic of the DOS Compatibility Feature is discussed.

DOS emulation enables DOS problem programs to run under control of OS without program conversion. The DOS Emulator program executes as an OS problem program under a nonzero storage protection key.

This publication is intended for use by persons involved in program maintenance, and system programmers who are altering the program design.

HOW TO USE THIS PUBLICATION

This program logic manual supplements the DOS Emulator program listing. The same labels are used on the PLM flowcharts, on the program listing, and within the module descriptions.

Effective use of this publication requires an understanding of the:

- Organization of the publication as a whole
- Organization of the DOS Emulator program into major operations
- Use of the Emulator module directory as a key for locating items related to each module
- Prerequisite and related publications
- Notational conventions used to describe the syntax (or format) of control statements

These topics are discussed below.

Organization of the Publication

In addition to the preface, a table of contents, a list of figures, and a list of abbreviations used in this manual, this publication has these major parts:

- "Summary of Changes," which summarizes the major technical changes reflected for version 2, level 1 and version 2, level 0.
- "Introduction," which briefly describes the capabilities of the DOS Emulator program, program linkages, and the interaction between the Emulator and OS.
- "Method of Operation," which describes the internal logic of the DOS Emulator program and the function of each routine.
- "Program Organization," which consists of the flowcharts for each routine and a summary of the function(s) of each module. (The summary always appears on the page preceding the first flowchart for the module.)

- "Directories," which lists the Emulator program labels alphabetically with their flowchart identifications and the section of the manual in which each label is discussed. In addition, there is a module directory, which can be used to locate key items about each module, a data area directory with information about each data area, lists of Emulator macros and field names.
- "Data Areas," which gives detailed layouts of control blocks, tables, and words to help identify information in storage dumps.
- "Diagnostic Aids," which gives debugging hints, Emulator dependencies on OS, DOS, and hardware, general register assignments, Emulator service aids, and message-to-module relationships.
- "Appendix," which describes the functions and formats of the two hardware instructions and associated lists used by the DOS Compatibility Feature.
- "Glossary," which defines new terms and, for quick recall, some terms defined in prerequisite reading.
- "Index," which is a subject index to this publication.

Organization into Categories

DOS Emulator program operations, defined in the first part of the "Method of Operation" section, fall into the following categories; initialization, synchronous interruptions, asynchronous interruptions, direct-access volume sharing, abnormal end conditions, and service aids.

The second part of the "Method of Operation" section (module descriptions) and "Program Organization" section (module summaries and flowcharts) are arranged in the same sequence for easy cross-reference.

Use of the Emulator Module Directory

The Emulator module directory in the "Directories" section can be used to find the location of the verbal description of each Emulator module, the flowchart(s) for each module, the summary of each module's functions, and other key information concerning that module.

Prerequisite and Related Literature

Effective use of this PLM requires a familiarity with DOS and an understanding of both the IBM System/370 Models 135, 145, and 155 and OS. A working knowledge of System/370 status switching, interruption mechanism, and input/output operations is also helpful. The following publications provide necessary background information or are referenced in the text:

Concepts and Facilities for DOS and TOS, GC24-5030

DOS IPL and Job Control Programs, Program Logic Manual, GY24-5086

DOS Logical IOCS, Volume 3: Sequential and Direct Access DASD Files, GY24-5088

DOS Logical Transient Programs, GY24-5152

DOS Supervisor and Physical Transients, GY24-5151

DOS System Programmer's Guide, GC24-5073

Emulating DOS Under OS on IBM System/370, GC26-3777

IBM System/360 and System/370 SRL Newsletter, GN20-0360

IBM System/360 Principles of Operation, GA22-6821

IBM System/370 Principles of Operation, GA22-7000

OS Data Management for System Programmers, GC28-6550

OS Data Management Macro Instructions, GC26-3794

OS JCL Reference, GC28-6704

OS Master Index to Logic Manuals, GY28-6717

OS Storage Estimates, GC28-6551

OS Supervisor Services and Macro Instructions, GC28-6646

OS System Control Blocks, GC28-6628

OS/VS Data Management Macro Instructions, GC26-3793

OS/VS Data Management for System Programmers, GC28-0631

OS/VS JCL Reference, GC28-0618

OS/VS Master Index of Logic, GY28-0603

OS/VS Supervisor Services and Macros, GC27-6979

OS/VS1 Storage Estimates, GC24-5094

OS/VS1 System Data Areas, SY28-0605

Notational Conventions

A uniform system of notation is used to describe the syntax (or format) of utility control statements. This notation describes which parameters are required and which are optional, the options available in expressing values, and the required punctuation.

Capital Letter Type

In the notation, capital letter type (LIST) is used to indicate specific values that must be coded exactly as shown.

Punctuation

The period (.), comma (,), semicolon (;), equal sign (=), and apostrophe (') are used for punctuation and must be coded as shown. These punctuation marks serve to separate the parameters of a utility control statement.

Brackets

Brackets ([ ]) indicate that the elements and punctuation they enclose are optional. The brackets themselves are for descriptive purposes only, and are not to be coded. For instance

```
value=element1,element2,element3[,element4]
```

indicates that "value=" must be followed by three required parameters (element1, element2, and element3) separated by commas. As indicated by the brackets, element4 is optional and need not be coded. If element4 is coded, however, the comma that immediately precedes it must also be coded.

When choices are available for an optional value, the choices appear in brackets, one choice above another, as follows:

```
value=element0[,element1]
              [,element2]
              [,element3]
```

in the above example, "value=" must be followed by element0. Optionally, element1, element2, or element3 can be coded.

Braces

Braces ({}) indicate a required choice. The braces themselves are for descriptive purposes only and are not to be coded. For example:

```
value={element1}
      {element2}
```

indicates that "value=" must be followed by either element1 or element2.

Underscoring

Underscoring indicates a default value -- a value assumed by the program if no value is coded. For example, given that no optional value is coded in the following:

```
value=[element1]
      [element2]
```

Element1 is assumed.

Ellipsis

An ellipsis (...) is used to indicate that one or more additional parameters or sets of parameters, each identically formatted, optionally can be added to the operand. For example, given the following:

```
value=element1,element2...
```

the ellipsis indicates that everything preceding the ellipsis and following the equal sign can be repeated.

CONTENTS

LIST OF FIGURES

FLOWCHARTS

Initialization

Synchronous Interruptions

New Programming Support

- Release 27 of DOS is supported.
- The DOS residence file (SYSRES) can be shared (two or more Emulators may share the residence file).
- The DOS residence volume can be shared (OS data sets may reside on the DOS residence volume).
- The Emulator service aids provide a new debug command (DIAG) that causes a diagnostic message to be printed when the DOS program restriction (modifying CCWs after an I/O request and before I/O completing) is violated. The service aids also provide a new TRACE command parameter (NOWRAP) that causes the trace table to be dumped each time it is full.

New Device Support

The Emulator now supports the following IBM devices:

- the 3420 Magnetic Tape unit
- the 3330 Disk Storage
- the 3505 Card Reader
- the 3525 Card Punch

SUMMARY OF CHANGES FOR VERSION 2, LEVEL 0

Direct-Access Volume Sharing

With the removal of the Version 1 requirement for private volumes, DOS and OS data sets can now reside on the same DASD volume. Use of this function is optional for one or more volumes in an Emulator job step. The DOS systems residence volume cannot be shared.

ISAM Compatibility

Unmodified DOS programs running under the Emulator can access OS format indexed sequential data sets. By changing their DOS indexed sequential files to OS format, Emulator users can process them using either their old DOS programs or newly-written OS programs.

BTAM Support

The Emulator supports DOS BTAM and permits the use of the HIO instruction for BTAM only.

Improved Initialization

Optionally available are an automatic DOS IPL and abbreviated emulator prompt. Use of these can speed up and simplify the operator action required when an emulation partition is initiated.

Improved Printer Support

If the format of printed output is such that the standard forms tape does not apply, the operator may change format. A new subparameter is provided in the DD statement to call a carriage tape image from the OS image library. This image specifies the physical carriage tape to be mounted on an IBM 1403 Printer.

Service Aids

The Emulator optionally provides routines for use as an aid to debugging.

ABBREVIATIONS USED IN THIS BOOK

Note that field names of data areas created, modified or interrogated by the Emulator ("Field Name Table") and symbolic routine names ("Emulator Module Directory") appear in the "Directories" section.

abn - abnormal
ACCW list - adjust CCW list
ADCONS - address constants
ADDR - address
AJF - adjustment factor
ASYN - asynchronous interruption address
async. - asynchronous
ASYNEXIT - user asynchroncus routine name
ATTN - attention

BDDD - base/displacement
BEBLK - beginning and end block
BG - background (DOS partition)
BISAM - basic indexed sequential access method
BLP - bypass label processing
BOF - beginning of file
BSAM - basic sequential access method
BTAM - basic telecommunications access method
BTR - DOS B-transient phase

c - character
CAW - channel address word
cc - condition code
CCB - command control block
cchh - cylinder/head
ce - channel end
CCW - channel command word (also, in listings, CCW address)
COMP - compare
COMREG - communications region (DOS)
COMTAB - communications table
cond. - conditions
CPU - central processing unit
CRnn - register contents compare
CSECT - control section
CSW - channel status word
CTEXT - COMTAB extension
cuu - channel and unit

DA - direct access
DADSM - direct-access device space management
DASD - direct-access storage device
DCB - data control block
DD - data definition
DEB - data extent block
DECB - data event control block
DIAG block - diagnostic block
DLBL - DASD label
DOS - disk operating system
DOSCOM - DOS communication region (see also COMREG)
DOSLOG - DOS system log
DRILIST - local execution list
DSCB - data set control block
DSECT - dummy section
DSN - data set name
DTF - define the file
DTFDA - DTF direct access
DTFIS - DTF ISAM
DTFPH - DTF physical
DTFSD - DTF sequential disk

EBCDIC - extended binary coded decimal interchange code
ECB - event control block
EMBLKS - Emulator and I/O control blocks
EMU - Emulator region
EMUCONS - Emulator constants area
EOF - end of file
EOV - end of volume
EOX - end of extent
ERP - error recovery procedure
EXCP - execute channel program
EXT - external

F - format (used in context with DSCB); foreground (used in context with DOS partitions; also PIB)
FCB - forms control buffer
FDAD - full direct-access address
FICL - first in class
FID - file ID
FIDBLK - file ID block

hex - hexadecimal
HIO - halt I/O

ID - identification
Init - initialization
Int. - interrupt
I/O - input/output
IOB - I/O block
IOCS - I/O control system
IPL - initial program loader
IPSW - PSW at interrupt
IS - indexed sequential
ISAM - indexed sequential access method
ISBLK - ISAM block
ISFMS - indexed sequential file management system
ISK - insert storage key

JCL - job control language
JFCB - job file control block

KEY - storage protection key

LCTL - load control
LDMD - DOS load module
LEX list - local execution list
LIOCS - logical I/O control system
LMAD - limit address
LOGIOB - constants and pointers associated with DOS system log
LPSW - load PSW
LTK - logical transient key
LUB - logical unit block

MFT - multiprogramming with a fixed number of tasks
MOD - module name
MVT - multiprogramming with a variable number of tasks

n - decimal number
N/A - not applicable
NICL - number in class
NODOS - DOS storage area omitted
NOP - no operation

opcode - operation code
OP/OPCD - CCW operation code
OPR - operation pointer
OS - operating system

p - page number
PC - program check (also, in listings, program interruption address)
PCEXIT - user PC exit routine name
PCIL - private core image library
PIB - program information block
PIK - program interrupt key
PRPGM - problem program
PSW - program status word
PUB - physical unit block

QISAM - queued indexed sequential access method
QSAM - queued sequential access method

R - register
RC - return code
RCCWLIST - adjust CCW data address list
Rnn - register number
RPSW - resume PSW
RTN - routine

SCK - set clock
SD - sequential disk
SEEK - seek address for DASD
shar. - sharing
SIDE - search ID equal
SIO - start I/O
SKE - search key equal
SLI - suppression of possible incorrect length indication
SNS - IOB sense bytes
SSK - set storage key
SSM - set system mask
STCTL - store control
STGCON - constants and pointers associated with staged devices
STIDC - store channel ID
STIDP - store CPU ID
SVC - supervisor call (also, in listings, SVC interruption address)
SVCEXIT - user SVC exit routine name
sync. - synchronous
SYSEMcuu - an Emulator DDname (SYSEM) associated with a
DOS channel and unit number (cuu)
SYSLOG - DOS system log
SYSREC - system recorder file

t - time (hour, minute, second)
TBL - table
TCB - task control block
TCH - test channel
TCU - terminal control unit
TEB - tape error block
TEBV - tape error by volume
TIC - transfer in channel (also, in listings, TIC address)
TIO - test I/O
TIOT - task input/output table
TRKBAL - track balance

UCB - unit control block
UCS - universal character set

VOL - volume
VS - virtual storage
VTOC - volume table of contents

WLR - wrong length record

X - hexadecimal

INTRODUCTION

The DOS Emulator program, IIVEMU, executes as a problem program under OS with an MFT, MVT, or VS1 control program. The DOS Emulator program (hereafter called the Emulator) works in conjunction with the DOS Compatibility Feature. The functional result is that DOS and DOS problem programs are able to execute successfully under control of the Operating System.

DOS EMULATION

DOS emulation is a combination of programming techniques and special machine features that permit DOS to operate under OS. The Emulator and, subsequently, DOS are loaded into an OS partition or region. When the Emulator or OS programs are in control of the CPU, the system is said to be in normal mode, that is, operating with true main-storage addresses.

When DOS programs are executing, the system is in local execution mode, that is, by means of a microprogram and/or hardware, DOS or local addresses are automatically adjusted to program addresses. When DOS is given control by the Emulator, the DOS Compatibility Feature effects the changeover to local execution mode. It does this by adding an adjustment factor (beginning address of DOS) to all addresses contained in each instruction as that instruction is processed (except for instructions that address channels and units). DOS and its problem programs execute in this mode unaided by the Emulator until a hardware interruption occurs.

Since DOS operates in local execution mode, supervisor calls and program interruptions must be intercepted by the DOS Compatibility Feature (a detailed discussion of this feature is in the Appendix) in order to bypass the OS interruption-handling mechanism.

In addition to interruption handling, DOS emulation performs most of the functions normally performed by hardware. I/O operations cannot be performed by DOS emulation and are accomplished by use of OS macros and data management services. Asynchronous interruptions (I/O, external, and machine check) are handled by OS in the usual manner except that control is returned to the Emulator before passing back to DOS. (See "Interruption Action When CPU is in Local Execution Mode" for more details.)

ENVIRONMENTAL CHARACTERISTICS

The Emulator is loaded into an OS partition or region (hereafter called the Emulator region). One of the Emulator's initialization routines (IIVGET) issues the GETMAIN macro instruction to obtain space within its region for DOS and for Emulator and OS control blocks to be used in I/O operations.

DOS is subsequently loaded into the Emulator region, beginning at a 4K boundary. The Emulator and OS control blocks occupy available space between the Emulator and DOS and/or between DOS and the end of the region. Figure 1 shows the Emulator's main-storage environment after DOS has been loaded.

At a minimum, the Disk Operating System consists of a DOS supervisor and a background partition (BG). The two optional foreground partitions (F1 and F2) are shown in Figure 1.

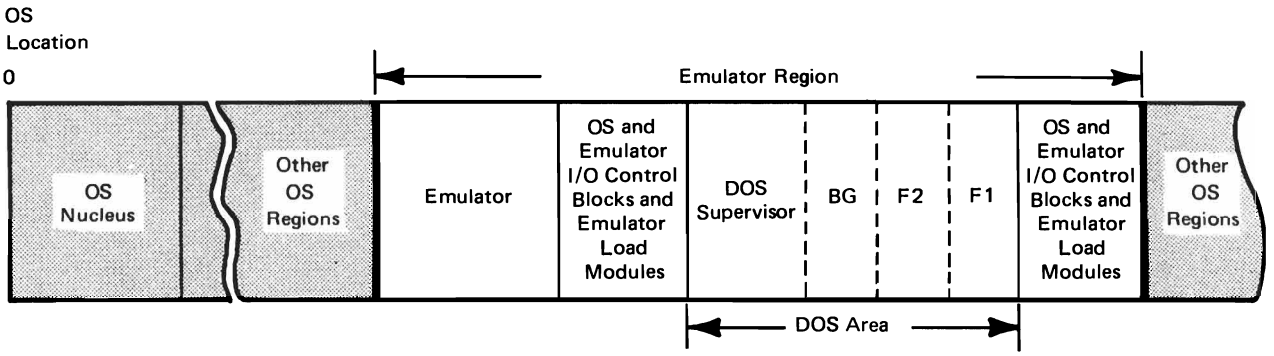


Figure 1. The Emulator's Main-Storage Environment

RESOURCE REQUIREMENTS

"Resource" is defined here as any input/output device, volume, or data set required by the Emulator. A resource may be dedicated, staged, or shared. If a resource is dedicated, it is reserved for the duration of an Emulator job step. If a resource is staged, input/output is placed in the OS input or output stream. If it is shared, OS job steps executing concurrently may access it.

There are three types of input/output devices: unit record, tape, and direct access. The DD statements for a DOS emulation run can specify whether a unit-record device is to be dedicated or whether its input/output is to be staged. A tape device by its very nature must be dedicated to the processing job. A direct-access device cannot be dedicated as such to an Emulator job. It is evident, however, that if a private volume is mounted on a direct-access device, the device is, in effect, dedicated to that volume. Direct-access volumes and data sets may be shared with other processing jobs, or data sets may be dedicated to an Emulator jobstep.

In terms of access method techniques and data set formats, private volumes must be requested in DD statements for a DOS emulation run involving DOS indexed sequential files since they have a different format from OS indexed sequential data sets and are unrecognizable to the OS indexed sequential access method.

OS ISAM volumes can be shared with other OS regions, but OS indexed sequential data sets should be shared for read only type operations.

OS sequential DASD and direct-access data sets and DOS sequential DASD and direct-access files are compatible, so they may be shared by more than one OS region for read only type operations.

To sum up, the DD statements for a DOS emulation run can specify:

- A group of OS devices to be dedicated for the duration of an Emulator jobstep,
- Volumes to be dedicated for the duration of an Emulator jobstep,
- Volumes that can be accessed by concurrent OS jobsteps,

- Files (data sets) to be dedicated for the duration of an Emulator jobstep,
- Files (data sets) that can be accessed by concurrent OS jobsteps, or
- DOS I/O requests for DOS unit record devices to be staged (spooled) into the OS I/O stream.

Dedicated Resources

Unit-record and tape devices. Where a dedicated device is of the same type as that used by DOS in its stand-alone environment, COMTAB (communications table) maps (assigns an OS device for each DOS device specified in a DD statement) the DOS device address to a corresponding, but not necessarily the same, OS device address. In most instances, identical device types are required, but some exceptions are:

Unit Specified on DD Statement	Unit Allowed in PUB Table
3211	1403U
3420	2400
3505	2540R
3525	2540P

In the example

```
//SYSEM180 DD UNIT=2400, data set parameters
```

COMTAB associates the DOS device address 180 with the address of the device that was allocated to the Emulator job. If the absolute unit address, such as UNIT=282 were coded, OS would assign the device at that address, if it were available.

Direct-access volumes. OS can be prevented from allocating temporary data sets on a DOS Emulator volume if the volume is defined as private in the PRESRES member of the library SYS1.PARMLIB, or the volume is offline until OS allocates it for the Emulator, and a DD statement in the following format is specified:

```
//SYSEM190 DD VOL=(PRIVATE,SER=111111),UNIT=2311,DISP={OLD}
                                                    {SHR}
```

where 111111 is the volume serial number of a private pack mounted on an IBM 2311 Disk Storage Drive. Because of the absence of the DSN parameter, this statement allows DOS access to any valid address on the entire volume by means of the Emulator end-of-extent appendage. See the module description for IGG019SA for further details.

Other DOS Emulators are prevented from using space on that volume by the use of ENQ and DEQ macro instructions. Other OS regions are unable to access or create any data sets on that volume, unless that volume is specifically requested in the VOL parameter of a DD statement.

Note: Two or more DCBs should never be opened concurrently for output to the same data set on a direct-access device. If exclusive control of the data set is not maintained and another DCB is opened before the current DCB is closed, the updated records can become permanently inaccessible.

DASD data sets. DASD data sets can be made private when the DISP parameter is coded as shown in this example:

```
                                                    {OLD}
//SYSEM190 DD VOL=SER=222222,UNIT=2311,DSN=A,DISP={NEW}
                                                    {MOD}
```

In the above example, if data set A were on a volume with multiple data sets, it would be the only data set on that volume inaccessible to other OS regions. The DISP parameter does not apply to DOS indexed sequential files because these files must always be processed on a private volume.

Staged Resources

Unit-record devices need not be dedicated to DOS emulation. Instead, a DOS emulation jobstream can request the use of the OS input stream and/or the use of the OS output stream. To take advantage of the former option, DOS program input is inserted between a //SYSEMcuu DD DATA card and a /* card in the DOS emulation jobstream. (The channel and unit (cuu) must represent a DOS PUB (physical unit block) table entry associated with a card reader assignment. The result of staging DOS input is that the OS reader/interpreter spools DOS program input to a DASD from which it can later be read.

DOS output can be staged if this card is inserted in the DOS emulation jobstream:

```
//SYSEMcuu DD SYSOUT=output class
```

(The cuu must represent a DOS PUB table entry associated with a card punch or printer assignment, and SYSOUT must indicate an output class.) OS will then provide the following services:

- When the Emulator is initiated, space will be reserved for DOS program output on a DASD.
- After DOS processing has ended, the output will be transcribed by a SYSOUT writer to an appropriate output device.

Shared Resources

In addition to being able to designate private volumes (discussed in "Dedicated Resources"), the Emulator is able to designate public volumes, to dedicate data sets to a particular DOS emulation run (also discussed in "Dedicated Resources") and to share data sets with other OS regions.

Since unit record and tape devices cannot be shared, this section will discuss only direct-access devices and, more specifically, DASD volumes and data sets. The discussion will not include DOS indexed sequential files, which must be processed on private volumes.

Both OS and DOS data sets can reside on the same volume. During emulation, OS performs data-set space allocation and DOS is prevented from performing space allocation. A data set, with the exception of OS indexed sequential data sets, must reside on one volume.

Figure 2 shows the Emulator methods of accessing DASD data sets. If a volume is not shared, the program may access any OS sequential DASD or direct-access data set, any DOS sequential DASD or direct-access file, or any DOS indexed sequential file. If a volume is shared, the Emulator program takes one of two paths. If an OS sequential DASD or direct-access data set or a DOS sequential DASD or direct-access file is being accessed, it is routed through Emulator module IIVDVS; if an OS indexed sequential data set, it is routed through Emulator module IIVIS. (For more details, see "Direct-Access Volume Sharing.")

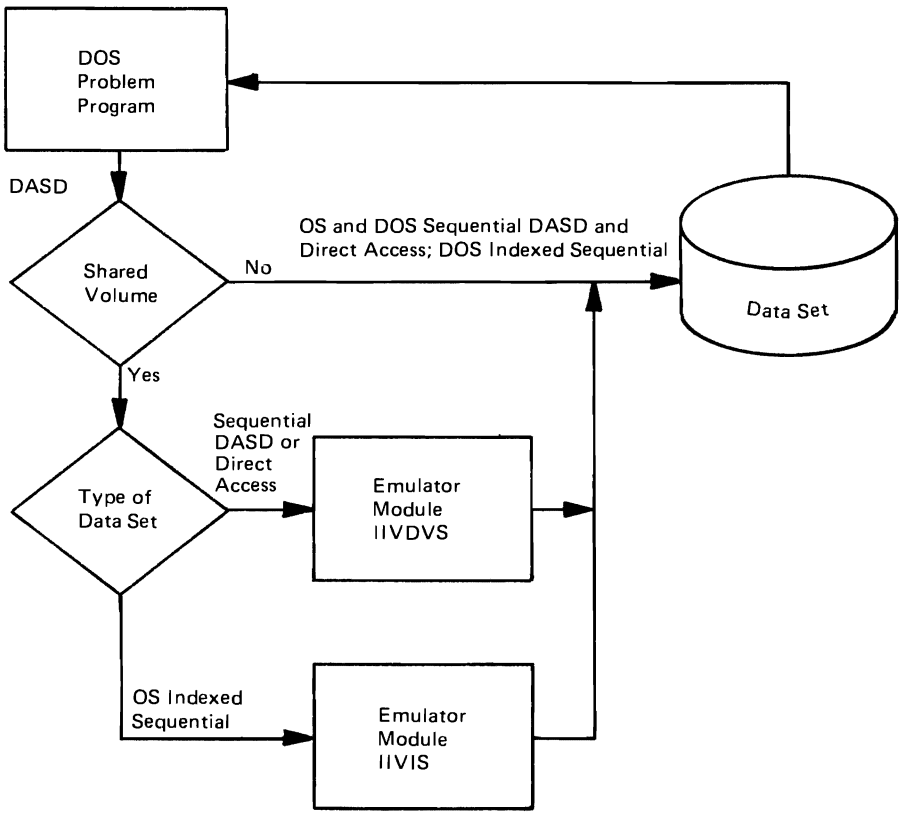


Figure 2. Methods of Accessing DASD Data Sets

OS direct-access and sequential DASD data sets and DOS direct-access and sequential DASD files. One DD statement must be provided for each data set or file on a shared volume. The DD statement must include a nontemporary data set name (dsname). The first DD statement must indicate the cuu contained in the DOS PUB table for that data set. Subsequent DD statements describe other data sets located on the volume defined by the first DD statement. DOS control statements (such as DLBL, EXTENT, and ASSGN) are still required for these files, but the extent information in the EXTENT statement need not be valid (however, it must be syntactically correct) since the Emulator overrides this information with the extents supplied by OS.

Typical DD statements for specific volumes are:

```
//SYSEM132 DD DSNAME=DOS.FILE.IDA,UNIT=2314,VOL=SER=123456
//anything  DD DSNAME=DOS.FILE.IDB,UNIT=2314,VOL=REF=*.SYSEM132
```

Typical DD statements for nonspecific volumes are:

```
//SYSEM132 DD DSNAME=DOS.FILE.IDA,UNIT=2314,DISP=(NEW,DELETE),SPACE=(CYL,(4,2))
//anything DD DSNAME=DOS.FILE.IDB,UNIT=2314,VOL=REF=*.SYSEM132,
            DISP=(NEW,DELETE),SPACE=(TRK,(10,3))
```

Note that DSNAME= must be followed by the name of the data set which must be the same name as coded in the file ID field of the DLBL statement. In addition, if DISP=SHR is coded, data sets may be shared. (If a data set is shared with another OS task, both tasks can read only.)

If two or more data sets on the same volume are used in a job step (see last example), the SYSEM ddname must be coded for the first data set; thereafter, a unique ddname not beginning with SYSEM must be coded for subsequent data sets. The DD statements are tied together when they specify the same volume serial number, when the SYSEM ddname is referred to, or when a SYSEM ddname with a nonspecific volume request is referred to.

OS indexed sequential data sets. The Emulator allows DOS problem programs to create and access OS indexed sequential data sets through OS indexed sequential access methods. (See "File Requirements" for a possible

The OS volume(s) on which an OS indexed sequential data set resides is described to the Emulator by the SYSEMcuu DD statement for the volume. The dsname is used to map to the proper OS data set by means of the Emulator ISAM mapping routine (IIVIS). The same DOS control statements mentioned in the discussion of direct-access and sequential DASD data sets are also required for OS indexed sequential data sets.

Typical DD statements for specific volumes for indexed sequential data sets are:

```
//SYSEM132 DD DSN=DOS.ISAM.FILEA(INDEX),UNIT=2314,DISP=NEW,
//            VOL=SER=111111,DCB=DSORG=IS,SPACE=(CYL,2)
//         DD DSN=DOS.ISAM.FILEA(PRIME),UNIT=2314,
//            VOL=SER=222222,DCB=*.SYSEM132,SPACE=(CYL,20)
//SYSEM133 DD DSN=DOS.ISAM.FILEB,UNIT=2314,DISP=OLD,
//            VOL=SER=333333,DCB=DSORG=IS
//anything DD DSN=DOS.ISAM.FILEC,UNIT=2314,DISP=OLD,
//            VOL=REF=*.SYSEM133,DCB=DSORG=IS
```

Typical DD statements for nonspecific volumes for indexed sequential data sets are:

```
//SYSEM132 DD DSN=DOS.ISAM.FILEA(INDEX),UNIT=2314,
//            DISP=NEW,DCB=DSORG=IS,SPACE=(CYL,2)
//         DD DSN=DOS.ISAM.FILEA(PRIME),UNIT=2314,
//            DCB=DSORG=IS,SPACE=(CYL,20)
//anything DD DSN=DOS.ISAM.FILEB,UNIT=2314,DISP=OLD,
//            VOL=REF=*.SYSEM132,DCB=DSORG=IS
```

EMULATOR/OS INTERACTION

This section discusses parts of OS that are used by the Emulator.

OS Macros

The Emulator communicates its service requests to OS with the OS macros listed in "Diagnostic Aids."

Emulator Appendages

To circumvent certain automatic functions and restrictions imposed by OS (see the module descriptions for IGG019SA and IGG019S1 for more details), two input/output appendages are required by the Emulator. Both appendages operate in the supervisor state under a protection key of 0. These modules

in the supervisor state under a protection key of 0. These modules automatically gain control at the appropriate time if the last 2 characters of their 8-character names are specified in the DCB macro instruction.

Requesting Bypass Label Processing (BLP)

The OS open routine, which is called when the Emulator issues an OPEN macro, must not process DOS tape labels. (DOS does its own label processing.) To prevent OS from processing DOS labels, BLP must be coded in the LABEL parameter of every DD card for a DOS tape volume. The OS Open routine will then find that the BLP bit is on in every JFCB for tape volumes, and it will bypass label processing.

Note: JFCBs are built by the OS reader/interpreter. All IBM reader procedures specify that the reader/interpreter is to ignore BLP if it is encountered on a DD card. Consequently, the BLP bit will not be on in a tape JFCB unless the user has modified the IBM reader procedure or written his own.

OPERATIONAL CONSIDERATIONS

Staged Output Considerations

The staged output of DOS jobs in a given emulation run is transcribed from a DASD to a printer or punch not piecemeal, but all at once, after the emulation run has terminated. Enough space to receive the combined output of the emulated DOS jobs should be reserved on the DASD. If more space than the system's default is needed, the SPACE parameter can be coded on the appropriate DD statements. The Emulator job is abnormally terminated if insufficient space is allotted.

Printer Overflow Simulation

DOS receives no indication that staged output to a printer is really output to a DASD with a later transcription to an OS printer. The Emulator, therefore, must simulate printer overflow indications when DOS output is staged to a printer.

The user may specify the length and format of the physical forms-control tape that will be mounted on nonFCB printers, such as the 1403, at the time the system output writer transcribes the hard copy to the printer. To do so, the user must allocate and catalog the image as a member of SYS1.IMAGELIB. The user forms-control image must be assembled and link-edited into SYS1.IMAGELIB before executing the Emulator. See the SETPRT macro in Data Management for System Programmers for OS or OS/VS for details.

The Emulator creates a forms-control image from the OS 3211 FCB image. If the DD statement FCB parameter is not specified, the Emulator standard forms-control image applies.

Additionally, a DOS dynamic load of the FCB will override user specifications. Special commands other than READ FCB and LOAD FCB will be rejected.

The printer overflow default image describes a standard-size page of 66 lines. The image is formatted with five lines of space at the top and bottom and 12 channels in between, distributed four lines apart. If, for a nonFCB printer,

a printer overflow indication is required by a DOS problem program, the physical carriage tape should conform to this format or the printed results will be unpredictable.

Emulator Separator Feature

The user may stage the output of a Disk Operating System being emulated that has more than one printer or card punch assigned to it by coding a DD statement for each DOS printer or punch assignment for which output will be staged. For example:

```
//SYSEM00E DD SYSOUT=A
//SYSEM01E DD SYSOUT=A
//SYSEM00D DD SYSOUT=B
//SYSEM01D DD SYSOUT=B
```

The Emulator's separator feature writes three separator cards before the punched output in each stacker of each staged punch and a separator page before the printed output of each staged printer. At the top of a separator page is the message:

```
IIV275I SYSEMcuu
```

where cuu identifies the channel and unit of the DOS staged printer.

Each separator card is punched in the following format:

Columns 1-34	hexadecimal FFs (12-11-0-7-8-9 punches)
Columns 35-36	blanks
Columns 37-44	the DOS address of the staged device (SYSEMcuu)
Columns 45-46	blanks
Columns 47-80	hexadecimal FFs

Note: The coding technique shown above for multiple printers and punches is also used to stage multiple card readers.

Data Set Requirements

DOS indexed sequential files. Only DOS indexed sequential files must be converted from DOS to OS format to allow them to be shared with other OS regions. Other types of DOS files need not be converted (see "Sequential DASD and direct-access data sets").

The left side of Figure 3 shows two DOS programs accessing a DOS data set before any conversion takes place. Program A is recoded to be run under OS (see right) and is now able to access any OS or OS-compatible data set, that is, an OS or DOS sequential DASD or direct-access or an OS indexed sequential data set. Program B, still in DOS format, may access an OS or OS-compatible data set only with the intervention of the Emulator. The DOS file may also be converted to OS format; in fact, a DOS indexed sequential file must be converted to allow OS and DOS volume sharing. If DOS indexed sequential files are changed to OS format, they can be processed by either their old DOS programs or newly-written OS programs.

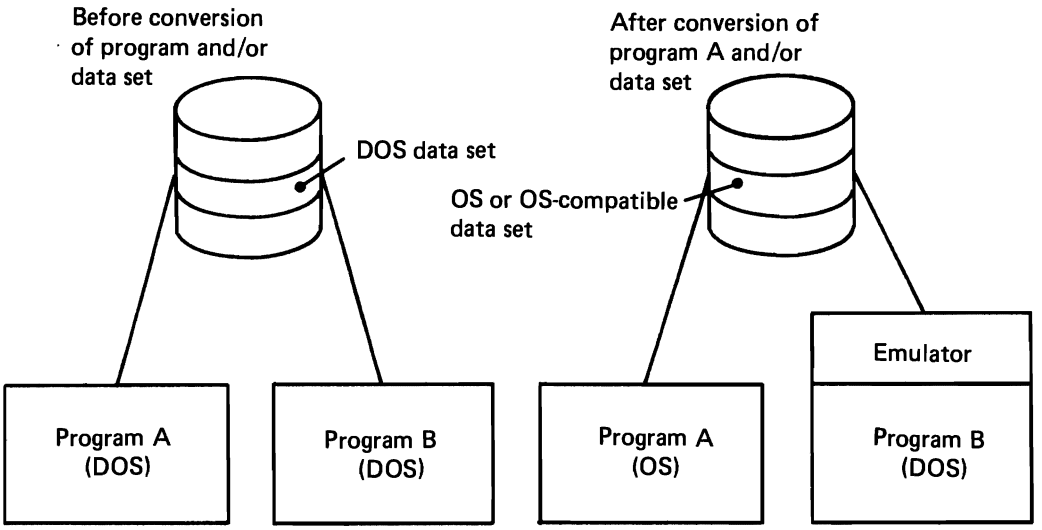


Figure 3. Two Programs Accessing a Data Set Before and After Conversion

Note: For more details concerning program and/or data set conversion, see Emulating DOS Under OS on IBM System/370, GC26-3777.

OS indexed sequential data sets. The Emulator contains a module (IIVIS) to allow a DOS program executing under the Emulator access to an OS indexed sequential data set if its logical record format is compatible with the design of the DOS problem program. In general, the only limitations on indexed sequential data sets relate to OS capabilities that do not exist in DOS.

An operational consideration when accessing OS DASD data sets with a DOS program under the Emulator is that OS optionally supports record deletion and also dummy records for formatted fixed-length records with keys. Dummy records must be recognized and record deletion handled by the DOS program.

Sequential DASD and direct-access data sets. Sequential DASD and direct-access data sets need not be converted. Module IIVDVS allows a DOS program executing under the Emulator to access an OS sequential DASD or direct-access data set or DOS sequential DASD or direct-access file on a shared volume.

The only limitations in a DOS program accessing an OS data set relate to OS capabilities that do not exist in DOS. For example, DOS does not maintain some identifier (format 1) DSCB data set descriptor fields used by an OS sequential DASD data set. For normal OS processing, this information can be supplied on the DD statement; however, utility functions such as those performed by IEHMOVE ignore information supplied in this manner. In addition, DOS FORTRAN unformatted records may not be acceptable to OS FORTRAN or data management.

Another problem that might be encountered by a DOS direct-access file is that nonpreformatted DOS files must be processed as undefined under OS. In addition, a DOS user cannot allocate data sets on a DASD (that is, code a physical address as a constant) under OS as OS does its own space allocation. (However, physical addresses may be used if absolute track allocation is used under OS.) Also, the 'M' of 'MBBCCHHR' (where M = extent, B = bin, C = cylinder, H = head, and R = record number) is defined differently in OS and DOS and will be compatible only for single-extent volumes.

BTAM. BTAM operates without change under emulation as a DOS access method, using DOS I/O error recovery and statistical recording procedures. Since the Emulator runs as an OS job, OS jobs running independently may include OS BTAM. Line groups are defined at the DOS level. To OS, each line is a line group consisting of one line. One OS DD statement must be included for each line to define the device (line) to the Emulator. The user must define to OS the same type, attributes, and features for the lines, control units, and devices that DOS has defined for its corresponding lines. For example,

```
//SYSEM068 DD UNIT=069
```

relates DOS line 068 to OS line 069. All characteristics of line 069, including device type, adapter type, TCU (terminal control unit) type and features, and line type, must be the same as those DOS expects for line 068. Lines used in the emulated DOS job must be dedicated to that job.

PHYSICAL CHARACTERISTICS

The basic Emulator module (IIVEMU) resides on the system program library (SYS1.LINKLIB) or on a private library containing an overlay structure and load modules (see Figure 4). (For further information concerning private libraries, refer to Supervisor Services and Macro Instructions (ATTACH macro), Data Management for System Programmers, and the JCL Reference, for OS or OS/VS.) The overlay structure is composed of a root segment and five overlay segments. The root segment is loaded first and given control by OS. IIVINT is the first module in the overlay segment to be loaded. When there is need for one of the other modules in the overlay segment, it is loaded over IIVINT, and so on.

MAIN-STORAGE REQUIREMENTS

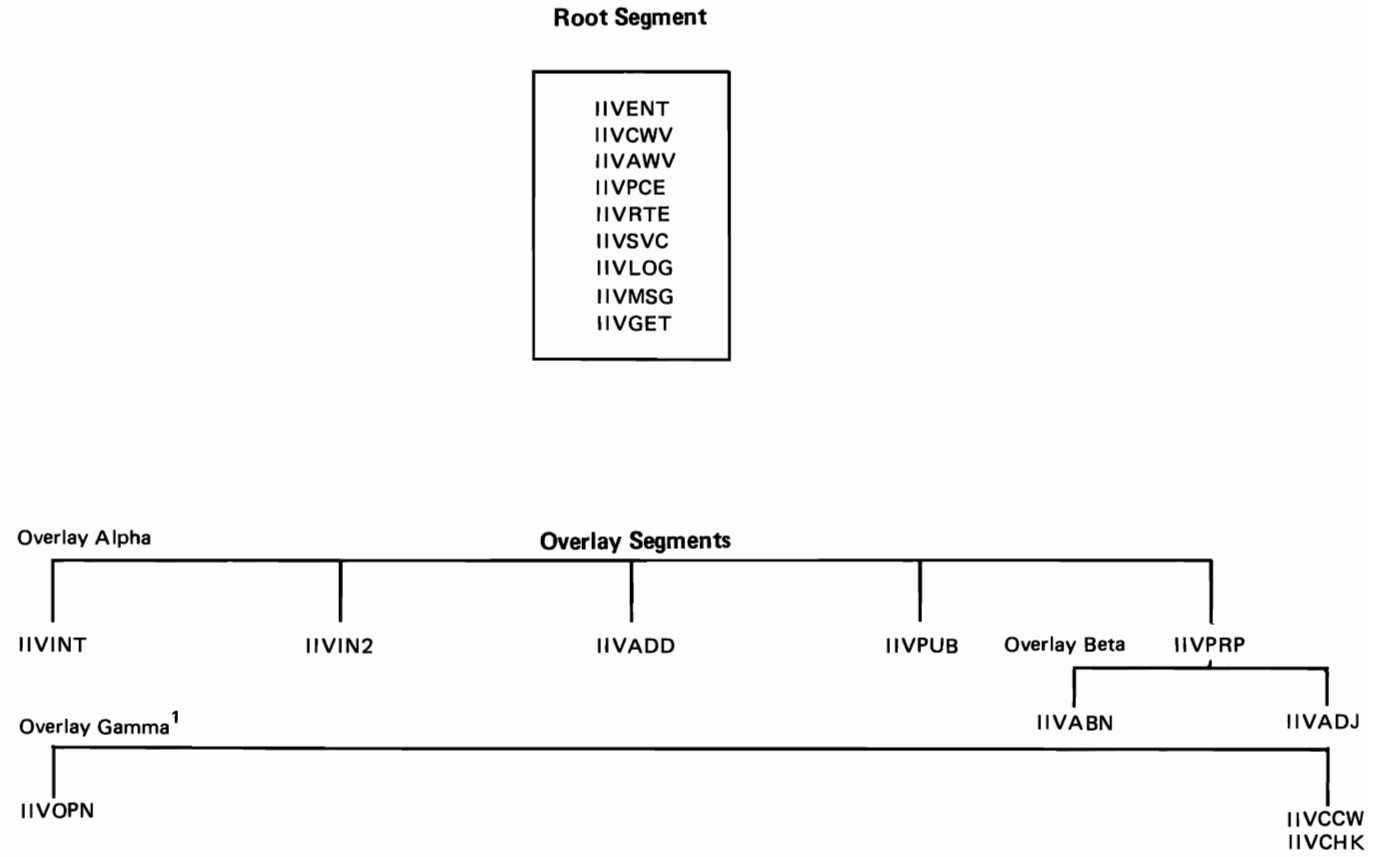
The minimum MFT or OS/VS1 storage requirement for a DOS emulation run is 23K bytes plus the size of the Disk Operating System to be emulated. An additional 6K bytes are required with an MVT control program because of the way MVT storage is managed. (Refer to Storage Estimates for OS or OS/VS for OS region requirements.)

Main-storage requirements are further increased by each of the following conditions that applies:

- If more than ten devices are used, add 250 bytes for each additional device (for I/O tables such as the DCB, IOB, COMTAB entry, etc.).

  Note: Do not include the 96 bytes required for the DCB. This value is already accounted for in the 250 bytes specified above.

Overlay Structure



Load Modules				
Loaded By	**Module Names**			**Required For**
IIVOPN	IIVFCB	IIVPU1	IIVRD2	Staged I/O
	IIVPOV	IIVPU2	IIVRD3	
	IIVPR1	IIVRD1	IIVRP1	
IIVIN2	IIVRAS			Service Aids
	IIVSDT	IIVSTG		Staged I/O
IIVPUB	IIVDVS	IIVIS		Shared Resource
	IIVGR2	IIVVIO		Allocation
IIVMSG	IIVMG1	IIVMG2	IIVMG3	Message Output
IIVPRP	IIVRAS			Service Aids
IIVRAS	IIVRCP	IIVSNP		Service Aids
IIVRCP	IIVACI	IIVPCI	IIVSCI	Service Aids
OS at Open	IGG019SA	IGG019S1		Appendages

[1] Overlay gamma occupies the region immediately following the largest overlay segment of overlay alpha.

Figure 4. Overlay Structure and Load Modules of IIVEMU

- If a direct-access file other than indexed sequential on a shared device is specified, 7.2K bytes are required. This figure includes space for up to five files. Add 175 bytes for each file that exceeds five files.

- If an indexed sequential data set is specified, 6.8K bytes are required for the data set (2.4K bytes for volume sharing and 4.4K bytes for ISAM), plus 400 bytes for each ADD or LOAD type DTF specified and 625 bytes for each RETRVE or ADDRTR type DTF specified. In addition, add the number of bytes required for the DCB parameters used for each data set. (See Storage Estimates for OS or OS/VS, and the information in Figure 5 to determine the number of bytes needed for the DCB parameters used.)

Number of Bytes Required for Indexed Sequential Data Sets

Modules for Volume Sharing	ISAM	ADD or LOAD	RETRVE or ADDRTR	DCB Parameters Used	Additional Indexed Sequential Data Sets
2.4K +	4.4K +	400 +	625 +	*n* Bytes +	*n* Bytes

Number of Bytes Required for Files on a Shared Volume and Indexed Sequential Data Sets

Modules for Volume Sharing	Modules for Files on Shared volume	ISAM	ADD or LOAD	RETRVE or ADDRTR	DCB Parameters Used	Additional Indexed Sequential Data Sets	Additional Files on Shared Volume
2.4K +	4.8K +	4.4K +	400 +	625 +	*n* Bytes +	*n* Bytes +	*n* Bytes

DTF Type	DCB Parameters Used	
LOAD	MACRF=PM, RECFM=F or FB, KEYLEN, LRECL (RKP take their values from the DTFIS)	
ADD	MACRF=WAC, RECFM=F or FB	
RETRVE	DCB1: MACRF=GL, PU, and SK or SI	DCB2: MACRF=RUSC, WUC
ADDRTR	DCB1: MACRF=GL, PU, and SK or SI	DCB2: MACRF=RUSC, WUAC

Notes:
Parameters that do not affect storage requirements have been omitted.
Two DCBs are used for the RETRVE and ADDRTR DTF types. DCB1 uses QISAM, DCB2 uses BISAM.
BLKSIZE takes its value from the DTFIS in all cases.
BUFNO=2 (the default value) in all cases.

Figure 5. Storage Estimates for ISAM

- If files on a shared device and indexed sequential data sets are specified, 7.2K bytes are required for device sharing plus 4.4K bytes for ISAM plus the number of bytes needed for DTF type and DCB parameters used.
- If any DD statement specifies DD DATA, DD *, or DD SYSOUT, add 1850 bytes (for the staged I/O routines).
- For each unique device type defined by a DD DATA, DD *, or DD SYSOUT statement, add 256 bytes (for the device command code translate table).
- If any DD statement specifies DD DATA, DD *, or DD SYSOUT, add the number of bytes used by QSAM. This number is variable and depends upon which, if any, QSAM modules are resident in the particular Operating System being used. Refer to Storage Estimates for OS or OS/VS, and the information in Figure 6 to arrive at this figure.
- If the service aids are used, 14K should be added to the storage estimates for the Emulator.

Number of Bytes Required for QSAM Data Sets Used in the Emulator

QSAM Modules of OS	DCB Parameters of Input Stream	DCB Parameters of Output Stream	Additional DCB Parameters of I/O Stream
n Bytes	+ *n* Bytes	+ *n* Bytes	+ *n* Bytes

OS Stream Used	DCB Parameters Used
OS Input Stream	DSORG=PS, RECFM=FB, MACRF=GL
OS Output Stream	DSORG=PS, RECFM=VBA, MACRF=PL

Notes:
BUFNO=2 for output. For input, the reader procedure value is used.
For output, BLKSIZE=9 plus the maximum number of bytes that can be transferred to the unit record device. The reader procedure value is used for input.
Buffering is simple. Scheduling is normal.

Figure 6. Storage Estimates for QSAM

METHOD OF OPERATION

To become completely familiar with the method of operation of the Emulator, the user of this manual must have a working knowledge of System/370 status switching, interruption mechanism, and input/output operations. An understanding of the DOS Compatibility Feature is also helpful. (See the Appendix for a complete description of this feature.) This section describes each of the four major operations of the Emulator and each Emulator module.

One of the four major operations of the Emulator is to initialize the OS region in which it resides. It accomplishes this by verifying parameters given on the DD statements, establishing the DOS storage area, initializing Emulator tables, and doing an initial program load of DOS. It concludes basic initialization procedures by passing control to DOS.

The second major operation of the Emulator is to deal with interruptions that occur during operation of DOS programs. These include synchronous (supervisor call and program check) and asynchronous (input/output, external, and machine check) interruptions.

The third major operation of the Emulator allows the user additional capabilities on an optional basis:

- Ability of multiple OS regions to concurrently access the same direct-access volume.
- Ability of unmodified DOS programs to access OS-formatted indexed sequential data sets.

The final major operation of the Emulator is to either terminate a job running in a given DOS partition or terminate any operations being performed by the Emulator. Termination takes place when serious user or program errors are detected.

These major operations are required (except the third, which is optional) to achieve the Emulator's single functional objective, which is to enable DOS problem programs to be run on a System/370 under OS. Refer to Figure 23 for an overall functional diagram of the Emulator.

INITIALIZATION

The Emulator is an OS problem program invoked by OS JCL. It receives control at entry point IIVENT. The address of the user parameter area (which contains operands from the PARM parameter of the Emulator EXEC statement) is passed to the Emulator at this time. Initialization routines verify parameters, establish DOS storage, initialize Emulator tables, and perform DOS initial program load.

Figure 7 shows the major steps in initializing the Emulator region and shows control passing from one routine to another. (Figure 16 entitled "OS Region at Beginning of DOS IPL" shows a map of the Emulator region before initialization. Figure 1 entitled "The Emulator's Main-Storage Environment" shows a map of the Emulator region after initialization.)

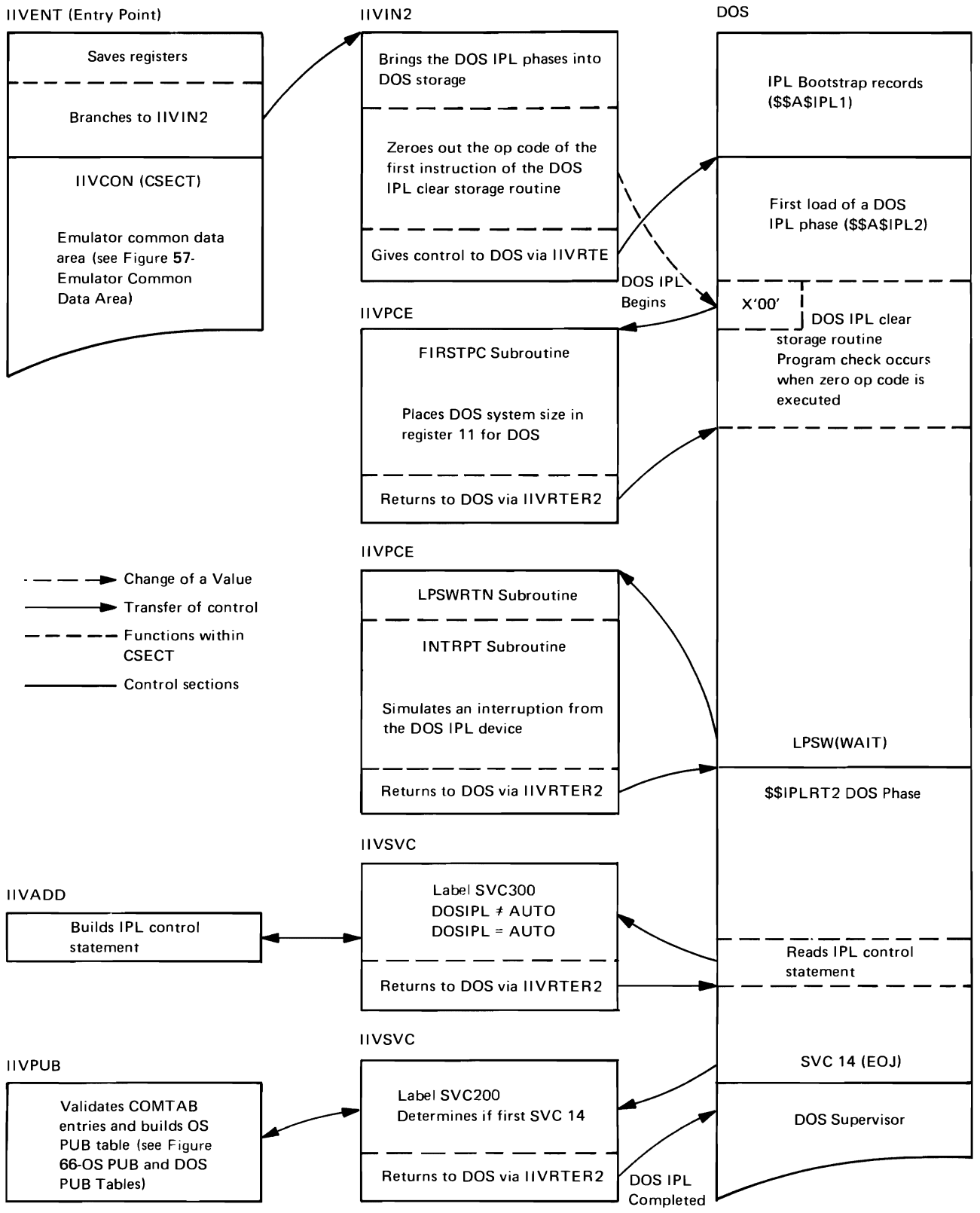


Figure 7. IPL of DOS Supervisor During Initialization

Verifying Parameters

The user parameter area is inspected to ensure that all required parameters have been specified. The values of the parameters are saved for use during the initialization operation. If parameter values are missing or invalid, the Emulator issues a WTOR to obtain them from the system operator. (See "Dependence on OS" in the section "Diagnostic Aids" for an explanation of the WTOR macro.)

Establishing the DOS Storage Area

The value specified by the DOSSYS parameter plus a 4K-byte boundary alignment factor is used in an OS GETMAIN macro to obtain the storage area into which DOS will later be loaded. The starting address is rounded up to the nearest 4K-byte boundary and the entire area is initialized to binary zeros. A FREEMAIN is then issued to release the extra 4K bytes at the beginning and end of the area. Subsequent GETMAINs obtain parts of this area for Emulator and OS I/O control blocks.

Initializing Emulator Tables

The following Emulator tables are built during the initialization operation.

Communications Table (COMTAB)

The communications table provides pertinent information about OS and DOS device correspondences. (See the section, "Data Areas" for the COMTAB format.)

One COMTAB entry is created for the DOSLOG system log (DOSLOG) and one for each device defined by a DD statement that contains a ddname beginning with the characters SYSE. After the table has been created, the entries are sorted in ascending order by their DOS channel and unit (cuu) numbers. The starting storage address of COMTAB is placed in the Emulator constants area (IIVCON).

COMTAB Extension

If a JFCB indicates a shared volume (contains a nontemporary dsname), a COMTAB extension entry is temporarily built and chained to the COMTAB entry. This temporary COMTAB extension consists of the ddname and dsname. The open flag (bit 7 in CTFLAG) and the shared volume flag (bit 1 in CTFLAG3) in COMTAB are set to indicate the presence of a temporary COMTAB extension.

The ddnames in the TIOT that do not begin with the characters SYS (or JOBLIB or STEPLIB) are counted; the resulting number is the number of data sets that reside on shared direct-access storage devices. The count is used to compute the storage needed for the COMTAB extension.

ISK/SSK Table

This table is used to record the keys that would be set by the SSK instruction if the DOS problem program were run under stand-alone DOS. When an ISK instruction is issued, the Emulator sets the storage key in conformance with the way in which it was set by the previous SSK instruction; the Emulator obtains the information from this table. This method of simulating storage protection does not provide true storage protection, which is provided by hardware under normal circumstances. It does, however, allow the DOS supervisor to check the location of control blocks to assure that they are in the proper task's area.

ECB Pointer Table

There is one event control block (ECB) for each DOS device. The ECB is located in a COMTAB entry. The Operating System uses this 4-byte control block to indicate the completion of an event (such as input, output, or timing) to the Emulator.

This table is a list of addresses of each ECB in the Emulator: one entry for a WTOR, one for the operator prompt, one for the timer, and one for each ECB contained in COMTAB.

Post ECB List

This list contains a 2-byte entry for each completed interruption not yet simulated to DOS and, therefore, outstanding to DOS. Each entry consists of the byte offset to the associated COMTAB entry and the completion code from the ECB. The list is used to ensure that I/O interruptions are simulated to DOS on a first-in, first-out basis.

OS Physical Unit Block (PUB) Table

The OS PUB table provides a one-to-one correspondence between a DOS device (whether dedicated or staged) and its associated OS device. Each 1-byte entry contains a byte offset to the corresponding COMTAB entry. An entry that contains X'FF' indicates the absence of an OS assignment.

Data Control Block (DCB)

One DCB is associated with each dedicated resource, except the DOS system log (DOSLOG) device, and one DCB is associated with every data set on a shared volume.

When DOS issues its first I/O request for a dedicated resource, IIVOPN obtains storage area for the DCB, initializes the DCB, and opens the DCB. The storage address for the dedicated resource DCB is placed in the corresponding COMTAB entry.

When DOS issues an OPEN for a shared data set, IIVDVS obtains storage area for the DCB, initializes the DCB, and opens the DCB. The storage address for the shared data set DCB is placed in the corresponding COMTAB extension entry.

Input/Output Block (IOB)

Each COMTAB entry, including the COMTAB entry for DOSLOG, contains an IOB. The IOB is initialized when the DCB is created (see "Data Control Block (DCB)"). The format and contents of the IOB varies, depending on whether the device is to be staged, is the DOSLOG device, or is neither.

DOS IPL

After the first part of initialization has been completed, the DOS IPL phases must be read into the DOS main-storage area. The Emulator builds channel programs and issues EXCPs to read the DOS bootstrap records into storage. The Emulator adjusts the CCW data addresses in the bootstrap records and again issues an EXCP. When the DOS bootstrap channel program is successfully completed, the DOS IPL phase $$A$IPL2 has been read into the DOS area of the Emulator region.

One of the functions of $$A$IPL2 is to clear main storage by moving zeros throughout main storage until a program interruption (addressing exception) occurs. This function cannot be performed in an emulated environment since it would destroy Emulator tables and control blocks. To bypass this function, the operation code of a move character (MVC) instruction in the clear storage loop of $$A$IPL2 is set to 0. When the DOS phase ($$A$IPL2) executes, the zeroed operation code causes a program interruption (operation exception); then the Emulator simulates the expected program interruption (addressing exception) to DOS. Control then passes to the DOS IPL routine.

Passing Control to DOS

Control is passed to DOS when the Emulator issues the execute local instruction. Conceptually, this instruction is much like an LPSW. (See the Appendix for a complete description of the execute local instruction.)

Before an execute local instruction can be issued for the first time, however, a PSW must be built in the first doubleword of the local execution list which effectively becomes the current PSW. Only the 3 low-order halfwords of the PSW are used by the execute local instruction. The instruction address portion of the local execution PSW contains the local (unadjusted) DOS instruction address. This address plus the adjustment factor (origin address field in the local execution list) points to an area in storage where the next DOS instruction to be executed resides.

When the execute local instruction is executed, the following sequence of events occurs:

- The 3 low-order halfwords of the current PSW are replaced by the corresponding fields of the local execution PSW.
- The contents of registers 14 and 15 are replaced by the third and fourth words of the local execution list.
- The CPU enters local execution mode.
- The next instruction to be executed is fetched by the CPU from the storage location specified by the DOS instruction address plus the adjustment factor.

DOS then retains control until a hardware interruption occurs, at which time the CPU leaves local execution mode.

INTERRUPTION ACTION WHEN CPU IS IN LOCAL EXECUTION MODE

The System/370 interruption mechanism, which stores the current PSW as an old PSW and fetches a new PSW, functions in a special fashion when the CPU is in local execution mode. To follow the machine interruption logic used by System/370, refer to Figure 8.

In general, the current PSW is stored in the first doubleword of the local execution list (local execution PSW) and the instruction address portion of the current PSW is replaced by one of the three possible interruption addresses located in the local execution list (see the "Data Areas" section for the format of the local execution list). If the cause of the interruption was asynchronous (I/O, external, or machine check), the current PSW is then stored as an old (OS) PSW and a new (OS) PSW is fetched. If the cause of the interruption was synchronous (SVC or program), the current PSW with the new instruction address remains current and the normal interruption mechanism of the CPU, and therefore the Operating System, is bypassed.

Synchronous Interruptions

Supervisor Call (SVC) Interruptions

The SVC instruction address field (bytes 28-31) of the local execution list contains the address of module IIVSVC. This module automatically gains control if an SVC interruption occurs while the CPU is in local execution mode.

The main function of IIVSVC is to simulate the interruption action normally performed by hardware in a stand-alone system. The local execution (DOS current) PSW is moved from the local execution list into the DOS SVC old PSW, and the DOS SVC new PSW is moved into the local execution PSW.

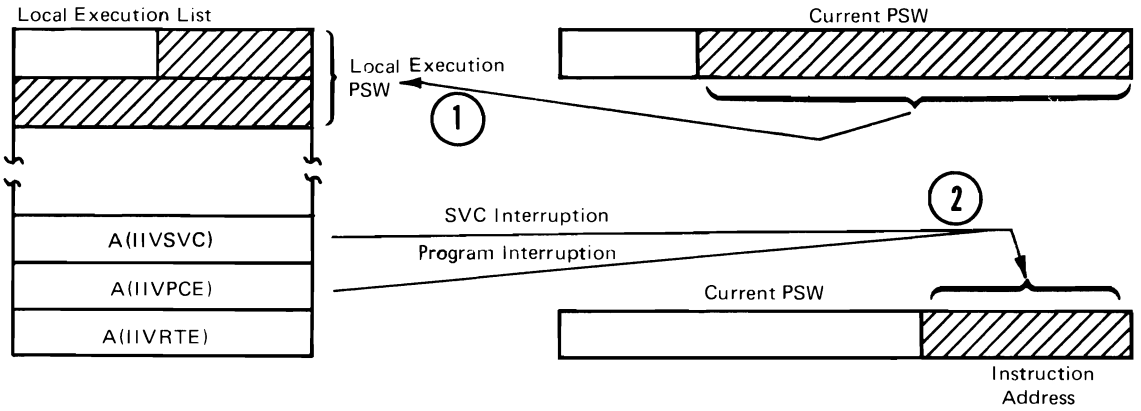
When the execute local instruction is eventually issued, the local execution PSW becomes the current PSW and the interruption action will have been simulated. Refer to "Passing Control to DOS" and the Appendix for further details.

An additional function of IIVSVC is to check the SVC number for the first SVC 4 and SVC 14. The first SVC 4 (DOS LOAD) signals that the DOS supervisor is in storage and ready to begin processing IPL input. The first SVC 14 (DOS EOJ) is assumed to signal the end of DOS IPL and initialization. If an OS indexed sequential or an OS or DOS sequential DASD or direct-access data set is being accessed, control is passed to module IIVGR2 to monitor the DOS-initiated supervisor call interruption.

Program Interruptions

The program interruption address field (bytes 32-35) of the local execution list contains the address of module IIVPCE. This module automatically gains control if a program interruption occurs while the CPU is in local execution mode. The different ways in which program interruptions are handled are divided into the following categories:

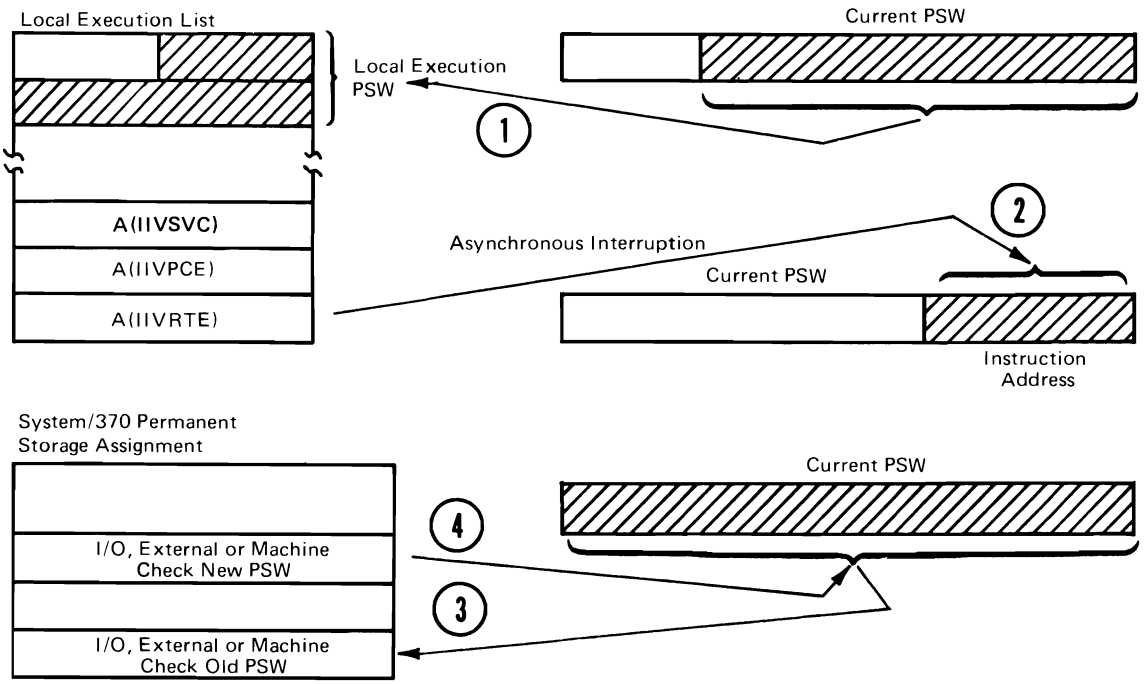
- First program interruption
- Store clock program interruption
- IPL interruption
- Normal program interruption
- Privileged operation



Synchronous (program or SVC) interruptions when the CPU is in local execution mode:

(1) The three low-order halfwords of the current PSW are stored in the local execution list.

(2) Then, depending on the type of interruption, the appropriate address from the local execution list is placed in the instruction address portion of the current PSW.



Asynchronous (input/output, external, or machine check) interruptions when the CPU is in local execution mode:

(1) The three low-order halfwords of the current PSW are stored in the local execution list.

(2) The instruction address portion of the current PSW is replaced by the asynchronous interruption address in the local execution list.

(3) The current PSW is stored in the permanent storage location of the I/O (or external or machine check, depending on the type of interruption) old PSW.

(4) The appropriate new PSW is moved from permanent storage to the current PSW.

Figure 8. System/370 Machine Interruption Logic

First Program Interruption

The first program interruption the Emulator expects to receive from DOS is an operation exception occurring in the Emulator-modified clear storage loop in DOS IPL phase $$A¶IPL2.

Store Clock Program Interruption

The System/370 store clock instruction is executed. The OS clock value is then adjusted by a value which was computed by the set clock routine in module IIVPCE.

IPL Interruption

The DOSIPL parameter passed to the Emulator in the PARM field of the EXEC statement specifies the DOS unit that will contain the DOS IPL statements. An I/O interruption from the device specified in the DOSIPL parameter is simulated to DOS when the Emulator encounters the first PSW with channel interruptions enabled and the wait bit set to 1.

Normal Program Interruption

Except for the first program interruption, IPL interruptions, and DOS supervisor-initiated privileged operations, the Emulator simulates the action of the hardware in a DOS stand-alone system by moving the local execution PSW to the DOS program old PSW and the DOS program new PSW to the local execution PSW. When the execute local instruction is eventually issued, the interruption action will have been simulated.

Privileged Operation

If the interruption code portion of the local execution PSW indicates that the program interruption was caused by a privileged operation exception, the local execution PSW is further examined to determine whether the problem program bit (bit 15) had been set by the DOS supervisor. If this bit is set to 1 (problem program state), the program interruption is passed on to DOS. (See "Normal Program Interruption" for details on how this type of interruption is handled.) If this bit is set to 0 (supervisor state), the instruction that caused the privileged operation exception determines the action taken by the Emulator.

Note: The first halfword of the local execution PSW is ignored by the execute local instruction. The contents of the fields within this area are maintained only to preserve the continuity of functions either not supported or simulated by the Emulator (such as status switching, storage protection, etc.)

The following privileged operations are simulated by the Emulator:

- ISK - insert storage key
- SSK - set storage key
- SSM - set system mask
- LPSW - load PSW
- LCTL - load control
- STCTL- store control
- STIDP- store CPU ID
- STIDC- store channel ID*
- SCK - set clock*
- TCH - test channel
- TIO - test I/O
- HIO - halt I/O
- SIO - start I/O

---

*The functions of these instructions are ignored, the condition code is set to zero and control is returned to DOS.

Privileged instructions RDD, WRD, and DIAGNOSE, and the DOS Compatibility Feature instructions (execute local and adjust CCW string) are treated as program interruptions. (See "Normal Program Interruption" for procedure.) The System/370 privileged instructions (LCTL, STCTL, STIDP, STIDC, SCK) are simulated only when a DOS Release 27 system is being emulated.

Insert storage key (ISK): The contents of the first operand register of the ISK instruction are replaced by the entry in the ISK/SSK table associated with the 2K block of storage specified in the second operand.

Set storage key (SSK): The key specified in the first operand register of the SSK instruction replaces the entry in the ISK/SSK table associated with the 2K block of storage specified in the second operand address.

Set system mask (SSM): The mask located at the storage address specified in the operand of the SSM instruction is placed in the first byte of the local execution PSW.

Load program status word (LPSW): The program status word located at the address specified in the LPSW instruction is moved into the local execution PSW. An additional function of the LPSW simulation subroutine is to intercept the first LPSW that specifies a PSW with interruptions enabled and the wait bit set to 1 during IPL. When this condition is met, an I/O interruption is simulated to DOS. (See "IPL Interruption" for procedure.)

Load control (LCTL): Control register information (starting at the DOS main-storage address specified in the LCTL instruction and continuing through as many storage words as the number of control registers specified) is moved into the control register field labeled CTLREGS in EMUCONS.

Store control (STCTL): Control register information is moved from the control register field labeled CTLREGS in EMUCONS to the DOS main-storage address specified by the STCTL instruction and continuing through as many storage words as the number of control registers specified.

Store CPU ID (STIDP): Eight bytes of CPU identification information are moved from a field (labeled CPUID) in EMUCONS to the DOS main-storage address specified in the STIDP instruction. The CPUID field in EMUCONS is initialized by the Emulator start I/O appendage. (See "I/O Appendages" in this section.)

Store channel ID (STIDC): The condition code field of the local execution PSW is set to zero (channel ID correctly stored). Control is returned to DOS.

Set clock (SCK): The resulting values of the difference between the OS time-of-day clock and the DOS time-of-day clock is computed and saved in EMUCONS. The DOS GETIME routine (SVC 34) is searched for the store clock instruction (SCK) and the opcode is set to zero. The condition code field of the local execution PSW is set to zero (clock value set). Control is returned to DOS.

Test channel (TCH): This instruction is simulated when the TCH simulation subroutine tests the device-busy flag in COMTAB. All COMTAB entries associated with DOS devices on the specified DOS channel are tested. If any of the devices are found to be busy, the condition code portion of the local execution PSW is set to 2 (channel busy). Otherwise, the condition code is set to 0 (channel available).

Test I/O (TIO): This instruction is simulated when the TIO simulation subroutine tests various fields in the COMTAB entry that corresponds to the device addressed in the TIO instruction.

The following TIO condition codes may be set in the local execution PSW and passed back to DOS to indicate which of the following conditions exists:

Condition Code	Meaning
0	Device not busy
1	CSW stored (CSW is moved from associated IOB into CSW area of DOS storage when I/O is complete)
2	Device busy (I/O not complete)
3	Not operational (no COMTAB entry found or CTFLAG, nonoperational flag, is set to 1)

Halt I/O (HIO): The OS macro IOHALT is issued to perform an HIO instruction. A condition code of 1 (CSW stored) with a CSW status of 0, is posted to the DOS supervisor, indicating the operation has been halted.

Start I/O (SIO): Because the Emulator executes as an OS problem program, a DOS request for input or output can be satisfied only with the assistance of OS data management. How the I/O request is actually handled is determined by the following criteria:

- If the I/O request is made to the DOS console device (DOSLOG), the request is interpreted and reissued in the form of an OS WTO or WTOR (see 1 below).
- If the I/O request is made to a unit-record device that is to be spooled from or to a temporary data set, the request is interpreted and reissued in the form of an OS GET or PUT (see 2 below).
- For all other I/O requests, the channel command word data addresses are adjusted and the request is reissued by means of OS channel programming (EXCP) techniques (see 3 below).

1. I/O operations - DOS system console. The only way a problem program, executing under OS, can communicate with the operator is to issue a WTO or a WTOR macro instruction by means of module IIVLOG. The parameters supplied to these macros will provide the message length, the storage address of the message to be issued and, in the case of a WTOR, the reply length and storage address.

   IIVLOG gains control at DOS SIO from module IIVPCE. DOS CCWs are located and checked for validity. If the CCW command code is for a write, the request is reissued as a WTO. If the command code is for a read or for a read chained to a write, a WTOR is issued. The WTO or WTOR length parameters are modified, as required by the Emulator. Data is moved to or from the DOS storage area.

2. I/O operations - staged unit-record devices. In a multiprogramming system, when the interpreter task of the Operating System encounters a DD * or DD DATA statement, a temporary data set is created on a direct-access device. Input data that follows the DD statement is spooled (temporarily written) to that data set until the /* delimiter is encountered. The data for any subsequent input requests is retrieved from the temporary data set.

   A DD statement specifying a SYSOUT class (for example, DD SYSOUT=A) causes OS to create a temporary data set on a DASD. All subsequent output data for that class is spooled to that temporary data set. At job termination, the data in the temporary data set is routed by the OS system writer to the unit-record device specified in the JCL.

Staged I/O receives control at DOS SIO. DOS CCWs are located, checked for validity, and interpreted. A QSAM GET or PUT macro is issued and the data is moved to or from the DOS area. Machine control characters (DOS CCW command codes) are placed in the output records to cause printer skipping and spacing and punch stacker selection. Unit-record device operations are simulated to DOS. A CSW and sense byte are maintained for each staged device. These indicators simulate unusual device conditions such as unit check or incorrect length as well as the usual channel end/device end condition.

If errors are discovered by staged I/O modules in the DOS CAW or in the CCW addressed by the CAW, the CSW stored (CC=01) condition is set in the local execution (current) PSW, the CSW is moved into the DOS low storage area, and control is returned to DOS. Under all other conditions, the DOS SIO condition code is set to 0 (successful). DOS does not receive the simulated interruption until it enables channel interruptions through an LPSW. The CSW built during the previously staged SIO simulation is now moved into the DOS low storage area.

Printer overflow is handled by module IIVPOV, which simulates the printer carriage tape operation to DOS. This function provides channel 9 and 12 indications for DOS in the same manner as the hardware does. (See "Printer Overflow Simulation" for more details.)

Staged I/O uses the following modules (see Figure 9):

IIVSTG - contains the main logic for staging both input and output.

IIVAWV - verifies the validity of the DOS CAW and locates the first CCW for the I/O operation.

IIVCWV - verifies the validity of the CCW being processed and follows the channel program logic by replacing the addresses of the TIC (transfer in channel) CCWs with the CCW addresses to which the TICs point.

IIVPOV - supports printer overflow. This module may be omitted from the Emulator linkage editor at the system programmer's discretion by coding PARM.LKED=LET.

IIVFCB - is the FCB image that resides in SYS1.IMAGELIB. Module IIVSTG converts this image to an Emulator format forms-control image (FCB2EM01).

IIVPR1 - contains the command code translate table for 1403 and 1443 Printers.

IIVRP1 - contains the command code translate table for a 2540 Card Read Punch.

IIVRD1 - contains valid reader commands in the command code translate table for a 1442 Card Read Punch.

IIVPU1 - contains valid punch commands in the command code translate table for a 1442 Card Read Punch.

IIVRD2 - has the same function as IIVRD1 for a 2520 Card Read Punch.

IIVPU2 - has the same function as IIVPU1 for a 2520 Card Read Punch.

IIVRD3 - contains the command code translate table for a 2501 Card Reader.

IIVPU3 - contains a command code translate table for a 3525 punch.

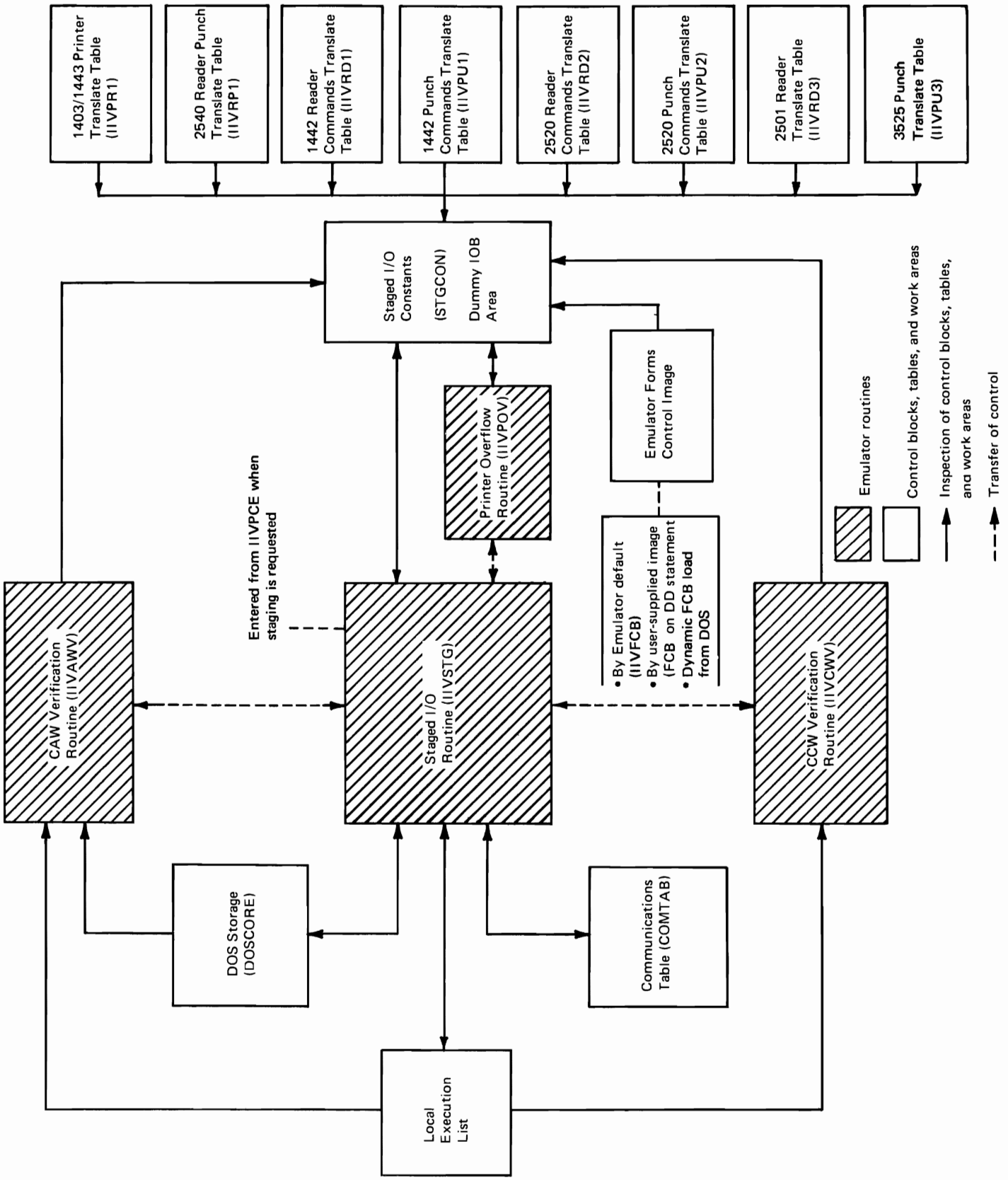


Figure 9. Staged I/O Control Program Flow

3. I/O operations - all other I/O requests. In general, by issuing the EXCP macro instruction, the Emulator requests the execution of the channel program specified in the IOB. To initiate execution of the channel program, the Operating System obtains its address from the IOB, places this address in the CAW, and issues an SIO instruction. (For further information concerning channel program execution, see Data Management for System Programmers for OS or OS/VS.)

   The channel program is supplied by DOS and is composed of CCWs on doubleword boundaries. Each CCW specifies a command to be executed and, for commands that initiate data transfer, the area to or from which the data is to be transferred.

   Before issuing the EXCP macro instruction, the Emulator must adjust the data addresses in each CCW through the use of adjust CCW string, a DOS Compatibility Feature instruction. (See the Appendix for a detailed description of this instruction.) When the input/output operation is complete, the CCW data addresses are readjusted to their original values.

   The DOS channel program is then executed with the following exceptions:

   - For direct-access devices, the DOS initial CCW sequence of SEEK-TIC or SEEK-SET FILE MASK-TIC is bypassed by the Emulator so that the OS function is not suppressed.

   - For magnetic-tape devices, the DOS initial SET MODE-TIC sequence, if present, is bypassed by the Emulator. If the set mode opcode is different than that contained in the OS DEB, a stand-alone set mode is issued so that the Emulator start I/O appendage can move the new set mode opcode to the DEB. The remainder of the DOS user's channel program is then executed.

I/O Appendages

A start I/O, end-of-extent, channel end, and abnormal end appendage (IGG019SA) and an abnormal end/channel end appendage (IGG019S1), both of which reside on SYS1.SVCLIB, are used by the Emulator.

Start I/O, End-of-Extent, Channel End, and Abnormal End Appendage

The Emulator issues an OPEN macro to open the VTOC of each dedicated direct-access volume to be used in an emulation run. OS OPEN builds a DEB in protected storage for each direct-access device, reflecting the upper and lower bounds of each VTOC data set. Any subsequent attempt to gain access to data outside of these boundaries results in an extent violation and termination of the I/O operation.

Since DOS problem programs must be able to refer to any location on a dedicated DOS volume, and not just the VTOC, the extent limits in each DEB must be changed to specify the extents of the cylinder being accessed each time an extent violation occurs. The Emulator's end-of-extent appendage, running with a protect key of 0, is able to enter protected storage and make this change before an SIO instruction is issued. This appendage is used only for DOS private volumes.

This does not hold true for data sets on shared volumes since the data set defined in the DD statement is opened and not the VTOC.

Prior to execution of the OS SIO and after the Emulator EXCP is issued, the start I/O appendage is entered to modify the DASD volume's file mask in the DEB. The file mask is modified in conformance with the DOS SFM command. If DOS does not issue an SFM command, the file mask is set to X'00'. The start I/O appendage is also entered to modify the tape set mode opcode in the DEB to conform to the mode requested by the DOS I/O supervisor.

Abnormal End/Channel End Appendage

The Emulator uses the abnormal end appendage to bypass the OS error recovery procedures for an end-of-cylinder condition on a direct-access device and a rewind-unload condition for tape or channel 9 is encountered for printer devices. When one of these conditions is detected, the abnormal end appendage resets the IOB exception flag to prevent the OS error recovery procedures from being entered. A normal return to the OS I/O supervisor is made and the I/O operation is posted complete.

The channel end appendage is entered for all tape and unit record devices at channel end. Upon receiving an incorrect length unit exception indication, the appendage turns off the IOB exception flag to prevent the OS I/O supervisor from loading error recovery routines. Otherwise, the I/O supervisor performs its normal processing. For chaining checks on tape, this appendage turns on the data chain bit in IOBFLAG1 to inform the tape error recovery procedures not to retry the channel program.

Asynchronous Interruptions

The asynchronous interruption address field (bytes 36-39) of the local execution list contains the storage address of module IIVRTE. This module gains control from OS after the processing of an asynchronous interruption that occurred while DOS was in control.

The main functions of this module are to process DOS-initiated asynchronous interruptions and to pass control to DOS by issuing the execute local instruction. (See "Passing Control to DOS" for details.)

Input/Output Interruptions

The ECBs that are pointed to by entries in the ECB pointer table are tested and a post ECB list entry is created for each ECB indicating I/O completion. When the system mask portion of the local execution PSW is set so that I/O interruptions are enabled and one or more post ECB list entries exist, a hardware interruption is simulated to DOS.

The CSW is moved from the IOB to DOS location 64. The local execution PSW is moved to the DOS I/O old PSW and the DOS channel and unit address is placed in the interruption code portion of the DOS I/O old PSW. The DOS I/O new PSW is then moved to the local execution PSW. If the interruption was initiated by a reply of ATTN to the Emulator prompt, the attention bit in the status portion of the DOS CSW is set to 1. When the execute local instruction is issued, the interruption is simulated.

Interval Timer Interruptions

The OS supervisor macro instruction STIMER is issued at 1-second intervals when the Emulator parameter DOSTIM=YES has been specified in the PARM field of the EXEC statement. This ensures the continuous maintenance of the DOS interval timer. Accuracy of the DOS interval timer will depend upon the priority of the Emulator job.

When the ECB associated with the expiration of a 1-second interval has been posted, DOS location 80 is decreased by 1 and the STIMER is reissued. When the DOS timer value becomes negative and the system mask portion of the local execution PSW is set so that external interruptions are enabled, a timer (external) interruption is simulated to DOS.

The local execution PSW is moved to the DOS external old PSW and the DOS external new PSW is moved to the local execution PSW. When the execute local instruction is issued, the interruption is simulated.

External Interruption Simulation

When an operator-initiated external interruption is required by a DOS problem program, the operator replies EXT to the Emulator prompt. If the system mask portion of the local execution PSW is set so that external interruptions are enabled, the local execution PSW is moved to the DOS external old PSW and the DOS external new PSW is moved to the local execution PSW. When the execute local instruction is issued, the interruption is simulated.

DIRECT-ACCESS VOLUME SHARING

With the removal of the Version 1 requirement for private volumes, DOS and OS data sets can now reside on the same DASD volume. This is optional for one or more volumes in an Emulator job step. DOS indexed sequential files, however, cannot be shared and must be accessed from dedicated volumes.

A data set located on a shared volume is defined through OS DD statements, and its allocation and maintenance in the VTOC is done through OS JCL by the OS direct-access device space management (DADSM) routines.

When DOS wants to allocate a file, as specified by DOS DLBL and EXTENT statements, the open routines maintain the VTOC of the DOS file to its corresponding volume by using the information in the DTF (type of file, extent sequence number) and the DLBL/EXTENT card image (location of the file on the volume, total number of extents, type of extents).

The volume-sharing simulation routines, IIVGR2, IIVDVS, and IIVVIO, update the DOS DTF and DLBL/EXTENT image to reflect the status of the file as allocated by OS. They also simulate a DOS access to the VTOC, either by issuing an OS OBTAIN macro when DOS issues a read to the volume label or an identifier (format 1) or extension (format 3) DSCB or by bypassing the I/O operation when DOS issues a write to the VTOC. In addition, module IIVDVS changes the flow of DOS B-transient phases so that some will be bypassed.

This section explains in more detail the flow of control between DOS B-transient phases and the Emulator, and outlines the relationship between the Emulator routines and the Emulator control blocks involved. Figure 10 gives an example of open/close processing for a sequential disk output file.

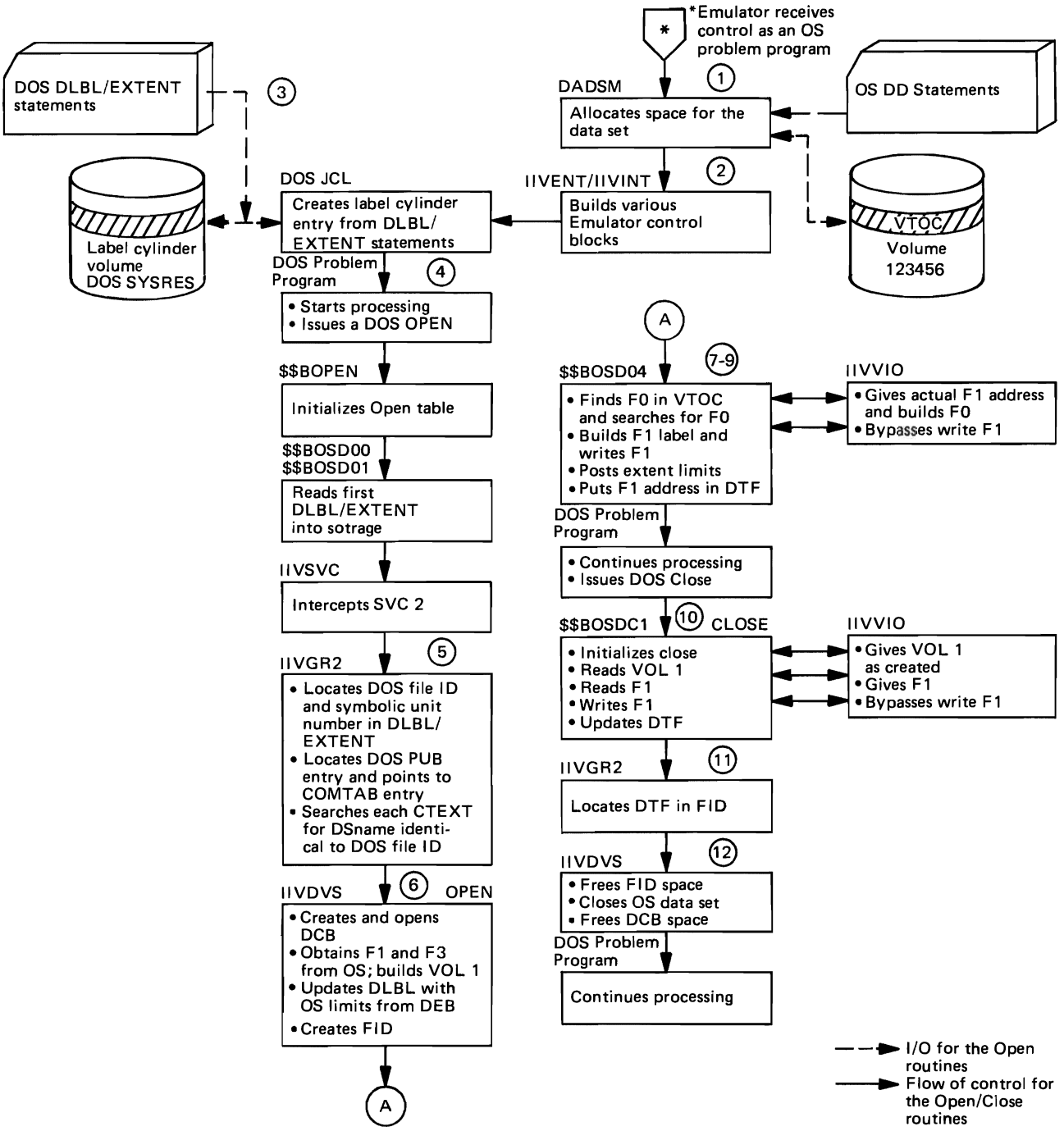


Figure 10. Example of Open/Close Processing for a Sequential Disk Output File

Note[1]: In order to understand the following, familiarity with DOS open/close processing is essential. See DOS Logical IOCS Volume 1: Introduction, GY24-5020, and Volume 3: Sequential and Direct Access DASD Files, GY24-5088, for details.

Note[2]: The encircled numbers correspond to the encircled numbers in the figure. Numbers (1) through (12) apply to OS sequential DASD and direct-access shared data sets and to DOS sequential DASD and direct-access shared files. Indexed sequential shared data sets are discussed in "OS Indexed Sequential Data Set Sharing" later in this section.

(1) The Emulator is called with the following OS control statements:

```
//EMU          JOB  ACT#,name,MSGLEVEL=1
//             EXEC PGM=IIVEMU,PARM=('XXXXXX')
                  .
                  .
                  .
//SYSEM191     DD   UNIT=2314,VOL=SER=123456,
//                  DSN=DOSFILA,DISP=(NEW,KEEP),
//                  SPACE=(CYL,(1))
```

OS JCL reads the DD statements and builds the corresponding JFCBs.

The OS initiator uses the JFCBs to perform space management on the requested volume as follows:

(a) Determine whether volume 123456 is mounted and, if not, look for an available device and issue a MOUNT message to the operator

(b) Allocate auxiliary storage space on the volume by searching the VTOC for an identifier (format 1) DSCB to be created and a free cylinder extent to be allocated to file DOSFILA

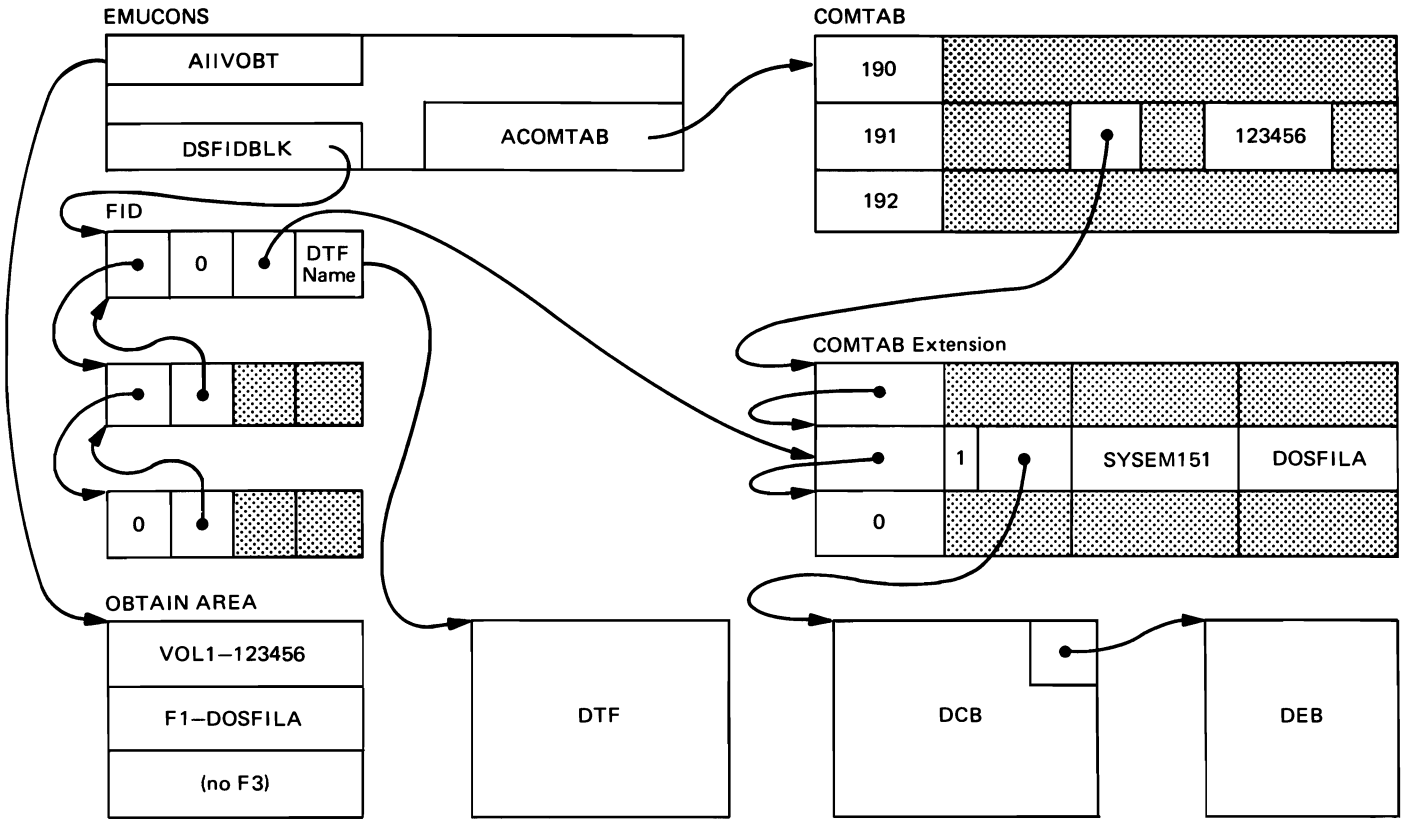
The VTOC of 123456 volume will then contain an identifier (format 1) DSCB describing DOSFILA file.

(2) The Emulator initialization routines read the JFCBs and build the Emulator control blocks. One COMTAB extension containing SYSEM191 in the CTDDNAME field and DOSFILA in the CTDSNAME field is created and chained to the COMTAB entry for DOSCUU=191. Refer to Figure 11 for the relationship of control blocks affected by IIVDVS.

(3) The DOS problem program is then called with the following control statements:

```
// JOB    DOS,...
   .
   .
// DLBL   DTF1,'DOSFILA',99/365
// EXTENT SYS001,,,10,10
// ASSGN  SYS001,X'191'
   .
   .
// EXEC   PROGRAM
/*
```

DOS JCL reads the DLBL and EXTENT statements and creates a DLBL/EXTENT record in the label cylinder of the DOS system-residence pack. Logical unit SYS001 is then assigned to physical unit 191.



Legend

ACOMTAB–Pointer to COMTAB

AIIVOBT–Pointer to Obtain Area

COMTAB–Communications Table

DCB–Data Control Block

DEB–Data Extent Block

DOSFILA–Data Set Name

DSFIDBLK–Pointer to FID

DTF–Define the file

EMUCONS–Emulator Constants Area DSECT

FID–File ID

F1–Format 1 (DSCB)

F3–Format 3 DSCB

SYSEM191–DD(data definition) Name

VOL1–Volume 1

123456–Volume Serial Number

190, 191, 192–DOS Device Addresses

Figure 11. Data Areas Affected by Open Processing (IIVDVS)

(4) The DOS problem program issues an OPEN macro for DTF1. After the DOS B-transient phases $$BOPEN and $$BOSD00 have been executed, phase $$BOSD01 locates, in the label cylinder, the DLBL/EXTENT record associated with DTF1, and builds the DLBL/EXTENT image in the open table, which is located in the DOS B-transient area. $$BOSDO1 releases control by issuing an SVC 2 instruction to load $$BOSDO1.

The Emulator supervisor call routine (IIVSVC) traps this SVC 2 and passes control to module IIVGR2 to monitor the call.

(5) IIVGR2 then locates the DOS file ID and logical unit number (here DOSFILA and SYS001) in the DLBL/EXTENT image, locates the DOS PUB entry associated with the logical unit (here 191), locates the corresponding COMTAB entry, and searches each associated COMTAB extension for a DSname identical to

the DOS file ID. When a match indicating a volume-shared file is found, module IIVGR2 sets the COMTAB extension address, open code, DTF address, and DOS LTK in the PARMLST field in EMUCONS. Then control is passed to module IIVDVS to perform OS open processing for the file described in the COMTAB extension entry.

(6) Module IIVDVS creates a FID containing a pointer to the COMTAB extension entry in the FIDCTXTN field. The name of the DTF is placed in the FIDTFNME field of the FID. IIVDVS then attaches this FID to the FID chain pointed to by DSFIDBLK in EMUCONS. The use count located in the COMTAB extension is increased by 1.

If the value of the use count equals 1, a DCB is created and opened, and its address is moved into the COMTAB extension. At this point, module IIVDVS issues an OS OBTAIN macro to get the identifier (format 1) DSCB address and contents from the VTOC and builds a volume 1 image, format 1 image, and format 3 image (if a format 3 DSCB was created) in the obtain work area pointed to by EMUCONS. The actual identifier (format 1) DSCB address is moved into the open table (see Section V for the format of the open table). Other indicators ('format 4 indicator' and 'return message indicator') are set in the open table to reflect the fact that DOS phases $$BOSDO1, $$BOSDO2, $$BOSDO3, and $$BOSDO8 have been bypassed.

(7) IIVDVS last updates the DLBL/EXTENT image in the B-transient area to reflect the type of extents and location of the extents as given by the OS DEB. IIVDVS returns control to IIVGR2 after setting $$BOSDO4 in NXTBTR in EMUCONS. IIVGR2 then updates the B-transient phase name with $$BOSDO4 and control is given to DOS B-transient phase $$BOSDO4.

Note that DOS open phases $$BOSDO1, $$BOSDO2, $$BOSDO3, and $$BOSDO8, which deal with space allocation in the VTOC, are not executed.

(8) $$BOSD04 searches the VTOC to find a free DSCB by issuing the following channel program:

- 'Seek, search on ID equal, TIC' to locate the beginning of the VTOC (address found in the open table)
- 'Read count, search on key equal to 0, TIC' to find the first free VTOC record (format 0) DSCB

Module IIVPCE intercepts the SIO and determines that the seek address is not contained in any of the DEBs attached to the COMTAB extension entries for this DOS cuu. IIVPCE then passes control to IIVVIO, which identifies the DOS channel program as a search DSCB, format 0 type, and simulates it by passing to DOS the count field of the actual identifier (format 1) DSCB as found in the obtain work area.

The same processing is done for the write identifier (format 1) DSCB channel program issued by $$BOSD04, which is recognized and bypassed by module IIVVIO.

(9) $$BOSDO4 then updates the DTF (with extent limits) or initializes it (with record number and identifier (format 1) DSCB address).

Note that the extent limits are obtained from the DLBL/EXTENT image in the DOS B-transient area. They are, therefore, the actual limits as determined by the OS space allocation routine because they were moved to the DLBL/EXTENT image by IIVDVS (see step (7)). Open processing is now complete. The DOS problem program will eventually issue a CLOSE macro for DTF1 and control will then be given to DOS B-transient phase $$BOSDC1.

(10) $$BOSDC1 updates the format 1 DSCB to indicate the file has been successfully created. The three channel programs issued (read volume 1, read format 1 DSCB, rewrite format 1 DSCB are intercepted by IIVPCE, and recognized and simulated by IIVVIO as explained in step (8).

$$BOSDC1 releases control by issuing an SVC 11 instruction, which is trapped by IIVSVC and recognized by IIVGR2 as the end of a DOS close operation.

(11) Module IIVGR2 identifies the file as a volume-shared file by searching the FID chain addressed by DSFIDBLK for a matching DTF name and LTK. A match indicates a volume-shared file, and control is passed to module IIVDVS to close the OS file.

(12) Module IIVDVS deletes the FID associated with the file from the FID chain. The use count in the corresponding COMTAB extension entry is decreased by 1. If the new use count is greater than 0, more than one DTF is accessing the same file and at least one DTF is still open.

A use count of 0 indicates that no more DTFs for the file remain open. An OS CLOSE macro is then issued, and the storage space occupied by the DCB is released.

Control is eventually returned to modules IIVGR2 and IIVSVC, and the latter issues an SVC 11 instruction.

End-of-Extent Processing

Upon reaching the end of an extent for an output file, the DOS problem program calls $$BOPEN to open the next extent.

The processing is like open processing except for two steps:

1. In updating the DLBL/EXTENT image (step (7)), module IIVDVS obtains the current extent sequence number from the DTF and searches the associated DEB for this extent. If this extent is not found, an OS EOV SVC is issued to obtain secondary allocation, if any is specified in the DD statement. Module IIVDVS then moves the limits of the current extent from the DEB to the DLBL/EXTENT image in the DOS B-transient area. If at the time the data set was created no secondary allocation was specified in the DD statement or if more than 16 extents have already been allocated, a message is issued and the DOS step is canceled.

2. In the next DOS phase to be executed (step (8)), IIVDVS sets $$BOSDO5 in the NXTBTR field of EMUCONS. This phase, when executed, issues a channel program to read the identifier (format 1) DSCB from the VTOC, and then tries to rewrite the DSCB with an additional extent. Both channel programs are intercepted by module IIVVIO and simulated.

Input Processing

Steps (1) through (6) are identical. IIVGR2 takes control when $$BOSDI1 is called, and IIVDVS moves $$BOSDI2 into the NXTBTR field in EMUCONS. $$BOSDI2 then attempts to read the identifier (format 1) DSCB. This attempt is intercepted by module IIVVIO, which gives back the actual OS identifier (format 1) DSCB.

Workfile Processing

The first call for $$BOSDW1 is similar to the open for an output file. The Emulator, in step (7), moves to the DLBL/EXTENT image the actual limits of the first extent as found in the DEB, replaces $$BOSDW1 and $$BOSDW2 in the

NXTBTR field of EMUCONS, and sets the indicators "extent open" and "SYSxxx in DTF" in the DTF.

A test is made to determine whether this extent is the last one allocated, and the bit indicating "last extent" is set on in the DLBL.

$$BOSDW2 updates the DTF to reflect the limits of the first extent (as set in the DLBL/EXTENT image) and calls $$BOSDW1 again. This second call for $$BOSDW1 is then trapped by module IIVGR2 and control is passed to module IIVDVS. If the "last extent" bit is on, control is returned to DOS, which terminates the open processing.

If the "last extent" bit is off, module IIVDVS moves the limits of the next extent from the DEB to the DLBL/EXTENT image. Note that for workfiles, module IIVDVS never issues an OS EOV SVC. Specification of the secondary space parameter in the DD statement will, therefore, not be useful.

End-of-extent processing for a workfile is similar to open processingg for an input file, because all the extents have been allocated when the file was opened and have already been created in the DSCB.

Module IIVVIO intercepts the "read format 1 DSCB" or "read format 3 DSCB" channel program issued by $$BOSDW3 and simulates it by returning the actual format 1 DSCB or format 3 DSCB as set in the obtain work area.

Direct-Access File Processing

Direct-access file processing is very similar to the open of a sequential disk input file or sequential disk output file. The main difference is that the DLBL/EXTENT image for a direct-access file contains all the extents for the output file.

For an output file, module IIVDVS (step (7)) obtains all the extents from the DEB and moves their limits to the DLBL/EXTENT image. Note that module IIVDVS never issues an OS EOV SVC for a direct-access file.

DOS Release 27 Processing

When emulating a DOS release 27 system, module IIVGR2 traps the SVC 2 for $$BOPEND and moves $$BOPEN in place of $$BOPEND in NXTBTR. Phase $$BOPEND, which initializes the system recorder file records by reading the VOL1 record for all assigned LUBs to disks, will then be bypassed.

DOS Release 27 Output Processing

Since $$BOSDO4 has been divided into two different open phases ($$BOSDO4 and $$BOSDO9) in DOS release 27, IIVDVS will bypass $$BOSDO4. IIVDVS moves $$BOSDO9 in place of NXTBTR and simulates $$BOSDO4 by setting DOS register 0 with the appropriate device type control factor and updating the volume sequence number in the DTF.

OS Indexed Sequential Data Set Sharing

Unmodified DOS programs running under the Emulator can gain access to and create OS-format data sets. However, the DOS user must comply with OS restrictions and requirements.

A volume-shared OS indexed sequential data set is defined through OS DD cards, and its allocation and maintenance in the VTOC of the corresponding volume is made at OS JCL time via OS DASD space management routines.

When DOS wants to access an OS indexed sequential data set, as defined by DOS DLBL/EXTENT statements, the Emulator gets the user to the proper data set by matching the DLBL file ID to the data set name in the DD statement.

The OS indexed sequential data set sharing simulation routine, IIVIS, updates the DOS DTF that reflects the status of the data set as allocated by OS. The open routine in IIVIS replaces the DOS ISFMS open B-transient phases.

The following section explains in more detail the flow of control between the Emulator and DOS, and outlines the relationship between the Emulator routines and the data areas affected by open processing (see Figure 12). Figure 13 is an example of processing OPEN and I/O macros for an OS indexed sequential data set.

Example of Processing

See numbers (1) through (5) in Figure 10 and steps (1) through (5) in the text under the heading "Direct-access Volume Sharing." These steps are identical to those taken when processing a shared OS indexed sequential data set, except

- $$BOPEN and $$BOPEN2 do the processing for indexed sequential data sets in the box showing $$BOSD00 and $$BOSD01 as processing phases,
- a match (COMTAB extension flag byte = 0) must indicate a shared OS indexed sequential data set, and
- control must be passed to module IIVIS to perform an OS open.

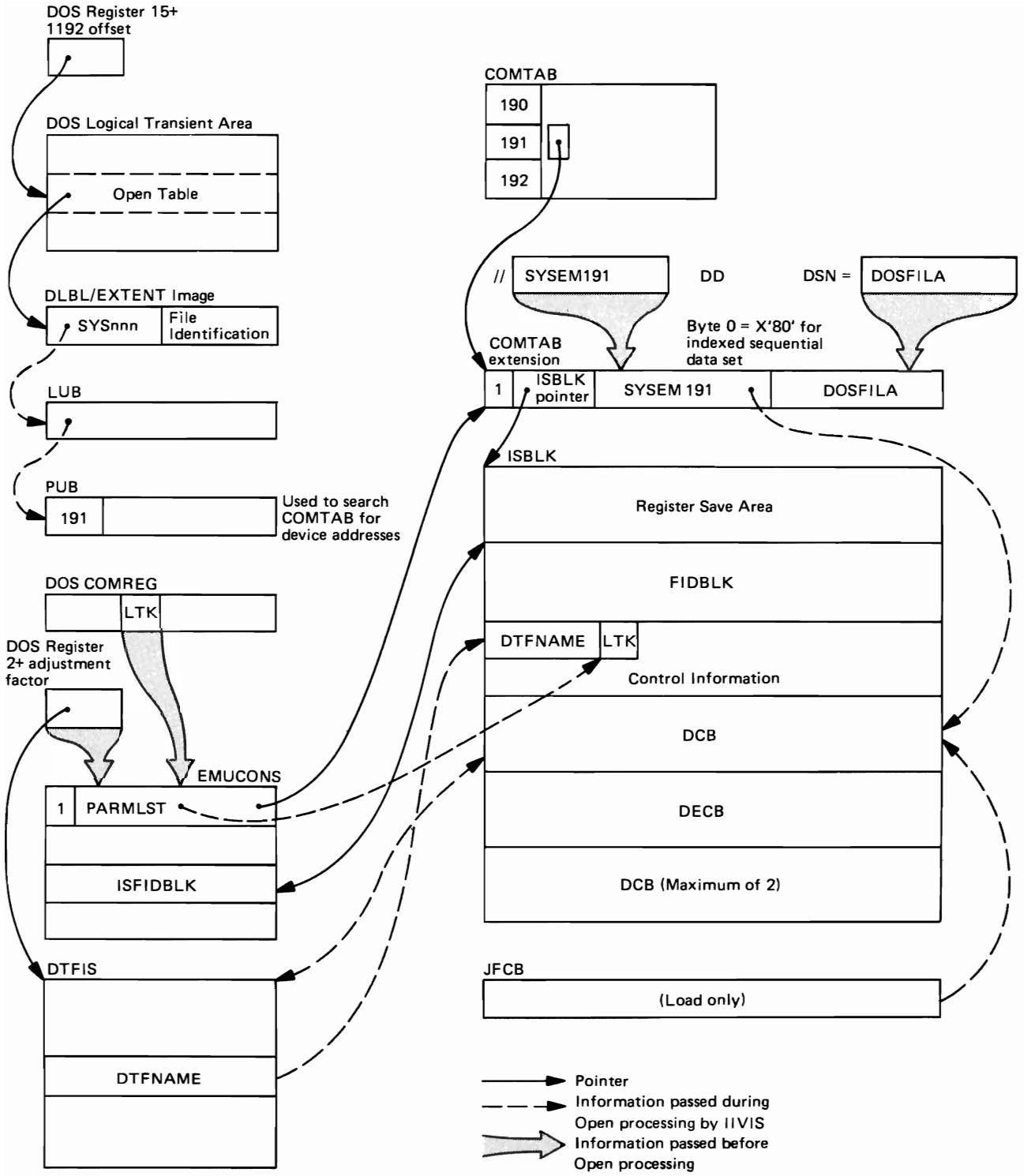


Figure 12. Data Areas Affected by Open Processing (IIVIS)

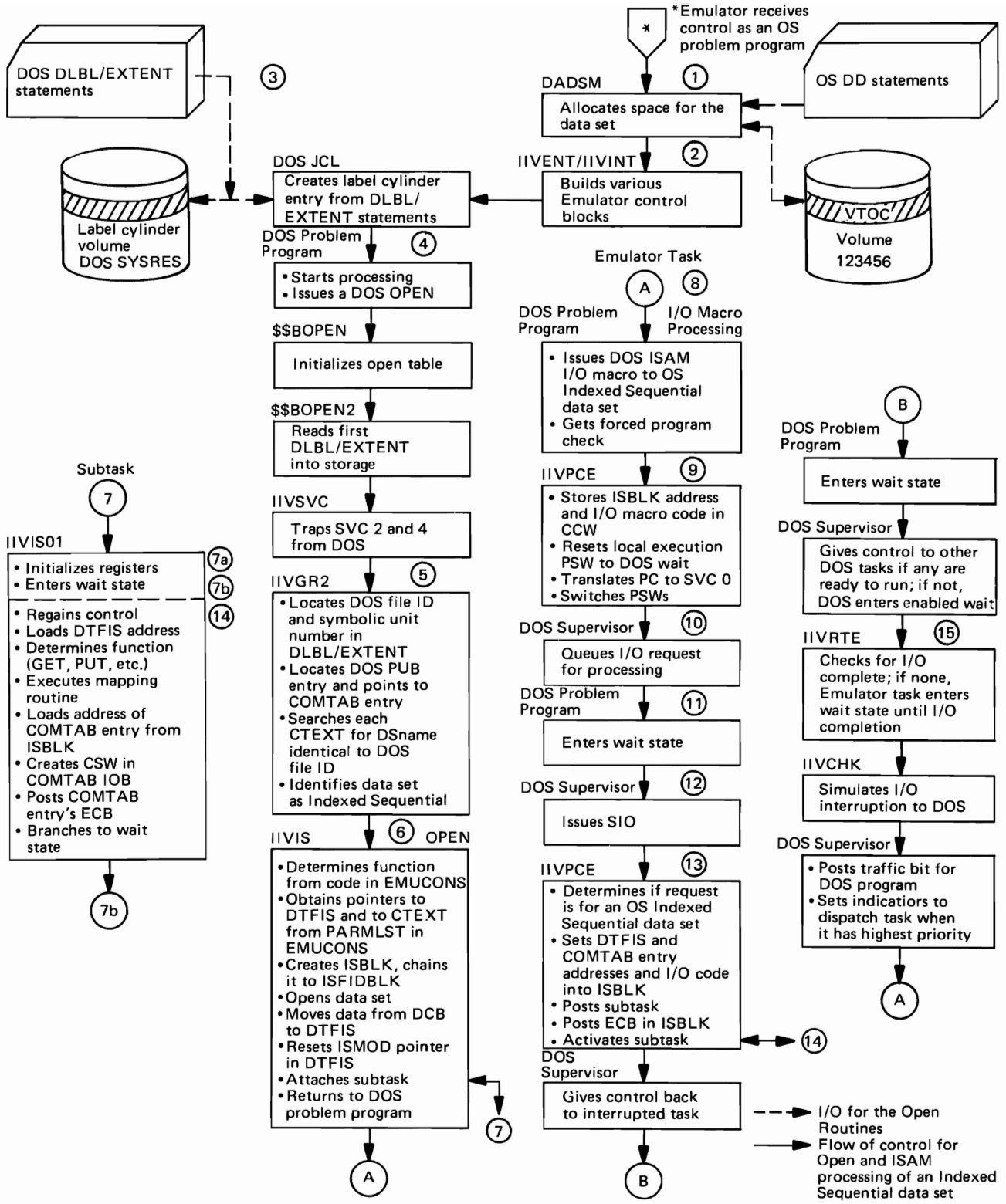


Figure 13. Example of Processing OPEN and I/O Macros for an OS Indexed Sequential Data Set

The following text is keyed to the encircled numbers in Figure 13.

(6) IIVIS determines the function (OPEN here, but it could also be CLOSE or SETL) to be performed by examining the code in EMUCONS. It also obtains the pointers to the DTFIS and to the COMTAB extension from EMUCONS. IIVIS creates a work area called the ISBLK, adds it to the ISFIDBLK chain, and opens the data set.

IIVIS then moves data from the DCB to the DTFIS and resets the ISMOD address pointer in DTFIS to trap I/O macro requests. It then attaches a subtask to perform I/O macro mapping and returns to the calling routine (IIVGR2).

(7a) The subtask, at entry point IIVIS01, initializes registers and (7b) goes into a wait state until the SIO subroutine (module IIVPCE) posts the ECB in the ISBLK.

(8) The DOS problem program gets a forced program check when it issues an I/O macro for the OS indexed sequential data set.

(9) IIVPCE, recognizing the program check as a special type, puts the address of the ISBLK associated with the DTFIS and the code of the I/O macro to be performed into a NOP CCW. It sets the local execution PSW address to a DOS WAIT macro contained in the DTFIS. IIVPCE then translates the forced program check to an SVC 0 interrupt, switches PSWs, and returns to the DOS supervisor.

(10) The DOS supervisor queues the I/O request for processing.

(11) The DOS problem program then executes the DOS WAIT macro.

(12) The DOS supervisor isseus a start I/O.

(13) The SIO subroutine (module IIVPCE) determines that the request is for an OS indexed sequential data set, loads the ISBLK address from the CCW, and moves the code from the CCW to the ISBLK control information field. It also moves the addresses of the DTFIS and the COMTAB entry associated with the I/O request into the ISBLK control information field.

The SIO subroutine then posts the ISBLK ECB to activate the subtask, increments the I/O count, and returns to the DOS supervisor at the instruction following the SIO.

(14) The activated subtask interprets the I/O macro code byte in the ISBLK control information field and goes to the related mapping routine. After executing the OS macro and moving the logical record to or from DOS storage, it maps pertinent information from the DCB to the DTFIS, indicates successful completion in the COMTAB IOB, posts the ECB in the COMTAB entry associated with the I/O device, and reenters the wait state.

(15) The Emulator task determines that input/output operations are completed on the device associated with the COMTAB entry and simulates an I/O interruption to the DOS supervisor.

Close Processing

After executing steps (4) through (5) indicated above, IIVIS determines that the close function is the one to be performed. It detaches the subtask, resets the DTFIS to its former state, closes the OS indexed sequential data set, removes ISBLK from the ISFIDBLK chain, frees the space, and returns to the DOS problem program.

SETL Processing

After executing steps (4) through (5) indicated above, IIVIS determines that the SETL function is the one to be performed, maps the DOS SETL to an OS SETL, and returns to the DOS problem program.

Shared DOS System Residence File

The DOS system residence file must reside on a standard DOS system residence volume.

The option is selected by coding the DOS system residence file ID as the DSN parameter in the SYSEMcuu DD statement that defines the volume for the DOS cuu specified in the DOSRES=cuu parameter of the EXEC statement. If DISP=OLD is coded, the DOS system residence file will not be shared although the DOS system residence volume will be treated as a device shared volume (may contain OS data sets). DISP=SHR must be coded on the DD statement in order to share the DOS system residence file.

When the DOS system residence file is being shared by two or more Emulator partitions, a separate cylinder on the DOS system residence volume must be provided for each Emulator partition to use for the DOS label cylinder.

A DD statement with the special ddname SYSEMLBL is required to define a single cylinder OS data set to be used for the DOS label cylinder. If this DD statement is missing, the standard DOS label cylinder contained within the DOS System residence file will be used. The DSN parameter may specify any valid data set name. Space allocation must be for one cylinder when the data set is created (DISP=NEW specified on the DD statement) and initialization of the data set must be performed with the DOS STDLABEL and PARSTD procedures.

DOS DLBL and EXTENT statements are not required in the DOS job stream for the DOS system residence file and DOS label cylinder data sets.

ABNORMAL END CONDITIONS

Serious user or program errors can cause the termination of either a DOS job or the Emulator itself. Each of the following error conditions will cause the cancellation of a DOS job requesting the specific I/O or DTF processing:

- Invalid CCW
- Emulator CCW chain table (BEBLK) overflow
- No seek or bin number for a 2321 data cell drive
- Invalid DOS logical unit
- DOS logical unit unassigned or assigned IGN for a shared device
- DTF points to a null DOS PUB table entry
- Cannot match file ID with dsname
- Incompatible file organization
- DCB and DTF device types incompatible
- DOS POINT MACRO not within extents of a given file
- Invalid DOS seek address
- Insufficient extent space in DOS DLBL/EXTENT image for DTF
- User labels specified in DTF but not in DSCB
- Cannot get more extents for file
- Cannot obtain F1 for file

Emulator termination will be caused by any of the following conditions:

- An attempt by DOS to load a hard wait PSW
- An invalid or undefined DOS IPL or DOSLOG device
- Insufficient storage for emulation
- Open for DOSRES was unsuccessful
- Could not find PUB entry in the DOS PUB table during DOS IPL (When DOSIPL ≠ AUTO)
- A DOS PUB entry did not exist for the DOSIPL device
- Device not supported by the Emulator
- DOS device address in DD statement not compatible with OS device type
- Invalid DOS device address specified in DD statement (when DOSIPL ≠ AUTO)
- Syntax errors in DEBUG statement when using card input
- Duplicate DOS device addresses
- Could not IPL DOS for DOS emulation
- Permanent I/O error while loading IPL routines
- Invalid automatic ADD request
- Error while canceling the Emulator

A dump of the contents of the DOS storage area and registers is taken only if a SYSSNAP DD statement is present and the error return code is other than 16, 20 or 24. Control is then returned to OS.

DETAILED ROUTINE DESCRIPTIONS

Each Emulator routine description consists of the following parts:

- A statement of the general function(s)
- A narrative description of the function(s)
- Operational diagrams when necessary

DOS Emulator Entry Routine (IIVENT) -- Flowchart 1A

IIVENT, the controlling routine during Emulator initialization, passes control to IIVINT and IIVIN2 so that these modules can perform initialization functions for the Emulator.

IIVENT saves the contents of the general purpose registers and the pointer to the user parameter area. The CSECT, IIVCON, which is used for communication between the modules of the Emulator, is defined in this module.

Initialization First-Load Routine (IIVINT) -- Flowcharts 2A-2M

IIVINT performs four functions in initializing the Emulator:

- Verifies parameters
- Establishes DOS storage
- Builds COMTAB
- Builds COMTAB extension

On entry to this module, the local execution list and adjust CCW data address list are aligned to 64-byte boundaries within the CSECT containing the Emulator constants. The Emulator base registers are initialized as follows:

- Register 9 contains the address of the local execution list.
- Register 10 contains the address of DOS storage.
- Register 11 contains the address of the Emulator constants area.

Verifies Parameters

The address of the user parameter area is obtained from IIVENT. The parameter area is scanned for the DOS channel and unit number of the DOS resident file (DOSRES=), the DOS IPL unit (DOSIPL=), the DOS console (DOSLOG=), and the DOS system size (DOSSYS=). These parameters are validated and checked to ensure that required parameters have been specified correctly. If a required parameter was not provided on the EXEC statement, a message is issued to the operator requesting the information.

IIVINT also looks for three other parameters:

```
[,AUTOEOJ={YES}]   specifies whether automatic ending of emulation is
[         {NO} ]   desired

[,DOSTIM={YES}]    specifies whether timer support is desired
[        {NO} ]

[,APROMPT={YES}]   requests an abbreviated initial prompt
[         {NO} ]
```

AUTOEOJ defaults to NO, DOSTIM defaults to YES, and APROMPT defaults to NO.

Establishes DOS Storage

DOS operates in the first block of storage obtained by the Emulator (Figure 14). The DOS storage size parameter is used to determine the amount of storage needed. The value specified in the DOSSYS parameter is increased by 4K bytes and the Emulator GETMAIN routine (IIVGET) is called. The extra 4K bytes are used to adjust DOS storage to a 4K boundary. After DOS storage is adjusted, the beginning address is saved and the remaining storage at each end of the DOS area is freed. All of DOS storage is initialized to zeros.

Builds COMTAB

The ddnames in the TIOT that begin with the characters SYSE are counted to calculate the number of devices allocated to DOS. The count is used to compute the size of COMTAB and is placed in the Emulator constants area (IIVCON) for future use in computing the size of other tables. The size of COMTAB is the DD entry count plus 1 for SYSLOG times the size of one COMTAB entry. Storage area is obtained for COMTAB by the Emulator GETMAIN routine (IIVGET).

Each COMTAB entry is initialized to 0 and the following data is entered:

- The DOS channel and unit address is entered from the TIOT ddname.
- The OS channel and unit address is entered from the UCB channel and unit address field.
- The OS device type is entered from the UCB.
- The CTFLAG fields are used to flag the entry for DOSLOG. They are also used to flag staged and shared devices. The JFCB is checked to determine whether a device is staged (JFCBTSDM set to X'20') or shared.
- If the JFCB indicated a staged device, CTFLAG2 is set to indicate whether the staging is for an input or output device (determined from byte 18 of the data set name).
- If the JFCB indicates a shared device, a COMTAB extension entry is temporarily built in the COMTAB entry. This temporary COMTAB extension consists of the ddname and dsname. The shared device flag in COMTAB is set to indicate the presence of a temporary COMTAB extension.
- The DOS SYSRES flag is set in the COMTAB entry associated with the DOS system residence volume.

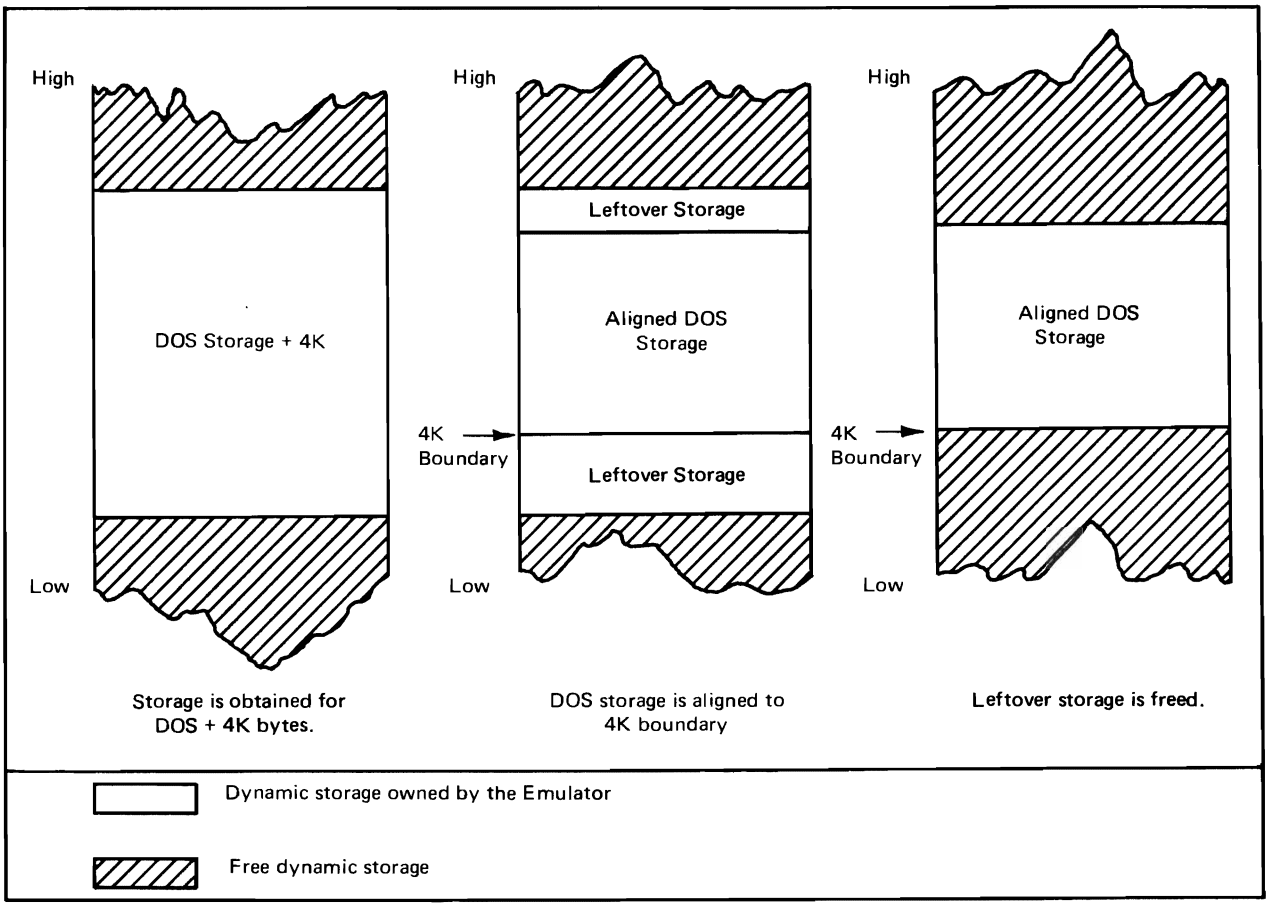


Figure 14. DOS Storage in Emulator Region

Builds COMTAB Extension

The number of ddnames in the TIOT that begin with three characters other than 'SYS', plus the DD statement labeled 'SYSEMLBL' if present, are counted to find out the number of data sets that reside on shared direct-access devices. The count is used to compute the storage needed for the COMTAB extension. This storage is obtained by means of module IIVGET.

An appropriate error message is issued followed by Emulator termination if:

- The OS cuu associated with the SYSEMLBL DD statement does not match the OS cuu for the DOS system residence volume.
- DISP=SHR was specified in the SYSEMLBL DD statement.

Each COMTAB extension is formatted with:

- The UCB channel and unit address (used during Emulator initialization only and not present in the COMTAB extension after initialization)
- The data set name from the JFCB
- The ddname from the TIOT entry
- A flag is set in the flag byte if the ddname is SYSEMLBL

IIVINT then returns to IIVENT.

The following Emulator subroutines are used by IIVINT:

SCAN Subroutine (Flowchart 2K). This subroutine is used by IIVINT to scan the user parameter area for a blank or comma. When entered, this subroutine computes the address and length of the next user parameter.

This subroutine has two returns to the user. One return indicates that the end of the area being scanned has been reached (scan completed). The other return gives the beginning address of a field in register 3 and the length of the user parameter in register 4.

YESORNO Subroutine (Flowchart 2K). This subroutine is used to check the validity of parameters that require YES or NO responses. Its return is into a branch table:

- Return address + 0 - parameter missing
- Return address + 4 - parameter = YES
- Return address + 8 - parameter = NO
- Return address + 12 - parameter invalid

CHKCUU Subroutine (Flowchart 2L). CHKCUU checks the parameter value given on the Emulator EXEC statement. It verifies the validity of a channel and unit address. This routine has two returns. It returns to the address in register 14 plus 4 if the cuu checked was missing, had an invalid length, or had an invalid range. If the cuu checked was valid, it returns to the address in register 14.

3066?

ASKOPR Subroutine (Flowchart 2L). This common subroutine is used by IIVINT to request required parameters from the operator from the operation that were omitted or correction if invalid parameters from the operator. It issues a WTOR to the console printer and waits on the reply. The reply is checked. If it was CANCEL, ASKOPR branches to the common emulation termination routine; otherwise, it returns to the calling routine.

DDSCAN Subroutine (Flowchart 2M). This subroutine scans the DDnames in the TIOT for entries with DDnames beginning with SYSE. If register 2 contains zeros, this subroutine gets the address of the TIOT from the Emulator constants area (IIVCON) and starts scanning the TIOT for an SYSE entry. If register 2 does not contain zeros, the address it contains is assumed to be the starting address within the TIOT for the scan.

There are three exits from this initialization subroutine. A return to the address in register 14 denotes that all TIOT entries have been examined. If this subroutine returns to the address in register 14 plus 4, the address of an SYSE entry is in register 2. When it returns to the address in register 14 plus 8, register 2 contains the address of a TIOT DDname that does not begin with SYS.

Initialization Second-Load Routine (IIVIN2) -- Flowcharts 3A-3F

The second-load initialization routine performs the following initialization:

- Moves temporary COMTAB extensions from COMTAB to available COMTAB extension
- Chains COMTAB entries to COMTAB extensions and chains together COMTAB extension entries
- Sorts COMTAB entries
- Builds and initializes other Emulator tables, such as the ISK/SSK table, the ECB pointer table, and the post ECB list
- Performs IPL from DOS resident volume

Moves Temporary COMTAB Extension

The COMTAB table is searched for temporary COMTAB extension entries. All temporary COMTAB extensions are moved to available COMTAB extension entries.

Chains COMTAB Entries and COMTAB Extensions

One COMTAB entry exists for every DOS PUB table entry being used. If a COMTAB entry is marked for direct-access device sharing, then there exists at least one COMTAB extension that is related to the COMTAB entry. A COMTAB extension entry describes a data set residing on a DASD shared volume. Chaining is required since there may be a group of related COMTAB extensions associated with one or more COMTAB entries.

Each COMTAB entry is chained to the first related COMTAB extension entry and all associated COMTAB extension entries are forward chained together (Figure 15).

Sorts COMTAB Entries

The COMTAB entries are sorted by the DOS channel and unit address and placed in ascending order.

This section of initialization is enqueued by means of a master ENQ to ensure that only one DOS Emulator will be building a COMTAB at any given time. To prevent more than one DOS Emulator from using the same OS direct-access device, the OS channel and unit address is also enqueued by means of a device ENQ. At the end of the device enqueueing routine, a master DEQ is performed. Other DOS Emulators may not use direct-access devices reserved by the device ENQ for this Emulator unless both Emulators specify DISP=SHR in their DD statements.

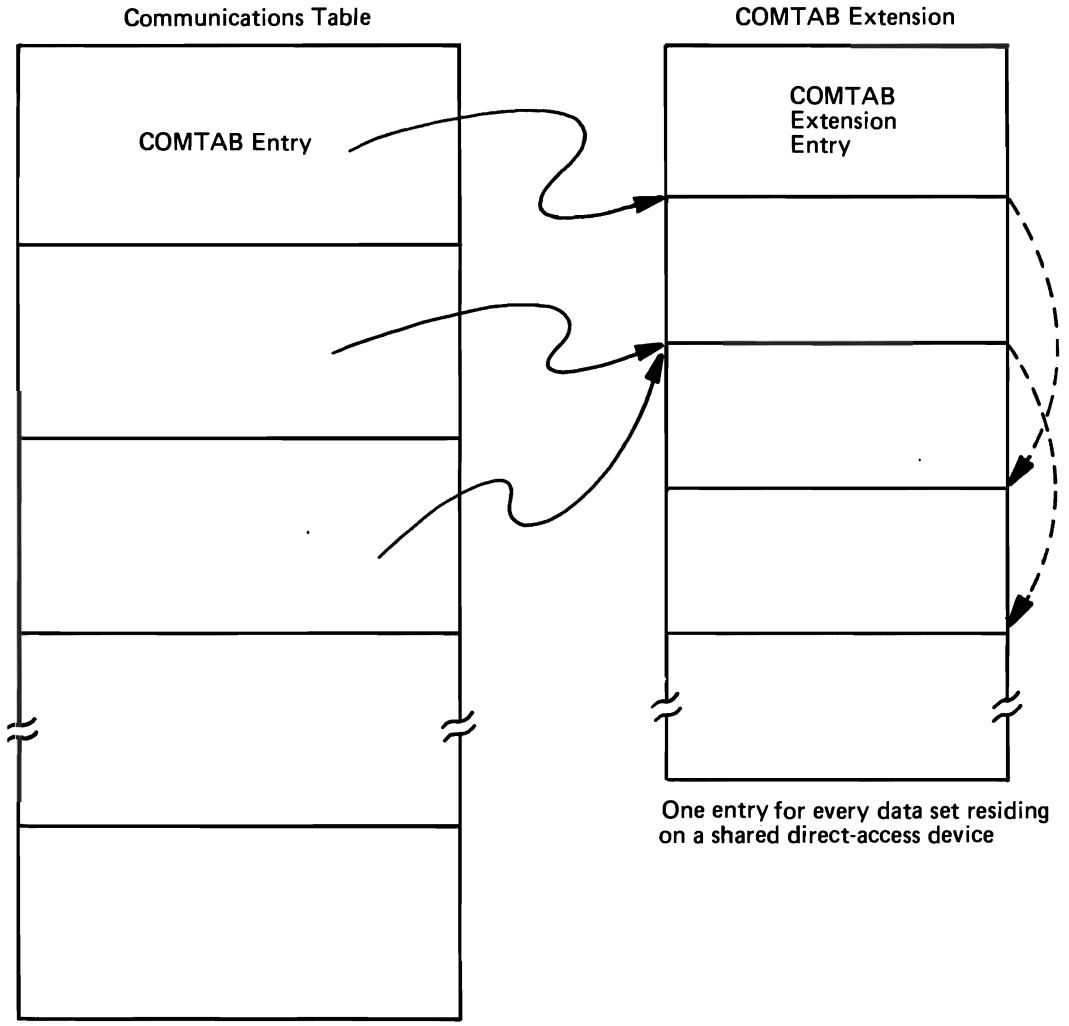


Figure 15. Relationship of COMTAB to COMTAB Extension

Builds and Initializes Other Emulator Tables

This section of initialization code obtains storage area and initializes the ISK/SSK table, the ECB pointer table, and the post ECB list.

- One byte for every 2K bytes of DOS storage is obtained for the ISK/SSK table. This table is initialized to zeros.
- The size of the ECB pointer table is the number of entries in COMTAB plus 3, times the length of one entry. Each entry consists of a 4-byte address. Storage is obtained for this table; the first entry is initialized with the address of the prompt ECB, the second with the address of the WTOR ECB, and the third with the address of the timer ECB. The rest of the entries are initialized sequentially with the addresses of the ECBs entered in COMTAB.
- Storage is obtained for the post ECB list so that the list contains a list of 2-byte entries, each representing a COMTAB entry. The table is initialized to zeros.

Performs IPL from DOS Resident Volume

The initialization second-load routine builds and opens a DCB for the DOS system residence volume or the DOS system residence file if the shared DOS system residence option has been selected. The COMTAB entry for the DOS resident volume is modified to reflect that the volume is open and to contain the address of the DCB. The data set specified in the SYSEMLBL DD statement is opened if present.

DOS IPL is a two-phase program consisting of two DOS programs:

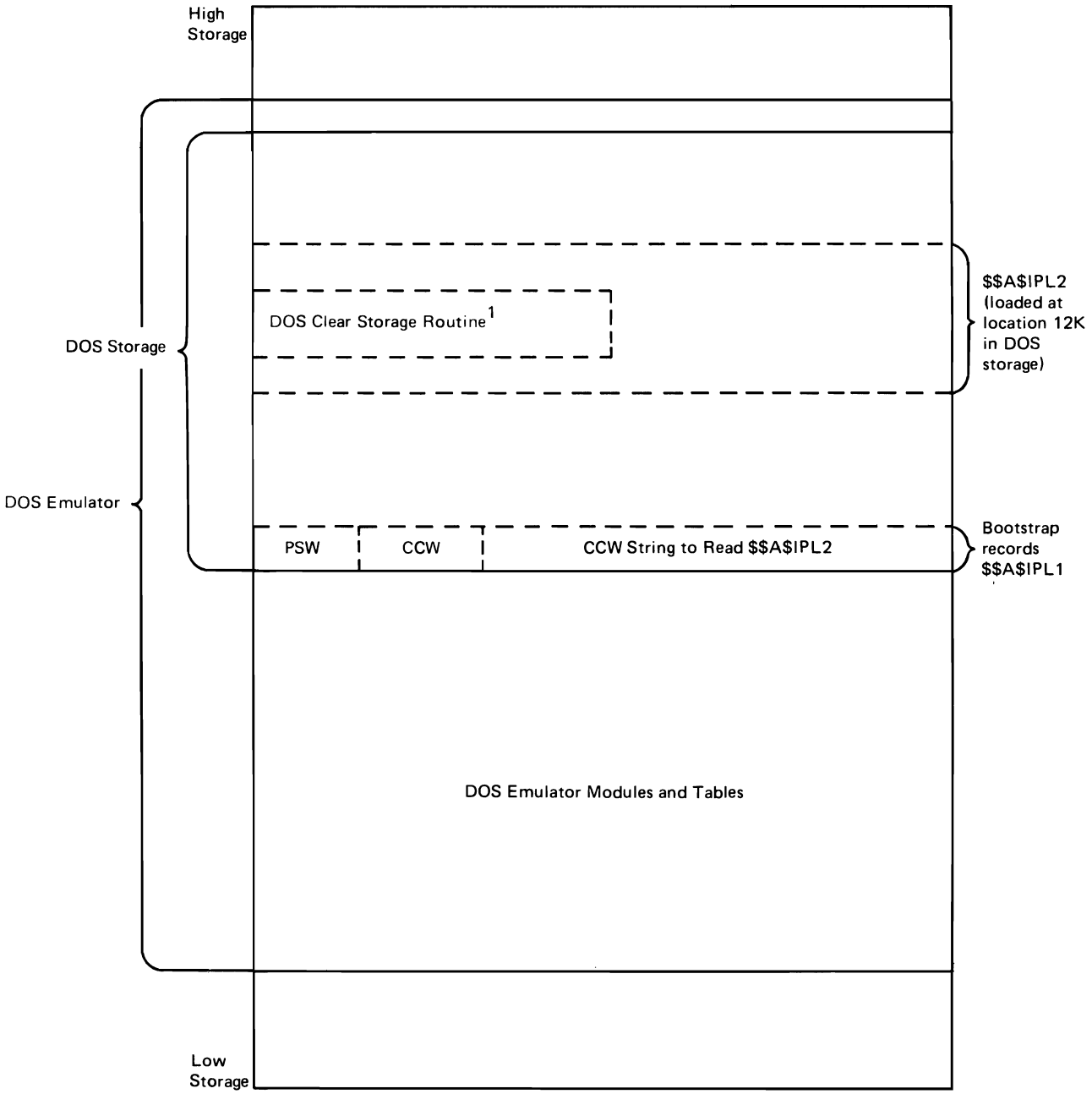
- $$A$IPL1, 64-byte bootstrap records, and
- $$A$IPL2, the first load of a DOS IPL phase of fewer than 4096 bytes.

The DOS $$A$IPL1 bootstrap records are located on DOS SYSRES at 00 00 1 (CC HH R) and 00 00 2 (CC HH R). The Emulator builds the necessary channel programs to read the bootstrap records into DOS storage at DOS location 0. The bootstrap records consists of a PSW at DOS location 0, followed by seven CCWs. The Emulator adjusts the data addresses in the last three CCWs. When these three CCWs are executed, the DOS IPL phase ($$AIPL2) is loaded at location 12K in DOS storage.

At this point, $$A$IPL2 in the DOS storage area is ready for executing its own IPL. IIVIN2 scans the $$A$IPL2 phase for the first DOS instruction in the loop that zeros the DOS storage area in search of the DOS system size. When the initialization second-load routine finds that instruction (an MVC) in $$A$IPL2, it replaces the operation code with zeros.

Later, when DOS executes this instruction, it gets a program check. Control is given to the Emulator program check executive routine (IIVPCE). The Emulator simulates a program interruption for addressing, places the DOS storage area ending address in register 11, and reenters local execution mode at the location pointed to by the DOS program new PSW.

If staged I/O support has been requested, module IIVIN2 loads IIVSTG and IIVSDT. If a //SYSDEBUG DD statement was present, module IIVIN2 calls IIVRAS to read debugging parameters. The initialization second-load routine gives control to module IIVRTE or, if error conditions were detected, module IIVENT. Figure 16 is a map of the Emulator region at this time.



[1] The Emulator zeroes out the op code of the first DOS instruction in the DOS clear storage routine.

Figure 16. OS Region at Beginning of DOS IPL

IPL Add Routine (IIVADD) -- Flowcharts 4A-4D

Module IIVADD is entered only when the DOSIPL parameter is coded with the AUTO option. When IIVSVC calls this routine, it provides a pointer to a DOS channel program that reads in the IPL control statements. When entered for the first time, it finds the DOS temporary PUB table and fetches the beginning address of COMTAB. Thereafter, processing begins with the next COMTAB entry in the table.

This routine checks the cuu entry in the DOS PUB table against the cuu entry in COMTAB and if a corresponding entry does not exist in COMTAB, the DOS PUB table entry is deleted. The routine also provides an automatic set date and time function during IPL.

The COMTAB is checked for DOS cuu's that are not in the DOS PUB table. If any are missing, this routine adds them to the table. To add a device to the DOS PUB table, this routine builds a DOS ADD control statement image and places it in the DOS input area. Upon return to DOS, the IPL control statement image provided by this routine will be processed.

When it is determined that all COMTAB entries have corresponding DOS PUB table entries, a set date and time IPL control statement is built and passed to DOS. The OS date and time are used.

Open Routine (IIVOPN) -- Flowcharts 5A-5D

This routine is called to open Emulator-allocated devices dynamically. The caller supplies a pointer to the COMTAB entry requiring the open. The open routine gets dynamic storage for the DCB, initializes it, and places its address in the COMTAB entry. The device is then opened to permit accessing.

Five major types of devices are opened by this routine (Figure 17):

- New volume mount
- Direct-access device
- Sequential device
- Staged device
- Teleprocessing device

Initialization

The calling routine places the address of the Emulator COMTAB entry for the device to be opened in register 0. Register 15 is initialized with the entry point to the open routine. The CTFLAG fields in the COMTAB entry indicate whether the open request is for a new volume mount or for a DASD, a sequential device, a teleprocessing device, or a staged device.

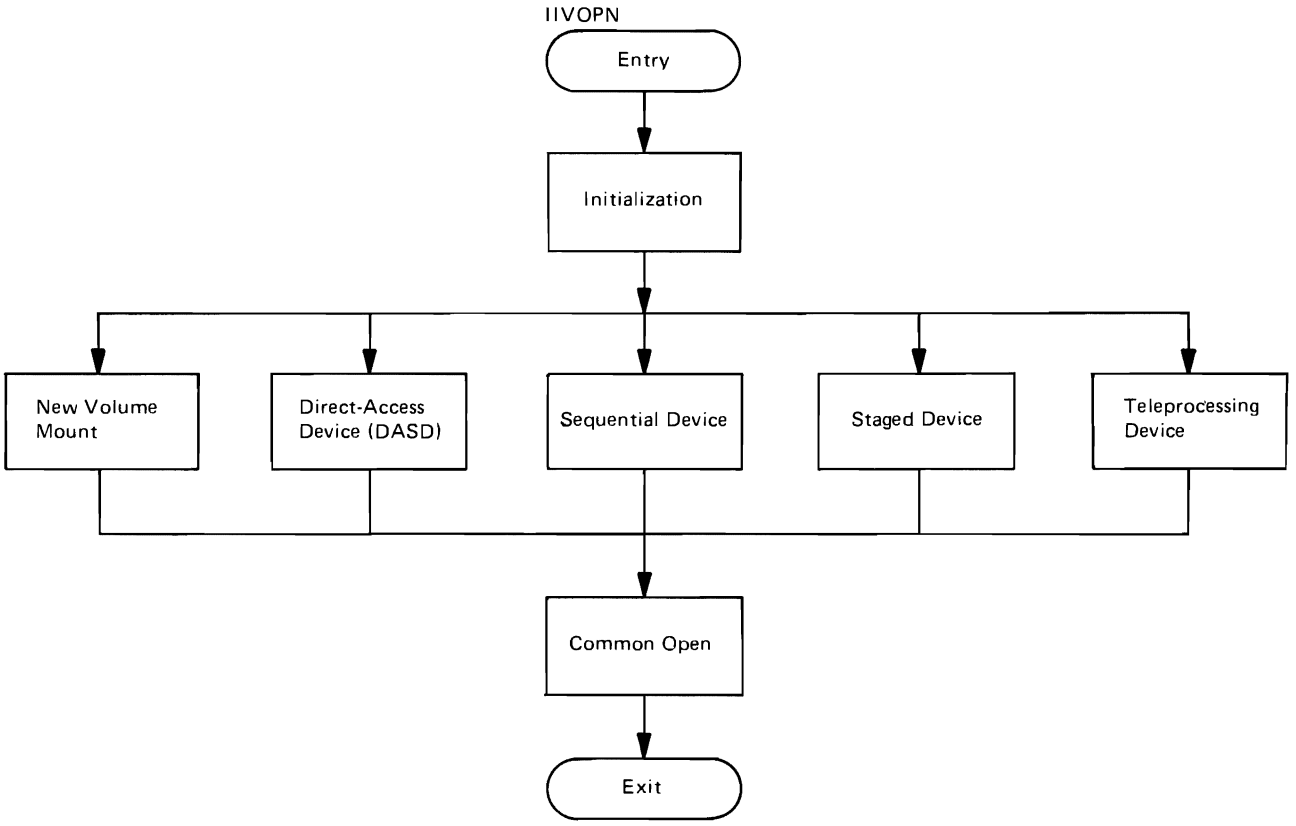


Figure 17. Open Subroutine Gross Flow

New Volume Mount

To perform an open operation for the purpose of mounting a new volume, this subroutine gets the address of the existing DCB from the COMTAB entry. It reads the JFCB into storage and changes the volume serial number. The new volume serial number is retrieved from the Emulator CSECT IIVCON, where it was placed by the prompt reply processor routine (IIVPRP).

Direct-Access Device

Storage is obtained for the DCB, which is formatted to describe a direct-access data set. The JFCB for this COMTAB entry is altered so that the data set name is the same as that of the VTOC data set (X'0404.....').

Sequential Device

Dynamic storage is acquired and formatted to describe a sequential data set. The JFCB is read into storage and modified for the bypassing of label processing.

Staged Device

storage is established for the staged I/O DCB, it is formatted for either an input or output direct-access data set, depending on the settings of CTFLAG2 bits 3 and 4. The IOB for staged I/O is a dummy IOB (STGCON) and is used as a parameter area by the staged I/O processing modules. It is formatted by the open routine according to the DOS device type. For more information on the staged I/O dummy IOB, see the staged I/O routine (IIVSTG). If the DCB was initialized to reflect an output data set, the open operation is performed within the staged I/O portion of the open routine; otherwise, the common open for input issues the open request.

When module IIVOPN is entered to open a staged printer, print overflow processing is invoked. Module IIVOPN loads the correct FCB image from SYS1.IMAGELIB and calls module IIVSTG to verify the image format and content. If module IIVSTG returns a unit check in the staged IOB CSW, an invalid FCB image has been obtained and emulation is terminated. If the SYS1.IMAGELIB data set could not be opened, the open routine issues a WTOR (IIV040D) to let the operator decide whether to continue without printer overflow support.

Teleprocessing Device

Dynamic storage for a DCB is acquired and formatted to describe a sequential data set.

Common Open

A common open operation for input is executed for most types of open requests by this portion of the open routine. If the data set (file) is opened successfully, this subroutine flags the COMTAB entry (CTFLAG=X'01'). If the open operation terminated unsuccessfully, an error code is returned to the caller in register 15. X'04' indicates a successful open, and X'00' indicates an error in opening the device.

When a direct-access device other than a staged I/O device is opened, the open routine issues an EXCP to read the format 4 DSCB record from the VTOC. This EXCP also allows the SIO appendage to extend the DEB extents. By analyzing the format 4 DSCB, the open routine determines whether the VTOC overlaps cylinder boundaries and issues a message if it does. This subroutine scans the VTOC for OS password data sets and does not permit use of volumes containing such data sets.

OS PUB Table Build Routine (IIVPUB) -- Flowcharts 6A-6F

This routine maps DOS I/O control blocks to OS I/O control blocks by means of the Emulator OS PUB table. (Storage is obtained by module IIVGET.) An OS PUB table entry points to a COMTAB entry, which contains addresses of OS control blocks needed for I/O operations.

This routine checks the DOS PUB table for a match of each COMTAB entry in the DOS channel and unit address field. If a corresponding DOS PUB table entry is not found, the Emulator is terminated. When a DOS PUB table entry for a COMTAB entry is found, the COMTAB entry offset (X'00' for the first

entry) is placed in the OS PUB table entry that corresponds to the DOS PUB entry. An X'FF' in an OS PUB entry indicates that no COMTAB entry existed for a DOS PUB table entry.

This routine also establishes a pointer to the DOS CE serviceability routines if these are supported (the CE table address field in the DOS background communications region extension points to these routines). If the timer function was requested by the user and is supported by DOS, the first STIMER is issued. The addresses of the DOS background SYSRDR LUB (logical unit block), background PIB (program information block), foreground 1 PIB, foreground 2 PIB, and beginning and ending addresses of the DOS B-transient area are obtained. These addresses are adjusted to true addresses and saved in IIVCON.

If an abbreviated prompt was requested by the user, the one-line initial Emulator prompt is formatted and issued. Otherwise, the standard three-line initial Emulator prompt is given.

If the Emulator JCL indicated a need for direct-access volume sharing or indexed sequential data set sharing, Emulator modules IIVDVS, IIVGR2, IIVVIO, and IIVIS are loaded.

If the shared DOS system residence option has been selected, the address of the cylinder used for DOS labels is saved in EMUCONS.

IIVPUB exits to the caller (IIVSVC).

GETMAIN/FREEMAIN Routine (IIVGET) -- Flowchart 7A

This routine performs OS GETMAINs and FREEMAINs.

GETMAIN

The caller provides IIVGET with the amount of storage needed. The OS GETMAIN macro is issued and the return codes checked. If storage was successfully obtained, this routine returns to the caller with the new storage address in register 1. If dynamic storage was unavailable, a message is issued to the console printer and emulation is terminated.

FREEMAIN

The caller provides IIVGET with the amount of storage (register 0) and the address of the first byte of the storage (register 1) to be freed. IIVGET issues an OS FREEMAIN macro and returns to the caller.

For all errors during a GETMAIN or FREEMAIN operation, other than storage not available, the operating system terminates the Emulator with a system completion code.

Program Check Executive Routine (IIVPCE) -- Flowcharts 8A-8U

All program interruptions that occur in the DOS area are routed to this routine. Upon entry, the 16 DOS general registers are saved in the EMUCONS save area and program addressability is established.

A check is made for a privileged operation interruption code in the current DOS PSW. When the program interruption is caused by an attempt to execute a privileged operation, control is passed to PCPRIVOP for further checking.

The first program interruption during IPL is intercepted, and control is passed to the FIRSTPC subroutine for processing. All other program interruptions are handled by a routine that causes the DOS current (local execution) PSW to be moved to the DOS program old PSW (location 40 in the DOS area) and the DOS program new PSW (location 104 in the DOS area) to be moved to the DOS current (local execution) PSW location to simulate a program interruption.

At the common exit point for all simulation routines (PC20), the routine loads the address of the Emulator asynchronous interrupt exit routine and exits to that point. A reentry point (PC10) for program interruption simulation is provided.

FIRSTPC Subroutine (Flowchart 8A)

To simulate the program interruption that normally occurs while main storage is being set to binary zeros during DOS supervisor IPL, module IIVIN2 places a 0 operation code in the DOS routine that performs this function. The resulting program interruption is the first one to be intercepted by the program check executive routine.

When given control, this subroutine checks for an operation exception interruption code in the current PSW; if the check is negative, control is returned to the program check executive routine at PC10 to simulate the program interruption. Otherwise, the interruption code is changed to addressing, the ending address of DOS main storage is placed in register 11 of the EMUCONS DOS register save area, and the FRSTPC bit in the IPLSW switch is set to 0. Control is then returned to the program check executive routine (entry point PC10) to simulate the program interruption.

PCPRIVOP Subroutine (Flowchart 8A)

All privileged operation interruptions are checked to determine if they are to be simulated. For such an interruption to qualify for simulation, the local execution PSW must show the DOS partition to be operating in the supervisor state (see Note in "Privileged Operation"), and the privileged operation must be one of the following which are either supported or bypassed by the Emulator.

- ISK - Insert storage key, operation code X'09'
- SSK - Set storage key, operation code X'08'
- SSM - Set system mask, operation code X'80'
- LPSW - Load PSW, operation code X'82'
- TCH - Test channel operation code X'9F'
- TIO - Test I/O, operation code X'9D'
- HIO - Halt I/O, operation code X'9E' (BTAM only)
- SIO - Start I/O, operation code X'9C'
- LCTL - Load control, operation code X'B7'
- STCTL- Store control, operation code X'B6'
- STIDP- Store CPU ID, operation code X'B202'
- STIDC- Store channel ID, operation code X'B203'
- SCK - Set clock, operation code X'B204'

When both conditions are met, control passes to the proper simulation routine. If both are not met, the routine exits to the program check executive routine (entry point PC10) to simulate a program interruption.

The condition code in the local execution PSW is set to 0 and control is passed to entry point PC20 if the privileged operation is to be bypassed. Two privileged instructions fall into this category; store channel ID (STIDC) and set clock (SCK).

ISK Simulation Subroutine (Flowchart 8C)

This subroutine simulates the ISK instruction. A branch and link to the FINDKEY subroutine is done to get the address of the R1 register of the ISK from the DOS register save area of EMUCONS. The FINDKEY subroutine also gives the address of the storage key in the ISK/SSK table for the block represented by the address in the R2 register. The storage key is moved from the table into bits 24-31 of the R1 register. Control is returned to the program check executive routine (entry point PC20).

SSK Simulation Subroutine (Flowchart 8C)

This subroutine simulates the SSK instruction. A branch and link to the FINDKEY subroutine is done to get the address of the R1 register in the DOS register save area of EMUCONS and the address in the ISK/SSK table of the storage key for the block represented by the address in the R2 register. Bits 24-31 of the R1 register are moved into the table. Control is returned to the program check executive routine (entry point PC20).

SSM Simulation Subroutine (Flowchart 8C)

This subroutine simulates the SSM instruction. The address of the new system mask is obtained with the FINDADDR subroutine and the byte at the address is moved into the first byte of the local execution PSW. Control is returned to the program check executive routine (entry point PC20).

LPSW Simulation Subroutine (Flowchart 8D)

This subroutine simulates the LPSW instruction. The FINDADDR subroutine is used to find the address of the new PSW. The eight bytes of data at that address in the DOS area are moved to the local execution PSW.

Before the routine exits, a check is made to see if this is the first LPSW during IPL. If not, exit is made to the program check executive routine (entry point PC20). The first LPSW during IPL should be an enabled wait for the IPL interruption. Control is passed to entry point INTRPT and the subroutine checks the first PSW loaded by the DOS IPL routines. When the wait bit is 1 and interruptions are enabled, the expected interruption is simulated according to the parameter on the EXEC statement. If these conditions are not met, control is returned to the program check executive routine at entry point PC20. When they are met, the first LPSW bit switch in IPLSW is set to 0 and the interruption type is determined by examination of the DOSIPL parameter.

When DOSIPL=EXT, an external interruption is simulated. The DOS current (local execution) PSW is moved to the external old PSW (location 24 in the DOS area), and the external new PSW (location 88 in the DOS area) is moved to the local execution PSW location.

When the DOSIPL value contains a DOS channel and unit address, that address is moved to the interruption code portion of the local execution PSW and an I/O interruption is simulated. The local execution PSW is moved to the I/O old PSW (location 56 in the DOS area), and the I/O new PSW (location 120 in the DOS area) is moved to the local execution PSW location in the local execution list. The CSW is set to 0 and the device end bit is turned on. If the DOSIPL device is the DOS console, the attention bit in the CSW is also turned on. Control is returned to the program check executive routine at entry point PC20.

TCH Simulation Subroutine (Flowchart 8E)

This subroutine simulates the TCH instruction. The FINDCHAN subroutine is used to obtain the channel and unit address from the instruction and the index and absolute address of the first device entry on the channel in COMTAB.

The CTFLAG device busy bit in COMTAB is tested to see if any of the devices on the channel are busy. If none are found to be busy, the condition code in the local execution PSW is set to 0 (channel available). If one is found to be busy, the NOP issued bit in CTFLAG3 is tested to determine if a device end status is being awaited. If this bit is set, the next device is checked for a busy condition; otherwise, the condition code is set to 2 (channel busy). Control is returned to the program check executive routine at entry point PC20.

TIO Simulation Subroutine (Flowcharts 8F-8G)

This subroutine simulates the TIO instruction. The FINDCHAN subroutine is used to obtain the channel and unit address from the instruction and the index and absolute address of the first device entry on the channel from COMTAB.

The COMTAB entries are scanned, starting with the first device on the channel, for the device being tested. If no match is found, the condition code in the local execution PSW is set to 3 (device not operational). When a match is made, a further test for device operational is made by testing the nonoperational bit in the CTFLAG byte of COMTAB. If it is 1, the device is not operational and the condition code is set to 3.

The CTFLAG device busy bit in COMTAB is tested for a value of 1 to determine whether it is busy. When the bit is 0, the device is not busy unless the device is a 2321 DASD unit (all 2321 bins used by DOS must be checked). When the device is not a 2321 or all bins are not busy, the current PSW condition code is set to 0 (device available). When the device is busy, further testing is required to determine if an interruption is pending.

The first check for interrupt pending is made on the ECB completion code. During IPL, the ECB is waited upon if it has not already been posted in order to save going through a TIO loop until I/O completion. Other times, the pending I/O table is scanned for an entry for the device. If the I/O is not complete, the local execution PSW condition code is set to 2 (device busy).

When an I/O operation has been completed, the ECB code, the COMTAB index, and the address of the device entry are passed to the check subroutine (at entry point CHECK2) to get the stored channel status word. Upon return, the condition code is set to 1 (CSW stored).

Control is returned to the program check executive routine at entry point PC20.

HIO Simulation Subroutine (Flowchart 8H)

This routine receives control when the privileged operation code HIO is intercepted. The routine simulates the HIO instruction to DOS. The FINDCHAN subroutine is used to obtain the channel and unit address from the instruction and the index and absolute address of the first device on the channel from COMTAB.

The COMTAB entries are scanned, starting with the first device on the channel, for the device being tested. If no match is found, the condition code in the local execution PSW is set to 3 (device not operational). When a match is made, the routine tests the nonoperational bit in the CTFLAG byte of COMTAB. If it is 1, the device is not operational and the condition code is set to 3. The CTFLAG device busy bit in COMTAB is tested for a value of 1 to determine whether it is busy or not. If the device is active, a halt I/O operation is performed.

This routine returns to the calling routine at the address specified in the return register of the calling routine. The return register is increased by a value that determines the proper branch instruction in a branch table. The calling sequence is as follows:

- Device not operational - Return to the program check executive routine at entry point PC20.
- Device not busy (flagged 'not in use' by the Emulator) - Set CSW status to 0, set current (local execution) PSW condition code to 'CSW stored,' and return to the program check executive routine at entry point PC20.

SIO Subroutine (Flowcharts 8J-8N)

The SIO routine issues an OS EXCP macro for the intercepted DOS SIO. DOS CCWs are used, the addresses being adjusted from local to true addresses. The SIO routine is given control by the program check executive routine when the privileged operation code is recognized as SIO.

The proper COMTAB entry is selected. (The COMTAB entries contain the various Emulator parameters required for each device.) Normally, the entry is found by means of DOS register 3, which contains a pointer to the appropriate DOS PUB entry at SIO. The position of the current PUB in the DOS PUB table is used as an index to an OS PUB table created in the Emulator's initialization phase. The indexed OS PUB entry contains an index of the proper COMTAB entry.

During IPL, before the DOS PUBs exist, and for CE serviceability routine operations (where DOS register 3 does not point to a PUB entry), the COMTAB entry is determined by a search for channel and unit address in COMTAB. The cuu is found by the FINDADDR subroutine. The DOS CAW local address is adjusted to its true address.

If the device is a 2321 Data Cell Drive, the COMTAB entry, determined as described above, is for the highest defined DOS bin number. The entry for the actual bin number to be used is determined by a search of all COMTAB entries having the same DOS cuu for the bin number contained in the DASD seek address. If a seek does not precede the channel program, the address of the last 2321 COMTAB entry for the device is loaded from its save position in the last COMTAB entry.

DOS is given a condition code of X'00' unless the device is busy (X'02') or the device is flagged as not operational (X'03'). If the data set has not been opened, the open routine (IIVOPN) is called.

If the device is a staged I/O device, the SIO count is increased and control is passed to module IIVSTG. If the device is DOSLOG, control is passed to module IIVLOG. If the I/O request was for a shared OS indexed sequential data set, the ECB in the associated ISBLK is posted and a branch is made to SIOCNT.

For direct-access I/O operations, the SEEKTEST subroutine is called to determine if the CCW operation code is seek. This routine also moves the correct DCB address from the COMTAB extension to the IOB for shared volumes (see SEEKTEST subroutine description for further details). The seek address is then moved to the IOB. The initial CCW sequence of SEEK-TIC or SEEK-SET FILE MASK-TIC is bypassed. That is, the IOB CCW start address is set to point to the address immediately after these commands.

For direct-access devices only, flags are set for the stand-alone sense operation or the stand-alone seek, since these commands do not have their addresses adjusted to true addresses by IIVCCW and, hence, must not be readjusted when the I/O interruption occurs. The stand-alone seek is not passed on to OS, but is posted complete to DOS. That is, the ECB is posted complete and the IOB CSW is posted.

For tape operations, the DOS initial SET MODE, if present, is moved to a save area in the device's COMTAB entry. This is necessary so that the DOS set mode opcode can be moved to the OS DEB. The CCW addresses are adjusted to true addresses beginning with the CCW immediately after these initial CCWs. If a rewind-unload is issued by DOS, an 'intervention required' message is printed by the Emulator if any further I/O is requested on that tape drive.

Module IIVCCW is called to adjust CCWs. The IIVCCW routine returns to the address in register 14 plus 0 (error) or 4 (normal). The error return indicates that an error condition was detected during CCW adjustment. The program check bit is set in the DOS CSW status, the condition code is set to 'CSW stored', and a return to entry point PC20 is made.

SEEKTEST Subroutine (Flowchart 8K)

This subroutine gains control to determine if the operation code of a DASD CCW is seek. If it is found to be seek, the address of the DASD seek is returned to the caller. Control is returned to the caller.

SEEKDVS Subroutine (Flowchart 8P)

The DASD seek address is then compared with each lower and upper read or write track address in the DEB. If the seek address does fall within a set of DEB extents, the associated DCB address is moved from the COMTAB extension to the IOB and control is returned to the caller. Otherwise, the next COMTAB extension is checked. If no match is found, control is given to module IIVVIO to determine whether the seek (and channel program) is for a DOS access to VOL1, F1, or F3. IIVVIO returns a condition code of 0 when the I/O has to be issued, a condition code of 4 when the I/O has to be simulated, or a condition code of 8 when the seek address is invalid. In the last case, the routine exits to the exit-ABEND error routine (IIVABN) after issuing the INVALID SEEK message.

FINDCHAN Subroutine (Flowchart 8Q)

This subroutine is used by the TCH, HIO, and TIO instructions to obtain the channel and unit address, the device entry index into COMTAB, and the absolute address of the device entry in COMTAB. The channel and unit address is obtained from the FINDADDR subroutine and stored at CHANUNIT in EMUCONS. The channel number is used to index into the CHNINDX table to determine whether the channel is supported by the Emulator. When the channel is not supported, the condition code in the local execution PSW is set to 3 (not operational) and exit is made to the program check executive routine entry point PC20.

The index into COMTAB for supported channels is obtained from the CHNINDX table and multiplied by the COMTAB entry length. The result plus the starting address of COMTAB is returned to the caller in register RW4EU. The channel index value is returned in register RW0EU. Return is to the address in register RB0CD.

FINDADDR Subroutine (Flowchart 8R)

This subroutine is used to determine the absolute value of the BDDD portion of a DOS privileged operation instruction. The base register bits are used as an index to locate the register value in the register save area. The 1 1/2-byte displacement value is added to the value contained in the register save area to obtain the effective address. If the instruction has an I/O code (TM for X'9C'), the address adjustment factor is not added; it is in all other cases. The resulting value is returned in register RW2EU and return is made to the caller by a branch to the address in register RL0EU.

FINDKEY Subroutine (Flowchart 8S)

This subroutine analyzes the ISK or SSK instruction to obtain the address of the R1 register in the instruction and the address of the entry in the ISK/SSK table that corresponds to the address value in the R2 register. When a main-storage address larger than the DOS area is specified in R2, an addressing program interruption is simulated.

The address of the R1 register is returned in register RW4EU and the address of the 1-byte entry in the ISK/SSK table is returned in register RW3EU.

Load/Store Control Register Subroutine (Flowchart 8T)

The two instructions (load control and store control) are simulated only to the extent that control register information is preserved in EMUCONS for retrieval by DOS.

Load Control Register. Control register information (starting at the DOS main-storage address specified in the LCTL instruction and continuing through as many storage words as the number of control registers specified) is moved into the control register field labeled CTLREGS in EMUCONS.

Store Control Register. Control register information is moved from the control register field labeled CTLREGS in EMUCONS to the DOS main-storage address specified by the STCTL instruction and continuing through as many storage words as the number of control registers specified.

Store CPU ID Subroutine (Flowchart 8S)

Eight bytes of CPU identification information are moved from a field (labeled CPUID) in EMUCONS to the DOS main-storage address specified in the STIDP instruction. The CPUID field in EMUCONS is initialized by the Emulator start I/O appendage.

Set Clock Subroutine (Flowchart 8U)

The Emulator set clock subroutine gains control when an SCK instruction (operation code is X'B204') is executed by DOS. This routine has two major functions:

- Compute and save in EMUCONS the absolute value of the difference between the OS time-of-day clock and the value addressed by the SCK instruction
- Locate and invalidate the STCK instruction (store clock) in the DOS SVC 34 routine. The main-storage address of the STCK instruction is saved in EMUCONS so that the program check (operation exception) can later be identified as the STCK instruction. No action is taken if the STCK instruction is not found.

Control is passed to entry point PC20 in IIVPCE for return to DOS.

Store Clock Subroutine (Flowchart 8U)

The Emulator store clock subroutine gains control after the program check caused by the DOS invalidated STCK instruction occurs. The address contained in the operation pointer field in the local execution list is compared with the address saved in EMUCONS by the set clock subroutine. A program check will be passed back to DOS if these addresses are not equal.

An 8-byte field in EMUCONS labeled TDELTA will contain the absolute value of the difference between the OS time-of-day value and the DOS time-of-day value. A switch labeled S370SW (in EMUCONS) will have bit 4 set to one if TDELTA is to be subtracted from the OS time-of-day value. The result is stored in DOS main storage addressed by the STCK instruction. Control is passed to entry point PC20 in IIVPCE for return to DOS.

Check I/O Routine (IIVCHK) -- Flowcharts 9A-9D

The main function of this routine is to post the DOS CSW when an I/O interruption occurs. The information posted is obtained from the IOB CSW.

The routine receives control from module IIVRTE when an outstanding posted I/O interruption is recognized. The routine is also entered by the TIO simulation subroutine (module IIVPCE) to post the DOS CSW.

The routine readjusts the CCW addresses to their local values. This is not required for operations flagged by module IIVPCE at SIO as being a stand-alone sense or seek.

For the shared DOS residence option, a DOS read to record 1 of the DOS system directory (CCHHR=X'0000000101') will cause the DOS label cylinder address to be moved from EMUCONS to the related field in the DOS input area. This is done to ensure that the proper label cylinder is being accessed.

For BTAM devices, the CCW chain issued is tested to determine if it is the chain used in polling with an autopoll wraplist. If it is, the fifth and sixth (and for start/stop devices, the seventh) CCWs are readjusted without using module IIVCCW. This is necessary since DOS BTAM modifies these CCW chains, causing the readjustment module IIVCCW to lose a path to these commands.

The routine issues a NOP operation to the device if the operation completes normally (completion code X'7F') and a channel-end only has been posted. This is performed to obtain device end conditions to be passed to DOS if the NOP is intercepted. If it is not intercepted, a CSW with only a device end condition is passed to DOS.

is performed to obtain device end conditions to be passed to DOS if the NOP is intercepted. If it is not intercepted, a CSW with only a device end condition is passed to DOS.

For all devices except DOS BTAM, if OS has posted a permanent I/O error for other than a rewind-unload operation, the routine posts a high error count to DOS to prevent DOS retries. The count is stored in the DOS PUB table. For tape, if TEBs are present, the count is stored in the TEB. If no TEBs are present, the count is stored in the TEBV if TEBVs are present.

For rewind-unload operations, two situations exist. In one case, an operator intervention is required, in which event the Emulator again issues a NOP as it does to force device end. The object of this NOP is to force OS to wait for the device to be readied, which will result in the Emulator simulating an unsolicited device end to DOS. The FPSW flag is on to indicate this situation. In the other case, intervention may not be required at all; therefore, the Emulator arbitrarily simulates an unsolicited device end to DOS. This is done as DOS has no indication of an actual unsolicited device end and may be waiting for one. The DEON flag is turned on to indicate both situations and the simulated interruption is queued in the post ECB list.

The routine transfers control to module IIVABN to post a channel end program check to DOS if module IIVCCW returns indicating an invalid CCW. This causes DOS to cancel the partition in which the error occurred.

Interpretive SYSLOG Routine (IIVLOG) -- Flowcharts 10A-10E

This routine interprets IBM 1052 Printer Keyboard CCW chains and issues WTO or WTOR as required by the channel program. All alarm commands are translated to NOP. Both data and command chaining are supported. Control is passed to this routine by the program check executive routine (IIVPCE).

Initialization

The dummy IOB for SYSLOG and the codes and flags used by the routine are initialized. The CAW verification routine (IIVAWV) and CCW verification routine (IIVCWV) are called to validate the DOS CAW and CCWs, respectively. If the CAW or a CCW is invalid and it was the first CCW of a chain, the PSW condition code is set to 01 and control is given to the exit subroutine. If the CAW and/or CCW is valid, the command code is validated and translated to an index value to be used with a branch table to go to the proper processing subroutine. The index values are:

00 - Invalid command code

04 - NOP/alarm

08 - Sense

10 - Write with automatic carriage return

14 - Write with no automatic carriage return

0C - Read

Invalid Command Code. This subroutine sets either the command reject bit in the sense byte and the unit check bit in the status byte or, if the 4 low-order bits of the command code are 0, the program check bit in the status byte.

NOP/Alarm. If the CCW is the first in a chain, the PSW condition code is set to 01. Control is given to the exit subroutine, bypassing the first CCW check in exit.

Sense. The sense byte is moved to the data area specified in the CCW and control is given to the exit subroutine.

Read. If a previous WTO is still outstanding, the WTO message is moved to the output area of the WTOR. The total reply length is obtained from the CCW and placed in the output area of the WTOR. The WTOR is then issued, the route switch is set on, and control is passed to the asynchronous interrupt exit routine (IIVRTE).

When the ECB associated with the WTOR is posted, control is passed to the remainder of the read subroutine. The reply is moved to the storage area specified by the data address in the CCW. If a continuation was requested in the reply, the data address and reply length are modified and control is passed to the read subroutine to issue another WTOR. If the reply is complete, control is passed to the exit subroutine.

Write. Tests are made to determine if data chaining or no automatic carriage return was specified in the previous CCW. If not, and the previous CCW was a write, the previously assembled WTO is issued. After the length and addresses of the current message are determined, a check is made to see if the length is greater than the maximum allowed for a WTO. If so, the message is broken into successive WTOs until the length of the remaining message is less than or equal to the maximum. One pending WTO is always left. Control is then passed to the exit subroutine.

Exit. If the CCW is the first in a chain, the PSW condition code is set to 00. The sense byte is cleared for the next command and the CCW address is increased to point to the next CCW.

- If the current chaining flags are on, control is returned to the initialization subroutine at the point where module IIVCWV is called.
- If no chaining flags are on, the device end and channel end bits in the status byte are set on.
- If a WTO is pending, the WTO is issued.
- If the route switch is on, the SYSLOG interruption pending flag is set on and control is passed to entry point IIVRTER2 in the route routine.
- If the route switch is off and the PSW condition code is 00, the SYSLOG interruption pending flag is set on.
- If the PSW condition code is 01, the DOS CSW is posted.

The routine exits to the asynchronous interrupt exit routine (IIVRTE) at entry point IIVRTER2.

Control will be returned to the IIVLOGR1 or IIVLOGR2 subroutines by the select routine if the SYSLOG interruption pending flag is on.

Dummy SYSLOG I/O Interruption

Control is passed to the IIVLOGR1 or the IIVLOGR2 subroutine by the select routine (IIVRTESL) when the SYSLOG interruption pending flag is on.

The SYSLOG interruption pending flag is set off, the DOS CSW is posted, and the DOS I/O interruption action is performed. Control is passed to the asynchronous interrupt exit routine (IIVRTER2).

Staged I/O Routine (IIVSTG) -- Flowcharts 11A-11N

Module IIVSTG contains the main logic required to stage input and output devices. During Emulator initialization, an IOB for each device defined in a DD statement is created. In the case of staged I/O, this area contains information that describes the unique characteristics of the device being staged, that is, print size, valid command codes, etc. This area also contains switches that indicate the function to be performed. (See STGCON in 'Data Areas' for format.)

Module IIVSTG receives control from module IIVPCE at DOS SIO. The contents of all registers are saved, and the channel status word in STGCON is reset to 0. STGSEN1 is moved to STGSEN0 and set to 0. STGSEN0 is given to DOS whenever a sense command is issued.

IIVAWV is called to validate the DOS CAW and initialize the CCW pointer in STGCON (STGCCW). Module IIVAWV sets the program check bit in the CSW contained in STGCON if it encountered any errors. Upon return from IIVAWV, a test is made to see if the program check bit was set. If it was, the CSW in STGCON is moved into location 64 in DOS storage, the condition code in the local execution PSW is set to CSW stored (01), and control is given to IIVRTER2.

If no errors were encountered, IIVCWV is called. This module checks the validity of the CCW addressed by STGCCW in STGCON. If the CCW is a TIC, the CCW pointer (STGCCW) is altered to point to the CCW addressed by the TIC. If any errors are encountered, the program check bit is set in the CSW contained in STGCON (STGCSW). Upon return from IIVCWV, a test is made to see if the program check bit was set. If it was, the following action is taken.

1. If the erroneous CCW is the first CCW in a chain (the CCW addressed by the CAW), the CSW is moved from STGCON to location 64 of DOS storage. The condition code in the local execution PSW is set to CSW stored (01), and control is given to IIVRTER2.

2. In all other cases, a hexadecimal 7F is moved into the high-order byte of the ECB for the device being staged. The pending interruption is later given to DOS in the same manner as for nonstaged devices. Control is given to IIVRTER2.

If no errors were discovered by IIVCWV, the data chaining bit is checked in the DOS CCW. If it is set, the program check bit is set in the CSW in STGCON and the same action is then taken as if the program check bit were set by module IIVCWV.

If the 4 low-order bits of the CCW command code are 0, the program check bit in the CSW is set to 1 and control is passed to IIVRTER2. Otherwise, the CCW operation code is translated according to the translate table whose address is contained in the STGOPCD field in STGCON. If the result of the translation is 0, the unit check bit in the CSW is set, the command reject bit in the current sense byte in STGCON (STGSEN1) is set, the ECB for the device being staged is posted, and control is given to IIVRTER2.

If the CCW command code is valid, it is checked for being a sense command. If it is a sense command, the sense information is taken from STGSEN0 and moved to the address contained in the DOS CCW plus the adjustment factor. STGSEN0 is then set to 0. If no errors were encountered up to this point, the command chaining bit is checked. If it is set, STGCCW is increased by 8 and the next CCW is processed. (If it is not set, refer to "Staged Output.")

If the command code is NOP, the command chaining bit is checked, If it is set, STGCCW is increased by 8 and the process is repeated for the next CCW.

STGMAX in STGCON contains the maximum number of bytes that can be transmitted to or from the unit-record device that is being staged. If the count field in the DOS CCW contains a value greater than the value in STGMAX, the residual count is computed and stored in the residual count field in the CSW. The incorrect length bit is also set in the CSW in STGCON if the SLI bit in the CCW being processed is not set.

Staged Input

If the first command to a staged reader is a feed-stacker select, the unusual sequence bit in STGSEN1 in STGCON is set (to simulate hardware procedures), the unit check bit is set in the status portion of the CSW in STGCON, the ECB for the device being staged is posted, and control is given to IIVRTER2. The same thing is done for any unusual sequence of commands. Successive commands specifying read without feed cause the same input record to be passed to DOS.

If EODAD (bit 5) in STGFLG is set, a /& is generated within the DOS input area and EOD is set in STGFLG to pass back a unit exception for the next input request. A GET is issued using get locate mode. A test is made for the SYNAD flag in STGFLG to see if there was a permanent error. If there was an error, the unit check bit is set in the CSW in STGCON and control is given to IIVRTER2.

If there were no errors and the suppress data transfer bit in the CCW is set, no data is moved into the DOS input area. Otherwise, the number of bytes specified in the count field of the CCW or STGMAX, whichever is less, is moved into the DOS input area. The first 2 bytes of the input data record are saved in STGDLM in STGCON. The EODAD subroutine uses this information to determine whether to generate a /*, /&, or both before sending back a unit exception. Control is then given to STGIO110 to check for command chaining.

Staged Output

If the output device being staged is a printer, the initial output request causes a skip-to-channel-1 command to be issued, followed by the DDname of the data set being written and by another skip to channel 1. If the device is a punch, the stacker bits in the command address portion of the CCW are examined to determine whether that particular stacker has been selected during this emulation run for the device being staged. If not, three separator cards that contain the DDname of the data set being staged are written before the output request is staged.

Print commands for the 3525 punch are compared with the last print command issued by DOS. Control is given to CMNDREJ if the line number indicated in the current command is not greater than the line number in the last command, otherwise, the last print command field in STGCON (labeled STGLCMD) is updated with the current print command.

The DCBLRECL field of the DCB is modified to reflect the length of the current output record. A PUT is issued and the buffer address is obtained from register 1. The SYNAD bit in STGFLG is tested to determine whether a permanent I/O error had occurred. If this bit is set, the unit check bit is set in the status portion of the CSW in STGCON and the device not operational bit (X'02') is set in the CTFLAG field of COMTAB. This, in effect, causes the DOS device to be permanently disabled for the remainder of the Emulator run. If no I/O error had occurred, the record descriptor word in the output buffer (which contains variable length records) is set with the value that was placed in DCBLRECL.

If the staged device is a printer and if module IIVPOV was included during the Emulator link-edit, a branch and link to that module is issued to simulate the carriage tape operation normally performed by the printer. This routine aids DOS printer overflow by simulating channel 9 and 12 interruptions to DOS.

When control is regained from IIVPOV, the status portion of the CSW in STGCON is tested for unusual conditions that might have occurred, such as unit check or incorrect length.

If any unusual conditions had occurred or if no command chaining is specified, the ECB for the staged device is posted and control is given to IIVRTER2. If command chaining is specified, the CCW pointer in STGCON (STGCCW) is increased by 8 and the routine starts processing the next CCW in the chain.

If the FIRSTCCW flag in STGFLG is not set after entry at CSWSTOR, a branch to INTPEND is issued. Otherwise, the staged device cuu is moved into the interruption field of the local execution PSW. The condition code in the local execution PSW is set to 1 (CSW stored). The CCW address is readjusted, increased by 8, and stored in the command address portion of the CSW in STGCON (address 64). Control is passed to the route routine (IIVRTER2).

After entry at INTPEND, the address of the last CCW processed is readjusted, increased by 8, and stored in the command address portion of the STGCON CSW. The condition code in the local execution PSW is set to 0, channel end and device end in the STGCON CSW are set, the ECB in COMTAB is posted, and control is given to the route routine (IIVRTER2).

SENSE is entered if the CCW being processed is a sense command (op code X'04'). The SYNAD bit in STGFLG is tested to determine whether a permanent input error had previously occurred. (IIVSTG does not receive control after a permanent output error.) If the SYNAD bit is set, the unit check bit in the CSW in STGCON is set to 1 and the SYNAD bit is reset to 0. This causes an 'error-on-recovery' condition when DOS regains control, and the DOS problem program that issued the erroneous read is terminated.

If there were no input errors, the residual count is computed and stored in the CSW in STGCON. The CCW data address is adjusted by the adjustment factor in the local execution list. The simulated sense byte (STGSEN0) is moved to the adjusted data address only if the suppress data-transfer bit in the CCW is set to 1. The incorrect length (ILC) bit is set in the CSW in STGCON if the SLI bit in the CCW is not set to 1 and if there is a residual count greater than 0 in the CSW. Control is then given back to the main routine to check for error conditions and CCW command chaining (see "Staged Output").

After entry at CMNDREJ, the command reject bit (bit 0) in STGSEN1 is set. The unit check bit is set to 1 in the status portion of the CSW in the dummy IOB area (STGCON). Control is passed to entry point INTPEND.

Read FCB Subroutine (Flowchart 11L)

This routine gains control at STGIO300 when a read FCB command is encountered by module IIVSTG. The data area is initialized to 0 up to the length specified in the count field of the CCW or 180 bytes, whichever is less. The Emulator forms-control image is converted to its FCB format, one line image at a time. The lines-per-inch flag is set in the first byte of the FCB if it is present in STGCHFLG in STGCON; the end-of-forms flag in the last byte of the FCB is set if the count in the CCW was equal to or greater than the length, in bytes, of the FCB image. The residual count, if it exists, is computed and placed in the CSW in STGCON, and the incorrect length bit is set if the SLI bit in the CCW is not set to 1. Control is then passed to STGIO110 to check for command chaining.

Load FCB Subroutine (Flowchart 11M)

This routine gains control at entry point STGIO200 when a load FCB command is encountered at the beginning of staged I/O or when module IIVOPN passes it control because an Emulator forms-control image must be created.

A FREEMAIN is issued to release the main-storage area where the existing image resides. A GETMAIN is then issued to obtain main storage for the new image.

The unit check bit is set in the status portion of the CSW in STGCON and a load check is set in the sense byte under the following conditions:

- The absence of the end-of-forms flag in the last position of the FCB data
- A value greater than X'0C' in any line position in the FCB data

The lines-per-inch flag, if present, is saved in STGCHFLG so that when an FCB is built by module IIVSTG when a read FCB command is issued, the FCB will be identical to that loaded previously. The STGCHFLG, STGCTP, STGLNPTR, and STGLNCNT fields in STGCON are reinitialized. Control is returned to the caller after the new Emulator forms-control image is built. See the module description for IIVPOV for further discussion of forms control.

EODAD Subroutine (Flowchart 11N)

This subroutine (entry point EODA) is given control by QSAM when an end-of-file (EOF) condition is encountered.

The STGDLM field in STGCON contains the first two characters of the input record from the last read operation. If STGDLM contains a /&, the unit exception bit in the status portion of the STGCON CSW is set. The EOD bit in STGFLG is set to 1 so that any future input request will result in a unit exception condition. The ECB for the device being staged is posted and control is given to the route routine (IIVRTER2).

If STGDLM contains a /*, a /& is generated at the address contained in the data address portion of the CCW being processed. The EOD switch in STGFLG is then set to 1 and the command chaining flag in the CCW is tested. If no command chaining was specified, the ECB is posted and control is given to IIVRTER2.

If STGDLM contains neither a /& nor a /*, the EODAD bit in STGFLG is set to 1. (This will cause a /& to be generated during the next input request.) A /* is generated at the address contained in the data address portion of the CCW being processed. If no command chaining was specified, the ECB is posted and control is given to IIVRTER2.

EODAD contains a secondary entry point (EOF010) used to blank out the DOS input area whenever a /* or a /& is to be generated.

SYNAD Subroutine (Flowchart 11N)

This subroutine (entry point SYNA) is given control by QSAM when a permanent input or output error is encountered. The SYNAD bit in STGFLG is set to 1 and control is returned to OS. The main routine, in turn, regains control from OS.

CAW Verification Routine (IIVAWV) -- Flowchart 12A

Module IIVAWV validates the DOS CAW and initializes the CCW pointer in STGCON. If the CAW is found to contain errors, the CSW program check bit is set and control is returned to the caller.

The following conditions will cause the program check bit in the CSW to be set:

- A nonzero value found in bits 4-7 of the CAW
- A command address not on a doubleword boundary
- A command address greater than the limit address in the local execution list

The following fields in STGCON may be modified by this routine:

- The status portion of the CSW maintained for this device (SCSWST2)
- The CCW pointer maintained for this device (STGCCW)
- The work byte in STGCON for this device (STGWK1)

CCW Verification Routine (IIVCWV) -- Flowchart 13A

Module IIVCWV validates and interprets DOS CCWs for IIVSTG and IIVLOG. If a CCW is found to be in error, the CSW program check bit is set and control is returned to the caller.

The following conditions will cause the program check bit in the CSW to be set:

- A nonzero value in bits 37-39 of the CCW
- A TIC as the first CCW (that is, the CCW addressed by the CAW)
- Two TICs with no intervening CCW
- A 0 in the count field of non-TIC CCWs
- A data address greater than the value in the limit address in the local execution list minus the count
- A command address in a TIC greater than the limit address in the local execution list
- A command address in a TIC not on a doubleword boundary

If the command code in the CCW being interpreted is a TIC, the CCW pointer in STGCON (STGCCW) is altered to point to the CCW addressed by the TIC.

The following fields in STGCON may be modified by this routine:

- The status portion of the CSW that is maintained for this device (SCSWST2)
- The CCW pointer that is maintained for this device (STGCCW)
- The work byte in STGCON for this device (STGWK1)

Printer Overflow Routine (IIVPOV) -- Flowchart 14A

Module IIVPOV maintains the simulated sense byte and status portion of the CSW for staged printers. IIVPOV simulates the carriage tape operation that is normally performed by the printer during output staging.

The in-storage, forms-control table defines to the Emulator the presence and position of forms-control channels for staged printers. Each entry in this table, addressed by STGFCT in STGCON, is a halfword in length. Only the low order 12 bits are utilized to represent one of 12 possible channels. Channels are ascending from left to right. The pointer is updated according to the CCW operation code. Within each entry, a 1 bit indicates a punch and a 0 bit indicates no punch.

Module IIVSTG issues as its first PUT for staged printers a skip to channel 1. In order to calibrate the forms-control pointer to the printed output, module IIVPOV receives control from IIVSTG with a dummy-skip-to-channel-1 command (only if channel 1 was defined in the FCB). The forms-control pointer is initially defined to point to the first entry in the table. The dummy skip to channel 1 causes the forms-control pointer to point to the first channel 1 entry in the table. Each DOS output request thereafter changes the value of the pointer. When the pointer reaches the end of the table, it is reinitialized and the process is repeated.

Upon entry, IIVPOV determines whether the operation code indicates a space command or skip command. If it is a space command, a branch to entry point SPACE is performed. Within SPACE, the line pointer is retrieved from STGCON and updated. If it points to the end of the table, it is reinitialized. A branch and link to entry point FETCH is performed. RW1EU contains the tape image for the current line upon return from FETCH. The image is right-adjusted.

A test for channel 12 is performed. If that bit is present, the unit exception bit is turned on in the simulated channel status word.

A test for channel 9 is performed. If that bit is present, the channel 9 indication (bit 7 in the simulated sense byte) and unit check bit in the simulated channel status word is set.

Multiple spaces are simulated one space at a time. Control is returned to retrieve the line pointer again if the command was a double or triple space. This process continues until all the spaces are simulated. Control is then returned to the main routine.

If the command code indicated a skip was to be performed, SKIP is entered. Within SKIP, the channel number is converted from its binary value to its storage format. A 12-bit constant (STGCHFLG) is tested. This constant contains a 1 bit if the corresponding channel is present in the in-storage carriage tape, and a 0 bit if not. The line pointer remains unchanged if there is a skip to a nonexistent channel. If the staged device has the FCB feature, a unit check and data check are passed back to DOS.

A branch and link to FETCH is performed. Upon return from FETCH, RW1EU contains the in-storage tape image for this line. The in-storage tape image is compared with the channel to be skipped to. Control is returned to the calling routine if the two are equal. Otherwise, the line pointer is updated and the next entry is checked. This process continues until a match is found.

Adjust CCW Data Address Routine (IIVCCW) -- Flowcharts 15A-15F

A basic knowledge of the format and function of the CCW is needed to become familiar with the operation of this routine. See System/360 Principles of Operation, GA22-6821, for this information.

The following input information must be passed to this routine by the calling routine: the address of the CCW string to be adjusted, the adjustment factor to be used, and the limit address of the emulated DOS program. This data is passed through the adjust CCW data address list. (See the Appendix for the format and function of this list.)

This routine adds an adjustment factor to the data address of each CCW in a string passed to this routine by the calling routine. When an SIO is issued by DOS with this string of CCWs, the data addresses in the CCWs will be local addresses. The adjustment factor passed to this routine will be positive. When this positive adjustment factor is added to the local data addressed in the CCWs, these addresses will be changed to true addresses that can be used by OS.

When OS finishes processing the SIO, the same string of CCWs is returned to this routine with a negative adjustment factor. When this factor is added to the CCW data addresses, the true data addresses will be changed to local data addresses that the emulated DOS program can use.

This routine uses two tables in its operation.

The first is called the beginning and end block (BEBLK) and consists of 30 8-byte entries. The first 4 bytes of an entry contain the true beginning address of the continuous group of CCWs in the CCW string being adjusted. The second 4 bytes contain the true end address of the continuous group of CCWs. The first bit of each entry is used as a completion indicator to show whether that group of CCWs has been processed. If the bit is 1, the group has been processed; if 0, it has not been processed.

When this routine encounters two different paths in the CCW string being processed, it continues processing one path and builds an entry in BEBLK indicating which path has not yet been processed. This entry is called an incomplete BEBLK entry. As each incomplete BEBLK entry is created, it is placed in BEBLK, starting at the end and working backward. The normal BEBLK entries start at the beginning of BEBLK and work forward. In the incomplete entry, the first 4 bytes are the TIC address, the fifth byte is the operation byte, and the last 3 bytes are the operation pointer. All of these values are taken from the adjust CCW data address list.

The other table used by this routine is called the status modifier table. This is a 256-byte translate and test table that contains the CCW command codes for the devices supported by the Emulator that cause the channel to return a status modifier condition when a CCW's condition is met.

Because it is possible to add the adjustment factor to a single CCW more than once, this routine uses a two-step method to adjust the data addresses in the CCW string. It is possible to add the adjustment factor more than once because the CCW string uses a CCW more than once in performing its operation.

During the first step, this routine adjusts the data address portion of each CCW in a string of CCWs with an adjustment factor of zero. This is done so that BEBLK entries can be created for each contiguous group of CCWs in the CCW string. As each BEBLK entry is created, this routine will branch to the combine subroutine to have duplicate groups of CCWs eliminated from BEBLK.

After the entire groups of CCW string has been processed and all duplicate groups of CCWs have been eliminated from BEBLK, the second step of this routine is performed. The data addresses of the CCWs defined by each entry in the BEBLK table are adjusted. The CCWs within each group that is represented by a BEBLK entry have their data addresses adjusted by an adjustment factor that is passed by the calling routine. Therefore, if this adjustment factor is positive, the CCW data addresses are changed from local to true addresses; if the adjustment factor is negative, the CCW data addresses are changed from true to local addresses.

If any addressability, protection, or specification errors are detected by the adjust CCW string instruction or BEBLK is filled, this routine returns control to the calling routine's return point.

If the byte count of a CCW in the CCW string being processed by this routine is added to the true address created for that CCW by this routine and the resultant address exceeds the emulated DOS program's limit address (passed to this routine by the calling routine), the SLI bit in this CCW is tested. If the SLI bit is on, this routine assumes that the storage area addressed by this CCW will not exceed the emulated DOS program's storage area and continues processing this CCW string.

Upon normal completion of this routine's operation, control is returned to the address 4 bytes beyond the calling routine's return point.

This routine sets the ABEND interception switch before adjusting CCWs. It then resets the switch after the adjustment is complete. If the ABEND interception is taken, the caller's registers are restored and control is passed to IIVAB4. At this time an ABEND error code of 16 is set in register 1 and control is returned to the caller.

Combine Subroutine (Flowchart 15F)

This routine eliminates any duplicate CCWs from the BEBLK table by combining an entry with a beginning and/or ending address that falls within the beginning and/or ending address of another entry, into the same entry. The routine also tests the last CCW of the current BEBLK entry to see if it is a TIC. If it is a TIC, this routine tests all BEBLK entries to see if the TIC command address is in BEBLK.

If this routine is able to combine the current BEBLK entry into any of the other entries, it turns the combine switch on to indicate to the calling routine that it has done so.

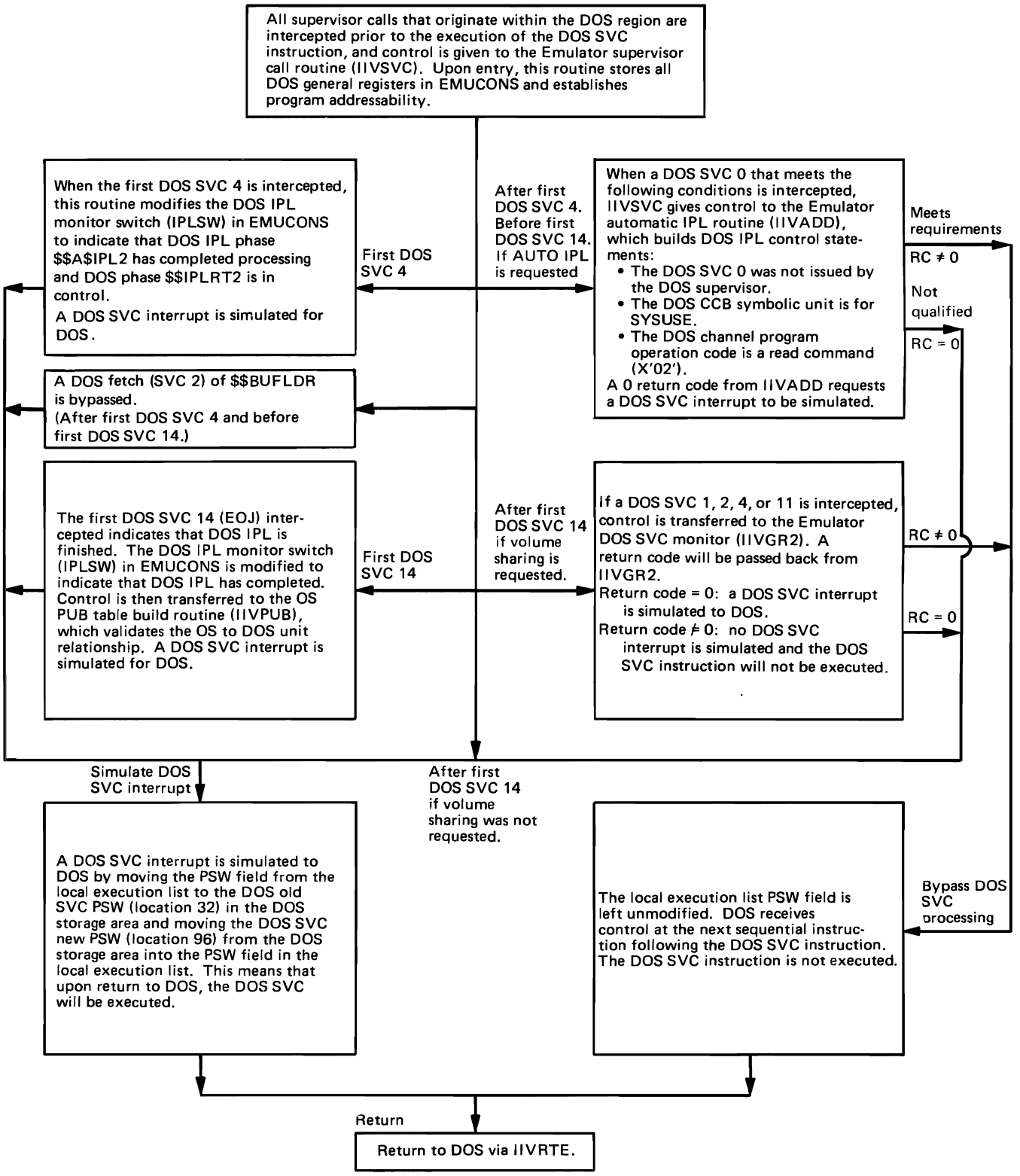
If this routine found the TIC command address in BEBLK, it turns the TIC-TO switch on to indicate to the calling routine that the TIC command address has already been processed by the adjust CCW data address routine.

If this routine is able to combine the current BEBLK entry into BEBLK, the CCW string being adjusted has looped into an area of itself that has already been processed. If this occurs and there are no incomplete paths of the CCW string to be processed, the theoretical end of the CCW string being adjusted has been reached.

CCW Adjustment Routine (IIVADJ) -- Flowcharts 16A-16B

This routine gains control from the IIVCCW or IIVRCW module and returns control to the IIVCCW or IIVRCW module upon completion of its function. Refer to the Appendix (Adjust CCW String Instruction) for a description of the function of this module.

Supervisor Call Routine (IIVSVC) -- Flowchart 17A



Start I/O, End-of-Extent, Channel End, and Abnormal End Appendage (IGG019SA) -- Flowcharts 18A-18B

The appendages are used to maintain (modify/restore) the DEBs for DASD and tape devices used by DOS. The DEB is built during open in protected storage (key of zero); it contains the file mask and extent information for direct-access devices and the set mode command for tape devices that are maintained by these appendages.

When the DEB is built by open, the initial entry address is in the DEB appendage address table for the start I/O and end-of-extent appendages if the device is direct access, and for the start I/O appendage if the device is tape. The initial entry modifies the addresses in the DEB appendage address table to point to the respective routines within the module.

The CPU ID is stored when this appendage is entered for the first time. CPU ID information will then be obtained from EMUCONS when the STIDP instruction is issued by DOS.

The addresses modified in a DEB are:

DASD DEB:

- End-of-extent address - EOERT000
- Start I/O address - SIORT000
- Channel end address - CERT000
- Abnormal end address - AERT000

TAPE DEB:

- Start I/O address - SIORT500

After initializing the DEB appendage address table, control is passed to the appropriate routine.

End-of-Extent Subroutine (EOERT000 - Flowchart 18A)

If the user label flag is on in the COMTAB (CTFLAG3 bit 0), the extent information in the DEB is saved and replaced with the user label extent from the COMTAB. Return is made to OS for a retry of the I/O request. Otherwise, the normal return is made to OS, which results in an out-of-extent condition to be posted in the event control block.

Start I/O (Tape) Subroutine (SIORT500) - Flowchart 18A)

The DOS set mode command is moved from the COMTAB to the DEB.

Start I/O (DASD) Subroutine (SIORT000 - Flowchart 18B)

The extent information in the DEB is saved and replaced with an extent limit of one cylinder (based on the cylinder in the IOB seek address) so that OS cylinder switching at end of extent will be inhibited. If the shared volume flag (CTFLAG3 bit 4) is not on, the DOS file mask is set in the DEB. Return to OS is always normal.

Abnormal End (DASD) Subroutine (AERT000 - Flowchart 18B)

The event control block in the IOB is tested to see if OS error recovery is completed. If not, the normal return is made so OS error recovery will be performed. Otherwise, control is passed to the restore subroutine (RSTOR000) to restore the extent in the DEB.

Channel End (DASD) Subroutine (CERT000 - Flowchart 18B)

The condition code is tested to see if an error condition has been detected at channel end. If an error condition is present, the normal return is made so OS error recovery will be performed. Otherwise, control is passed to the restore subroutine (RSTOR000) to restore the extent in the DEB.

Restore DEB Extent Subroutine (RSTOR000 - Flowchart 18B)

The extent saved by SIORT000 or EOERT000 is moved into the DEB. The return to OS is always normal.

Abnormal End/Channel End Appendage (IGG019S1) -- Flowchart 19A

This Emulator module does double duty as an abnormal end appendage for teleprocessing devices and a channel end and abnormal end appendage for tape and unit-record devices.

The abnormal end appendage bypasses OS error recovery procedures for teleprocessing I/O errors. When a magnetic tape, unit record, or teleprocessing (BTAM) error is detected, control is given to the appropriate section of the appendage. The appendage turns off the IOB exception bit and returns to the I/O supervisor (0 displacement from the address in register 14) to bypass the OS error recovery procedures. If the appendage receives control for an error other than those mentioned above, it immediately returns to the I/O supervisor, allowing the OS error recovery procedures to handle the error processing. The abnormal end appendage also routes error conditions for Emulator-initiated I/O (as opposed to DOS-initiated I/O) to the OS error recovery procedures.

In the role of channel end appendage, this module routes all incorrect length and unit exception conditions for tape and unit record devices at channel end to DOS. (DOS does not consider incorrect length a permanent error condition.) It does this by turning off the exception bit in the IOB and returning to the OS I/O supervisor, which then posts the I/O operation complete.

After a tape rewind-unload has been issued or a BTAM intervention required condition occurs, the not ready bit is turned on in the UCB and a not ready flag is set in the COMTAB entry. For unit-record devices, if a unit exception occurs, the IOB esception flag is turned off, suppressing OS error recovery procedures.

Asynchronous Interrupt Exit Routine (IIVRTE) -- Flowcharts 20A-20G

When the OS supervisor has completed processing any asynchronous interruption that occurred while DOS was operating in local execution mode, it gives control to this routine at entry point IIVRTE. When this routine finishes processing, it returns control to DOS by issuing the execute local instruction which returns the CPU to local execution mode.

If the one-second STIMER interval has expired, this routine branches to the timer interrupt check routine to process the interruption. If the interval has not expired, the routine branches to the asynchronous interrupt check routine to see if any of the Emulator ECBs pointed to by the ECB pointer table

are posted. If one or more of the Emulator ECBs are posted, this routine branches to the select routine, which passes control to the appropriate Emulator routine to process the interruption.

If the interruption was not for the Emulator, DOS is checked to see if it is in the wait state. If DOS is in the wait state and interruptions are disabled (hard wait), control is given to the ABEND error routine (IIVABN) to cancel the Emulator job. If DOS is in the wait state and the AUTOEOJ option has been specified, control is passed to the end of job routine to see if DOS has any more work to do.

If DOS is in the wait state, this routine goes into the wait state until completion of an outstanding I/O operation. When this occurs, control is given to the appropriate Emulator routine to simulate completion of the I/O operation to DOS.

If DOS is not in the wait state and the timer option has been specified, the fourth byte (local address X'53') of the emulated DOS timer is increased by 1. When this byte reaches a value of 256, it is reset to 0. The purpose of this operation is to show DOS a constantly changing timer value when the timer is tested to see if it is operational. This has no real effect on the actual time.

After the DOS interval timer has been updated, or if the timer option was not specified, control is given to DOS through the execute local instruction, thus placing the CPU in local execution mode. For an explanation of the operation of this instruction, see the "DOS Compatibility Feature" in the Appendix.

STAE Exit Routine (Flowchart 20A)

This routine gains control for a scheduled OS ABEND. The primary function of this routine is to determine whether further ABEND processing is to be allowed or bypassed. ABEND processing is bypassed only if the bypass ABEND switch (OPTFLGR2 in IIVCON=X'02') is on or a //SYSABEND DD statement is present in the OS jobstream. The switch is set on by IIVOPN before opening a nonshared volume and by IIVCCW before CCW address adjustment takes place. The STAE retry routine is scheduled to receive control and the routine returns to the OS ABEND processing routines.

Route Routine (Flowchart 20B)

The entry point for this routine is IIVRTER2. Following completion of their operations, the interpretive SYSLOG (IIVLOG), staged I/O (IIVSTG), asynchronous interrupt exit (IIVRTE), and program check executive (IIVPCE) routines return to this routine at entry point IIVRTER2 to return control to DOS. The LPSW simulation, SSM simulation, ISK simulation, SSK simulation, TIO simulation, TCH simulation, SIO, and Check I/O routines return to the IIVRTER2 entry point by means of the program check executive routine. Module IIVRAS also exits to IIVRTER2.

At entry point IIVRTER2, this routine tests to see if any more interruptions can be processed before control is given to DOS. If any further interruptions can be processed, control is given to the appropriate Emulator routine. If not, control is given to DOS through the execute local instruction.

Select Routine (Flowchart 20C)

This routine routes control to the appropriate Emulator routine that handles pending asynchronous interruptions of the emulated DOS program.

The routine first tests to see if the system mask portion of the local execution PSW is enabled for external interruptions. If it is, a test is then made to see if a timer interruption is pending. If a timer interruption is pending, control is passed to the timer interrupt check routine.

The routine next tests to see if there is an interruption for the Emulator prompt WTOR. If there is, control is passed to the prompt reply processor routine (IIVPRP).

If DOS was disabled for external interruptions, or none of the above interruptions was pending, the system mask portion of the local execution PSW is tested to see if I/O interruptions are enabled. If the mask is enabled for I/O interruptions, a test is then made to see if the operator has responded to the last WTOR issued by the interpretive SYSLOG routine. If he has responded, control is passed to the interpretive SYSLOG routine. If he has not responded, this routine tests to see if the interpretive SYSLOG routine had issued a WTO to the operator and is waiting for DOS to enable for I/O interruptions. If so, control is passed to the interpretive SYSLOG routine.

A test is then made to see if there are any I/O interruptions to be handled for the emulated DOS program. If there are, control is passed to the check I/O routine. If there were no asynchronous interruptions pending, control is passed to the route routine at entry point IIVRTER2.

STAE Retry Routine (Flowchart 20C)

This routine gains control from the OS ABEND processing routines when the STAE exit routine determines that a bypass is to be affected. The STAE retry routine reissues the OS STAE macro to reactivate the ABEND interception procedure. Since any outstanding WTORs are canceled by the OS ABEND routines before passing control to the STAE retry routine, the WTORs are reestablished. Control is passed to the program setting the bypass switch to an address stored in STAERTN (IIVCON) by the program. Each module that sets the bypass ABEND switch and the action taken by the module is as follows:

Modules Requesting ABEND Interception

Module name	Function being performed when requesting interception	Action taken upon regaining control
IIVCCW IIVRCW	Adjusting CCW data addresses	Channel program check returned to DOS
IIVDVS	Opening a file	Cancel DOS job; issue message IIV256I
	Reading the VTOC using an OS OBTAIN macro	Cancel DOS job; issue message IIV261I
	Issuing an OS EOV macro to get an additional extent	Cancel DOS job; issue message IIV260I
IIVOPN	Opening a dedicated device for SIO processing	A 'device not operational' indication is returned to DOS; message IIV018I is issued

Timer Interrupt Check Subroutine (Flowchart 20D)

This subroutine is entered from the select routine (IIVRTESL) to check for a DOS timer external interruption. The timer pending switch is set off and the DOS timer is tested for a value greater than 0. If the value is greater than 0, the time indicator switch is set on.

The DOS timer is then decreased by 1 second and again tested for a value greater than 0. If the value is less than 0 and the time indicator switch is on, the time indicator switch is set off, the interruption code in the local execution PSW is set to indicate a timer external interruption, the local execution PSW is moved to the DOS external old PSW, and the DOS external new PSW is moved to the local execution PSW.

An OS STIMER for 1 second is issued, and control is given to the route routine at entry point IIVRTER2.

Timer Interrupt Subroutine (Flowchart 20D)

This routine is the STIMER completion exit routine for the STIMER macro issued by the Emulator. It gains control from OS when the interval specified by the STIMER macro instruction, issued in the timer interrupt check routine, has expired.

The timer interrupt pending switch is set to 1 to indicate that the STIMER interval has expired. The DRI switch is turned off, and the timer ECB is posted so that if the Emulator was in the wait state, it receives control from OS to indicate expiration of the STIMER interval.

The timer ECB is one of the ECBs pointed to by the ECB pointer table and thus is one of the ECBs waited on by the asynchronous interrupt exit routine when it enters the wait state. After the timer ECB is posted, this routine passes control back to the OS control program that called it.

Asynchronous Interrupt Check Subroutine (Flowcharts 20E-20F)

This routine tests all Emulator ECBs to see if any of them have been posted by OS as complete. As each posted ECB is found, this routine sets a switch on or increases a counter to indicate to other Emulator routines that an asynchronous interruption is pending and must be processed.

Except for SYSLOG, a 2-byte entry is created in the post ECB list when an ECB for an I/O device has been found posted. The first byte contains the offset value of the I/O device in COMTAB. The second byte contains the status byte of the posted ECB. After all Emulator ECBs have been tested, this routine returns control to the calling routine.

End-of-Job Routine (Flowchart 20G)

This routine gains control from IIVRTE when DOS tries to load a wait PSW with interruptions enabled and the end-of-job switch (EOJSW) is set to 1. It also gains control from the prompt reply processor (IIVPRP) at the time the operator command EOJ is received. If it is determined that all DOS processing is complete, the Emulator is terminated.

The following conditions are checked to determine if all DOS processing is complete:

- The number of outstanding I/O operations must be equal to zero.
- The first byte of the background PIB must contain X'82' to indicate that the DOS background partition is waiting for an interruption to restart.
- In a multiprogramming system, both foreground PIBs must contain X'80' to indicate that they are both active.
- The DOS SYSRDR LUB must be valid.
- The IOB CSW unit exception bit for the background SYSRDR device must be 1 to indicate end-of-file on the device.
- The background SYSRDR device IOB address is found by mapping the SYSRDR LUB index for the DOS PUB associated with it to the COMTAB entry for the device. This mapping is done through the OS PUB table.

If all of these conditions are met, emulation is ended by a return to OS by means of the Exit ABEND error routine (IIVABN).

Prompt Reply Processor Routine (IIVPRP) -- Flowcharts 21A-21E

This module receives control from the select routine (IIVRTESL) when the console operator replies to the operator prompt message issued by the Emulator. The following Emulator commands are recognized:

EOJ — The operator wishes to terminate the Emulator region after all DOS partitions become inactive. The end-of-job switch (EOJSW) is set to notify the end-of-job routine that emulation may be ended when all DOS partitions are inactive.

EXT — A DOS external interruption is desired. The interruption code is set, and a DOS external interruption is simulated.

ATTN — The DOS attention routines are desired. The interruption code is set, and a DOS I/O interruption is simulated.

MAPIO — The operator requires the corresponding channel and unit numbers of the DOS and OS I/O units. The operand is tested to determine whether the request is for all or selected DOS cuu's, and messages to the operator are constructed accordingly.

MOUNT — The operator wishes to mount a new DASD volume. The operand is tested for the DOS cuu and new volume serial number. An OS CLOSE macro for the DCB that corresponds to the DOS cuu is issued, the new volume serial number is inserted in the JFCB, and an OPENJ macro is issued.

DEBUG — The operator wishes to snap dump Emulator control blocks when certain events occur in DOS. See "Emulator Service Aids" for information about the modules that perform trace and snap functions.

All commands and their operands, if any, are verified and appropriate error messages are issued if errors are detected. The error message becomes the text of the subsequent prompt. See Emulating DOS Under OS on IBM System/370, GC26-3777, for more information about prompts.

Control is always returned to the select routine (IIVRTESL).

SVC Monitor Routine (IIVGR2) -- Flowcharts 22A-22D

This module monitors SVC 1, 2, 4, and 11 calls from DOS, recognizes OS sequential DASD and direct-access shared data sets, DOS sequential DASD and direct-access shared files, or OS indexed sequential shared data sets, and passes control to the volume-sharing simulation routine (IIVDVS) or the ISAM mapping routine (IIVIS), as the case may be. IIVGR2 is entered from module IIVSVC when a DOS SVC is trapped and sequential DASD or direct-access shared or OS indexed sequential shared data sets are specified by the DD statement.

If the shared DOS system residence option is in effect or a shared DOS PCIL is being used, and the SVC number is 1 or 4, the phase name is checked for a call from MAINT, $LNKEDT, or CORGZ. If equal, an * is placed in the last position of the phase name to cause DOS to issue the 'phase not found' message OS06I. An Emulator message, IIV208A, is also issued. This will protect the core image library from being accidentally updated.

The routine calls IIVDVS when an OPEN or CLOSE macro or an end-of-extent condition is identified for a sequential DASD or direct-access shared data set (file). It calls IIVIS when an OPEN, CLOSE, SETL, SETFL, or ENDFL macro is identified for an OS indexed sequential shared data set.

IIVGR2 uses a five-character table to identify the B-transient phase being called by DOS, after which the data set must be identified as sequential DASD or direct-access shared, OS indexed sequential shared, or DOS dedicated. The two phases are:

Open Phase

DOS provides a 'file ID' field in the DLBL/EXTENT image, and a SYS number in the DTF or DLBL/EXTENT image.

By using the SYS number and the DOS cuu pointed to by the LUB, the routine identifies the corresponding COMTAB entry. This entry indicates either a DOS dedicated file, in which case control returns to IIVSVC to let DOS process the OPEN macro, or a sequential DASD or direct-access shared or OS indexed sequential shared data set (file), in which case the COMTAB extension is searched for a match between the DOS program DTF name and the DTF name in the FID.

If a match is not found, the message MISSING DD STATEMENT is issued and control is given to DOS, which issues an SVC 6 (cancel). If a match is found, the routine sets the code, the COMTAB extension address, the DTF address, and the LTK in the parameter list located in EMUCONS before calling IIVDVS or IIVIS to perform the open.

Nonopen Phase

The routine searches a FID for a match between the DTF names. If a match is not found, it returns to IIVSVC to let DOS execute the phase. If a match is found, it sets the DTF and FID addresses in the parameter list located in EMUCONS before calling IIVDVS or IIVIS as before.

After calling IIVDVS or IIVIS, the routine checks register 15 for an error code. If there is one, it issues an SVC 6 and returns to module IIVSVC. If there is not an error code, it determines from the NXTBTR field in EMUCONS which B-transient phase has to be executed. If none is specified, the routine returns to module IIVSVC, which gives control to DOS by means of module IIVRTE to execute the instruction following the SVC instruction. If a B-transient

name is given, the routine moves it in place of the existing one and returns to module IIVSVC, which then returns control to DOS to execute the SVC 2 instruction.

To sum up, IIVGR2 returns to IIVSVC (and later to DOS) to execute:

- the SVC 2, 4, or 11 when a sequential DASD or direct-access shared or OS indexed sequential shared data set (file) is not being processed
- or the instruction following the SVC instruction when the supervisor call has been simulated by the Emulator

In case of an error, the routine returns to module IIVSVC in order to execute a DOS SVC 6, which cancels the DOS step.

Device Sharing Simulation Routine (IIVDVS) -- Flowcharts 23A-23H

The routines in this module move the extent limits from OS (DEB) to DOS (DLBL/EXTENT card image) so that DOS will use the actual extent limits of the file as determined from OS JCL.

In addition, the following control blocks are created or updated to reflect the fact that an OS file is being processed:

- DCB - created and opened when DOS requests that a file be opened; closed when DOS requests that a file be closed or at the end of a DOS job step (except for DOS system files)
- DTF - maintained to reflect the status of the file as determined by OS
- Open table - updated to reflect the actual seek addresses in the VTOC and some specific indicators (see open table below)
- COMTAB extension and FID - maintained for the Emulator to identify sequential DASD and direct-access shared data sets (files)
- EMUCONS - maintained as a communication area between IIVDVS, IIVGR2, and IIVPCE
- Obtain area - created and maintained for IIVVIO to return VOL1, identifier and extension DSCBs to DOS

Processing of a sequential DASD or direct-access shared data set (file) is divided into seven routines, all included in IIVDVS (Figure 18).

IIVDVS expects the following control blocks and registers to be set by DOS open phases:

- Register 7 points to DOS communication region.
- Register 2 points to the current DTF.
- Open table indicators (X'4A3' of B-transient phase) set for special open.
- Address of DLBL (X'4A8' of B-transient phase).
- The image of the first DLBL/EXTENT statement should have been read into storage (open/end of extent only) at address X'378' of B-transient phase (for sequential disk) or in label area (direct access).

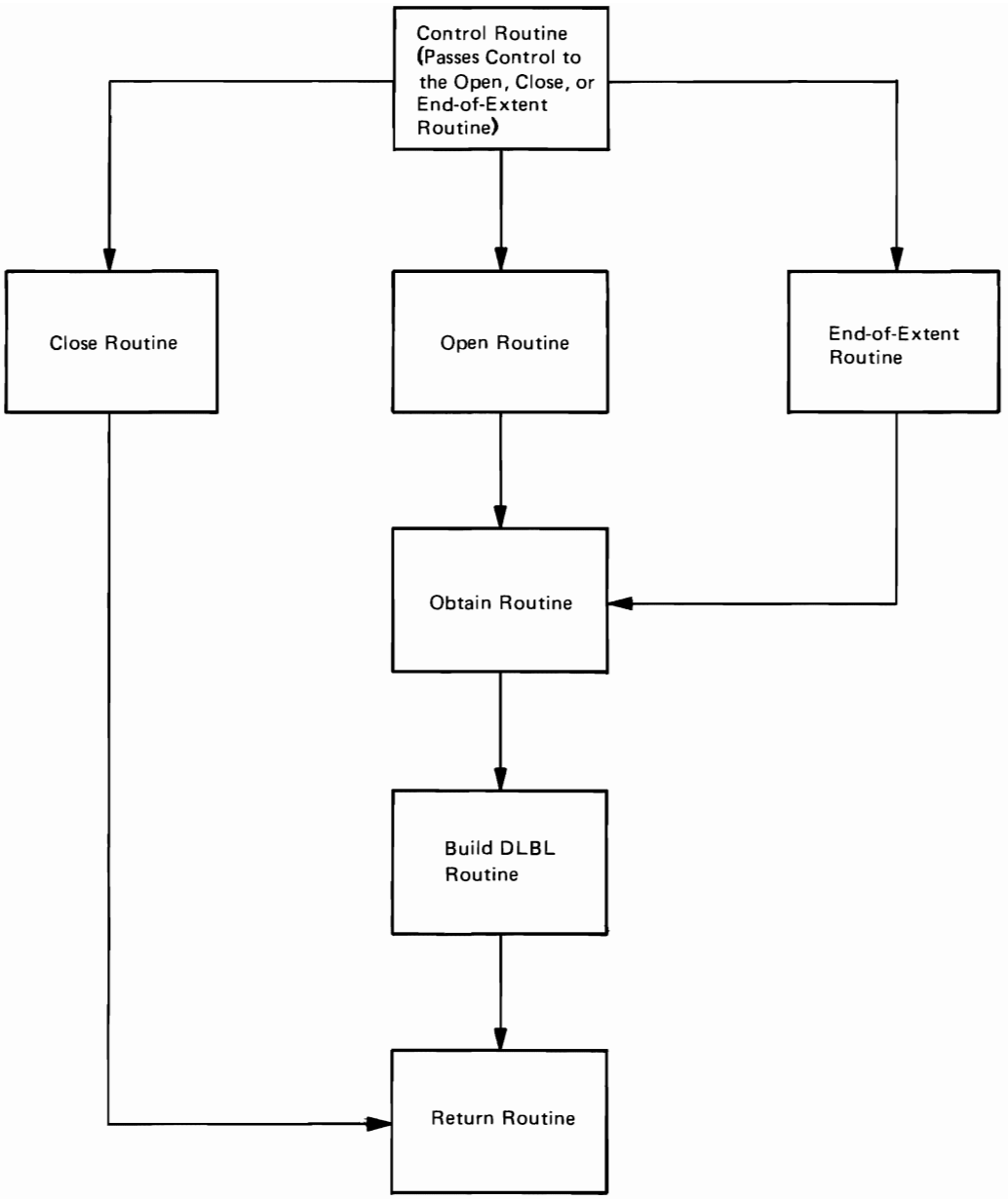


Figure 18. General Flow of IIVDVS

IIVDVS updates the following control blocks and DOS registers to reflect the fact that some DOS open phases have been bypassed:

DOS register

- Register 6 is set to point to the DLBL/EXTENT card image.

Open table

- Open table indicator (X'4A3' of B-transient phase) is set to show special open for SYSLNK.
- Message indicator (X'6A1' of B-transient phase) is set to show the type of DTF being processed.

- VTOC in-storage indicator (X'4A0' of B-transient phase) is set to show that the VTOC has been read into.
- VTOC lower and upper limit (X'3E0' of B-transient phase) are set with the identifier (format 1) DSCB address (sequential disk only).
- Seek bucket (X'3FC' of B-transient phase) will contain the address of identifier (format 1) DSCB in VTOC.
- Logical unit in VTOC CCB (X'3E8' of B-transient phase) will contain the logical unit of the file (input sequential only).
- Search argument (X'409') will contain the file ID of the file (input sequential only).

DTF

- Open communications byte in DTF will reflect current status.
- Volume sequence number in DTF will contain X'0001' when opening a sequential output file.
- DTF flag 1 (X'15' of DTF) indicators such as DELETFL=NO will be turned on for work files.
- VTOC address (X'020' of DTF) will contain the identifier (format 1) DSCB address in VTOC for work file.
- User label address (X'05C' of DTF) will contain the disk address of user label extent for DTFDA.

DLBL

- User label seek address (X'30' of DTF) will contain the disk address of user label when $$BOSDO6 is called.
- Search argument (X'3C') and LIOCS search argument (X'4C') will contain the actual disk address of the first extent when $$BOSDO6 is called.
- DLBL/EXTENT card image will contain the extent limits, extent type, extent sequence number, logical unit, volume serial number, expiration and creation dates, and DLBL indicator according to the OS information about the file.

Control Routine (Flowchart 23A)

This routine gains control from IIVGR2 and examines a parameter list (PARMLST) located in EMUCONS. PARMLST contains the address of the DTF to be processed (PARMDTFA), the key of the partition owning the DTF (PARMLTK), a code indicating the operation to be performed (PARMCODE), and (in PARMFDCS) a pointer to the associated COMTAB extension for an open function or a pointer to the associated FID for an end of extent or close function.

The routine sets standard linkage conventions (save area and registers), and then initializes registers to point to COMTAB extension, FID, DTF, and DLBL.

According to PARMCODE, the routine exits to the open, end-of-extent, or close routines in this module.

Open Routine (Flowcharts 23A-23B)

This routine is entered from the control routine when PARMCODE indicates an open function is to be performed. EMUCONS should contain the following DOS phase names in the fields indicated:

Type	OLDBTR	NXTBTR	NXTBTR after Open routine
Sequential output	$$BOSD01	$$BOSDO1	$$BOSDO1
Sequential input	$$BOSD01	$$BOSDI1	$$BOSDI1
Sequential work	$$BOSD01	$$BOSDW1	$$BOSDW2
Direct access	$$BOPEN2	$$BODAIN	$$BODAIN

The routine obtains storage for a new FID by means of module IIVGET, attaches this FID to the chain of FIDs pointed to by DSFIDBLK in EMUCONS, and sets in it the COMTAB extension pointer (FIDCTXTN), the DTF name (FIDTFNME), and the key (FIDLTK).

If the file is not yet opened (CTDCBUC=0), the routine obtains storage for a DCB by calling IIVGET and initializes the DCB with the following:

- DSORG=PS
- MACRF=(E)
- DEVD=DA
- Appendages (same as for dedicated volumes)
- DDname from CTDDNAME
- Exit list to point to an inactive exit list

An OS OPEN for output is issued for the data set associated with this DCB, and the DCB is checked for a successful open. If an error occurs, message CANNOT OPEN is transmitted and the routine branches to the return routine. If no error occurs during the open, CTDCBPTR is set in the COMTAB extension, the track balance (DCBTRKBAL) is set to 0, and the address of the last record (DCBFDAD) is set to point to the last track of the last extent so that OS will not try to write an EOF record at close.

The use count for this file (contained in the CTDCBUC field of the COMTAB extension) is increased by one. The device types, as specified in the DOS DTF and OS DCB, are checked for identity. If not identical, the message NO MATCHING DEVICE TYPE is issued and control is given to the return routine.

For direct-access files, control is given to the obtain routine. For sequential disk files, register 6 is set to point to the DLBL/EXTENT card image.

Depending upon the type of file as indicated by the name of the DOS phase to be fetched, the following control blocks are updated:

sequential input (DOS calling $$BOSDI1)

- The volume sequence number in the DTF is set to 1.
- The LIOCS switches in the DTF are reset for DTFSD.
- The message code in the open table is set to 3.

sequential output (DOS calling $$BOSDO1)

- The new volume switch is reset in the DTF.
- The open bit in the DTF is reset for a system file.
- The volume and extent sequence numbers are set to their values in the DLBL/EXTENT card image.
- The message code in the open table is set to 4.

sequential work file (DOS calling $$BOSDW1)

- Indicators 'DELETFL=NO', 'SYSxxx in DTF', and 'extent open' are set in the DTF.
- The message code in the open table is set to 9.
- The NXTBTR in EMUCONS is set to $$BOSDW2.

Control is then given to the obtain routine.

End-of-Extent Routine (Flowcharts 23C-23D)

This routine is entered from the control routine when PARMCODE specifies EOX. The end-of-extent routine's main purpose is to give an additional extent to DOS for sequential input or output files.

However, the meaning of EOX has been extended to cover all cases where DOS calls the B-transient phases for a file that is already opened, except for close conditions, which are processed in the close routine. These cases can be divided into four main categories as fcllows:

Obtain Only Cases. The DOS B-transient phase to be executed next will access the VTOC.

Type	OLDBTR	NXTBTR
Any	Any	$$BOMSG1
Direct access input, user label	$$BODAI1	$$BODAU1
Direct access user label	$$BOFLPT	$$BODAU1
Sequential disk output, trailer label	$$BOSDC1	$$BOSDO6
Sequential disk input, trailer label	$$BOSDC1	$$BOSDI3
Sequential disk input header label	$$BOSDI1	$$BOSDI3
Direct access input	$$BODAIN	$$BODAI1

The end-of-extent routine will indicate that an obtain is to be performed and branch to the obtain routine. Control eventually returns to DOS, and the access to the VTOC will be routed and handled by module IIVVIO.

User Labels Cases. The DOS B-transient phases to be executed next will create the user label track.

Type	OLDBTR	NXTBTR before	NXTBTR after
Sequential disk output	$$BOSDO4 $$BOSDO9	$$BOSDO6 $$BOSDO6	$$BOSDO6 $$BOSDO6

For sequential disk output files with user labels (when running with DOS release 26), the routine sets the appropriate user label track address and data track address in the DTF prior to branching to the obtain routine. When running DOS release 27, the correct user label and data track address in the DTF have already been set by the obtain routine.

End-of-Extent Cases. The DOS problem program calls the B-transient phases in order to open (or, rather get the next extent for the file.

Type	OLDBTR	NXTBTR before	NXTBTR after
Sequential disk output	$$BOSDC1	$$BOSDO1	$$BOSDO5
Sequential disk output	$$BOSD01	$$BOSDO1	$$BOSDO5
Sequential disk input	$$BOSD01	$$BOSDI1	$$BOSDI2
Sequential disk work file	$$BOSD01	$$BOSDW3	$$BOSDW3
Sequential disk work file	$$BOSDC1	$$BOSDW3	$$BOSDW3

The routine checks whether the required extent is present in the DEB; if it is, the routine branches to the obtain routine, which updates the DTF and branches to the build DLBL routine to create a DLBL/EXTENT card image reflecting the extent.

If the required extent is not present in the DEB, the routine issues a message and eventually the DOS job step is canceled; for an output file only, the routine issues an OS EOV to get secondary allocation from OS if any was specified in the OS DD statement.

Other Cases

Type	OLDBTR	NXTBTR before	NXTBTR after
Work file	$$BOSDW2 $$BOFLPT	$$BOSDW1 $$BOSDW1	$$BOSDW2 $$BOSDW2
Direct access output	$$BODAIN	$$BODAO1	XXXXXXXX

where XXXXXXXX = $$BODAUI or $$BOFLPT or $$BOPEN.

$$BOSDW1 is called to get the next DLBL/EXTENT statement, if any. This DOS phase will be bypassed by moving $$BOSDW2 in NXTBTR, except when the routine reaches the last extent as indicated in the DEB, in which case NXTBTR will not be updated and $$BOSDW1 will complete the DOS open processing for the work file. For direct access output, NXTBTR is updated as described above and the routine branches to the return routine.

Other special cases are:

- Call for B-transient phase for sequential input at EOF ($$BOSDI1 is called by $$BOSDO1). The end-of-extent routine will then let DOS execute $$BOSDI1 by branching to the return routine.

- Call for a B-transient phase for sequential output at the end-of-extent time ($$BOSDO1 is called by $$BOSD01) whenever the end-of-extent address is specified in the DTF. The end-of-extent routine will then let DOS execute $$BOSDO1 by branching to the return routine.

- Call for a B-transient phase for sequential work file when a DOS POINT macro is issued ($$BOSDW3 is called by $$BOSD01). The end-of-extent routine will first check the validity of the given seek address prior to branching to the current work file end-of-extent case and issue a message if the seek address does not fit into any extent in the DEB.

Close Routine (Flowchart 23E)

This routine is entered from the control routine, and the following DOS program names must be set in the indicated fields of EMUCONS:

Type	OLDBTR field	NXTBTR field
Sequential disk	$$BCLOSE	$$BOSDC1
Sequential disk	$$BOSDC1	$$BCLOSE
Sequential disk	$$BOSDC2	$$BCLOSE
Sequential output user label	$$BOSDI3	$$BCLOSE
Sequential input user label	$$BOSDO6	$$BCLOSE
Sequential disk	$$BOSDC1	PRPGM
Sequential disk	$$BOSDC2	PRPGM
Sequential input user label	$$BOSDI3	PRPGM
Sequential output user label	$$BOSDO6	PRPGM
End of job	--	$$BEOJ
Direct access	$$BCLOSE	$$BODACL

When OLDBTR contains $$BCLOSE, this routine branches directly to the obtain routine in order to build VOL1 (volume 1), identifier (format 1) and extension (format 3) DSCB images, so that the next DOS phases can read from the VTOC if needed.

When NXTBTR contains PRPGM, which indicates that DOS is issuing an SVC 11 instruction, the routine identifies a time close condition ($$BOSDI3 or $$BOSDO6 issuing an SVC 11 may be for header labels at open). When a time close condition is recognized, the use count (CTDCBUC) is decreased and tested for a 0. If the use count is not 0, control is given to the return routine to free the current FID and return to module IIVGR2. If the use count is 0, the routine issues an OS CLOSE for the file, frees the storage used by the DCB, resets the pointer to the DCB in the CTDCBPTR field of EMUCONS, and branches to the return routine to free the current FID.

Obtain Routine (Flowcharts 23F-23G)

This routine is entered from the open, end-of-extent, or close routines. It initializes a constant area (IIVOBTE1), which is pointed to by AIIVOBE1 in EMUCONS, with all of the VTOC information needed by DOS. The following fields in the obtain work area are initialized:

OBVOL1	21 characters of a VOL1 label
OBF1LBL	identifier (format 1) DSCB for the data set
OBF3LBL	extension (format 3) DSCB for the data set, if needed
OB1COUNT	format 1 count information
OB3COUNT	format 3 count information

OBF1LBL and OBF3LBL are built by means of the OBTAIN macro. The obtain routine then updates the following control blocks:

Direct-Access Files

The user label address is moved into DTFDA.

Sequential Work Files

The VTOC address is moved from the obtain area into DTFSD in the form CHHR for DOS releases 25 and 26, and CCHR for DOS release 27.

Sequential Input Files

In order to simulate the bypassing of the DOS phase $$BOSDI1 for both open and end-of-extent cases, the following control blocks are updated:

- Logical unit (SYS number) in the open table
- Binary number in the open table and DTF
- Identifier (format 1) DSCB address in the open table seek bucket and limt bucket
- File ID (dsname) in the open table
- NXTBTR (in EMUCONS) to be $$BOSDI2

Sequential Output Files

The first three control blocks described for sequential input files are updated to simulate the bypass of $$BOSDO1 and $$BOSDO2. In addition:

- 'VTOC read' indicator in the open table is set on
- 'Special open' indicator in the DTF is reset if not a system file (filename must start with IJSYS)
- NXTBTR in EMUCONS is set to contain $$BOSDO5 at end-of-extent time, $$BOSDO4 at open time (if not release 27), $$BOSDO9 at open time (for release 27)

When running DOS release 27 only, the bypass of $$BOSDO4 is simulated by:

- Setting DOS register 0 with the appropriate control factor
- Increasing the volume sequence number in the DTF
- Moving identifier (format 1) DSCB address in the DTF
- Moving the actual user label address in the DTF and setting the 'header user label' indicator in the DTF

The obtain routine then exits to the build DLBL routine after setting the 'last extent' indicator in the DLBL/EXTENT card image if the extent being processed is the last one in the DEB.

In case of sequential output files, the 'last extent' indicator is set only if the extent is the last one in the DEB and if there is no secondary allocation in the DD statement.

Return Routine (Flowchart 23H)

This routine is entered from the other routines in IIVDVS when the function to be performed is completed. If everything is in order, a completion code of 0 is set in register 15 before returning to IIVGR2.

When an error must be indicated to IIVGR2, a value of 1 is set in register 15 and a message is issued to the operator by means of module IIVMSG.

Control returns to IIVGR2 after its registers have been restored.

Build DLBL Routine (Flowchart 23H)

This routine is entered from the obtain routine. Registers DCBREG, DTFREG, DLBLREG, DEBREG, and CTXTNREG must previously have been set. The routine first sets the following indicators in the DLBL/EXTENT card image:

Extent limits converted to DASD address	on
Next extent on a new pack on a new volume	off
New volume on same unit	off
Bypass extent	off
No extent card	off

The volume serial number is then moved from the OS UCB into the DLBL volume serial and file serial. If the file is a sequential disk file, the extent type and extent upper and lower limits are moved into DLBL from the corresponding extent in the DEB, and the routine exits to the return routine.

If the file is a direct-access file, the same processing is repeated for every extent found in the DEB. When there is not enough space in the DLBL/EXTENT card image to contain all the extents described in the DCB, a message INSUFFICIENT EXTENT SPACE is transmitted, and control is given to the return routine.

ISAM Mapping Routine (IIVIS) -- Flowcharts 24A-24L

Main Task Control Executive Routine (Flowchart 24A)

This routine is the entry point for mapping OPEN, CLOSE, SETFL ENDFL, and SETL functions. Control is passed by module IIVGR2 when a B-transient phase is required to process an OS indexed sequential data set. The required function is mapped from the DOS request to the equivalent OS request.

When this routine is entered, the following information is supplied:

- Register 11 contains the address of the EMUCONS area in which a parameter list for this module is contained
- Register 10 contains the address of DOS storage

- The 2-word parameter list contains the following:
  - The first byte of the first word is the function code (OPEN, CLOSE, SETL, etc.)
  - The last 3 bytes of the first word contain the DTFIS table address
  - The second word contains the address of the COMTAB extension (CTEXT) for OPEN, or the file ID block (FIDBLK) address at all other times

After checking for a valid function code, control is passed to the proper mapping routine or an error message is issued and control is returned to DOS.

Open Mapping Routine (Flowchart 24B)

This routine is entered each time a DOS OPEN is issued for an OS indexed sequential data set. Main storage is obtained in which to build the ISBLK, which is added to the chain of ISBLKs. The starting ISBLK address is contained in the ISFIDBLK field in EMUCONS. Each ISBLK holds an OS register save area, control information area, DCB(s) area, and a DECB area. A RDJFCB macro instruction is issued to obtain JCL information.

The type of DTFIS that is being opened determines the additional work still required.

LOAD Open Mapping Subroutine (Flowchart 24B)

For a LOAD DTFIS table (type X'24') where the DLBL open code is C for create, the DCB is completed from one of three sources: the DTFIS, the JFCB, or by default value. The priority of input varies by field as indicated in Figure 19.

Mapped to OS DCB field	Mapped from DOS DTFIS field	Mapped from JFCB field	Default value
DCBBLKSI	IJHKBKLN(2)	JFCBLKSI(1)	N/A
DCBLRECL	IJHKLGLN(1)	N/A	N/A
DCBKEYLE	IJHKLGLN+2(1)	N/A	N/A
DCBRECFM	IJHKOPCO	N/A	N/A
DCBRKP	IJHKNRCD+12(1)	N/A	N/A
DCBMACR	N/A	N/A	(PM) (1)
DCBCYLOF	Number of tracks per cylinder (IJHKHNDV+4) (2)	JFCCYLOF(1)	3(3)
DCBOPTCD	IJHKOPCD(2)	JFCOPTCD(1)	Always R(3) reorganization criteria
DCBBUFNO	N/A	JFCBUFNO(1)	2(2)
DCBNTM	N/A	JFCNTM(1)	3(2)

Figure 19. Sources of Input to DCB Fields at OS Indexed Sequential Data Set Creation

When these values have been initialized, an OPEN TYPE=J macro instruction is issued. If the open fails, the DOS task is terminated upon return of control to DOS. If the data set is successfully opened, the IJHKLPDR and IJHKPRCT fields in the DTFIS table are loaded from the DCBLPDA and DCBNRFC fields, respectively, of the DCB.

For a LOAD DTFIS table (type X'24') where the DLBL open code is E for extension load, only the OPEN TYPE=J macro instruction is issued and, if the data set is successfully opened, the IJHKLPDR and IJHKPRCT fields of the DTFIS table are loaded.

When the data set has been successfully opened for a LOAD DTFIS, the code used to trap and map I/O macro instructions is moved into the DTFIS table in the CCW build area (IJHKRDWR) and the ISMOD address in the DTFIS is modified to point to this code. Control is passed to the subtask attaching routine.

ADD, RETRVE, ADDRTR Open Mapping Subroutine (Flowchart 24B)

This subroutine opens an OS indexed sequential data set for ADD, RETRVE, or ADDRTR DTFIS tables, types X'25', X'26', and X'27', respectively. When the data set has been successfully opened, an OS work area is obtained to support the add function of ADD or ADDRTR type DTFIS tables.

The DTFIS table fields significant to the Emulator are completed from the information contained in the OS ISAM DCB (see Figure 20).

Mapped to DOS DTFIS field	Mapped from OS DCB field
IJHCRESZ	DCBLRECL
IJHCKYSZ	DCBKEYLE
IJHCBLSZ	DCBBLKSI
IJHCKYLC	DCBRKP
IJHACPRC	DCBNREC
IJHACOTC	DCBRORG3
IJHACOFC	DCBRORG1
IJHACORC	DCBNOREC
IJHCCLPA	DCBLPDA

Figure 20. Source of Input to Supported DTFIS Fields at Open of ADD, RETRVE, and ADDRTR

Control is passed to the subtask attaching routine.

Subtask Attaching Routine (Flowchart 24B)

This routine's primary function is to attach an OS subtask to perform the DOS ISAM macro mapping. A subtask is used so that other DOS tasks can continue to run when it is necessary to await the completion of the I/O required to access the OS indexed sequential data set.

The entry point for the subtask is at IIVIS01 within the IIVIS CSECT. This entry point is established by issuing an IDENTIFY macro instruction upon the first entry into the open mapping routine and setting a swtich in the first byte of AIIVIS in EMUCONS to indicate this macro has been issued. Also upon first entry, a CHAP macro instruction is issued to reduce the dispatching priority of the Emulator task by an order of 1 so that the subtask(s) can run at a higher priority when attached.

After the IDENTIFY and CHAP macro instructions have been issued, the address of the ISBLK is placed in register 1 and all 16 of the OS registers are stored in the register save area of the ISBLK. An ATTACH macro instruction is then issued to create the subtask.

The open mapping routine has completed its function at this point and control is returned to the calling routine.

Close Mapping Routine (Flowchart 24C)

This routine is entered each time a DOS CLOSE macro is issued to an OS indexed sequential data set. Any work area obtained to support the add function of ADD or ADDRTR DTFIS tables is released. The subtask that was attached to handle the DOS ISAM macro instructions to the OS indexed sequential data set is removed by issuance of an OS DETACH macro instruction. The data set is closed by issuing a CLOSE macro instruction. The ISBLK is unchained from the string of other ISBLKS and the area is released. Control is returned to the calling routine.

SETL Mapping Routine (Flowchart 24D)

This subroutine maps a DOS SETL macro instruction to an OS SETL macro instruction. The DTFIS must be for either RETRVE or ADDRTR, and IOAREAS must be nonzero to allow execution. If execution is not possible, the DOS task is abnormally terminated with an SVC 50 instruction.

If an OS SETL macro instruction is outstanding, an OS ESETL macro instruction must be issued to clear the SETL status in the DCB. If the DCB is not open to support the type of OS SETL macro, the DCB is closed, the DCBMACR field is respecified, and the DCB is reopened. Then the OS SETL macro instruction corresponding to the DOS SETL is issued. The mapping is performed as follows in Figure 21:

DOS SETL	OS SETL
Type BOF	Type B
Type KEY	Type K
Type GKEY	Type KC
Type ID	Type I

Figure 21. Mapping of DOS SETL to OS SETL

Control is passed to the status mapping routine so that any error conditions can be returned to the DOS problem program by means of the DTFIS status byte.

SETFL and ENDFL Mapping Routine. No function in OS corresponds to the DOS functions of SETFL and ENDFL, so any attempt to map them results in a NOP and control returns to the DOS problem program.

Subtask Control Executive Routine (Flowchart 24E)

This routine is entered when the open mapping routine issues the ATTACH macro instruction. The OS supervisor registers are saved and the routine's registers are loaded from the ISBLK register save area (pointed to by register 1). The subtask control executive routine then issues a WAIT macro instruction and enters the wait state until called upon to dispatch an ISAM macro instruction mapping request.

When the ECB in the ISBLK is posted by the SIO routine in module IIVPCE, the subtask control executive routine gets control, determines the validity of the request, and passes control to the proper mapping routines. After mapping is complete, the subtask control executive routine is reentered and again goes into the wait state until the next request.

Get Mapping Routine (Flowchart 24F)

This routine maps the DOS ISAM GET macro instruction to the OS ISAM GET macro instruction. Prior to issuing the GET, a check is made to ensure that a SETL was previously issued. If a SETL had not previously been issued, the DCB is checked to ensure it is open for QISAM; if not, the DOS task is terminated by an SVC 50. If the DCB is open for QISAM, the GET macro is issued and the OS ISAM module issues a SETL for the beginning of file.

When a SETL has been issued, the OS GET is issued. A successful completion allows this routine to move the logical record from the OS buffer into the DOS area reserved for it (specified by the IOAREAS parameter of the DTFIS macro). The key for unblocked files is also moved. The DOS address of the logical record is stored in the DTFIS table and the record is moved into the WORKS area, if the WORKS area is specified. The DASD address of the block is obtained from the DCB work table and stored in the DTFIS table for reference by the problem program. If the record is to be processed in IOAREAS, then code is created so that the address is loaded when the DOS task is reentered.

Final status is mapped to the DTFIS table by the status mapping routine.

Put Mapping Routine (Flowchart 24G)

This routine maps the DOS ISAM PUT macro to the OS ISAM PUTX macro. If a GET macro has not been issued to this data set by the problem program prior to the PUT, the DOS problem program is abnormally terminated by means of an SVC 50 instruction.

When the record is processed in the WORKS area, it is first moved to the IOAREAS logical record location. The logical record is moved from IOAREAS into the OS buffer logical record location, and an OS PUTX macro is issued.

Upon return from the PUTX, the status information is mapped from the DCB to the DTFIS table by the status mapping routine.

ESETL Mapping Routine (Flowchart 24G)

This routine maps the DOS ISAM ESETL macro to the OS ESETL macro. If a SETL macro has not been issued for the data set prior to the ESETL, the routine returns to the subtask control executive routine. Otherwise, an OS ESETL is issued. Control is passed to the status mapping routine.

Read Key Mapping Routine (Flowchart 24H)

This routine maps the DOS READ KEY macro to the OS READ macro, type KU. The DTFIS must be built for random retrieval before mapping is possible. If the IOAREAR address is zero, the problem program is abnormally terminated with an SVC 50 instruction.

A check is made to determine that a DCB is open for BISAM macro processing.

The key argument is obtained from the DTFIS table and used in an OS READ type KU macro instruction. Upon return from OS, final status is mapped from the DECB to the DTFIS table by the status mapping routine.

Write Key Mapping Routine (Flowchart 24H)

This routine maps the DOS WRITE KEY macro to the OS WRITE macro, type K. The DTFIS must provide for random retrieval and a DOS READ KEY must have been issued to the file before mapping is possible. If these conditions are not met, the DOS problem program is abnormally terminated by an SVC 50 instruction.

Any error on the previous READ KEY will not allow the WRITE KEY to be performed, so control is returned to the DOS problem program. When all restrictions have been met, the updated record is moved from the WORKR area (when specified) into the IOAREAR logical record location. The record is then moved from IOAREAR in DOS to the OS buffer. An OS WRITE type K macro instruction is issued to write the logical record back into the file.

Final status is mapped to the DTFIS table by passing control to the status mapping routine.

Write NEWKEY Mapping Routine (Flowchart 24J)

This routine maps the DOS WRITE NEWKEY macro to the OS WRITE type KN macro. The DTFIS table must provide for random adding of logical records before mapping can be accomplished. If this condition is not met, the DOS problem program is abnormally terminated by an SVC 50 instruction.

The key argument is moved to the save area in the DTFIS table, its address is stored in the DECB, and an OS WRITE type KN macro instruction is issued.

Final status is mapped to the DTFIS table by the status mapping routine.

WAITF Mapping Routine (Flowchart 24K)

This routine simulates the DOS WAITF macro instruction for DOS WRITE NEWKEY, READ KEY, and WRITE KEY processing. If one of these macro instructions has not been issued, the problem program is abnormally terminated by an SVC 50 instruction.

For either WRITE macro instruction, a CHECK macro instruction is issued to determine if the I/O operation has finished. This routine exits to the status mapping routine.

When the I/O is a READ KEY, an OS CHECK macro instruction is issued and the DASD address and logical record are moved to the DOS problem program storage locations. The logical record is moved into the WORKR area, if specified. If IOREG is specified, code is created in the DTFIS CCW build area to load the register with the address when the DOS problem program regains control. This routine exits to the status mapping routine.

ISAM Mapping Subroutines

SVC 50 Subroutine (Flowchart 24L)

This subroutine causes an SVC 50 instruction to be the next instruction issued when the DOS task acquires control after the I/O operation has been completed. This is done so that the DOS task will be canceled for violating a DOS restriction. The user must check his program logic to determine the cause.

OPENFAIL Subroutine (Flowchart 24L)

This subroutine causes the DOS OPEN message writer to be the next B-transient phase called because of errors that have occurred during the process of opening an OS indexed sequential data set. This causes the DOS task to terminate.

SYNAD Subroutine (Flowchart 24L)

This subroutine is entered from the OS ISAM logic modules when error conditions have been detected while performing OS ISAM macro instruction requests.

The DCBMACRF field is interrogated so that the proper status mapping routine can be called.

Status Mapping Subroutine (Flowchart 24L)

This subroutine is the common exit point for all subtask I/O mapping routines and the SETL mapping routine. Its function is to map the statistics fields and last prime data record address from the DCB to the DTFIS. This subroutine also maps completion status from either the DECB or the DCB exceptional condition bytes, fields DECBEXC1-DECBEXC2 and DCBEXCD1-DCBEXCD2, respectively, to the DTFIS exception condition status byte, field IJHCSTBY (see Figure 22).

Entry is at one of three entry points, depending on whether the OS ISAM DCB is used for QISAM load, QISAM scan, or BISAM mode. Each entry point contains instructions to set up registers to point to the proper OS exceptional condition bytes and the corresponding OS-to-DOS mapping table. The mapping table consists of a DOS bit, which is set on in byte IJHCSTBY in the DTFIS table, if the corresponding bit is on in one of the OS exception bytes.

Each OS exceptional condition byte is checked independently for a value of 0. If both are 0, IJHCSTBY in the DTFIS table is set to 0 and the status mapping subroutine exit is entered. If a byte is found to be nonzero, the proper point in the map table is set and the byte is scanned from left to right a bit at a time until all 8 bits have been checked. When a bit is found to be 1, the corresponding bit configuration byte in the mapping table is ORed into the DTFIS status byte (IJHCSTBY).

When both bytes have been interrogated, the status mapping subroutine exit is entered. This section of the subroutine determines whether the Emulator task or a subtask is in control by interrogating a control byte, TAFLAG3, in the ISBLK. When TAFLAG3 is 0, the Emulator task is in control and a branch is made to the main task control executive routine.

When a subtask is in control, the completion of the OS ISAM I/O request must be signaled to the Emulator task. This is accomplished by loading the address of the COMTAB entry for the request from the ISBLK, creating a CSW in the IOB in the COMTAB entry, and issuing a POST macro instruction for the ECB in the COMTAB entry. Control is then passed to the subtask control executive routine which will zero TAFLAG3 in ISBLK and enter the wait state.

Mapped to DOS DTFIS field	Mapped from OS DCB field	
IJHACRRC	DCBNREC	Prime data record count
IJHACOTC	DCBRORG3	Number of independent overflow records
IJHACOFC	DCBRORG1	Number of cylinder overflow area full
IJHACORC	DCBNOREC	Overflow record count
IJHCCLPA	DCBLPDA	Address of last prime data record
IJHCSTBY	QISAM (Load mode)	
	DCBEXCD1	
X'20'	X'20'	
X'80'	X'04'	
	DCBEXCD2	
X'02'	X'80'	
X'04'	X'40'	
X'80'	X'20'	
X'40'	X'08'	
	QISAM (Scan mode)	
	DCBEXCD1	
X'10'	X'80'	
X'08'	X'40'	
X'10'	X'10'	
X'80'	X'08'	
X'80'	X'04'	
X'10'	X'02'	
X'10'	X'01'	
	DCBEXCD2	
X'80'	X'20'	
X'01'	X'10'	
	BISAM	
	DECBEXC1	
X'10'	X'80'	
X'40'	X'40'	
X'02'	X'20'	
X'80'	X'10'	
X'80'	X'08'	
X'10'	X'04'	
X'01'	X'02'	
X'04'	X'01'	

Figure 22. Mapping of DCB Fields to DTFIS Fields After Processing of Each I/O Macro

VTOC I/O Simulation Routine (IIVVIO) -- Flowcharts 25A-25E

The VTOC I/O simulation routines are used to provide VTOC label information and actual user label extent information to DOS when a data set (file) on a shared volume is being opened.

When the SIO subroutine (IIVPCE) detects a request for I/O on a shared volume and the seek address cannot be matched with the extents in the DEB for that volume, the request is assumed to be for the accessing of VTOC labels or user labels for the data set (file) by a DOS open routine. Control is passed to IIVVIO to verify the assumption and provide the requested I/O.

Routines VIOA through VIOJ are used to determine the type of access requested. Routines VIOIO through VIOIOF provide the I/O simulation.

VIOA - Analyze CCW Command Code Routine (Flowchart 25A)

The CCW command code is matched with a table containing the acceptable commands, and the appropriate processing routine is executed. If no match is found, control is passed to VIOERRX.

Each entry in the command code table is a fullword, containing the command code in the high order byte and the address of the routine to be executed in the 3 low order bytes.

Command Code		Routine
03	NOP	VIOB
07	Seek	VIOC
08	TIC	VIOD
31	Search ID equal	VIOE
29	Search key equal	VIOF
12	Read count	VIOG
06	Read data	VIOH
0E	Read key/data	VIOI
05	Write data	VIOJ
0D	Write key/data	VIOJ

Note: The multitrack bit is ignored.
The type byte is set by routines VIOB through VIOJ to indicate the type of access requested.

Flag Values	Request
80	Search ID equal for VOL1 label
40	Search ID equal for (format 1) label
20	Search ID equal for (format 3) label
10	Read count
08	Search key equal
04	NOP
02	Read data
01	Read key/data
FF	Write

VIOB - NOP Command Code Routine (Flowchart 25B)

The NOP flag is set on in the type byte. Control is passed to VIONXT.

VIOC - Seek Command Code Routine (Flowchart 25B)

If the user label extent is present in the obtain work area and the seek address is for the user label CCHH, the user label flag is set on in the COMTAB (CTFLAG3) and the extent is moved to the COMTAB for use by the end-of-extent appendage. The return code is set to 00 to notify the calling routine an EXCP is to be issued, and return is made to the calling routine.

If the seek address is for CCHH 0000 or for the CCHH of the VTOC, control is passed to VIONXT.

Otherwise, the return code is set to 08 to notify the calling routine that an error condition has been detected and return is made to the calling routine.

VIOD - TIC Command Code Routine (Flowchart 25B)

It is assumed that the TIC is one that follows a search. Control is passed to VIONXT.

VIOE - Search ID Equal Command Code Routine (Flowchart 25B)

If the ID is for the VOL1 label (CCHHR = 00003), the VOL1 flag is set on in the type byte. If the ID is equal to the format 1 label ID, the format 1 flag is set on in the type byte. If the ID is equal to the format 3 label ID, the format 3 flag is set on in the type byte. Control is always passed to VIONXT.

VIOF - Search Key Equal Command Code Routine (Flowchart 25B)

The search key equal flag is set on in the type byte. Control is passed to VIONXT.

VIOG - Read Count Command Code Routine (Flowchart 25C)

The read count flag is set on in the type byte, and the count address is saved. Control is passed to VIONXT.

VIOH - Read Data Command Code Routine (Flowchart 25C)

The read data flag is set on in the type byte, and the data address is saved. Control is passed to VIONXT.

VIOI - Read Key/Data Command Code Routine (Flowchart 25C)

The read key/data flag is set on in the type byte, and the data address is saved. Control is passed to VIONXT.

VIOJ - Write Data and Write Key/Data Command Codes Routine (Flowchart 25C)

The type byte is set to X'FF' to indicate no data is to be transferred. DOS will be writing the VTOC label so only simulation of the I/O interruption is required. Control is passed to VIONXT.

VIONXT - Get Next CCW Routine (Flowchart 25C)

If the current CCW command code is a TIC and if the command chain bit is on in the CCW, the CCW pointer is increased to point to the next CCW in the chain. Control is then passed to VIOA. Otherwise, control is passed to VIOIO to select the correct simulation routine.

The following routines determine the type of I/O requested and provide any data required by the DOS open routine request. The type byte flags are tested for the combinations set by the command code routines VIOB through VIOJ. The data transferred to the DOS data areas is the real VTOC label data built by the obtain subroutine within module IIVDVS. The obtain work area containing the VTOC labels is in module IIVDVS starting at label IIVOBTE1. The address of IIVOBTE1 is in EMUCONS at label AIIVOBE1.

The type byte must indicate one of the following CCW chains or the I/O request is considered invalid.

VIOWKD - Write a VTOC label
(1) Seek (CCHH equal obtained format 1 CCHH)
(2) Search ID equal
(3) TIC to search (2)
(4) Write key/data
(5) Search ID equal
(6) TIC to search (5)
(7) Read key/data skip on

VIORDV1 - Read VOL1 label
(1) Seek (CCHH 0000)
(2) Search ID equal (CCHHR 00003)
(3) TIC to search (2)
(4) Read key/data

VIORDV12 - Read VOL1 label data only
(1) Seek (CCHH 0000)
(2) Search ID equal (CCHHR 00003)
(3) TIC to search (2)
(4) Read data

VIORDF3 - Read format 3 label
(1) Seek (CCHH equal obtained format 1 or 3 CCHH)
(2) Search ID equal (CCHHR equal obtained format 3 CCHHR)
(3) TIC to search (2)
(4) Read key/data

VIORDF1K - Read format 1 label data only
(1) Seek (CCHH equal obtained format 1 CCHH)
(2) Search ID equal (CCHHR equal obtained format 1 CCHHR)
(3) TIC to search (2)
(4) Search key equal multi-track
(5) TIC to search (4)
(6) Read data

VIORDF1C - Read format 1 label count and data
(1) Seek (CCHH equal obtained format 1 CCHH)
(2) Search ID equal (CCHHR equal obtained format 1 CCHHR)
(3) TIC to search (2)
(4) Read count multi-track
(5) Search key equal
(6) TIC to read count (4)
(7) Read data

VIORDF1D - Read format 1 label key and data
(1) Seek (CCHH equal obtained format 1 CCHH)
(2) Search ID equal (CCHHR equal obtained format 1 CCHHR)
(3) TIC to search (2)
(4) Read key/data

VIORDF0 - Read format 0 count field
(1) Seek (CCHH equal obtained format 1 CCHH)
(2) Search ID equal (CCHHR equal obtained format 1 CCHHR)
(3) TIC to search (2)
(4) Read count multi-track
(5) Search key equal
(6) TIC to read count (4)
(7) NOP

VIOIO - VTOC I/O Simulation Selection (Flowchart 25C)

The type byte is matched with a table containing the combined type codes, and the appropriate processing routine is executed. If no match is found, control is passed to VIOERRX. Each entry in the type code table is a fullword containing the combined type code in the high order byte and the address of the routine to be executed in the 3 low order bytes.

Combined Type Code		Routine
FF	Write	VIOIOA
81	Read VOL1 key/data	VIOIOB
82	Read VOL1 data	VIOIOB1
21	Read F3 key/data	VIOIOC
4A	Read F1 data	VIOIOD
5A	Read F1 count/data	VIOIOD
41	Read F1 key/data	VIOIOE
1C	Read F0 count	VIOIOF
5C	Read F0 count	VIOIOF

VIOIOA - Write Simulation Routine (Flowchart 25D)

Control is passed to VIOIOSIM.

VIOIOB - Read VOL1 Label Simulation Routine (Flowchart 25D).
VIOIOB1

The VOL1 label is moved from the obtain work area to the address specified in the read CCW. Control is then passed to VIOIOSIM.

VIOIOC - Read Format 3 Label Simulation Routine (Flowchart 25D)

The F3 label in the obtain work area is moved to the address specified in the read CCW. Control is then passed to VIOIOSIM.

VIOIOD - Read Format 1 Label (Data) Simulation Routine (Flowchart 25D)

The data area of the F1 label in the obtain work area is moved to the address specified in the read CCW. Control is then passed to VIOIOSIM.

VIOIOE - Read Format 1 Label (Key/Data) Simulation Routine (Flowchart 25E)

The F1 label in the obtain work area is moved to the address specified in the read CCW. Control is then passed to VIOIOSIM.

VIOIOF - Read Format 0 Count Field Simulation Routine (Flowchart 25E)

The count field of the F1 label in the obtain work area is moved to the address specified in the read count CCW. Control is then passed to VIOIOSIM.

VIOERRX - Error Exit Routine (Flowchart 25E)

The return code 08 is set in register 15, and return is made to the calling routine.

VIOIOSIM - Simulation Exit Routine (Flowchart 25E)

The return code 04 is set in register 15, and return is made to the calling routine.

Exit ABEND Error Routine (IIVABN) -- Flowcharts 26A-26B

This routine is entered when simulation of some error condition to DOS or termination of the Emulator is required.

The routine examines the error code passed from the module in which the error was found. The proper message is printed if the calling routine has not already printed it. The error codes are:

- 04 - Return to OS (any queued devices are dequeued by means of the DEQ macro)
- 08 - DOS hard wait
- 12 - Invalid or no IPL device defined
- 16 - Invalid CCW found by IIVCCW
- 20 - Emulator CCW chain table size exceeded
- 24 - No seek or no bin number for IBM 2321 Data Cell Drive

If a DOS partition is being canceled, the channel end and program check bits in the DOS CSW are turned on to force DOS to cancel the partition. Partition cancelation occurs for errors found by the IIVCCW routine, for example, invalid CCW data address or CCW not on a doubleword boundary (error code 16), if the Emulator CCW chain table size is exceeded (error code 20), or if the IIVPCE SIO routine finds invalid 2321 CCWs (error code 24). Exit is made to the caller for return to DOS if the Emulator region is to be canceled. CCWs for active devices are adjusted to local addresses, and an OS snap dump of DOS registers 0 through 15 and DOS storage is taken if the JCL statement //SYSSNAP DD SYSOUT=A was included for the Emulator job step. A service aid printout instead of the snap dump will be taken if the DEBUG option is specified. The routine returns to OS. The Emulator is canceled if the DOS supervisor attempts to enter a hard wait (error code 08), if there is insufficient main-storage space to run the Emulator (error code 04), or if an invalid or no IPL device is defined (error code 12).

Message Writer Routine (IIVMSG) -- Flowchart 27A

This routine prints all messages issued by the Emulator. See Figure 88 entitled "Message-to-Module Relationship" for specific message codes, message numbers, and module names. Emulator routines can request a message to be printed by issuing an Emulator macro called EMUMSG. This macro generates the code required to pass control to this routine. The parameters passed to IIVMSG are used to select the message text to be printed, complete the message with any supplemental text provided by the caller, and determine if the request is for a WTO or WTOR.

Emulator messages are contained within three text modules: IIVMG1, IIVMG2, and IIVMG3. These modules contain messages appropriate for issuance by the Emulator during initialization (IIVMG1), after initialization (IIVMG2), and when volume and data set sharing has been requested (IIVMG3).

Each message is identified with a unique message code. Within each text module, messages are assigned sequential numbers from 1 to 99. These numbers are added to the appropriate module identifier to form the message code used in the EMUMSG macro call. The module identifier for IIVMG1 is 0, for IIVMG2 is 100, and for IIVMG3 is 200. The module identifier determines both the message and the required module.

Two Emulator macros, MGTXT and MSGCOD™, are used to create the message text modules. MGTXT is issued for every message contained in each of the text modules. The macro generates control bytes used for supplemental text and the constants for the text itself. MSGCODT is issued once in each module, and it must be issued after all the MG™XT calls. MSGCOD™ generates a message index table.

The three macros, EMUMSG, MGTXT, and MSGCOD™, are described below.

```
EMUMSG   MSGCODE={code}
                 {(reg)}

                {data address}
        [,FILL={D(data address)} ]
                {(reg)}

                 {reply address}
        [,REPLY={D(reply address)}
                 {(reg)}

             {ecb address}
        ,ECB={D(ecb address)}
             {(reg)}

        ,REPLEN={reply length} ]
                {(reg)}
```

where MSGCODE is a required parameter:

code is the message code described above (IIVMG1, 1-99; IIVMG2, 101-199; IIVMG3, 201-299).

(reg) is one of registers 0 through 12, which contains the message code in the low-order byte.

FILL is an optional parameter:

data address is the symbolic address of the supplemental text.

D(data address) is the same as data address except the symbolic address is in a DSECT.

(reg) is one of registers 1 through 12, which contains the text address.

REPLY (see note below):

reply address is the address of the reply area.

D(reply address) is the same as reply address, except the address is a DSECT label.

(reg) is one of register 1 through 12, which contains the reply address.

ECB (see note below):

ecb address is the address of the ECB.

D(ecb address) is the DSECT address of the ECB.

(reg) is one of registers 1 through 12, which contains the ECB address.

REPLEN (see note below):

reply length is the length of the required reply.

(reg) is one of registers 1 through 12, which contains the reply length in its low-order byte.

Note: The parameters REPLY, ECB, and REPLEN must all be present or all omitted. Any parameter not enclosed by brackets must be present.

```
MGTXT    {'text'}
         {(...,'text',...,n,...)}
```

where 'text' is the message text in quotes; use this form when no supplemental text is required. (...,'text',...,n,...) is used when supplemental text is required. The information within the parentheses represents any combination of 'text' and n, separated by commas. The n represents a field length for the supplemental data. The placement of the n relative to the 'text' determines the displacement of the field to be filled.

For example, MGTXT ('THIS IS AN', 8) generated code would be:

```
DC AL2(8)            Length of supplemental data
DC AL1(128)          Last entry indicator
DC AL1(11)           Byte position within the message text
DC C'THIS IS AN '
DC CL8' '
```

Assuming the above macro call is the fifth issuance of MGTXT in module IIVMG2, then the message code is module identifier (100) + 5 = 105.

To print this message, the requesting routine issues the macro EMUMSG:

```
EMUMSG MSGCODE=105,FILL=DATA
```

where DATA DC CL8'EXAMPLE.'

The resulting message will be:

THIS IS AN EXAMPLE.

Actually the message text must contain the message identifier (IIVxxxT) and space for an 8-byte job name, which the message writer always fills in. All MGTXT macro calls must, therefore, begin as follows:

```
MGTXT 'IIVxxxT  jobname  ...'
      (jobname=space for 8-byte Emulator job name)

or    ('IIVxxxT jobname  ...',...)
```

where xxx = message ID and T = message type.

MSGCODT No required parameters.

This macro is issued once in each message text module. It must be issued after all the MGTXT calls. Global values in the MGTXT macro are used to generate a table containing the following information:

```
DC   A(maximum message number)
          or
DC   A(number of MGTXT calls)
DC   AL1(length of message 1)
DC   AL3(address of message 1 including any control
     bytes for supplemental text)
     .
     .
     .
Repeat for message 2, and so on.
```

Service Aids Initialization Routine (IIVRAS) -- Flowcharts 28A-28B

The major function of module IIVRAS is to control the program flow of the service aids modules IIVRCP and IIVSNP.

At entry to this routine, the DCB for SYSSNAP is opened and address constants in CSECTs IIVRCN and IIVCON are initialized. If register 0 contains zero at the time of entry, the service aids command processor IIVRCP is loaded into main storage. Control is then passed to IIVRCP to process a debug statement. Module IIVRCP is deleted after control is returned. Module IIVSNP is then loaded, and its main-storage address is placed in CSECT IIVRCN. Control is then returned to the caller.

If register 0 contained a value of 4 at entry time, the Emulator snap dump routine IIVSNP is loaded into main storage. Control is then passed to module IIVSNP to snap the Emulator wraparound trace table. This will normally happen at Emulator end-of-job time. Control is returned to the caller after the trace table is snapped.

If register 0 contained a value greater than 4, register 1 must contain a main-storage address of a two-byte length field. This field contains the length minus one of a character string that describes the cause of the snap dump (OPTION IN EFFECT gives the cause of the snap dump). The character string immediately follows the length field. Control is passed to module IIVSNP to snap Emulator control blocks. After the Emulator storage is snapped, control is returned to the caller.

The following routines in IIVRAS gain control depending on the type of interruption that occurs:

IIVRASPC - Program Check Intercept Initialization Routine (Flowchart 28B)

Control is gained at this entry point after the occurrence of a program check interruption when the CPU is in local execution mode. DOS registers are saved and Emulator registers are restored. This subroutine exits to module IIVPCI, which further interrogates the program check interruption.

IIVRASVC - Supervisor Call Intercept Initialization Routine (Flowchart 28B)

Control is gained at this entry point after the occurrence of a supervisor call interruption when the CPU is in local execution mode. DOS registers are saved and Emulator registers are restored. This subroutine exits to module IIVSCI, which will further interrogate the supervisor call interruption.

IIVRASYN - Asynchronous Intercept Initialization Routine (Flowchart 28B)

Control is gained at this entry point after the occurrence of an asynchronous interruption when the CPU is in local execution mode. DOS registers are saved and Emulator registers are restored. If requested, control is passed to a user asynchronous exit routine. Control will then be passed to module IIVSNP if an optional snap dump is indicated. DOS registers are then restored and this routine exits to the Emulator module IIVRTE.

Command Processor Routine (IIVRCP) -- Flowcharts 29A-29P

This routine gains control from IIVRAS in order to read a DEBUG statement from card input or a DEBUG console reply to the Emulator prompt.

Each command of the DEBUG statement is checked for syntax validity. Each keyword or parameter is then analyzed and corresponding indicators are set in RASCONS to be used by the service aids modules. These modules (IIVACI, IIVPCI, IIVSCI) are loaded by IIVRCP with user exit modules (if any) when an END command is encountered. Control then returns to IIVRAS, which will delete IIVRCP and resume emulation.

Snap Dump and Trace Formatting Routine (IIVSNP) -- Flowcharts 30A-30K

The major function of module IIVSNP is to format the Emulator control blocks and trace table. A nonzero value in register 0 at entry to this routine will cause only the trace table to be printed. The main line code consists of calls to subroutines that convert main storage data to EBCDIC and write the formatted data.

Trace Table Subroutine (Flowchart 30E)

The entry point label of the trace table subroutine is RAS14000. Control is returned to the caller if the number of active trace table entries is zero. Otherwise, each entry is converted from its internal format (see Figure 84, Internal Trace Table Format) to its printed format (see Figure 86 (Part 6 of 7). Service Aids Snap Dump). Trace table entries are printed starting with the most recent entry. Each succeeding line represents an older entry. The trace table pointers are reset immediately after snapping so no two snaps will reflect the same events.

Write Subroutine (Flowchart 30H)

The entry point label of the write subroutine is RAS15000. When this subroutine is entered, register 1 contains the address of a two-byte length field, which contains the length in bytes minus one of a character string; the character string immediately follows the length field. Each line is blocked until the buffer is filled, at which time the entire buffer is written out. A secondary entry point (RAS15200) will cause the buffer to be truncated and written. Control is returned to the caller.

Snap Subroutine (Flowchart 30K)

The entry point label of the snap subroutine is RAS16000.
Register contents upon entry to this subroutine are:

- Register 0 - the length in bytes of main storage to be snapped
- Register 1 - the starting address of the main storage to be snapped

Control is first passed to the EBCDIC conversion subroutine (entry point is RAS17000) to convert hexadecimal data to EBCDIC. The EBCDIC data is then formatted in fullword sections for readability by the data formatting subroutine (entry point is RAS18000). Control is then passed to the write subroutine (entry point is RAS15000). Data is snapped 32 bytes at a time until the length (passed in register 0) is reached. Only the first line of a set of duplicate lines is printed so that the volume of output is minimized.

EBCDIC Conversion Subroutine (Flowchart 30G)

The entry point label of the EBCDIC conversion subroutine is RAS17000.
The register contents upon entry to this subroutine are:

- Register 0 - the number of bytes to convert
- Register 1 - the main-storage address of the hexadecimal data to be converted

The number of bytes contained in register 0 is rounded to the next fullword. A maximum of 72 bytes can be converted at any one time. Output data is placed in a work area labeled BUFF1. After conversion, control is returned to the caller.

Data Formatting Subroutine (Flowchart 30J)

The entry point label of the data formatting subroutine is RAS18000.
The register contents upon entry to this subroutine are:

- Register 0 - the number of bytes to format
- Register 1 - the beginning of main-storage address to contain the formatted output

Input data is found in a work area labeled BUFF1. EBCDIC data is formatted in fullword segments for readability. Control is returned to the caller.

Program Check Intercept Routine (IIVPCI) -- Flowcharts 31A-31C

The major function of module IIVPCI is to intercept DOS program check interruptions. IIVPCI receives control from module IIVRAS (entry point is IIVRASPC) after DOS registers have been saved and Emulator registers have been restored. Control will be passed to module IIVSNP to snap Emulator control blocks if a snap for DOS program checks was requested and if a nonprivileged operation program check or if a nonEmulator supported privileged operation exception is encountered. Snap dumps during DOS IPL are bypassed unless specifically requested because of the number of program checks that occur during that time.

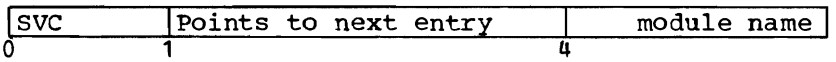
The first of two snaps of the Emulator control blocks and main storage will be taken for user selected privileged operations that are issued by DOS. This dump will be a picture of the Emulator region before instruction simulation by the Emulator. The second snap will be issued by module IIVACI. This dump will contain a picture of the Emulator region after the instruction is simulated.

User selected privileged operations will be entered into the Emulator trace table if a trace table was requested. Since all required trace information is not available at program check time, a flag is set in IIVRCN (RASCONS) to indicate that that entry must be completed by module IIVACI. Just before passing control to Emulator module IIVPCE, control will be passed to a program check user exit routine if one was specified. An optional snap dump will be taken if the exit routine returns to the address in register 14 plus a displacement of four.

Supervisor Call Intercept Routine (IIVSCI) -- Flowchart 32A

Module IIVSCI receives control from module IIVRAS (entry point is IIVRASVC) after the occurrence of a supervisor call interruption while the CPU is in local execution mode.

If an SVC snap dump was requested, a four byte field at hexadecimal location 1A8 in CSECT IIVRCN will contain the starting main-storage address of a chain of SVC control blocks. Each entry in the chain is 12 bytes in length and has the following format:



Control is passed to module IIVSNP to snap Emulator control blocks if the interruption field in the local execution PSW matches the SVC number in byte 0 of an entry. A value of zero in the next entry field (byte 1) indicates the last entry in the chain. In the case of SVC numbers 2 and 4, the main-storage address contained in DOS register 1 is adjusted and compared with the 8 character module name in byte 4 of the entry if that field contains a nonzero value. An equal compare will cause a snap dump to be taken.

If SVC or ALL were specified for trace, the trace table will be updated.

The supervisor call interruption will then be partially recorded in the trace table if SVC or AIL are specified for TRACE. Just before exiting to module IIVSVC, control will be passed to a supervisor call user exit routine if one was specified. An optional snap dump will be taken if the user exit routine returns to the address contained in register 14 plus a displacement of four. This routine restores DOS registers and exits to the Emulator module IIVSVC to process the SVC instruction.

Asynchronous Intercept Routine (IIVACI) -- Flowcharts 33A-33H

Module IIVACI receives control from the various Emulator modules when control is to be returned to DOS. The address constant ART20 (which ordinarily contains the main-storage address of IIVRTER2) is replaced by module IIVRCP with the entry point address of IIVACI.

Since this routine gains control from other Emulator modules, it is assumed that registers are set up with standard Emulator values. If the current trace entry is flagged incomplete, it is completed according to the type of interruption it represents. The trace table pointers are then updated. Asynchronous interruptions (I/O, EXT, TIMER) are then checked and entered if requested.

Control is passed to module IIVSNP to issue the second of two snap dumps if indicated by module IIVPCI. This snap dump will reflect Emulator main storage after a valid DOS privileged operation was simulated by the Emulator.

The PSW and COMP snap functions are also handled in module IIVACI. If these options are specified, a snap dump will be taken whenever the conditions are satisfied. Asynchronous interruptions (I/O, EXT, TIMER) are checked and a snap dump is taken if requested. The local execution user exit routine is given control just before passing control to module IIVRTE at entry point IIVRTER2. An optional snap dump is taken if control is returned to the address in register 14 plus a displacement of four.

Service Aids Adjust CCW Data Address Routine (IIVRCW) -- Flowcharts 34A-34H

The main function of IIVRCW is the same as IIVCCW. The DOS local addresses in the data address portion of the CCWs will be adjusted to OS true addresses or readjusted from OS true addresses to DOS local addresses.

The first path of IIVRCW is similar to IIVCCW. BEBLK entries are created according to the channel program to be processed. The BEBLK entries contain pointers to the beginning and ending addresses of each consecutive set of CCWs found in the channel program.

The second step of IIVPCW is slightly different from IIVCCW in that the CCWs will not be adjusted or readjusted in the DOS area, but will leave the DOS channel programs unchanged.

Start I/O Time

IIVRCW copies the DOS channel program into a buffer located in the DIAG block (the block is created by IIVRCP when the DIAG command routine is entered). The data address portions of the CCWs in the buffer are then adjusted to OS true addresses and the TIC addresses will point to the corresponding CCW in the buffer. The IOB is then modified to reflect that the OS EXCP should be issued on the channel program located in the DIAG block.

I/O Completion Time

IIVRCW is called by IIVCHK when the OS EXCP is complete. The data address portions of the CCWs in the buffer are readjusted to DOS local addresses. The TIC addresses will point to the corresponding CCW in the DOS area. When readjustment is complete, the channel program located in the DIAG block should be identical to the channel program located in the DOS area. A check is made on each CCW and message IIV281I is issued if the CCWs do not match. The IOB is then modified to reflect that the I/O is completed on the DOS channel program (the CSW will be pointing to the corresponding CCW in the DOS area).

Functional Organization of Emulator Interruption Handling

Flowcharts

FUNCTIONAL ORGANIZATION OF EMULATOR INTERRUPTION HANDLING

In Figure 23, the major Emulator interruption handling functions are grouped relative to the modules that perform them.

FLOWCHARTS

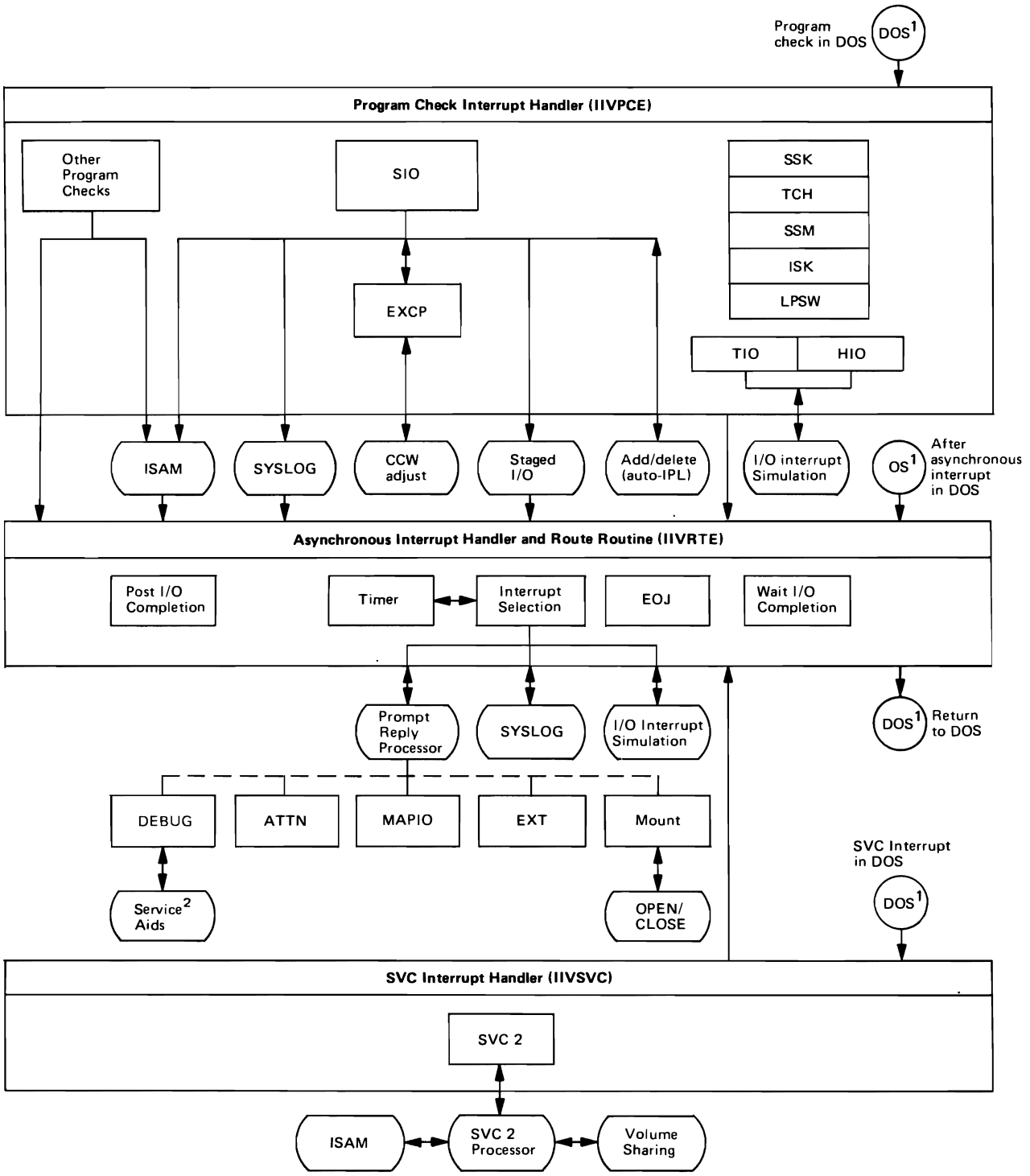
This section contains flowcharts of Emulator routines. In general, the flowcharts have the following characteristics:

- Symbols, or labels, are used where possible to aid readers in locating instruction sequences in listings.
- Where CPU control passes from one routine or subroutine to another, terminal blocks are always used. Off page connectors are used only when a single routine or subroutine extends over two or more pages.
- Information necessary to understand a module and its interaciton with the rest of the Emulator is given in text preceding each major module.
- Redundancy is sometimes used to avoid excessive cross-referencing; the same function may sometimes be shown in two or more flowcharts, even though, in each case, it is performed with the same set of instructions.
- Each flowchart identifier begins with a numerical value ranging from one to two digits. The numerical prefix is unique for each module. The flowchart identifiers end with an alphabetic character ranging from A-Z, omitting the letters I and O to avoid confusion with the digits 1 and 0. The alphabetic character following the numerical prefix of the flowchart identifier, starts with the letter A and progresses alphabetically for each flowchart of a module.

The flowcharts are divided into seven categories:

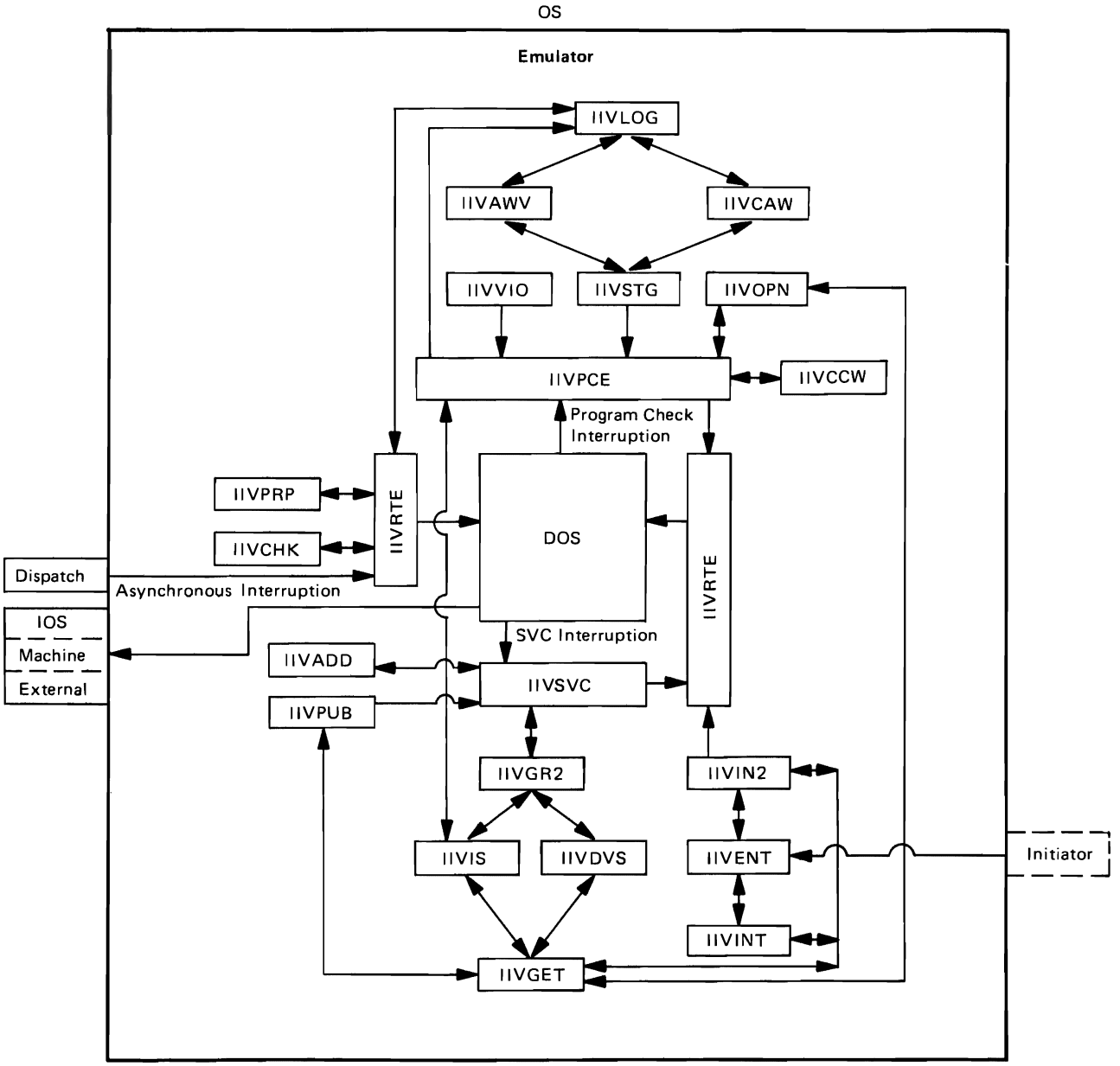
- Initialization
- Synchronous interruptions
- Asynchronous interruptions
- Direct-acces volume sharing
- Abnormal end conditions
- Message writer
- Emulator service aids

Figure 24 shows the module relationships of the Emulator.



1. Depending on the DEBUG options taken, service aids may first receive control.
2. Service aids can also be activated during initialization if a //SYSDEBUG DD statement is present.

Figure 23. Functional Organization of Emulator Interruption Handling



Note: Module IIVRAS functions are not included in this diagram (see "Emulator Service Aids" for information concerning the relationship of IIVRAS to other Emulator modules).

Figure 24. Emulator Module Relationship

Initialization

DOS Emulator Entry Routine (Flowchart 1A)

Module name: IIVENT

Entry point name: IIVENT

Major functions:

- Saves registers
- Saves pointer to user parameter area
- Establishes a CSECT of constants (EMUCONS) at assembly time

Entered from: OS

Modules called: IIVINT

Exits to:

- IIVIN2
- IIVABN

OS macros issued: SAVE

Input: Register 1 points to the pointer to the user parameter area

Output: Register 7 points to the user parameter area

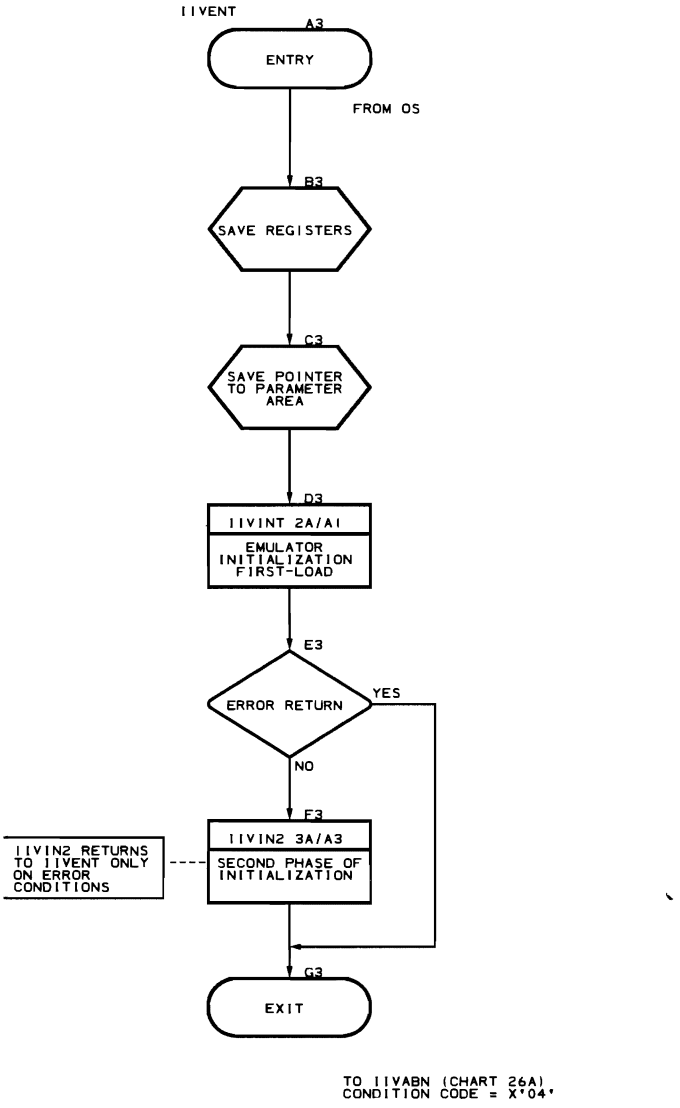
Return codes: X'04' on exit to IIVABN

Tables/work areas: None

Errors detected: None

Messages requested: None

Flowchart 1A. DOS Emulator Entry Routine (IIVENT)



Initialization First-Load Routine (Flowcharts 2A-2M)

Module name: IIVINT

Entry point name: IIVINT

Major functions:

- Verifies parameters
- Establishes DOS storage area
- Initializes COMTAB
- Initializes COMTAB extension

Entered by: IIVENT

Modules called:

- IIVGET
- IIVMSG

Exits to:

- Caller
- IIVABN

OS macros issued:

- WAIT
- EXTRACT
- SAVE
- RDJFCB
- RETURN

Input: Register 7 points to the user parameter area

Output:

- Register 9 points to local execution list
- Register 10 points to DOS storage area
- Register 11 points to IIVCON

Return codes: None

Tables/work areas:

- TIOT
- COMTAB
- UCB
- JFCB
- Local execution list
- Adjust CCW data address list
- EMUCONS
- COMTAB extension

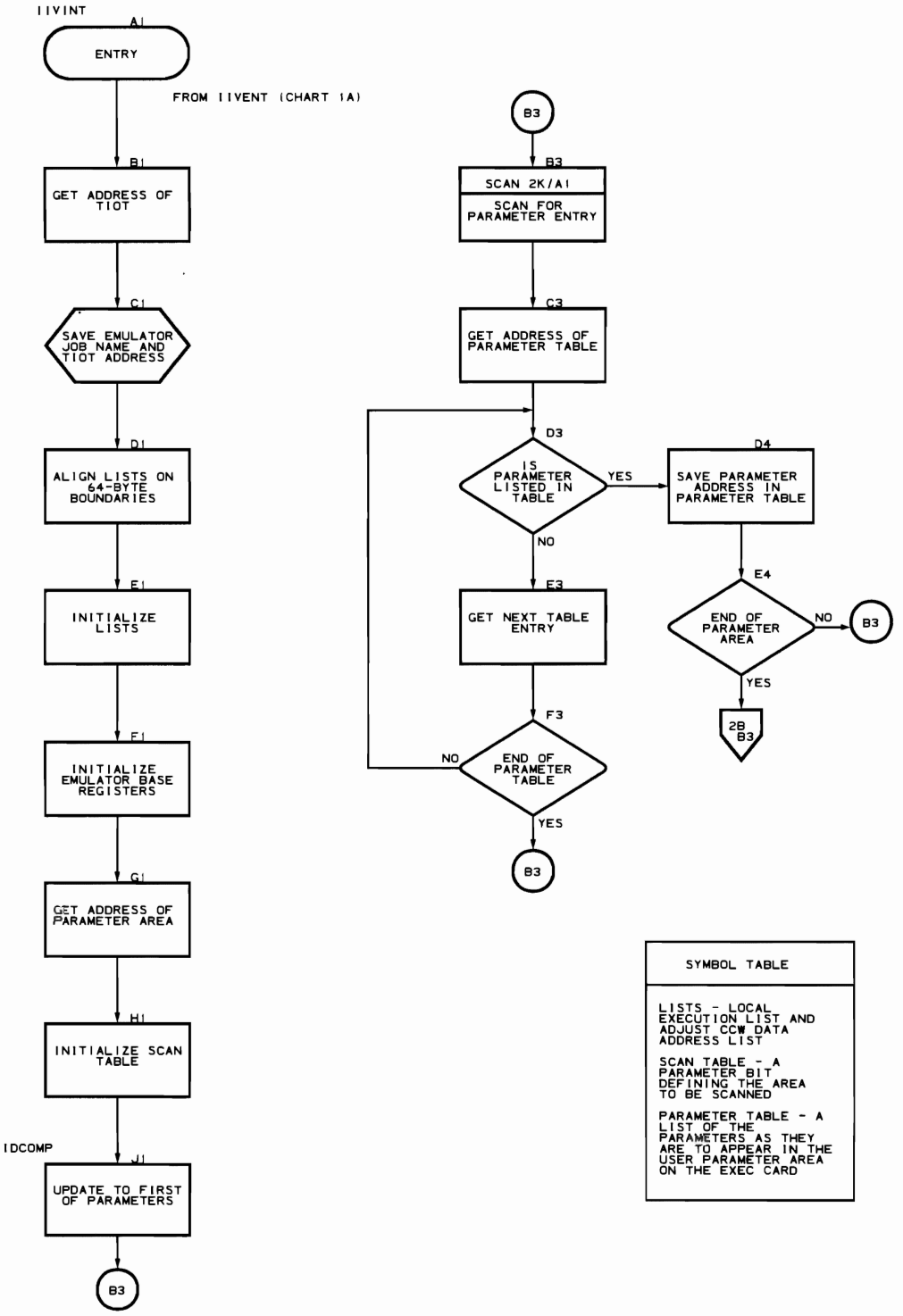
Errors detected:

- Invalid Emulator parameters
- No DOSRES DD statement
- Invalid device assigned to Emulator
- Invalid Emulator DDname
- DISP=SHR specified in SYSEMLBL DD statement
- OS cuu from SYSEMLBL not the same as OS cuu from DOS system residence file

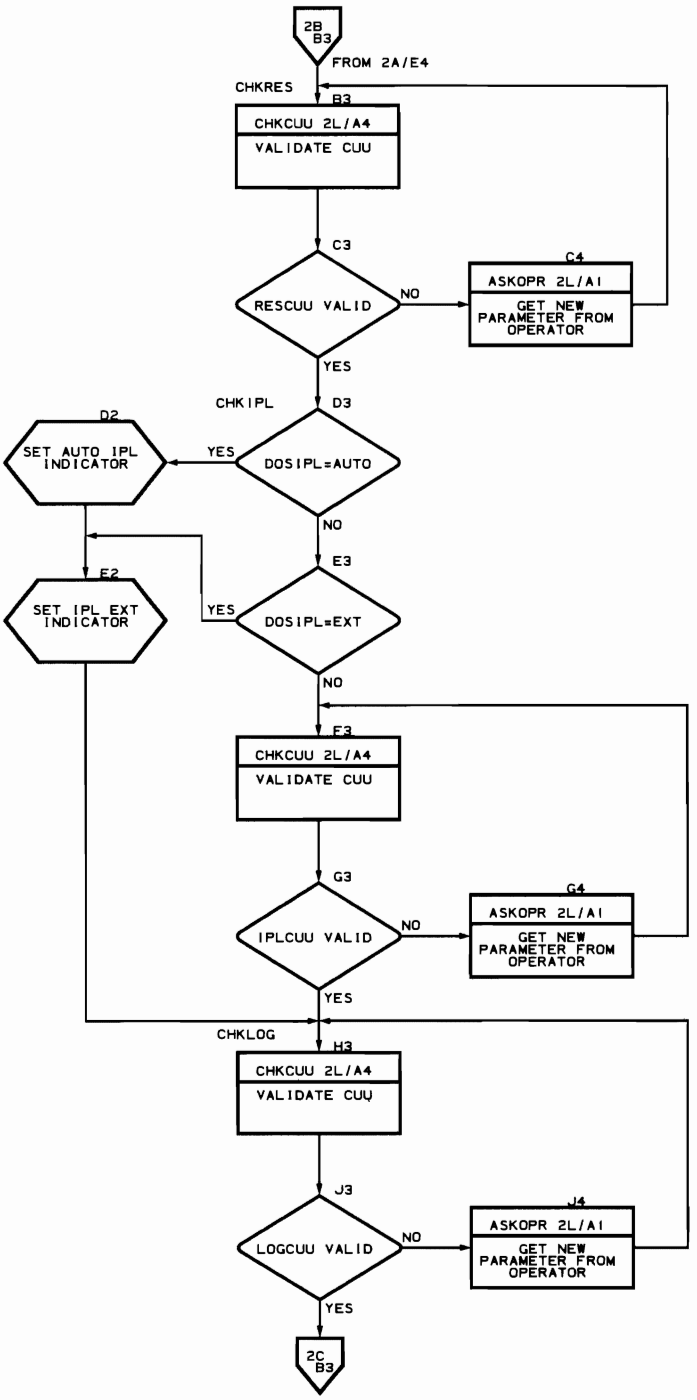
Messages requested:

- IIV002D
- IIV003D
- IIV004D
- IIV005D
- IIV006D
- IIV007D
- IIV008D
- IIV011I
- IIV019I
- IIV022I
- IIV025I
- IIV026I

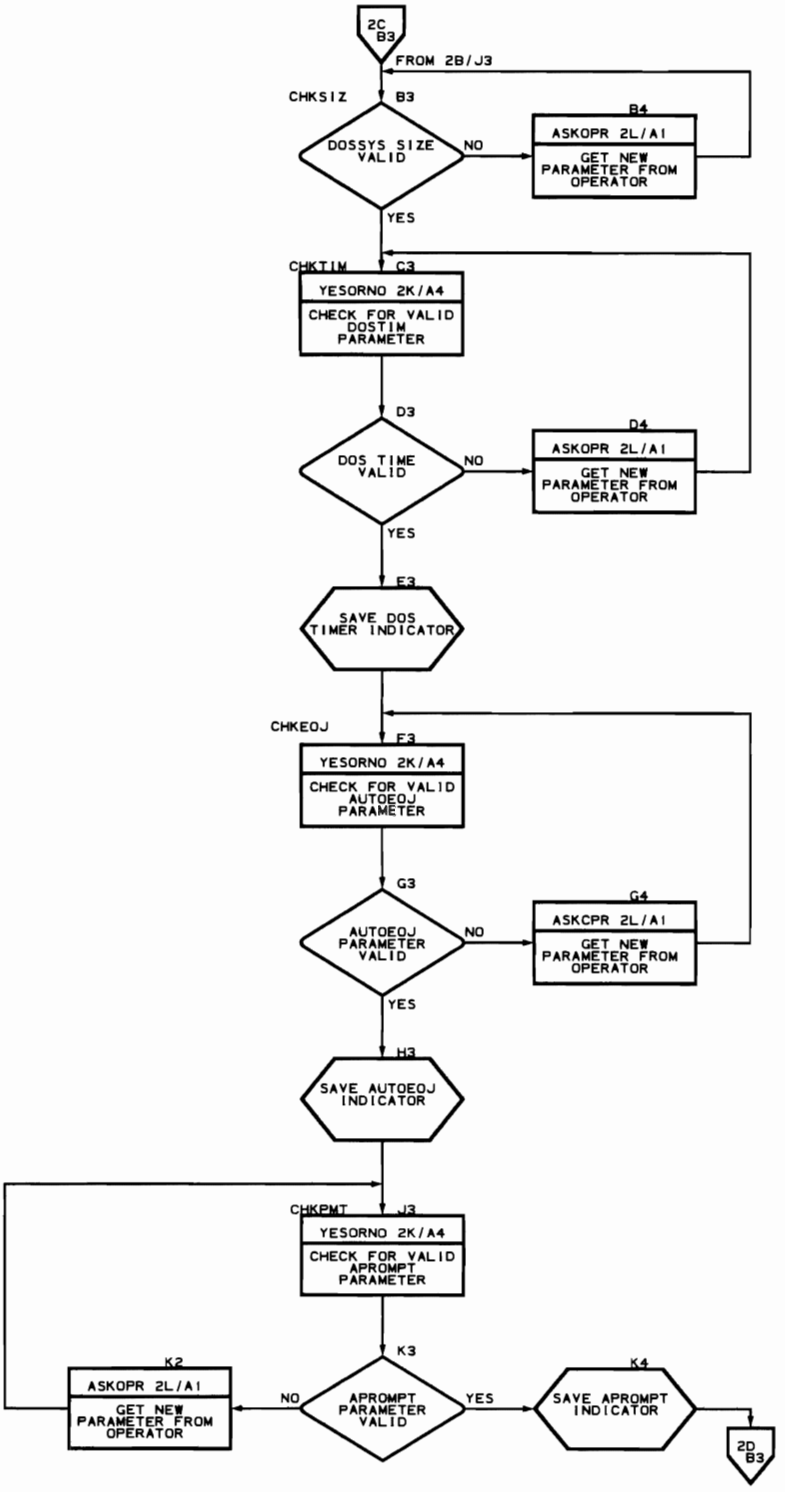
Flowchart 2A. Initialization First-Load Routine (IIVINT Part 1 of 12)



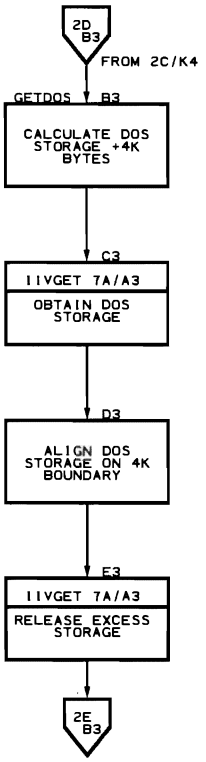
Flowchart 2B. Initialization First-Load Routine (IIVINT Part 2 of 12)



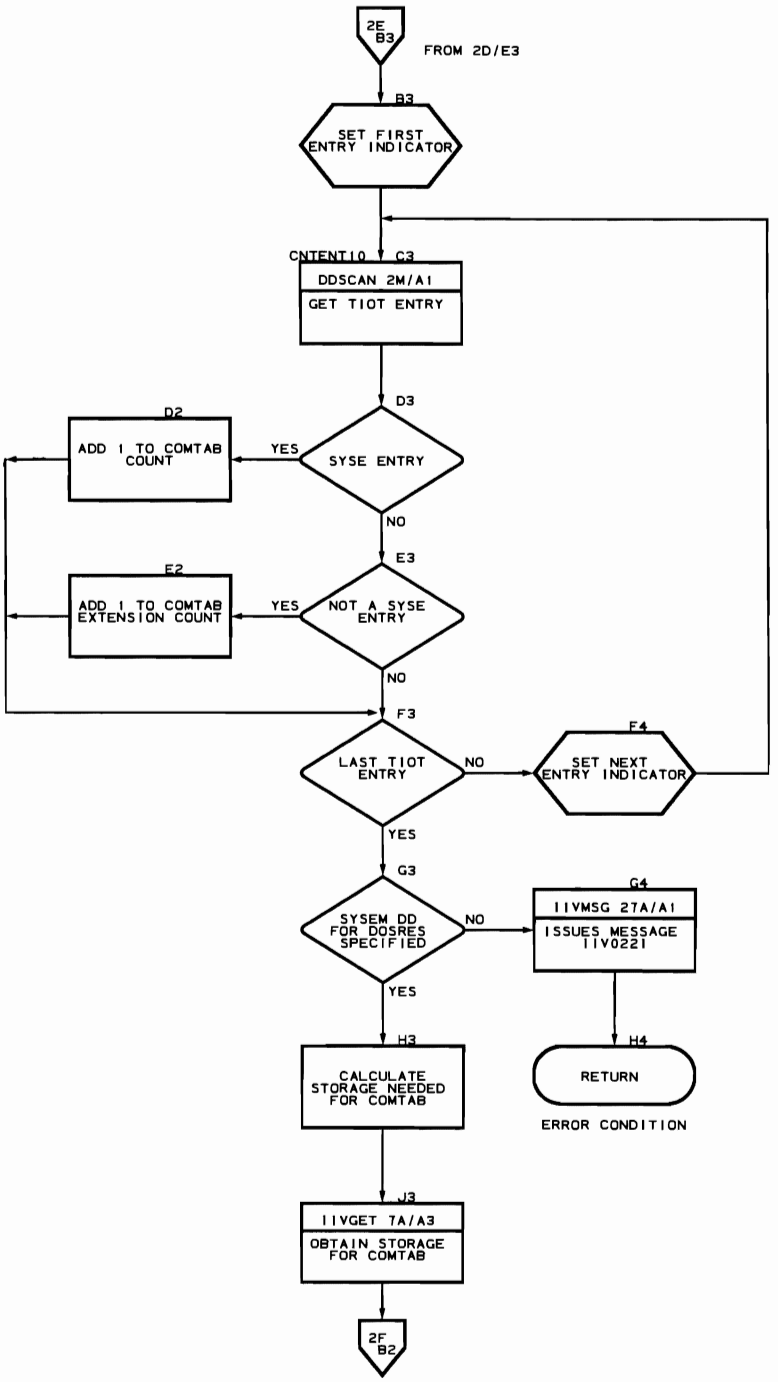
Flowchart 2C. Initialization First-Load Routine (IIVINT Part 3 of 12)



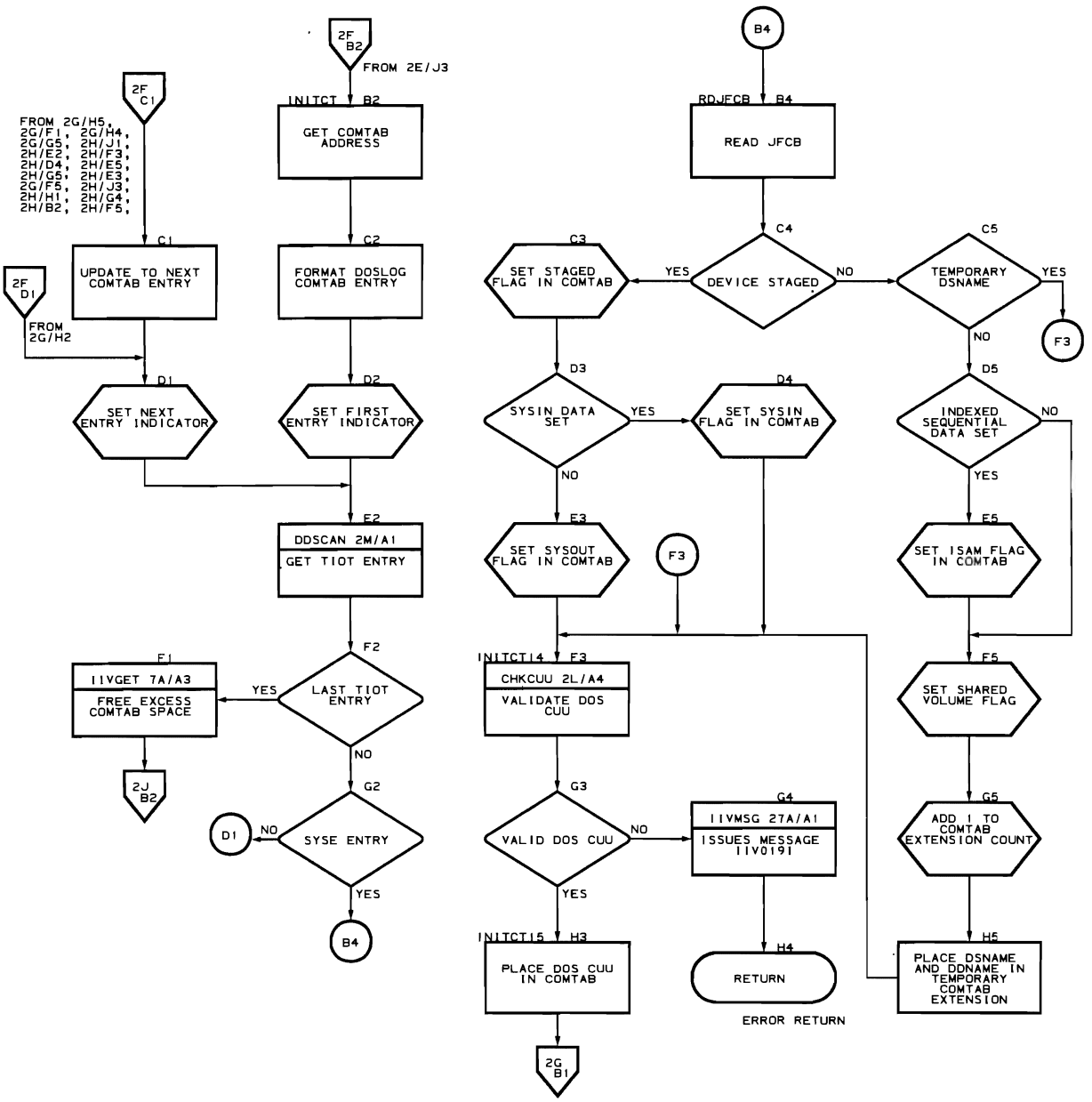
Flowchart 2D. Initialization First-Load Routine (IIVINT Part 4 of 12)



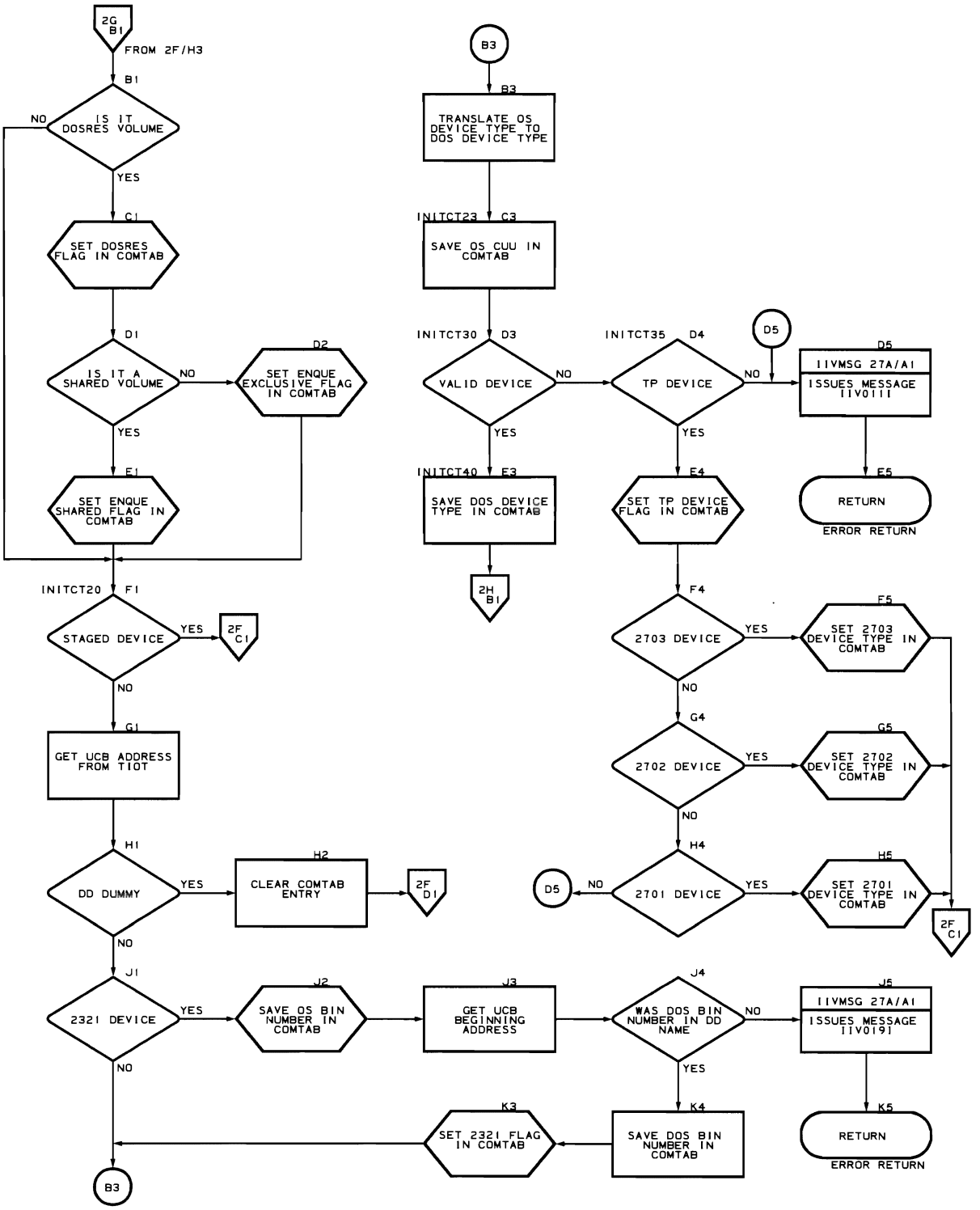
Flowchart 2E. Initialization First-Load Routine (IIVINT Part 5 of 12)



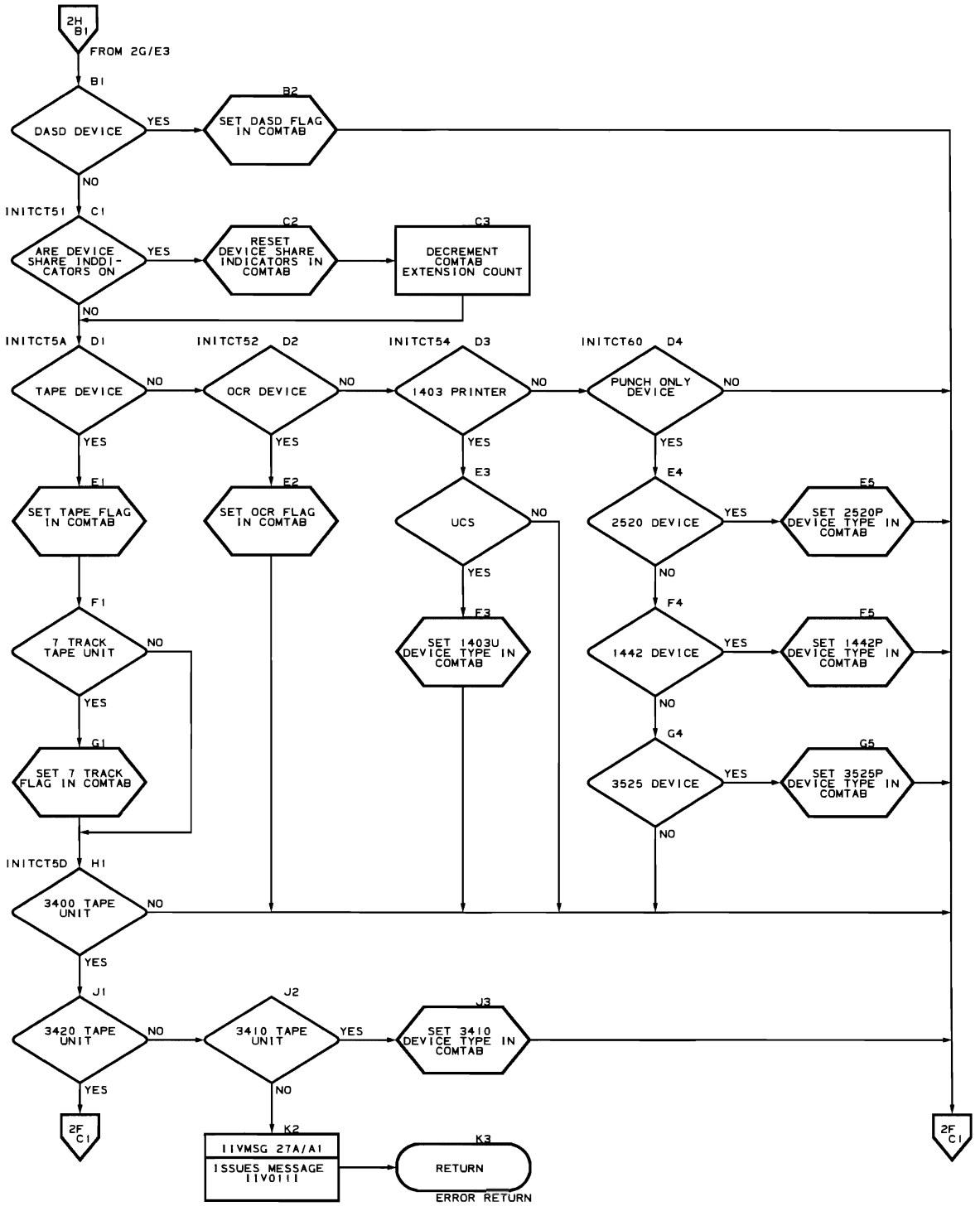
Flowchart 2F. Initialization First-Load Routine (IIVINT Part 6 of 12)



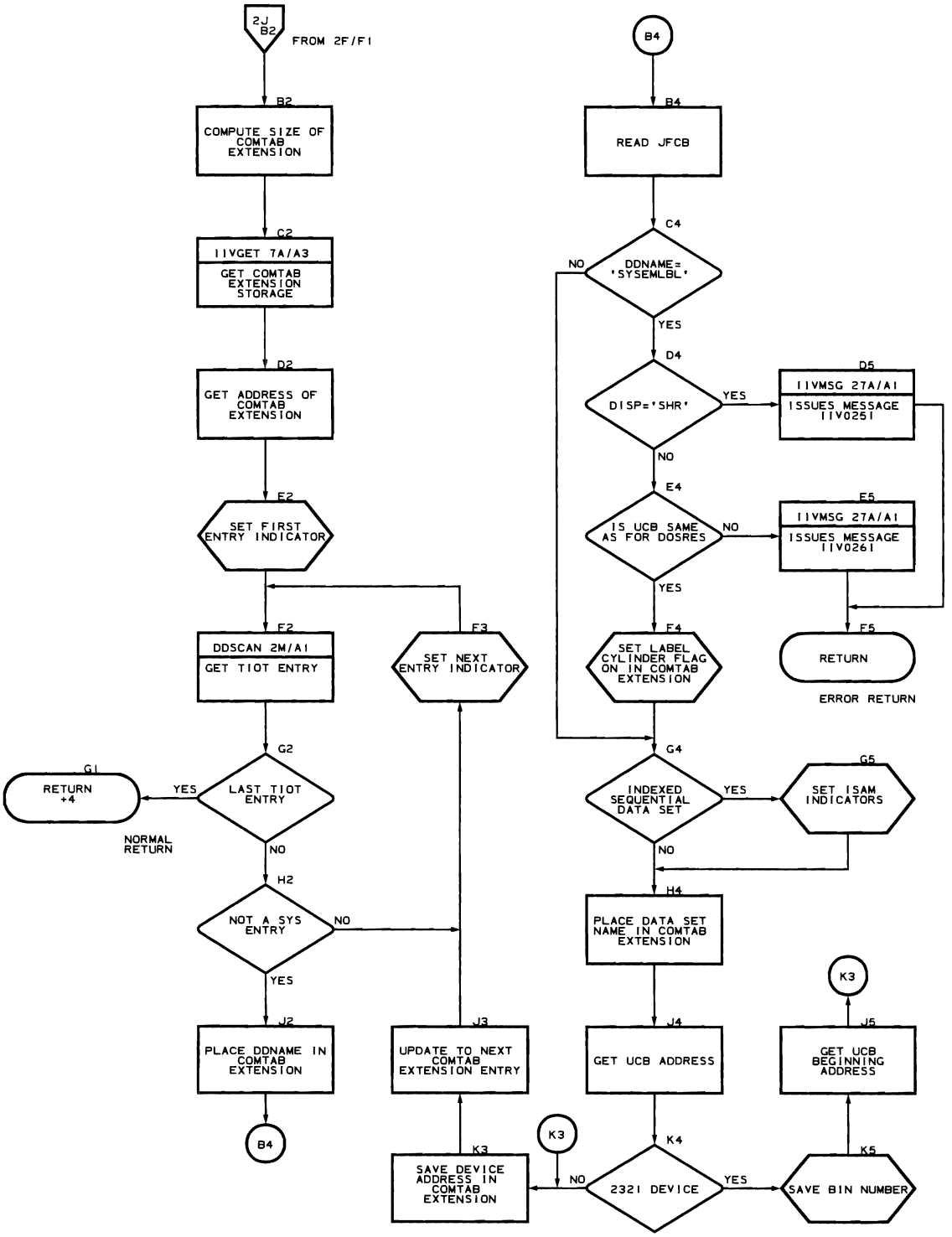
Flowchart 2G. Initialization First-Load Routine (IIVINT Part 7 of 12)



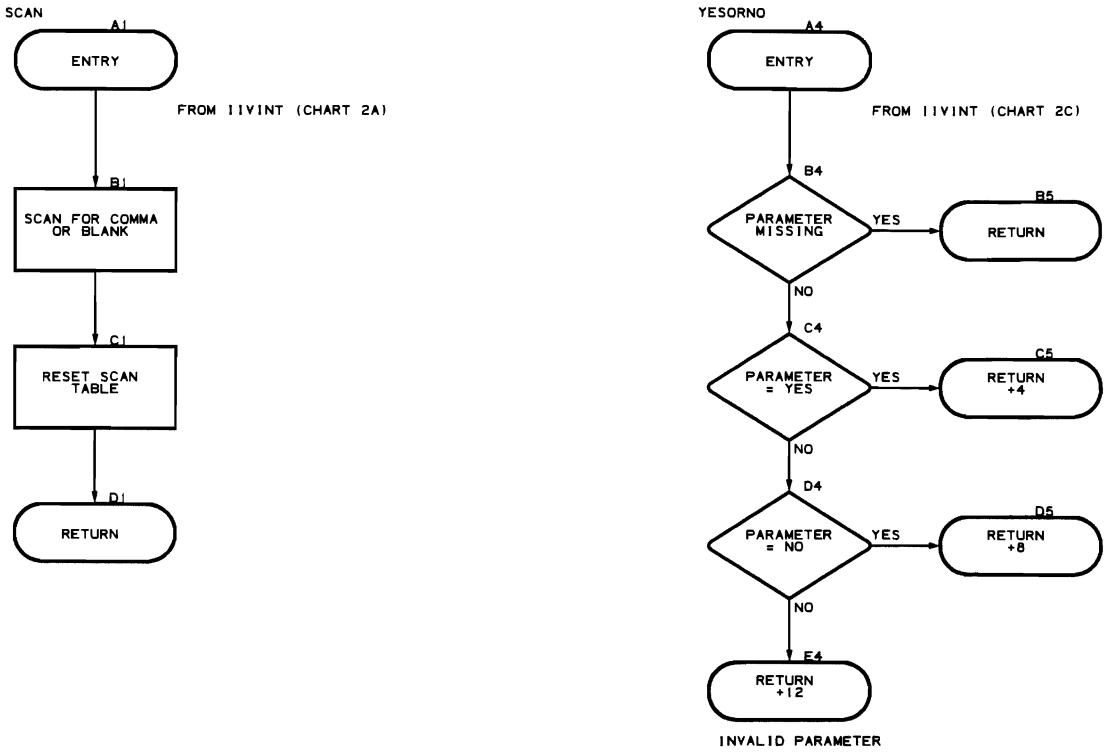
Flowchart 2H. Initialization First-Load Routine (IIVINT Part 8 of 12)



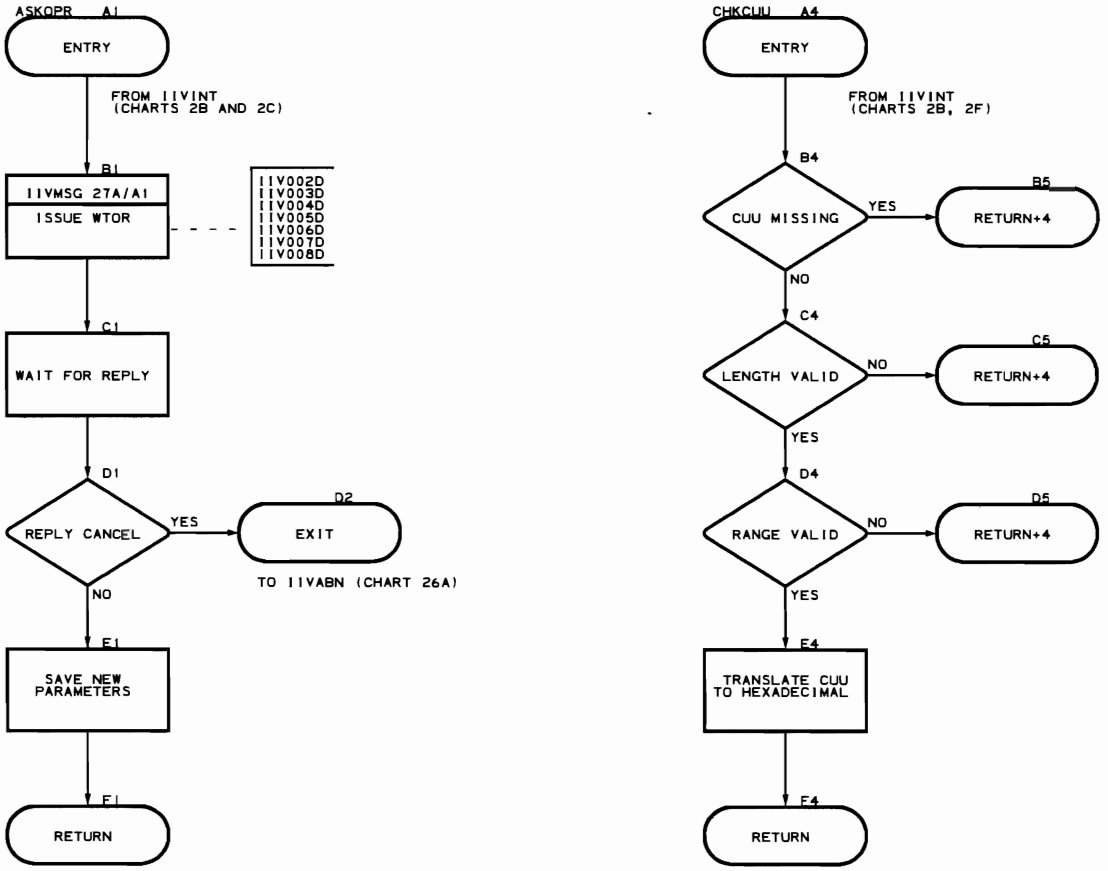
Flowchart 2J. Initialization First-Load Routine (IIVINT Part 9 of 12)



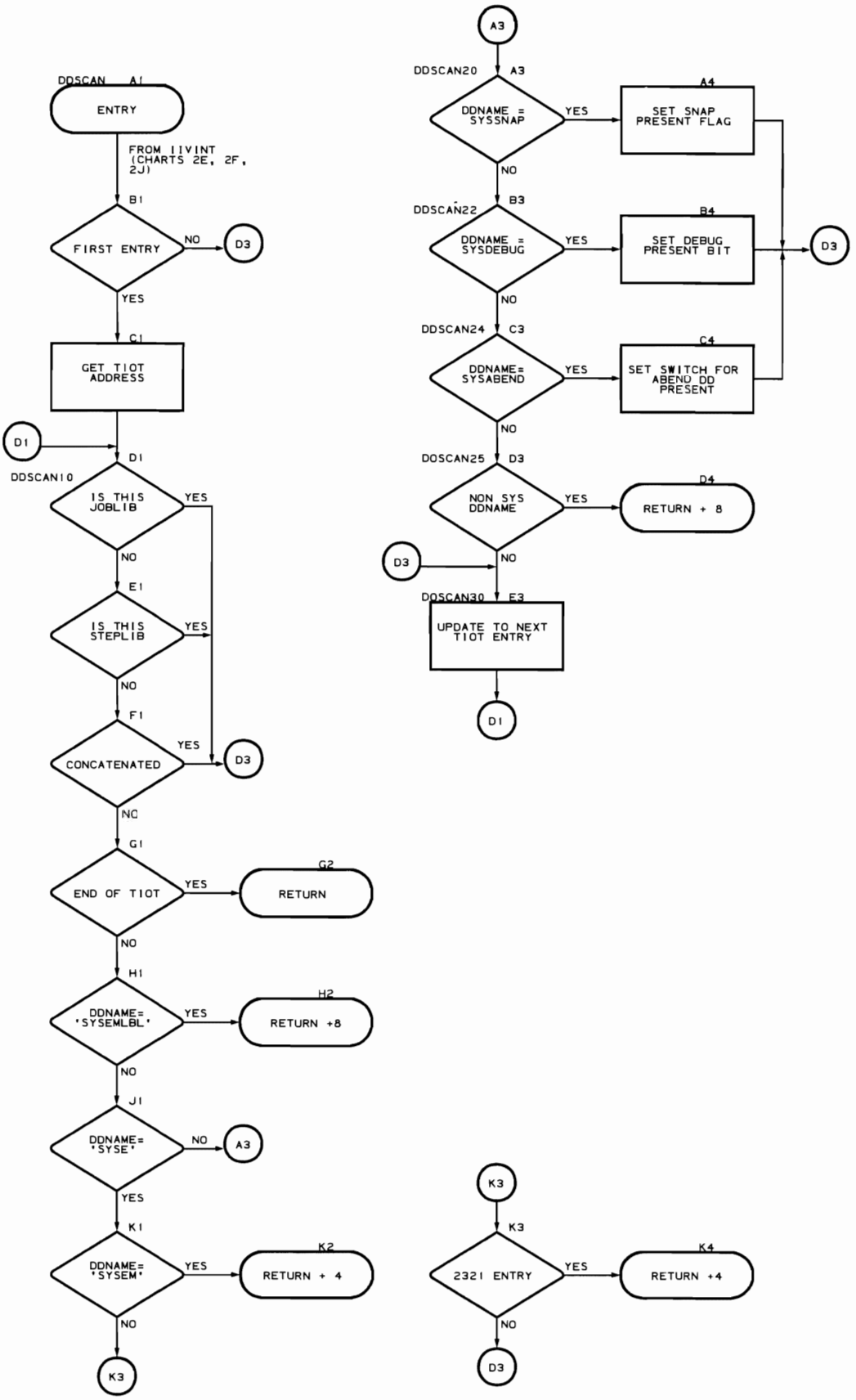
Flowchart 2K. SCAN and YESORNO Subroutines (IIVINT Part 10 of 12)



Flowchart 2L. ASKOPR and CHKCUU Subroutines (IIVINT Part 11 of 12)



Flowchart 2M. DDSCAN Subroutine (IIVINT Part 12 of 12)



Initialization Second-Load Routine (Flowcharts 3A-3F)

Module name: IIVIN2

Entry point name: IIVIN2

Major functions:

- Performs COMTAB and COMTAB extension chaining
- Sorts COMTAB table
- Builds other Emulator tables
- IPLs DOS from the DOS resident volume
- Saves store CPU ID results (STIDP)

Entered by: IIVENT

Modules called:

- IIVOPN
- IIVGET
- IIVMSG
- IIVRAS
- IIVRTE

Exits to:

- IIVRTE
- Returns to caller on error conditions

OS macros issued:

- EXCP
- ENQ
- WAIT
- LOAD
- DEQ
- SAVE
- RETURN
- DCBD
- OPEN

Input:

- Register 9 points to local execution list
- Register 10 points to DOS storage
- Register 11 points to IIVCON

Output: None

Return codes: None

Tables/work areas:

- COMTAB
- COMTAB extension
- ISK/SSK table
- ECB pointer table
- Post ECB list
- Local execution list
- EMUCONS
- DOS storage

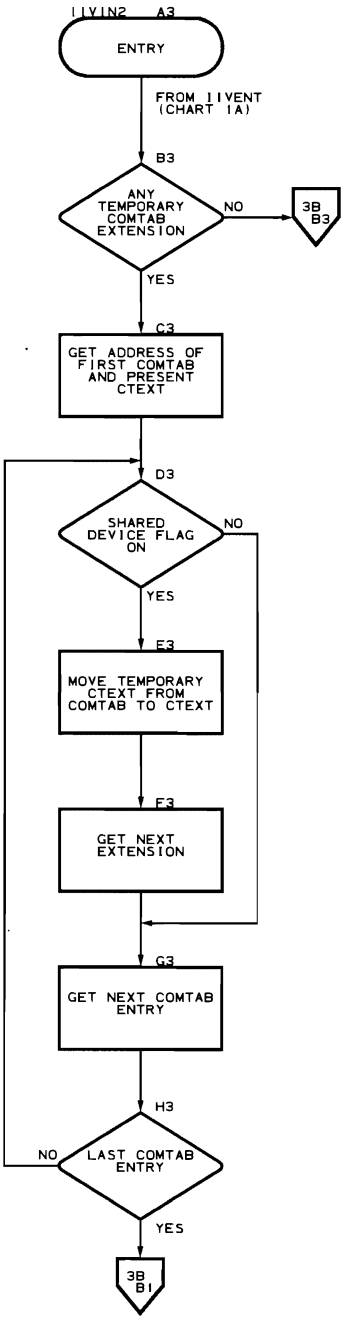
Errors detected:

- Duplicate DOS device address
- Could not IPL DOS for DOS emulation
- Could not open DOS SYSRES DCB
- I/O error on DOS SYSRES
- Missing DD statement for DOS SYSRES volume
- DDname does not map to a DOS device
- Invalid device sharing request
- Invalid starting address for DOS SYSRES
- DOSRES label cylinder in use

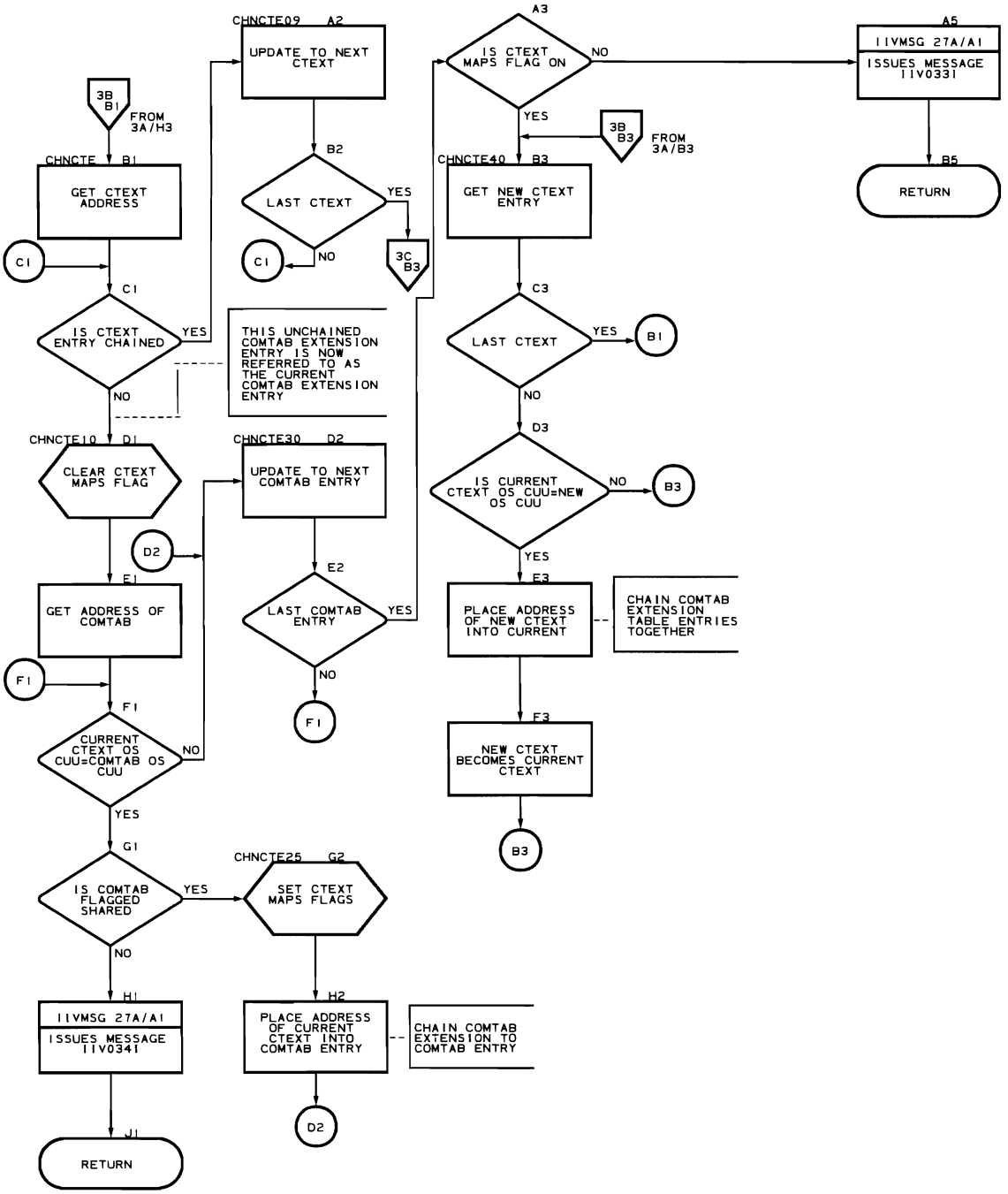
Messages requested:

- IIV012I
- IIV013I
- IIV015I
- IIV020I
- IIV022I
- IIV027I
- IIV028I
- IIV033I
- IIV034I

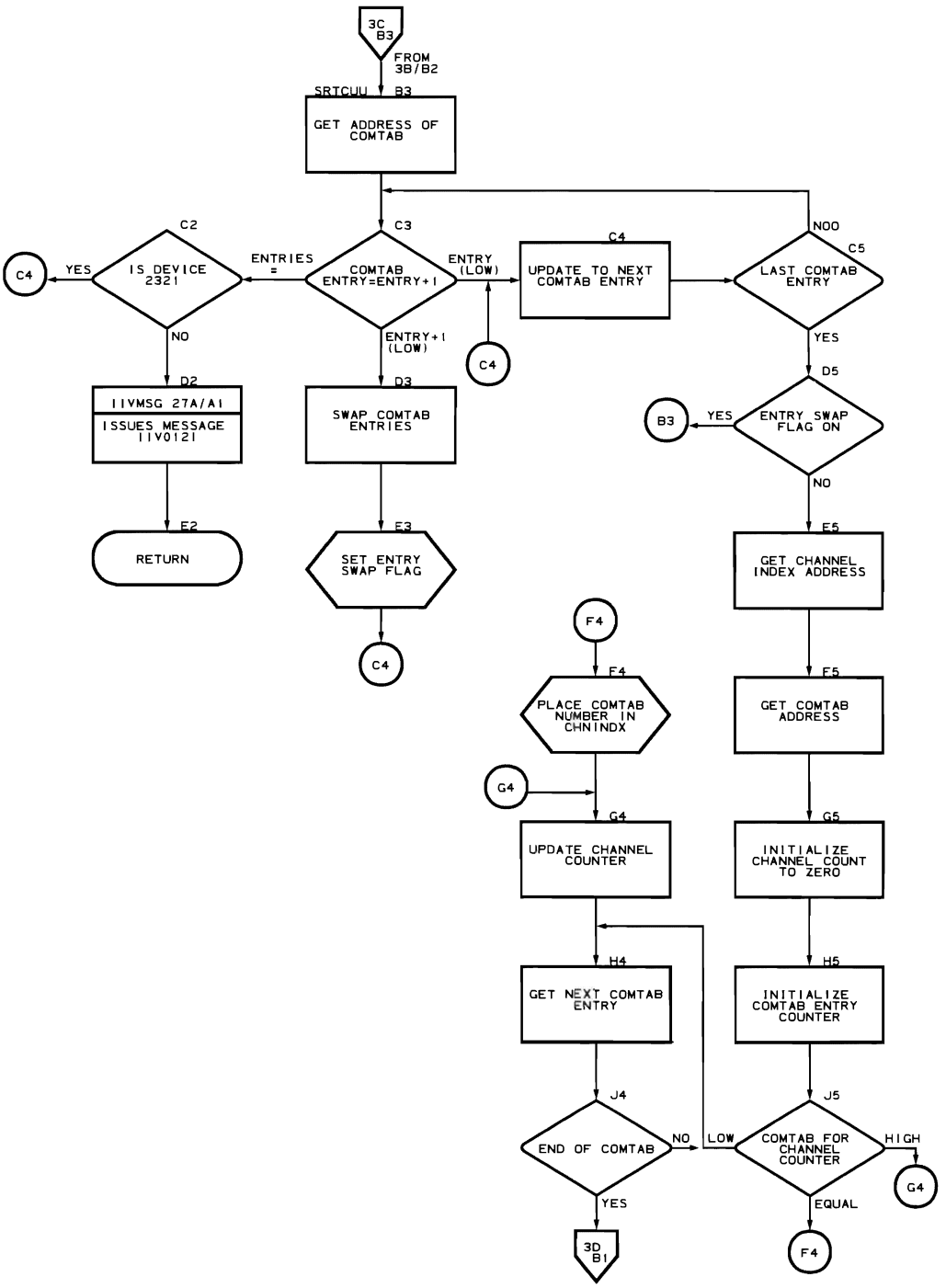
Flowchart 3A. Initialization Second-Load Routine (IIVIN2 Part 1 of 6)



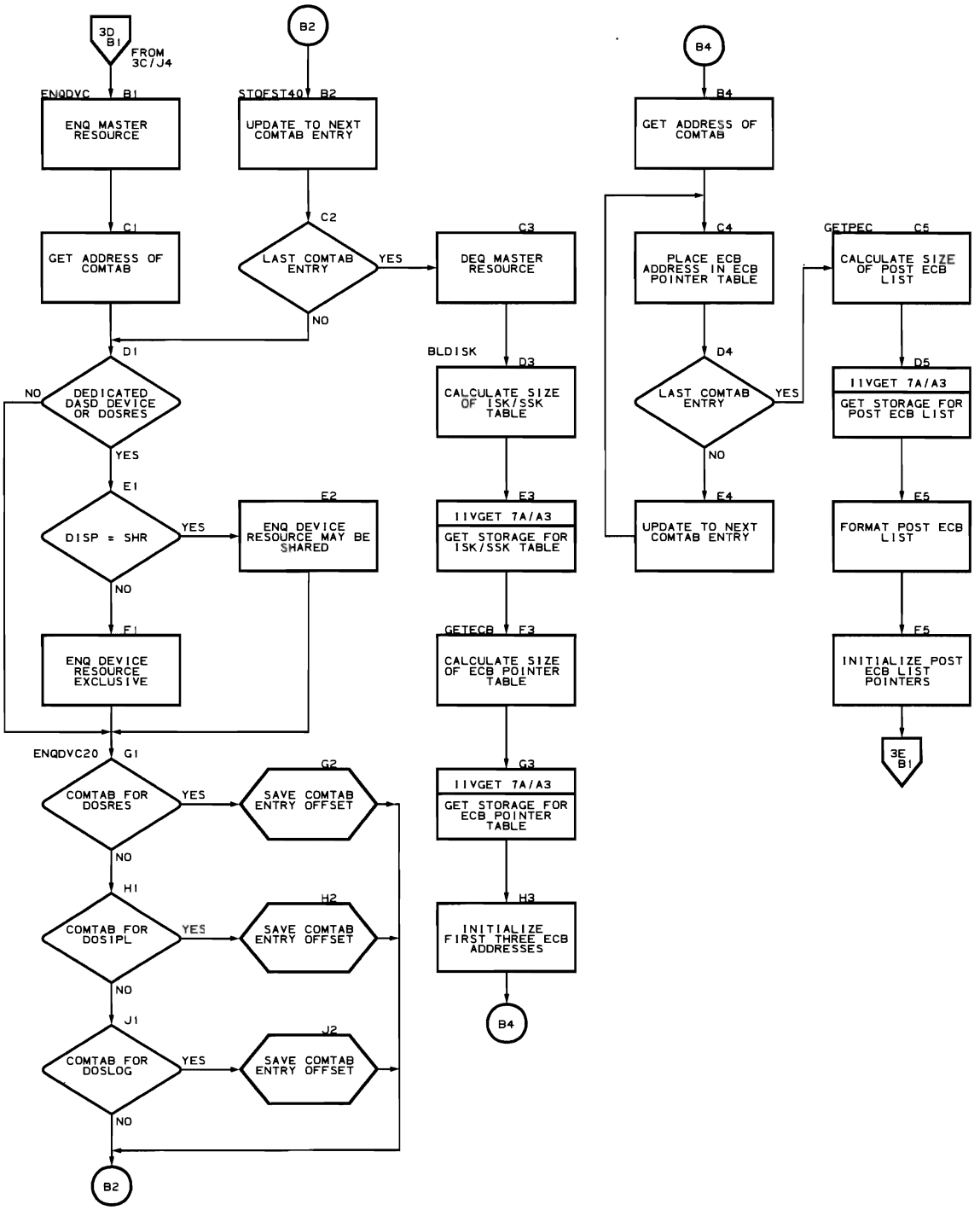
Flowchart 3B. Initialization Second-Load Routine (IIVIN2 Part 2 of 6)



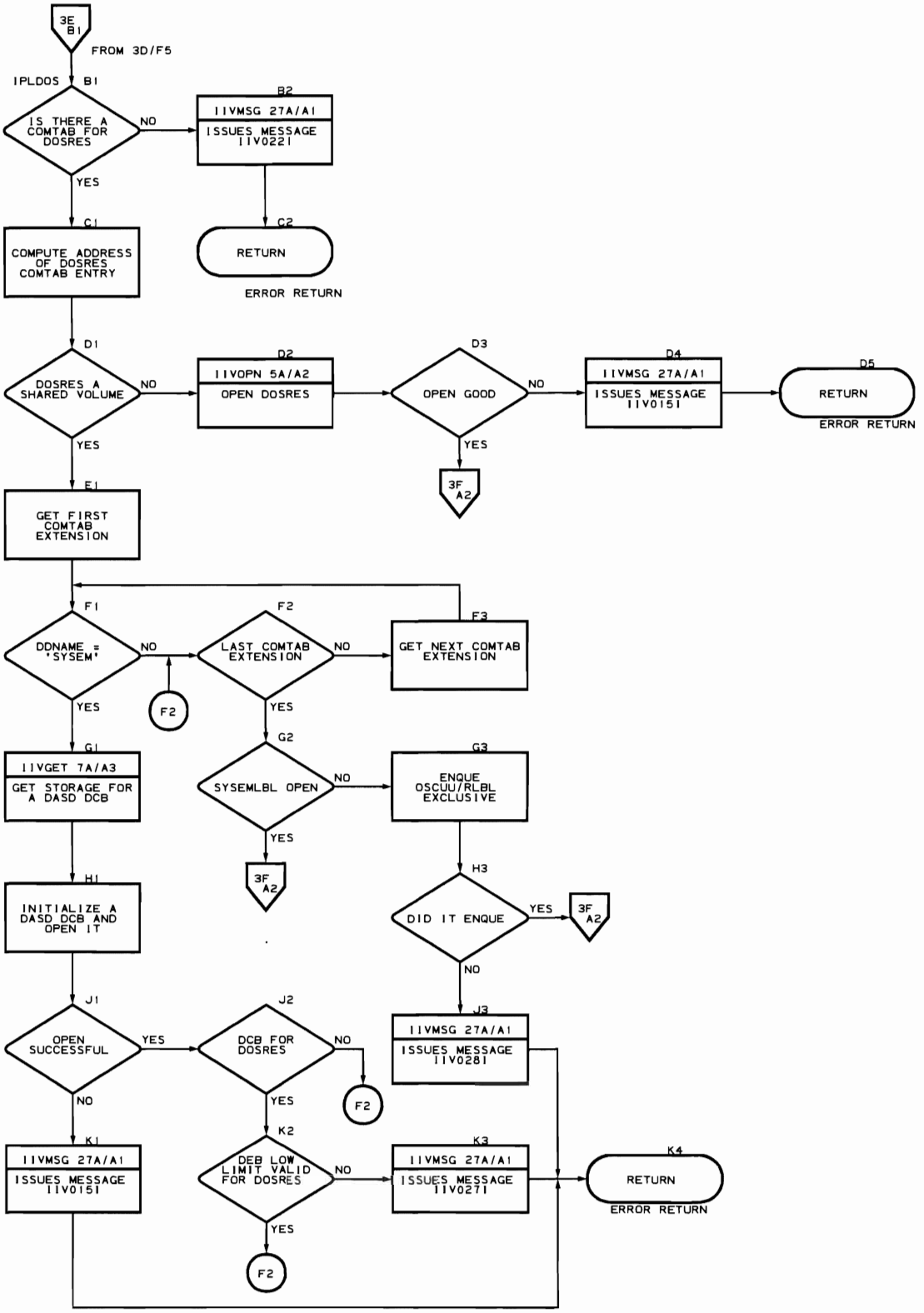
Flowchart 3C. Initialization Second-Load Routine (IIVIN2 Part 3 of 6)



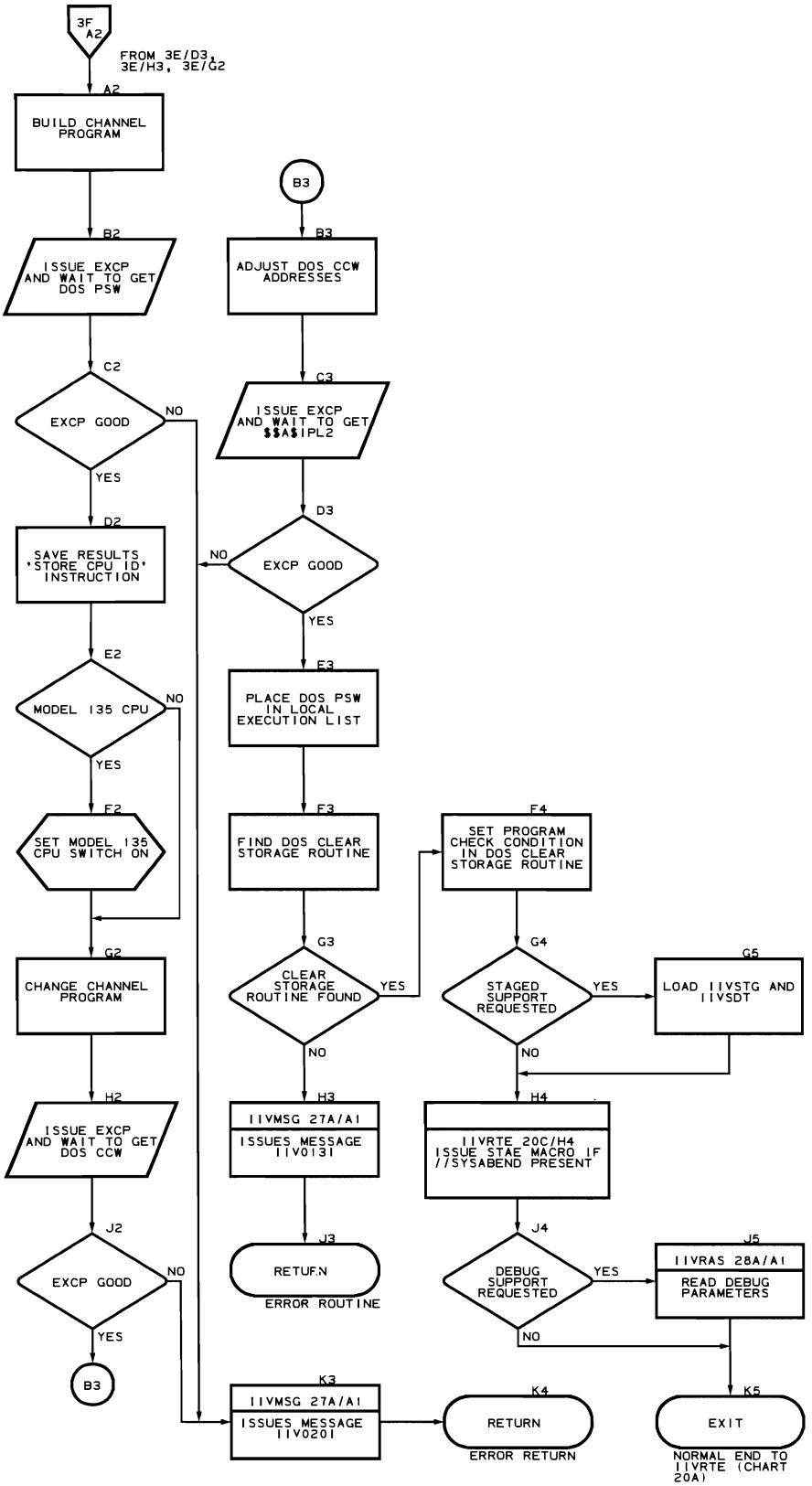
Flowchart 3D. Initialization Second-Load Routine (IIVIN2 Part 4 of 6)



Flowchart 3E. Initialization Second-Load Routine (IIVIN2 Part 5 of 6)



Flowchart 3F. Initialization Second-Load Routine (IIVIN2 Part 6 of 6)



IPL Add Routine (Flowcharts 4A-4D)

Module name: IIVADD

Entry point name: IIVADD

Major functions: Performs automatic add, delete, set date and time

Entered by: IIVSVC

Modules called: IIVMSG

Exits to: IIVSVC

OS macros issued:

- TIME
- SAVE

Input:

- Register 1 points to the DOS CCW
- Register 9 points to the local execution list
- Register 10 points to DOS storage
- Register 11 points to IIVCON

Output: An add, delete, or set date and time statement is placed in the DOS input area.

Return codes: None

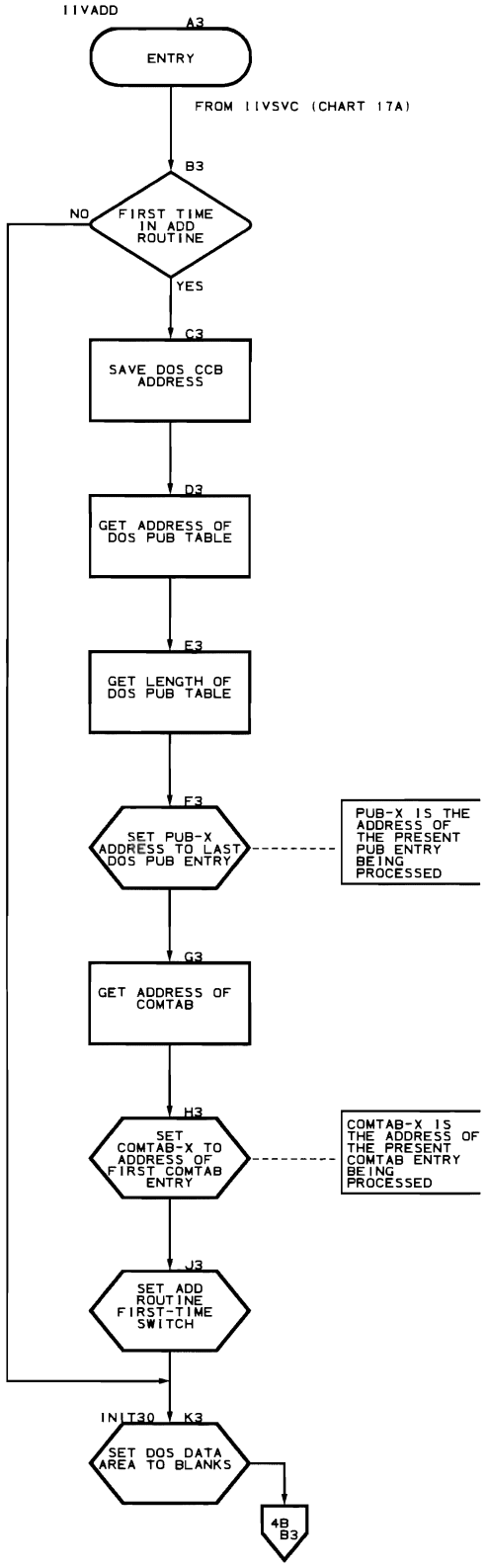
Tables/work areas:

- DOS PUB
- COMTAB
- EMUCONS

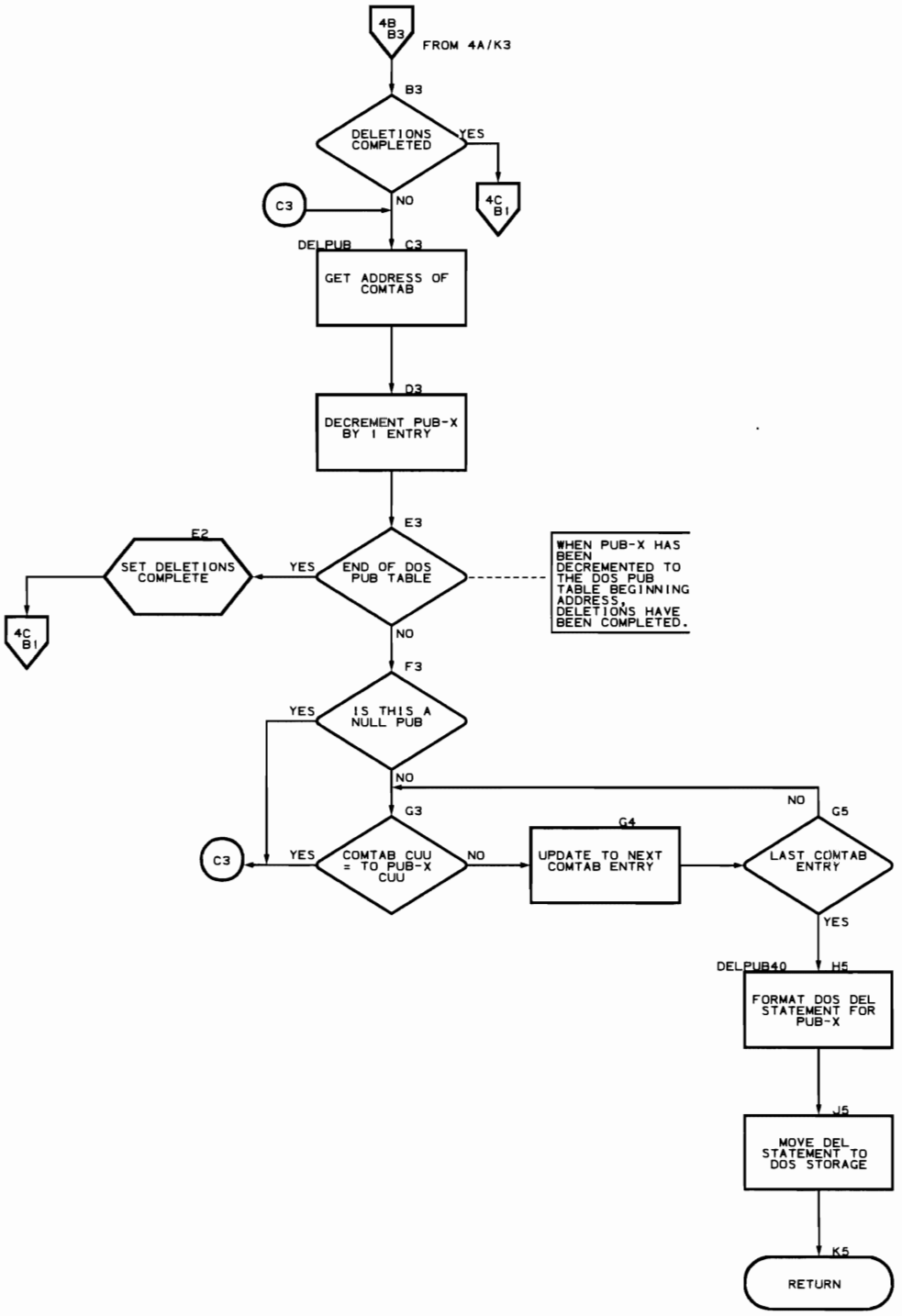
Errors detected: Invalid automatic add reqeust

Messages requested: IIV032I

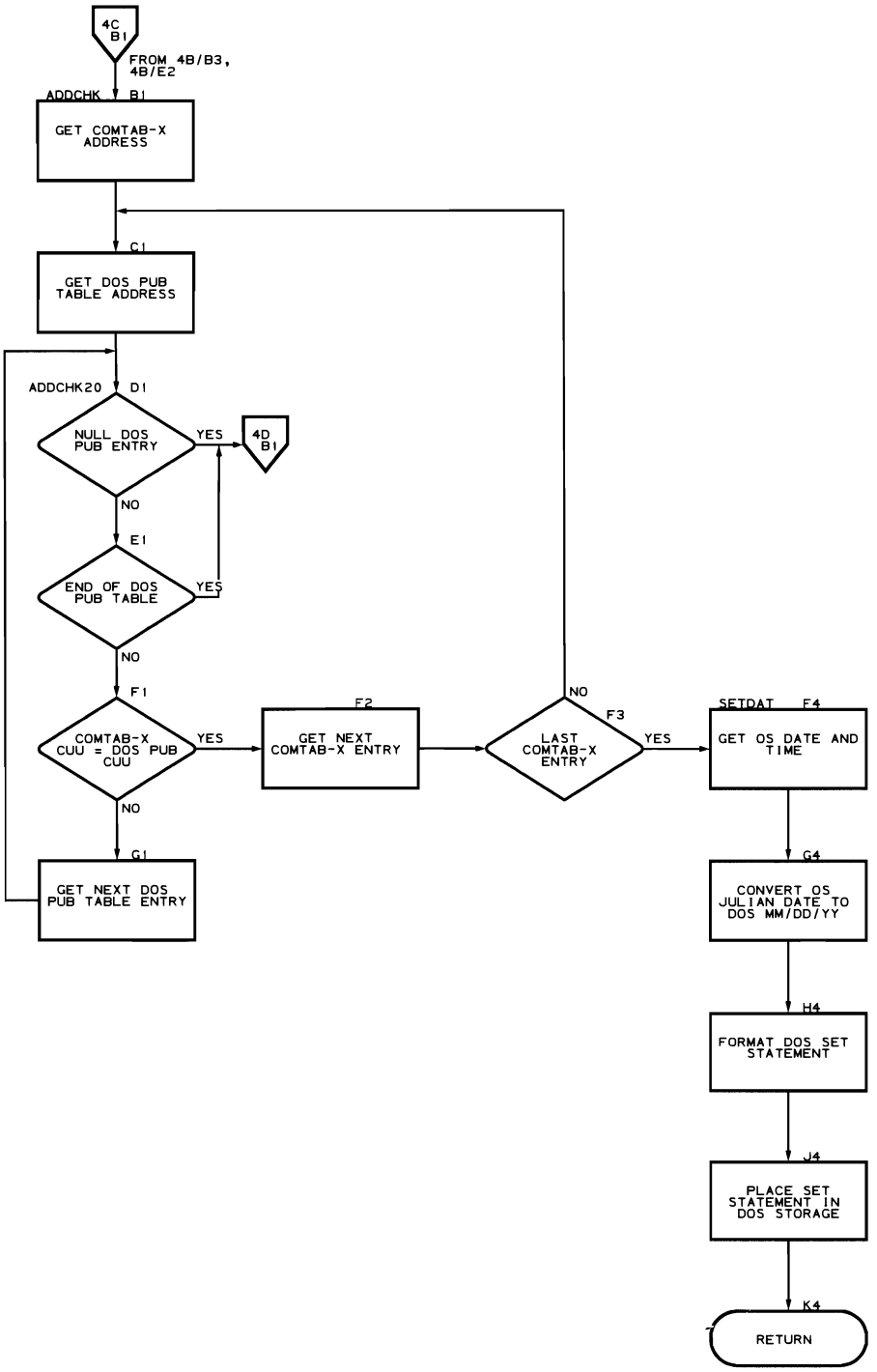
Flowchart 4A. IPL Add Routine (IIVADD Part 1 of 4)



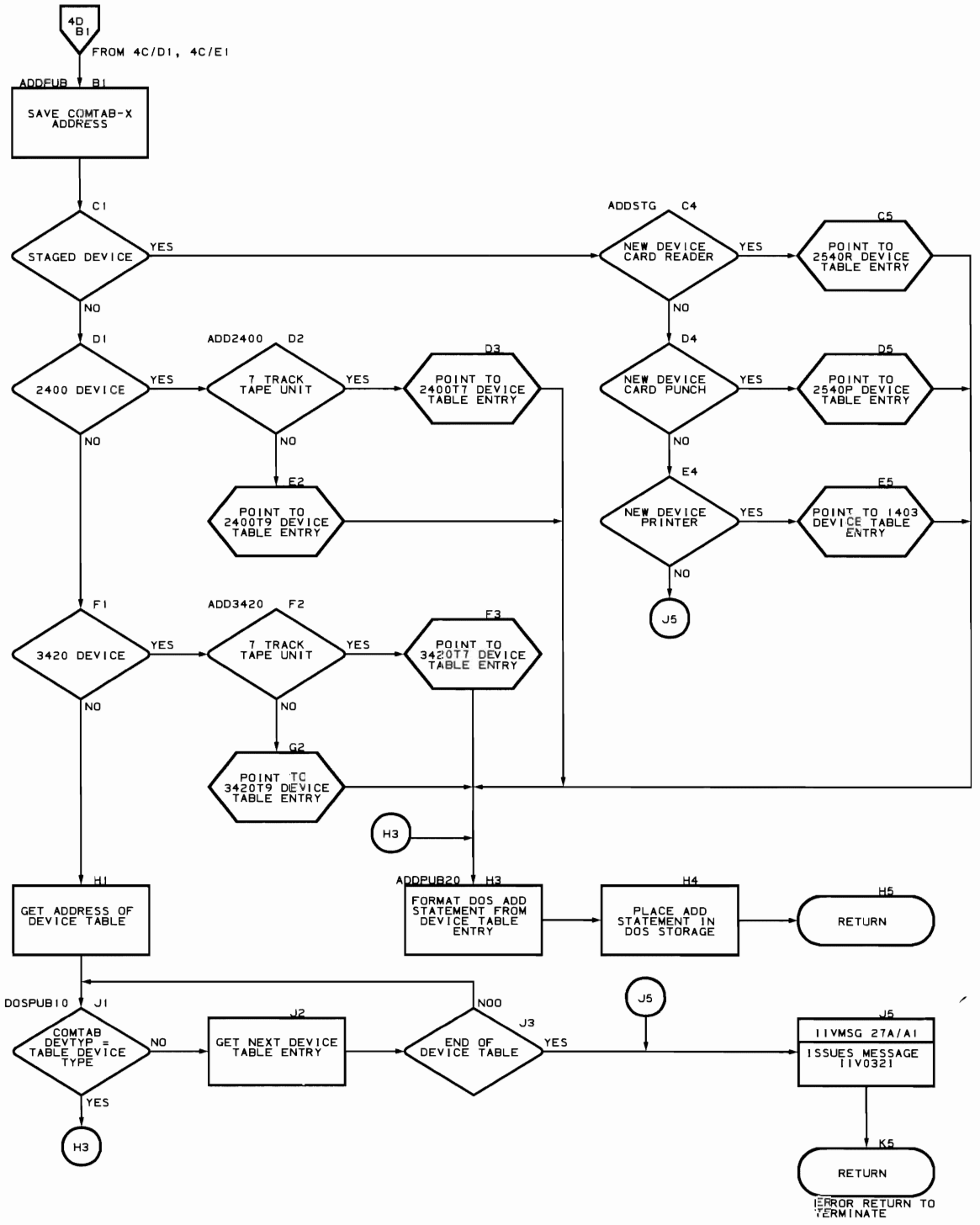
Flowchart 4B. IPL Add Routine (IIVADD Part 2 of 4)



Flowchart 4C. IPL Add Routine (IIVADD Part 3 of 4)



Flowchart 4D. IPL Add Routine (IIVADD Part 4 of 4)



Open Routine (Flowcharts 5A-5D)

Module name: IIVOPN

Entry point name: IIVOPN, OPEN95

Major functions: Opens Emulator-allocated dedicated resources dynamically

Entered by:

- IIVPCE
- IIVIN2
- IIVPRP
- IIVRTE

Modules called:

- IIVGET
- IIVMSG

Exits to:

- IIVABN
- Returns to caller
- IIVSTG

OS macros issued:

- DCB
- EXCP
- WAIT
- OPEN TYPE=J
- LOAD
- RDJFCB
- IMGLIB
- SAVE
- DELETE
- DCBD

Input:

- Register 0 points to the COMTAB entry to be processed
- Register 9 points to the local execution list
- Register 10 points to DOS storage
- Register 11 points to IIVCON

Output:

- COMTAB CTFLAG bit 7 is set on
- Opened DCB
- Formatted IOB/formatted STGCON

Return codes:

- X'00'=unsuccessful open
- X'04'=successful open
- X'04'=exit to IIVABN

Tables/work areas:

- COMTAB
- IOB
- JFCB
- DEB
- PUB
- EMUCONS
- DOS COMREG
- DOS low storage area
- DSCBs in VTOC

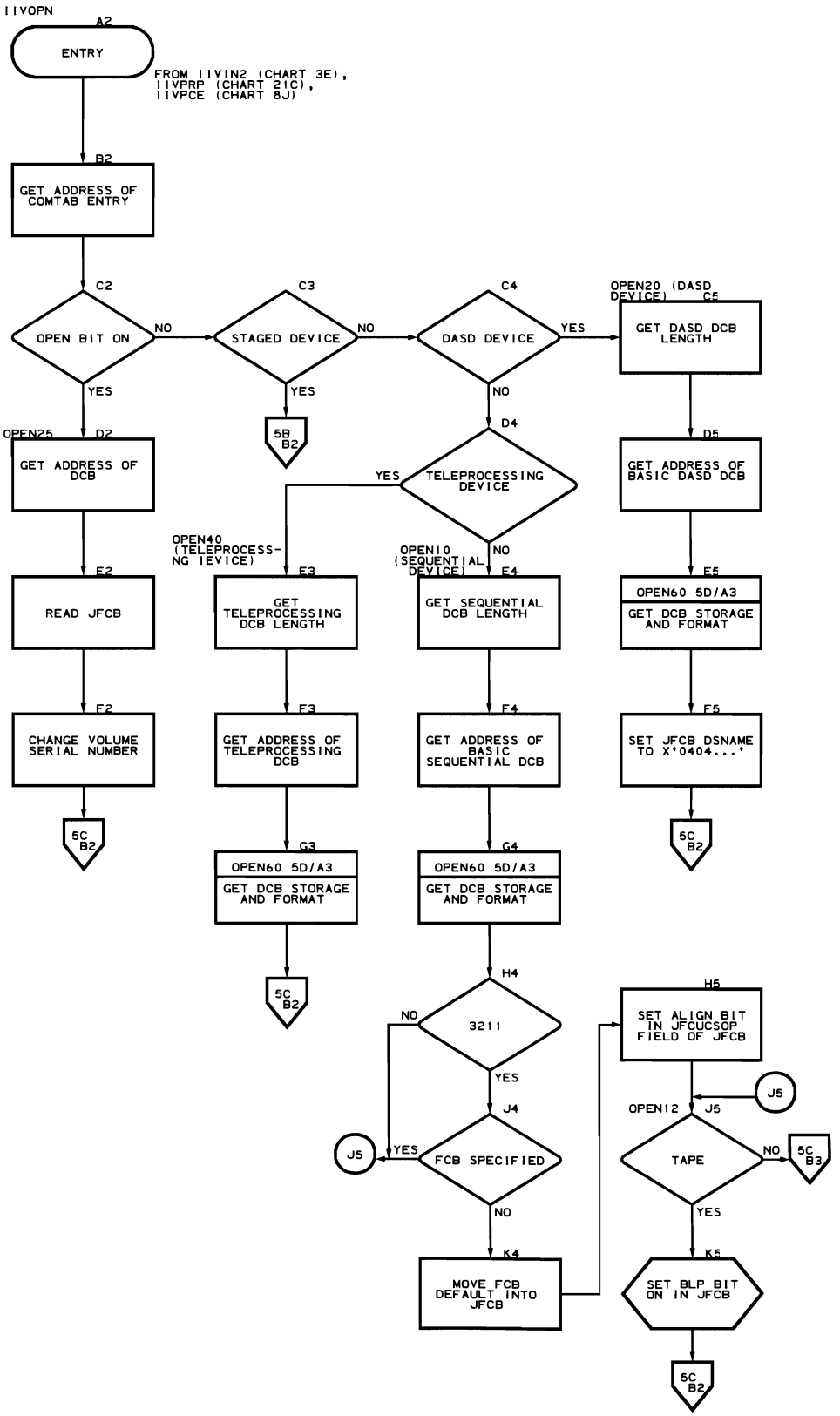
Errors detected:

- Device not stageable
- Open was unsuccessful
- VTOC overlaps cylinder boundary
- Could not find PUB entry in the DOS PUB table during DOS IPL
- Volume contains OS password data sets
- Invalid FCB image
- OPEN for SYS1.IMAGELIB failed

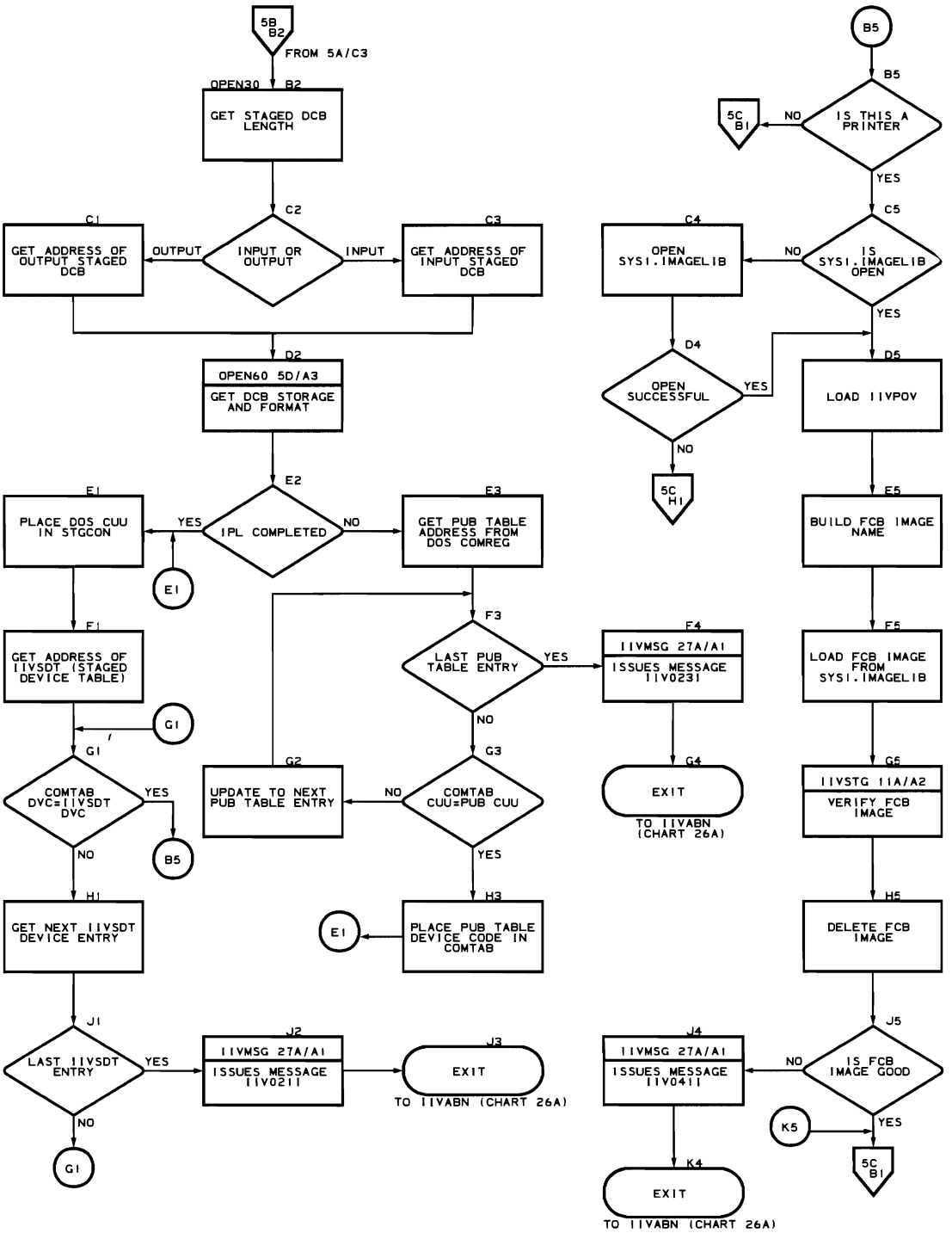
Messages requested:

- IIV018I
- IIV021I
- IIV023I
- IIV024I
- IIV040D
- IIV041I
- IIV050D

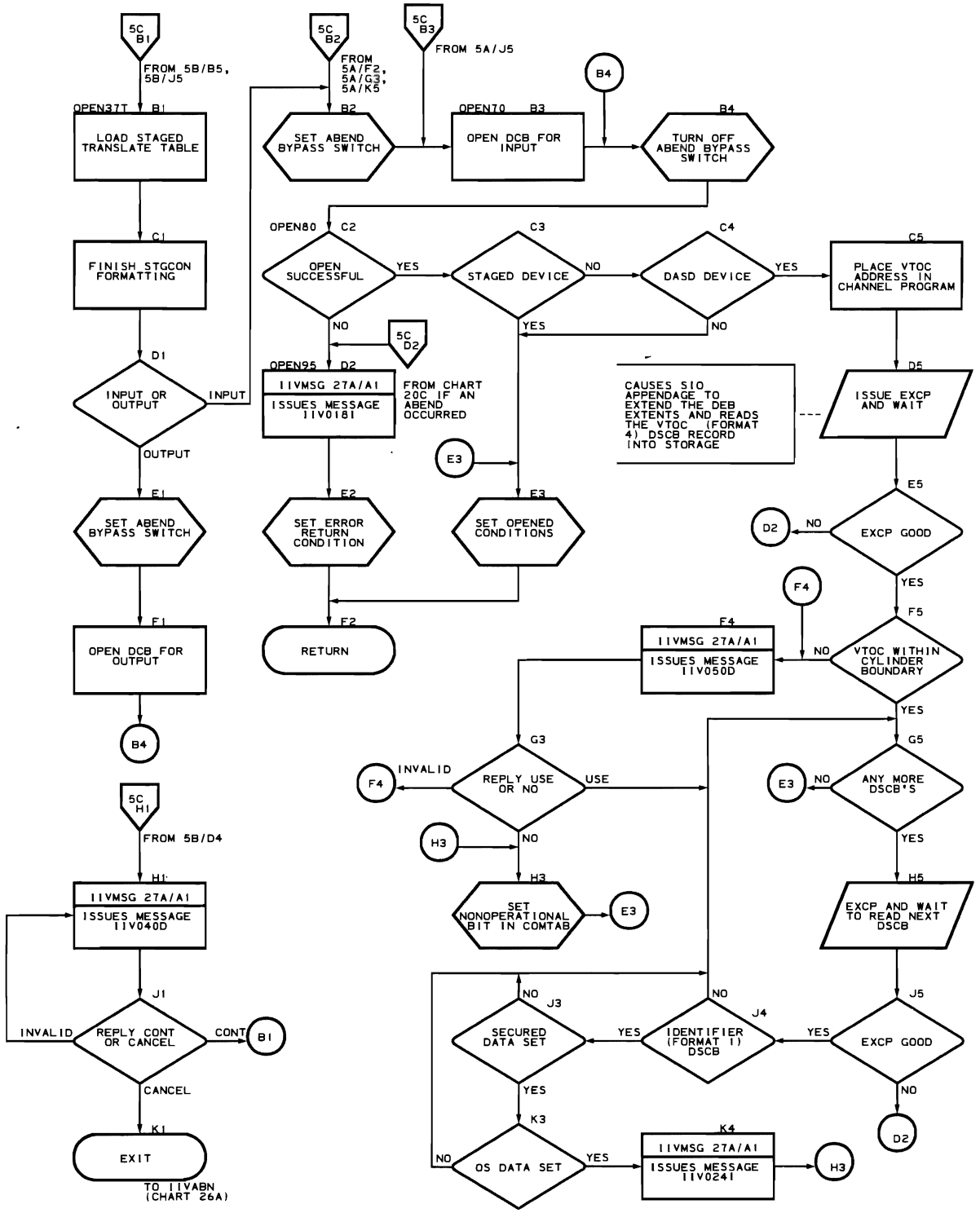
Flowchart 5A. Open Routine (IIVOPN Part 1 of 4)



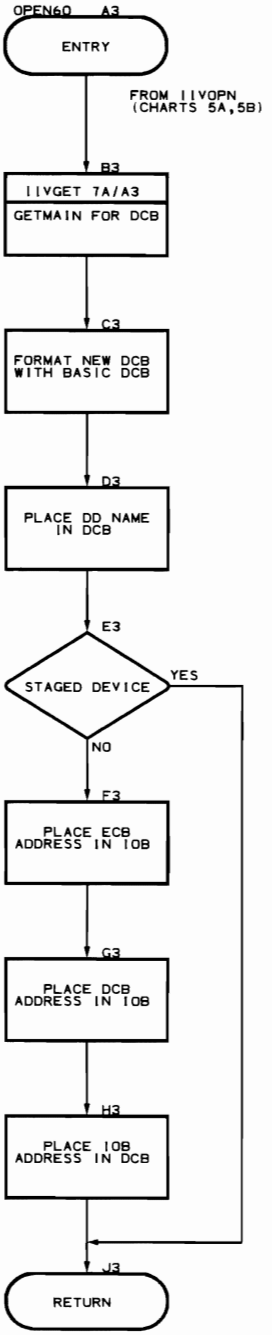
Flowchart 5B. Open Routine (IIVOPN Part 2 of 4)



Flowchart 5C. Open Routine (IIVOPN Part 3 of 4)



Flowchart 5D. OPEN60 Routine (IIVOPN Part 4 of 4)



OS PUB Table Build Routine (Flowcharts 6A-6F)

Module name: IIVPUB

Entry point name: IIVPUB

Major functions:

- Maps DOS I/O control blocks to Emulator I/O control blocks by means of the OS PUB table
- Issues initial Emulator prompt

Entered by: IIVSVC

Modules called: IIVGET, IIVMSG

Exits to:

- Caller
- Return+4 to caller on error condition

OS macros issued:

- LOAD
- STIMER
- SAVE
- RETURN

Input:

- Register 9 points to local execution list
- Register 10 points to DOS storage
- Register 11 points to IIVCON

Output: Initial prompt

Return codes: None

Tables/work areas:

- Local execution list
- EMUCONS
- COMTAB
- DOS communications region
- DOS PUB table
- DOS storage
- OS PUB table

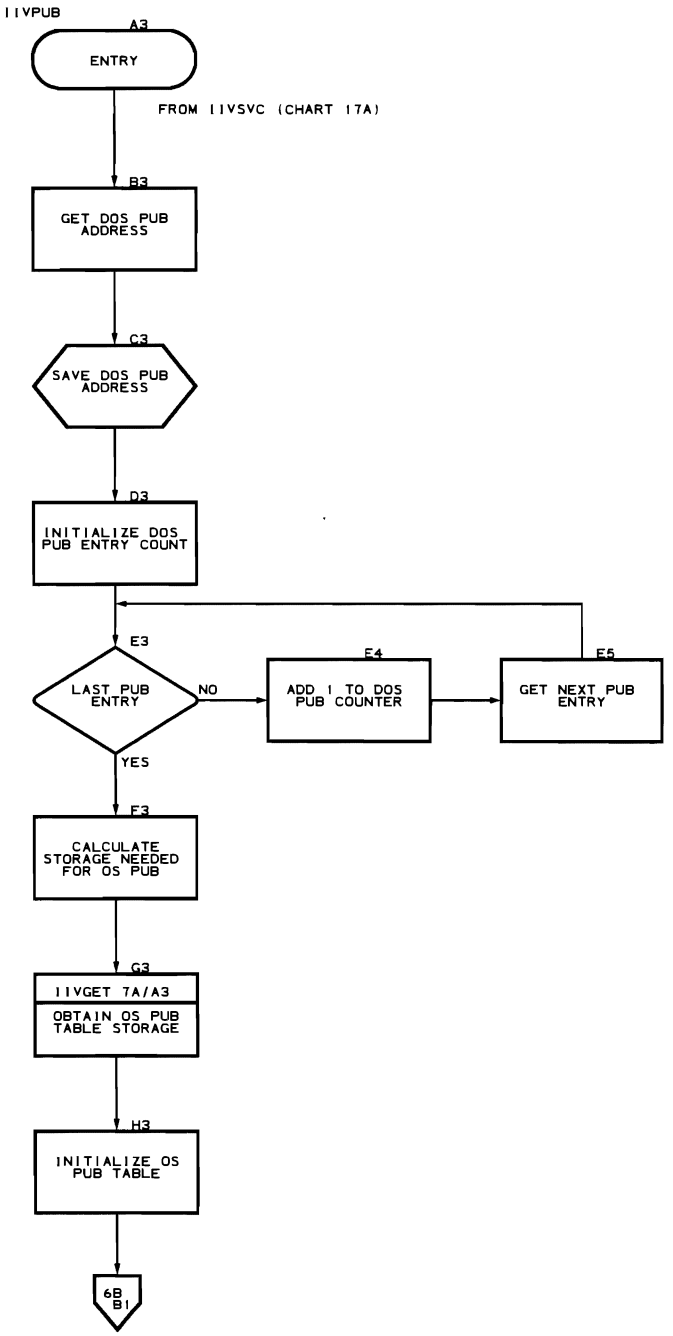
Errors detected:

- A DOS PUB entry did not exist
- Devices not compatible
- Device not stageable

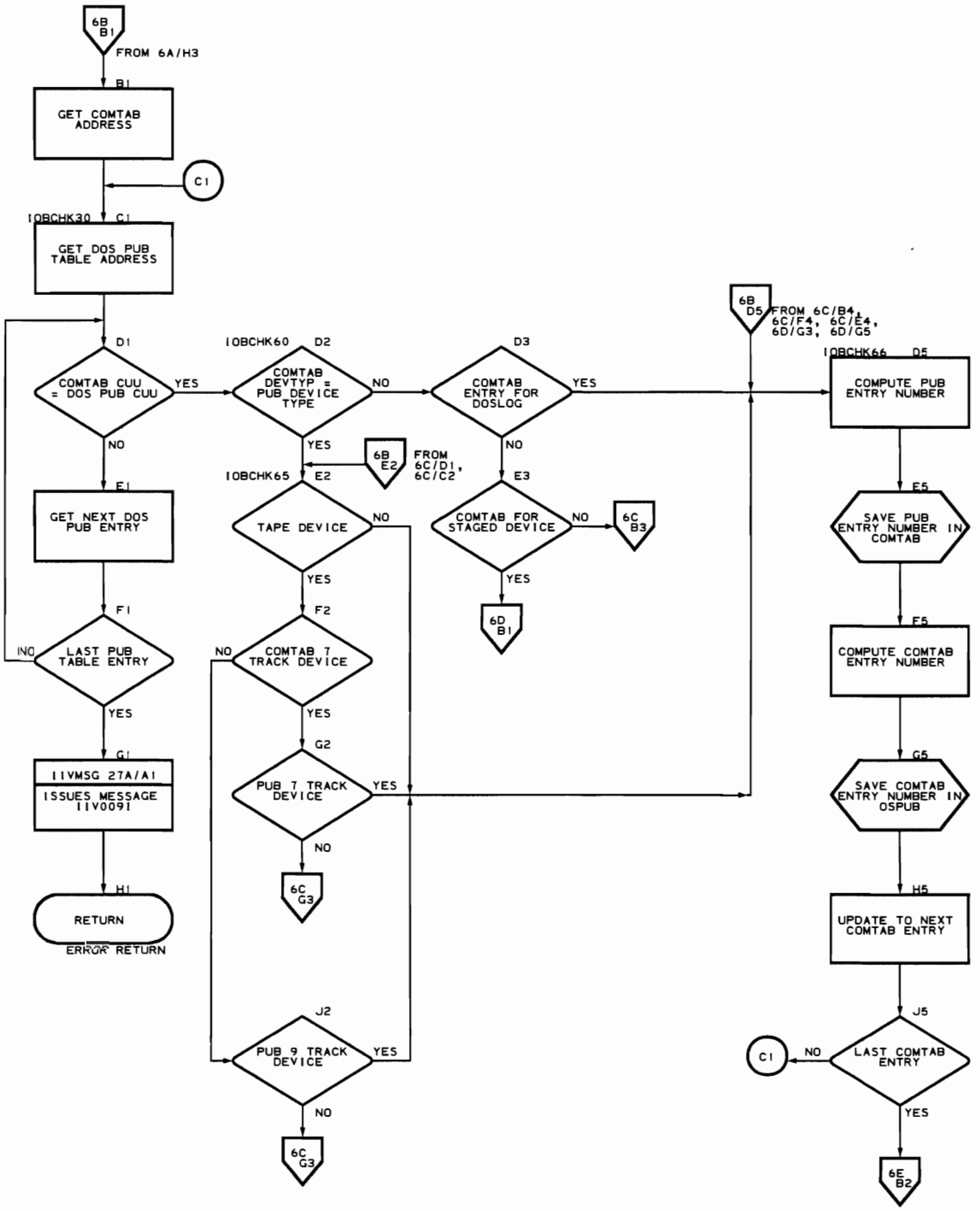
Messages requested:

- IIV009I
- IIV017I
- IIV021I
- IIV100E

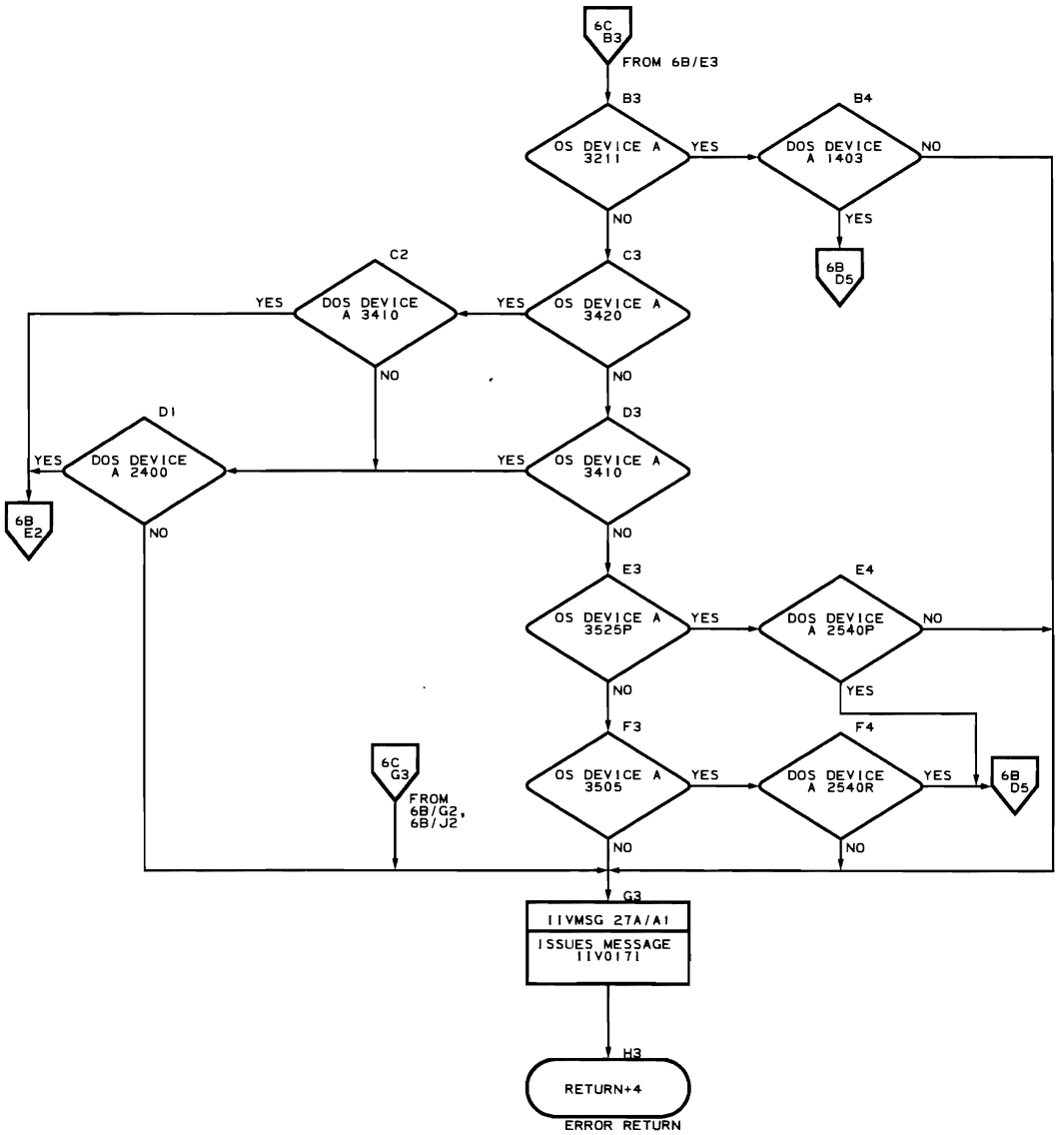
Flowchart 6A. OS PUB Table Build Routine (IIVPUB Part 1 of 6)



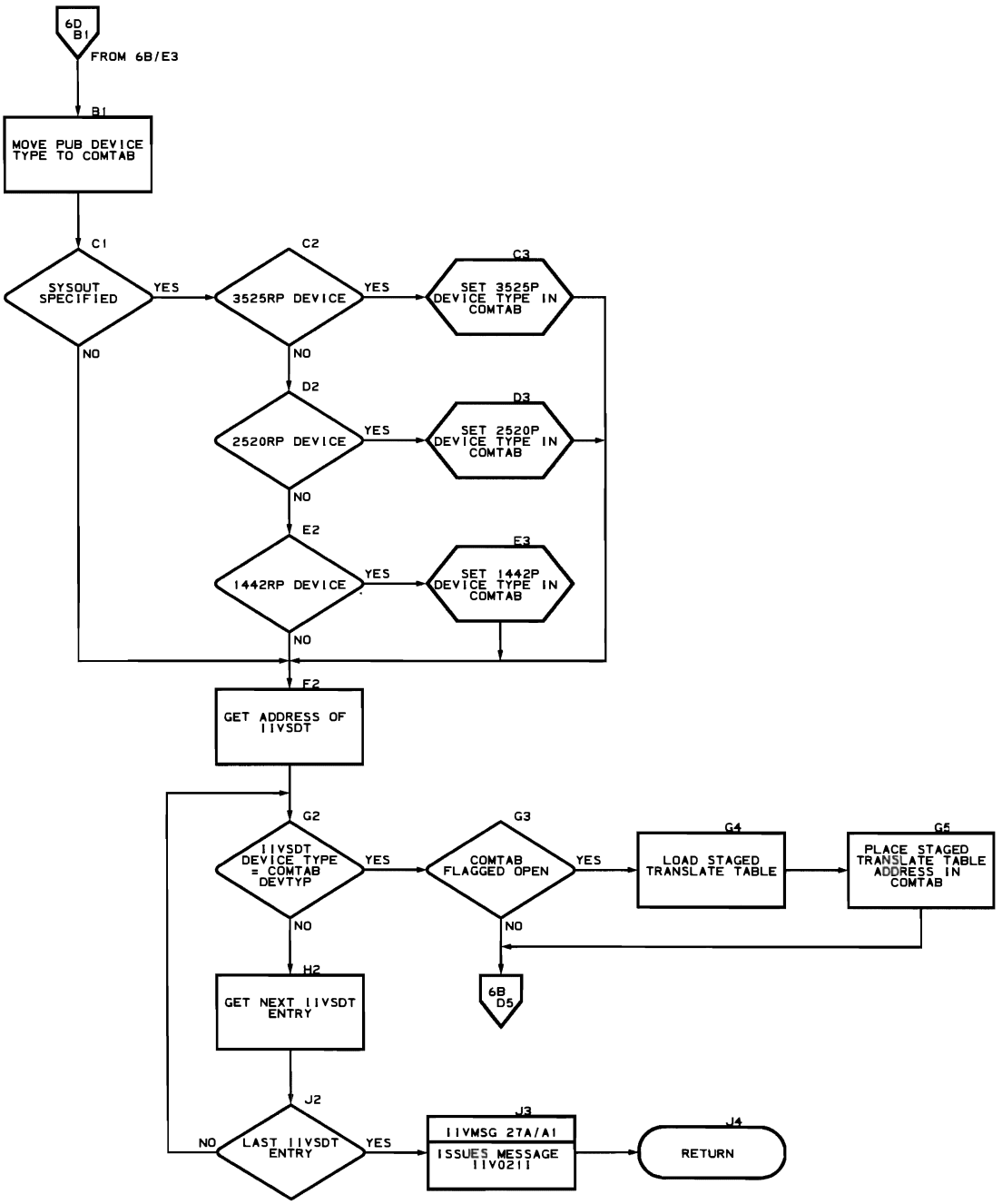
Flowchart 6B. OS PUB Table Build Routine (IIVPUB Part 2 of 6)



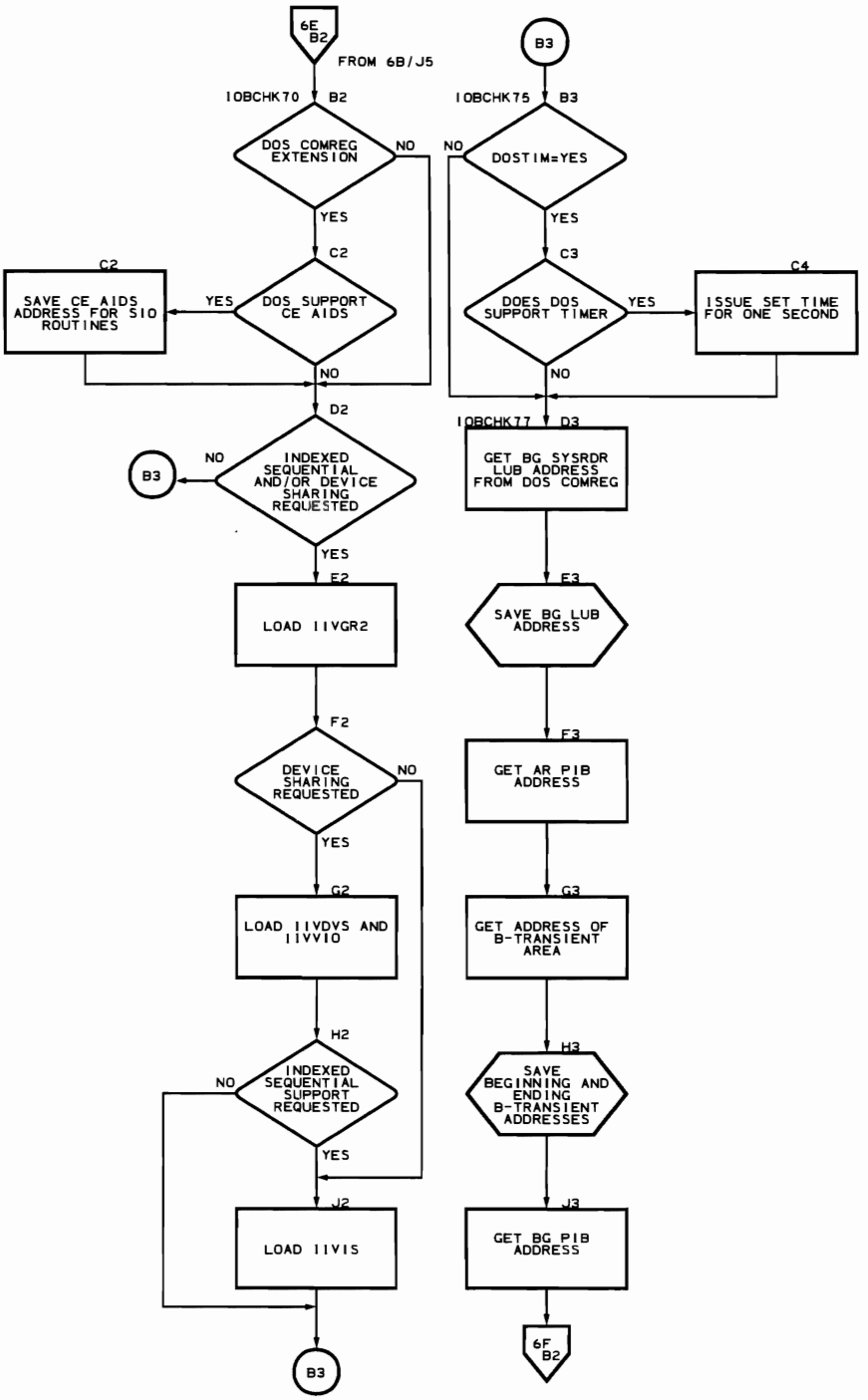
Flowchart 6C. OS PUB Table Build Routine (IIVPUB Part 3 of 6)



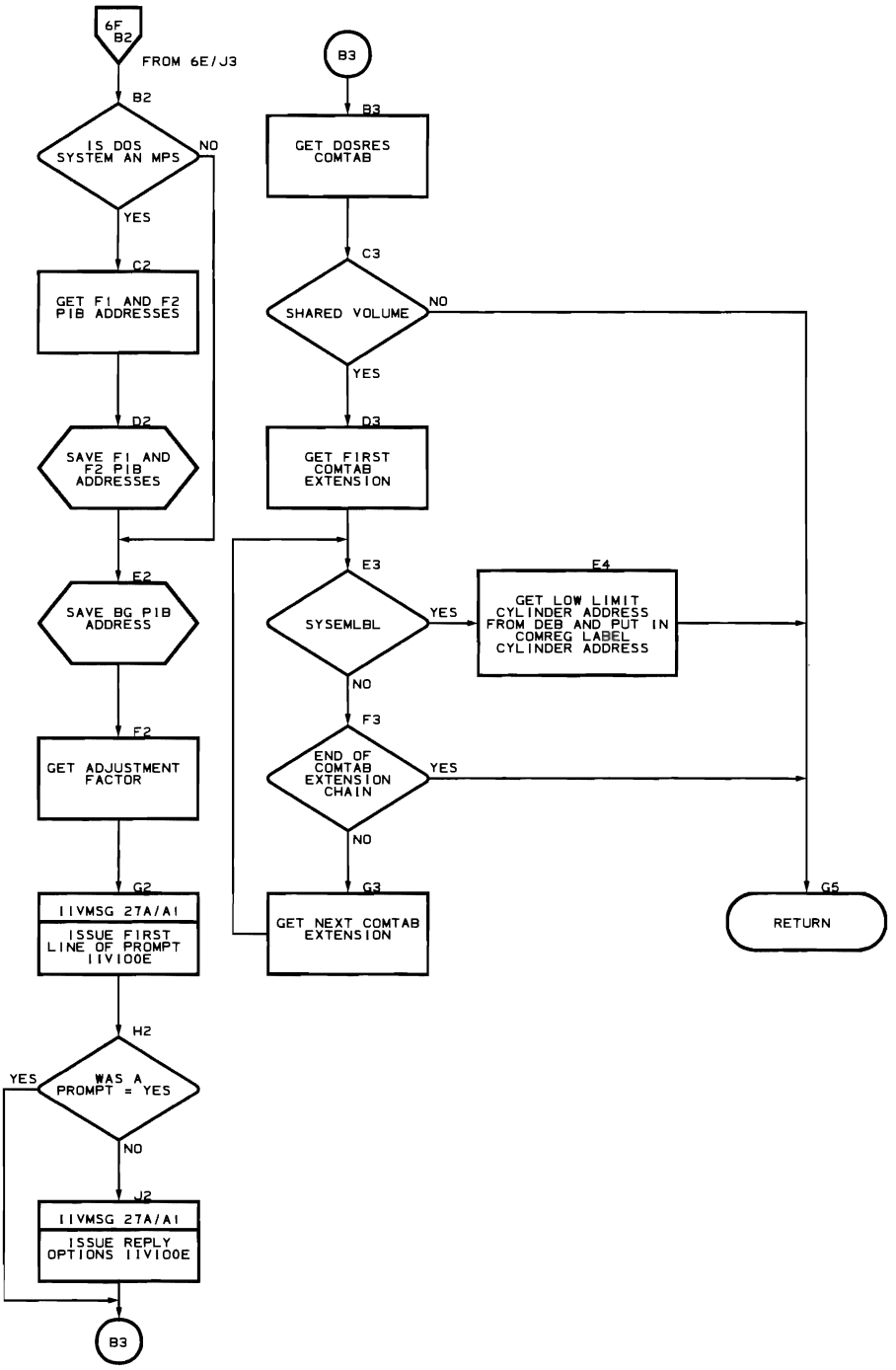
Flowchart 6D. OS PUB Table Build Routine (IIVPUB Part 4 of 6)



Flowchart 6E. OS PUB Table Build Routine (IIVPUB Part 5 of 6)



Flowchart 6F. OS PUB Table Build Routine (IIVPUB Part 6 of 6)



GETMAIN/FREEMAIN Routine (Flowchart 7A)

Module name: IIVGET

Entry point name: IIVGET

Major functions:

- Obtains dynamic storage
- Frees storage

Entered by:

- IIVINT
- IIVIN2
- IIVPUB
- IIVOPN
- IIVDVS
- IIVIS

Modules called: IIVMSG

Exits to:

- Returns to caller
- IIVABN on error conditions

OS macros issued:

- GETMAIN
- FREEMAIN
- SAVE

Input:

GETMAIN	FREEMAIN
• Register 0 - amount of storage needed	• Register 0 - length of storage to be released
• Register 1 - zeros	• Register 1 - address of storage to be released

Output: None

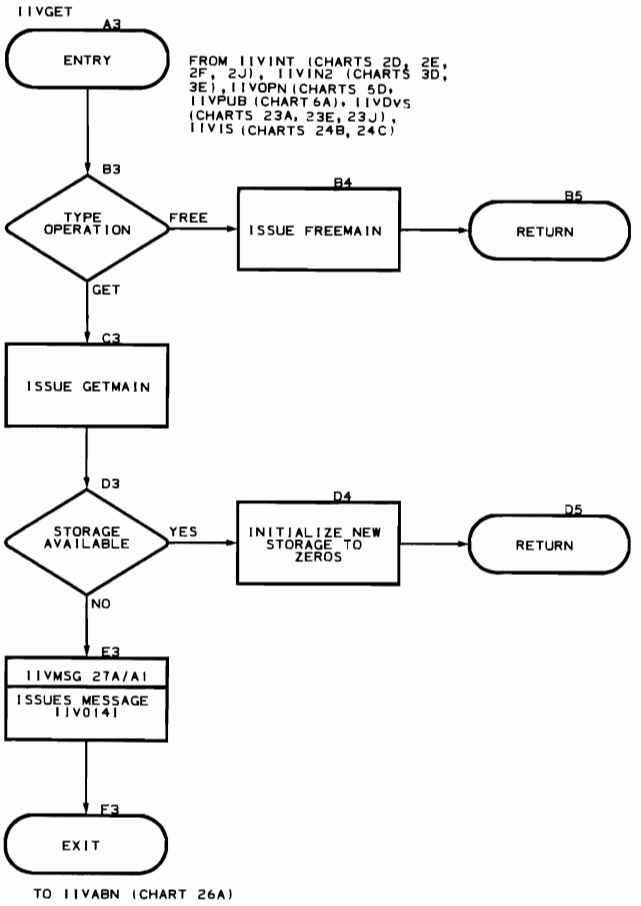
Return codes: X'04' on exit to IIVABN

Tables/work areas: EMUCONS

Errors detected: Storage not available

Messages requested: IIV014I

Flowchart 7A. GETMAIN/FREEMAIN Routine (IIVGET)



Program Check Executive Routine (Flowcharts 8A-8U)

Module name: IIVPCE

Entry point name: IIVPCE

Major functions:

- Checks for privileged operation interruption to determine if it is simulated.
- Checks for first program interruption to determine if it is an operation exception interruption.
- Simulates all other program interruptions.
- SSMRTN simulates the SSM instruction.
- SSKRTN simulates the SSK instruction.
- ISKRTN simulates the ISK instruction.
- LPSWRTN simulates the LPSW instruction.
- Checks for first LPSW that causes control to pass to INTRPT.
- INTRPT checks for wait bit = 1 and enabled interruptions; if these conditions are met, INTRPT simulates the expected interruption.
- TCHRTN simulates the TCH instruction.
- TIORTN simulates the TIO instruction.
- HIORTN simulates the HIO instruction.
- CTLRTN simulates the LCTL and STCTL instructions.
- STIDPRTN simulates the STIDP instruction.
- If the DOS device is not staged or is not the DOS system console device, IIVPCE issues an OS EXCP macro for the intercepted DOS SIO, using DOS CCWs with adjusted addresses (from local to true).
- Initial DOS CCWs for seek and set file mask are not passed to OS.
- Stand-alone seek and sense operations are simulated as complete to DOS but not given to OS.
- Obtains the channel and unit address, the device entry index into COMTAB, and the absolute address of the device entry in COMTAB.
- Determines the absolute value of the BDDD portion of an instruction within the DOS partition or determines channel and unit address from I/O instruction.
- Dissects the ISK or SSK instruction to obtain corresponding addresses of R1 register and entry in ISK/SSK Table.
- STIDC instruction is ignored.
- SCKRTN simulates the DOS SCK instruction. Disables STCK instruction in the DOS SVC 34 routine.
- STCKRTN simulates the STCK instruction for DOS SVC 34.

Entered by:

- DOS via a program interruption
- IIVPCI

Modules called:

- IIVCCW
- IIVMSG
- IIVOPN
- IIVABN
- IIVVIO
- IIVCHK
- IIVRCW

Exits to:

- IIVLOG
- IIVSTG
- IIVRTE
- IIVABN

OS macros issued:

- WAIT
- EXCP
- IOHALT
- POST

Input:

- DOS registers
- Local execution list
- DOS program new PSW
- Pointer to instruction
- Pointer to local execution list
- Pointer to Emulator constants
- Pointer to DOS area
- Base register
- Pointer to current PUB in location REG3
- Pointer to CCB in location REG1

Output:

- DOS program old PSW
- Base register
- Pointer to instruction
- Pointer to local execution list
- Pointer to DOS area
- Pointer to Emulator constants

- System mask in current PSW
- Storage key value in ISK/SSK table
- Storage key value in instruction register
- Pointer to COMTAB entry for channel
- Index of channel within COMTAB
- Channel and unit address
- Absolute address of location in DOS area or channel and unit address from I/O instruction

Return codes: Error code 12 for IIVABN

Tables/work areas:

- Local execution list
- Current PSW
- ISK/SSK table
- COMTAB
- DCB
- ECB
- CSW
- CAW
- IOB
- CCW
- DOS I/O old PSW
- DOS I/O new PSW
- Post ECB list
- TEB
- PUB

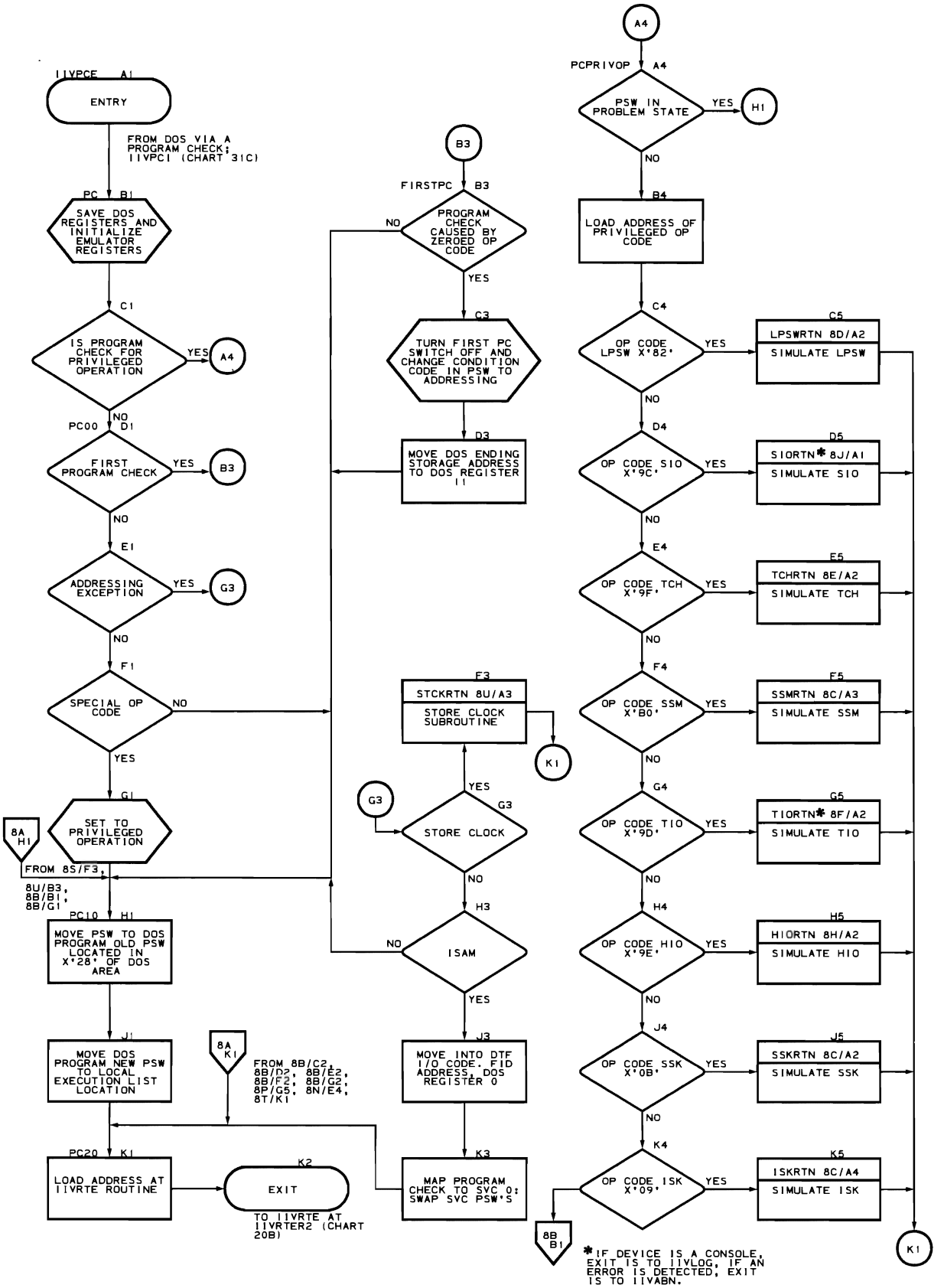
Errors detected:

- For nondedicated device - invalid seek address
- Invalid or no IPL device specified
- For IBM 2321 - no seek or no bin number defined

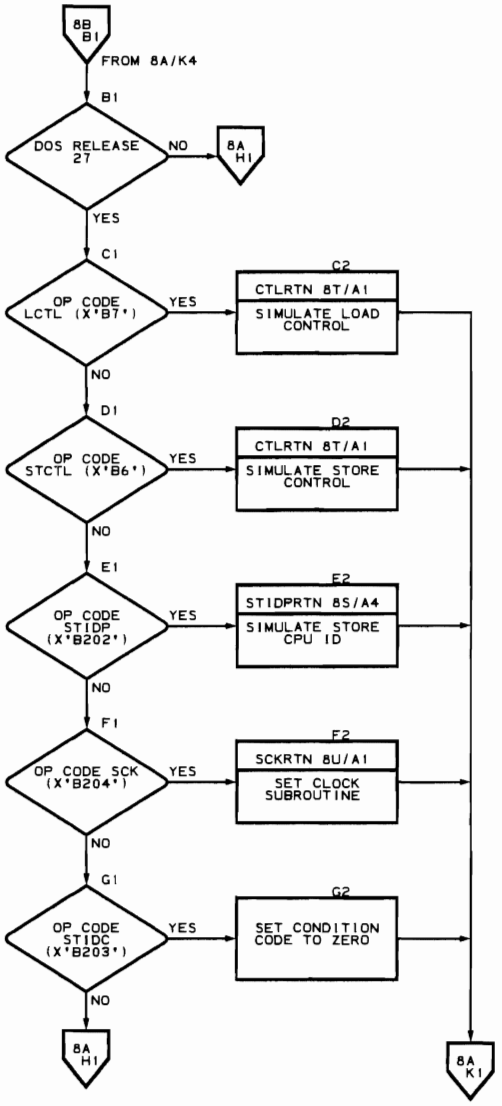
Messages requested:

- IIV160A
- IIV263I

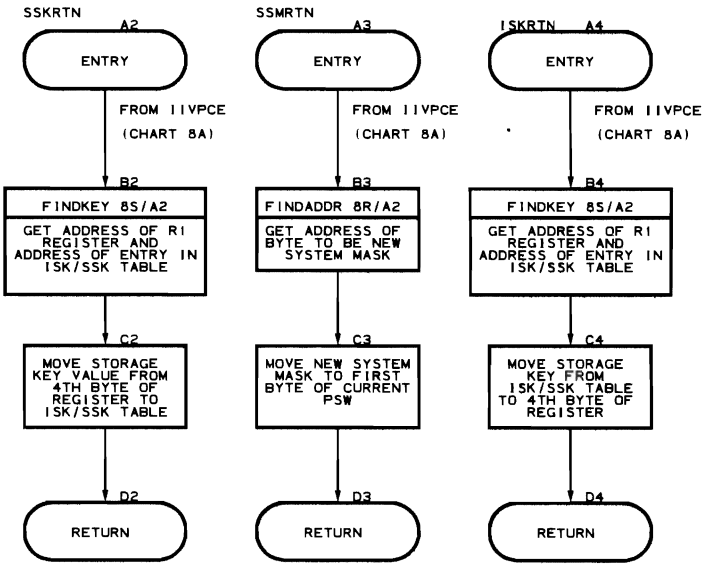
Flowchart 8A. Program Check Executive Routine (IIVPCE Part 1 of 19)



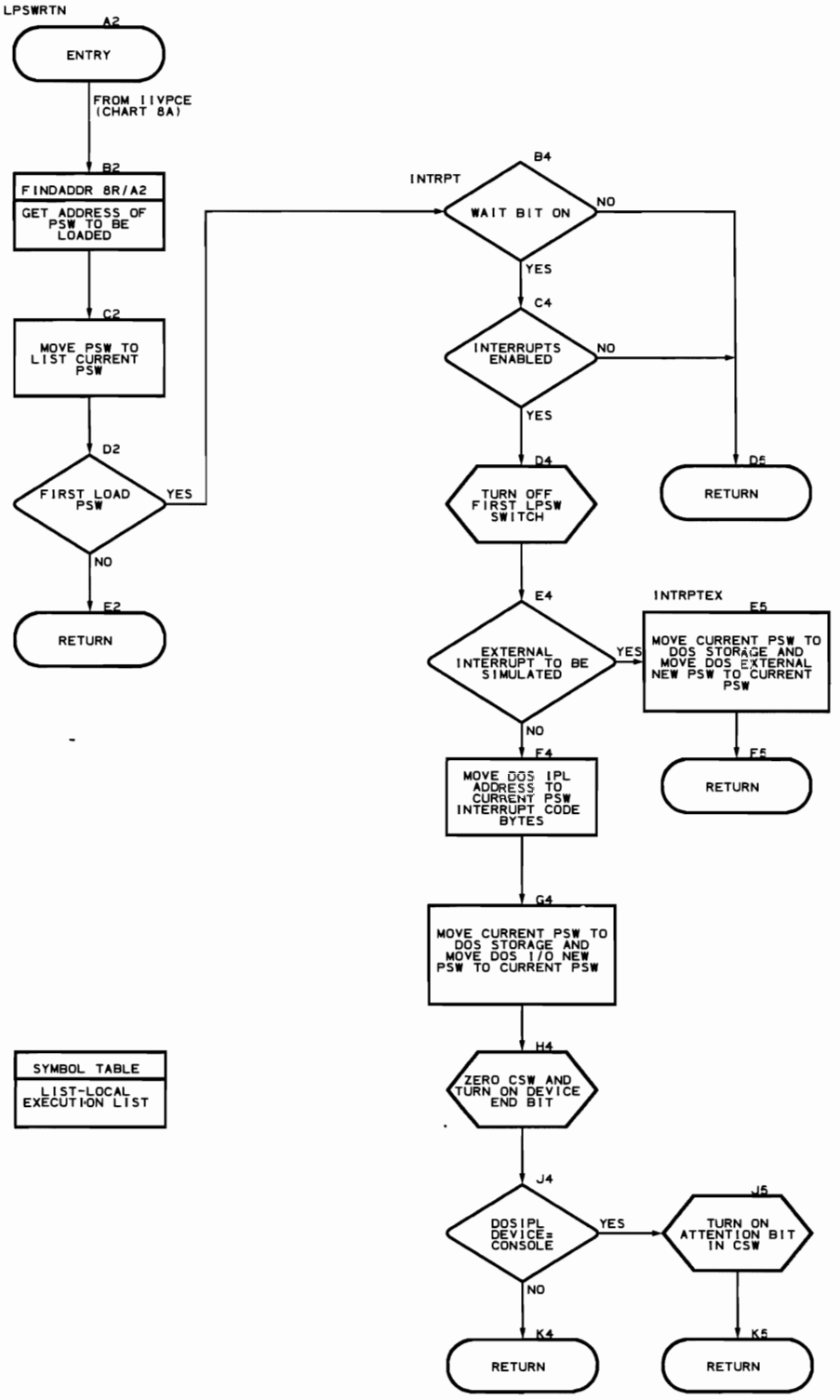
Flowchart 8B. Program Check Executive Routine (IIVPCE Part 2 of 19)



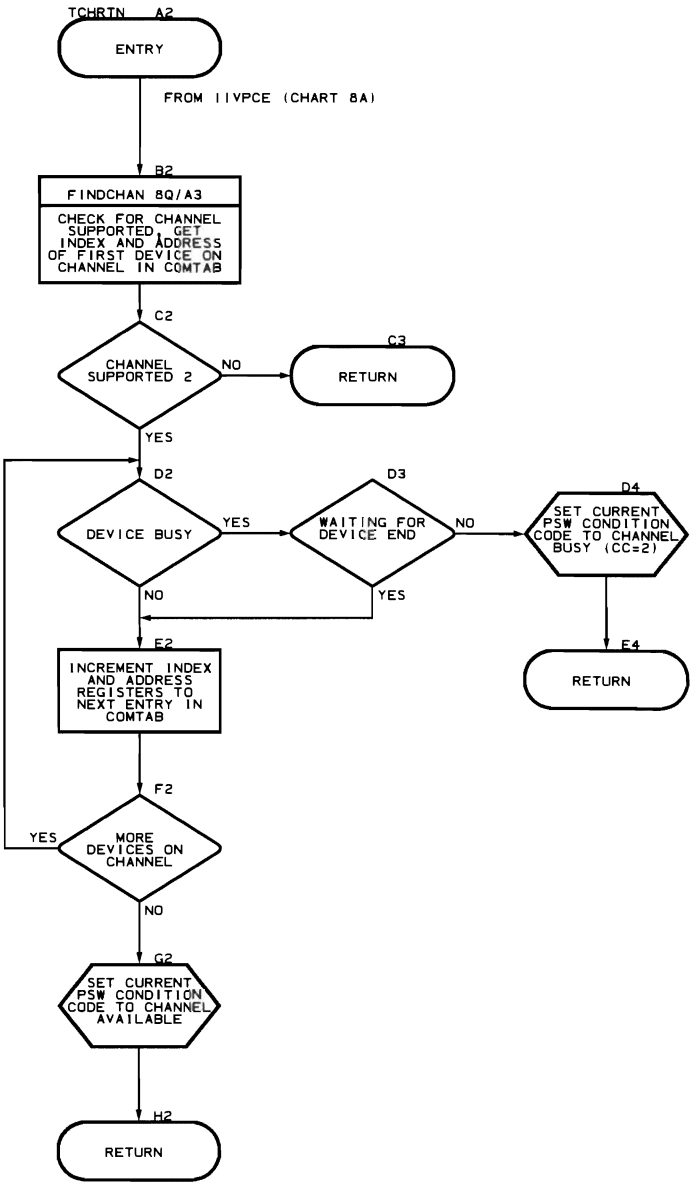
Flowchart 8C. SSK, SSM, and ISK Simulation Subroutines (IIVPCE Part 3 of 19)



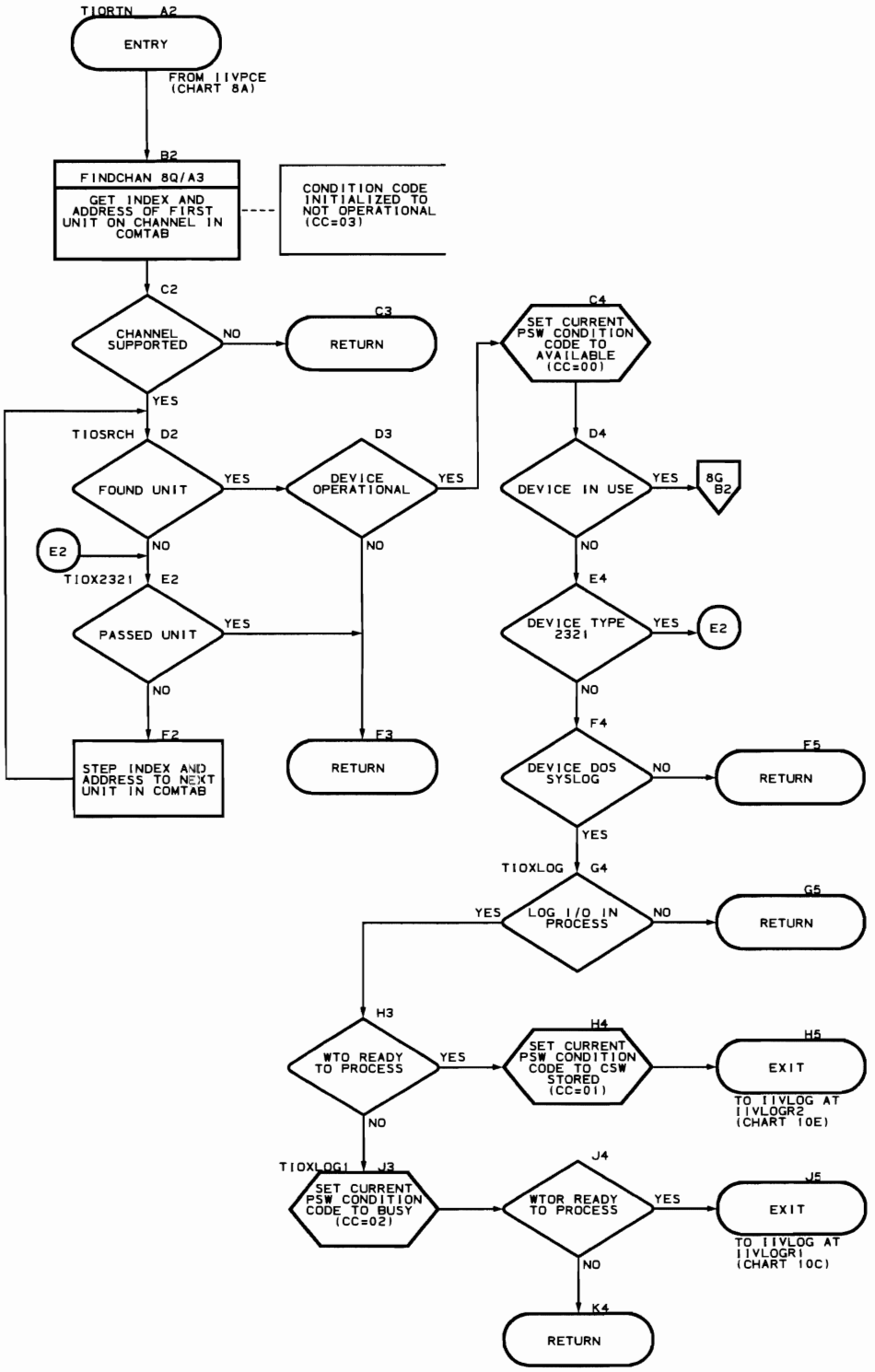
Flowchart 8D. LPSW Simulation Subroutine (IIVPCE Part 4 of 19)



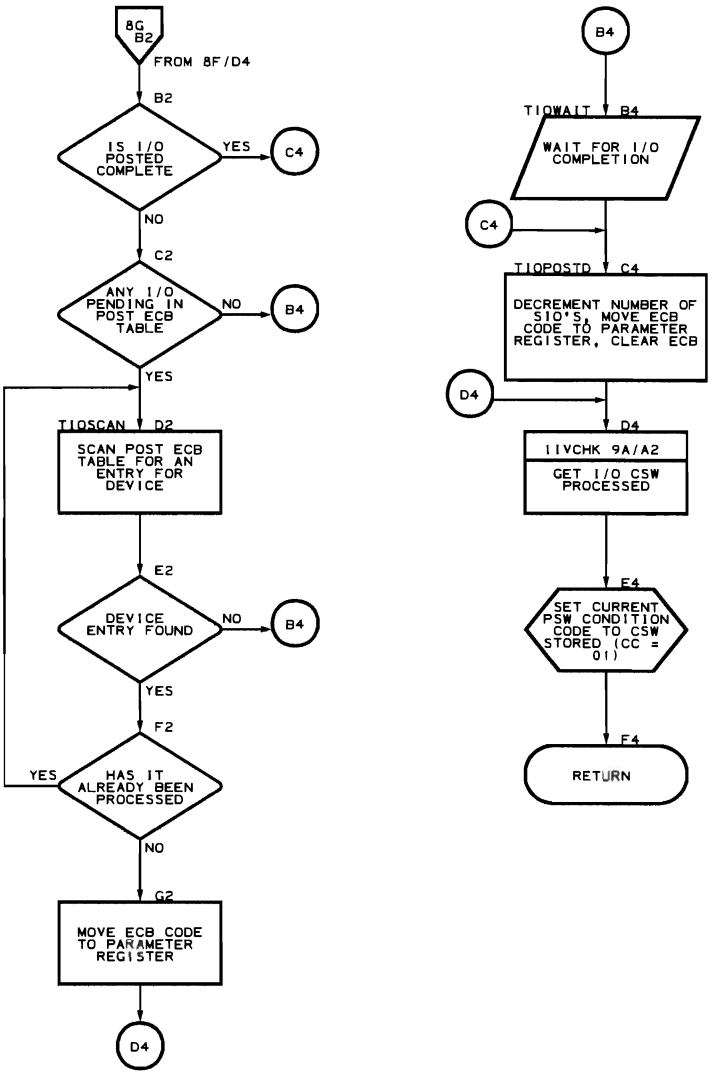
Flowchart 8E. TCH Simulation Subroutine (IIVPCE Part 5 of 19)



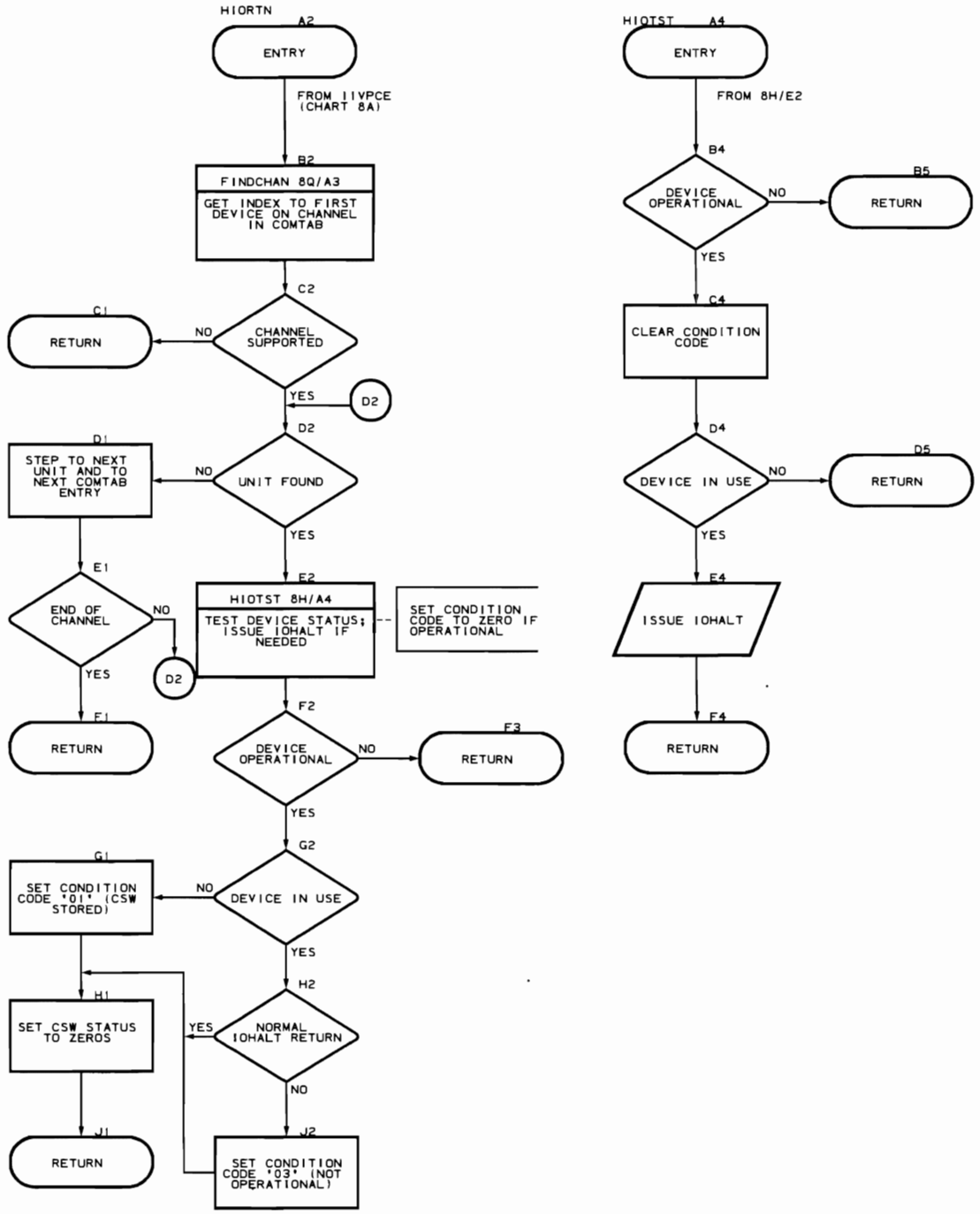
Flowchart 8F. TIO Simulation Subroutine (IIVPCE Part 6 of 19)



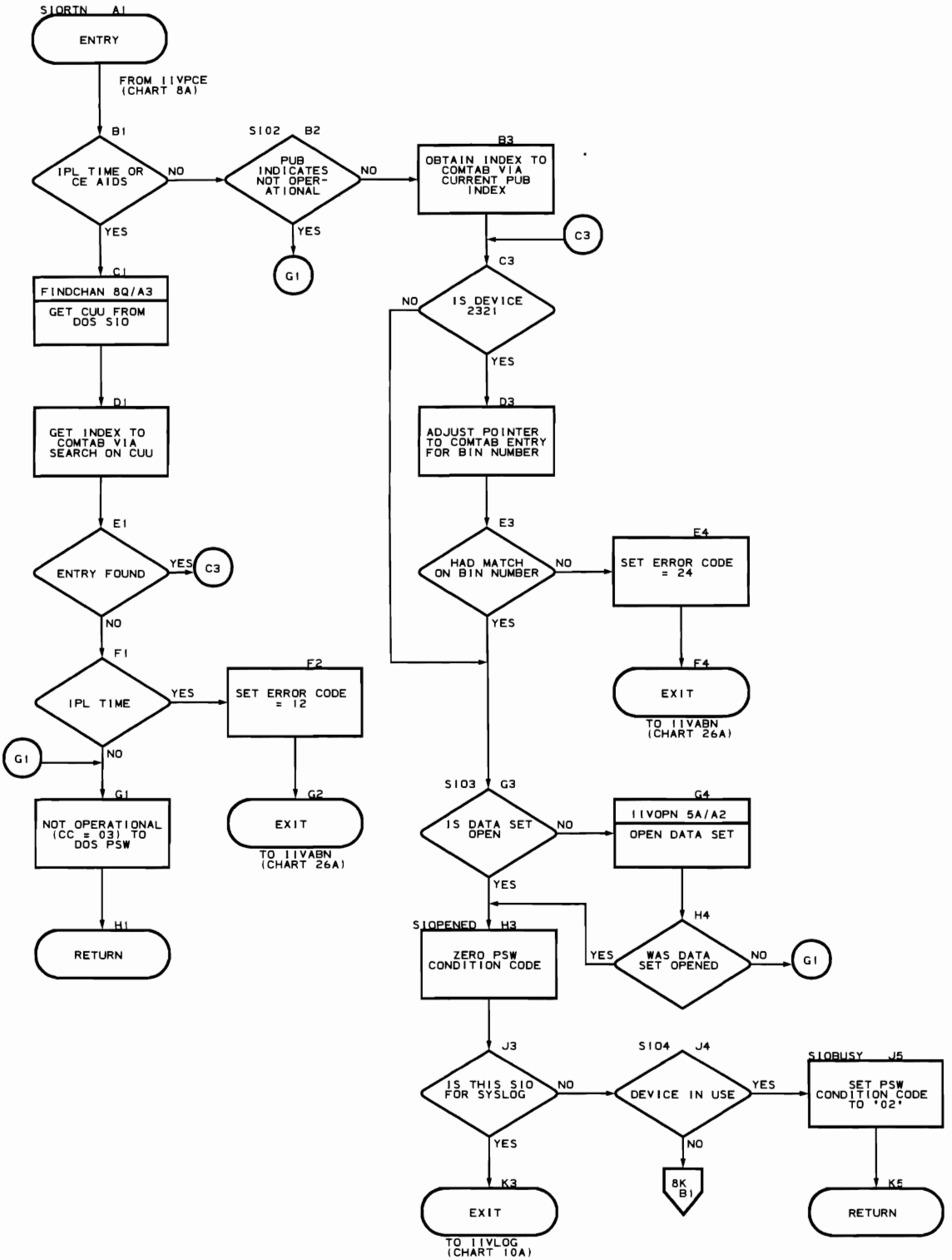
Flowchart 8G. TIO Simulation Subroutine (IIVPCE Part 7 of 19)



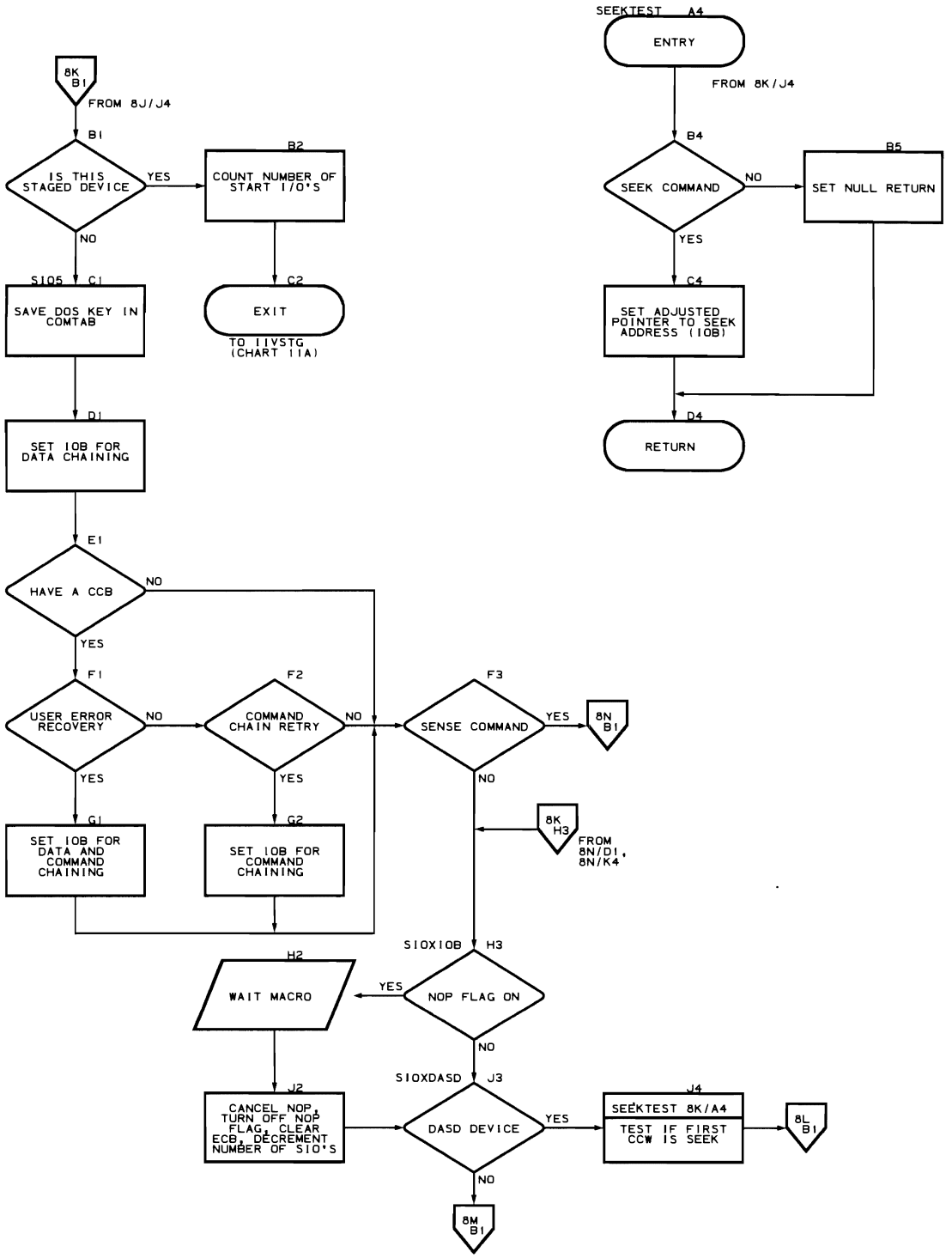
Flowchart 8H. HIO Simulation Subroutine (IIVPCE Part 8 of 19)



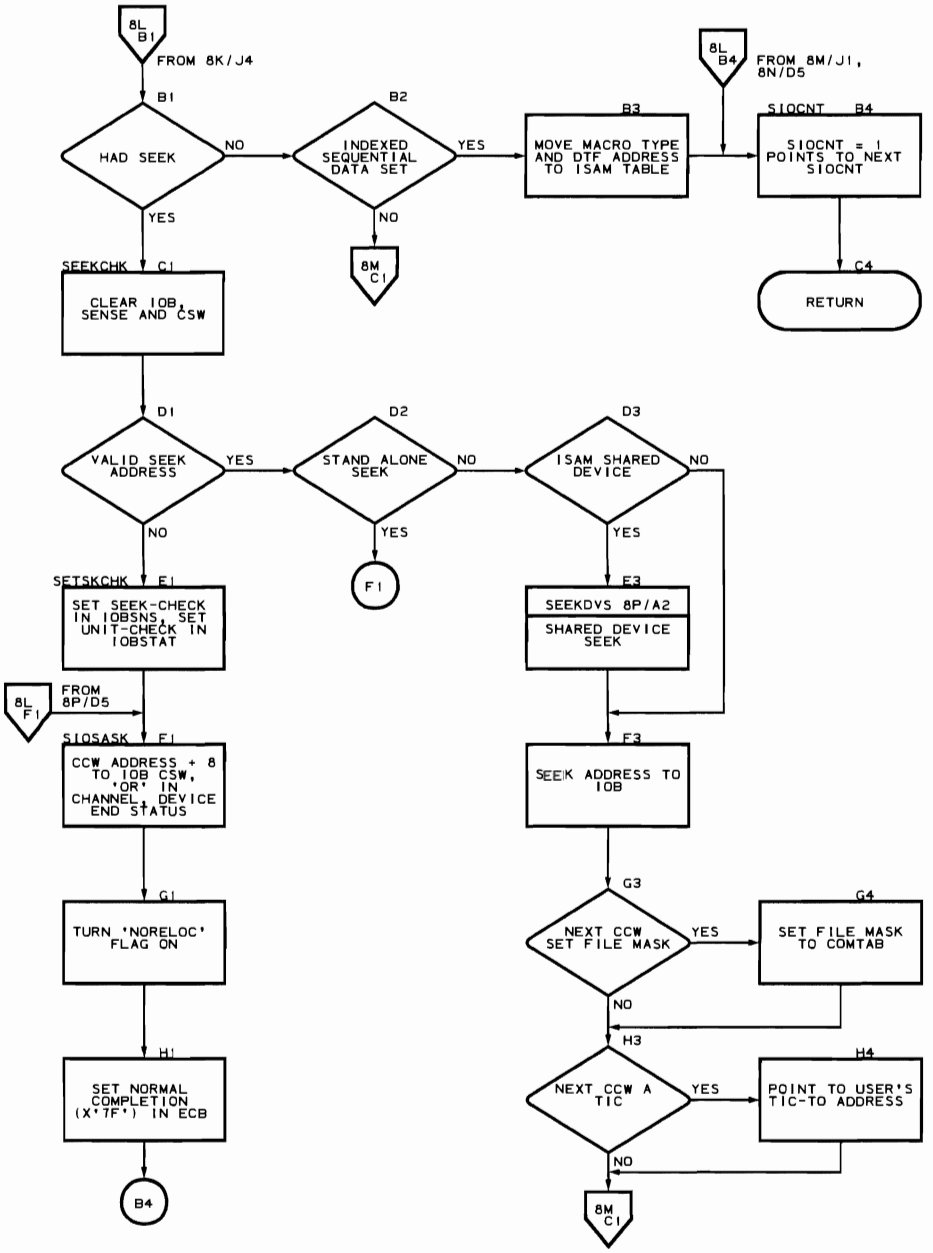
Flowchart 8J. SIO Subroutine (IIVPCE Part 9 of 19)



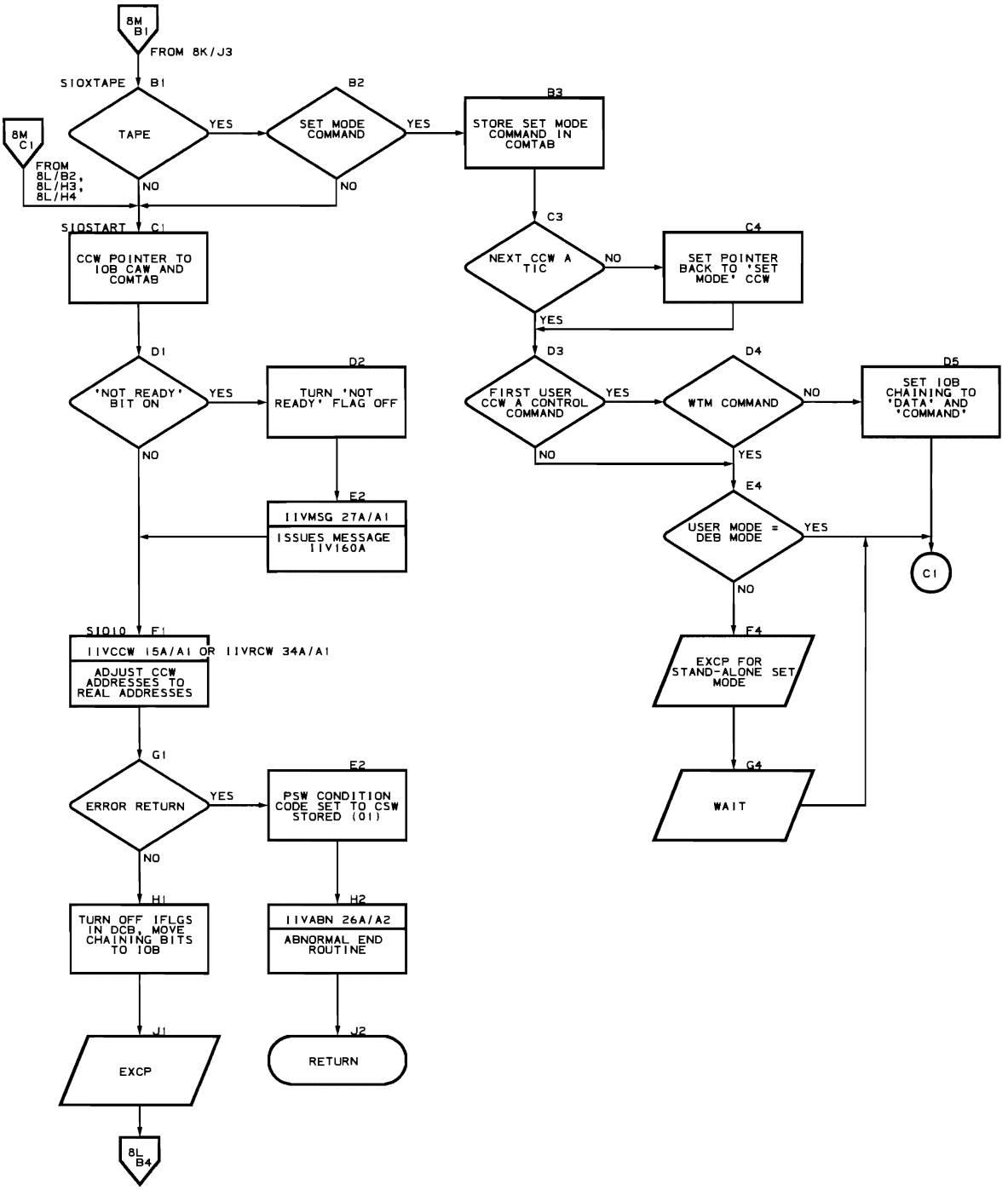
Flowchart 8K. SIO and SEEKTEST Subroutines (IIVPCE Part 10 of 19)



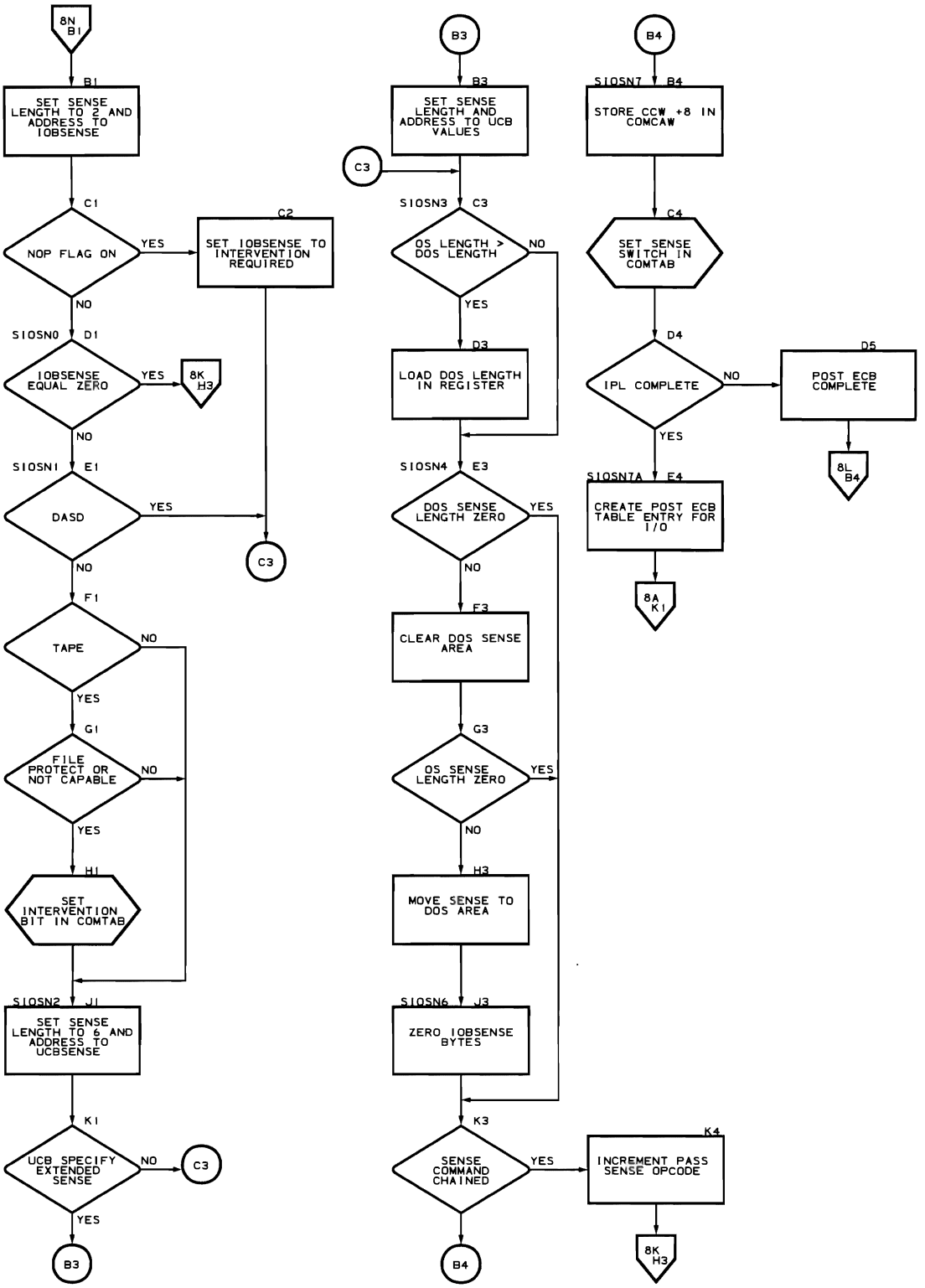
Flowchart 8L. SIO Subroutine (IIVPCE Part 11 of 19)



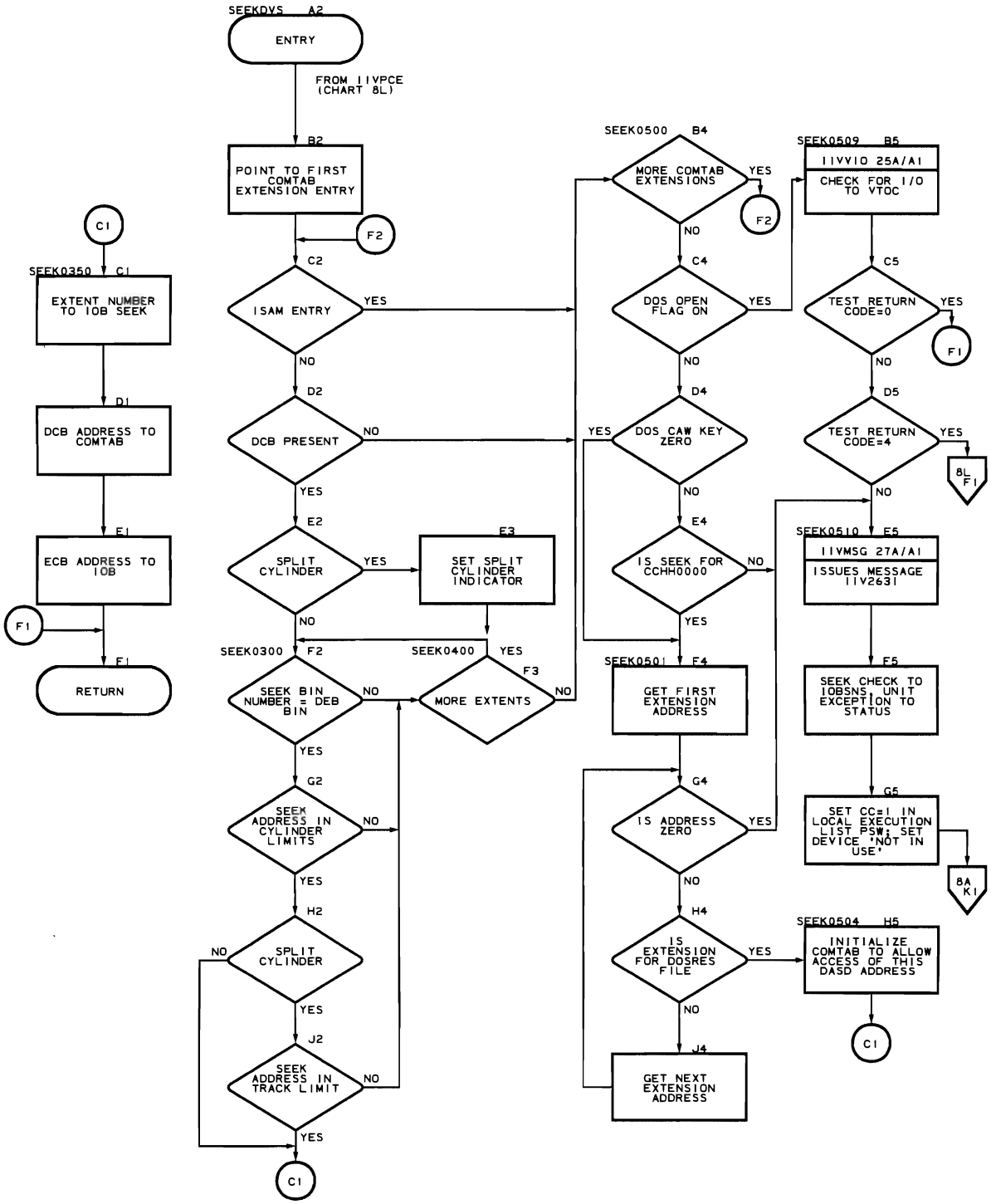
Flowchart 8M. SIO Subroutine (IIVPCE Part 12 of 19)



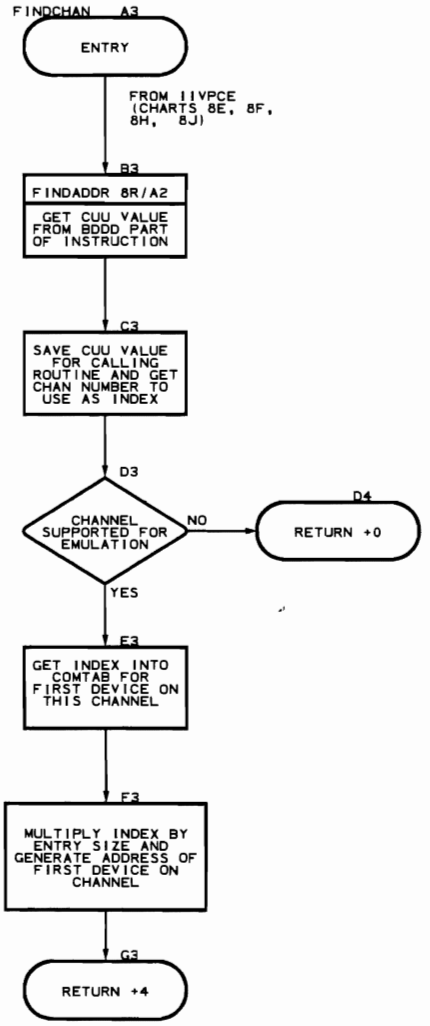
Flowchart 8N. SIO Subroutine (IIVPCE Part 13 of 19)

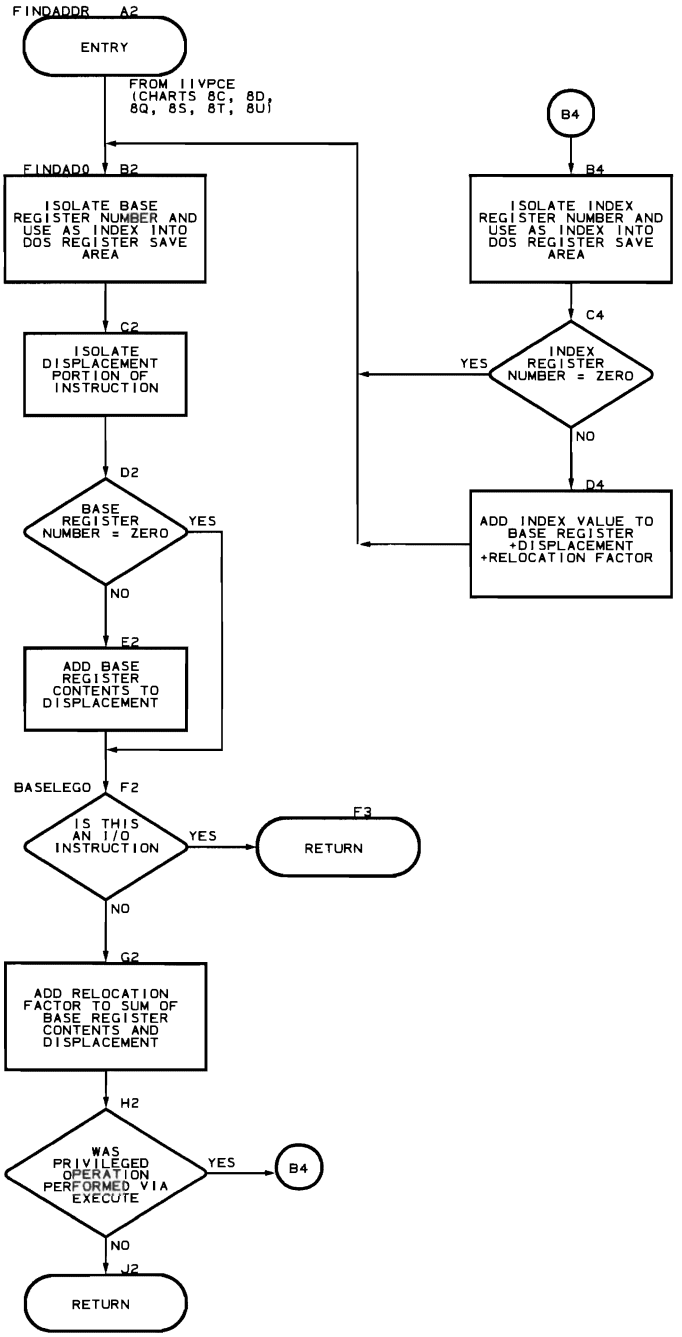


Flowchart 8P. SEEKDVS Subroutine (IIVPCE Part 14 of 19)

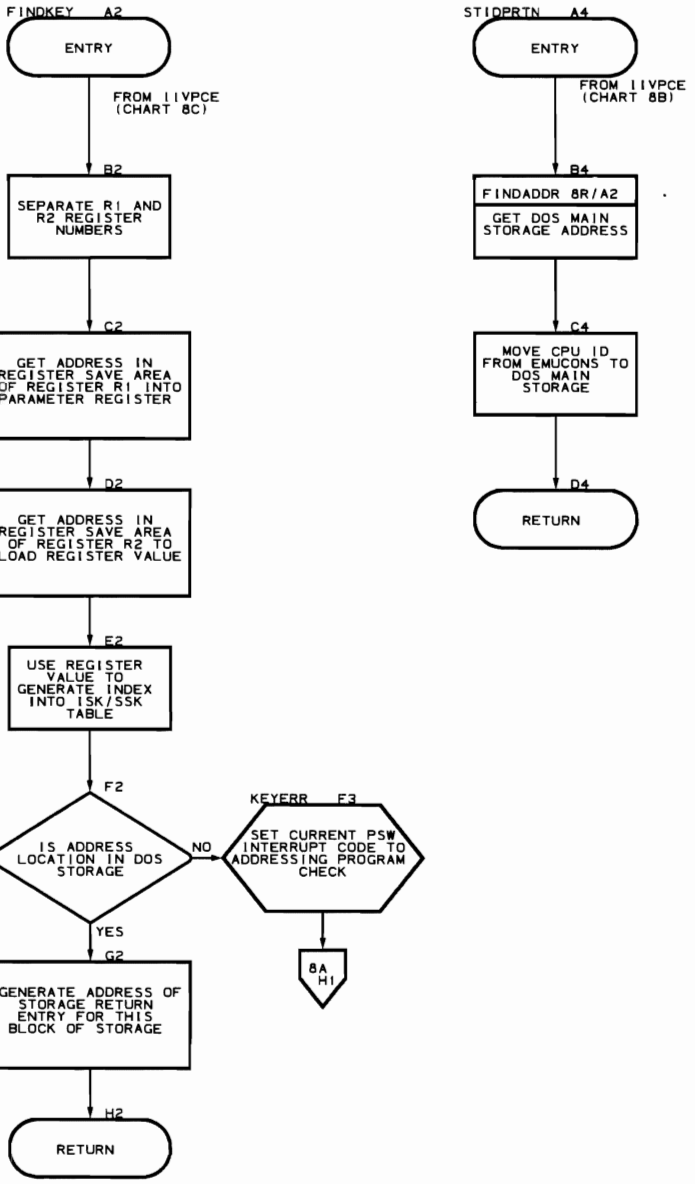


Flowchart 8Q. FINDCHAN Subroutine (IIVPCE Part 15 of 19)

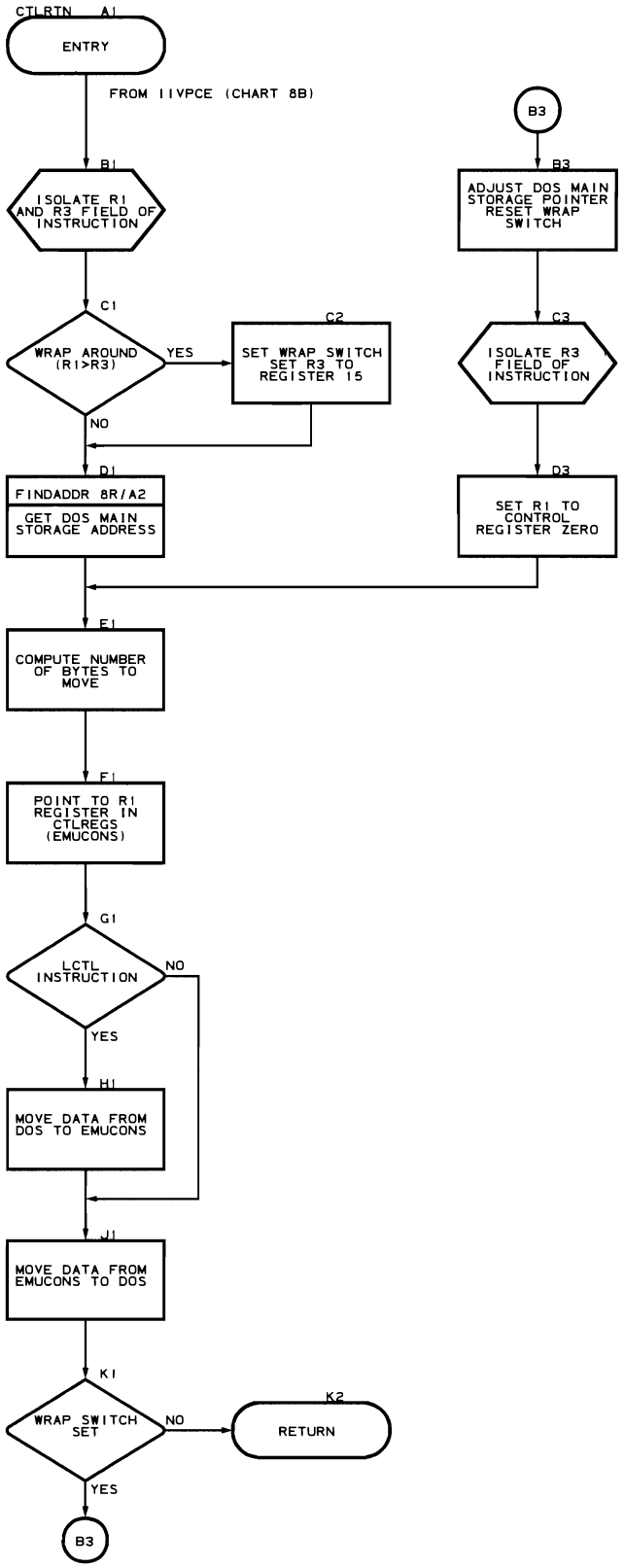



FINDADDR A2
ENTRY
FROM IIVPCE (CHARTS 8C, 8D, 8Q, 8S, 8T, 8U)
FINDAD0 B2
ISOLATE BASE REGISTER NUMBER AND USE AS INDEX INTO DOS REGISTER SAVE AREA
C2
ISOLATE DISPLACEMENT PORTION OF INSTRUCTION
D2
BASE REGISTER NUMBER = ZERO
YES
NO
E2
ADD BASE REGISTER CONTENTS TO DISPLACEMENT
BASELEGO F2
IS THIS AN I/O INSTRUCTION
YES
F3
RETURN
NO
G2
ADD RELOCATION FACTOR TO SUM OF BASE REGISTER CONTENTS AND DISPLACEMENT
H2
WAS PRIVILEGED OPERATION PERFORMED VIA EXECUTE
YES
B4
NO
J2
RETURN
B4
B4
ISOLATE INDEX REGISTER NUMBER AND USE AS INDEX INTO DOS REGISTER SAVE AREA
C4
INDEX REGISTER NUMBER = ZERO
YES
NO
D4
ADD INDEX VALUE TO BASE REGISTER +DISPLACEMENT +RELOCATION FACTOR

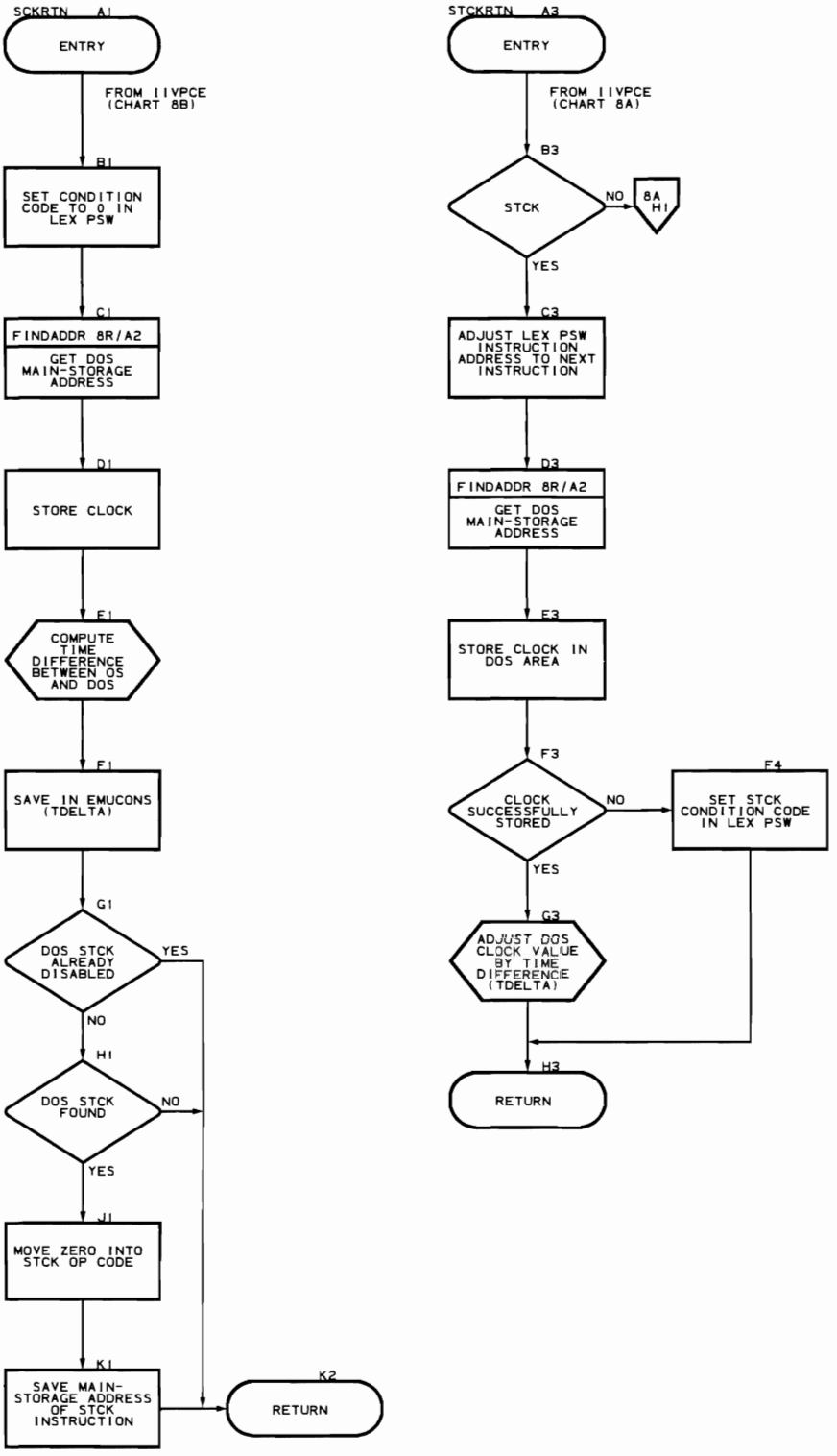
Flowchart 8S. FINDKEY and Store CPU ID Subroutines (IIVPCE Part 17 of 19)



Flowchart 8T. Load/Store Control Register Subroutine (IIVPCE Part 18 of 19)



Flowchart 8U. Set Clock, Store Clock Subroutines (IIVPCE Part 19 of 19)



Check I/O Routine (Flowcharts 9A-9D)

Module name: IIVCHK

Entry point name: IIVCHK

Major functions:

- Posts the CSW to DOS at I/O interruption
- If permanent error occurs, posts high error count to DOS to bypass DOS error recovery procedures
- Readjusts DOS CCWs from true to local addresses
- Provides DOS label cylinder address when DOS system residence volume is shared

Entered from:

- IIVRTE
- IIVPCE

Modules called:

- IIVCCW
- IIVABN (if invalid CCW)
- IIVMSG
- IIVRCW

Exits to:

- Caller

OS macros issued: EXCP (to issue NOP)

Input:

- Local execution list
- Pointer to EMUCONS
- Pointer to DOS area
- Base register

Output:

- DOS CSW posted
- High error count (254) posted to DOS PUB table (or TEB or TEBV, if appropriate)
- IOB sense bytes
- IOB status bits
- DOS label cylinder address in DOS input area when DOS system directory record 1 read

Return codes:

Register 15 - 0 = simulate I/O interruption

4 = ignore current entry in post ECB list and get next entry

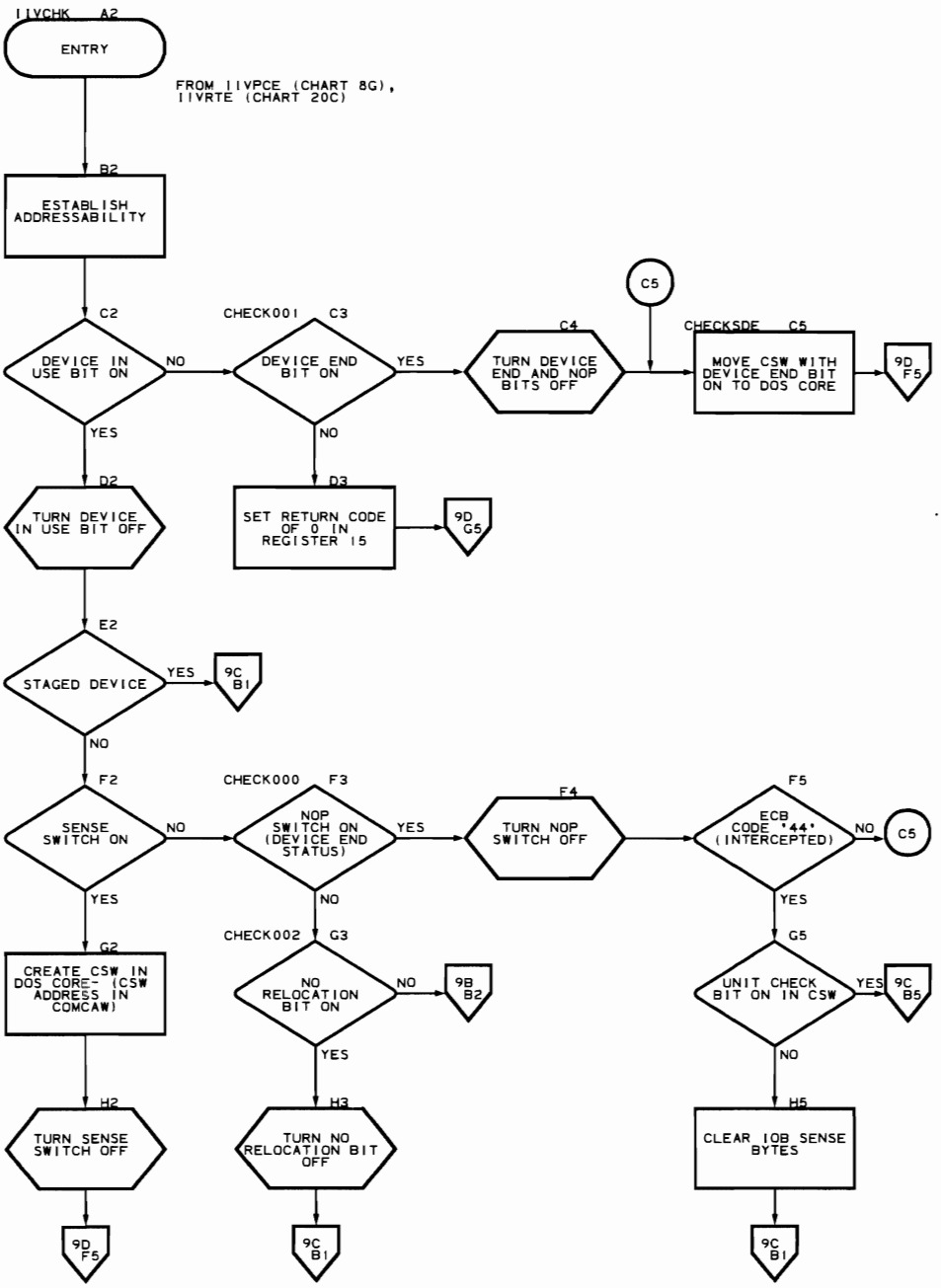
Tables/work areas:

- TEB
- COMTAB
- IOB
- Local execution list
- DOS communications region
- DOS PUB

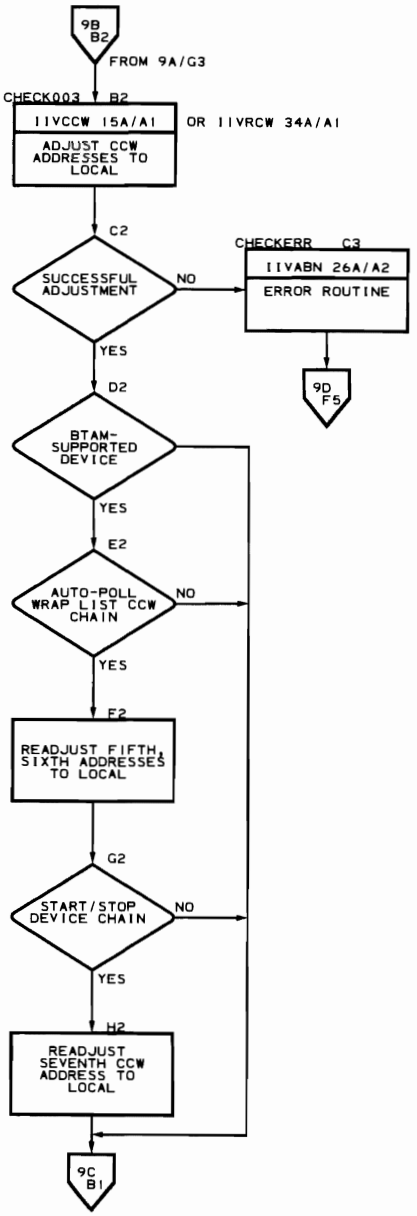
Errors detected: An error detected by IIVCCW or IIVRCW causes this module to pass control to IIVABN.

Messages requested: IIV160A

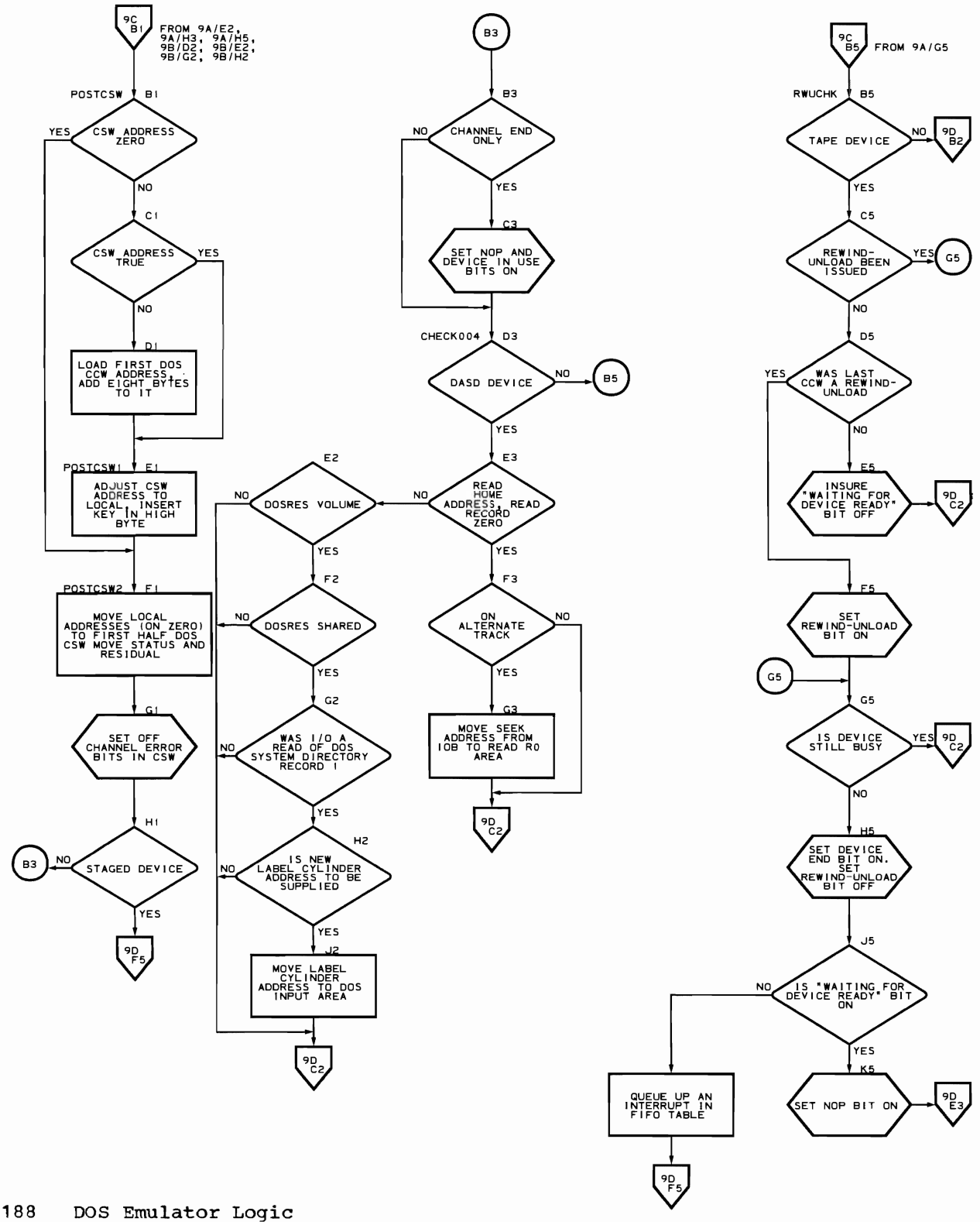
Flowchart 9A. Check I/O Routine (IIVCHK Part 1 of 4)



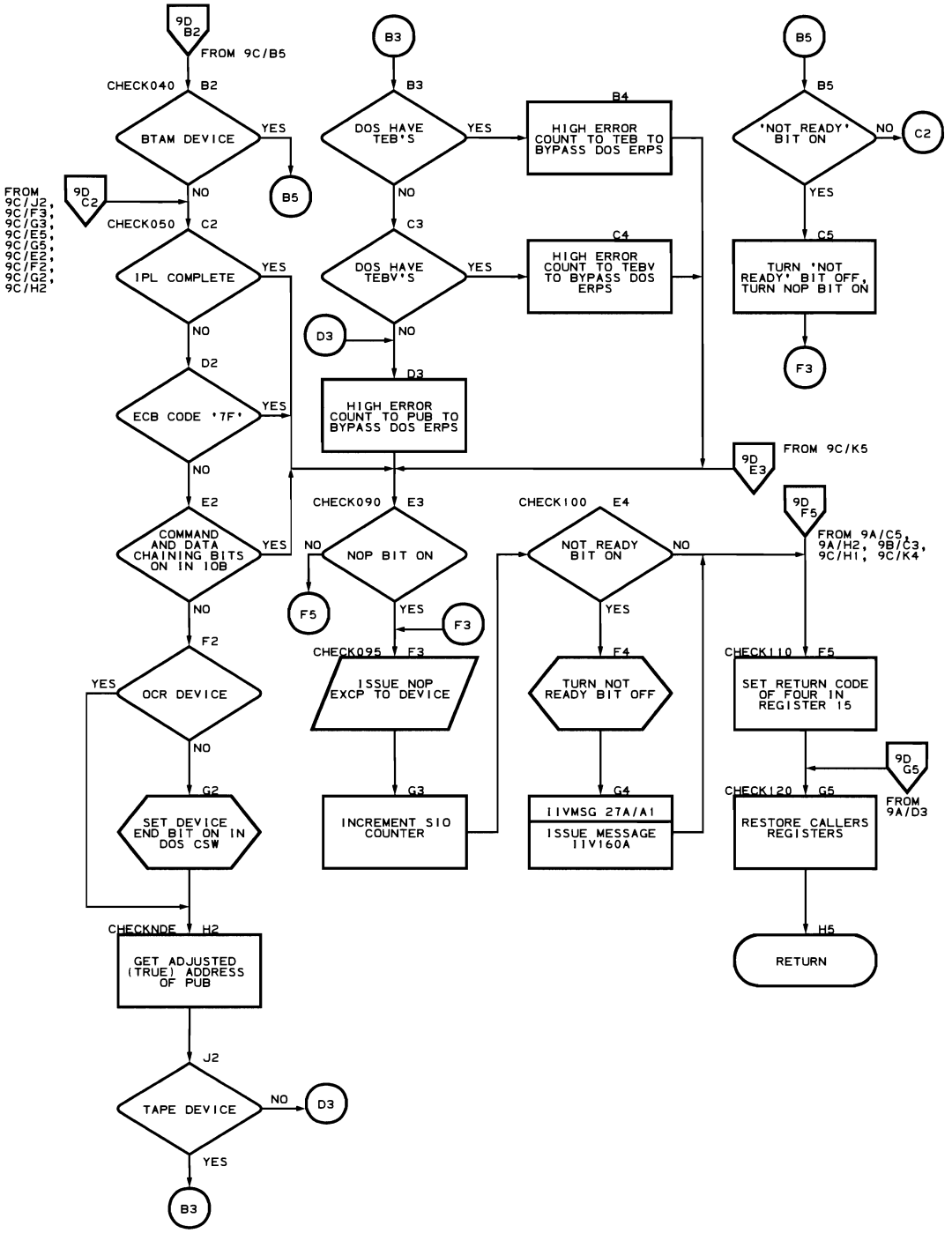
Flowchart 9B. Check I/O Routine (IIVCHK Part 2 of 4)



Flowchart 9C. Check I/O Routine (IIVCHK Part 3 of 4)



Flowchart 9D. Check I/O Routine (IIVCHK Part 4 of 4)



Interpretive SYSLOG Routine (Flowcharts 10A-10E)

Module name: IIVLOG

Entry point name:

- IIVLOG
- IIVLOGR1
- IIVLOGR2
- IIVLOGR3

Major functions: Interprets keyboard-printer CCW chains and issues WTO or WTOR macros as required

Entered from:

- IIVPCE
- IIVRTE

Modules called:

- IIVAWV
- IIVCWV

Exits to:

- IIVPCE
- IIVRTE

OS macros issued:

- WTO
- WTOR

Input:

- Pointer to DOSCORE
- Pointer to Emulator constants (IIVCON)
- Pointer to local execution list
- Replies to WTOR issued by this module

Output:

- WTO/WTOR messages requested by DOS supervisor and/or problem programs
- Updated DOS PSW in local execution list
- Updated DOS CCW data areas
- Updated DOS CSW

Return codes: None

Tables/work areas:

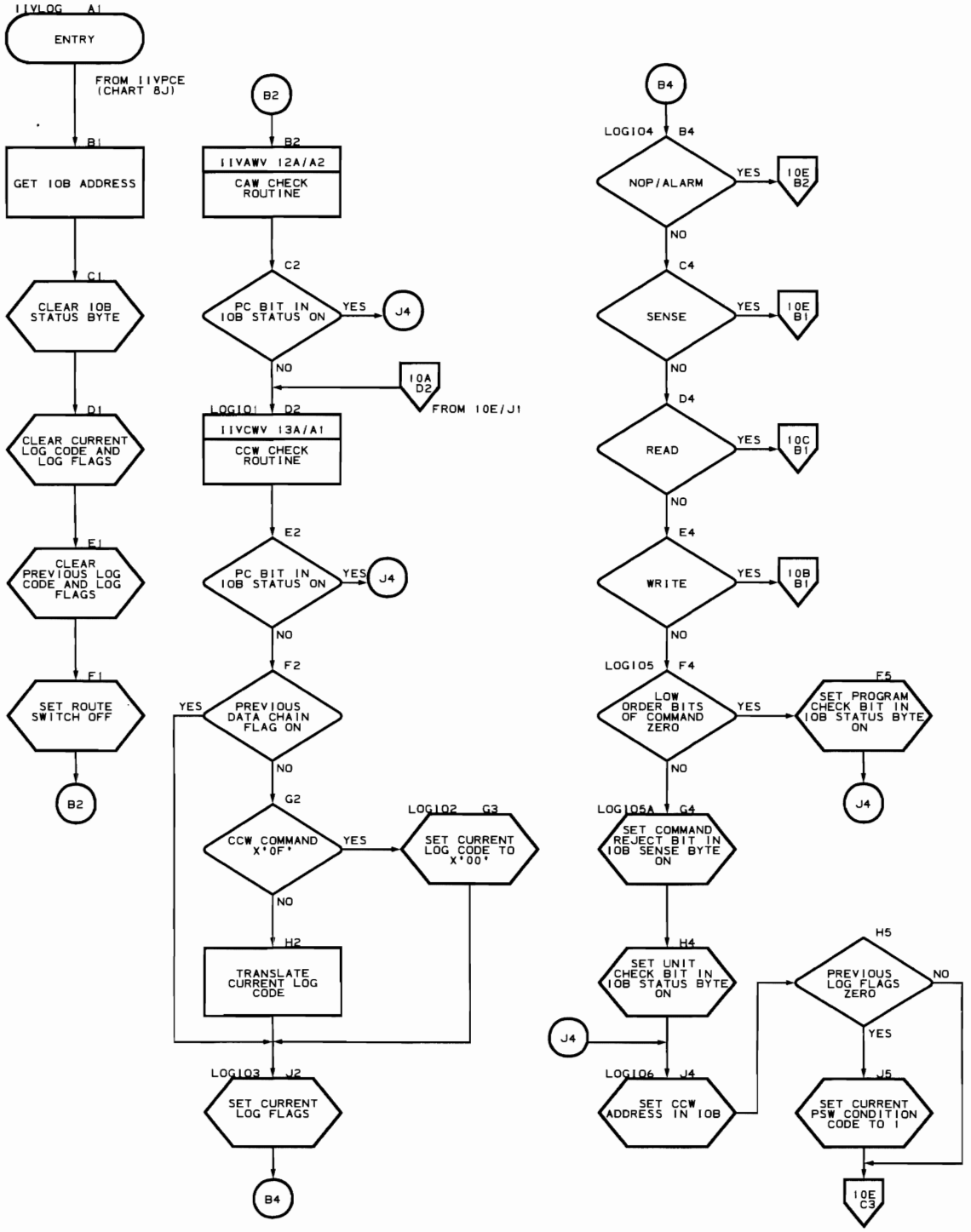
- DOSCORE:
  - CAW
  - I/O old PSW
  - I/O new PSW
  - CCW strings
  - CCW data areas
  - CCW
- COMTAB for DOSLOG: LOGIOB (dummy IOB for DOS SYSLOG device)
- Local execution list: DOS current PSW

Errors detected: CAW/CCW errors as detected by IIVAWV and IIVCWV

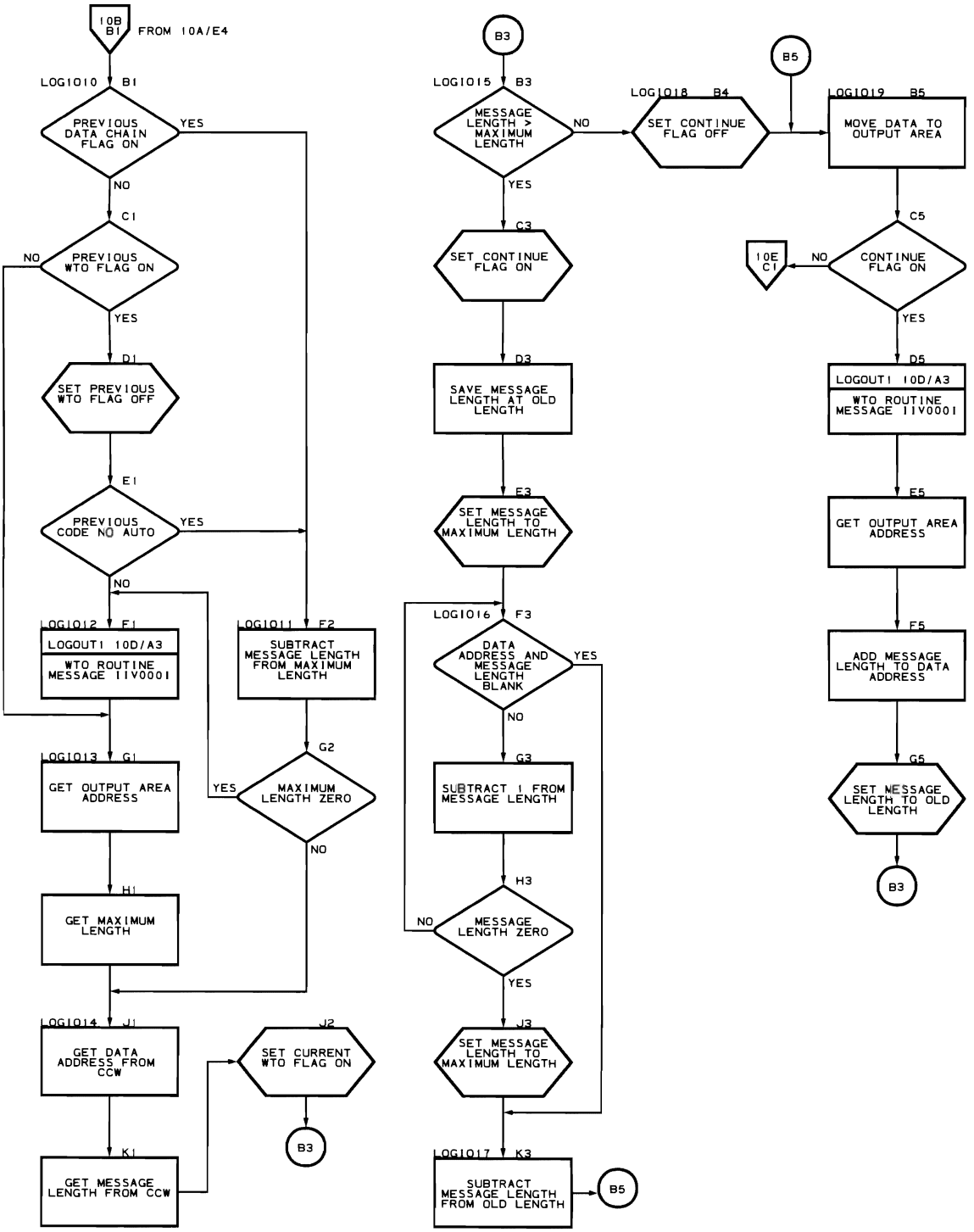
Messages issued:

- IIV000A
- IIV000I

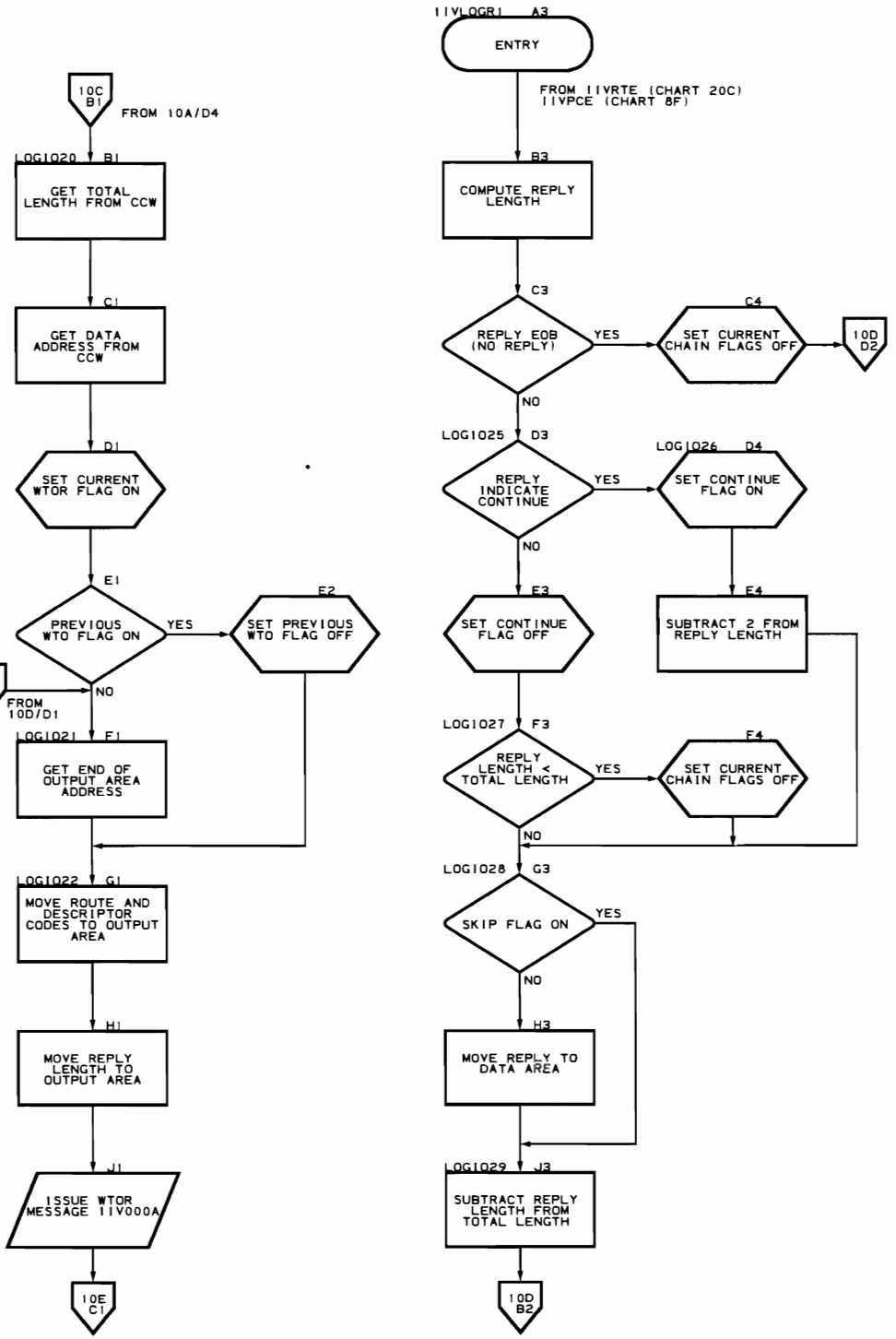
Flowchart 10A. Interpretive SYSLOG Routine (IIVLOG Part 1 of 5)



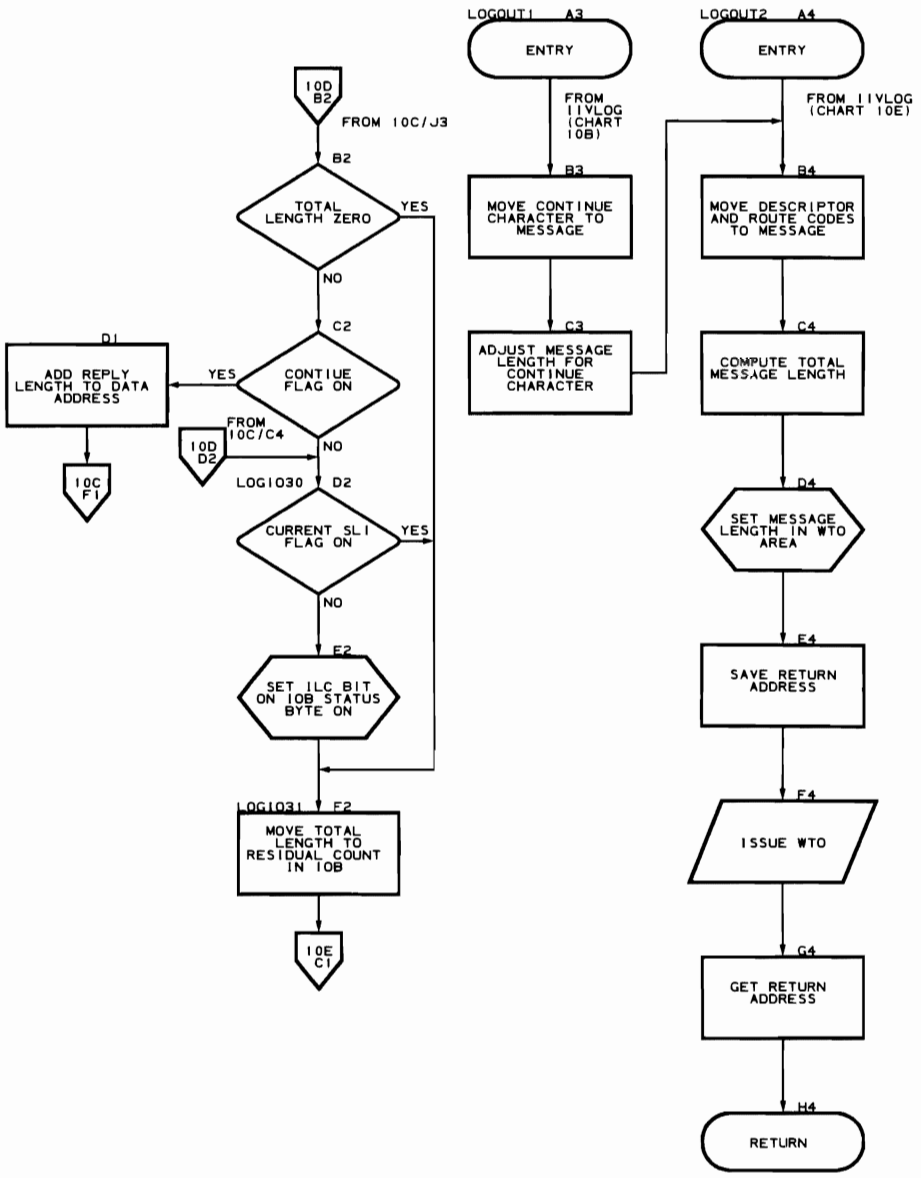
Flowchart 10B. Interpretive SYSLOG Routine (IIVLOG Part 2 of 5)



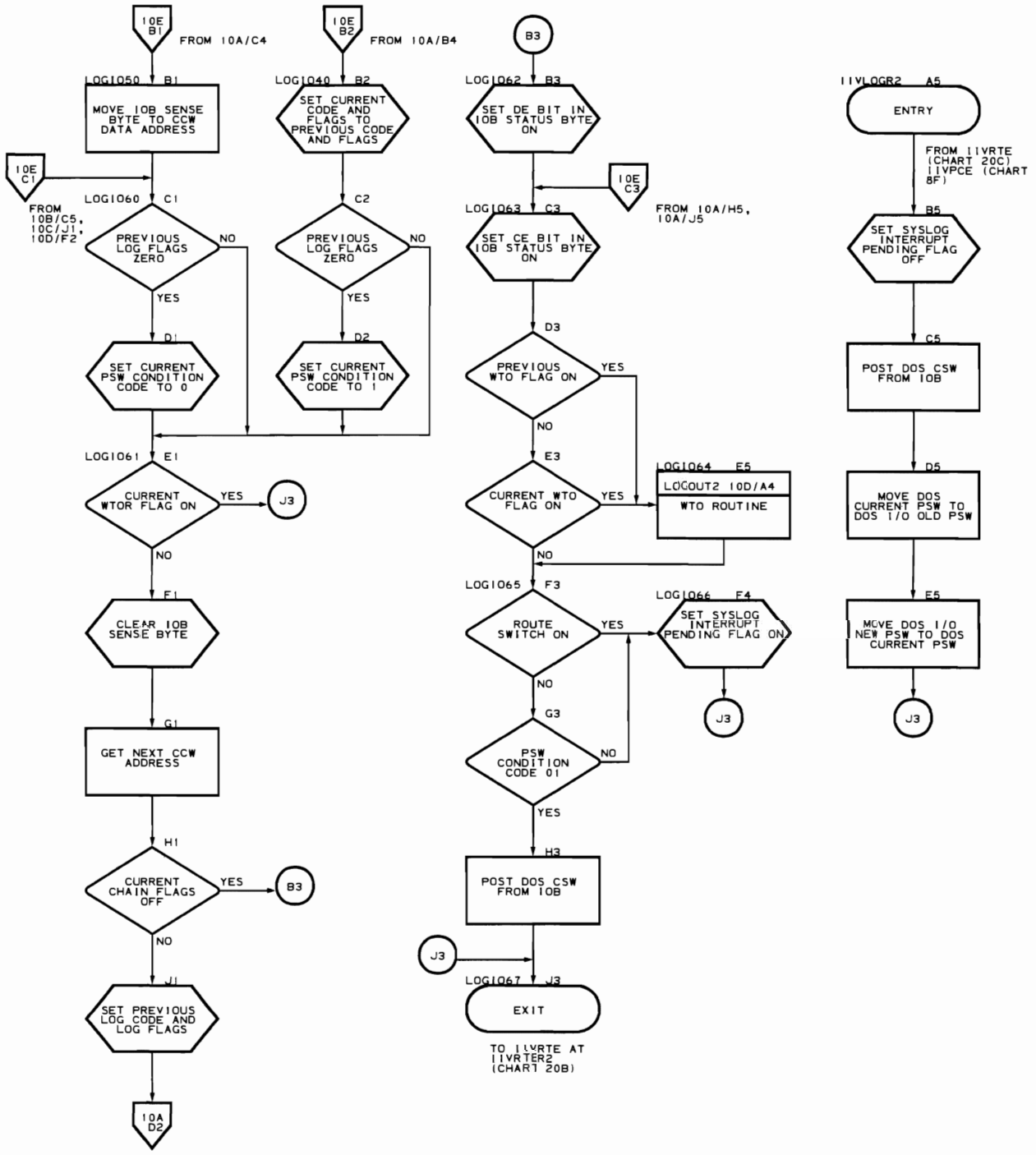
Flowchart 10C. IIVLOGR1 Subroutine (IIVLOG Part 3 of 5)



Flowchart 10D. LOGOUT1 and LOGOUT2 Subroutines (IIVLOG Part 4 of 5)



Flowchart 10E. IIVLOGR2 Subroutine (IIVLOG Part 5 of 5)



Staged I/O Routine (Flowcharts 11A-11N)

Module name: IIVSTG

Entry point names:

- IIVSTG
- EODA
- SYNA
- STGIO300

Major functions:

- Furnishes the interface between the DOS Emulator and the spooling facilities (temporary data sets) provided for unit record devices by OS
- During initialization creates an Emulator forms-control image for staged printers from an OS FCB image
- Intercepts DOS FCB load for 3211 and recreates the Emulator forms control image
- Intercepts the DOS READ FCB channel command and builds an FCB in DOS main storage from the Emulator forms control image
- Intercepts the DOS CHECK READ channel command and computes and places the value of the FCB address register in DOS main storage

Entered by:

- IIVPCE
- IIVOPN
- OS (QSAM) upon end of data
- OS (QSAM) upon occurrence of synchronous error

Modules called:

- IIVAWV
- IIVCWV
- IIVPOV
- IIVMSG

Exits to:

- IIVRTE
- OS (QSAM)

OS macros issued:

- PUT
- DCBD
- GET
- SAVE
- RETURN
- GETMAIN
- FREEMAIN

Input:

- Data is passed to DOS following an input request
- Register 0 contains the entry number into COMTAB for this I/O request. Zero indicates the first entry.
- Register 10 contains the starting address of DOS
- Register 11 contains the address of IIVCON
- Register 14 contains the address of IIVRTER2
- The OS FCB image is passed from the SYS1.IMAGELIB

Output:

- Output data from DOS that is written out to temporary data sets that are later processed by the Operating System's output writer
- Emulator forms-control image
- Operator information messages that indicate unusual occurrences (such as SYNAD conditions)

Return codes: None

Tables/work areas:

- DOSCORE - DOS low storage area
- Local execution list
- COMTAB
- STGCON - dummy IOB area
- DOS CCWs
- Emulator forms-control image

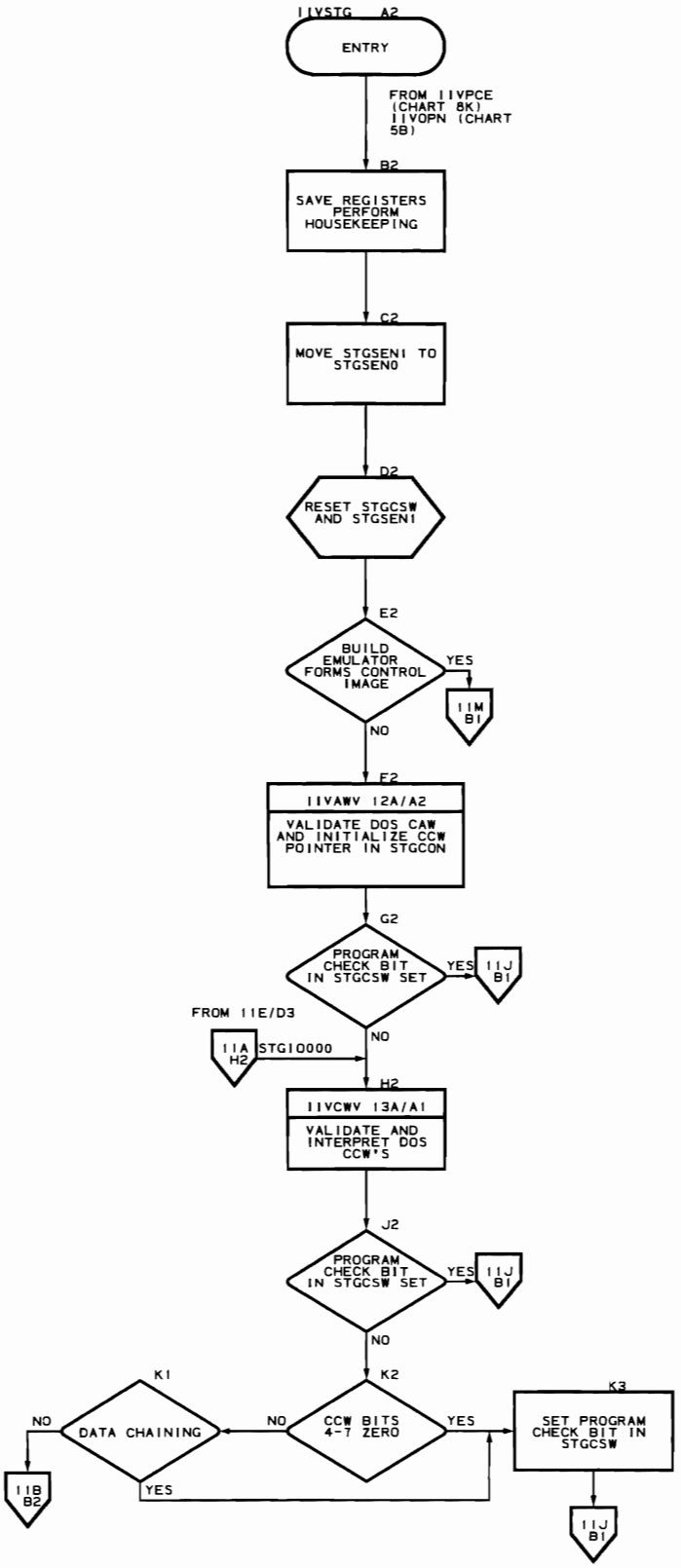
Errors detected:

- SYNAD subroutine is entered after a permanent I/O error
- Program check bit set in CSW
- Unit check bit set in CSW if error detected in OS FCB image
- Invalid command sequences for staged 2540 Reader
- Command reject condition for all staged devices

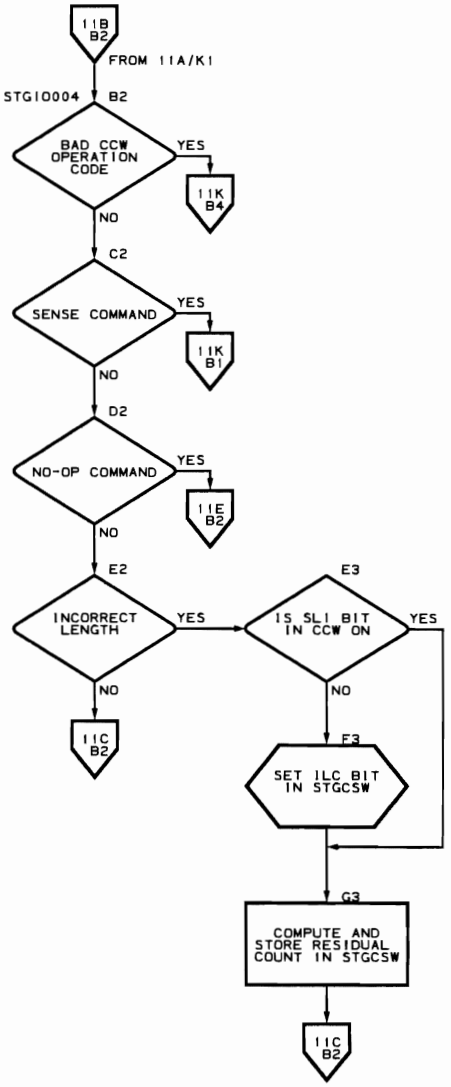
Messages requested or issued:

- IIV140I
- IIV275I

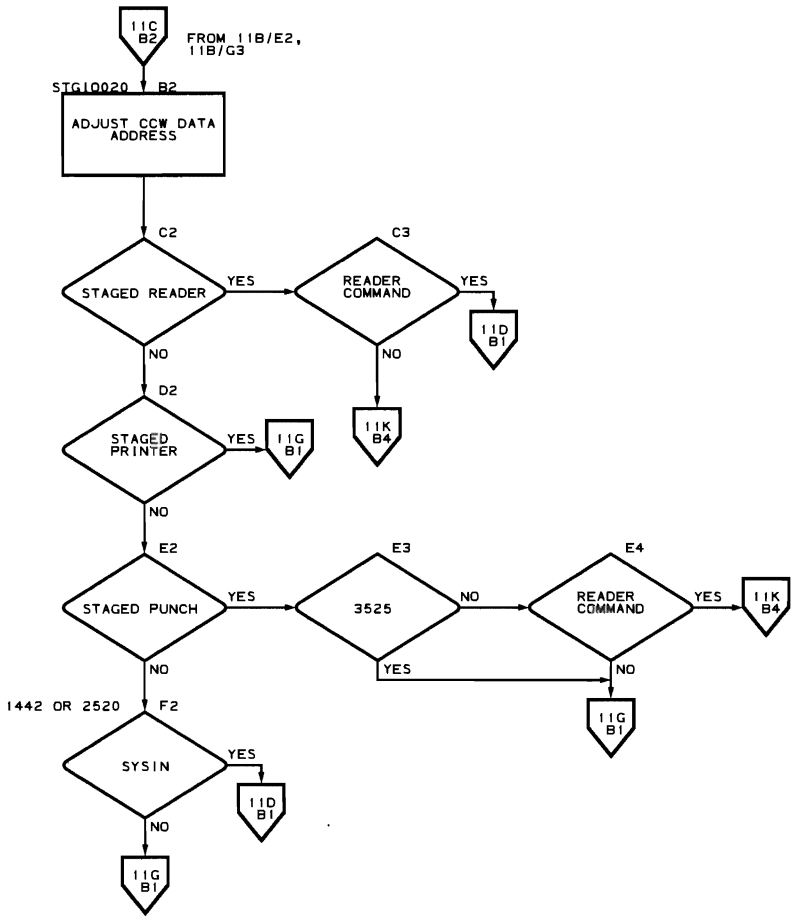
Flowchart 11A. Staged I/O Routine (IIVSTG Part 1 of 13)



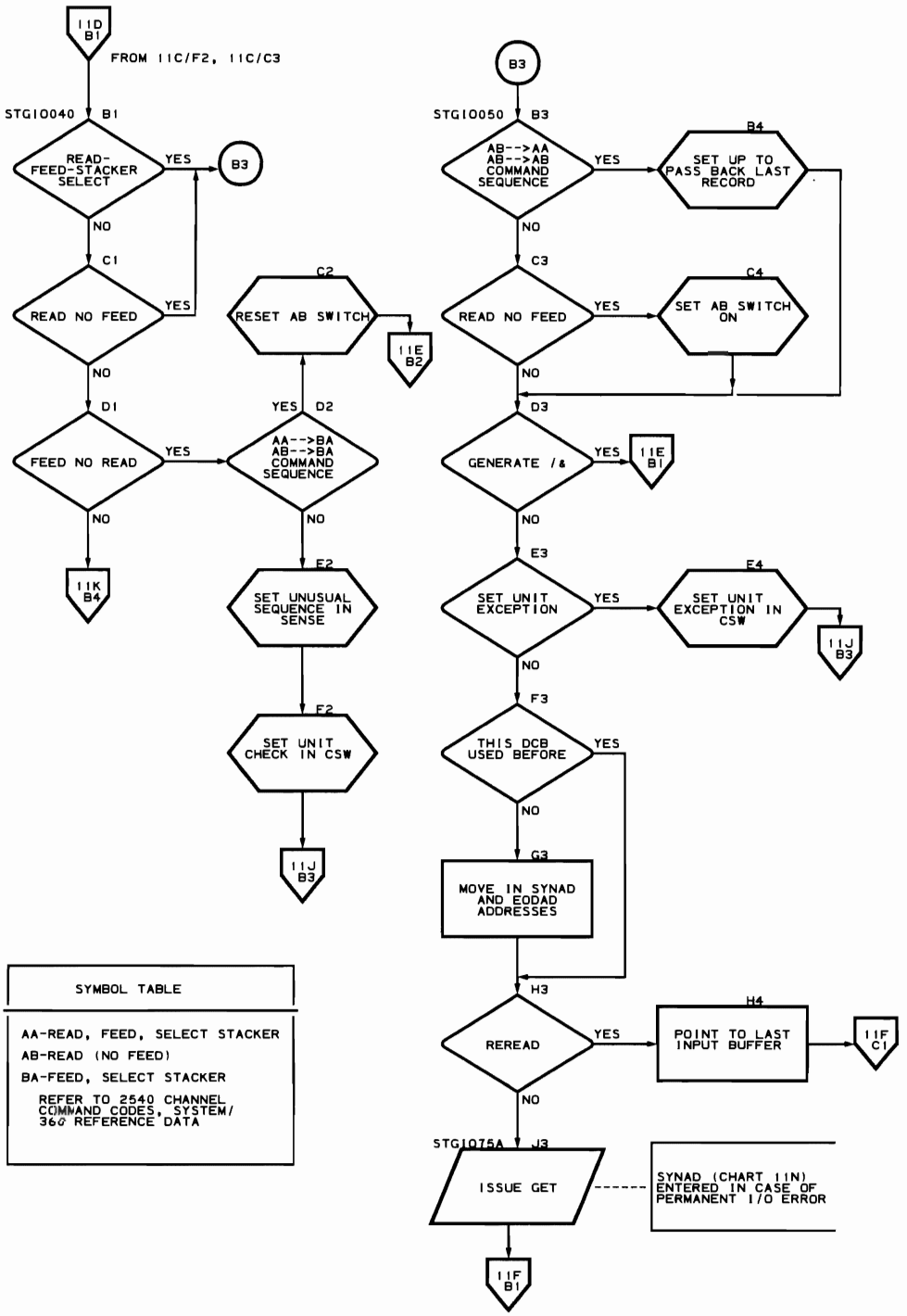
Flowchart 11B. Staged I/O Routine (IIVSTG Part 2 of 13)



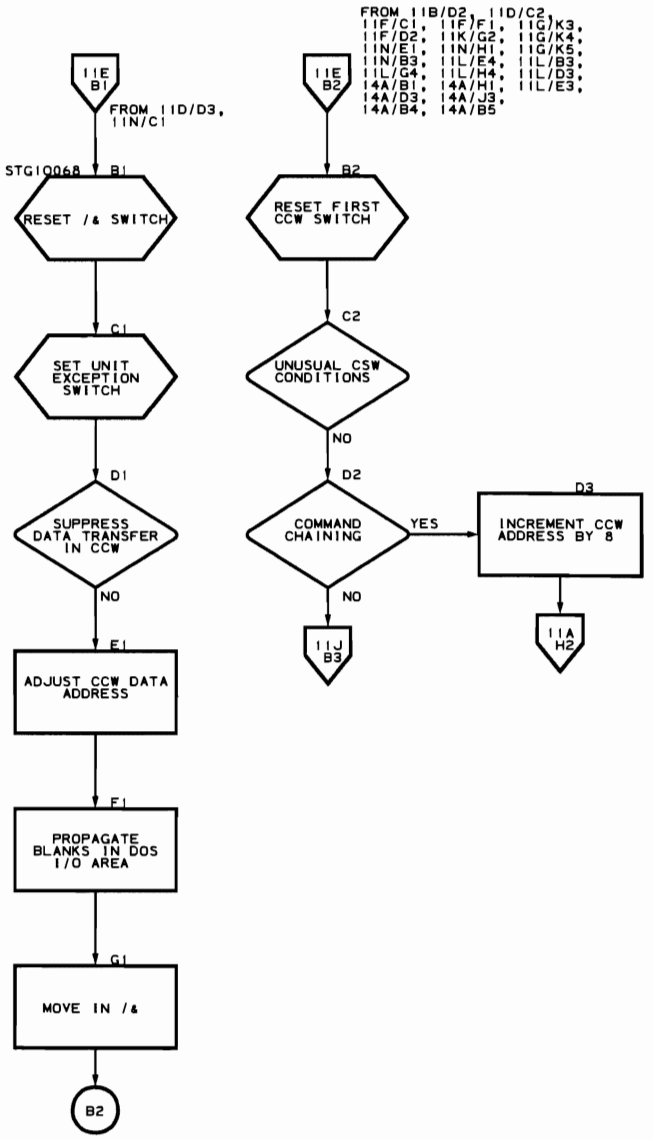
Flowchart 11C. Staged I/O Routine (IIVSTG Part 3 of 13)



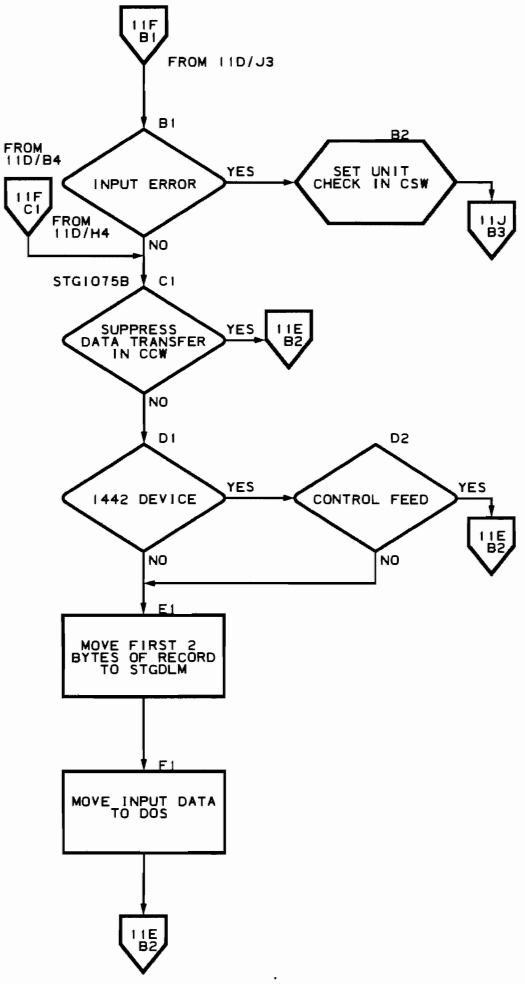
Flowchart 11D. Staged I/O Routine (IIVSTG Part 4 of 13)



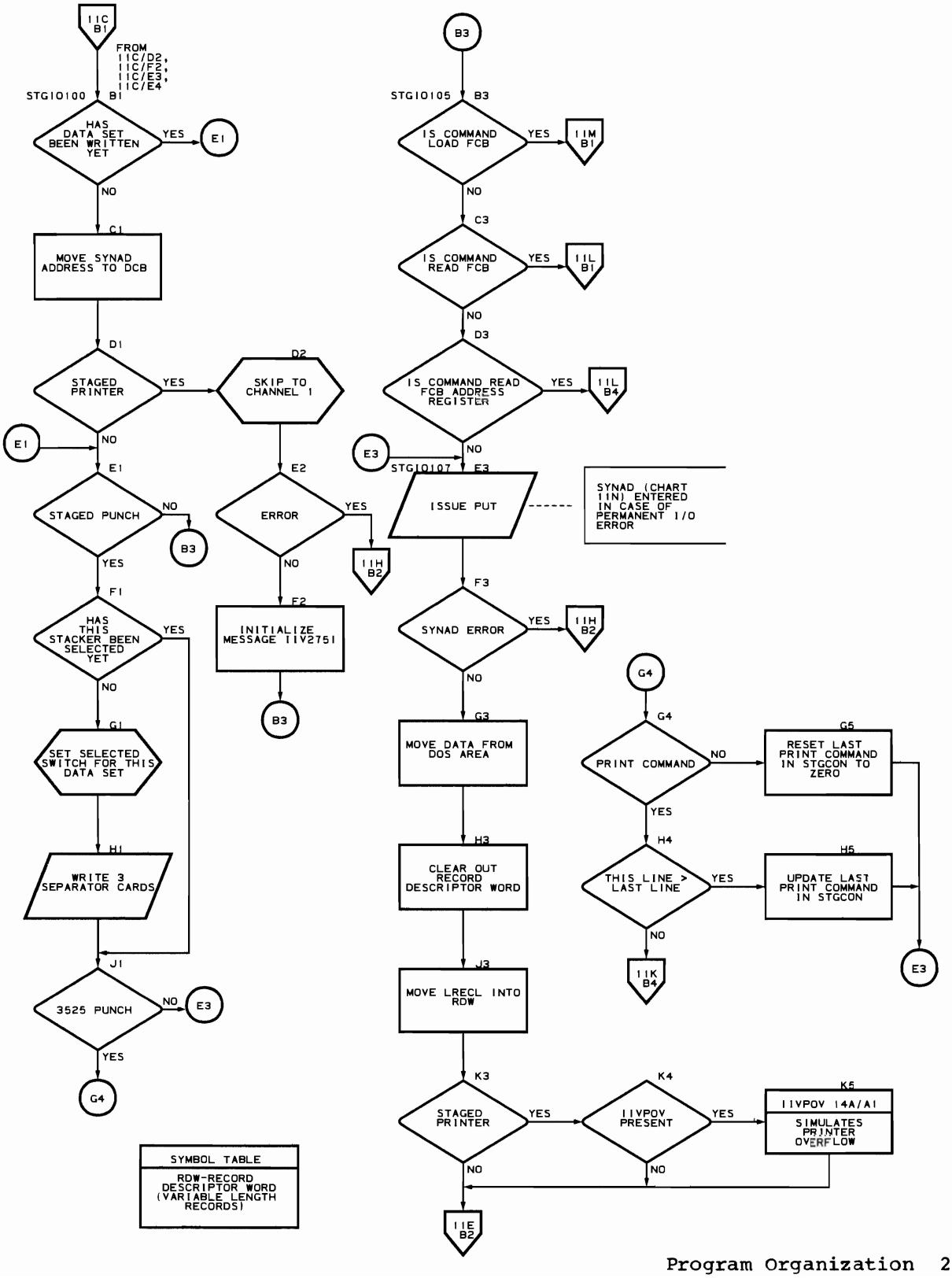
Flowchart 11E. Staged I/O Routine (IIVSTG Part 5 of 13)



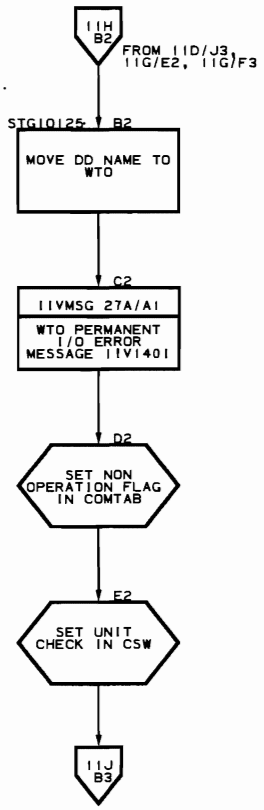
Flowchart 11F. Staged I/O Routine (IIVSTG Part 6 of 13)



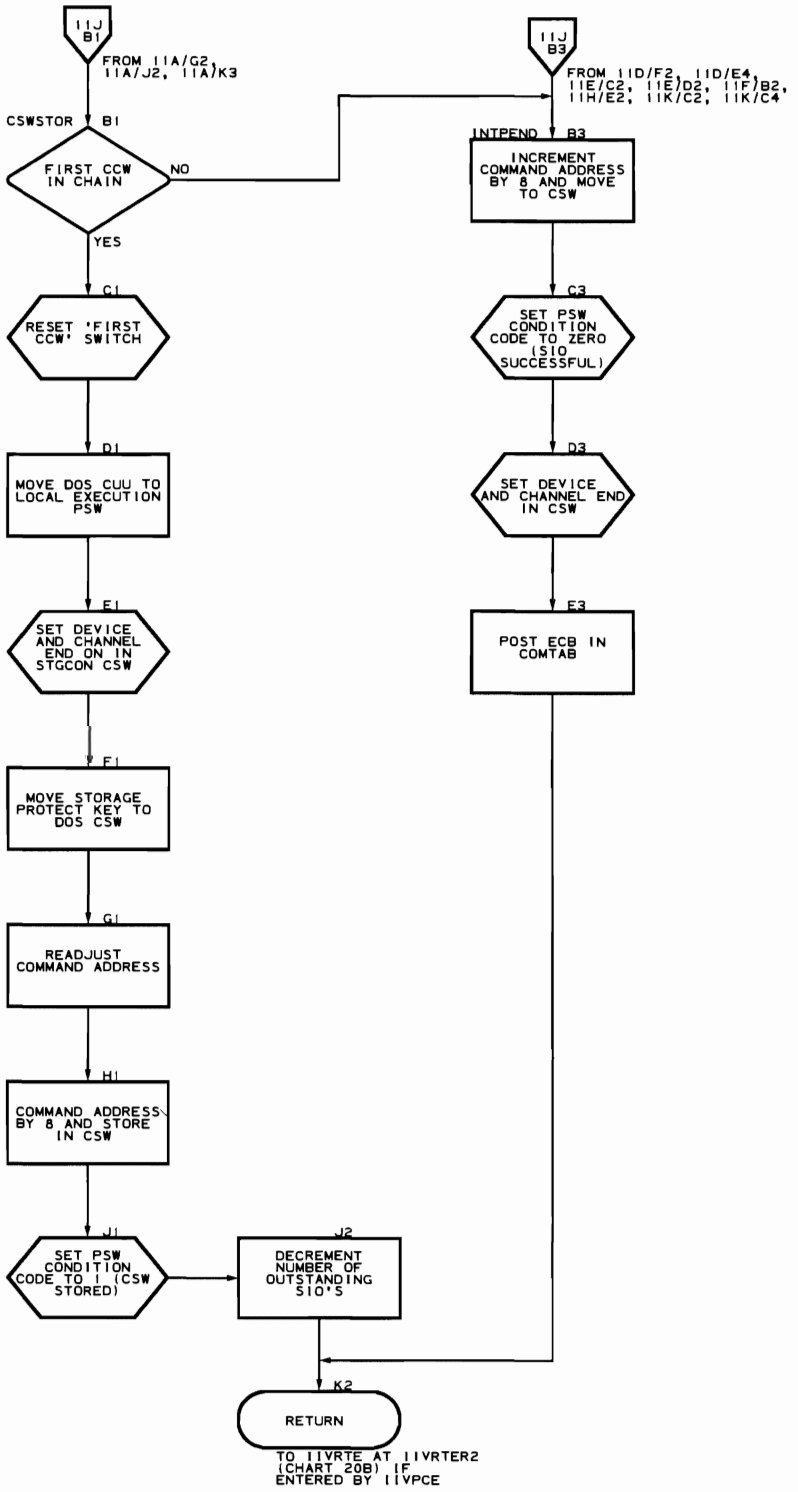
Flowchart 11G. Staged I/O Routine (IIVSTG Part 7 of 13)



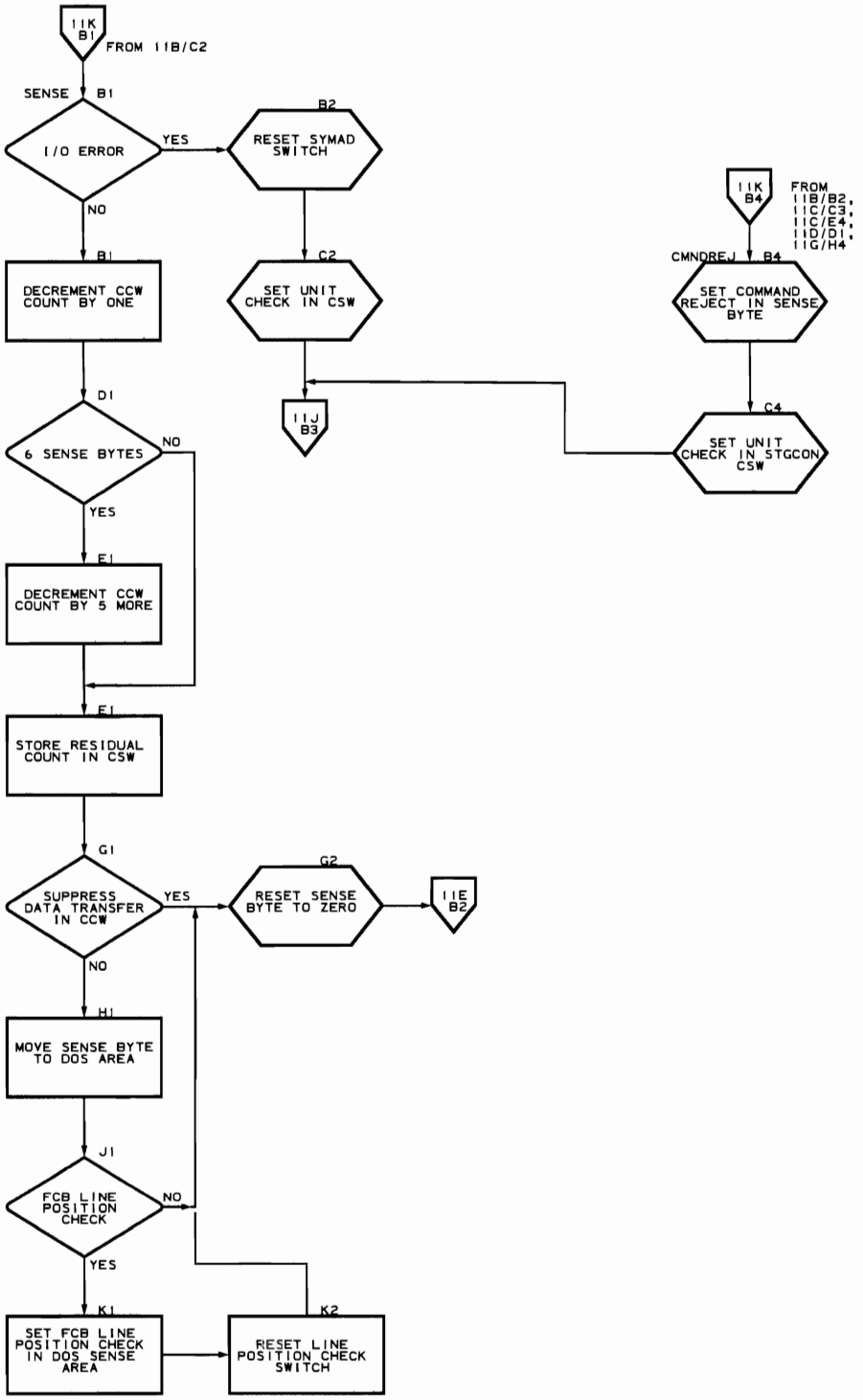
Flowchart 11H. Staged I/O Routine (IIVSTG Part 8 of 13)



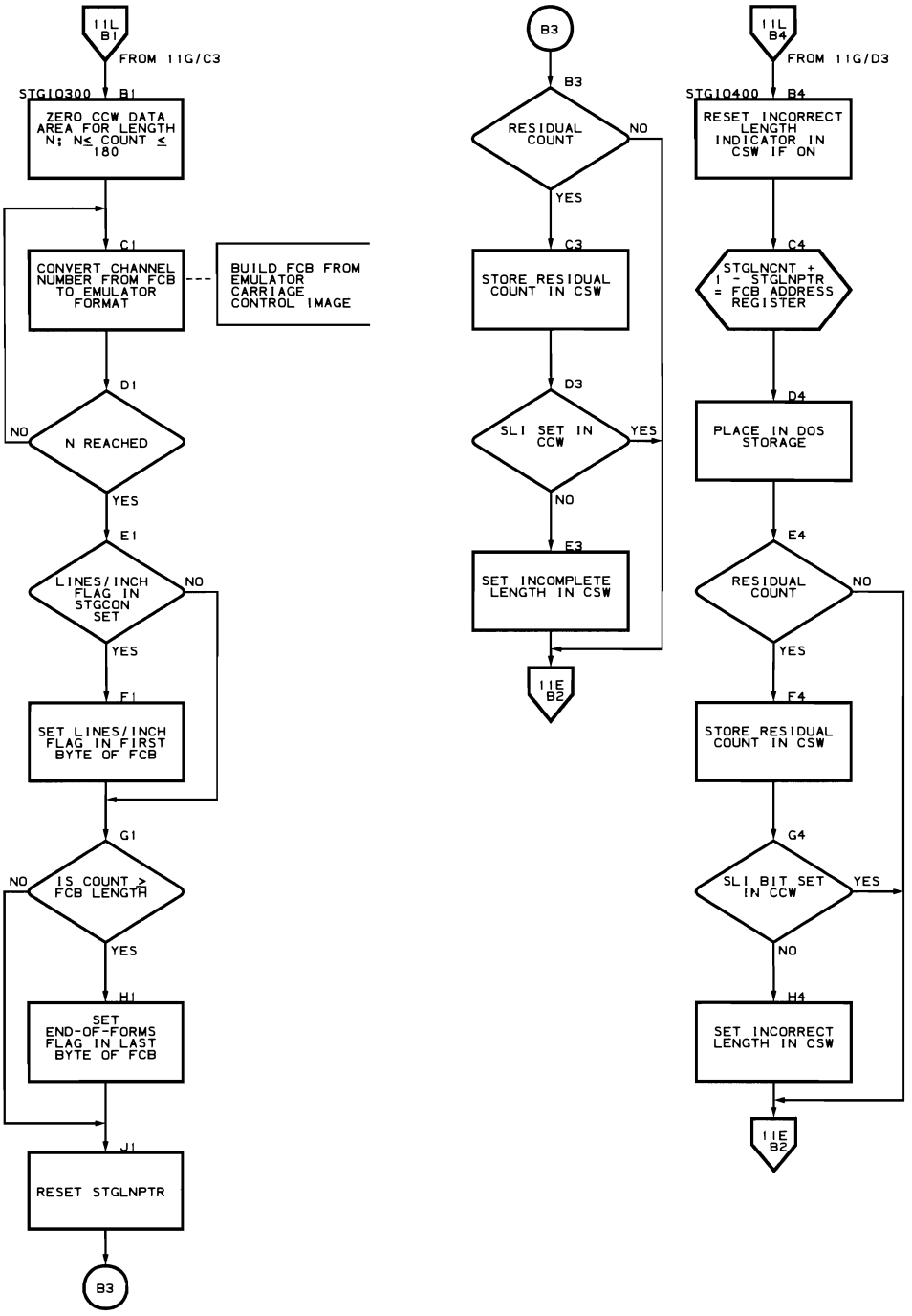
Flowchart 11J. Staged I/O Routine (IIVSTG Part 9 of 13)



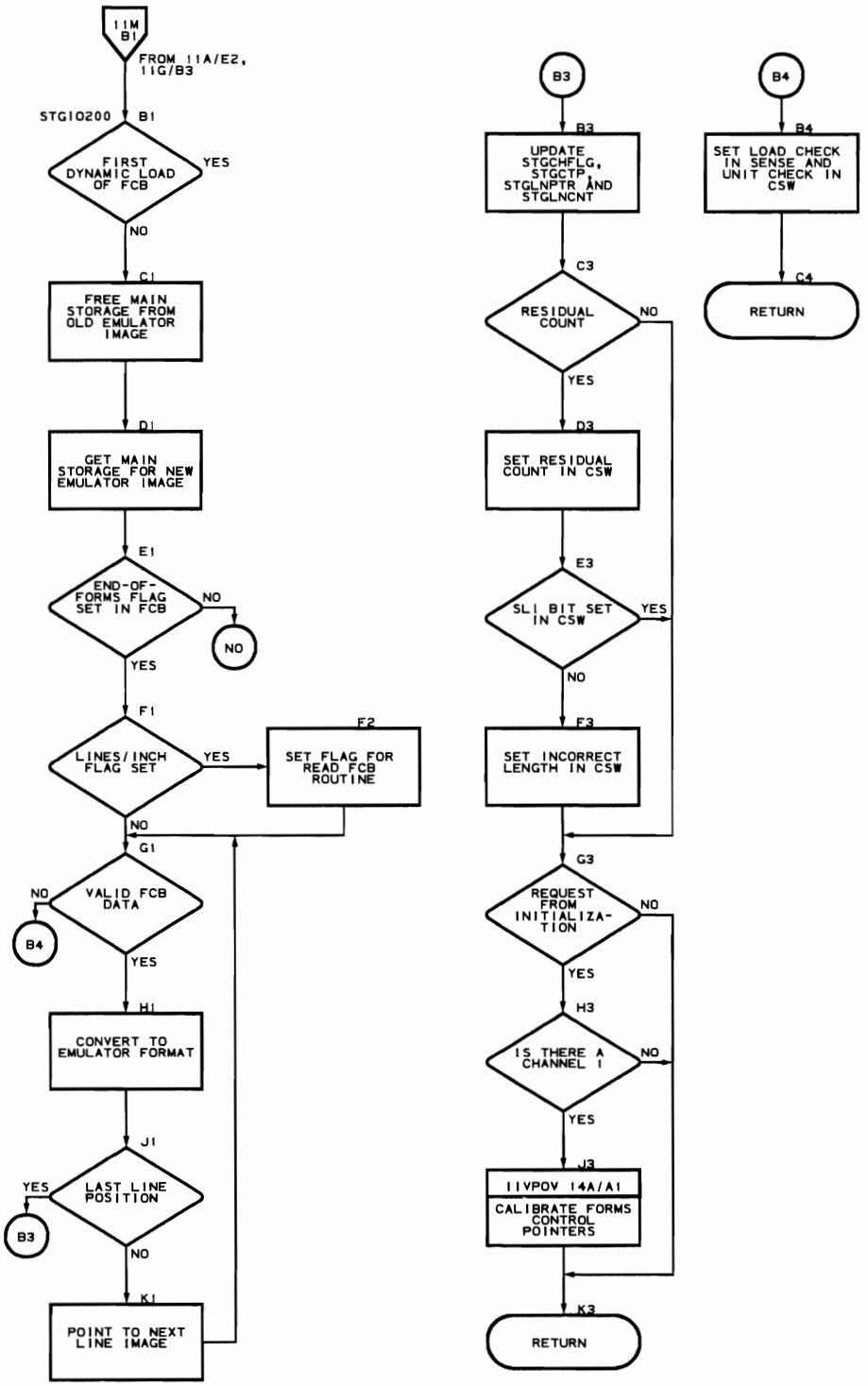
Flowchart 11K. Staged I/O Routine (IIVSTG Part 10 of 13)



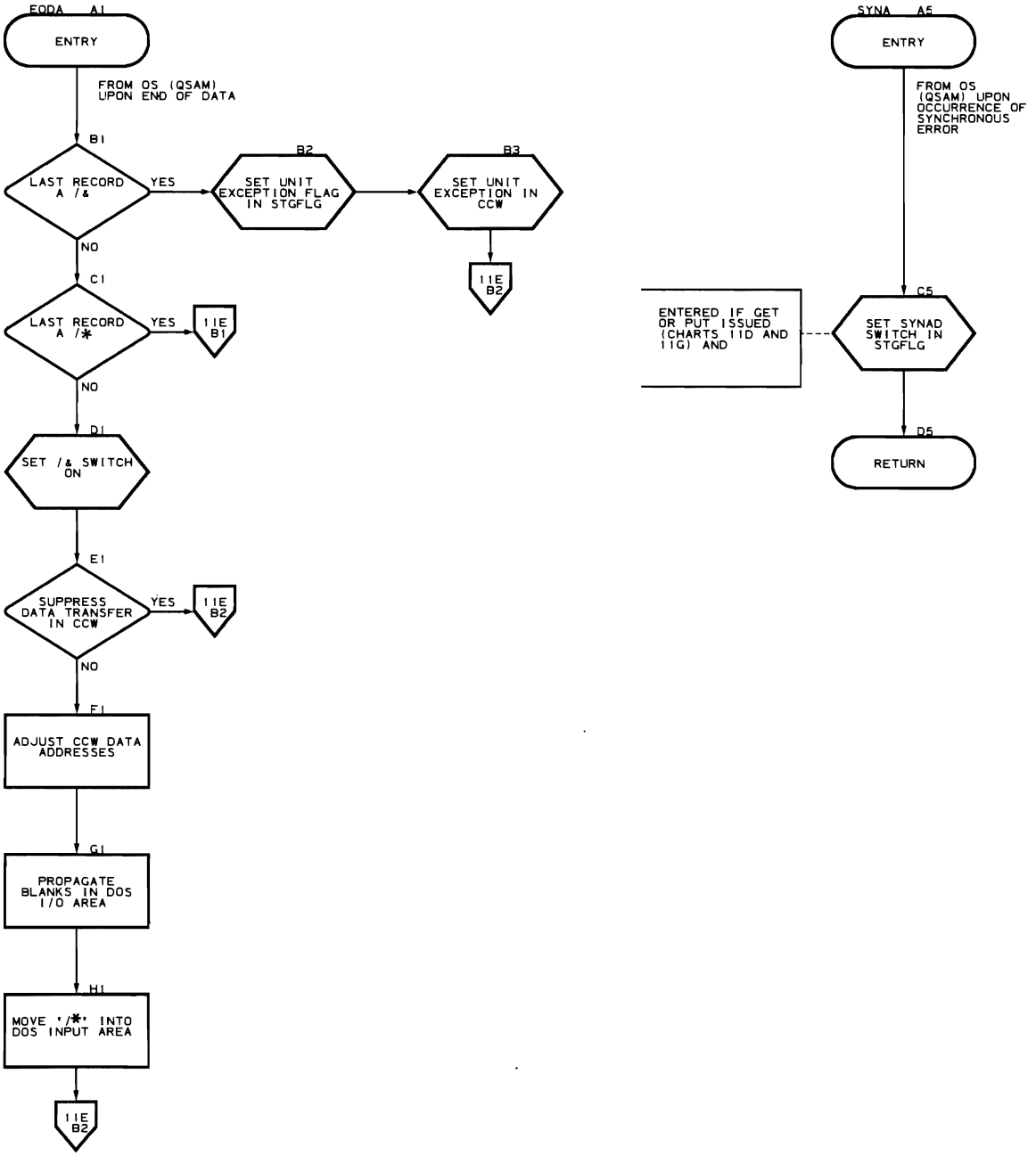
Flowchart 11L. Read FCB Subroutine (IIVSTG Part 11 of 13)



Flowchart 11M. Load FCB Subroutine (IIVSTG Part 12 of 13)



Flowchart 11N. EODAD and SYNAD Subroutines (IIVSTG Part 13 of 13)



CAW Verification Routine (Flowchart 12A)

Module name: IIVAWV

Entry point name: IIVAWV

Major functions: Validates the DOS CAW and initializes the CCW pointer in STGCON for modules IIVSTG and IIVLOG

Entered by:

- IIVSTG
- IIVLOG

Modules called: None

Exits to: Caller

OS macros issued: None

Input: DOSCORE (DOS CAW)

Output:

- COMTAB (storage protection key)
- STGCON (CCW pointer, CSW)

Return codes: None

Tables/work areas:

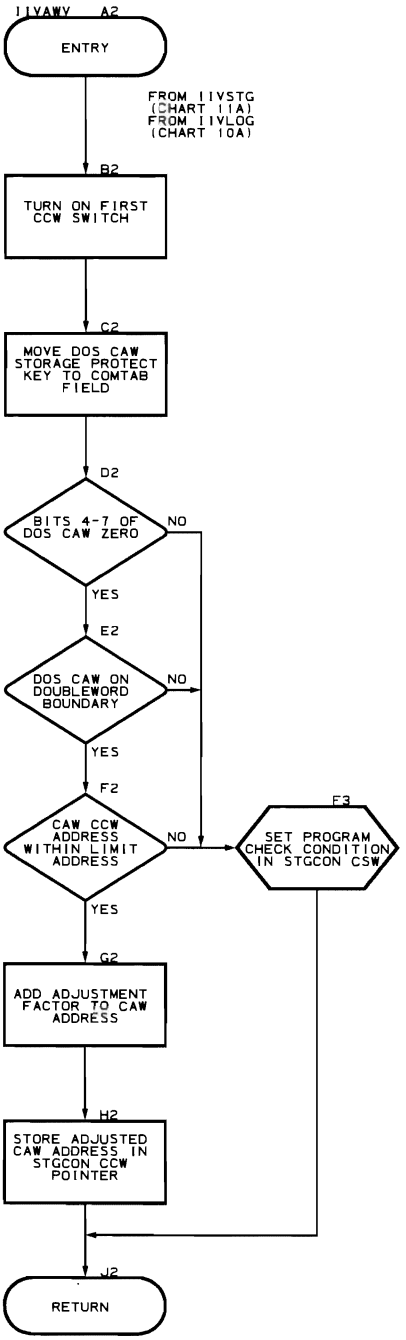
- STGCON - dummy IOB area
- DOSCORE - DOS low storage
- Local execution list
- COMTAB

Errors detected:

- Nonzero value in bits 4-7 of DOS CAW
- Command address not on a doubleword boundary
- Command address greater than the limit address in local execution list

Messages requested: None

Flowchart 12A. CAW Verification Routine (IIVAWV)



CCW Verification Routine (Flowchart 13A)

Module name: IIVCWV

Entry point name: IIVCWV

Major functions:

- Validates and interprets DOS CCWs for modules IIVSTG and IIVLOG
- Changes pointers from TIC CCWs to TIC-TO-CCWs

Entered by:

- IIVSTG
- IIVLOG

Modules called: None

Exits to: Caller

OS macros issued: None

Input:

- STGCON: STGFLG (first CCW), STGCCW (CCW pointer)
- DOS: CCW
- Local execution list: DRILIMAD (limit address of DOS)

Output:

STGCON: STGCCW (CCW pointer), STGCSW (channel status word)

Return codes: None

Tables/work areas:

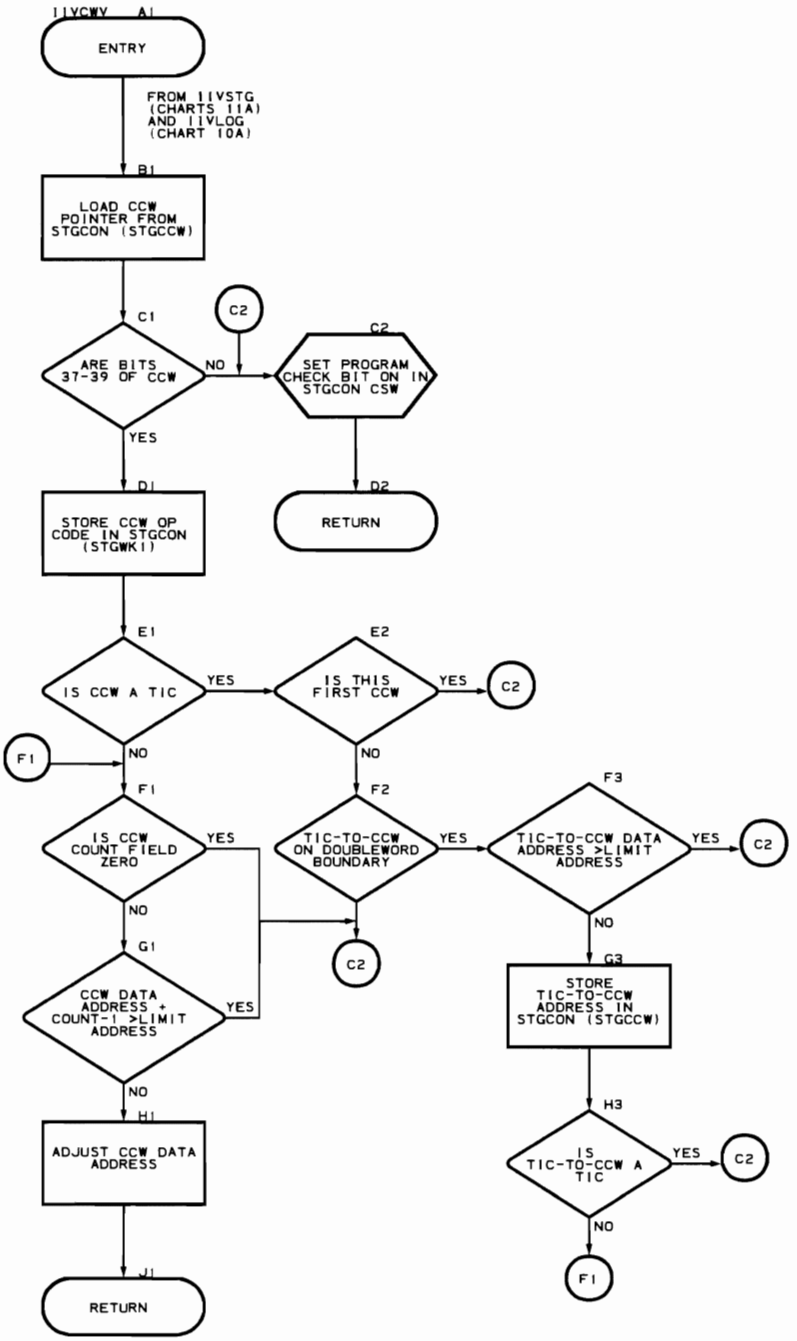
- STGCON (dummy IOB area)
- Local execution list

Errors detected:

- Nonzero value is contained in bits 37-39 of CCW
- First CCW (CCW addressed by CAW) is a TIC
- Count field in CCW is zero
- TIC/TIC sequence of CCWs was encountered
- Data address and count greater than the limit address in the local execution list
- Command address in a TIC CCW is greater than the limit address in the local execution list
- Command address in a TIC CCW is not on a doubleword boundary

Messages requested: None

Flowchart 13A. CCW Verification Routine (IIVCWV)



Printer Overflow Routine (Flowchart 14A)

Module name: IIVPOV

Entry point name: IIVPOV

Major functions:

- Maintains the simulated sense byte and status portions of the CSW for staged printer devices
- Simulates the carriage tape operation normally performed by the printer during output staging

Entered by:

- IIVSTG

Modules called: None

Exits to: Caller

OS macros issued: None

Input: CCW operation code (STGCCW)

Output:

- Status portion of CSW
- Sense byte (STGSEN1)

Return codes: None

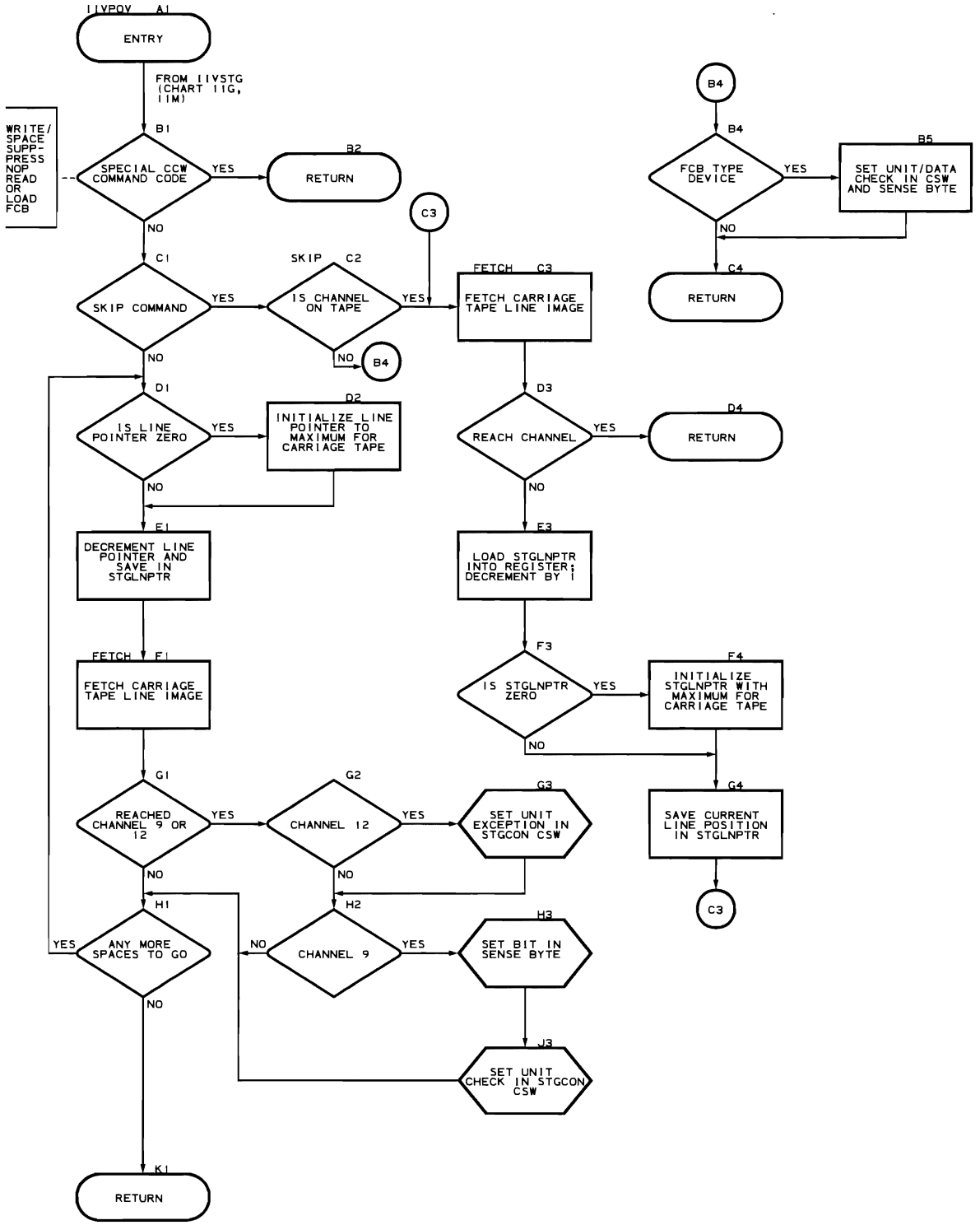
Tables/work areas:

- Emulator forms-control image created by IIVSTG
- STGLNPTR index into Emulator forms-control image created by IIVSTG
- STGCSW channel status word (STGCON)
- STGSEN1 sense byte (STGCON)
- STGLNCNT lines/page (STGCON)
- STGCHFLG printer carriage tape channel flags (STGCON)
- STGCTP address of Emulator forms-control image created by IIVSTG

Errors detected: None

Messages requested: None

Flowchart 14A. Printer Overflow Routine (IIVPOV)



Adjust CCW Data Address Routine (Flowcharts 15A-15F)

Module name: IIVCCW

Entry point name:

- IIVCCW
- RCCWAB4

Major functions: Adjusts data addresses in a string of CCWs prior to their being passed to OS for an I/O operation and after completion of the CCW function in OS

Entered by:

- IIVRTE
- IIVPCE
- IIVCHK

Modules called: IIVADJ

Exits to:

- IIVPCE
- IIVCHK
- Caller

OS macros issued: SAVE

Input:

- Adjustment factor
- Pointer to a CCW string
- Pointer to the local execution list
- Pointer to the Emulator constants area (IIVCON)
- Pointer to COMTAB
- Local limit address

Output: Adjusted CCW string

Return codes:

- Error code 16 for IIVABN
- Error code 20 for IIVABN

Tables/work areas:

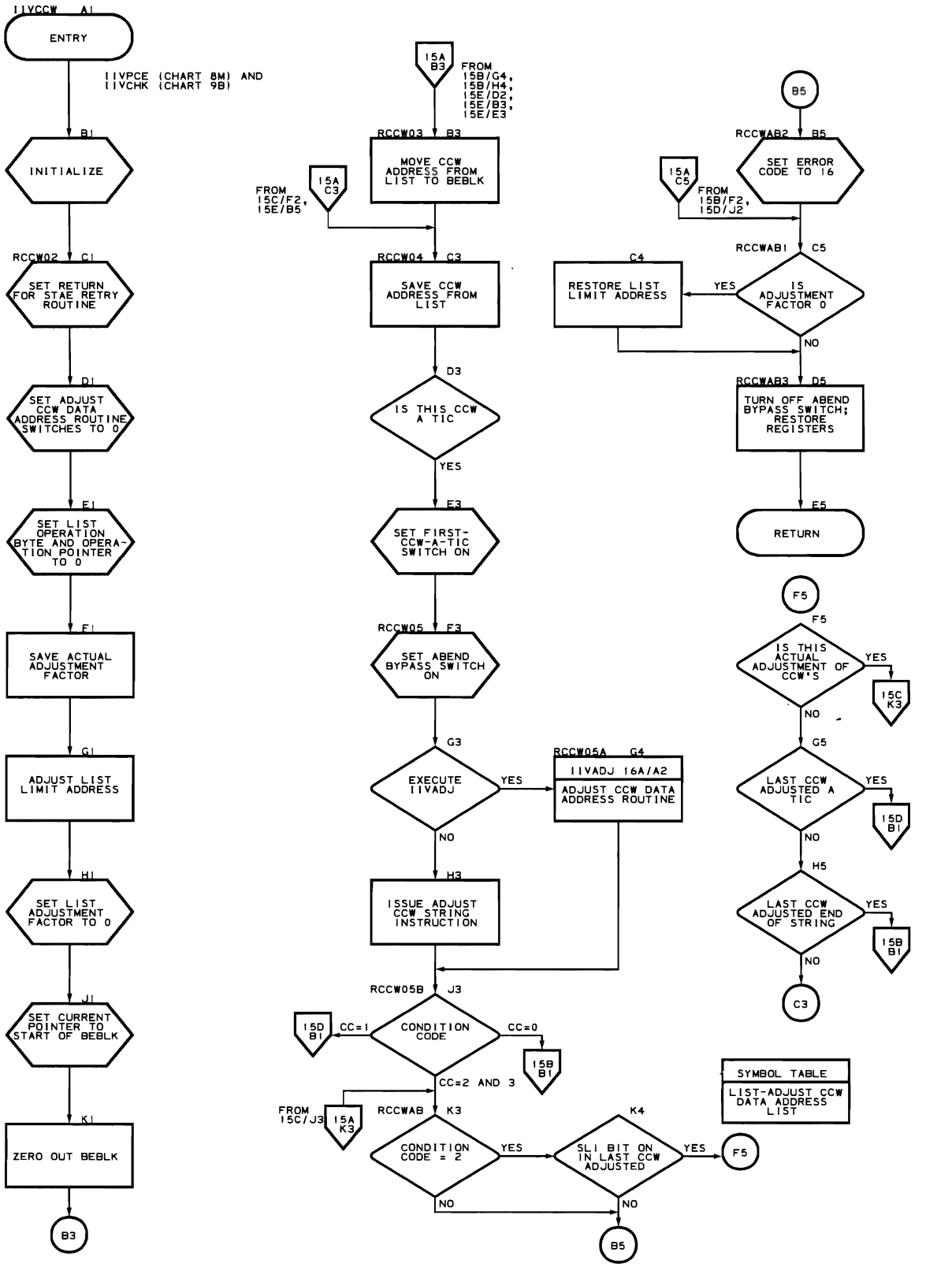
- Adjust CCW data address list (located in IIVCON)
- BEBLK CCW addresses for adjusting
- Status modifier table used to find status modifier-type CCWs
- RCCWSAVE register save area

Errors detected:

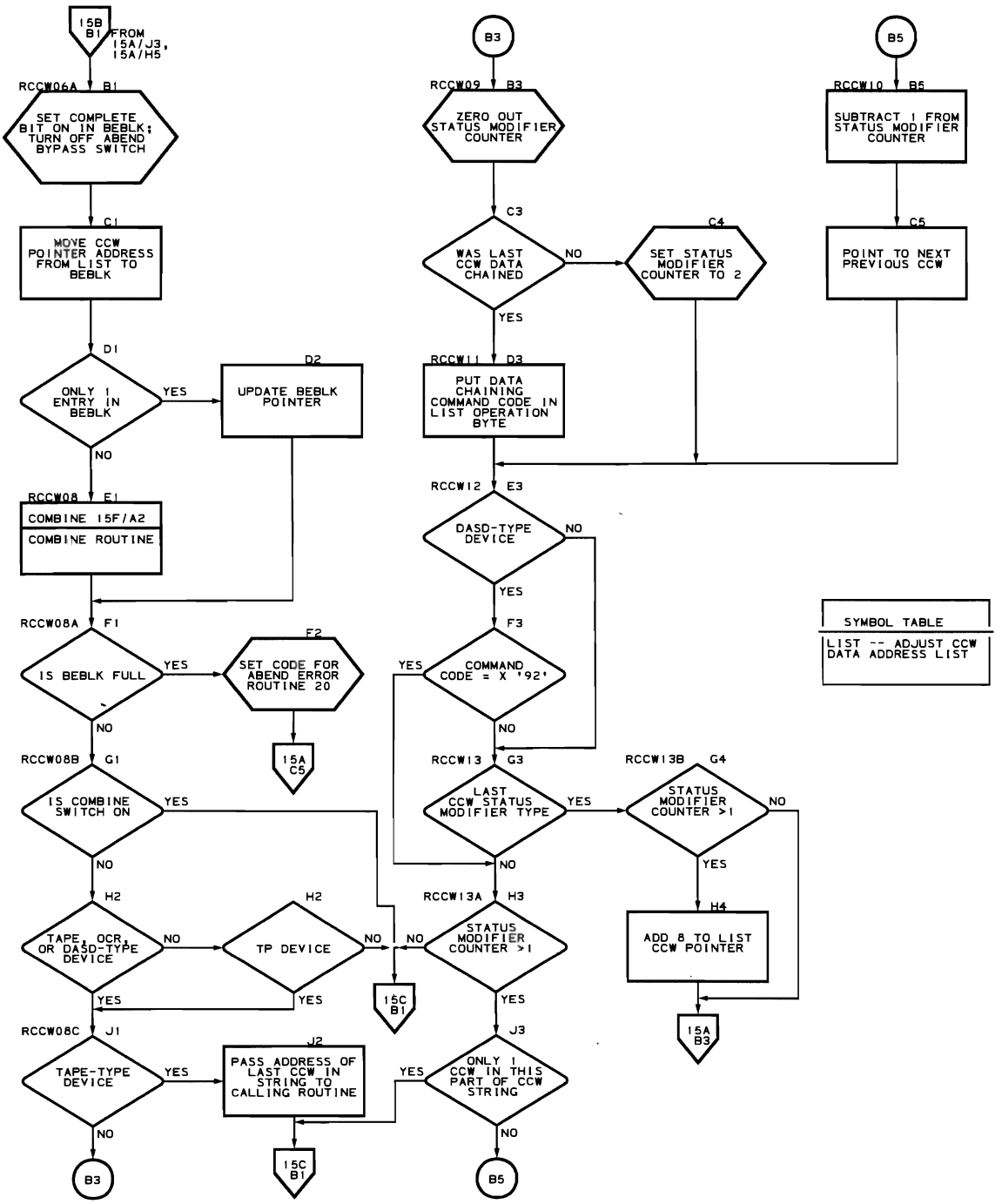
- Adjusted addresses not within DOS storage area
- BEBLK is filled and CCW adjusting cannot proceed

Messages requested: None

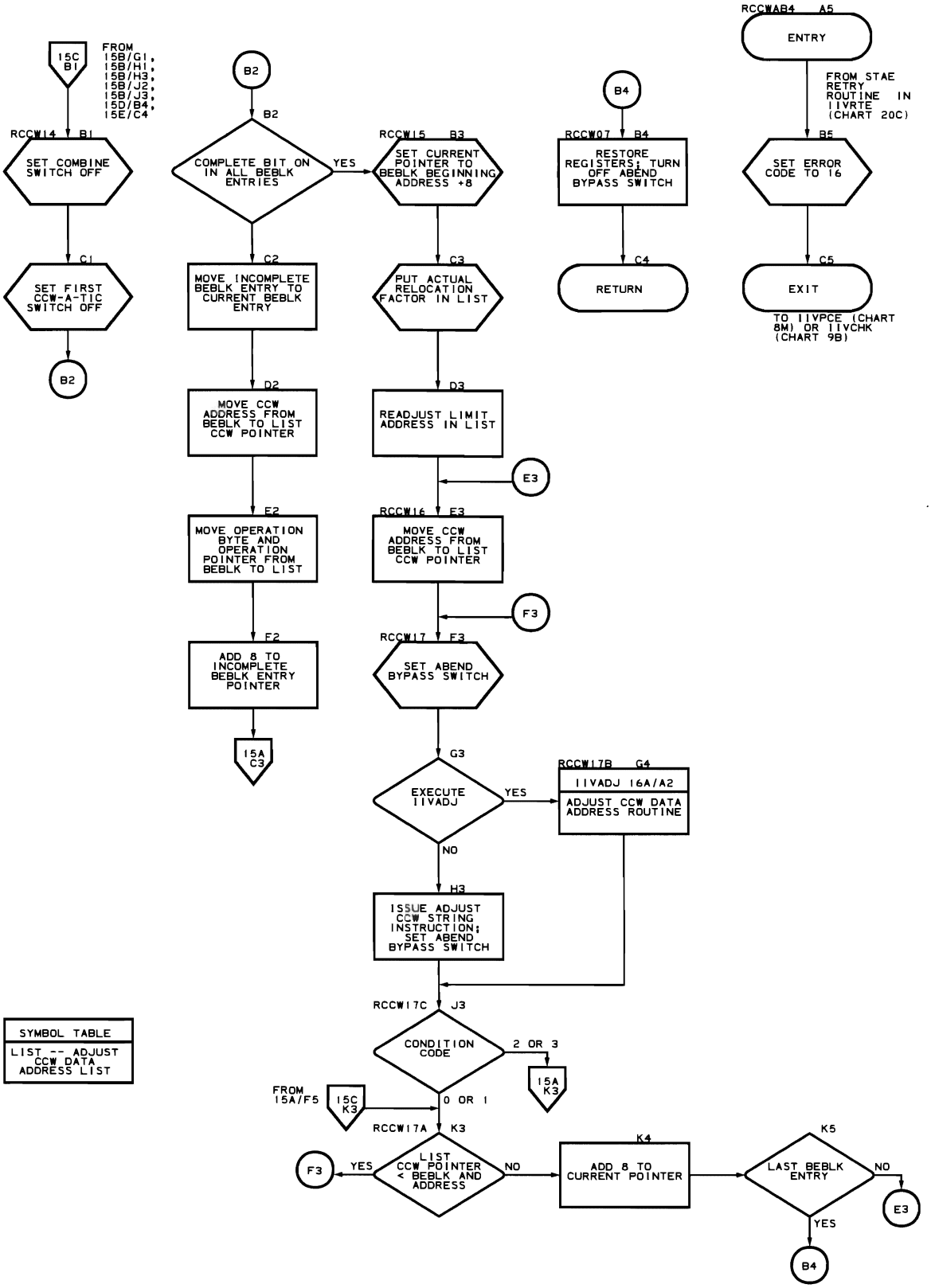
Flowchart 15A. Adjust CCW Data Address Routine (IIVCCW Part 1 of 6)



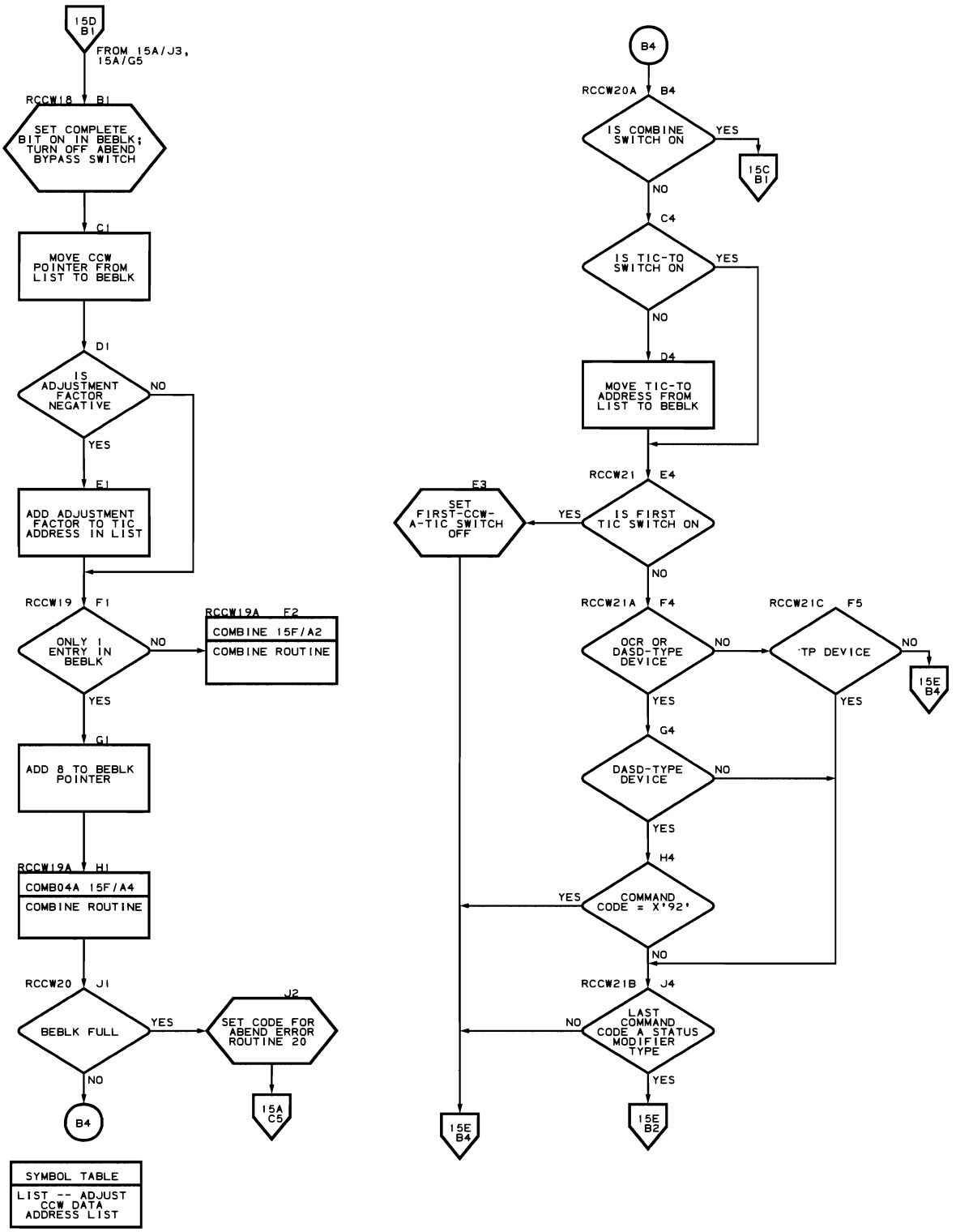
Flowchart 15B. Adjust CCW Data Address Routine (IIVCCW Part 2 of 6)



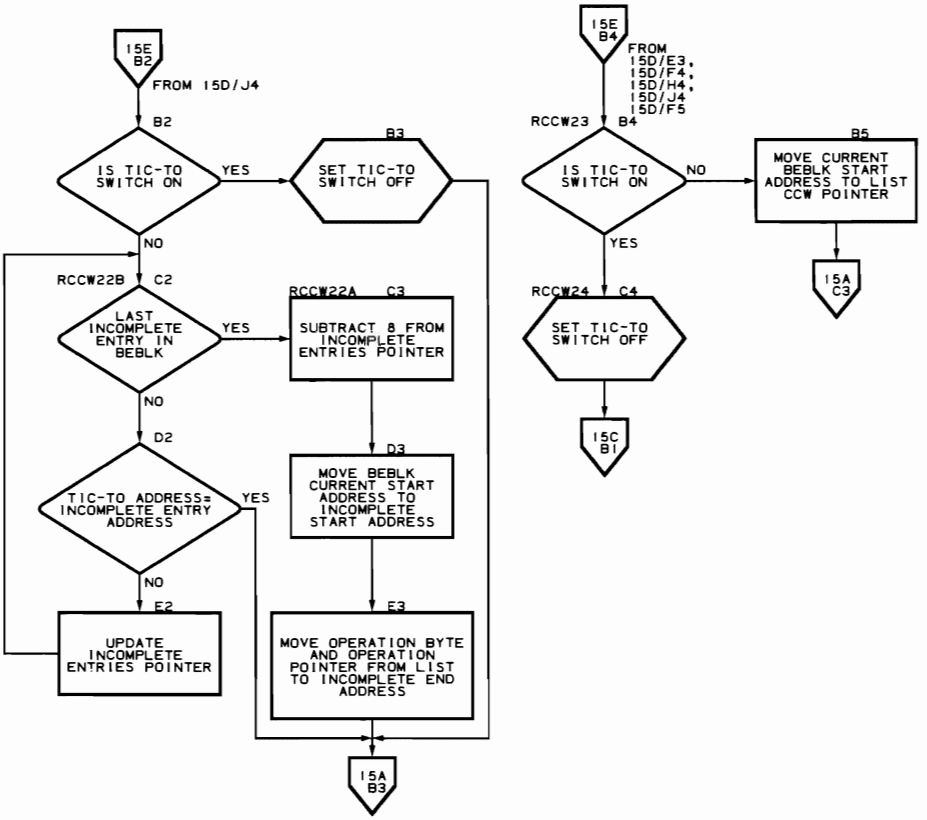
Flowchart 15C. Adjust CCW Data Address Routine (IIVCCW Part 3 of 6)



Flowchart 15D. Adjust CCW Data Address Routine (IIVCCW Part 4 of 6)

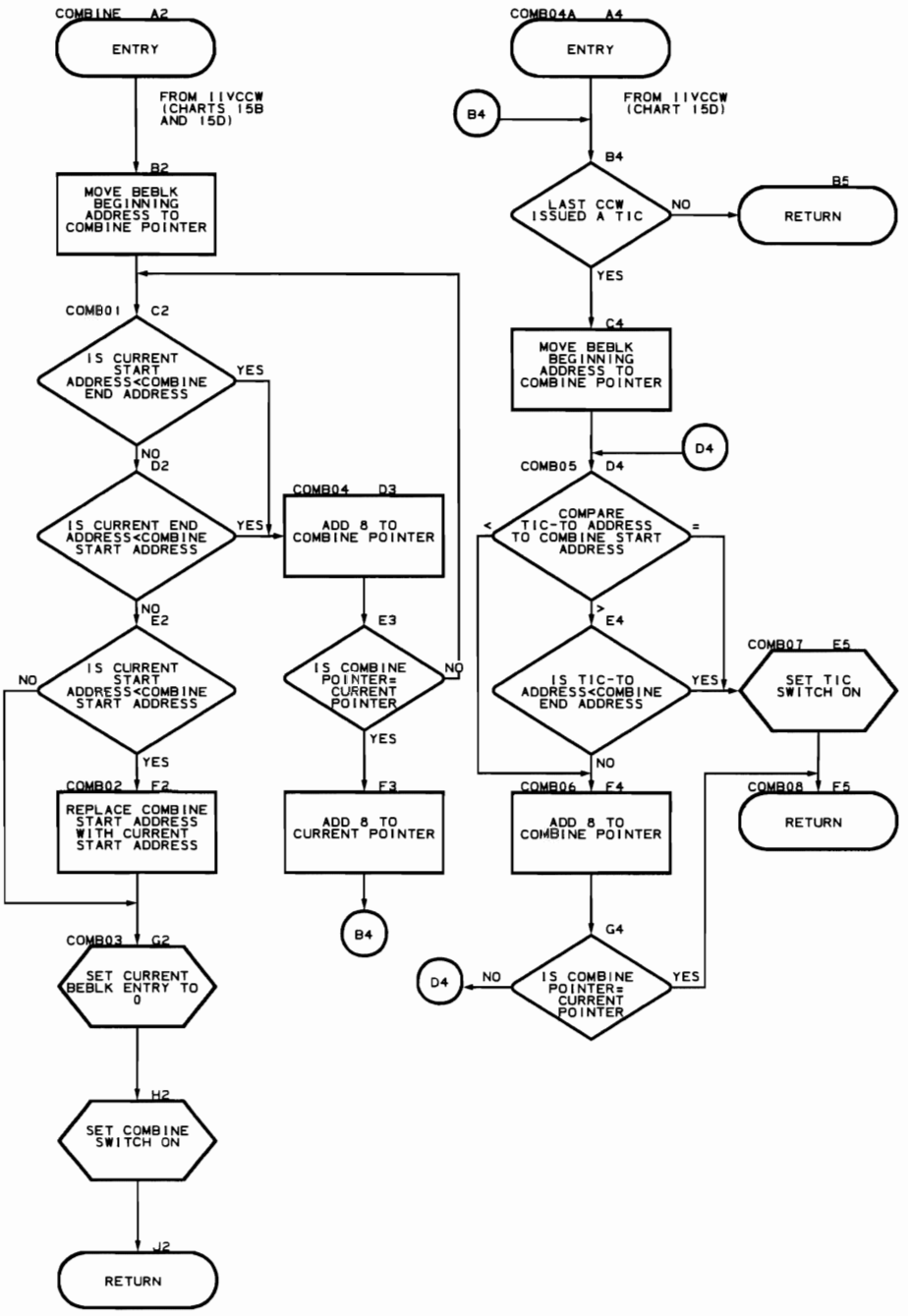


Flowchart 15E. Adjust CCW Data Address Routine (IIVCCW Part 5 of 6)



SYMBOL TABLE
LIST - ADJUST CCW DATA ADDRESS LIST

Flowchart 15F. Combine and COMBO4A Subroutines (IIVCCW Part 6 of 6)



CCW Adjustment Routine (Flowcharts 16A-16B)

Module name: IIVADJ

Entry point name: IIVADJ

Major functions: Adjusts CCW data addresses

Entered by:

- IIVCCW
- IIVRCW

Modules called: None

Exits to: Caller

OS macros issued: None

Input:

- Adjustement factor
- Pointer to a CCW string
- Pointer to the local execution list
- Local limit address

Output: Adjusted CCW string

Return codes: None

Tables/work areas:

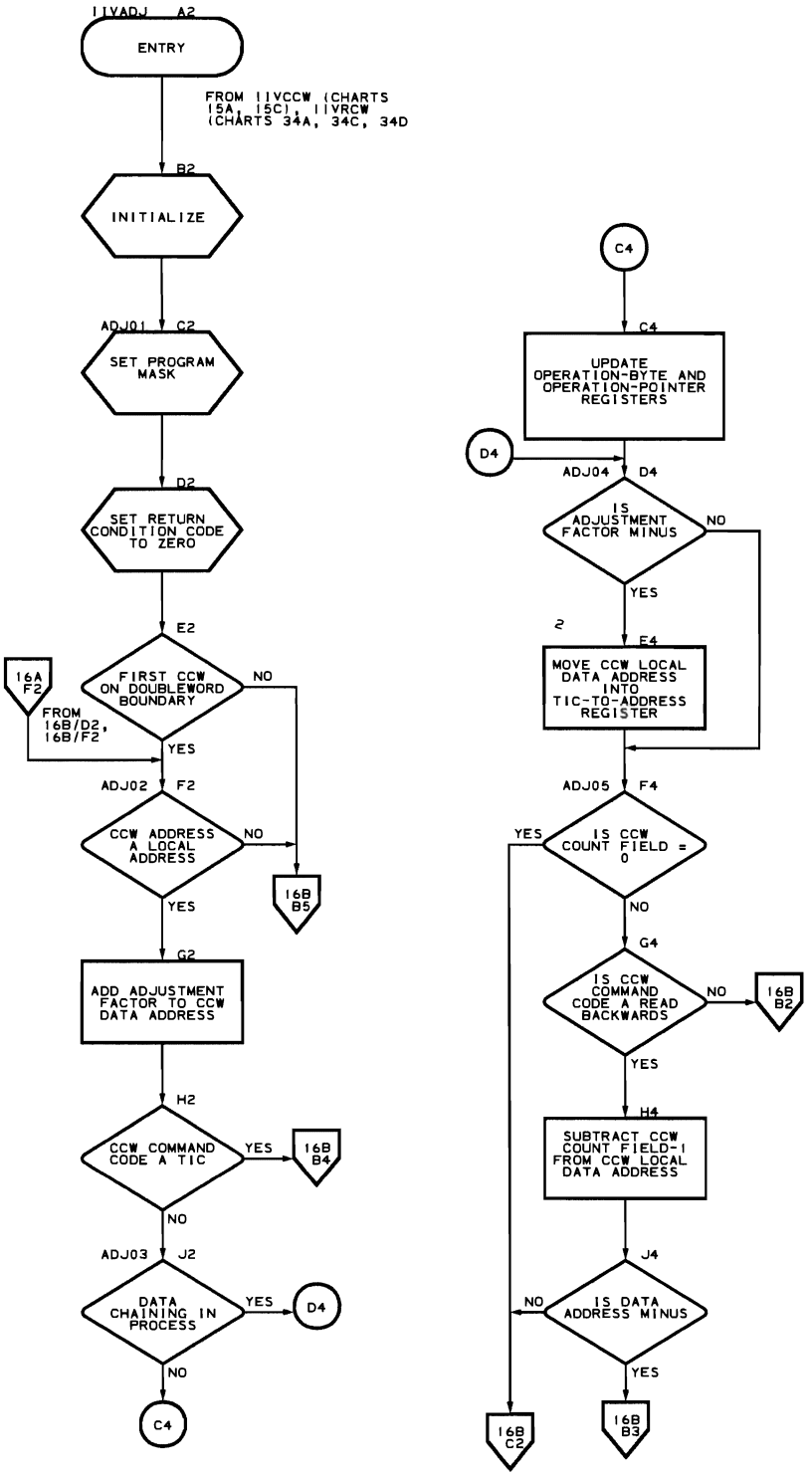
- Adjust CCW data address list (located in IIVCON)
- ADJSAVE register save area

Errors detected:

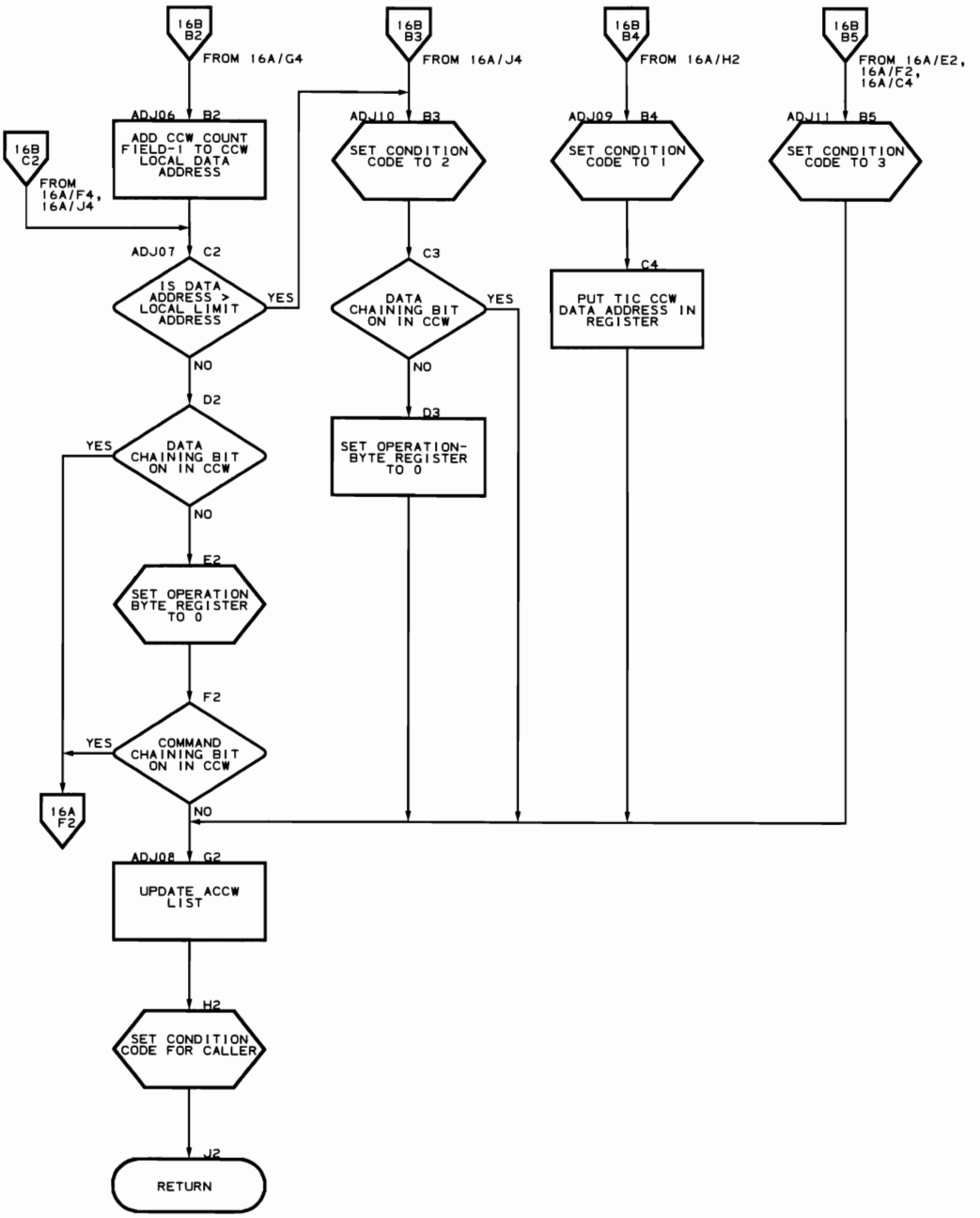
- Adjusted addresses not within DOS storage area
- CCW not on doubleword boundary
- Invalid CCW address

Messages requested: None

Flowchart 16A. CCW Adjustment Routine (IIVADJ Part 1 of 2)



Flowchart 16B. CCW Adjustment Routine (IIVADJ Part 2 of 2)



Supervisor Call Routine (Flowchart 17A)

Module name: IIVSVC

Entry point name: IIVSVC

Major functions: Directs DOS SVCs to proper Emulator modules

Entered from:

- DOS when DOS issues a supervisor call
- IIVSCI

Modules called:

- IIVADD
- IIVPUB
- IIVGR2

Exits to:

- IIVRTE (normal exit)
- IIVABN (error exit)

OS macros issued: None

Input: DOS registers

Output: Manipulates PSWs for supervisor calls

Return codes: X'04' to IIVABN for termination

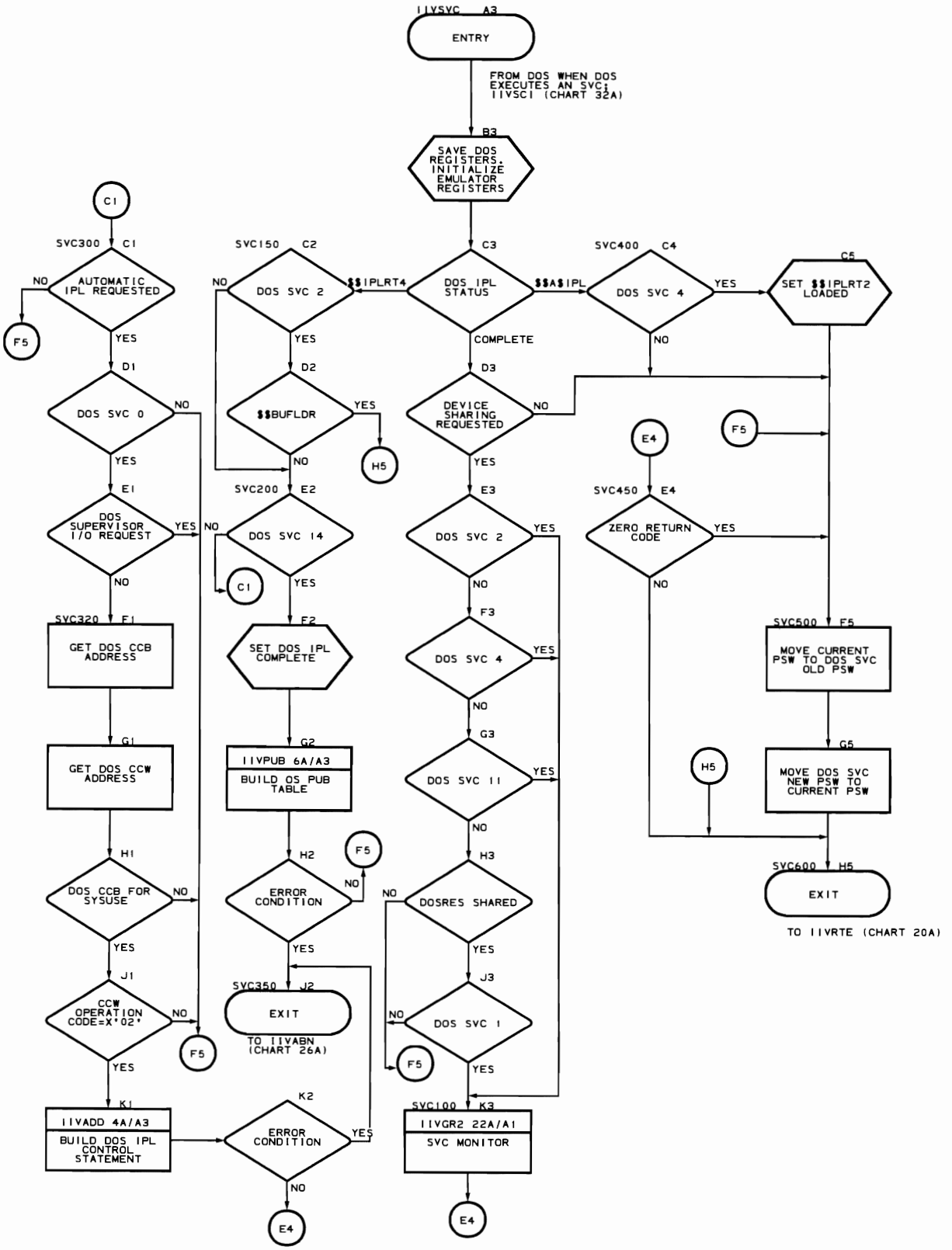
Tables/work areas:

- EMUCONS
- DOS storage
- DOS CCB
- DOS COMREG
- Local execution list

Errors detected: None

Messages requested: None

Flowchart 17A. Supervisor Call Routine (IIVSVC)



Start I/O, End-of-Extent, Channel End, and Abnormal End Appendage (Flowchart 18A-18B)

Module name: IGG019SA

Entry point name:

- IGG019SA
- SIORT000
- AERT000
- CERT000

Major functions:

- Maintains (modifies/restores) DEB information so that DOS I/O requests may be done by OS IOCS
- Uses the DOS mode set command for all tape I/O
- Uses the DOS file mask for all I/O on dedicated DASD volumes
- Inhibits automatic cylinder switching for all DASD I/O
- Makes the user label track associated with a shared volume data set accessible to the DOS OPEN/EOF routines

Entered by: OS input/output supervisor routines

Modules called: None

Exits to: Caller

OS Macros issued: None

Input:

- Pointer to DEB
- Pointer to DCB
- Pointer to UCB
- Pointer to IOB

Output: Modified DEB extent and/or device modifier fields as required to support DOS I/O requests

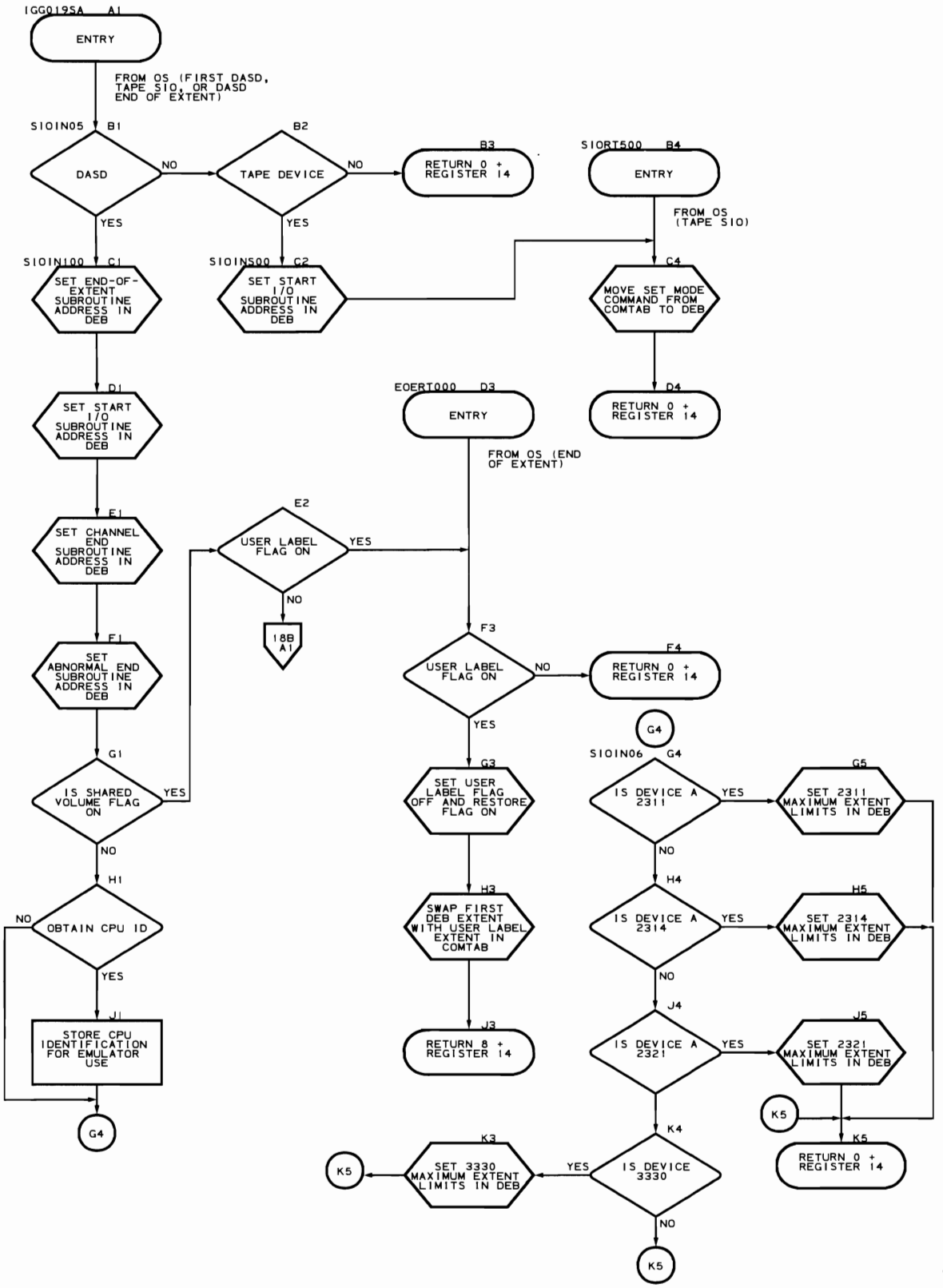
Return codes: None

Tables/work areas: COMTAB, DEB

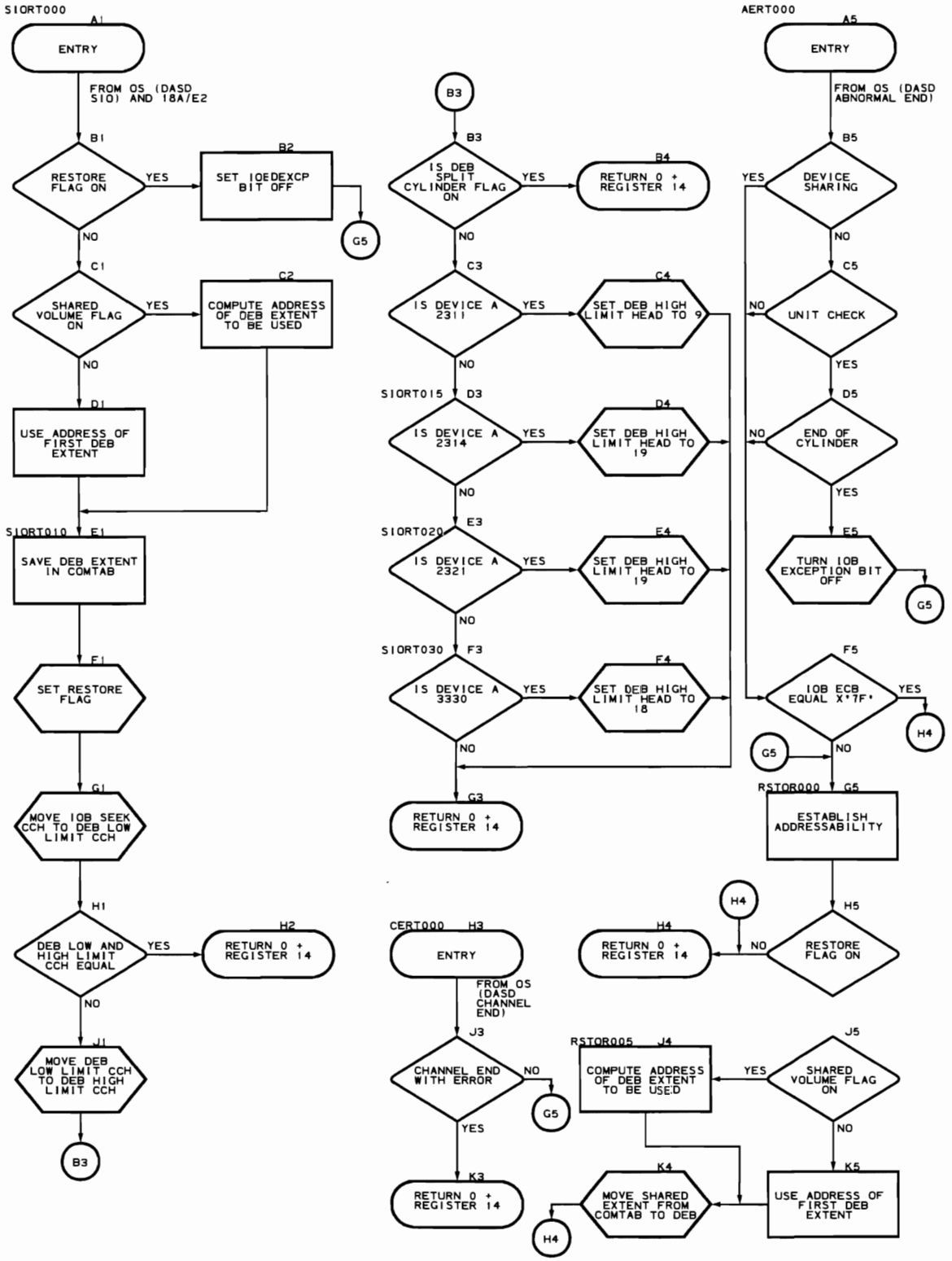
Errors detected: None

Messages requested: None

Flowchart 18A. Start I/O, End-of-Extent, Channel End, and Abnormal End Appendage (IGG019SA Part 1 of 2)



Flowchart 18B. Start I/O, End-of-Extent, Channel End, and Abnormal End Appendage (IGG019SA Part 2 of 2)



Abnormal End/Channel End Appendage (Flowchart 19A)

Module name: IGG019S1

Entry point name: IGG019S1

Major functions:

- Flags device 'not ready' condition after rewind-unload has been issued
- Prevents OS error recovery procedures from being loaded for incorrect length conditions on tape and unit record devices
- Prevents OS error recovery procedures from being entered for BTAM 'intervention required' condition
- Prevents OS error recovery procedures from being entered for unit record 'unit exception' conditions

Entered from: OS input/output supervisor interruption handler

Modules called: None

Exits to: Caller

OS macros issued: None

Input:

- Pointer to DCB
- Pointer to DEB
- Pointer to UCB
- Pointer to IOB

Output:

- IOB exception condition bit
- ECB completion code in IOB
- UCB interception bit
- UCB 'not ready' bit
- COMTAB entry 'not ready' bit

Return codes: None

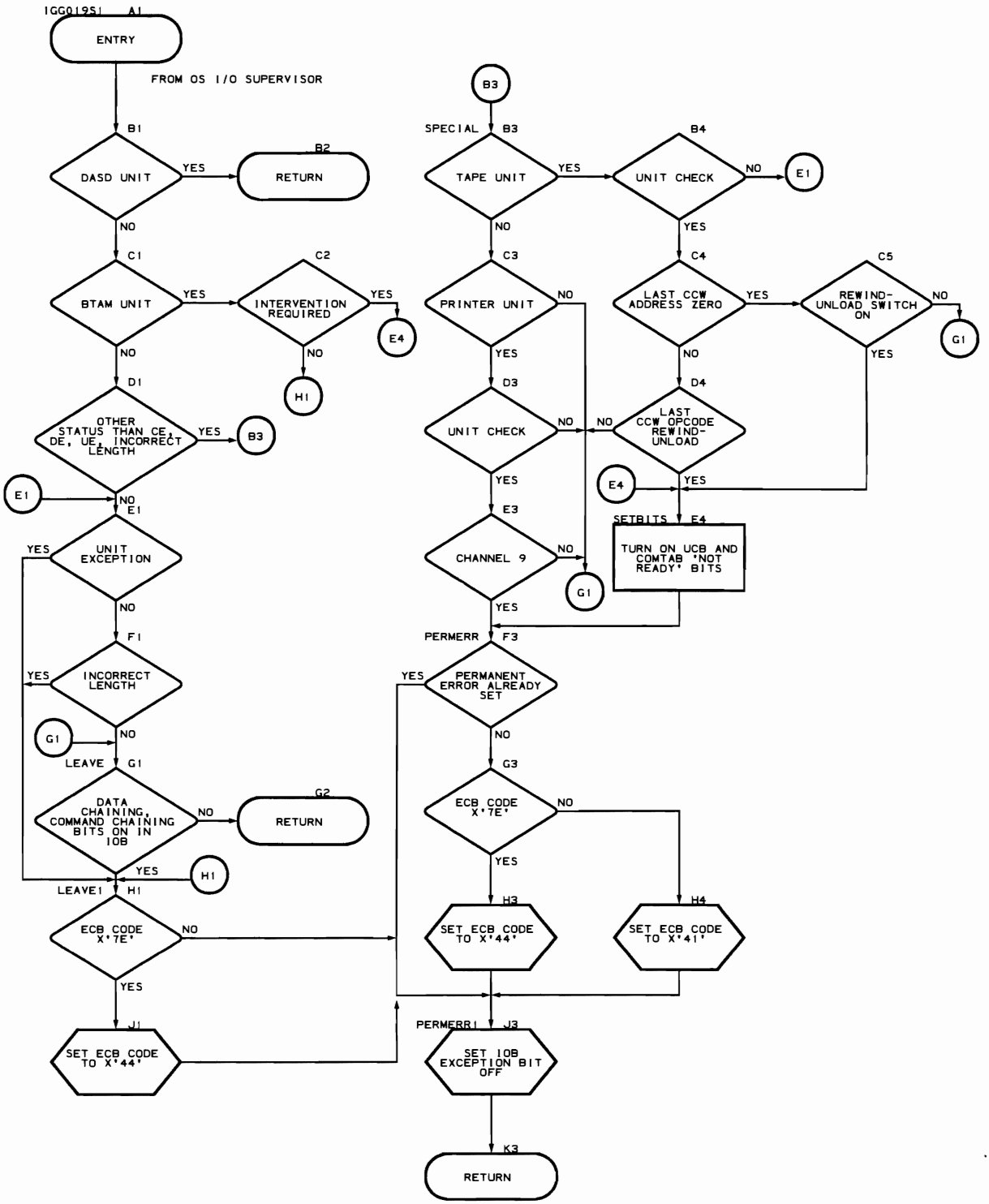
Tables/work areas:

- IOB
- UCB
- COMTAB

Errors detected: None

Messages requested: None

Flowchart 19A. Abnormal End/Channel End Appendage (IGG019S1)



Asynchronous Interrupt Exit Routine (Flowcharts 20A-20G)

Module name: IIVRTE

Entry point names:

- IIVRTE
- IIVRTER2
- IIVRTETM
- IIVRTEOJ
- IIVRTEST
- IIVRTESR
- IIVRTESY

Major functions:

- First routine to gain control following asynchronous interruptions in local execution mode
- Serves as central routine for all asynchronous interruption handlers
- Serves as central return point for all Emulator routines
- Determines if any interruptions are pending for DOS and gives control to the proper routine to process them
- Checks for end-of-job conditions
- Issues the STIMER macro, decrements the DOS timer, and simulates timer interruptions
- Issues the DOS compatibility instruction (see execute local instruction in Appendix) to return to adjusted DOS storage area
- Issues WAIT macro when DOS-requested input/output operations are not completed
- Issues STAE macro if a //SYSABEND DD statement is not present, schedules STAE retry routine, reissues any STAE retry routine, reissues any canceled WTORs

Entered from:

- OS supervisor, OS abend processing routines
- IIVINT
- IIVPCE
- IIVSTG
- IIVLOG
- IIVIN2
- IIVSVC
- IIVRAS
- IIVACI

Modules called:

- IIVPRP
- IIVMSG
- IIVCHK

Exits to:

- Caller
- DOS
- IIVABN
- IIVLOG
- IIVCCW
- IIVOPN
- IIVRCW

OS macros issued:

- POST
- STAE
- WAIT
- STIMER

Input: Except for postasynchronous interruption entries, pointer to IIVCON

Output:

- Post ECB list entries for IIVPCE
- Updated DOS timer value
- Local execution list with contents required by OS

Return codes: 16 = DOS entered a hard wait

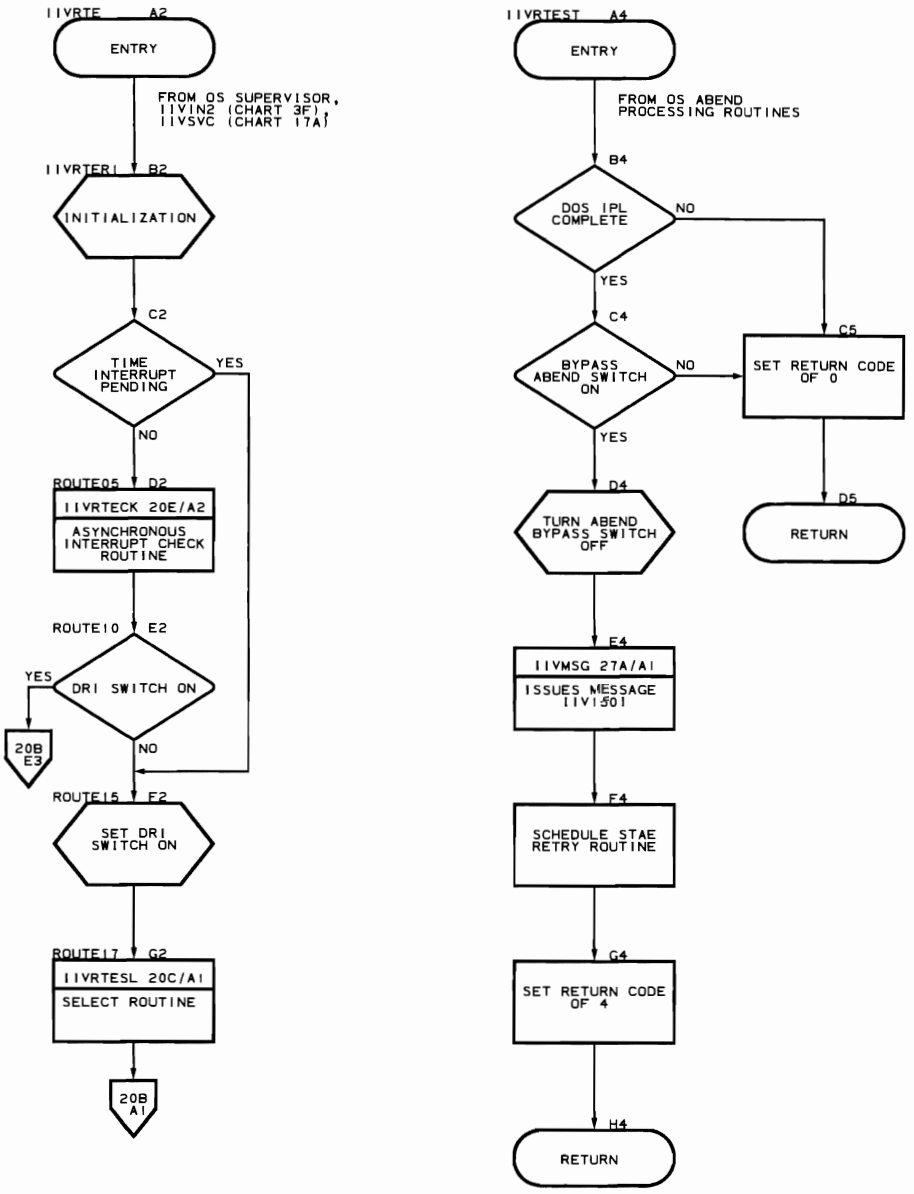
Tables/work areas:

- Local execution list (located in IIVCON)
- COMTAB
- ECBLIST (dynamic storage) - ECB list to scan
- PENDSN (IIVCON) - interruption pending switch
- BREGSAVE (IIVCON) - register save area
- Post ECB List - index and condition codes of ECBs found posted
- BASEREGS (IIVCON) - Emulator base registers
- DOSCORE - access DOS timer

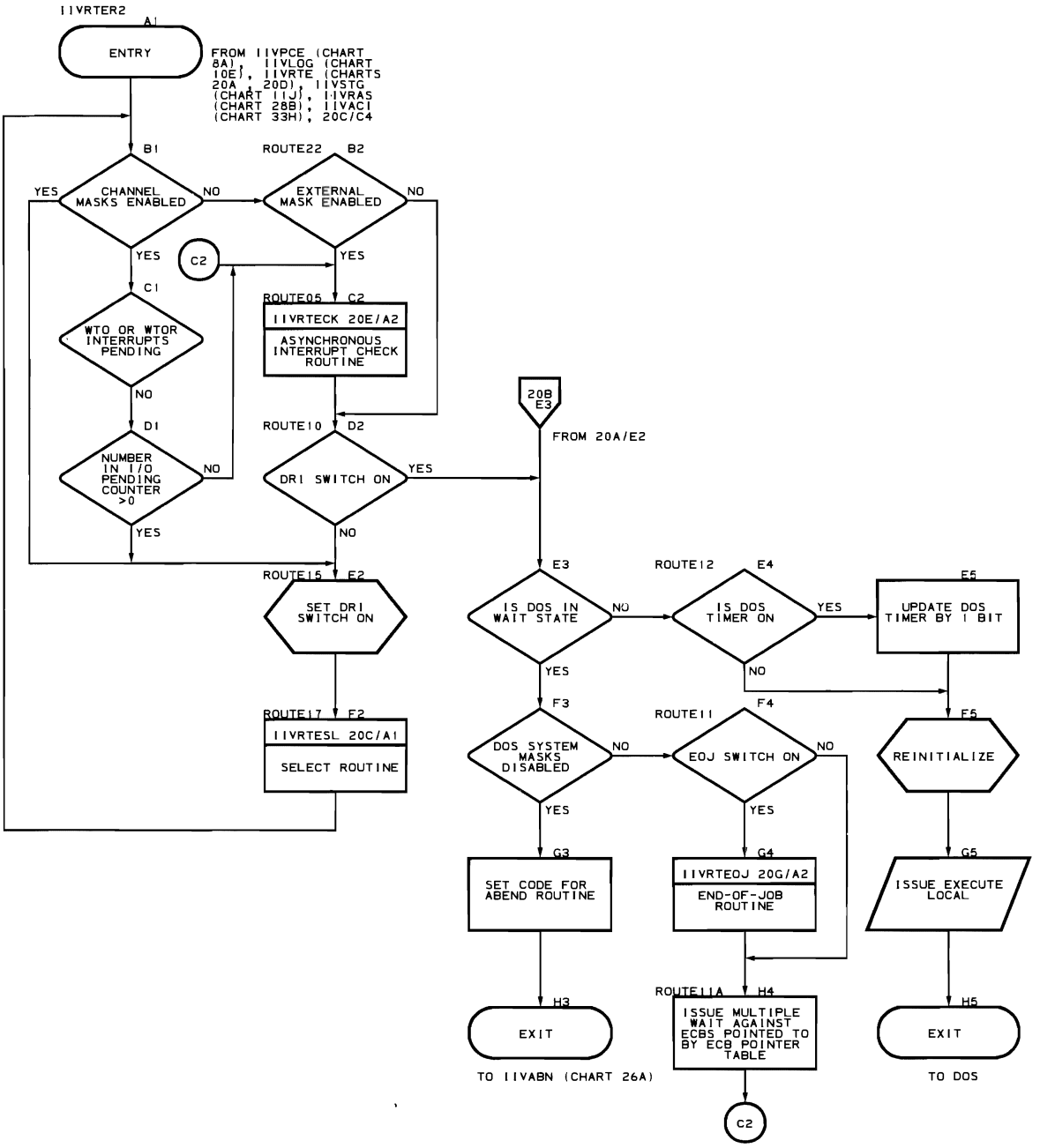
Errors detected: Detects ABEND condition signified by DOS hard wait PSW

Messages requested: IIV150I

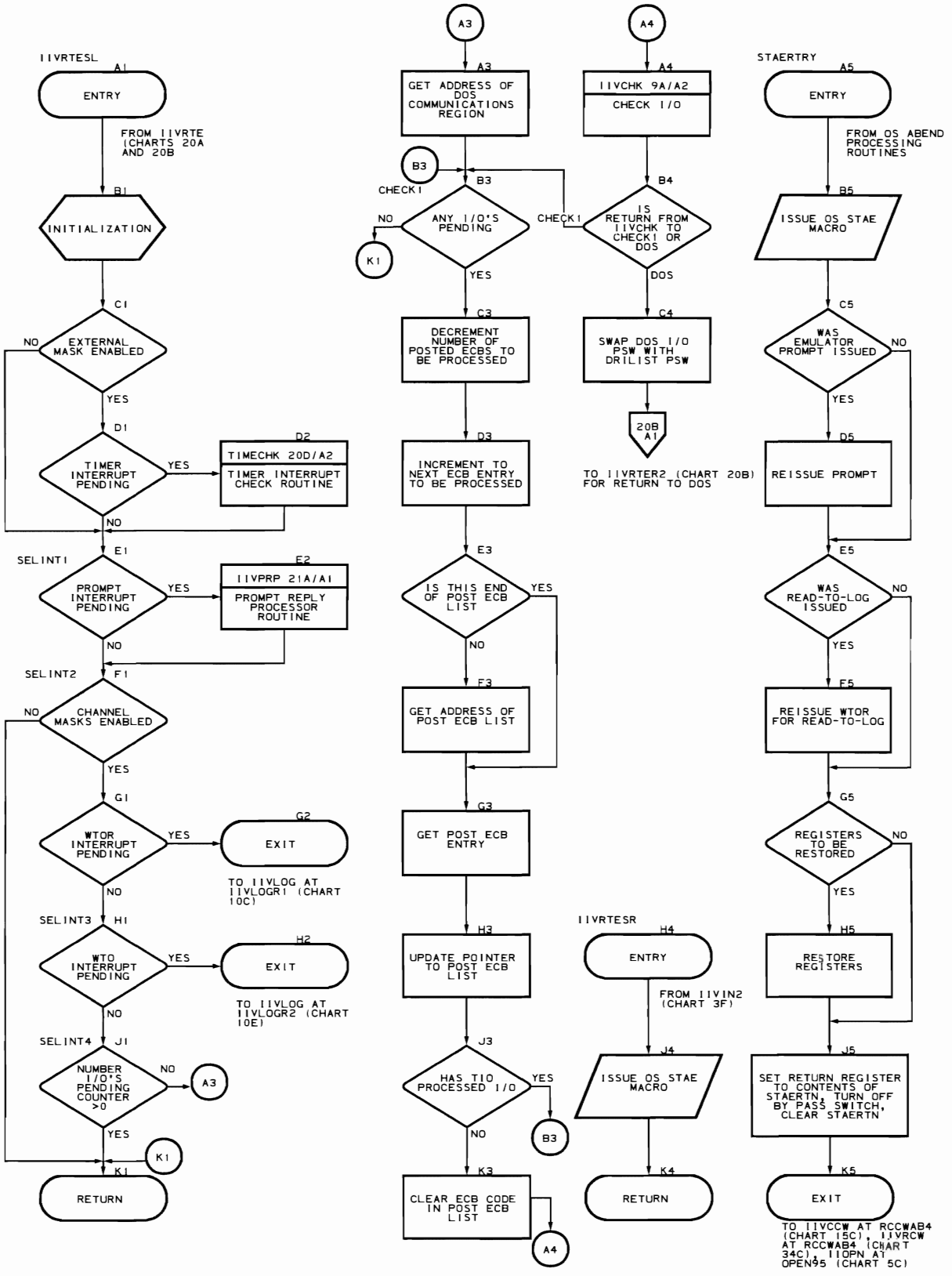
Flowchart 20A. Asynchronous Interrupt and STAE Exit Routines (IIVRTE Part 1 of 7)



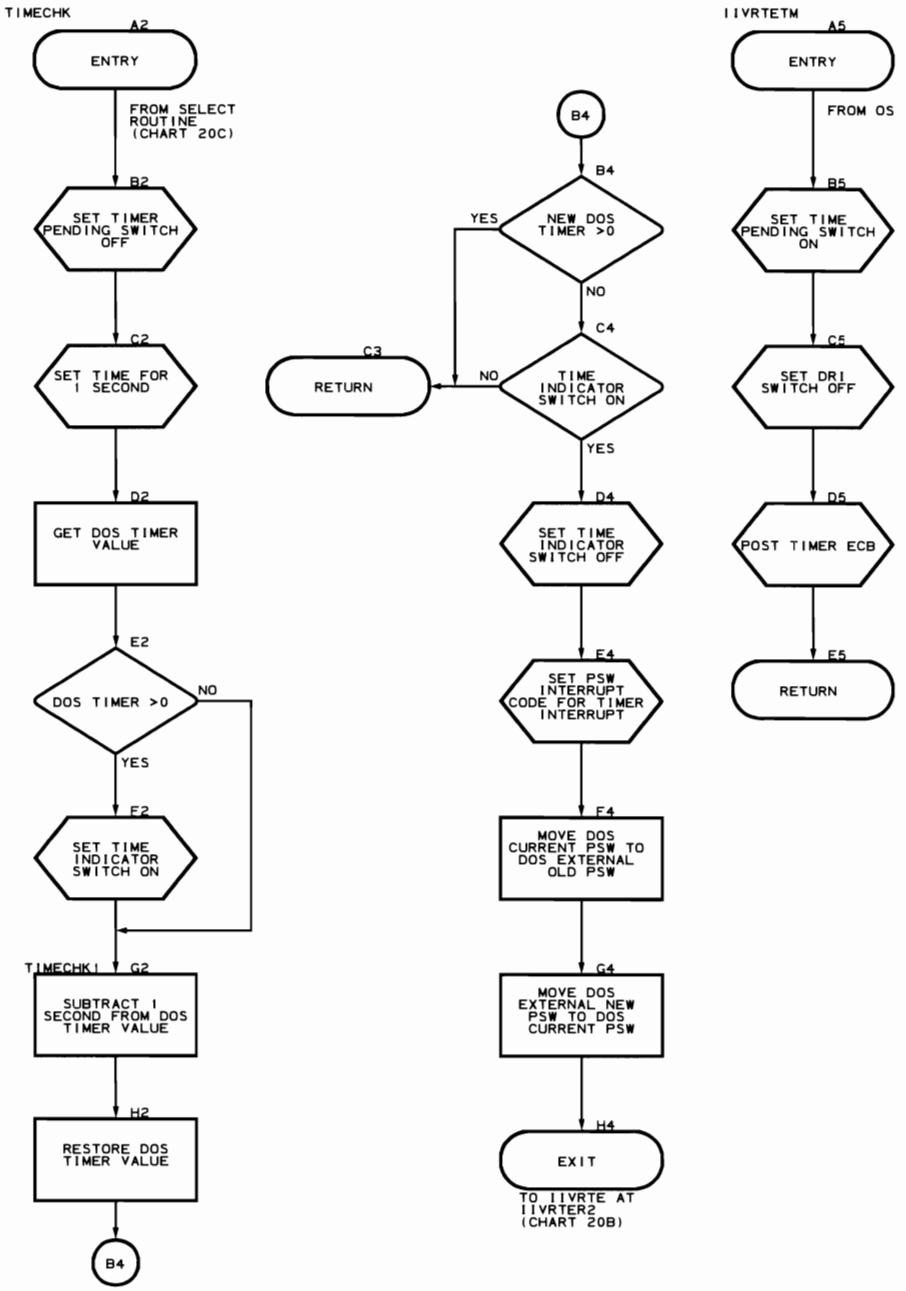
Flowchart 20B. Route Routine (IIVRTE Part 2 of 7)



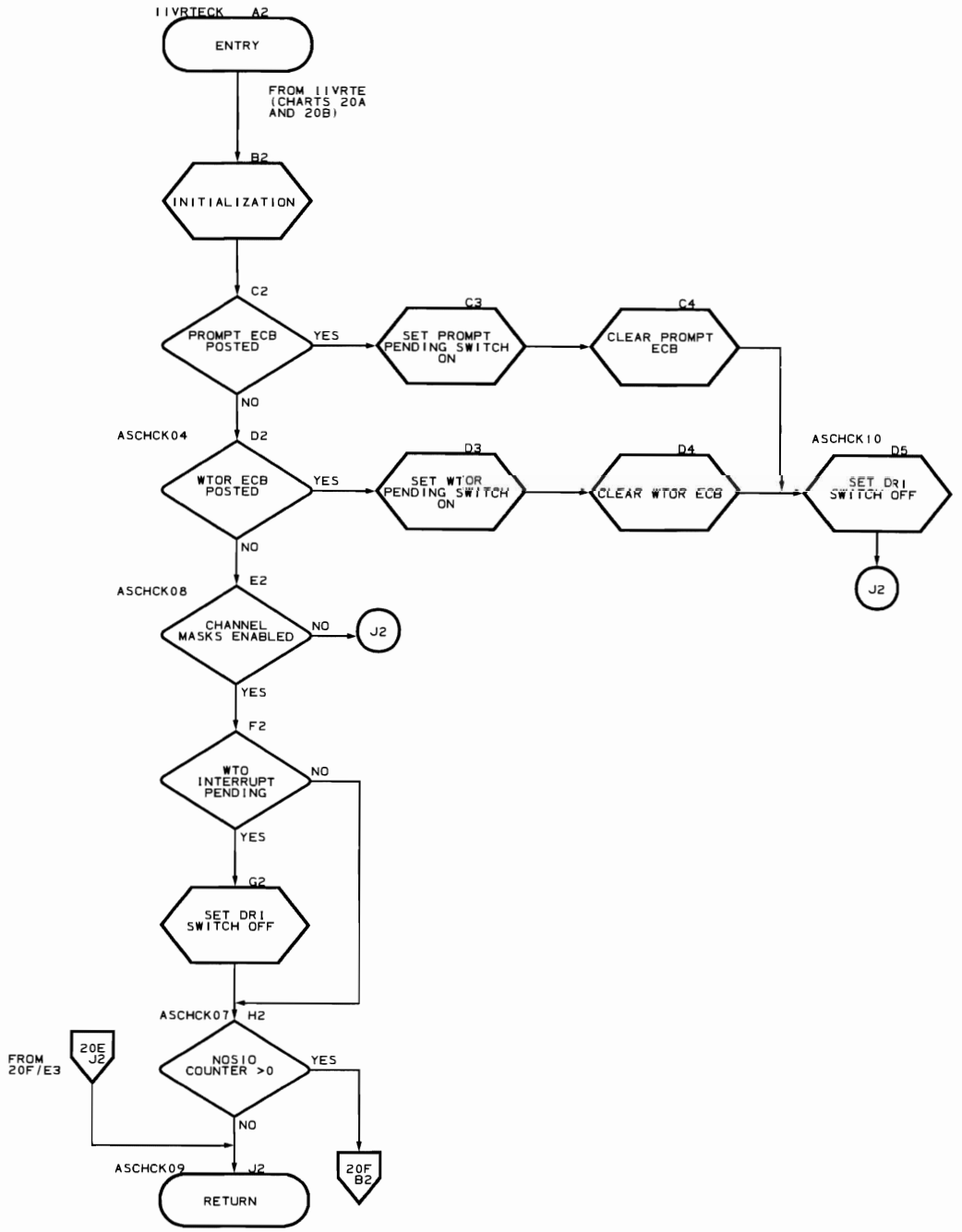
Flowchart 20C. Select and STAE Retry Routines (IIVRTE Part 3 of 7)



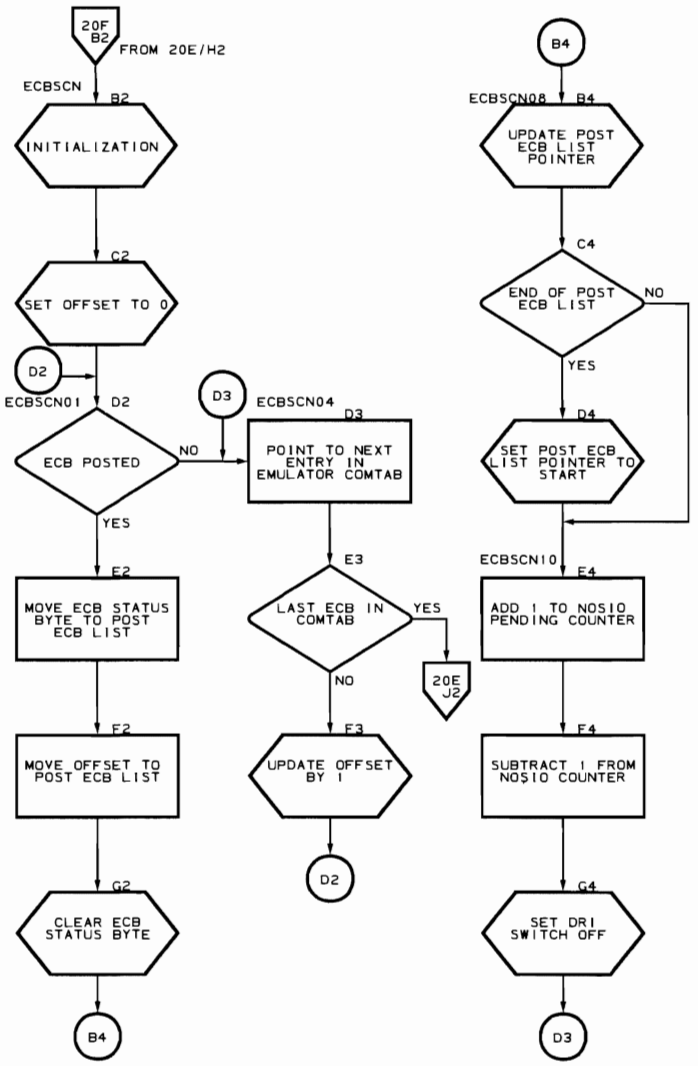
Flowchart 20D. Timer Interrupt Check and Timer Interrupt Subroutines (IIVRTE Part 4 of 7)



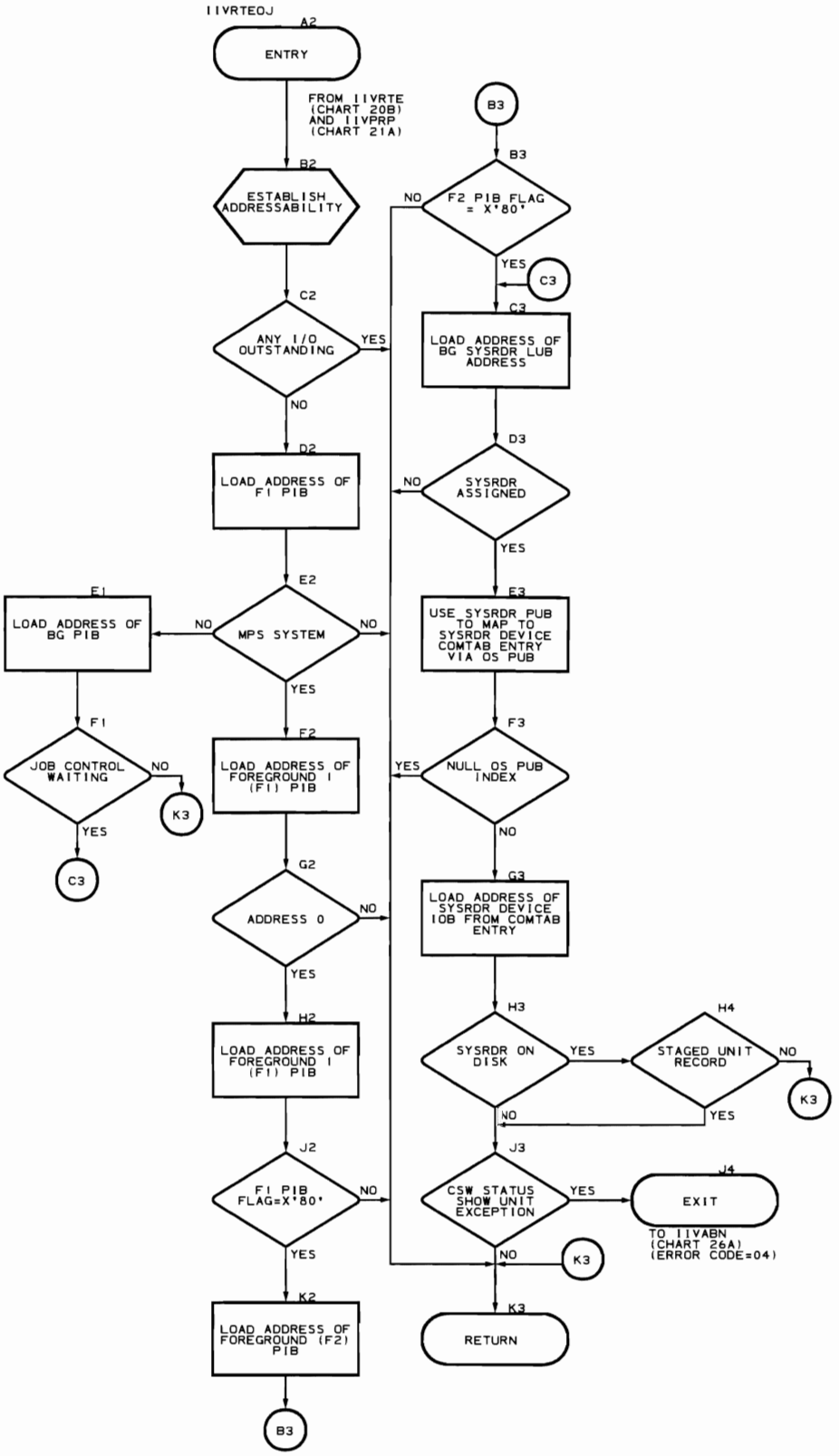
Flowchart 20E. Asynchronous Interrupt Check Subroutine (IIVRTE Part 5 of 7)



Flowchart 20F. Asynchronous Interrupt Check Subroutine (IIVR™E Part 6 of 7)



Flowchart 20G. End-of-Job Routine (IIVRTE Part 7 of 7)



Prompt Reply Processor Routine (Flowcharts 21A-21E)

Module name: IIVPRP

Entry point name: IIVPRP

Major functions: Gets control when console operator replies to an operator prompt message from the Emulator

Entered by: IIVRTE

Modules called:

- IIVOPN
- IIVMSG
- IIVRAS
- IIVRTEOJ

Exits to: Caller

OS macros issued:

- SAVE
- CLOSE
- LOAD
- RETURN

Input:

- Pointer to DOS storage area
- Pointer to IIVCON
- Pointer to local execution list
- Reply to previous prompt WTOR

Output:

- Prompt for further operator communication
- Messages to operator dependent on prompt reply being processed
- Updated DOS CSW
- Updated DOS PSW in local execution list
- Updated external and I/O PSWs in DOSCORE

Return codes: None

Tables/work areas:

- IIVCON (constants and addresses)
- Local execution list (DOS current PSW)
- DOSCORE:
  - External old PSW
  - External new PSW
  - I/O old PSW
  - I/O new PSW
  - PUB
  - CSW
  - COMTAB

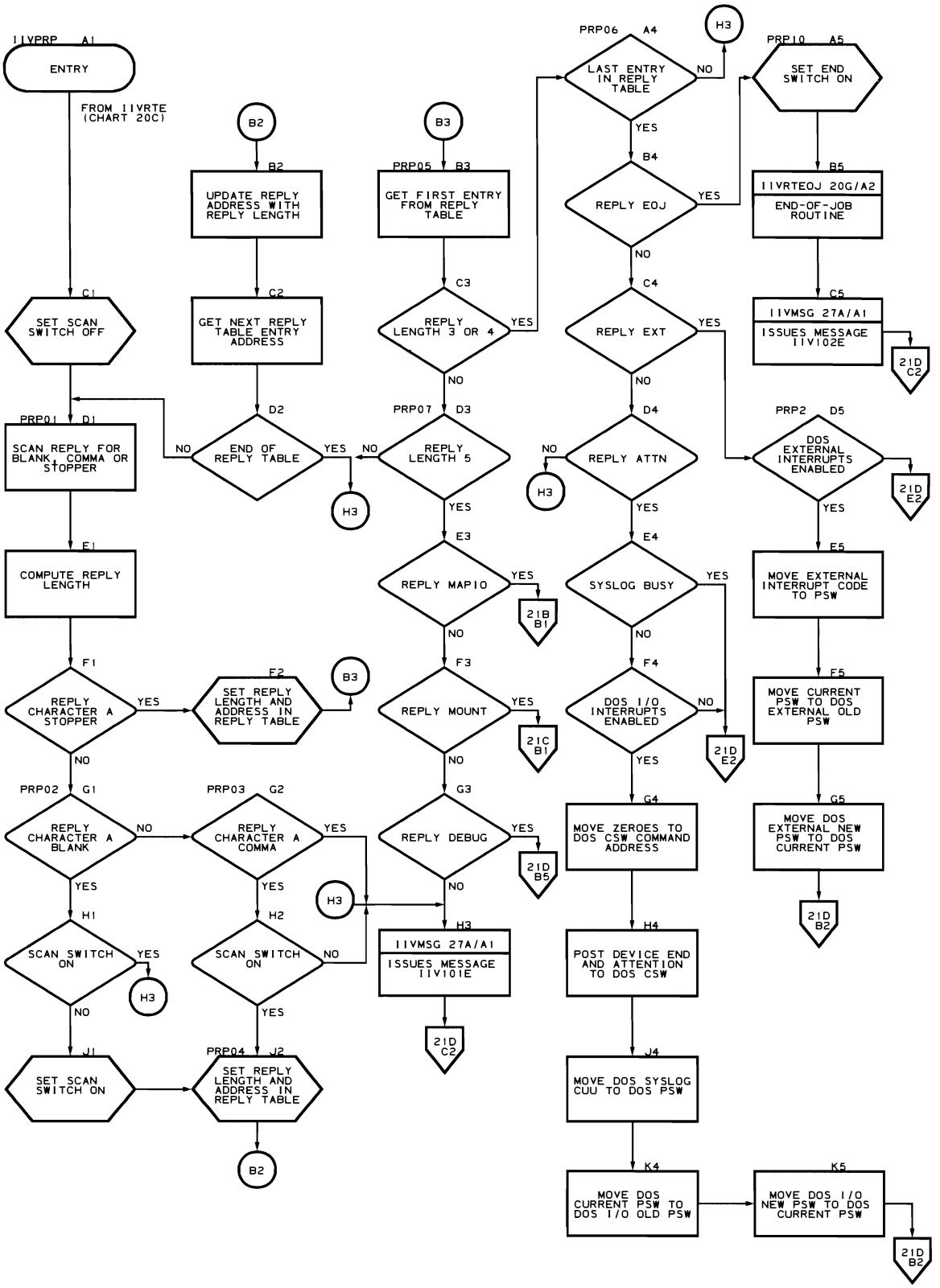
Errors detected:

- Replies are checked for a valid Emulator command (EOJ,EXT,ATTN, MAPIO,MOUNT,DEBUG)
- Invalid operands for MOUNT and MAPIO

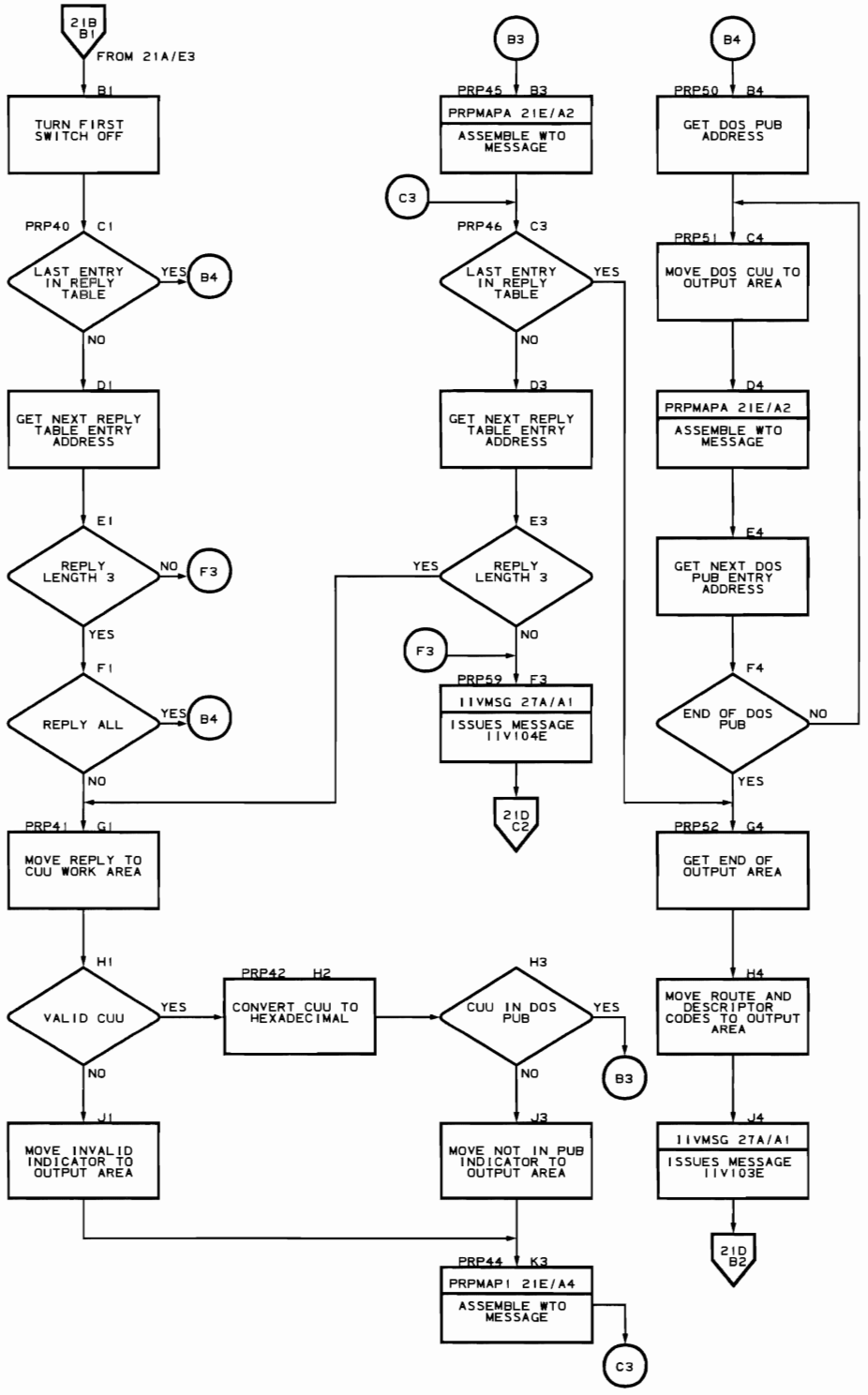
Messages requested:

- IIV101E
- IIV102E
- IIV103E
- IIV104E
- IIV105E
- IIV106E
- IIV107E
- IIV108E
- IIV109E

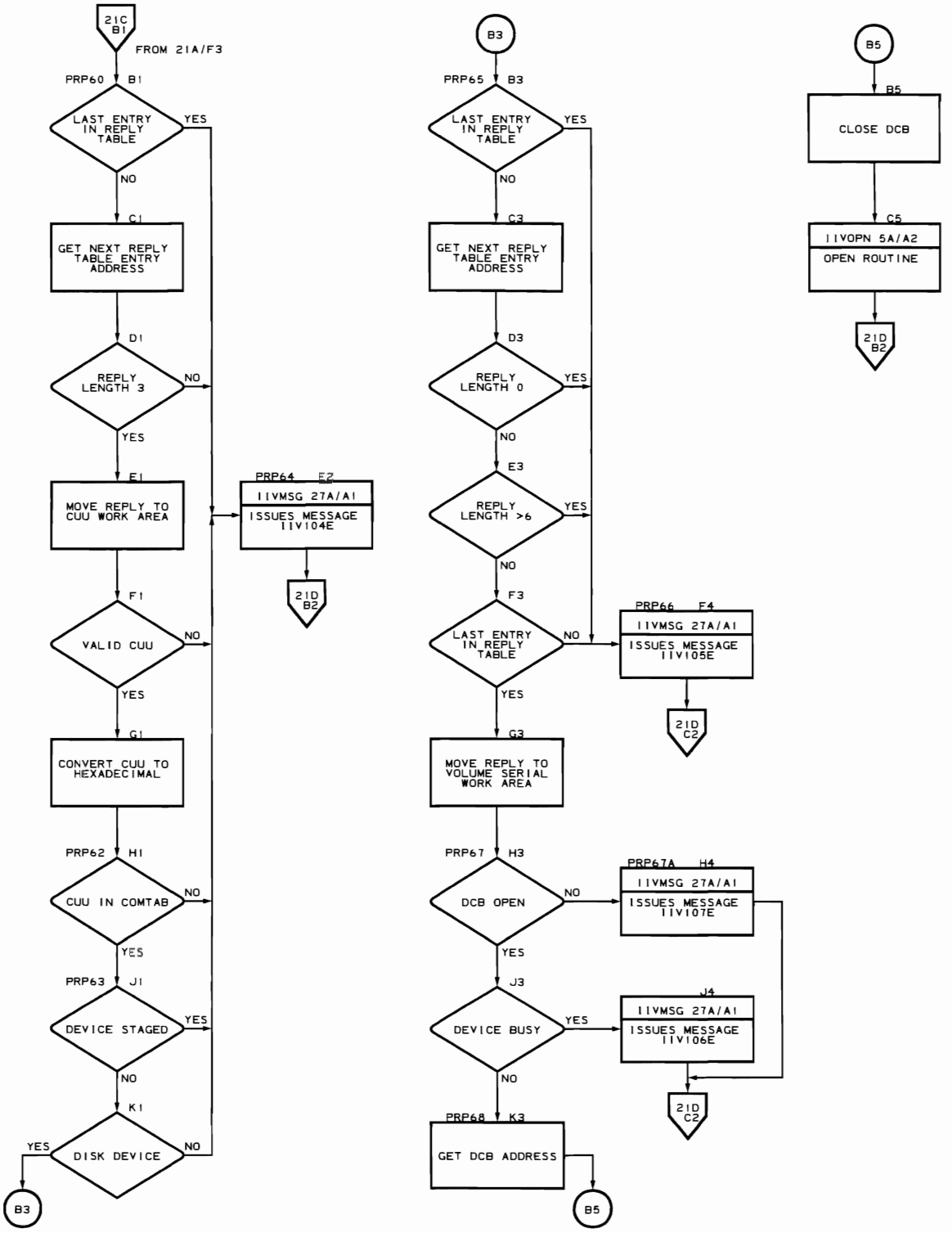
Flowchart 21A. Prompt Reply Processor Routine (IIVPRP Part 1 of 5)



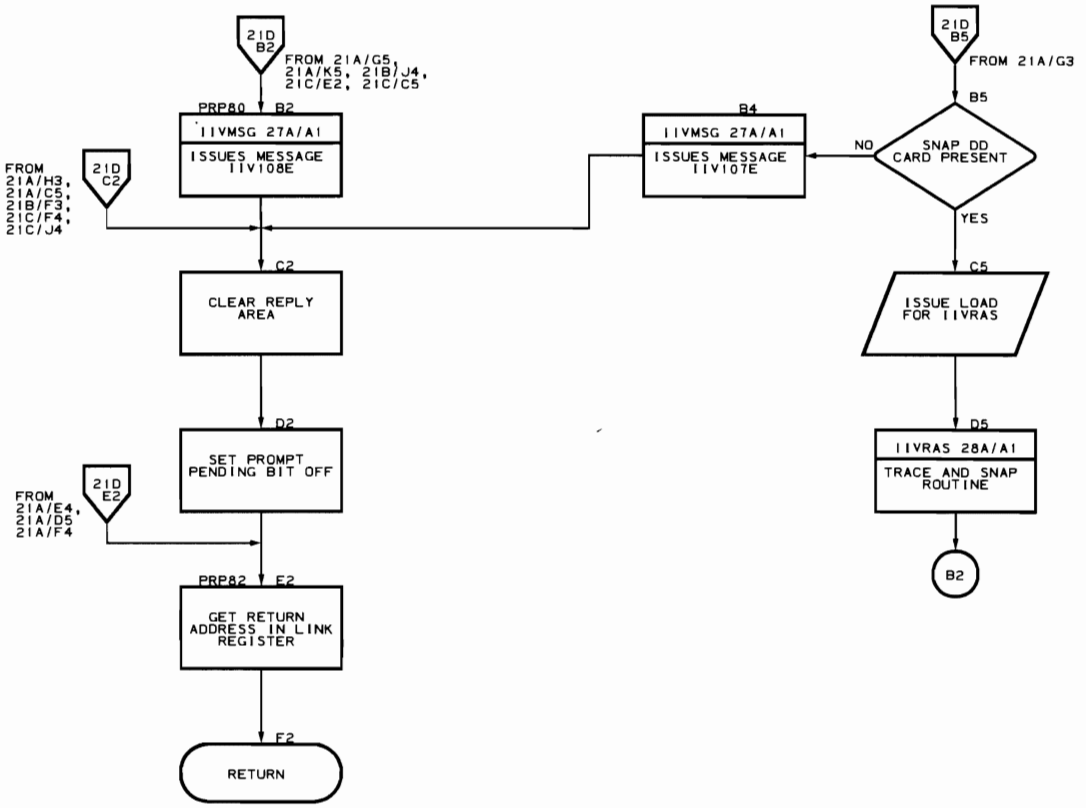
Flowchart 21B. Prompt Reply Processor Routine (IIVPRP Part 2 of 5)



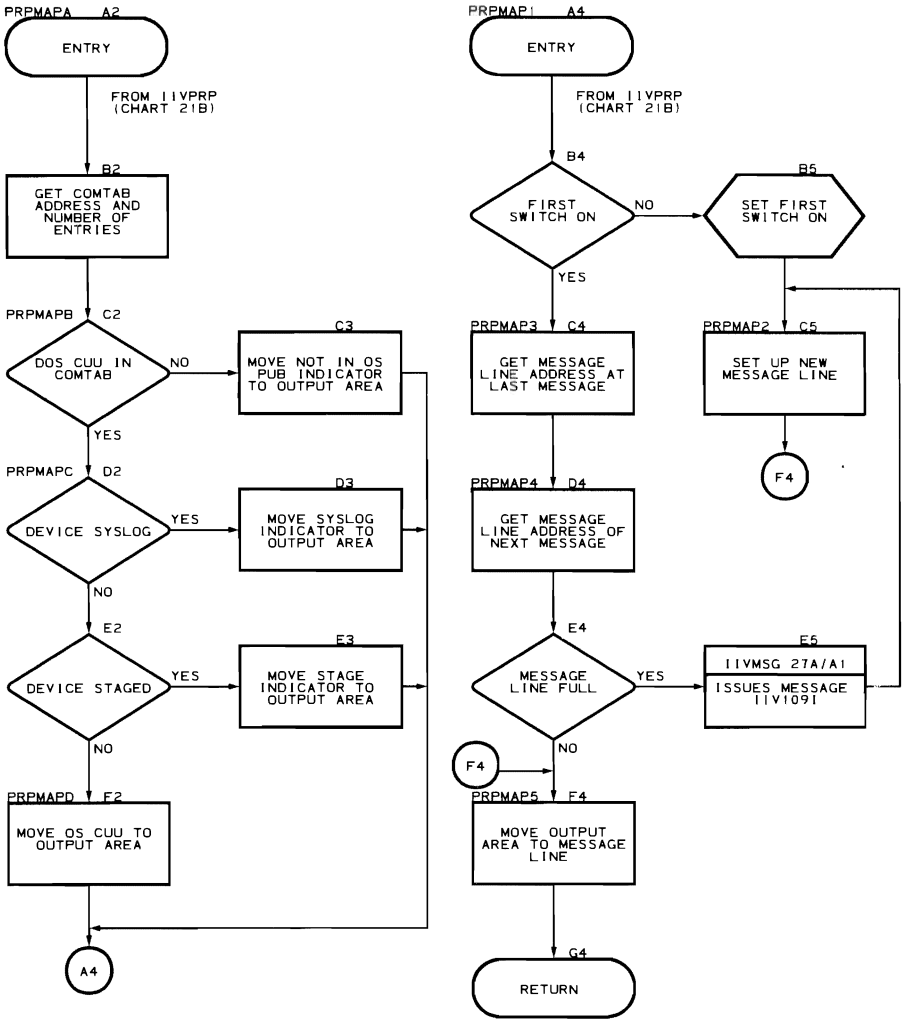
Flowchart 21C. Prompt Reply Processor Routine (IIVPRP Part 3 of 5)



Flowchart 21D. Prompt Reply Processor Routine (IIVPRP Part 4 of 5)



Flowchart 21E. PRPMAPA and PRPMAP1 Subroutines (IIVPRP Part 5 of 5)



SVC Monitor Routine (Flowcharts 22A-22D)

Module name: IIVGR2

Entry point name: IIVGR2

Major functions:

- Monitors SVC 1, 2, 4, and 11 calls from DOS
- Inhibits DOS execution of library maintenance programs when DOS system residence volume is a shared volume
- Recognizes volume-shared files or OS indexed sequential shared data sets
- Calls IIVDVS or IIVIS to perform the needed functions

Entered by: IIVSVC

Modules called:

- IIVIS
- IIVMSG
- IIVDVS

Exits to: Caller

OS macros issued: RETURN

Input:

- Register 9 must contain a pointer to the local execution list.
- Register 10 must contain a pointer to DOS storage.
- Register 11 must contain a pointer to IIVCON.

Output: None

Return codes: Contained in register 15

Tables/work areas:

- Local execution list
- IIVCON
- DLBL
- DTF
- FID
- COMTAB
- COMTAB extension

Errors detected:

- Invalid logical unit
- Unit unassigned or assigned in error
- Invalid physical unit
- No DD card or invalid DLBL

Messages requested:

- IIV208I
- IIV250I
- IIV251I
- IIV252I
- IIV253I
- IIV254I
- IIV255I

DOS SVC Tables. Figure 25 shows the DOS SVC tables. Whenever DOS issues an SVC 2, the Emulator traps the SVC and determines which routine in IIVGR2 gains control from SVC2TAB or SVC2PP. The factors that affect which routine in IIVGR2 gains control are the caller that issues the SVC 2 and the B-transient phase called.

The SVC11TAB lists the B-transients that issue an SVC 11. The Emulator traps this SVC to determine which phase issued the SVC so control can be passed to routine GR2CLS in IIVGR2.

The SVC4TAB shows which B-transient phase issues an SVC 4 and the phase it calls. The Emulator traps this SVC 4 and gives control to GR2EOJ in IIVGR2.

SVC2TAB (when SVC 2 is issued)

CALLER	CALLED	ROUTINE IN IIVGR2	CHART
$$BOSD01	$$BOSDO1	TSTOPN	22B/B2
	$$BOSDO7	GR2EOX	22C/B1
	$$BOSDI1	TSTOPN	22B/B2
	$$BOSDW1	GR2OPN	22B/B1
	$$BOSDW3	GR2EOX	22C/B1
$$BOSDO4	$$BOSDO6	GR2EOX	22C/B1
$$BOSDO6	$$BCLOSE	GR2CLS	22C/B2
$$BOSD09	$$BOSD06	GR2EOX	22C/B1
$$BOSDI1	$$BOSDI3	GR2EOX	22C/B1
$$BOSDI3	$$BCLOSE	GR2CLS	22C/B2
$$BOSDI4	$$BOSDI3	GR2EOX	22C/B1
$$BOSDW2	$$BOSDW1	GR2EOX	22C/B1
$$BOPEN2	$$BODAIN	GR2OPN	22B/B1
	$$BOIS01	GR2OPN	22B/B1
	$$BORTV1	GR2OPN	22B/B1
$$BODAIN	$$BODAO1	GR2EOX	22C/B1
	$$BODAI1	GR2EOX	22C/B1
$$BODAI1	$$BODAU1	GR2EOX	22C/B1
$$BOFLPT	$$BOSDW1	GR2EOX	22C/B1
	$$BODAU1	GR2EOX	22C/B1
$$BOPEN	$$BOPEND	RMSRESET	22D/B4
$$BCLOSE	$$BODACL	GR2CLS	22C/B2
	$$BOSDC1	GR2CLS	22C/B2
	$$BCIS0A	GR2CLS	22C/B2
	$$BOPEND	GR2CLS	22C/B2
$$BOSDC1	$$BOSDO1	GR2EOX	22C/B1
	$$BOSDW3	GR2EOX	22C/B1
	$$BCLOSE	GR2CLS	22C/B2
	$$BOSDO6	GR2EOX	22C/B1
	$$BOSDI3	GR2EOX	22C/B1

Figure 25 (Part 1 of 2). DOS SVC Tables

SVC2TAB (when SVC 2 is issued) (continued)

CALLER	CALLED	ROUTINE IN IIVGR2	CHART
$$BOSDC2	$$BCLOSE	GR2CLS	22C/B2
$$BOMSG2	$$BODSPV	GR2DPV	22D/B5
	$$BOVDMP	GR2DPV	22D/B5
	$$BODSPV	GR2DPV	22D/B5
$$BOSDMW	$$BOVDMP	GR2DPV	22D/B5

SVC2PP (when SVC 2 issued for SETFL, ENDFL, or SETL)

CALLER	CALLED	ROUTINE IN IIVGR2	CHART
Problem program	$$BSETFL	ISSETFL	22D/B1
	$$BENDFL	ISENDFL	22D/B2
	$$BSETL	ISSETL	22D/B3

SVC11TAB (when SVC 11 is issued)

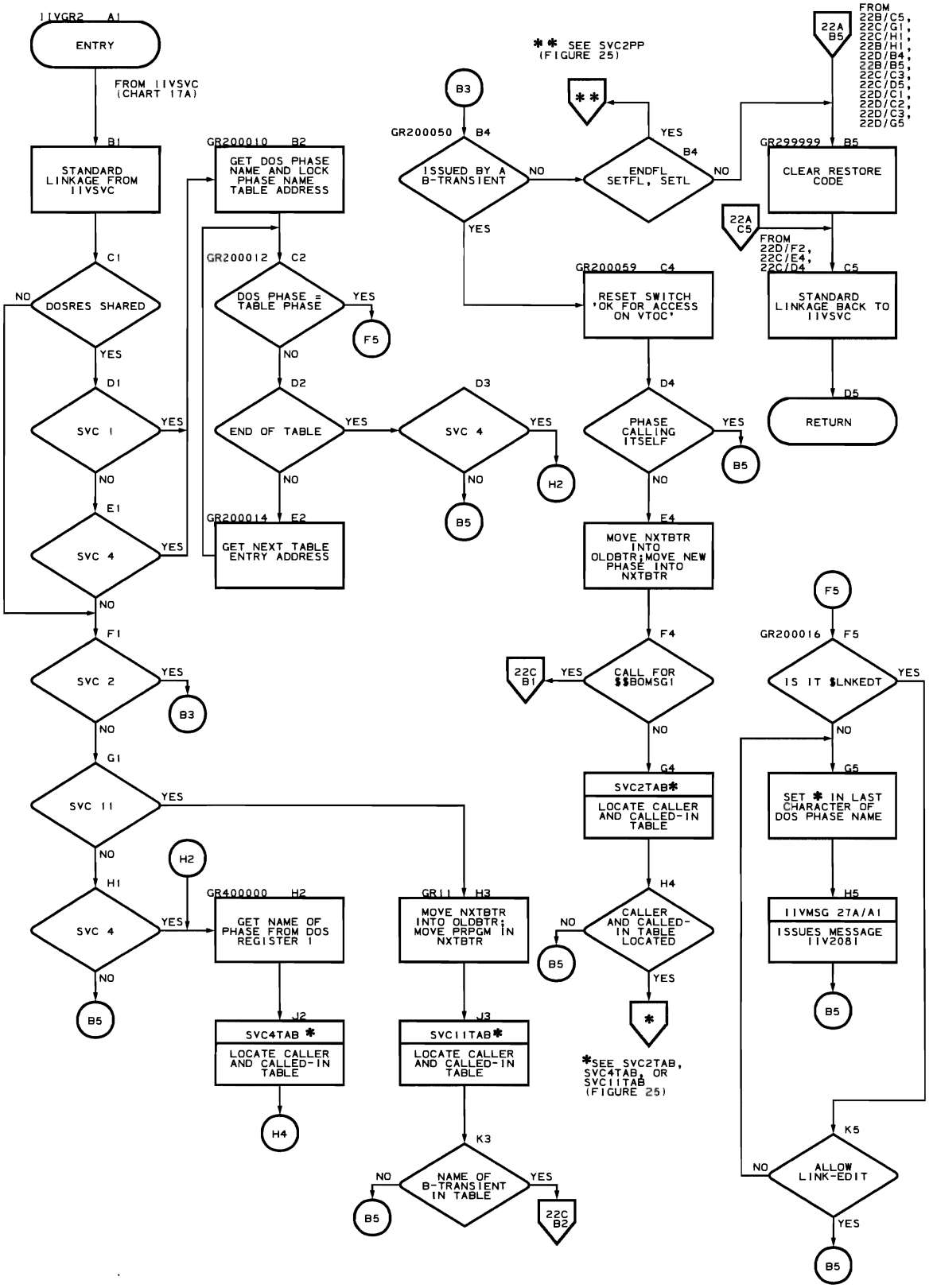
CALLER	ROUTINE IN IIVGR2	CHART
$$BOSDC1	GR2CLS	22C/B2
$$BOSDC2	GR2CLS	22C/B2
$$BOSDI3	GR2CLS	22C/B2
$$BOSD06	GR2CLS	22C/B2

SVC4TAB (When SVC 4 is issued)

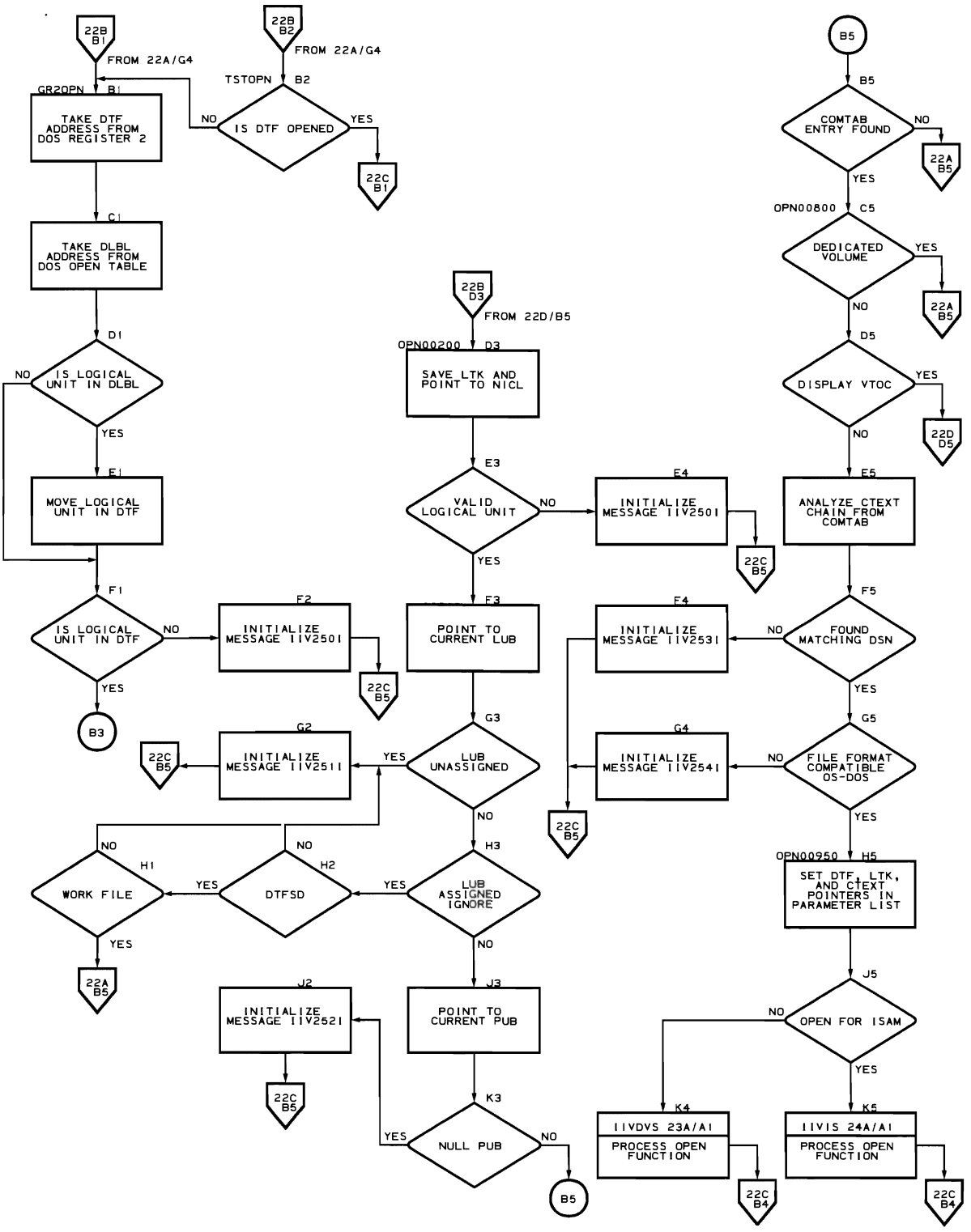
CALLER	CALLED	ROUTINE IN IIVGR2	CHART
$$BEOJ	$JOBCTLA	GR2EOJ	22C/B3

Figure 25 (Part 2 of 2). DOS SVC Tables

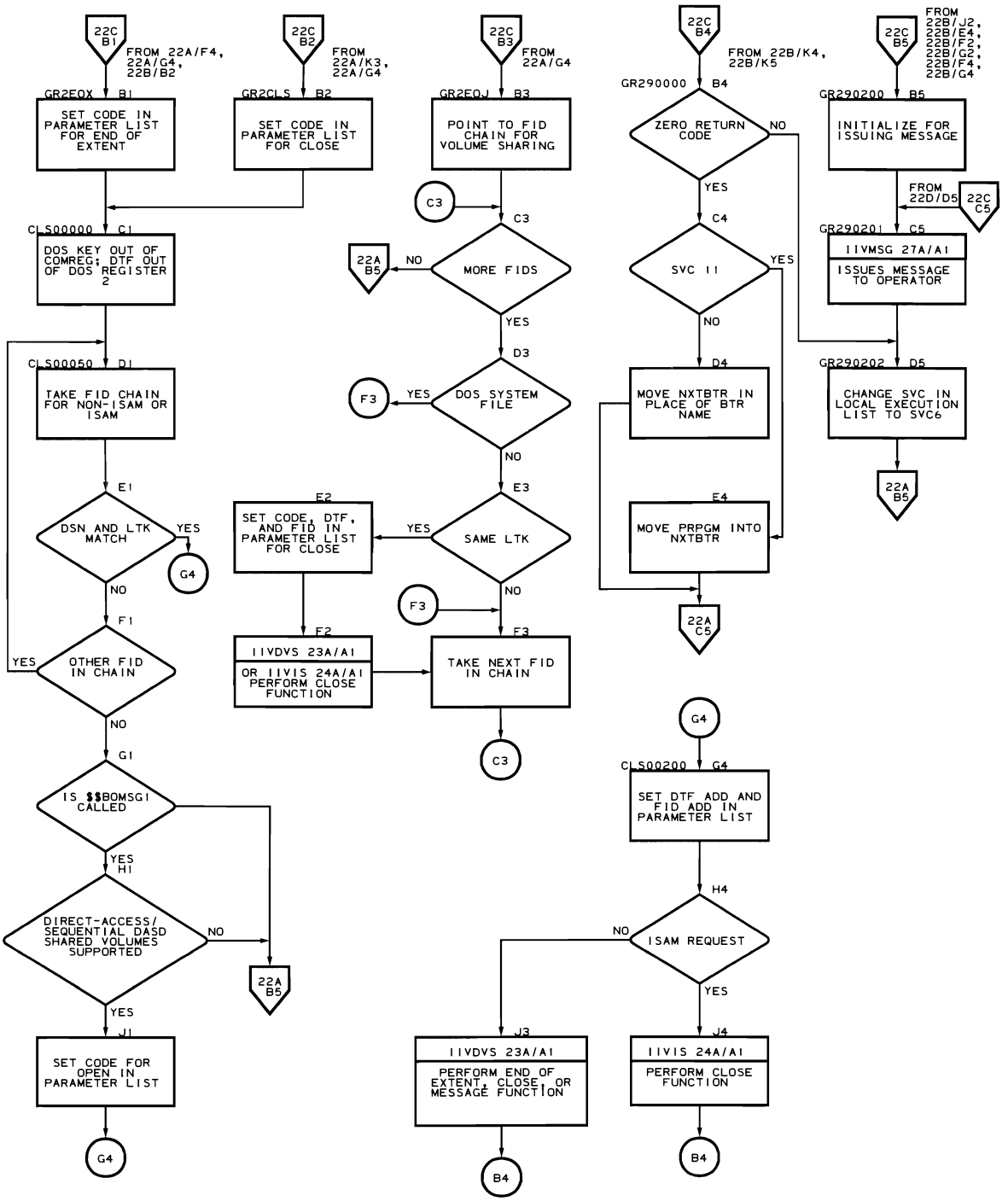
Flowchart 22A. SVC Monitor Routine (IIVGR2 Part 1 of 4)



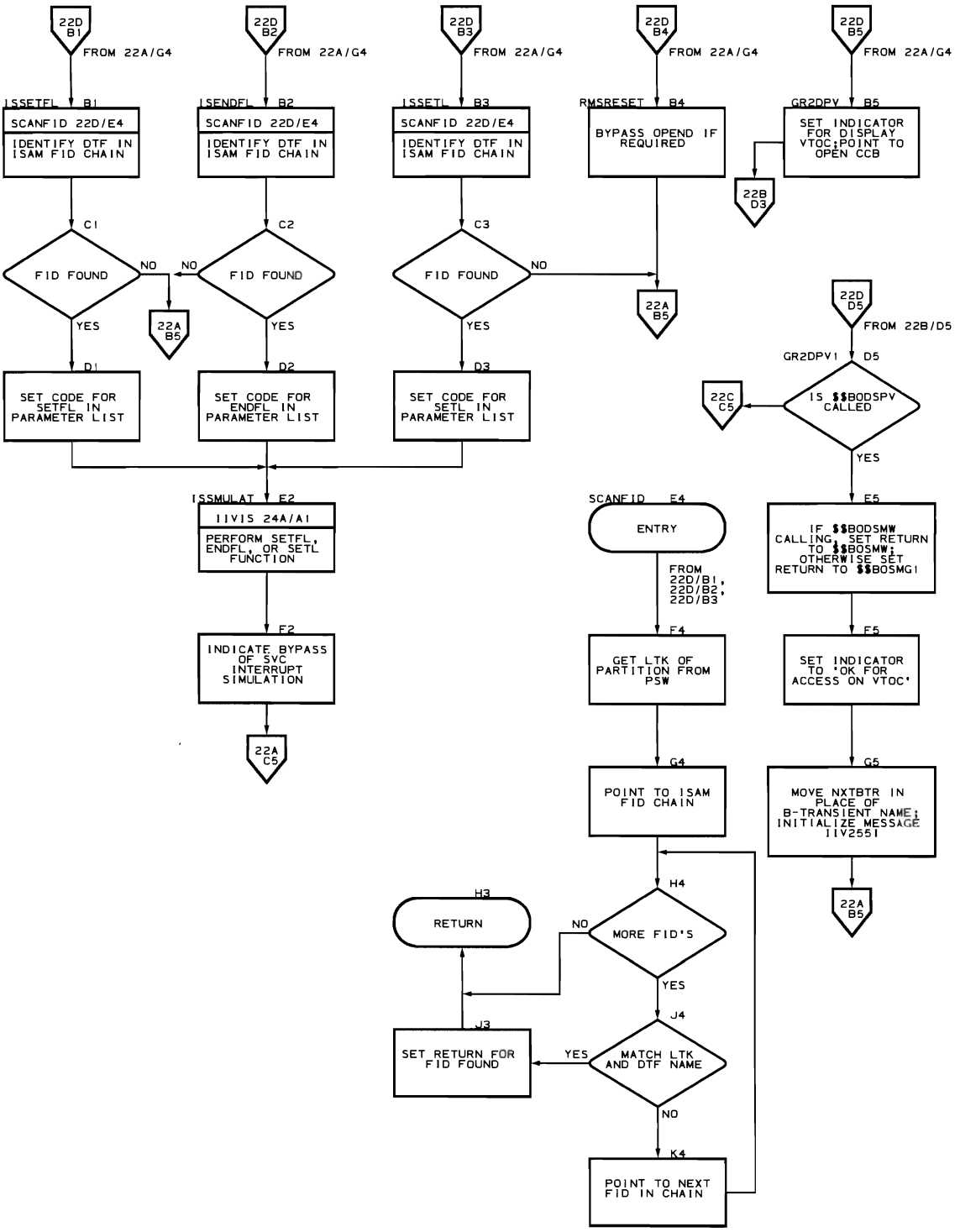
Flowchart 22B. SVC Monitor Routine (IIVGR2 Part 2 of 4)



Flowchart 22C. SVC Monitor Routine (IIVGR2 Part 3 of 4)



Flowchart 22D. SVC Monitor Routine (IIVGR2 Part 4 of 4)



Device Sharing Simulation Routines (Flowcharts 23A-23H)

Module name: IIVDVS

Entry point name: IIVDVS

Major functions:

- Moves the actual limits of the current extent allocated by OS into the DLBL/EXTENT card image
- Creates (in the obtain routine) VOL1 (volume 1) identifier (format 1), and extension (format 3) DSCB images to be used by module IIVVIO
- Controls the flow of DOS B-transient phases to be executed for open, end-of-extent, and close operations

Entered from: IIVGR2

Modules called:

- IIVGET
- IIVMSG

Exits to: IIVGR2

OS macros issued:

- OPEN
- EOV
- CLOSE
- OBTAIN
- DCBD

Input:

- DTF address, LTK, COMTAB extension pointer (at open time) or FID pointer (at end of extent or close), and PARMCODE in the PARMLST field of EMUCONS.
- COMTAB extension with dsname, ddname, and DCB pointer (at end of extent or close).
- FID with forward and backward pointers, COMTAB extension pointer, DTF name, and DOS LTK (at end of extent or close).
- OLDBTR and NXTBTR fields in EMUCONS.
- Register 10 points to DOS storage.
- Register 11 points to EMUCONS.
- Register 14 contains the return address.
- Register 15 points to module IIVDVS.

Output:

- Register 15 contains return code.
- NXTBTR field in EMUCONS contains next B-transient phase to be executed.
- FID created (at open time) or suppressed (at close).
- COMTAB extension updated with DCB pointer and use count (at open and close).

Return codes:

- 0 = gives control to next B-transient phase pointed to by the NXTBTR field in EMUCONS.
- ≠ 0 = cancels DOS partition.

Tables/work areas:

- COMTAB extension
- FID
- DTF
- DLBL
- Open table
- DOS communication region
- EMUCONS

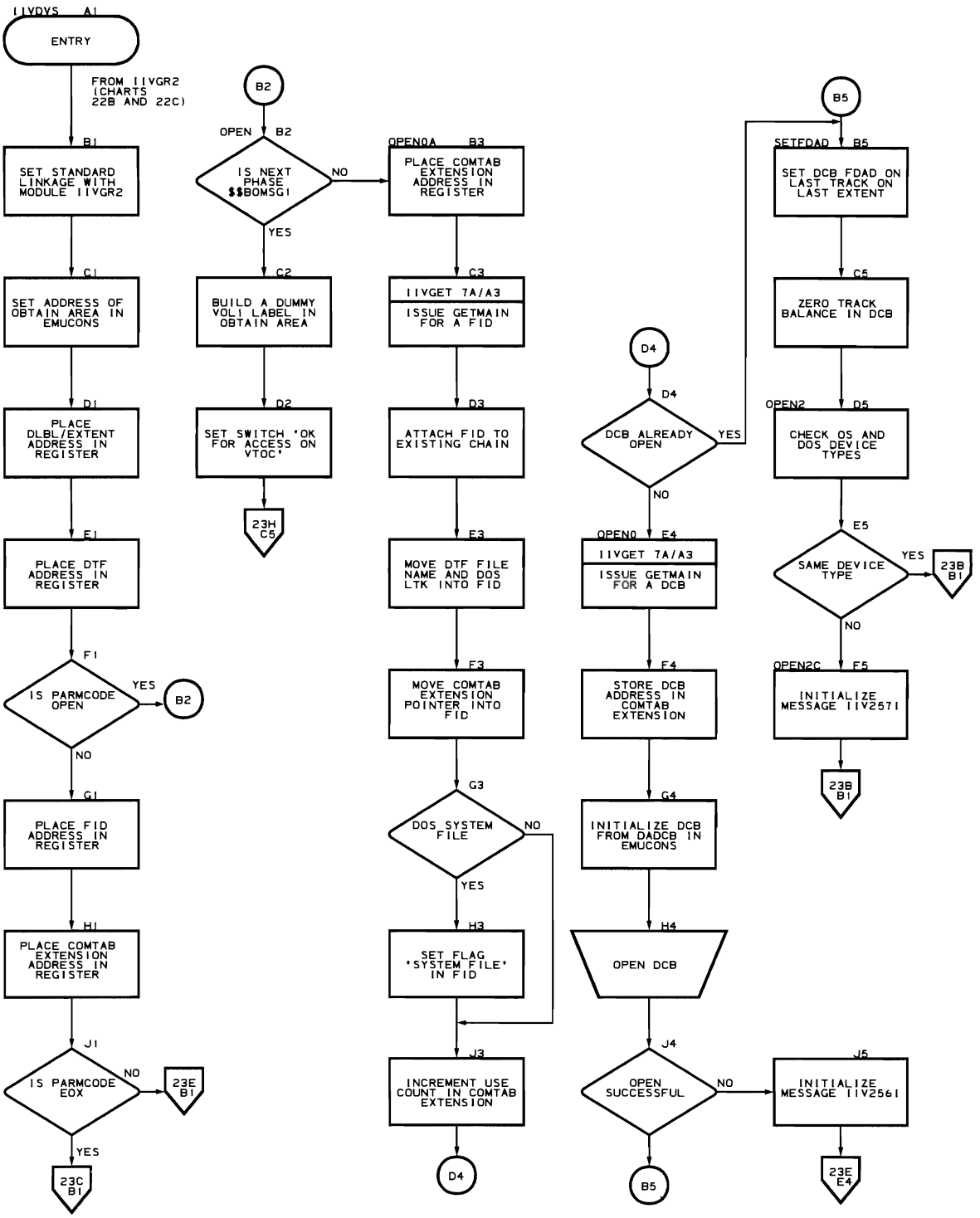
Errors detected:

- Open failure for a data set
- DCB and DTF device types incompatible
- Insufficient extent space in DOS DLBL/EXTENT image for DTF
- User labels specified in DTF but not in DSCB
- Cannot get more extents for file
- Cannot obtain identifier (format 1) DSCB for a data set
- Point outside extents of a data set

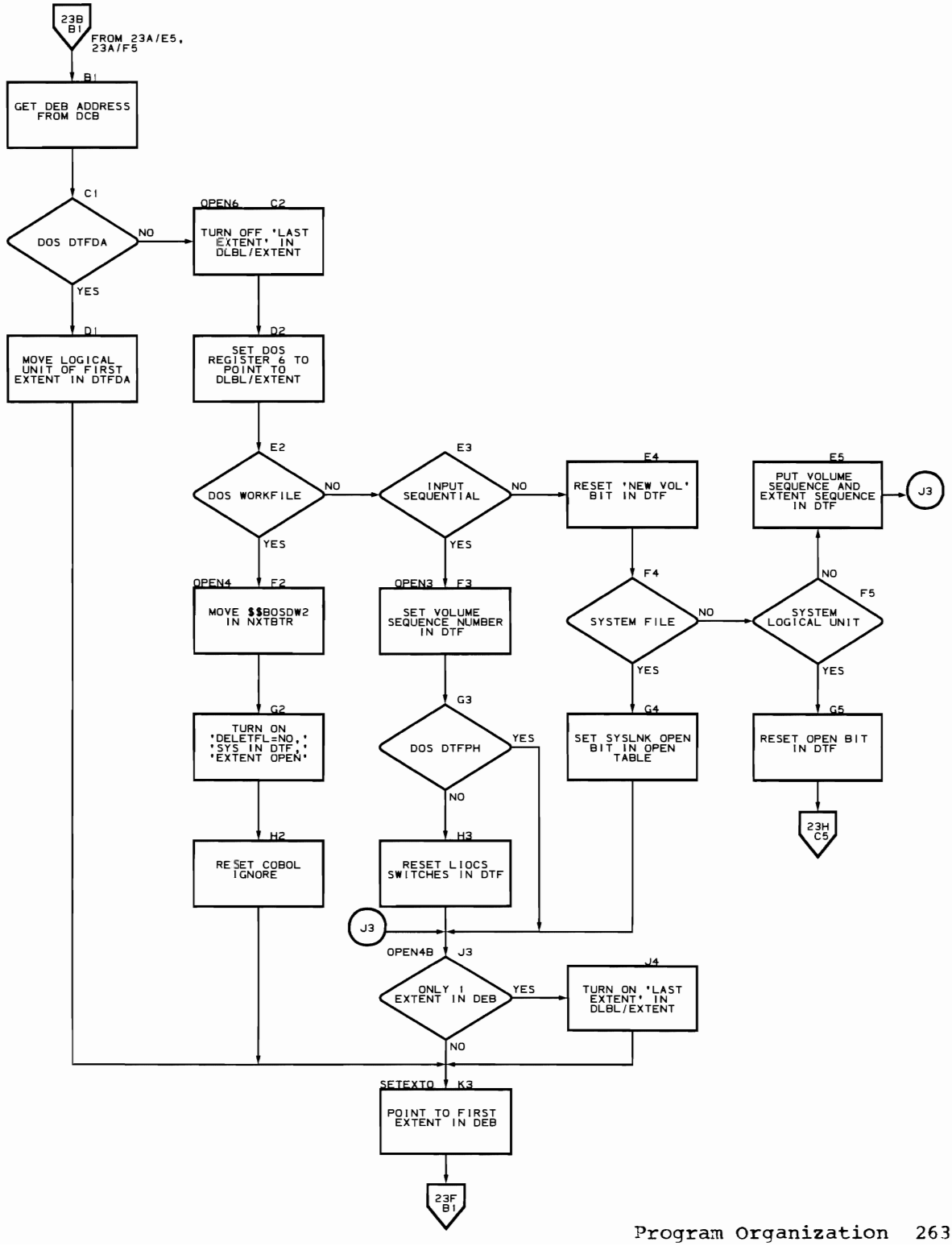
Messages requested:

- IIV256I
- IIV257I
- IIV258I
- IIV259I
- IIV260I
- IIV261I
- IIV262I

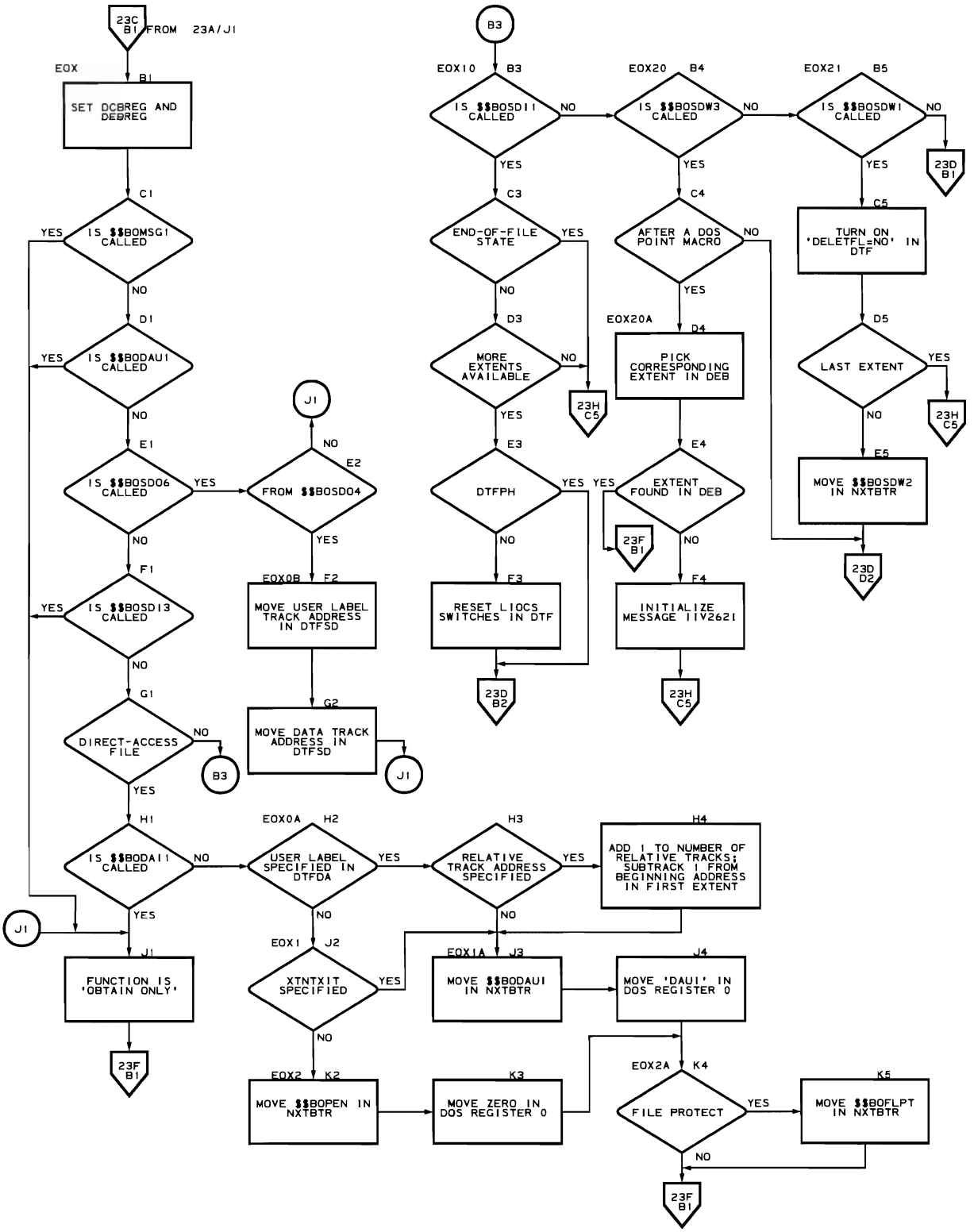
Flowchart 23A. Device Sharing Simulation Routine (IIVDVS Part 1 of 8)



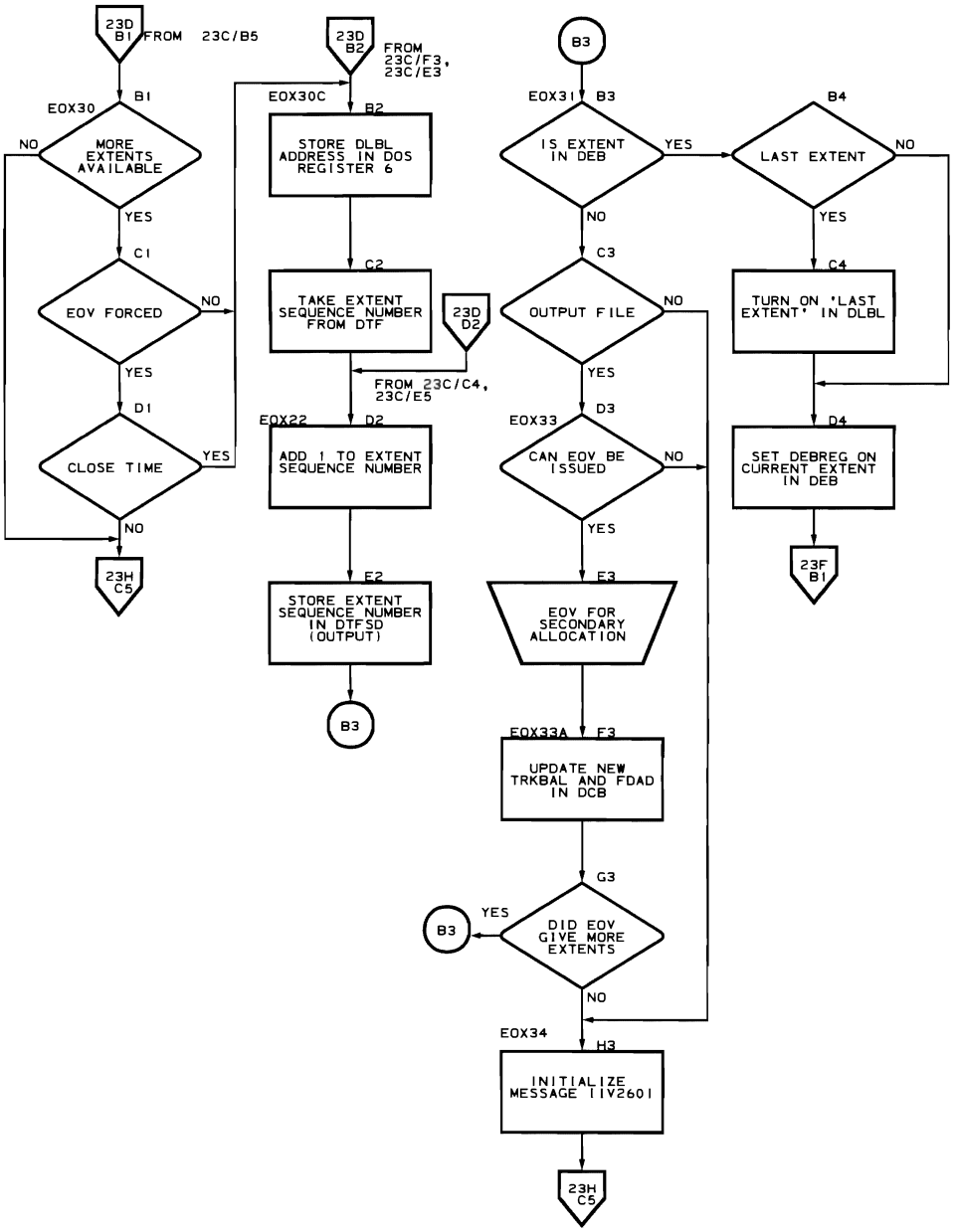
Flowchart 23B. Device Sharing Simulation Routine (IIVDVS Part 2 of 8)



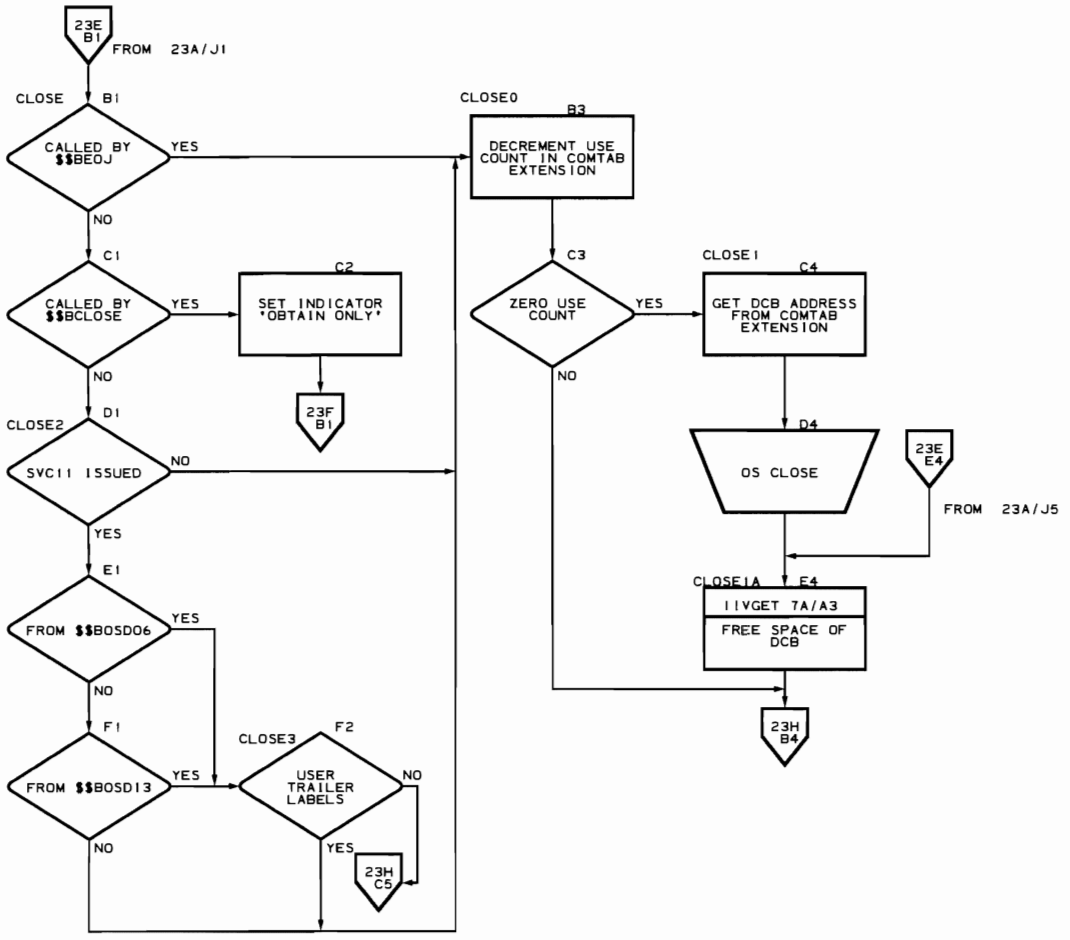
Flowchart 23C. Device Sharing Simulation Routine (IIVDVS Part 3 of 8)



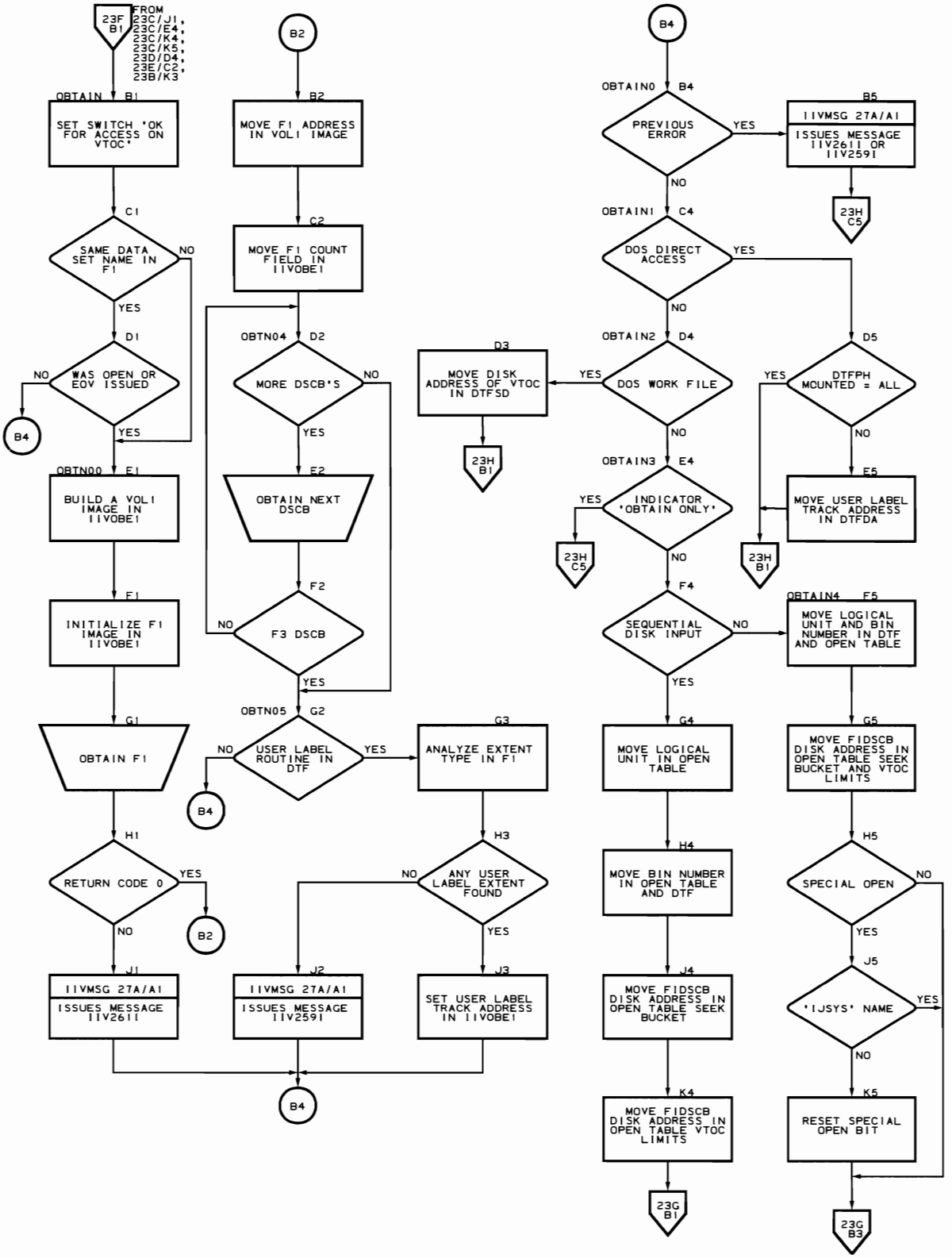
Flowchart 23D. Device Sharing Simulation Routine (IIVDVS Part 4 of 8)



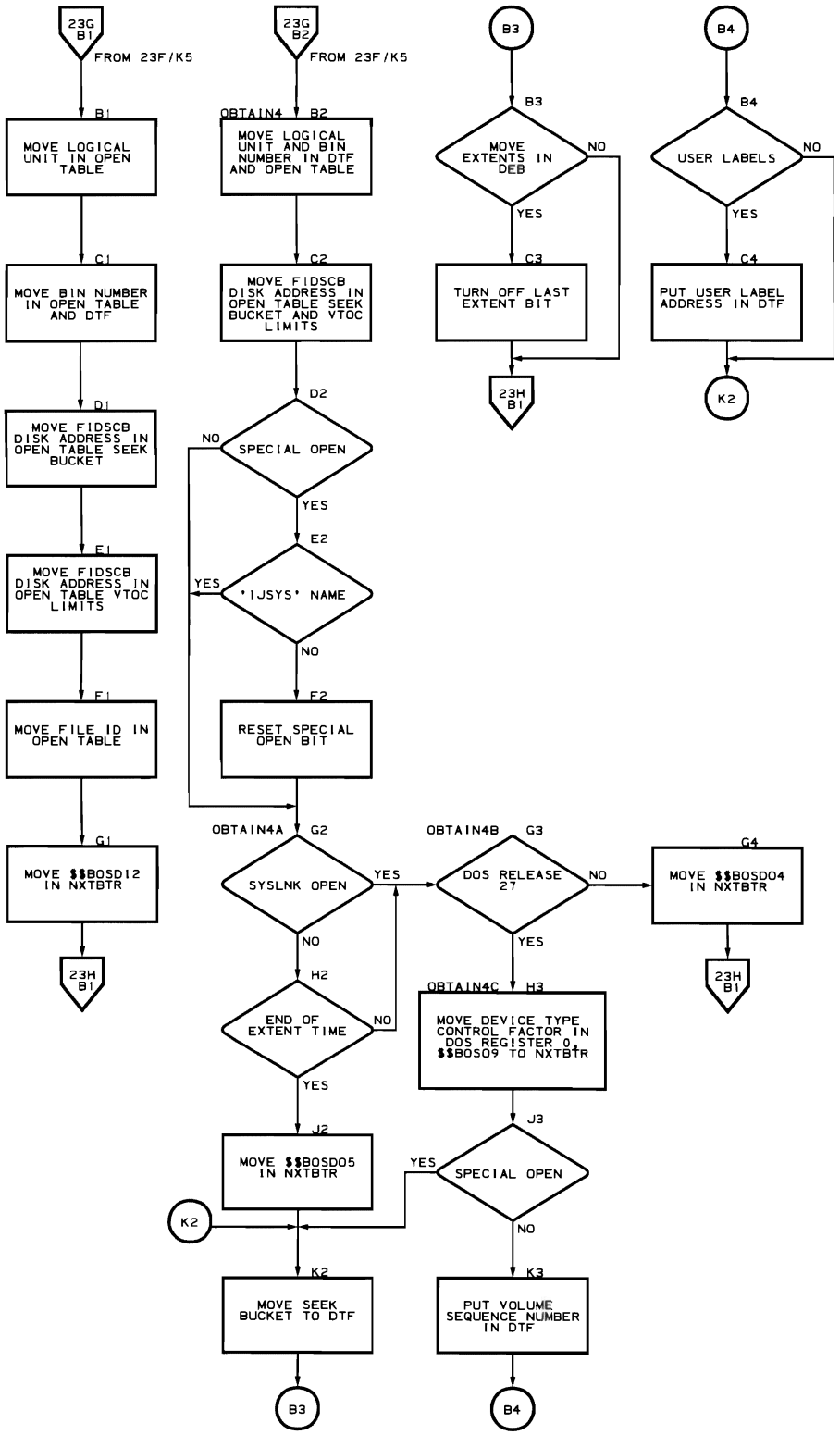
Flowchart 23E. Device Sharing Simulation Routine (IIVDVS Part 5 of 8)



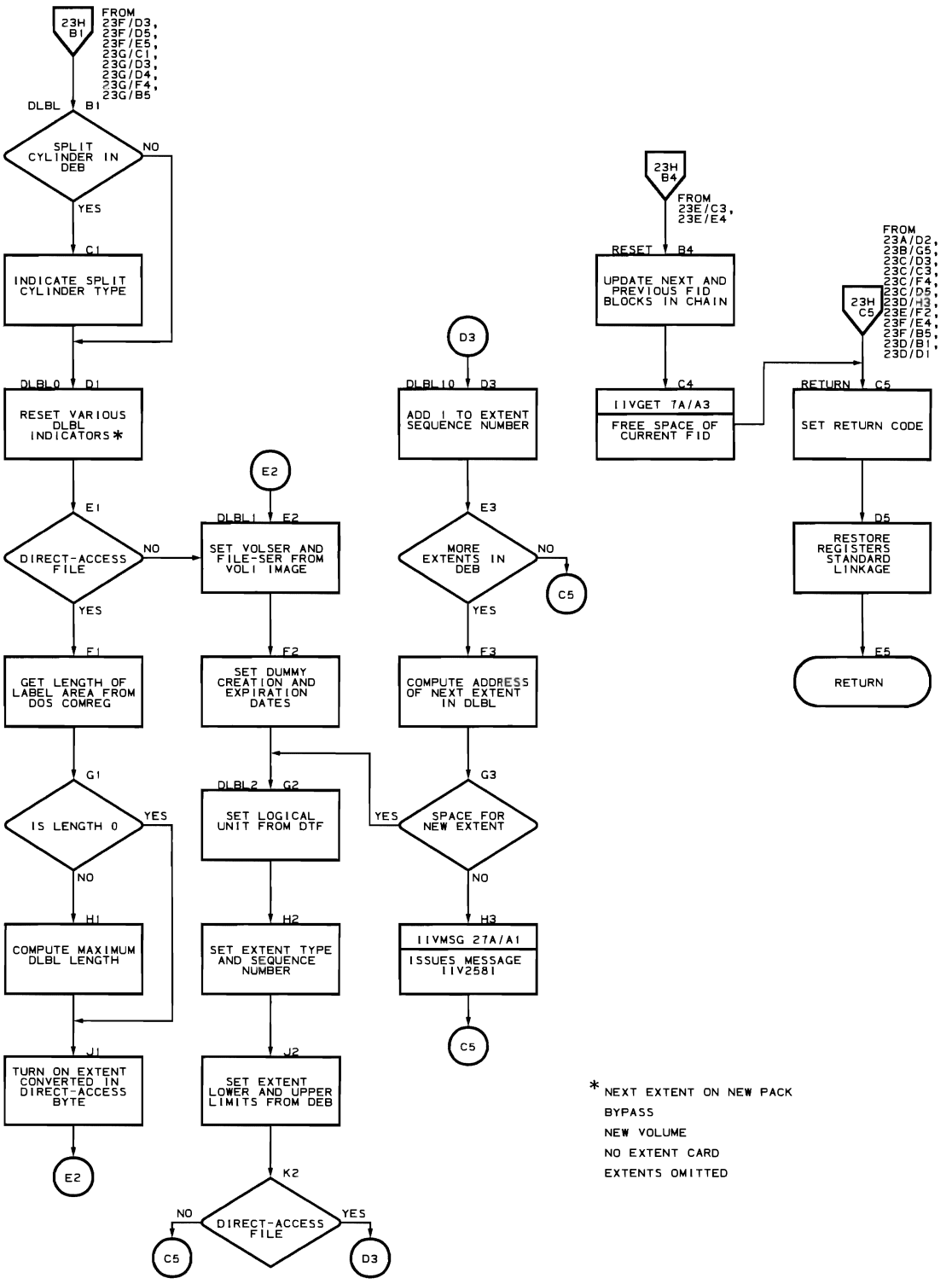
Flowchart 23F. Device Sharing Simulation Routine (IIVDVS Part 6 of 8)



Flowchart 23G. Device Sharing Simulation Routine (IIVDVS Part 7 of 8)



Flowchart 23H. Device Sharing Simulation Routine (IIVDVS Part 8 of 8)



ISAM Mapping Routine (Flowcharts 24A-24L)

Module name: IIVIS

Entry point names:

- IIVIS
- IIVIS01

Major function: To map requested DOS ISAM I/O macro instructions to enable accessing OS indexed sequential data sets.

Entered from:

- IIVGR2
- By specifying entry point in ATTACH macro instruction in IIVIS open mapping routine

Modules called: IIVGET

Exits to:

- Caller
- By issuing DETACH macro instruction in IIVIS close mapping routine

OS macros issued:

- OPEN
- OPEN TYPE=J
- CLOSE
- SETL
- RDJFCB
- ATTACH
- DETACH
- IDENTIFY
- CHAP
- ESETL
- GET
- PUT
- PUTX
- READ KU
- WRITE
- CHECK
- SAVE
- RETURN
- WAIT
- DCBD
- POST
- DCB
- WRITE K
- WRITE KN

Input:

- Pointer to DOS low storage (DOSCORE)
- Pointer to EMUCONS (DSECT for IIVCON)
- Parameter list in EMUCONS (ISAMCODE, ISAMDTFA, ISAMFDCX)
- Pointer to local execution list
- Return address
- Register 1 points to ISBLK (obtained by IIVIS open mapping routine) which contains base registers in register save area section.
- When removed from wait state by POST macro instruction issued by SIO routine (module IIVPCE), ISBLK control information contains I/O macro code, address of DTFIS table, and address of COMTAB entry

Output:

- ISBLK and OPEN DCB
- Error code
- Logical records to DOS problem program from OS indexed sequential data set
- Logical records to OS indexed sequential data set from DOS problem program

Return codes:

- 0 in register 15 - switch SVC PSWs

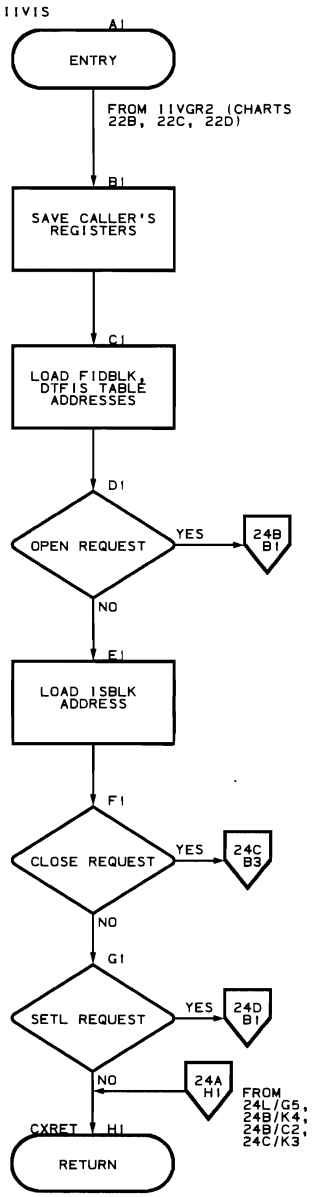
Tables/work areas:

- COMTAB extension
- FIDBLK
- EMUCONS
- ISBLK
- DOS DTFIS
- DCB

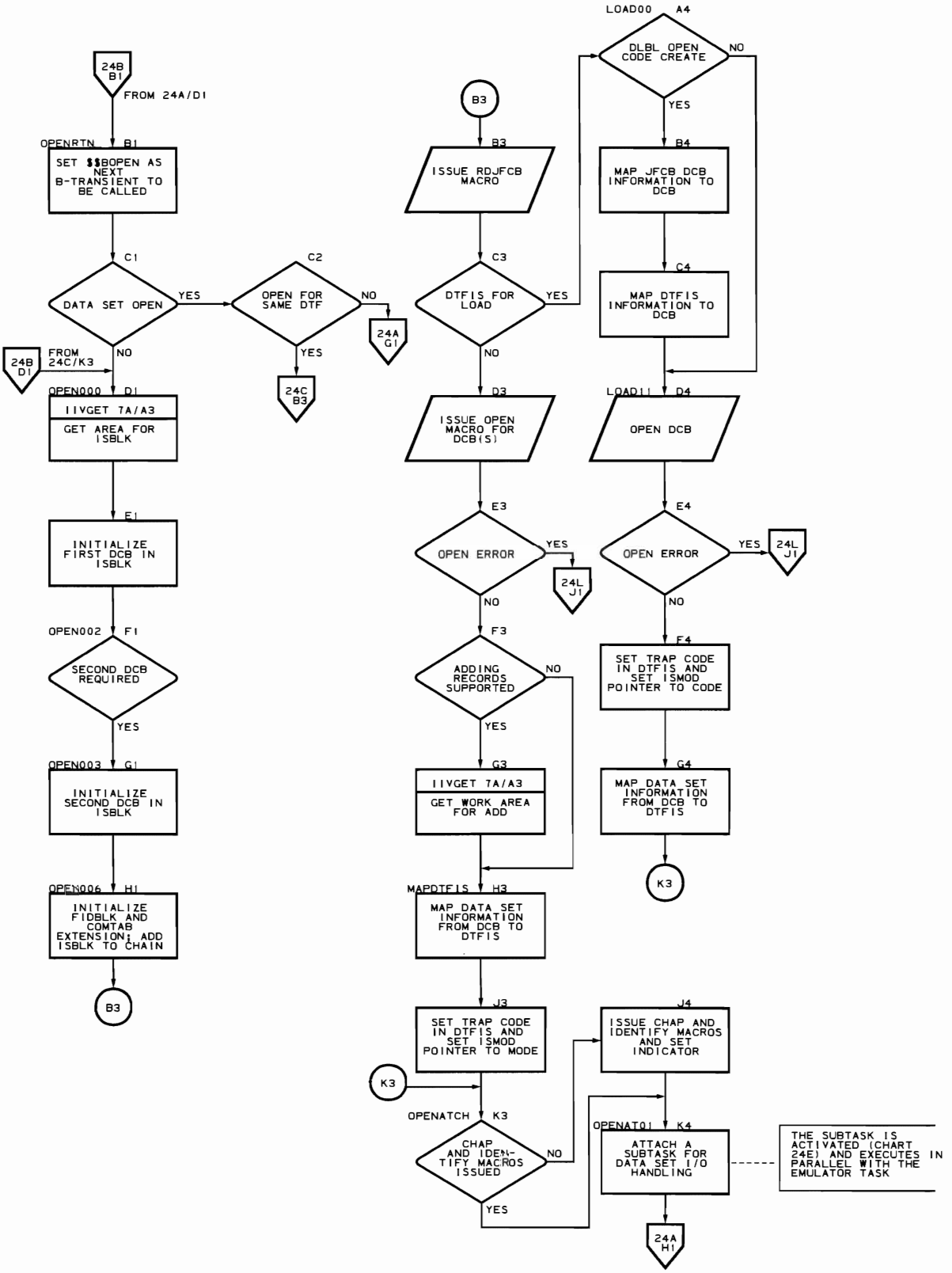
Errors detected: OPEN macro instruction failure

Messages requested: None

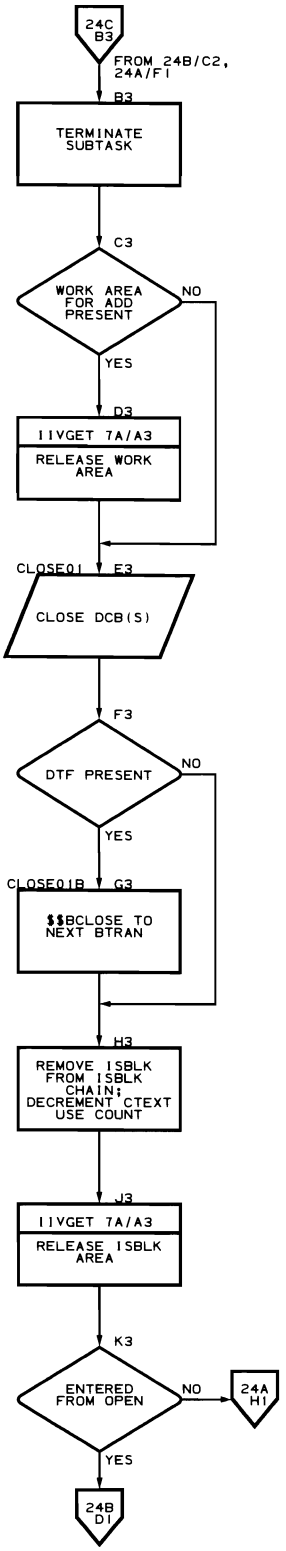
Flowchart 24A. Main Task Control Executive Routine (IIVIS Part 1 of 11)



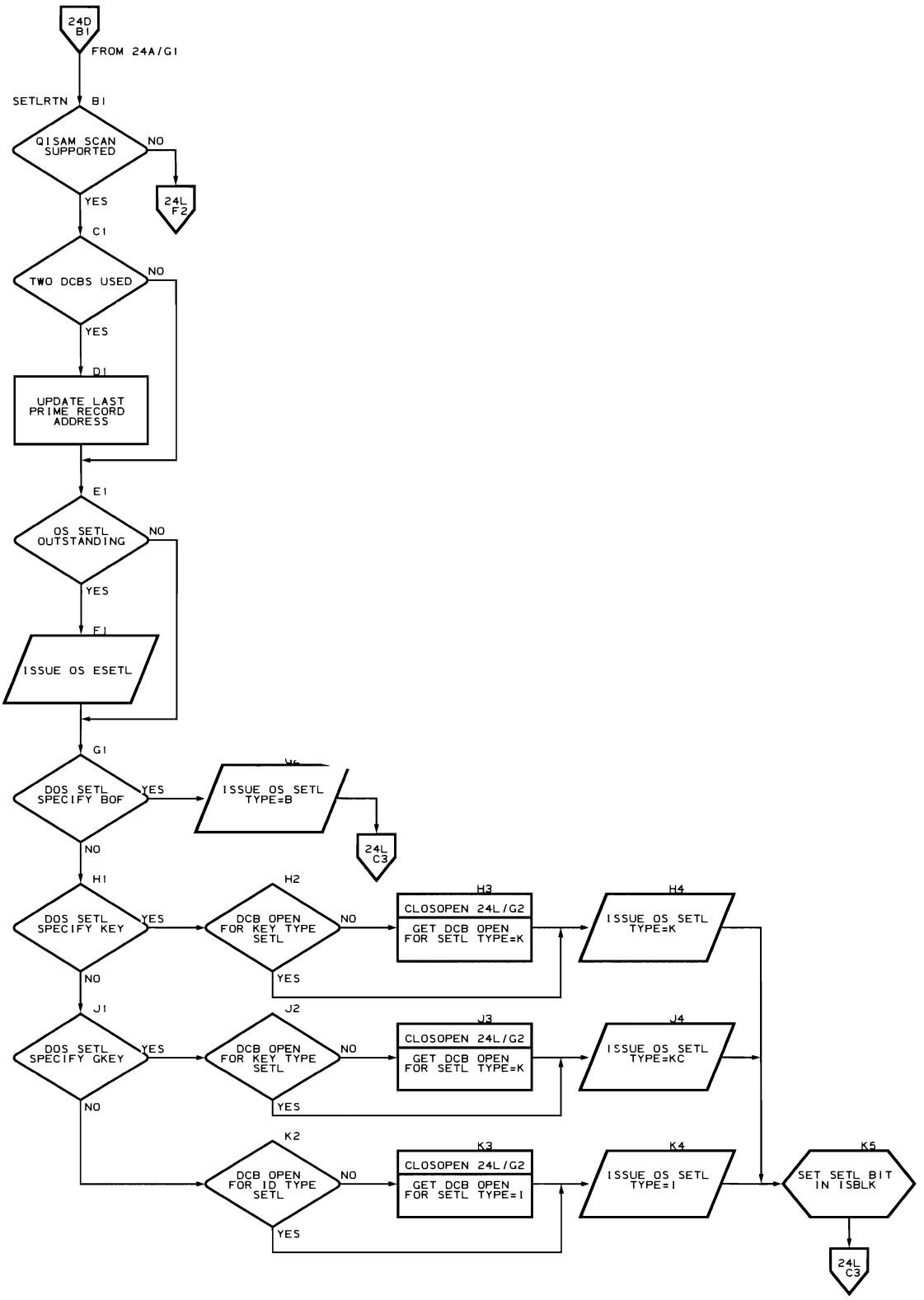
Flowchart 24B. Open Mapping Routine (IIVIS Part 2 of 11)



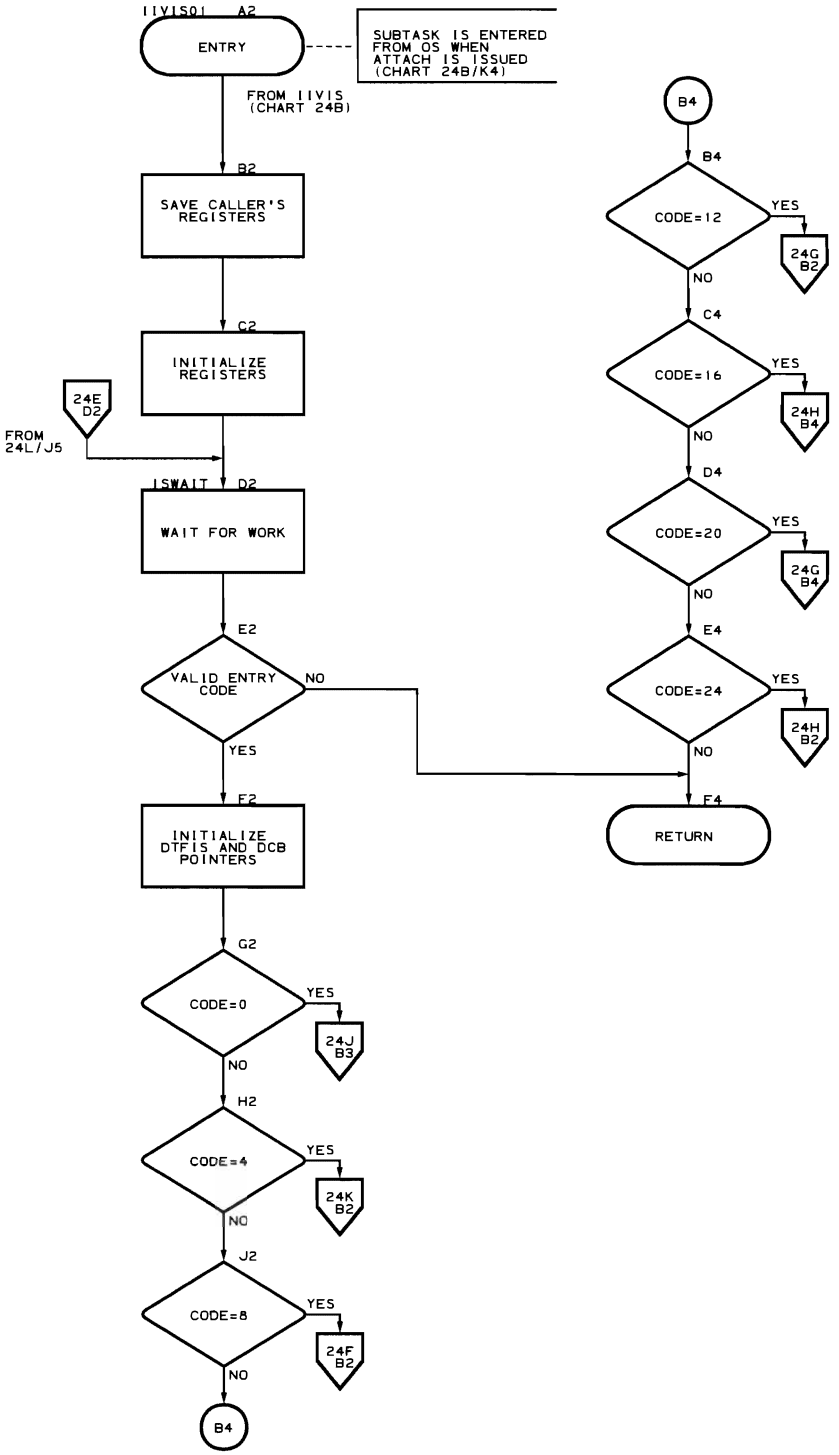
Flowchart 24C. Close Mapping Routine (IIVIS Part 3 of 11)



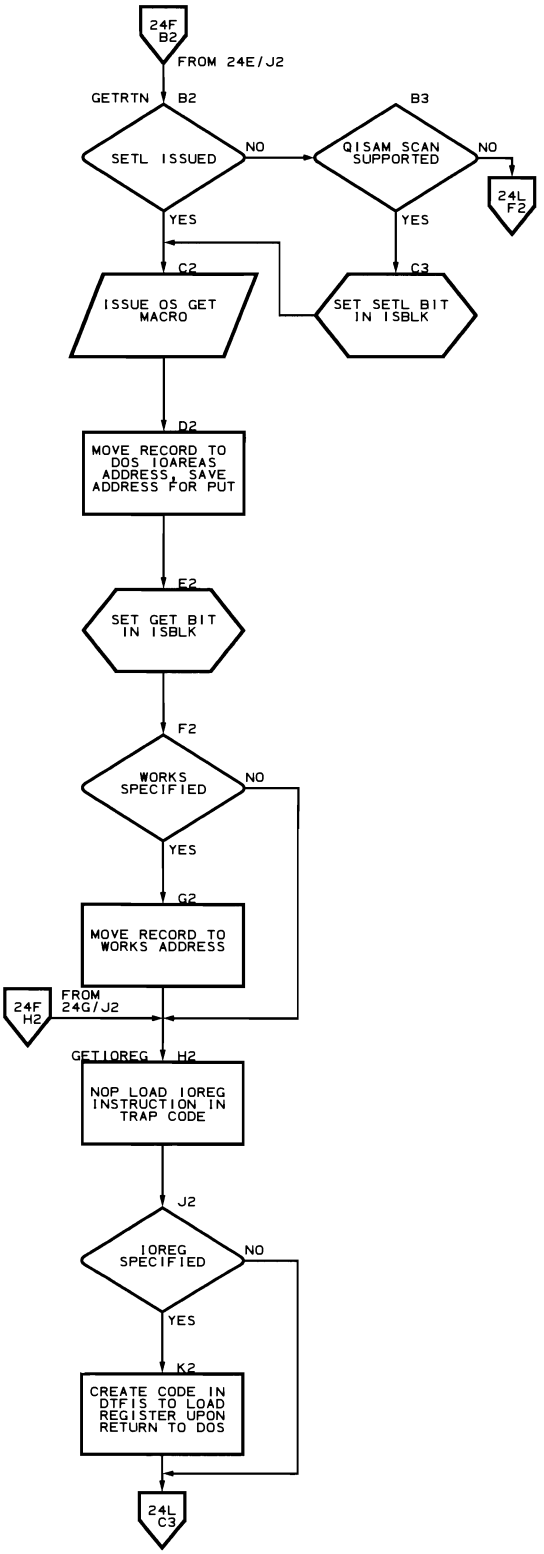
Flowchart 24D. SETL Mapping Routine (IIVIS Part 4 of 11)



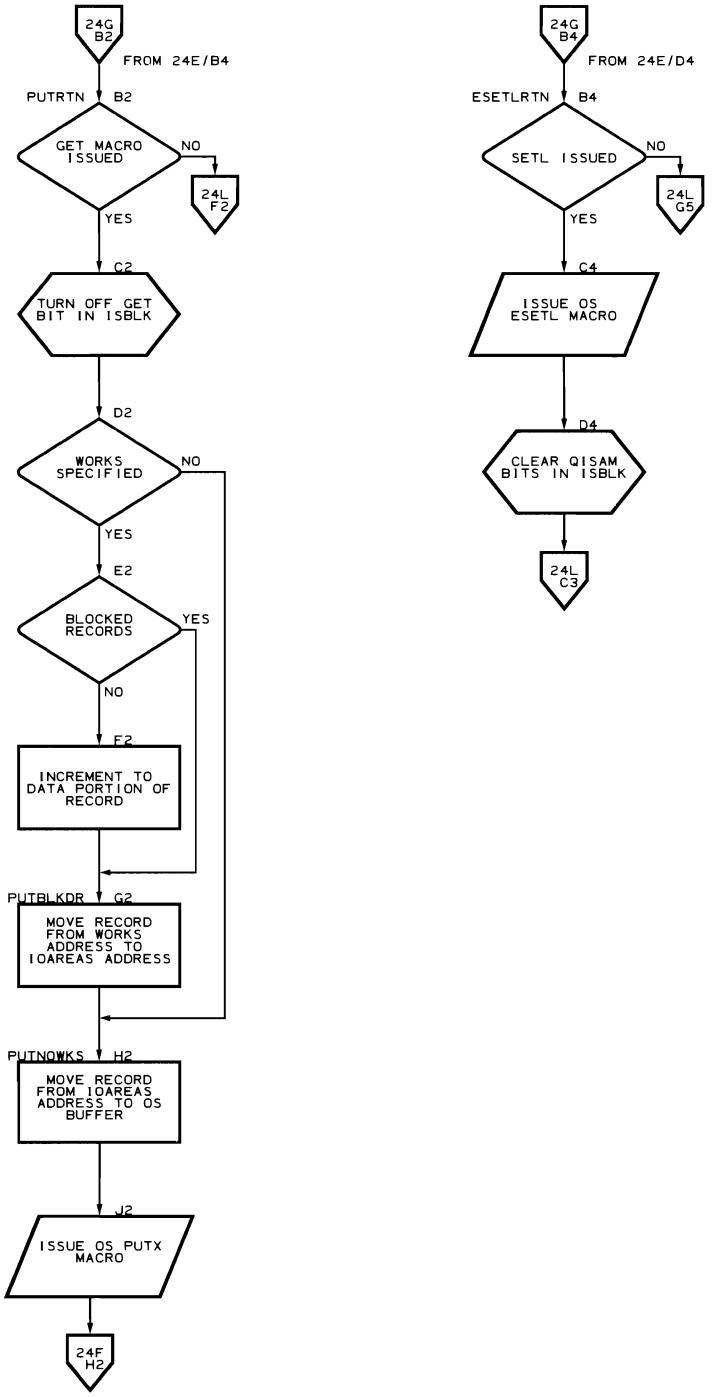
Flowchart 24E. Subtask Control Executive Routine (IIVIS Part 5 of 11)



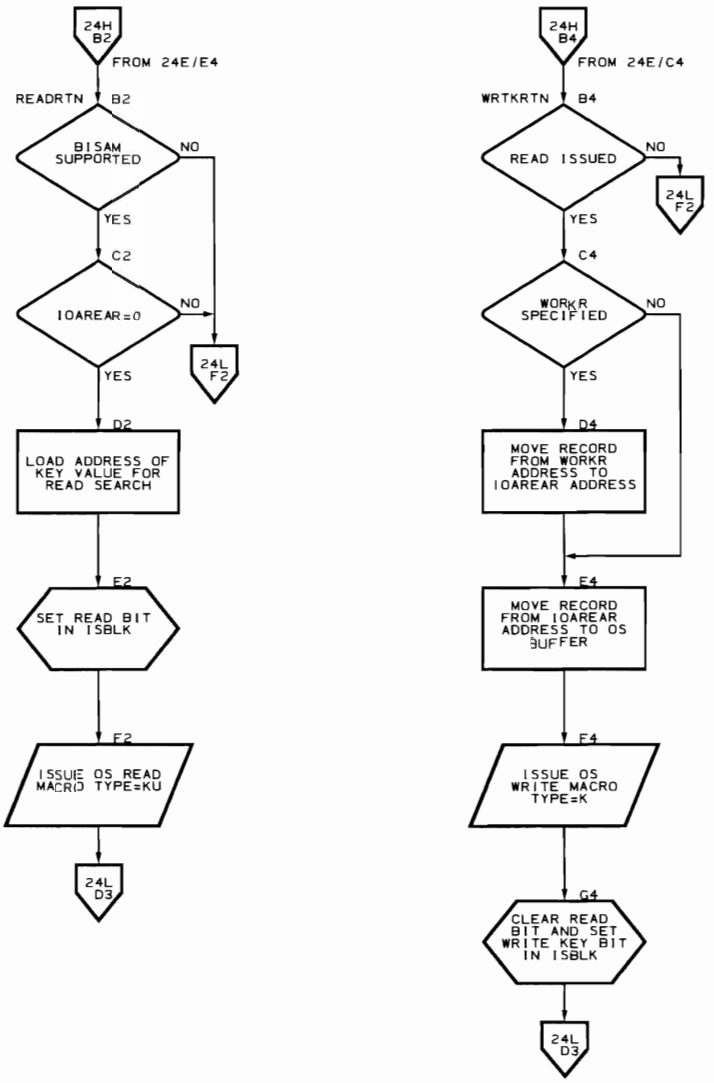
Flowchart 24F. Get Mapping Routine (IIVIS Part 6 of 11)



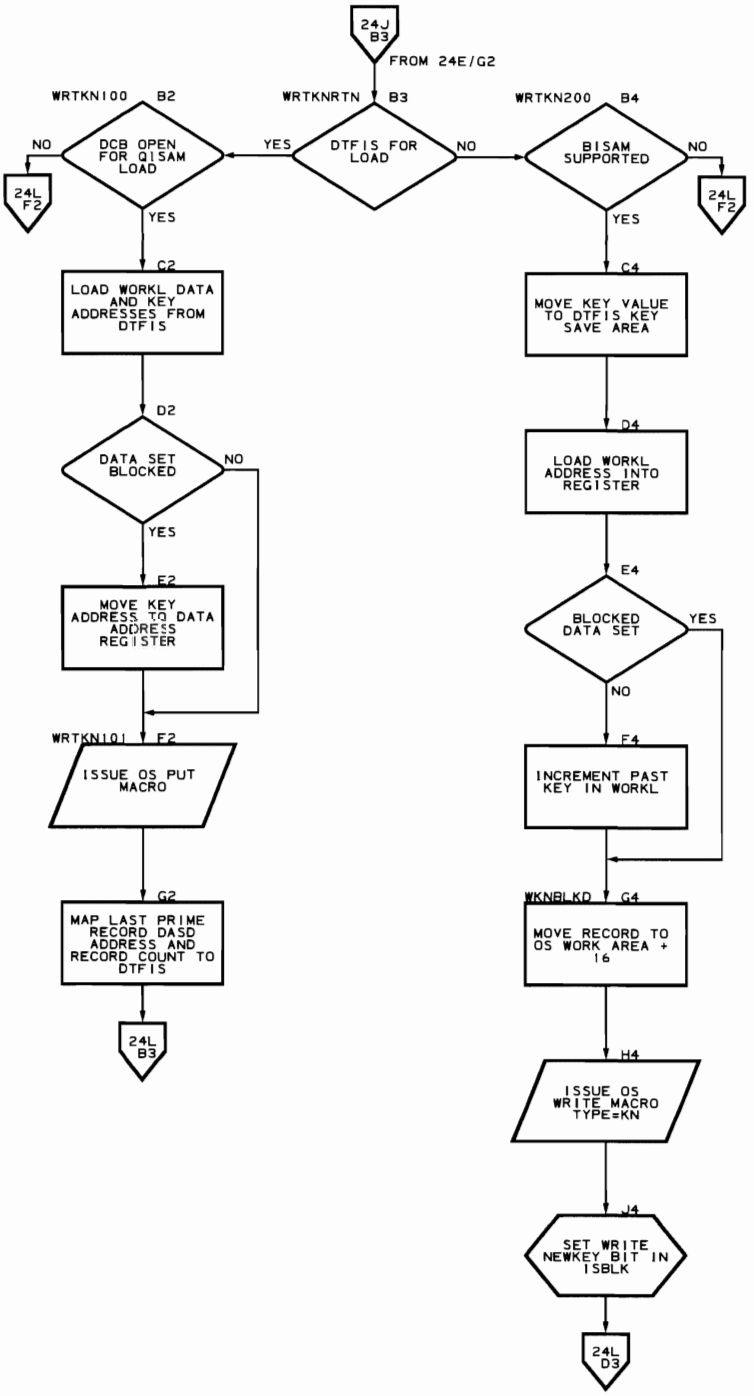
Flowchart 24G. Put and ESETL Mapping Routine (IIVIS Part 7 of 11)



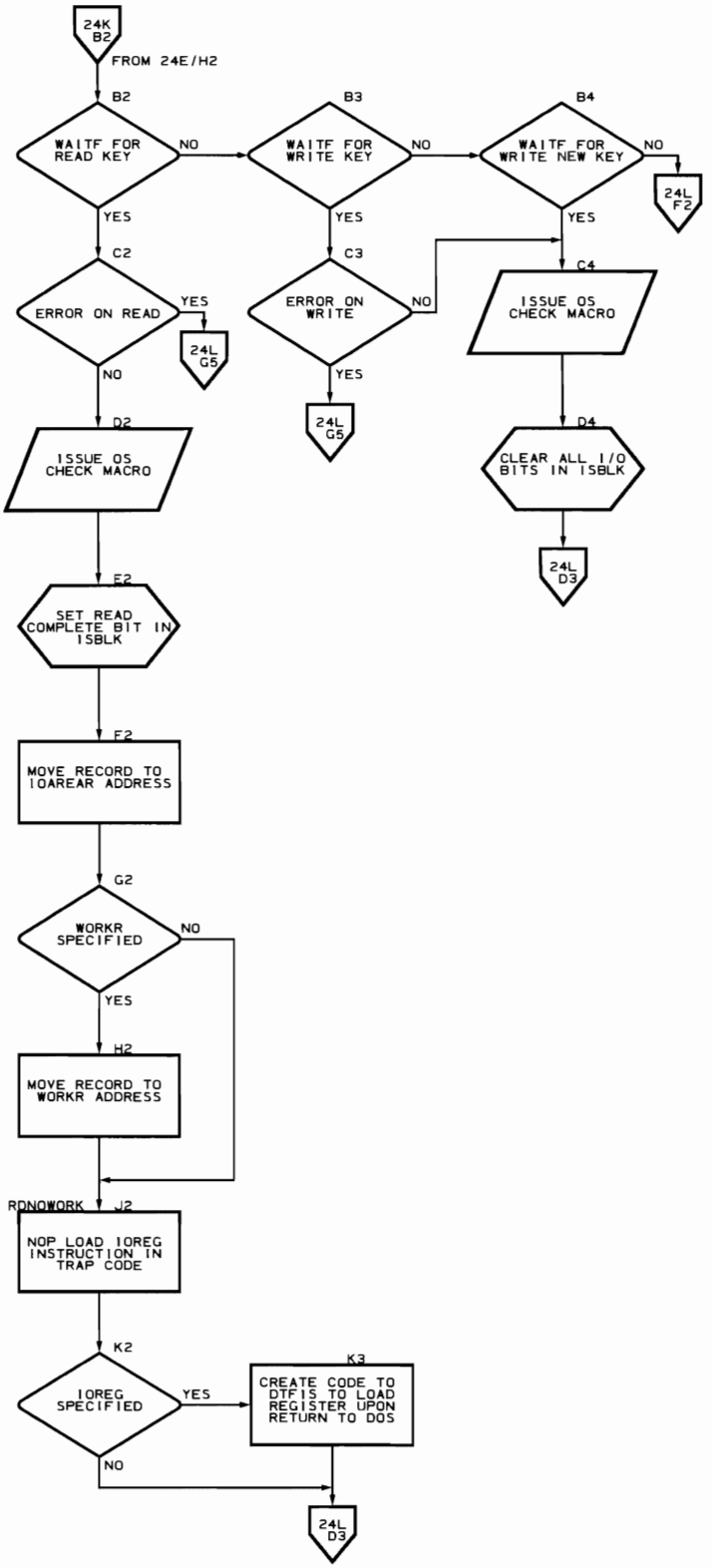
Flowchart 24H. Read Key and Write Key Mapping Routines (IIVIS Part 8 of 11)



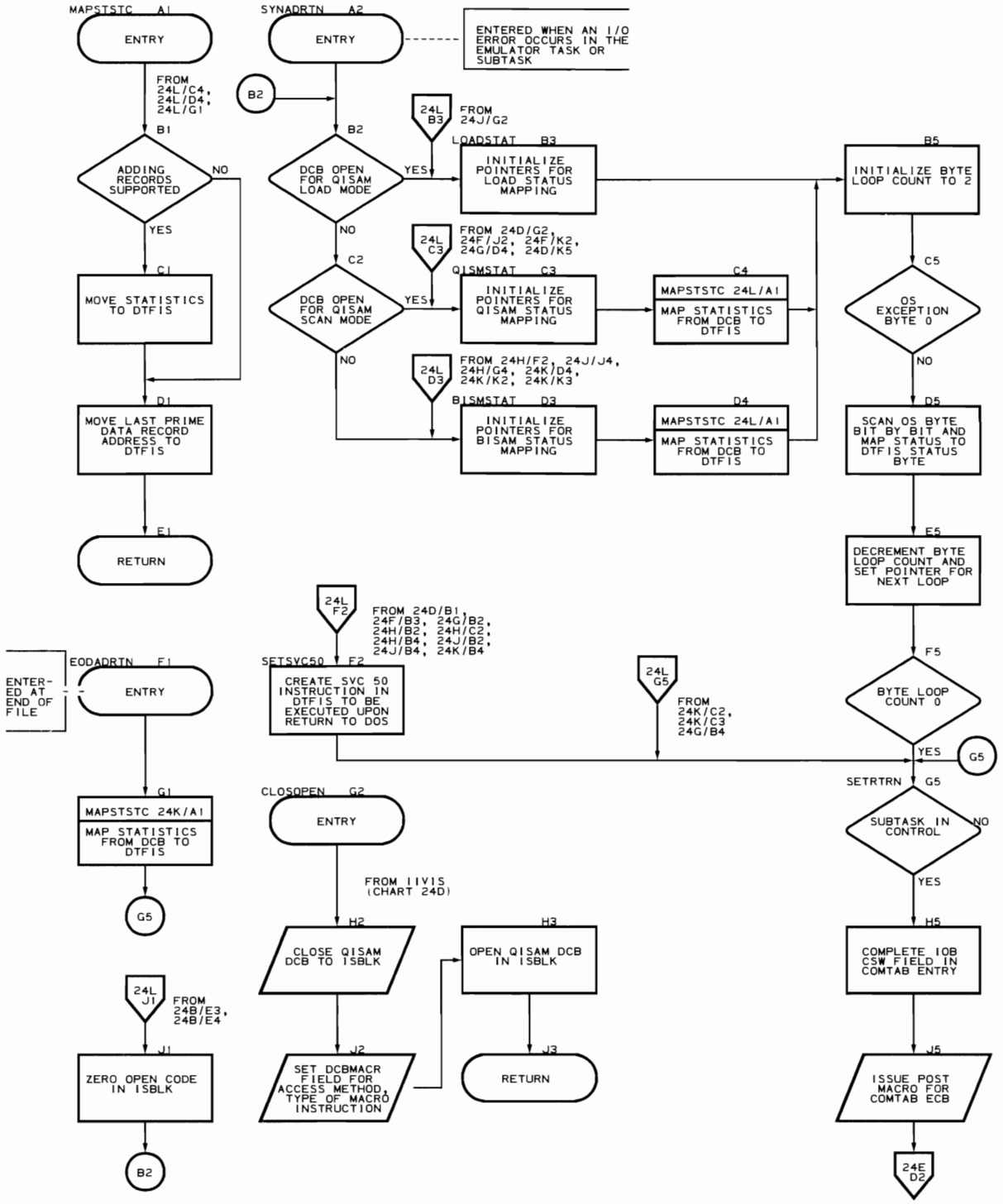
Flowchart 24J. Write NEWKEY Mapping Routine (IIVIS Part 9 of 11)



Flowchart 24K. WAITF Mapping Routine (IIVIS Part 10 of 11)



Flowchart 24L. EODAD, SYNAD, and Status Mapping Routines (IIVIS Part 11 of 11)



VTOC I/O Simulation Routines (Flowcharts 25A-25E)

Module name: IIVVIO

Entry point name: IIVVIO

Major functions (for shared volumes only):

- Provides simulation of VTOC I/O from DOS open phases
- Provides user label extent information from actual user label I/O

Entered from: IIVPCE

Modules called: None

Exits to: caller

OS macros issued:

- SAVE
- RETURN

Input:

- Pointer to DOS CCWs for I/O request
- Pointer to the COMTAB for the volume on which I/O is requested

Output:

- Updated DOS CCW data areas (VTOC labels)
- User label switch set to specify user label I/O requested

Return codes:

- 00 - EXCP to be issued
- 04 - I/O completely simulated
- 08 - CCW chain (channel program) not recognized

Tables/work areas:

- DOSCORE (CCW strings, CCW data areas)
- Obtain work area (F0, F1 and F3 label values for the volume)
- COMTAB (entry for the volume on which I/O is requested)

Errors detected: Invalid seek address (unrecognizable CCW string or seek address not within VTOC limits or user label extent)

Messages requested: None

Command and Type Code Tables for the VTOC I/O Simulation Routine. The address of the routine to be executed, when exiting from the VTOC I/O simulation routine (IIVVIO chart 25A), is in the branch register. This address depends on the table (see Figure 26) used at entry to VIOA2.

COMMAND CODE TABLE

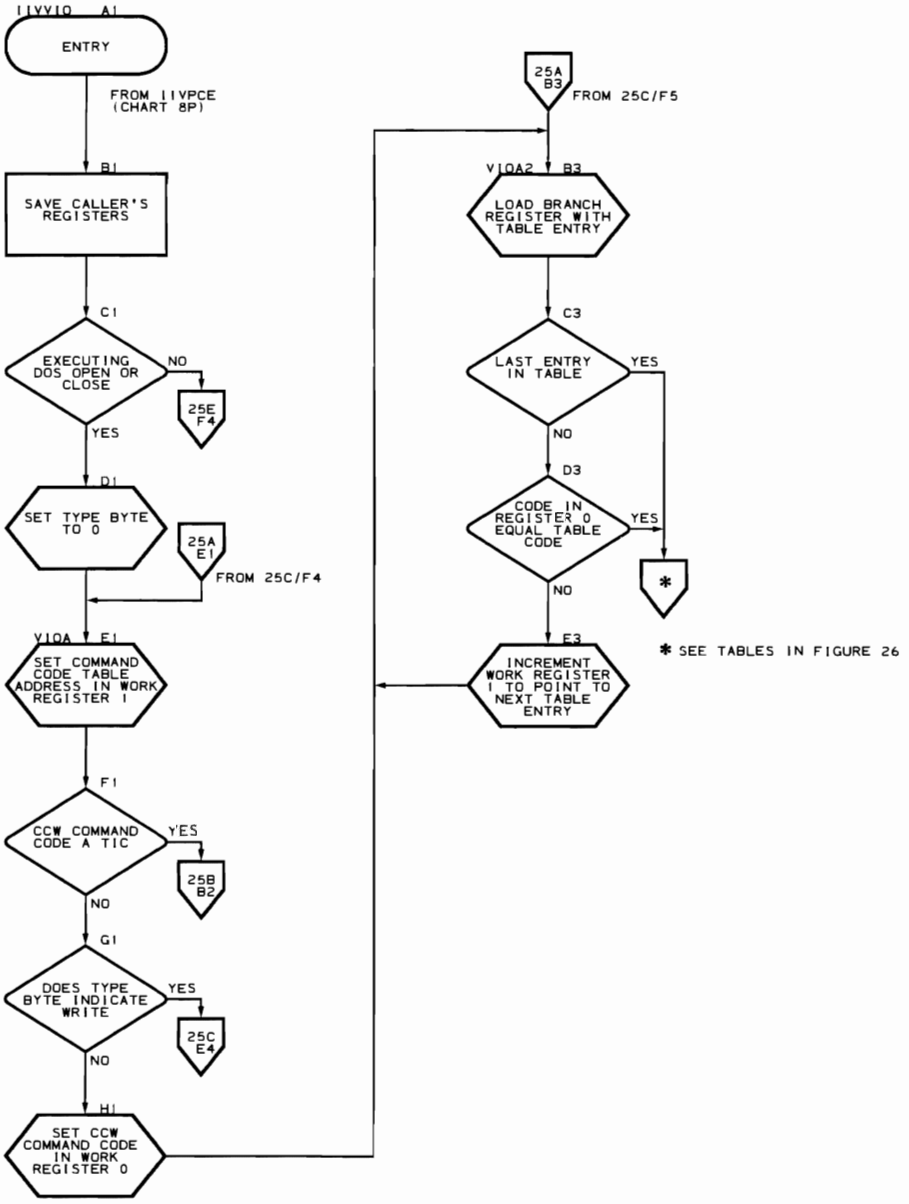
Code	Routine	Chart
X'03'	VIOB	25B/B1
X'07'	VIOC	25B/E1
X'08'	VIOD	25B/B2
X'31'	VIOE	25B/B3
X'29'	VIOF	25B/E2
X'12'	VIOG	25C/B1
X'06'	VIOH	25C/B2
X'0E'	VIOI	25C/B3
X'05'	VIOJ	25C/B4
X'0D'	VIOJ	25C/B4
X'00'	VIOERRX	25E/F4

TYPE CODE TABLE

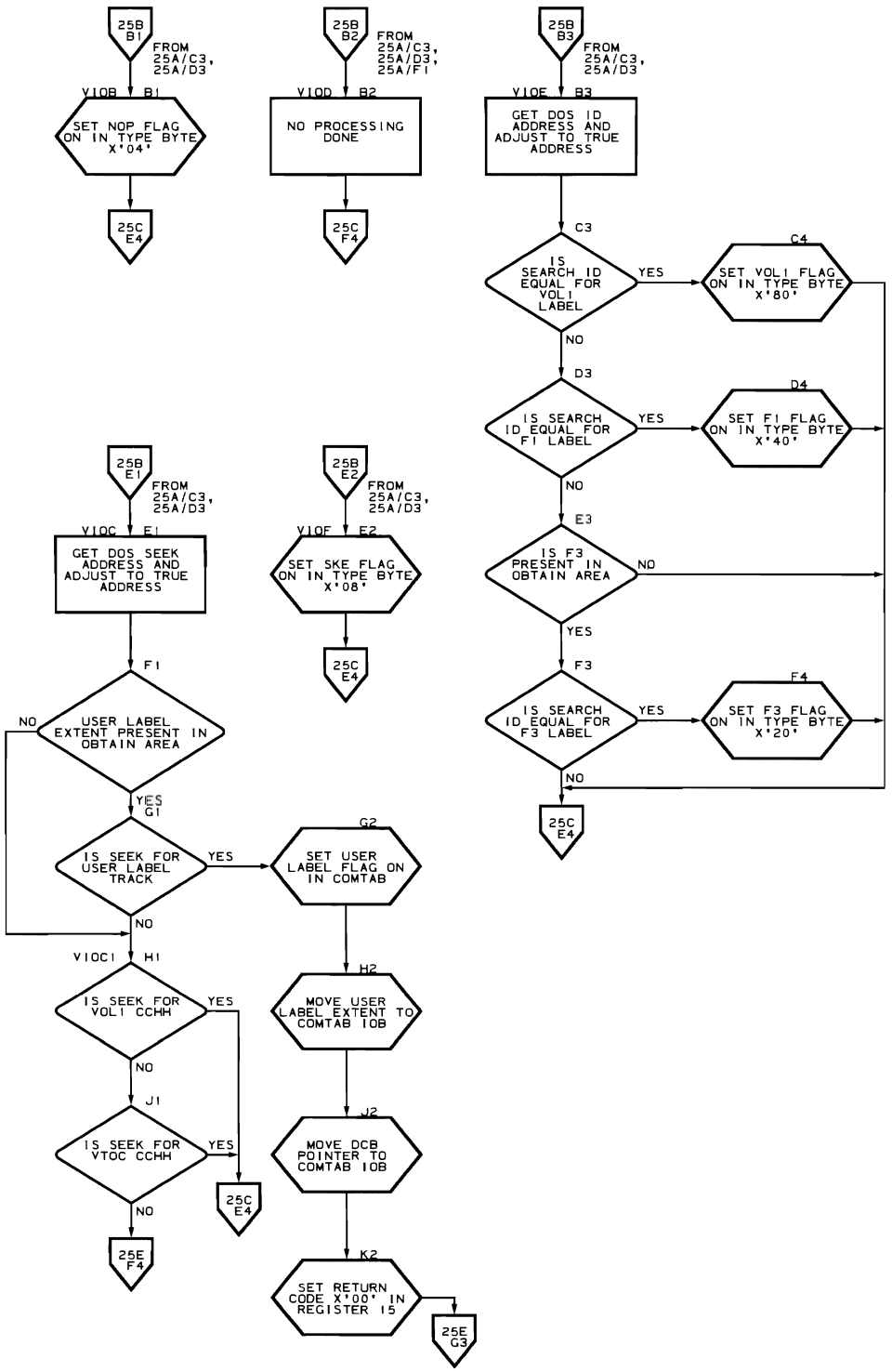
Code	Routine	Chart
X'FF'	VIOIOA	25D/B1
X'81'	VIOIOB	25D/E1
X'82'	VIOIOB1	25D/G2
X'21'	VIOIOC	25D/B3
X'4A'	VIOIOD	25D/B5
X'5A'	VIOIOD	25D/B5
X'41'	VIOIOE	25E/B1
X'1C'	VIOIOF	25E/B3
X'5C'	VIOIOF	25E/B3
X'00'	VIOERRX	25E/F4

Figure 26. Command and Type Code Tables for the VTOC I/O Simulation Routine

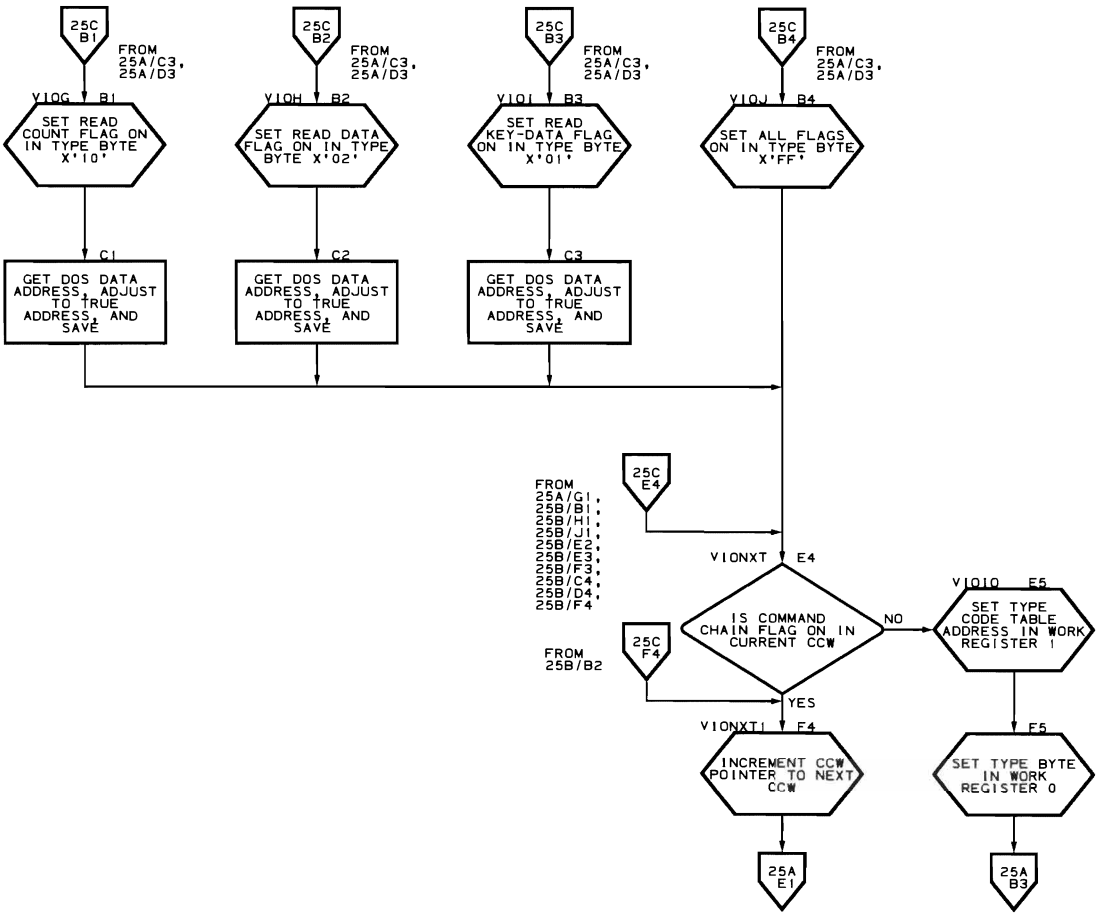
Flowchart 25A. VTOC I/O Simulation Routine (IIVVIO Part 1 of 5)



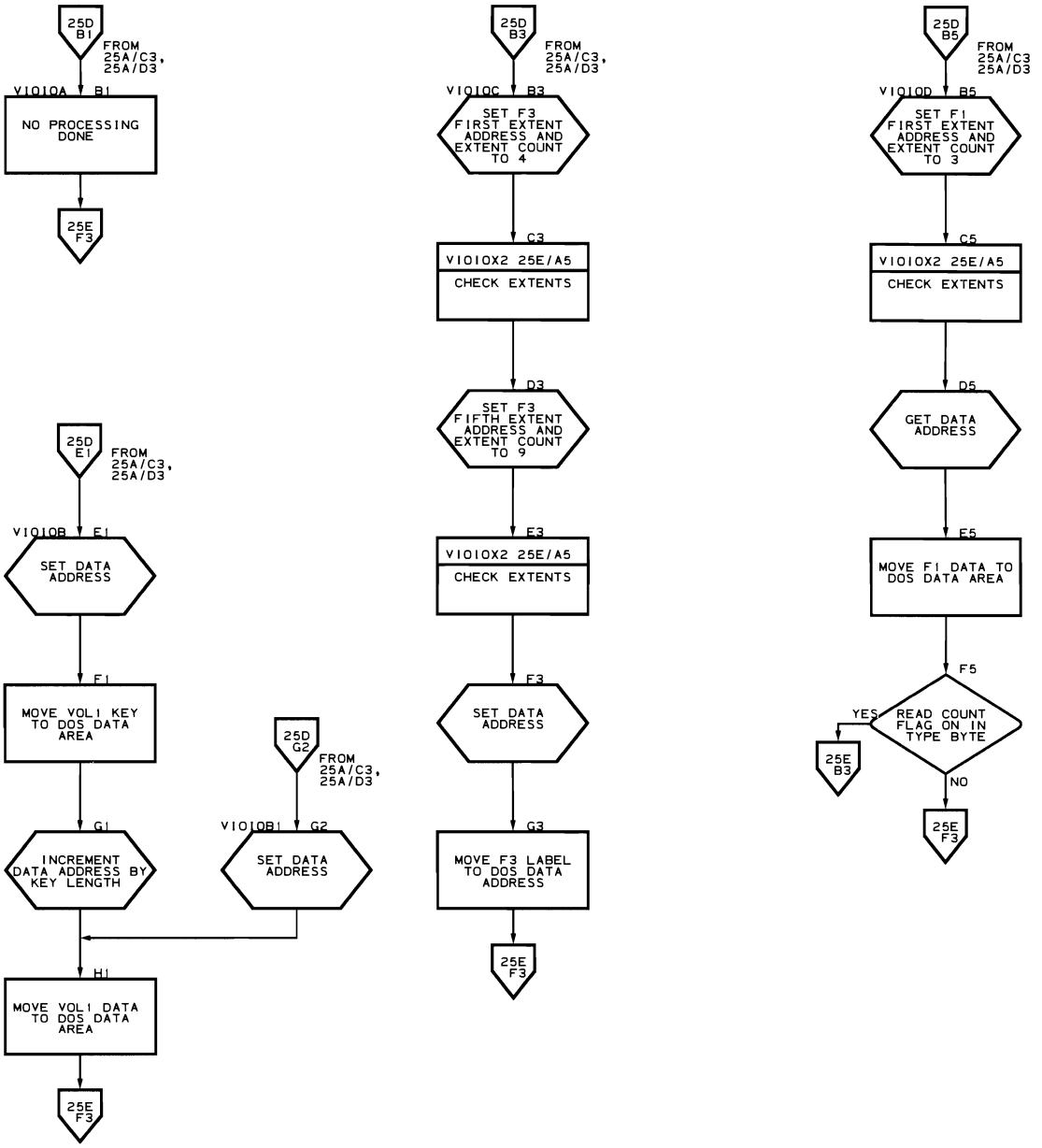
Flowchart 25B. VTOC I/O Simulation Routine (IIVVIO Part 2 of 5)



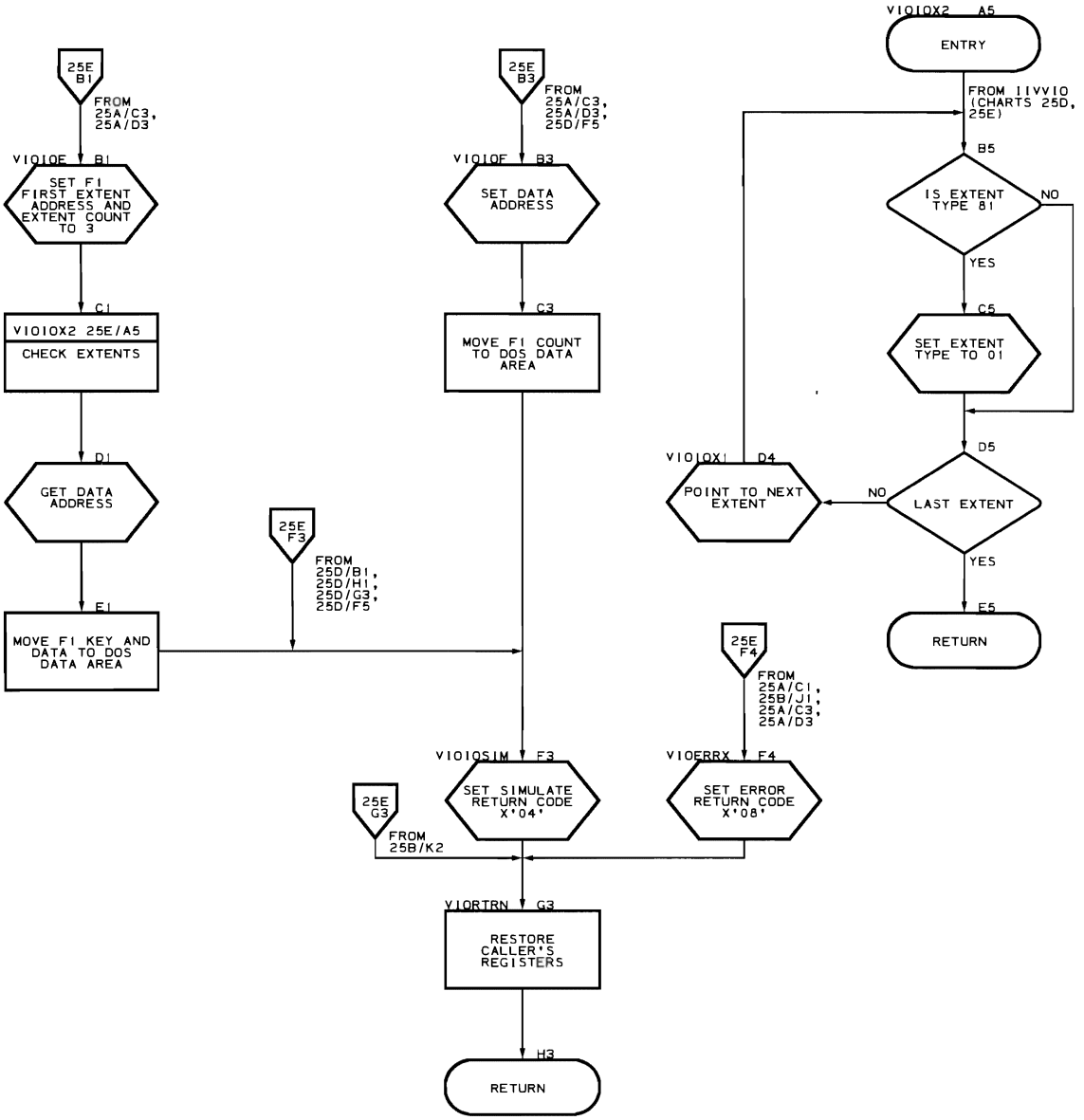
Flowchart 25C. VTOC I/O Simulation Routine (IIVVIO Part 3 of 5)



Flowchart 25D. VTOC I/O Simulation Routine (IIVVIO Part 4 of 5)



Flowchart 25E. VTOC I/O Simulation Routine (IIVVIO Part 5 of 5)



Exit-ABEND Error Routine (Flowcharts 26A-26B)

Module name: IIVABN

Entry point name: JIVABN

Major functions:

- Determines if the return to OS at the end of emulation is normal
- Examines the error code passed from the module in which the error was found and prints the proper message if the calling routine has not already done so
- Dequeues any queued resources
- Posts 'CSW stored' condition code and status bits for channel end-program check to DOS if error concerns only one DOS partition
- Snaps DOS registers and storage using SNAP macro if module IIVRAS not present; if IIVRAS is present, an IIVRAS printout is given
- Returns to OS if emulation ending

Entered by:

- IIVINT
- IIVPCE
- IIVOPN
- IIVRTE
- IIVENT
- IIVGET
- JIVCHK
- IIVSVC
- IIVSNP
- IIVRCP

Modules called:

- IIVRAS
- IIVMSG

Exits to:

- Caller
- OS

OS macros issued:

- SNAP
- DEQ
- LOAD
- SAVE
- RETURN
- DCB
- OPEN

Input:

- Error code in register RP1EU
- Addressability in register RB0CD
- Addressability to COMTAB in RP0EU if entered from IIVCCW at SIO time

Output:

- Snap of contents of DOS registers and storage area at abnormal end of emulation
- DOS CSW and condition code if DOS should end a partition

Return codes: None

Tables/work areas:

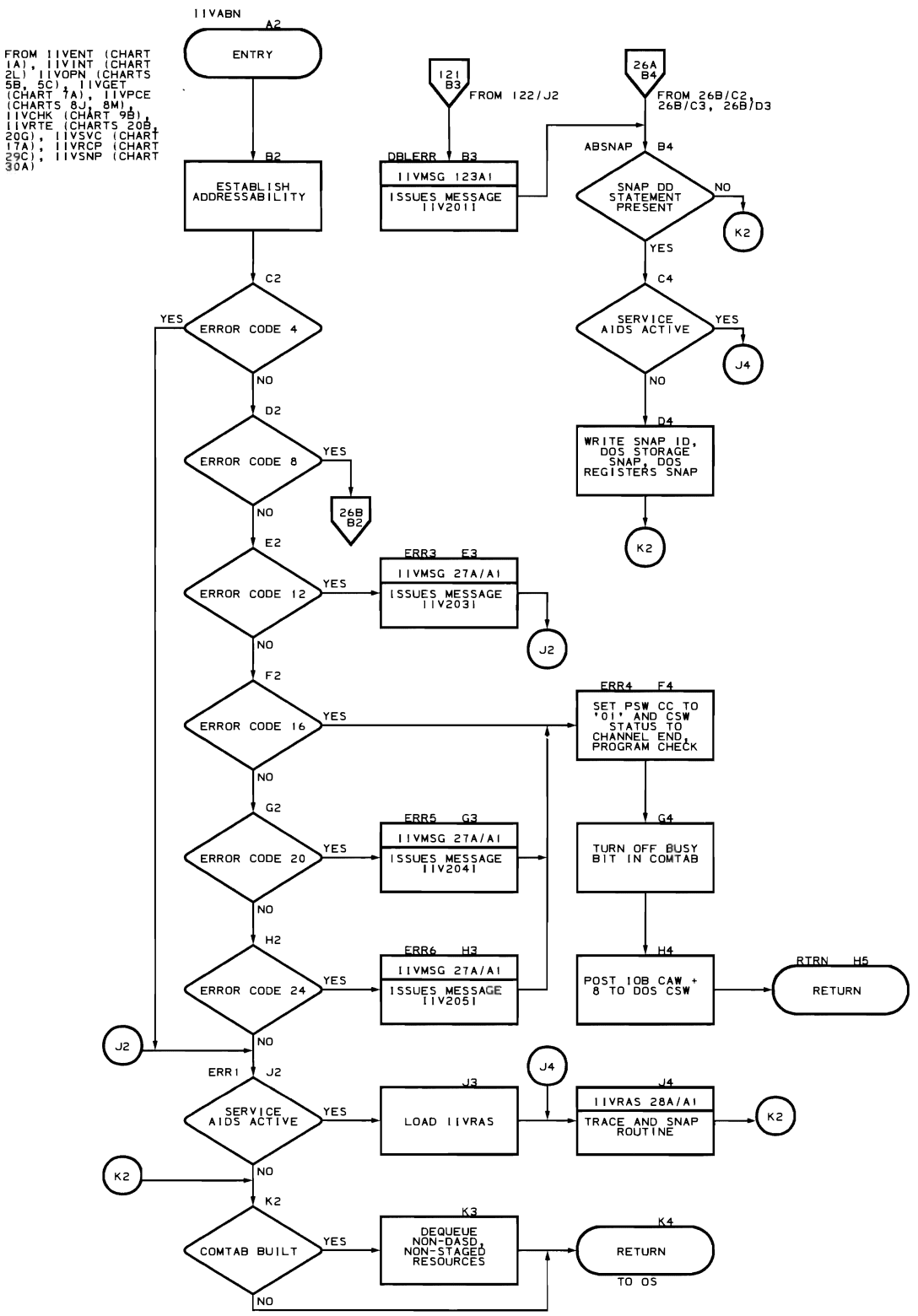
- CSW
- COMTAB
- CCW
- DCB for SNAPs
- IIVCON - Emulator constant area
- Local execution list
- IIVRAS constants

Errors detected: None

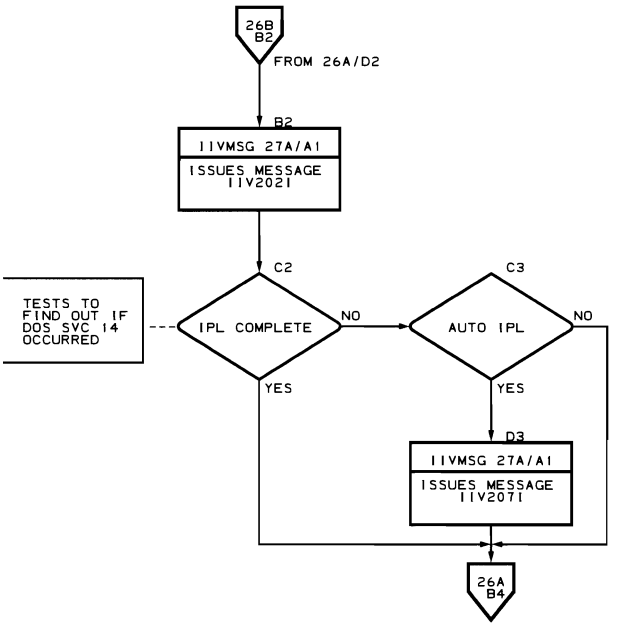
Messages requested:

- IIV202I (IIVPCE detected error)
- IIV203I
- IIV204I (IIVCCW detected error)
- IIV205I (IIVPCE detected error)
- IIV207I

Flowchart 26A. Exit-ABEND Error Routine (IIVABN Part 1 of 2)



Flowchart 26B. Exit-ABEND Error Routine (IIVABN Part 2 of 2)



Message Writer

Message Writer Routine (Flowchart 27A)

Module name: IIVMSG

Entry point name: IIVMSG

Major function: Prints messages using OS WTO or WTOR services. The messages are on behalf of all the Emulator modules.

Entered by:

- IIVINT
- IIVIN2
- IIVPUB
- IIVCHK
- IIVADD
- IIVPRP
- IIVGET
- IIVOPN
- IIVSTG
- IIVRTE
- IIVABN
- IIVPCE
- IIVRCP
- IIVGR2
- IIVDVS
- IIVRCW

Modules called:

- IIVMG1
- IIVMG2
- IIVMG3

Exits to: Caller

OS Macros issued:

- WTO
- LOAD
- SAVE
- WTOR
- DELETE
- RETURN

Input:

- Message code
- Supplemental data, if applicable
- Parameter list with reply address, length, and ECB, if applicable

Output: Message printed to the console

Return codes: None

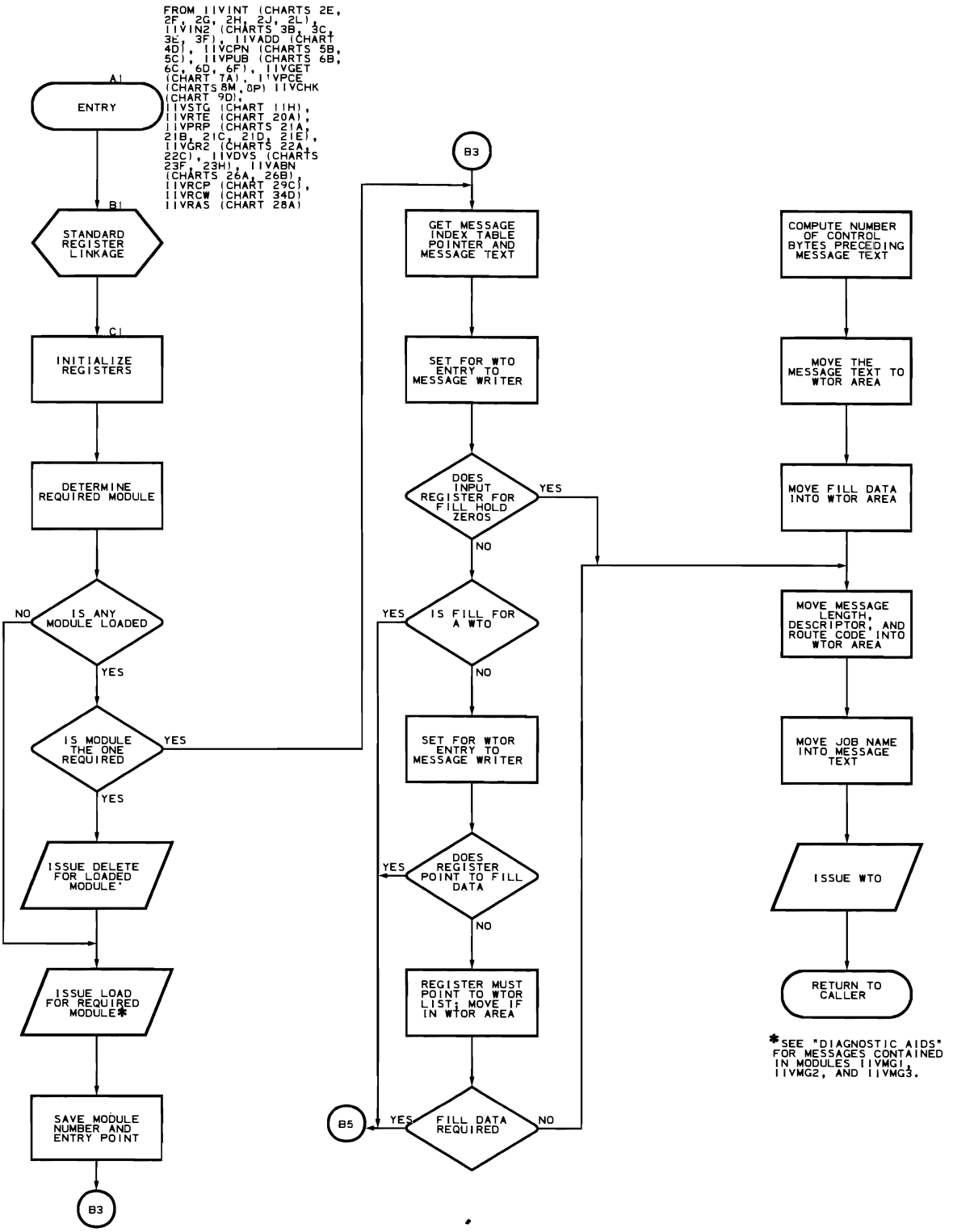
Tables/work areas:

- MSGINDX table in each of the message text modules, IIVMG1, IIVMG2, and IIVMG3
- EMUCONS

Errors detected: None

Messages requested: None

Flowchart 27A. Message Writer Routine (IIVMSG)



Service Aids Initialization Routine (Flowcharts 28A-28B)

Module name: IIVRAS

Entry point names:

- IIVRAS
- IIVRASPC
- IIVRASVC
- IIVRASYN

Major functions:

- Initializes fields in IIVRCN
- Establishes addressability after program checks, supervisor calls, and asynchronous interruptions during local execution mode

Entered by:

- IIVIN2
- IIVABN
- IIVPRP
- Hardware (IIVRASVC, IIVRASPC, and IIVRASYN addresses placed in SVC, PC, and ASYNC interruption fields respectively in the LEX list)

Modules called:

- IIVRCP
- Asynchronous user exit routine (IIVRASnn)
- IIVSNP
- IIVMSG

Exits to:

- Caller
- IIVPCI
- IIVSCI
- IIVRTE

OS macros issued:

- SAVE
- DELETE
- RETURN
- TIME
- LOAD
- OPEN

Input:

- Fields in IIVRCN (RASCONS)
- Register 0 - 0 = caller from IIVPRP or IIVIN2
  4 = snap out trace table, caller from IIVABN
  8 = snap DOS hard wait, caller from IIVABN
- Register 1 - contains option in effect if register 0 contains 8

Output: Initializes fields in IIVRCN

Return codes: None

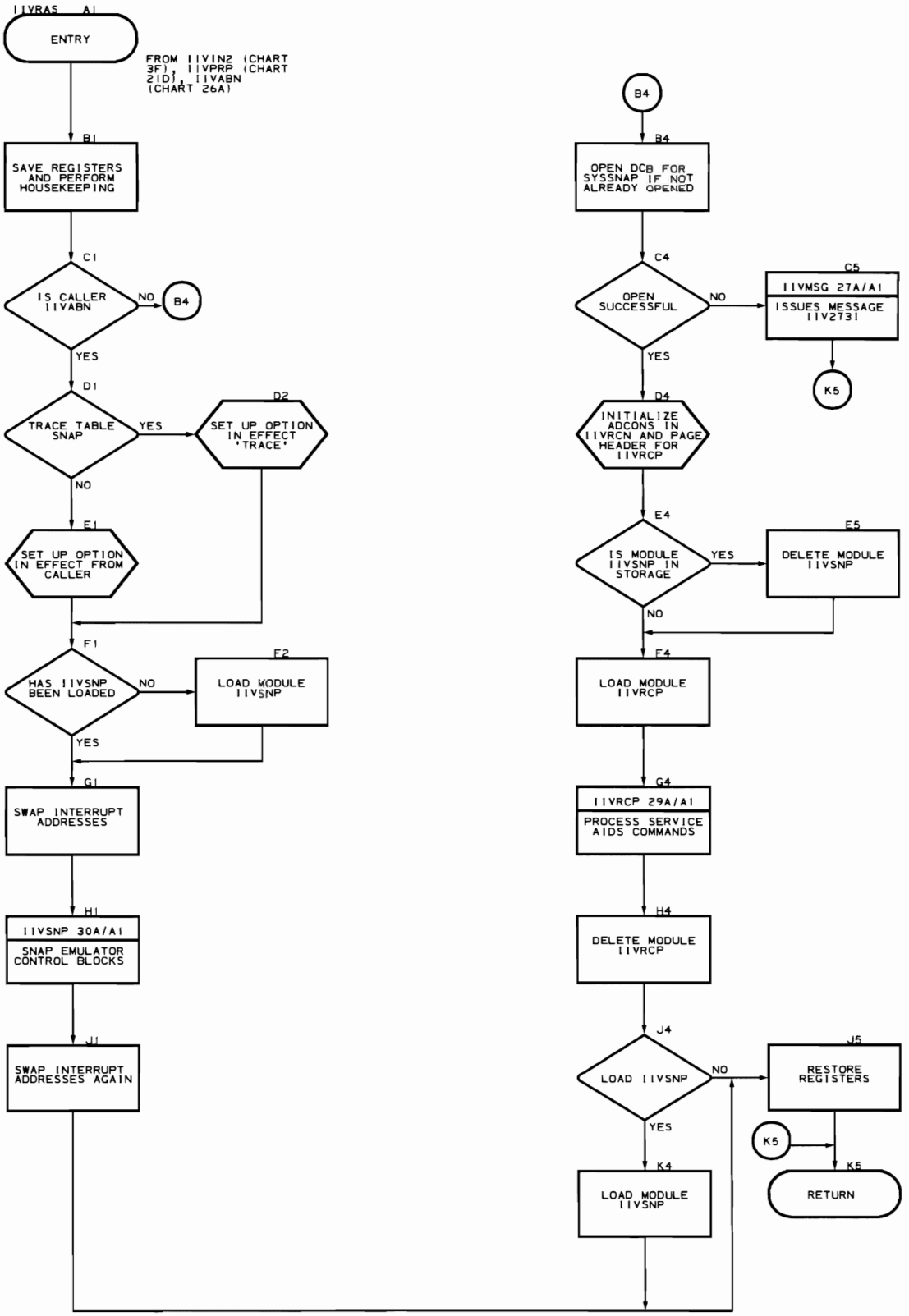
Tables/work areas:

- IIVRCN (RASCONS)
- Local execution list
- IIVCON (EMUCONS)

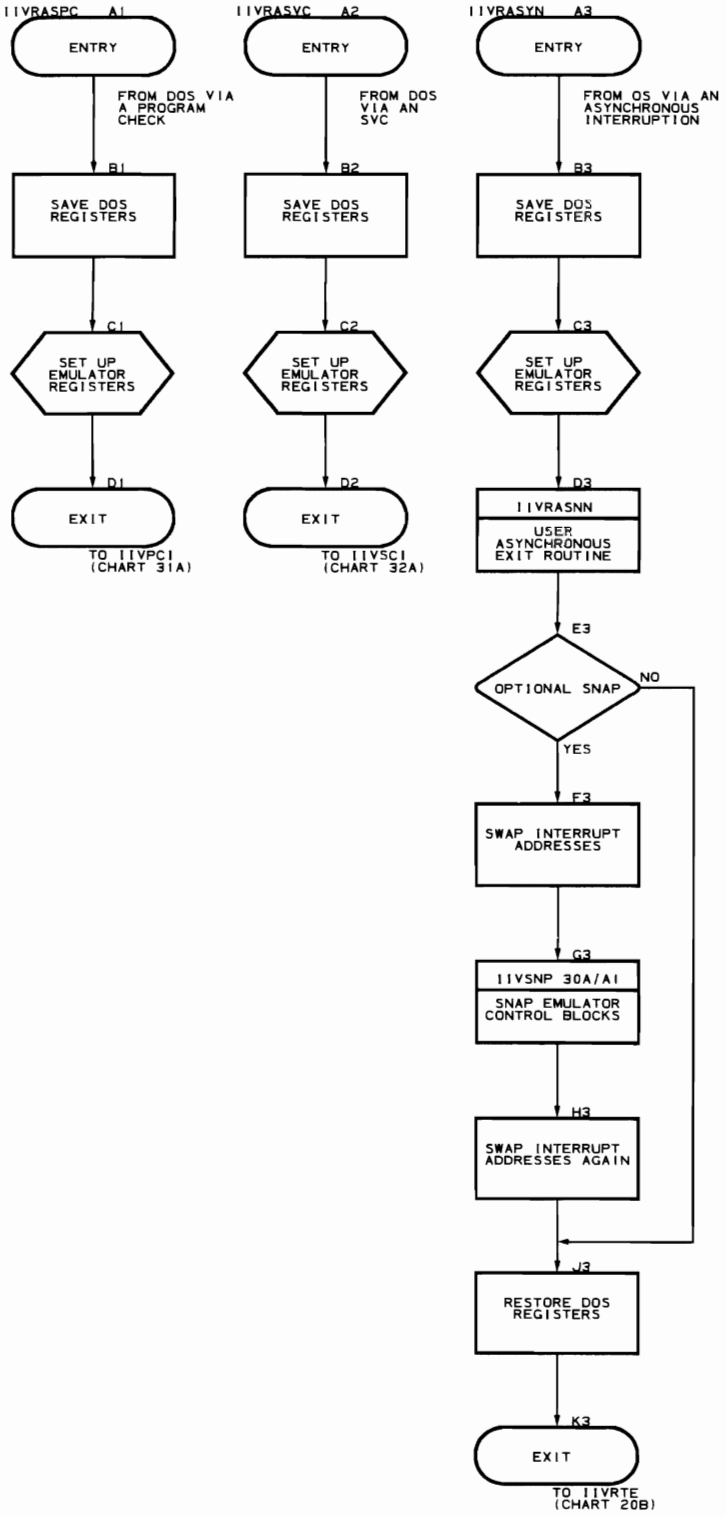
Errors detected: None

Messages requested: IIV273I

Flowchart 28A. Service Aids Initialization Routine (IIVRAS Part 1 of 2)



Flowchart 28B. IIVRASPC, IIVRASVC, and IIVRASYN Subroutines (IIVRAS Part 2 of 2)



Command Processor Routine (Flowcharts 29A-29P)

Module name: IIVRCP

Entry point name: IIVRCP

Major functions:

- Reads a debug statement from card input or console reply
- Checks the validity of the statement
- Sets adequate indicators in RASCONS
- Loads service aids processor IIVACI, IIVPCI, IIVSCI

Entered by: IIVRAS

Modules called: IIVMSG

Exits to:

- IIVABN
- Caller

OS macros issued:

- OPEN
- CHECK
- DELETE
- FREEMAIN
- WRITE
- LOAD
- GETMAIN
- GET
- CLOSE
- WAIT
- DCBD
- DCB
- SAVE

Input:

- Pointer to local execution list
- Pointer to Emulator constants
- Service aids constants

Output: Indicators and switches in RASCONS

Return codes: In register 15

Tables/work areas:

- Local execution list
- EMUCONS
- RASCONS
- DCB for SYSDEBUG data set

Errors detected: Syntax errors in debug statement

Messages requested or issued:

- IIV270I (on console)
- IIV271I (on console)
- IIV274I (on console)
- IIV276I

Dictionaries for IIVRCP. The CMDDICT (Figure 27) lists each command, the routine that initializes each command, the routine that resets each command after an error is detected, and the associated dictionary to handle the parameters or keyword parameters of each command.

The SNPDICT (Figure 27) lists each parameter or keyword parameter (under heading 'Name of Keyword') of the SNAP command and the name and chart number of the routine to process each parameter.

The TRCDICT (Figure 27) lists each parameter or keyword parameter (under heading 'Name of Keyword') of the TRACE command and the name and chart number of the routine to process each parameter.

The STODICT (Figure 27) lists each parameter or keyword parameter (under heading 'Name of Keyword') of the STORAGE command and the name and chart number of the routine to process each parameter.

The EXIDICT (Figure 27) lists each keyword parameter (under heading 'Name of Keyword') of the EXIT command and the name and chart number of the routine to process each keyword parameter.

The DIAGDICT (Figure 27) lists the keyword parameter of the DIAG command and the name and chart number of the routine to process the parameter.

CMDDICT - Command Dictionary

Name of Command	Routine to Initialize		Routine to Reset		Associated Keyword Dictionary
	Name	Chart	Name	Chart	
DIAG	DIAGINIT	29L/A1	DIAGINV	29L/A1	DIAGDICT
END	ENDINIT	29N/A1			
EXIT	EXIINIT	29M/A1	EXIINV	29M/A1	EXIDICT
SNAP	SNPINIT	29G/A1	SNPINV	29G/A2	SNPDICT
STORAGE	STOINIT	29K/A1	STOINV	29K/A1	STODICT
TRACE	TRCINIT	29E/A1	TRCINV	29E/A2	TRCDICT

SNPDICT - Snap Dictionary

Name of Keyword	Routine to Process		
	Name	Chart	
ALL	SNPALL	29J/G3	
ATTN	SNPATTN	29H/D4	
COMP*	SNPCOMP	29H/A1	note1
CUU*	SNPCUU	29J/A1	
EXT	SNPEXT	29H/A5	
HIO	SNPHIO	29J/A5	
INT	SNPINT	29H/D5	
IO	SNPIO	29J/G2	
ISK	SNPISK	29J/D3	
LPSW	SNPLPSW	29J/D5	
PC*	SNPPC	29H/G4	
PSW*	SNPPSW	29G/C2	
SIO	SNPSIO	29J/A2	
SSK	SNPSSK	29J/D4	
SSM	SNPSSM	29J/D2	
SVC*	SNPSVC	29G/A3	note2
TCH	SNPTCH	29J/A4	
TIMER	SNPTIMER	29H/A4	
TIO	SNPTIO	29J/A3	

note1 If COMP=A111111=hhhhhhhh is specified, the routine name to process hhhhhhhh is SNPCPSUB. (See chart 29H/A3.)

note2 If SVC=PHASE=(modname1,modname2) is specified, the routine name to process modname1 cr modname2 is SNPSUBOP. (See chart 29G/H5.)

* This indicates a keyword parameter. All others are called parameters.

Figure 27 (Part 1 of 2). Command Processor Routine Dictionaries

TRCDICT - Trace Dictionary

Name of Keyword	Routine to Process	
	Name	Chart
ALL	TRCALL	29F/H5
ATTN	TRCATTN	29F/G3
CUU*	TRCCUU	29F/A2
EXT	TRCEXT	29F/A4
HIO	TRCHIO	29E/C5
INT	TRCINT	29F/D3
IO	TRCIO	29F/A1
ISK	TRCISK	29E/F3
LPSW	TRCLPSW	29E/F5
NOWRAP	TRCNWRAP	29F/E1
WRAP	TRCWRAD	29F/H1
NUMBER*	TRCNUMBR	29F/D4
SIO	TRCSIO	29E/C2
SSK	TRCSSK	29E/F4
SSM	TRCSSM	29E/F2
SVC	TRCSVC	29F/A5
TCH	TRCTCH	29E/C4
TIMER	TRCTIMER	29F/A3
TIO	TRCTIO	29E/C3
WRAP	TRCWRAP	29F/H1

STODICT - Storage Dictionary

Name of Keyword	Routine to Process	
	Name	Chart
ALL	STOALL	29K/A5
DOS	STODOS	29K/E2
EMBLKS	STOEMBLK	29K/A3
EMU	STOEMU	29K/A4
NODOS	STONODOS	29K/E3
NUMBER*	STONUMBR	29K/E4

EXIDICT - Exit Dictionary

Name of Keyword	Routine to Process	
	Name	Chart
AS*	EXIAS	29M/A5
LEX*	EXILEX	29M/A4
PC*	EXIPC	29M/A3
SVC*	EXISVC	29M/A2

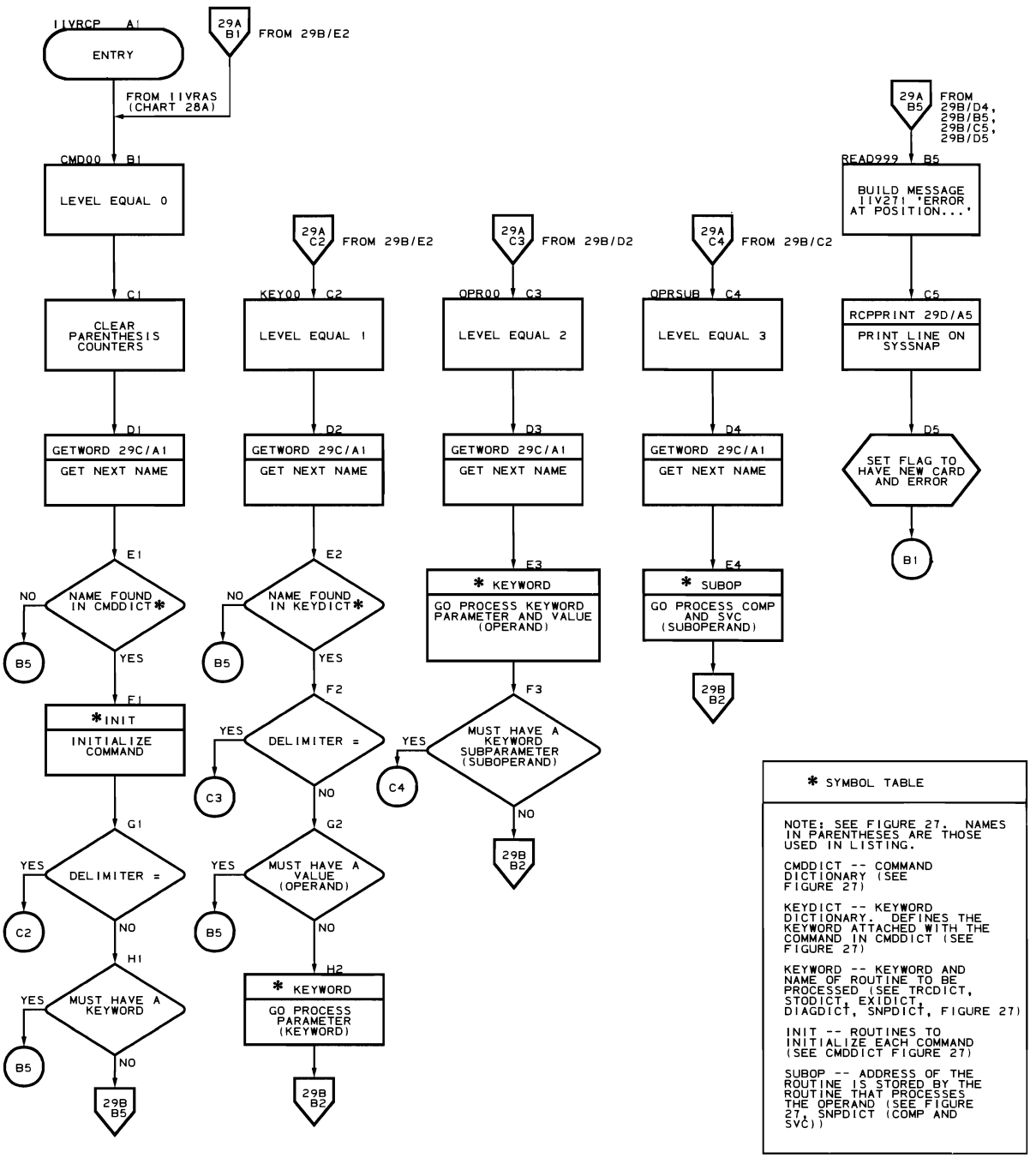
DIAGDICT -- Diagnostic Dictionary

Name of Keyword	Routine to Process	
	Name	Chart
CCWCHK	DIA3CCW	29L/A4

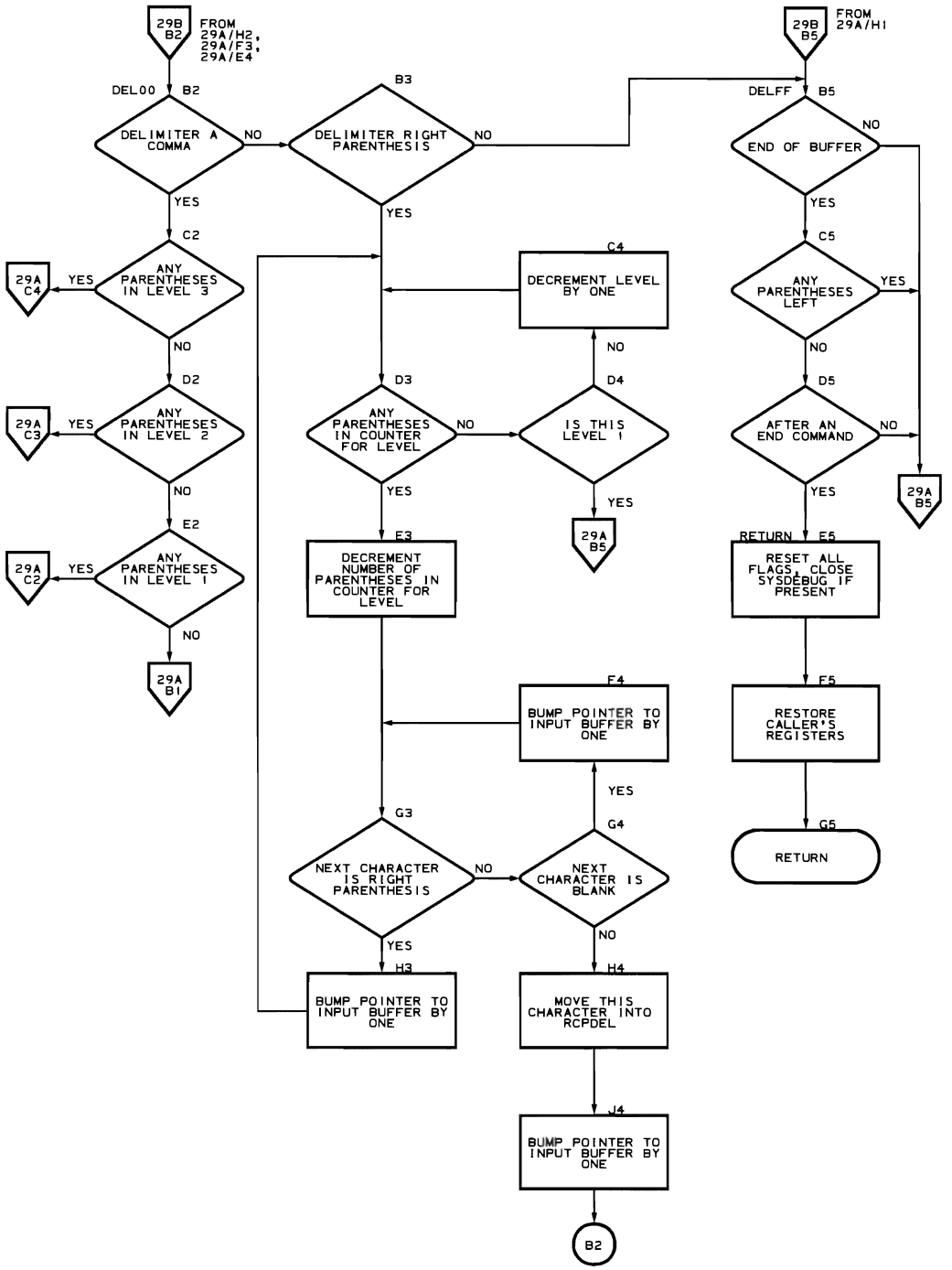
* This indicates a keyword parameter. All others are called parameters.

Figure 27 (Part 2 of 2). Command Processor Routine Dictionaries

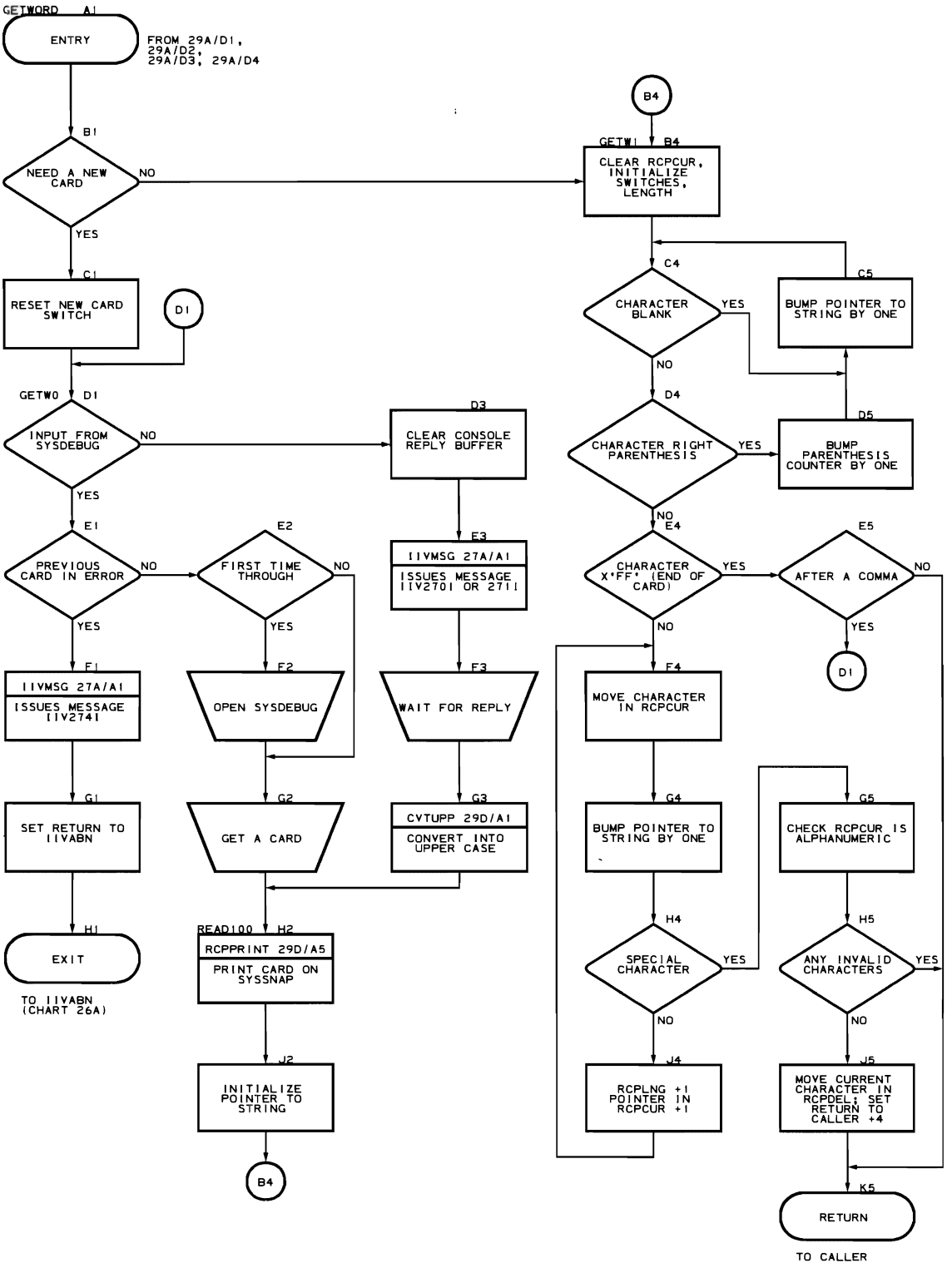
Flowchart 29A. Command Processor Routine (IIVRCP Part 1 of 14)



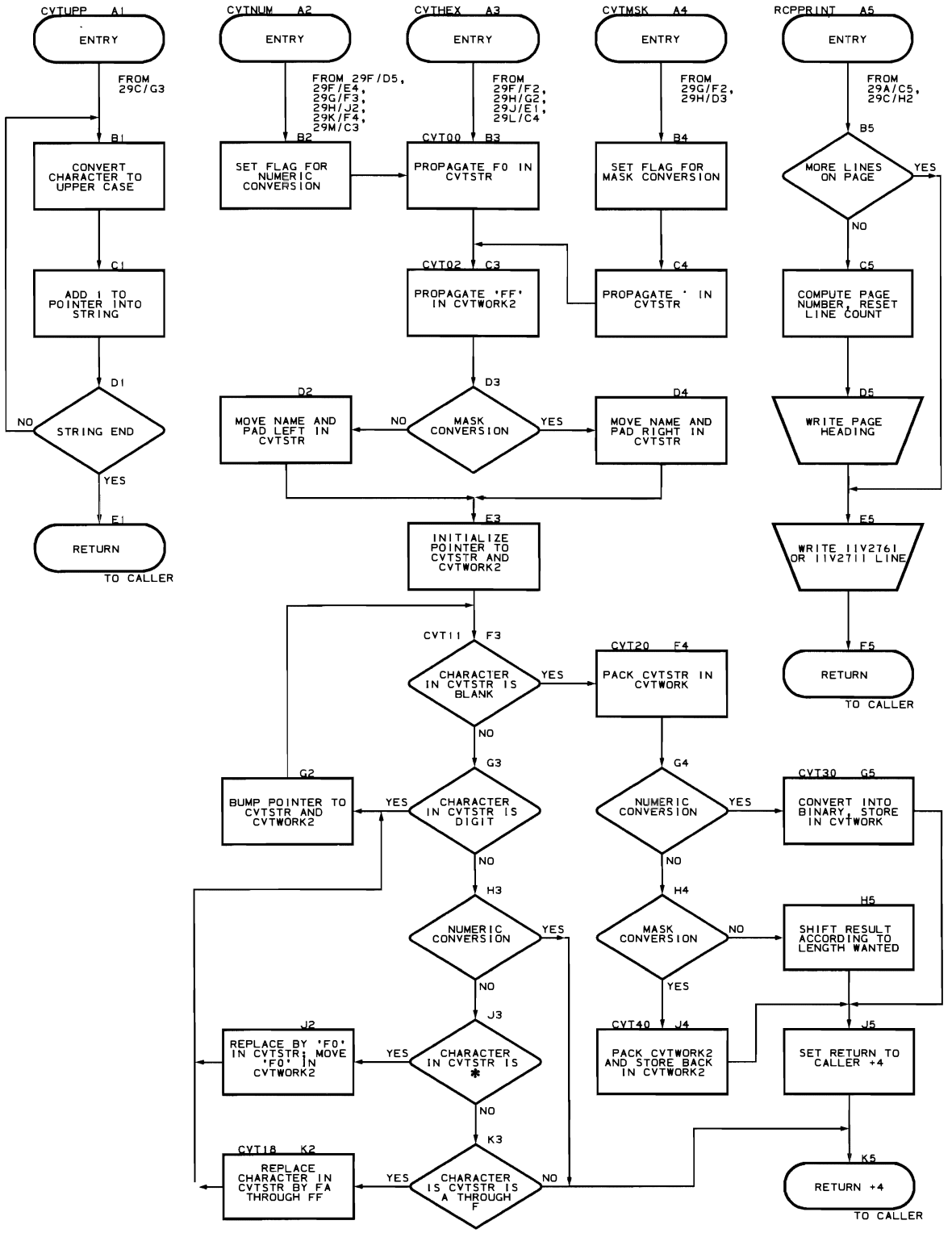
Flowchart 29B. Command Processor Routine (IIVRCP Part 2 of 14)



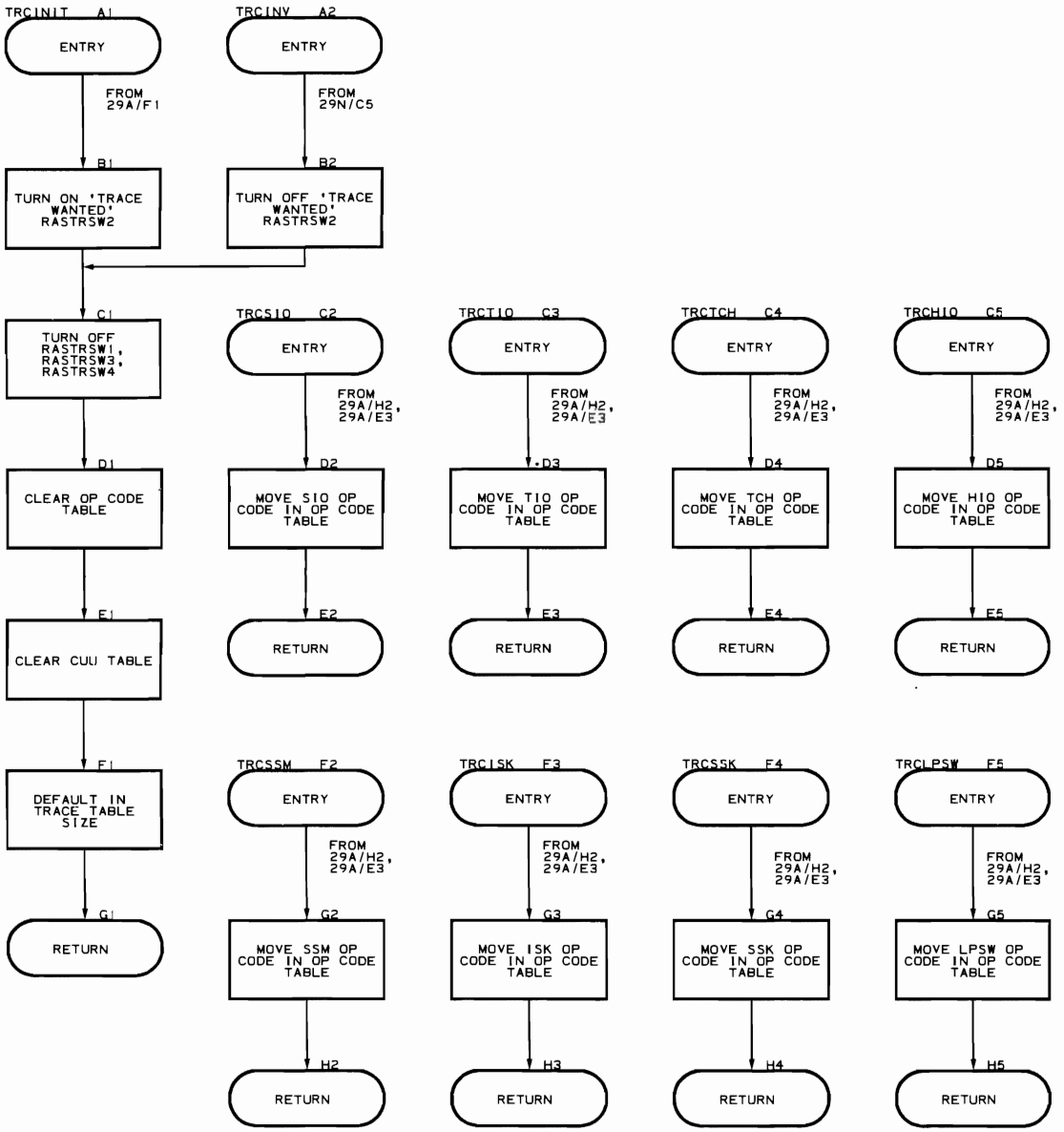
Flowchart 29C. GETWORD Subroutine (IIVRCP Part 3 of 14)



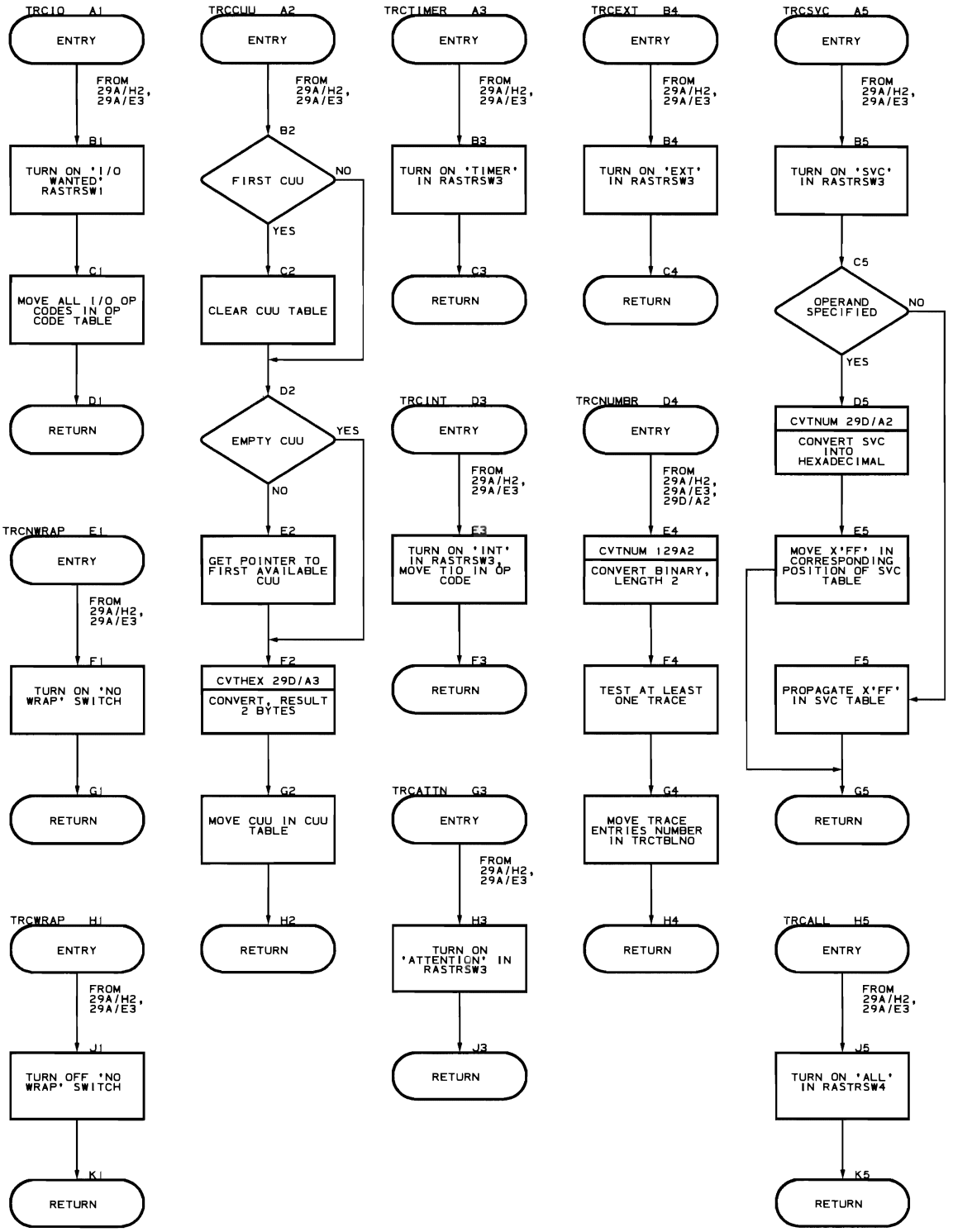
Flowchart 29D. CVT and RCPPRINT Subroutines (IIVRCP Part 4 of 14)



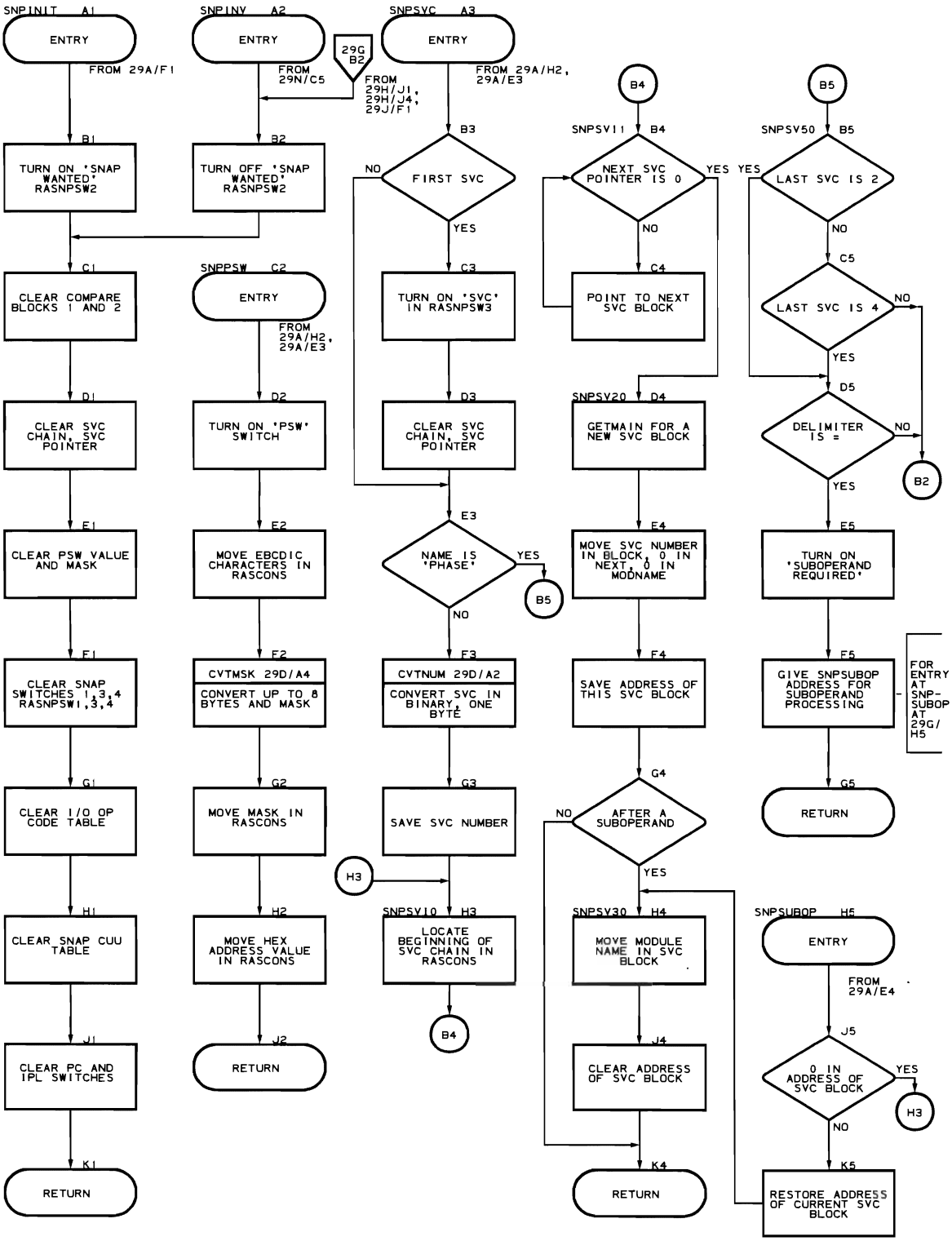
Flowchart 29E. Trace Subroutines (IIVRCP Part 5 of 14)



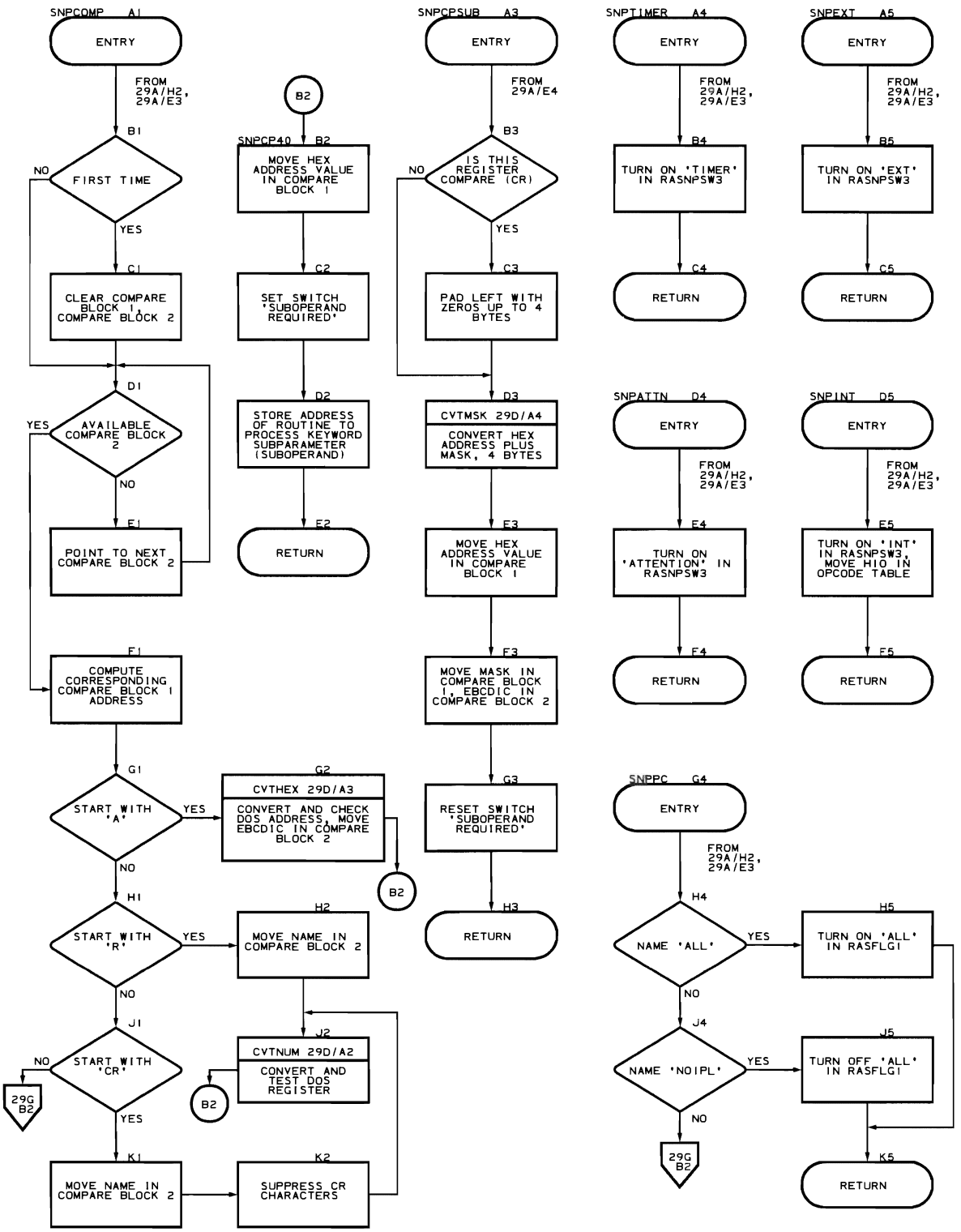
Flowchart 29F. Trace Subroutines (IIVRCP Part 6 of 14)



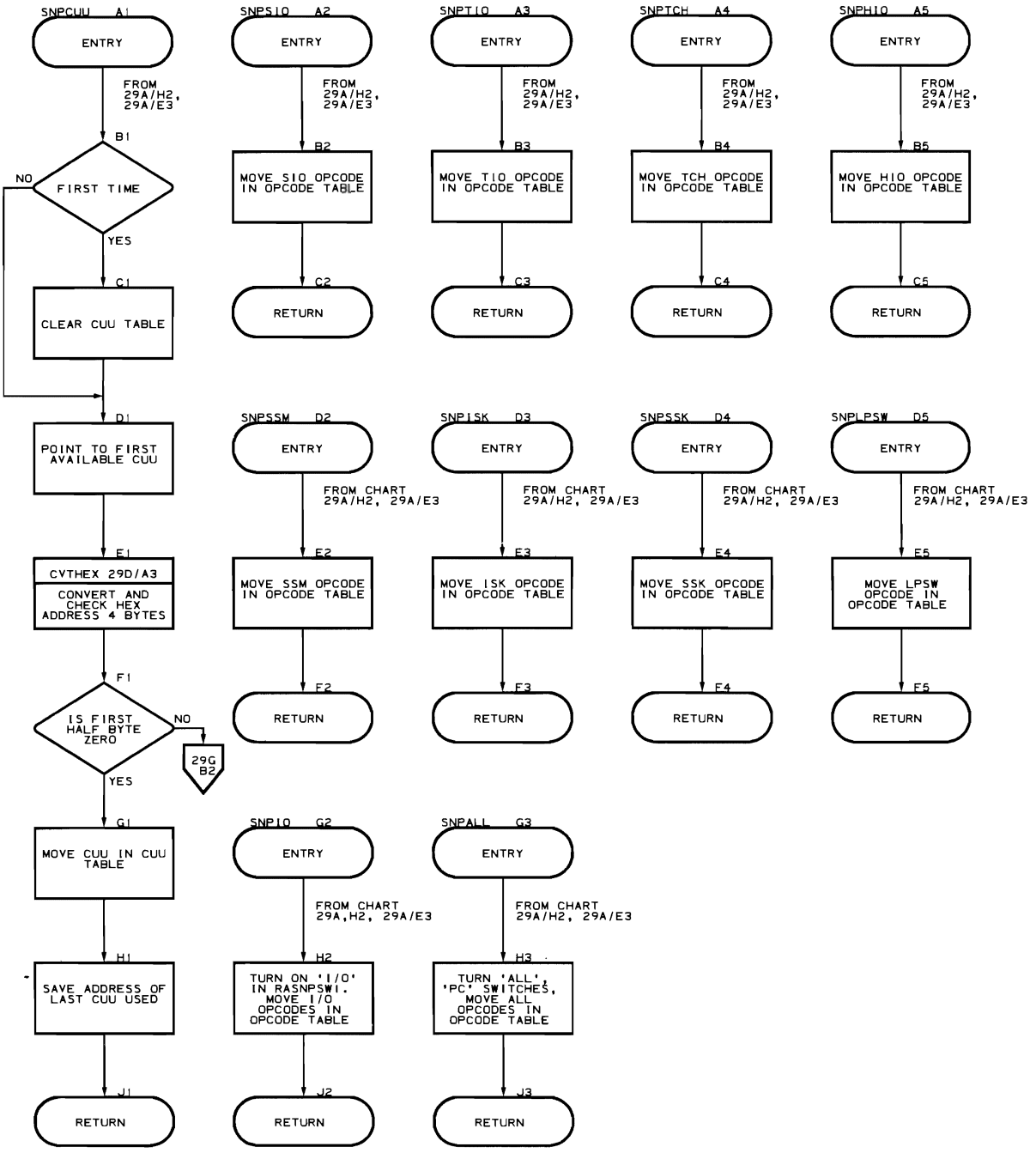
Flowchart 29G. Snap Subroutines (IIVRCP Part 7 of 14)



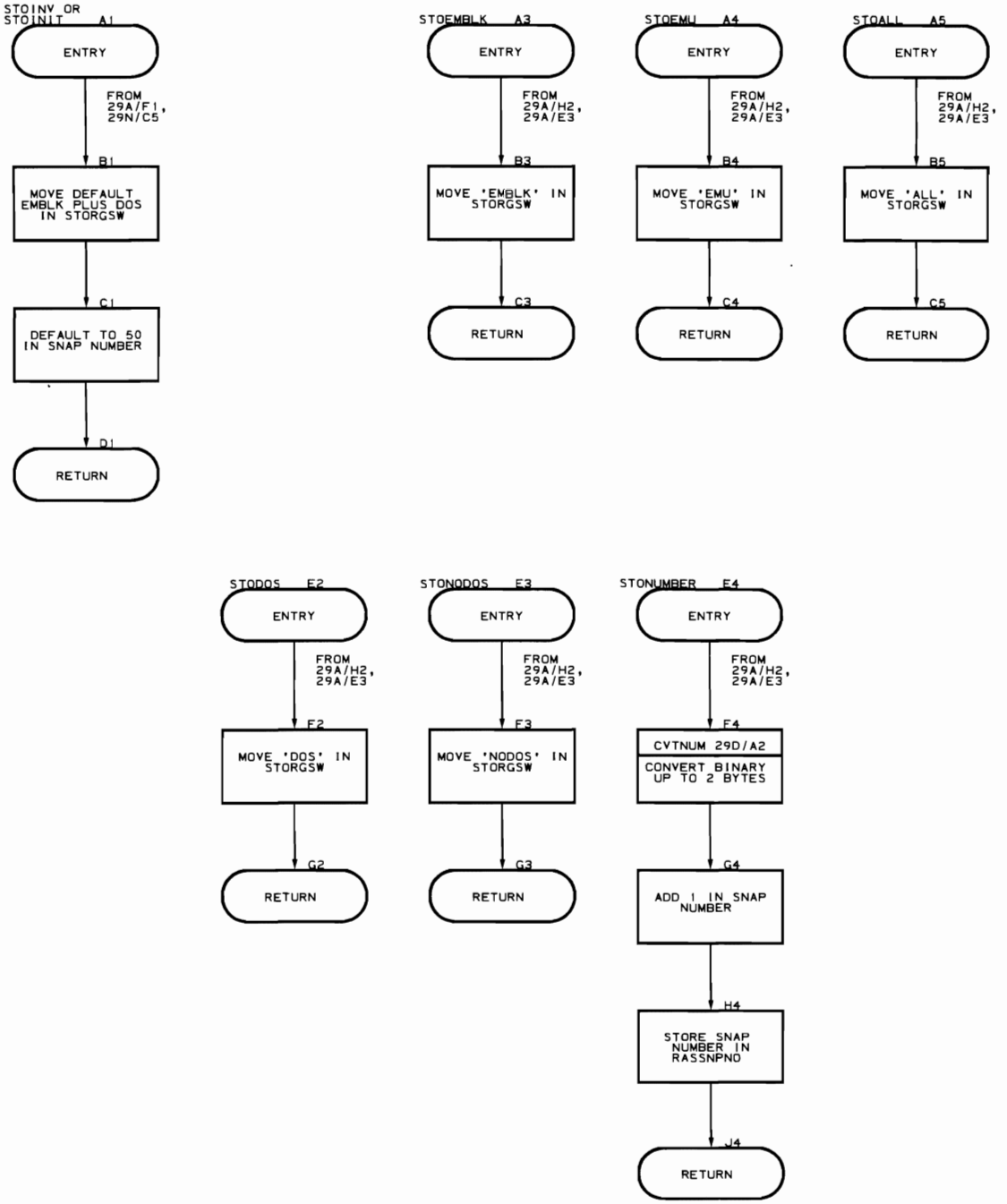
Flowchart 29H. Snap Subroutines (IIVRCP Part 8 of 14)



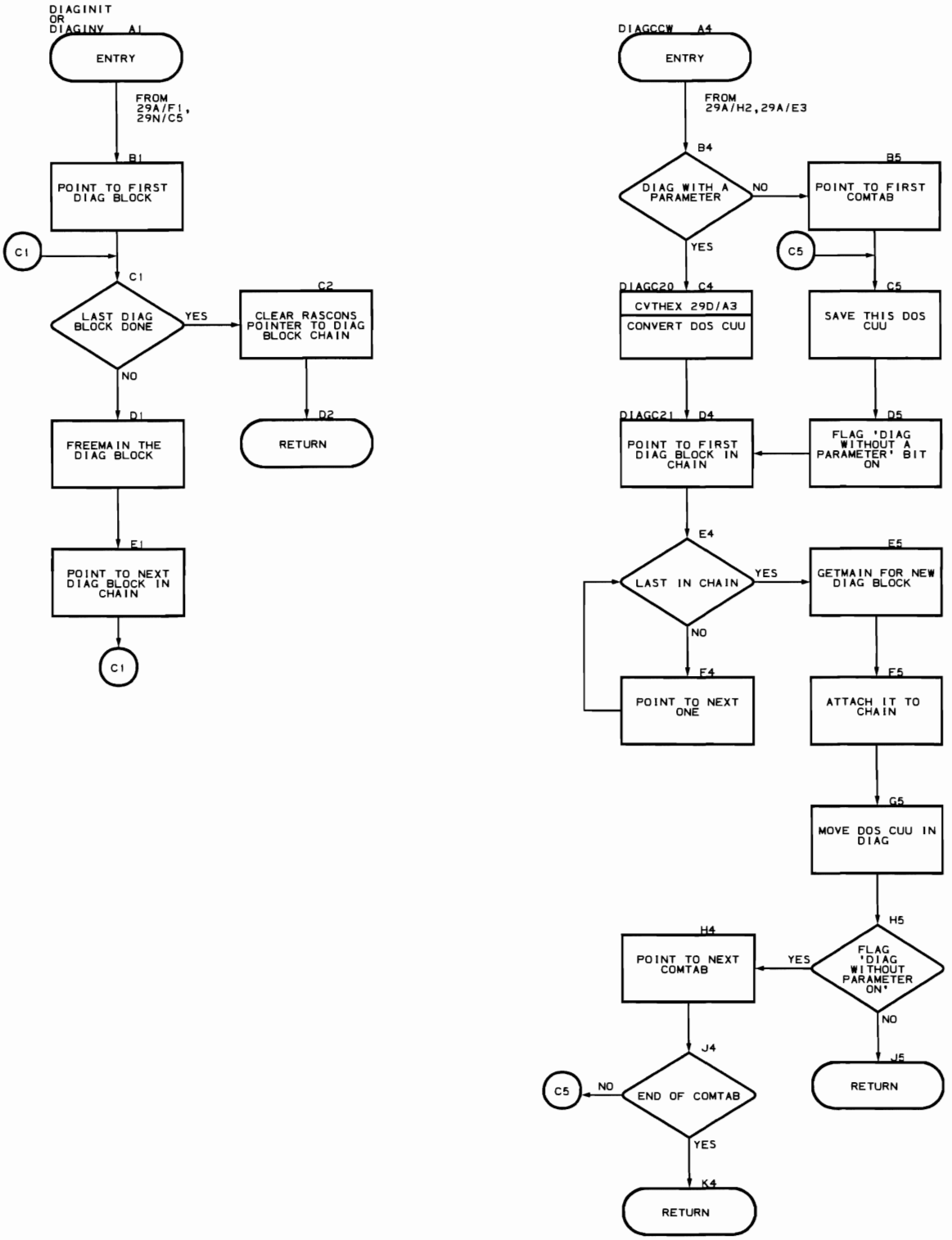
Flowchart 29J. Snap Subroutines (IIVRCP Part 9 of 14)



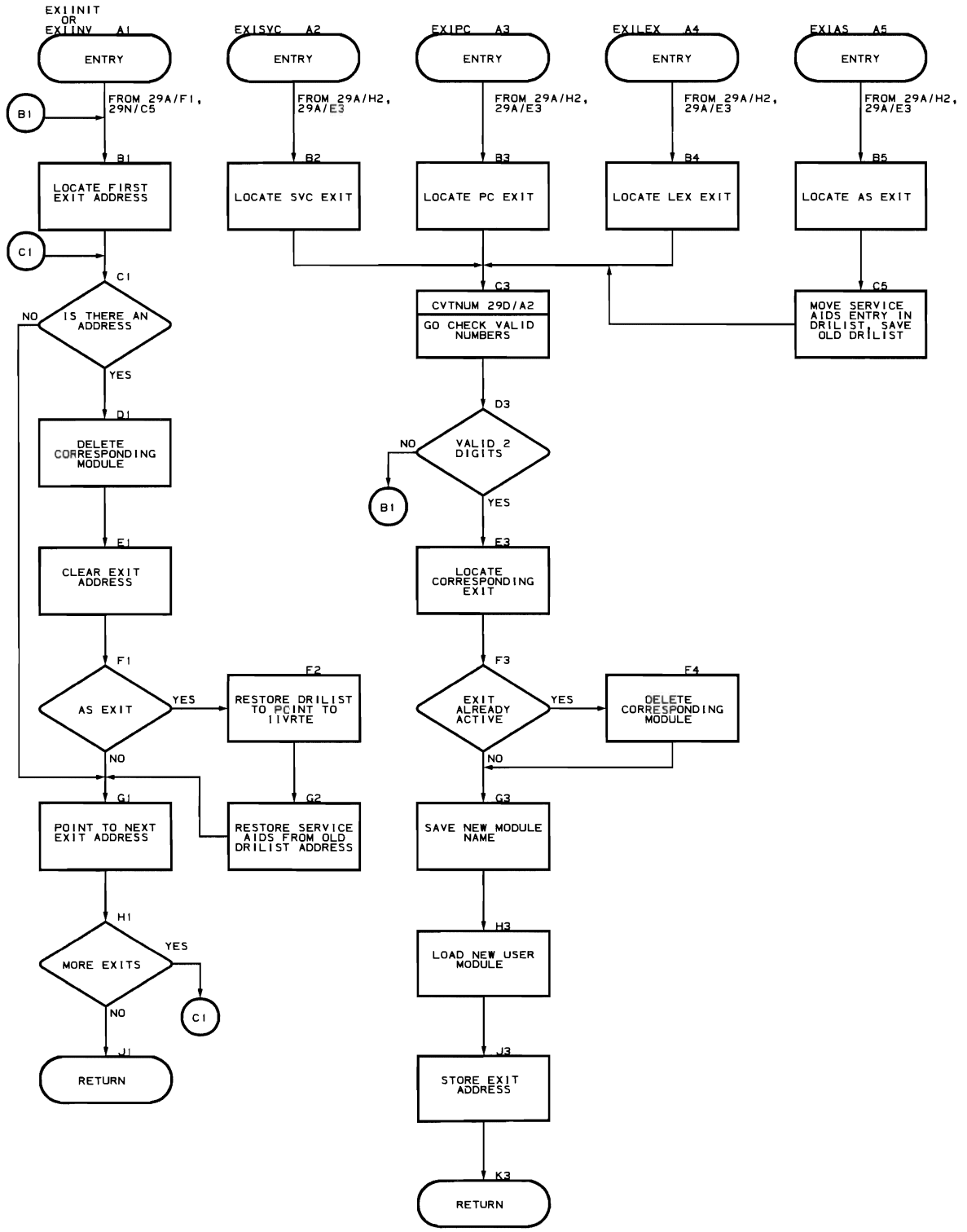
Flowchart 29K. Storage Subroutines (IIVRCP Part 10 of 12)



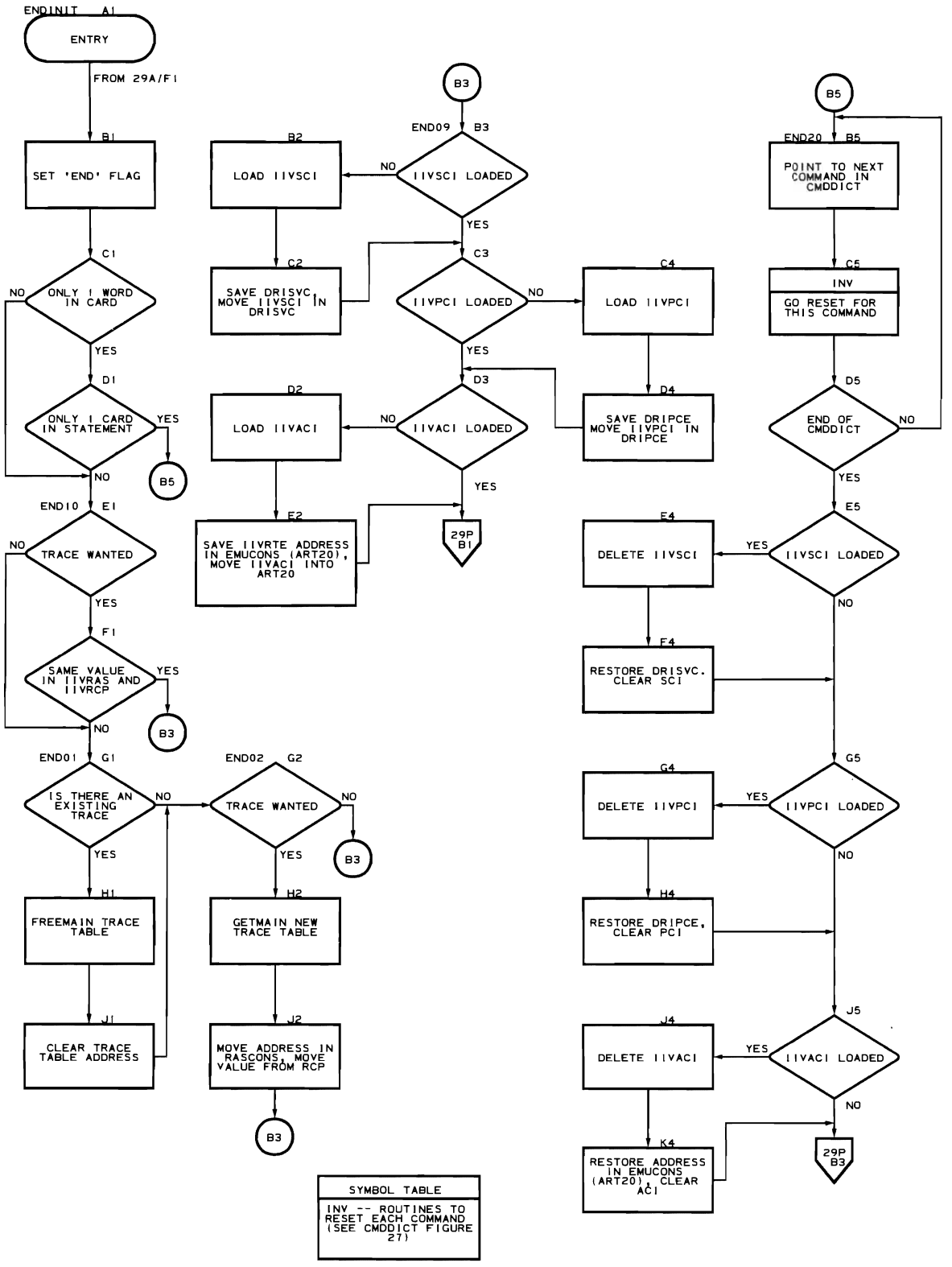
Flowchart 29L. Diagnostic Subroutines (IIVRCP Part 11 of 14)



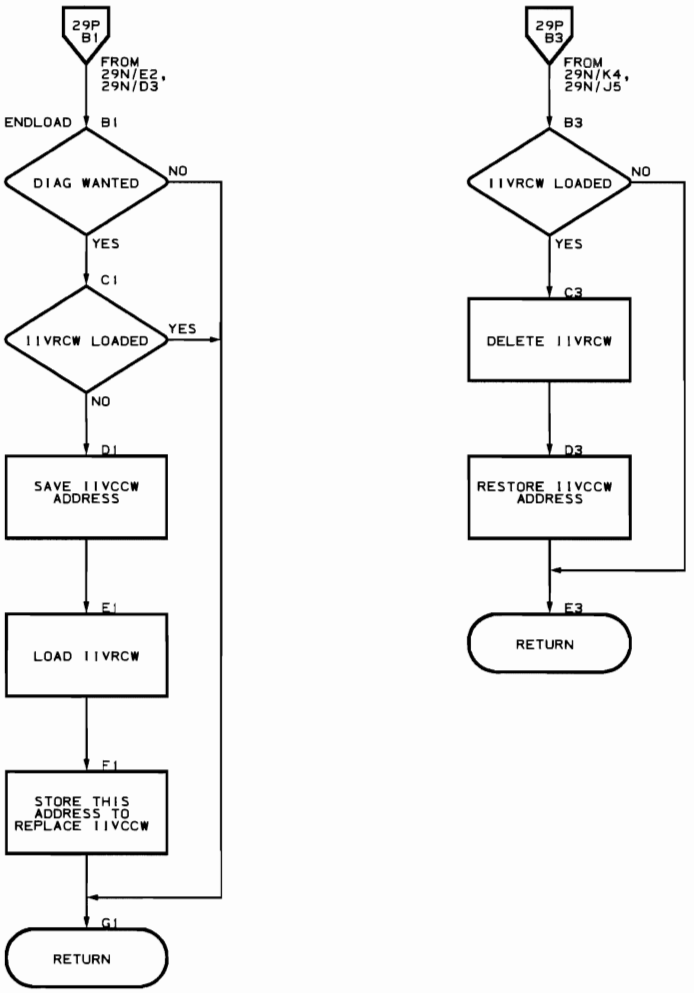
Flowchart 29M. Exit Subroutines (IIVRCP Part 12 of 14)



Flowchart 29N. End Subroutine (IIVRCP Part 13 of 14)



Flowchart 29P. End Subroutine (IIVRCP Part 14 of 14)



Snap Dump and Trace Formatting Routine (Flowcharts 30A-30K)

Module name: IIVSNP

Entry point name: IIVSNP

Major functions:

- Formats Emulator control blocks, trace table, DOS main storage
- Issues OS snap macro (PDATA=ALL or SDATA=ALL)

Entered by:

- IIVRAS
- IIVACI
- IIVPCI
- IIVSCI

Modules called: None

Exits to:

- Caller
- IIVABN

OS macros issued:

- SAVE
- RETURN
- TIME
- SNAP
- WRITE
- CHECK
- DCBD

Input:

- Register 1 - option in effect message preceded by a 2-byte field (length minus one)
- Register 0 - nonzero indicates trace table snap only
- DOS storage
- RASCONS
- Emulator control blocks

Output: Formatted snap dump

Return codes: None

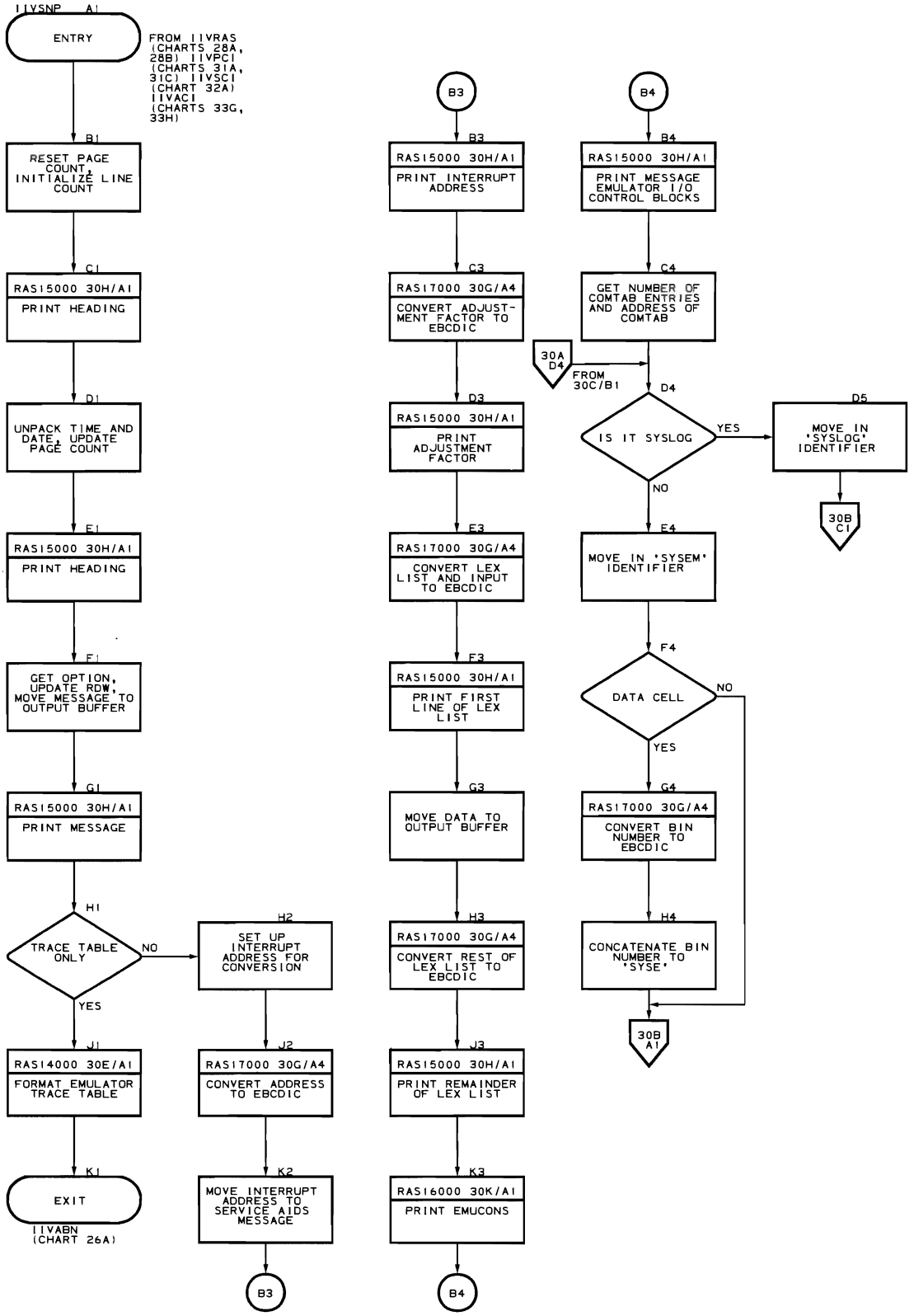
Tables/work areas:

- IIVCON (EMUCONS)
- IIVRCN (RASCONS)
- Local execution list
- DOS storage
- COMTAB

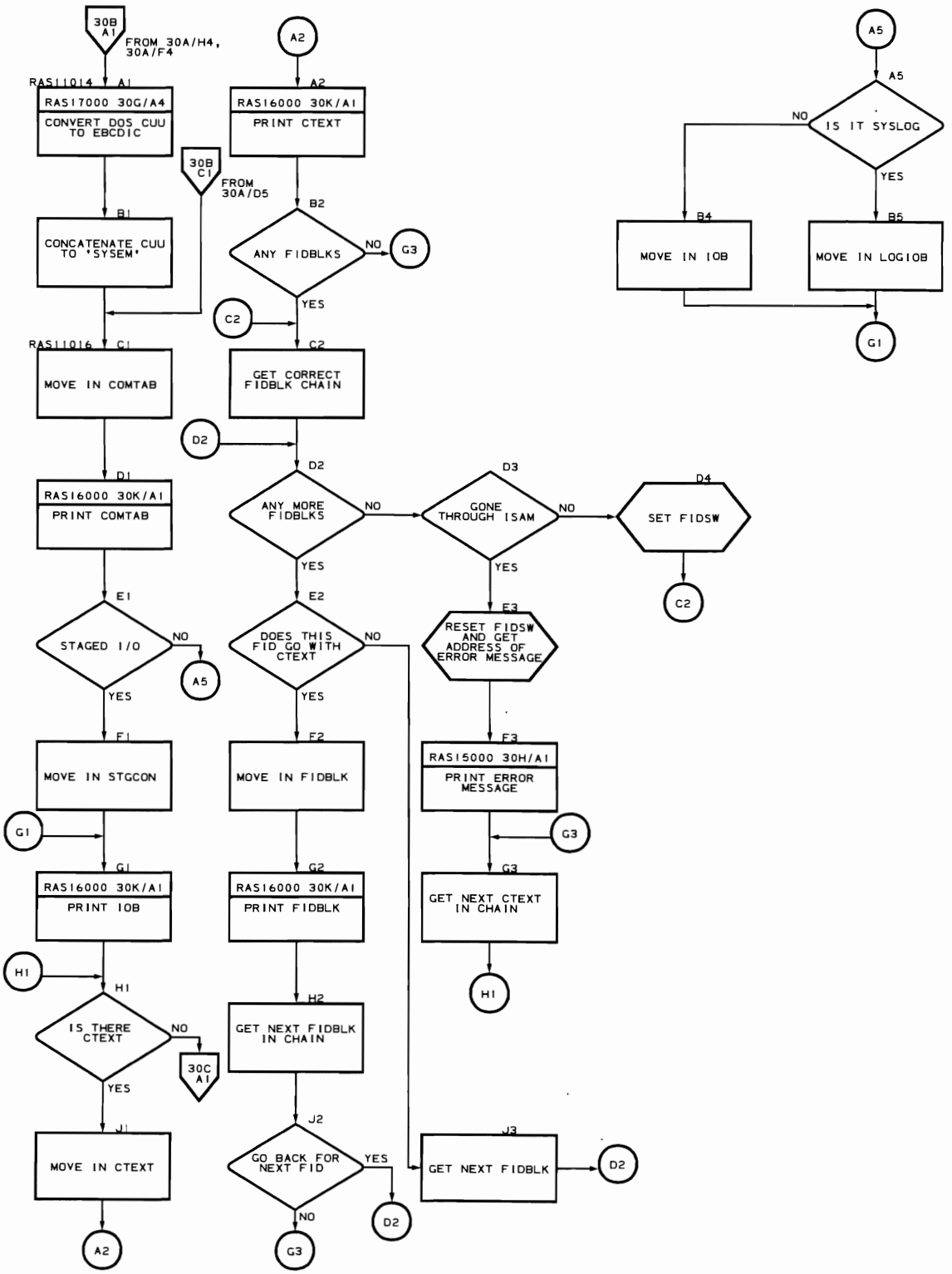
Errors detected: Invalid FIDBLK chain while formatting Emulator I/O control blocks

Messages requested: None

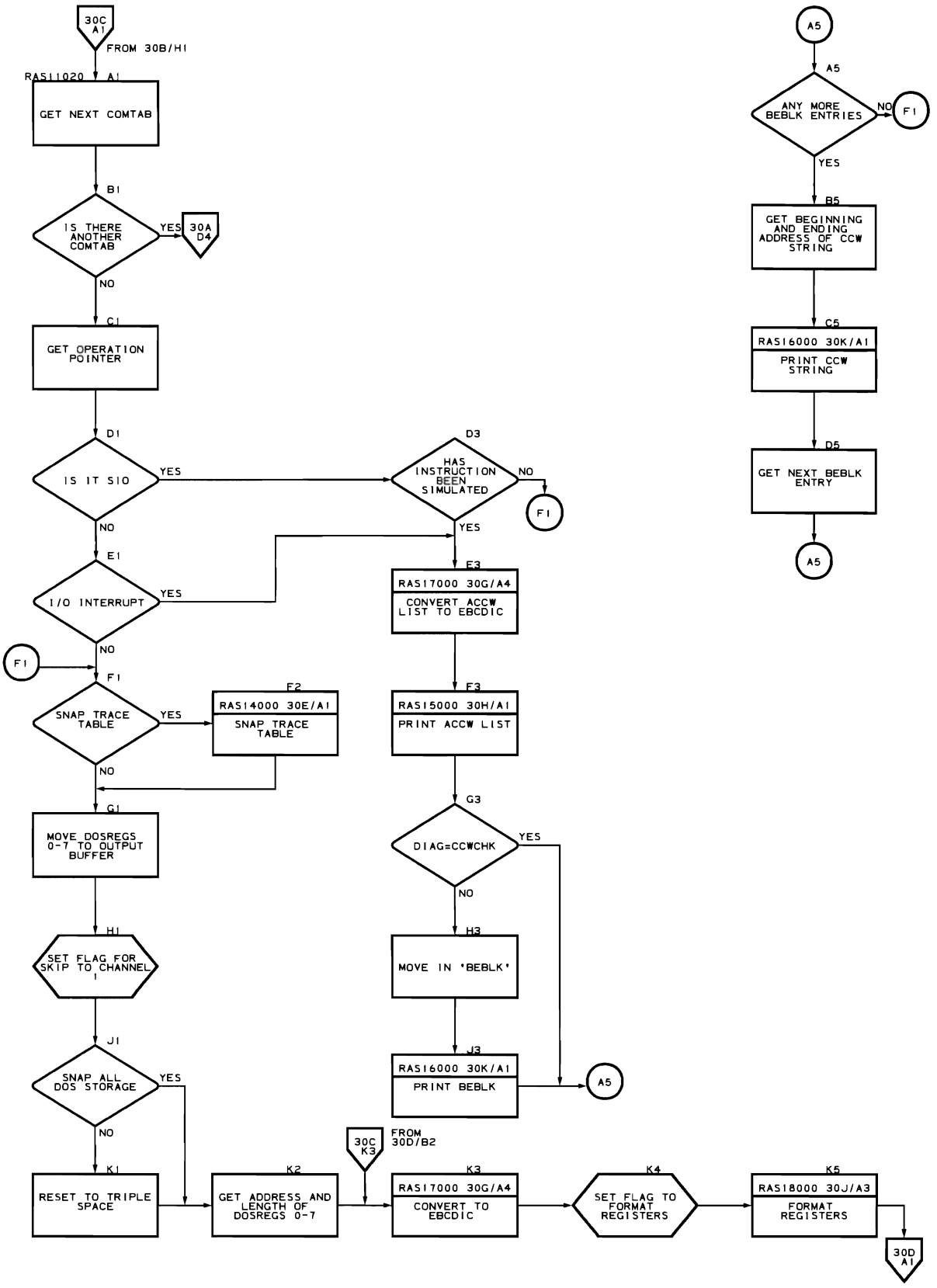
Flowchart 30A. Snap Dump and Trace Formatting Routine (IIVSNP Part 1 of 10)



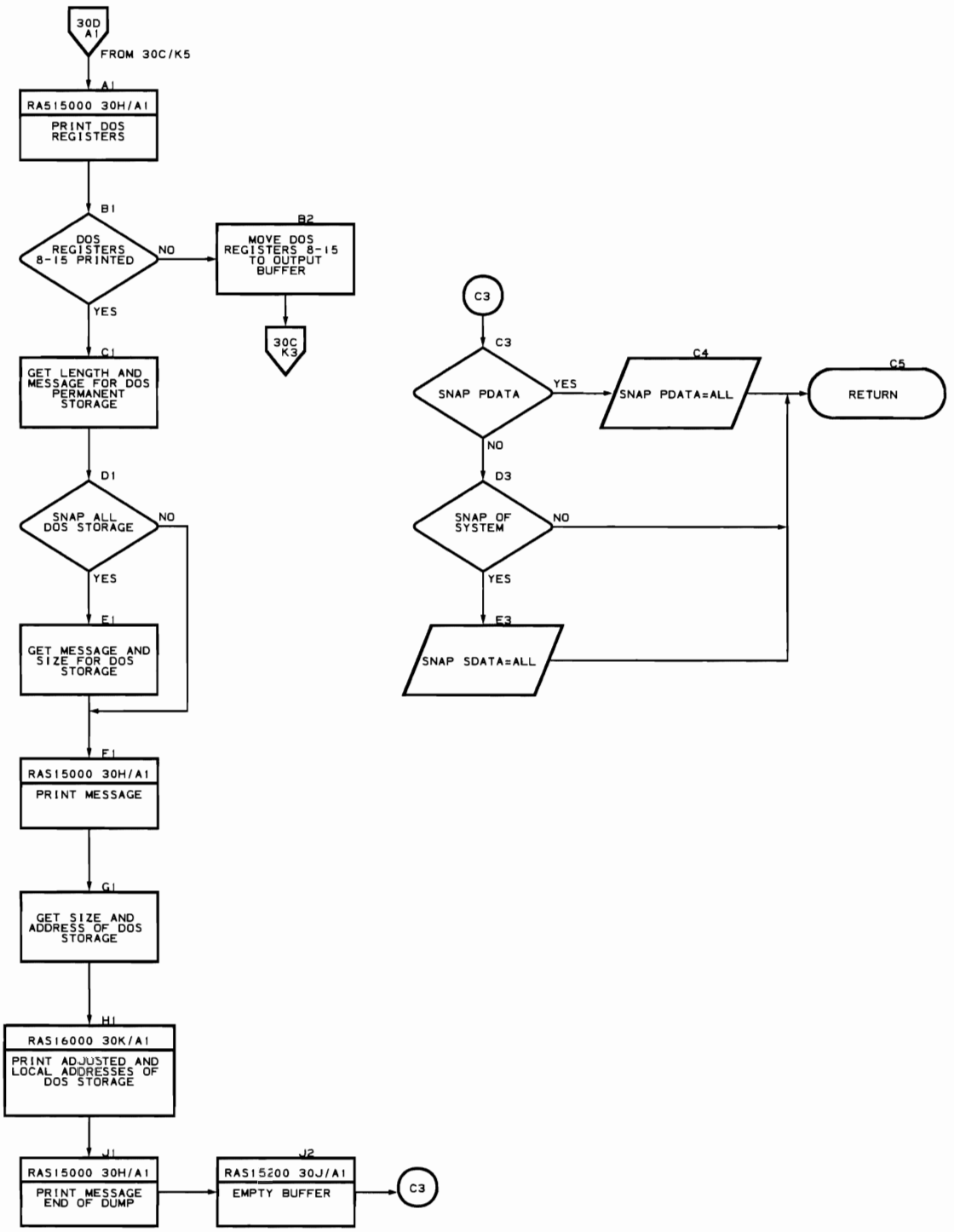
Flowchart 30B. Snap Dump and Trace Formatting Routine (IIVSNP Part 2 of 10)



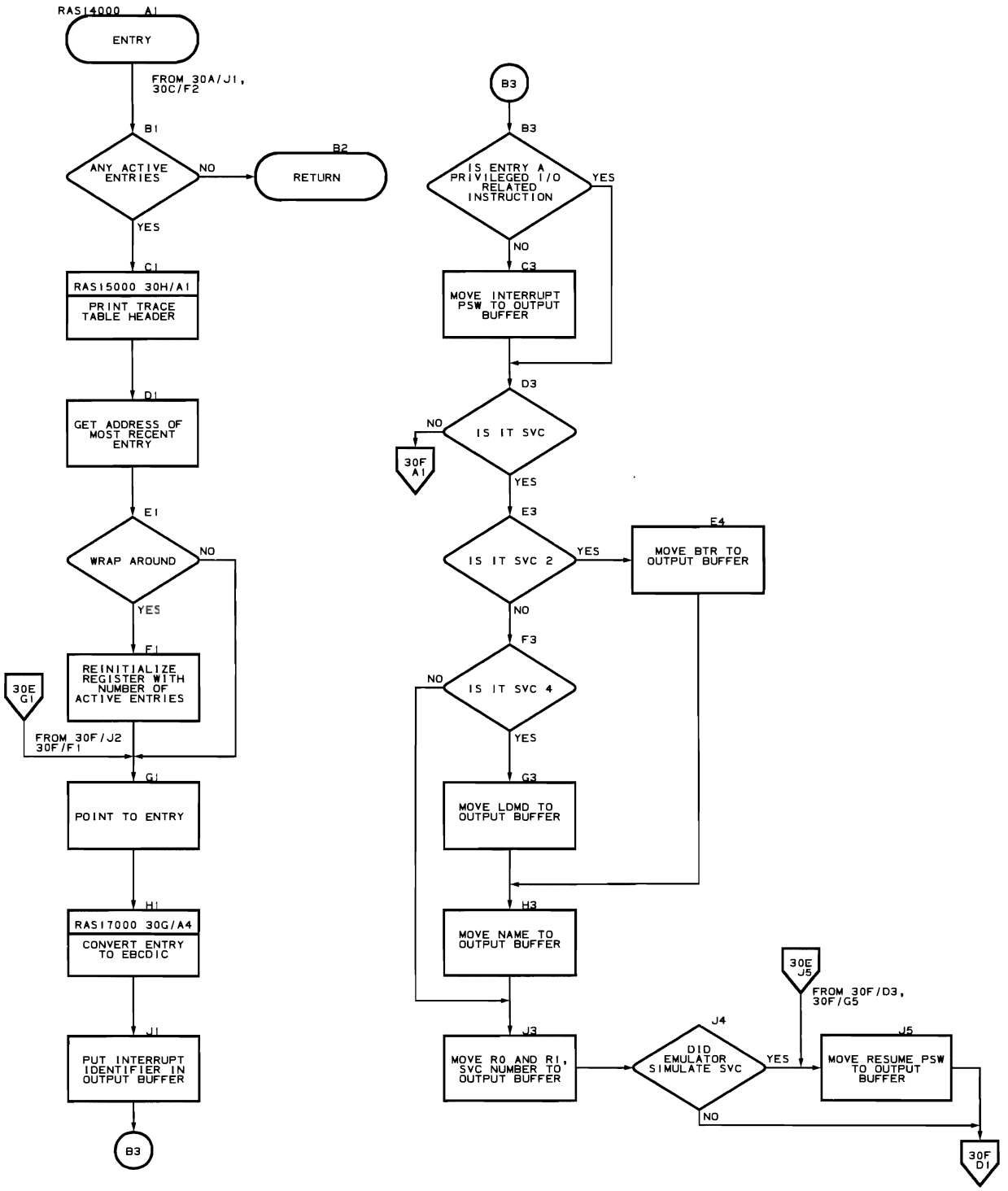
Flowchart 30C. Snap Dump and Trace Formatting Routine (IIVSNP Part 3 of 10)



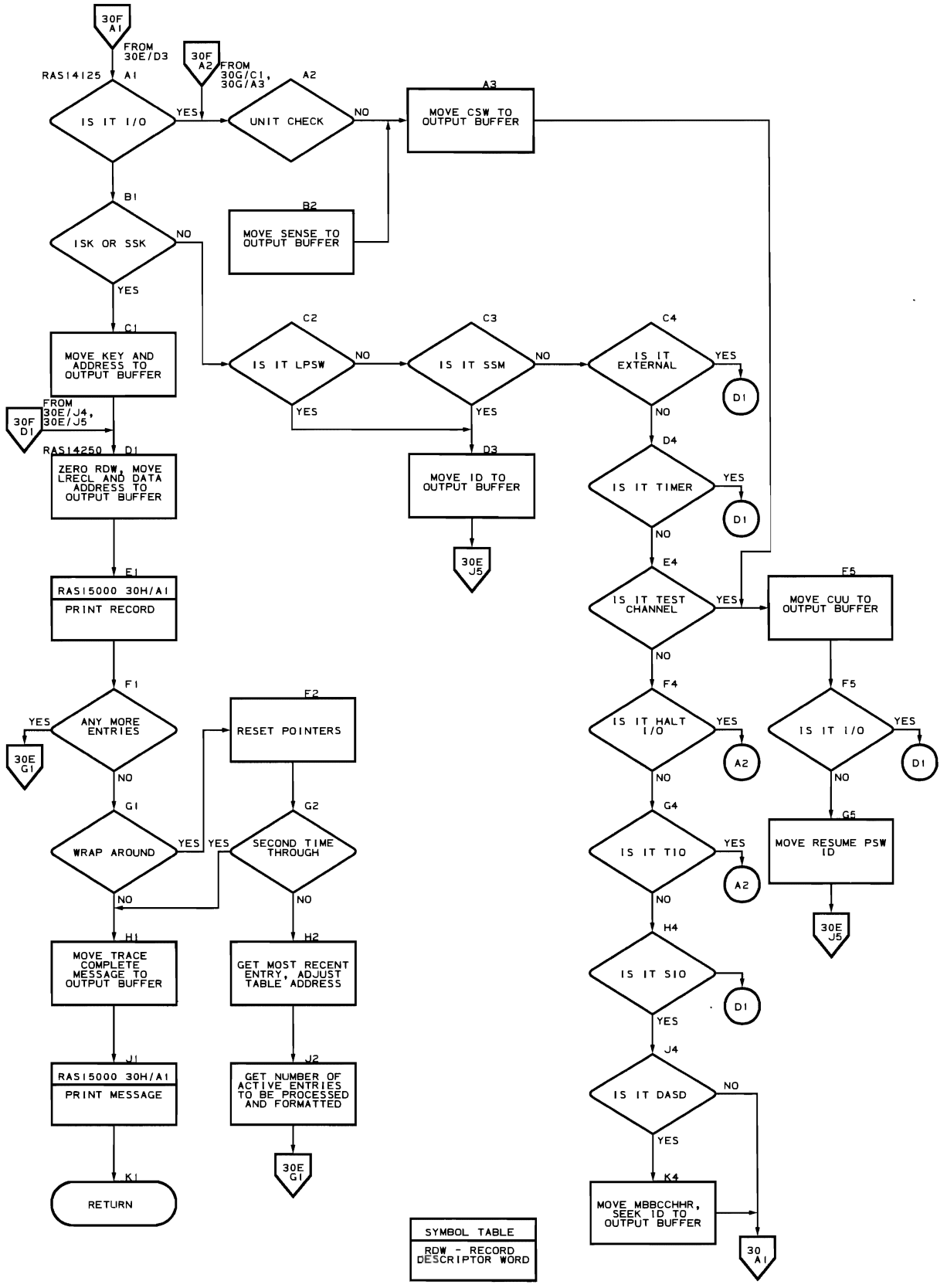
Flowchart 30D. Snap Dump and Trace Formatting Routine (IIVSNP Part 4 of 10)



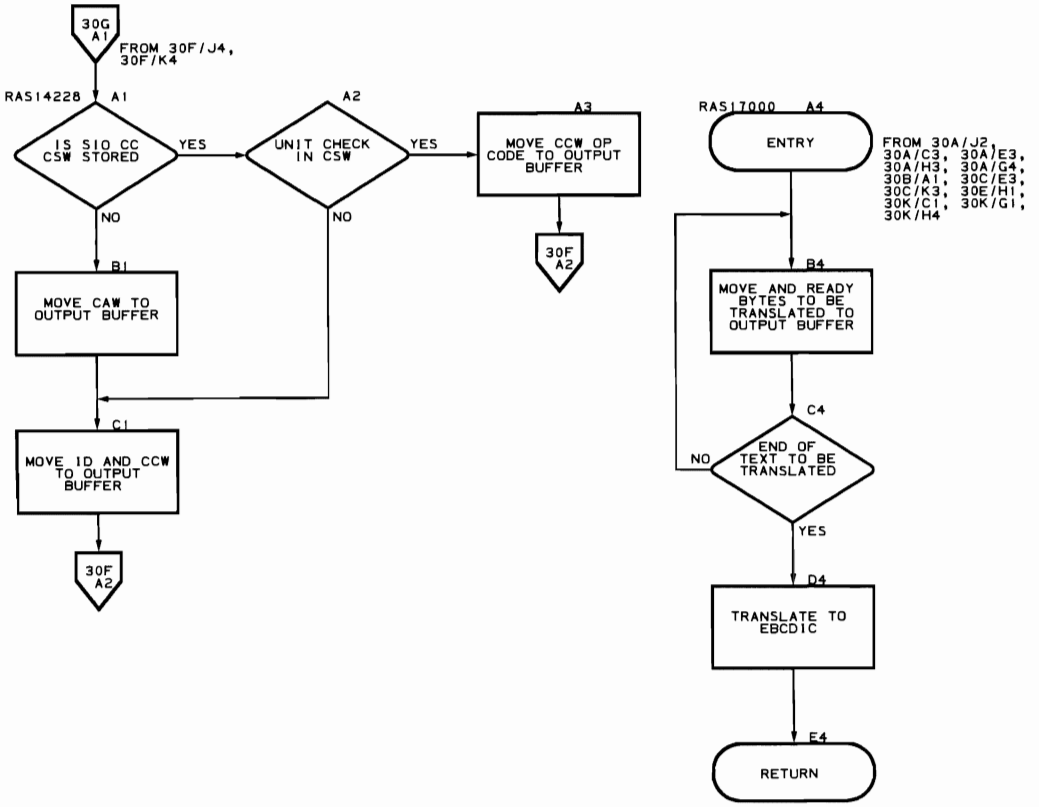
Flowchart 30E. Trace Table Subroutine (IIVSNP Part 5 of 10)



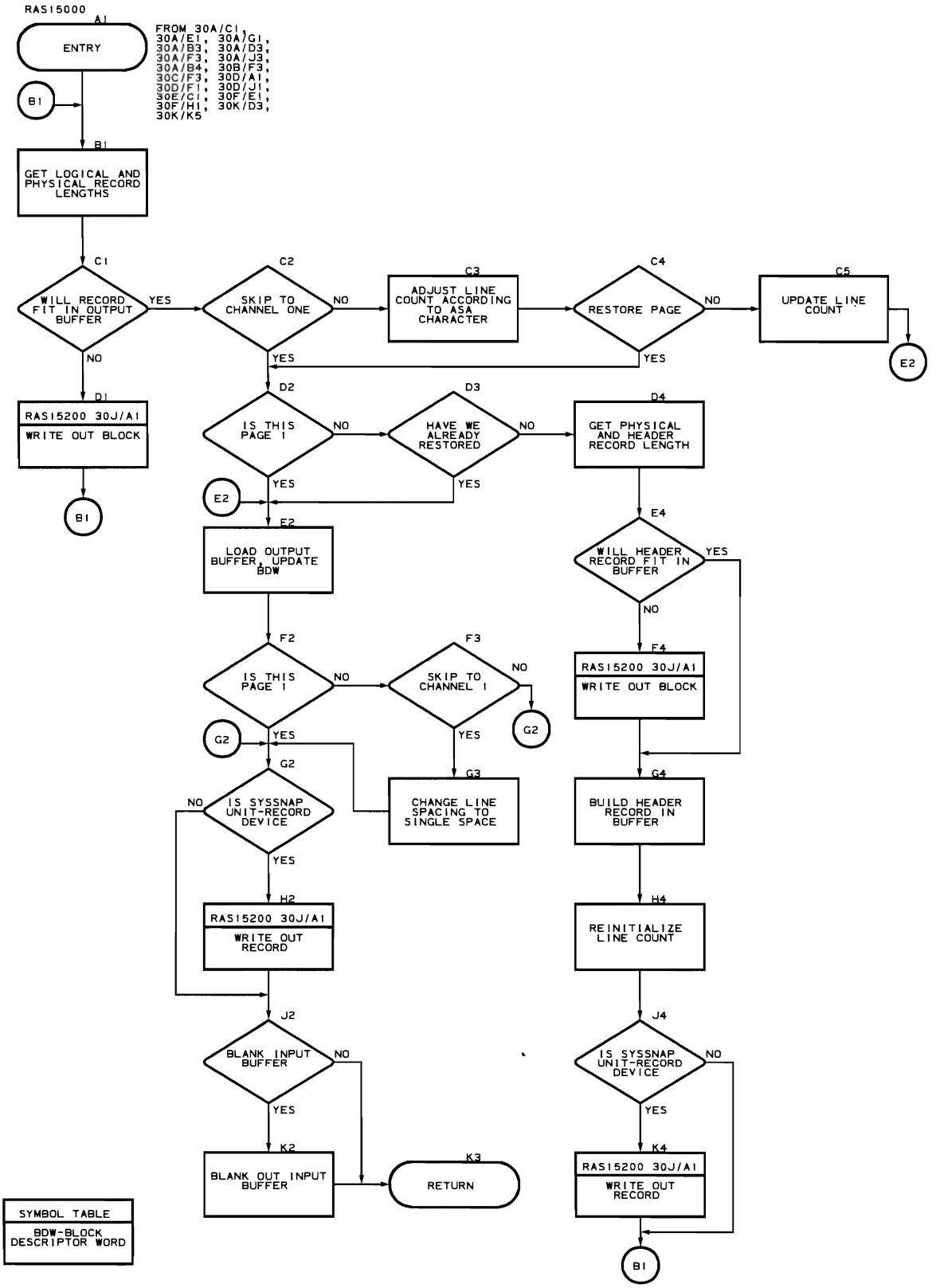
Flowchart 30F. Trace Table Subroutine (IIVSNP Part 6 of 10)



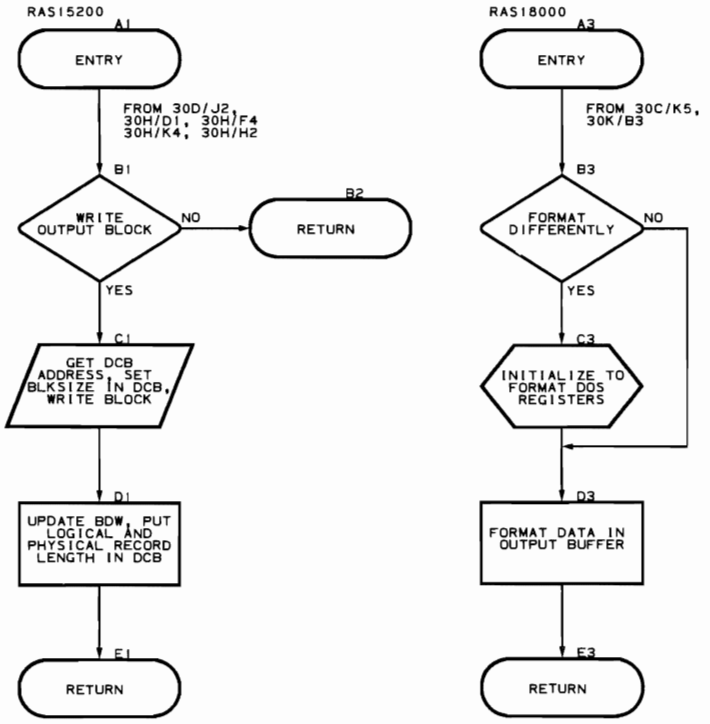
Flowchart 30G. Trace Table and EBCDIC Conversion Subroutines (IIVSNP Part 7 of 10)



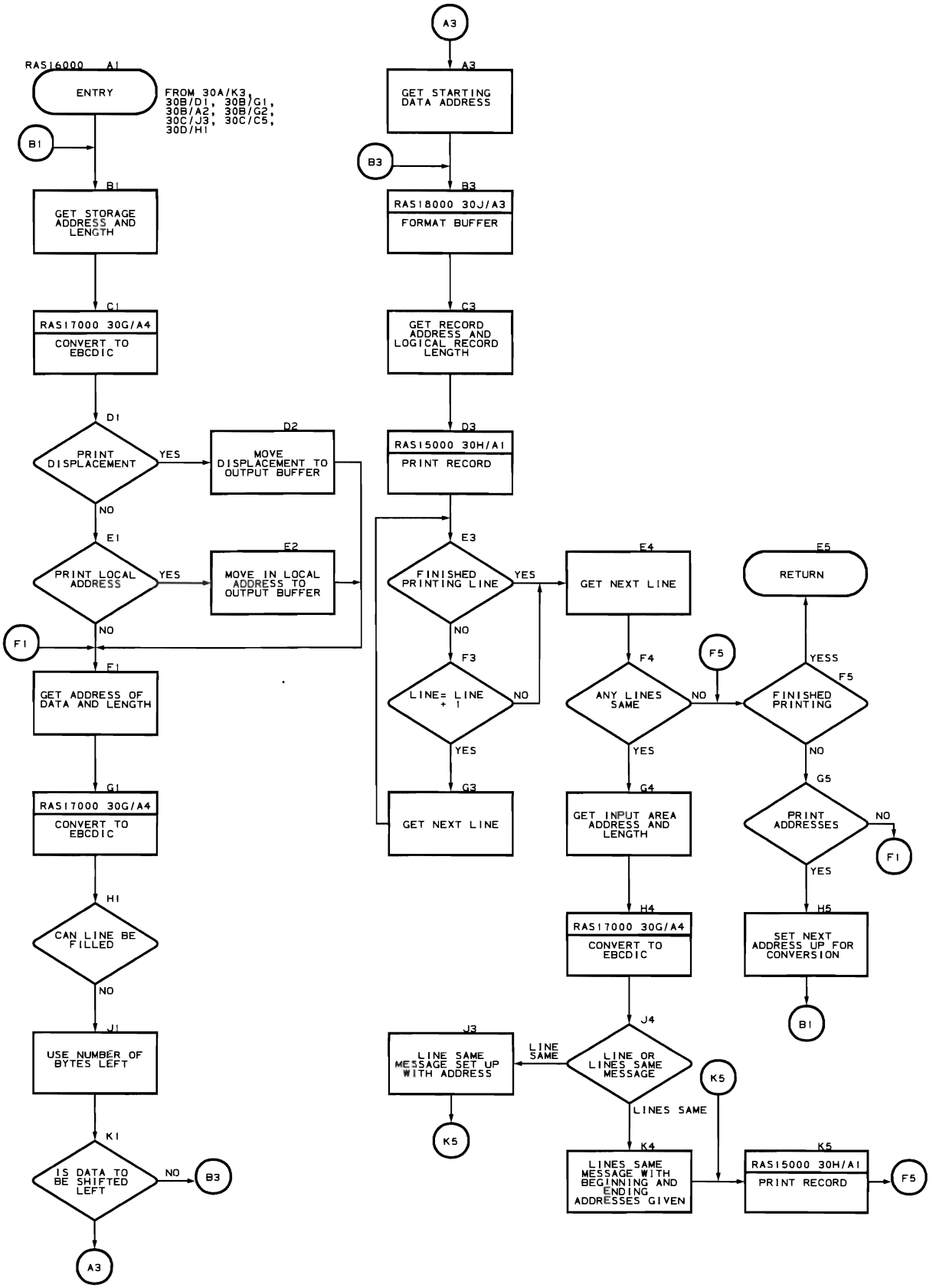
Flowchart 30H. Write Subroutines (IIVSNP Part 8 of 10)



Flowchart 30J. Write and Data Formatting Subroutines (IIVSNP Part 9 of 10)



Flowchart 30K. Snap Subroutine (IIVSNP Part 10 of 10)



Program Check Intercept Routine (Flowcharts 31A-31C)

Module name: IIVPCI

Entry point name: IIVPCI

Major functions:

- Trace and snap privileged operations
- Give control to a program check user exit if specified

Entered by: IIVRAS

Modules called:

- IIVSNP
- IIVRASnn (program check user exit routine)

Exits to: IIVPCE

OS macros issued: None

Input:

- Fields in IIVRCN (RASCONS)
- DOS storage

Output: Fields in IIVRCN (RASCONS)

Return codes: None

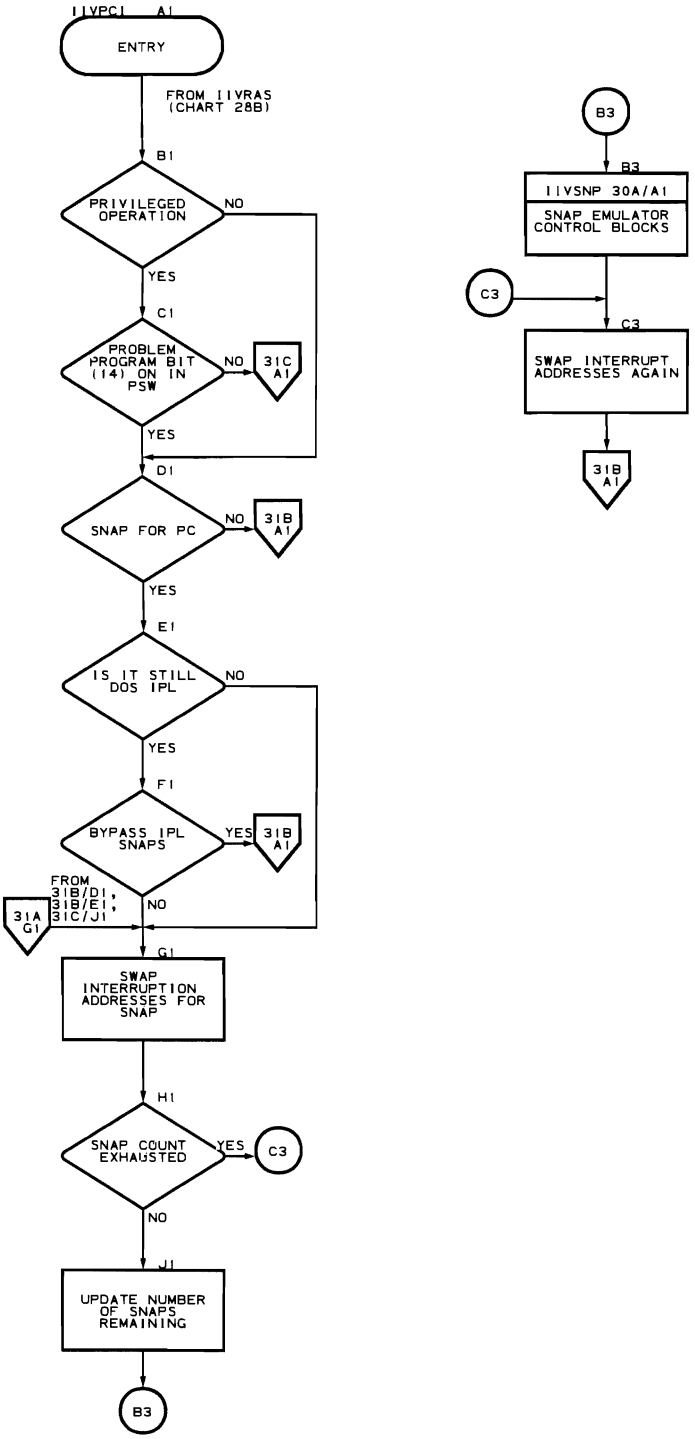
Tables/work areas:

- IIVRCN (RASCONS)
- IIVCON (EMUCONS)
- Local execution list
- DOS storage

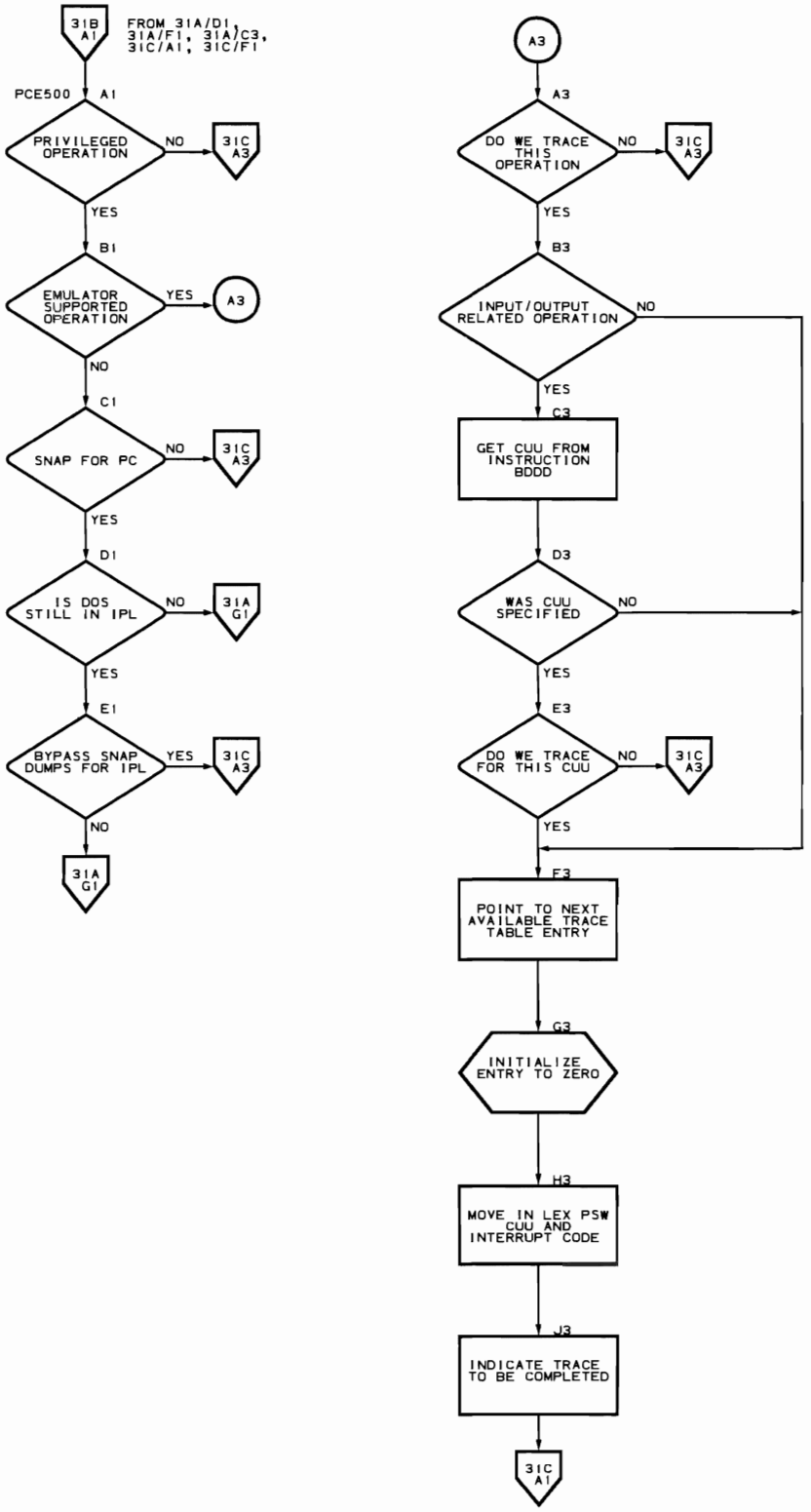
Errors detected: None

Messages requested: None

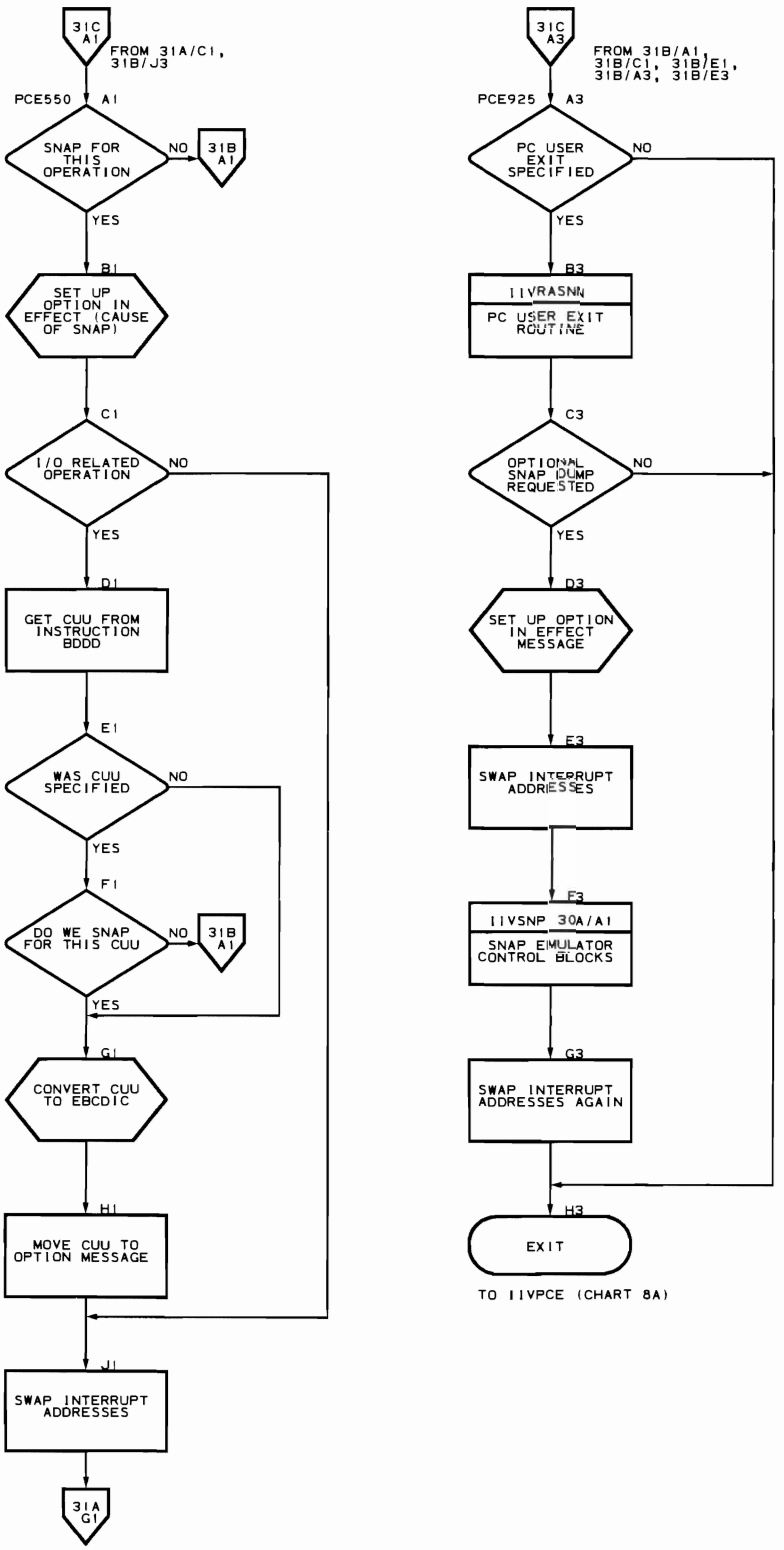
Flowchart 31A. Program Check Intercept Routine (IIVPCI Part 1 of 3)



Flowchart 31B. Program Check Intercept Routine (IIVPCI Part 2 of 3)



Flowchart 31C. Program Check Intercept Routine (IIVPCI Part 3 of 3)



Supervisor Call Intercept Routine (Flowchart 32A)

Module name: IIVSCI

Entry point name: IIVSCI

Major functions:

- Intercepts DOS supervisor call when TRACE=SVC or SNAP=SVC=nn is specified
- Gives control to user exit routines

Entered by: IIVRAS

Modules called:

- IIVSNP
- IIVRASnn (supervisor call user exit routine)

Exits to: IIVSVC

OS macros issued: None

Input:

- Fields in IIVRCN (RASCONS
- DOS storage

Output: Fields in IIVRCN (RASCONS)

Return codes: None

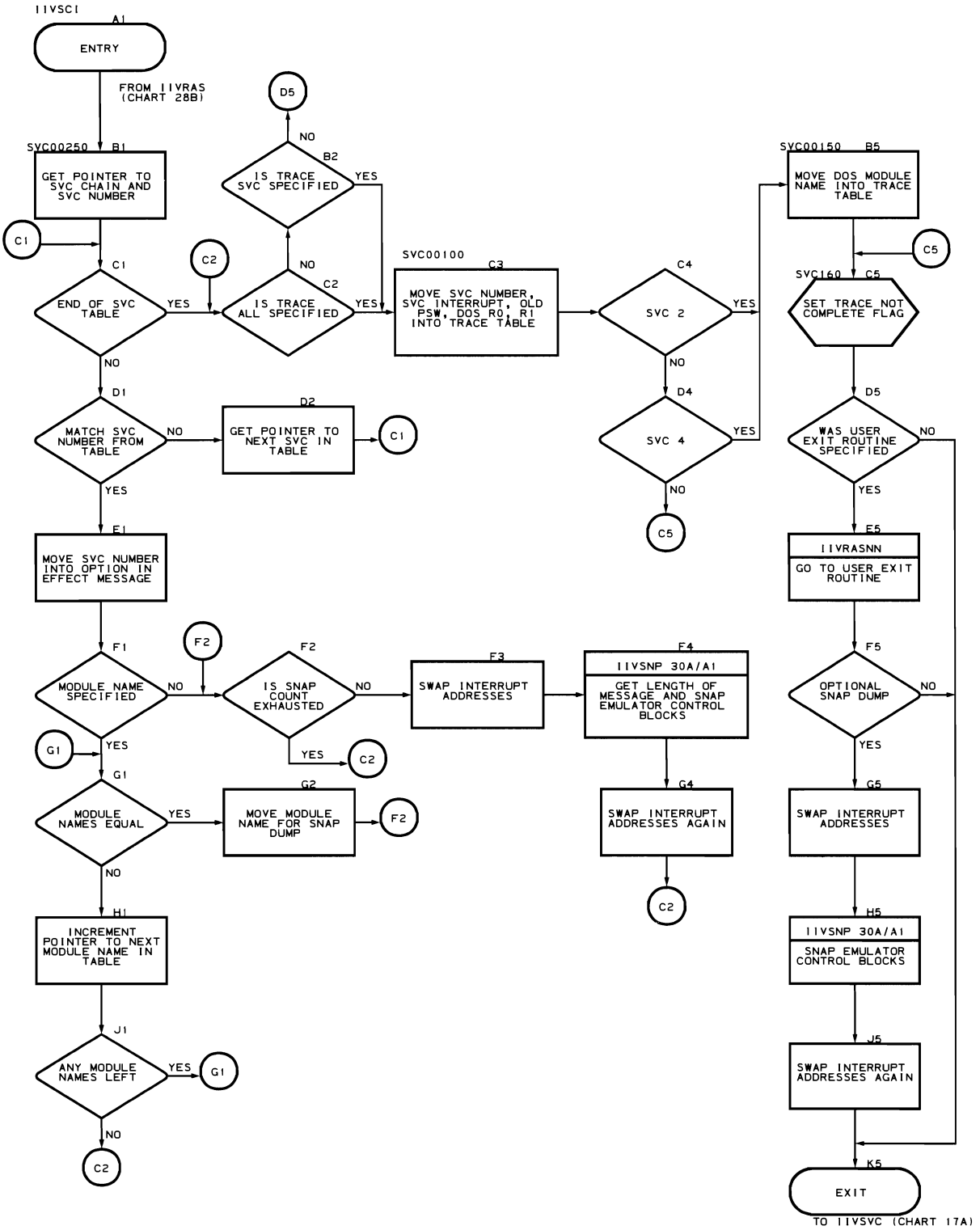
Tables/work areas:

- IIVRCN (RASCONS)
- IIVCON (EMUCONS)
- Local execution list
- DOS storage

Errors detected: None

Messages requested: None

Flowchart 32A. Supervisor Call Intercept Routine (IIVSCI)



Asynchronous Intercept Routine (Flowcharts 33A-33H)

Module name: IIVACI

Entry point name: IIVACI

Major functions:

- PSW compare
- Trace entry completion
- Second snap for privileged operations
- Local execution exit routine
- Check for asynchronous interruptions (trace and snap)
- Give control to user exit routines

Entered by:

- Entry point address placed in EMUCONS (ART20 by IIVRCP)
- IIVPCE
- IIVLOG
- IIVRTE
- IIVSTG
- IIVRAS

Modules called:

- IIVSNP
- IIVRASnn (local execution user exit routine)

Exits to: IIVRTE

OS macros issued: None

Input:

- Fields in module IIVRCN (RASCONS)
- COMTAB
- DOS storage

Output: Fields in module IIVRCN (RASCONS)

Return codes: None

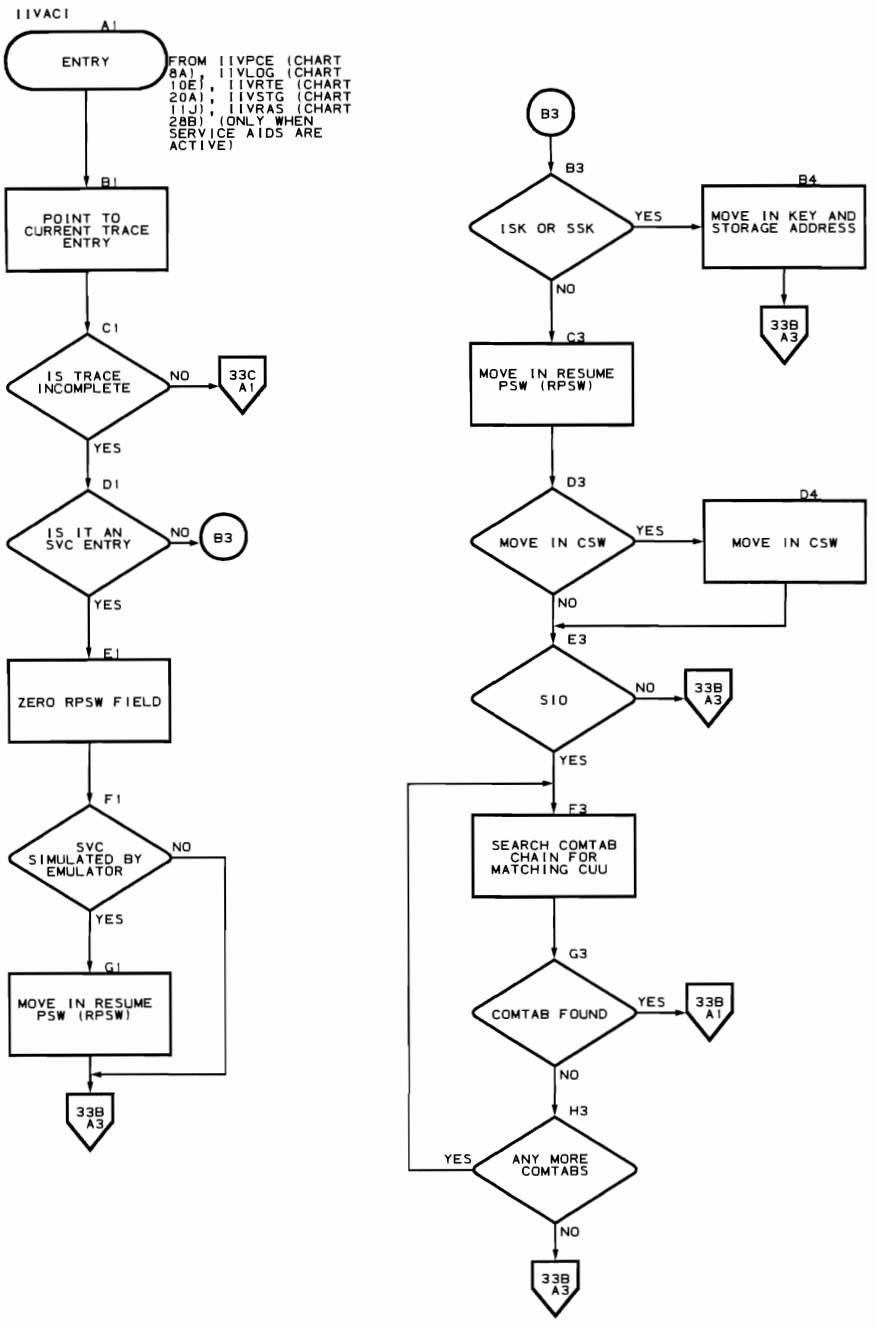
Tables/work areas:

- IIVRCN (RASCONS)
- EMUCONS
- Local execution list
- COMTAB
- DOS storage

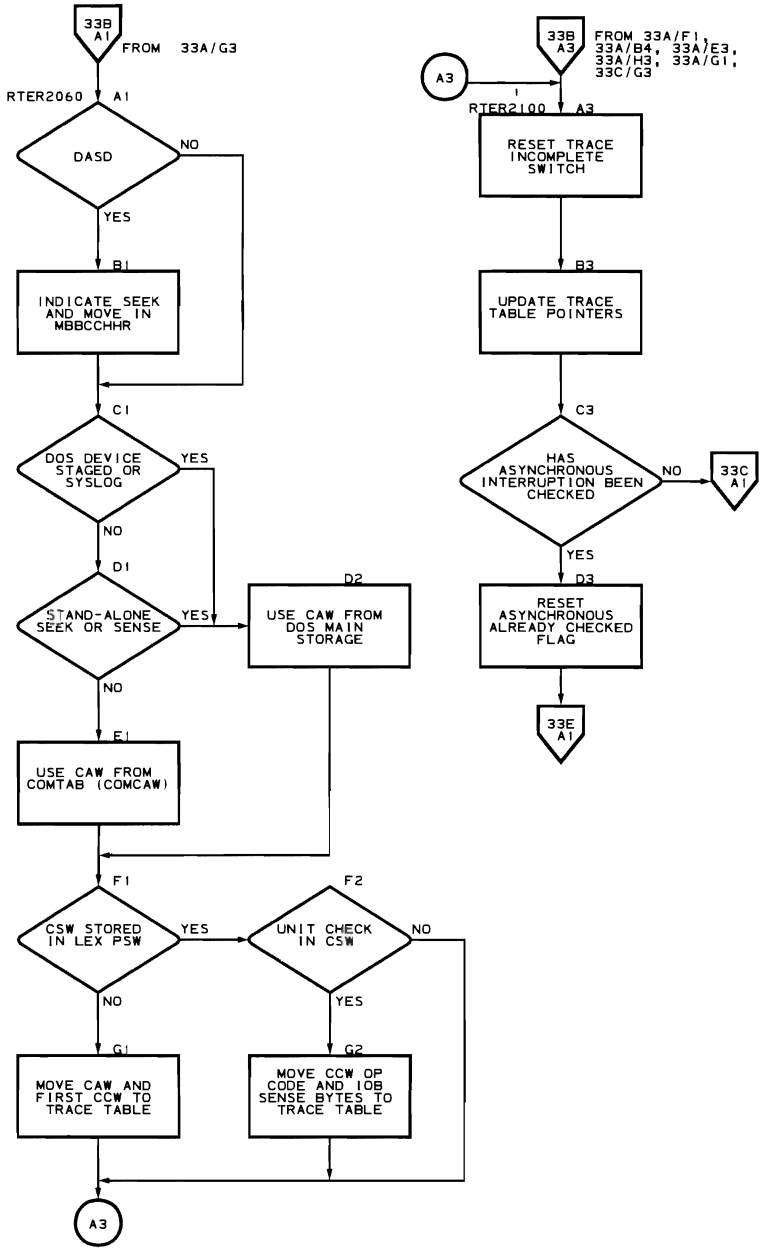
Errors detected: None

Messages requested: None

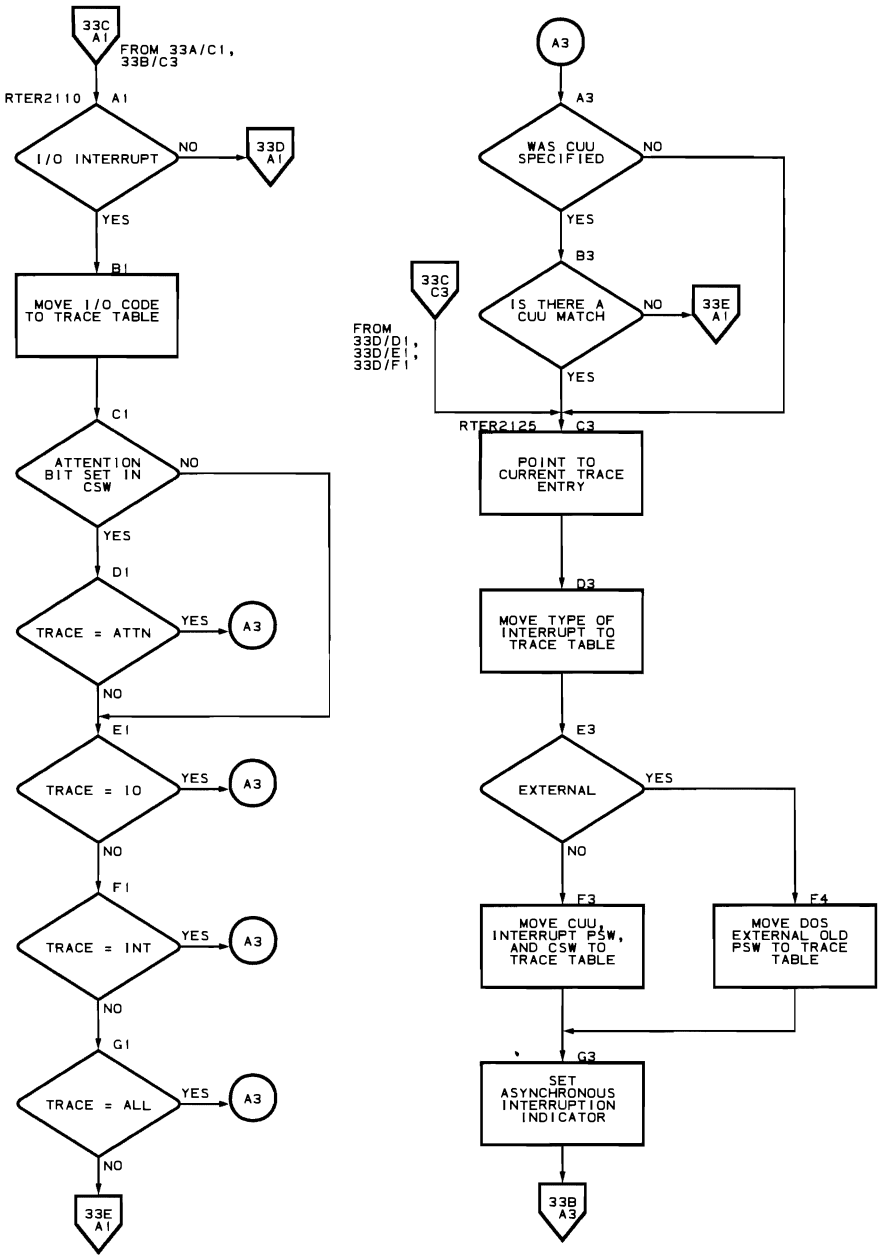
Flowchart 33A. Asynchronous Intercept Routine (IIVACI Part 1 of 8)



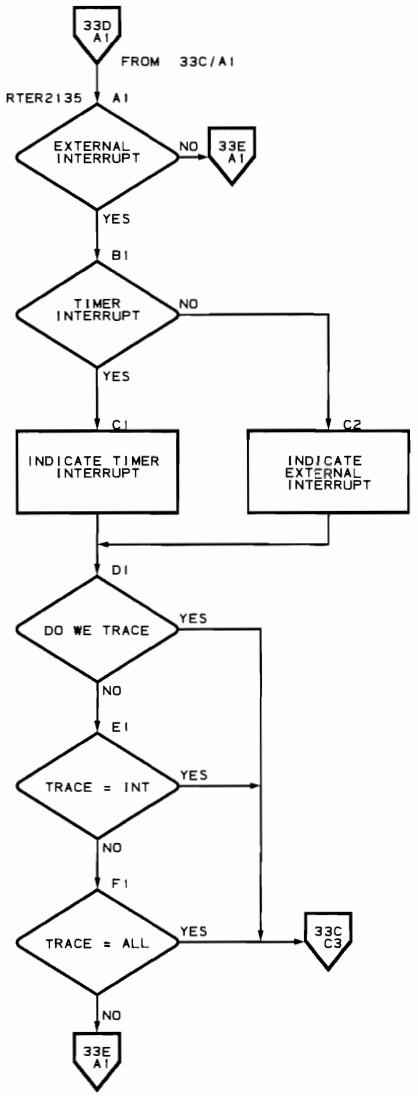
Flowchart 33B. Asynchronous Intercept Routine (IIVACI Part 2 of 8)



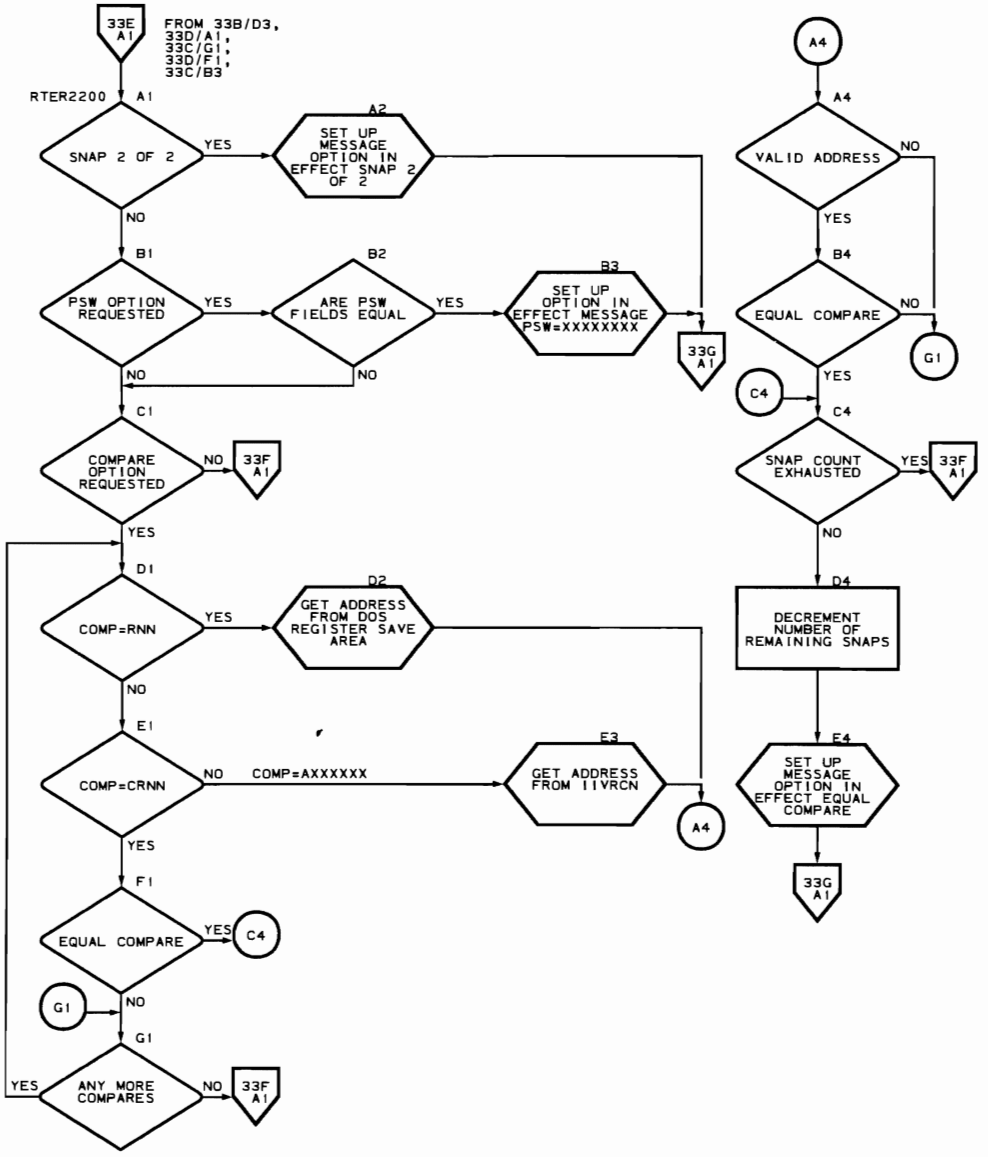
Flowchart 33C. Asynchronous Intercept Routine (IIVACI Part 3 of 8)



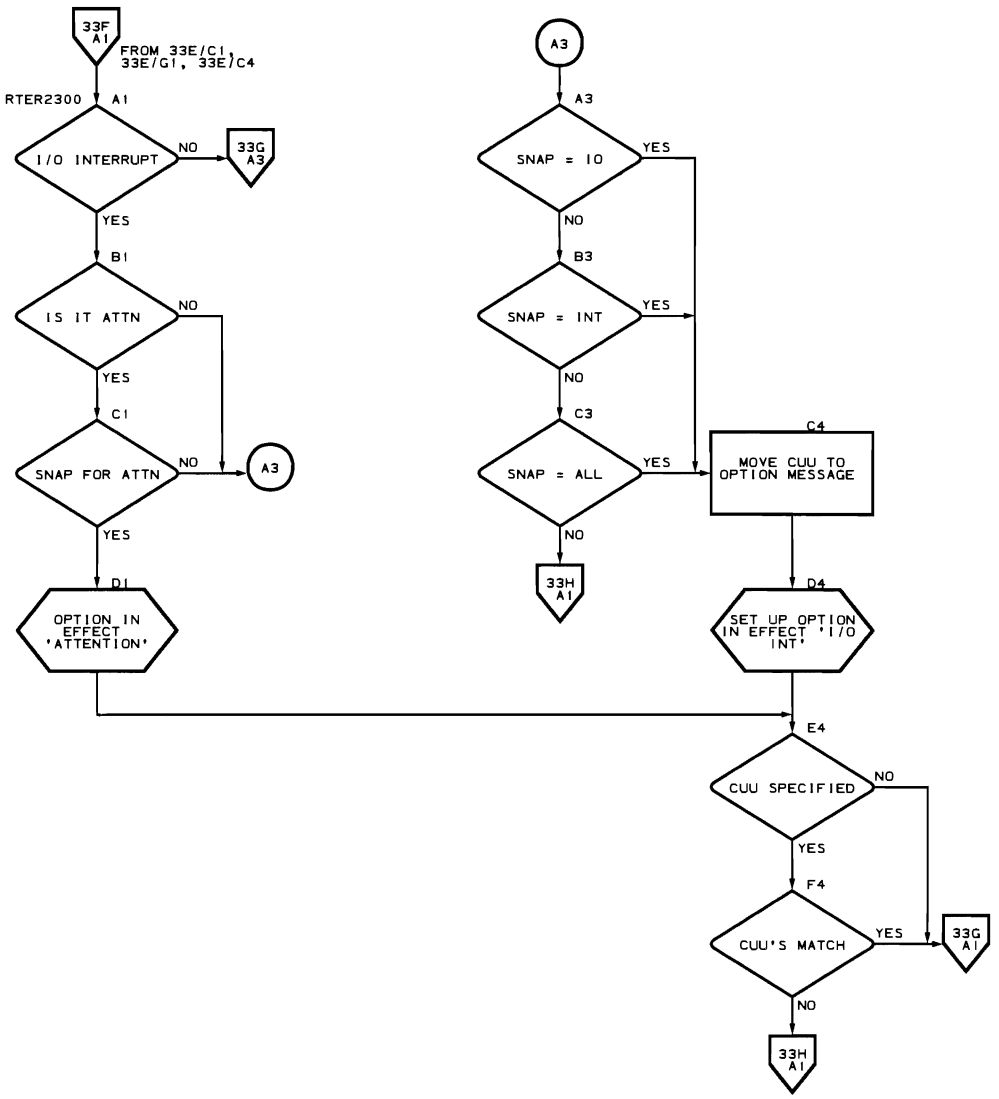
Flowchart 33D. Asynchronous Intercept Routine (IIVACI Part 4 of 8)



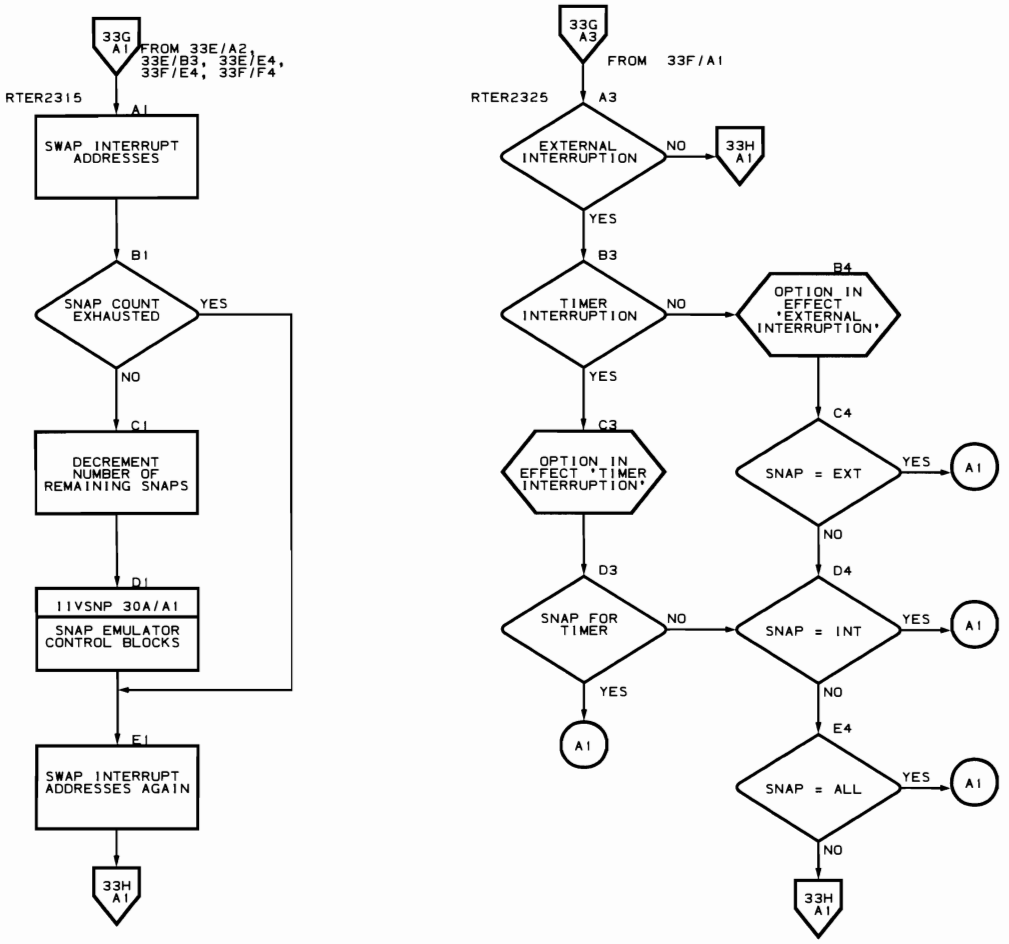
Flowchart 33E. Asynchronous Intercept Routine (IIVACI Part 5 of 8)



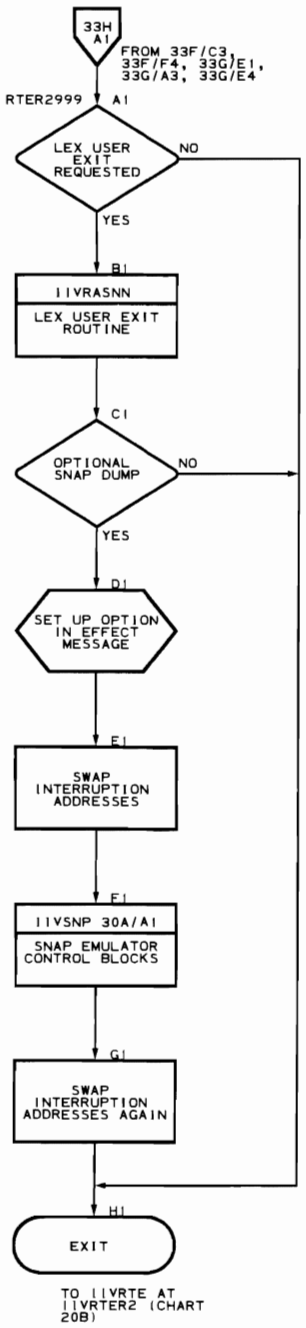
Flowchart 33F. Asynchronous Intercept Routine (IIVACI Part 6 of 8)



Flowchart 33G. Asynchronous Intercept Routine (IIVACI Part 7 of 8)



Flowchart 33H. Asynchronous Intercept Routine (IIVACI Part 8 of 8)



Service Aids Adjust CCW Data Address Routine (Flowcharts 34A-34H)

Module name: IIVRCW

Entry point name: IIVRCW, RCCWAB4

Major functions:

- Adjusts data addresses in a string of CCWs before being passed to OS for an I/O operation and after completion of the CCW function in OS
- The CCW's are adjusted in the DIAG block built by IIVRCP when the DIAG feature is called

Entered by:

- IIVPCE
- IIVCHK
- IIVRTE

Modules called:

- IIVADJ
- IIVMSG

Exits to:

- IIVPCE
- IIVCHK
- Caller

OS macros issued: None

Input:

- Adjustment factor
- Pointer to a CCW string
- Pointer to the local execution list
- Pointer to the Emulator constants area (IIVCON)
- Pointer to COMTAB
- Local limit address

Output: Adjusted CCW string

Return codes:

- Error code 16 for IIVABN
- Error code 20 for IIVABN

Tables/work areas:

- Adjust CCW data address list (located in IIVCON)
- BEBLK CCW addresses for adjusting
- Status modifier table used to find status modifier-type CCWs
- RCCWSAVE register save area
- DIAG block

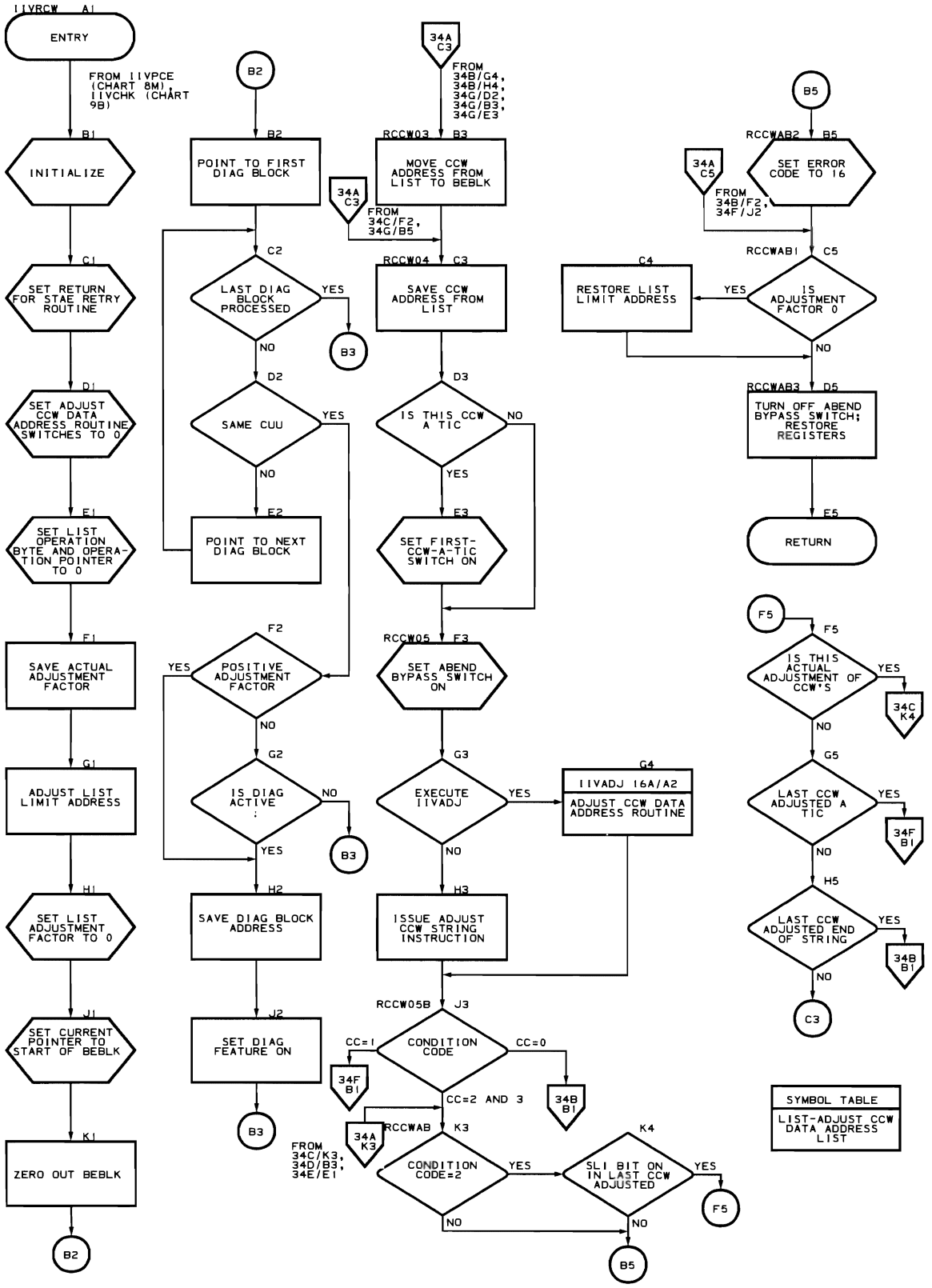
Errors detected:

- BEBLK is filled and CCW adjusting cannot proceed
- Adjusted addresses not within DOS storage area

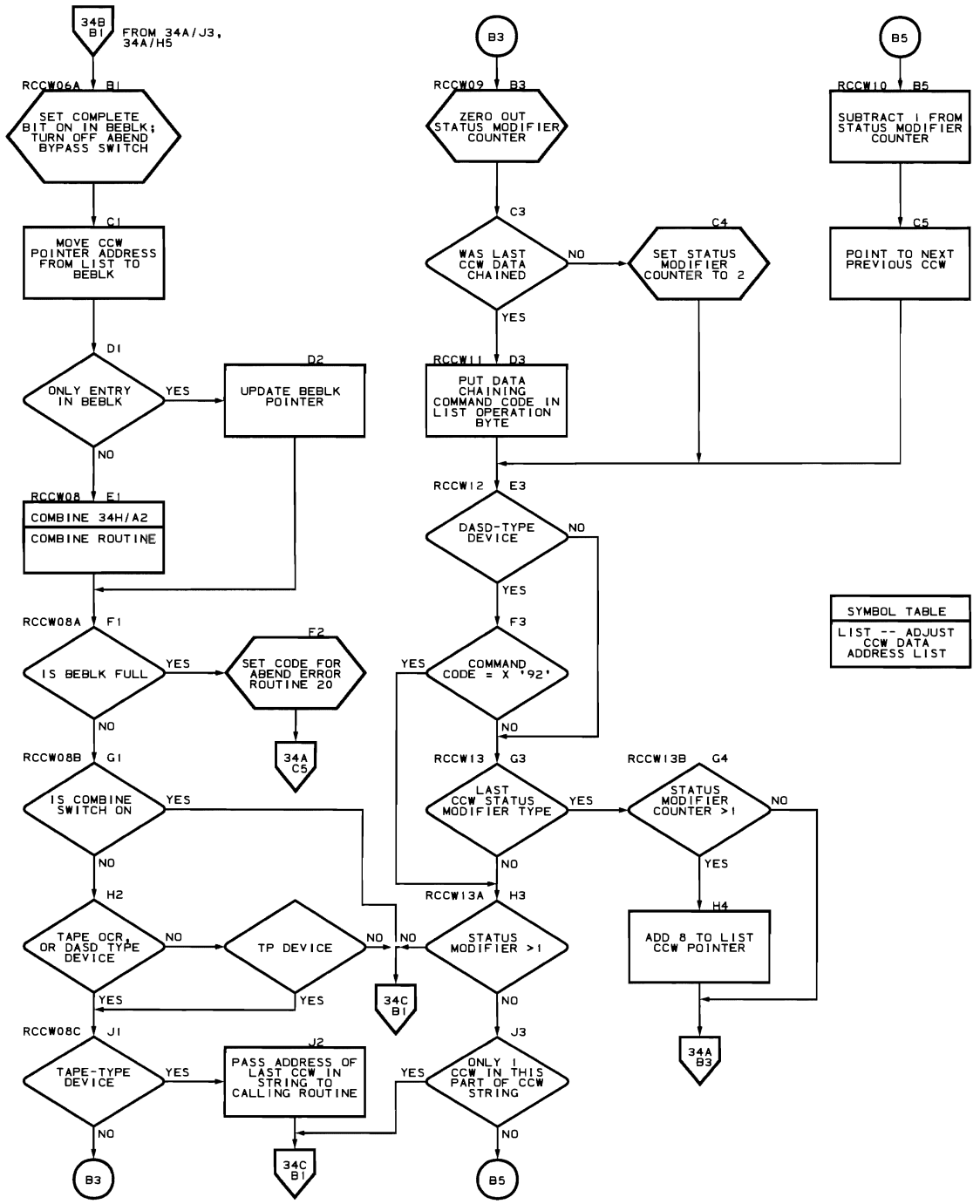
Messages requested:

- IIV280I
- IIV281I

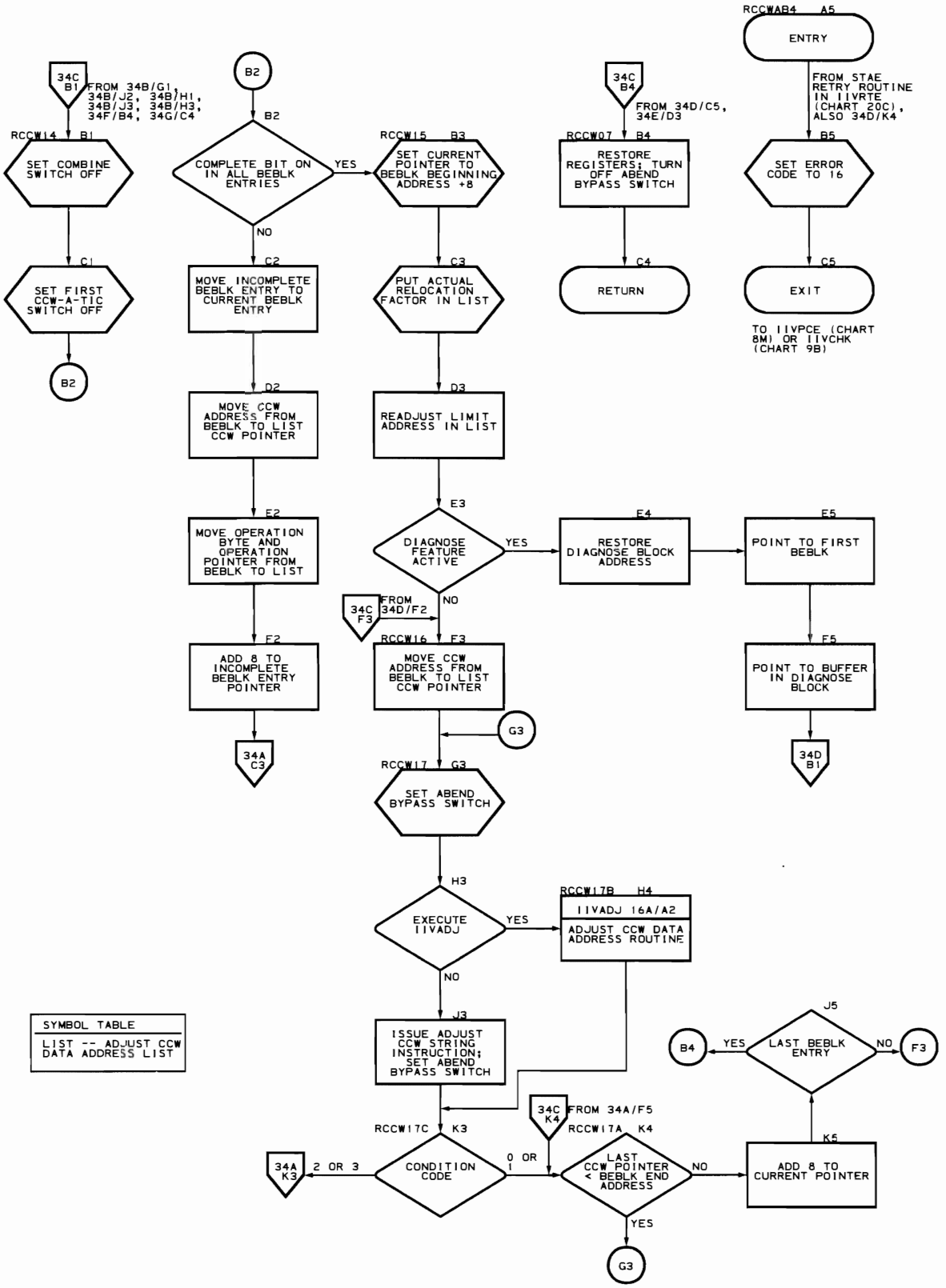
Flowchart 34A. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 1 of 8)



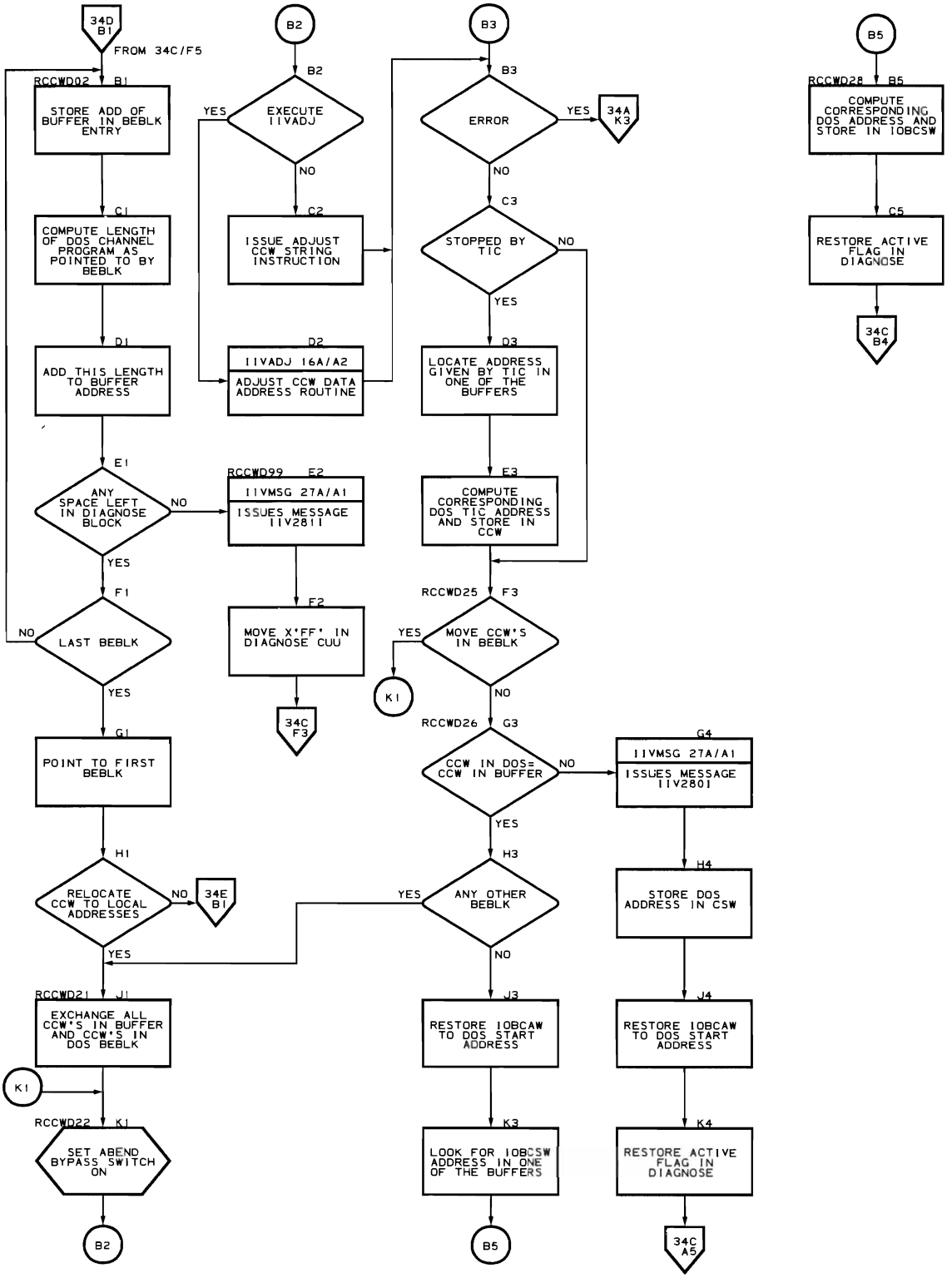
Flowchart 34B. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 2 of 8)



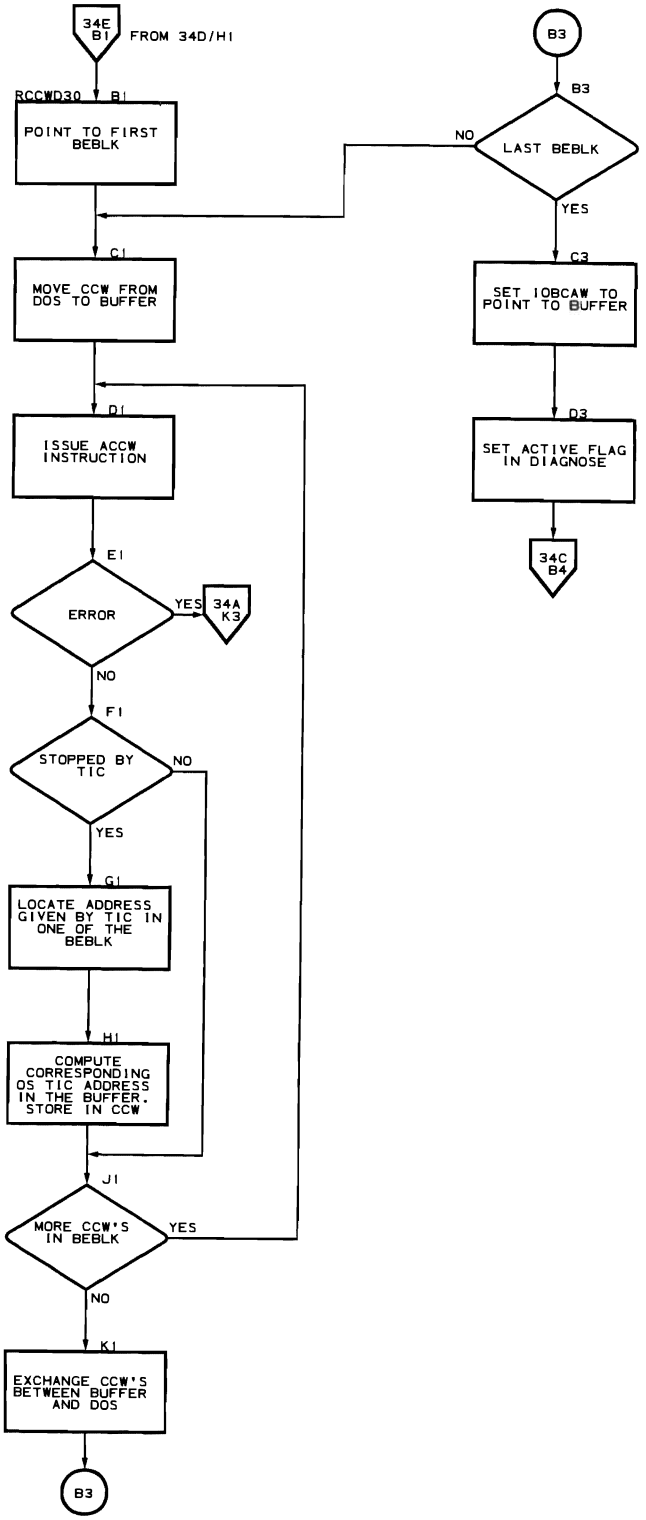
Flowchart 34C. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 3 of 8)



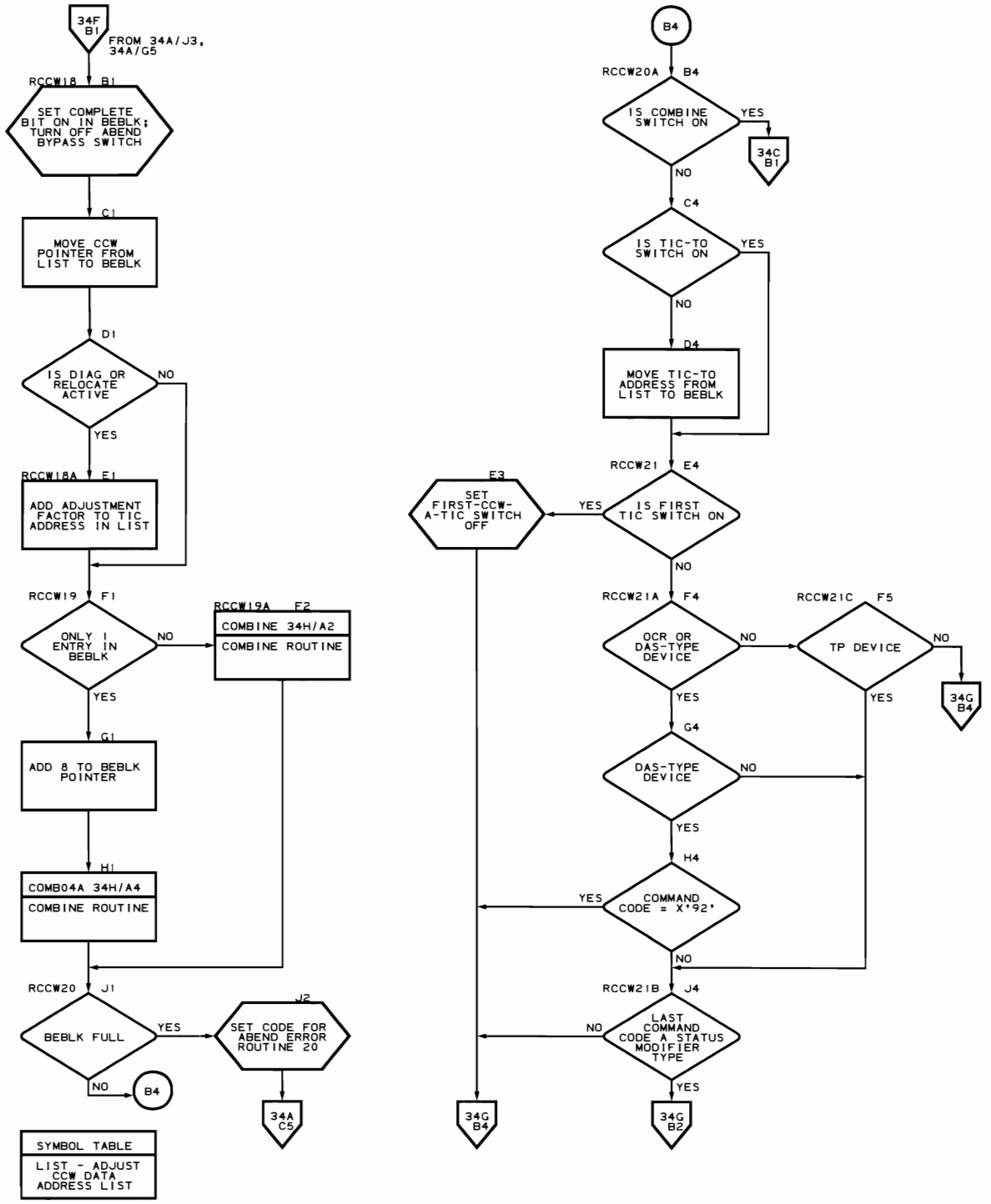
Flowchart 34D. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 4 of 8)



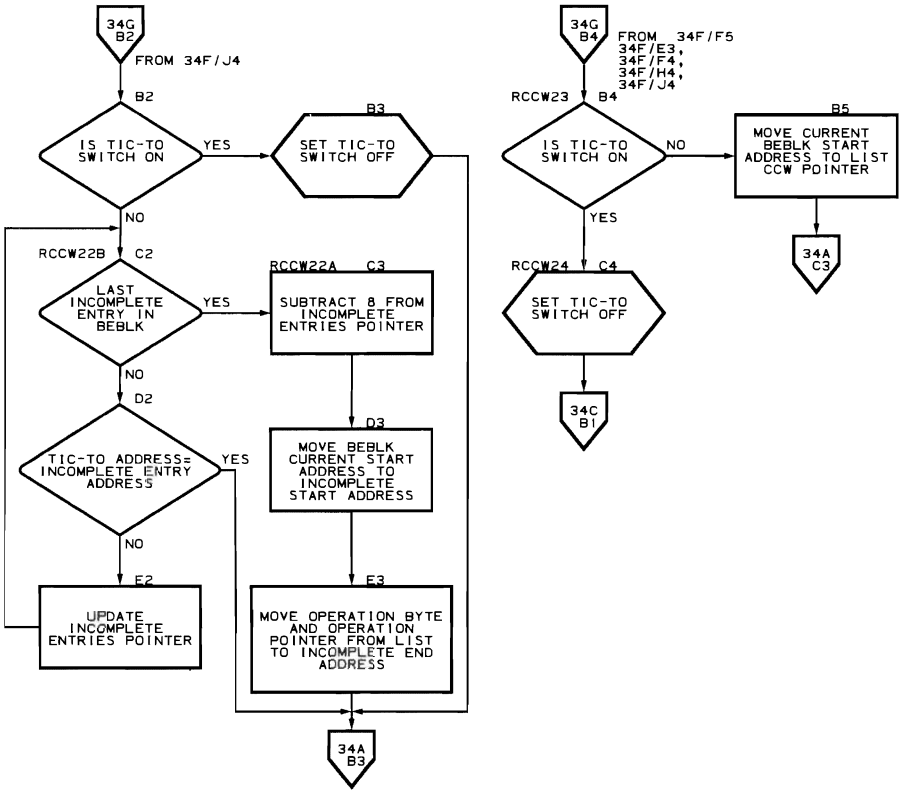
Flowchart 34E. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 5 of 8)



Flowchart 34F. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 6 of 8)

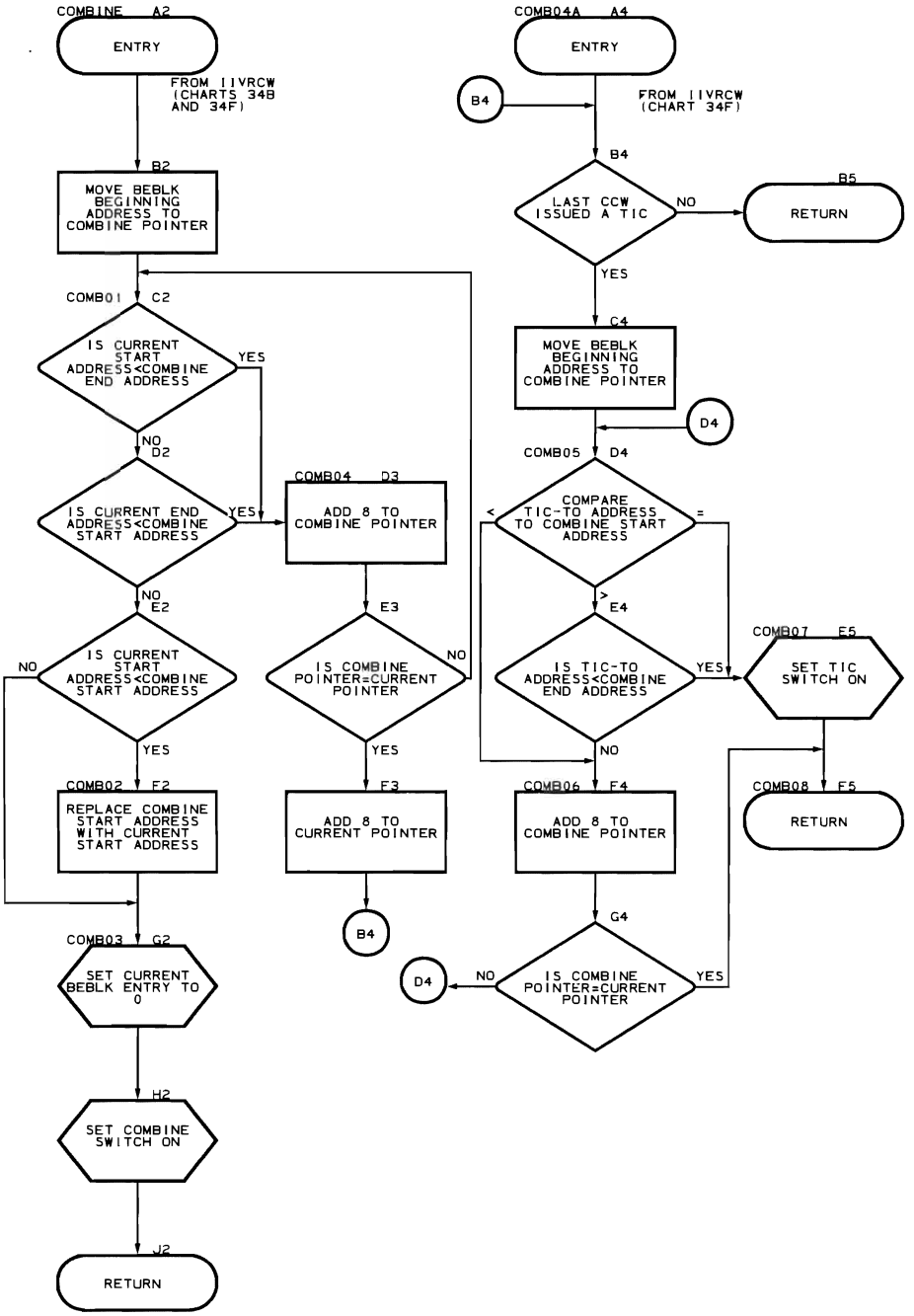


Flowchart 34G. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 7 of 8)



SYMBOL TABLE
LIST - ADJUST CCW DATA ADDRESS LIST

Flowchart 34H. Service Aids Adjust CCW Data Address Routine (IIVRCW Part 8 of 8)



DIRECTORIES

Emulator Module Directory

Data Area Directory

Emulator Macros

Symbol Table

Field Name Table

Figure 28 contains information about the physical structure of the Emulator. The column headings and their meanings are:

Module Name. The name of the object module containing the routine, table, or transient area being described. The microfiche cards containing program listings are ordered by object module names.

Function. The Emulator routines are grouped according to operation: initialization, synchronous interruptions, asynchronous interruptions, direct-access volume sharing, abnormal end conditions, message writer, and service aids. The flowcharts in the section "Program Organization" are divided into these groups. For a precise definition of function as it relates to a particular routine, see the functional and routine descriptions given in the section "Method of Operation."

Name of Routine, Table, Transient Area. The name of the routine, table, or transient area being described.

Entry Point Name. Symbolic labels for routine entry points.

Control Section Name. The name of the control section containing the item.

PLM Reference - Description Page. The page on which a particular item is described.

PLM Reference - Flowchart Number. The flowchart number for the appropriate routine.

Module Name	Function	Name of Routine, Table, Transient Area	Entry Point Name	Control Section Name	PLM Reference Descrip-tion Pg.	PLM Reference Chart No.
IGG019SA	SIO,EOX,abn. end, ce	Start I/O, end-of-extent, channel end, and abnormal end appendage	IGG019SA	IGG019SA	73	18A-18B
IGG019S1	Abn. end, ce	Abnormal end/channel end appendage	IGG019S1	IGG019S1	74	19A
IIVABN	Abnormal end cond.	Exit-ABEND error routine	IIVABN	IIVABN	101	26A-26B
IIVACI	Service aids	Asynchronous intercept routine	IIVACI	IIVACI	108	33A-33H
IIVADD	Init.	IPL add routine	IIVADD	IIVADD	51	4A-4D
IIVADJ	Sync.int.	CCW adjustment routine	IIVADJ	IIVADJ	71	16A-16B
IIVAWV	Sync.int.	CAW verification routine	IIVAWV	IIVAWV	68	12A
IIVCCW	Sync.int.	Adjust CCW data address routine	IIVCCW	IIVCCW	69	15A-15F
IIVCHK	Sync.int.	Check I/O routine	IIVCHK	IIVCHK	61	9A-9D
IIVCWV	Sync.int.	CCW verification routine	IIVCWV	IIVCWV	68	13A
IIVDVS	D.A. volume shar.	Device sharing simulation routine	IIVDVS	IIVDVS	80	23A-23H
IIVENT	Init.	DOS Emulator entry routine	IIVENT	IIVENT	43	1A
IIVENT	Init.	DOS Emulator constants area (EMUCONS)	--	IIVCON	435	N/A
IIVFCB	Sync.int.	Carriage tape storage image table	IIVFCB	IIVFCB	26	N/A
IIVGET	Init.	GETMAIN/FREEMAIN routine	IIVGET	IIVGET	54	7A
IIVGR2	D.A. volume shar.	SVC monitor routine	IIVGR2	IIVGR2	79	22A-22D
IIVINT	Init.	Initialization first-load routine	IIVINT	IIVINT	43	2A-2M
IIVIN2	Init.	Initialization second-load routine	IIVIN2	IIVIN2	47	3A-3F
IIVIS	D.A. volume shar.	ISAM mapping routine	IIVIS	IIVIS	88	24A-24L
IIVLOG	Sync.int.	Interpretive SYSLOG routine	IIVLOG	IIVLOG	62	10A-10E

Figure 28 (Part 1 of 2). Emulator Module Directory

Module Name	Function	Name of Routine, Table, Transient Area	Entry Point Name	Control Section Name	PLM Reference Descrip- tion Pg.	Chart No.
IIVLOG	Sync.int.	Interpretive SYSLOG routine Entry point 1	IIVLOGR1	IIVLOG	63	10C
IIVLOG	Sync.int.	Interpretive SYSLOG routine Entry point 2	IIVLOGR2	IIVLOG	63	10E
IIVMSG	Message writer	Message writer routine	IIVMSG	IIVMSG	101	27A
IIVMG1	Message writer	Initialization message texts	--	IIVMG1	101	N/A
IIVMG2	Message writer	Post-initialization message texts	--	IIVMG2	101	N/A
IIVMG3	Message writer	Volume sharing and service aids message texts	--	IIVMG3	101	N/A
IIVOPN	Init.	Open routine	IIVOPN	IIVOPN	51	5A-5D
IIVPCE	Sync.int.	Program check executive routine	IIVPCE	IIVPCE	54	8A-8U
IIVPCI	Service aids	Program check intercept routine	IIVPCI	IIVPCI	107	31A-31C
IIVPOV	Sync.int.	Printer overflow routine	IIVPOV	IIVPOV	69	14A
IIVPRP	Async.int.	Prompt reply processor routine	IIVPRP	IIVPRP	78	21A-21E
IIVPR1	Sync.int.	1403/1443 printer translate table	--	IIVPR1	26	N/A
IIVPUB	Init.	OS PUB table build routine	IIVPUB	IIVPUB	53	6A-6F
IIVPU1	Sync.int.	1442 punch commands translate table	--	IIVPU1	26	N/A
IIVPU2	Sync.int.	2520 punch commands translate table	--	IIVPU2	26	N/A
IIVPU3	Sync.int.	3525 punch commands translate table	--	IIVPU3	26	N/A
IIVRAS	Service aids	Service aids initialization routine	IIVRAS	IIVRAS	104	28A-28B
IIVRAS	Service aids	Constants area (RASCONS)	--	IIVRCN	441	N/A
IIVRCP	Service aids	Command processor routine	IIVRCP	IIVRCP	105	29A-29F
IIVRCW	Service aids	Service aids adjust CCW data address routine	IIVRCW	IIVRCW	108	34A-34E
IIVRD1	Sync.int.	1442 reader commands translate table routine	--	IIVRD1	26	N/A
IIVRD2	Sync.int.	2520 reader commands translate table	--	IIVRD2	26	N/A
IIVRD3	Sync.int.	2501 translate table	--	IIVRD3	26	N/A
IIVRP1	Sync.int.	2540 translate table	--	IIVRP1	26	N/A
IIVRTE	Async.int.	Asynchronous interrupt exit routine	IIVRTE	IIVRTE	74	20A-20G
IIVRTE	Async.int.	Asynchronous interrupt check subroutine	IIVRTECK	IIVRTE	77	20E-20F
IIVRTE	Async.int.	End-of-Job routine	IIVRTEOJ	IIVRTE	77	20G
IIVRTE	Async.int.	Asynchronous interrupt exit routine Entry point 1	IIVRTER1	IIVRTE	74	20A
IIVRTE	Async.int.	Route routine, entry point 2	IIVRTER2	IIVRTE	75	20B
IIVRTE	Async.int.	Select routine	IIVRTESL	IIVRTE	76	20C
IIVRTE	Async.int.	Timer interrupt subroutine	IIVRTETM	IIVRTE	77	20D
IIVRTE	Async.int.	Timer interrupt check subroutine	TIMECHK	IIVRTE	77	20D
IIVSCI	Service aids	Supervisor call intercept routine	IIVSCI	IIVSCI	107	32A
IIVSDT	Init.	Staged device table	--	IIVSDT	49	N/A
IIVSNP	Service aids	SNAP dump and trace formatting routine	IIVSNP	IIVSNP	105	30A-30K
IIVSTG	Sync.int.	Staged I/O routine	IIVSTG	IIVSTG	64	11A-11N
IIVSVC	Sync.int.	Supervisor call routine	IIVSVC	IIVSVC	72	17A
IIVVIO	D.A. volume shar.	Device sharing VTOC I/O routine	IIVVIO	IIVVIO	96	25A-25E

Figure 28 (Part 2 of 2). Emulator Module Directory

DATA AREA DIRECTORY

Figure 29 contains a summary of information about the major data areas used by the Emulator. The column headings and their meanings are:

Data Area Name. Names the data area and its abbreviation, if there is one.

Beginning Symbol. Identifies the beginning of the data area and can be used to locate the data area in the program listings.

Creation. Contains some indication as to the origin of the data area.

"Requestor" means that creation and maintenance of the data area is the responsibility of the requestor. Other requestors do not have to explicitly create the needed data area.

Storage Area. Indicates where the data area is in main storage.

Size. Gives the size of the data area.

Means of Access. Indicates the most commonly used way of referring to the data area or one of its fields.

Data Area Name	Beginning Symbol	Created by	Storage Area	Size	Means of Access
Adjust CCW data address list	RCCWLIST	Emulator	Emulator area	24 bytes	Beginning symbol
Beginning and end block	BEBLK	Emulator	Emulator area	240 bytes	Beginning symbol
Channel address word (CAW)	DOSCAW	DOS	DOS supervisor area	Word	DOS storage area plus 72
Channel command word (CCW)	None	Requestor when needed	DOS problem program area	Doubleword	Address of requestor's first CCW is contained in the DOS CAW
Channel status word (CSW)	DOSCSW	Emulator	DOS supervisor area	Doubleword	DOS storage area plus 64
Command control block (CCB)	DOSCCB	DOS	DOS problem program area	16 byte entry for each channel program	Register 1 at SIO execution
Communications table (COMTAB)	COMTAB	Emulator	Emulator area	84 byte entry for each device + 1 for SYSLOG	Address in IIVCON
COMTAB extension	CTEXT	Emulator	Emulator area	64 bytes	Address in IIVCOM
DASD label (DLBL)	DLBLDS	DOS	DOS B-transient area + 1192 bytes	Variable	Address in DOS open table for sequential disk; in problem program label area for direct-access disks
Data control block (DCB)	IHADCB (DSECT)	Emulator	Emulator area	Variable	COMTAB - symbolic names equated to offsets of fields
Data event control block (DECB) -- BISAM	DECBECB	Emulator	Emulator area	26 bytes	Immediately follows FIDBLK
Data extent block (DEB)	None	OS during open	Protected storage Emulator area	Varies with device and access method	DCB - symbolic names equated to offsets of fields

Figure 29 (Part 1 of 3). Data Area Directory

Data Area Name	Beginning Symbol	Created by	Storage Area	Size	Means of Access
Data set control block (DSCB) -- format 1	OBF1DATA	OS space allocation routine or DOS open routine	VTOC	140 bytes	OBTAIN
Data set control block (DSCB) -- format 3	OBF3LBL	OS space allocation routine or DOS open routine	VTOC	140 bytes	OBTAIN
Data set control block (DSCB) -- format 4	None	Utility programs	VTOC	140 bytes	Channel program
DIAG block	DIAGBLK	Emulator	Emulator area	112 bytes	Address in IIVCON
DTFIS ADD-RETRVE-ADDRTR table	IJHCTBL1	DOS	DOS problem program area	Variable	DOS register 0 (points to fullword constant) at SETL; DOS register 1 at GET/PUT/ LSETL/READ/WRITE/WAITF; DOS register 2 at OPEN/CLOSE
DTFIS LOAD table	IJHKTABL	DOS	DOS problem program area	Variable	DOS register 0 (points to fullword constant) at SETTL/ENDFL; DOS register 1 at WRITL; DOS register 2 at OPEN/CLOSE
Event control block (ECB)	None	Emulator	COMTAB	Word	COMTAB
ECB pointer table	ECBLIST (pointer)	Emulator	Emulator area	4 byte entry for each DCB, SYSLOG, and 3 special ECBs	Address in IIVCON
File ID block (FIDBLK)	FIDBLK (DSECT)	Emulator	Emulator area	16 bytes	Address in IIVCON
IIVCON	IIVCON	Emulator	Emulator area	1208 bytes	External reference (see Emulator linkage editor map)
IIVRCN	IIVRCN	Emulator	Emulator area	558 bytes	Address in module IIVRAS and IIVCON
Input/output block (IOB)	IOBDSECT	Emulator	Emulator area	10 words	COMTAB
ISAM block	ISBLK (DSECT)	Emulator	Emulator area	388 or 624 bytes	Address in ISFEGSV\ field (following FIDBLK field, which is pointed to by IIVCON)
ISK/SSK table	ISSKTAB (pointer)	Emulator	Emulator area	1 byte entry for each 2K block of DOS storage	Address in IIVCON
Job file control block (JFCB)	JFCBAREA	OS	Auxiliary storage	176 bytes	RDJFCB macro (IIVCON)
Local execution list	ALRILIST (point to DRILIST)	Emulator	Emulator area	10 words	Address in IIVCON
Logical unit block (LUB)	None	DOS	DOS supervisor area	2 byte entry for each logical unit in LUB table	Address in IIVCON and DOS communication region

Figure 29 (Part 2 of 3). Data Area Directory

Data Area Name	Beginning Symbol	Created by	Storage Area	Size	Means of Access
Open table	--	DOS	DOS B-transient area	208 bytes	Address in DOS register 15 when open phases are in control
OS PUB table	None	Emulator	Emulator area	1-byte entry for each PUB entry	Address in IIVCON
Physical unit block (PUB)	None	DOS	DOS supervisor area	8 byte entry for each device in DOS PUB table	Address in IIVCON and DOS communication region
Post ECB list	None	Emulator	Emulator area	2 byte entry for each device	Address in IIVCON
Program information block (PIB)	DOSPIB	DOS	DOS supervisor area	16 bit entry for each PIB in PIB table	Address in IIVCON and DOS communication region
Program status word (PSW)	--	DOS	Adjustment factor plus	Doubleword	DOS storage assignment plus adjustment factor
DOS SVC old PSW	DOSSVOLD		32(20)		
DOS SVC new PSW	DOSSVNEW		96(60)		
DOS program old PSW	DOSPGOLD		40(28)		
DOS program new PSW	DOSPGNEW		104(68)		
DOS I/O old PSW	DOSIOOLD		56(38)		
DOS I/O new PSW	DOSIONEW		120(78)		
DOS external old PSW	DOSXTOLD		24(18)		
DOS external new PSW	DOSXTNEW		88(58)		
DOS machine check old PSW	DOSMCOLD		48(30)		
DOS machine check new PSW	DOSMCNEW		112(70)		
Staged I/O constants block (STGCON)	STGCON	Emulator	Dummy IOB	10 words	CONTAB
Tape error block (TEB)	None	DOS	DOS supervisor area	6 bytes	PUB
Tape error by volume (TEBV)	None	DOS	DOS supervisor area	Variable	Address in DOS BG COMREG extension
Task input/output table (TIOT)	None	OS	MFT-OS problem program area MVT -- protected storage	Variable	Address in IIVCON
Unit control block (UCB)	None	OS	OS nucleus	Variable	DEB
Volume label	OBVOL1	Utility programs	Cylinder 0, track 0, record 3	80 characters	IIVCON

Figure 29 (Part 3 of 3). Data Area Directory

EMULATOR MACROS

The Emulator program makes use of several macros, most of which are DSECTs, to describe common Emulator data areas. The macros are identified and described as follows (Figure 30):

Emulator Macro	Issuing Module(s)	Function
COMTAB	IIVGR2, IIVPCE, IIVVIO IIVPRP, IIVLOG, IIVIS IIVABN, IIVCHK, IIVSNP, IIVACI	Creates a DSECT that describes one entry of the Emulator communications table.
CTEXT	IIVGR2, IIVDVS, IIVPCE, IIVVIO, IIVIS, IIVSNP	Creates a DSECT that describes a COMTAB extension entry.
DEBD	IIVPCE, IIVDVS	Creates a DSECT named IIVDEB that describes a DEB.
DLBLD	IIVGR2, IIVDVS	Creates a DSECT named DLBLDS that describes the DLBL card image.
DOSCCB	IIVPCE	Creates a DSECT that describes the DOS CCB.
DOSCOM	IIVGR2, IIVDVS, IIVPCE, IIVCHK	Creates a DSECT that describes the DOS communication region.
DOSCORE	IIVPRP, IIVLOG, IIVCHK, IIVRAS, IIVACI	Creates a DSECT that describes the first 128 bytes of DOS storage.
DOSPIB	IIVOPN	Creates a DSECT that describes the DOS PIB.
DOSPUB	IIVPRP, IIVCHK	Creates a DSECT that describes one DOS PUB table entry.
DOSREGS	All modules	Equates labels to the 16 general purpose registers.
DRILIST	All modules (contained in EMUCONS)	Creates a DSECT that describes the local execution list and the adjust CCW data address list.
DTFD	IIVGR2, IIVDVS	Creates a DSECT named DTFDS that describes the DTF.
DTFISDST	IIVIS	Creates a DSECT that describes the DTFIS tables.

Figure 30 (Part 1 of 3). Emulator Macros

Emulator Macro	Issuing Module(s)	Function
EMUCONS	All modules	Describes the Emulator constant area, which contains data constants common to most Emulator modules. EMUCONS can be used in two different ways: • If the positional parameter is omitted, it generates a CSECT called IIVCON. This CSECT is the common data area of the Emulator. The CSECT option is used by the Emulator entry routine (IIVENT). • If the positional parameter DSECT is coded, this macro generates a DSECT called EMUCONS, which describes the Emulator constant CSECT, IIVCON.
EMUMSG	IIVINT, IIVADD, IIVIN2, IIVPUB, IIVPRP, IIVABN, IIVSTG, IIVRTE, IIVPCE, IIVGET, IIVOPN, IIVGR2, IIVDVS, IIVRCP, IIVRAS, IIVRCW, IIVCHK	Provides linkage to the Emulator message writer (IIVMSG), which sends messages to the operator and programmer.
FIDBLK	IIVGR2, IIVDVS, IIVIS, IIVSNP	Creates a DSECT that describes the FID.
FMTRC	IIVSNP	Creates a DSECT that describes the formatted trace table printed when the TRACE command is invoked by IIVSNP.
ISBLK	IIVIS	Creates a DSECT that describes the ISAM block (ISBLK).
MGTXT	IIVMG1, IIVMG2, IIVMG3	Generates the message text and controls information for supplemental text.
MSGCODT	IIVMG1, IIVMG2, IIVMG3	Generates an index table to locate and process the message text.
OBTNWK	IIVDVS, IIVVIO	Creates a work area for VTOC labels used by IIVVIO and IIVDVS. If the keyword parameter DSECT is present, only a DSECT to describe the area is generated.

Figure 30 (Part 2 of 3). Emulator Macros

Emulator Macro	Issuing Module(s)	Function
RASCONS	IIVRAS, IIVRCP, IIVSNP, IIVACI, IIVPCI, IIVSCI	Describes Emulator service aids constant area, which contains data constants required by all service aids modules. RASCONS can be used in two different ways: • If the keyword parameter is omitted, it generates a CSECT called IIVRCN. This CSECT is the common data area of the service aids modules. The CSECT option is used by module IIVRAS. • If the keyword parameter DSECT is coded, this macro generates a DSECT called RASCONS, which describes the CSECT, IIVRCN.
STGTAB	IIVPUB, IIVOPN	Creates a DSECT that describes module IIVSDT.
TRCDSCT	IIVACI, IIVPCI, IIVSCI, IIVSNP	Creates a DSECT that describes the internal trace table format (see figure entitled "Internal Trace Table Format").

Figure 30 (Part 3 of 3). Emulator Macros

SYMBOL TABLE

The symbols (or labels) in Figure 31 are contained in flowcharts in the "Program Organization" section of this publication. The list can be helpful when program listings are being used. If there is a symbol at, or nearby the instructions in question, refer to this list to locate a flowchart, a routine name, or a module name that could either answer your question or provide a context that might lead to an answer.

Users of this list should be aware of its limitations; in particular, the flowcharts do not contain all labels to be found in the program listings. A full description of the characteristics of the flowcharts appears in the "Program Organization" section.

Symbol	Chart	Routine	Module
ABSNAP	26A	Exit-ABEND error routine	IIVABN
ADDCHK	4C	IPL add routine	IIVADD
ADDCHK20	4C	IPL add routine	IIVADD
ADDPUB	4D	IPL add routine	IIVADD
ADDSTG	4D	IPL add routine	IIVADD
ADD2400	4D	IPL add routine	IIVADD
ADD3400	4D	IPL add routine	IIVADD
ADJ01	16A	CCW adjustment routine	IIVADJ
ADJ02	16A	CCW adjustment routine	IIVADJ
ADJ03	16A	CCW adjustment routine	IIVADJ
ADJ04	16A	CCW adjustment routine	IIVADJ
ADJ05	16A	CCW adjustment routine	IIVADJ
ADJ06	16B	CCW adjustment routine	IIVADJ
ADJ07	16B	CCW adjustment routine	IIVADJ
ADJ08	16B	CCW adjustment routine	IIVADJ
ADJ09	16B	CCW adjustment routine	IIVADJ
ADJ10	16B	CCW adjustment routine	IIVADJ
ADJ11	16B	CCW adjustment routine	IIVADJ
AERT000	18B	Abnormal end appendage	IGG019SA
ASCHCK04	20E	Asynchronous interrupt check subroutine	IIVRTE
ASCHCK07	20E	Asynchronous interrupt check subroutine	IIVRTE
ASCHCK08	20E	Asynchronous interrupt check subroutine	IIVRTE
ASCHCK09	20E	Asynchronous interrupt check subroutine	IIVRTE
ASCHCK10	20E	Asynchronous interrupt check subroutine	IIVRTE
ASKOPR	2L	ASKOPR subroutine	IIVINT
BASELEG0	8R	FINDADDR subroutine	IIVPCE
BISMSTAT	24L	EODAD, SYNAD, and status mapping routines	IIVIS
BLDISK	3D	Initialization second-load routine	IIVIN2
CERT000	18B	Channel end appendage	IGG019SA
CHECKERR	9B	Check I/O routine	IIVCHK
CHECKNDE	9D	Check I/O routine	IIVCHK
CHECKSDE	9A	Check I/O routine	IIVCHK
CHECK000	9A	Check I/O routine	IIVCHK
CHECK001	9A	Check I/O routine	IIVCHK
CHECK002	9A	Check I/O routine	IIVCHK
CHECK003	9B	Check I/O routine	IIVCHK
CHECK004	9C	Check I/O routine	IIVCHK
CHECK040	9D	Check I/O routine	IIVCHK
CHECK050	9D	Check I/O routine	IIVCHK
CHECK090	9D	Check I/O routine	IIVCHK
CHECK095	9D	Check I/O routine	IIVCHK
CHECK1	20C	Select routine	IIVRTE
CHECK100	9D	Check I/O routine	IIVCHK
CHECK110	9D	Check I/O routine	IIVCHK
CHECK120	9D	Check I/O routine	IIVCHK
CHKCUU	2L	CHKCUU subroutine	IIVINT
CHKEOJ	2C	Initialization first-load routine	IIVINT
CHKIPL	2B	Initialization first-load routine	IIVINT
CHKLOG	2B	Initialization first-load routine	IIVINT
CHKPMT	2C	Initialization first-load routine	IIVINT
CHKRES	2B	Initialization first-load routine	IIVINT
CHKSIZ	2C	Initialization first-load routine	IIVINT
CHKTIM	2C	Initialization first-load routine	IIVINT
CHNCTE	3B	Initialization second-load routine	IIVIN2
CHNCTE09	3B	Initialization second-load routine	IIVIN2
CHNCTE10	3B	Initialization second-load routine	IIVIN2
CHNCTE25	3B	Initialization second-load routine	IIVIN2
CHNCTE30	3B	Initialization second-load routine	IIVIN2
CHNCTE40	3B	Initialization second-load routine	IIVIN2
CLOSE	23E	Device sharing simulation routine	IIVDVS
CLOSERTN	24C	Close mapping routine	IIVIS

Figure 31 (Part 1 of 15). Symbol Table

Symbol	Chart	Routine	Module
CLOSE0	23E	Device sharing simulation routine	IIVDVS
CLOSE01	24C	Close mapping routine	IIVIS
CLOSE1	23E	Device sharing simulation routine	IIVDVS
CLOSE1A	23E	Device sharing simulation routine	IIVDVS
CLOSE2	23E	Device sharing simulation routine	IIVDVS
CLOSE3	23E	Device sharing simulation routine	IIVDVS
CLOSOPEN	24L	EODAD, SYNAD, and status mapping routines	IIVIS
CLS00000	22C	SVC monitor routine	IIVGR2
CLS00050	22C	SVC monitor routine	IIVGR2
CLS00200	22C	SVC monitor routine	IIVGR2
CMD00	29A	Command processor routine	IIVRCP
CMNDREJ	11K	Staged I/O routine	IIVSTG
CNTENT10	2E	Initialization first-load routine	IIVINT
COMBINE	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB01	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB02	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB03	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB04	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB04A	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB05	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB06	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB07	15F	Combine subroutine	IIVCCW
	34H	Service aids adjust CCW data address routine	IIVRCW
COMB08	15F	Adjust CCW data address routine	IIVCCW
	34H	Service aids combine subroutine	IIVRCW
CSWSTOR	11J	Staged I/O routine	IIVSTG
CTLRTN	8T	Load/store control register subroutine	IIVPCE
CVTHEX	29D	CVT subroutine	IIVRCP
CVTMSK	29D	CVT subroutine	IIVRCP
CVTNUM	29D	CVT subroutine	IIVRCP
CVTUPP	29D	CVT subroutine	IIVRCP
CVT00	29D	CVT subroutine	IIVRCP
CVT02	29D	CVT subroutine	IIVRCP
CVT11	29D	CVT subroutine	IIVRCP
CVT18	29D	CVT subroutine	IIVRCP
CVT20	29D	CVT subroutine	IIVRCP
CVT30	29D	CVT subroutine	IIVRCP
CVT40	29D	CVT subroutine	IIVRCP
CXRET	24A	Main task control executive routine	IIVIS
DDSCAN	2M	DDSCAN subroutine	IIVINT
DDSCAN10	2M	DDSCAN subroutine	IIVINT
DDSCAN20	2M	DDSCAN subroutine	IIVINT
DDSCAN22	2M	DDSCAN subroutine	IIVINT
DDSCAN24	2M	DDSCAN subroutine	IIVINT

Figure 31 (Part 2 of 15). Symbol Table

Symbol	Chart	Routine	Module
DDSCAN25	2M	DDSCAN subroutine	IIVINT
DDSCAN30	2M	DDSCAN subroutine	IIVINT
DELFF	29B	Command processor routine	IIVRCP
DELPUB	4B	IPL add routine	IIVADD
DELPUB40	4B	IPL add routine	IIVADD
DEL00	29B	Command processor routine	IIVRCP
DIAGCCW	29L	Diagnostic subroutines	IIVRCP
DIAGC20	29L	Diagnostic subroutines	IIVRCP
DIAGC21	29L	Diagnostic subroutines	IIVRCP
DIAGINIT	29L	Diagnostic subroutines	IIVRCP
DIAGINV	29L	Diagnostic subroutines	IIVRCP
DLBL	23H	Device sharing simulation routine	IIVDVS
DLBL0	23H	Device sharing simulation routine	IIVDVS
DLBL0A	23H	Device sharing simulation routine	IIVDVS
DLBL1	23H	Device sharing simulation routine	IIVDVS
DLBL10	23H	Device sharing simulation routine	IIVDVS
DLBL2	23H	Device sharing simulation routine	IIVDVS
DOSPUB10	4D	IPL add routine	IIVADD
DVSMSG	23H	Device sharing simulation routine	IIVDVS
ECBSCN	20F	Asynchronous interrupt check subroutine	IIVRTE
ECBSCN01	20F	Asynchronous interrupt check subroutine	IIVRTE
ECBSCN04	20F	Asynchronous interrupt check subroutine	IIVRTE
ECBSCN08	20F	Asynchronous interrupt check subroutine	IIVRTE
ECBSCN10	20F	Asynchronous interrupt check subroutine	IIVRTE
ENDINIT	29N	End subroutine	IIVRCP
ENDLOAD	29P	End subroutine	IIVRCP
END01	29N	End subroutine	IIVRCP
END02	29N	End subroutine	IIVRCP
END09	29N	End subroutine	IIVRCP
END10	29N	End subroutine	IIVRCP
END20	29N	End subroutine	IIVRCP
ENQDVC	3D	Initialization second-load routine	IIVIN2
ENQDVC20	3D	Initialization second-load routine	IIVIN2
EODA	11N	EODAD subroutine	IIVSTG
EODADRTN	24L	EODAD routine	IIVIS
EOERT000	18A	End-of-extent appendage	IGG019SA
EOX	23C	Device sharing simulation routine	IIVDVS
EOX0A	23C	Device sharing simulation routine	IIVDVS
EOX0B	23C	Device sharing simulation routine	IIVDVS
EOX1A	23C	Device sharing simulation routine	IIVDVS
EOX10	23C	Device sharing simulation routine	IIVDVS
EOX2	23C	Device sharing simulation routine	IIVDVS
EOX2A	23C	Device sharing simulation routine	IIVDVS
EOX20	23C	Device sharing simulation routine	IIVDVS
EOX20A	23C	Device sharing simulation routine	IIVDVS
EOX21	23C	Device sharing simulation routine	IIVDVS
EOX22	23C	Device sharing simulation routine	IIVDVS
EOX3E	23D	Device sharing simulation routine	IIVDVS
EOX30	23C	Device sharing simulation routine	IIVDVS
EOX30C	23D	Device sharing simulation routine	IIVDVS
EOX31	23D	Device sharing simulation routine	IIVDVS
EOX32	23D	Device sharing simulation routine	IIVDVS
EOX33	23D	Device sharing simulation routine	IIVDVS
EOX33A	23D	Device sharing simulation routine	IIVDVS
EOX34	23D	Device sharing simulation routine	IIVDVS
EOX35	23D	Device sharing simulation routine	IIVDVS
EOX36	23D	Device sharing simulation routine	IIVDVS
ERR1	26A	Exit-ABEND error routine	IIVABN
ERR3	26A	Exit-ABEND error routine	IIVABN
ERR4	26A	Exit-ABEND error routine	IIVABN
ERR5	26A	Exit-ABEND error routine	IIVABN

Figure 31 (Part 3 of 15). Symbol Table

Symbol	Chart	Routine	Module
ERR6	26A	Exit-ABEND error routine	IIVABN
ESETLRTN	24G	ESETL mapping routine	IIVIS
EXIAS	29M	Exit subroutines	IIVRCP
EXIINIT	29M	Exit subroutines	IIVRCP
EXIINV	29M	Exit subroutines	IIVRCP
EXILEX	29M	Exit subroutines	IIVRCP
EXIPC	29M	Exit subroutines	IIVRCP
EXISVC	29M	Exit subroutines	IIVRCP
FETCH	14A	Printer overflow routine	IIVPOV
FINDADDR	8R	FINDADDR subroutine	IIVPCE
FINDAD0	8R	FINDADDR subroutine	IIVPCE
FINDCHAN	8Q	FINDCHAN subroutine	IIVPCE
FINDKEY	8S	FINDKEY subroutine	IIVPCE
FIRSTPC	8A	Program check executive routine	IIVPCE
GETDOS	2D	Initialization first-load routine	IIVINT
GETECB	3D	Initialization second-load routine	IIVIN2
GETIOREG	24F	Get mapping routine	IIVIS
GETPEC	3D	Initialization second-load routine	IIVIN2
GETRTN	24F	Get mapping routine	IIVIS
GETWORD	29C	GETWORD subroutine	IIVRCP
GETW0	29C	GETWORD subroutine	IIVRCP
GETW1	29C	GETWORD subroutine	IIVRCP
GR11	22A	SVC monitor routine	IIVGR2
GR2CLS	22C	SVC monitor routine	IIVGR2
GR2DPV	22D	SVC monitor routine	IIVGR2
GR2DPV1	22D	SVC monitor routine	IIVGR2
GR2DPV3	22D	SVC monitor routine	IIVGR2
GR2EOJ	22C	SVC monitor routine	IIVGR2
GR2EOX	22C	SVC monitor routine	IIVGR2
GR2OPN	22B	SVC monitor routine	IIVGR2
GR200010	22A	SVC monitor routine	IIVGR2
GR200012	22A	SVC monitor routine	IIVGR2
GR200014	22A	SVC monitor routine	IIVGR2
GR200016	22A	SVC monitor routine	IIVGR2
GR200050	22A	SVC monitor routine	IIVGR2
GR200059	22A	SVC monitor routine	IIVGR2
GR290000	22C	SVC monitor routine	IIVGR2
GR290200	22C	SVC monitor routine	IIVGR2
GR290201	22C	SVC monitor routine	IIVGR2
GR290202	22C	SVC monitor routine	IIVGR2
GR299999	22A	SVC monitor routine	IIVGR2
GR400000	22A	SVC monitor routine	IIVGR2
HIORTN	8H	HIO simulation subroutine	IIVPCE
HIOTST	8H	HIO simulation subroutine	IIVPCE
IDCOMP	2A	Initialization first-load routine	IIVINT
IGG019SA	18A	Start I/O appendage	IGG019SA
IGG019S1	19A	Abnormal end/channel end appendage	IGG019S1
IIVABN	26A	Exit-ABEND error routine	IIVABN
IIVACI	33A	Asynchronous intercept routine	IIVACI
IIVADD	4A	IPL add routine	IIVADD
IIVADJ	16A	CCW adjustment routine	IIVADJ
IIVAWV	12A	CAW verification routine	IIVAWV
IIVCCW	15A	Adjust CCW data address routine	IIVCCW
IIVCHK	9A	Check I/O routine	IIVCHK
IIVCWV	13A	CCW verification routine	IIVCWV
IIVDVS	23A	Device sharing simulation routine	IIVDVS
IIVENT	1A	DOS Emulator entry routine	IIVENT
IIVGET	7A	GETMAIN/FREEMAIN routine	IIVGET
IIVGR2	22A	SVC monitor routine	IIVGR2
IIVINT	2A	Initialization first-load routine	IIVINT
IIVIN2	3A	Initialization second-load routine	IIVIN2
IIVIS	24A	Main task control executive	IIVIS

Figure 31 (Part 4 of 15). Symbol Table

Symbol	Chart	Routine	Module
IIVIS01	24E	Subtask control executive	IIVIS
IIVLOG	10A	Interpretive SYSLOG routine	IIVLOG
IIVLOGR1	10C	IIVLOGR1 subroutine	IIVLOG
IIVLOGR2	10E	IIVLOGR2 subroutine	IIVLOG
IIVMSG	27A	Message writer routine	IIVMSG
IIVMSG15	27A	Message writer routine	IIVMSG
IIVMSG20	27A	Message writer routine	IIVMSG
IIVMSG25	27A	Message writer routine	IIVMSG
IIVMSG45	27A	Message writer routine	IIVMSG
IIVMSG60	27A	Message writer routine	IIVMSG
IIVMSG65	27A	Message writer routine	IIVMSG
IIVMSG80	27A	Message writer routine	IIVMSG
IIVMSG90	27A	Message writer routine	IIVMSG
IIVOPN	5A	Open routine	IIVOPN
IIVPCE	8A	Program check executive routine	IIVPCE
IIVPCI	31A	Program check intercept routine	IIVPCI
IIVPOV	14A	Printer overflow routine	IIVPOV
IIVPRP	21A	Prompt reply processor routine	IIVPRP
IIVPUB	6A	OS PUB table build routine	IIVPUB
IIVRAS	28A	Service aids initialization routine	IIVRAS
IIVRASPC	28B	IIVRASPC subroutine	IIVRAS
IIVRASVC	28B	IIVRASVC subroutine	IIVRAS
IIVRASYN	28B	IIVRASYN subroutine	IIVRAS
IIVRCP	29A	Command processor routine	IIVRCP
IIVRCW	34A	Service aids adjust CCW data address routine	IIVRCW
IIVRTE	20A	Asynchronous interrupt exit routine	IIVRTE
IIVRTECK	20E	Asynchronous interrupt check subroutine	IIVRTE
IIVRTEOJ	20G	End-of-job routine	IIVRTE
IIVRTER1	20A	Asynchronous interrupt exit routine	IIVRTE
IIVRTER2	20B	Route routine	IIVRTE
IIVRTESL	20C	Select routine	IIVRTE
IIVRTESR	20C	STAE retry routine	IIVRTE
IIVRTEST	20A	STAE exit routine	IIVRTE
IIVRTETM	20D	Timer interrupt subroutine	IIVRTE
IIVSCI	32A	Supervisor call intercept routine	IIVSCI
IIVSNP	30A	Snap dump and trace formatting routine	IIVSNP
IIVSTG	11A	Staged I/O routine	IIVSTG
IIVSVC	17A	Supervisor call routine	IIVSVC
IIVVIO	25A	VTOC I/O simulation routine	IIVVIO
INITCT	2F	Initialization first-load routine	IIVINT
INITCT13	2F	Initialization first -load routine	IIVINT
INITCT15	2F	Initialization first-load routine	IIVINT
INITCT20	2G	Initialization first-load routine	IIVINT
INITCT23	2G	Initialization first-load routine	IIVINT
INITCT30	2G	Initialization first-load routine	IIVINT
INITCT35	2G	Initialization first-load routine	IIVINT
INITCT5A	2H	Initialization first-load routine	IIVINT
INITCT5D	2H	Initialization first-load rouine	IIVINT
INITCT51	2H	Initialization first-load routine	IIVINT
INITCT52	2H	Initialization first-load routine	IIVINT
INITCT54	2H	Initialization first-load routine	IIVINT
INITCT60	2H	Initialization first-load routine	IIVINT
INIT30	4A	IPL add routine	IIVADD
INTPEND	11J	Staged I/O routine	IIVSTG
INTRPT	8D	LPSW simulation subroutine	IIVPCE
INTRPTEX	8D	LPSW simulation subroutine	IIVPCE
IOBCHK30	6B	OS PUB table build routine	IIVPUB
IOBCHK60	6B	OS PUB table build routine	IIVPUB
IOBCHK65	6B	OS PUB table build routine	IIVPUB
IOBCHK66	6B	OS PUB table build routine	IIVPUB
IOBCHK70	6E	OS PUB table build routine	IIVPUB

Figure 31 (Part 5 of 15). Symbol Table

Symbol	Chart	Routine	Module
IOBCHK75	6E	OS PUB table build routine	IIVPUB
IOBCHK77	6E	OS PUB table build routine	IIVPUB
IPLDOS	3E	Initialization second-load routine	IIVIN2
ISENDFL	22D	SVC monitor routine	IIVGR2
ISKRTN	8C	ISK simulation subroutine	IIVPCE
ISSETFL	22D	SVC monitor routine	IIVGR2
ISSETL	22D	SVC monitor routine	IIVGR2
ISSMULAT	22D	SVC monitor routine	IIVGR2
ISWAIT	24E	Subtask control executive routine	IIVIS
KEYERR	8S	FINDKEY subroutine	IIVPCE
KEY00	29A	Command processor routine	IIVRCP
LEAVE	19A	Abnormal end/channel end appendage	IGG019S1
LEAVE1	19A	Abnormal end/channel appendage	IGG019S1
LOADSTAT	24L	EODAD, SYNAD, and status mapping routines	IIVIS
LOAD00	24B	Open mapping routine	IIVIS
LOAD11	24B	Open mapping routine	IIVIS
LOGIO1	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO10	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO11	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO12	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO13	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO14	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO15	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO16	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO17	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO18	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO19	10B	Interpretive SYSLOG routine	IIVLOG
LOGIO2	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO20	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO21	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO22	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO25	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO26	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO27	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO29	10C	IIVLOGR1 subroutine	IIVLOG
LOGIO3	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO30	10D	LOGOUT1 and LOGOUT2 subroutines	IIVLOG
LOGIO31	10D	LOGOUT1 and LOGOUT2 subroutines	IIVLOG
LOGIO4	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO40	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO5	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO5A	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO50	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO6	10A	Interpretive SYSLOG routine	IIVLOG
LOGIO60	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO61	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO62	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO63	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO64	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO65	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO66	10E	IIVLOGR2 subroutine	IIVLOG
LOGIO67	10E	IIVLOGR2 subroutine	IIVLOG
LOGOUT1	10D	LOGOUT1 subroutine	IIVLOG
LOGOUT2	10D	LOGOUT2 subroutine	IIVLOG
LPSWRTN	8D	LPSW simulation subroutine	IIVPCE
MAPDTFIS	24B	Open mapping routine	IIVIS
MAPSTSTC	24L	Status mapping routine	IIVIS
OBTAIN	23F	Device sharing simulation routine	IIVDVS
OBTAIN0	23F	Device sharing simulation routine	IIVDVS
OBTAIN1	23F	Device sharing simulation routine	IIVDVS
OBTAIN2	23F	Device sharing simulation routine	IIVDVS

Figure 31 (Part 6 of 15). Symbol Table

Symbol	Chart	Routine	Module
OBTAIN3	23F	Device sharing simulation routine	IIVDVS
OBTAIN4	23G	Device sharing simulation routine	IIVDVS
OBTAIN4A	23G	Device sharing simulation routine	IIVDVS
OBTAIN4B	23G	Device sharing simulation routine	IIVDVS
OBTAIN4C	23G	Device sharing simulation routine	IIVDVS
OBTN0	23F	Device sharing simulation routine	IIVDVS
OBTN00	23F	Device sharing simulation routine	IIVDVS
OBTN04	23F	Device sharing simulation routine	IIVDVS
OBTN05	23F	Device sharing simulation routine	IIVDVS
OBTN4	23F	Device sharing simulation routine	IIVDVS
OBTN5	23F	Device sharing simulation routine	IIVDVS
OPEN	23A	Device sharing simulation routine	IIVDVS
OPENATCH	24B	Open mapping routine	IIVIS
OPENAT01	24B	Open mapping routine	IIVIS
OPENRTN	24B	Open mapping routine	IIVIS
OPEN0	23A	Device sharing simulation routine	IIVDVS
OPEN0A	23A	Device sharing simulation routine	IIVDVS
OPEN000	24B	Open mapping routine	IIVIS
OPEN002	24B	Open mapping routine	IIVIS
OPEN003	24B	Open mapping routine	IIVIS
OPEN006	24B	Open mapping routine	IIVIS
OPEN10	5A	Open routine	IIVOPN
OPEN12	5A	Open routine	IIVOPN
OPEN2	23A	Device sharing simulation routine	IIVDVS
OPEN2C	23A	Device sharing simulation routine	IIVDVS
OPEN20	5A	Open routine	IIVOPN
OPEN25	5A	Open routine	IIVOPN
OPEN3	23B	Device sharing simulation routine	IIVDVS
OPEN30	5B	Open routine	IIVOPN
OPEN37T	5C	Open routine	IIVOPN
OPEN4	23B	Device sharing simulation routine	IIVDVS
OPEN40	5A	Open routine	IIVOPN
OPEN4B	23B	Device sharing simulation routine	IIVDVS
OPEN5	23B	Device sharing simulation routine	IIVDVS
OPEN6	23B	Device sharing simulation routine	IIVDVS
OPEN60	5D	OPEN60 routine	IIVOPN
OPEN70	5C	Open routine	IIVOPN
OPEN80	5C	Open routine	IIVOPN
OPEN95	5C	Open routine	IIVOPN
OPN00200	22B	SVC monitor routine	IIVGR2
OPN00800	22B	SVC monitor routine	IIVGR2
OPN00950	22B	SVC monitor routine	IIVGR2
OPRSUB	29A	Command processor routine	IIVRCP
OPR00	29A	Command processor routine	IIVRCP
PC	8A	Program check executive routine	IIVPCE
PCE500	31B	Program check intercept routine	IIVPCI
PCE550	31C	Program check intercept routine	IIVPCI
PCE925	31C	Program check intercept routine	IIVPCI
PCPRIVOP	8A	Program check executive routine	IIVPCE
PC00	8A	Program check executive routine	IIVPCE
PC10	8A	Program check executive routine	IIVPCE
PC20	8A	Program check executive routine	IIVPCE
PERMERR	19A	Abnormal end/channel end appendage	IGG019S1
PEPMERR1	19A	Abnormal end/channel appendage	IGG019S1
POSTCSW	9C	Check I/O routine	IIVCHK
POSTCSW1	9C	Check I/O routine	IIVCHK
POSTCSW2	9C	Check I/O routine	IIVCHK
PRPMAPA	21E	PRPMAPA subroutine	IIVPRP
PRPMAPB	21E	PRPMAPA subroutine	IIVPRP
PRPMAPC	21E	PRPMAPA subroutine	IIVPRP
PRPMAPD	21E	PRPMAPA subroutine	IIVPRP
PRPMAP1	21E	PRPMAP1 subroutine	IIVPRP

Figure 31 (Part 7 of 15). Symbol Table

Symbol	Chart	Routine	Module
PRPMAP2	21E	PRPMAP1 subroutine	IIVPRP
PRPMAP3	21E	PRPMAP1 subroutine	IIVPRP
PRPMAP4	21E	PRPMAP1 subroutine	IIVPRP
PRPMAP5	21E	PRPMAP1 subroutine	IIVPRP
PRP01	21A	Prompt reply processor routine	IIVPRP
PRP02	21A	Prompt reply processor routine	IIVPRP
PRP03	21A	Prompt reply processor routine	IIVPRP
PRP04	21A	Prompt reply processor routine	IIVPRP
PRP05	21A	Prompt reply processor routine	IIVPRP
PRP06	21A	Prompt reply processor routine	IIVPRP
PRP07	21A	Prompt reply processor routine	IIVPRP
PRP10	21A	Prompt reply processor routine	IIVPRP
PRP20	21A	Prompt reply processor routine	IIVPRP
PRP30	21A	Prompt reply processor routine	IIVPRP
PRP40	21B	Prompt reply processor routine	IIVPRP
PRP41	21B	Prompt reply processor routine	IIVPRP
PRP42	21B	Prompt reply processor routine	IIVPRP
PRP44	21B	Prompt reply processor routine	IIVPRP
PRP45	21B	Prompt reply processor routine	IIVPRP
PRP46	21B	Prompt reply processor routine	IIVPRP
PRP50	21B	Prompt reply processor routine	IIVPRP
PRP51	21B	Prompt reply processor routine	IIVPRP
PRP52	21B	Prompt reply processor routine	IIVPRP
PRP59	21B	Prompt reply processor routine	IIVPRP
PRP60	21C	Prompt reply processor routine	IIVPRP
PRP62	21C	Prompt reply processor routine	IIVPRP
PRP63	21C	Prompt reply processor routine	IIVPRP
PRP64	21C	Prompt reply processor routine	IIVPRP
PRP65	21C	Prompt reply processor routine	IIVPRP
PRP66	21C	Prompt reply processor routine	IIVPRP
PRP67	21C	Prompt reply processor routine	IIVPRP
PRP67A	21C	Prompt reply processor routine	IIVPRP
PRP68	21C	Prompt reply processor routine	IIVPRP
PRP80	21D	Prompt reply processor routine	IIVPRP
PRP82	21D	Prompt reply processor routine	IIVPRP
PUTBLKDR	24G	Put mapping routine	IIVIS
PUTNOWKS	24G	Put mapping routine	IIVIS
PUTRTN	24G	Put mapping routine	IIVIS
QISMSTAT	24L	EODAD, SYNAD and status mapping routines	IIVIS
RAS14000	30E	Formatting subroutines	IIVSNP
RAS14228	30G	Trace table subroutine	IIVSNP
RAS15000	30H	Print subroutines	IIVSNP
RAS15200	30J	Print subroutines	IIVSNP
RAS16000	30K	Print multiple line subroutine	IIVSNP
RAS17000	30G	Formatting subroutines	IIVSNP
RAS18000	30J	Print subroutines	IIVSNP
RCCWAB	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCWAB1	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCWAB2	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCWAB3	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCWAB4	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW

Figure 31 (Part 8 of 15). Symbol Table

Symbol	Chart	Routine	Module
RCCWD02	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD20	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD22	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD30	34E	Service aids adjust CCW data address routine	IIVRCW
RCCW02	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCW03	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCW04	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCWD25	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD26	34D	Service aids adjust CCW data address routine	IIVRCN
RCCWD26	34D	Service aids adjust CCW data address routine	IIVRCN
RCCWD28	34D	Service aids adjust CCW data address routine	IIVRCN
RCCWD99	34D	Service aids adjust CCW data address routine	IIVRCN
RCCW05	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCW05A	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCW05B	15A	Adjust CCW data address routine	IIVCCW
	34A	Service aids adjust CCW data address routine	IIVRCW
RCCW06A	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW07	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW08	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW08A	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW08B	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW08C	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW09	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW10	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW

Figure 31 (Part 9 of 15). Symbol Table

Symbol	Chart	Routine	Module
RCCW11	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW12	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW13	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW13A	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW13B	15B	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW14	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW15	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW16	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW17	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW17A	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW17B	15C	Adjust CCW data address routine	IIVCCW
	34B	Service aids adjust CCW data address routine	IIVRCW
RCCW17C	15C	Adjust CCW data address routine	IIVCCW
	34C	Service aids adjust CCW data address routine	IIVRCW
RCCW18	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW18A	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW19	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW19A	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW20	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW20A	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW21	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW

Figure 31 (Part 10 of 15). Symbol Table

Symbol	Chart	Routine	Module
RCCW21A	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW21B	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW21C	15D	Adjust CCW data address routine	IIVCCW
	34F	Service aids adjust CCW data address routine	IIVRCW
RCCW22A	15E	Adjust CCW data address routine	IIVCCW
	34G	Service aids adjust CCW data address routine	IIVRCW
RCCW22B	15E	Adjust CCW data address routine	IIVCCW
	34G	Service aids adjust CCW data address routine	IIVRCW
RCCW23	15E	Adjust CCW data address routine	IIVCCW
	34G	Service aids adjust CCW data address routine	IIVRCW
RCCW24	15E	Adjust CCW data address routine	IIVCCW
	34G	Service aids adjust CCW data address routine	IIVRCW
RCCWD25	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD26	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD28	34D	Service aids adjust CCW data address routine	IIVRCW
RCCWD99	34D	Service aids adjust CCW data address routine	IIVRCW
RCPPRINT	29D	RCPPRINT subroutine	IIVRCP
RDJFCB	2F	Initialization first-load routine	IIVINT
RDNOWORK	24K	WAITF mapping routine	IIVIS
READRTN	24H	Read key mapping routine	IIVIS
READ100	29C	GETWORD subroutine	IIVRCP
READ999	29A	Command processor routine	IIVRCP
RESET	23H	Device sharing simulation routine	IIVDVS
RETURN	23H	Device sharing simulation routine	IIVDVS
	29B	Command processor routine	IIVRCP
RETURN1	23H	Device sharing simulation routine	IIVDVS
RMSRESET	22D	SVC monitor routine	IIVGR2
ROUTE05	20A	Asynchronous interrupt exit routine	IIVRTE
	20B	Route routine	IIVRTE
ROUTE10	20A	Asynchronous interrupt exit routine	IIVRTE
	20B	Route routine	IIVRTE
ROUTE11	20B	Route routine	IIVRTE
ROUTE11A	20B	Route routine	IIVRTE
ROUTE12	20B	Route routine	IIVRTE
ROUTE15	20A	Asynchronous interrupt exit routine	IIVRTE
	20B	Route routine	
ROUTE17	20A	Asynchronous interrupt exit routine	IIVRTE
	20B	Route routine	IIVRTE
ROUTE22	20B	Route routine	IIVRTE
RSTOR000	18B	Channel end appendage	IGG019SA
RSTOR005	18B	Channel end appendage	IGG019SA
RTER2060	33B	Asynchronous intercept routine	IIVACI
RTER2100	33B	Asynchronous intercept routine	IIVACI
RTER2110	33C	Asynchronous intercept routine	IIVACI
RTER2125	33C	Asynchronous intercept routine	IIVACI
RTER2135	33D	Asynchronous intercept routine	IIVACI
RTER2200	33E	Asynchronous intercept routine	IIVACI
RTER2300	33F	Asynchronous intercept routine	IIVACI
RTER2315	33G	Asynchronous intercept routine	IIVACI

Figure 31 (Part 11 of 15). Symbol Table

Symbol	Chart	Routine	Module
RTER2325	33G	Asynchronous intercept routine	IIVACI
RTER2999	33H	Asynchronous intercept routine	IIVACI
RTRN	26A	Exit-ABEND error routine	IIVABN
RWUCHK	9C	Check I/O routine	IIVCHK
SCAN	2K	Scan subroutine	IIVINT
SCANFID	22D	SVC monitor routine	IIVGR2
SCKRTN	8U	Set clock subroutine	IIVPCE
SEEKCHK	8L	SIO subroutine	IIVPCE
SEEKDVS	8P	SEEKDVS subroutine	IIVPCE
SEEKTEST	8K	SEEKTEST subroutine	IIVPCE
SEEK0300	8P	SEEKDVS subroutine	IIVPCE
SEEK0350	8P	SEEKDVS subroutine	IIVPCE
SEEK0400	8P	SEEKDVS subroutine	IIVPCE
SEEK0500	8P	SEEKDVS subroutine	IIVPCE
SEEK0501	8P	SEEKDVS subroutine	IIVPCE
SEEK0504	8P	SEEKDVS subroutine	IIVPCE
SEEK0509	8P	SEEKDVS subroutine	IIVPCE
SEEK0510	8P	SEEKDVS subroutine	IIVPCE
SELINT1	20C	Select routine	IIVRTE
SELINT2	20C	Select routine	IIVRTE
SELINT3	20C	Select routine	IIVRTE
SELINT4	20C	Select routine	IIVRTE
SENSE	11K	Staged I/O routine	IIVSTG
SETBITS	19A	Abnormal end/channel end appendage	IGG019S1
SETDAT	4C	IPL add routine	IIVADD
SETEXT0	23B	Device sharing simulation routine	IIVDVS
SETFDAD	23A	Device sharing simulation routine	IIVDVS
SETLRTN	24D	SETL mapping routine	IIVIS
SETRTRN	24L	EODAD, SYNAD, and status mapping routines	IIVIS
SETSKCHK	8L	SIO subroutine	IIVPCE
SETSVC50	24L	EODAD, SYNAD, and status mapping routines	IIVIS
SIOBUSY	8J	SIO subroutine	IIVPCE
SIOCNT	8L	SIO subroutine	IIVPCE
SIOINS00	18A	Start I/O appendage	IGG019SA
SIOIN05	18A	Start I/O appendage	IGG019SA
SIOIN06	18A	Start I/O appendage	IGG019SA
SIOIN100	18A	Start I/O appendage	IGG019SA
SIOIN105	18A	Start I/O appendage	IGG019SA
SIOIN110	18A	Start I/O appendage	IGG019SA
SIOPENED	8J	SIO subroutine	IIVPCE
SIORTN	8J	SIO subroutine	IIVPCE
SIORT000	18B	Start I/O appendage	IGG019SA
SIORT010	18B	Start I/O appendage	IGG019SA
SIORT015	18B	Start I/O appendage	IGG019SA
SIORT020	18B	Start I/O appendage	IGG019SA
SIORT030	18B	Start I/O appendage	IGG019SA
SIORT500	18A	Start I/O appendage	IGG019SA
SIOSASK	8L	SIO subroutine	IIVPCE
SIOSN	8N	SIO subroutine	IIVPCE
SIOSN0	8N	SIO subroutine	IIVPCE
SIOSN1	8N	SIO subroutine	IIVPCE
SIOSN2	8N	SIO subroutine	IIVPCE
SIOSN3	8N	SIO subroutine	IIVPCE
SIOSN4	8N	SIO subroutine	IIVPCE
SIOSN6	8N	SIO subroutine	IIVPCE
SIOSN7	8N	SIO subroutine	IIVPCE
SIOSN7A	8N	SIO subroutine	IIVPCE
SIOSTART	8M	SIO subroutine	IIVPCE
SIOXDASD	8K	SIO subroutine	IIVPCE
SIOXIOB	8K	SIO subroutine	IIVPCE

Figure 31 (Part 12 of 15). Symbol Table

Symbol	Chart	Routine	Module
SIOXTAPE	8M	SIO subroutine	IIVPCE
SIO10	8M	SIO subroutine	IIVPCE
SIO2	8J	SIO subroutine	IIVPCE
SIO3	8J	SIO subroutine	IIVPCE
SIO4	8J	SIO subroutine	IIVPCE
SIO5	8K	SIO subroutine	IIVPCE
SKIP	14A	Printer overflow routine	IIVPOV
SNPALL	29J	Snap subroutines	IIVRCP
SNPATTN	29H	Snap subroutines	IIVRCP
SNPCOMP	29H	Snap subroutines	IIVRCP
SNPCPSUB	29H	Snap subroutines	IIVRCP
SNPCP40	29H	Snap subroutines	IIVRCP
SNPCUU	29J	Snap subroutines	IIVRCP
SNPEXT	29H	Snap subroutines	IIVRCP
SNPHIO	29J	Snap subroutines	IIVRCP
SNPINIT	29G	Snap subroutines	IIVRCP
SNPINT	29H	Snap subroutines	IIVRCP
SNPINV	29G	Snap subroutines	IIVRCP
SNPIO	29J	Snap subroutines	IIVRCP
SNPISK	29J	Snap subroutines	IIVRCP
SNPLPSW	29J	Snap subroutines	IIVRCP
SNPPC	29H	Snap subroutines	IIVRCP
SNPPSW	29G	Snap subroutines	IIVRCP
SNPSIO	29J	Snap subroutines	IIVRCP
SNPSSK	29J	Snap subroutines	IIVRCP
SNPSSM	29J	Snap subroutines	IIVRCP
SNPSUBOP	29G	Snap subroutines	IIVRCP
SNPSVC	29G	Snap subroutines	IIVRCP
SNPSV10	29G	Snap subroutines	IIVRCP
SNPSV11	29G	Snap subroutines	IIVRCP
SNPSV20	29G	Snap subroutines	IIVRCP
SNPSV30	29G	Snap subroutines	IIVRCP
SNPSV50	29G	Snap subroutines	IIVRCP
SNPTCH	29J	Snap subroutines	IIVRCP
SNPTIMER	29H	Snap subroutines	IIVRCP
SNPTIO	29J	Snap subroutines	IIVRCP
SPECIAL	81	Abnormal end/channel end appendage	IGG019S1
SRTCUU	3C	Initialization second-load routine	IIVIN2
SSKRTN	8C	SSK simulation subroutine	IIVPCE
SSMRTN	8C	SSM simulation subroutine	IIVPCE
STAERTRY	20C	STAE retry routine	IIVRTE
STCKRTN	8U	Store clock subroutine	IIVPCE
STGIO000	11A	Staged I/O routine	IIVSTG
STGIO004	11B	Staged I/O routine	IIVSTG
STGIO020	11C	Staged I/O routine	IIVSTG
STGIO040	11D	Staged I/O routine	IIVSTG
STGIO050	11D	Staged I/O routine	IIVSTG
STGIO068	11E	Staged I/O routine	IIVSTG
STGIO100	11G	Staged I/O routine	IIVSTG
STGIO105	11G	Staged I/O routine	IIVSTG
STGIO	107	Staged I/O routine	IIVSTG
STGIO110	11E	Staged I/O routine	IIVSTG
STGIO125	11H	Staged I/O routine	IIVSTG
STGIO200	11M	Load FCB subroutine	IIVSTG
STGIO300	11L	Read FCB subroutine	IIVSTG
STGIO400	11L	Read FCB subroutine	IIVSTG
STGIO75A	11D	Staged I/O routine	IIVSTG
STGIO75B	11F	Staged I/O routine	IIVSTG
STIDPRTN	8S	Store CPU ID subroutine	IIVPCE
STOALL	29K	Storage subroutines	IIVRCP
STODOS	29K	Storage subroutines	IIVRCP
STOEMBLK	29K	Storage subroutines	IIVRCP

Figure 31 (Part 13 of 15). Symbol Table

Symbol	Chart	Routine	Module
STOEMU	29K	Storage subroutines	IIVRCP
STOFST40	3D	Initialization second-load routine	IIVIN2
STOINIT	29K	Storage subroutines	IIVRCP
STOINV	29K	Storage subroutines	IIVRCP
STONODOS	29K	Storage subroutines	IIVRCP
STONUMBR	29K	Storage subroutines	IIVRCP
SVC00100	32A	Supervisor call intercept routine	IIVSCI
SVC00150	32A	Supervisor call intercept routine	IIVSCI
SVC00250	32A	Supervisor call intercept routine	IIVSCI
SVC100	17A	Supervisor call routine	IIVSVC
SVC150	17A	Supervisor call routine	IIVSVC
SVC160	32A	Supervisor call intercept routine	IIVSCI
SVC200	17A	Supervisor call routine	IIVSVC
SVC300	17A	Supervisor call routine	IIVSVC
SVC320	17A	Supervisor call routine	IIVSVC
SVC350	17A	Supervisor call routine	IIVSVC
SVC400	17A	Supervisor call routine	IIVSVC
SVC450	17A	Supervisor call routine	IIVSVC
SVC500	17A	Supervisor call routine	IIVSVC
SVC600	17A	Supervisor call routine	IIVSVC
SYNA	11N	SYNAD subroutine	IIVSTG
SYNADRTN	24L	EODAD, SYNAD, and status mapping routines	IIVIS
TCHRTN	8E	TCH simulation subroutine	IIVPCE
TIMECHK	20D	Timer interrupt check subroutine	IIVRTE
TIMECHK1	20D	Timer interrupt check subroutine	IIVRTE
TIOPOSTD	8G	TIO simulation subroutine	IIVPCE
TIORTN	8F	TIO simulation subroutine	IIVPCE
TIOSCAN	8G	TIO	IIVPCE
TIOSRCH	8F	TIO simulation subroutine	IIVPCE
TIOWAIT	8G	TIO simulation subroutine	IIVPCE
TIOXLOG	8F	TIO simulation subroutine	IIVPCE
TIOXLOG1	8F	TIO simulation subroutine	IIVPCE
TIOX2321	8F	TIO simulation subroutine	IIVPCE
TRCALL	29F	Trace subroutines	IIVRCP
TRCATTN	29F	Trace subroutines	IIVRCP
TRCCUU	29F	Trace subroutines	IIVRCP
TRCEXT	29F	Trace subroutines	IIVRCP
TRCHIO	29E	Trace subroutines	IIVRCP
TRCINIT	29E	Trace subroutines	IIVRCP
TRCINT	29F	Trace subroutines	IIVRCP
TRCINV	29E	Trace subroutines	IIVRCP
TRCIO	29F	Trace subroutines	IIVRCP
TRCISK	29E	Trace subroutines	IIVRCP
TRCLPSW	29E	Trace subroutines	IIVRCP
TRCNUMBR	29F	Trace subroutines	IIVRCP
TRCNWRAP	29F	Trace subroutines	IIVRCP
TRCSIO	29E	Trace subroutines	IIVRCP
TRCSSK	29E	Trace subroutines	IIVRCP
TRCSSM	29E	Trace subroutines	IIVRCP
TRCSVC	29F	Trace subroutines	IIVRCP
TRCTCH	29E	Trace subroutines	IIVRCP
TRCTIMER	29F	Trace subroutines	IIVRCP
TRCTIO	29E	Trace subroutines	IIVRCP
TRCWRAP	29F	Trace subroutines	IIVRCP
TSTOPN	22B	SVC monitor routine	IIVGR2
VIOA	25A	VTOC I/O simulation routine	IIVVIO
VIOA2	25A	VTOC I/O simulation routine	IIVVIO
VIOB	25B	VTOC I/O simulation routine	IIVVIO
VIOC	25B	VTOC I/O simulation routine	IIVVIO
VIOC1	25B	VTOC I/O simulation routine	IIVVIO
VIOD	25B	VTOC I/O simulation routine	IIVVIO

Figure 31 (Part 14 of 15). Symbol Table

Symbol	Chart	Routine	Module
VIOE	25B	VTOC I/O simulation routine	IIVVIO
VIOERRX	25E	VTOC I/O simulation routine	IIVVIO
VIOF	25B	VTOC I/O simulation routine	IIVVIO
VIOG	25C	VTOC I/O simulation routine	IIVVIO
VIOH	25C	VTOC I/O simulation routine	IIVVIO
VIOI	25C	VTOC I/O simulation routine	IIVVIO
VIOIO	25C	VTOC I/O simulation routine	IIVVIO
VIOIOA	25D	VTOC I/O simulation routine	IIVVIO
VIOIOB	25D	VTOC I/O simulation routine	IIVVIO
VIOIOB1	25D	VTOC I/O simulation routine	IIVVIO
VIOIOC	25D	VTOC I/O simulation routine	IIVVIO
VIOIOD	25D	VTOC I/O simulation routine	IIVVIO
VIOIOE	25E	VTOC I/O simulation routine	IIVVIO
VIOIOF	25E	VTOC I/O simulation routine	IIVVIO
VIOIOSIM	25E	VTOC I/O simulation routine	IIVVIO
VIOIOX1	25E	VTOC I/O simulation routine	IIVVIO
VIOIOX2	25E	VTOC I/O simulation routine	IIVVIO
VIOJ	25C	VTOC I/O simulation routine	IIVVIO
VIONXT	25C	VTOC I/O simulation routine	IIVVIO
VIONXT1	25C	VTOC I/O simulation routine	IIVVIO
VIORTRN	25E	VTOC I/O simulation routine	IIVVIO
WKNBLKD	24J	Write NEWKEY mapping routine	IIVIS
WRTKNRTN	24J	Write NEWKEY mapping routine	IIVIS
WRTKN100	24J	Write NEWKEY mapping routine	IIVIS
WRTKN101	24J	Write NEWKEY mapping routine	IIVIS
WRTKN200	24J	Write NEWKEY mapping routine	IIVIS
WRTKRTN	24H	Write key mapping routine	IIVIS
YESORNO	2K	YESORNO subroutine	IIVINT

Figure 31 (Part 15 of 15). Symbol Table

FIELD NAME TABLE

The field names in Figure 32 are contained in data areas which are initialized, modified, or interrogated by the Emulator. The list can be of help in identifying a field name with its data area and its location within that area. Although field names for DOS and OS data areas are included, they are limited to those significant to the Emulator. A full description of each field is included in the "Data Areas" section.

Field Name	Hexadecimal Location	Data Area
AASCHCK	4E0	EMUCONS
ABADR	4E4	EMUCONS
ABEBLK	4C4	EMUCONS
ABGPIB	44C	EMUCONS
ABGSYSRD	448	EMUCONS
ACOMTAB	98	EMUCONS
ACOMTABX	C4	EMUCONS
ACOMTBEX	C0	EMUCONS
ACTEXEND	C8	EMUCONS
ADDRWRK	1CC	EMUCONS
ADOSBTND	494	EMUCONS
ADOSBTR	490	EMUCONS
ADOSPUB	9C	EMUCONS
ADRILIST	50C	EMUCONS
AEMUCONS	514	EMUCONS
AEOJCHK	4D4	EMUCONS
AF1PIB	454	EMUCONS
AF2PIB	450	EMUCONS
AIIVADJ	5B8	EMUCONS
AIIVCCW	5B4	EMUCONS
AIIVDVS	45C	EMUCONS
AIIVGET	518	EMUCONS
AIIVGR2	458	EMUCONS
AIIVISM	460	EMUCONS
AIIVMSG	4C0	EMUCONS
AIIVOBE1	508	EMUCONS
AIIVPCI	188	RASCONS
AIIVPOV	478	EMUCONS
AIIVSCI	194	RASCONS
AIIVSDT	488	EMUCONS
AIIVVIO	48C	EMUCONS
AOPEN	4EC	EMUCONS
AOSPUB	1D0	EMUCONS
APC	4FC	EMUCONS
ARASCONS	464	EMUCONS
ARASNP	1A4	RASCONS
ARASPCE	180	RASCONS
ARASRTE	198	RASCONS
ARASSVC	18C	RASCONS
AROUTE	504	EMUCONS
AROUTEEP	500	EMUCONS
ART20	4F0	EMUCONS
ASTCBADR	D8	EMUCONS
ASTGIO	4E8	EMUCONS
ASVC	4F8	EMUCONS
ASYNEXAD	1BC	RASCONS
ASYNEXRN	1D8	RASCONS
ATIMER	4F4	EMUCONS
BADCUU	1F7	EMUCONS
BASEREGS	50C	EMUCONS
BHDR1	238	RASCONS

Figure 32 (Part 1 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
BREGSAVE	168	EMUCONS
CAWSCNAD	4C8	EMUCONS
CCB	3E8	Open table
CCWSCNAD	4CC	EMUCONS
CCW2ADDR	31	COMTAB
CEAIDS	1D4	EMUCONS
CHNINDX	A0	EMUCONS
CMPBLK	58	RASCONS
CMPBLK2	D0	RASCONS
CMPLEN	A	RASCONS
COMCAW	24	COMTAB
COMCCWSA	21	COMTAB
COMFMSK	28	COMTAB
COMKEY	20	COMTAB
COMTABEN	A8	EMUCONS
CPUID	5BC	EMUCONS
CSCDLB	378	Open table
CTDCBPTR	8	COMTAB extension
CTDCBUC	1	COMTAB extension
CTDDNAME	C	COMTAB extension
CTDSNAME	14	COMTAB extension
CTECHPTR	4	COMTAB extension
CTEXTCNT	FE	EMUCONS
CTEXTFLG	0	COMTAB extension
CTEXTPTR	4	COMTAB
CTFLAG	12	COMTAB
CTFLAG2	13	COMTAB
CTFLAG3	14	COMTAB
CTFLAG4	15	COMTAB
CTLREGS	5C4	EMUCONS
CTLR0	5C4	EMUCONS
CTLR14	5C4	EMUCONS
CTLR15	600	EMUCONS
CTLR2	5CC	EMUCONS
CTULEXT	16	COMTAB
CTXTNSAV	16	COMTAB
CUUSAVE	1E0	RASCONS
DADCB	348	EMUCONS
DADCBLEN	48	EMUCONS
DBUGDCB	46C	EMUCONS
DCBBLKSI	3E	DCB
DCBDDNAM	28	DCB
DCBEODAD	21	DCB
DCBEXCD1	50	DCB
DCBEXCD2	51	DCB
DCBIFLGS	2C	DCB
DCBKEYLE	10	DCB
DCBLPDA	B8	DCB
DCBLRECL	52	DCB
DCBMAC	35	DCB
DCBMACR	32	DCB
DCBNOREC	D6	DCB
DCBNREC	9C	DCB
DCBOFLGS	30	DCB
DCBOPTCD	34	DCB
DCBPTR	4C	COMTAB
DCBRECFM	24	DCB
DCBRKP	3C	DCB
DCBRORG1	E0	DCB
DCBRORG2	CE	DCB
DCBRORG3	98	DCB

Figure 32 (Part 2 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
DCBSYNAD	38	DCB
DCBTDC	95	DCB
DCELEBNR	38	UCB
DEBBINUM	24	DEB
DEBCEA	-18	DEB
DEBDVMOD	20	DEB
DEBENDCC	2A	DEB
DEBENDHH	2C	DEB
DEBEOEA	-24	DEB
DEBNMTRK	2E	DEB
DEBSIOA	-20	DEB
DEBSTRCC	26	DEB
DEBSTRHH	28	DEB
DEBXCEA	-14	DEB
DECBAREA	C	DECB
DECBDCBA	8	DECB
DECBEXC1	18	DECB
DECBEXC2	19	DECB
DECBKEY	14	DECB
DECBLOGP	10	DECB
DEVTYP	B	COMTAB
DIAGBLK	0	RASCONS
DIAGBUF	8	RASCONS
DIAGCUU	4	RASCONS
DIAGFLG	6	RASCONS
DIAGLNG	74	RASCONS
DIAGNXT	0	RASCONS
DIAGREL	80	RASCONS
DOSBIN	E	COMTAB
DOSCAW	0	CAW
DOSCCB	0	CCB
DOSCSW	0	CSW
DOSCUU	8	COMTAB
DOSRFTB	608	EMUCONS
DOSSIZE	E0	EMUCONS
DRELEASE	605	EMUCONS
DSFIDBLK	10C	EMUCONS
DS1DSIND	5D	DSCB-F1
DS1FMTID	2C	DSCB-F1
DS1PTRDS	87	DSCB-F1
DS4DEVDT	4A	DSCB-F4
DS4DSREC	32	DSCB-F4
DS4VTOCE	69	DSCB-F4
DS4VTOCI	3A	DSCB-F4
DTFFLNME	16	DTFIS tables
DTFLMODA	11	DTFIS tables
DTFRCTGD	44	DTFIS ADD-RETRVE-ADDRTR table
DTFTYPE	14	DTFIS tables
ECB	0	COMTAB
ECB	0	ECB
ECBLIST	E8	EMUCONS
ECBLIST	0	ECB pointer table
EMUCCW1	29	COMTAB
EMUCCW2	30	COMTAB
EMUCONS	0	EMUCONS
EMULBLAD	60C	EMUCONS
EMUPATCH	624	EMUCONS
EMUSAVE	16C	EMUCONS
ENDPTECB	BC	EMUCONS
EOJSW	101	EMUCONS
EXMSG	1E5	RASCONS

Figure 32 (Part 3 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
EXTMSK	1FA	EMUCONS
FCBCCW	480	EMUCONS
FCBCCWCT	486	EMUCONS
FCBCCWDA	481	EMUCONS
FCBCCWFG	484	EMUCONS
FCBCCWOP	480	EMUCONS
FCBCCWX	485	EMUCONS
FIDBLK	48	ISBLK
FIDCTXTN	8	FIDBLK
FIDLTK	13	FIDBLK
FIDNXT	0	FIDBLK
FIDPRV	4	FIDBLK
FIDTFNME	C	FIDBLK
F4INCORE	4A0	Open table
F4INDIC	4A0	Open table
HDRDATE	297	RASCONS
HDRPAGE	2B0	RASCONS
HDRTIME	27D	RASCONS
HDR1	240	RASCONS
HEXTAB	13B	EMUCONS
HYPLIST	0	EMUCONS
H1	1BE	EMUCONS
IIVCON	0	EMUCONS
IIVCONB1	4D8	EMUCONS
IIVCONB2	4DC	EMUCONS
IIVCONB3	51D	EMUCONS
IIVRCN	0	RASCONS
IJHAARAD	C8	DTFIS ADD-RETRVE-ADDRTR table
IJHACFID	A0	DTFIS ADD-RETRVE-ADDRTR table
IJHACLNK	B2	DTFIS ADD-RETRVE-ADDRTR table
IJHACOCR	80	DTFIS ADD-RETRVE-ADDRTR table
IJHACOFC	9A	DTFIS ADD-RETRVE-ADDRTR table
IJHACORC	9C	DTFIS ADD-RETRVE-ADDRTR table
IJHACOTC	98	DTFIS ADD-RETRVE-ADDRTR table
IJHACPRC	7C	DTFIS ADD-RETRVE-ADDRTR table
IJHACRID	98	DTFIS ADD-RETRVE-ADDRTR table
IJHACTIN	A8	DTFIS ADD-RETRVE-ADDRTR table
IJHACTNA	88	DTFIS ADD-RETRVE-ADDRTR table
IJHACTOA	90	DTFIS ADD-RETRVE-ADDRTR table
IJHACUSE	CC	DTFIS ADD-RETRVE-ADDRTR table
IJHADKEY	D0	DTFIS ADD-RETRVE-ADDRTR table
IJHAISKY	DA	DTFIS ADD-RETRVE-ADDRTR table
IJHCATB2	64	DTFIS ADD-RETRVE-ADDRTR table
IJHCATB3	66	DTFIS ADD-RETRVE-ADDRTR table
IJHCBLSZ	4E	DTFIS ADD-RETRVE-ADDRTR table
IJHCCCB	0	DTFIS ADD-RETRVE-ADDRTR table
IJHCCCW	8	DTFIS ADD-RETRVE-ADDRTR table
IJHCCLPA	42	DTFIS ADD-RETRVE-ADDRTR table
IJHCKYLC	5E	DTFIS ADD-RETRVE-ADDRTR table
IJHCKYSZ	4C	DTFIS ADD-RETRVE-ADDRTR table
IJHCOPT	15	DTFIS ADD-RETRVE-ADDRTR table
IJHCRARA	C	DTFIS ADD-RETRVE-ADDRTR table
IJHCRESZ	4A	DTFIS ADD-RETRVE-ADDRTR table
IJHCRKEY	10	DTFIS ADD-RETRVE-ADDRTR table
IJHCRTR	2B	DTFIS ADD-RETRVE-ADDRTR table
IJHCRWOR	14	DTFIS ADD-RETRVE-ADDRTR table
IJHCSADR	68	DTFIS ADD-RETRVE-ADDRTR table
IJHCSTBY	1E	DTFIS ADD-RETRVE-ADDRTR table
IJHKADCN	B8	DTFIS load table
IJHKBKLN	4E	DTFIS load table
IJHKCCB	0	DTFIS load table

Figure 32 (Part 4 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
IJHKCCOD	1E	DTFIS load table
IJHKLGLN	4A	DTFIS load table
IJHKLPDR	42	DTFIS load table
IJHKOPCO	15	DTFIS load table
IJHKPRCT	6C	DTFIS load table
IJHKRDWR	98	DTFIS load table
IJHRADSV	48	DTFIS ADD-RETRVE-ADDRTR table
IJHROVCN	54	DTFIS ADD-RETRVE-ADDRTR table
IJHRREGS	46	DTFIS ADD-RETRVE-ADDRTR table
IJHSCADR	30	DTFIS ADD-RETRVE-ADDRTR table
IJHSDB1	18	DTFIS ADD-RETRVE-ADDRTR table
IJHSIOAR	8	DTFIS ADD-RETRVE-ADDRTR table
IJHSLIOR	1C	DTFIS ADD-RETRVE-ADDRTR table
IMGLBDCB	468	EMUCONS
IOB	38	COMTAB
IOBCSW	9	IOB
IOBDCBPT	15	IOB
IOBECBPT	5	IOB
IOBFLAG1	0	IOB
IOBSEEK	21	IOB
IOBSEEKM	20	IOB
IOBSENS0	2	IOB
IOBSENS1	3	IOB
IOBSTART	11	IOB
IPLCUU	CE	EMUCONS
IPLOFST	D3	EMUCONS
IPLSTMT	498	EMUCONS
IPLSW	1D8	EMUCONS
ISAMCODE	104	EMUCONS
ISAMDTFA	105	EMUCONS
ISAMFDCX	108	EMUCONS
ISAMLIST	104	EMUCONS
ISCODE	68	ISBLK
ISCOMTAB	6C	ISBLK
ISDCB	78	ISBLK
ISDCB1	184	ISBLK
ISDECB	164	ISBLK
ISDTFA	69	ISBLK
ISDTFLMA	180	ISBLK
ISECB	5C	ISBLK
ISFIDBLK	110	EMUCONS
ISFLAGS	74	ISBLK
ISLIST	68	ISBLK
ISREGSAV	0	ISBLK
ISREGSVA	64	ISBLK
ISSKTAB	F8	EMUCONS
ISTCB	60	ISBLK
ISWKNARA	70	ISBLK
JFCBAREA	394	EMUCONS
JFCBBS	3FA	EMUCONS
JFCBDSJN	3AA	EMUCONS
JFCBDSN	394	EMUCONS
JFCBDSNM	0	JFCB
JFCBDS18	3A5	EMUCONS
JFCBDS19	3A6	EMUCONS
JFCBDS8	39C	EMUCONS
JFCBIB	3C8	EMUCONS
JFCBIND2	3EB	EMUCONS
JFCBIP	3D6	EMUCONS
JFCBLKSI	66	JFCB
JFCBLTYP	42	JFCB

Figure 32 (Part 5 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
JFCBMSK1	3E0	EMUCONS
JFCBTSDM	34	JFCB
JFCBUFIN	3ED	EMUCONS
JFCBUFL	3EE	EMUCONS
JFCBUFNO	3EC	EMUCONS
JFCBVOLS	76	JFCB
JFCBVSER	40A	EMUCONS
JFCBVSR1	40A	EMUCONS
JFCBVSR3	40D	EMUCONS
JFCCYLOF	402	EMUCONS
JFCDSORG	3F6	EMUCONS
JFCFCBID	3CC	EMUCONS
JFCKEYLE	3F1	EMUCONS
JFCLRECL	3FC	EMUCONS
JFCNCP	3FE	EMUCONS
JFCNTM	3FF	EMUCONS
JFCOPTCD	3F9	EMUCONS
JFCRECFM	3F8	EMUCONS
JFCRKP	400	EMUCONS
LABADDR	4A8	Open table
LEXADR	1B8	RASCONS
LEXMSG	1E3	RASCONS
LEXRTN	1D0	RASCONS
LHDR1	23C	RASCONS
LIMTBCKT	3E0	Open table
LINECNT	2BA	RASCONS
LOGCUU	D0	EMUCONS
LOGIO	4D0	EMUCONS
LOGOFST	D4	EMUCONS
LOPTMSG1	20A	RASCONS
LPSWSVE	222	RASCONS
MODNM	201	RASCONS
MSGIN	4A1	Open table
MSGINDIC	4A1	Open table
MSGOUT	4A1	Open table
MSGWF	4A1	Open table
NBRENT	FC	EMUCONS
NBR2K	E4	EMUCONS
NOIOPEND	1BC	EMUCONS
NOSIO	1C0	EMUCONS
NXTBTR	114	EMUCONS
OLDBTR	11C	EMUCONS
OPENLIST	598	EMUCONS
OPTBFLGS	4A3	Open table
OPTBFLPT	4A3	Open table
OPTBLNK	4A3	Open table
OPTBSYFL	4A3	Open table
OPTCUU	211	RASCONS
OPTFLGR2	100	EMUCONS
OPTION	20C	RASCONS
OPTMSG1	20C	RASCONS
OSBIN	10	COMTAB
OSCUU	C	COMTAB
PAGECNT	2BC	RASCONS
PARMCLSE	20	EMUCONS
PARMCODE	104	EMUCONS
PARMDTFA	104	EMUCONS
PARMEOX	24	EMUCONS
PARMFDCX	108	EMUCONS
PARMFLG	1C2	EMUCONS
PARMLST	104	EMUCONS

Figure 32 (Part 6 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
PARMLTK	108	EMUCONS
PARMOPEN	1C	EMUCONS
PCEXITAD	1B4	RASCONS
PCEXRTN	1C8	RASCONS
PECBPTR1	B0	EMUCONS
PECBPTR2	B4	EMUCONS
PENDSW	1D9	EMUCONS
POSTECB	B8	EMUCONS
PROMPECB	4DC	EMUCONS
PRPCONTF	594	EMUCONS
PRPCUUC	4BC	EMUCONS
PRPCUUWK	4B8	EMUCONS
PRPMSGJN	1DF	EMUCONS
PRPMSG99	1DA	EMUCONS
PRPRPLY	51D	EMUCONS
PRPRPLYZ	51C	EMUCONS
PRPSW	1F5	EMUCONS
PRPVSER	1EF	EMUCONS
PSWMSK2	172	RASCONS
PSWSAVE	168	RASCONS
PSWSVE	224	RASCONS
PSWSW	17C	RASCONS
PTCHAREA	3C8	RASCONS
PUBINDX	A	COMTAB
QLIST	59C	EMUCONS
QNAME	5A8	EMUCONS
RASART20	1A0	RASCONS
RASCCW	2C4	RASCONS
RASCONS	0	RASCONS
RASDIAG	2C0	RASCONS
RASFLG1	17E	RASCONS
RASNPCDS	36	RASCONS
RASNPCUU	40	RASCONS
RASNPSW1	54	RASCONS
RASNPSW2	55	RASCONS
RASNPSW3	56	RASCONS
RASNPSW4	57	RASCONS
RASOPCDS	0	RASCONS
RASOPLEN	8	RASCONS
RASOTHER	19C	RASCONS
RASPC	184	RASCONS
RASPCSW	17F	RASCONS
RASPSWSV	160	RASCONS
RASSNPNO	34	RASCONS
RASSVC	190	RASCONS
RASTRCDS	14	RASCONS
RASTRCMX	C	RASCONS
RASTRCNO	10	RASCONS
RASTRCNX	E	RASCONS
RASTRCTB	8	RASCONS
RASTRCUU	1C	RASCONS
RASTRLEN	A	RASCONS
RASTRSVC	2C8	RASCONS
RASTRSW1	30	RASCONS
RASTRSW2	31	RASCONS
RASTRSW3	32	RASCONS
RASTRSW4	33	RASCONS
REGA	150	EMUCONS
REGB	154	EMUCONS
REGC	158	EMUCONS
REGD	15C	EMUCONS

Figure 32 (Part 7 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
REGE	160	EMUCONS
REGF	164	EMUCONS
REGSAV	128	EMUCONS
REGSAVE	1C4	EMUCONS
REG0	128	EMUCONS
REG1	12C	EMUCONS
REG2	130	EMUCONS
REG3	134	EMUCONS
REG4	138	EMUCONS
REG5	13C	EMUCONS
REG6	140	EMUCONS
REG7	144	EMUCONS
REG8	148	EMUCONS
REG9	14C	EMUCONS
RESCUU	CC	EMUCONS
RESOFST	D2	EMUCONS
RFACTOR	510	EMUCONS
RNAME	5B0	EMUCONS
RNMBIN	5B3	EMUCONS
RNMCUU	5B0	EMUCONS
RNMELEN	4	EMUCONS
SCANBEG	244	EMUCONS
SCANCHR	240	EMUCONS
SCANEND	23C	EMUCONS
SCANLNG	248	EMUCONS
SCANSTP	243	EMUCONS
SCANTBL	23C	EMUCONS
SECOND	1B4	EMUCONS
SECONDA	1B8	EMUCONS
SEKADR	3FC	Open table
SNPCUULN	A	RASCONS
SNPLEN	8	RASCONS
SNPNO	21C	RASCONS
SRCHKEY	409	Open table
STAEREGS	470	EMUCONS
STAERTN	474	EMUCONS
STCKADDR	620	EMUCONS
STGBUF	18	STGCON
STGCCW	1C	STGCON
STGCHFLG	10	STGCON
STGCON	38	COMTAB
STGCSW	9	STGCON
STGCUU	24	STGCON
STGDLM	20	STGCON
STGCTP	18	STGCON
STGFLG	0	STGCON
STGFLG2	27	STGCON
STGINTR	AC	EMUCONS
STGLNCNT	4	STGCON
STGLNPTR	6	STGCON
STGMAX	12	STGCON
STGOPCD	14	STGCON
STGSEN0	2	STGCON
STGSEN1	3	STGCON
STGWK1	1	STGCON
STORGSW	17D	RASCONS
STPTBL	24C	EMUCONS
SVCCHN	1A8	RASCONS
SVCEXAD	1B0	RASCONS
SVCEXRTN	1C0	RASCONS
S370SW	604	EMUCONS

Figure 32 (Part 8 of 9). Field Name Table

Field Name	Hexadecimal Location	Data Area
TAFLAG1	74	ISBLK
TAFLAG2	75	ISBLK
TAFLAG3	76	ISBLK
TAFLAG4	77	ISBLK
TDELTA	610	EMUCONS
TEBLEN	0	TEBV
TEBTAB	0	TEB
TEBV	18	TEBV
TIMEIND	1F6	EMUCONS
TIMERECB	124	EMUCONS
TIOEDDNM	4	TIOT
TIOEFSRT	1	TIOT
TIOELNGH	0	TIOT
TIOTADR	DC	EMUCONS
TRCDLEN	8	RASCONS
TRCLEN	12	RASCONS
TRNSFLD	F0	EMUCONS
TRNSFLDA	F1	EMUCONS
TRNSFLD1	F0	EMUCONS
TRNSFLD3	F0	EMUCONS
TRNSFLD4	F0	EMUCONS
TRNSFLD5	F0	EMUCONS
TSBLEN	1	TEBV
TYPSAVE	1E2	RASCONS
UCBTYP	10	UCB
VIIVCON	1AC	RASCONS
VOLLABI	0	VOL label
VOLNO	3	VOL label
VOLSERNO	4	VOL label
VOLVTOC	B	VOL label
WKAREA	618	EMUCONS
WTOECB	444	EMUCONS
WTORECB	4D8	EMUCONS
XLIST	390	EMUCONS

Figure 32 (Part 9 of 9). Field Name Table

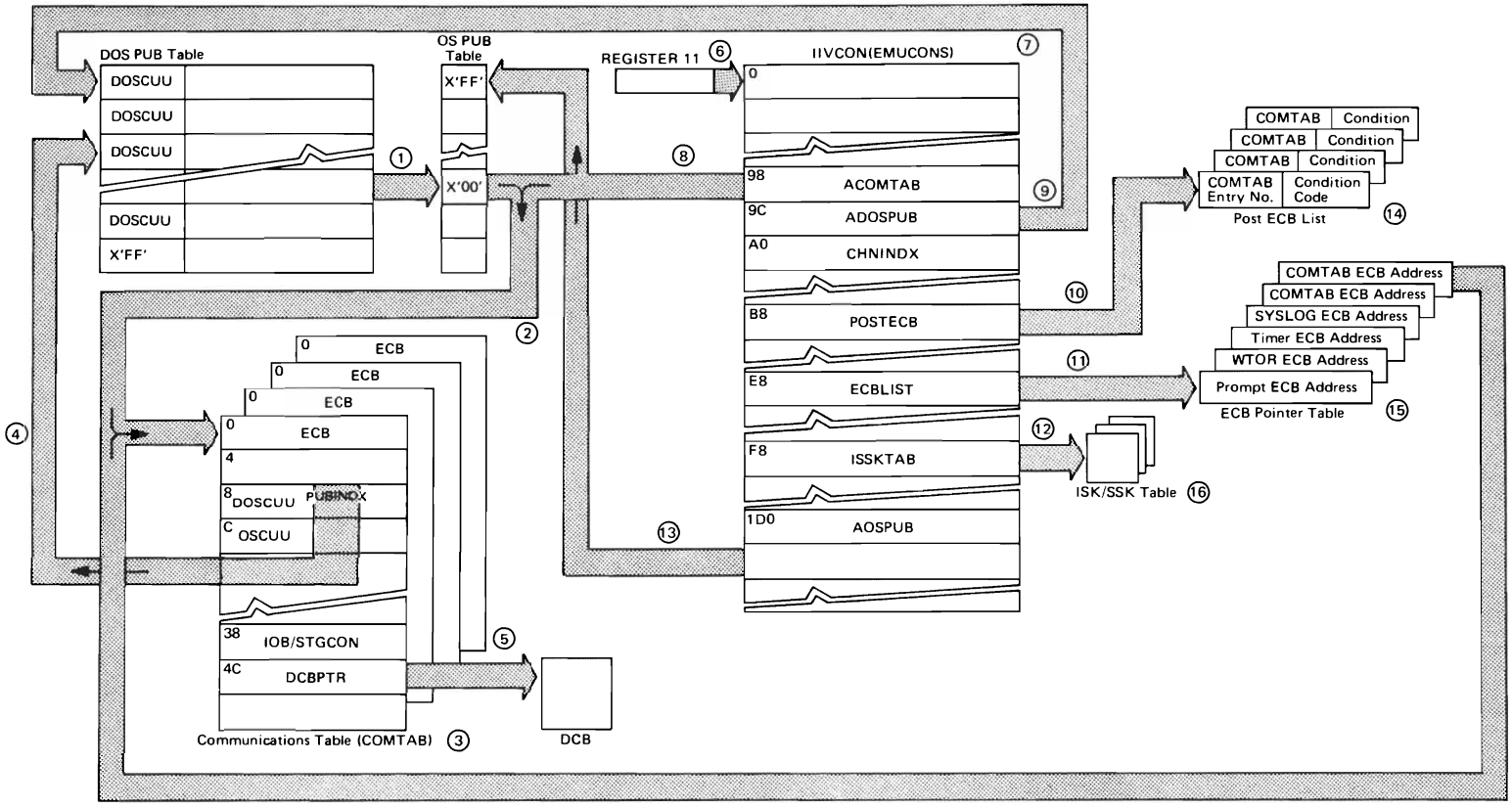
DATA AREAS

Data Area Relationships

Data Area Layouts

DATA AREA RELATIONSHIPS

The major paths by which Emulator routines can reach specific data areas are shown in Figures 33, 34, and 35.

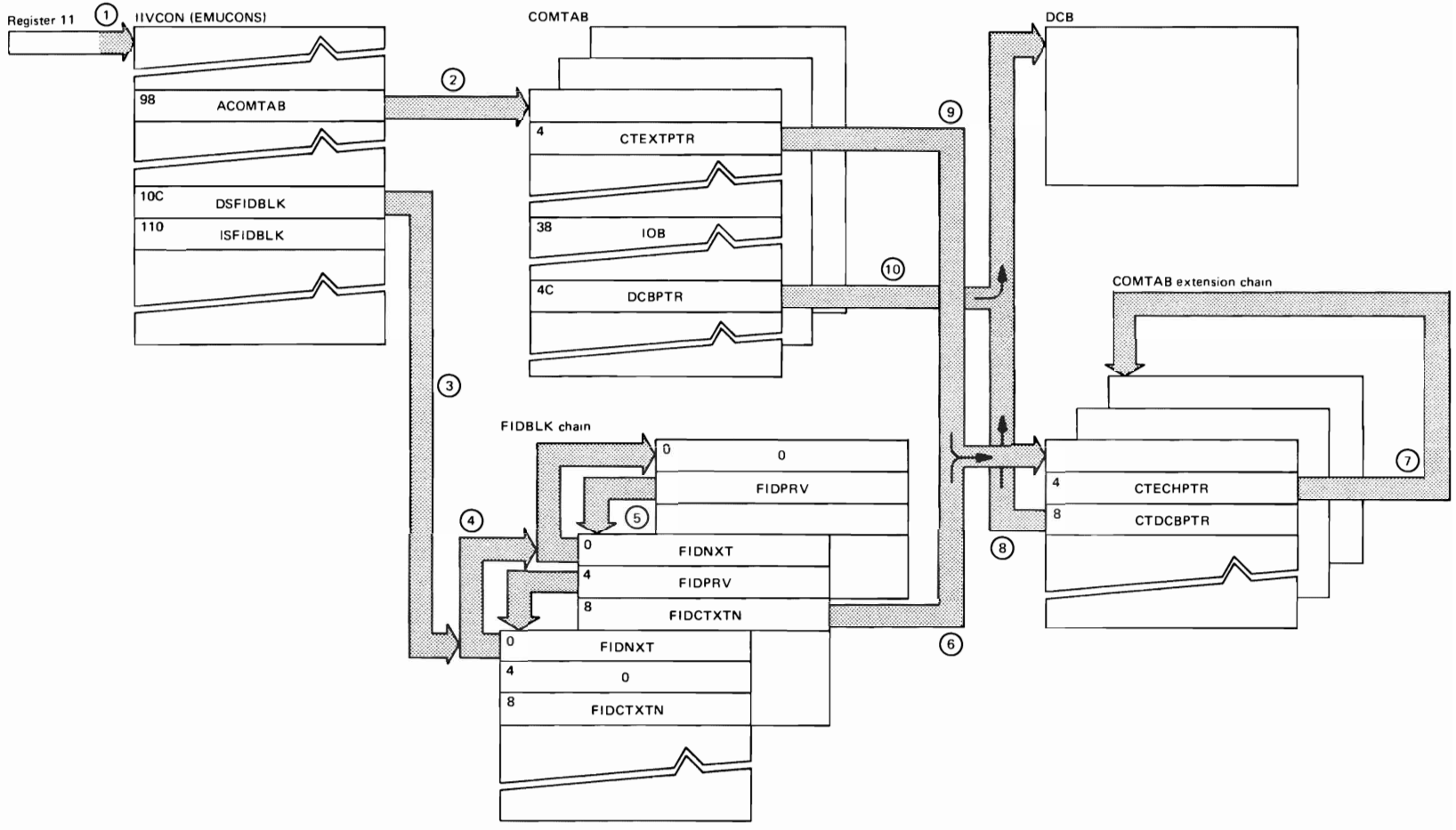


KEY:

DOS PUB table	(1)	A one-to-one relationship exists between the DOS PUB table and the OS PUB table. An X'FF' in the last byte of this table acts as a table delimiter.
OS PUB table	(2)	An OS PUB entry specifies a COMTAB entry number An X'FF' entry indicates that no COMTAB entry exists for a DOS PUB table entry.
COMTAB	(3)	The communications table has one 88-byte entry for every device allocated to emulation plus an entry for SYSLOG.
	(4)	The byte labeled PUBINDX is an index into the DOS PUB table where the PUB entry for this DOS device can be found.
	(5)	Bytes X'4C-4F' labeled DCBPTR point to the appropriate DCB.
DCB		For every data set to be processed by a program, there is a corresponding data control block to indicate its current use.
Register 11	(6)	This register points to the first byte of IIVCON.
IIVCON	(7)	The CSECT IIVCON contains data constants common to most Emulator modules.
	(8)	Bytes X'98-9B' labeled ACOMTAB point to COMTAB.
	(9)	Bytes X'9C-9F' labeled ADOSPUB point to the DOS PUB table.
	(10)	Bytes X'B8-BB' labeled POSTECB point to the post ECB list.
	(11)	Bytes X'E8-EB' labeled ECBLIST point to the ECB pointer table.
	(12)	Bytes X'F8-FB' labeled ISSKTAB point to the ISK/SSK table.
	(13)	Bytes X'1D0-1D3' labeled AOSPUB point to the OS PUB table.
Post ECB list	(14)	The first byte points to a COMTAB entry. The second byte contains the condition code from the ECB when it was last posted.
ECB Pointer table	(15)	This table is a list of 4-byte addresses to the ECBs for the devices being used (or DOS emulation plus addresses for SYSLOG and three special ECBs.
ISK/SSK table	(16)	This table has a 1-byte entry for each 2K block of DOS storage that contains the appropriate protect key for each block.

Figure 33. Data Area Relationships (When Resources are Dedicated or Staged)

Figure 34. Data Area Relationships (When Direct-Access Data Sets/Files Other Than OS Indexed Sequential are Shared)



KEY:

Register 11 — (1) This register points to the first byte of IIVCON.

IIVCON (EMUCONS) — (2) Bytes X'98-9B' (labeled ACOMTAB) point to COMTAB.

(3) Bytes X'10C-10F' (labeled DSFIDBLK) point to a FIDBLK in the FIDBLK chain.

FIDBLK — (4) Bytes X'0-3' (labeled FIDNXT in each FIDBLK) point to the following FIDBLK in the chain.

(5) Bytes X'4-7' (labeled FIDPRV in each FIDBLK) point to a previous FIDBLK in the chain. (This field is zero in the first FIDBLK in the chain.)

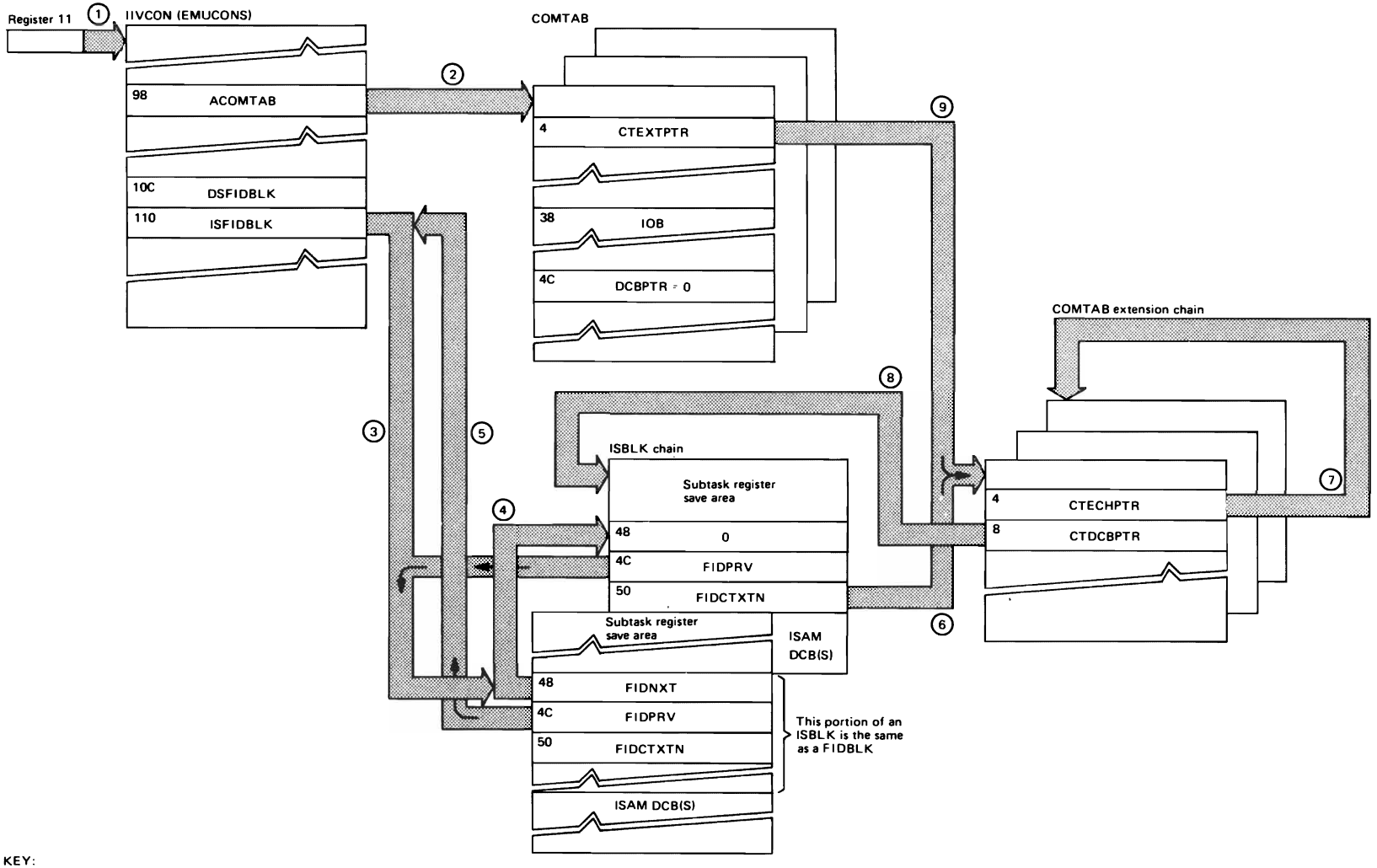
(6) Bytes X'8-B' (labeled FIDCTXTN in each FIDBLK) point to the appropriate COMTAB extension for the data set in the chain of COMTAB extensions.

COMTAB Extension — (7) Bytes X'4-7' (labeled CTECHPTR in each COMTAB extension) point to the next COMTAB extension on this device.

(8) Bytes X'8-B' (labeled CTDCBPTR in each COMTAB extension) point to the DCB for this data set.

COMTAB — (9) Bytes X'4-7' (labeled CTEXTPTR in each COMTAB) point to the COMTAB extension for this entry.

(10) Bytes X'4C-4F' (labeled DCBPTR in each COMTAB) point to the DCB for this data set.



KEY:

Register 11 — (1) This register points to the first byte of IIVCON.

IIVCON (EMUCONS) — (2) Bytes X'98-9B' (labeled ACOMTAB) point to COMTAB.

(3) Bytes X'110-113' (labeled ISFIDBLK) point to FIDBLK in the ISBLK.

ISBLK — (4) Bytes X'48-4B' (labeled FIDNXT in each FIDBLK in the ISBLK) point to the following FIDBLK in the chain.

(5) Bytes X'4C-4F' (labeled FIDPRV in each FIDBLK) point to the previous FIDBLK in the chain. However, this field in the first FIDBLK in the chain points to the ISFIDBLK field in IIVCON.

(6) Bytes X'50-53' (labeled FIDCTXTN in each FIDBLK) point to the appropriate COMTAB extension in the chain of COMTAB extensions.

COMTAB Extension — (7) Bytes X'4-7' (labeled CTECHPTR in each COMTAB extension) point to the next COMTAB extension on this device.

(8) Bytes X'8-B' (labeled CTDCBPTR in each COMTAB extension) point to the associated ISBLK in the ISBLK chain.

COMTAB — (9) Bytes X'4-7' (labeled CTEXTPTR in each COMTAB) point to the COMTAB extensions associated with this volume.

Figure 35. Data Area Relationships (When OS Indexed Sequential, Direct-Access Data Sets are Shared)

DATA AREA LAYOUTS

This section describes control blocks unique to the Emulator, control blocks in the Emulator region that are referenced by both the Emulator and OS, and data areas in DOS and OS that are referenced by the Emulator.

In some cases, major portions of the block not relevant to the functions of the Emulator are omitted. In those cases, a reference is made to a publication where a more complete description can be found.

Some of the data areas (for example, the CAW, CSW, and the PSW) are fully described in the publication IBM System/360 Principles of Operation. The descriptions of those data areas are therefore condensed in this section to minimize references to the Principles of Operation publication.

The symbolic names shown in individual data area fields represent the offset, in bytes, from the beginning of a table to the field. Access is gained to a specific field by using an instruction in which the beginning address of the table (usually contained in a register) is the base address, and the symbolic field name represents the displacement.

There are places where the symbolic field names will differ from the names used in other publications. Names used here were taken from Emulator listings and where differences exist, a nonemulator program may refer to the field by the other name. (To resolve name differences, compare offsets; if the offsets match, the names represent the same field.)

Usage of the data area fields can be traced in the Emulator listings by first locating the symbolic field names in the cross-reference table at the back of the listings and then noting where the names are used. Where no symbolic name appears in a data area field, the field is probably not referred to by the Emulator.

The page format used for data area field descriptions is identical to that used in the publication System Control Blocks for OS or System Data Areas for OS/VS. The field headings and their meanings are:

```
**************************************************************************
*        Bytes and Field     Hex.                                        *
*Offset  Alignment Name      Dig. Field Description, Contents, Meaning*
**************************************************************************
```

Offset: The numeric address of the field relative to the beginning of the data area. The first number is the offset in decimal, followed (in parentheses) by the hexadecimal equivalent.

Example: 16(10)

Bytes and Alignment: The size (number of bytes) of the field and its alignment relative to the fullword boundary.

Examples:

4	-	a 4-byte field beginning on a word boundary
..2	-	a 2-byte field beginning on a halfword boundary
...1	-	a 1-byte field in the low-order byte of a word
...3	-	a 3-byte field beginning at the low-order byte of a word (and running into the next word)

Field Name: A name that identifies the field. This column is also used to show the bit settings of flag fields, that is, the state of bits in a byte. When the column is used to show the state of bits (0, 1) in a flag byte, it is shown as follows:

....	.... The 8 bit positions (0-7) in a byte. For ease of scanning, the high-order (left-hand) 4 bits are separated from the low-order 4 bits.
X...	.... A reference to bit 0.
1...	.... Bit 0 is on.
0...	.... Bit 0 is off.
....	..xx A reference to bits 6 and 7.

Bit settings that are significant are shown and described. Bit settings that are not presently significant are described as reserved bits. Do not use these bits because the Operating System may make use of them in the future.

Hex.dig.(hexadecimal digits): The contents of the field expressed as hexadecimal digits.

Field Description, Contents, Meaning: The use of the field.

Adjust CCW List

Initialized by: IIVINT

Modified by: IIVCCW, IIVPCE, IIVABN, IIVADJ, adjust CCW string instruction

Interrogated by: IIVCCW, IIVSNP, IIVADJ, and adjust CCW string instruction

Pointer in: EMUCONS and Emulator register 9 + X'40' offset

The Adjust CCW list (Figure 36) contains information necessary for modifying the data addresses in channel command words so that they address the main-storage area assigned to the emulated environment. See the Appendix for details.

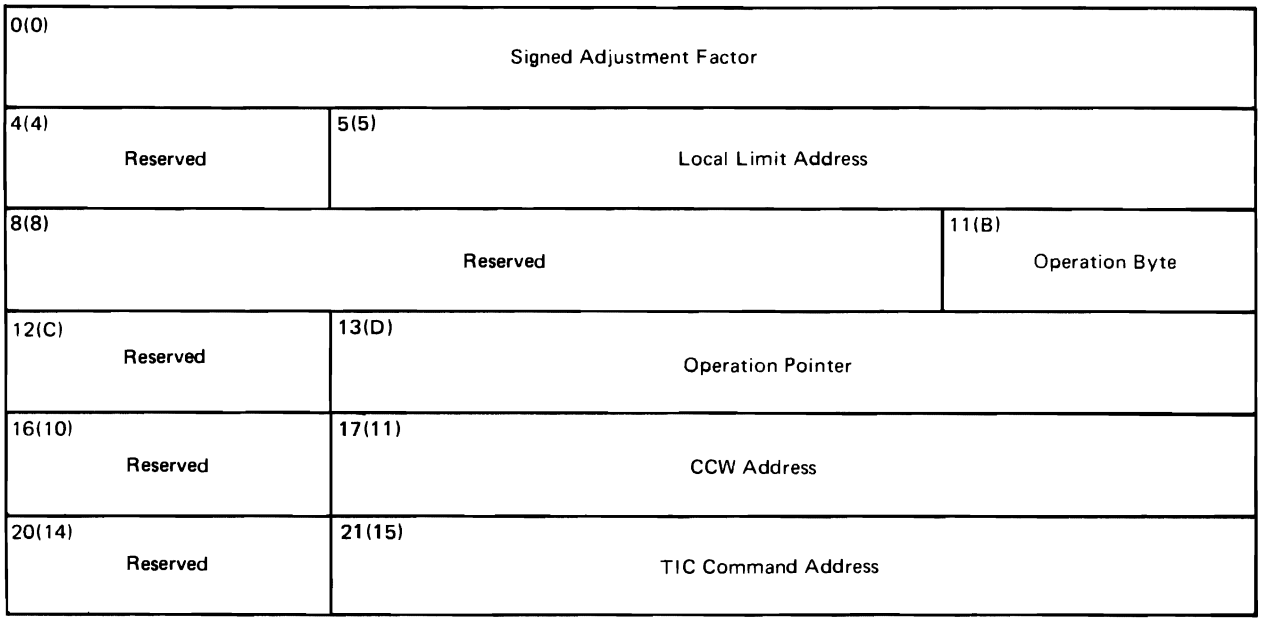


Figure 36. Adjust CCW List

Adjust CCW List Description

Offset	Bytes and Field Alignment Name	Field Description, Contents, Meaning
0(0)	4	Signed adjustment factor. This area holds a fullword, binary, signed number on a 4096-byte boundary (the three low-order hexadecimal digits equal 000). This factor is algebraically added to the data addresses of the CCWs. A positive number is used to adjust local CCW data addresses to true addresses; a negative number is used to reconvert true data addresses to local addresses.
5(5)	3	Local limit address. This word holds the maximum address usable by the emulated DOS programs. The address value must be on a 4096-byte boundary minus 1 (the three low-order hexadecimal digits equal FFF). The adjusted data address computed from the CCW data address plus the adjustment factor (see above) must fall between address 0 and the local limit address.
11(B)	...1	Operation byte. This field carries the command code for CCWs that are data chained together. The operation byte is set to the command of the first CCW of a data chain and is used to indicate direction when computing the extreme data addresses of the CCWs that are data chained together.

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
			The field is set to zero when the CCW being processed is not data chained. Therefore, a nonzero field found upon initialization of an instruction indicates that the first CCW to be adjusted is part of a chain.
13(D)	3		Operation pointer. This word contains the true address of the CCW that originated the operation byte for the last nonTIC CCW adjusted.
17(11)	3		CCW address. This field holds the true address of the first CCW of the string. The adjust CCW string instruction (see Appendix) adds the adjustment factor to the data address portion of each CCW in the CCW string until: (1) The end of the string has been reached (condition code = 0); (2) A TIC (transfer in channel) command has been encountered (condition code = 1); (3) The local storage area specified by the last CCW adjusted falls at least partially outside the limits of 0 and the local storage limit address (condition code = 2); or (4) The next CCW to be adjusted falls at least partially outside the limits of 0 and the local storage limit address or is not located on a doubleword boundary (condition code = 3). Whenever one of these four conditions terminates the instruction, the condition code is set and the address +8 of the last CCW adjusted is placed into the CCW address field. If a data chain was in process, the command code and address of the CCW containing the command are set in the operation byte and operation pointer fields of this list.
21(15)	3		TIC command address. If the last CCW adjusted was a TIC, this field contains the command address from the TIC CCW. If the TIC CCW is within a data chain, the operation byte and operation pointer fields of this list contain the values set for the first CCW of the data chain. If the TIC CCW is not within a data chain, the operation byte field of this list is set to zero, the CCW address field of this list is set to the address +8 of the TIC CCW, and the operation pointer field of this list is set to the address of the last CCW adjusted before the TIC.

Beginning and End Block (BEBLK)

Initialized by: IIVCCW

Modified/Interrogated by: IIVCCW, IIVSNP

Pointer in: EMUCONS

The BEBLK table (Figure 37 Part 1 of 2) in the Emulator routine IIVCCW contains an 8-byte entry for each group of CCWs. The first four bytes of an entry contain the starting address of a group of CCWs within the CCW chain presently being processed by the IIVCCW routine. This starting address will be the address of one of the following CCWs (numbers below correspond to numbers in Figure 37 Part 2 of 2):

(1) The first CCW in the CCW chain being processed.

(2) The CCW addressed by a TIC CCW (only if that CCW addresses a group of CCWs other than the group that the TIC CCW itself terminates).

(3) The CCW that follows a TIC CCW (only if the TIC CCW addresses a CCW within the group that the TIC CCW itself terminates).

The end of a group of CCWs is determined by encountering one of the following CCWs (Figure 37 Part 2 of 2):

(4) A TIC CCW.

(5) The end of the CCW chain (the command chaining and data chaining bits set to zero).

The last four bytes in each BEBLK table entry contain the address +8 of the CCW that terminates the group of CCWs begun at the address contained in the first four bytes.

BEBLK

0(0) Beginning CCW Address	4(4) Ending CCW Address +8
8(8) Beginning CCW Address	12(C) Ending CCW Address +8
16(10) Beginning CCW Address	20(14) Ending CCW Address +8
24(18) Beginning CCW Address	28(1C) Ending CCW Addr +8
32(20) Beginning CCW Address	36(24) E
40(28) Beginning CCW Address	

Figure 37 (Part 1 of 2). Beginning and End Block (BEBLK)

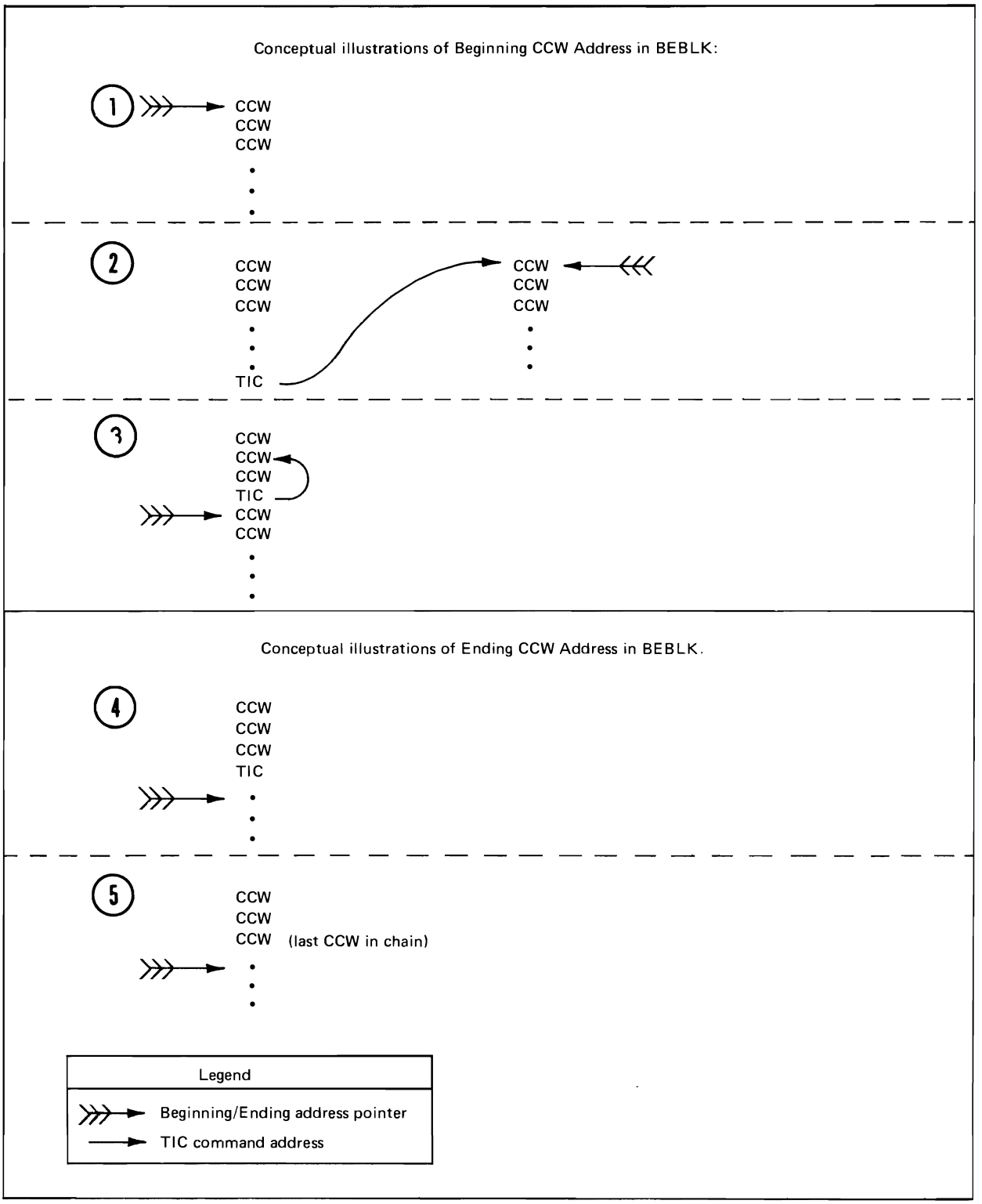


Figure 37 (Part 2 of 2). Beginning and End Block (BEBLK)

Channel Address Word

Initialized by: DOS

Interrogated by: IIVPCE, IIVLOG, IIVAWV, IIVSCI

Pointer in: Emulator register 10 (points to DOS storage) + X'48' offset

The channel address word (Figure 38) is referred to by a channel during execution of a start I/O instruction. It is the means by which the channel can determine the main-storage location from which it should fetch the first channel command word (CCW). The channel address word is permanently assigned to main-storage location 72.

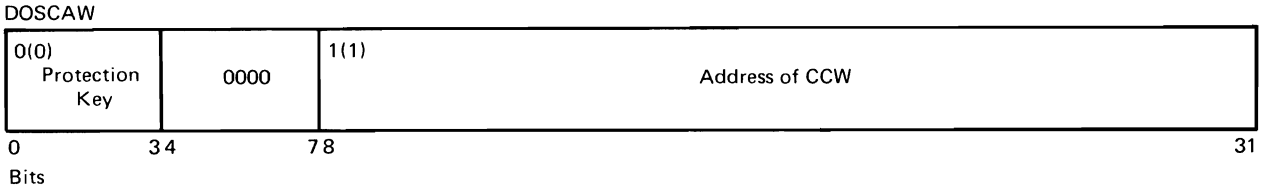


Figure 38. Channel Address Word

Channel Address Word Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0 (0)	1	DOSCAW	Protection key. In systems having the data set protection feature, contains the requestor's protection key. Otherwise, contains zeros.
1 (1)	.3		The main-storage address of the first channel command word (CCW) to be executed.

Channel Command Word

Initialized by: DOS

Modified by: IIVABN, IIVPCE, IIVCCW, IIVADJ, and adjust CCW string instruction

Interrogated by: IIVLOG, IIVSTG, IIVCWV, IIVCCW, IIVVIO, IIVCHK, IIVACI, IIVADJ

Pointer in: DOS CCB + X'08' offset

The channel command word (CCW, Figure 39) indicates to a channel what I/O operation it should start. For operations involving data transfer, the CCW also indicates the main-storage location into which data is to be placed or read from, and how many bytes of data are to be transferred.

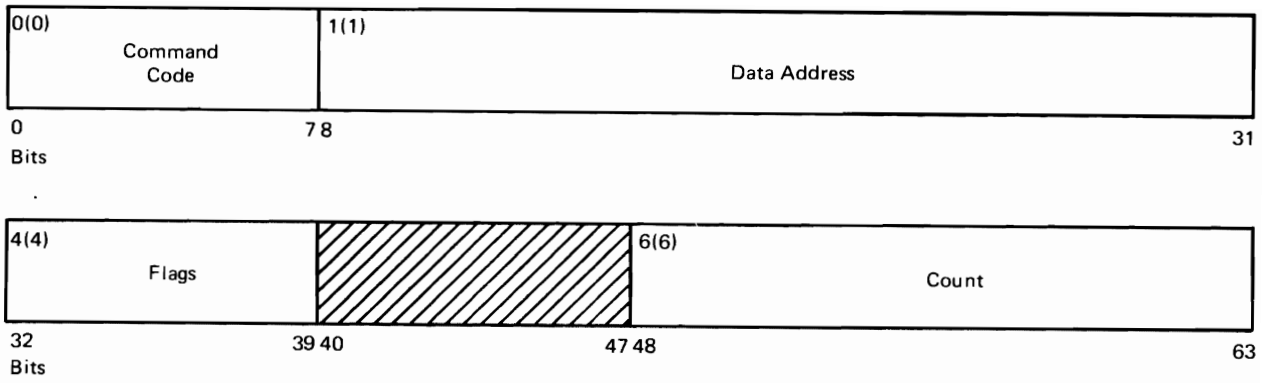


Figure 39. Channel Command Word

Channel Command Word Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0 (0)	1		Command code. Specifies the operation to be performed.
1 (1)	.3		Data address. Specifies the main-storage location of a data area. Depending upon the command code, data is either read from or placed into the data area during an I/O operation involving data transfer.
4 (4)	1	10..	Chain-data (CD) flag. The data area designated by the next CCW is to be used with the current operation.
		01..	Chain-command (CC) flag. The operation specified by the command code of the next CCW is to be initiated on normal completion of the current operation.
		..1.	Suppress-length-indication (SLI) flag. The incorrect length status bit in the channel status word is not to be set by the channel if it detects an incorrect-length condition.
		...1	Skip flag. Data transfer to main storage is to be suppressed. This flag is valid only for read, read backward, and sense operations.
		.... 1...	Program controlled interruption (PCI) flag. Not significant to Emulator.
		.... .000	Must be zeros for every CCW except for those CCWs specifying a transfer in channel operation.
6 (6)	..2		Count. Specifies the number of bytes in the data area associated with this CCW.

Channel Status Word

Initialized by: Emulator

Modified by: OS, IIVPCE, IIVSTG, IIVCWV, IIVAWV, IIVPOV, IIVLOG

Pointer in: Emulator register 10 (points to DOS storage) + X'40' offset

Interrogated by: IIVPCI, IIVACI

Explanation: IIVPCE gets the 7 low-order bytes of the last CSW from the IOBCSW field in the IOB and the high-order byte from the COMKEY field of COMTAB and places it in the CSW location in DOS storage. It is the CSW in the IOB that is modified by OS and the Emulator modules.

The channel status word (CSW, Figure 40) indicates to a program the status of an I/O device, control unit, channel, and subchannel. The CSW is assigned permanently to main-storage location 64. Information is stored in the CSW by a channel after an I/O interruption and also during execution of the start I/O, test I/O, and halt I/O instructions.

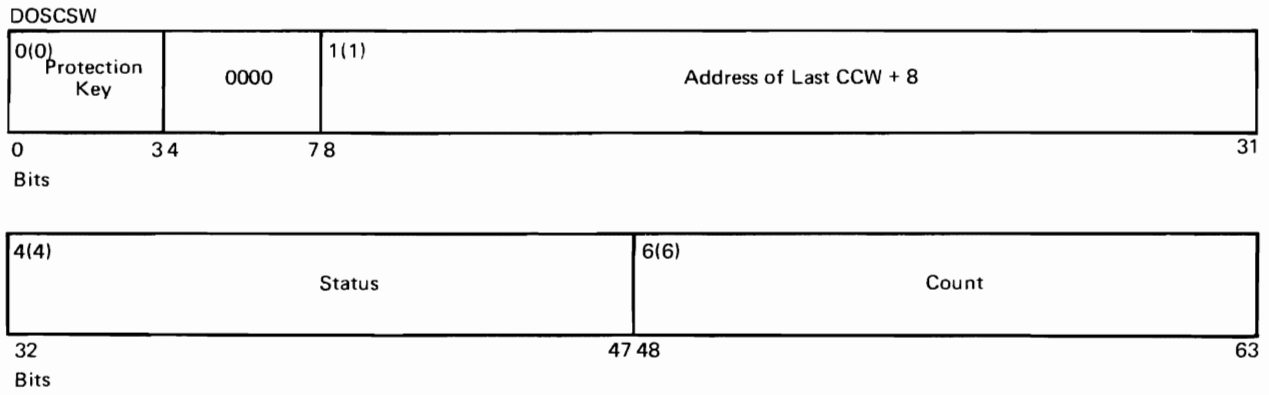


Figure 40. Channel Status Word

Channel Status Word Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1	xxxx	Protection key. In systems having the data set protection feature, contains the requestor's protection key. Otherwise, contains zeros.
		 0000	Always zeros.
1(1)	.3		A main-storage address that is eight higher than the address of the last executed CCW.
4(4)	2	Status byte 1	
		1...	Attention
		.1..	Status modifier
		..1.	Control unit end
		...1	Busy
		.... 1...	Channel end
		.... .1..	Device end
		.... ..1.	Unit check
		1	Unit exception
		Status byte 2	
		1...	Program-controlled interruption
		.1..	Incorrect length
		..1.	Program check
		...1	Protection check
		.... 1...	Channel data check
		.... .1..	Channel control check
		.... ..1.	Interface control check
		1	Chaining check
6(6)	..2		Count. The number of bytes of data that remained to be transferred after the last CCW was executed.

Command Control Block

Initialized by: DOS

Interrogated by: IIVADD, IIVDVS, IIVGR2, IIVPCE

Pointer in: DOS register 1

The DOS command control block (CCB) is used for communication between physical IOCS and the problem program. A 16-byte field is required for each channel program executed by physical IOCS.

Note in Figure 41 that byte 2, bit 0, is significant to the Emulator. Bit 0 is normally set on at channel end to signify that the I/O operation was completed. Bytes 9-11, also significant to the Emulator, contain the address of the CCW associated with this CCB.

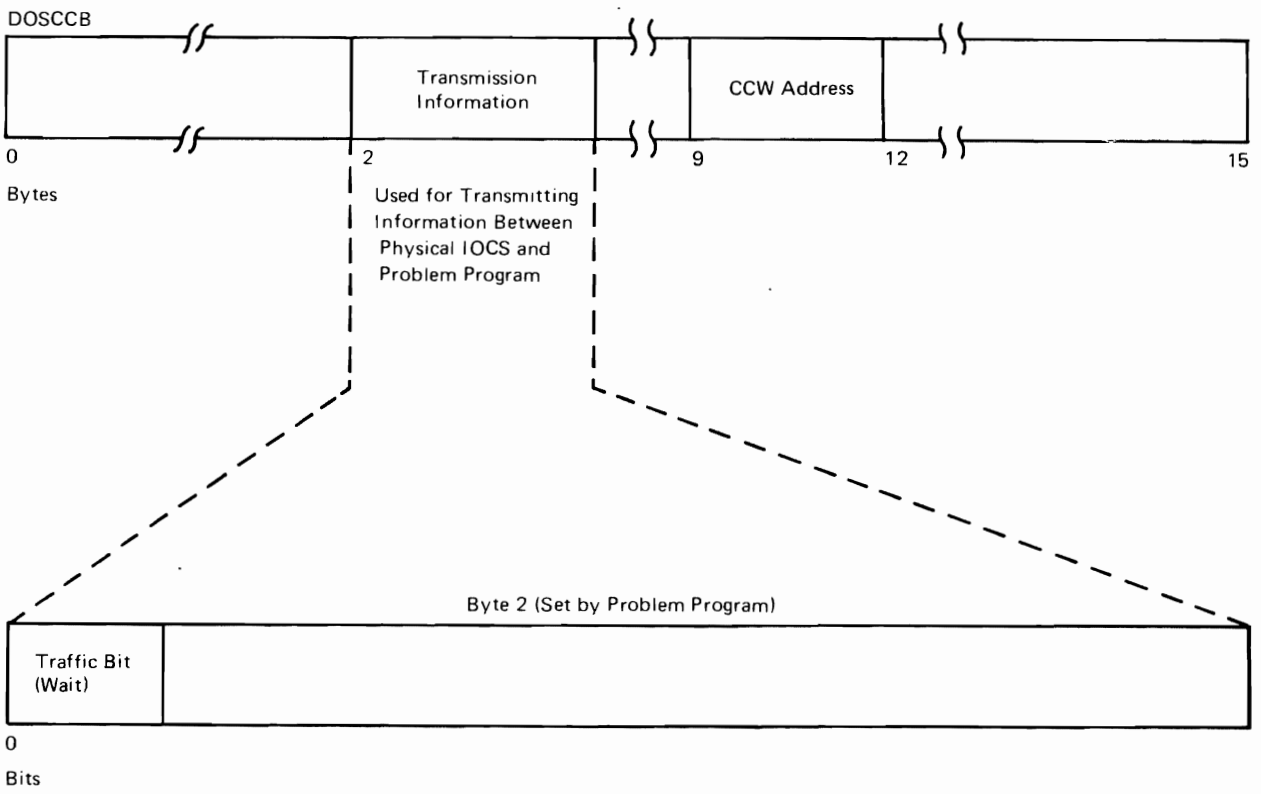


Figure 41. Command Control Block Field Used by the Emulator

Communications Table

Initialized by: IIVINT

Modified by: IIVOPN, IIVPCE, IIVSTG, IIVAWV, IIVABN, IIVCHK, IIVINT

Interrogated by: IIVRTE, IIVPRP, IGG019S1, IGG019SA, IIVGR2, IIVACI, IIVIN2

Pointer in: EMUCONS

The communications table (COMTAB, Figure 42) has one entry for every device allocated to emulation plus an entry for SYSLOG. Each entry is formatted as illustrated in Figure 42.

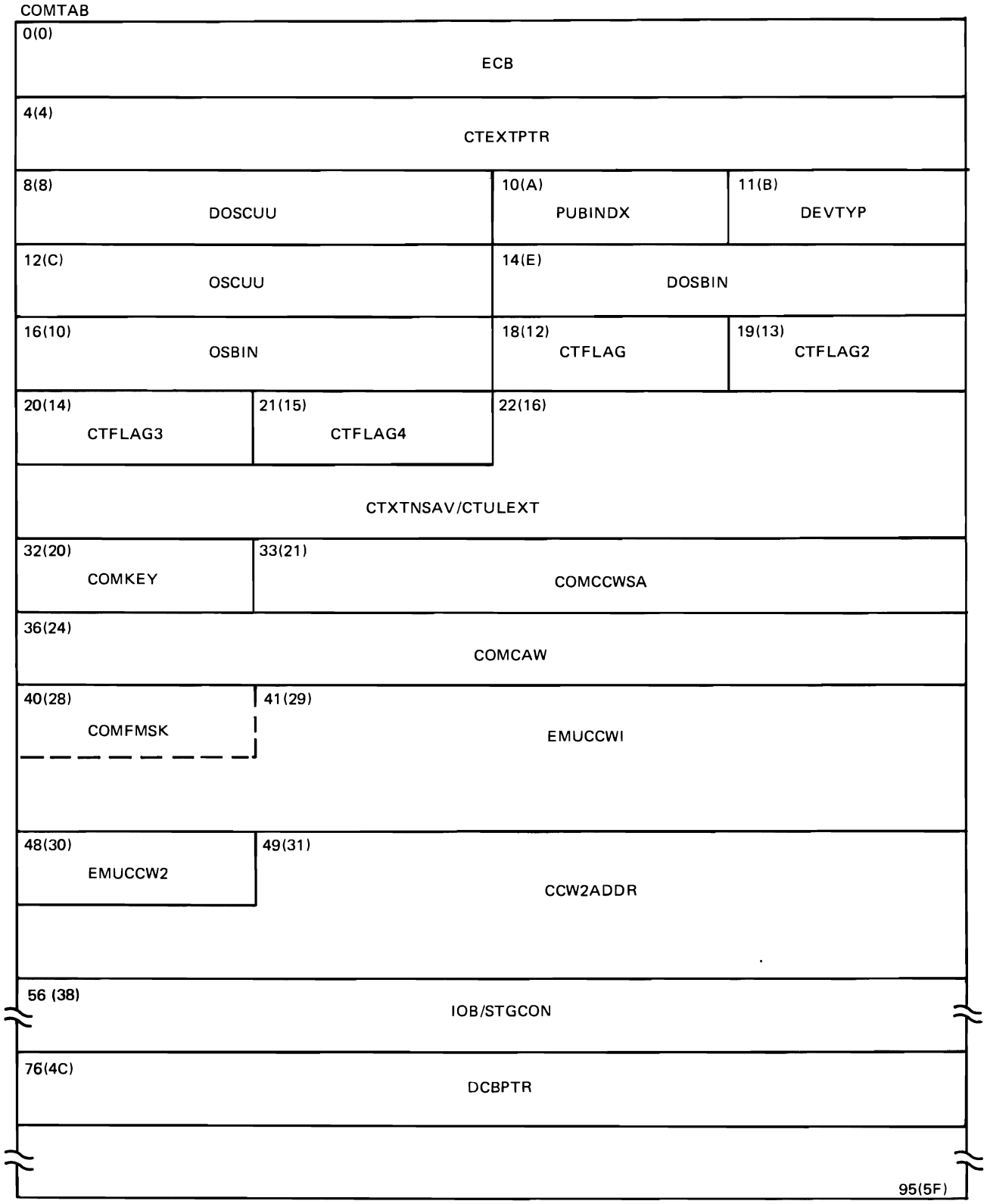


Figure 42. Communications Table

Communications Table Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0 (0)	4	ECB	The actual ECB for this device.
4 (4)	4	CTEXTPTR	The pointer to the COMTAB extension for this entry.
8 (8)	2	DOSCUU	The DOS channel and unit address as known by the DOS supervisor.
10 (A)	..1	PUBINDX	The index into the DOS-PUB table where the PUB entry for this DOS device can be found. The first PUB entry is index value zero.
11 (B)	...1	DEVTYP	The one-byte DOS device type as given in the DOS-PUB table.
12 (C)	2	OSCUU	The OS channel and unit address.
14 (E)	..2	DOSBIN	The DOS 2321 bin number from the DDname.
16 (10)	2	OSBIN	The OS 2321 bin number from the UCB.
18 (12)	..1	CTFLAG	Communications flag. 1... .... Entry for a staged device. .1.. .... Entry for DOS SYSLOG. ..1. .... Check I/O to test I/O exit flag. ...1 .... Device end flag. .... 1... Sense switch. .... .1.. Device in use .... ..1. Nonoperational flag. ....1 Device is open.
19 (13)	...1	CTFLAG2	Communications flag 2. 1... .... Tape device. .1.. .... Direct-access device. ..1. .... Optical character reader. ...1 .... Staged SYSIN device. .... 1... Staged SYSOUT device. .... .1.. 2321 device. .... ..1. Do not suppress error retry. ....1 File protection flag.
20 (14)	1	CTFLAG3	Communications flag 3 1... .... Use OS error recovery procedures for this I/O request. .1.. .... Shared volume indicator. ..1. .... Indexed sequential data set. ...1 .... User label swap switch. .... 1... Telecommunications device. .... .1.. Seven-track tape unit. .... ..1. NOP-issued flag. ....1 Stand-alone seek flag.

Communications Table Description (Continued)

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
21(15)	.1	CTFLAG4	Communications flag 4.
			1... .... Disposition is SHR.
			.1.. .... Not ready mask.
			..1. .... DOS resident device.
			.... 1... Restore DEB extents indicator.
			...x .x.x Not used.
			.... ..1. Stand-alone CCW (not chained) set mode issued.
22(16)	..10	CTXTNSAV	Extent from DEB at SIO time.
		or	
22(16)	..10	CTULEXT	User label extent at DOS open time.
32(20)	1	COMKEY	Save area used to pass storage protection key at SIO time to DOS (during I/O interruption time).
33(21)	.3	COMCCWSA	Save area for CCW command address; nonzero value implies NOP has been issued by Emulator.
36(24)	4	COMCAW	Pointer to CCW command chain to be adjusted from local to true address.
40(28)	1	COMFMSK	File mask for DOS DASD.
		or	
40(28)	8	EMUCCW1	Emulator area for set-mode CCW for tape.
48(30)	1	EMUCCW2	Emulator area for TIC CCW for tape commands.
49(31)	.7	CCW2ADDR	The first three bytes contain the address portion of TIC CCW for tape; the last four bytes, the remainder of the TIC CCW.
56(38)	40	IOB/ STGCON	The actual IOB (including a 4-byte field called DCBPTR, the DCB pointer) or STGCON in the case of a staged device.
76(4C)	4	DCBPTR	(See IOB/STGCON field)

COMTAB Extension

Initialized by: IIVINT

Modified by: IIVDVS, IIVIS

Interrogated by: IIVPCE, IIVGR2, IIVDVS, IIVIS

Pointer in: COMTAB

The COMTAB extension (CTEXT, Figure 43) has one entry for every file residing on a shared direct-access volume. If a COMTAB entry is marked for direct-access device sharing, then at least one COMTAB extension entry exists that is related to the COMTAB entry. Each entry is formatted as illustrated in Figure 43.

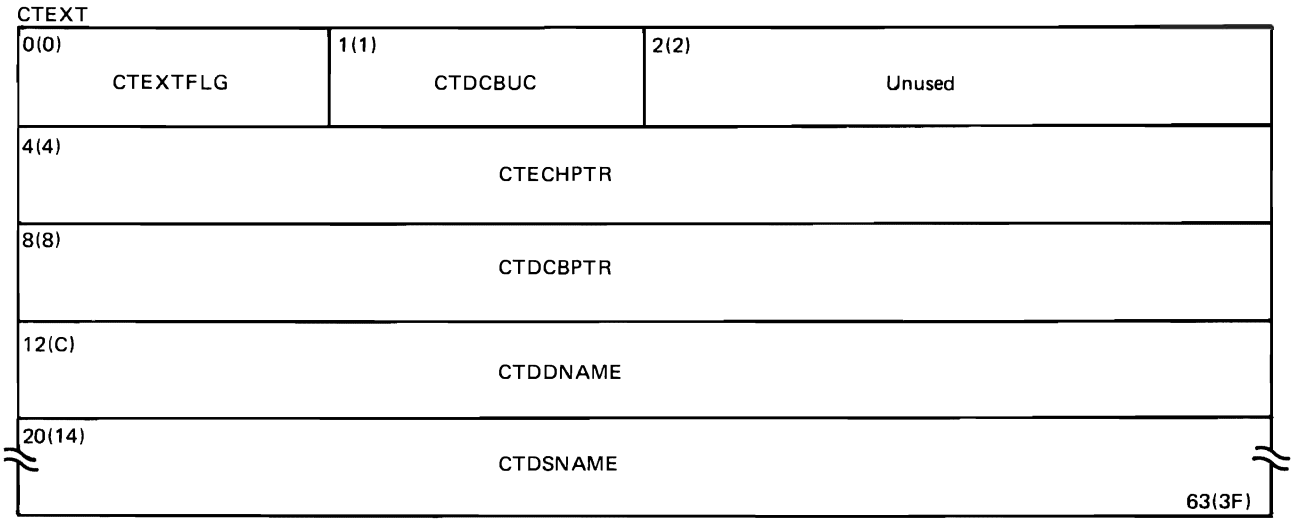


Figure 43. COMTAB Extension

COMTAB Extension Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1	CTEXTFLG	COMTAB extension flag. 1... .... Indexed sequential data set. .1.. .... End of CTEXT chain indicator. ..xx xxxx Not used.
1(1)	.1	CTDCBUC	DCB use count.
4(4)	4	CTECHPTR	Pointer to the next COMTAB extension on this device.
8(8)	4	CTDCBPTR	Pointer to the DCB for this data set.
12(C)	8	CTDDNAME	DD (data definition) name.
20(14)	44	CTDSNAME	DS (data set) name.

DASD Label (DLBL)

Initialized by: DOS

Modified by: DOS, IIVDVS, IIVIS

Interrogated by: IIVGR2, IIVDVS, IIVIS, DOS

Pointer in: DOS open table

The DLBL represents the internal format of a DOS DLBL extent statement read from the label cylinder by DOS OPEN phases. Figure 44 shows the fields in the DLBL that are significant to the Emulator and the field description tells which modules modify or interrogate each field.

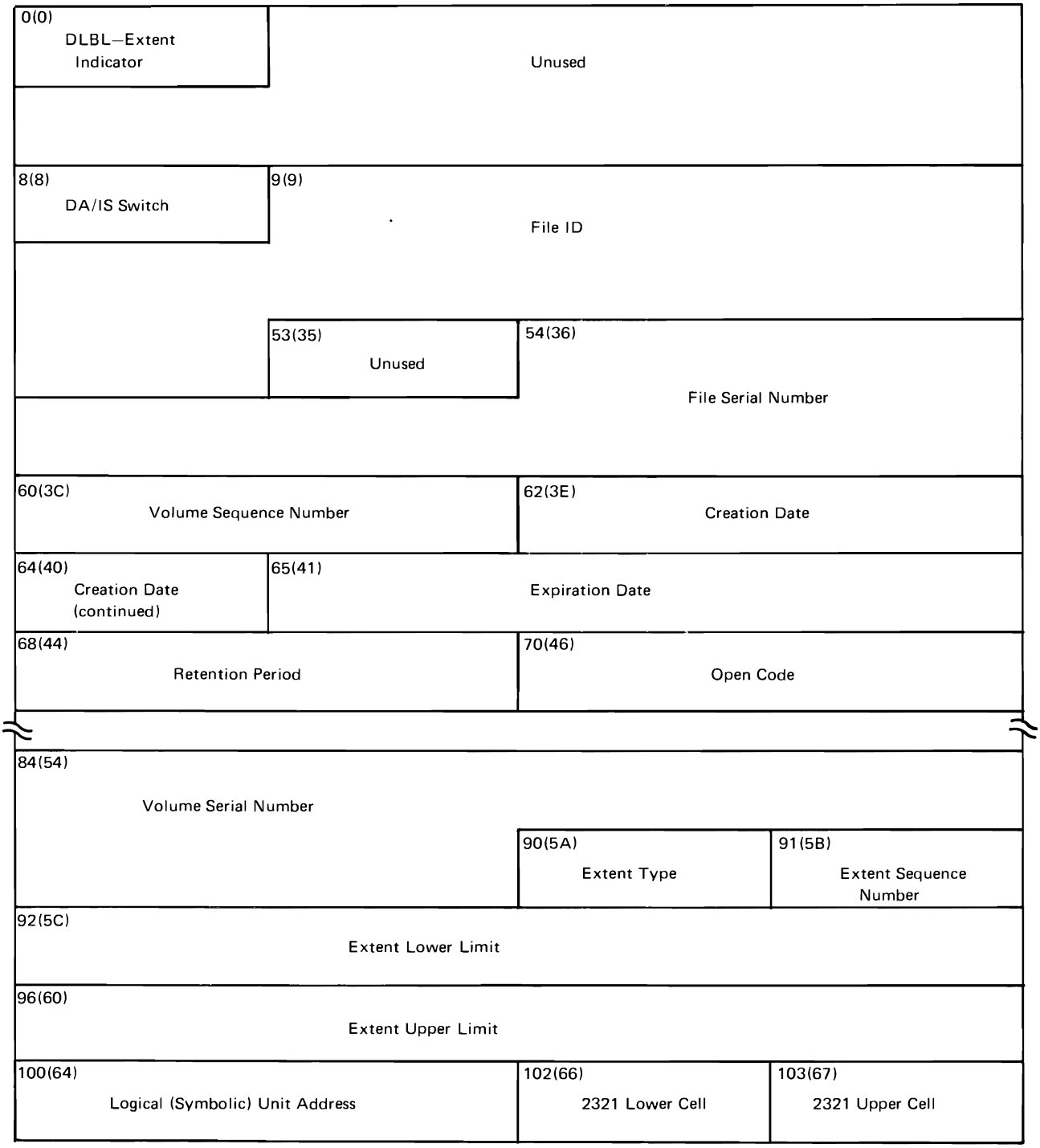


Figure 44. DASD Label

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1	DLBL-EXTENT	SD (sequential disk) 1... .... Next extent on a new pack. .1.. .... Last extent. ..1. .... Bypass extent. ...1 .... New volume on same unit. .... 1... Extent limits omitted. .... .1.. Extent converted to DASD address. .... ..1. No EXTENT/XTENT card. ....x Not used. For direct access or indexed sequential file management system, this byte indicates the number of extents. Modified/interrogated by IIVDVS.
8(8)	1	DA/IS switch	xxxx ..xx Not used. .... 1... Extent limits omitted. .... .1.. Extent converted to DASD address. Modified by IIVDVS.
9(9)	44	File ID	File identifier including generation and version numbers. If field is missing on DLBL card, file name padded with blanks is inserted. Interrogated by IIVGR2.
54(36)	6	File serial number	Vclume serial number from first extent. Modified by IIVDVS.
60(3C)	2	Volume sequence number	Always initialized to X'0001'. Modified by IIVDVS.
63(3E)	3	Creation date	Initialized with 3 bytes of X'00'. Modified by IIVDVS.
65(41)	3	Expiration date	If date is in the form YYDDD, it is converted to YDD. If date is in retention period form, 1 to 4 characters, the field is padded with binary zeros. Modified by IIVDVS.
68(44)	2	Retention period	Converted to a 2-byte number and inserted in this field. Modified by IIVDVS.
70(46)	1	Open code	DLBL type: S = sequential D = direct access C or E = indexed sequential file management system where: C = load create function E = load extend function Interrogated by IIVIS.
84(54)	6	Volume serial number	Volume serial number for extent. Modified by IIVDVS.

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
90(5A)	1	Extent Type	Same codes as in Format 1 label: X'00' = next three fields do not indicate any extent. X'01' = prime data area (ISFMS) or consecutive area, etc. (that is, the extent containing the user's data records). X'02' = Overflow area of an ISFMS file. X'04' = Cylinder index or master index of an ISFMS file. X'40' = User label track area. X'8n' = Shared cylinder indicator, where n = 1, 2, or 4. Modified by IIVDVS.
91(5B)	1	Extent sequence number	Number of extents as determined by the extent card sequence. Modified by IIVDVS.
92(5C)	4	Extent lower limit and	Before the OPEN, DLBL/EXTENT information is in the relative track form of HHNNT followed by three bytes of binary zeros.
96(60)	4	extent upper limit	HH = Relative (to 0) start address in tracks. NN = Number of tracks. T = 0 or upper track number for split cylinder in SD files. Following an OPEN on DLBL/EXTENT cards, or whenever DLAB/XTENT cards are used, the extent lower and upper limits are each in the CCHH format. Modified by IIVDVS.
100(64)	2	Logical (symbolic) unit address	This 2-byte field identifies the logical unit with the same code as that used in a CCB. The first byte identifies the unit class: X'00' = System logical unit X'01' = Programmer logical unit The second byte identifies the logical unit within its class. Thus X'0003' denotes SYSLST and X'0103' denotes SYS003. Interrogated by IIVGR2 and IIVDVS.
102(66)	1	2321 lower cell	2321 extent lower and upper cell limit. This 2-byte field contains zeros for 2311/2314/2319 disk. Modified by IIVDVS.
103(67)	1	2321 upper cell	

Data Control Block

Initialized by: IIVOPN (EXCP, QTAM, BTAM), IIVIS (ISAM), IIVDVS

Defined by: IIVRAS (BSAM)

Modified/interrogated by: IIVSTG, IIVINT, IIVIN2, IIVOPN, IIVPCE (EXCP, QSAM, BTAM), IIVIS (ISAM), IIVSNP (BSAM)

Pointer in: IOB, COMTAB extension

Data control blocks (DCBs) describe the current use of a data set. For every data set to be processed by a program, there is a corresponding DCB.

The foundation sections serve the same purpose in all DCBs although the formats may vary slightly for different access method routines. Although they are maintained primarily by data management routines, certain fields in the DCBs do contain a limited amount of information maintained or referred to by the Emulator. Figure 45 illustrates the format of this DCB. Descriptions of the fields follow the illustration.

Device Interface (ISAM only)

16(10) DCBKEYLE	Unused

Foundation Extension

Unused	33(21) DCBEODAD
36(24) DCBRECFM	Unused

Foundation

Before Open

40(28) DCBDDNAM		
48(30) DCBOFLGS	Unused	50(32) DCBMACR

After Open

44(2C) DCBIFLGS	Unused

Figure 45 (Part 1 of 2). Data Control Block Fields Used by the Emulator

Access Method Interface – ISAM, BSAM, QSAM

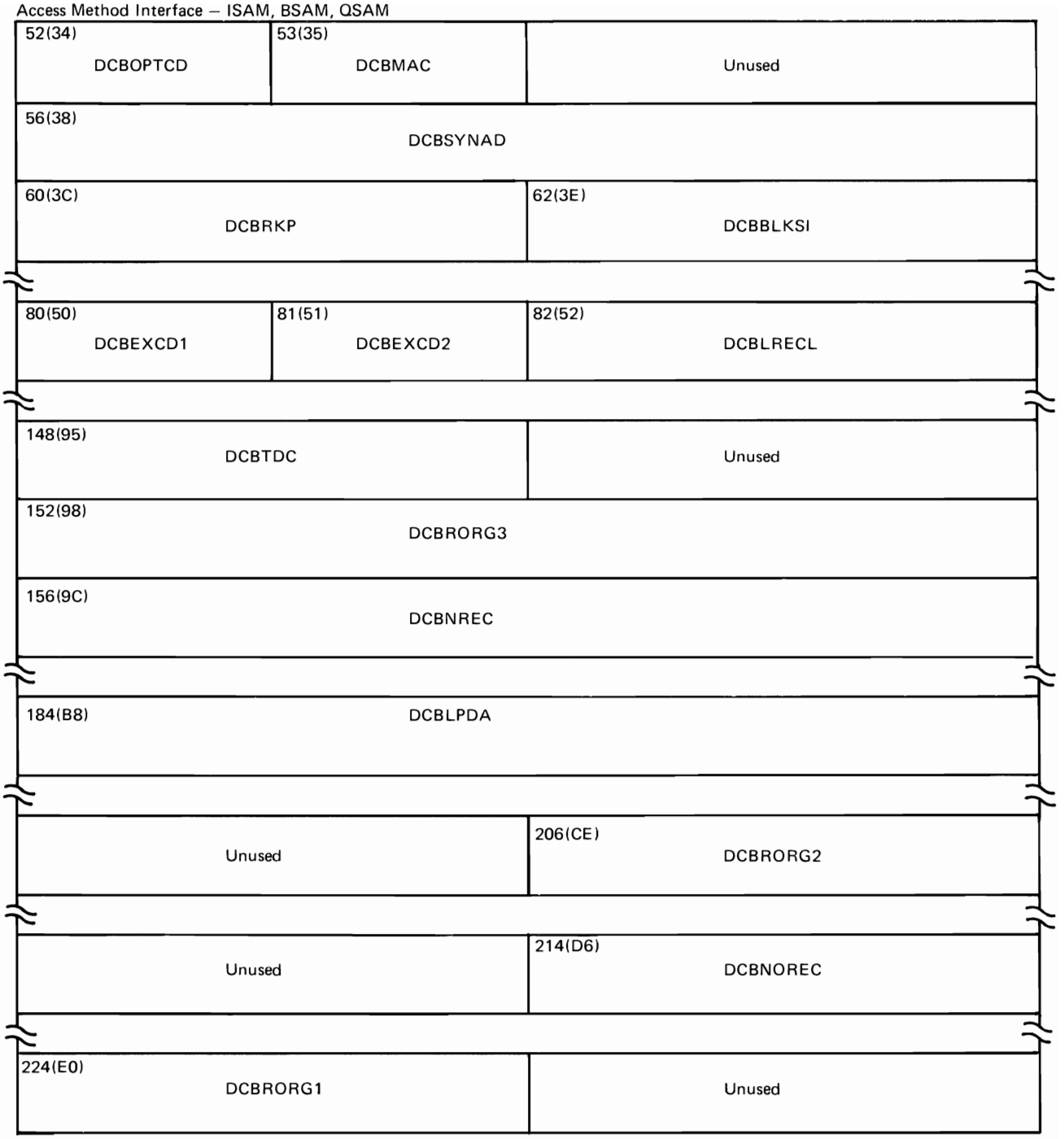


Figure 45 (Part 2 of 2). Data Control Block Fields Used by the Emulator

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
16(10)	1	DCBKEYLE	Key length (used by ISAM).
33(21)	.3	DCBEODAD	EODAD exit address (used by ISAM and QSAM).
36(24)	1	DCBRECFM	Record format (used by ISAM).
40(28)	8	DCBDDNAM	DD statement data set name (used by IIVINT, IIVIN2, IIVOPN and ISAM).
44(2C)	1	DCBIFLGS	Permanent error condition is indicated when bits 0 and 1 are on (used by EXCP).
48(30)	1	DCBOFLGS	Successful open is indicated when bit 3 is on (used by IIVOPN and ISAM).
50(32)	..2	DCBMACR	Macro instruction code (used by ISAM).
52(34)	1	DCBOPTCD	Option code (used by ISAM).
53(35)	.1	DCBMAC	Extension of the macro instruction operation code field (used by ISAM).
56(38)	4	DCBSYNAD	SYNAD exit address (used by ISAM).
or			
57(39)	.3	DCBSYNAD	Staged I/O SYNAD exit address (used by QSAM).
60(3C)	2	DCBRKP	Relative key position (used by ISAM).
62(3E)	..2	DCBBLKSI	Blocksize (used by ISAM and BSAM).
80(50)	1	DCBEXCD1	Condition flags (used by ISAM).
81(51)	.1	DCBEXCD2	Condition flags (used by ISAM).
82(52)	..2	DCBLRECL	Logical record length for variable length records (used by ISAM, BSAM, and QSAM).
148(95)	2	DCBTDC	Tag deletion count (used by ISAM).
152(98)	4	DCBRORG3	Count of accesses to overflow records other than the first (used by ISAM).
156(9C)	4	DCBNREC	Number of logical records in prime data area (used by ISAM).
184(B8)	8	DCBLPDA	Direct-access address of last prime data record in prime data area (used by ISAM).
206(CE)	..2	DCBRORG2	Number of tracks left in overflow area (used by ISAM).
214(D6)	..2	DCBNOREC	Number of logical records in overflow area (used by ISAM).
224(E0)	2	DCBRORG1	Number of full cylinder overflow areas (used by ISAM).

Data Event Control Block -- BISAM

Initialized by: IIVIS

Modified/interrogated by: IIVIS

Pointer in: Register 1 at WRITE and CHECK time

The data event control block (DECB) is created when a READ or WRITE macro instruction is expanded. It contains information about the input or output operation requested by the macro instruction. Figure 46 shows the DECB fields used in BISAM that are significant to the Emulator.

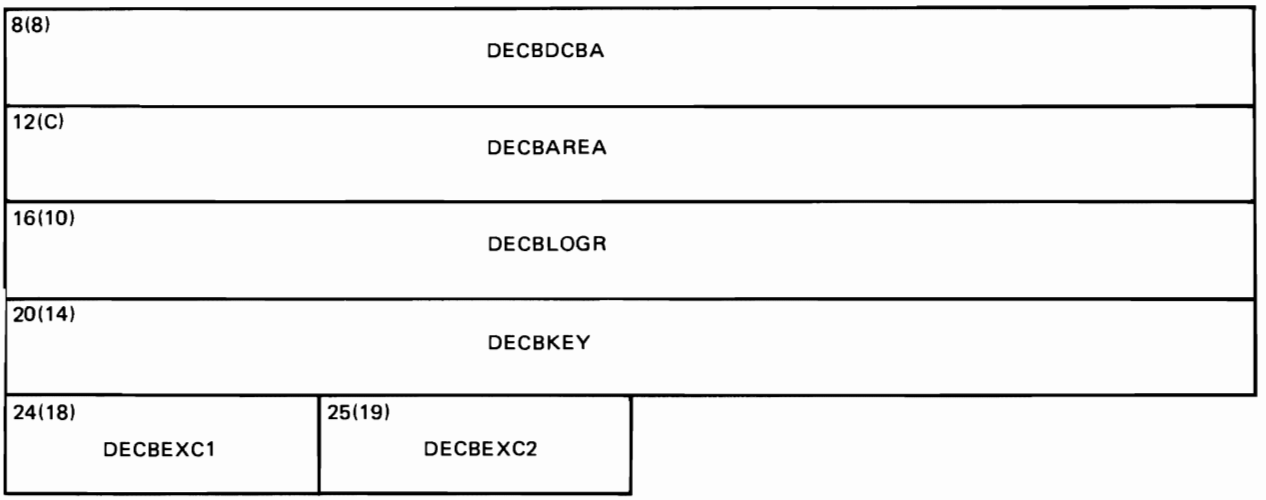


Figure 46. Data Event Control Block -- BISAM Fields Used by the Emulator

Data Event Control Block -- BISAM Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
8 (8)	4	DECBDCBA	Address of the DCB to which this I/O request is related.
12 (C)	4	DECBAREA	Address of the area in storage for the record.
16 (10)	4	DECBLOGR	Address of the logical record (also used by IIVIS in QISAM mode to store the current logical record).
20 (14)	4	DECBKEY	Address of the key portion of the record.
24 (18)	1	DECBEXC1	Exceptional condition code.
			1... .... Record not found.
			.1.. .... Record length check.
			..1. .... Space not found in which to add a record.
			...1 .... Invalid request.
			.... 1... Uncorrectable I/O error.
			.... .1.. Unreachable block.
			.... ..1. Overflow record.
			....1 Duplicate record presented for inclusion in the data set.

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
25(19)	.1	DECBEXC2	Exceptional condition code.
			.... ..1. Execution of the last channel program was instituted by an asynchronous routine.
			....1 Previous macro instruction was READ KU.
			xxxx xx.. Reserved bits.

Data Extent Block

Initialized by: OS

Modified by: IGG019SA

Interrogated by: IIVOPN, IIVDVS, IIVPCE

Pointer in: DCB

The data extent block (DEB) contains an extension of information in the data control block (DCB). Every DEB is associated with a DCB, and the two point to each other. The DEB contains information concerning the physical characteristics of a data set.

Each DEB consists of one 32-byte base plus:

- One 4-byte extension if the data set is to be processed on unit-record or magnetic tape devices.
- One 16-byte extension for each extent if the data set is to be processed on a direct-access device.

Figure 47 shows the DEB fields used by the Emulator. A complete description of the DEB is contained in the publication System Control Blocks for OS or System Data Areas for OS/VS.

Appendage Table[1]

-36(-24)	DEBEOEA
-32(-20)	DEBSIOA
≈	≈
-24(-18)	DEBCEA
-20(-14)	DEBXCEA

Direct-Access Storage Device Section [2]

+32(+20) DEBDVMOD	Unused

+36(+24) DEBBINUM	+38(+26) DEBSTRCC
+40(+28) DEBSTRHH	+42(+2A) DEBENDCC
+44(+2C) DEBENDHH	+46(+2E) DEBNMTRK

[1] The Emulator includes the start-I/O appendage, channel-end appendage, and abnormal-end appendage routines.

[2] The start-I/O appendage modifies these fields to extend the extent to cover the whole DASD volume.

Figure 47. Data Extent Block (Ordinary) Fields Used by the Emulator

Data Extent Block Description

Offset	Bytes and Field Alignment	Name	Field Description, Contents, Meaning
			APPENDAGE TABLE
-36(-24)	4	DEBEOEA	Address of the end-of-extent appendage
-32(-20)	4	DEBSIOA	Address of the start I/O appendage routine.
-24(-18)	4	DEBCEA	Address of the channel end appendage routine.
-20(-14)	4	DEBXCEA	Address of the abnormal end appendage routine.
			DIRECT-ACCESS STORAGE DEVICES
+32(+20)	1	DEBDVMOD	Device modifier: file mask

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
+36(+24)	2	DEBBINUM	Bin number.
+38(+26)	..2	DEBSTRCC	Cylinder address for the start of an extent limit.
+40(+28)	2	DEBSTRHH	Read/write track address for the start of an extent limit.
+42(+2A)	..2	DEBENDCC	Cylinder address for the end of an extent limit.
+44(+2C)	2	DEBENDHH	Read/write track address for the end of an extent limit.
+46(+2E)	..2	DEBNMTRK	Number of tracks allocated to a given extent.

Data Set Control Block -- Identifier (Format 1)

Initialized by: OS space allocation routine or DOS open routine

Modified by: OS, DOS

Interrogated by: OS, DOS, IIVDVS, IIVVIO, IIVOPN

Located in: VTOC

This data set control block (DSCB) describes the characteristics and up to three extents of a data set. Figure 48 shows the format of the fields of a format 1 DSCB that are significant to the Emulator. Descriptions of the fields follow the illustration.

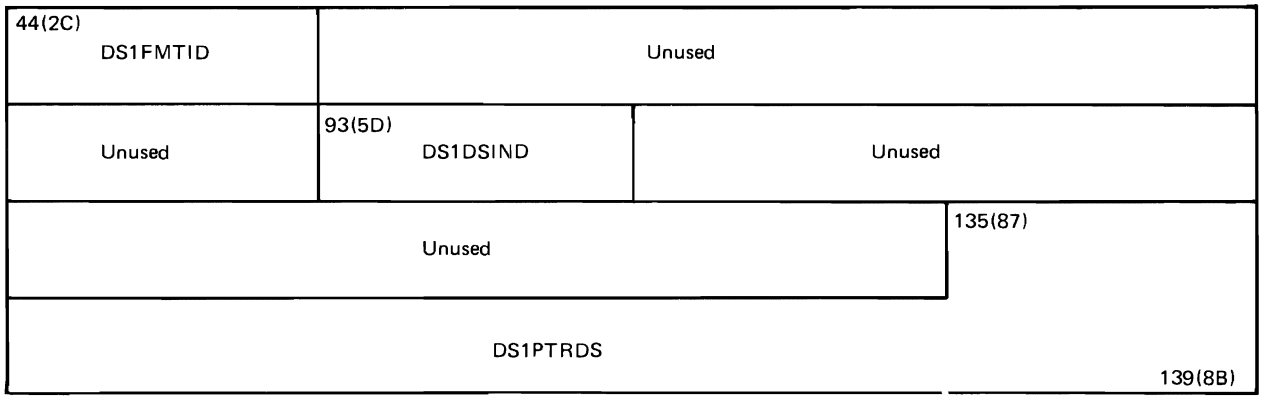


Figure 48. Data Set Control Block -- Identifier (Format 1) Fields Used by the Emulator

Data Set Control Block -- Format 1 Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
44(2C)	1	DS1FMTID	Format identifier; hexadecimal F1 identifies this as a format 1 DSCB.
93(5D)	.1	DS1DSIND	Data set indicators. 1... .... This is the last volume on which this data set normally resides ..1. .... Block length must always be a multiple of 8 bytes. ...x .x.. Data set security. ...1 .0.. Password is requested to read or write. ...1 .1.. Password is required to write but not to read. .x.. x.xx Reserved bits.
135(87)	...5	DS1PTRDS	Pointer to an index (format 2) DSCB, if data set has IS organization, or pointer to an extension (format 3) DSCB if data set has sequential or direct organization and more than 3 extents. This pointer has the format CCHHR. Contains binary zeros if no additional DSCB is pointed to.

Data Set Control Block -- Extension (Format 3)

Initialized by: OS space allocation routine or DOS open routine

Modified by: OS, DOS

Interrogated by: OS, DOS, IIVDVS, IIVVIC, IIVOPN

Pointer in: DSCB -- identifier (format 1)

This data set control block (DSCB) describes up to 13 additional extents that cannot be described in an identifier (format 1) DSCB. All fields are significant to the Emulator. It is pointed to by an identifier (format 1) DSCB. (For format see System Control Blocks for OS or System Data Areas for OS/VS.)

Data Set Control Block -- VTOC (Format 4)

Initialized by: volume-initializing utilities

Modified by: OS, DOS

Interrogated by: OS, DOS, IIVOPN

Identified by: A 44-byte entry of X'04'

This data set control block (DSCB) describes the volume table of contents (VTOC) data set. It is always the first DSCB in the VTOC. Figure 49 shows the format of the fields a VTOC (format 4) DSCB that are significant to the Emulator. Descriptions of the fields follow the illustration.

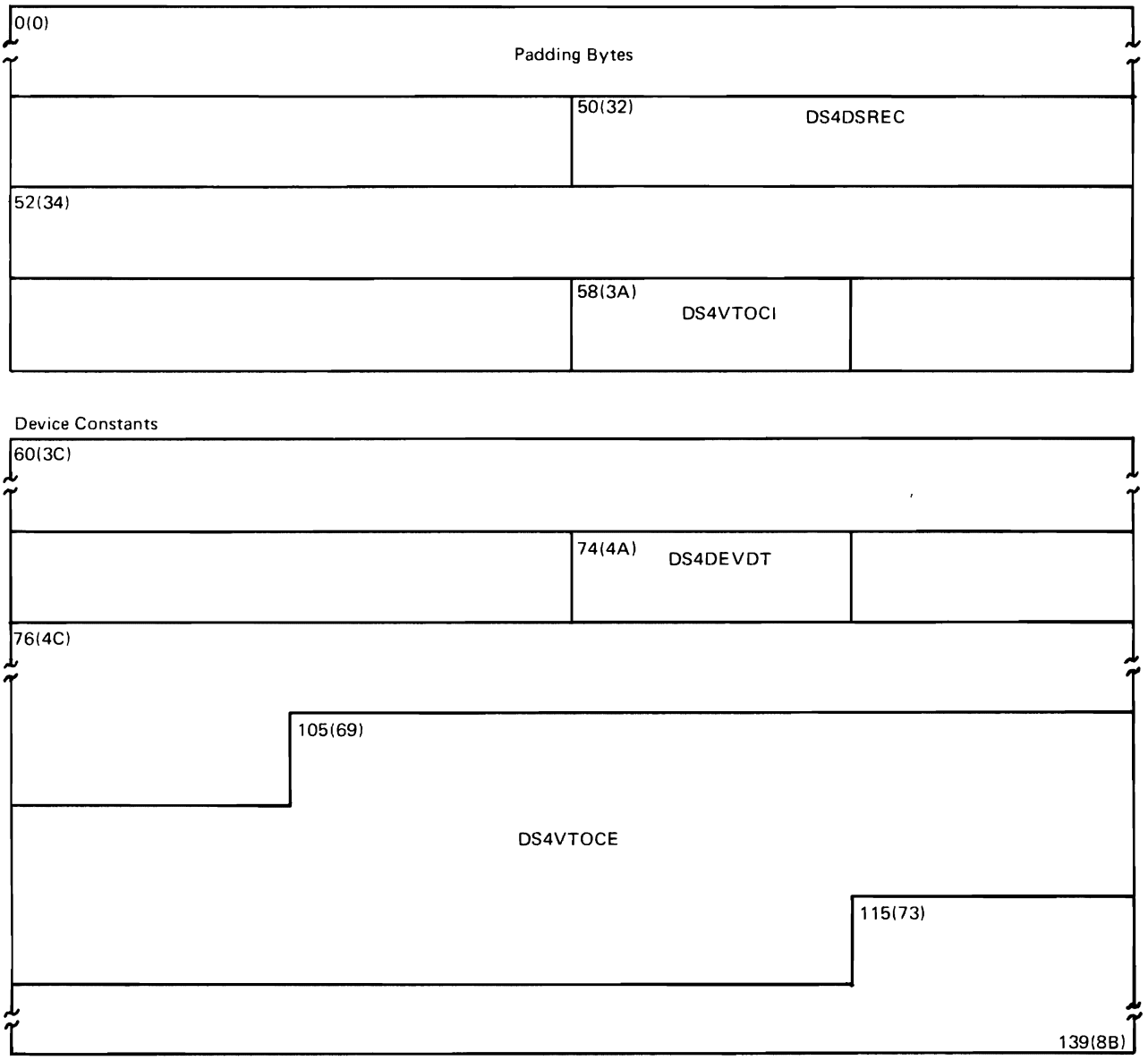


Figure 49. Data Set Control Block -- VTOC (Format 4) Fields Used by the Emulator

Data Set Control Block -- Format 4 Description

Offset	Bytes and Alignment	Field Name	Hex. Dig.	Field Description, Contents, Meaning
0(0)	44	(Padding Bytes)		Hexadecimal 04 in each byte.
50(32)	..2	DS4DSREC		Number of available free VTOC record (format 0) DSCBs in the VTOC.
58(3A)	..1	DS4VTOCI		VTOC indicators.
		1...		Either no free space (format 5) DSCBs exist, or they do not reflect the true status of the volume.
		.... 1...		Accurate free space and shared extent (format 5 and 6) DSCBs now exist, and bit 0 has been turned off. This volume may contain data sets produced by the Disk Operating System; the Operating System access methods may not be able to process these data sets.
		.... .1..		A DADSM function has been prematurely terminated. Possible VTOC errors exist.
		.xxx ..xx		(Reserved bits)
74(4A)	..1	DS4DEVDT		Number of full DSCBs that can be contained on one track (44-byte key plus 96-byte data length).
105(69)	.10	DS4VTOCE		Extent description of the VTOC.
105(69)		Byte 1		Data set extent type indicator.
			00	Following 9 bytes do not indicate any extent.
			01	The extent contains the data blocks (user's blocks) or is a prime area for indexed sequential data sets).
			02	The extent is an overflow area (for indexed sequential data sets only).
			04	The extent is an index area (for indexed sequential data sets only).
			40	The first extent description describes the user label extent.
			80	The extent described is sharing one or more cylinders with one or more data sets.
			81	The extent described begins and ends on cylinder boundaries, that is, the extent is composed of one or more cylinders.
106(6A)		Byte 2		Extent sequence number (M)
				Uniquely identifies each separate extent on a given volume for a data set. For all organizations

Offset	Bytes and Alignment	Field Name	Hex. Dig.	Field Description, Contnets, Meaning
				but indexed sequential, the first extent of the data set on each volume is identified with zero in this field. The first extent on each volume of an indexed sequential data set is identified with a value of one in the field. Additional extents on the volume are identified with sequentially increasing binary values. This field is always zero for an extent field pointing to a user label track.
107(6B)		Bytes 3-6		Lower limit of this extent (CCHH) Contains the cylinder and the track address specifying the starting point of this extent.
111(6F)		Bytes 7-10		Upper limit of this extent (CCHH) Contains the cylinder and track address specifying the ending point of this extent.

Diagnostic Block (DIAG Block)

Initialized by: IIVRCP

Modified by: IIVRCP, IIVRCW

Interrogated by: IIVRCW, IIVRCP

Pointer in: RASCONS (RASDIAG)

The diagnostic block (Figure 50) is used to adjust and readjust the data address portion of the CCWs from DOS local addresses to OS true addresses, then back to DOS local addresses.

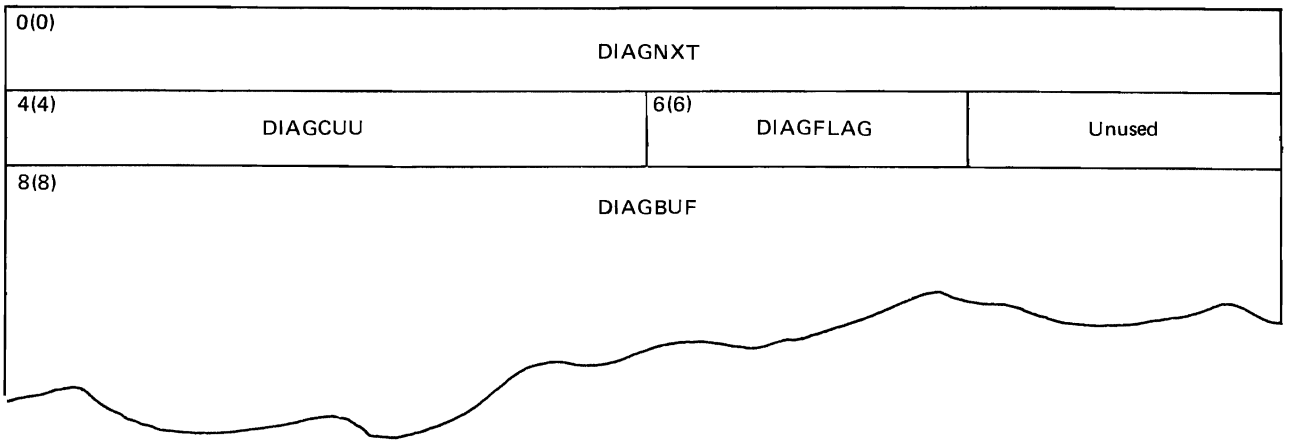


Figure 50. Diagnostic Block

Diagnostic Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	4	DIAGNXT	Pointer to the next diagnostic block.
4(4)	2	DIAGCUU	The DOS cuu for the channel program.
6(6)	1	DIAGFLAG	X'80' if relocation of CCWs is done.
8(8)	104	DIAGBUF	Buffer area for the CCWs.

DTFIS ADD-RETRVE-ADDRTR and DTFIS Load Tables

Initialized by: IIVIS

Modified by: IIVIS

Interrogated by: IIVIS, IIVGR2

Pointer in: DOS register 1 when a DOS I/O macro is issued

When the DTFIS (define the file for indexed sequential) macro instruction is encountered at assembly time, the assembler builds a DTF table tailored to the DTF parameters. The table contains:

- A device CCB.
- A V-type statement used by the linkage editor to resolve the linkage to the logic module associated with this DTF.
- Logic indicators such as one I/O area, two I/O areas, and device type.
- Addresses of all of the areas and controls used by this device (except work files).

Those parts of the DTFIS ADD-RETRVE-ADDRTR table and the DTFIS Load table that are significant to the Emulator as shown in Figures 51 and 52, respectively.

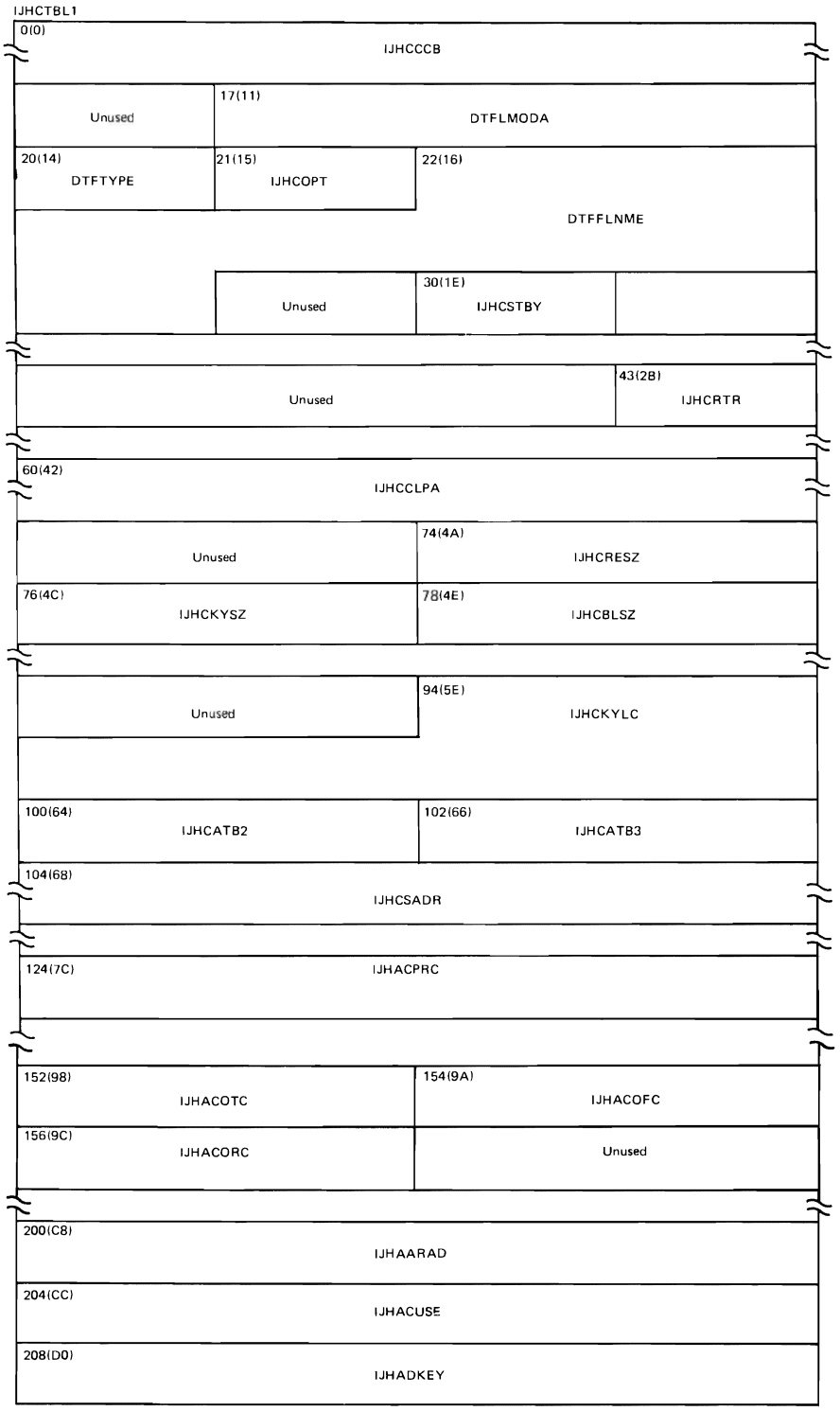


Figure 51 (Part 1 of 2). DTFIS ADD-RETRVE-ADDRTR Table

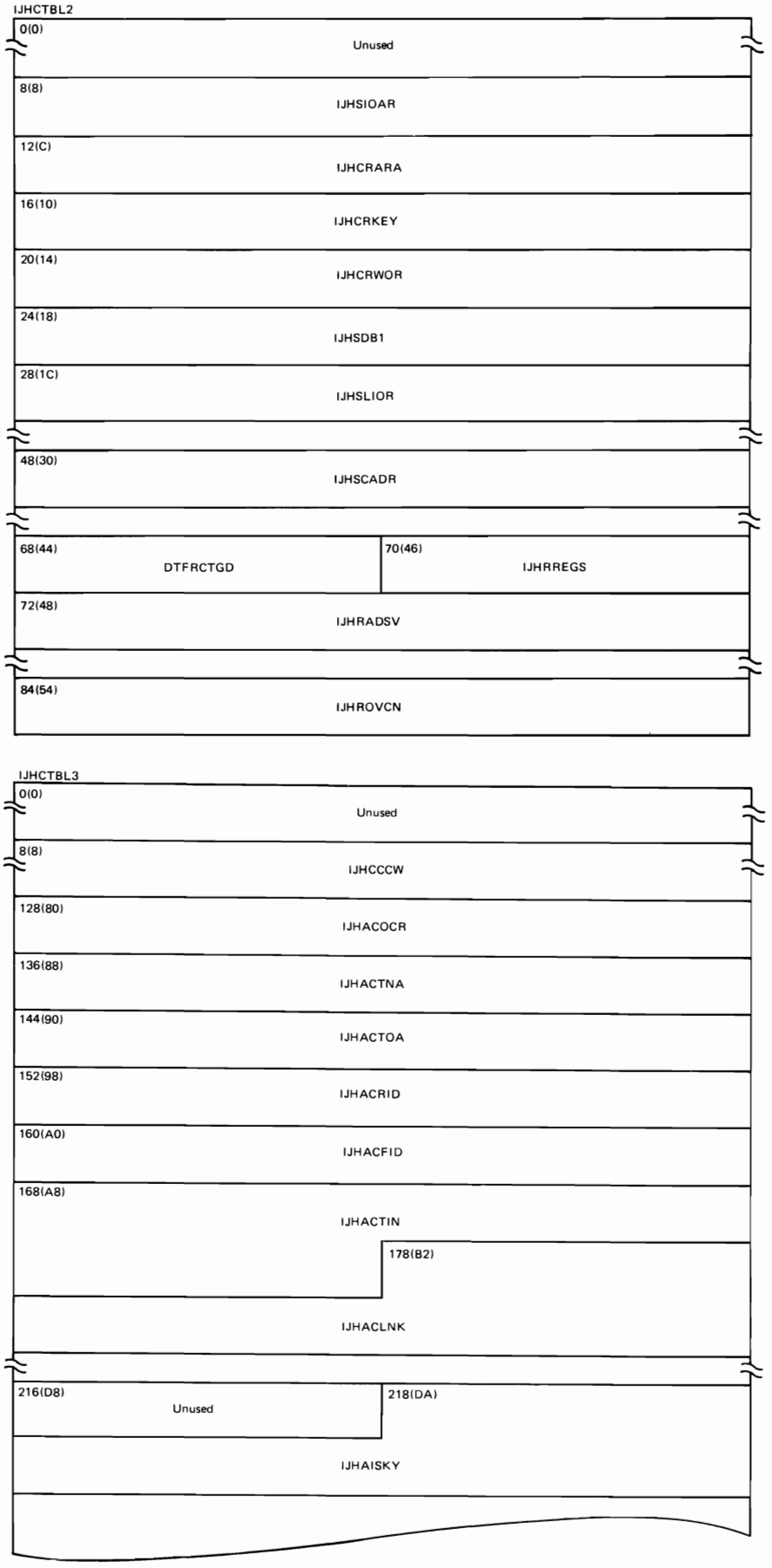


Figure 51 (Part 2 of 2). DTFIS ADD-RETRVE-ADDRTR Table

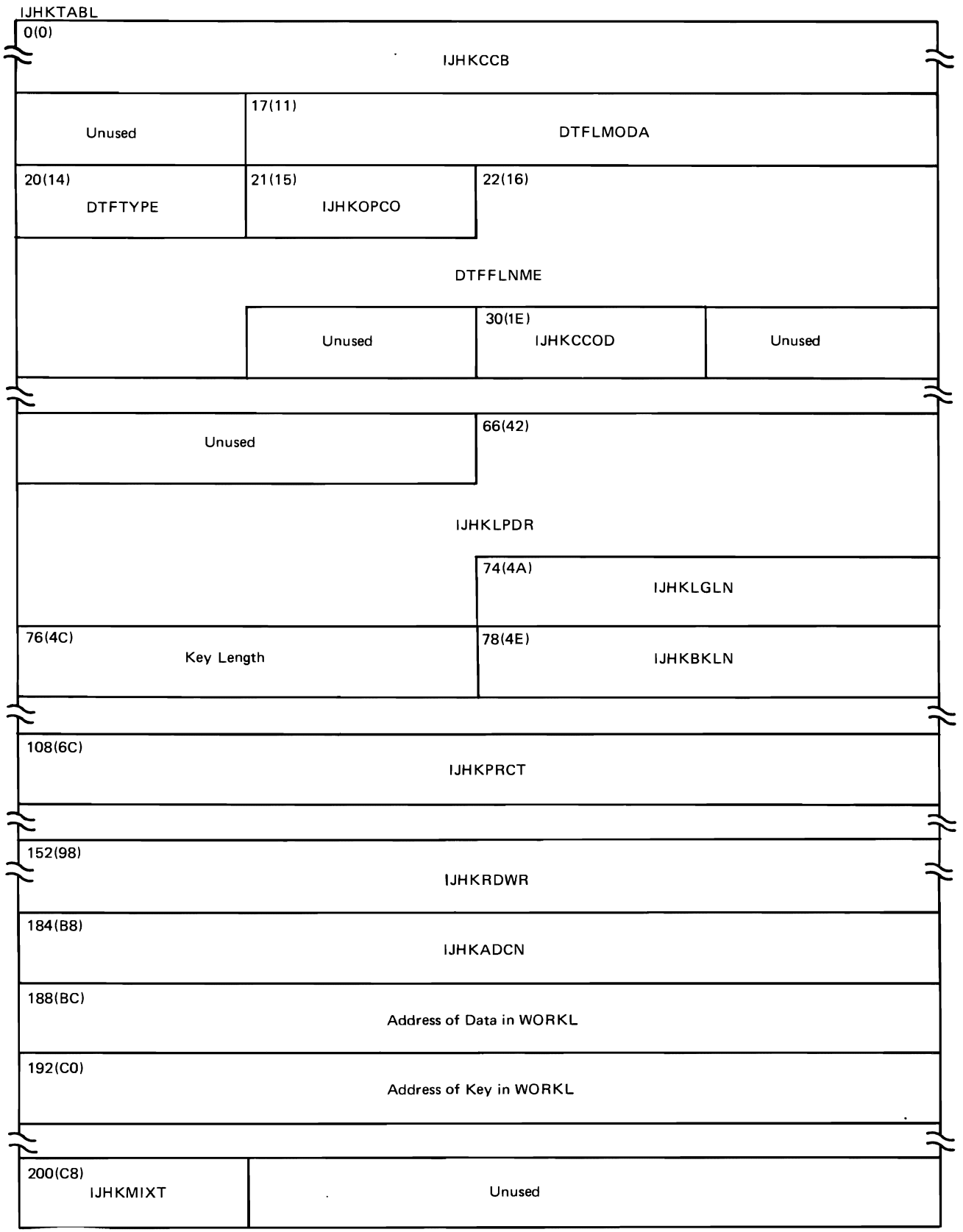


Figure 52. DTFIS Load Table

DTFIS ADD-RETRVE-ADDRTR Table Description (Part 1)

Offset	Bytes	DSECT Label *	Hex. Dig.	Field Description, Contents, Meaning
*Note:	Within IIVIS, IJH=DTF.			
0(0)	16	IJHCCCB		Command control block (CCB).
17(11)	3	DTFLMODA		Logic module address.
20(14)	1	DTFTYPE		File type for open/close as follows:
			25	Add
			26	RETRVE
			27	ADDRTR
21(15)	1	IJHCOPT		Option byte.
		..1.		Cylinder overflow.
		.... 1...		Blocked records.
22(16)	7	DTFFLNME		DTF file name.
30(1E)	1	IJHCSTBY		Status byte.
		1...		Uncorrectable DASD Error (except WLR).
		.1..		WLR error.
		..1.		EOF (sequential).
		...1		No record found.
		.... 1...		Illegal identification specified.
		.... .1..		Duplicate record sensed.
		.... ..1.		Overflow area full.
		1		Record retrieved from overflow area.
43(2B)	1	IJHCRTR		Retrieval byte.
		1...		WORKR area specified.
		.1..		WORKS area specified.
60(42)	8	IJHCCLPA		Last prime data record address (MBBCCHHR).
74(4A)	2	IJHCRESZ		Logical record length (RECSIZE).
76(4C)	2	IJHCKYSZ		Key length (KEYLEN).
78(4E)	2	IJHCBLSZ		Block size (logical record length times number of records in the block).
94(5E)	6	IJHCKYLC		Key location (KEYLOC) for blocked records.
100(64)	2	IJHCATB2		Displacement of part 2 of the DTFIS table from start of part 1.
102(66)	2	IJHCATB3		Displacement of part 3 of the DTFIS table from start of part 1.
104(68)	10	IJHCSADR		Seek/search address area.
124(7C)	4	IJHACPRC		Prime data record count.

Offset	Bytes	DSECT Label	Hex. Dig.	Field Description, Contents, Meaning
152(98)	2	IJHACOTC		Number of independent overflow tracks.
154(9A)	2	IJHACOFC		Number of full cylinder overflow areas.
156(9C)	2	IJHACORC		Overflow record count.
200(C8)	4	IJHAARAD		A(&IOAREAL) - address of IOAREAL, the I/O area used for adding records to a file.
204(CC)	4	IJHACUSE		A(&WORKL) - address of WORKL, work area containing user data records to be added to the file.
208(D0)	4	IJHADKEY		A(&filename.K) - address of the add key area.

DTFIS ADD-RETRVE-ADDRTR Table Description (Part 2)

Offset	Bytes	DSECT Label	Hex. Dig.	Field Description, Contents, Meaning
8(8)	4	IJHSIOAR		Address of IOAREAS, I/O area used for sequential retrieval.
12(C)	4	IJHCRARA		Address of IOAREAR, I/O area used for random retrieval or address of IOAREA2 (if specified) for sequential retrieval.
16(10)	4	IJHCRKEY		Address of KEYARG, field containing user-supplied key used for random read/write operations and sequential retrieval initiated by key.
20(14)	4	IJHCRWOR		Address of WORKR, work area used for random retrieval.
24(18)	4	IJHSDB1		Current sequential I/O area address.
28(1C)	4	IJHSLIOR		Load I/O register for sequential, or 4-byte NOP instruction for random.
48(30)	8	IJHSCADR		Current sequential DASD address (MBBCCHHR).
68(44)	2	DTFRCTGD		Number of records tagged for deletion.
70(46)	2	IJHRREGS		IOREG for random (or 2-byte NOP for sequential).
72(48)	4	IJHRADSV		Record pointer within I/O area for write (for random retrieval).
84(54)	4	IJHROVCN		An overflow record that is other than the first record in a chain of such records.

DTFIS ADD-RETRVE-ADDRTR Table Description (Part 3)

Offset	Bytes	DSECT Label	Hex. Dig.	Field Description, Contents, Meaning
8(8)	60	IJHCCCW		I/O trap code build area.
128(80)	8	IJHACOCR		Cylinder overflow control record (COCR).
136(88)	8	IJHACTNA		Current track index normal entry count field.
144(90)	8	IJHACTOA		Current track index overflow entry count field.
152(98)	8	IJHACRID		Current prime data record count field.
160(A0)	8	IJHACFID		Current overflow record count field.
168(A8)	10	IJHACTIN		Track index normal entry data field.
178(B2)	10	IJHACLNK		Current overflow record sequence-link field.
218(DA)	6	IJHAISKY		MVC 0(&KEYLEN,13),0(12) - unblocked MVC 0(&KEYLEN,13),&KEYLOC-1(12) - blocked utility MVC for key.
236(EC)	--			Key area for add only. Number of bytes depends on key length, KEYLEN.

DTFIS Load Table Description

Offset	Bytes	DSECT Label	Hex. Dig.	Field Description, Contents, Meaning
0(0)	16	IJHKCCB		Command control block (CCB).
17(11)	3	DTFLMODA		Logic module address.
20(14)	1	DTFTYPE	24	File type for OPEN/CLOSE: LOAD.
21(15)	1	IJHKOPCO		Option byte.
		..1.		Cylinder overflow option.
		.... 1...		Blocked records.
22(16)	7	DTFFLNME		File name.
30(1E)	1	IJHKCCOD		Status byte.
		1...		Uncorrectable DASD error (except WLR error).
		.1..		WLR error.
		..1.		Prime data area full.
		...1		Cylinder index area not large enough to reference prime data area. Set on only if error detected at SETFL time.
		.... 1...		Master index not large enough to reference prime data area. Set on only if error detected at SETFL time.
		.... .1..		Duplicate record.
		.... ..1.		Sequence error.
		1		No EOF record written in prime data area.
66(42)	8	IJHKLPDR		Address of last prime data record (MBBCCHHR).
74(4A)	2	IJHKLGLN		Logical record length.
76(4C)	2			Key length.
78(4E)	2	IJHKBKLN		Block length (logical record length times number of records in the block).
108(6C)	4	IJHKPRCT		Prime data record counter (logical records).
152(98)	32	IJHKRDWR		I/O trap code build area.
184(B8)	4	IJHKADCN		Address of IOAREAL.
188(BC)	4			Address of data in WORKL. (FIXBLK = address of WORKL; FIXUNB = address of WORKL + key length.)
192(C0)	4			Address of key in WORKL. (FIXBLK = address of WORKL + KEYLOC - 1; FIXUNB = address of WORKL.)
200(C8)	1	IJHKMIXT		Master index indicator. X'01' indicates a master index is used; X'00' indicates no master index is used.

Event Control Block

Initialized by: IIVINT

Modified by: IIVPCE, IIVSTG, IIVRTE, IIVCHK

Pointer in: IOB

The event control block (ECB, Figure 52) is used for communication between various components of the control program, as well as between processing programs and the control program. It is located in COMTAB and is used as specified in EXCP for testing of normal or abnormal completion of I/O requests.

Figure 53. Event Control Block

Event Control Block Description

Offset	Bytes and Alignment	Field Name	Hex Dig.	Field Description, Contents, Meaning
0 (0)	1	ECB		Flags and completion code.
		1...		Waiting for an event to occur. The WAIT macro instruction has been issued but the channel program has not been completed.
		.1..		The event has occurred. The channel program has been completed; a zero bit indicates it has not been completed.
		.xxx xxxx		COMPLETION CODE:
		0111 1111	7F	The channel program has terminated without error.
		0100 0001	41	The channel program has terminated with a permanent error.
		0100 0010	42	The channel program has terminated because a direct-access extent address has been violated.
		0100 0100	44	The channel program has been intercepted because of a permanent error associated with a device end for the previous request. The intercepted request may be reissued.
		0100 1000	48	A request element for the channel program has been made available after having been purged.
		0100 1111	4F	A direct-access device error recovery routine was unable to read the home address record or record 0.

ECB Pointer Table

Initialized by: IIVINT

Interrogated by: OS WAIT macro, IIVRTE

Pointer in: EMUCONS

The ECB pointer table (Figure 53) is a list of 4-byte addresses to the ECBs for devices being used for DOS emulation. The list contains an ECB address for each SYSE card plus an address for SYSLOG and three special ECBs. The special ECBs are for the Emulator prompt, WTOR, and timer. This table is initiated by the initialization routine.

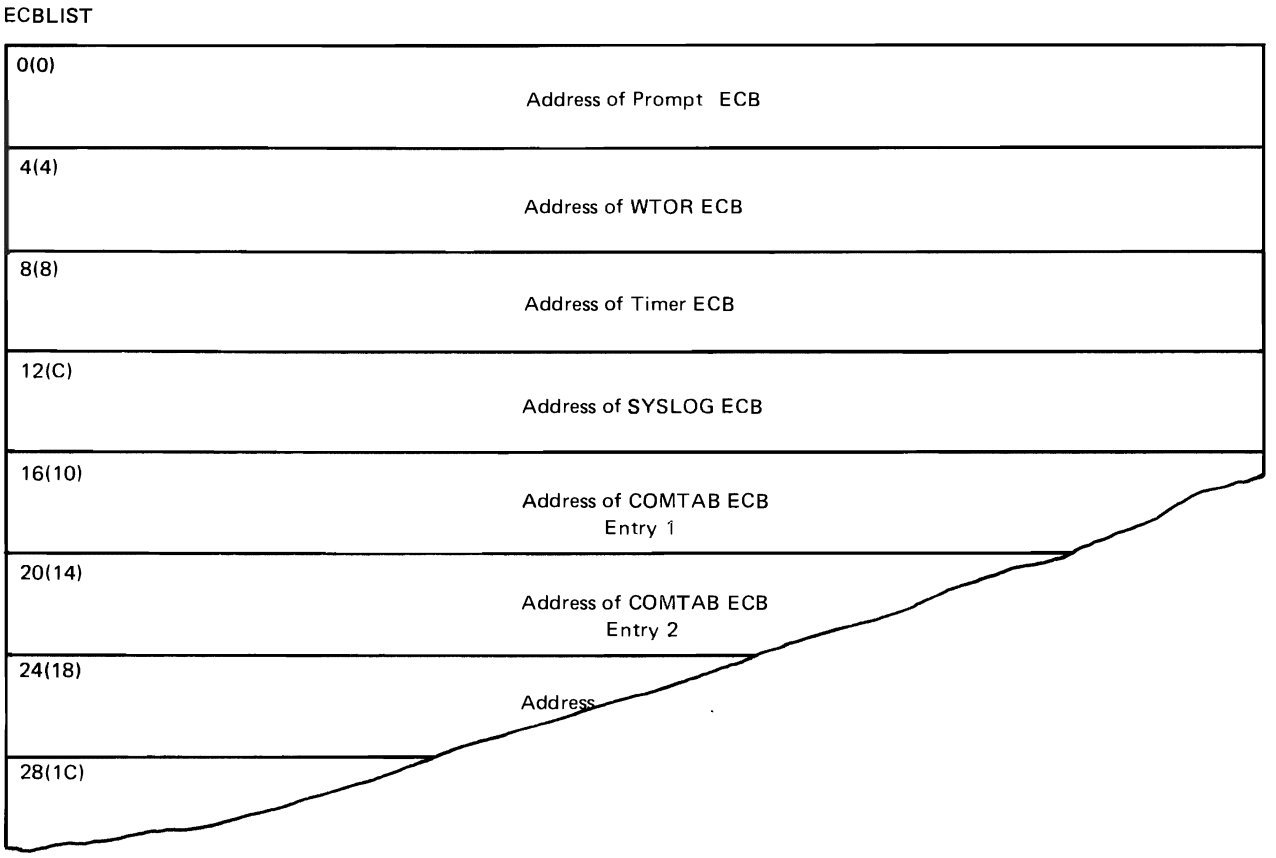


Figure 54. ECB Pointer Table

File ID Block

Initialized by: IIVIS, IIVDVS

Modified by: IIVIS, IIVDVS

Interrogated by: IIVIS, IIVDVS, IIVGR2, IIVSNP

Pointer in: EMUCONS

A file ID block (FIDBLK, Figure 54) is created for each opened file to map a DOS OPEN DTF to an OS OPEN DCB. The FID table is pointed to by the DSFIDBLK or ISFIDBLK fields in EMUCONS. Each FIDBLK entry is formatted as follows:

FIDBLK

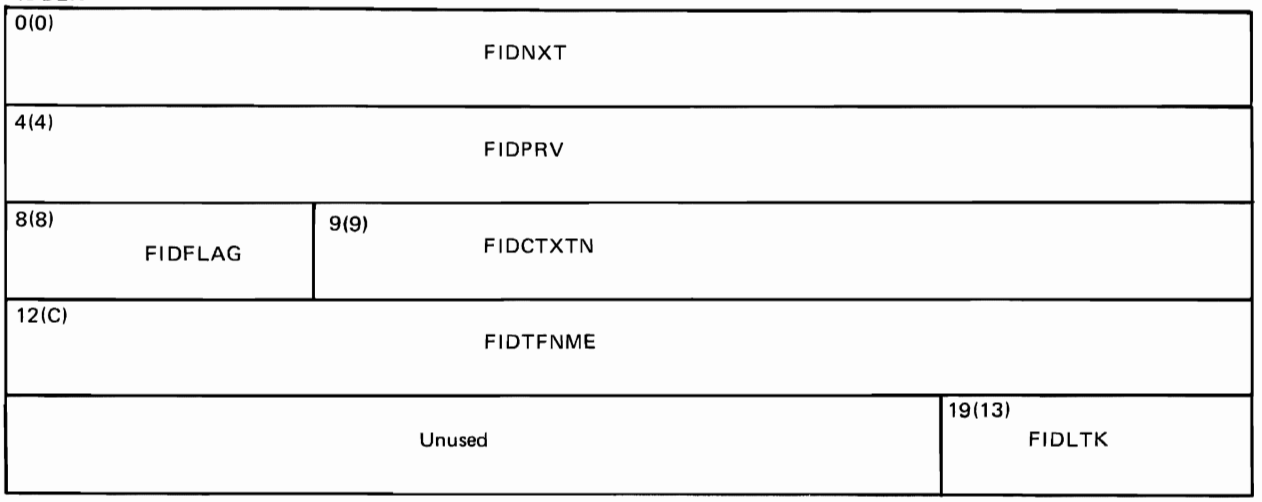


Figure 55. File ID Block

File ID Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	4	FIDNXT	Address of next FID block.
4(4)	4	FIDPRV	Address of previous FID block.
8(8)	1	FIDFLAG	X'80' indicates a system file. X'40' indicates a SYSREC file.
9(9)	3	FIDCTXTN	Pointer to COMTAB extension.
12(C)	7	FIDTFNME	Name of the DTF.
19(13)	1	FIDLTK	DOS partition of DTF.

IIVCON

Initialized by: IIVINT

Modified by: All Emulator modules

Pointer in: Emulator register 11

IIVCON (Figure 56) is a CSECT assembled with IIVENT containing data constants common to most Emulator modules. Figure 57 is a listing of the contents of IIVCON.

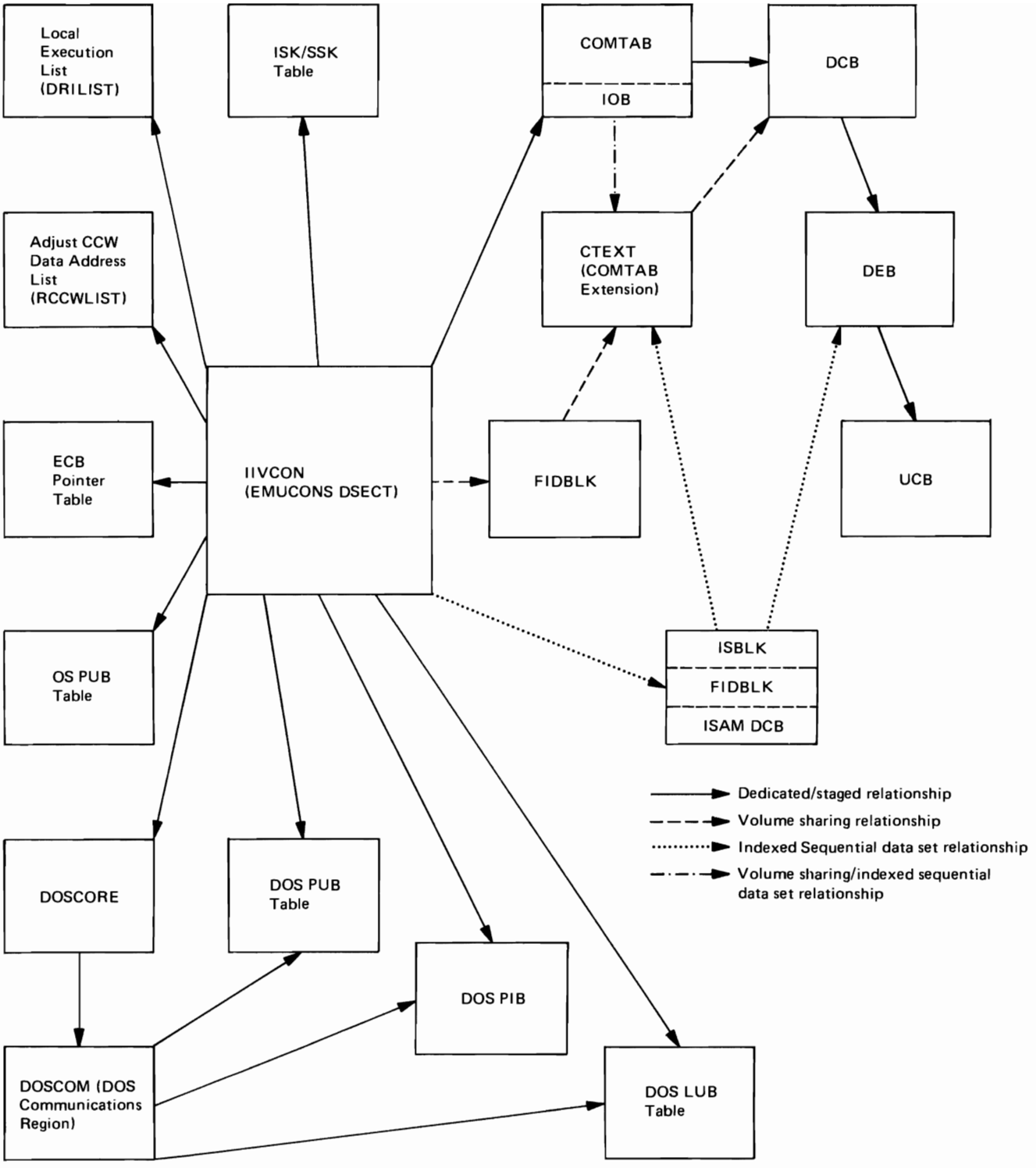


Figure 56. Relationship of IIVCON to Other Major Emulator Data Areas

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                        3+*******************************************************************
                                        4+*                                                                 *
                                        5+*         EMULATOR DATA CONSTANTS CSECT                           *
                                        6+*                                                                 *
                                        7+*******************************************************************

000000                                  9+IIVCON   CSECT
000000                                 10+HYPLIST  DS    0CL152
000000 0000000000000000                11+         DC    16F'0' .                 SPACE REQ TO ALIGN TO 64 BYTE
000040 0000000000000000                12+         DC    10F'0' .                 SPACE REQ FOR DRI LIST
000068 0000000000000000                13+         DC    6F'0' .                  FORCE ALIGNMENT FOR RCCW LIST
000080 0000000000000000                14+         DC    6F'0' .                  SPACE REQ FOR RCCW LIST

000098 00000000                        16+ACOMTAB  DC    A(0) .                   ADDR  OF COMMUNICATIONS TABLE
00009C 00000000                        17+ADOSPUB  DC    A(0) .                   ADDR OF DOS PUB TBL
0000A0 FFFFFFFFFFFFFFFF                18+CHNINDX  DC    8X'FF' .                 CHANNEL INDEX TABLE
0000A8 00000000                        19+COMTABEN DC    A(0) .                   COMM TABLE
0000AC 00000000                        20+STGINTR  DC    A(0) .                   STAGED INTERP. ROUTINE
0000B0 00000000                        21+ECBPTR1  DC    A(0) .                   ECB LIST POINTER ONE
0000B4 00000000                        22+ECBPTR2  DC    A(0) .                   ECB LIST POINTER TWO
0000B8 00000000                        23+POSTECB  DC    A(0) .                   BEGINNING ADDR OF POST ECB LIST
0000BC 00000000                        24+ENDPTECB DC    A(0) .                   POST ECB LIST DELIMITING ADDR
0000C0 00000000                        25+ACOMTBEX DC    A(0) .                   ADDRESS OF COMTAB EXTENSION 2-0
0000C4 00000000                        26+ACOMTABX DC    A(0) .                   PARM FOR AUTO ADD ROUTINE    2-0
0000C8 00000000                        27+ACTEXEND DC    A(0) .                   ENDING ADDR OF COMTAB EXT    2-0
0000CC 0000                            28+RESCUU   DC    H'0' .                   DOS SYSRES CUU
0000CE 0000                            29+IPLCUU   DC    H'0' .                   DOS IPL DVC CUU
0000D0 0000                            30+LOGCUU   DC    H'0' .                   DOS LOG DVC CUU
0000D2 00                              31+RESOFST  DC    X'0' .                   COMTAB ENT NO OF DOS SYSRES
0000D3 00                              32+IPLOFST  DC    X'0' .                   COMTAB ENT NO OF DOS IPL DVC
0000D4 00                              33+LOGOFST  DC    X'0' .                   COMTAB ENT NO OF DOS LOG DVC
0000D5 000000
0000D8 00000000                        34+ASTCBADR DC    A(0) .                   ASYNC TCB ADDRESSS
0000DC 00000000                        35+TIOTADR  DC    F'0' .                   ADDRESS OF TIOT
0000E0 00000000                        36+DOSSIZE  DC    F'0' .                   DOS SYSTEM SIZE
0000E4 0000                            37+NBR2K    DC    H'0' .                   NUMBER OF 2K INCREMENTS IN
0000E6 0000
0000E8 00000000                        38+ECBLIST  DC    A(0) .                   ADDR OF ECB LIST
                                       39+*
0000F0                                 40+TRNSFLD  DS    0D .                     TRANSLATE WORK AREA
0000F0                                 41+TRNSFLD5 DS    0CL5 .                   LNTH 5
0000F0                                 42+TRNSFLD4 DS    0CL4 .                   LNTH 4
0000F0                                 43+TRNSFLD3 DS    0CL3 .                   LNTH 3
0000F0 00                              44+TRNSFLD1 DC    X'00' .                  FIRST BYTE
0000F1 00000000000000                  45+TRNSFLDA DC    XL7'00' .                LAST 7 BYTES
                                       46+*
0000F8 00000000                        47+ISSKTAB  DC    A(0) .                   ISK/SSK TABLE ADDRESS
0000FC 0000                            48+NBRENT   DC    H'0' .                   NUMBER OF DEVICES ALLOCATED
0000FE 0000                            49+CTEXTCNT DC    H'0' .                   NUMBER OF COMTAB EXT ENTRIES2-0
000100 00                              50+OPTFLGR2 DC    XL1'0' .                 OPTIONS FLAG                 2-0
                                       51+*        EQU   X'80'                    ISAM SUPPORT REQUESTED       2-0
                                       52+*        EQU   X'40'                    VOLUME SHARING REQUESTED     2-0
                                       53+*        EQU   X'20'                    AUTO IPL REQUESTED           2-0
                                       54+*        EQU   X'10'                    ABBREVIATED PROMT REQUESTED  2-0
                                       55+*        EQU   X'08'                    SYSSNAP DD CARD PRESENT      2-0
                                       56+*        EQU   X'04'                    STAGED SUPPORT REQUESTED     2-0
                                       57+*        EQU   X'02'                    BYPASS ABEND FLAG            2-0
                                       58+*        EQU   X'01'                    SYSDEBUG DD CARD PRESENT     2-0
000101 00                              59+EOJSW    DC    XL1'0' .                 END OF EMULATION SWITHCH
000104                                 60+PARMLST  DS    0F .                     PARMS - IIVGP2/IIVDVS/IIVIS  2-0
000104                                 61+PARMDTFA DS    0A .                     DTF POINTER                  2-0
000104 00                              62+PARMCODE DC    XL1'0' .                 CODE                         2-0
00001C                                 63+PARMOPEN EQU   28 OPEN CODE
000020                                 64+PARMCLSE EQU   32 CLOSE CODE
000024                                 65+PARMEOX  EQU   36 END OF EXTENT CODE
000105                                 66+         DS    AL3(0) .                 DTF ADDRESS                  2-0
000108.                                67+PARMFDCX DS    0A .                     FIDBLK OR CTEXT POINTER      2-0
000108                                 68+PARMLTK  DS    AL1 .                    LOGICAL TRANSIENT KEY        2-0
000109                                 69+         DS    AL3 .                    FIDBLK OR CTEXT ADDRESS      2-0
000104                                 70+         ORG   PARMLST .                                             2-0
000104                                 71+ISAMLIST DS    0F .                     PARMS IIVISCX                2-0
000104 00                              72+ISAMCODE DC    AL1(0) .                 CODE                         2-0
000105 000000                          73+ISAMDTFA DC    AL3(0) .                 DTF ADDRESS                  2-0
000108 00000000                        74+ISAMFDCX DC    AL4(0) .                 FIDBLK OR CTEXT ADDRESS      2-0
00010C                                 75+         ORG
00010C 00000000                        76+DSFIDBLK DC    A(0) .                   FIRST FIDBLK ON D/S LIST     2-0
000110 00000000                        77+ISFIDBLK DC    A(0) .                   FIRST FIDBLK ON SSAM LIST    2-0
000114 D7D9D7C7D4404040                78+NXTBTR   DC    CL8'PRPGM' NEXT BTR TO BE EXECUTED     2-0
00011C D7D9D7C7D4404040                79+OLDBTR   DC    CL8'PRPGM' NAME OF BTR ISSUING SVC2    2-0
000124 00000000                        80+TIMERECB DC    F'0' .                   DUMMY TIMER ECB FOR LPSW

                                       82+***  SAVE AREA FOR DOS REGISTERS  ***

000128                                 84+         DS    0F .                     FOR ALIGNMENT
000128                                 85+REGSAV   DS    0CL64
000128                                 86+REG0     DS    F .                      DOS REG 0
00012C                                 87+REG1     DS    F .                      DOS REG 1
000130                                 88+REG2     DS    F .                      DOS REG 2
000134                                 89+REG3     DS    F .                      DOS REG 3
000138                                 90+REG4     DS    F .                      DOS REG 4
00013C                                 91+REG5     DS    F .                      DOS REG 5
000140                                 92+REG6     DS    F .                      DOS REG 6
000144                                 93+REG7     DS    F .                      DOS REG 7
000148                                 94+REG8     DS    F .                      DOS REG 8
00014C                                 95+REG9     DS    F .                      DOS REG 9
000150                                 96+REGA     DS    F .                      DOS REG 10
000154                                 97+REGB     DS    F .                      DOS REG 11
000158                                 98+REGC     DS    F .                      DOS REG 12
00015C                                 99+REGD     DS    F .                      DOS REG 13
000160                                100+REGE     DS    F .                      DOS REG 14
000164                                101+REGF     DS    F .                      DOS REG 15

000168 00000000                       103+BREGSAVE DC    F'0' .                   REGISTER SAVE
00016C 0000000000000000               104+EMUSAVE  DC    18F'0' .                 EMULATOR REGISTER SAVE AREA
0001B4 00012C00                       105+SECOND   DC    F'76800' .               ONE SECOND TIME INTERVAL
```

Figure 57 (Part 1 of 4). Emulator Common Data Area

```
DOS EMULATOR - COMMON DATA AREA

 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

0001B8 CC009600                   106+SECONDA  DC    F'38400' .              ONE SECOND FOR STIMER
0001BC 0000                       107+NOIOPEND DC    H'0' .                  NO OF I/O PENDING
0001BE 0001                       108+H1       DC    H'1' .                  HALF WORD CONSTANT OF ONE
0001C0 0000                       109+NOSIO    DC    H'0' .                  NO. SIO FUNCTIONS ACTIVE
0001C2 00                         110+PARMFLG  DC    XL1'00' .               INITIALIZATION REMINDER FLAG
0001C3 00
0001C4 00C0000000000000          111+REGSAVE  DC    2F'0' .                 REGISTER SAVE
0001CC 00000000                   112+ADDRWRK  DC    A(0) .                  ADDRESS WORK AREA
0001D0 00000000                   113+AOSPUB   DC    A(0) .                  ADDR OF OS PUB TABLE
0001D4 00004E20                   114+CEAIDS   DC    A(20000) .              ADDR OF DOS CE AIDS AREA

0001D8 1F                         116+IPLSW    DC    B'00011111' .           INITIALIZATION (IPL) SWITCH
                                  117+*                                      BIT 0 - UNUSED
                                  118+*                                      BIT 1 - UNUSED
                                  119+*                                      BIT 2 - SYSABEND DD PRESENT 2-1
                                  120+*                                      BIT 3 - I/O FOR IPL READS
                                  121+*                                      BIT 4 - I/O TO READ SUPERVISOR
                                  122+*                                      BIT 5 - 1ST LOAD PSW
                                  123+*                                      BIT 6 - 1ST PC FOR CORE CLEARIN
                                  124+*                                      BIT 7 - 1ST SVC 14 (EOJ)

0001D9 01                         126+PENDSW   DC    X'01' .                 INTERRUPT PENDING SWITCH
                                  127+*                                      BIT 0 - LOG BUSY              2.1
                                  128+*                                      BIT 1 - UNUSED
                                  129+*                                      BIT 2 - MODEL 135 CPU         2.1
                                  130+*                                      BIT 3 - WTO INTERRUPT
                                  131+*                                      BIT 4 - TIME INTERRUPT
                                  132+*                                      BIT 5 - PROMPT INTERRUPT
                                  133+*                                      BIT 6 - WTOR INTERRUPT
                                  134+*                                      BIT 7 - DRI SWITCH

0001DA                            136+PRPMSG99 DS    0CL21 .                 EMULATOR
0001DA 5CC5D4E440                 137+         DC    C'*EMU ' .              OPERATOR
0001DF D1D6C2D5C1D4C540           138+PRPMSGJN DC    C'JOBNAME ' .           PROMPT
0001E7 4040D9D6D4D7E35C           139+         DC    C' PROMPT*' .           MSG
0001EF 404040404040               140+PRPVSER  DC    CL6' ' .                VOL SER WORK AREA
0001F5 01                         141+PRPSW    DC    X'01' .                 INITIAL PROMPT SW
0001F6 00                         142+TIMEIND  DC    X'00' .                 TIME INDICATOR SWITCH
0001F7 F7F0F0                     143+BADCUU   DC    C'700' .                INVALID CHANNEL ADDR
0001FA FF00                       144+EXTMSK   DC    X'FF00' .               EXTERNAL REQUEST MASK

                                  146+*        TRANSLATE TABLE TO CONVERT TO HEX                          *
00013B                            147+HEXTAB   EQU   *-C'A'
0001FC FAFBFCFDFEFF               148+         DC    X'FAFBFCFDFEFF' .       A - F
000202 00000000000000000          149+         DC    XL41'0' .               SPACE BETWEEN
00022B F0F1F2F3F4F5F6F7           150+         DC    C'0123456789ABCDEF' .   0 - 9

                                  152+*        EMULATOR SCAN TABLE                                        *
00023C                            154+         DS    0F .                    ALIGNMENT
00023C                            155+SCANTBL  DS    0CL16
00023C C00C0010                   156+SCANEND  DC    A(0) .                  END ADDRESS OF SCAN AREA
000240 000000                     157+SCANCHR  DC    XL3'0' .                SCAN CHAR
000243 00                         158+SCANSTP  DC    X'0' .                  SCAN STOP CHARACTER
000244 00000000                   159+SCANBEG  DC    A(0) .                  NEW SCAN START ADDRESS
000248 00000000                   160+SCANLNG  DC    F'0' .                  REMAINING SCAN AREA LENGTH

                                  162+*        TRANSLATE TABLE TO STOP ON BLANK OR COMMA                  *
00024C 00000000000000000          164+STPTBL   DC    XL64'0' .               DO NOT STOP
00028C 40                         165+         DC    C' ' .                  STOP
00028D 00000000000000000          166+         DC    XL42'0' .               DO NOT STOP
0002B7 6B                         167+         DC    C',' .                  STOP
0002B8 00000000000000000          168+         DC    XL148'0' .              DO NOT STOP


                                  171+*                    DATA CONTROL BLOCK
                                  172+*
000348                            173+         ORG   *-4 TO ELIMINATE UNUSED SPACE
000348                            174+DADCB    DS    0F'0' ORIGIN ON WORD BOUNDARY
00034C                            175+         ORG   *+4 TO ORIGIN GENERATION

                                  177+*                    DIRECT ACCESS DEVICE INTERFACE

00034C 00000000000C0000           179+         DC    BL12'0' EDAD,DVTBL
000358 00000000                   180+         DC    A(0) IKEYLE,DEVT,TRBAL

                                  182+*                    COMMON ACCESS METHOD INTERFACE

00035C 00                         184+         DC    AL1(0) BUFNO
00035D 000001                     185+         DC    AL3(1) BUFCB
000360 0000                       186+         DC    AL2(0) BUFL
000362 2000                       187+         DC    BL2'0010000000000000' DSORG
000364 00C00001                   188+         DC    A(1) IOBAD

                                  190+*                    FOUNDATION EXTENSION

000368 00                         192+         DC    BL1'00000000' BFTEK,BFLN,HIARCHY
000369 000001                     193+         DC    AL3(1) EODAD
00036C 00                         194+         DC    BL1'00000000' RECFM
00036D 000390                     195+         DC    AL3(XLIST) EXLST

                                  197+*                    FOUNDATION BLOCK

000370 E2E8E2C5D4404040           199+         DC    CL8'SYSEM' DDNAME
000378 02                         200+         DC    BL1'00000010' OFLGS
000379 00                         201+         DC    BL1'00000000' IFLG
00037A F004                       202+         DC    BL2'1111000000000100' MACR

                                  204+*                    EXCP APPENDAGE LIST
```

Figure 57 (Part 2 of 4). Emulator Common Data Area

```
DOS EMULATOR - COMMON DATA AREA

 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

000370 00                         206+          DC    BL1'00000000' BER1
000371 000000                     207+          DC    XL3'000000' RER1
000380 00000000                   208+          DC    XL4'00000000' UMSK
000384 E8C1                       209+          DC    CL2'YA' EOEA
000386 F040                       210+          DC    CL2'0' PCIA
000388 E8C1                       211+          DC    CL2'YA' SIOA
00038A F040                       212+          DC    CL2'0' CENDA
00038C F040                       213+          DC    CL2'0' XENDA
00038E F040                       214+          DC    CL2'0' AERR
3C0048                            215+DADCBLEN  EQU   *-DADCB .             LENGTH OF DASD DCB

000390 87                         217+XLIST     DC    X'87' .               DCB EXIT LIST FOR PDJFCB MACRO
000391 000394                     218+          DC    AL3(JFCBAREA) .       USAGE

000394                            220+JFCBAREA  DS    0F .                  JFCB WORK AREA
000394                            221+JFCBDSN   DS    0CL44 .               JFCB DATA SET NAME          2-0
000394                            222+          DS    CL8 .                 *                           2-0
00039C                            223+JFCBDS8   DS    CL2 .                 *                           2-0
00039E                            224+          DS    CL7 .                 *                           2-0
0003A5                            225+JFCBDS18  DS    CL1 .                 *
0003A6                            226+JFCBDS19  DS    CL1 .                 *                           2-0
0003A7                            227+          DS    CL3 .                 *                           2-0
0003AA                            228+JFCBDSJN  EQU   * .                   *                           2-0
0003AA                            229+          DS    CL22 .                *                           2-0
0003C0                            230+          DS    XL8 .                 NOT USED
0003C8                            231+JFCBIP    DS    XL1 .                 JFCB INTERFACE BYTE
0003C9                            232+          DS    XL3 .                 NOT USED                    2-0
0003CC                            233+JFCFCBID  DS    XL4 .                 FCB IDENTIFIER              2-0
0003D0                            234+          DS    CL6 .                 NOT USED                    2-0
0003D6                            235+JFCBIP    DS    XL1 .                 JFCB LABEL TYPE
0003D7                            236+          DS    XL9 .                 NOT USED                    2-0
0003E0                            237+JFCBMSK1  DS    XL1 .                 JFCB DATA MANAGMENT         2-0
                                  238+*                                     MASK BYTE 5                 2-0
0003E1                            239+          DS    XL10 .                NOT USED                    2-0
0003EB                            240+JFCBINDZ  DS    XL1 .                 JFCB FLAG                   2-0
0003EC                            241+JFCBUFNO  DS    XL1 .                 JFCB NUMBER OF BUFFERS      2-0
                                  242+*                                     REQUIRED FOR THIS DATA      2-0
                                  243+*                                     SET                         2-0
0003ED                            244+JFCBUFIN  DS    XL1 .                 JFCB BUFFER HIERAPCHY,      2-0
                                  245+*                                     BUFFERING TECHNIQUE,        2-0
                                  246+*                                     BUFFER ALIGNMENT            2-0
0003EE                            247+JFCBUFL   DS    XL2 .                 JFCB BUFFER LENGTH          2-0
0003F0                            248+          DS    XL1 .                 NOT USED                    2-0
0003F1                            249+JFCKEYLE  DS    XL1 .                 JFCB DIRECT ACCESS KEY      2-0
                                  250+*                                     LENGTH                      2-0
0003F2                            251+          DS    XL4 .                 NOT USED                    2-0
0003F6                            252+JFCBSORG  DS    XL1 .                 JFCB DATA SET ORGANIZATION  2-0
0003F7                            253+          DS    XL1 .                 NOT USED                    2-0
0003F8                            254+JFCRECFM  DS    XL1 .                 JFCB RECORD FORMAT          2-0
0003F9                            255+JFCOPTCD  DS    XL1 .                 JFCB OPTION CODES           2-0
0003FA                            256+JFCBBS    DS    XL2 .                 MAX BLOCK SIZE
0003FC                            257+JFCLRECL  DS    XL2 .                 JFCB LOGICAL RECORD         2-0
                                  258+*                                     LENGTH                      2-0
0003FE                            259+JFCNCP    DS    XL1 .                 JFCB NUMBER CHANNEL         2-0
                                  260+*                                     PROGRAMS                    2-0
0003FF                            261+JFCNTM    DS    XL1 .                 JFCB NUMBER OF TRACKS       2-0
                                  262+*                                     MASTER INDEX                2-0
000400                            263+JFCRKP    DS    XL2 .                 JFCB RELATIVE KEY           2-0
                                  264+*                                     POSITION                    2-0
000402                            265+JFCCYLOF  DS    XL1 .                 JFCB CYLENDER OVERFLOW      2-0
000403                            266+          DS    XL7 .                 NOT USED                    2-0
00040A                            267+JFCBVSER  DS    0CL6 .                JFCB VOLUME SERIAL NUMBER
00040A                            268+JFCBVSR1  DS    CL3 .                 *
00040D                            269+JFCBVSR3  DS    CL3 .                 *
000410                            270+          DS    CL52 .                NOT USED

000444 00000000                   272+WTOECB    DC    F'0' .                DUMMY ECB FOR SYSLOG
000448 00000000                   273+ABGSYSPD  DC    A(0) .                ADDR OF BG SYSRDR LUB
00044C 00000000                   274+ABGPIB    DC    A(0) .                ADDR OF BG PIB
000450 00000000                   275+AF2PIB    DC    A(0) .                ADDR OF F2 PIB
000454 00000000                   276+AF1PIB    DC    A(0) .                ADDR OF F1 PIB

000458                            278+AIIVGR2   DS    A .                   ADDR OF SVC 2 SUPRVSR ROUT  2-0
00045C                            279+AIIVDVS   DS    A .                   ADDR OF SHARED VOL ROUTINE  2-0
000460                            280+AIIVISM   DS    A .                   ADDR OF ISAM ROUTINE        2-0
000464 00000000                   281+ARASCONS  DC    A(0) .                PTR TO SERVICE AID CNTRL BLK2-0
000468 00000000                   282+IMGLBDCB  DC    A(0) .                PTR TO IMAGELIB DCB         2-0
00046C 00000000                   283+DBUGDCB   DC    A(0) .                SNAP DCB POINTER            2-0
000470 00000000                   284+STAEREGS  DC    A(0) .                PTR TO SAVE AREA            2-0
000474 00000000                   285+STAERTN   DC    A(0) .                RETRY POINT                 2-0
000478 00000000                   286+AIIVPOV   DC    A(0) .                PRINT OVERFLOW ROUTINE      2-0

000480                            288+FCDCCW    DS    0D .                  FCB DUMMY CCW (STS PRINTER) 2-0
000480 00                         289+FCBCCWOP  DC    AL1(0) .              *  OP CODE                  2-0
000481 000000                     290+FCBCCWDA  DC    AL3(0) .              *  PTR TO FCB IMAGE + 2     2-0
000484 00                         291+FCBCCWFG  DC    AL1(0) .              *  CCW FLAGS                2-0
000485 00                         292+FCBCCWX   DC    AL1(0) .              *                           2-0
000486 0000                       293+FCBCCWCT  DC    AL2(0) .              *  CCW DATA TRNSFER LENG    2-0

000488                            295+AIIVSDT   DS    A .                   ADDR OF STAGED DEVICE TABLE 2-0
00048C                            296+AIIVVIO   DS    A .                   ADDR OF IIVVIO              2-0
000490 00000000                   297+ADDSBTR   DC    A(0) .                ADDR OF DOS BTRANSIENT AREA 2-0
000494 00000000                   298+ADDSETND  DC    A(0) .                ADDR OF DOS BTRANSIENT END  2-0
000498 40404040404040404040       299+IPLSTMT   DC    CL32' ' .             LAST AUTO IPL STMT          2-0

0004B8                            301+PRPCUUWK  DS    CA .                  CUU TRANSLATE WORK AREA
0004B8 03                         302+          DC    AL1(3) .              LENGTH 3
0004B9 0004BC                     303+          DC    AL3(PRPCUUC) .        CHAR CUU WRK ADDR
0004BC 404040                     304+PRPCUUC   DC    CL3' ' .              CHANNEL AND UNIT
```

Figure 57 (Part 3 of 4). Emulator Common Data Area

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

0004BF 0C
0004C0 00000000                    306+AIIVMSG  DC    V(IIVMSG) .              ADDR OF MESSAGE WRITER      2-0
0004C4 00000000                    307+ABEBLK   DC    V(IIVCCWBB) .            ADDR OF DEADLOCK            2-0
0004C8 00000000                    308+CAWSCNAD DC    V(IIVAWV) .              ADDR OF IIVAWV
0004CC 00000000                    309+CCWSCNAD DC    V(IIVCWV) .              ADDR OF IIVCWV
0004D0 00000000                    310+LOGIO    DC    V(IIVLOG) .              ADDR OF IIVLOG
0004D4 00000000                    311+AEOJCHK  DC    V(IIVRTEJJ) .            EOJ CHECK ROUTINE ADDR
                                   312+*
0004D8                             313+IIVCONB1 EQU   *
                                   314+         ENTRY IIVCONB1
0004D8 00000000                    315+WTORECB  DC    F'0' .                   WTOR ECB
                                   316+*
0004DC                             317+IIVCONB2 EQU   *
                                   318+         ENTRY IIVCONB2
0004DC 00000000                    319+PRUMPECB DC    F'0' .                   PROMPT ECB
                                   320+*
0004E0 00000000                    321+AASCHCK  DC    V(IIVRTECK) .            ADDR OF IIVRTECK
0004E4 00000000                    322+ABADR    DC    V(IIVABN) .              ADDR OF ABEND ERROR RTN
0004E8 00000000                    323+ASTGIO   DC    A(0) .                   ADDR OF IIVSTG
0004EC 00000000                    324+ADPEN    DC    V(IIVOPN) .              ADDR OF OPEN ROUTINE

0004F0 00000000                    326+ART20    DC    V(IIVRTER2) .            ADDR OF IIVRTER2
0004F4 00000000                    327+ATIMER   DC    V(IIVRTETM) .            ADDR OF IIVRTETM
0004F8 00000000                    328+ASVC     DC    V(IIVSVC) .              ADDR OF SVC ROUTINE         2-0
0004FC 00000000                    329+APC      DC    V(IIVPCE) .              ADDR OF PC ROUTINE
000500 00000000                    330+AROUTEEP DC    V(IIVRTE) .              ADDR OF IIVTRE
000504 00000000                    331+AROUTE   DC    V(IIVRTER1) .            AADR OF IIVRTER1
000508 00000000                    332+AIIVOBE1 DC    A(0) .                   ADDR OF OBTAIN WORK AREA    2-0
00050C                             333+BASEREGS DS    0XL12 .                  COMMON BASE REGISTER VALUES
00050C 00000000                    334+ADRILST  DC    A(0) .                   DRILIST ADDRESS
000510 00000000                    335+RFACTOR  DC    A(0) .                   DOS CORE ADDRESS
000514 00000000                    336+AEMUCONS DC    A(IIVCON) .              EMUCONS ADDRESS
000518 00000000                    337+AIIVGET  DC    V(IIVGET) .              ADDRESS OF GETMAIN ROUTINE  2-0

00051C 00                          339+PRPRPLY2 DC    X'00' .                  ZERO CLEAR CHARACTER        2-0
00051D                             340+IIVCONB3 EQU   * .                      2-0
                                   341+         ENTRY IIVCONB3 .              2-0
00051D 0000000000000000            342+PRPRPLY  DC    XL119'0' .               REPLY AREA FOR PROMPT       2-0
000594 40C3                        343+PRPCONTF DC    C' C' .                  CONTINUATION CHARACTER      2-0

000598                             345+OPENLIST DS    0F
000598 80                          346+         DC    AL1(128) OPTION BYTE
000599 000348                      347+         DC    AL3(DADCB) DCB ADDRESS


00059C                             349+QLIST    DS    0F
00059C FF                          350+         DC    AL1(255) LAST ENTRY INDICATION
00059D 04                          351+         DC    AL1(RNMELEN) LENGTH
00059E 40                          352+         DC    BL1'01000000' SDSI
00059F 00                          353+         DC    AL1(0) RETURN CODE FIELD
0005A0 000005A8                    354+         DC    A(QNAME) QNAME ADDRESS
0005A4 000EC5B0                    355+         DC    A(RNAME) RNAME ADDRESS
0005A8 C9C9E5C5D4E44040            356+QNAME    DC    CL8'IIVEMU' .            Q RESOURCE NAME
0005B0                             357+RNAME    DS    0CL4 .                   R RESOURCE NAME
0005B0 404040                      358+RNMCUU   DC    CL3' ' .                 OS CUU
0005B3 40                          359+RNMBIN   DC    CL1' ' .                 OS BIN NUMBER
000004                             360+RNMELEN  EQU   L'RNAME .                LENGTH OF RNAME FIELD


0005B4 00000000                    362+AIIVCCW  DC    V(IIVCCW) IIVCCW ADDR                 2-1
0005B8 00000000                    363+AIIVADJ  DC    V(IIVADJ) IIVADJ ADDR                 2-1
0005BC 0000000000000000            364+CPUID    DC    XL8'0' CPU ID FIELD             2.1


0005C4                             366+CTLREGS  DS    0F .                     DOS CONTROL REGISTER FIELD  2-1
0005C4 00CC03E0                    367+CTLR0    DC    XL4'E0' .                TM4, INT AND EXT SIG MASKS  2-1
0005C8 00000000                    368+CTLR1    DC    F'0' .                   UNUSED                      2-1
0005CC FFFFFFFF                    369+CTLR2    DC    F'-1' .                  CHANNEL MASKS               2-1
0005D0 00000000                    370+CTLR3    DC    F'0' .                   UNUSED                      2-1
0005D4 00000000                    371+CTLR4    DC    F'0' .                   UNUSED                      2-1
0005D8 00000000                    372+CTLR5    DC    F'0' .                   UNUSED                      2-1
0005DC 00000000                    373+CTLR6    DC    F'0' .                   UNUSED                      2-1
0005E0 00000000                    374+CTLR7    DC    F'0' .                   UNUSED                      2-1
0005E4 00000000                    375+CTLR8    DC    F'0' .                   UNUSED                      2-1
0005E8 00000000                    376+CTLR9    DC    F'0' .                   UNUSED                      2-1
0005EC 00000000                    377+CTLR10   DC    F'0' .                   UNUSED                      2-1
0005F0 00000000                    378+CTLR11   DC    F'0' .                   UNUSED                      2-1
0005F4 00000000                    379+CTLR12   DC    F'0' .                   UNUSED                      2-1
0005F8 00000000                    380+CTLR13   DC    F'0' .                   UNUSED                      2-1
0005FC C40000C0                    381+CTLR14   DC    X'C4',XL3'0' .           MCH HANDLING                2-1
000600 00000200                    382+CTLR15   DC    F'512' .                 MCEL PTR (MCH)              2-1


000604 00                          384+S370SW   DC    X'0' .                   SWITCH FOR 370              2-1
                                   385+*        EQU   X'20' .                  SHARED DOSPES VOLUME FLG    2-1
000605 08                          386+DRELEASE DC    X'08' DOS RELEASE SUPPORT       2-1
000001                             387+REL25    EQU   X'01' 25                        2-1
000002                             388+REL25E   EQU   X'02' 25E                       2-1
000004                             389+REL26    EQU   X'04' 26                        2-1
000008                             390+REL27    EQU   X'08' 27                        2-1
000606 0000                        391+         DC    2X'00' SPARE FLAGS               2-1
000608 00000300                    392+DOSRFTB  DC    A(0) ADD OF DOS RF TABLE        2-1
00060C 0000                        393+EMULBLAD DC    H'0' .                   LABEL CYL ADDR -CCC-        2-1
00060E 0000
000610 0000000000000000            394+TDELTA   DC    D'0' TIME DIFF BTWN OS AND DOS  2-1
000618 0000000000000000            395+WKAREA   DC    D'0' WORK AREA                  2-1
000620 00000000                    396+STCKADDR DC    A(0) ADDRESS OF SVC 34 STCK INST 2-1
000624 BBBBBBBBBBBBBBBB            397+EMUPATCH DC    132DX'BB' ABOUT 5 PEPCEBT             2-1
```

Figure 57 (Part 4 of 4). Emulator Common Data Area

IIVRCN

Initialized by: IIVRAS

Modified/interrogated by: IVRAS, IIVSNP, IIVRCP, IIVSCI, IIVPCI, IIVACI

Pointer in: IIVCON

IIVRCN (Figure 58) is a CSECT assembled with module IIVRAS containing data constants common to the serivce aids modules. Figure 59 is a listing of the contents of IIVRCN.

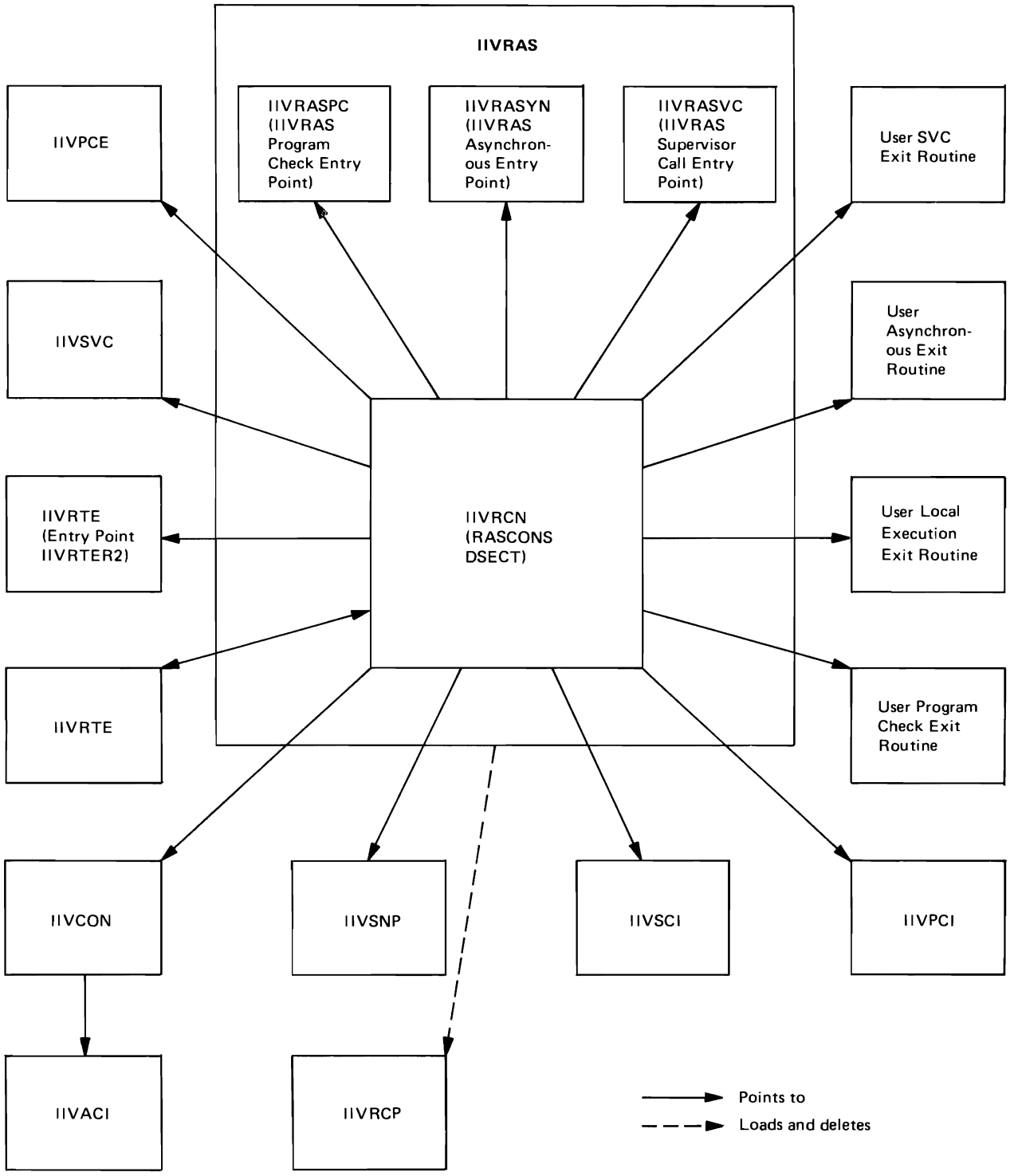


Figure 58. Relationship of IIVRCN to Other Service Aids Modules, Other Emulator Modules, IIVCON, and User Routines

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

CC0000                             3+IIVRCN   CSECT
CC0000                             4+RASCONS  EQU   IIVRCN
000000 08098C829C9D9E9F            5+RASOPCDS DC    XL8'080980829C9D9E9F' .  VALID OP CODES
CC0008                             6+RASOPLEN EQU   *-RASOPCDS .             LENGTH OF TABLE

000008 C00CCCCO                    8+RASTRCTB DC    A(0) .                   ADDRESS OF TRACE TABLE
CC000C 0000                        9+RASTRCMX DC    H'0' .                   NUMBER OF ACTIVE ENTRIES
CC000E C000                       10+RASTRCNX DC    H'0' .                   NEXT AVAILABLE ENTRY
000010 0000                       11+RASTRCNO DC    H'00' .                  TRACE TABLE ENTRIES
CC0012 0024                       12+TRCLEN   DC    H'36' .                  LENGTH OF EACH TRACE ENTRY

000014 C00C0CC00CC00000           14+RASTRCDS DC    XL8'0' .                 SPECIFIED TRACE OP CODES
                                  15+*        EQU   X'08CC00000C000000' .    SSK TRACE
                                  16+*        EQU   X'00090000CC000000' .    ISK TRACE
                                  17+*        EQU   X'0000800000000000' .    SSM TRACE
                                  18+*        EQU   X'CC000C82CCC00000' .    LPSW TRACE
                                  19+*        EQU   X'000000009C000000' .    SIO TRACE
                                  20+*        EQU   X'C00000000090000' .     TIO TRACE
                                  21+*        EQU   X'CCC0000000009E00' .    HIO TRACE
                                  22+*        EQU   X'000000000000009F' .    TCH TRACE
000008                            23+TRCDLEN  EQU   *-RASTRCDS .             LENGTH OF TABLE

CC001C 0000000000000000           25+RASTRCUU DC    5F'0' .                  TRACE CUU TABLE
CC000A                            26+RASTRLEN EQU   (*-RASTRCUU)/2 .         SIZE OF TABLE

000030 00                         28+RASTRSW1 DC    X'00' .                  TRACE SWITCH 1
                                  29+*        EQU   X'01' .                  TRACE 'IO ' PARM SPECIFIED
000031 00                         30+RASTRSW2 DC    X'00' .                  TRACE SWITCH 2
CC0032 00                         31+RASTRSW3 DC    X'00' .                  TRACE SWITCH 3
                                  32+*        EQU   X'10' .                  TRACE 'SVC' PARM SPECIFIED
                                  33+*        EQU   X'01' .                  TRACE 'EXT' PARM SPECIFIED
                                  34+*        EQU   X'02' .                  TRACE 'INT' PARM SPECIFIED
                                  35+*        EQU   X'04' .                  TRACE 'ATTN PARM SPECIFIED
                                  36+*        EQU   X'08' .                  TRACE 'TIMER PARM SPECIFIED
CC0033 00                         37+RASTRSW4 DC    X'00' .                  TRACE SWITCH
                                  38+*        EQU   X'01' .                  TRACE 'ALL' PARM SPECIFIED



CC0034 C033                       40+RASSNPNC DC    H'51' .                  SNAP TABLE ENTRIES

000036 C0C00CC00CCC0000           42+RASNPCDS DC    XL8'0' .                 SPECIFIED SNAP OP CODES
                                  43+*        EQU   X'08C0C00000C0C000' .    SSK SNAP
                                  44+*        EQU   X'0009000000000000' .    ISK SNAP
                                  45+*        EQU   X'C0008C0000000000' .    SSM SNAP
                                  46+*        EQU   X'C000008200000000' .    LPSW SNAP
                                  47+*        EQU   X'000000009C000000' .    SIO SNAP
                                  48+*        EQU   X'C00000000090000' .     TIO SNAP
                                  49+*        EQU   X'0000000000009E00' .    HIO SNAP
                                  50+*        EQU   X'000000000000009F' .    TCH SNAP
000008                            51+SNPLEN   EQU   *-RASNPCDS .             LENGTH OF TABLE

CC003E CCCC
000040 CCCCCCC00CCC0000           53+RASNPCUU DC    5F'0' .                  SNAP CUU TABLE
00C00A                            54+SNPCUULN EQU   (*-RASNPCUU)/2 .         SIZE OF CUU TABLE

CC0054 0C                         56+RASNPSW1 DC    X'00' .                  SNAP SWITCH 1
                                  57+*        EQU   X'01' .                  SNAP 'IO ' PARM SPECIFIED
CC0055 C0                         58+RASNPSW2 DC    X'00' .                  SNAP SWITCH 2
CC0056 00                         59+RASNPSW3 DC    X'00' .                  SNAP SWITCH 3
                                  60+*        EQU   X'10' .                  SNAP 'SVC' PARM SPECIFIED
                                  61+*        EQU   X'01' .                  SNAP 'EXT' PARM SPECIFIED
                                  62+*        EQU   X'02' .                  SNAP 'INT' PARM SPECIFIED
                                  63+*        EQU   X'04' .                  SNAP 'ATTN' PARM SPECIFIED
                                  64+*        EQU   X'08' .                  SNAP 'TIMER' PARM SPECIFIED
000057 CC                         65+RASNPSW4 DC    X'00' .                  SNAP SWITCH 4    SNAP ALL
                                  66+*        EQU   X'01' .                  SNAP 'ALL' PARM SPECIFIED
```

Figure 59 (Part 1 of 3). Emulator Service Aids Common Data Area

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

000058 C0C00CC0CCCC0000             68+CMPBLK   DC    XL120'0' COMP INPUT AREA
00000C                              69+CMPLEN1  EQU   12 .                    LENGTH OF CMPBLK1
CC000A                              70+CMPLEN   EQU   (*-CMPBLK)/CMPLEN1 .   TABLE SIZE
0000D0 C00C0CC00CC00000             71+CMPBLK2  DC    XL140'0' .             COMP INPUT SAVE AREA
CC00DE                              72+CMPLEN2  EQU   14 .                    LENGTH OF CMPBLK2



00015C CC00CCC0
000160 0000000C0000000000           74+RASPSWSV DC    D'0' .                 PSW SAVE AREA
000168 CCCCCCC00CC0C000             75+PSWSAVE  DC    XL10'0' .
000172 C00C0CC00CC00000             76+PSWMSK2  DC    XL10'0' .              MASK FOR PSW OPTION
00017C 00                           77+PSWSW    DC    X'00' .                PSW SWITCH



00017D 00                           79+STORGSW  DC    X'00' ,                STORAGE SWITCH
                                    80+*        EQU   X'00' .                STORAGE=(EMBLKS,DOS)
                                    81+*        EQU   X'01' .                STORAGE=(EMU,DOS)
                                    82+*        EQU   X'02' .                STORAGE=(ALL,ECS)
                                    83+*        EQU   X'10' .                STORAGE=(,NODCS)



00017E 00                           85+RASFLG1  DC    X'0' .                 FLAG BYTE
                                    86+*        EQU   X'01'                  TRACE NOT COMPLETE
                                    87+*        EQU   X'02'                  PROGRAM CHECK INTERRUPT
                                    88+*        EQU   X'04'                  CAUSE OF SNAP IS EQUAL COMP



CC017F 00                           90+RASPCSW  DC    X'0' .                 PROGRAM CHECK SWITCH



000180 CCCC00C0                     92+ARASPCE  DC    A(0) .                 REPLACEMENT ADDR FOR DRIPC
CC0184 0C000000                     93+RASPC    DC    A(0) .                 V(IIVPCE) - INIT BY IIVRCP
CC0188 CCCCCCC0                     94+AIIVPCI  DC    A(0) .                 A(IIVPCI)

00018C 000000C0                     96+ARASSVC  DC    A(0) .                 REPLACEMENT ADDR FOR DRISVC
CC0190 00000000                     97+RASSVC   DC    A(0) .                 V(IIVSVC) - INIT BY IIVRCP
000194 00C0CCC0                     98+AIIVSCI  DC    A(0) .                 A(IIVSCI)

000198 00000000                    100+ARASRTE  DC    A(0) .                 REPLACEMENT ADDR FOR DRIOTHER
CC019C CCCCCCC0                    101+RASCTHER DC    A(0) .                 V(IIVRTE) - INIT BY IIVRCP

CC01A0 C000CCC0                    103+RASART2C DC    A(0) .                 V(IIVRTER2) - INIT BY IIVRCP

CC01A4 00000000                    105+ARASNP   DC    A(C) .                 V(IIVSNP)

CC01A8 CCCCCCC0                    107+SVCCHN   DC    A(0) .                 POINTER TO BEGINNING OF CHAIN
0001AC 00000000                    108+VIIVCON  DC    A(0) .                 INITIALIZED BY IIVRAS
CC01B0 C0000C00                    109+SVCEXAD  DC    A(0) SVC EXIT ROUTINE ADDR
0001B4 C0000CC0                    110+PCEXITAD DC    A(0) PC EXIT ROUTINE ADDR
0001B8 00000000                    111+LEXADR   DC    A(0) LEX EXIT RTN ADDRESS
CC01BC CCCCCCC0                    112+ASYNEXAD DC    A(0) ASYN EXIT ROUTINE ADDR
0001C0 C9C9E5E9C1E24040            113+SVCEXRTN DC    CL8'IIVRAS  ' SVC EXIT ROUTINE NAME
0001C8 C9C9E5E9C1E24040            114+PCEXRTN  DC    CL8'IIVRAS  ' PC EXIT ROUTINE NAME
CC01D0 C9C9E5E9C1E24040            115+LEXRTN   DC    CL8'IIVRAS  ' LEX EXIT RTN NAME
0001D8 C9C9E5E9C1E24040            116+ASYNEXRN DC    CL8'IIVRAS  ' ASYN EXIT ROUTINE NAME



CC01E0 C000                        118+CUUSAVE  DC    H'0' .                 CHANNEL AND UNIT SAVE AREA
0001E2 00                          119+TYPSAVE  DC    X'0' .                 TYPE OF INTERRUPT
CC01E3 C023                        120+LEXMSG   DC    AL2(L'EXMSG-1) LENGTH MINUS ONE
CC01E5 E2D5C1D740C4E4D4            121+EXMSG    DC    C'SNAP DUMP INVOKED BY MODULE XXXXXXXX' MSG
CC0201                             122+         ORG   *-L'LEXRTN RESET IC
CC0201                             123+MODNM    DS    XL8 LOAD MODULE NAME
000209                             124+         ORG   EXMSG+L'EXMSG RESET IC
000209 00
CC020A                             125+         CNOP  2,4 ALLIGN ON HALF WORD BOUNDRY
00020A 0015                        126+LOPTMSG1 DC    AL2(L'OPTMSG1-1) .     LENGTH MINUS ONE
00020C E7E7E7E740E7E7E7            127+OPTMSG1  DC    C'XXXX XXX   SNAP X CF X' .    MSG
```

Figure 59 (Part 2 of 3). Emulator Service Aids Common Data Area

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

CC020C                           128+          ORG   OPTMSG1 .                RESET IC
CC020C                           129+OPTION    DS    XL4 .                    CAUSE
000210                           130+          DS    X FILLER
CC0211                           131+OPTCUU    DS    XL3 CUU
00021C                           132+          ORG   OPTMSG1+L'OPTMSG1-6 .    POINT TO SNAP NO
CC021C                           133+SNPNO     DS    XL6 .                    SNAP X OF X
CC0222                           134+          ORG   OPTMSG1+L'OPTMSG1 .      RESET IC
000222 0013                      135+LPSWSVE   DC    AL2(L'PSWSVE-ONE) .      LENGTH OF MSG
000224 D7E2E67EE7E7E7E7          136+PSWSVE    DC    C'PSW=XXXXXXXXXXXXXXXX' .MSG
CC0238 0C81CCC0                  137+BHDR1     DC    AL2(L'HDR1+8),AL2(0) BDW
00023C 007D0000                  138+LHDR1     DC    AL2(L'HDR1+4),AL2(0) RDW
CC0240 40404C4040404040          139+HDR1      DC    CL121' ' HEADER RECORD
000240                           140+          ORG   HDR1 RESET IC
000240 F1D6E261F3F6F040          141+          DC    C'1OS/360  DOS EMULATOR  SERVICE AIDS ' ID
CC0264                           142+          DS    CL20 FILLER
000278 E3C9D4C540                143+          DC    CL5'TIME' ID
CC027C E7E7E7E7E7E7              144+HDRTIME   DC    C'XXXXXX' TIME
CCC283                           145+          DS    CL15 FILLER
000292 C4C1E3C540                146+          DC    CL5'DATE' ID
CC0297 E7E7E7E7E7                147+HDRDATE   DC    C'XXXXX' DATE
00029C                           148+          DS    CL15 FILLER
0002AB D7C1C7C540                149+          DC    CL5'PAGE' ID
CC02B0 E7E7E7E7                  150+HDRPAGE   DC    C'XXXX' PAGE
0002B9                           151+          ORG   HDR1+L'HDR1 RESET IC
C002B9 00
CC02BA 0001                      152+LINECNT   DC    H'1' .                   LINE COUNT FOR IIVPCP
0002BC 0000                      153+PAGECNT   DC    H'0' .                   PAGE COUNT FOR IIVPCP
CC02BE CC00
0002C0 CCCCCCC0                  154+RASDIAG   DC    F'0' ADDR OF FIRST DIAG BLOCK    2-1
0002C4 00000000                  155+RASCCW    DC    F'0' SAVE IIVCCW ADDRESS         2-1
CC02C8 000000C0000000.00         156+RASTRSVC  DC    XL255'00' .              SVC TRACE TABLE
C003C7 00
0003C8 0000000000000000          157+PTCHAREA  DC    10D'0' .                 MAINTENENCE PATCH AREA
CC0000                           158+DIAGBLK   DSECT  DIAGNOSTIC BLOCK FOR CCW    2-1
CC0000                           159+DIAGNXT   DS    F ADDR OF NXT BLOCK          2-1
000004                           160+DIAGCUU   DS    CL2 DOS CUU                    2-1
CC0006                           161+DIAGFLAG  DS    CL1 FLAG BYTE                  2-1
CC0080                           162+DIAGREL   EQU   X'80' ON IF RELOC+SAVE       2-1
CCCCC7                           163+          DS    CL1 UNUSED                     2-1
CC0CC8                           164+DIAGBUF   DS    CL108 BUFFER(SAVE FOR DOS CH.PGM) 2-1
000074                           165+DIAGLNG   EQU   *-DIAGBLK LENGTH OF DIAG BLK          2-1
```

Figure 59 (Part 3 of 3). Emulator Service Aids Common Data Area

Input/Output Block

Initialized by: IIVOPN

Modified by: IIVPCE, IIVINT, IGG019S1, OS, IIVCHK

Interrogated by: IIVPCE, IIVCHK

Located in: COMTAB (X'38' offset)

The input/output block (IOB) is the primary means of communication between a requestor of an I/O operation and the I/O supervisor. All of the information passed between the requestor and the I/O supervisor is either contained in the IOB or pointed to by the IOB.

Although the I/O supervisor uses IOBs, it neither creates them nor disposes of them; IOBs belong to the requestor of an I/O operation.

Only those parts of the IOB that are significant to the Emulator are shown in Figure 60 below.

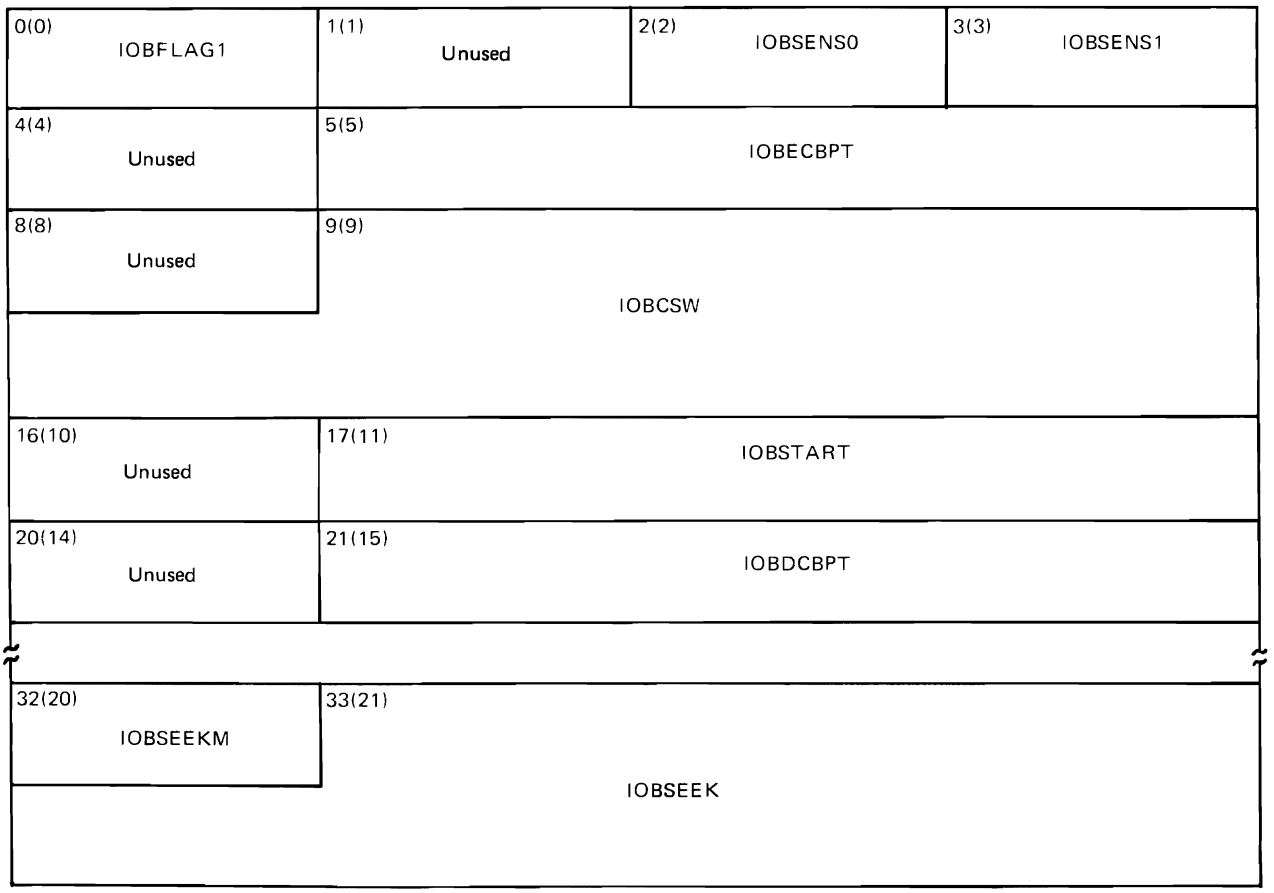


Figure 60. Input/Output Block Fields Used by the Emulator

Input/Output Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1	IOBFLAG1	Flag byte 1.
		00..	No chaining.
		01..	Command chaining.
		10..	Data chaining.
		11..	Both command and data chaining.
		.	.
		.	.
		.	.
		.... ..1.	Not a related I/O request.
			Note: Bits 0, 1, and 6 are set as required by EXCP.
2(2)	..1	IOBSENS0	Sense byte 1 is tested or passed to DOS (see IIVCHK for details).
3(3)	...1	IOBSENS1	Sense byte 2 is tested or passed to DOS (see IIVCHK for details).
5(5)	.3	IOBECBPT	Address of the ECB to be posted upon completion of the I/O event.
9(9)	.7	IOBCSW	Low order seven bytes of the last CSW for this request are passed to DOS when I/O interruptions are simulated.
17(11)	.3	IOBSTART	Address of the first CCW in the channel program to be executed.
21(15)	.3	IOBDCBPT	Address of the DCB needed for this request.
32(20)	1	IOBSEEKM	M = the number of the DEB extents for the data set to be read or written.
33(21)	.7	IOBSEEK	BBCCHHR = the seek address of a direct-access volume:

	Disk
Bytes 1-2	Zeros
Bytes 3-4	Cylinder number
Bytes 5-6	Read/write head number
Bytes 7	Record number

ISAM Block (ISBLK)

Initialized by: IIVIS

Modified by: IIVIS

Interrogated by: IIVIS, IIVGR2 (FIDBLK only), and IIVPCE (FIDBLK only)

Pointer in: ISREGSVA field (following FIDBLK field, which is pointed to by EMUCONS)

The ISAM block (ISLBK, Figure 61) is built by the Emulator when a DTF is opened and is used at I/O macro time to map the DOS DTF to the OS DCB.

ISBLK

0(0)	ISREGSAV
72(48)	FIDBLK
92(5C)	ISECB
96(60)	ISTCB
100(64)	ISREGSVA
104(68)	ISLIST
112(70)	ISWKNARA
116(74)	ISFLAGS
120(78)	ISDCB
356(164)	ISDECB
384(180)	ISDTFLMA
388(184)	ISDCB1
623(26F)	

Figure 61. ISAM Block (ISBLK)

ISAM Block (ISBLK) Description

Offset	Bytes and Alignment	Name Field	Hex. Dig.	Field Description, Contents, Meaning
0(0)	72	ISREGSAV		Register save area.
72(48)	32	FIDBLK		File identification block.
92(5C)	4	ISECB		Event control block used for communication between the Emulator main task and subtask.
96(60)	4	ISTCB		Address of task control block for subtask associated with ISBLK.
100(64)	4	ISREGSVA		Address of ISREGSAV.
104(68)	8	ISLIST		Parameter list used to communicate an I/O request from the main task to a subtask.
		Byte 1		
		ISCODE		I/O request code.
104(68)			00	WRITE NEW KEY.
			04	WAITF.
			08	GET.
			0C	PUT.
			10	WRITE KEY.
			14	ESETL.
			18	READ KEY.
		Bytes 2-4		
105(69)		ISDTFA		Address of DTFIS table.
		BYTES 5-8		
108(6C)		ISCOMTAB		Address of the COMTAB entry to which this I/O request is related.
112(70)	4	ISWKNARA		Address of the WRITE type KN work area.
116(74)	4	ISFLAGS		Indicators.
		Byte 1		
		TAFLAG1		I/O request status indicator.
116(74)		1...		GET successful.
		.1..		READ KEY issued.
		..1.		WRITE KEY issued.
		...1		WRITE NEWKEY issued.
		.... 1...		READ successful.
		xxx		Not used.
		Byte 2		
117(75)	1	TAFLAG2		OPEN/SETL status indicator.
		1...		Open for QISAM.
		11..		Open for QISAM load.
		1.1.		Open for QISAM search by record or generic key.
		1..1		Open for QISAM search by device address.
		...1		Set for all SETLs.
		.... 1...		Open for BISAM.
		1		Open for BISAM in DCB 2.
		.... 11..		Open for BISAM add in DCB 1.

Offset	Bytes and Alignment	Name Field	Hex. Dig.	Field Description, Contents, Meaning
		.... 1.1.		Open for BISAM RETRVE in DCB 1.
		.... 1.11		Open for BISAM RETRVE in DCB 2.
		.... 111.		Open for BISAM ADDRTR in DCB 1.
		 1111		Open for BISAM ADDRTR in DCB 2.
		Byte 3		
		TAFLAG3		Subtask/main task control indicator.
118(76)		1111 1111		Subtask in control.
		0000 0000		Main task in control.
		Byte 4		
119(77)		TAFLAG4		Not used.
120(78)	236	ISDCB		QISAM load mode: DCB when creating file. QISAM scan mode: DCB when sequentially retrieving file. BISAM mode: DCB only when adding records to file.
356(164)	28	ISDECB		BISAM-mode data event control block used when accessing file for random retrieval or add mode.
384(180)	4	ISDTFLMA		Save area for address of DTF (define the file) logic module.
388(184)	236	ISDCB1		BISAM-mode data control block when randomly retrieving and/or adding records to file.

ISK/SSK Table

Initialized by: IIVINT

Modified by: IIVPCE

Pointer in: EMUCONS

The insert storage key/set storage key (ISK/SSK) table contains one 1-byte entry for each 2K block of DOS dynamic storage. It is initialized to zeros by the initialization routine. The appropriate protection key for each block is stored in its related byte entry.

Job File Control Block

Initialized by: OS

Modified by: IIVINT, IIVOPN

Interrogated by: IIVINT

Obtained by: Issuing RDJFCB macro instruction

The job file control block (JFCB) is an internal representation of a DD statement.

Only those parts of a JFCB that are significant to the Emulator are described in Figure 62.

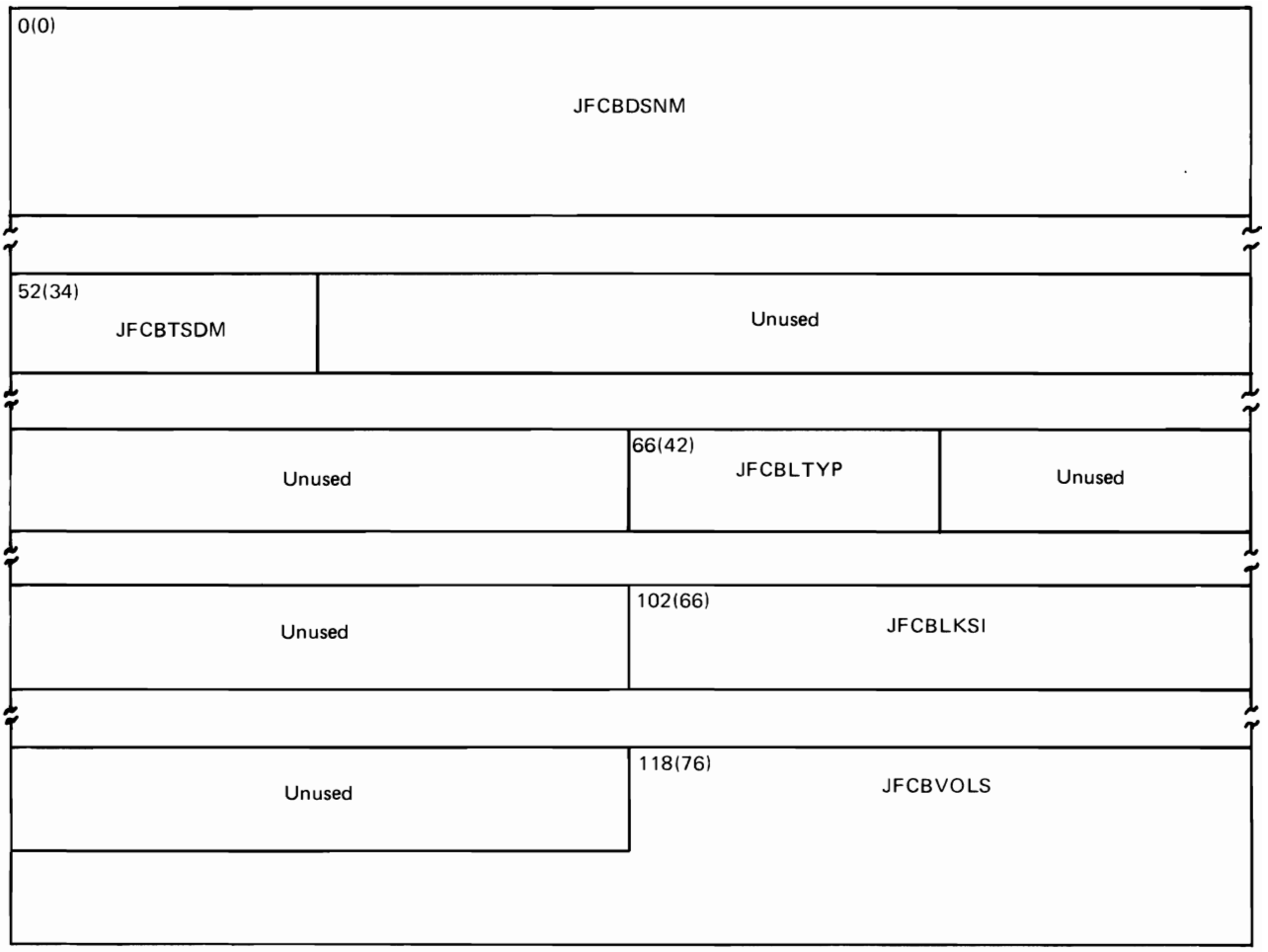


Figure 62. Job File Control Block Fields Used by the Emulator

Job File Control Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	44	JFCBDSNM	This field is initialized to 44 bytes of X'04' to indicate the VTOC data set name. The tenth character is examined to determine SYSIN and SYSOUT files.
52(34)	1	JFCBTSDM	A X'20' in this byte is assumed to mean a SYSIN or SYSOUT file.
66(42)	..1	JFCBLTYP	This byte is set to X'10' to assure BLP option.
102(66)	..2	JFCBLKSI	If the blocksize for staged I/O is not specified in a DD statement, this field holds a default blocksize (89 bytes for punch, 141 for print).
118(76)	..6	JFCBVOLS	The OPEN routines move volume identification (VOLID) into this field for mount requests.

Local Execution List

Initialized by: IIVINT

Modified by: IIVINT, IIVPCE, IIVLOG, IIVSTG, IIVPRP, execute local instruction, IIVRAS, IIVSNP, IIVSCI, IIVACI, IIVPCI, IIVRCP

Interrogated by: IIVCCW, IIVRTE, IIVAWV, IIVCWV, IIVPCE, IIVGR2, IIVRAS, IIVSNP, IIVSCI, IIVACI, IIVPCI, IIVRCP

Pointer in: EMUCONS and Emulator register 9

The local execution list (Figure 63) contains information that enables the Emulator to acquire control after an interruption and subsequently return control to DOS. See the Appendix for details concerning DOS emulation.

<table>
<tr><td colspan="4">0(0)
Programming Use</td><td>2(2)
Interruption Code</td></tr>
<tr><td>4(4)
ILC</td><td>CC</td><td>PM</td><td colspan="2">5(5)
Local Instruction Address</td></tr>
<tr><td colspan="5">8(8)
Register 14</td></tr>
<tr><td colspan="5">12(C)
Register 15</td></tr>
<tr><td colspan="2">16(10)
Reserved</td><td colspan="3">17(11)
True Origin Address</td></tr>
<tr><td colspan="2">20(14)
Reserved</td><td colspan="3">21(15)
Local Limit Address</td></tr>
<tr><td colspan="2">24(18)
Reserved</td><td colspan="3">25(19)
True Operation Pointer</td></tr>
<tr><td colspan="2">28(1C)
Reserved</td><td colspan="3">29(1D)
SVC Interruption Address</td></tr>
<tr><td colspan="2">32(20)
Reserved</td><td colspan="3">33(21)
Program Interruption Address</td></tr>
<tr><td colspan="2">36(24)
Reserved</td><td colspan="3">37(25)
Asynchronous Interruption Address</td></tr>
</table>

Figure 63. Local Execution List Fields Used by the Emulator

Local Execution List Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	2		The Emulator cannot change the condition of the first two bytes.
2(2)	..2		Interruption code. Identifies the cause or source of an interruption. When an interruption occurs while in local execution mode, the interruption code is placed in the local execution list.
4(4)		xx..	Instruction length code (ILC). When an interruption occurs while in local execution mode, the current value of the ILC is placed in this area of the local execution list.
4(4)		..xx	Condition code (CC). When an interruption occurs while in local execution mode, the current value of the condition code is placed in this area of the local execution list.
4(4)		 xxxx	Program mask (PM). When an interruption occurs while in local execution mode, the current value of the program mask is placed in this area of the local execution list.
5(5)	.3		Instruction address. When an interruption occurs while in local execution mode, the address of the next sequential instruction is placed in this area of the local execution list.
8(8)	4		Register 14. When an interruption occurs while in local execution mode, the contents of register 14 are placed into this area of the local execution list.
12(C)	4		Register 15. When an interruption occurs while in local execution mode, the contents of register 15 are placed into this area of the local execution list.
17(11)	3		True origin address. This area holds the address of the Emulator program corresponding to address 0 in the emulated environment. It must be on a 4096-byte boundary (the three low-order hexadecimal digits equal 000). The origin address is added to the local limit address to determine that the maximum local address falls within the storage limits of the system.
21(15)	3		Local limit address. This area holds the maximum address usable by the emulated DOS programs. The address value must be on a 4096-byte boundary -1 (the three low-order hexadecimal digits equal FFF). All local instruction and operand addresses are

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
			checked by the DOS Compatibility Feature (in Models 135 and 145) against the local limit address to determine whether they fall within the adjusted DOS storage area. Any address found to be greater than the local limit address causes an addressing exception. The DOS Compatibility Feature in Model 155 does not do this checking. However, all addresses contained in the local execution list are checked for boundary alignment.
25(19)	3		True operation pointer. When an interruption condition is raised while in local execution mode, the true address of the last executed instruction (except for the object of an execute instruction, in which case it is the true address of the execute instruction itself) is stored in this area of the local execution list.
29(1D)	3		SVC interruption address. When an SVC interruption occurs, the SVC interruption address from the local execution list replaces the instruction address of the current PSW. This area holds the address of the Emulator SVC interruption routine.[1]
33(21)	3		Program interruption address. When a program interruption occurs, the program interruption address from the local execution list replaces the instruction address of the current PSW. This area holds the address of the Emulator program interruption routine.[2]
37(25)	3		Asynchronous interruption address. When an external or I/O interruption occurs, the asynchronous interruption address from the local execution list replaces the instruction address of the current PSW.[3] A machine check interruption causes unpredictable results.

[1]The Emulator SVC interruption routine (IIVSVC) determines if device or data set sharing is involved. If not, IIVSVC moves the first two words from the local execution list to the DOS SVC old PSW location 32(20). It then moves the DOS SVC new PSW into the first two words of the local execution list. The DOS SVC interruption routine then handles the interruption in the usual manner. If device or data set sharing is involved, the SVC is checked to see if it is a 2, 4, or 11. If not, IIVSVC swaps PSWs in the manner described above. If the interruption is an SVC 2, 4, or 11, module IIVGR2 further analyzes it and then returns to IIVSVC with an indication of whether SVCs should be swapped or control passed to the next instruction following the SVC. For further information concerning DOS emulation interruption handling procedures, see the explanation of the DOS Compatibility Feature in the Appendix.

[2]The Emulator program interruption routine (IIVPCE) moves the first two words from the local execution list to the DOS program old PSW location 40(28). It then moves the DOS program new PSW into the first two words of the local execution list. The DOS program interruption routine then handles the interruption in the usual manner.

[3]The interruption is then handled in the normal OS manner by storing the old PSW and loading the appropriate new PSW.

Logical Unit Block

Initialized by: DOS

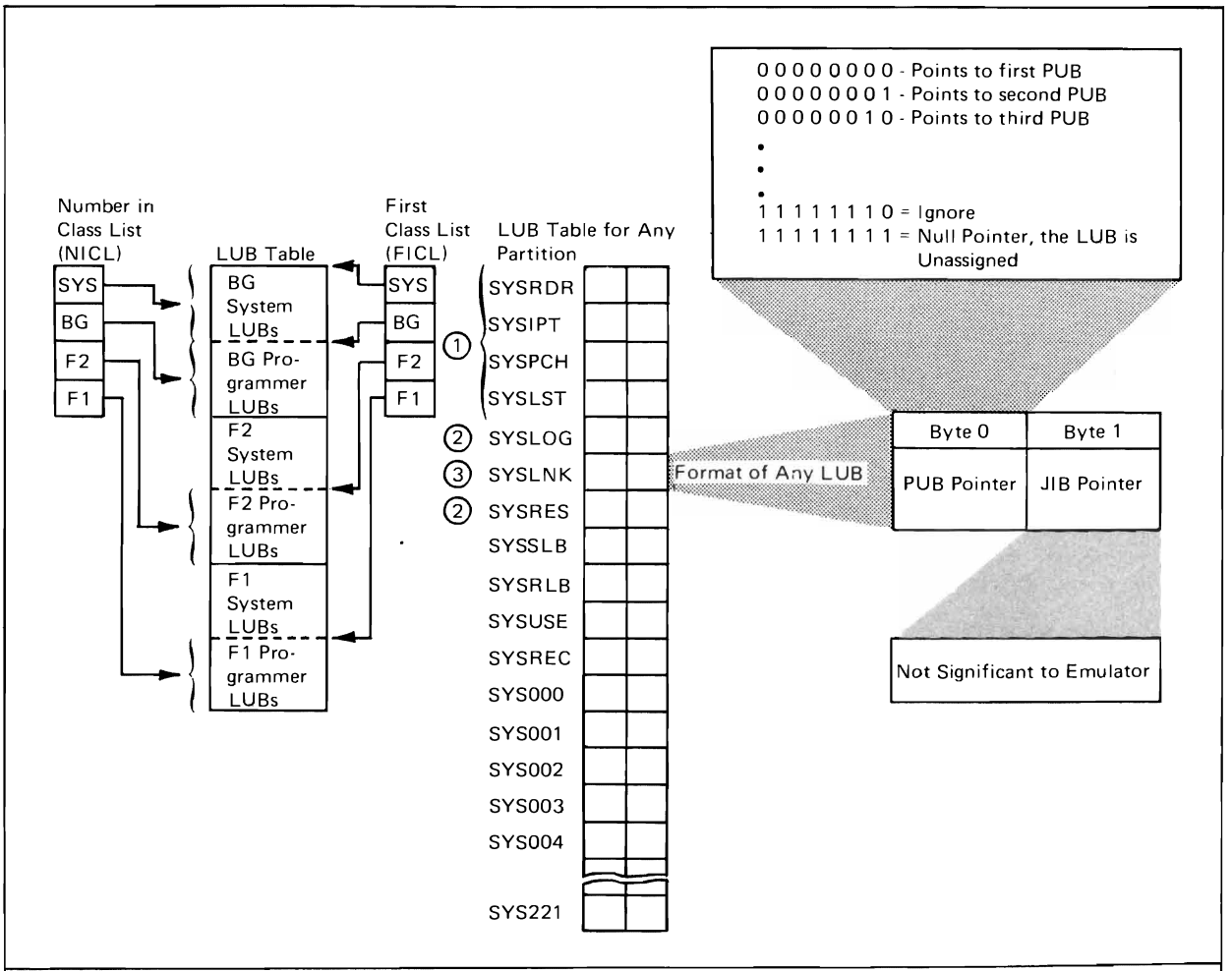
Interrogated by: IIVOPN, IIVRTE, IIVGR2

Pointer in: DOS COMREG

The logical unit block (LUB) is a 2-byte entry in the LUB table for each logical unit specified. The first byte is used as a PUB pointer; the second byte is not significant to the Emulator.

LUB Table

The LUB table (Figure 64) contains a 2-byte entry for each logical unit specified. The first entry must be for SYSRDR.



① When in single program initiation mode (foreground 1 or 2): must be unit record device and can be referenced by the program.

② When in single program initiation mode (foreground 1 or 2): can be referenced by the program.

③ Cannot be referenced by foreground programs.

Note: The location of the LUB table is found in bytes 76 and 77 (X'4C'-'4D') of the communications region. Label DOSLUB identifies the first byte of the table.

Figure 64. Logical Unit Block Table

Open Table

Initialized by: DOS

Interrogated by: IIVIS, IIVGR2, IIVDVS

Pointer in: B-transient area in DOS register 15 at SVC 2 time

The open table (Figure 65), located in the B-transient area, is a common section for all OPEN/CLOSE phases and is initialized by phase 1 of the OPEN or CLOSE monitor. Only the fields listed in the figure are significant to the Emulator.

Address Dec.	Hex	Bytes	Field Name	Field Description, Contents, Meaning
888	378	104	CSCDLB	DLBL image (sequential only).
992	3E0	8	LIMTBCKT	Limit (lower and upper CCHH).
1000	3E8	8	CCB	CCB used by DOS open.
1018	3FC	7	SEKADR	Seek bucket (CCHHR).
1033	409	44	SRCHKEY	Search argument.
1184	4A0	1	F4INDIC	VTOC DSCB indicator.
			F4INCORE	X'01' = on if VTOC DSCB read into core.
1185	4A1	1	MSGINDIC	Monitor ID for message routine.
			MSGOUT	X'F4' = message ID for sequential output.
			MSGIN	X'F3' = message ID for sequential input.
			MSGWF	X'F9' = message ID for sequential work file.
1187	4A3	1	OPTBFLGS	OPEN indicators.
			OPTBLNK	X'40' = SYSLNK open.
			OPTBSYFL	X'02' = SYSTEM file.
			OPTBFLPT	X'01' = file protection.
1192	4A8	4	LABADDR	Address of DLBL.

Figure 65. Open Table Fields Used by the Emulator

Physical Unit Block

Initialized by: DOS

Modified/Interrogated by: IIVPCE, IIVCHK

Interrogated by: IIVOPN, IIVPRP, IIVGR2

Pointer in: DOS COMREG

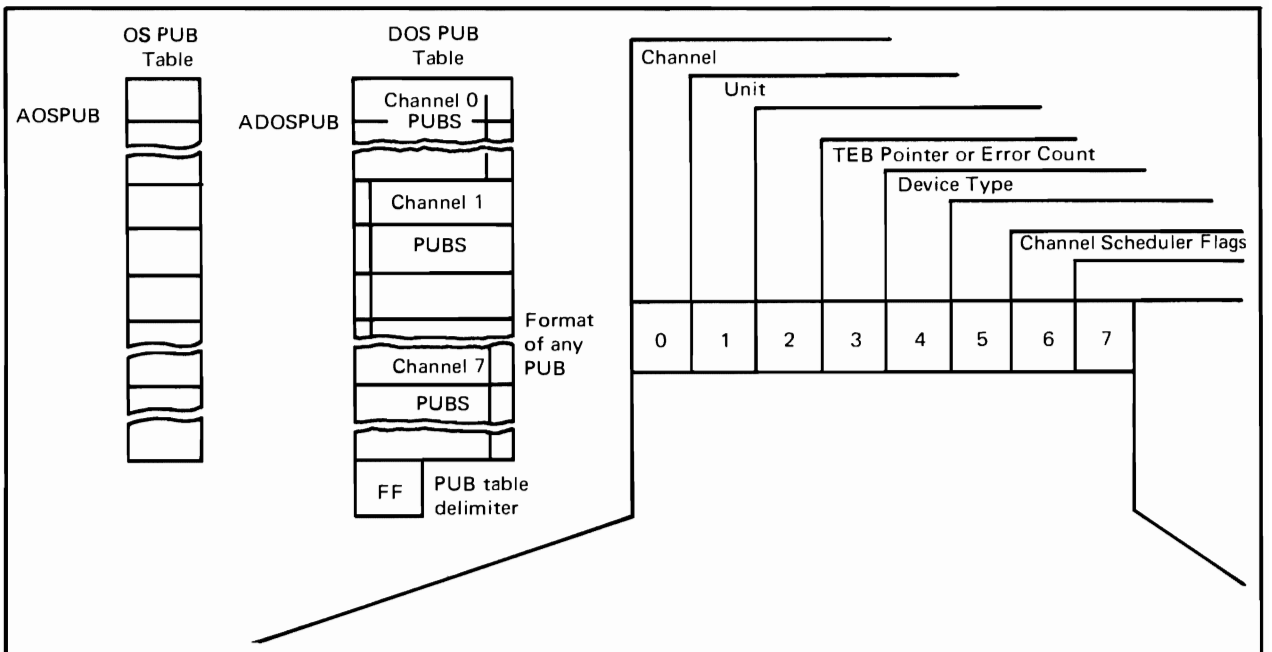
The DOS PUB and OS PUB tables provide a one-to-one correspondence between a DOS device and its associated OS device.

DOS PUB Table

The physical unit block (PUB) is an 8-byte entry in the DOS PUB table (Figure 66) for each physical unit specified. Byte 0 contains the channel number; byte 1, the device unit number; byte 3, the TEB pointer or the error count; byte 4, the device type code; and byte 6, the channel scheduler flags. These are the only bytes significant to the Emulator.

OS PUB Table

The OS PUB table (Figure 66) contains a 1-byte entry for each DOS PUB. If there is no OS equivalent for a DOS PUB entry, the OS PUB entry will be X'FF'.



Byte 0 - Channel number (hex 0-7, FF=NULL).

Byte 1 - I/O device unit number.

Byte 3 - If device is a magnetic tape unit* and TEBS are specified, this byte is a TEB pointer (hex 1, 2, 3...).

If device is a magnetic tape unit* but TEBS are not specified, this byte is an error counter.

If device is not a magnetic tape unit*, this byte is an error counter.

The error count must indicate to the DOS error routines when it has reached the maximum number of retries.

*2400 series or 2495 Tape Cartridge Reader.

Byte 4 - Device type codes are mapped with corresponding OS device type codes during Emulator initialization.

Byte 6 - A 1 in bit 7 indicates a 7-track tape unit.

Note: Only the fields noted above in any PUB are significant to the Emulator. Bytes 64 and 65 of the DOS communications region contain the address of the PUB table entry.

Figure 66. OS PUB and DOS PUB Tables

Post ECB List

Initialized by: IIVINT

Modified/interrogated by: IIVPCE, IIVRTE, IIVCHK

Pointer in: EMUCONS

The post ECB list contains one 2-byte entry for each entry in the communications table (COMTAB). The first byte is an index into COMTAB. The second byte is the condition code from the ECB when it was last posted.

An entry is made for each ECB posted by OS except for the three special ECBs (prompt, WTOR, and timer) and the SYSLOG ECB. The entries represent I/O interruptions to be simulated to DOS on a first-in, first-out queue.

Four pointers are used to process this table:

- PECBPTR1 This pointer to an entry in the table indicates that it just finished I/O interruption processing.
- PECBPTR2 This pointer indicates the next entry available in the table.
- POSTECB This pointer indicates the first entry in the table.
- ENDPTECB This pointer indicates the last entry +2 bytes to prepare for wraparound to the first entry in the table after the last entry is processed.

Program Information Block

Initialized by: DOS

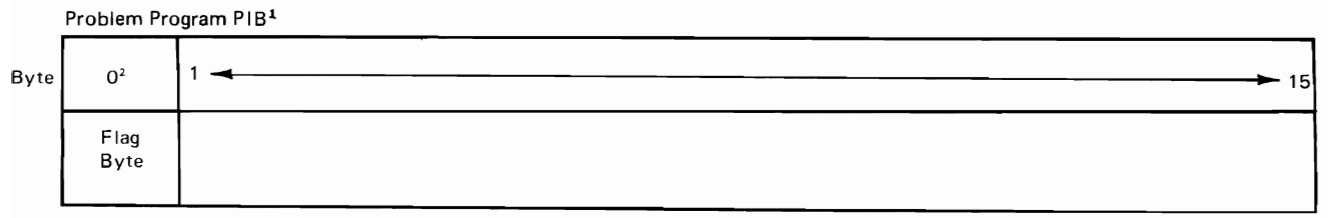
Interrogated by: IIVOPN

Pointer in: DOS COMREG

The first part of a program information block (PIB) contains program status information about DOS. Only byte 0 is significant to the Emulator (Figure 67).

The second part contains the address of the communications region (Figure 68), which is significant to the Emulator.

Both parts of the PIB must be in the following order: all bound, background, foreground 2, and foreground 1.



[1] Bytes 90-91 of the communications region contain the address of the first part of the PIB table (see Figure 77—DOS Communications Region Fields used by the Emulator).

[2] Byte 0 (PIBFLG) must indicate the following:
Background PIB must have X'82' when waiting for a CCB to be posted.
F2 and F1 PIBs must have X'80' if they exist and are inactive.

Figure 67. First Part of Program Information Block Table

Problem Program PIB [1]

Byte	0 ◄► 1[2]	2 ◄► 15
BG PIB	Address of BG Communications Region	
F2 PIB	Address of Area Communications Region	
F1 PIB	Address of Area Communications Region	

[1] Bytes 124-125 of the communications region contain the address of the second part of the PIB table (see Figure 77—DOS Communications Region Fields used by the Emulator).

[2] Background, F2 and F1 PIBs must contain a pointer to their associated communications region (COMREG) in bytes 0-1.

Figure 68. Second Part of Program Information Block Table

Program Status Word

Initialized by: DOS

Modified by: IIVOPN, IIVPCE, IIVLOG, IIVPRP, IIVSTG, IIVGR2

Interrogated by: IIVRTE, IIVACI, IIVPCI, IIVSCI, IIVSNP

Located in: Local execution list

The program status word (PSW, Figure 69) is a two-way communication link between a CPU and a program. All PSWs occupy permanently assigned main-storage locations. The PSWs are:

Name	Symbolic Name	Location
DOS SVC old PSW	DOSSVOLD	32(20)
DOS SVC new PSW	DOSSVNEW	96(60)
DOS program old PSW	DOSPGOLD	40(28)
DOS program new PSW	DOSPGNEW	104(68)
DOS I/O old PSW	DOSIOOLD	56(38)
DOS I/O new PSW	DOSIONEW	120(78)
DOS external old PSW	DOSXTOLD	24(18)
DOS external new PSW	DOSXTNEW	88(58)
DOS machine check old PSW	DOSMCOLD	48(30)
DOS machine check new PSW	DOSMCNEW	112(70)

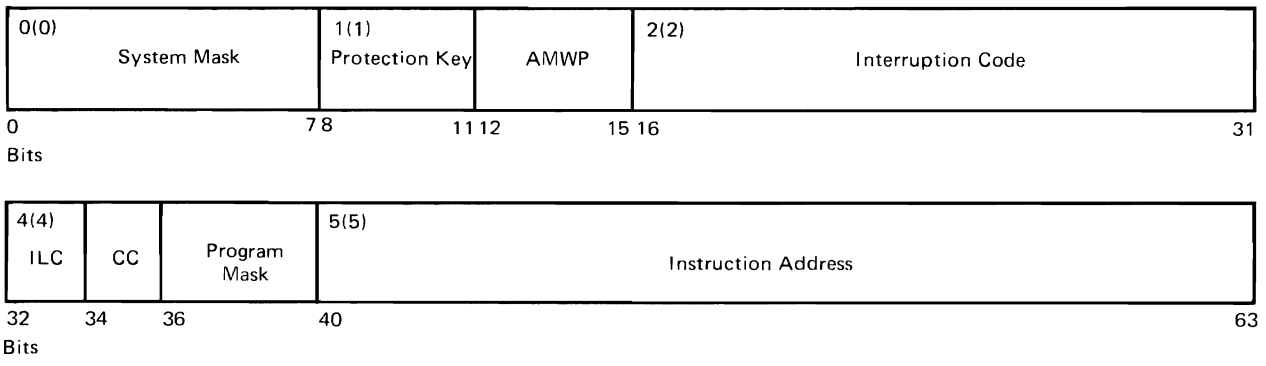


Figure 69. Program Status Word

Program Status Word Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1		System mask. This field is used to determine if DOS is enabling interruptions. Each bit represents a potential interruption source. A system mask bit of one allows the corresponding source to cause an interruption. A mask bit of 0 prevents interruptions from occurring; they remain pending.
		1...	Channel 0.
		.1..	Channel 1.
		..1.	Channel 2.
		...1	Channel 3.
		.... 1...	Channel 4.
		.... .1..	Channel 5.
		.... ..1.	Channel 6.
		1	External interruptions (timer interruption key, and external signals of the direct control feature).
1(1)	4 bits	xxxx	Protection key. This key is matched with a storage key whenever data is stored in or fetched from a location that is protected.
1(1)	4 bits	 xxxx	AMWP bits. The WP bits are tested to determine wait or problem state. The four bits indicate:
		.... x...	Not significant to the Emulator.
		.... .1..	Machine check interruptions can occur.
		.... .0..	Machine check interruptions will remain pending.
		.... ..1.	The CPU is in the wait state.
		.... ..0.	The CPU is in the running state.
		1	The CPU is in the problem state.
		0	The CPU is in the supervisor state.
2(2)	.2		Interruption code. Identifies the cause or source of an interruption. When simulating an I/O interruption, the Emulator stores the address of the channel end and device that caused the interruption here. 4(4) xx.. .... Instruction length code (ILC). For program or supervisor call interruptions, contains the length, in halfwords, of the last interpreted instruction. For I/O, external, and machine check interruptions, the ILC is unpredictable.

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
4(4)		..xx	Condition code (CC). Reflects the results of the execution of an instruction. Modified by the Emulator.
4(4)		 xxxx	Program mask (PM). Each bit represents a potential interruption source. A program mask bit of 1 allows the source to cause an interruption. A mask bit of 0 prevents the interruption from occurring.
		.... 1...	Fixed-point overflow.
		.... .1..	Decimal overflow.
		.... ..1.	Exponent underflow.
		1	Significance.
5(5)	.3		Instruction address. The leftmost byte of the instruction to be executed.

Staged I/O Constants Block

Initialized by: IIVOPN

Modified/Interrogated by: IIVSTG, IIVAWV, IIVCWV, IIVPOV

Located in: COMTAB at X'38' offset

The staged I/O constants block (STGCON, Figure 70) which resides in the COMTAB IOB area for staged devices, contains data unique to the particular device being staged. The Emulator both creates and uses the STGCON.

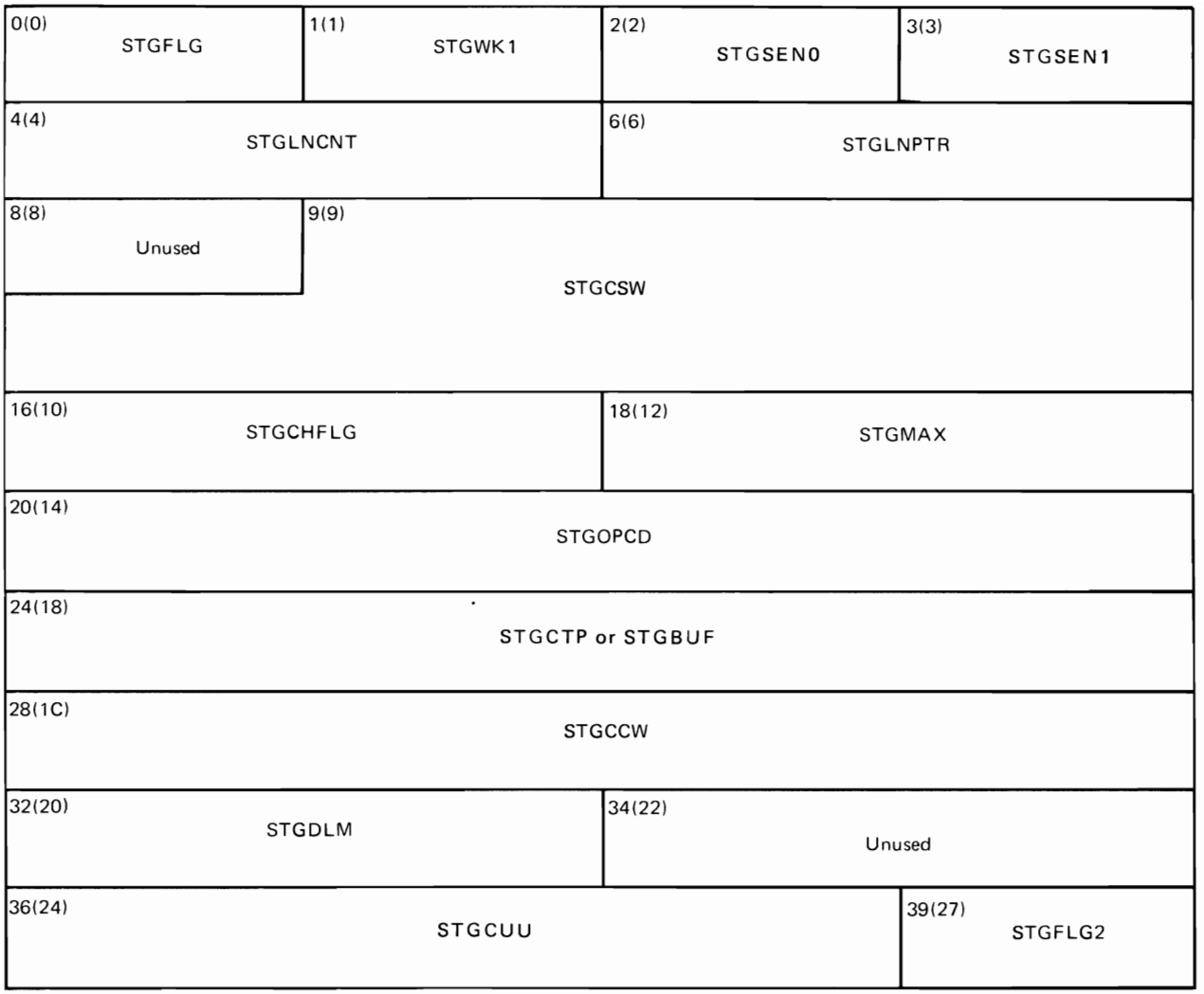


Figure 70. Staged I/O Constants Block (STGCON)

Staged I/O Constants Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	1	STGFLG	Flag byte 1.
		1...	Staged reader.
		.1..	Staged printer.
		..1.	Staged punch.
		...1	SYNAD (I/O error).
		.... 1...	Read, no feed, command encountered.
		.... .1..	EODAD (generate /& next SIO).
		.... ..1.	EOD (unit exception next SIO).
		1	First CCW in a chain.
1(1)	.1	STGWK1	Work byte.
2(2)	..1	STGSEN0	Sense information for last I/O operation.

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
3(3)	...1	STGSEN1	Sense information for current I/O operation.
4(4)	2	STGLNCNT	Number of lines per page (required by module IIVPOV).
6(6)	..2	STGLNPTR	Index into in-storage carriage tape.
8(8)	1	Unused	Reserved for future use.
9(9)	.7	STGCSW	Simulated channel status word.
16(10)	2	STGCHFLG	In-storage carriage tape channel flags.
18(12)	..2	STGMAX	Maximum number of bytes that can be transmitted to or from this device.
20(14)	4	STGOPCD	Opcode translate table address.
24(18)	4	STGCTP	Printer: in-storage carriage tape address, or
		STGBUF	Reader: address of the last OS input buffer, if a staged reader
28(1C)	4	STGCCW	DOS CCW address.
32(20)	2	STGDLM	DOS JCL delimiter.
34(22)	..2	Unused	Reserved for future use.
36(24)	3	STGCUU	DOS cuu in EBCDIC.
39(27)	...1	STGFLG2	Flag byte 2.
		xxxx	Reserved for future use.
		.... 1...	3211 line position check.
		.... .1..	Punch stacker RP3 has been previously selected.
		.... ..1.	Punch stacker P2 has been previously selected.
		1	Punch stacker P1 has been previously selected.

Tape Error Block

Initialized by: DOS

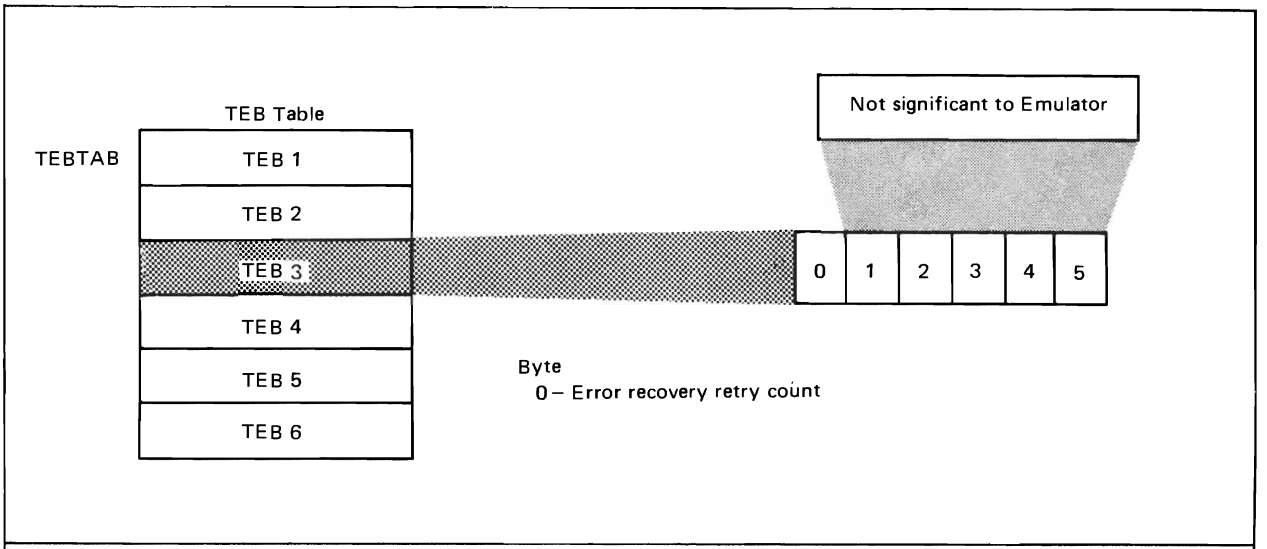
Modified by: IIVCHK

Pointer in: DOS COMREG

The tape error block (TEB, Figure 71) is generated for each 2400 Series Magnetic Tape or 2495 Tape Cartridge Reader unit and is the index for I/O error retry suppression by the Emulator.

A TEB is referenced from byte 3 of a magnetic tape unit PUB.

Only byte 0 of the 6-byte TEB is significant to the Emulator. This byte keeps track of the error recovery retry count; the count is set to 254 when the Emulator wants to suppress DOS retries.



One TEB is generated for each 2400 Series magnetic tape or 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 magnetic tape unit PUB.

Bytes 70 and 71 (X'46'-'47') of the communications region contain the address of the TEB table entry. Label TEBTAB identifies the first byte of the table.

Figure 71. Tape Error Block

Tape Error by Volume

Initialized by: DOS

Modified by: IIVCHK

Pointer in: DOS BG communication region extension

The tape error by volume (TEBV) is a DOS table composed of one status block and (n) error blocks and pointed to by the TEBVTAB field in the DOS BG COMREG extension. The TEBLEN and TSBLEN fields are used in conjunction with the TEBV index field in the PUB (byte 3) to locate the TEBV retry counter. This counter is set to a higher count to force DOS I/O retries. For further information concerning the TEBV, see the DOS System Programmer's Guide.

Only those parts of the TEBV that are significant to the Emulator are described in Figure 72.

Decimal displacement	Label	Byte length	Description
0	TEBLEN	1	Length of TEBV error block (for each error block generated)
1	TSBLEN	1	Length of TEBV status block (4, 6, or 22 bytes)
.			
.			
.			
24	TEBV	1	Retry counter

Figure 72. Tape Error by Volume Fields Used by the Emulator

Task Input/Output Table

Initialized by: OS

Interrogated by: IIVINT

Pointer in: EMUCONS

The task input/output table (TIOT) maps JFCBs to UCBs.

Only those parts of the TIOT (Figure 73) that are significant to the Emulator are described below.

DD Entry

0(0) TIOELNGH	Unused
4(4) TIOEDDNM	

Device Entry

Unused	1(1) TIOEFSRT

Figure 73. Task Input/Output Table Fields Used by the Emulator

Task Input/Output Table Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
			DD entry: A DD entry includes a device entry. Before allocation, there may be several device entries in each DD entry.
0(0)	1	TIOELNGH	This field contains the length of each DD entry and is used to scan the TIOT during initialization.
4(4)	8	TIOEDDNM	This field is scanned to determine DOS units.
			Device entry: During the execution of a problem program, contains 1 device entry for each allocated device.
1(1)	.3	TIOEFSRT	Devices other than 2321: address of the UCB.
			2321 Data Cell Drive: address of the description in the UCB of the cell in the bin.

Unit Control Block

Initialized by: OS, IGG019S1

Interrogated by: IIVINT, IGG019S1, IGG019SA

Pointer in: DEB

There is a unit control block (UCB) for each device attached to the system. It describes the characteristics of the device to the I/O supervisor and is used by the job scheduler during allocation of the device.

Only those parts of the UCB (Figure 74) that are significant to the Emulator are described below.

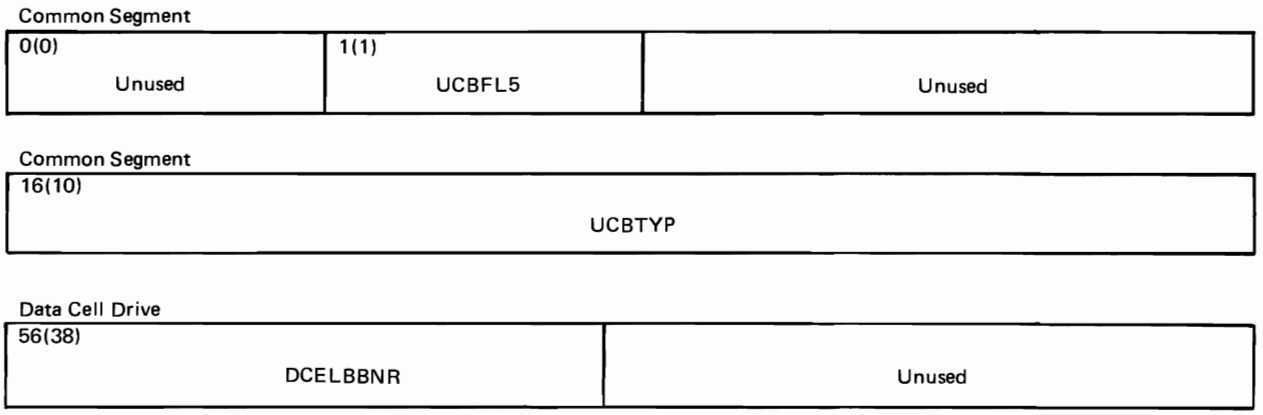


Figure 74. Unit Control Block Fields Used by the Emulator

Unit Control Block Description

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
1(1)	.1	UCBFL5 .... 1...	UCBEXTSN-UCB+24 (UCBNBRSN) contains the number of bytes of sense information and UCB+25 (UCBSNADR) contains the address of the sense information.
16(10)	4	UCBTYP	Bytes 3 and 4 of this field are used to identify the device class and unit type, respectively, for the purpose of cross-referencing them with DOS device types during initialization. The SIO appendage modifies the DEB, based on the device type. The abnormal end appendage determines whether OS or DOS is to handle I/O errors, based on the device type.
56(38)	2	DCELBBNR	Bin number for a data cell drive.

Volume Label

Initialized by: IIVDVS (simulates volume label used by DOS)

Interrogated by: DOS, IIVVIO, IIVDVS

Located in: first record (tape) or cylinder 0, track 0, record 3 (DASD)

A volume label is 80 characters long and identifies the volume and its owner. On direct-access volumes, it is record number 3, which follows the two IPL records. It is recorded as an 84-byte physical record consisting of a 4-byte key area containing VOL1 and an 80-byte data area. Figure 75 shows the fields in the volume label that are significant to the Emulator.

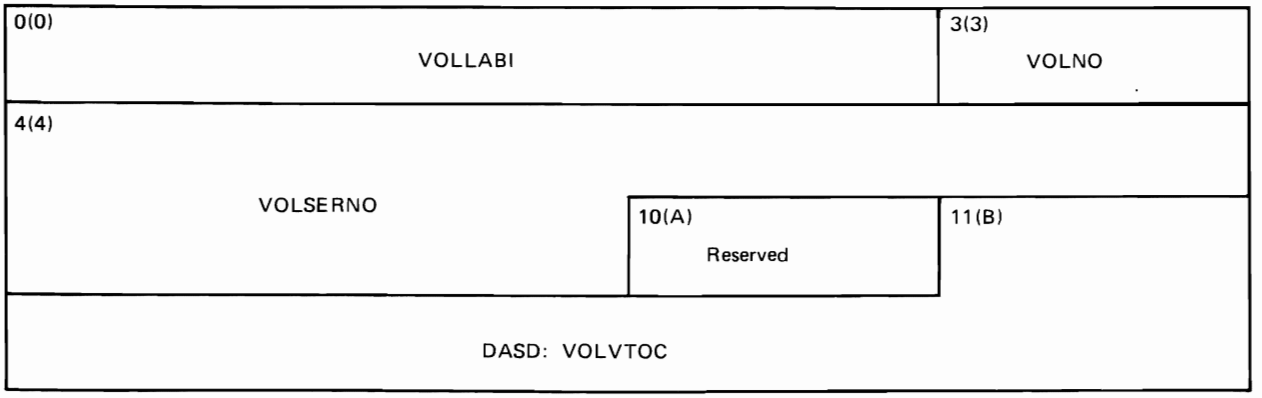


Figure 75. Volume Label Fields Used by the Emulator

Offset	Bytes and Alignment	Field Name	Field Description, Contents, Meaning
0(0)	3	VOLLABI	Label identifier - VOL.
3(3)	...1	VOLNO	Volume label sequence number - 1.
4(4)	6	VOLSERNO	Volume serial number that uniquely identifies the volume. This field may contain from one to six alphabetic or numeric characters, left justified with blanks in the remainder of the field.
11(B)	...5	VOLVTOC	Direct-access storage: The CCHHR address of the VTOC DSCB on this volume.

DIAGNOSTIC AIDS

Hints for Debugging

Emulator Dependencies on DOS, OS, and Hardware

- Dependence on DOS
- Dependence on OS
- Dependence on Hardware

Service Aids

Emulator General Register Assignments

Emulator Message-to-Module Relationships

HINTS FOR DEBUGGING

When error conditions such as program checks, channel program checks, or channel protection checks occur, they may be due to violations of Emulator restrictions by DOS programs. One example is the modification of a CCW by a DOS problem program between its issuance of an EXCP and the WAIT. The following information may be of assistance in debugging such conditions.

The conditions listed below, found in DOS and/or Emulator storage, may indicate violations:

- Channel program CCW addresses have been adjusted only part way through a group of CCWs by module IIVCCW (adjust CCW data address routine).
- Channel programs contain data addresses that appear to be improperly adjusted (from true to local addresses or vice versa).
- Channel programs or CCWs are properly adjusted, but are not currently in use (that is, no active COMTAB entries point to the channel program in question).
- Input channel programs have the SLI bit off, and the sum of its data address and the count field exceeds the address of the DOS area.
- Program check occurs in module IIVCCW during execution of the adjust CCW string instruction.
- DOS CSW status indicates a program check. The CSW program check bit is set by module IIVABN when module IIVCCW could not adjust the channel program or the BEBLK was full.

Emulator tables and fields that may be useful in locating channel programs and determining I/O status are:

- COMTAB - contains one entry per device with all I/O-related information, including a pointer to the last channel program used for the device (DCB, IOB pointers, ECB, etc.).
- COMTAB - while examining I/O request and/or the IOB, keep the following in mind:
  - Stand-alone seeks are simulated by use of the CSW field contained in the IOB. All other fields in the IOB relate to the last EXCP issued. The same applies to stand-alone sense commands where the IOB sense field is nonzero at the time the sense is issued, except that the simulated CSW is stored in the EMUCCW1 field of COMTAB.
  - The IOB is also used for issuing NOP commands within the Emulator and, therefore, the information contained in the IOB is not always associated with a DOS request.
- NOSIO - SIO counter in CSECT IIVCON. The value in this counter represents the number of DOS I/O operations initiated by OS EXCPs whose associated I/O interruptions have not been queued in the post ECB list.
- NOIOPEND - number of queued I/O interruptions to be simulated to DOS by IIVPCECK, entry point to the check I/O routine (IIVCHK). The value in this counter represents the number of entries in the post ECB list. This counter is also located in IIVCON.

Note: DOS programs violating storage protection requirements can also produce unpredictable results.

EMULATOR DEPENDENCIES ON DOS, OS, AND HARDWARE

Dependence on DOS

DOS Low Storage and Communication Regions

The following area in DOS low storage and communication region is referred to by the Emulator. This listing briefly describes what the area contains that is relevant to the Emulator.

DOS Area	Address (Hex)	Description of Use
Low storage (Figure 76)	0-7F	Bytes 0-127 are as assumed to exist as in stand-alone DOS.
	14	As in stand-alone DOS, contains the pointer to the background (BG) communications region.
DOS BG communication region (COMREG) (Figure 77)	2C	Contains the label area length.
	34	X'08' in this byte must indicate timer support.
	35	X'40' in this byte must indicate a multiprocessing system.
	38	X'08' in this byte must indicate that the job is to be canceled.
	40	Contains the address of the PUB (physical unit block).
	46	Contains the address of the TEB (tape error block).
	48	Contains the address of FICL.
	4A	Contains the address of NICL.
	4C	Contains the LUB.
	58	Contains the LIOCS communication bytes.
	5A	Must contain a 2-byte address of the first part of the PIB.
	6E	Contains the logical transient key.
	7C	Must contain a 2-byte address of the second part of the PIB table.
	88	Contains a pointer to the communication region extension; if the latter does not exist, must contain zeros.

DOS Area	Address (Hex)	Description of Use
DOS F1 and F2 COMREGs (Figure 77)	38	X'08' in this byte must indicate that the job is to be canceled.
DOS BG COMREG Extension (Figure 78)	0	Must contain an address pointer to the CE table or zeros.
	12	Contains the PIK.
	20	Contains the address of the TEBV table.

0(0) ← Reset to Zeros after IPL → 13								
20(14) Comm Region (COMREG) Address	24(18) External Old PSW	32(20) SVC Old PSW	40(28) Program Old PSW	48(30) Machine Check Old PSW	56(38) I/O PSW	64(40) CSW	72(48) CAW	76(4C) BG Job Duration
80(50) System Timer	84(54) System Timer of Day	88(58) External New PSW	96(60) SVC New PSW	104(68) Program Check New PSW	112(70) Machine Check New PSW			120(78) I/O New PSW

Figure 76. DOS Low Storage

COMREG[1]

0(0)	44(2C) Label Area Length	52(34) Machine Configuration Byte	53(35) System Configuration Byte	54(36)
56(38) Job Control Byte	57(39)	64(40) Address of PUB	66(42)	70(46) Address of TEB
72(48) Address of FICL	74(4A) Address of NICL	76(4C) Address of LUB	78(4F)	88(58) LIOCS Communication Bytes
90(5A) Address of First Part of PIB Table	92(5C)	110(6E) Logical Transient Key	112(70)	124(7C) Address of Second Part of PIB Table
126(7E)	136(88) Pointer to Communication Region Extension			

[1]The address of the communications region is in fixed location X'14' - '17' (see Figure 76 - DOS Low Storage).

Figure 77. DOS Communications Region Fields Used by the Emulator

BGXTNSN

0(0)	4(4)	18(12)	20(14)	32(20)	
CE Table Address		Program Interrupt Key (PIK)		TEBV Table Address	63(3F)

Figure 78. DOS Background Communications Region Extension Fields Used by the Emulator

DOS Control Blocks

The DOS control blocks in Figure 79 are referred to by the Emulator. This directory briefly describes how each block is used. The field names listed are those contained in the DOS supervisor assembly listings. The blocks are further described in the "Data Areas" section.

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
CCB	CCBCOM1	2	2	1	Bit 1 is normally set on at channel end to signify that the I/O operation was completed.
		9	9	3	Bytes 9-11 must contain the address of the CCW associated with this CCB.
LUB		0	0	2	The first 2-byte entry in each LUB table must be for SYSRDR. Byte 0 is used as a PUB pointer.
PIB	PIBFLG	0	0	1	The F1 and F2 program areas must have X'80' when they are inactive. The BG program area must have X'82' when it is waiting for a CCB to be posted.
PIB	PIBCOMRA	0	0	2	Bytes 0 and 1 of the second part of PIB must contain the address of the corresponding communications region.
PUB	PUBCHANN	0	0	1	Must contain the channel address.
PUB	PUBDEVND	1	1	1	Must contain the device address.
PUB	PUBERR	3	3	1	Must contain either the error retry count or the TEB index for I/O error retry suppression by the Emulator.
PUB	PUBDEVTY	4	4	1	Contains device-type codes that are mapped with corresponding OS device type codes during Emulator initialization.
PUB	PUBCSFLG	6	6	1	A 1 in bit 7 indicates a 7-track tape unit.
TEB	TEBERRCT	0	0	1	Error retry count is set to 254 when the Emulator wants to suppress DOS retries.
TEBV	TEBLEN	0	0	1	Length of TEBV error block (for each error block generated).
TEBV	TSBLEN	1	1	1	Length of TEBV status block (4, 6, or 22 bytes).
TEBV	TEBV	24	18	1	Retry counter.

Figure 79. DOS Control Blocks

DOS IPL and Initialization

The Emulator also depends on DOS for the following information:

- IPL records - Simulation of DOS IPL by the Emulator is based on the IPL records description in the IBM System/360 Disk Operating System, IPL and Job Control Programs.
- $$A$IPL2 - This phase is loaded during DOS initialization. The Emulator scans for the DOS clear main-storage routine in this phase. The operation code of a move character (MVC) instruction in the clear main-storage routine is then set to 0. (See 'DOS IPL' in "Method of Operation" section.)
- First LPSW - The Emulator requires that the first LPSW instruction following an SVC 4 must signal that DOS is ready to begin processing IPL input.
- First SVC 14 - The first SVC 14 received must signal the end of DOS IPL and initialization.

DOS SIO

When the Emulator intercepts an SIO not originating from a DOS CE serviceability routine, registers 1 and 3 must contain the addresses of the associated CCB and PUB entry, respectively.

CE SIO

When the Emulator intercepts an SIO originating from a DOS CE serviceability routine, it is assumed by the Emulator to be executed from a higher storage location than the normal (DOS SIO) request. The contents of registers 1 and 3 are not significant, as after a DOS SIO.

DOS Time of Day

The address of the DOS SVC table must be contained in the last nonzero halfword preceding the DOS communications table.

B-Transient Phases, Sequence, and Table

- $$BOPEN or $$BOPEN2 - One of these DOS phases must set registers 2, 6, and 7 with the DTF pointer, the XTENT card image pointer, and the COMREG pointer, respectively.
- B-transient sequence - The sequence of calls between the B-transient phases, for OPEN, CLOSE, and EOV must be followed or results will be unpredictable.
- Open table - The format and fields significant to the Emulator for this table are found in "Data Areas."

- DLBL and EXTENT statements or, in the alternative, VOL, DLAB, and XTENT statements, supply the information needed to build SYSRES DASD labels.

- ISAM I/O macro calls - All calls for the logic (I/O access) phases must culminate with BAL 14,XXX(15) where register 15 is loaded from the DTFIS table (displacement X'10') with the address of a logic phase, register 1 contains the address of the DTFIS table, and XXX is the displacement into the standard ISMOD branch table for the desired I/O operation. Register 14 contains the return address for the calling routine.

- DTFIS - Defined within the problem program. The format and fields significant to the Emulator are found in "Data Areas".

Dependence on OS

The following OS facilities are used by the Emulator:

Macros

OS macros used by the Emulator are shown in Figure 80.

Note:

- Each module summary in the Program Organization section lists the macros issued by that particular module.

The macros in the following list can be better understood in the context of overall Emulator operation, which is presented in "Method of Operation." For a basic description of the operands of these macros and their operation, refer to Supervisor Services and Macro Instructions and Data Management Macro Instructions for OS or OS/VS.

OS Macro	Emulator module issuing macro	Use of Macro
ATTACH	IIVIS	Issued to attach (create) a subtask to process ISAM I/O macros at open. The subtask is assigned a higher dispatching priority than the Emulator task.
CHAP	IIVIS	Issued to reduce the dispatching priority of the Emulator task to expedite subtask processing.
CHECK	IIVIS	Issued to test completion of direct retrieval ISAM macro instructions, such as READ K, WRITE K, and WRITE KN.
	IIVSNP	Issued to test completion of WRITE.
	IIVRCP	Issued to test completion of WRITE.
CLOSE	IIVPRP	Issued after a MOUNT reply (to a prompt) has been entered. After emulation is finished, OS closes the Emulator-related data sets.
	IIVDVS	Issued after a DOS CLOSE macro has been trapped by IIVGR2 for a sequential DASD or direct-access data set residing on a shared volume.
	IIVIS	Issued to close OS indexed sequential data sets.
	IIVRCP	Issued to close the SYSDEBUG data set.
DCB	IIVABN, IIVCON, IIVOPN, IIVRCP, IIVIS	Issued to create the DCB format according to file organization and access methods.
DCBD	IIVSNP, IIVRCP, IIVDVS, IIVIS IIVOPN, IIVIN2, IIVSTG	Issued to generate a DSECT that shows the structure of a DCB.
DELETE	IIVRAS	Issued to delete IIVSNP and IIVRCP.
	IIVRCP	Issued to delete IIVACI, IIVPCI, and IIVSCI.
	IIVMSG	Issued to delete IIVMSG1, IIVMG2, and IIVMG3.
	IIVOPN	Issued to delete forms-control image.
DEQ	IIVIN2, IIVABN	Issued to release control of an OS device.
DETACH	IIVIS	Issued to remove (eliminate) a subtask at close.
ENQ	IIVIN2	Issued to request exclusive control of an OS volume for the issuing Emulator job. This prevents two Emulator jobs from using the same OS volume concurrently.

Figure 80 (Part 1 of 5). OS Macros Used by the Emulator

OS Macro	Emulator module issuing macro	Use of Macro
EOV	IIVDVS	Issued when additional space is required by means of a secondary allocation parameter on a sequential DASD output data set.
ESETL	IIVIS	Issued to terminate sequential retrieval.
EXCP	IIVPCE, IIVCHK	Issued on behalf of DOS I/O requests that are not directed to the console and not to be staged.
	IIVIN2	Issued to read the DOS bootstrap channel program.
	IIVOPN	Issued to ensure protection of OS password data sets.
EXTRACT	IIVINT	Issued to obtain the address of the TIOT from the TCB.
GET	IIVSTG	Issued on behalf of DOS input requests when staging has been indicated. The input operation is always to an OS DASD.
	IIVIS	Issued to sequentially retrieve records from an OS indexed sequential data set.
	IIVRCP	Issued to read service aids control statements.
GETMAIN/ FREEMAIN	IIVGET	Issued (GETMAIN) for the size of DOS plus 4K bytes. After DOS is aligned on a 4K boundary, IIVGET issues a FREEMAIN to release the extra 4K bytes; subsequent GETMAINs obtain space in the same area(s) for Emulator tables and for the DOS residence file's DCB and IOB.
	IIVSTG	GETMAIN issued for main storage for forms-control image. FREEMAIN issued to free old forms-control image if old one exists.
	IIVRCP	GETMAIN issued to obtain main storage for the trace table, SVC blocks, and DIAG blocks. FREEMAIN issued to release main storage obtained from the GETMAIN.
IDENTIFY	IIVIS	Issued to add an entry point to IIVIS for use by the subtask.
IMGLIB	IIVOPN	Issued to open the SYS1.IMAGELIB data set.
IOHALT	IIVPCE	Issued to terminate I/O on a tele-processing device when DOS performs an HIO operation.

Figure 80 (Part 2 of 5). OS Macros Used by the Emulator

OS Macro	Emulator module issuing macro	Use of Macro
LOAD	IIVOPN, IIVPUB	Issued to selectively bring staged I/O modules into main storage when staged I/O is requested, depending on the unit record device type in the DOS PUB table.
	IIVRCP	Issued to load IIVACI, IIVPCI, and IIVSCI, IIVRCW, user exit routines.
	IIVIN2	Issued to load Emulator modules.
	IIVPRP	Issued to load IIVRAS.
	IIVPUB	Issued to fetch volume-sharing modules.
	IIVRAS	Issued to load IIVSNP and IIVRCP.
	IIVMSG	Issued to load IIVMG1, IIVMG2, and IIVMG3.
	IIVABN	Issued to load IIVRAS.
OBTAIN	IIVDVS	Issued to examine and modify the DSCBs associated with a sequential DASD or direct-access data set located on a shared volume.
OPEN	IIVDVS	Issued after a DOS OPEN macro has been trapped by IIVGR2 for a sequential DASD or direct-access data set residing on a shared volume.
	IIVIS	Issued to open OS indexed sequential data sets.
	IIVIN2	Issued to open DOS residence file if it is shared.
	IIVRCP	Issued to open the SYSDEBUG data set.
	IIVABN	Issued to open a DCB with a ddname of SYSNAP.
	IIVRAS	Issued to open the SYSSNAP data set.
OPEN TYPE=J	IIVOPN	Issued to open sequential DASD and direct-access data sets (TYPE=J specifies that the Emulator's JFCBs are already in main storage).
	IIVIS	Issued to open an OS indexed sequential data set for initial loading of that data set.
POST	IIVRTE	Issued by the timer interrupt check subroutine to set a completion code in an ECB. The completion code indicates that a time interval of 1 second, established with an STIMER, has expired.
	IIVPCE	Issued to initiate subtask processing for IIVIS.

Figure 80 (Part 3 of 5). OS Macros Used by the Emulator

OS Macro	Emulator module issuing macro	Use of Macro
	IIVIS	Issued to signal the completion of subtask processing to the Emulator task.
PUT	IIVSTG	Issued on behalf of DOS output requests when staging has been indicated. The output operation is always to an OS DASD.
	IIVIS	Issued to store records in ascending order by key to an OS indexed sequential data set.
PUTX	IIVIS	Issued to rewrite sequentially retrieved records to an OS indexed sequential data set.
RDJFCB	IIVOPN, IIVIS, IIVINT	Issued so that these modules can examine and modify the JFCBs.
READ KU	IIVIS	Issued to retrieve records from an OS indexed sequential data set by key.
RETURN	IIVIS, IIVINT, IIVRAS, IIVSNP, IIVVIO, IIVMSG, IIVGR2, IIVABN, IIVSTG, IIVPUB, IIVIN2, IIVPRP	Issued to restore register contents and return to the calling routine.
SAVE	IIVENT, IIVINT, IIVIS, IIVSNP, IIVRCP, IIVRAS, IIVVIO, IIVMSG IIVSTG, IIVPRP, IIVPUB, IIVCCW, IIVIN2, IIVADD, IIVGET, IIVOPN, IIVABN	Issued to save contents of caller's registers.
SETL	IIVIS	Issued to start sequential retrieval.
SNAP	IIVABN	Issued if the Emulator run must be terminated and if the JCL statement //SYSSNAP DD SYSOUT=A was included for the Emulator job step. The contents of the entire DOS storage area of the Emulator region and the DOS registers are dumped.
	IIVSNP	Optionally issued to obtain a snap dump.
STAE	IIVRTE	Issued to enable the Emulator to retain control if an OS ABEND condition occurs because of an error in the DOS problem program so that DOS, not OS, can cancel the job.

Figure 80 (Part 4 of 5). OS Macros Used by the Emulator

OS Macro	Emulator module issuing macro	Use of Macro
STIMER	IIVRTE, IIVPUB	Issued by the timer interrupt check subroutine and OS PUB table build routine to establish a time interval of 1 second if the user has requested that the Emulator simulate the DOS timer. Upon expiration of this 1-second interval for the timer interrupt check subroutine, the OS control program passes control to the timer interrupt routine as soon as the Emulator becomes the highest priority task in the ready condition. Since the Emulator must compete with other tasks for control, the timer interrupt routine may not be entered immediately upon expiration of the 1-second interval. Also, it is possible that DOS external interruptions may be disabled when the interval expires. Consequently, the Emualator cannot simulate the DOS timer with strict accuracy.
TIME	IIVADD, IIVSNP, IIVRAS	Issued to get the present OS date and time of day.
WAIT	IIVIN2, IIVIS, IIVOPN, IIVPCE, IIIVRTE, IIVRCP	Issued when processing cannot continue until an interruption occurs.
WRITE	IIVSNP	Issued to dump Emulator control blocks.
	IIVRCP	Issued to write the SYSSNAP data set.
WRITE K, KN	IIVIS	Issued to update and add, respectively, records to an OS indexed sequential data set.
WTO, WTOR	IIVMSG, IIVLOG	Used by the Emulator to issue messages for DOS and by IIVMSG to output Emulator messages. A DOS message is reformatted and issued as a WTO or WTOR by IIVLOG.

Figure 80 (Part 5 of 5). OS Macros Used by the Emulator

Bypass Label Processing

See "Requesting Bypass Label Processing" in the "Introduction" for details.

Access Methods

For the indexed sequential access method (ISAM), module IIVIS is dependent on the location of the fields in the DCB, the DECB, and the JFCB.

Appendages

Appendage modules IGG019SA and IGG019SI are used by the Emulator (see also "Emulator Appendages" in the Introduction and "I/O Appendages" in the Method of Operation section). These modules perform the following operations:

IGG019SA

- Updates DASD extents to inhibit automatic cylinder switching procedures.
- Updates DASD extents to allow accessing of the user label track associated with a sequential DASD or direct-access data set on a shared volume.
- Updates DASD file mask to reflect DOS file mask if the I/O is for a dedicated volume.
- Updates tape set mode to reflect DOS set mode.
- Issues store CPU ID instruction (STIDP).

IGG019S1

- Suppresses OS error recovery for teleprocessing I/O.
- Suppresses OS error recovery for special tape operations (see IGG019S1 module description).

OS Control Blocks

The OS control blocks in Figure 81 are referred to by the Emulator. This directory briefly describes how each block is used. The blocks are further described in the "Data Areas" section.

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
DCB	DCBKEYLE	16	10	1	Contains the key length.
DCB	DCBEODAD	33	21	3	Contains the EODAD exit address.
DCB	DCBRECFM	36	24	1	Contains the record format.
DCB	DCBDDNAM	40	28	8	Contains the DD statement data set name.
DCB	DCBIFLGS	44	2C	1	Indicates a permanent error condition when bits 0 and 1 are on.
DCB	DCBOFLGS	48	30	1	Indicates a successful open when bit 3 is on.
DCB	DCBMACR	50	32	2	Contains the macro instruction operation code.
DCB	DCBOPTCD	52	34	1	Contains the option code.

Figure 81 (Part 1 of 5). OS Control Blocks

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
DCB	DCBMAC	53	35	1	Contains extension of the macro instruction operation code field.
DCB	DCBSYNAD	56	38	4	Contains the SYNAD exit address.
	or				
DCB	DCBSYNAD	57	39	3	Contains the staged I/O SYNAD exit address.
DCB	DCBRKP	60	3C	2	Contains the relative key position.
DCB	DCBBLKSI	62	3E	2	Contains the blocksize.
DCB	DCBEXCD1	80	50	1	Contains condition flags.
DCB	DCBEXCD2	81	51	1	Contains condition flags.
DCB	DCBLRECL	82	52	2	Contains the logical record length for variable length records.
DCB	DCBTDC	148	95	2	Contains the tag deletion count.
DCB	DCBRORG3	152	98	4	Contains the count of accesses tc overflow records other than the first.
DCB	DCBNREC	156	9C	4	Contains the number of logical records in the prime data area.
DCB	DCBLPDA	184	D8	8	Contains the direct-access address of the last prime data record in the prime data area.
DCB	DCBRORG2	206	CE	2	Contains the number of tracks lef in the overflow area.
DCB	DCBNOREC	214	D6	2	Contains the number of logical records in the overflow area.
DCB	DCBRORG1	224	E0	2	Contains the number of full cylinder overflow areas.
DEB	DEBEOEA	-36	-24	4	Contains the end-of-extent appendage address.
DEB	DEBSIOA	-32	-20	4	Contains the SIO appendage address.
DEB	DEBCEA	-24	-18	4	Contains the channel end appendage address.

Figure 81 (Part 2 of 5). OS Control Blocks

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
DEB	DEBXCEA	-20	-14	4	Contains the abnormal end appendage address.
DEB	DEBDVMOD	32	20	1	Contains the file mask.
DEB	DEBBINUM	36	24	2	The SIO appendage modifies fields to extend the extent to cover the entire DASD volume (for dedicated volumes only).
	DEBSTRCC	38	26	2	
	DEBSTRHH	40	28	2	
	DEBENDCC	42	2A	2	
	DEBENDHH	44	2C	2	
DEB	DEBNMTRK	46	2E	2	Contains the number of tracks allocated on the volume.
DECB	DECBDCBA	8	8	4	Contains the address of the DCB.
DECB	DECBAREA	12	C	4	Contains the address of the area in storage for the record.
DECB	DECBLOGR	16	10	4	Contains the address of the logical record (also used by IIVIS in QISAM mode to store the current logical record).
DECB	DECBKEY	20	14	4	Contains the address of the key portion of the record.
DECB	DECBEXC1	24	18	1	Contains the exceptional condition code.
DECB	DECBEXC2	25	19	1	Contains the exceptional condition code.
DSCB	DS1FMTID	44	2C	1	Contains the format identifier (X'F1' identifies this as (format 1) DSCB).
DSCB	DS1DSIND	93	5D	1	Contains data set indicators in bits 0, 2, 3, and 5.
DSCB	DS1PTRDS	135	87	5	Contains the pointer to an index (format 2) DSCB, if data set has IS organization, or pointer to an extension (format 3) DSCB if data set has sequential or direct-access organization and more than 3 extents.
DSCB VTOC (Format 4)		0	0	44	Contains padding bytes.
DSCB	DS4DSREC	50	32	2	Contains the number of available free VTOC record (format 0) DSCBs in the VTOC.

Figure 81 (Part 3 of 5). OS Control Blocks

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
DSCB	DS4VTOCI	58	3A	1	Contains VTOC indicators.
DSCB	DS4DEVDT	74	4A	1	Contains the number of full DSCBs that can be held on one track.
DSCB	DS4VTOCE	105	69	10	Contains extent description of the VTOC.
IOB	IOBFLAG1	0	0	1	Bits 0, 1, and 6 are set as required by EXCP (data chaining, command chaining, and related bit).
IOB	IOBSEN0	2	2	1	Sense byte 1 is tested or passed to DOS (see IIVCHK for details).
IOB	IOBSEN1	3	3	1	Sense byte 2 is tested or passed to DOS (see IIVCHK for details).
IOB	IOBECBPT	5	5	3	Contains the address of the ECB.
IOB	IOBCSW	9	9	7	When I/O interruptions are simulated, the 7 low-order bytes of the last CSW are passed to DOS.
IOB	IOBSTART	17	11	3	Initialized with the first CCW for EXCP.
IOB	IOBDCBPT	21	15	3	Initialized with the DCB address for EXCP.
IOB	IOBSEEKM	32	20	1	M = the number of DEB extents as required by EXCP.
IOB	IOBSEEK	33	21	7	BBCCHHR = the seek address for a direct-access volume.
JFCB	JFCBDSNM	0	0	44	This field is modified so that the VTOC can be opened. The tenth character is examined to determine SYSIN and SYSOUT files for staged devices.
JFCB	JFCBTSDM	52	34	1	SYSIN or SYSOUT file is assumed if X'20' is set in this byte.
JFCB	JFCBLTYP	66	42	1	Set to X'10' to assure BLP (bypass label processing) option.

Figure 81 (Part 4 of 5). OS Control Blocks

Control Block Name	Field Name	Displacement Decimal	Hex	Bytes	Description of Use
JFCB	JFCBLKSI	102	66	2	Contains a default block-size for staged I/O if not specified in the DD statement (89 bytes for punch, 141 for print).
JFCB	JFCBVOLS	118	76	6	The Emulator open routine moves in the volume serial number or identification (VOLID) for mount requests. Used to locate the TIOT by means of EXTRACT.
TIOT	TIOELNGH	0	0	1	Contains the length of each DD entry. Used to scan the TIOT during initialization.
TIOT	TIOEFSRT	1	1	3	Contains the UCB address of the device allocated.
TIOT	TIOEDDNM	4	4	8	Contains an 8-byte DDname used to determine DOS units.
UCB	UCBTYP16		10	4	Used to identify device types for cross-referencing with DOS device types. The SIO appendage modifies the DEB based on device type. The abnormal end appendage determines whether OS or DOS is to handle I/O errors, based on the device type.
UCB	DCELBBNR	56	38	3	Bin number for a data cell drive.

Figure 81 (Part 5 of 5). OS Control Blocks

Dependence on Hardware

The Emulator depends upon hardware for information as follows:

- PSW - The system mask field in the program status word is used to determine whether DOS is enabling interruptions.

  The WP bits (14-15) are tested by the Emulator to determine wait or problem state.

  The interruption code bits (16-31) are used by the Emulator to store the channel and device address when simulating an I/O interruption.

  The condition code bits (34-35) are modified by the Emulator.

- CAW - The key and address fields in the DOS channel address word are accessed by the Emulator.

- CSW - All fields in the channel status word are used or tested by the Emulator.

The formats of the above words can be found in the "Data Areas" section.

EMULATOR SERVICE AIDS

Some of the problems that might cause a DOS program to be canceled may be due to violations of the Emulator restrictions such as a CCW modified by DOS between the issuance of an EXCP and WAIT macros. The service aids may help in confirming the suspected cause of the problem by providing a dump at the time the violation occurred.

The Emulator contains certain modules (IIVRAS, IIVRCP, IIVSNP, IIVPCI, IIVSCI, IIVACI, IIVRCW), referred to herein as the service aids, that enable the installation systems programmer or the IBM programming systems representative to localize problems that might occur during emulation.

The service aids enable the Emulator to:

- issue a dump of all or part of the Emulator region when specific events occur
- trace specific events and record them in a variable size trace table
- give control to user exit routines when specific events occur

An additional 14K bytes of storage plus BSAM storage requirements are needed in order to utilize the service aids. BSAM storage requirements can be computed by referring to Storage Estimates for OS or OS/VS. The DCB parameters used for the output data set are DSORG=PS, MACRF=W, RECFM=VBA. Buffering is simple, scheduling is normal, and output goes to a DASD without record overflow.

Before using the service aids, a review of "Hints for Debugging" and Emulator limitations and restrictions in the "Introduction" may help localize a problem. It would also be helpful to review the "Method of Operation" section of this publication. Note that the control blocks that appear in the formatted snap dumps can all be found in the "Data Areas" section.

Use of the Emulator Service Aids

The Emulator service aids can be invoked in one (or both) of two ways:

- by replying DEBUG to an Emulator prompt message on the system console

  A reply of DEBUG to an Emulator prompt message causes the following message to be issued:

  IIV270A jobname ENTER OR CONTINUE DEBUG STATEMENT

  The debug statement may then be entered on the console and when completed (as indicated by the END command), the Emulator prompt message is reissued and emulation resumes.

- by including a //SYSDEBUG DD * card with the Emulator job step

  A debug statement that is entered through the OS input stream may span 1 or more cards and must be immediately preceded by a //SYSDEBUG DD * card and followed by a /* card.

A SYSSNAP DD statement, such as //SYSSNAP DD SYSOUT=A, must be included to define an output data set. This data set will contain:

- the debug statement(s) entered during emulation and error messages associated with it (if any)
- the formatted snap dumps requested
- the trace table (if specified) containing the events recorded before the Emulator's termination

If both snap dumps and a trace table are specified, a trace table containing the events recorded at the time the snap is taken will be printed with every snap dump.

The Debug Statement

A debug statement can be written in free form, that is, it need not begin in a particular column and it may contain any number of blank characters between elements and/or delimiters (commas, parentheses, equal signs) of the statement. Debug statements can contain commands, parameters, keyword parameters, and keyword subparameters. Each keyword parameter or keyword subparameter must be followed by an equal sign and its corresponding value or values.

There are six commands: SNAP, TRACE, STORAGE, EXIT, DIAG, and END. Each command, with the exception of END, must contain at least one parameter or keyword parameter. The commands, parameters, or keyword parameters may appear in any order in the debug statement, but a keyword subparameter must follow its associated value. If more than one parameter or keyword parameter is entered in a debug statement, the parameters must be enclosed in parentheses. A debug statement is composed of:

```
debug statement = [[ ][ command1[ ,command2,...) ], ]END

                  parameter1                    parameter2
command = [ (]      or                  [,        or                  ,...) ]
                  keyword parameter1            keyword parameter2

keyword parameter=[ (]value[ ,keyword subparameter1 [ ,keyword subparameter2,
                    ... ]][ ,... ) ]

keyword subparameter = [ (]value1[ ,value2,... ) ]
```

Whenever a keyword parameter is repeated within a single debug statement, only the last keyword parameter specified will be in effect. For example,

```
SNAP = (SVC =2, SVC =4), END
```

causes a SNAP to be taken only when DOS issues an SVC 4 since the first keyword parameter (SVC=2) is overridden by the second keyword parameter. Each time a command is repeated in a statement (with the exception of END) the first command is overriden by the second. A SNAP would be taken for each I/O operation on cuu=132, but not for each SVC 2 in the following example:

```
SNAP = (SVC =2), SNAP = (IO,CUU= 132), END
```

Figure 82 shows how to code a debug statement.

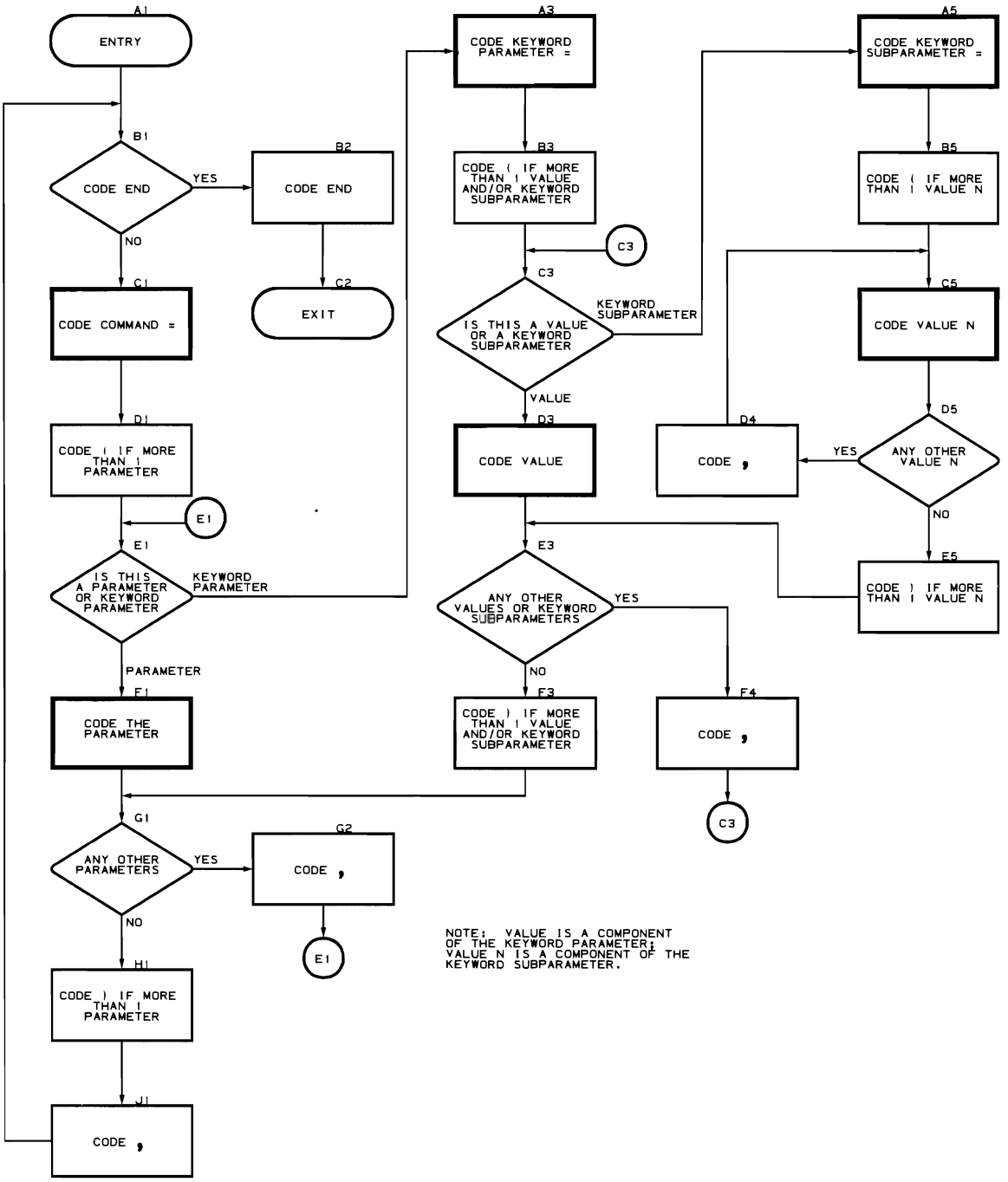


Figure 82. How to Code a Debug Statement

Some points to consider when coding a debug statement are:

- The debug statement may be contained in more than one card or console reply. In this case:

  1. The statement can only be interrupted after a comma. Statements can only be interrupted when a comma follows a parameter or keyword parameter and its corresponding value, or a keyword subparameter and its corresponding value.

  2. If the service aids are invoked by including a //SYSDEBUG DD * card, the next record of the data set defined by the DD statement is read and processed.

  3. If the service aids are invoked by replying DEBUG to an Emulator prompt, the message

     IIV270A jobname ENTER OR CONTINUE DEBUG STATEMENT

     is issued and the reply is read and processed.

- The end of a debug statement is indicated by the END command. When an END command is encountered following other debug commands, functions defined by the other debug commands, such as SNAP and TRACE, are activated. When an END command is encountered and is the only command in the DEBUG statement, all the existing functions, if any, are inactivated.

- The options specified in the debug statement remain in effect until a new debug statement is entered through the system console or until the end of the Emulator job.

- Whenever a PHASE keyword subparameter is identified, all values encountered are processed; thus,

  SNAP = (SVC =4, PHASE=A, PHASE= (B,C)),END

  will cause a SNAP to be taken when DOS issues an SVC 4 to load A, B, or C.

- If an error occurred when entering the debug statement on the console, the following message will be issued

  IIV271A jobname DEBUG COMMAND ERROR AT POSITION xxx-RESPECIFY

  The debug statement may be reentered with the command in error or the entire statement may be respecified.

- If any errors are detected when input is from the data set defined by the SYSDEBUG DD statement, the message

  IIV274I jobname

  is issued and emulation is terminated.

- If a syntax error is detected during the processing of a debug statement from the console, all functions, if any, are inactivated only for the command being processed. For example, if

  TRACE=ALL, SNAP=(SVC=2,END

  is issued from the console, an error message

  IIV271A jobname

  is issued (END in the example is interpreted as an incorrect parameter for the command SNAP since there is no closing parenthesis before the second comma). Only the command in error (SNAP in this example) and any subsequent commands (END in this example) must be correctly respecified to be processed. In the above example, a reply

  SNAP = (SVC =2), END

  will cause a TRACE to be taken and a SNAP to be issued when DOS issues an SVC 2. If only END were respecified, only the TRACE command would be activated.

TRACE Command

The format for a TRACE command is

```
            parameter1                            parameter2
TRACE = [ ( ]or                          [,or                              ,...) ]
            keyword parameter1=value1             keyword parameter2=value2
```

See also "Trace Table" for further information concerning the function of this command.

The following are valid parameters or keyword parameters of TRACE. Only the keyword parameters NUMBER and CUU have values.

ALL — When ALL is specified, all of the events caused by the following parameters or keyword parameters are recorded in the trace table as they occur.

ATTN — If the attention bit is set in the CSW, the cuu, the PSW at the time of interruption and the CSW are recorded in the trace table. If the unit check bit is set in the CSW, the two IOB sense bytes are also recorded. This parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

  ATTN cuu IPSW hhhhhhhh hhhhhhhh CSW hhhh hhhh SNS hhhh

CUU = — [( ] cuu [,cuu,...)]
Events associated with I/O operations will only be recorded in the trace table if the operation is to the specified channel and unit. These events include SIO, TIO, TCH, HIO, and I/O interruptions.

EXT — When an interruption is external or the DOS interval timer occurs, the PSW at the time of interruption is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

  EXT IPSW hhhh0040 hhhhhhhh
  TMR IPSW hhhh0080 hhhhhhhh

HIO When a halt I/O instruction is issued by DOS, the cuu, the resume PSW, and the CSW are recorded in the trace table. In addition, if the unit check bit is set in the CSW, the two IOB sense bytes are recorded. This parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

HIO cuu RPSW hhhhhhhh hhhhhhhh CSW hhhhhhhh hhhhhhhh SNS hhhh

INT When INT is specified, I/O interruptions, external interruptions, and timer interruptions are recorded in the trace table. See also the parameters IO, EXT, and TIMER for further information.

IO All interruptions associated with I/O operations are recorded in the trace table. These include SIO, TIO, TCH, and HIO. In addition, I/O interruptions cause the PSW at interruption, the channel and unit address, and the CSW to be recorded in the trace table. If the unit check bit is set in the CSW, the two IOB sense bytes are also recorded. I/O interruptions are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

I/O cuu IPSW hhhhhhhh hhhhhhhh CSW hhhhhhhh hhhhhhhh SNS hhhh

ISK When an insert storage key instruction is issued by DOS, the PSW at interruption, the storage address of DOS, and the storage protection key are recorded in the trace table. This parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

ISK IPSW hhhhhhhh hhhhhhhh ADDR 11111111 KEY h

LPSW When a load PSW instruction is issued by DOS, the PSW at interruption and the resume PSW are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

LPSW IPSW hhhhhhhh hhhhhhhh RPSW hhhhhhhh hhhhhhhh

NUMBER = {50}
{nnnnn}

This keyword parameter is used to redefine the default trace table size. If NUMBER is not specified, the default value is 50. The maximum number of entries is 65,534. As each entry is 36 bytes in length, any significant increase in the table size should be reflected in the job step region parameter.

SIO When a start I/O instruction is encountered, the information recorded in the trace table depends on the condition code and the type of device. If the device type is direct access, the DASD seek address will be recorded. If CC=01 (CSW stored) and the unit check bit is not set in the CSW, the cuu, the RPSW, the CSW, and the CCW addressed by the DOS CAW (for staged or SYSLOG devices, or if the operation is a stand-alone sense or seek) are recorded in the trace table. If, however, the device is not staged or SYSLOG, or if the operation is not a stand-alone sense or seek, the CCW address is obtained from the COMCAW field of the COMTAB entry for that device. This parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

SIO cuu SEEK hhhhhhhh hhhhhhhh RPSW hhhhhhhh hhhhhhhh

CAW hhhhhhhh CCW1 hhhhhhhh hhhhhhhh

If CC=01 and the unit check bit is set in the CSW, the cuu, RPSW, CSW, the operation code of the CCW referred to above, and the first two sense bytes obtained from the IOB are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
SIO cuu RPSW hhhhhhhh hhhhhhhh CSW hhhhhhhh hhhhhhhh

     SNS hhhh OP hh
```

If CC≠01, the cuu, RPSW, CAW, and either the first CCW or the CCW pointed to by the COMCAW field in COMTAB (depending on the same device types and operations noted previously) are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
SIO cuu RPSW hhhhhhhh hhhhhhhh CSW hhhhhhhh hhhhhhhh
     CCW1 hhhhhhhh hhhhhhhh
```

SSK When a set storage key instruction is issued by DOS, the PSW at interruption, the storage address, and the storage protection key are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
SSK IPSW hhhhhhhh hhhhhhhh ADDR 11111111 KEY h
```

SSM When a set system mask instruction is issued by DOS, the PSW at interruption and the resume PSW are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
SSM IPSW hhhhhhhh hhhhhhhh RPSW hhhhhhhh hhhhhhhh
```

SVC When a supervisor call is issued by DOS, the PSW at interruption and the contents of DOS general purpose registers 0 and 1 are recorded in the trace table. In addition, if the interruption is not taken (in other words, if the function of the DOS SVC routine is performed by the Emulator), the resume PSW is also recorded in the trace table. If the SVC is 2 (DOS fetch), the name of the DOS B-transient phase is recorded. If the SVC is 4 (DOS load), the name of the DOS load module is recorded. This keyword parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
SVC nnn IPSW hhhhhhhh hhhhhhhh RPSW hhhhhhhh hhhhhhhh BTR cccccccc

     R0 hhhhhhhh R1 hhhhhhhh

SVC nnn IPSW hhhhhhhh hhhhhhhh LDMD cccccccc R0hhhhhhhh

     R1 hhhhhhhh
```

```
SVC = [ (] n {[,n,...) ]}
```

n This value of the SVC keyword parameter is the supervisor call number for which entries will be made in the trace table. The value n represents a decimal number from 0 to 255.

TCH When a test channel instruction is issued by DOS, the cuu and the resume PSW are recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
TCH cuu RPSW hhhhhhhh hhhhhhhh
```

TIMER When an interruption due to the emulated DOS interval timer occurs, the PSW at interruption is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

```
TMR IPSW hhhhhhhh hhhhhhhh
```

TIO When a test I/O instruction is issued by DOS, the cuu, the resume PSW, and the CSW are recorded in the trace table. In addition, if the unit check bit is set in the CSW, the two IOB sense bytes are recorded. This parameter is recorded in the trace table (see Figure 86 (Part 6 of 7). Service Aids Snap Dump) as:

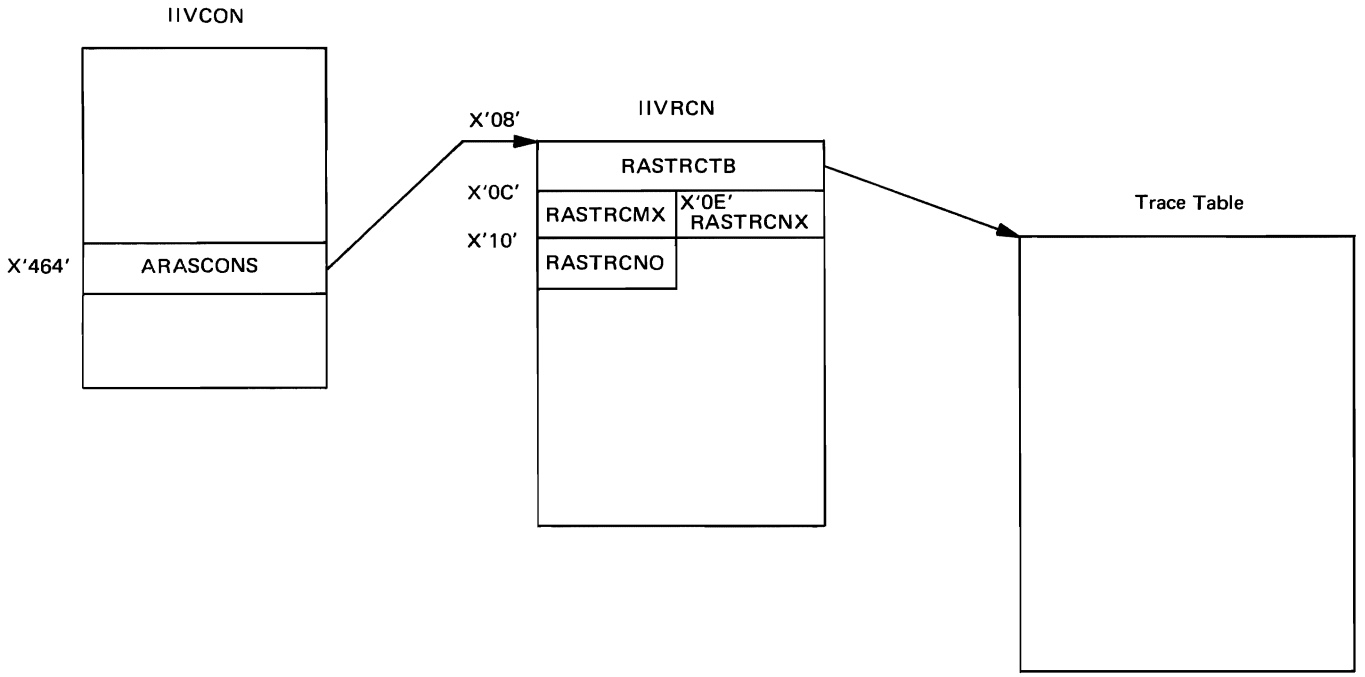
TIO cuu RPSW hhhhhhhh hhhhhhhh CSW hhhhhhhh hhhhhhhh SNS hhhh

WRAP
NOWRAP WRAP is the default parameter of the TRACE command and will cause old entries in the trace table to be overlaid by new entries when the trace table is full. NOWRAP causes the trace table to be snapped every time it is full.

Trace Table. The trace table is an optional feature specified by the command TRACE. The events to be traced are specified in parameters in the TRACE command. When the table is filled, old entries are overlaid with new entries, starting at the beginning address of the trace table.

The trace table is printed along with the Emulator-formatted snap dumps as they occur. All trace table pointers are reset immediately after the TRACE table is printed. This ensures that no two formatted traces reflect the same events. If no snap dumps are taken, the trace table will be printed only once at Emulator end-of-job time.

Unusual circumstances may cause abnormal termination of the Emulator region before the trace table is printed. Refer to Figures 83 and 84 for the procedure to locate and interpret an unformatted trace table.



Notes:
The address of the Emulator service aids constants area (IIVRCN) is located in a 4-byte field at X'464' in IIVCON
The address of the Emulator trace table is located in a 4-byte field at X'08' in IIVRCN.
The number of active trace entries is located in a 2-byte field at X'0C' in IIVRCN.
The address of the next available entry in the trace table is located by multiplying 36 (trace table entry length) times the value contained in the 2-byte field at X'0E' in IIVRCN and adding the result to the beginning address of the trace table.
The size of the trace table in 36-byte entries is located in a 2-byte field at X'10' in IIVRCN.
The most recent entry in the trace table can be located by subtracting 36 from the address of the next available entry.
Successive subtractions yield addresses of former entries.

Figure 83. How to Locate the Trace Table

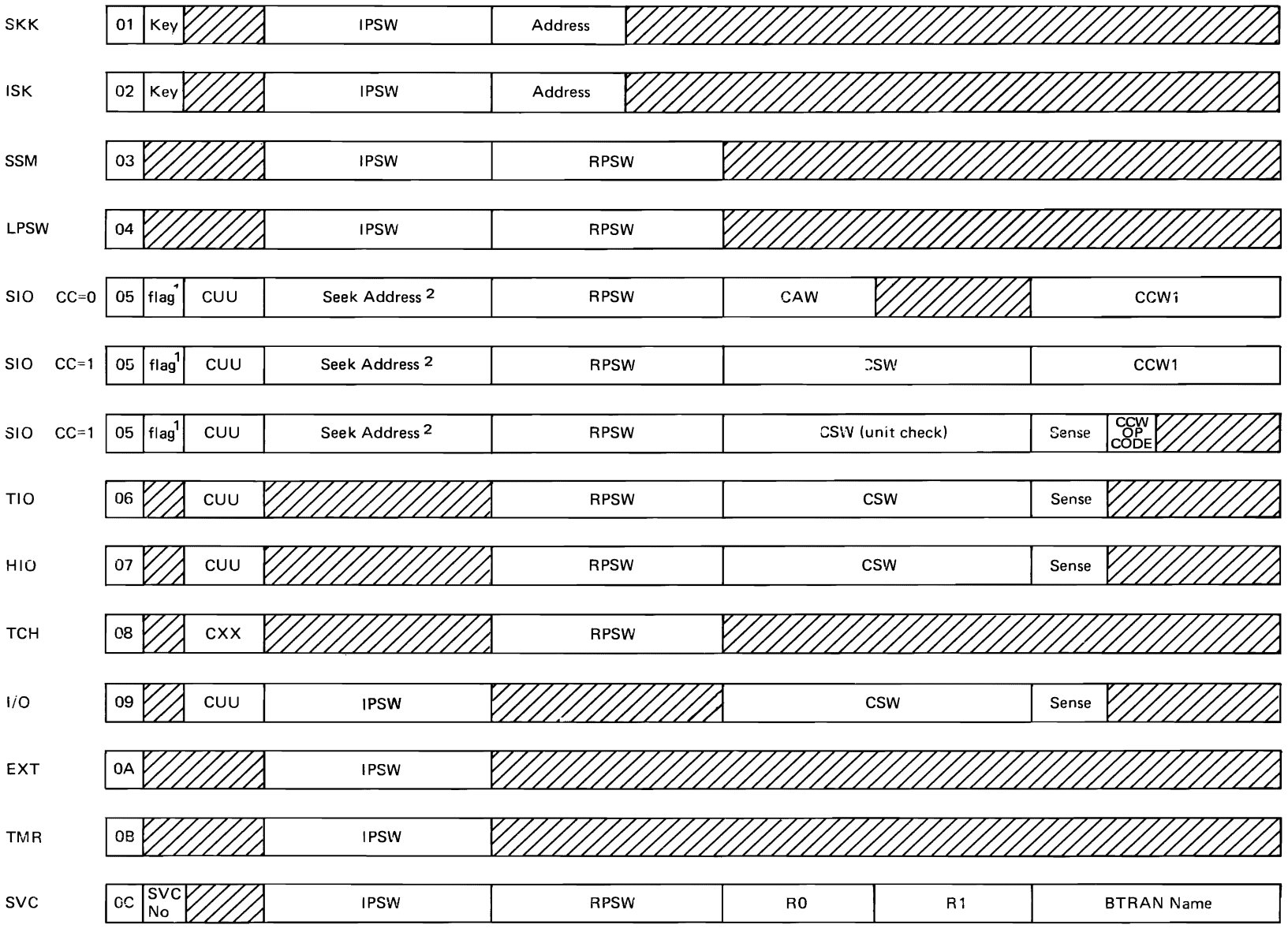


Figure 84. Internal Trace Table Format

[1]Flag — X'01' indicates a DASD; the other 7 bits in this byte are reserved.

[2]The seek address refers to MBBCCHHR (for DASD only).

The format for a SNAP command is

```
                parameter1
SNAP = [ (]       or
                keyword parameter1=value1[,keyword subparameter1=value2
                                                        ,...) ]

                [,keyword subparameter2=value3,...]
```

The following parameters, keyword parameters, keyword subparameters, and corresponding values of the SNAP command define the conditions under which an Emulator dump can be taken.

ALL This parameter causes an Emulator snap dump to be taken, as described for all other parameters or keyword parameters.

ATTN This parameter causes an Emulator snap dump to be taken whenever an I/O interruption is simulated to DOS and the attention bit is set in the DOS CSW.

```
                {A111111  =}
COMP = [ (]     {Rnn      =} {h}[ {h} {h} {h} {h} {h} {h} {h} ]  [,...) ]
                {CRnn     =} {*}[ {*} {*} {*} {*} {*} {*} {*} ]
```

1 This digit represents a DOS local address in main storage whose contents will be compared with the corresponding data as shown by values replacing h. Leading zeros are not required for the address.

Rnn This value represents a DOS general purpose register that contains an address in main storage. The main storage addressed by the indicated register will be compared with the corresponding data.

CRnn This value indicates that the contents of the indicated DOS general purpose register is to be compared with the corresponding data.

nn This value can be one or two digits representing any decimal value from 0-15.

* The * indicates that the corresponding four bits in the same position are not to be examined.

h The h represents the data to be compared and can be any valid hexadecimal digit from 0-F. The data can be any valid combination of hexadecimal digits or *'s, up to a total of 8 digits (4 bytes).

The effective length of the compare field for A111111 and Rnn will only be as long as the number of digits provided. In the case of CRnn, leading zeros are padded to the left.

Note: If the debug statement

```
SNAP = COMP = A0020 = 0E40, END
```

were coded, an Emulator snap dump would be taken if DOS storage locations 20 and 21 contained the hexadecimal values of 0E and 40, respectively. If, however, this condition is met while DOS is in local execution mode and the values in storage locations 20 and 21 are then changed prior to a hardware interruption, the condition will never be detected by the Emulator and thus no snap dump will be taken.

CUU = [(] cuu [,cuu,...)]
This keyword parameter causes an Emulator snap dump to be taken only for the channel and unit specified for I/O related instructions and interruptions.

EXT This parameter causes an Emulator snap dump to be taken whenever an external interruption is simulated to DOS.

HIO* This parameter causes an Emulator snap dump to be taken whenever DOS issues a halt I/O instruction.

INT This parameter causes an Emulator snap dump to be taken for I/O, external, and timer (asynchronous) interruptions.

IO This parameter causes an Emulator snap dump to be taken whenever DOS issues the SIO, TIO, HIO, or TCH instructions. In addition, a snap dump will be taken for all I/O interruptions.

ISK* This parameter causes an Emulator snap dump to be taken whenever DOS issues an insert storage key instruction.

LPSW* This parameter causes an Emulator snap dump to be taken whenever DOS issues a load PSW instruction.

PC = {NOIPL}
{ALL}

This keyword parameter causes an Emulator snap dump to be taken whenever a nonprivileged operation program check occurs. If the problem state bit was set in the local execution PSW when the privileged operation occurred, a snap dump will also be taken. If PC without operands or PC=NOIPL is specified, program checks during DOS IPL are ignored. If PC=ALL is specified, program checks will also be snapped during DOS IPL. The SPACE parameter should then be included in the SYSSNAP DD statement since there is a fairly large volume of output.

PSW = {h}[{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}{h}]
{*}[{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}{*}]

* The * indicates that the corresponding four bits in the same position of the local execution PSW are not to be examined.

h The h can be any valid hexadecimal digit from 0-F.

Digits and *'s can be placed in any order. If fewer than 16 characters are specified, *'s are padded to the right up to a total of 16. In other words, the remaining digits in the local execution PSW are not examined.

The local execution PSW is examined before passing control to DOS. An Emulator snap dump will be taken if there is an equal compare between all the indicated fields to be examined (as specified by valid digits in place of h) and those in the local execution PSW.

Note: If all the characters are *'s, a snap dump will be taken every time control is passed to DOS.

SIO* This parameter causes an Emulator snap dump to be taken whenever DOS issues a start I/O instruction.

SSK* This parameter causes an Emulator snap dump to be taken whenever DOS issues a set storage key instruction.

SSM* This parameter causes an Emulator snap dump to be taken whenever DOS issues a set system mask instruction.

```
SVC =    [(] n      {[,n,...)]}
                    {[,n,...,PHASE =[(] modname1 [,modname2,...)])[,...]}
```

n — This value is the supervisor call number for which snaps are to be taken.

PHASE — This keyword subparameter indicates that snaps are taken only if specific DOS B-transient phases and load modules are being fetched or loaded. PHASE is valid only if the SVC number is 2 or 4 and immediately precedes PHASE.

modname — This value may be any combination of alphameric characters or $, up to a total of 8 characters. If less than 8 characters are specified, the name will be padded right with blanks to 8 characters.

An Emulator snap dump is taken if the indicated SVC interruption occurs. If the PHASE keyword subparameter was specified, snaps are taken only if DOS register 1 points to one of the indicated module names at the time the indicated SVC interruption occurs.

TCH* — This parameter causes an Emulator snap dump to be taken whenever DOS issues a test channel instruction.

TIMER — This parameter causes an Emulator snap dump to be taken whenever an interval timer interruption is simulated to DOS.

TIO* — This parameter causes an Emulator snap dump to be taken whenever DOS issues a test I/O instruction.

* Another dump is issued after the Emulator has simulated the instruction.

STORAGE Command

The format for a STORAGE command is

```
              parameter1                [,parameter2,...)]
STORAGE = [(]   or
              keyword parameter=value
```

The following parameters and keyword parameter of the STORAGE command define the main storage to be dumped whenever a snap dump is taken. The parameters may be specified in any order.

ALL — This parameter causes the OS SNAP macro to be issued (SDATA=ALL) in addition to the formatted dump of the Emulator control blocks, which is printed whenever a snap dump is taken.

DOS — This parameter causes the DOS registers and entire DOS main-storage area to be dumped whenever a snap dump is taken. (DOS is the default value if STORAGE is not specified.)

EMBLKS — This parameter causes a formatted dump of the Emulator control blocks to be printed whenever a snap dump is taken. (EMBLKS is the default value if STORAGE is not specified.)

EMU — This parameter causes the OS SNAP macro to be issued (PDATA=ALL) in addition to the formatted dump mentioned previously.

NODOS — This parameter causes the DOS registers and only the DOS permanent storage area to be dumped whenever a snap dump is taken.

NUMBER = 50
nnnnn

This keyword parameter defines the number of snaps to be taken (maximum number is 65,534). Snap dumps are bypassed when this value is reached. If NUMBER is not specified, the default value is 50.

EXIT Command

The format for an EXIT command is

EXIT = [( ] keyword parameter1=value1[,keyword parameter2=value2,...)]

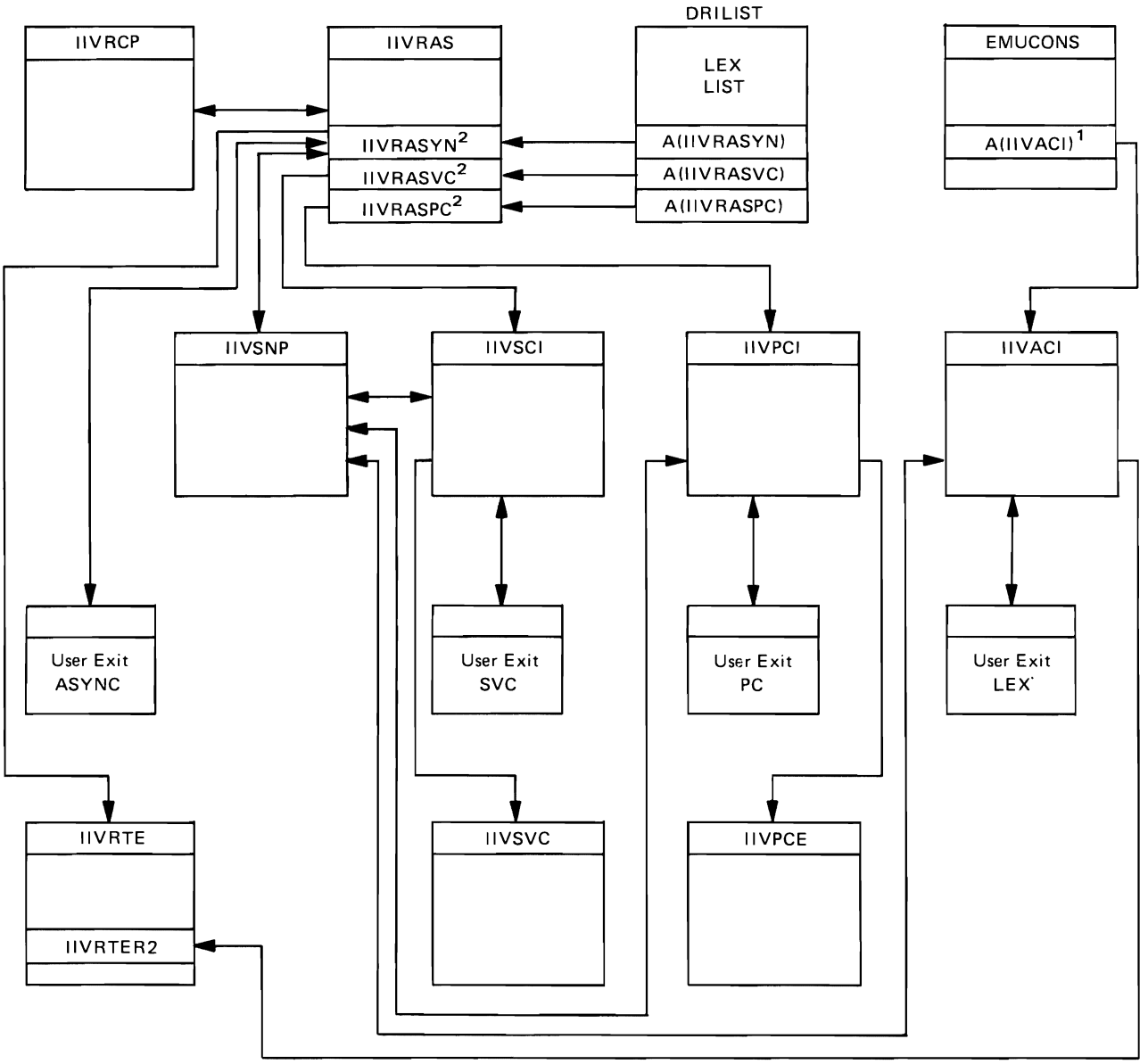
The EXIT command allows user-written routines to be given control at various points during the emulation process.

Before the user routine can be executed, it must reside on the system program library (SYS1.LINKLIB) or on a private library. The full member name of the routine is 8 bytes in length, but the first 6 bytes are required to be the characters IIVRAS. The last 2 characters must be provided by the user as an identification (nn) and can be any two numeric characters.

User-written exit routines can provide additional control when solving emulation problems. Emulator control blocks can be examined, as well as individual bit settings anywhere in the Emulator region.

A user exit routine may receive control under the conditions described in the following keyword parameters and corresponding values.

Refer to Figure 85 for overall control flow.



[1]The label for this address constant is ART20. This address constant is the return point for all Emulator modules.

[2]These entry points are entered after the associated hardware interruptions have occurred.

Figure 85. Control Flow of the Emulator Service Aids

AS = nn — An asynchronous interruption occurred while the CPU was in local execution mode. Control is passed to the asynchronous user exit routine before passing control to the Emulator (entry point IIVRTE). The nn is a 2-byte identification of an asynchronous user exit routine that has been entered on the system program library (SYS1.LINKLIB) or on a private library.

LEX = nn — Control is to be returned to DOS after a program check, supervisor call, or asynchronous interruption occurs. Control is passed to the local execution user exit routine before passing control to the Emulator (entry point IIVRTER2). The nn is a 2-byte identification of a local execution user exit routine that has been entered on the system program library (SYS1.LINKLIB) or on a private library.

PC = nn — A DOS program check interruption occurred. This condition includes privileged operations, which will subsequently be handled by the Emulator. Control is passed to the program check user exit routine before passing control to the Emulator (entry point IIVPCE). The nn is a 2-byte identification of a program check user exit routine that has been entered on the system program library (SYS1.LINKLIB) or on a private library.

SVC = nn — A DOS supervisor call interruption occurred. Control is passed to the supervisor call user exit routine before passing control to the Emulator (entry point IIVSVC). The nn is a 2-byte identification of a supervisor call user exit routine that has been entered on the system program library (SYS1.LINKLIB) or on a private library.

At entry to an exit routine, the contents of the registers are as follows:

- Register 9 contains the address of the local execution list.
- Register 10 contains the true address of the DOS main storage (adjustment factor).
- Register 11 contains the address of IIVCON (EMUCONS).
- Register 14 contains the return address. An optional snap dump will be taken if the user exit routine returns to the address in register 14 plus a displacement of 4. A zero displacement will cause a normal return.
- Register 15 contains the address of the exit routine entry point.

Standard linkage must be used. It is the responsibility of the user to ensure all registers are saved and restored.

Note: The ddname for user data sets must start with the three characters SYS. Any other combination of characters will be interpreted as an erroneous volume-sharing request.

DIAG Command

The format for a DIAG command is

```
DIAG = CCWCHK [=[ (]cuu[,...)]]
```

One of the main restrictions of the Emulator is in running DOS programs which modify CCWs or use CCW data addresses between an I/O request and its completion. The result is unpredictable and could cause a loop in DOS as well as a DOS program check or an Emulator program check. The restriction is required because

the Emulator adjusts the DOS CCWs in place (in DOS main storage) and issues an OS EXCP against these CCWs. These CCWs contain OS (true) addresses from the time the OS EXCP is issued and the time OS posts I/O completion in the ECB.

The DIAG feature of version 2, level 1 of the Emulator allows removal of the restriction on programs using CCW data addresses (read only operations) during I/O requests. In addition, the DIAG command allows identification of DOS programs that modify the CCWs.

Rather than adjusting the DOS CCWs in DOS main storage, a special module (IIVRCW) is loaded when the DIAG feature is requested. IIVRCW will copy the DOS CCWs in a special save area, adjust them to their OS addresses in this area (called the DIAG block) and allows the Emulator to issue I/O requests from this area only.

The DOS CCWs in DOS main storage will remain unchanged between the I/O request and its completion. When IIVRCW readjusts the CCWs in the DIAG block back to DOS (local) addresses, a check will be made to determine whether the DOS CCWs have been modified by the DOS program from the time of the DOS SIO and the completion of the I/O as indicated by OS. If the CCWs have been modified, a message (IIV280I) will be issued to indicate which CCW has been modified. A channel program check is returned to the DOS program.

How to Request the DIAG Feature

CCWCHK [=[(]cuu[,...)]]

The cuu's specified indicate which DOS unit address will have the feature attached. If no cuu is specified, all the DOS unit addresses defined during the Emulation run (specified on SYSEMcuu statements) will have the DIAG feature except for staged devices and log devices.

DIAG limitations: When the DIAG command is being processed by IIVRCP, a DIAG block is created for each cuu specified. The total length of the block is 112 bytes (104 of these bytes will be used to save the DOS channel program). If a DOS channel program issued on a DOS unit with the DIAG feature contains more than 13 CCWs, message IIV281I is issued by module IIVRCW and the DIAG feature is reset for the DOS cuu involved.

The Emulator region should contain enough space for the DIAG blocks and module IIVRCW plus the 11K required for the service aids function.

END Command

When coded with other debug commands, END indicates termination of the debug statement. When coded as the only command in a debug statement, END will inactivate all previous functions, if any.

Examples of Valid and Invalid Debug Statements

The following are valid debug statements using one card/console reply:

- TRACE =•(EXT, NUMBER = (120)), END

  (At Emulator end of job, a trace table of 120 entries will be written to the SYSSNAP data set. The trace table will contain DOS external interruption entries, if they occur.)

- SNAP = COMP = A0020 = 0E40, END

  (An Emulator snap dump will be taken if DOS storage locations 20 and 21 contain the hexadecimal values of 0E and 40, respectively. See also the note in the explanation of the keyword parameter COMP under the SNAP command.)

- SNAP = (SVC = ((2, 3, 4,PHASE= ($$BOPEN,$$BCLOSE)))),END

  (An Emulator snap dump will be taken whenever DOS executes an SVC 2 or SVC 3. In addition, a snap dump will be taken by the Emulator whenever DOS executes an SVC 4 and DOS general purpose register 1 points to either the phase name $$BOPEN or $$BCLOSE.)

The following are valid debug statements using multiple card/console replies:

- TRACE = (SIO,
  TIO),

- SNAP = SVC = (1,
  2, END

  (DOS start I/O and test I/O instructions will be recorded in the Emulator trace table as they occur. Since the keyword parameter NUMBER was not included, a default value of 50 entries will be used.)

  (An Emulator snap will be issued whenever an SVC 1 or SVC 2 instruction is executed by DOS. If any start I/O or test I/O instructions were issued by DOS before the snap, the trace table will be included as part of the snap dump. If there are any entries in the trace table between the time the last snap dump was taken and Emulator end of job, the trace table will be snapped again at Emulator end of job.)

The following are invalid debug statements using one card/console reply

- TRACE = SIO, TIO, END

  (SIO, TIO are not enclosed in parentheses.)

- SNAP = (SVC = (2), END
  (Unbalanced parentheses.)

- SN AP = ALL, END

  (Blank embedded in SNAP.)

The following are invalid DEBUG statement using multiple card/console replies:

- SNAP = (ALL, SVC = (,
  2, 3)), END

  (Statements can only be interrupted when a comma follows a parameter, keyword parameter and its corresponding value, or a keyword subparameter and its corresponding value.)

Examples of Service Aids Usage to Diagnose Problems

Problem 1

Symptom:

DOS program canceled due to a DOS program check or channel program check and a DOS storage dump is produced.

Assumptions:

The following assumptions are made in this example:

1. The program check occurs only under emulation.

2. The DOS Emulator restrictions have been reviewed for possible violation by the user program.

Probable cause:

With the symptoms and assumptions listed above, the following causes are suspected;

1. A DOS module is using the data address in a CCW that is currently active (for example a WAIT macro not issued to ensure I/O completion). This type of violation usually results in a variety of program checks, depending on the data address and its usage. In any case, it is desirable to obtain dumps and any other helpful diagnostic information at the time the symptom occurs.

Code the following statement to obtain information that may help identify the cause of this problem.

```
SNAP=(PC,COMP=R14=35DE4),END
```

Explanation:

Specifying PC tells the Emulator to produce the desired dumps whenever a nonprivileged operation program check occurs. Specifying COMP=CR14=35DE4 tells the Emulator to produce the desired snap dumps whenever register 14 contains the value 00035DE4. The register and the value are arbitrary, but in this example the value may represent an adjusted data address from a CCW.

If the suspected cause was responsible for the problem, the dumps produced should help reveal the DOS modules involved when the problem occurred and consequently help locate the code violating the restriction.

2. A DOS module is modifying a CCW chain that is currently active. The modification involves the data address or the chaining bits. This violation usually results in a channel program check from the channel or from an Emulator simulation during CCW adjustment.

By using the service aids, diagnostic information may be obtained before the channel program check is passed to DOS. Since the cause of the problem may be that the channel program is being modified, the only areas of interest are the Emulator control blocks (such as relative I/O blocks) and DOS storage. Therefore, the default STORAGE specifications will be used again and the following SNAP command will be used:

```
SNAP=COMP=A45=20,END
```

COMP=A45=20 tells the service aids to produce the required snap dumps when the contents of DOS location X'45' is equal to 20. Location X'45' is byte 2 of the CSW status and hexadecimal 20 represents a channel program check. The dumps should aid in revealing the information in the following list.

a. The channel program check was simulated by the Emulator (the status of the IOB does not indicate a channel program check).

b. The channel program that caused the channel program check.

c. The module involved in causing the channel program check, since the modification occurred shortly before the dump.

A more effective method of obtaining an Emulator snap dump could be employed if it can be determined exactly how the channel program is being modified. The COMP specification can then be used to compare with a specific CCW to obtain the dump closer to the time the CCW is being modified. If the chain bit is being moved or ORed in to add more CCWs to a CCW string, then the following SNAP command might be used to take a snap dump:

```
SNAP = CCMP=A234DC=600000D0,END
```

The following example shows a CCW in the CCW string before and after modification:

```
Before modification:
     .
     .
     .
860260042000C0D0
     .
     .
     .
After modification:
     .
     .
     .
860260046000C0D0
     .
     .
     .
```

Problem 2

Symptom:

A DOS problem program is canceled due to an invalid direct-access seek address after the Emulator message IIV263I is issued.

Assumptions:

The following assumptions are made in this example:

1. The program check occurs only under emulation.

2. The volume involved in a nondedicated volume.

3. The DOS supervisor was generated without the DASD file protection option.

Probable cause:

It is suspected that the DOS problem program is violating an Emulator restriction regarding nondedicated volumes.

Service aids usage:

The DOS message indicates the contents of the CSW, the address of the CCB and the seek address. If a DOS dump is not available, an Emulator snap dump can

be created at the time DOS issues the SIO. This is accomplished by coding the following debug statement:

(sample statement 1) SNAP=COMP=A44=0E,STORAGE=ALL,END

First identify the file involved; the CCB address should help to locate the DTF (in most cases, the CCB immediately begins the DTF). The DTF name can then be located and, in turn, be used to locate the corresponding DLBL/EXTENT cards. If they have been cataloged on the SYSRES label cylinder, the DOS LSERV program can be used. The DLBL/EXTENT cards will contain the 'file-id' of the file; and therefore, its 'data set name'. The dsname is then used to locate the corresponding DD statement.

In addition the DTF should indicate whether it has been opened by DOS (open bit on and volume sequence number and extent description initialized).

Another way to check that DOS actually issued an open for the file is by coding:

(sample statement 2) SNAP= SVC=(2,PHASE=$$BOPEN),END

An Emulator snap dump will be issued for each DOS open. No further debugging is required if it can be determined that a DOS open was never issued to the file in question, since a direct-access volume sharing restriction was violated.

At this point, the DTF has been located, and the corresponding OS DD statement has been identified. The last snap dump issued by the Emulator (sample statement 1) lists the Emulator control blocks:

- COMTAB extensions are listed immediately below their related COMTAB. To identify the associated CTEXT, match the DD name of the involved file with the name listed at the left of each COMTAB extension.

  CTDCBPTR (displacement 8 into the CTEXT) points to the related DCB. Then locate the DEB to find the actual extents of the file as allocated by OS.

- File identification blocks (FIDBLK) are listed below their related COMTAB extensions. Verify that a FIDBLK has been created for the DTF (the DTF name is listed immediately to the left of each FIDBLK entry) for the executing DOS partition. Displacement X'13' into the FIDBLK contains the PID/PIK of the DOS partition issuing the open.

At this point, the DCB and its related DEB have been located and it has been verified that the Emulator actually opened the corresponding OS file.

The presence of message IIV263I indicates that the Emulator module IIVPCE did not find the given DOS seek address in any of the DEB's attached to the corresponding COMTAB; additionally the Emulator module IIVVIO could not identify the channel program as one issued by specific DOS open routines to maintain the VTOC.

It must first be checked whether or not the I/O request came from the problem program either by looking in the snap dump created (sample statement 1 - the instruction address portion of the DOS SVC old PSW should point to the next sequential instruction after the DOS SVC 0 instruction) or by looking at the dump created by DOS (the PSW is listed immediately before the dump).

An Emulator error can be suspected if the I/O request came from a B-transient phase and the seek address indicates either the volume label (cylinder 0, head 0) or the VTOC (the VTOC address should be set in the DTF).

Since all Emulator processing for shared volumes is highly dependent upon DOS open phases, the DOS open phases and release number should be checked. The DOS Release number must be 25, 26, or 27 and the DOS open phases must not be modified by the installation.

A violation of an Emulator restriction can be suspected if the request came from a problem program. Some possible violations are:

- The problem program is expecting an extent at a given address. Solution: The DD statement should contain absolute track allocation.
- The problem program is expecting either:
  1. an extent larger than the one defined in the SPACE parameter of the DD statement
  2. an extent type which does not match the one given in the DD statement
  3. a total number of extents different than those provided.

In all the above cases the DD statement must be modified to reflect an extent status compatible with the problem program before resubmitting the job.

Interpreting Dumps

The following explanation of the contents of an Emulator snap dump is interspersed with samples taken from a snap dump. Capital letters represent the headings found in all dumps and lower case letters represent information that varies with each dump. Each lower case letter used indicates the format of the information and the number of letters indicates the length.

- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents 1 byte of character information
- a represents 1/2 byte of information used for true addresses
- l represents 1/2 byte of information used for local addresses

```
*******************************************************************************
*** Emulator Service Aids Requested**                                         *
*                                                                             *
*OS/360 DOS EMULATOR  VERSION  d  LEVEL  d  TIME  dddddd  DATE  ddddd  PAGE  dddd*
*                                                                             *
*OPTION IN EFFECT  option                                                     *
*******************************************************************************
```

Figure 86 (Part 1 of 7). Service Aids Snap Dump

EMULATOR SERVICE AIDS REQUESTED

identifies the data that fcllows as debugging output.

VERSION d

is the version of the Emulator being executed.

LEVEL d

is the level of the Emulator being executed.

TIME dddddd

is the hour (first two digits), minute (second two digits), and second (last two digits) when the Emulator service aids routines began processing.

DATE ddddd

is the year (first two digits) and day of the year (last three digits).

PAGE dddd

is the page number that appears at the top of each page.

OPTION IN EFFECT option

is the reason the snap dump was taken. The service aids SNAP options appear in the listing as follows:

cccc cuu SNAP 1 of 2 or cccc cuu SNAP 2 of 2

where cccc is one of the following privileged instructions:

- HIO - halt I/O instruction
- ISK - insert storage key instruction
- LPSW - load PSW instruction
- SIO - start I/O instruction
- SSK - set storage key instruction
- SSM - set system mask instruction
- TCH - test channel instruction
- TIO - test I/O instruction

cuu is the channel and unit number of the DOS device. This field will appear only for an I/O related instruction.

SNAP 1 of 2
indicates that the snap dump was taken at the time of the hardware interruption and before control is passed to the Emulator.

SNAP 2 of 2
indicates that the snap dump is taken before passing control to a user local execution exit routine if one was specified.

ATTENTION
indicates that an I/O interruption was simulated to DOS and the attention bit in the CSW was set to 1.

DOS HARD WAIT
indicates that the PSW specified in the operand portion of a DOS LPSW instruction had its wait bit set to 1 and all interruptions disabled. This option will appear automatically only if the service aids routines are in storage at the time the condition occurs.

EMULATOR TRACE REQUESTED
indicates that the following lines will contain the formatted Emulator trace table.

EQUAL COMPARE ADDR llllll STORAGE hhhhhhhh or
EQUAL COMPARE REG dd STORAGE hhhhhhhh or
EQUAL COMPARE REG dd CONTAINS hhhhhhhh
indicates that an equal compare condition has occurred. ADDR llllll is the DOS local main-storage address. STORAGE hhhhhhhh is the contents of the addressed main storage.
REG dd is the DOS general purpose register that contains a local main-storage address or
REG dd is the DOS general purpose register that contains the data that was compared.
dd is the register number.
CONTAINS hhhhhhhh is the compare data supplied by the user.

EXTERNAL INTERRUPTION
indicates that an external interruption was simulated to DOS.

INPUT/OUTPUT INTERRUPTION cuu
indicates that an input/output interruption was simulated to DOS.
cuu is the channel and unit address of the DOS device.

PROGRAM CHECK
indicates that a DOS program check interruption occurred. This condition includes privileged operations not supported by the Emulator, but does not include the supported privileged operations.

PSW = hhhhhhhh hhhhhhhh
indicates that the specified fields of the local execution PSW contain the indicated values.

SNAP DUMP INVOKED BY MODULE cccccccc
indicates that the user exit routine represented by cccccccc requested an optional snap dump.

SUPERVISOR CALL (SVC d) or
SUPERVISOR CALL (SVC d) MODULE cccccccc
indicates that a supervisor call instruction was issued by DOS. (SVC d) is the supervisor call instruction. MODULE cccccccc is the 8-character B-transient phase or load module name addressed by DOS general purpose register 1 at the time of the interruption (for SVC 2 or 4 only).

TIMER INTERRUPTION
indicates that a timer interruption was simulated to DOS.

```
*********************************************************************************************
*INTERUPT AT 111111 (aaaaaa)                                                                 *
*                                                                                            *
*DOS ADJUSTMENT FACTOR aaaaaa                                                                *
*                                                                                            *
*LEX LIST aaaaaa PSW hhhhhhhh hhhhhhhh R14 hhhhhhhh R15 hhhhhhhh AJF aaaaaaaa LMAD 11111111 *
*                                  OPR aaaaaaaa SVC aaaaaaaa PC aaaaaaaa ASYN aaaaaaaa      *
*********************************************************************************************
```

Figure 86 (Part 2 of 7). Service Aids Snap Dump

INTERRUPT AT llllll (aaaaaa)
is the address of the last instruction executed while the CPU was in local execution mode. The local address is represented by llllll. The adjusted address is represented by aaaaaa.

DOS ADJUSTMENT FACTOR aaaaaa
is an address in the Emulator region that corresponds to location 0 of the adjusted DOS storage area. This value is obtained from the adjustment factor field of the local execution list.

LEX LIST aaaaaa
is the starting address of the local execution list.

PSW hhhhhhhh hhhhhhhh
is the PSW field (first eight bytes) of the local execution list.

R14 hhhhhhhh
is the register 14 field of the local execution list.

R15 hhhhhhhh
is the register 15 field of the local execution list.

AJF aaaaaaaa
    is the adjustment factor field in the local execution list.

LMAD 11111111
    is the limit address field in the local execution list. This value corresponds to the DOSSYS parameter in the Emulator execute statement.

OPR aaaaaaaa
    is the true operation pointer field in the local execution list. This field points to the last instruction executed while the CPU was in the local execution mode.

SVC aaaaaaaa
    is the SVC interruption address field of the local execution list. The value contained in this field should be the entry point address of module IIVSVC.

PC aaaaaaaa
    is the program interruption address field of the local execution list. The value contained in this field should be the entry point address of module IIVPCE.

ASYN aaaaaaaa
    is the asynchronous interruption field of the local execution list. The value contained in this field should be the entry point address of module IIVRTE.

```
*****************************************************************************************************
*CSECT IIVCON (EMUCONS)                                                                             *
*      ADDR      DSPL    STORAGE                                                                     *
*      aaaaaa    hhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*      aaaaaa    hhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*      aaaaaa    hhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*                LINES aaaaaa TO aaaaaa SAME AS ABOVE                                              *
*      aaaaaa    hhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*      aaaaaa    hhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*                LINE aaaaaa SAME AS ABOVE                                                         *
*****************************************************************************************************
```

Figure 86 (Part 3 of 7). Service Aids Snap Dump

CSECT IIVCON (EMUCONS)
    identifies the next lines as the contents of the main-storage area occupied by the CSECT IIVCON.

aaaaaa
    is the main-storage address of the line.

hhhh
    is the displacement from zero of the line.

LINES aaaaaa TO aaaaaa SAME AS ABOVE
    are the starting addresses of the first and last lines for a group of lines that are identical to the line immediately preceding.

LINE aaaaaa SAME AS ABOVE
    is the starting address of a line that is identical to the line immediately preceding.

```
*****************************************************************************************************
***EMULATOR I/O CONTROL BLOCKS**                                                                     *
*SYSLOG            COMTAB  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh                   *
*                  LOGIOB  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh                                                  *
*cccccccc¹         COMTAB  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh                   *
*                  STGCON  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh                                                  *
*cccccccc¹         COMTAB  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh                   *
*                  IOB     aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh                                                  *
*cccccccc¹         CTEXT   aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*                                  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  *
*cccccccc²        FIDBLK  aaaaaa  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh                              *
*****************************************************************************************************
```

Figure 86 (Part 4 of 7). Service Aids Snap Dump

****EMULATOR I/O CONTROL BLOCKS****
identifies the next lines as the contents of the Emulator input/output control blocks. The name of each control block precedes its address and contents.

SYSLOG
identifies the control blocks that follow as being related to the DOS system log.

cccccccc¹
identifies the control blocks that follow as being related to the DD statement with the ddname cccccccc¹.

ccccccc²
is the DOS DTF file name associated with the Emulator file identification block.

COMTAB
identifies the control block as an Emulator communications table entry.

LOGIOB
identifies the control block as the work and constants area for Emulator module IIVLOG.

STGCON
identifies the control block as the work and constants area for the unit record staging modules.

IOB
identifies the control block as an OS input/output block.

CTEXT
identifies the control block as an Emulator COMTAB extension.

FIDBLK
identifies the control block as an Emulator file identification block.

aaaaaa
is the starting address of the control block identified on the same line.

If an invalid FIDBLK chain is detected, the following entry will be recorded in the snap dump:

INVALID FIDBLK CHAIN
indicates that the DCB use count in the preceding COMTAB extension entry was greater than the number of file identification blocks whose third word pointed back to that COMTAB extension entry.

```
*********************************************************************************************
*CCW STRING BEGINNING/ENDING BLOCK (BEBLK)                                                  *
*               aaaaaa   bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee  bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee *
*                        bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee  bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee *
*                        bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee  bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee *
*                        bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee  bbbbbbbb eeeeeeee bbbbbbbb eeeeeeee *
*                                                                                            *
*CCW(S)         aaaaaa   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*                        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh                  *
*********************************************************************************************
```

Figure 86 (Part 5 of 7). Service Aids Snap Dump

CCW BEGINNING/ENDING BLOCK (BEBLK)
identifies the next lines as the contents of the main-storage area occupied by the Emulator control block BEBLK.

aaaaaa
is the starting address of the control block.

bbbbbbbb
is the starting main-storage address of a group of CCWs whose data addresses were adjusted by the Emulator module IIVCCW.

eeeeeeee
is the ending main-storage address of a group of CCWs whose data addresses were adjusted by the Emulator module IIVCCW.

CCW(S)
identifies the next lines as the contents of the main-storage area occupied by a group of CCWs whose main-storage addresses are found in the Emulator control block BEBLK.

aaaaaa
is the starting address of the group of CCWs.

```
*********************************************************************************************************
*EMULATOR TRACE TABLE STARTING WITH MOST RECENT ENTRY                                                   *
*SIO  cuu  SEEK hhhhhhhh hhhhhhhh  RPSW hhhhhhhh hhhhhhhh  CAW  hhhhhhhh           CCW1 hhhhhhhh hhhhhhhh *
*SIO  cuu                          RPSW hhhhhhhh hhhhhhhh  CSW  hhhhhhhh hhhhhhhh  SNS  hhhh     OP hh    *
*SIO  cuu                          RPSW hhhhhhhh hhhhhhhh  CSW  hhhhhhhh hhhhhhhh  CCW1 hhhhhhhh hhhhhhhh *
*TIO  cuu                          RPSW hhhhhhhh hhhhhhhh  CSW  hhhhhhhh hhhhhhhh  SNS  hhhh              *
*HIO  cuu                          RPSW hhhhhhhh hhhhhhhh  CSW  hhhhhhhh hhhhhhhh  SNS  hhhh              *
*TCH  cuu                          RPSW hhhhhhhh hhhhhhhh                                                *
*I/O  cuu  IPSW hhhhhhhh hhhhhhhh                          CSW  hhhhhhhh hhhhhhhh  SNS  hhhh              *
*ATTN cuu  IPSW hhhhhhhh hhhhhhhh                          CSW  hhhhhhhh hhhhhhhh  SNS  hhhh              *
*ISK       IPSW hhhhhhhh hhhhhhhh  ADDR 11111111 KEY h                                                   *
*SSK       IPSW hhhhhhhh hhhhhhhh  ADDR 11111111 KEY h                                                   *
*SSM       IPSW hhhhhhhh hhhhhhhh  RPSW hhhhhhhh hhhhhhhh                                                *
*EXT       IPSW hhhhhhhh hhhhhhhh                                                                        *
*TMR       IPSW hhhhhhhh hhhhhhhh                                                                        *
*LPSW      IPSW hhhhhhhh hhhhhhhh  RPSW hhhhhhhh hhhhhhhh                                                *
*SVC  ddd  IPSW hhhhhhhh hhhhhhhh  RPSW hhhhhhhh hhhhhhhh  BTR  cccccccc           R0 hhhhhhhh R1 hhhhhhhh*
*SVC  ddd  IPSW hhhhhhhh hhhhhhhh                          LDMD cccccccc           R0 hhhhhhhh R1 hhhhhhhh*
**TRACE COMPLETE*                                                                                      *
*********************************************************************************************************
```

Figure 86 (Part 6 of 7). Service Aids Snap Dump

EMULATOR TRACE TABLE STARTING WITH MOST RECENT ENTRY
identifies the next lines as entries in the Emulator trace table. The first line printed is the most recent entry. Each succeeding line represents an older entry. The name at the beginning of each line identifies the type of entry on that line:

- ATTN input/output interruption entry with the attention bit in the CSW set to one
- EXT external interruption entry

- HIO halt input/output (HIO) entry
- I/O input/output interruption entry
- ISK insert storage key (ISK) entry
- LPSW load PSW (LPSW) entry
- SIO start input/output (SIO) entry
- SSK set storage key (SSK) entry
- SSM set system mask (SSM) entry
- TCH test channel (TCH) entry
- TIO test input/output (TIO) entry
- SVC supervisor call (SVC) interruption entry
- TMR timer interruption entry

ADDR 11111111
is the local address in main storage.

BTR cccccccc
is the name of the B-transient phase when the supervisor call is an SVC 2 (DOS fetch).

CAW hhhhhhhh
is the DOS channel address word. This value is taken from DOS permanent storage location X'48' if the device is either staged or the DOS system log. In all other cases, the value is taken from location X'24' in the corresponding COMTAB entry.

CCW1 hhhhhhhh hhhhhhhh
is the channel command word addressed by the CAW. Refer to preceding explanation of CAW for further details.

CSW hhhhhhhh hhhhhhhh
is the DOS channel status word. The CSW is taken from location X'40' in DOS permanent storage.

cuu
is the channel and unit for the indicated I/O operation.

IPSW hhhhhhhh hhhhhhhh
is the local execution PSW when the interruption occurs.

KEY h
is the storage protection key associated with the ISK or SSK instruction.

LDMD cccccccc
is the name of the load module when the supervisor call is an SVC 4 (DOS load).

ddd
is the SVC number.

OP hh
is the opcode of the CCW addressed by the CAW. Refer to preceding explanation of CAW for further details.

RPSW hhhhhhhh hhhhhhhh
is the resume PSW. This is the local execution PSW that will be used when control is returned to DOS.

R0 hhhhhhhh
is the contents of DOS general purpose register zero.

R1 hhhhhhhh
is the contents of DOS general purpose register one.

SEEK hhhhhhhh hhhhhhhh
is the seek address for direct-access devices. This value is taken from the IOB.

SNS hhhh
is the first two sense bytes. This field will only appear if the unit check bit in the CSW is set to one. The sense bytes are taken from the IOB.

```
*****************************************************************************************************
*DOS REGS 0-7   hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh       *
*         8-15  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh       *
*                                                                                                   *
*               DOS PERMANENT STORAGE                                                               *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               END OF DUMP                                                                         *
*                                                                                                   *
*               DOS STORAGE                                                                         *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               LINES aaaaaa TO aaaaaa SAME AS ABOVE                                                *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               LINE aaaaaa SAME AS ABOVE                                                           *
*               aaaaaa 111111 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh *
*               END OF DUMP                                                                         *
*****************************************************************************************************
```

Figure 86 (Part 7 of 7). Service Aids Snap Dump

DOS REGS 0-7
8-15
identifies the contents of the DOS general purpose registers at the time the Emulator service aids routines issued the snap dump.

DOS PERMANENT STORAGE
identifies the next four lines as the DOS permanent storage assignment (first 128 bytes).

aaaaaa
is the adjusted main-storage address of the line.

111111
is the local main-storage address of the line.

DOS STORAGE
identifies the next lines as the contents of the main-storage area assigned to DOS.

aaaaaa
is the adjusted main-storage address of the line.

111111
is the local main-storage address of the line.

LINES aaaaaa TO aaaaaa SAME AS ABOVE
are the starting addresses of the first and last lines for a group of lines that are identical to the line immediately preceding.

LINE aaaaaa SAME AS ABOVE
is the starting address of a line that is identical to the line immediately preceding.

END OF DUMP
indicates that the snap dump of DOS main storage is completed.

EMULATOR GENERAL REGISTER ASSIGNMENTS

Figure 87 shows the general register assignments for the resident parts of the DOS Emulator.

Register	Symbol	Assignment
0	RP0EU	Parameter passing register
1	RP1EU	Parameter passing register
2	RW0EU	Work register 1
3	RW1EU	Work register 2
4	RW2EU	Work register 3
5	RW3EU	Work register 4
6	RW4EU	Work register 5
7	RW5EU	Work register 6
8	RB1CD	Base register 2 (used as base for modules)
9	RB2CD	Pointer to local execution list
10	RB3DS	Pointer to DOS storage
11	RB4DS	Base register for IIVCON
12	RB5DS	Base register 6 (DSECT)
13	RB6DS	Base register 7 (DSECT)
14	RL0EU	Link register 1
15	RB0CD	Base register 1 (subroutine)

Figure 87. General Register Assignments

MESSAGE-TO-MODULE RELATIONSHIP

Figure 88 shows the relationship between each Emulator message, the module that requests or issues that message, and the module in which the message text is contained.

Message Number	Module requesting message	Message Code	Module containing text
IIV000A	IIVLOG	--	IIVLOG
IIV000I	IIVLOG	--	IIVLOG
IIV002D	IIVINT	1,8	IIVMG1
IIV003D	IIVINT	2,9	IIVMG1
IIV004D	IIVINT	3,8	IIVMG1
IIV005D	IIVINT	4,10	IIVMG1
IIV006D	IIVINT	5,11	IIVMG1
IIV007D	IIVINT	6,11	IIVMG1
IIV008D	IIVINT	7,11	IIVMG1
IIV009I	IIVPUB	22	IIVMG1
IIV011I	IIVINT	13	IIVMG1
IIV012I	IIVIN2	15	IIVMG1
IIV013I	IIVIN2	16	IIVMG1
IIV014I	IIVGET	123	IIVMG2
IIV015I	IIVIN2	17	IIVMG1
IIV017I	IIVPUB	23	IIVMG1
IIV018I	IIVOPN	128	IIVMG2
IIV019I	IIVINT	14	IIVMG1
IIV020I	IIVIN2	18	IIVMG1
IIV021I	IIVPUB, IIVOPN	24	IIVMG1
IIV022I	IIVINT, IIVIN2	19	IIVMG1
IIV023I	IIVOPN	124	IIVMG2
IIV024I	IIVOPN	129	IIVMG2
IIV025I	IIVINT	25	IIVMG1
IIV026I	IIVINT	26	IIVMG1
IIV027I	IIVIN2	27	IIVMG1
IIV028I	IIVIN2	28	IIVMG1
IIV032I	IIVADD	12	IIVMG1
IIV033I	IIVIN2	20	IIVMG1
IIV034I	IIVIN2	21	IIVMG1
IIV040D	IIVOPN	125,126	IIVMG2
IIV041I	IIVOPN	127	IIVMG2
IIV050D	IIVOPN	130,131	IIVMG2
IIV100E	IIVPUB	101,102,103	IIVMG2
IIV101E	IIVPRP	104	IIVMG2
IIV102E	IIVPRP	105	IIVMG2
IIV103E	IIVPRP	106	IIVMG2
IIV104E	IIVPRP	107	IIVMG2
IIV105E	IIVPRP	108	IIVMG2
IIV106E	IIVPRP	109	IIVMG2
IIV107E	IIVPRP	110	IIVMG2
IIV108E	IIVPRP	112	IIVMG2
IIV109I	IIVPRP	111	IIVMG2
IIV140I	IIVSTG	119	IIVMG2
IIV150I	IIVRTE	120	IIVMG2
IIV160A	IIVPCE, IIVCHK	121	IIVMG2
IIV202I	IIVABN	113	IIVMG2
IIV203I	IIVABN	115	IIVMG2
IIV204I	IIVABN	116	IIVMG2
IIV205I	IIVABN	117	IIVMG2
IIV207I	IIVABN	114	IIVMG2
IIV208I	IIVGR2	221	IIVMG3

Figure 88 (Part 1 of 2). Message-to-Module Relationships

Message Number	Module requesting message	Message Code	Module containing text
IIV250I	IIVGR2	201	IIVMG3
IIV251I	IIVGR2	202	IIVMG3
IIV252I	IIVGR2	203	IIVMG3
IIV253I	IIVGR2	204	IIVMG3
IIV254I	IIVGR2	205	IIVMG2
IIV255I	IIVGR2	206	IIVMG3
IIV256I	IIVDVS	207	IIVMG3
IIV257I	IIVDVS	208	IIVMG3
IIV258I	IIVDVS	209	IIVMG3
IIV259I	IIVDVS	210	IIVMG3
IIV260I	IIVDVS	211	IIVMG3
IIV261I	IIVDVS	212	IIVMG3
IIV262I	IIVDVS	213	IIVMG3
IIV263I	IIVPCE	214	IIVMG3
IIV270A	IIVRCP	216	IIVMG3
IIV271A	IIVRCP	217	IIVMG3
IIV273A	IIVRCP	218	IIVMG3
IIV274A	IIVRCP	220	IIVMG3
IIV275I	IIVSTG	--	IIVSTG
IIV276I	IIVRCP	--	IIVRCP
IIV280I	IIVRCW	215	IIVMG3
IIV281I	IIVRCW	219	IIVMG3

Figure 88 (Part 2 of 2). Message-to-Module Relationships

APPENDIX

DOS Compatibility Feature

DOS COMPATIBILITY FEATURE

DOS emulation is assisted by the DOS Compatibility Feature to an extent determined by which System/370 model is being used. The execute local instruction is provided as a part of this feature. This instruction deals with the emulated environment, which consists of the CPU and storage of the emulated system (DOS). The instruction is used to achieve the direct execution of System/370 nonprivileged instructions and to provide the Emulator with the information necessary for such procedures as simulating privileged instructions, changing the state of the emulated CPU, and executing the input/output operations of the emulated system.

The emulation of channels and input/output devices is considered to be a function of the Emulator. On some models an additional instruction, adjust CCW string, is provided to assist in converting channel programs (data addresses are modified in the CCWs) in the emulated environment to channel programs within the Emulator and vice versa.

The storage of the emulated system must be embedded within the storage available to the Emulator. Addresses relative to the storage of the emulated environment are called local addresses; the corresponding addresses relative to the storage of the Emulator are called program addresses. The size of the emulated storage and the relationship between local and program addresses are provided by the operand of the execute local instruction, the local execution list. Other parameters of the emulated environment provided by the local execution list are the condition code, program mask, local instruction address, and the contents of general registers 14 and 15 of the emulated CPU.

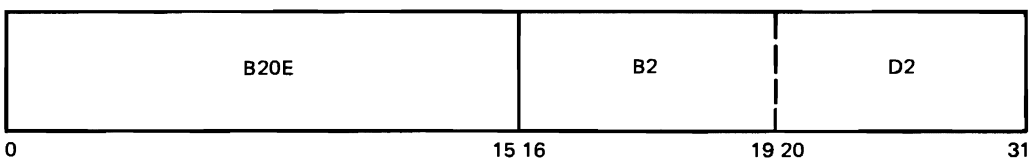
Execution of the execute local instruction causes the CPU to be placed in local execution mode. General registers 14 and 15 are loaded from the local execution list, and the PSW is modified by replacing the condition code, program mask, and the instruction address with values from the list. In local execution mode the CPU functions as the CPU of the emulated environment and executes nonprivileged instructions contained therein, treating all instruction and data addresses as local to that environment.

Any interruption condition terminates local execution mode. An interruption is said to be synchronous if it is a result of instruction execution within the emulated environment; if it is caused by conditions or events external to the emulated environment it is said to be asynchronous. The current parameters of the emulated environment are stored in the local execution list, and if the interruption was synchronous, the appropriate interruption code, instruction length code, and the program address of the instruction causing the interruption are also stored. The instruction address in the PSW is replaced by the program address, appropriate to the interruption type, of the next instruction of the Emulator. If the interruption was synchronous, instruction execution proceeds with the modified PSW. If the interruption was asynchronous to local instruction execution, which would be the case, for example, with an input/output interruption, the modified PSW is stored in the permanent storage location for the interruption type, a new PSW is loaded, and instruction sequencing proceeds under control of the new PSW.

Refer to IBM System/370 Principles of Operation for more information on system structure and modes, hardware instructions, input/output operations, interruption action, and status switching.

Execute Local Instruction

$D_2(B_2)$ [S]



The local execution list (Figure 89), operand of the execute local instruction, provides the CPU with the parameters necessary for an emulated environment when the CPU is placed in local execution mode. The local execution list is 40 bytes in length and must begin on a 64-byte boundary.

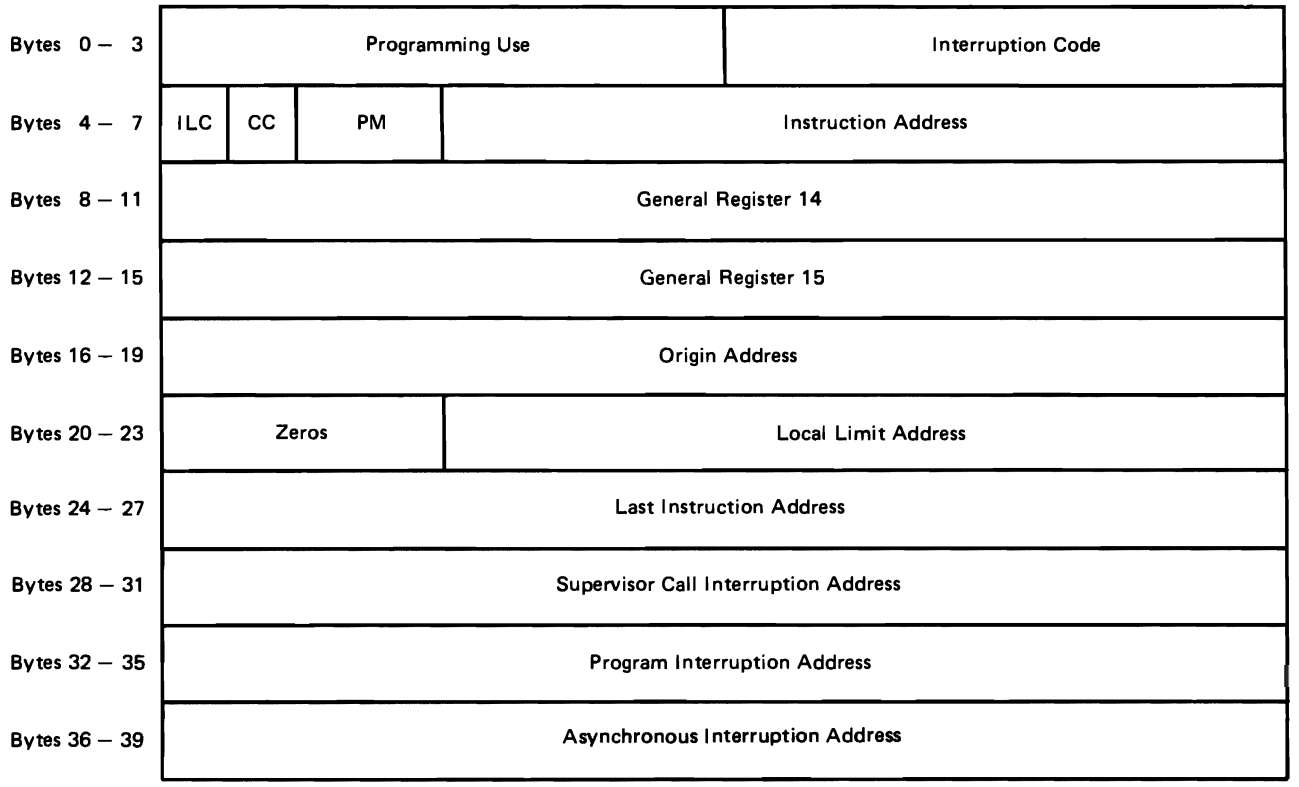


Figure 89. Local Execution List

The condition code, program mask, and instruction address in the current PSW are replaced by the values in the corresponding subfields of the local PSW field (bytes 0-7) of the local execution list. General registers 14 and 15 are loaded, respectively, with the values contained in the general register 14 and general register 15 fields of the local execution list. The CPU is placed in local execution mode.

The modified PSW is not checked for program interruptions during execution of the execute local instruction. The checking occurs as part of the execution of the next instruction.

Any program event exception resulting from completion of the execute local instruction is held pending termination of local execution mode. The program event recording code and the program event address, which is either the program address of the execute local instruction or the program address of an execute instruction, as appropriate, may be stored immediately or when the interruption is actually taken. If successful-branch, storage-alteration, or general-register-alteration monitoring is active, the associated program event will be indicated, regardless of what bytes of storage or what general registers were actually altered by the execute local instruction and regardless of the general-register-alteration mask.

Resulting Condition Code:

Upon completion of the execute local instruction, the code is set according to the condition code loaded from the local PSW field of the local execution list.

Program Interruptions:

Operation: The instruction is not installed. The operation is suppressed.

Protection: The operand location is protected, and the key in storage associated with the operand does not match the protection key in the PSW. The operation is suppressed.

Addressing: The address of the local execution list is invalid, or the value formed by summing the origin address and the local limit address exceeds the maximum valid program address if dynamic address translation is not active or exceeds 16,777,215 if dynamic address translation is active. The operation is suppressed.

Specification: The operand address does not specify a 64-byte boundary, the origin address is not a multiple of 4096, or the local limit address is not one less than a multiple of 4096. The operation is suppressed.

Segment Translation: The address of the local execution list cannot be translated. The operation is nullified.

Page Translation: The address of the local execution list cannot be translated. The operation is nullified.

Translation Specification: A translation specification exception was detected while translating the address of the local execution list. The operation is suppressed.

Special Operation: The CPU is already in local execution mode. The operation is suppressed. The interruption will be indicated to the program which placed the CPU in local execution mode.

Program Event: The execute local instruction has been nullified or suppressed, and attempted execution of the instruction resulted in an instruction-fetching event. The program event condition is indicated in the interruption code concurrently with the code denoting the exception condition which caused nullification or suppression. Successful-branch, storage-alteration, and general-register alteration events are not possible; they cannot occur with the exception conditions causing nullification or suppression.

Program Event (deferred until local execution mode is terminated): Execution of the instruction generated a program-event exception. The instruction has been completed.

Local Execution List

The local execution list, which is the operand of the execute local instruction, provides the CPU with the parameters necessary for an emulated environment when the CPU is placed in local execution mode. Upon termination of local execution mode by an interruption condition, the modified values of these parameters are stored at locations in the local execution list. Included in the list is the address of the first instruction in the Emulator to be executed following termination of local execution mode. The following is a detailed description of each field and subfield in the 40-byte list:

- The Local PSW Field (bytes 0-7) contains the PSW for the emulated environment. The PSW is in the BC-mode format.

  The Programming Use Subfield (bits 0-15) is an area in which the Emulator may place bits 0-15 of the current BC-mode PSW of the emulated environment. This field is neither inspected nor altered by execution

of the execute local instruction. The system-mask, protection key, machine check mask, BC-mode, wait state, and problem state bits of the current PSW are unaltered by execution of the execute local instruction. Also unaltered are the following functions: program-event recording (although it is inhibited during local execution mode), and dynamic address translation for program addresses.

The Interruption Code Subfield (bits 16-31) is identical to the interruption code field in the BC-mode PSW. Upon termination of local execution mode by a synchronous interruption, the 16-bit interruption code associated with the exception condition in the emulated environment is placed in this field. The contents of this field are unpredictable when local execution mode has been terminated by an asynchronous interruption.

The Instruction Length Code Subfield (bits 32-33) is identical to the instruction length code field in the BC-mode PSW. When local execution mode is terminated by a synchronous interruption, the value stored in this field is the same as that stored in the old PSW for the same interruption condition in normal mode. The contents of this field are unpredictable when local execution mode has been terminated by an asynchronous interruption.

The Condition Code Subfield (bits 34-35) replaces the current condition code upon completion of the execute local instruction. Upon termination of local execution mode because of any interruption condition, the current condition code is stored in this field.

The Program Mask Subfield (bits 36-39) replaces the program mask in the current PSW upon completion of the execute local instruction. Upon termination of local execution mode because of any interruption condition, the current program mask is stored in this field.

The Local Instruction Address Subfield (bits 40-63) replaces the instruction address in the current PSW upon completion of the execute local instruction. It is the local address of the first instruction to be executed in local execution mode. Upon termination of local execution mode because of any interruption condition, the current local instruction address is stored in this field.

- The General Register 14 Field (bytes 8-11) provides a fullword value which is loaded into general register 14 upon completion of the execute local instruction. Upon termination of local execution mode because of any interruption condition, the current contents of general register 14 are stored in this field.

- The General Register 15 Field (bytes 12-15) provides a fullword value which is loaded into general register 15 upon completion of the execute local instruction. Upon termination of local execution mode because of any interruption condition, the current contents of general register 15 are stored in this field.

- The Origin Address Field (bytes 16-19) specifies the program address that corresponds to local address 0 for the emulated environment. This address must be a multiple of 4096, that is, the three low-order hexadecimal digits must be zeros. When this field is fetched from the local execution list, the high-order byte is ignored; it is reserved and should be set to zero.

The CPU, in converting a local address to a program address, adds the value obtained from this field to the local address. The contents of this field are not altered by the execution of the execute local instruction or by the local execution mode termination process.

- The Local Limit Address Field (bytes 20-23) specifies the maximum address permitted the emulated environment. The address must specify one less than a multiple of 4096, that is, the three low-order hexadecimal digits must be X'F's. The high-order byte of this field is reserved and should be set to zero. The contents of this field are not altered by execution of the execute local instruction or by the local execution termination process.

- The Last Instruction Address Field (bytes 24-27) provides, when local execution mode is terminated by a synchronous interruption, the program address of the instruction causing the interruption. If, however, the instruction causing the interruption was the subject of an execute instruction, the program address of the execute instruction is stored in this field. The high-order byte is set to zero. The contents of this field are unpredictable when local execution mode has been terminated by an asynchronous interruption.

- The Supervisor Call Interruption Address Field (bytes 28-31) specifies the program address of the first instruction to be executed subsequent to termination of local execution mode by a supervisor call instruction. After the current instruction address has been stored in the local instruction address subfield, the contents of the supervisor call interruption address field replace the current instruction address in the PSW. The high-order byte of this field is ignored. It is reserved and should be set to zero. The contents of this field are not altered by execution of the execute local instruction or by the local execution mode termination process.

- The Program Interruption Address Field (bytes 32-35) specifies the program address of the first instruction to be executed subsequent to termination of local execution mode by a program interruption condition other than one of the address translation exceptions. After the current instruction address has been stored in the local instruction address subfield, the contents of the program interruption address field replace the current instruction address in the PSW. The high-order byte of this field is ignored. It is reserved and should be set to zero. The contents of this field are not altered by execution of the execute local instruction or by the local execution mode termination process.

- The Asynchroncus Interruption Address Field (bytes 36-39) specifies the program address of the first instruction to be executed subsequent to termination of local execution mode by any asynchronous interruption. After the current instruction address has been stored in the local instruction address subfield, the contents of the asynchronous interruption address field replace the current instruction address in the PSW. The high-order byte in this field is ignored. It is reserved and should be set to zero. The contents of this field are not altered by execution of the execute local instruction or by the local execution mode termination process.

The contents of the local PSW, general register 14, general register 15, and last instruction address fields are unpredictable while the CPU is in local execution mode.

Programming Note: Alteration of the local execution list while the CPU is in local execution mode gives unpredictable results. For those models which check the validity of local addresses, the local execution list should be placed at a location not addressable while the CPU is in local execution mode. For those models which do not, the local execution list should be placed at an address that is less than the origin address so as to minimize the chance of error.

Loss of CPU addressability to the local execution list once the CPU has been placed in local execution mode will make proper termination of local execution mode impossible. The key in storage associated with the local execution list

should not be altered while the CPU is in local execution mode. If dynamic address translation is active, no translation table entry required for translation of the address of the local execution list should be altered or invalidated while the CPU is in local execution mode.

Local Execution Mode

Execution of an execute local instruction places the CPU in local execution mode; attached channels are not affected. Program event recording is inhibited for subsequent instruction execution while the CPU remains in local execution mode, and any program event exception resulting from execution of the execute local instruction remains pending until the CPU is removed from local execution mode. The CPU remains in local execution mode until an interruption or a CPU reset occurs.

In local execution mode, all instruction and operand addresses are local to the emulated environment. Addition of the value contained in the origin address field of the local execution list converts these addresses to program addresses.

Instruction execution in local execution mode proceeds under the protection in force at the time the execute local instruction was executed.

Except as noted below, any installed nonprivileged System/370 instruction may be executed while the CPU is in local execution mode. If a privileged instruction is encountered, a privileged operation exception is recognized, regardless of the problem state bit of the current PSW. If a monitor call instruction or any emulation instruction is encountered, a special operation exception is recognized. For any of these exceptions, the operation is suppressed, and a program interruption takes place. Conditions causing a privileged operation exception are checked for before conditions that would cause a special operation exception.

For some models, when a local address specifies any part of an instruction or data outside the available local storage, as defined by the local limit address, an addressing exception is recognized. A program interruption takes place, and the instruction is suppressed or terminated as appropriate. Such an addressing exception has priority over a segment translation or page translation exception resulting from attempted translation of the corresponding program address. If dynamic address translation is active, an addressing exception of lower priority may also be caused by an invalid address in the segment or page tables; no distinction is made among the various causes of addressing exceptions.

Alteration of the local execution list while the CPU is in local execution mode gives unpredictable results.

The condition code and program mask in the current PSW may be altered by instruction execution while the CPU is in local execution mode. The setting of the condition code by instruction execution is unaffected by local execution mode.

The CPU remains in local execution mode when it is placed in the stopped state, and, if a program event exception is pending as a result of the execute local instruction which placed the CPU in local execution mode, it remains pending.

If the store status function is executed while the CPU is in local execution mode, the program address of the local execution list is stored in the word at absolute storage location 268, and bit 0 of this word is set to one. The instruction address in the PSW stored in the doubleword at absolute storage location 256 is either the local instruction address or the corresponding program address, depending upon the model. (If the execute local instruction is installed, execution of the store status function causes all bits of the word at absolute storage location 268 to be set to zeros when the CPU is not in local execution mode.)

Termination of Local Execution Mode

Any interruption or any reset removes the CPU from local execution mode.

Interruptions are divided into two classes: the class of asynchronous interruptions which encompasses all those interruptions caused by conditions external to the emulated environment, and the class of synchronous interruptions which are those resulting from instruction execution while in local execution mode. The class of synchronous interruptions is further divided into two subclasses: supervisor call interruptions arising from execution of a supervisor call instruction while in local execution mode, and program interruptions due to exception conditions generated by instruction execution while in local execution mode. Program interruptions resulting from segment translation, page translation, and translation specification exceptions are asynchronous interruptions, since a condition causing such an interruption is external to the emulated environment.

In local execution mode, the recognition of any interruption condition for which the CPU is enabled (other than a pending program event exception resulting from the execution of the execute local instruction which placed the CPU in local execution mode) causes the current contents of general registers 14 and 15, the condition code, the program mask, and the instruction address to be stored in the corresponding fields of the local execution list. If the interruption causes nullification, the instruction address is the local address of the nullified instruction; otherwise, it is the updated instruction address. Additionally, if the interruption is a synchronous interruption, the interruption code associated with the exception, the instruction length code, and the program address of the last executed instruction are stored; otherwise the contents of these fields in the local execution list are unpredictable. For a program interruption resulting from an odd-numbered instruction address, the instruction length code is unpredictable, and the last instruction address is the program address corresponding to the odd-numbered local instruction address. If the last executed instruction was the subject of an execute instruction, the program address of the execute instruction is stored in the last instruction address field.

After the instruction address has been stored in the local execution list, the instruction address in the current PSW is replaced by the supervisor call interruption address, the program interruption address, or the asynchronous interruption address from the local execution list, as appropriate. The modified PSW is not checked for program interruptions during the process of termination of local execution mode; these checks occur as part of the execution of the next instruction.

The CPU is then removed from local execution mode, and if the interruption was a synchronous interruption, the interruption condition is cleared. If a program event exception is pending as a result of the execute local instruction that placed the CPU in local execution mode, the exception is now recognized.

If the interruption was a synchronous interruption, and if no program event exception is recognized, instruction execution proceeds as specified by the modified PSW. If the interruption was an asynchronous interruption, or if a program event exception has been recognized, the modified PSW, with any ancillary information, is stored in the permanent storage locations for the highest priority interruption condition pending, a new PSW is loaded from the appropriate permanent storage location, and the highest priority interruption condition is cleared. The state of the CPU and subsequent instruction execution are as specified by this new PSW, which also specifies whether any interruption conditions of lower priority are accepted or kept pending.

If the CPU was removed from local execution mode because of a segment translation, a page translation, or a translation specification exception, and if a program event exception is recognized when local execution mode is terminated, both conditions are indicated concurrently in the program interruption code stored.

An indication of an enabled input/output or external interruption condition may cause the CPU to initiate termination of local execution mode. In such a case, should a pending program event exception be recognized, it is permissible for the CPU to complete termination of local execution mode and subsequently take the program interruption, even though the indication of the initial interruption condition be canceled.

Under certain nonrecoverable machine check conditions, the CPU may be unable to store into the local execution list or to fetch the asynchronous interruption address from the list. In such cases, local execution mode is terminated, any pending program event exception is lost, and the contents of the local execution list are unpredictable, as is the instruction address in the machine check old PSW.

Otherwise, if upon termination of local execution mode the CPU is unable to store into the local execution list or to fetch the appropriate interruption address from the list, a protection exception is recognized, regardless of the reason why the list cannot be accessed. Any supervisor or program interruption conditions resulting from instruction execution in local execution mode are cleared. A program event exception pending from execution of the execute local instruction may be lost or may be indicated concurrently with the protection exception. Local execution mode is terminated. The contents of the local execution list and the instruction address in the old PSW are unpredictable. On some models, if the CPU is unable to access the local execution list upon termination of local execution mode, the CPU will enter an internal loop trying to terminate local execution mode; in these cases, the CPU must be reset.

Any reset of the CPU in local execution mode results in termination of local execution mode. Execution of the current CPU operation is terminated, and all pending program and supervisor call interruption conditions are cleared. The contents of the local execution list are unpredictable. After a CPU reset while in local execution mode, the instruction address in the old PSW is either the local instruction address or the corresponding program address if the reset occurred while the CPU was in the stopped state; otherwise, it is unpredictable. After any other reset the PSW is cleared to zeros.

Programming Notes: When any address translation exception occurs while the CPU is in local execution mode, the instruction address in the program old PSW resulting from the interruption is the asynchronous interruption address from the local execution list, not the program address of the instruction causing the interruption.

Instruction execution in local execution mode may alter the program mask. After termination of local execution mode, the program mask in the current PSW is the same as that which has been stored in the local execution list.

Model Dependencies: The System/370 Model 155 does not check local addresses against the local limit address unless the dynamic address translation facility is installed.

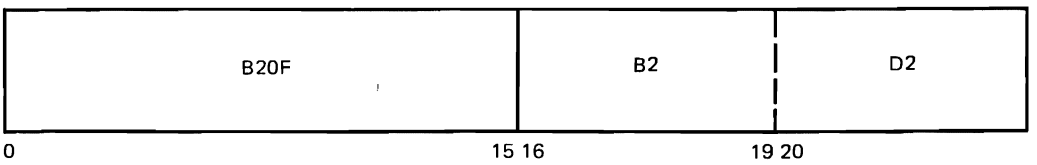
The System/370 Model 155 stores a PSW containing the program address corresponding to the local instruction address when the store status function is executed while the CPU is in local execution mode. On all other models, the instruction address in the PSW stored is the local instruction address.

After a CPU reset while in the stopped state, the instruction address in the PSW is the program address corresponding to the local instruction address in the System/370 Model 155. On all other models, the instruction address is the local instruction address.

The System/370 Model 155 is unable to recover when it finds itself unable to access the local execution list upon termination of local execution mode.

Adjust CCW String Instruction

$D_2(B_2)$ [S]



As part of the DOS Emulation Feature, an instruction, called the adjust CCW string instruction, is provided on some CPU models to assist the Emulator in locating the CCWs of a channel program, testing the validity of the addresses of CCWs and their storage areas, and adjusting their data addresses. On other models, these functions are accomplished by the Emulator.

Consecutive doublewords are interpreted as a string of CCWs, and the data address fields of these CCWs are adjusted by the addition of a signed adjustment factor. The string of CCWs begins at a location designated in the CCW address field of a list specified by the operand address. The list, called the ACCW list, begins on a 64-byte boundary and contains these fields:

Bytes 0 – 3	Signed Adjustment Factor
Bytes 4 – 7	Local Limit Address
Bytes 8 – 11	Operation Byte
Bytes 12 – 15	Operation Pointer
Bytes 16 – 19	CCW Address
Bytes 20 – 23	TIC Data Address

Figure 90. ACCW list

The signed adjustment factor used to adjust the data address field is contained in the ACCW list. The factor is a nonnegative value to convert addresses local to the emulated environment to program addresses, and it is a negative value to convert program addresses to local addresses. Adjustment consists in

replacing the data address in a CCW with the 24 low-order bits of the algebraic sum of the adjustment factor and the data address, where the latter is considered to be an unsigned binary number. The data address fields of consecutive CCWs are adjusted until one of the following conditions occurs:

- The last CCW of the string, as indicated by the absence of both the chain data and chain command flags, has been adjusted.
- A CCW whose command code specifies transfer in channel has been adjusted.
- A CCW specifying indirect data addressing has been adjusted.
- A CCW has been adjusted that specifies a local storage area at least partially outside the limits of local address 0 and the local limit address.
- The local address of the next CCW to be adjusted falls outside the area bounded by local address 0 and the local limit address, or the local address does not specify a doubleword boundary.

Anyone of these conditions completes execution of the adjust CCW string instruction and sets the condition code to indicate the reason for completion. The program address + 8 of the last CCW adjusted is placed in the ACCW list. If data chaining was in progress, the command code and the program address of the CCW containing the command code are stored in the operation byte field and the operation pointer field, respectively, of the ACCW list. Otherwise, zeros are stored in the operation byte field. If the last CCW adjusted specifies transfer in channel, the TIC data address field of the ACCW list contains the data address, prior to adjustment, that was in the CCW.

Execution of the instruction may be interrupted by an external event. If an interruption condition is presented to the CPU during execution of the instruction, with the CPU enabled for the interruption, the interruption is taken upon completion of the current unit of operation. A unit of operation is the operation within the instruction execution required for complete adjustment of a CCW. The instruction address stored in the old PSW as a result of the interruption is the address of the adjust CCW string instruction, or the address of an execute instruction if the adjust CCW string was the object of such an instruction. The contents of the ILC and CC fields in the old PSW are unpredictable. The contents of the ACCW list are such that execution of the adjust CCW instruction may be resumed by loading the old PSW stored as a result of the interruption.

Resulting Condition Code:

0 End of the CCW string; the last CCW adjusted specifies neither data chaining nor command chaining.

1 A CCW specifying transfer in channel was the last CCW adjusted.

2 The last CCW adjusted specifies a local storage area at least partially outside the limits of local address 0 and the local limit address.

3 The local address of the next CCW to be adjusted does not specify a doubleword boundary or is outside the limits of local address 0 and the local limit address, or the last CCW adjusted specifies indirect data addressing.

Program Interruptions:

Operations: The instruction is not installed. The operation is suppressed.

Protection: The operand location is protected, and the key in storage does not match the protection key. The operation is suppressed for fetching and terminated for storing; or a CCW in the designated string is protected, and the operation is terminated.

Addressing: The address of the ACCW list is invalid, and the operation is suppressed; or the address of a CCW in the designated string is invalid, and the operation is terminated.

Specification: The operand address does not specify a 64-byte boundary, or the signed adjustment factor is not a multiple of 4096, or the local limit address is not one less than a multiple of 4096. The operation is suppressed.

Segment Translation: The address of the ACCW list or of a CCW cannot be translated. The operation is nullified.

Page Translation: The address of the ACCW list or of a CCW cannot be translated. The operation is nullified.

Translation Exception: A translation specification exception is detected while translating the first operand address or the address of a CCW. The operation is terminated.

Special Operation: The adjust CCW string instruction was encountered while the CPU was in local execution mode. The operation is suppressed, and the interruption is indicated to the program that placed the CPU in local execution mode.

Program Event: An instruction-fetching or storage alteration event has been encountered. The operation is completed, nullified, or terminated. (See the detailed description of program-event recording.)

The ACCW List

The operand of the adjust CCW string instruction is the ACCW list. The following is a detailed description of each field in the 24-byte list at the initiation of the instruction execution:

The Signed Adjustment Factor Field (bytes 0-3) is a signed fullword binary number whose value must be a multiple of 4096 (that is, the three low-order hexadecimal digits must be zeros).

The Local Limit Address Field (bytes 4-7) provides the maximum address permitted the emulated environment. This address must be one less than a multiple of 4096 (that is, the three low-order hexadecimal digits must be X'FFF'). Local CCW addresses and the extreme local addresses of the storage area defined for each CCW by the data address, command code, and count must fall between local address 0 and the local limit address. The high-order byte of this field is reserved, and should be set to zero.

The Operation Byte Field (bytes 8-11) when zero, byte 11 indicates that the initial CCW to be adjusted contains its own command code. When byte 11 is not zero, the first CCW to be adjusted is considered to be part of a sequence of CCWs that are data chained together, and byte 11 provides the command code for this sequence. When fetched from the ACCW list, the three high-order bytes of this field are ignored; they should be set to zeros.

The Operation Pointer Field (bytes 12-15) contains the address of the CCW that originated the operation byte for the last CCW adjusted that did not specify transfer in channel. When the field is fetched from the ACCW list the high-order byte is ignored; it is reserved and should be set to zeros.

The CCW Address Field (bytes 16-19) provides the address of the first CCW to be adjusted. When this field is fetched from the ACCW list, the high-order byte is ignored; it is reserved and should be set to zeros.

The TIC Data Address Field (bytes 20-23) is ignored upon initiation of instruction execution.

The contents of the signed adjustment factor field and the local limit address field are not altered by the adjust CCW string instruction. At the end of the operation the contents of the remaining fields depend upon the manner in which execution tock place. In general, if the operation was suppressed, the contents of the entire list are unaltered, while if the operation was terminated, the contents of bytes 8-23 of the ACCW list are unpredictable. Upon instruction nullification or completion, the contents of the remaining fields are as follows:

- For the operation byte field:
  - Bytes 8-10 will always be zeros.
  - Upon completion with condition code 0, bute 11 will be zero.
  - Upon completion with condition code 1, if the TIC command was data chained into, byte 11 will contain the command code in force for the string of CCWs data chained together; otherwise, it will be zero.
  - Upon completion with condition code 2, byte 11 will contain the command code in force for the last CCW adjusted if that CCW specifies data chaining; otherwise, it will be zero.
  - Upon completion with condition code 3:

    If no CCWs have been adjusted, byte 11 is unaltered.

    If the last CCW adjusted specifies indirect data addressing, byte 11 will contain the command code in force for that CCW.

    If the last CCW adjusted does not specify indirect data addressing, byte 11 will contain the command code in force for the last CCW adjusted if that CCW specifies data chaining; otherwise it will be zero.

    Upon nullification, if the last CCW adjusted specifies data chaining, byte 11 will contain the command code in force for that CCW; otherwise, it will be zero.
- For the operation pointer field:
  - The high-order byte will be set to zero.
  - Upon completion with condition code 1, if the TIC command was data chained into, the operation pointer field will contain the program address of the CCW at the start of the chain of CCWs data chained together. Otherwise, the operation pointer field will contain the program address of the CCW containing the command code in force prior to fetching the TIC command.

- Upon completion with condition code 3, when no CCWs have been adjusted, the operation pointer field will contain the same address as upon initiation of instruction execution.

- Under all other conditions, the operation pointer field will contain the program address of the CCW containing the command code in force for the last CCW adjusted.

- For the CCW address field:

  - The high-order byte will be set to zero.

  - Upon completion with condition code 3, when no CCWs have been adjusted, the CCW address field will contain the same address as upon initiation of instruction execution.

  - Under all other conditions, the CCW address field will contain the program address + 8 of the last CCW adjusted.

- The TIC data address field:

  - Upon completion with condition code 1, the TIC data address field contains zero in byte 20 and the data address from the TIC command, prior to its adjustment, in bytes 21-23.

  - Under all other conditions, the TIC data address field is unaltered by execution of the adjust CCW string instruction.

Upon nullification due to a segment-translation exception or a page translation exception encountered when attempting to access the initial CCW, the data contents of the ACCW list are unaltered. The high-order byte of the operation pointer field and the CCW address field and bytes 8-10 of the operation byte field may either be unaltered or set to zeros.

Programming Note:

If a CCW specifies transfer in channel, no limit checking of the data address occurs. The data address is adjusted, and the instruction is completed with condition code 1.

If a CCW specifies indirect data addressing, no limit checking of the data address occurs. The data address is adjusted, and the instruction is completed with condition code 3.

During execution of the adjust CCW string instruction, the contents of the ACCW list are unpredictable. In particular, the CPU may alter the contents of the list after the adjustment of each CCW in the string.

If the CPU is unable to access the ACCW list after initiation of the adjust CCW string instruction, the operation will be terminated. The cause of termination will be indicated as a program interruption due to protection exception.

This condition may arise independently of any other interruption conditions, or it may be detected only when an attempt is made to store into the ACCW list in order to interrupt the adjust CCW string instruction. In the latter case, the program interruption due to protection exception is taken (concurrently with a program event exception if necessary). Any address translation exceptions are lost. If the interruption condition is of lower priority than the program interruption, the program interruption occurs first, and the other interruption is subject to the control of the mask bits in the program new PSW.

Model Dependencies:

The adjust CCW string instruction is available on Models 145 and 155.

GLOSSARY

The terms in this glossary are defined as they pertain to this document.

ACCW list: See adjust CCW list.

address translation: The process of changing the address of a data item or an instruction from its virtual address to its real storage address.

adjust CCW list: The area of main storage that contains data pertinent to the adjust CCW string instruction (See Appendix).

adjust CCW string: The instruction used to modify CCW command addresses so they address the main-storage area assigned to the DOS storage area in the Emulator region.

adjustment factor: A constant that is added to the address fields of DOS instructions as they are executed.

appendages: Emulator routines that provide additional control over I/O operations during channel program execution. An appendage may receive control when a channel end, abnormal end, end-of-extent, or a start I/O condition occurs. Refer to Data Management for System Programmers for OS or OS/VS.

asynchronous interruptions: Interruptions that occur without regular time relationships; unexpected or unpredictable with respect to the execution of a program's instructions (I/O, machine check, external).

background partition (BG): In DOS, storage is divided into three partitions. The partition with the lowest priority is the background partition (see 'foreground partition').

basic telecommunications access method (BTAM): Controls transmission and reception of messages over telecommunications lines in response to READ, WRITE and CONTROL macro instructions issued in the user's problem program. The primary functions of BTAM are channel program generation at object time and, at the user's option, buffer management.

beginning and end block (BEBLK): Contains one 8-byte entry for each group of CCWs that contain the beginning and ending addresses +8 of each group.

bootstrap: A technique or device designed to bring a program into main storage by means of its own action, for example, a subroutine whose first few instructions are sufficient to bring the rest of it into the computer from an input device.

B-transient area: A special area of the DOS supervisor reserved for B-transient routines.

B-transient routines: DOS routines that deal primarily with the logical input/output control system. These routines are located in the core image library.

channel address word (CAW): Contains the address of the first CCW.

channel command word (CCW): Indicates to a channel what I/O operation it should start. For operations involving data transfer, also indicates the main-storage location into which data is to be placed or read from, and how many bytes of data are to be transferred.

channel status word (CSW): Indicates to a program the status of an I/O device, control unit, channel, and subchannel.

command control block (CCB): A DOS control block containing information pertinent to DOS data set processing.

common area: A control section used to reserve a main-storage area that can be referred to by other modules. communications table (COMTAB): Contains one entry for every DOS device allocated to emulation plus an entry for SYSLOG.

compatible data set: When a data set does not have to be changed in format to be accessed by the Emulator, DOS, or OS.

COMTAB extension: The control block that is either chained to a COMTAB entry or to another COMTAB extension entry. There is one COMTAB extension entry for every shared or OS indexed sequential data set.

conversion: The process of changing from one form of representation to another, for example, converting indexed sequential data sets from DOS to OS format.

DASD label (DLBL): Internal format of a DOS DLBL extent statement read from the label cylinder by DOS OPEN phases.

data control block (DCB): A control block through which the information required by access routines to store and retrieve data is communicated to the access routines.

data definition (DD) statement: A job control statement that describes a data set associated with a particular job step.

data event control block (DECB) -- BISAM: A control block through which information concerning input or output operations is communicated to BISAM.

data extent block (DEB): Contains one or more extent entries and other control information for the data set with which it is associated.

data set: The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. See also file.

data set control block--extension (format 3) (DSCB): A control block that describes up to 13 additional extents that cannot be described in a format 1 DSCB.

data set control block--identifier (format 1) (DSCB): A control block that describes the characteristics and up to three extents of a data set.

data set control block--VTOC (format 4) (DSCB): The 140-byte block, always the first DSCB in the VTOC, that describes the VTOC data set.

data set name: A 44 character name that identifies an OS data set. This name must be identical to the DOS file ID to indicate a DOS shared file. (See also DSNAME and file ID.)

dedicated device: A device assigned specifically to operate under DOS. It is inaccessible to OS programs until released from this assignment.

define the file (DTF): A macro instruction that describes the characteristics of a logical file, indicates the type of processing for the file, and specifies the main-storage areas and routines to be used.

define the file for indexed sequential (DTFIS): Same as the definition for a DTF except that a DTFIS describes the characteristics of a logical file for ISAM.

DIAG block: The area of main storage used to adjust and readjust the data address portions of the CCWs from DOS local addresses to OS true addresses, then back to DOS local addresses.

diagnostic block: See DIAG block.

direct-access storage device (DASD): A type of storage device wherein access to the next position from which information is to be obtained is in no way dependent on the position from which information was previously obtained.

disk operating system (DOS): Wherever the term DOS is used in this manual, it refers to DOS system and/or DOS problem programs.

DLBL/EXTENT card image: Contains the extent limits, extent type, extent sequence number, logical unit, volume serial number, expiration and creation dates, and DLBL indicator according to the OS information about the file.

DOS compatibility feature: A hardware feature that permits execution of DOS programs within the OS environment.

DSNAME: A parameter in a DD statement that is used by the operating system to locate the data set on a volume. (See also data set name.)

DTFIS ADD-RETRVE-ADDRTR table: A DTF table built at assembly time and tailored to the DTF parameters associated with the add and retrieve functions of ISAM.

DTFIS load table: A DTF table built at assembly time and tailored to the DTF parameters associated with the load function of ISAM.

dummy record: A record that takes up space in a file but is not actually a record of that file.

dynamic address translation (DAT): A hardware feature that performs the translation of a virtual storage address to a real storage address during execution of an instruction.

ECB pointer table: A list of 4-byte addresses for SYSLOG, the three Emulator ECBs (prompt, WTOR, and timer), and the ECBs for staged devices and devices being used for emulation.

EMUCONS: A dummy section (DSECT) containing data constants common to most Emulator modules. See also IIVCON.

emulation: The combination of programming techniques and special machine features that permits DOS programs to operate in the OS environment.

Emulator (as used in this publication): Designates the DOS Emulator program IIVEMU. See also emulation.

Emulator region: The main-storage area (OS) in which the Emulator program, Emulator control blocks, and the DOS main-storage area reside.

event control block (ECB): A control block used to represent the status of an event.

execute: To carry out an instruction or perform a routine.

execute local: The instruction used to place the system in local execution mode and to adjust the instructions and instruction operands within the storage area assigned to the adjusted DOS storage area.

extent: The physical locations on input/output devices occupied by, or reserved for, a particular data set.

FCB: See forms-control buffer.

FCB image: An image of the carriage control used by the IBM 3211 Printer and the Emulator. The image must be assembled and link-edited into SYS1.IMAGELIB. (See forms-control buffer.)

file: The major unit of data storage and retrieval in the disk operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. See also data set.

file ID block (FIDBLK): A block for each opened file to map a DOS OPEN DTF to an OS OPEN DCB.

foreground partition: In DOS, storage is divided into three partitions. The partitions with highest and next-to-highest priority are foreground partitions.

forms-control buffer: The forms-control buffer (FCB) is 180 positions and is used to store vertical formatting information. Each position corresponds to a line on the form. The FCB is addressed in synchronism with carriage movement. Channel codes used for skipping control are stored at the FCB addresses corresponding to the desired line positions on the form. A flag bit off or on at FCB address 1 determines six or eight lines per inch vertical spacing respectively. The flag bit on at an address other than 1 identifies the corresponding line as the last line on the form. The FCB is loaded by the "load FCB" command.

forms-control image: A term applied to the Emulator. (See FCB image.)

hard wait: A condition in which the system is placed in the wait state with all interruptions disabled.

IIVCON: A control section containing data constants common to most Emulator modules. See also EMUCONS.

IIVRCN: A control section cintaining data contents common to the service aids modules. See also RASCONS.

image: See FCB image.

initial program load (IPL): The initialization procedure that loads the nucleus or supervisor and begins normal operation.

input/output block (IOB): A block used for communication between the problem program and the system; provides the addresses of the other control blocks and maintains information about the channel program, such as the type of chaining and the progress of I/O operations.

integrated Emulator: An Emulator designed to be executed under control of a system control program in a multiprogramming environment.

interface: A shared boundary, such as OS and DOS control blocks that are modified or examined by the Emulator.

ISAM block (ISBLK): A table built by the Emulator when a DTFIS is opened and used at I/O macro time to map to the OS data set.

ISK/SSK table: Contains one 1-byte entry for each 2K block of DOS dynamic storage.

job file control block (JFCB): A control block holding the internal format of an OS DD statement.

job (JOB) statement: The control statement that identifies the beginning of a series of job control statements for a single job.

library: A collection of related files (DOS) or data sets (OS).

load modules: Reside on a link or job library for the purpose of being loaded into main storage.

local address: An unadjusted DOS main-storage address.

local execution list: The area of main storage that contains data that is pertinent to the execute local instruction (see Appendix).

local execution mode: That state the CPU enters when the execute local instruction is issued (see Appendix).

local execution PSW: Effective DOS current PSW when CPU is in local execution mode.

logical unit block (LUB): A DOS control block that associates a DOS logical unit with its corresponding physical unit.

map: Ao establish a correspondence between the elements of one set and the elements of another set, such as to assign to every DOS device a device of the same type or a DOS macro to an OS macro.

module: An entity associated with a single assembly or compilation. The smallest unit used for distribution and maintenance purposes, for example, the set of statements in some source language that is compiled, or the resulting code.

normal mode: That state of the CPU during execution of OS or Emulator programs.

open table: A common section located in the B-transient area for all DOS OPEN/CLOSE phases initialized by phase 1 of the OPEN or CLOSE monitor.

OS PUB table: Contains a 1-byte entry for each DOS PUB that functions as an index into COMTAB.

page table: A table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

page translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set.

partition: See region.

password data set: An OS data set that is protected from other users by means of a password that must be given to the operating system at the time the data set is accessed.

phase: Under DOS, the smallest complete unit that can be referred to in the core image library. Each overlay of a program or the program itself, if it contains no overlays, is a single, complete phase.

physical unit block (PUB): A DOS control block that contains information pertinent to the device characteristics of an input/output unit.

post ECB list: Contains one 2-byte entry for each entry in COMTAB, the first of which is an index into COMTAB and the second the completion code from the ECB when it was last posted.

preformatted DOS file: The capacity records are reset to reflect empty tracks on the file of a disk pack or data cell. This is done in preparation for creating a new file.

private volume: In OS, a mounted volume that the system can allocate to an output data set only when a specific volume request is made. A private volume is demounted after its last use in a job step.

privileged instruction: An instruction that can only be executed when the CPU is in supervisor state.

problem program: Any program that is not part of the Operating System: a routine that solves problems, sorts and merges records, performs computations, etc., as opposed to a control program or a language translator.

processing program: A general term for any program that is not a control program.

program event recording (PER): A hardware feature used to assist in debugging programs by detecting program events.

program information block (PIB): A DOS control block that contains information pertinent to DOS task management.

program status word (PSW): A doubleword in main storage used to control the order in which instructions are executed and to hold and indicate the status of the system in relation to a particular program.

prompt: A special Emulator message that allows the operator to communicate with the Emulator at any time.

protection key: An indicator, associated with a task, that appears in the program status word whenever the task is in control, and that must match the storage keys of all storage blocks it is to use.

public volume: In OS, a mounted volume that the system can allocate to an output data set for which a nonspecific volume request is made. A public volume remains mounted until the device on which it is mounted is required by another volume.

RASCONS: A dummy section (DSECT) containing data contents common to the service aids modules.

reader/interpreter: The part of the control program that controls the reading, transcription, and interpretation of an input job stream.

region: An area of main storage set aside for a job.

resident volume (DOSRES): The direct-access volume in which the DOS IPL program and supervisor reside.

resume PSW: The PSW DOS continues with whether or not modified by the Emulator, i.e., the local execution PSW.

root segment: That segment of an overlay program that remains in main storage at all times during the execution of the overlay program; the first segment in an overlay program.

routine: An ordered set of instructions that may have some general or frequent use. A module may consist of one or several routines.

segment table (SGT): A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

segment table entry (STE): An entry in the segment table that indicates the length, location, and availability of a corresponding page table.

segment translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set.

shared data set: When other OS job steps may access a data set concurrently with the Emulator.

shared volume: When other OS job steps may access a volume concurrently with the Emulator.

simulation (as used in this publication): The programming techniques used in the Emulator to produce, as nearly as possible, the results that DOS instructions and hardware functions produce in a stand-alone DOS environment.

snap dump (as used in this publication): A selective dynamic formatted dump performed at various points in a program run.

soft wait: That state the CPU enters when the wait bit of the current PSW is set to one and interruptions are enabled.

spooling: All primary input streams are read from an input device and temporarily stored on a DASD in a format convenient for later processing by the Operating System and user programs. System and selected user print and punch output are similarly stored on a DASD until a convenient time for writing hard copy.

staged I/O: The ability to request input/output operations without regard to the characteristics of the input/output devices; partitions compete for these devices on a priority basis or on a first-in, first-out determinant.

staged I/O constants block (STGCON): The block residing in the IOB area for staged devices that contains data unique to the particular device being staged.

stand-alone DOS: The disk operating system as it functions in its own environment (contrasted to DOS when it is emulated).

stand-alone emulator: An Emulator that can be executed only on a computing system totally dedicated to that program.

subroutine: A routine that can be part of another routine.

supervisor: The medium through which the use of resources is coordinated and the flow of operations through the CPU is maintained; hence, a control program that executes in the supervisor state.

supervisor call (SVC): An instruction that causes an SVC interruption in the hardware to give control to a control program routine (called an SVC routine) for some specific action, such as reassigning parts of main storage or retrieving data from an I/O device. This interruption mechanism differs when the program is running in local execution mode. See the Appendix for details.

synchronous interruptions: Interruptions that occur with a regular or predictable time relationship (program check, SVC).

SYS1.IMAGELIB: The OS system library in which UCS and FCB images reside.

SYS1.LINKLIB: The OS system library that contains executable programs and modules.

tape error block (TEB): A DOS control block that contains information pertinent to error recovery processing.

tape error block by volume (TEBV): This is a DOS table composed of one status block and (n) error blocks and pointed to by the TEBVTAB field in the DOS BG COMREG extension.

task: A unit of work for the central processing unit from the standpoint of the control program, that is, the basic multiprogramming unit under the control program.

task I/O table (TIOT): A table that contains information pertinent to input and output processing. One TIOT entry is created by the Operating System for each DD statement supplied in the job step.

translation specification exception: A program interruption that occurs when a page table entry, segment table entry, or the control register pointing to the segment table contains infcrmation in an invalid format.

trap (as used in this publication): Intercepting a given DOS event and recognizing certain specific DOS or Emulator conditions.

true address: An actual main-storage address (OS) within the Emulator region.

unit control block (UCB): An OS control block that contains information pertinent to the device characteristics of an input/output unit.

virtual storage: Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism, for example, a drum or disk pack, or a recording medium that is mounted and demounted as a unit, for example, a data cell.

volume label: Uniquely identifies the volume.

volume table of contents (VTOC): A table, associated with a direct-access volume, that describes each data set on the volume.

write to operator (WTO): A macro instruction that causes a message contained within the macro to be written to the operator's console.

write to operation with reply (WTOR): A macro instruction that causes a message contained within the macro to be written to the operator's console. An operator reply is required.

Indexes to program logic manuals are consolidated in the publication OS Master Index to Logic Manuals, GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

A

B

C

D

E

F

G

H

I

S

T

U

V

W

Y

Order Number GY26-3741-3

DOS Emulator Logic Printed in USA Order Number GY26-3741-3

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)

READER'S COMMENT FORM

DOS Emulator Logic

Order Number GY26-3741-3

Your comments about this publication will help us to produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name _____

Job Title _____

Address _____

_____ Zip _____

No postage necessary if mailed in the U S A

Order Number GY26-3741-3

YOUR COMMENTS, PLEASE . . .

This publication is one of a series which 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.

fold fold

FIRST CLASS
PERMIT NO. 1359
White Plains, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Department 813 (LGP)
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Programming Publications, Dept. 813 (LGP)

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)

DOS Emulator Logic Printed in USA Order Number GY26-3741-3

IBM Technical Newsletter

File Number	S360-35
Base Publication Number	GY26-3741-3
This Newsletter Number	GN26-8021
Date	August 14, 1972
Previous Newsletter Number(s)	None

DOS EMULATOR LOGIC (on IBM System/370 Under OS)

This Technical Newsletter provides replacement pages for the DOS Emulator Logic publication, Order Number GY26-3741-3.

These replacement pages remain in effect for subsequent versions of the DOS Emulator program unless specifically altered. Pages to be inserted and/or removed are listed below:

title page - xii
xix-xxii
1-2
7-14
27-28
395-398
417-418
421-422
467-468
477-478
489-490
521-540
Reader's Comment Forms (new mailing address)

A change to the text is indicated by a vertical line to the left of the change.

Summary of Amendments

The DOS Emulator program, which runs under OS/MFT and OS/MVT, now runs under OS/VS1 as well. The support of OS/VS1 is reflected in these replacement pages. Any reference in this manual to main storage pertains to virtual storage for OS/VS.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Programming Publications, Department D78, San Jose, California 95114

PRINTED IN U.S.A.