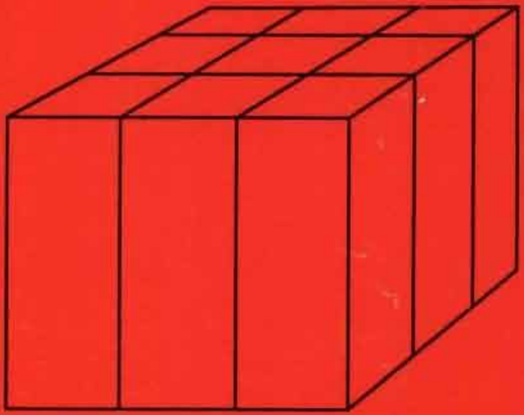
IBM Virtual Storage Extended/Advanced Functions Diagnosis Reference

Serviceability Aids



IBM Virtual Storage Extended/Advanced Functions Diagnosis Reference

Serviceability Aids

Program Number 5666-301

Order Number LY33-9115-01
File No. S370/4300-37

Second Edition (January 1987)

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

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this document is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

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

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

IBM Corporation
Dept. 6R1BP
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

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

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

PREFACE

IBM serviceability aid programs supplement various supervisor and transient functions. The programs provide the means to trace specific events during execution of a program, assist in software debugging, generate dumps, and list error statistics.

This publication is intended primarily for use by IBM personnel responsible for program service. The publication gives an overview of the functions of the serviceability aids, of their design, and their organization; it supplements the program listings.

Related Publications

Functional descriptions of the serviceability programs in this manual, as well as some not described here, are included in the following publication:

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

The reader should also be familiar with:

IBM 4300 Processors Principles of Operation, GA22-7070, or
IBM System/370 Principles of Operation, GA22-7000, and
VSE/Advanced Functions System Management Guide, SC33-6191
VSE/Advanced Functions Planning and Installation, SC33-6193
VSE/Advanced Functions System Control Statements, SC33-6198
VSE/SP Diagnosis Guide, SC33-6112

Information on how to use Info/Analysis as an online dump management tool may be found in the following publications:

Interactive System Productivity Facility General Information, GC34-2078
Interactive System Productivity Facility for VSE Installation and Customization, SC34-2080
Interactive System Productivity Facility Dialog Management Services, SC34-2088
Interactive System Productivity Facility Dialog Management Services Examples, SC34-2085
Interactive System Productivity Facility Diagnosis, SC34-2132

For complete information on VSE/Advanced Functions program logic, consult the following VSE/Advanced Functions Diagnosis Reference publications:

Supervisor, LY33-9107.

Error Recovery and Recording Transients, LY33-9108.
Logical Transients and $IJBSxxx Phases, LY33-9109.
Initial Program Load and Job Control, LY33-9110.
Linkage Editor, LY33-9112.
Librarian, LY33-9111.

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

Organization of the Manual

The manual consists of the following sections:

- Part 1: SDAID (System Debugging Aid)
- Part 2: Dump Utility Programs (DOSVSDMP)
- Part 3: Dump Analysis Programs (IJBXDBUG and IJBXSDA)
- Part 4: Info/Analysis
- Part 5: Parser (IJPARSER)
- Part 6: LSERV (Label Information Area Display)

CONTENTS

FIGURES

SDAID: INTRODUCTION

The SDAID program provides for VSE users, including IBM programming support, the tracing facilities needed for efficient problem analysis. The program allows the user to set up ten different traces per SDAID session (execution) with a variety of trace output being produced.

The following types of events may be traced:

BRanch	successfully executed branch instructions
CAncel	program (main task) cancel or EOJ
EXTernal	external interruptions
INSTruction	instruction(s) execution
IO	I/O interrupts
MONitorcall	MC instruction execution
PAGing	page fault interruptions
PGMCheck	program check interruptions
PGMLoad	phase load request or actual load
REGister	register alterations
SIO	SIO instruction execution
STorage	storage alterations
SVC	Supervisor calls
VTAMBU	buffer usage in ACF/VTAM
VTAMIO	VTAM I/O related events

The trace output of SDAID may be written to a line printer, to a tape device, or into a wrap-around buffer. If trace data is written into a wrap-around buffer then the user may specify an output device (printer or tape) to give out the contents of the wrap-around buffer on explicite request.

Any SDAID session consists of two major steps: The first step is the SDAID initialization (SDAID set-up) where the requested traces are prepared. The SDAID initialization is controlled by the commands SDAID, OUTDEV, TRACE, and READY. The second step is the SDAID execution (event tracing) where trace data is collected. The SDAID execution is controlled by the commands STRTSD, STOPSD, and ENDSD.

The following overview describes the function of the SDAID commands:

SDAID invokes the SDAID program. It must be the first command submitted for an SDAID session.

OUTDEV defines the device on which SDAID is to record trace information. The command is mandatory; it must precede the READY command.

TRACE defines the program events whose occurrences are to be traced; the command must precede the READY command.

READY signals to SDAID that all control input from the console is complete. The READY command must be preceded by an OUTDEV command and at least one TRACE command.

STRTSD requests SDAID to start executing the requested trace operation(s).

STOPSD requests SDAID to stop executing the requested trace operation(s).

ENDSD terminates the current SDAID session by freeing all resources that were used by the SDAID program. ENDSD may be given at any time after an SDAID command has been issued. The ENDSD command may be used to flush the set-up process or to terminate the SDAID execution.

For more information about the SDAID commands, see VSE/Advanced Functions Diagnosis: Service Aids.

SDAID: PROGRAM DESIGN AND ORGANIZATION INFORMATION

INITIALIZATION

The SDAID set-up process may be invoked from a user partition (set-up via SYSIN) or via the attention supervisor task (set-up via attention commands). If the set-up via SYSIN is used, the phase SDAID is called (directly or via a JCL procedure) and that phase submits the commands SDAID, OUTDEV, TRACE, and READY to the SDAID adapter phase $$BATTN3. If the SDAID set-up is done via the attention task, then the attention commands SDAID, OUTDEV, TRACE, and READY are passed from the LTA phase $$BATTNA to the SDAID adapter phase $$BATTN3.

$$BATTN3 loads the SDAID message library IJSDMSG and the initialization manager IJSDROT into the system GETVIS area if they are not already there.

$$BATTN3 now invokes the initialization manager IJSDROT. This phase determines the entered command, checks for proper command sequence, loads all needed phases and tables into the system GETVIS area and invokes the parser, phase IJPARSER, for command syntax checking.

When the READY command is issued, then the initialization manager IJSDROT determines which execution phases are to be loaded to satisfy all specified trace functions. In /370 mode the execution phases are loaded into real storage taken from a reserved area of at least 48k bytes between the end of the supervisor and the begin address of the private partitions. The size of that reserved SDAID area may be increased during IPL using the SDSIZE parameter of the SYS command. In ECPS:VSE mode the execution phases are loaded into the system GETVIS area and are PFIXED.

After the STRTSD command is entered, the new PSWs and the control registers are modified so that SDAID is enabled to trace the specified events.

IJSDAID - SYSIN COMMAND PROCESSOR

Function:
Prepares complete SDAID commands and transfers them to the SDAID adapter phase $$BATTN3.

Phase Name:
SDAID

Module Name:
IJSDAID.

Entry point:
IJSDAID.

Input:
OUTDEV and TRACE commands from SYSIN or SYSLOG and procedure parameters from the SVA.

Output:

```
SDAID commands: SDAID, OUTDEV, TRACE,
                READY, ENDSD.
```

These commands are submitted to $$BATTN3.
For information they are displayed on SYSLOG, too.

Phases called: $$BATTN3
Parameters passed to $$BATTN3:
Register 0 contains the address of an interface area, called BATTNA.

Macros used:
DTFCN, DTFDI

Sequence of Operation
Reads input parameters from SYSIPT or SYSLOG.
If substitution of parameters is required retrieves SVA parameters via PARMMAC macro.
Substitutes missing parameters by default values.
Passes SDAID commands to $$BATTN3.
Prepares an OUTDEV command (if required).
Writes information messages to SYSLOG or SYSLST.
On errors issues ENDSD command to flush the set-up process.
Returns to JCL via RETURN macro.

Internal Subroutines

SDOUTD:
Prepare the OUTDEV command
retrieve PRINTER, TAPE, BUFFER parameter

SDBUTERM:
Process BUFFOUT parameter

SDCONS:
Display SDAID commands
Pass commands to $$BATTN3.

SDENDSD:
Prepare ENDSD command.

SDTRAREA:
Retrieve AREA parameter.

GETVAL:
Retrieve parameters from SVA (via PARMMAC)

BUFFOUT:
Move parameter to output buffer.

SDINPUT:
Read input from SYSLOG or SYSIPT.
Scan input area.
Substitute placeholders by retrieved
parameters or by default values.

Information Messages:

The messages 4C40I to 4C61I are prepared by IJSDAID.
They are printed|displayed by IORTN ($$BATTN3).

Normal exit:

to JCL via the RETURN Macro.

Error exit:

to JCL via the RETURN Macro.

$$BATTN3 - SDAID TO SYSTEM ADAPTER

Function:

This phase has 2 functions:
1. calls the phase IJSDROT if an SDAID command is entered.
2. performs all I/O operations for SYSLOG and SYSLST.

Entry point:

This phase has 2 entry points:
1. $$BATTN3: entry to process an SDAID command.
2. IORTN: entry to perform the I/O operations.

Input:

For function 1:
The address of the interface area BATTNA is contained in register 15 (if $$BATTN3 is called from $$BATTNA) or in register 0 (if $$BATTN3 is called from IJSDAID). BATTNA contains the SDAID command and flag bytes.

For function 2:
An indicator for the requested function (READ/WRITE), the address of the I/O area.

Output:

For function 2: The I/O operation required.

Normal exit:

For function 1: returns to $$BATTNA or IJSDAID.
For function 2: returns to calling phase.

Error exit:

Same as normal exit.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: $$BATTN3

```
Internal                      External
routines                      routines  Function
--------------------------------------------------------------------------

$$BATTN3                                Controls processing of an SDAID command
                                            (PAGE 6).

  |
  |
  |-->SDLOAD                            Loads phases IJSDMSG and IJSDROT.
  |
  |-------------------------->IJSDROT   Main entry all SDAID commands (page 8).
  |
  |-->ENDSD                             Releases IJSDMSG and IJSDROT.
  |
  +-->IOSMOD                            I/O for SYSLOG and SYSLST.
```

IJSDROT - SDAID INITIALIZATION MANAGER

Function:

The SDAID initialization manager loads all data areas, control blocks, and phases needed to process an SDAID command.

Entry point:

Same as module name.

Input:

Command name and specified operands.

Output:

Updated information in control blocks.

Normal exit:

Returns to phase $$BATTN3.

Error exit:

Same as normal exit.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDROT

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDROT                                 Main entry all SDAID commands (PAGE 8).

  |
  |
  |-->ROTOUTD                           Processes the OUTDEV command.
  |
  |     |
  |     |-------------------->IJSDCT2   Sets up OUTDEV cmd directory (page 17).
  |     |-------------------->IJSDPIF   Checks command syntax (page 23).
  |     |-->ROTFREV                     Releases storage in SVA.
  |     |
  |     |     |
  |     |     +-------------->IJSDMSG   Writes an error message (page 14).
  |     +-->ROTGETS                     Computes size of an SDAID phase.
  |
  |           |
  |           +-------------->IJSDMSG   Writes an error message (page 14).
  |
  |-->ROTTRA                            Processes the TRACE command.
  |
  |     |
  |     |-------------------->IJSDCT3   Sets up TRACE command directory (page
  |     |8).
  |     |-------------------->IJSDPIF   Checks command syntax (page 23).
  |     |-->ROTFREV                     Releases storage in SVA.
  |     |
  |     |     |
  |     |     +-------------->IJSDMSG   Writes an error message (page 14).
  |     +-->ROTGETS                     Computes size of an SDAID phase.
  |
  |           |
  |           +-------------->IJSDMSG   Writes an error message (page 14).
  |
  |-->ROTRDY                            Processes the READY command.
  |
  |     |
  |     |-->ROTFRAL                     Releases initialization phases.
  |     |
  |     |     |
  |     |     +-->ROTFREV               Releases storage in SVA.
  |     |
  |     |
  |     |
|     |
|     |           |
|     |           +-------->IJSDMSG   Writes an error message (page 14).
|     |-->ROTFREV                     Releases storage in SVA.
|     |
|     |     |
|     |     +-------------->IJSDMSG   Writes an error message (page 14).
|     |-->ROTGETS                     Computes size of an SDAID phase.
|     |
|     |     |
|     |     +-------------->IJSDMSG   Writes an error message (page 14).
|     |-->ROTGETR                     Gets real storage from page pool.
|     |
|     |     |
|     |     +-------------->IJSDMSG   Writes an error message (page 14).
|     |-->ROTGETV                     Gets storage from SVA.
|     |
|     |     |
|     |     +-------------->IJSDMSG   Writes an error message (page 14).
|     |-->ROTLOAD                     Loads an SDAID phase into SVA.
|     |-------------------->IJSDINT   Establishes new PSWs in EDTCB (page 27).
|     |-------------------->IJSDIOS   Processes I/O interruptions (page 31).
|     +-------------------->IJSDRDY   Processes the READY command (page 26).
|
|-->ROTSDAID                          Processes the READY command.
|
|     |
|     |-------------------->IJSDEDT   Allocates execution ctl blocks (page
|     |3).
|     |-------------------->IJSDIDT   Builds initialization table (page 12).
|     |-------------------->IJSDTCB   Manages TCB queue (page 15).
|     +-->ROTGETS                     Computes size of an SDAID phase.
|
|           |
|           +-------------->IJSDMSG   Writes an error message (page 14).
|
|-->ROTSTOP                           Processes the STOPSD command.
|
|     |
|     +-------------------->IJSDMSG   Writes an error message (page 14).
|
|-->ROTSTRT                           Processes the STRTSD command.
|
|     |
|     +-------------------->IJSDMSG   Writes an error message (page 14).
|
+-->ROTENDSD                          Processes the ENDSD command.
```

```
|
|-->ROTSTOP                     Processes the STOPSD command.
|
|     |
|     +-------------->IJSDMSG   Writes an error message (page 14).
|-->ROTFRER                     Releases storage in page pool.
|-->ROTFREV                     Releases storage in SVA.
|
|     |
|     +-------------->IJSDMSG   Writes an error message (page 14).
|-->ROTPFREE                    Frees pages in SVA.
|
|     |
|     +-------------->IJSDMSG   Writes an error message (page 14).
|-->ROTFREV                     Releases storage in SVA.
|
|     |
|     +-------------->IJSDMSG   Writes an error message (page 14).
+-->ROTFRAL                     Releases initialization phases.

      |
      +-->ROTFREV               Releases storage in SVA.

            |
            +-------->IJSDMSG   Writes an error message (page 14).
```

IJSDIDT - SDAID INITIALIZATION DATA TABLE

Function:

This phase allocates all data areas needed during SDAID initialization. The data areas are cleared and anchored in the IDTCB.

Input:

Parameters: none.

Output:

Formatted Initialization Data Table (IDTCB).

Normal exit:

Returns to calling phase.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDIDT

Internal routines	External routines	Function
IJSDIDT		Builds initialization table (PAGE 12).

IJSDEDT - SDAID EXECUTION DATA TABLE

Function:

This phase allocates all data areas needed during SDAID execution. The data areas are cleared and anchored in the EDTCB.

Entry point:

Same as module name.

Input:

None.

Output:

Initialized control blocks.

Normal exit:

Returns to the calling module IJSDROT.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDEDT

```
Internal                       External
routines                       routines  Function
-----------------------------------------------------------------------------

IJSDEDT                                  Allocates execution ctl blocks (PAGE
                                             13).
```

IJSDMSG - MESSAGE LIBRARY AND MESSAGE WRITER

Function:

The SDAID message library contains all SDAID messages. It also contains the message manager, which prepares the messages and writes them on SYSLOG and SYSLST.

Entry point:

Same as module name.

Input:

A parameter list is submitted in register 1. This list contains the message number to be processed, possibly the address and length of text to be inserted in predefined slots, and an identifier for the end of the list.

Output:

The requested message written on SYSLOG and SYSLST.

Exit:

Returns to the calling routine.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDMSG

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDMSG                                 Writes an error message (PAGE 14).

  |
  |
  |-->MSGLOOK                           Searches message library.
  |
  +-->MSGOUT                            Prepares and display message.

        |
        |-->MSGUPD                      Inserts text in message slots.
        +-------------------->IOSMOD    I/O for SYSLOG and SYSLST
```

IJSDTCB - TRACE CONTROL BLOCK MANAGER

Function:

The SDAID trace control block manager has the following functions:

- Initializes trace table (TRTCB), trace control block queue (TCBCBQ), and trace control block extension queue (TCECBQ).
- Enqueues a TCB in its appropriate chain.
- Appends a TCB to a master TCB and enqueue it in its appropriate TCB chain.
- Appends a TCE to a master TCB and enqueue it in its appropriate TCE chain.

Entry point:

Same as module name.

Input:

Parameters: interface established in EDTCB.

Output:

Updated trace table (TRTCB) and trace control block queue.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDTCB

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDTCB                                 Manages TCB queue (PAGE 15).

  |
  |
  |-->TCBAPP                            Appends a TCB.
  |
  |-->TCBENQ                            Enqueues a TCB.
  |
  |     |
  |     |-------------------->IJSDMSG   Writes an error message (page 14).
  |     +-->TCBAPP                      Appends a TCB.
  |
  +-->TCBINI                            Initializes TCB-QUEUE.
```

IJSDCT2 - OUTDEV COMMAND TABLE

Function:

The SDAID OUTDEV command table contains the definitions of nodes and parameters for the OUTDEV command. (See "Parser" chapter). It also contains a routine to establish the command directory in the initialization data table (IDTCB).

Entry point:

Same as module name.

Input:

IDTPTR contains the address of the initialization data table IDTCB.

Output:

The command directory established in IDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDCT2

Internal routines	External routines	Function
IJSDCT2		Sets up OUTDEV cmd directory (PAGE 17).

IJSDCT3 - TRACE COMMAND TABLE

Function:
The SDAID TRACE command table contains the definitions of the nodes and parameters for the TRACE command. (See "Parser" chapter). It also contains a routine to establish the command directory in the initialization data table (IDTCB).

Entry point:
Same as module name.

Input:
IDTPTR contains the address of the initialization data table IDTCB.

Output:
The command directory established in IDTCB.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDCT3

Internal routines	External routines	Function
IJSDCT3		Sets up TRACE command directory (PAGE 18).

IJSDOUT - SDAID OUTDEV COMMAND PROCESSOR

Function:

The SDAID OUTDEV command processor checks the syntax and the semantics for the OUTDEV command.
It saves the specified parameters in the execution data table.

Entry point:

Same as module name.

Input:

CPCB: IJPARSER control block containing the SIF for the OUTDEV command.

Output:

Updated control block EDTCB according the specified parameters of the OUTDEV command.

Normal exit:

Returns to calling routine.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDOUT

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDOUT                                 Processes the OUTDEV command (PAGE 19).

  |
  |
  |-->SDOUT01                           Processes primary output device.
  |
  |-->SDOUT02                           Processes size of wrap around buffer.
  |
  |-->SDOUT03                           Processes secondary output device.
  |
  |     |
  |     +--------------------->IJSDMSG   Writes an error message (page 14).
  |
  +-->SDOUT04                           Validates PRINTER/TAPE device.

        |
        |--------------------->IJSDMSG   Writes an error message (page 14).
        |
        |
```

IJSDTRA - SDAID TRACE COMMAND PROCESSOR

Function:

The SDAID TRACE command processor builds for each trace command a TRACE control block (TCBCB) and saves all specified parameters and values in it.

Entry point:

Same as module name.

Input:

CPCB: IJPARSER control block containing the SIF for the TRACE command.

Output:

Updated control blocks EDTCB and TCBCB according the specified parameters of the TRACE command.

Normal exit:

Returns to the calling routine.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDTRA

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDTRA                                 Processes the TRACE command (PAGE 21).

  |
  |
  |-->INIT                              Initializes work areas.
  |
  |-->PARMS                             Processes all parameters.
  |
  |-->CHKTCB                            Determines number of TCBs.
  |
  |     |
  |     +-->TRAEMSG
  |
  |-->ENQTCB                            Enqueues all TCBs.
  |
  |     |
  |     |-------------------->IJSDTCB   Manages TCB queue (page 15).
  |     +-->TRAEMSG
  |
  |-->ENQTCE                            Enqueues all TCEs.
  |
  |     |
  |     +-------------------->IJSDTCB   Manages TCB queue (page 15).
  |
  +-->ENQAPP                            Enqueues appendage TCEs.
```

IJSDPIF - SDAID-TO-IJPARSER INTERFACE

Function:

This phase is the only communication medium from the PARSER to the SDAID program.

Entry points:

IJSDPIF: called by IJSDROT to transfer control to the PARSER for syntax checking.
PIFCHK: called by the PARSER to transfer control to the processing routine for the actual command.
PIFIOF: called by the PARSER for I/O processing.

The entry points PIFCHK and PIFIOF are set in the Parser Control Block CPCB and only used by the PARSER.

Input:

For entry PIFCHK: the address of the SIF entry for which a semantic processing is to be done.
For entry PIFIOF: for write request- the text to be written in IJPARSER I/O area in IDTCB.

Normal exit:

For entry IJSDPIF: as provided by the PLS/II return statement
For entry PIFCHK: to IJPARSER.
For entry PIFIOF: to IJPARSER.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDPIF

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDPIF                                 Checks command syntax (PAGE 23).

  |
  |
  |-------------------------->PARSER
  |
  |-->PIFRC0                            Processes IJPARSER return code 0.
  |
  |     |
  |     |-->CONTINU                     Enlarges I/O area.
  |     |
  |     |     |
  |     |     |-->PIFFREV               Releases storage in SVA.
  |     |     |
  |     |     |     |
  |     |     |     +-------->IJSDMSG   Writes an error message (page 14).
  |     |     +-->PIFGETV               Gets storage from SVA.
  |     |
  |     |           |
  |     |           +-------->IJSDMSG   Writes an error message (page 14).
  |     |-------------------->IJSDMSG   Writes an error message (page 14).
  |     |-------------------->IJSDOUT   Processes the OUTDEV command (page 19).
  |     |-------------------->IJSDTRA   Processes the TRACE command (page 21).
  |     +-------------------->IOSMOD    I/O for SYSLOG and SYSLST
  |
  +-->PIFRC4                            Processes IJPARSER return code 4.

        |
        +-------------------->IOSMOD    I/O for SYSLOG and SYSLST
```

FLOW OF CONTROL FOR ENTRY POINT: PIFCHK

```
Internal                  External
routines                  routines  Function
-----------------------------------------------------------------

PIFCHK                              Processes semantic checking.

  |
  |
  |----------------------->IJSDOUT  Processes the OUTDEV command (page 19).
  |
  +----------------------->IJSDTRA  Processes the TRACE command (page 21).
```

FLOW OF CONTROL FOR ENTRY POINT: PIFIOF

```
Internal                  External
routines                  routines  Function
-----------------------------------------------------------------

PIFIOF                              Performs I/O for IJPARSER.

  |
  |
  |----------------------->IOSMOD   I/O for SYSLOG and SYSLST
  |
  +-->PIFHELP

       |
       +------------------>IOSMOD   I/O for SYSLOG and SYSLST
```

IJSDRDY - READY COMMAND PROCESSOR

Function:

Scans the Trace Control Block queues (TCB queues).
Prepares the PER mask, PER range and low core PSWs.
These values will be inserted into low core locations
x'58' (Ext new PSW), x'60' (SVC new PSW),
x'68' (program new PSW), x'78' (I/O new PSW)
and into the control registers 8 to 11 at STRTSD time.

Entry point:

Same as module name.

Input:

None.

Output:

Updated information in control blocks EDTCB and GDTCB.

Normal exit:

Returns to the calling routine.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDRDY

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDRDY                                 Processes the READY command (PAGE 26).

  |
  |
  |-------------------------->IJSDMSG   Writes an error message (page 14).
  |
  |-->PERADDR                           Determines PER range.
  |
  +-->PERPID                            Saves IDs of PER partitions.
```

EVENT RECORDING ROUTINES

IJSDINT - INTERRUPT PROCESSOR

Function:

The SDAID interrupt processor controls the processing of those program events which are specified to be traced. It invokes the pertinent event processor in order to generate the trace output.

Entry points:

IJSDINT: Establishes interrupt addresses
         (called from IJSDROT)
EXTENT: entry for EXT interrupts
IOENT: entry for I/O interrupts
PGMENT: entry for PGM interrupts
SVCENT: entry for SVC interrupts

Input:

The interrupt information in fixed storage locations.

Output:

It is done by the invoked procedures.

Normal exit:

Returns to the interrupted program via LPSW-instruction.

Error exit:

Not applicable.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: SVCENT

```
Internal                       External
routines                       routines  Function
-----------------------------------------------------------------------------

SVCENT                                   Entry point for SVC interrupts.

  |
  |
  |-------------------------->IJSDZSV   Processes SVC events (page 58).
  |
  |-------------------------->IJSDZPL   Processes program load events (page 52).
  |
  +-->COMMON                             Closes OUTDEV and returns to SUP.
```

FLOW OF CONTROL FOR ENTRY POINT: EXTENT

```
Internal                       External
routines                       routines  Function
-----------------------------------------------------------------------------

EXTENT                                   Entry point for EXT interrupts.

  |
  |
  |-------------------------->IJSDZEX   Processes EXTERNAL interrupts (page 39).
  |
  +-->COMMON                             Closes OUTDEV and returns to SUP.
```

FLOW OF CONTROL FOR ENTRY POINT: IOENT

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IOENT                                   Entry point for I/O interrupts.

  |
  |
  |-------------------------->IJSDIOS   Processes I/O interruptions (page 31).
  |
  |-------------------------->IJSDZIO   Processes I/O events (page 45).
  |
  +-->COMMON                            Closes OUTDEV and returns to SUP.
```

FLOW OF CONTROL FOR ENTRY POINT: PGMENT

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------

PGMENT                                  Entry point for PGM interrupts.

  |
  |
  |------------------------->IJSDZBR   Generates Branch trace record (page 33).
  |
  |------------------------->IJSDZBU   Proc. BUFFER OVERFLOW events (page 35).
  |
  |------------------------->IJSDZCA   Processes CANCEL events (page 37).
  |
  |------------------------->IJSDZGA   Proc. REGISTER alter events (page 41).
  |
  |------------------------->IJSDZIN   Proc instruction fetch events (page 43).
  |
  |------------------------->IJSDZMC   Processes monitor call events (page 47).
  |
  |------------------------->IJSDZPA   Processes program load events (page 49).
  |
  |------------------------->IJSDZPC   Processes program check event (page 50).
  |
  |------------------------->IJSDZPL   Processes program load events (page 52).
  |
  |------------------------->IJSDZSA   Proc. storage alter events (page 54).
  |
  |------------------------->IJSDZSI   Processes SIO events (page 56).
  |
  |------------------------->IJSDZVT   Processes VTAMIO events (page 60).
  |
  +-->COMMON                            Closes OUTDEV and returns to SUP.
```

IJSDIOS - I/O INTERRUPT PROCESSOR

Function:

Processes I/O Interruptions.
Stacks I/O Interrupts and presents them to the supervisor when the supervisor is able to handle them.
If a unit check occurs, then retrieves SENSE data and stacks it, too.

Entry points:

IJSDIOS (called from IJSDROT)
Initializes the I/O stack.

SDAIOM (called by IJSDINT)
Presents a stacked I/O interrupt to the I/O supervisor of VSE.
This is a response to a special monitor call from the supervisor requesting I/O interrupts.

SDAIOP (called by IJSDINT)
Puts interrupt info into I/O stack.
If unit check, then stacks SENSE info, too.

SDAIOI (directly entered on I/O interrupts.)
During I/O on OUTDEV the I/O new PSW in low core points to SDAIOI)..
Puts interrupt info into I/O stack.
If unit check, then stacks SENSE info, too.

Input:

None.

Output:

None.

Normal exit:

Returns to the calling procedure.
Returns to the interrupted program (if entered at SDAIOI).

IJSDSTP - SDAID STOP-ON-EVENT PROCESSOR

Function:

Stops the system after an event has occurred and is processed by SDAID.
The system is put into a wait state with '00EEEE' is the address part of the wait PSW. To go out of the wait state, the operator must press the external interrupt key.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table EDTCB.

Output:

None.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDZBR - SDAID BRANCH TRACE PROCESSOR

Function:

The SDAID Branch Trace processor traces all or selected successfully executed BRANCH instructions in a specified program area.
It generates a branch trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table EDTCB.
TRTPTR contains the address of the trace table (TRT) entry for branch traces.
TCBPTR contains the address of the first block in the TCB-queue for branch traces.

Output:

The branch trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZBR

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZBR                                 Generates Branch trace record (PAGE 33).

  |
  |
  |-->AREAVAL                           Determines event area.
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->RECGEN1                           Generates the BR trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        |-------------------->IJSDWRP   Determines I/O function (page 64).
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        +-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
```

IJSDZBU - SDAID BUFFER OVERFLOW PROCESSOR

Function:

The SDAID buffer overflow trace processor traces all buffer overflow conditions.
It writes the contents of the SDAID wrap around buffer on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the Execution Data Table (EDTCB).
TRTPTR	contains the address of the Trace Table (TRT) entry for buffer-overflow traces.
TCBPTR	contains the address of the first block in the TCB-queue for buffer overflow traces.

Output:

The wrap-around buffer written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZBU

```
Internal                      External
routines                      routines  Function
-------------------------------------------------------------

IJSDZBU                                 Proc. BUFFER OVERFLOW events (PAGE 35).

  |
  |
  |-------------------------->IJSDPWB   Prints wrap buffer and tape (page 74).
  |
  |-------------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
  |
  +-------------------------->IJSDWRT (page 66).
```

IJSDZCA - SDAID CANCEL TRACE PROCESSOR

Function:

The SDAID cancel trace processor traces all program termination conditions which occurred in a specified partition. It generates a cancel trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for cancel traces.
TCBPTR	contains the address of the first block in the TCB-queue for cancel traces.

Output:

The cancel trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZCA

```
Internal                       External
routines                       routines  Function
----------------------------------------------------------------------------

IJSDZCA                                  Processes CANCEL events (PAGE 37).

  |
  |
  |-->AREAVAL                            Determines area of event.
  |
  |-->OCCUR                              Determines occurrence of event.
  |
  |-->RECGEN1                            Generates CANCEL trace record.
  |
  +-->WRITE                              Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZEX - SDAID EXTERNAL INTERRUPT PROCESSOR

Function:

The SDAID external interrupt processor traces all external interrupts.
It generates an external interrupt trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for EXT traces.
TCBPTR	contains the address of the first block in the TCB-queue for EXT traces.

Output:

The external interrupt trace record written on the specified output device.

Normal exit:

Returns to the interrupt program via LPSW.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZEX

```
Internal                       External
routines                       routines  Function
-----------------------------------------------------------------------------

IJSDZEX                                  Processes EXTERNAL interrupts (PAGE 39).

  |
  |
  |-->OCCUR                              Determines occurrence of event.
  |
  |-->RECGEN1                            Generates the EXT trace record.
  |
  +-->WRITE                              Writes trace record to OUTDEV.

        |
        |--------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |--------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |--------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +--------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZGA - SDAID REGISTER ALTER TRACE PROCESSOR

Function:

The SDAID register alter trace processor traces all instructions which alter the contents of one or more registers. It generates a register alter trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for register alter traces.
TCBPTR	contains the address of the first block in the TCB-queue for register alter traces.

Output:

The register alter trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZGA

```
Internal                     External
routines                     routines  Function
---------------------------------------------------------------------------

IJSDZGA                                Proc. REGISTER alter events (PAGE 41).

  |
  |
  |-->AREAVAL                          Determines area of event.
  |
  |-->OCCUR                            Detemines occurrence of event.
  |
  |-->OPTION                           Determines instructions to be traced.
  |
  |-->RECGEN1                          Generates register alter trace record.
  |
  |-->REGLST                           Saves registers used in instruction.
  |
  +-->WRITE                            Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT  Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP  Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB  Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP  Determines I/O function (page 64).
```

IJSDZIN - SDAID INSTRUCTION TRACE PROCESSOR

Function:

The SDAID instruction trace processor traces all or selected instructions in a specified program area.
It generates an instruction trace and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for INSTRUCTION traces.
TCBPTR	contains the address of the first block in the TCB-queue for INSTRUCTION traces.

Output:

The instruction trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZIN

```
Internal                          External
routines                          routines  Function
---------------------------------------------------------------------------

IJSDZIN                                     Proc instruction fetch events (PAGE 43).

  |
  |
  |-->AREAVAL                               Determines area of event.
  |
  |-->OCCUR                                 Determines occurrence of event.
  |
  |-->OPTION                                Determines instructions to be traced.
  |
  |-->RECGEN1                               Generates INST trace record.
  |
  +-->WRITE                                 Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT       Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP       Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB       Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP       Determines I/O function (page 64).
```

IJSDZIO - SDAID I/O TRACE PROCESSOR

Function:

The SDAID I/O interrupt trace processor traces the interrupts from all or selected I/O units, control units or channels. It generates an I/O interrupt trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for I/O traces.
TCBPTR	contains the address of the first block in the TCB-queue for I/O traces.

Output:

The I/O interrupt trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

```
FLOW OF CONTROL FOR ENTRY POINT:  IJSDZIO

Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZIO                                 Processes I/O events (PAGE 45).

  |
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->OPTION                            Determines I/O events to be traced.
  |
  |-->RECGEN1                           Generates I/O trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |--------------------->IJSDDAT  Collects data for OUTPUT (page 67).
        |--------------------->IJSDSTP  Stops SDAID on option HALT (page 32).
        |--------------------->IJSDWRB  Writes trace rec. to BUFFER (page 62).
        +--------------------->IJSDWRP  Determines I/O function (page 64).
```

IJSDZMC - SDAID MONITOR CALL TRACE PROCESSOR

Function:

The SDAID monitor call event processor traces all or selected monitor call instructions in a specified program area. It generates a monitor call trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for monitorcall traces.
TCBPTR	contains the address of the first block in the TCB-queue for monitorcall traces.

Output:

The monitor call trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZMC

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDZMC                                 Processes monitor call events (PAGE 47).

  |
  |
  |-->AREAVAL                           Determines area of event.
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->OPTION                            Determines MC events to be traced.
  |
  |-->RECGEN1                           Generates MC trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZPA - SDAID PAGING EVENT PROCESSOR

Function:

The SDAID paging event processor traces all page-fault interruptions, generates paging trace records and writes them on the specified output device (OUTDEV).

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry For SVC traces in case of an SVC-interrupt, for PGMLOAD traces in case of a monitorcall event.

TCBPTR contains the address of the first block in the TCB-queue for SVC or PGMLOAD traces.

Output:

The PAGING trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

IJSDZPC - SDAID PROGRAM CHECK EVENT PROCESSOR

Function:

The SDAID program check processor traces all or selected program interrupts in a specified program area.
It generates a PGMCHECK trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).
TRTPTR contains the address of the trace table (TRT) entry for program check traces.
TCBPTR contains the address of the first block in the TCB-queue for program check traces.

Output:

The PGMCHECK trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZPC

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZPC                                 Processes program check event (PAGE 50).

  |
  |
  |-->AREAVAL                           Determines area of event.
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->OPTION                            Determines PGMCHECK events to be traced.
  |
  |-->PGMPAG                            Processes Page Fault events.
  |
  |     |
  |     +-------------------->IJSDZPA   Processes program load events (page 49).
  |
  |-->RECGEN1                           Generates the PGMC trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZPL - SDAID PROGRAM LOAD TRACE PROCESSOR

Function:

The SDAID program load event processor traces selected SVC-interrupts and monitor call interrupts used for the program load function, generates a program load trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TRTPTR contains the address of the trace table (TRT) entry for SVC traces in case of an SVC-interrupt, for PGMLOAD traces in case of a monitorcall event.

TCBPTR contains the address of the first block in the TCB-queue for SVC or PGMLOAD traces.

Output:

The PGMLOAD trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZPL

```
Internal                      External
routines                      routines  Function
----------------------------------------------------------------------------

IJSDZPL                                 Processes program load events (PAGE 52).

  |
  |
  |-->PGMLMON                           Analyzes program load monitor call.
  |
  |     |
  |     |-->GETPNM2                     Processes loaded phase.
  |     |-->MODDUMP                     Saves load and end address for DUMP.
  |     |-->MODIFY                      Checks for OVERLOAD/OVERLAY.
  |     |-->PLADDR                      Saves load and end address of phase.
  |     +-->SETHDL                      Processes HDL option.
  |
  |-->PGMLSVC                           Analyzes program load SVC.
  |
  |     |
  |     |-->AREAVAL                     Determines area of SVC event.
  |     +-->GETPNM1                     Gets name of phase to be traced.
  |
  +-->RECGEN                            Generates PGML trace record.

        |
        +-->WRITE                       Writes trace record on OUTDEV.

              |
              |-------------->IJSDDAT   Collects data for OUTPUT (page 67).
              |-------------->IJSDSTP   Stops SDAID on option HALT (page 32).
              |-------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
              +-------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZSA - SDAID STORAGE ALTER TRACE PROCESSOR

Function:

The SDAID storage alter trace processor traces all instructions which alter a specified program area in storage.
It generates a storage alter trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for storage alter traces.
TCBPTR	contains the address of the first block in the TCB-queue for storage alter traces.

Output:

The storage alter trace record written on the specified OUTPUT device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZSA

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZSA                                 Proc. storage alter events (PAGE 54).

  |
  |
  |-->AREAVAL                           Determines area of event.
  |
  |     |
  |     +-->STORAD                      Computes address of altered storage.
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->PATTERN                           Processes storage pattern.
  |
  |-->RECGEN1                           Generates STOR trace record.
  |
  +-->WRITE                             Writes trace record on OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZSI - SDAID SIO TRACE PROCESSOR

Function:

The SDAID SIO trace processor traces the SIO instruction issued by the supervisor.
It generates an SIO trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for SIO traces.
TCBPTR	contains the address of the first block in the TCB-queue for SIO traces.

Output:

The SIO trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZSI

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZSI                                 Processes SIO events (PAGE 56).

  |
  |
  |-->OCCUR                             Determines occurrence of SIO.
  |
  |-->OPTION                            Determines SIO to be traced.
  |
  |-->RECGEN1                           Generates SIO trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |--------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |--------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |--------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +--------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZSV - SDAID SVC TRACE PROCESSOR

Function:

The SDAID SVC trace processor traces all or selected SVC interrupts in a specified program area.
It generates an SVC trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for SVC traces.
TCBPTR	contains the address of the first block in the TCB-queue for SVC traces.

Output:

The SVC trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZSV

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDZSV                                 Processes SVC events (PAGE 58).

  |
  |
  |-->AREAVAL                           Determines area of SVC.
  |
  |-->OCCUR                             Determines occurrence of SVC.
  |
  |-->OPTION                            Determines SVCs to be traced.
  |
  |-->PGMLSVC                           Processes PGMLOAD SVCs.
  |
  |     |
  |     +-------------------->IJSDZPL   Processes program load events (page 52).
  |
  |-->RECGEN1                           Generates SVC trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDZVT - SDAID VTAM TRACE PROCESSOR

Function:

The SDAID VTAM trace processor traces all VTAM buffer pool usages.
It generates a VTAM buffer pool trace record and writes it on the specified output device.

Entry point:

Same as module name.

Input:

EDTPTR	contains the address of the execution data table (EDTCB).
TRTPTR	contains the address of the trace table (TRT) entry for VTAM traces.
TCBPTR	contains the address of the first block in the TCB-queue for VTAM traces.

Output:

The VTAM trace record written on the specified output device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDZVT

```
Internal                      External
routines                      routines  Function
------------------------------------------------------------------------

IJSDZVT                                 Processes VTAMIO events (PAGE 60).

  |
  |
  |-->OCCUR                             Determines occurrence of event.
  |
  |-->RECGEN1                           Generates VTAM trace record.
  |
  +-->WRITE                             Writes trace record to OUTDEV.

        |
        |-------------------->IJSDDAT   Collects data for OUTPUT (page 67).
        |-------------------->IJSDSTP   Stops SDAID on option HALT (page 32).
        |-------------------->IJSDWRB   Writes trace rec. to BUFFER (page 62).
        +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDWRB - SDAID TRACE BUFFER MANAGER

Function:

This routine writes the trace records in an internal buffer in the following modes:

IF OUTDEV B=nnn or OUTDEV B=nnn P=cuu is specified, the buffer is filled in wrap-around mode.

IF OUTDEV B=nnn TAPE=cuu or OUTDEV tape=cuu is specified, the buffer is filled and written on tape on overflow.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

Output:

Updated information in control block EDTCB.

Normal exit:

Returns to the calling procdure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDWRB

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDWRB                                 Writes trace rec. to BUFFER (PAGE 62).

  |
  |
  |-------------------------->IJSDPWB   Prints wrap buffer and tape (page 74).
  |
  |-------------------------->IJSDWRT (page 66).
  |
  +-->OVERPR                            Processes buffer overflow.

        |
        +-------------------->IJSDZBU   Proc. BUFFER OVERFLOW events (page 35).
```

IJSDWRP - SDAID PRINTER I/O MANAGER

Function:
This module writes the SDAID trace lines to the printer specified by the OUTDEV command.

Entry point:
Same as module name.

Input:
EDTPTR contains the address of the execution data table (EDTCB).

Output:
The trace record written on the specified printer.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDWRP

```
Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDWRP                                 Printer or tape I/O (PAGE 64).

  |
  |
  |-->FCLOSE                            Controls the CLOSE request.
  |
  |     |
  |     |-->WRCINIT                     Initializes counters.
  |     |-->WRCLOSE                     Closes the printer.
  |     |-->WRCRETRY
  |     |-->WRCTFLG
  |     +-->WRCWAIT                     Waits for I/O completion.
  |
  |           |
  |           +-->WRCHECK               Analyzes CSW.
  |
  |-->FPUT                              Controls the PUT request.
  |
  |     |
  |     |-->WRCINIT                     Initializes counters.
  |     |-->WRCLOSE                     Closes the printer.
  |     |-->WRCOPEN                     OPEN printer for SDAID.
  |     |-->WRCPUT                      Initiates OUTPUT operation.
  |     |
  |     |     |
  |     |     +-->WRCSIO                Issues SIO.
  |     |
  |     |           |
  |     |           |-------->IJSDIOS   Processes I/O interruptions (page 31).
  |     |           +-->WRCHECK         Analyzes CSW.
  |     |-->WRCRETRY
  |     |-->WRCTFLG
  |     +-->WRCWAIT                     Waits for I/O completion.
  |
  |           |
  |           +-->WRCHECK               Analyzes CSW.
  |
  +-->WRCTFLG
```

IJSDWRT - SDAID TAPE I/O MANAGER

Function:
This module writes the SDAID trace lines to the tape specified by the OUTDEV command.

Entry point:
Same as module name.

Input:

EDTPTR	contains the address of the Execution Data Table (EDTCB).

Output:
The trace record written on the specified tape.

Normal exit:
Returns to the calling procedure.

Error exit:
Same as exit normal.

External references:
See "Diagnostic Aids" section of this chapter.

IJSDDAT - SDAID TRACE DATA COLLECTOR

Function:

The SDAID trace data collector produces the header records for the data records according to the specified output parameters of the TRACE command. This header record contains information about the type of data, the address, and size of the data, etc. It is used:

- -to write the data into a buffer if the output device is tape,
- -to write the data into the wrap-around buffer,
- -to convert the data to EBCDIC and write it on a printer.

The type of data to be recorded is saved in a so called data word in the TCBCB for the actual program event during processing of the OUTPUT parameters of the TRACE command. For each active flag bit in the data word, a data record containing the size of the data record, the address of data, and the number of data bytes is generated. If the data area is larger than 2K bytes, a master record and one or more continuation records are generated. The master record keeps a block of data from the beginning of the data area to the next 2K boundary. A continuation record keeps a 2K-byte block beginning at a 2K boundary. The last continuation record keeps the remaining part of the data area beginning at a 2K boundary. But at least a master record is always generated.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table (EDTCB).

TCBPTR contains the address of the first block in the TCB-queue for cancel traces.

Output:

The header record for the data is stored in EDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDDAT

```
Internal                     External
routines                     routines  Function
--------------------------------------------------------------------------

IJSDDAT                                Collects data for OUTPUT (PAGE 67).

  |
  |
  |-->DATBUFF                          Collects BUFF data.
  |
  |     |
  |     |-------------------->IJSDPWB  Prints wrap buffer and tape (page 74).
  |     +-------------------->IJSDWRB  Writes trace rec. to BUFFER (page 62).
  |
  |-->DATCCB                           Collects CCB data.
  |
  |     |
  |     +-->VALADDR                    Checks addressability of data.
  |
  |-->DATCCW                           Collects CCW and CCWD data.
  |
  |-->DATCHQ                           Collects CHQ data.
  |
  |     |
  |     +-->VALADDR                    Checks addressability of data.
  |
  |-->DATCOMR                          Collects COMREG data.
  |
  |-->DATCREG                          Collects CREG data.
  |
  |-->DATERRBL                         Collects ERRBL data.
  |
  |     |
  |     +-->VALADDR                    Checks addressability of data.
  |
  |-->DATFREG                          Collects FREG data.
  |
  |-->DATGREG                          Collects GREG data.
  |
  |-->DATLOCO                          Collects LOCO data.
  |
  |-->DATLTA                           Collects LTA  data.
  |
  |
  |
  |
|
|     |
|     +-->VALADDR                      Checks addressability of data.
|
|-->DATLUB                             Collects LUB data.
|
|-->DATPIB                             Collects PIB data.
|
|-->DATPTA                             Collects PTA  data.
|
|     |
|     +-->VALADDR                      Checks addressability of data.
|
|-->DATPTAB                            Collects PTAB data.
|
|-->DATPUB                             Collects PUB data.
|
|-->DATSCOM                            Collects SYSCOM data.
|
|-->DATSUPV                            Collects SUPV data.
|
|     |
|     +-->VALADDR                      Checks addressability of data.
|
|-->DATTOD                             Collects TOD  data.
|
|-->DATTTAB                            Collects TTAB data.
|
|-->DATVDMP                            Collects DUMP data.
|
|     |
|     +-->VALADDR                      Checks addressability of data.
|
|-->DATVT1                             Collects data for VTAM trace.
|
|------------------------->IJSDCVT    Converts OUTPUT to EBCDIC (page 70).
|
+------------------------->IJSDWRB    Writes trace rec. to BUFFER (page 62).
```

IJSDCVT - CONVERT OUTPUT TO EBCDIC

Function:

This module converts the data specified by the OUTPUT parameter of the TRACE command to EBCDIC and writes it to the printer.

Entry point:

Same as module name.

Input:

An argument list is passed in register 1. It contains THEN address and number of bytes of data to be converted.

Output:

The header record for the data is stored in EDTCB.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

```
FLOW OF CONTROL FOR ENTRY POINT:  IJSDCVT

Internal                      External
routines                      routines  Function
---------------------------------------------------------------------------

IJSDCVT                                 Converts OUTPUT to EBCDIC (PAGE 70).

  |
  |
  |-->CVTBUFF
  |
  |-->CVTCCB                            Converts and prints CCB .
  |
  |     |
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTCCW                            Converts and prints OUTPUT(CCW/CCWD).
  |
  |     |
  |     |-->CVTSTOR                     Converts and prints data.
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTCHQ                            Conv. and prints CHQ for OUTPUT(IOTAB).
  |
  |     |
  |     |-->CVTSTOR                     Converts and prints data.
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTCOMR                           Converts and prints OUTPUT(COMR).
  |
  |     |
  |     |-->CVTSTOR                     Converts and prints data.
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTCREG                           Converts and prints OUTPUT(CREG).
  |
  |     |
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTERRBL                          Conv.and prts. ERRBL for OUTPUT(IOTAB).
  |
  |     |
  |     |-->CVTSTOR                     Converts and prints data.
  |     +-------------------->IJSDWRP   Determines I/O function (page 64).
  |
  |-->CVTFREG                           Converts and prints OUTPUT(FREG).
```

```
|
|     |
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTGREG                            Converts and prints OUTPUT(GREG).
|
|     |
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTIORB                            Converts and prints IORB.
|
|     |
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTLOCO                            Converts and prints OUTPUT(LOCORE).
|
|     |
|     |-->CVTSTOR                      Converts and prints data.
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTLTA                             Converts and prints OUTPUT(LTA).
|
|     |
|     |-->CVTSTOR                      Converts and prints data.
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTLUB                             Conv. and prints LUB for OUTPUT(IOTAB).
|
|     |
|     |-->CVTSTOR                      Converts and prints data.
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTMD                              Converts and prints OUTPUT(DUMP PHASE).
|
|     |
|     |-->CVTSTOR                      Converts and prints data.
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTPIB                             Converts and prints OUTPUT(PIB).
|
|     |
|     |-->CVTSTOR                      Converts and prints data.
|     +--------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTPTA                             Converts and prints OUTPUT(PTA).
|
|
|
|
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTPUB                            Conv. and prints PUB for OUTPUT(IOTAB).
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTSCOM                           Converts and prints OUTPUT(SCOM).
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTSUPV                           Converts and prints OUTPUT(SUPV).
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTTOD                            Converts and prints OUTPUT(TOD).
|
|     |
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTTTAB                           Converts and prints OUTPUT(TTAB).
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
|-->CVTVDMP                           Converts and prints OUTPUT(DUMP).
|
|     |
|     |-->CVTSTOR                     Converts and prints data.
|     +-------------------->IJSDWRP   Determines I/O function (page 64).
|
+-->CVTVT1                            Converts and prints VTAM buffer pool.

      |
      +-------------------->IJSDWRP   Determines I/O function (page 64).
```

IJSDPWB - PRINT WRAP BUFFER AND TAPE

Function:

The print wrap-around buffer processor gets control when an overflow of this buffer has been occurred. It prints the content of the wrap-around buffer.

Entry point:

Same as module name.

Input:

EDTPTR contains the address of the execution data table EDTCB.

Output:

The content of the wrap-around buffer is printed on the specified output device.

Normal exit:

Returns to the calling procedure.

Error exit:

Same as exit normal.

External references:

See "Diagnostic Aids" section of this chapter.

FLOW OF CONTROL FOR ENTRY POINT: IJSDPWB

```
Internal                     External
routines                     routines  Function
-------------------------------------------------------------------------

IJSDPWB                                Prints wrap buffer and tape (PAGE 74).

  |
  |
  |-->CVTBR                            Converts BR trace record to EBCDIC.
  |
  |-->CVTBU
  |
  |-->CVTCA                            Converts CAN trace record to EBCDIC.
  |
  |-->CVTEX                            Converts EXT trace record to EBCDIC.
  |
  |-->CVTGA                            Converts STOR trace record to EBCDIC.
  |
  |-->CVTIN                            Converts INST trace record to EBCDIC.
  |
  |-->CVTIO                            Converts IO trace record to EBCDIC.
  |
  |-->CVTMC                            Converts MC trace record to EBCDIC.
  |
  |-->CVTPA                            Converts PAG trace record to EBCDIC.
  |
  |-->CVTPC                            Converts PGMCHECK trace record.
  |
  |-->CVTPL                            Converts PGML trace record to EBCDIC.
  |
  |-->CVTSA                            Converts STOR trace record to EBCDIC.
  |
  |-->CVTSI                            Converts SIO trace record to EBCDIC.
  |
  |-->CVTSV                            Converts SVC record to EBCDIC.
  |
  |-->CVTVT1                           Converts VTAM trace records to EBCDIC.
  |
  |-------------------------->IJSDCVT  Converts OUTPUT to EBCDIC (page 70).
  |
  +-------------------------->IJSDWRP  Determines I/O function (page 64).
```

SDAID: DATA AREA INFORMATION

This section is provided to give an overview of the structures of the SDAID control blocks.

The following control blocks are described:

1. The Global Data Table GDTCB.

2. The Initialization Data Table IDTCB.

3. The Execution Data Table EDTCB.

4. The Trace Table TRTCB.

5. The Trace Control Block Queue TCBCBQ.

6. The Trace Control Block TCBCB.

Figure 1 on page 77 is provided to give you an overview of the control block structure for SDAID.

The remainder of this section shows the layout and explains all fields of the data areas used by SDAID.

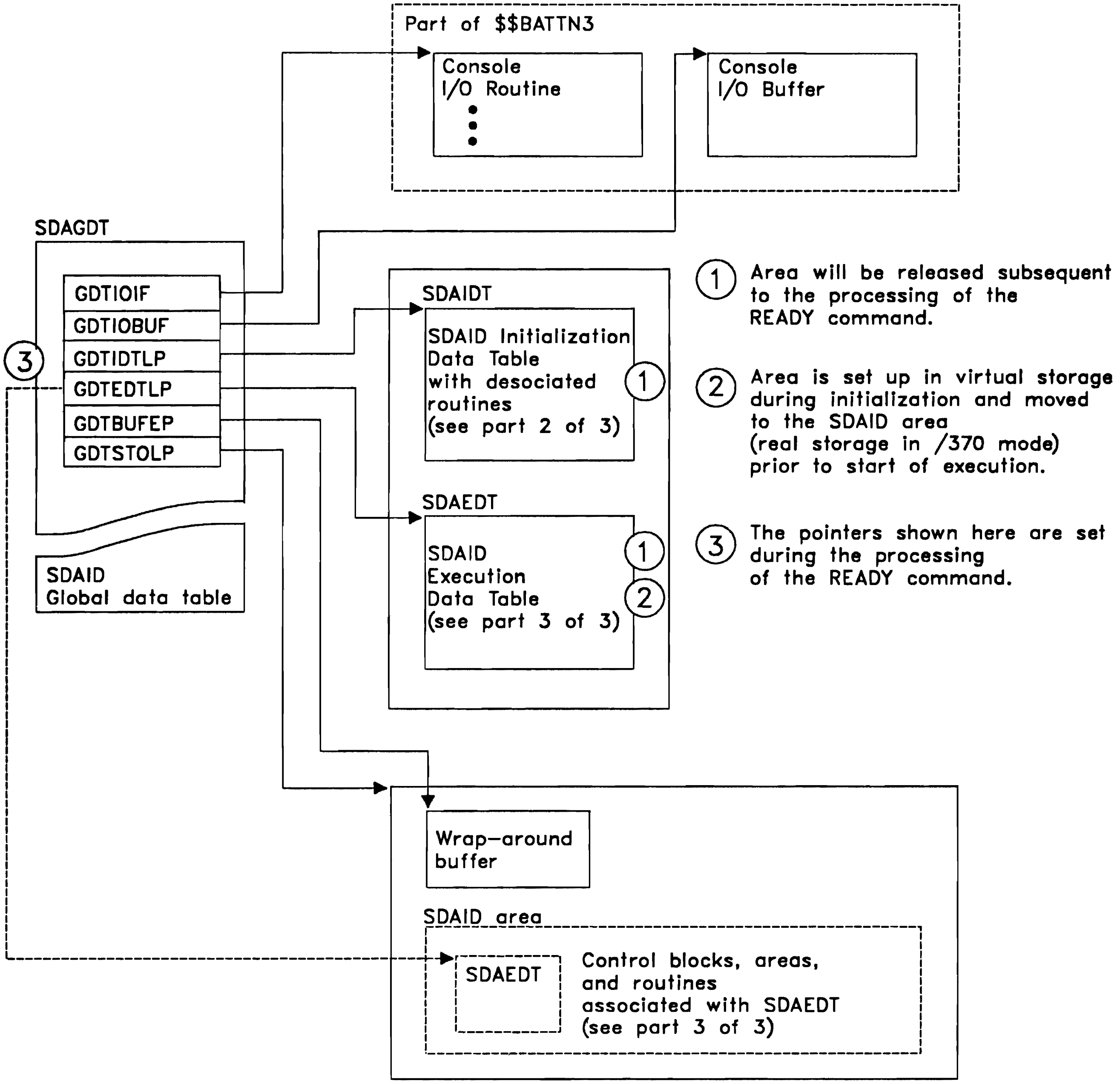
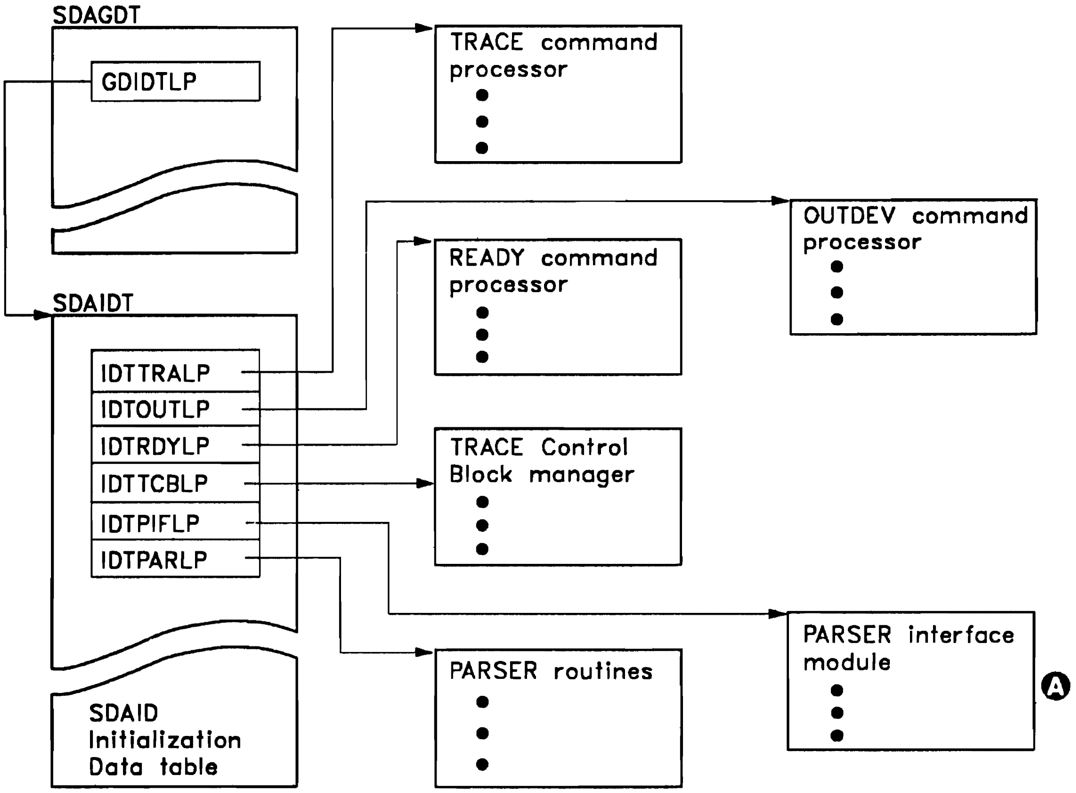
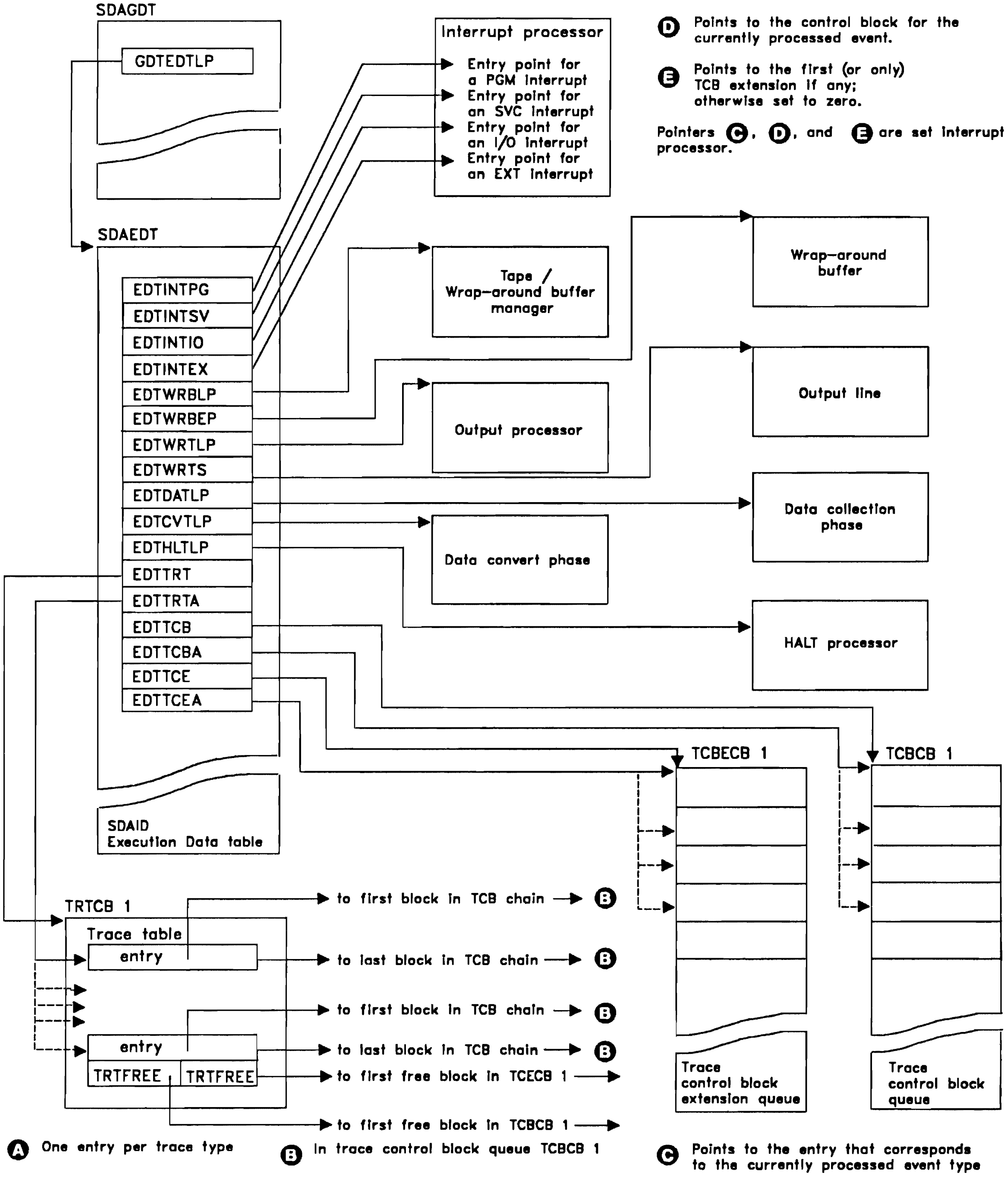

Part of $$BATTN3
Console I/O Routine
Console I/O Buffer
SDAGDT
GDTIOIF
GDTIOBUF
GDTIDTLP
GDTEDTLP
GDTBUFEP
GDTSTOLP
3
SDAID Global data table
SDAIDT
SDAID Initialization Data Table with desociated routines (see part 2 of 3)
1
SDAEDT
SDAID Execution Data Table (see part 3 of 3)
1
2
1 Area will be released subsequent to the processing of the READY command.
2 Area is set up in virtual storage during initialization and moved to the SDAID area (real storage in /370 mode) prior to start of execution.
3 The pointers shown here are set during the processing of the READY command.
Wrap-around buffer
SDAID area
SDAEDT
Control blocks, areas, and routines associated with SDAEDT (see part 3 of 3)

Figure 1 (Part 1 of 3). Structure of SDAID Control Blocks and Data Areas



A Controls required communication between the SDAID initialization routines and the PARSER routines, which are needed during SDAID initialization.

Figure 1 (Part 2 of 3). Structure of SDAID Control Blocks and Data Areas


SDAGDT
GDTEDTLP
Interrupt processor
Entry point for a PGM interrupt
Entry point for an SVC interrupt
Entry point for an I/O interrupt
Entry point for an EXT interrupt
SDAEDT
EDTINTPG
EDTINTSV
EDTINTIO
EDTINTEX
EDTWRBLP
EDTWRBEP
EDTWRTLP
EDTWRTS
EDTDATLP
EDTCVTLP
EDTHLTLP
EDTTRT
EDTTRTA
EDTTCB
EDTTCBA
EDTTCE
EDTTCEA
Tape / Wrap-around buffer manager
Wrap-around buffer
Output processor
Output line
Data collection phase
Data convert phase
HALT processor
TCBECB 1
TCBCB 1
SDAID Execution Data table
TRTCB 1
Trace table
entry
entry
TRTFREE
TRTFREE
to first block in TCB chain → B
to last block in TCB chain → B
to first block in TCB chain → B
to last block in TCB chain → B
to first free block in TCECB 1 →
to first free block in TCBCB 1 →
Trace control block extension queue
Trace control block queue

D Points to the control block for the currently processed event.

E Points to the first (or only) TCB extension if any; otherwise set to zero.

Pointers C, D, and E are set interrupt processor.

A One entry per trace type

B In trace control block queue TCBCB 1

C Points to the entry that corresponds to the currently processed event type

Figure 1 (Part 3 of 3). Structure of SDAID Control Blocks and Data Areas

GDTCB - SDAID Global Data Table

The GDTCB contains information needed when the SDAID program is active. It is located at displacement X'88' of SYSCOM, and is part of the supervisor. F*n indicates flag bytes defined in the following lists.

Created by	Modified by	Used by
IJSDROT	IJSDROT, IJSDRDY	all SDAID routines

0	GDTNAME							
8	F*1	F*2	F*3	F*4	F*5	GDTPROL		
10	GDTROTLP				GDTROTLG			
18	GDTMSGLP				GDTMSGLG			
20	F*5	GDTIORTN			GDTIOBUF			
28	GDTIDTLP				GDTIDTLG			
30	GDTIDTEP							
38	GDTEDTLP				GDTEDTLG			
40	GDTEDTEP							
48	GDTSTOLP				GDTSTOLG			
50	GDTPGMN							
58	GDTSVCN							
60	GDTION							
68	GDTEXTN							
70	GDTXR08				GDTXR09			
78	GDTXR10				GDTXR11			
80	GDTPWBLP				GDTPWBLG			

<table>
<tr><td>88</td><td colspan="2">GDTCCWT1</td><td colspan="3">GDTCCWT2</td></tr>
<tr><td>90</td><td colspan="2">GDTSENSE</td><td colspan="3">GDTOUT</td></tr>
<tr><td>98</td><td colspan="2">GDTTIBA</td><td colspan="3"></td></tr>
<tr><td>A0</td><td colspan="2">GDTPIB1</td><td colspan="3">GDTPIB2</td></tr>
<tr><td>A8</td><td>GDTATID</td><td>GDTMTID</td><td>GDTSTID</td><td>ERP</td><td></td></tr>
<tr><td>B0</td><td colspan="2">GDTWRBIP</td><td colspan="3">GDTWRBLG</td></tr>
<tr><td>B8</td><td colspan="2">GDTDCBUF</td><td colspan="3">GDTJIBEX</td></tr>
<tr><td>C0</td><td colspan="2">GDTBEGV</td><td colspan="3">GDTRTCOM</td></tr>
<tr><td>C8</td><td colspan="2">DEMONDCT</td><td colspan="3">DEMONLK2</td></tr>
<tr><td>D0</td><td colspan="2">reserved</td><td colspan="3">reserved</td></tr>
<tr><td>D8</td><td colspan="2">GDTINTT1</td><td colspan="3">GDTINTT2</td></tr>
<tr><td>E0</td><td colspan="2">GDTINTT3</td><td colspan="3">reserved</td></tr>
<tr><td>E8</td><td colspan="5">GDTCPCMD</td></tr>
<tr><td>F0</td><td colspan="5">GDTCPCMD (continued)</td></tr>
<tr><td>F8</td><td colspan="2">GDTSIMP</td><td colspan="3">GDTSIML</td></tr>
<tr><td>100</td><td colspan="2">GDTSIMT</td><td colspan="3">GDTSIMD</td></tr>
</table>

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	GDTCB	F8	Global Data Table GDTCB
0	GDTNAME	8	Control block name ('GDTCB')
8	GDTFL1	1	F*1 flag byte 1
			<u>Bits defined in GDTFL1</u>
8	GDTFL11	X'80'	Not used
8	GDTFL12	X'40'	OUTDEV command specified
8	GDTFL13	X'20'	TRACE command specified
8	GDTFL18	X'01'	READY command specified
9	GDTFL2	1	F*2 flag byte 2
			<u>Bits defined in GDTFL2</u>
9	GDTFL21	X'80'	SDAID command specified
9	GDTFL22	X'40'	STOPSD command specified
9	GDTFL23	X'20'	STRTSD command specified
9	GDTFL24	X'10'	ENDSD command specified
A	GDTFL3	1	F*3 flag byte 3
			<u>Bits defined in GDTFL3</u>
A	GDTFL31	X'80'	Not used
A	GDTFL32	X'40'	OUTDEV command in progress
A	GDTFL33	X'20'	TRACE command in progress
A	GDTFL37	X'02'	READY command in progress
A	GDTFL38	X'01'	READY command was processed
B	GDTFL4	1	F*4 flag byte 4
			<u>Bits defined in GDTFL4</u>
B	GDTFL41	X'80'	SDAID command in progress
B	GDTFL42	X'40'	STOPSD command in progress
B	GDTFL43	X'20'	STRTSD command in progress
B	GDTFL44	X'10'	ENDSD command in progress

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
C	GDTFL5	4	Reserved
10	GDTROT	8	Anchor for IJSDROT
10	GDTROTLP	4	Load address of phase or 0
			Bits defined in GDTROTLP
10	GDTROTF1	X'80'	Release IJSDROT
14	GDTROTLG	4	Phase size in hex or 0
			Anchor for message library and manager IJSDMSG
18	GDTMSG	8	Anchor for IJSDMSG
18	GDTMSGLP	4	Load address of phase or 0
1C	GDTMSGLG	4	Phase size in hex or 0
	Anchor for I/O routine in $$BATTN3		
20	GDTIOIF	8	Anchor for IORTN in $$BATTN3
20	GDTIOFL	1	F*5 flag byte
			Bits defined in GDTIOFL
20	GDTIOFLW	X'80'	Indicates WRITE request
20	GDTIOFLR	X'40'	Indicates READ request
20	GDTIOFLP	X'20'	Write on SYSLST
21	GDTIORTN	3	Address of I/O routine
24	GDTIOBUF	4	Address of console I/O buffer

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Anchor for Initialization Data Table IJSDIDT		
28	GDTIDT	C	Anchor for IJSDIDT
28	GDTIDTLP	4	Load address of phase or 0
2C	GDTIDTLG	4	Phase size in hex or 0
30	GDTIDTEP	4	Address of Initialization Data Table IDTCB
	Anchor for Execution Data Table IJSDEDT		
38	GDTEDT	10	Anchor for IJSDEDT
38	GDTEDTLP	4	Load address of phase or 0
3C	GDTEDTLG	4	Phase size in hex or 0
40	GDTEDTEP	4	Address of Execution Data Table EDTCB.
	Anchor for SDAID execution storage		
48	GDTSTO	8	
48	GDTSTOLP	4	Start address of execution area
4C	GDTSTOLG	4	Size of area in hex or 0
	Save areas for NEW PSWs		
50	GDTPGMN	8	Save area for PGM NEW PSW
54	GDTPGMA	4	Entry point for PGM interrupts
58	GDTSVCN	8	Save area for SVC NEW PSW
5C	GDTSVCA	4	Entry point for SVC interrupts
60	GDTION	8	Save area for I/O NEW PSW
64	GDTIOA	4	Entry point for I/O interrupts
68	GDTEXTN	8	Save area for EXT NEW PSW
6C	GDTEXTA	4	Entry point for EXT interrupts
	Save area for control registers		
70	GDTXR	10	Save area for control registers
70	GDTXR08	4	Control register 8
70	GDTXR08C	4	
74	GDTXR09	4	Control register 9
74	GDTXR09C	4	
78	GDTXR10	4	Control register 10
78	GDTXR10C	4	
7C	GDTXR11	4	Control register 11
7C	GDTXR11C	4	
	Anchor for print buffer phase IJSDPWB		
80	GDTPWB	8	Anchor for IJSDPWB
80	GDTPWBLP	4	Phase load address or 0
84	GDTPWBLG	4	Phase size in hex or 0

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Pointers to data areas in DOS/VSE supervisor		
88	GDTCCW	28	
88	GDTCCWT1	4	Address of Device Translation Table (DEVTRTAB)
8C	GDTCCWT2	4	Address of Device Specific Table (DEVLIST)
90	GDTSENSE	4	Address of sense buffer
94	GDTOUT	4	Outdevice characteristics
94	GDTOUTCH	2	Outdevice channel number
96	GDTOUTDT	1	Outdevice device type
97	GDTOUTX	1	Outdevice flag byte
98	GDTTIBA	4	Address of TIB area
9C	GDTINTIB	2	TID of set-up partition
A0	GDTPIB1	4	Address of program information block, first part
A4	GDTPIB2	4	Address of program information block, second part
A8	GDTATID	2	TID of highest syst task
AA	GDTMTID	2	TID of highest main task
AC	GDTSTID	2	TID of highest sub-task
AE	GDTTIDER	1	TID of ERP task
	Anchor for wrap-around buffer		
B0	GDTWRB	8	Anchor for wrap/tape buffer
B0	GDTWRBEP	4	Start address of buffer or 0
B4	GDTWRBLG	4	Size of buffer in hex or 0
	Fields used by DOS/VSE dump component		
B8	GDTDCBUF	4	
BC	GDTCOM2	4	
BC	GDTJIBEX	4	
C0	GDTBEGV	4	Begin of virtual area
C4	GDTRTCOM	4	Address of interface area to $$BATTNA or IJSDAID
	Used by DEMON component		
C8	DEMONDCT	4	
CC	DEMONLK2	4	
	Supervisor I/O tables		
D8	GDTINTT1	4	Supervisor I/O table 1
DC	GDTINTT2	4	Supervisor I/O table 2
E0	GDTINTT3	4	Supervisor I/O table 3
E8	GDTCPCMD	16	CP command work area

IDTCB - SDAID Initialization Data Table

The IDTCB is loaded by IJSDROT at initialization time (SDAID command entered). It contains an entry for each of the initialization routines, the I/O areas and the command tables with its directories. The IDTCB is anchored in the Global Data Table (GDTCB).

Created by	Modified by	Used by
IJSDIDT	all initialization routines	

0	IDTCBNAM	
8	IDTCMDLP	IDTCMDLG
10	IDTCMDT	
18	IDTTRALP	IDTTRALG
20	IDTOUTLP	IDTOUTLG
28	IDTRDYLP	IDTRDYLG
30	reserved	reserved
38	IDTTCBLP	IDTTCBLG
40	IDTTCBF	IDTTCBP
48	IDTTCBM1	IDTTCBM2
50	IDTTCBA1	IDTTCBA2
58		
60		
68	IDTPIFLP	IDTPIFLG
70	IDTPARLP	IDTPARLG
78	IDTPARIP	IDTPARRP

80	IDTPARFL	IDTCPCBP
88	IDTIOAD	IDTIOLG
90	IDTIONXT	reserved
98	reserved	reserved
A0	IDTCT2LP	IDTCT2LG
A8	IDTCT3LP	IDTCT3LG
B0	IDTCT4LP	IDTCT4LG
B8	reserved (28 BYTES)	
D4	IDTCDOD (28 BYTES)	
F0	IDTCDTR (28 BYTES)	
10C	IDTCDOP (28 BYTES)	
128	(28 BYTES)	
144	IDTPARIO (80 BYTES)	
194	IDTPARER (80 BYTES)	

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	IDTCB	1E4	Initialization data table IDTCB
0	IDTCBNAM	8	Control block name ('IDTCB')
8	IDTDATA1	1DC	Control block data
8	IDTCMD	10	Anchor for command table
8	IDTCMDLP	4	Address of phase or 0
C	IDTCMDLG	4	Phase size in hex or 0
10	IDTCMDT	4	Address of actual command table
	Anchor for TRACE command processor IJSDTRA		
18	IDTTRA	8	Anchor for phase IJSDTRA
18	IDTTRALP	4	Address of phase or 0
1C	IDTTRALG	4	Phase size in hex or 0
	Anchor for OUTDEV command processor IJSDOUT		
20	IDTOUT	8	Anchor for phase IJSDOUT
20	IDTOUTLP	4	Address of phase or 0
24	IDTOUTLG	4	Phase size in hex or 0
	Anchor for READY command processor		
28	IDTRDY	8	Anchor for phase IJSDOUT
28	IDTRDYLP	4	Address of phase or 0
2C	IDTRDYLG	4	Phase size in hex or 0
	Anchor and interface to trace control block manager IJSDTCB		
38	IDTTCB	20	Anchor for phase IJSDTCB
38	IDTTCBLP	4	Address of phase or 0
3C	IDTTCBLG	4	Phase size in hex or 0
40	IDTTCBX	18	Interface to IJSDTCB
40	IDTTCBF	4	Function to be performed
44	IDTTCBP	4	Position of enqueued element
48	IDTTCBM1	4	ID of master TCB
4C	IDTTCBM2	4	Position of master TCB
50	IDTTCBA1	4	ID of appendage TCB
54	IDTTCBA2	4	Position of appended TCB
	Anchor for parser interface module IJSDPIF		
68	IDTPIF	8	Anchor for IJSDPIF
68	IDTPIFLP	4	Address of phase or 0
6C	IDTPIFLG	4	Phase size in hex or 0

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Anchor and interface to PARSER		
70	IDTPAR	18	Anchor for PARSER
70	IDTPARLP	4	Address of phase or 0
74	IDTPARLG	4	Phase size in hex
78	IDTPARX	10	Interface to PARSER
78	IDTPARIP	4	Address of PARSER input area
7C	IDTPARRP	4	Address of PARSER reply area
80	IDTPARFL	4	PARSER flag byte
			Bits defined in IDTPARFL
80	IDTPARF1	X'80'	Check exit active
80	IDTPARF2	X'40'	Check semantic error
80	IDTPARF5	X'08'	
84	IDTCPCBP	4	Address of PARSER control block CPCB
	Anchor for command input area		
88	IDTIO	10	Anchor for I/O area
88	IDTIOAD	4	Address of I/O area 0
8C	IDTIOLG	4	Size of I/O area or 0
90	IDTIONXT	4	Address of continuation
	Anchor for command tables		
98	IDTCT	20	Anchor for command tables
A0	IDTCT2	8	Anchor for OUTDEV cmd table
A0	IDTCT2LP	4	Address of command table or 0
A4	IDTCT2LG	4	Size of table in hex or 0
A8	IDTCT3	8	Anchor for TRACE command table
A8	IDTCT3LP	4	Address of command table or 0
AC	IDTCT3LG	4	Size of table in hex or 0
	Command directory for the SDAID commands		
B8	IDTCDIR	8C	Command directory
D4	IDTCDOD	1C	Directory for OUTDEV command
F0	IDTCDTR	1C	Directory for TRACE command
	I/O areas		
144	IDTPARIO	50	PARSER I/O area 1
194	IDTPARER	50	PARSER I/O area 2

EDTCB - SDAID Execution Data Table

The execution data table contains information needed during SDAID's tracing, and is anchored in the global data table (GDTCB).

Created by	Modified by	Used by
IJSDEDT	all SDAID phases	

0	EDTCBNAM	
8	FL1 FL2 FL3	EDTGDTPT
18	EDTIDTPT	
20	EDTSCMPT	EDTPCMPT
20	EDTINTLP	EDTINTLG
28	EDTINTPG	EDTINTSV
30	EDTINTIO	EDTINTEX
38	EDTWRBLP	EDTWRBLG
40	EDTWRBEP	EDTWRBSZ
48	EDTWRTLP	EDTWRTLG
50	EDTWRTF	EDTWRTS
58	EDTWRTL	
60	EDTIOSLP	EDTIOSLG
68	EDTIOSF	EDTIOSCT
70	EDTIOSAR	EDTIOSSZ
78	EDTIOSPP	
80	EDTDATLP	EDTDATLG
88	EDTDATF	

```
 90 |      EDTCVTLP      |      EDTCVTLG      |
    |=========================================|
 98 |      EDTHLTLP      |      EDTHLTLG      |
    |=========================================|
 A0 |      EDTTRT        |      EDTTRTA       |
    |--------------------|--------------------|
 A8 |                    |      EDTTRTC       |
    |--------------------|--------------------|
 B0 |      EDTTCB        |      EDTTCBA       |
    |--------------------|--------------------|
 B8 |      EDTTCE        |      EDTTCEA       |
    |=========================================|
 C0 |FLG1|FLG2| F1 | F2  |EDTOUTAD  | MD |TRID|
    |=========================================|
 C8 |      EDTTMC        | PEM|      | EDTTREG|
    |--------------------|--------------------|
 D0 |      EDTTBEG       |      EDTTEND       |
    |--------------------|--------------------|
 D8 |      EDTPPER       |SVEX|SVSV|SVPG|SVIO |
    |--------------------|--------------------|
 E0 |      EDTMSKLP      |     EDTMSKEP       |
    |--------------------|--------------------|
 E8 |      EDTMSKLG      |                    |
    |=========================================|
 F0 |      EDTLLTA       |      EDTULTA       |
    |--------------------|--------------------|
 F8 |      EDTLSVA       |      EDTUSVA       |
    |--------------------|--------------------|
100 |      EDTLPTA       |      EDTUPTA       |
    |--------------------|--------------------|
108 |      EDTEOS        |      EDTEOR        |
    |--------------------|--------------------|
110 |      EDTBEGV       |      EDTEOV        |
    |--------------------|--------------------|
118 |      EDTNPART      |      EDTNTASK      |
    |--------------------|--------------------|
120 |      EDTTIBA       |                    |
    |--------------------|--------------------|
128 |      EDTPIB1       |      EDTPIB2       |
```

130	EDTATID	EDTMTID	EDTSTID	DVTY
138	EDTEXTN			
140	EDTSVCN			
148	EDTPGMN			
150	EDTION			
158	EDTSEXT			
160	EDTOPSW			
168	EDTPID	EDTPID1	EDTIOAD	EDTSID
170	EDTOEV1		EDTSEV1	
178	EDTBLKLP		EDTBLKLG	
180	EDTCAW		EDTCCB	
188	EDTCSW			
190	EDTCLOCK			
198	EDTSAV13			
1A0	EDTSAV1 (72 BYTES)			
1E8	EDTSAV2 (64 BYTES)			
228	EDTSAV3 (64 BYTES)			
268	EDTTREC (96 BYTES)			
2C8	EDTTREC1 (120 BYTES)			
340	EDTVTREC			
348	EDTWORK (80 BYTES)			
398	EDTPGOLD			
3A0	EDTMCSAV		EDTPGSAV	

3A8	EDTIOMAD	EDTIOPAD
3B0	EDTSTCKM	
3B8	EDTDMPLP	EDTDMPLG
3C0	EDTDEBLP	EDTDEBLG
3C8	EDTDEBF	EDTTREX
3D0	EDTIDALP	EDTIDALG

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	EDTCB	3C0	Structure for Execution Data Table EDTCB.
0	EDTCBNAM	8	Control block name ('EDTCB')
8	EDTDATA1	98	Area 1
8	EDTFL1	1	Flag byte 1
9	EDTFL2	1	Flag byte 2
			Bits defined in EDTFL2
9	EDTFL21	X'80'	PER event mixed with other
9	EDTFL22	X'40'	PER event mixed with MC
9	EDTFL23	X'20'	PER event mixed with PGM
A	EDTFL3	1	Flag byte 3
			Bits defined in EDTFL3
A	EDTFL31	X'80'	System operates in ECPS mode
A	EDTFL33	X'20'	VM handshaking active
A	EDTFL34	X'10'	SDAID is in debug mode
A	EDTFL35	X'08'	Suppress output generation
A	EDTFL36	X'04'	Suppress tracing of any event
A	EDTFL37	X'02'	COUNT option not specified
A	EDTFL38	X'01'	Virtual addressing
B	EDTFL4	1	Flag byte 4
			Bits defined in EDTFL4
B	EDTFL41	X'80'	Suppress output for deferred I/O interrupts
C	EDTGDTPT	4	Address of Global Data Table (GDTCB).
10	EDTIDTPT	4	Address of Initialization Data Table (IDTCB).
18	EDTSCMPT	4	Address of DOS SYSCOM
1C	EDTPCMPT	4	Address of DOS COMRG
20	EDTINT	18	Anchor for IJSDINT
20	EDTINTLP	4	Address of phase or 0
24	EDTINTLG	4	Phase size in hex or 0

28	EDTINTPG	4	Entry point for PGM interrupts
2C	EDTINTSV	4	Entry point for SVC interrupts
30	EDTINTIO	4	Entry point for I/O interrupts
34	EDTINTEX	4	Entry point for EXT interrupts

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Anchor and interface to wrap buffer manager IJSDWRB		
38	EDTWRB	10	Interface to IJSDWRB
38	EDTWRBFL	1	Flag byte
			Bits defined in EDTWRBFL
38	EDTWRBPC	X'80'	Record buffer if PGM check
38	EDTWRBOF	X'40'	Record buffer if overflow
38	EDTWRBEN	X'20'	Record buffer if SDAID end
38	EDTWRBBU	X'10'	Record buffer on event
38	EDTWRBBT	X'02'	Write buffer on tape
38	EDTWRBOV	X'01'	Buffer overflow occurred
39	EDTWRBLP	3	Address of phase or 0
3C	EDTWRBLG	4	Phase size in hex or 0
40	EDTWRBEP	4	Address of wrap buffer
44	EDTWRBSZ	4	Size of wrap buffer or 0
	Anchor and interface for output processor		
48	EDTWRTLP	4	Address of phase or 0
4C	EDTWRTLG	4	Phase size in hex
50	EDTWRTX	C	Interface definition
50	EDTWRTF	4	Function to be performed
54	EDTWRTS	4	Address of line to be written
58	EDTWRTL	4	Number of bytes to be written
	Anchor and interface to I/O stack		
60	EDTIOS	20	Anchor for IJSDIOS
60	EDTIOSLP	4	Address of phase or 0
64	EDTIOSLG	4	Phase size in hex or 0
68	EDTIOSF	4	Function to be performed
6C	EDTIOSCT	4	Number of outstanding dev ends
6C	EDTIOSNO	4	Number of outstanding dev ends
70	EDTIOSAR	4	Address of I/O stack
74	EDTIOSSZ	4	Size of I/O stack
78	EDTIOSPP	4	Pub ptr to outdev channel
	Anchor and interface for data collector IJSDDAT		
80	EDTDAT	10	Anchor for IJSDDAT
80	EDTDATLP	4	Address of phase or 0
84	EDTDATLG	4	Phase size in hex
88	EDTDATF	4	Function bytes

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Anchor and interface for data converter IJSDCVT		
90	EDTCVT	8	Anchor for IJSDCVT
90	EDTCVTLP	4	Address of phase or 0
94	EDTCVTLG	4	Phase size in hex or 0
	Anchor for HALT processor IJSDSTP		
98	EDTHLT	8	Anchor for IJSDSTP
98	EDTHLTLP	4	Address of phase or 0
9C	EDTHLTLG	4	Phase size in hex or 0
	Anchor for trace tables and control blocks		
A0	EDTDATA2	28	Trace control block area
A0	EDTTRT	4	Address of trace table TRTCB
A4	EDTTRTA	4	Address of actual TRTCB entry
AC	EDTTRTC	4	Number of traces of same type
B0	EDTTCB	4	Address of Trace Control Block Queue TCBCBQ
B4	EDTTCBA	4	Address of active TCBCB
B8	EDTTCE	4	Address of TCBCB extension
BC	EDTTCEA	4	Address of active TCECB
	Output device information		
	Status		
C0	EDTDEV	2	
C0	EDTDFLG1	1	Outdev flag byte 1
			Bits defined in EDTDFLG1
C0	EDTDVOPN	X'80'	Outdevice is open
C0	EDTIOSS1	X'40'	I/O is started in interval
C0	EDTIOSS2	X'20'	I/O is started, now busy
C0	EDTIOSS3	X'10'	I/O pending
C0	EDTDSDA	X'08'	OUTDEV SIO in progress
C0	EDTDSICE	X'04'	Channel end
C0	EDTDSIDE	X'02'	Device end
C0	EDTDSIXX	X'01'	CU busy
C1	EDTDFLG2	1	OUTDEV flag byte 2
			Bits defined in EDTDFLG2
C1	EDTOUTFT	X'80'	First time switch
C1	EDTCTLX	X'40'	Channel error
C1	EDTISTCK	X'20'	I/O stack not empty
C1	EDTIOSIM	X'10'	I/O interr simulation reqd
C1	EDTOUTDV	X'08'	Unit check from OUTDEV
C1	EDTPBUF	X'04'	Print buffer not empty
C1	EDTDWTCE	X'02'	Wait for channel end
C1	EDTDWTDE	X'01'	Wait for device end

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Device Information		
C2	EDTOUT	6	
C2	EDTOUTF1	1	V*1 flags
			Bits defined in EDTOUTF1
C2	EDTOUTPR	X'80'	Printer is outdevice
C2	EDTOUTMT	X'40'	Tape is outdevice
C2	EDTOUTCN	X'20'	Console is outdevice
C2	EDTOUTBU	X'08'	Buffer is outdevice
C2	EDTOUTTD	X'01'	Tape-dump active
C3	EDTOUTF2	1	V*2 flags
			Bits defined in EDTOUTF2
C3	EDTOUTPA	X'80'	Physical address specified
C3	EDTOUTLA	X'40'	Logical address specified
C4	EDTOUTAD	2	Phys/loc address of OUTDEV
C4	EDTOUTCH	1	Channel addr of OUTDEV
C6	EDTOUTMD	1	Tape mode
C7	EDTTRID	1	ID of trace in progress
	Preparation area for NEW PSWs and control registers		
C8	EDTCREG	10	Control register prep. area
C8	EDTTCR8	4	Control register 8
C8	EDTTMC	4	Monitor call mask
CC	EDTTCR9	4	Control register 9
CC	EDTTPER	4	PER control register
CC	EDTTPEM	1	PER event masks
CE	EDTTREG	2	Reg.alter mask
D0	EDTTCR10	4	Control register 10
D0	EDTTBEG	4	PER start address
D4	EDTTCR11	4	Control register 11
D4	EDTTEND	4	PER end address
D8	EDTPPER	4	Flags for part. using PER
DB	EDTPPERS	X'01'	PER in supervisor
DC	EDTSVEX	1	Saved system mask Ext new PSW
DC	EDTSVEXF	1	
DD	EDTSVSV	1	Saved system mask Svc new PSW
DD	EDTSVSVF	1	
DE	EDTSVPG	1	Saved system mask Pgm new PSW
DE	EDTSVPGF	1	
DF	EDTSVIO	1	Saved system mask I/O new PSW
DF	EDTSVIOF	1	
E0	EDTMSK	C	System masks save area
E0	EDTMSKLP	4	Addr of control block
E4	EDTMSKEP	4	Addr of save area
E8	EDTMSKLG	4	Size of control block

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Miscellaneous addresses		
F0	EDTSBDY	30	
F0	EDTLLTA	4	Start address of LTA
F4	EDTULTA	4	End address of LTA
F8	EDTLSVA	4	Start address of SVA
FC	EDTUSVA	4	End address of SVA
100	EDTLPTA	4	Start address of PTA
104	EDTUPTA	4	End address of PTA
108	EDTEOS	4	End of supervisor
10C	EDTEOR	4	End of real storage
110	EDTBEGV	4	Begin of virtual area
114	EDTEOV	4	End of virtual storage
118	EDTNPART	4	Number of supported partitions
11C	EDTNTASK	4	Number of supported subtasks
120	EDTGDT1	18	
120	EDTTIBA	4	Address of TIB area
128	EDTPIB1	4	Address if first part of Program Information Block
12C	EDTPIB2	4	Address of 2nd part of Program Information Block
130	EDTATID	2	TID of highest system task
132	EDTMTID	2	TID of highest main task
134	EDTSTID	2	TID of highest sub-task
136	EDTDEVTY	2	PUB device type
			Save area for modified NEW PSWs.
138	EDTDATA4	60	
138	EDTEXTN	8	EXT NEW PSW save area
138	EDTEXTNM	1	System mask
140	EDTSVCN	8	SVC NEW PSW save area
140	EDTSVCNM	1	System mask
148	EDTPGMN	8	PGM NEW PSW save area
148	EDTPGMNM	1	System mask
150	EDTION	8	I/O NEW PSW save area
150	EDTIONM	1	System mask
158	EDTSEX	8	SDAID EXT wait PSW
15D	EDTSEXA	3	EXT NEW PSW address

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Saved old PSW after interrupt		
160	EDTOPSW	8	Saved old PSW
160	EDTOPSM	1	System mask
161	EDTOPFL1	1	
			Bits defined in EDTOPFL1
161	EDTOPID	xxxx....	Protection key
161	EDTOPCMW	xxxx	CMWP bits
162	EDTOPFL2	1	
162	EDTOPCC	..xx....	Condition code
162	EDTOMSK	xxxx	Program mask
165	EDTOPADD	3	Instruction address
168	EDTPID	2	ID of interrupted task
16A	EDTPID1	2	SYSLOG ID
16C	EDTIOADD	2	I/O address at interrupt
16C	EDTIOAD	2	I/O address at interrupt
16C	EDTIOCH	1	I/O channel address
16D	EDTIOCU	1	I/O cu and device address
16E	EDTSID	2	ID of current space
	Occurred program events		
170	EDTOEV1	1	Code for occurred events
			Bits defined in EDTOEV1
170	EDTOEVSV	X'80'	SVC interrupt
170	EDTOEVIO	X'40'	I/O interrupt
170	EDTOEVEX	X'20'	EXT interrupt
170	EDTOEVPG	X'10'	PGM interrupt
171	EDTOEV2	1	Code for occurred MC events
			Bits defined in EDTOEV2
171	EDTOEVMC	X'80'	User monitor call
171	EDTOEVPA	X'40'	Paging event
171	EDTOEVSI	X'20'	SIO event
171	EDTOEVFL	X'10'	Program load event
171	EDTOEVVT	X'08'	VTAM event
171	EDTOEVBU	X'04'	Buffer overflow event
171	EDTOEVCA	X'02'	Cancel event
172	EDTOEV3	1	Reserved
173	EDTOEV4	1	Code for occurred PER events

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
			Bits defined in EDTOEV4
173	EDTOEVIN	X'80'	Instruction fetch event
173	EDTOEVGA	X'40'	Register alter event
173	EDTOEVBR	X'20'	Successful branch event
173	EDTOEVSA	X'10'	Storage alter event
	Specified program events		
174	EDTSEV1	1	Code for specified events
			Bits defined in EDTSEV1
174	EDTSEVSV	X'80'	SVC interrupt
174	EDTSEVIO	X'40'	I/O interrupt
174	EDTSEVEX	X'20'	EXT interrupt
174	EDTSEVPG	X'10'	PGM interrupt
175	EDTSEV2	1	Code for specified MC event
			Bits defined in EDTSEV2
175	EDTSEVMC	X'80'	User monitor call
175	EDTSEVPA	X'40'	Paging event
175	EDTSEVSI	X'20'	SIO event
175	EDTSEVFL	X'10'	Program load event
175	EDTSEVVT	X'08'	VTAM event
175	EDTSEVBU	X'04'	Buffer overflow event
175	EDTSEVCA	X'02'	Cancel event
176	EDTSEV3	1	Code for specified PER event
			Bits defined in EDTSEV3
176	EDTSEVBR	X'80'	Successful branch event
176	EDTSEVIN	X'40'	Instruction fetch event
176	EDTSEVSA	X'20'	Storage alter event
176	EDTSEVGA	X'10'	Register alter event
177	EDTSEV4	1	Miscellaneous flags
			Bits defined in EDTSEV4
177	EDTSEVZZ	X'80'	Any trace using SDAID MC
178	EDTBLK	8	SDAID area statistics
178	EDTBLKLP	4	Addr of SDAID execution area
17C	EDTBLKLG	4	Size of SDAID area

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	CAW save area		
180	EDTCAW	4	CAW save area
184	EDTCCB	4	CCB for IO and SIO trace
188	EDTCSW	8	CSW save area
189	EDTCSCMD	3	Address of last CCW + 8
18C	EDTCSUST	1	Unit status
			Bits defined in EDTCSUST
18C	EDTCSATT	X'80'	Attention
18C	EDTCSMOD	X'40'	Status modifier
18C	EDTCSCUE	X'20'	Control unit end
18C	EDTCSBUS	X'10'	Unit or contr unit busy
18C	EDTCSCHE	X'08'	Channel end
18C	EDTCSDE	X'04'	Device end
18C	EDTCSUCK	X'02'	Unit check
18C	EDTCSUEX	X'01'	Unit exception
18D	EDTCSCST	1	Channel status
			Bits defined in EDTCSCST
18D	EDTCSPCI	X'80'	Program controlled interr
18D	EDTCSICL	X'40'	Incorrect length
18D	EDTCSPCK	X'20'	Channel program check
18D	EDTCSPRO	X'10'	Channel protection check
18D	EDTCSDCK	X'08'	Channel data check
18D	EDTCSCCC	X'04'	Channel control check
18D	EDTCSIFC	X'02'	Channel interface check
18D	EDTCSCHC	X'01'	Channel chaining check
18E	EDTCSCNT	2	Residual count
190	EDTCLOCK	8	TOD clock

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Work and save areas		
198	EDTDATA5	228	Control block area 5
198	EDTSAV13	4	Register 13 save area
1A0	EDTSAV1	48	Intermodule save area
1E8	EDTSAV2	40	Save area for registers, saved at interrupt time
1E8	EDTSR00	4	Register 0
1E8	EDTSR00C	4	
1EC	EDTSR01	4	Register 1
1EC	EDTSR01C	4	
1F0	EDTSR02	4	Register 2
1F0	EDTSR02C	4	
1F4	EDTSR03	4	Register 3
1F4	EDTSR03C	4	
1F8	EDTSR04	4	Register 4
1F8	EDTSR04C	4	
1FC	EDTSR05	4	Register 5
1FC	EDTSR05C	4	
200	EDTSR06	4	Register 6
200	EDTSR06C	4	
204	EDTSR07	4	Register 7
204	EDTSR07C	4	
208	EDTSR08	4	Register 8
208	EDTSR08C	4	
20C	EDTSR09	4	Register 9
20C	EDTSR09C	4	
210	EDTSR10	4	Register 10
210	EDTSR10C	4	
214	EDTSR11	4	Register 11
214	EDTSR11C	4	
218	EDTSR12	4	Register 12
218	EDTSR12C	4	
21C	EDTSR13	4	Register 13
21C	EDTSR13C	4	
220	EDTSR14	4	Register 14
220	EDTSR14C	4	
224	EDTSR15	4	Register 15
224	EDTSR15C	4	
228	EDTSAV3	40	Save area for control registers

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Preparation area for trace records		
268	EDTTREC	60	Trace record core image
2C8	EDTTREC1	78	Trace record EBCDIC format
340	EDTVTREC	4	Address of VTAM1 trace record
348	EDTWORK	50	Temp work area
348	EDTWORK1	8	Workarea 1
350	EDTWORK2	8	Workarea 2
350	EDTWORK3	4	Workarea 2a
354	EDTWORK4	4	Workarea 2b
358	EDTWORK5	8	Workarea 3
360	EDTWORK6	8	Workarea 4
368	EDTWORK7	4	Workarea 5
	Save area for OLD PSWs if a PGM or MC event mixed with		
	A PER event has occurred		
398	EDTPGOLD	8	Save area for PGM OLD PSW
3A0	EDTMCSAV	4	Save area for MC codes
3A4	EDTPGSAV	4	Save area for PGM codes
	Interface to I/O stack		
3A8	EDTIOMAD	4	I/O entry point for MCs
3AC	EDTIOPAD	4	I/O interr call entry
3B0	EDTSTCKM	4	I/O interr addr in PSW
	Interface to SDAID DEBUG		
3B8	EDTDMP	8	
3B8	EDTDMPLP	4	Address of phase or 0
3BC	EDTDMPLG	4	Phase size in hex
3C0	EDTDEB	10	
3C0	EDTDEBLP	4	Address of phase or 0
3C4	EDTDEBLG	4	Phase size in hex
3C8	EDTDEBF	4	Function bytes
3CC	EDTTREX	4	Page exception address
3D0	EDTIDALP	4	Printer buffer address
3D4	EDTIGALG	4	Length of print buffer

TRTCB - SDAID Trace Table

The trace table contains information as to where the trace control blocks to a specific type of event to be traced are located and how many events of this type are to be traced.
The trace table consists of a number of entries which is equal to the number of event types that can be traced.
Each entry in the trace table is 28 bytes (7 times 4) long and keeps the pointer to the first and last element in the appropriate trace control block queue, the address of the event processor, the number of specified events and the number of occurred events.

```
    ┌──────────────────────┐
 0  │       TRTCBNAM       │
    ├───────────┬──────────┼───────────┐
 8  │ TRTFREE   │ TRTCNTF  │ TRTCNTU   │
    ├───────────┼──────────┼───────────┤
14  │ TRTEFREE  │ TRTECNTF │ TRTECNTU  │
    ├───────────┴──────────┴───────────┴──────────────────────┐
20  │              ENTRY 1                                    │
    ├─────────────────────────────────────────────────────────┤
34  │              ENTRY 2                                    │
    ├─────────────────────────────────────────────────────────┤
    =                                                         =
    ├─────────────────────────────────────────────────────────┤
    │              ENTRY ($TRANBR)                            │
    └─────────────────────────────────────────────────────────┘
```

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	TRTCB	1C	Structure of one entry in the trace table TRTCB
0	TRTFXX	4	Address of 1st TCB in chain
4	TRTLXX	4	Address of last TCB in chain
8	TRTCXX	4	Number of TCBS in chain
C	TRTELP	4	Load address of event processor
10	TRTELG	4	Phase size in hex or 0
14	TRTCLP	4	Number of occurred events
18	TRTCLG	4	

TRTCB1 - SDAID Trace Table Entry

The based structure TRTCB1 describes the format and organization of an entry in the trace table.

TRTFXX	TRTLXX	TRTCXX	TRTELP	TRTELG	TRTCLP	TRTCLG

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
0	TRTCB1	1E0	Trace table structure
0	TRTCBNAM	8	Control block name
8	TRTSTAT	C	Statistic information
8	TRTFREE	4	Address of next free element in TCB-QUEUE
C	TRTCNTF	4	Number of free TCBS in queue
10	TRTCNTU	4	Number of TCBS in use
14	TRTEXT	C	Extension for storage DUMP
14	TRTEFREE	4	Address of next free element in TCE queue.
18	TRTECNTF	4	Number of free elements in TCE queue.
1C	TRTECNTU	4	Number of TCES in use
20	TRTBLK	1C0	Begin of trace table
20	TRTBLK1	1C0	Number of entries depends on variable $TRANBR, use structure TRTCB for mapping the entries
1E0	TRTEND	0	End of trace table

TCBCB - SDAID Trace Control Block

The trace control block queue keeps information specified by the TRACE command(s).
The trace control block queue consists of a number of entries equal to the number of events that can be specified.
The based structure TCBCB describes the format and the organization of an element in the trace control block queue. V*n indicates flag bytes defined in the following lists.

<table>
<tr><td>0</td><td>V*1</td><td>V*2</td><td>V*3</td><td>V*4</td><td>V*5</td><td>V*6</td><td>V*7</td><td>V*8</td></tr>
<tr><td>8</td><td colspan="8">TCBESD</td></tr>
<tr><td>10</td><td colspan="8">TCBFL9</td></tr>
<tr><td>18</td><td colspan="4">TCBACT</td><td colspan="2">TCBPID</td><td colspan="2">TCBSID</td></tr>
<tr><td>20</td><td>V*9</td><td colspan="3">TCBLADD1</td><td colspan="4">TCBLADD2</td></tr>
<tr><td>28</td><td>V*10</td><td colspan="3">TCBUADD1</td><td colspan="4">TCBUADD2</td></tr>
<tr><td>30</td><td colspan="2">TCBTRC</td><td colspan="2">TCBTRE</td><td colspan="2">TCBTRA</td><td>OCCF</td><td>V*11</td></tr>
<tr><td>38</td><td colspan="4">TCBCNT</td><td colspan="4">TCBCCWD</td></tr>
<tr><td>40</td><td colspan="4">TCBRTCB</td><td colspan="4">TCBETCB</td></tr>
<tr><td>48</td><td colspan="4">TCBTBWD</td><td colspan="4">TCBTFWD</td></tr>
<tr><td>50</td><td colspan="4">TCBBWD</td><td colspan="4">TCBFWD</td></tr>
</table>

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
0	TCBCB	58	Structure of a trace control block TCBCB
0	TCBBL1	30	Control block area 1
0	TCBTYP	1	V*1 type of TCB
1	TCBFL2	1	V*2 flag byte 2
			Bits defined in TCBFL2
1	TCBFL21	X'80'	TCB is waiting for trigger
1	TCBFL22	X'40'	Trigger event has occurred
1	TCBFL23	X'20'	TCB is implicit specified
1	TCBFL24	X'10'	Stop when event is recorded
1	TCBFL25	X'08'	Occurence counter overflow
1	TCBFL26	X'04'	Occurrence specified
1	TCBFL27	X'02'	TCB is active
2	TCBFL3	1	V*3 flag byte 3
			Bits defined in TCBFL3
2	TCBFL31	X'80'	Code for trace area
3	TCBFL4	1	V*4 flag byte 4
			Bits defined in TCBFL4
3	TCBFL41	1	Number of specified elements in list, for STOR trace the length of the specifid pattern
4	TCBFL5	1	v*5 flag byte 5
			Bits defined in TCBFL5
4	TCBFL51	X'80'	Trace all events of the specified trace type
4	TCBFL52	X'40'	Trace all $I/O units, $svcs, instructions, registers, program checks, phases, monitor calls
4	TCBFL53	X'20'	Trace specific control units, 1 pattern specified for REG and STOR trace, instruction trace with branches only
4	TCBFL54	X'10'	Trace specific channels, 2 patterns specified for register trace
4	TCBFL55	X'08'	Option 'ALL' specified for pgml trace, reg trace with two patterns in wrap-around mode

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
4	TCBFL56	X'04'	Option 'REQ' specified for pgml trace
4	TCBFL57	X'02'	Option 'HDL' specified for pgml trace
	Trace area		
5	TCBFL6	1	V*6 flag byte 6
			Bits defined in TCBFL6
5	TCBFL61	X'80'	Trace in user partition
5	TCBFL62	X'40'	Trace in SUPERVISOR
5	TCBFL63	X'20'	Trace in a PHASE
5	TCBFL64	X'10'	Trace in any area ('ALL')
5	TCBFL65	X'08'	Trace in LTA
5	TCBFL66	X'04'	Trace in SVA
5	TCBFL67	X'02'	Trace in PTA
5	TCBFL68	X'01'	Trace in address space
6	TCBFL7	1	V*7 flag byte 7
			Bits defined in TCBFL7
6	TCBFL71	X'80'	'ADDRESS' specification
6	TCBFL72	X'40'	'OFFSET' specification
6	TCBFL73	X'20'	Real partition specified
6	TCBFL75	X'08'	Suppress trace in JCL
6	TCBFL77	X'02'	Trace in SVA phase
6	TCBFL78	X'01'	Specified phase is active
	TCB subtype		
7	TCBFL8	1	V*8 flag byte 8
			Bits defined in TCBFL8
7	TCBFL81	X'80'	SVC-TCB for PGMLoad, PGM-TCB for paging trace
7	TCBFL82	X'40'	SVC-TCB for vtamio trace
7	TCBFL83	X'20'	SIO-TCB for vtamio trace
7	TCBFL84	X'10'	IO-TCB for vtamio trace
	Event specific data		
8	TCBESD	8	Patterns for stor and reg trace
10	TCBFL9	8	Phase name

OFFSET		BYTES AND	
(HEX)	FIELD NAME	BIT PATTERN	DESCRIPTION
	Output specifications		
18	TCBACT	4	Output specifications
18	TCBACT1	1	Flag byte 1
			Bits defined in TCBACT1
18	TCBACTBU	X'80'	Dump wrap buffer
18	TCBACTMO	X'40'	Dump phase
18	TCBACTVD	X'20'	DUMP
18	TCBACTDP	X'01'	PHASE/DUMP
19	TCBACT2	1	Flag byte 2
			Bits defined in TCBACT2
19	TCBACTVT	X'80'	Dump VTAM usage buffers
19	TCBACTTB	X'20'	Dump Task Control Blocks
19	TCBACTPB	X'10'	Dump Partition Control Blocks
19	TCBACTLU	X'08'	Dump LUB
19	TCBACTPU	X'04'	Dump PUB
19	TCBACTCQ	X'02'	Dump Channel Queue
19	TCBACTEB	X'01'	Dump Error Block
1A	TCBACT3	1	Flag byte 3
			Bits defined in TCBACT3
1A	TCBACTSV	X'40'	Dump SUPERVISOR
1A	TCBACTLO	X'20'	Dump low core
1A	TCBACTCX	X'10'	Dump full CCW data (CCWD)
1A	TCBACTCW	X'08'	Dump Channel program
1A	TCBACTCB	X'04'	Dump CCB
1A	TCBACTRB	X'02'	Dump IORB
1A	TCBACTPI	X'01'	Dump PIB
1B	TCBACT4	1	Flag byte 4
			Bits defined in TCBACT4
1B	TCBACTPT	X'80'	Dump PTA
1B	TCBACTLT	X'40'	Dump LTA
1B	TCBACTTO	X'20'	Dump TOD
1B	TCBACTSC	X'10'	Dump SYSCOM
1B	TCBACTPC	X'08'	Dump COMRG
1B	TCBACTCR	X'04'	Dump control registers
1B	TCBACTFR	X'02'	Dump floating point registers
1B	TCBACTGR	X'01'	Dump general registers

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	Trace area specifications		
1C	TCBPID	2	ID of specified partition
1E	TCBSID	2	ID of specified space
20	TCBLIM	10	Trace area limits
20	TCBLFL1	1	V*9 lower limit flag byte
			Bits defined in TCBLFL1
20	TCBLFL11	X'80'	Lower value = '0'
20	TCBLFL12	X'40'	Lower value is specified
21	TCBLADD1	3	Specified lower limit
24	TCBLADD2	4	True lower limit
28	TCBUFL1	1	V*10 upper limit flag byte
			Bits defined in TCBUFL1
28	TCBUFL11	X'80'	Upper value = '*'
28	TCBUFL12	X'40'	Upper value is specified
29	TCBUADD1	3	Specified upper limit
2C	TCBUADD2	4	True upper limit
30	TCBBL2	28	Control block area 2
	Occurrence specifications		
30	TCBOCC	7	Occurrence specification
30	TCBTRB	2	Start value
32	TCBTRE	2	Stop value
34	TCBTRA	2	Actual occurrence of event
36	TCBOCCF	1	Used in paging trace
			Bits defined in TCBOCCF
36	TCBOCC1	X'80'	Occurrence within range for lower level option
37	TCBFLB	1	Flag byte
			Bits defined in TCBFLB
37	TCBFLB1	X'80'	Terminate SDAID on event
37	TCBFLB2	X'40'	'COUNT' option specified
37	TCBFLB3	X'20'	'COUNT'-COUNTER overflow
37	TCBFLB4	X'10'	Reserved

OFFSET (HEX)	FIELD NAME	BYTES AND BIT PATTERN	DESCRIPTION
	COUNT parameter fields		
38	TCBCNT	4	'COUNT' counter
3C	TCBCCWD	4	Value of CCWD
	Chain pointers		
40	TCBRTCB	4	Not used
44	TCBETCB	4	Offset to DUMP chain
48	TCBTBWD	4	BWD offset to appended TCB
4C	TCBTFWD	4	FWD offset tp appended TCB
50	TCBBWD	4	BWD offset to master TCB
54	TCBFWD	4	FWD offset to master TCB

PHASE CROSS REFERENCE LIST

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
$$BATTN3	IJSDROT	$$BATTNA IJSDAID	GDTCB SDAID M GDTCB R	MAC LOAD MAC GETVIS MAC LOAD MAC FREEVIS MAC EXCP MAC WAIT MAC EXCP MAC WAIT MSG 4C07I
IJSDROT	IJSDPIF IJSDCT2 IJSDPIF IJSDCT3 IJSDPIF IJSDEDT IJSDIOS IJSDRDY IJSDIDT IJSDEDT IJSDTCB IJSDMSG IJSDWRB IJSDMSG	$$BATTN3	GDTCB R IDTCB M SYSCOM * M IDTCB R	MAC MESSAGE MAC GETVIS MAC MESSAGE MAC FREEVIS MAC MESSAGE SVC 100 SVC 55 SVC 54 SVC 4 MSG 4C01I MSG 4C04I MSG 4C05I MSG 4C06I MSG 4C07I MSG 4C36I MSG 4C37I
IJSDIDT		IJSDROT	SYSCOM * R IDTCB M	
IJSDEDT		IJSDROT	GDTCB R	

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDMSG	IOSMOD	IJSDROT IJSDROT IJSDTCB IJSDOUT IJSDTRA IJSDPIF IJSDPIF IJSDRDY		
IJSDTCB	IJSDMSG	IJSDROT IJSDTRA	IDTCB R	MSG 4C09I
IJSDCT2		IJSDROT	IDTCB M	
IJSDCT3		IJSDROT	IDTCB M	
IJSDOUT		IJSDPIF IJSDPIF	CPCB R EDTCB M PUBOWNER * R IORB * M GDTCB M	MAC WAIT MAC MESSAGE SVC 0 MSG 4C10I MSG 4C11I MSG 4C12I MSG 4C15I MSG 4C27I
IJSDTRA	IJSDTCB IJSDMSG	IJSDPIF IJSDPIF	CPCB R	MSG 4C13I MSG 4C14I MSG 4C22I MSG 4C23I MSG 4C24I MSG 4C25I MSG 4C28I MSG 4C29I MSG 4C30I MSG 4C31I MSG 4C32I MSG 4C33I MSG 4C34I MSG 4C35I MSG 4C39I

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDPIF	PARSER	IJSDROT	CPCB R	MAC GETVIS
	IJSDOUT	IJSDROT	IDTCB R	MAC MESSAGE
	IJSDTRA			MAC FREEVIS
	IJSDMSG			MAC MESSAGE
	IOSMOD			MSG 4C02A
	IJSDMSG			MSG 4C07I
	IJSDOUT			MSG 4C03I
	IJSDTRA			
	IOSMOD			
IJSDRDY	IJSDMSG	IJSDROT	TRTCB R	MSG 4C38I
			TCBCB R	MSG 4C16I
			EDTCB R	MSG 4C17I
IJSDINT	IJSDIOS	IJSDROT	GDTCB R	
	IJSDZBR		EDTCB M	
	IJSDZBU		LOCORE R	
	IJSDZCA		TRTCB R	
	IJSDZEX		TCBCB R	
	IJSDZGA			
	IJSDZIN			
	IJSDZIO			
	IJSDZMC			
	IJSDZPA			
	IJSDZPC			
	IJSDZPL			
	IJSDZSA			
	IJSDZSI			
	IJSDZSV			
	IJSDZVT			
IJSDZBR	IJSDDAT	IJSDINT	EDTCB M	MAC CHKTRG
	IJSDSTP		TRTCB R	MAC CHKPART
	IJSDWRB		TCBCB R	MAC CHKAREA
	IJSDWRP			MAC SDAOCC
				MAC RECOUT
IJSDZBU	IJSDPWB	IJSDINT		MAC SDAOCC
	IJSDWRT	IJSDWRB		
	IJSDSTP			

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDZCA	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC CHKTRG MAC CHKPART MAC SDAOCC MAC RECOUT
IJSDZEX	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC SDAOCC MAC RECOUT
IJSDZGA	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC CHKTRG MAC CHKPART MAC CHKAREA MAC REALAD MAC SDAOCC MAC RECOUT
IJSDZIN	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC CHKTRG MAC CHKPART MAC CHKAREA MAC SDAOCC MAC RECOUT
IJSDZIO	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT		MAC SDAOCC MAC RECOUT
IJSDZMC	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC CHKTRG MAC CHKPART MAC CHKAREA MAC SDAOCC MAC RECOUT
IJSDZPC	IJSDZPA IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TRTCB R TCBCB R	MAC CHKTRG MAC CHKPART MAC CHKAREA MAC SDAOCC MAC RECOUT
IJSDZPL	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT IJSDZSV	FCHWORK * R TRTCB M TCBCB R EDTCB M	MAC CHKPART MAC CHKAREA MAC SDAOCC MAC RECOUT

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDZSA	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB R TCBCB R TRTCB M	MAC CHKTRG MAC CHKPART MAC CHKPAGE MAC SDAOCC MAC RECOUT
IJSDZSI	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT	EDTCB M TCBCB R TRTCB M	MAC SDAOCC MAC RECOUT
IJSDZSV	IJSDZPL IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT		MAC CHKTRG MAC CHKPART MAC CHKAREA MAC SDAOCC MAC RECOUT
IJSDZVT	IJSDDAT IJSDSTP IJSDWRB IJSDWRP	IJSDINT		MAC SDAOCC MAC RECOUT
IJSDWRB	IJSDWRT IJSDPWB IJSDWRT IJSDZBU	IJSDROT IJSDZBR IJSDZCA IJSDZEX IJSDZGA IJSDZIN IJSDZIO IJSDZMC IJSDZPC IJSDZPL IJSDZSA IJSDZSI IJSDZSV IJSDZVT IJSDDAT IJSDDAT	EDTCB R TRTCB R	

Phase	Calls	Called	Control Blocks and	Macros, SVCs
Name	to	by	Data Areas	Messages
IJSDWRP	IJSDIOS	IJSDZBR		
		IJSDZCA		
		IJSDZEX		
		IJSDZGA		
		IJSDZIN		
		IJSDZIO		
		IJSDZMC		
		IJSDZPC		
		IJSDZPL		
		IJSDZSA		
		IJSDZSI		
		IJSDZSV		
		IJSDZVT		
		IJSDCVT		
		IJSDPWB		
		IJSDPWB		
IJSDDAT	IJSDWRB	IJSDZBR	EDTCB M	
	IJSDCVT	IJSDZCA	COMREG * R	
	IJSDPWB	IJSDZEX	SYSCOM * R	
	IJSDWRB	IJSDZGA	PUBTAB * R	
		IJSDZIN	CHQ * R	
		IJSDZIO	PUB * R	
		IJSDZMC	CCB * R	
		IJSDZPC	DEVTRTAB * R	
		IJSDZPL	DEVLIST * R	
		IJSDZSA	PIB1TAB * R	
		IJSDZSI	PIB2TAB * R	
		IJSDZSV	NICL * R	
		IJSDZVT	LUB * R	
			PCB * R	
			EDTCB SDAID M	
			TIBATAB * R	
			TIB * R	
			TCB * R	
			TCBCB R	
			GDTCB R	
IJSDCVT	IJSDWRP	IJSDDAT		
		IJSDPWB		
		IJSDPWB		

Phase Name	Calls to	Called by	Control Blocks and Data Areas	Macros, SVCs Messages
IJSDPWB	IJSDCVT IJSDWRP IJSDCVT	IJSDZBU IJSDWRB IJSDDAT		
IJSDDEB	IJSDPWB	IJBXSDA IJBXDM5		

* VSE/Advanced Functions
M = modified   R = referenced

PHASE DIRECTORY

The phase directory is organized by symbolic phase name. It describes the contents of each phase with its subroutine(s) by segment name and function, allowing you to quickly find any desired code.

In the following tables, the phase name appears in the first (leftmost) column. The second column contains an entry-point label, the label of an internal procedure (subroutine or segment). The third column differentiates between entry points (EP), procedures (PR) and segments (SE). The fourth column contains the page number where the logic of the procedure is described and the fifth column contains an abstract of the function of the procedure.

Phase name	Segment name	Type	Page	Function
IJSDAID	IJSDAID	EP	4	Builds and submits SDAID commands
$$BATTN3	$$BATTN3	EP	6	Controls processing of an SDAID command
	SDLOAD	PR		Loads phases IJSDMSG and IJSDROT
	ENDSD	PR		Releases IJSDMSG and IJSDROT
	IOSMOD			
IJSDROT	IJSDROT	EP	8	Main entry all SDAID commands
	ROTOUTD	PR		Processes the OUTDEV command
	ROTTRA	PR		Processes the TRACE command
	ROTRDY	PR		Processes the READY command
	ROTSDAID	PR		Processes the READY command
	ROTSTOP	PR		Processes the STOPSD command
	ROTSTRT	PR		Processes the STRTSD command
	ROTENDSD	PR		Processes the ENDSD command
	ROTFRAL	PR		Releases initialization phases
	ROTGETV	PR		Gets storage from SVA
	ROTFREV	PR		Releases storage in SVA
	ROTPFIX	PR		PFIXes storage in SVA
	ROTPFREE	PR		Frees pages in SVA
	ROTGETR	PR		Gets real storage from page pool
	ROTFRER	PR		Releases storage in page pool

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
	ROTGETS	PR		Computes size of an SDAID phase
	ROTLOAD	PR		Loads an SDAID phase into SVA
IJSDIDT	IJSDIDT	EP	12	Builds initialization table
IJSDEDT	IJSDEDT	EP	13	Allocates execution ctl blocks
IJSDMSG	IJSDMSG	EP	14	Writes an error message
	MSGLOOK	PR		Searches message library
	MSGOUT	PR		Prepares and display message
	MSGUPD	PR		Inserts text in message slots
	MSGNOTF	PR		Processes invalid message number
	MSGLIB	SE		SDAID message library
IJSDTCB	IJSDTCB	EP	15	Manages TCB queue
	TCBINI	PR		Initializes TCB-QUEUE
	TCBENQ	PR		Enqueues a TCB
	TCBAPP	PR		Appends a TCB
	DAREA			
IJSDCT2	IJSDCT2	EP	17	Sets up OUTDEV cmd directory
	DOUTDEV	SE		OUTDEV command definition
IJSDCT3	IJSDCT3	EP	18	Sets up TRACE command directory
	DTRACE	SE		TRACE command definition
IJSDOUT	IJSDOUT	EP	19	Processes the OUTDEV command
	SDOUT01	PR		Processes primary output device
	SDOUT02	PR		Processes size of wrap around buffer
	SDOUT03	PR		Processes secondary output device
	SDOUT04	PR		Validates PRINTER/TAPE device
IJSDTRA	IJSDTRA	EP	21	Processes the TRACE command
	INIT	SE		Initializes work areas
	PARMS	SE		Processes all parameters
	CHKEX	SE		
	CHKTCB			Determines number of TCBs
	ENQTCB	SE		Enqueues all TCBs
	ENQTCE	SE		Enqueues all TCEs
	TRAEMSD	PR		Processes error messages
IJSDPIF	IJSDPIF	EP	23	Checks command syntax
	PIFRCO	SE		Processes IJPARSER return code 0

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
	PIFRC4	SE		Processes IJPARSER return code 4
	CONTINU	SE		Enlarges I/O area
	PIFGETV	PR		Gets storage from SVA
	PIFFREV	PR		Releases storage in SVA
	PIFCHK			
	PIFIOF			
	PIFHELP	SE		Processes the HELP function
IJSDRDY	IJSDRDY	EP	26	Processes the READY command
	PERADDR	PR		Determines PER range
	PERPID	PR		Saves IDs of PER partitions
IJSDINT	IJSDINT	EP	27	Establishes new PSWs in EDTCB
	SVCENT			
	IOENT			
	EXTENT			
	PGMENT			
	COMMON	PR		Closes OUTDEV and returns to SUP
	SDINT			
IJSDZBR	IJSDZBR	EP	33	Generates Branch trace record
	AREAVAL	SE		Determines event area
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates the BR trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts record to EBCDIC
IJSDZBU	IJSDZBU	EP	35	Proc. BUFFER OVERFLOW events
IJSDZCA	IJSDZCA	EP	37	Processes CANCEL events
	AREAVAL	SE		Determines area of event
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates CANCEL trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to ebcdic
IJSDZEX	IJSDZEX	EP	39	Processes EXTERNAL interrupts
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates the EXT trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZGA	IJSDZGA	EP	41	Proc. REGISTER alter events

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
	AREAVAL	SE		Determines area of event
	REGLST	SE		Saves registers used in instruction
	OPTION	SE		Determines instructions to be traced
	OCCUR	SE		Detemines occurrence of event
	RECGEN1	SE		Generates register alter trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZIN	IJSDZIN	EP	43	Proc instruction fetch events
	AREAVAL	SE		Determines area of event
	OPTION	SE		Determines instructions to be traced
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates INST trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZIO	IJSDZIO	EP	45	Processes I/O events
	OPTION	SE		Determines I/O events to be traced
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates I/O trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZMC	IJSDZMC	EP	47	Processes monitor call events
	AREAVAL	SE		Determines area of event
	OPTION	SE		Determines MC events to be traced
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates MC trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZPC	IJSDZPC	EP	50	Processes program check event
	PGMPAG	SE		Processes Page Fault events
	AREAVAL	SE		Determines area of event
	OPTION	SE		Determines PGMCHECK events to be traced
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates the PGMC trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZPL	IJSDZPL	EP	52	Processes program load events
	PGMLSVC	PR		Analyzes program load SVC
	AREAVAL	SE		Determines area of SVC event

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
	GETPNM1	SE		Gets name of phase to be traced
	PGMLMON	PR		Analyzes program load monitor call
	GETPNM2	SE		Processes loaded phase
	MODIFY	SE		Checks for OVERLOAD/OVERLAY
	PLADDR	SE		Saves load and end address of phase
	MODDUMP	SE		Saves load and end address for DUMP
	SETHDL	SE		Processes HDL option
	OCCUR	PR		Determines occurrence of event
	RECGEN	PR		Generates PGML trace record
	WRITE	SE		Writes trace record on OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZSA	IJSDZSA	EP	54	Proc. storage alter events
	AREAVAL	SE		Determines area of event
	PATTERN	SE		Processes storage pattern
	STORAD	SE		Computes address of altered storage
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates STOR trace record
	WRITE	SE		Writes trace record on OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZSI	IJSDZSI	EP	56	Processes SIO events
	OPTION	SE		Determines SIO to be traced
	OCCUR	SE		Determines occurrence of SIO
	RECGEN1	SE		Generates SIO trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZSV	IJSDZSV	EP	58	Processes SVC events
	PGMLSVC	SE		Processes PGMLOAD SVCs
	AREAVAL	SE		Determines area of SVC
	OPTION	SE		Determines SVCs to be traced
	OCCUR	SE		Determines occurrence of SVC
	RECGEN1	SE		Generates SVC trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC
IJSDZVT	IJSDZVT	EP	60	Processes VTAMIO events
	OCCUR	SE		Determines occurrence of event
	RECGEN1	SE		Generates VTAM trace record
	WRITE	SE		Writes trace record to OUTDEV
	RECGEN2	SE		Converts trace record to EBCDIC

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
IJSDWRB	IJSDWRB	EP	62	Writes trace rec. to BUFFER
	OVERPR	SE		Processes buffer overflow
IJSDWRP	IJSDWRP	EP	64	Determines I/O function
	FPUT	SE		Controls the PUT request
	FCLOSE	SE		Controls the CLOSE request
	WRCOPEN	PR		OPEN taper for SDAID
	WRCLOSE	PR		Closes the tape
	WRCINIT	PR		Initializes counters
	WRCPUT	PR		Initiates OUTPUT operation
	WRCWAIT	PR		Waits for I/O completion
	WRCSIO	PR		Issues SIO
	WRCHECK	PR		Analyzes CSW
IJSDDAT	IJSDDAT	EP	67	Collects data for OUTPUT
	DATGREG	PR		Collects GREG data
	DATFREG	PR		Collects FREG data
	DATCREG	PR		Collects CREG data
	DATCOMR	PR		Collects COMREG data
	DATSCOM	PR		Collects SYSCOM data
	DATLTA	PR		Collects LTA data
	DATPTA	PR		Collects PTA data
	DATPIB	PR		Collects PIB data
	DATCCB	PR		Collects CCB data
	DATCCW	PR		Collects CCW and CCWD data
	DATTOD	PR		Collects TOD data
	DATLOCO	PR		Collects LOCO data
	DATSUPV	PR		Collects SUPV data
	DATERRBL	PR		Collects ERRBL data
	DATCHQ	PR		Collects CHQ data
	DATPUB	PR		Collects PUB data
	DATLUB	PR		Collects LUB data
	DATPTAB	PR		Collects PTAB data
	DATTTAB	PR		Collects TTAB data
	DATVT1	PR		Collects data for VTAM trace
	DATVDMP	PR		Collects DUMP data
	DATBUFF	PR		Collects BUFF data
	VALADDR	PR		Checks addressability of data
IJSDCVT	IJSDCVT	EP	70	Converts OUTPUT to EBCDIC
	CVTGREG	PR		Converts and prints OUTPUT(GREG)
	CVTFREG	PR		Converts and prints OUTPUT(FREG)
	CVTCREG	PR		Converts and prints OUTPUT(CREG)

Phase directory (cont'd.)

Phase Name	Segment Name	Type	Page	Function
	CVTCOMR	PR		Converts and prints OUTPUT(COMR)
	CVTSCOM	PR		Converts and prints OUTPUT(SCOM)
	CVTLTA	PR		Converts and prints OUTPUT(LTA)
	CVTPTA	PR		Converts and prints OUTPUT(PTA)
	CVTPIB	PR		Converts and prints OUTPUT(PIB)
	CVTCCB	PR		Converts and prints CCB
	CVTIORB	PR		Converts and prints IORB
	CVTCCW	PR		Converts and prints OUTPUT(CCW/CCWD)
	CVTTOD	PR		Converts and prints OUTPUT(TOD)
	CVTLOCO	PR		Converts and prints OUTPUT(LOCORE)
	CVTSUPV	PR		Converts and prints OUTPUT(SUPV)
	CVTERRBL	PR		Conv.and prts. ERRBL for OUTPUT(IOTAB)
	CVTCHQ	PR		Conv. and prints CHQ for OUTPUT(IOTAB)
	CVTPUB	PR		Conv. and prints PUB for OUTPUT(IOTAB)
	CVTLUB	PR		Conv. and prints LUB for OUTPUT(IOTAB)
	CVTTTAB	PR		Converts and prints OUTPUT(TTAB)
	CVTVT1	PR		Converts and prints VTAM buffer pool
	CVTVDMP	PR		Converts and prints OUTPUT(DUMP)
	CVTMD	PR		Converts and prints OUTPUT(DUMP PHASE)
	CVTSTOR	PR		Converts and prints data
IJSDPWB	IJSDPWB	EP	74	Prints wrap buffer and tape
	CVTSV	SE		Converts SVC record to EBCDIC
	CVTPC	SE		Converts PGMCHECK trace record
	CVTIN	SE		Converts INST trace record to EBCDIC
	CVTGA	SE		Converts STOR trace record to EBCDIC
	CVTBR	SE		Converts BR trace record to EBCDIC
	CVTSA	SE		Converts STOR trace record to EBCDIC
	CVTPA	SE		Converts PAG trace record to EBCDIC
	CVTPL	SE		Converts PGML trace record to EBCDIC
	CVTMC	SE		Converts MC trace record to EBCDIC
	CVTIO	SE		Converts IO trace record to EBCDIC
	CVTSI	SE		Converts SIO trace record to EBCDIC
	CVTEX	SE		Converts EXT trace record to EBCDIC
	CVTVT1	SE		Converts VTAM trace records to EBCDIC
	CVTCA	SE		Converts CAN trace record to EBCDIC

MESSAGE-TO-PHASE CROSS REFERENCE LIST

Message Number	Phase Name	Segment Name
4C01I	IJSDROT	IJSDROT
4C02A	IJSDPIF IJSDAID	PIFRCO IJSDAID
4C03I	IJSDPIF	PIFHELP
4C04I	IJSDROT	IJSDROT
4C05I	IJSDROT	IJSDROT
4C06I	IJSDROT	IJSDROT
4C07I	$$BATTN3 IJSDROT	$$BATTN3 ROTFREV ROTGETR ROTGETS ROTGETV ROTPFIX ROTPFREE
	IJSDPIF	PIFFREV PIFGETV
4C08I	IJSDCT3	IJSDCT3
4C09I	IJSDOUT	SDOUT04
4C10I	IJSDOUT	SDOUT04
4C11I	IJSDOUT	SDOUT04
4C12I	IJSDOUT	SDOUT03
4C13I	IJSDTRA	SDTRA16
4C14I	IJSDTRA	SDTRA16
4C15I	IJSDOUT	SDOUT04

Message directory(continued).

Message Number	Phase Name	Segment Name
4C16I	IJSDRDY	IJSDRDY
4C17I	IJSDRDY	IJSDRDY
4C22I	IJSDTRA	TRAEMSD
4C23I	IJSDTRA	TRAEMSD
4C24I	IJSDTRA	TRAEMSD
4C25I	IJSDTRA	TRAEMSD
4C27I	IJSDOUT	SDOUT04
4C28I	IJSDTRA	TRAEMSD
4C29I	IJSDTRA	TRAEMSD
4C30I	IJSDTRA	TRAEMSD
4C31I	IJSDTRA	TRAEMSD
4C32I	IJSDTRA	CHKTCB TRAEMSD
4C33I	IJSDTRA	CHKTCB TRAEMSD
4C34I	IJSDTRA	TRAEMSD
4C35I	IJSDTRA	TRAEMSD
4C36I	IJSDROT	ROTSTRT
4C37I	IJSDROT	ROTSTOP
4C38I	IJSDRDY	IJSDRDY
4C39I	IJSDTRA	SDTRA21D
4C40I	IJSDAID	SDCONS
4C41I	IJSDAID	IJSDAID

Message directory(continued).

Message Number	Phase Name	Segment Name
4C42I	IJSDAID	IJSDAID
4C43I	IJSDAID	IJSDAID
4C44I	IJSDAID	IJSDAID
4C45I	IJSDAID	IJSDAID
4C46I	IJSDAID	IJSDAID
4C47I	IJSDAID	IJSDAID
4C48I	IJSDAID	IJSDAID
4C51I	IJSDAID	BUFFOUT
4C52I	IJSDAID	SDOUTD
4C53I	IJSDAID	SDOUTD
4C54I	IJSDAID	SDOUTD
4C55I	IJSDAID	SDTRAREA
4C56I	IJSDAID	SDCONS
4C57I	IJSDAID	SDCONS
4C60I	IJSDAID	IJSDAID
4C61I	IJSDAID	IJSDAID

DUMP UTILITY: INTRODUCTION

The dump utility program DOSVSDMP combines these functions:

- the creation of a stand-alone dump program,
- the printing and formatting of SDAID output tapes,
- the printing of dump output tapes.

Dump Program Creation

The program DOSVSDMP is used to create a stand-alone dump program on cards, on a diskette, or on a tape reel. The generated stand-alone dump program is placed aside to be used in emergency situations when the VSE system is in a hard or soft wait situation or in a system loop.

The stand-alone dump program on cards or on diskette writes its dump output directly to a printer device.

The tape resident stand-alone dump program writes the dump data onto the tape behind the dump program. The dump on tape contains dump data and a symptom string. The symptom string supplies environment and error information and control block locator entries. The symptom string is built in order to supply Info/Analysis with all information required to print and format the dump.

Info/Analysis will provide a facility to read a stand-alone dump tape and transfers its contents into the VSE dump library (tape onload function). Info/Analysis will then process the stand alone dump.

Printing and Formatting of SDAID Tapes

The program DOSVSDMP is used to print and format SDAID tapes. DOSVSDMP processes two types of SDAID tapes. The first type of tapes contains the dynamic output of an SDAID trace with OUTDEV assigned to a tape device. The other type of tapes contains SDAID buffer data. If the SDAID program writes its output into a wrap-around buffer, the contents of this buffer can be written onto a tape with the attention DUMP command.

Printing of Dump Tapes

Info/Analysis provides the facility to onload dump tapes into the VSE dump library. If this feature is used, the Info/Analysis functions for dump viewing and dump printing are available for the VSE stand-alone dumps.

If the user does not have enough space in the dump library to onload a stand-alone dump, he may print the stand-alone dump tape with DOSVSDMP. DOSVSDMP prints the symptom record and the virtual dump without any formatting.

INVOCATION OF DOSVSDMP:

DOSVSDMP may be invoked via an EXEC statement followed by a parameter for a direct invocation of its functions. If DOSVSDMP is invoked without parameters, it prompts the console operator for the required functions.

DOSVSDMP accepts the input parameter only if it is invoked from SYSRDR and only for the functions 'Create SA-dump on tape' and 'Print SDAID tape'. The format of the direct invocation of DOSVSDMP is as follows:

```
    // EXEC DOSVSDMP,PARM='PRINT SDAID TAPE=cuu File=nnnn'
    // EXEC DOSVSDMP,PARM='PRINT SDAID TAPE=SYSnnn File=nnnn'
    // EXEC DOSVSDMP,PARM='CREATE DUMP TAPE=cuu'
    // EXEC DOSVSDMP,PARM='CREATE DUMP TAPE=SYSnnn'
```

DUMP UTILITY: PROGRAM DESIGN AND ORGANIZATION INFORMATION

MODULES

The dump utility program DOSVSDMP consists of the following modules:

- IJBXDMP (Initialization Routine: Prompt console operator for additional input or check the operands of the parameter and transfer control to IJBXDM5 or IJBXDM7),
- IJBXDM1 (tape input/output subroutines),
- IJBXDM2 (error messages for DOSVSDMP, console I/O routines, SYSLST output routines),
- IJBXDM5 (prints SDAID and dump tapes),
- IJBXDM7 (generates stand-alone dump programs on card, diskette, or tape),
- IJBXDM8 (builds section 6 entries of symptom record for the control blocks),
- IJBXDM9 (reads console messages from hard copy file and builds section 6 entries of symptom record for them).

MACROS

The macros DMPSCTX, IJBXSCT6, IJBXRC, and IJBXLIO are required to assemble all the stand-alone dump modules.

The macro DMPSCTX contains the common fields, constants and flags which are used for inter-module communication between the different stand-alone dump modules.

The macro IJBXSCT6 contains the equates used for the symptom record.

The macro IJBXRC contains the dump record structure.

The macro IJBXLIO is the interface to the logical dump retrieval routine, the SYSLST, SYSLOG and the message handler routine.

PHASES

The phase/module relationship is as follows:

Phase	Module
DOSVSDMP	IJBXDMP
DOSVSDX2	IJBXDM2
DOSVSDX1	IJBXDM1 IJBXDM5
DOSVSDX7	IJBXDM7
DOSVSDX8	IJBXDM8
DOSVSDX9	IJBXDM9

If DOSVSDMP is called for printing an SDAID tape or a dump tape, then the phases DOSVSDMP, DOSVSDX2 and DOSVSDX1 are loaded into core.

If DOSVSDMP is called in order to create a stand-alone dump program, DOSVSDMP, DOSVSDX2 and DOSVSDX7 are loaded into core.

When DOSVSDX7 gets control to create a stand-alone dump tape, it loads the phase DOSVSDX8 into the I/O buffer of IJBXDM1 and copies its contents onto the tape. In the same way it loads the phase DOSVSDX9 and copies its contents onto tape.

STORAGE REQUIREMENTS

DOSVSDMP together with all additionally loaded phases requires about 166K bytes of storage. This includes an I/O buffer of 99K bytes. A VSE partition of 166K (including 48K GETVIS area) is sufficient for DOSVSDMP.

FLOW OF CONTROL OF DOSVSDMP

When DOSVSDMP is called from Job Control, the module IJBXDMP is entered. It loads the base registers, initializes the interface area to other modules, and loads the phase DOSVSDX2. In prompting mode the selection menu 1 is displayed. For the text of this menu see "Dump Utility Menus" on page 179.
The operator may now specify the activity to be performed.
If dump program creation is required (option 1 or requested in the parameter), the phase DOSVSDX7 is loaded. The dump program is created under control of DOSVSDX7. The phase DOSVSDX7 contains a skeleton of the IPL bootstrap record and of the stand-alone dump program. For dump program creation on tape this skeleton - taylored to the actual environment - is written onto tape. After that DOSVSDX7 loads the phases DOSVSDX8 and DOSVSDX9 into the I/O buffer of IJBXDM1 to write them onto tape as part of the dump program. The structure and the logic flow of the generated dump program is described in "Flow of Control of the Generated Dump Program" on page 135.

If dump program creation is on a card punch or diskette device the operator is prompted to specify the printer address for the stand-alone dump program. Then the phase DOSVSDX7 is loaded into core. The card or diskette dump is created under control of DOSVSDX7.

If printing of a dump tape is required (option 2), an OC exit routine is activated, menu 2 is displayed, and the phase DOSVSDX1 is loaded. Tape printing is performed under control of IJBXDM5. For the text of menu 2 refer to "Dump Utility Menus" on page 179.

If SDAID tape printing is required (option 3 or requested in the parameter), an OC exit routine is activated and the phase DOSVSDX1 is loaded. Tape printing is performed under control of IJBXDM5. The module IJBXDM5 is able to process two types of SDAID tapes: The first type consists of the SDAID trace tapes and the other type consists of the SDAID buffer tapes built by the attention dump command. SDAID tapes which have been built by the attention DUMP command contain a symptom record. The records belonging to the symptom record have an identification field of X'00000001' in bytes 0 - 3. IJBXDM5 skips all records with X'00000001' in the first 4 bytes. There is no difference in processing the SDAID data records for both types of SDAID tapes. IJBXDM5 loads the SDAID formatting routine IJSDDEB and passes the SDAID data to it for formatting and printing.

The modules IJBXDM1 and IJBXDM2 contain subroutines which are called when they are required by other modules. When the required service is completed, control is returned to the calling module.
The module IJBXDM1 contains the tape I/O routines.
The module IJBXDM2 contains the console I/O routines, the SYSLST output routines and a collection of all console messages issued by the program DOSVSDMP.

Figure 2 on page 147 shows the flow of control for DOSVSDMP execution.

THE GENERATED DUMP PROGRAM

The generated dump program is structured in such a way, that the loss of dumped data is kept at a minimum. The generated dump program is therefore divided into two parts: The first part is a small bootstrap program, which is loaded into a low core area reserved exclusively for the stand-alone dump program. It saves some pages of low core to get space for the main routine of the stand-alone dump program. This bootstrap program executes differently for the card and diskette-resident dump program on the one hand, and for the tape resident dump program on the other hand. The second part of the dump program is the dump main routine DMPROG. It is loaded into the already saved core locations. This dump program DMPROG dumps the pages from core or from the page data set. Its structure does not differ much for the different types of the stand-alone dump program.

If the dump program was created on cards or diskette, the storage is printed directly at dump time. The bootstrap program therefore prints five pages of low core to get space for the main part of the dump program. If the dump program is loaded from tape, then the bootstrap program saves ten pages into a working data set on tape (file 2 on tape). These saved pages are later on inserted into the dump file. The following chapter describes the flow of control of the different bootstrap programs and of the generated dump program DMPROG.

In ECPS:VSE mode the machine save key should be used to save the machine status and page zero. In /370 mode the store status key should be used to save the machine status. In ECPS:VSE the stand-alone dump contains a **complete** picture of the dumped system; in /370 mode the dump program destroys low core locations 0 - x'17' and x'300 - x'400'.

FLOW OF CONTROL OF THE GENERATED DUMP PROGRAM

Card and Diskette Dump Program

The flow of control of the diskette stand-alone dump program does not differ from the flow of control of the card resident stand-alone dump program. Instead of reading a card a record of 80 bytes is read from the diskette unit.

SYSTEM IN /370 MODE:

- STEP 1

  The first card contains the CCWs to read the next three cards into the core (location x'300').

- STEP 2

  These three cards prepare the card dump , e.g. save PSWs and registers, and read the next two cards which overlay card three and four. These cards read the next two cards overlaying the previous two cards and this continues until card 50 is read. These total 50 cards print the PSWs, registers and the first eight pages of storage. The last two cards (card 49 and 50) read 22 cards into location x'400' (print routine PRINTIO) and 82 cards into location x'B00' (dump main routine DMPROG).

- STEP 3

  DMPROG prints the symptom record and the storage beginning with page eight (x'4000'). If a virtual dump is not possible, e.g. paging is not possible, only the core pages are printed.

SYSTEM IN ECPS :VSE MODE:

- STEP 1

  The first card contains the CCWs to read the next two cards into low core (location x'C8').

- STEP 2

  The second and third card read the next 22 cards into locations x'168'-x'800'. These cards contain a small program which prints page one until page eight.

  Via the RSP instruction the saved page zero and the status information are fetched.
  Page zero is printed.

  After this the stand-alone dump main routine DMPROG is read into location x'B00'.

- STEP 3

  The program DMPROG prints the symptom record and the machine save information (if save information is valid).

  Then it prints the virtual storage beginning with page nine. If a virtual dump is not possible only the addressable pages are printed.

Restrictions for the Use of the Generated Dump Program

- If the stand-alone dump program on card or diskette has been created on a /370 system it can dump a /370 system only. Conversely, if it is created in a system generated with MODE=E, it can be used on a system with MODE=E only.

Tape Resident Dump Program

The system is IPLed from the tape unit.

- STEP 1 (TAPEIPL)

  The first record contains the CCW to read the next record into core location x'300'.

- STEP 2 (ALTDMP)

  The program ALTDMP checks the machine type. If the stand-alone dump program runs on an E-machine, the status of pages 1 - 9 is tested. If at least one of them is not addressable, IPL3 is loaded to make them addressable. Then ALTDMP saves the first ten pages of storage in the file 2 of the stand-alone dump tape and reads the next record into core location x'B00' (dump main routine DMPROG).

- STEP 3 (IPL3)

  This program is only called if one of the first ten pages is not addressable. If a page is connected, it is made addressable. If the page is disconnected, it is connected and made addressable. If no page frame is free, some pages at the end of virtual storage are disconnected.

- STEP 4 (DMPROG)

  (See next Chapter.)

Dump Program Main Routine DMPROG

The stand-alone dump main routine DMPROG is executed at location x'B00' and is common to card/diskette and tape dump.

- STEP 1

  DMPROG investigates whether the system to be dumped has some essential characteristics of a VSE system like

  - VSE/AF supervisor
  - page table (/370 system only)

- dynamic address translation (/370 only)
- IPL complete

If the basic prerequisite is not met, the DMPROG takes a so called real dump. This real dump contains the pages of the processor storage retrieved without address translation. Otherwise a full dump of virtual pages is written onto tape or onto printer via the print routine PRINTIO.

For tape dump the symptom record and the dumped data are written into file 3 of the stand-alone dump tape. For more details of this file refer to "Format of the Stand-alone Dump Tape" on page 139.

For the card or diskette resident dump program, the dump is complete after step 1.

- STEP 2

  For creating the control block locator information of the VSE/AF control blocks the DMPROG must be shifted to the end of storage. This is necessary to read the VSE/AF supervisor into its original location and to analyze it there.

  - /370 SYSTEM

    DMPROG is shifted to the end of real storage minus 32K.

  - ECPS:VSE MODE

    DMPROG looks for 16 2K-pages adjacent to one another. These pages must be addressable, connected or disconnected.

    The search begins with the page that is given by AFCC-16 (AFCC = available-frame-count-capacity). If the last 16 pages are not addressable, connected or disconnected the start address is reduced by 2K. This algorithm takes place until 16 pages are found. If this is not possible the execution of DMPROG is stopped.

  Otherwise DMPROG is shifted and some addresses are relocated.

- STEP 3

  When the DMPROG has been shifted to the end of store the supervisor is read from dump tape (file 3) into its original location in low core. That is necessary for DOSVSDX8 to find the control block locator information.

  Phases DOSVSDX8 and DOSVSDX9 are loaded adjacent to DMPROG. These phases are fetched from the stand-alone dump tape file one. After execution of DOSVSDX8 and DOSVSDX9, DMPROG issues the completion message.

Creating Symptom Record Section 6 Entries

DOSVSDX8 is called from the stand-alone dump program DMPROG. DOSVSDX8 builds section 6 entries of some important VSE/AF4 control blocks. A list of these control blocks is given in the chapter "The Section 6 Extension Records" on page 143.

Dumping Console Messages

DOSVSDX9 is called by the stand-alone dump program DMPROG after all the section 6 records for the control blocks have been written to tape. DOSVSDX9 writes the most recent messages (a maximum of 200 messages) including most recent screen image and an information message as section 6 entries to the dump tape. If the hardcopy file is not accessible, only the last screen image and information messages are dumped. If there is no display support only an information message is dumped.

FORMAT OF THE STAND-ALONE DUMP TAPE

The stand-alone dump tape consists of three files. The first file contains the dump program. It is built by the program DOSVSDX7 during dump program creation. The second file is a work file which is used by the stand-alone dump program to save temporary information. The third file is the dump output. These three files are separated by single tape marks. The last file is followed by two tape marks. The format of the three files is as follows:

File 1

The 24 Byte IPL Record (TAPEIPL): This record permits IPL from the tape device.

The 256 Byte IPL Record (ALTDMP): This record is read by the 24 byte IPL record. It writes ten pages of processor storage onto tape. It then reads the stand-alone dump program (DUMPROG) from tape into the already dumped storage and branches to it.

The IPL extension record (IPL3): If required, this program is invoked by ALTDMP to make some low core pages addressable.

The Description Record: This record contains the copyright statement and a description of the contents of the stand-alone dump tape.

The DMPROG Record: This record contains the stand-alone dump program. It writes a full storage dump to the tape device from which it was IPLed, together with a symptom record.

The program to build section 6 entries: This record is a copy of the phase DOSVSDX8. It writes control block locator entries (section 6 entries) onto tape.

The HCFile program: This record is a copy of the phase DOSVSDX9. It reads up to 200 console messages from the hard copy file and writes them as section 6 entries onto the dump tape.

File 2

The second file is a working data set for the dump program. It is being used for saving data, while the dump is being taken. IPL2 writes the first ten pages of processor storage into this file. The dump program later on retrieves these pages and writes them into file 3. The record length in file 2 is 2048 bytes.

File 3

The third file contains the symptom record and dumped data. Each record of the third file is 2056 bytes long. The first four bytes of each record contain a record identification field.

The records belonging to the symptom string describe the hardware and software environment on which the dump has been taken. In addition they contain error symptoms and control block locator information. The records of the symptom string have an identification field of X'00000001'.

The data records contain in their identification field the starting address of the dumped page. The dumped pages are retrieved from processor storage with or without address translation or they are retrieved from the page data set. A detailed description of the record types of file 3 is given in the next two chapters.

DATA RECORDS IN FILE 3

Before file 3 is written to tape, the stand-alone dump program investigates whether the system to be dumped has some essential characteristics of a VSE system. It is checked whether the processor storage contains a VSE/AF supervisor and whether the page table (/370 systems only) can be found.

If this basic prerequisite is not met, the dump program takes a so-called real dump. This real dump contains the pages of the processor storage retrieved without address translation. In this case the symptom 'MODE=OTHER' is entered into section 5 of the symptom record.

When the processor storage contains a VSE supervisor, the dump program writes a virtual dump onto tape. The dumped pages are retrieved from processor storage with address translation or they are retrieved from the page data set.

Invalid pages are not dumped.

Virtual Pages

The data records contain in their identification field the virtual address of the dumped page. Since any block of dumped data starts on page boundary byte three of the identification record is always X'00'. Byte zero of the identification record contains the flag value X'02' or X'01'. Byte zero is X'02' if the page has been retrieved from core (with address translation). Byte zero is X'01' if the page has not been found in core, but has been retrieved from the page data set (page is connected or disconnected). Bytes 4 - 2051 contain the dumped page. Byte 2052 contains the storage key. Bytes 2053 - 2055 contain the real address of the dumped page if it has been retrieved from core.

Real Pages

The data records contain in their identification field the real address of the dumped page. Since any block of dumped data starts on page boundary byte three of the identification record is always X'00'. Byte zero of the identification record contains the value X'06' to show that this is a 'real page', which has been retrieved from core without any address translation.

Bytes 4 - 2051 contain the dumped page. Byte 2052 contains the storage key. Bytes 2053 - 2055 contain the real address of the dumped page (same as in bytes 1-3).

THE SYMPTOM STRING

Besides the data records file 3 contains the symptom string records. A description of the symptom string is given in the Diagnosis Reference Guide for Info/Analysis. This chapter only explains which fields of the symptom record are used in the VSE stand-alone dump.

The symptom string consists of a header record and one or more section 6 extension records. All records of the symptom string have an identification field of X'00000001'. This identification field does not belong to the symptom record. That means that offsets within the symptom record are calculated relative to the byte following the identification field.

The header record is the first record of file 3. Section 6 extension records may occur interspersed between the data records or they appear at the end of the storage dump.

Header Record

The first two bytes of the header record (following the identification field of X'00000001') contain the characters C'SR'. The stand-alone dump program only fills in section 1 (environment section), section 2 (offsets to other sections), and section 5 (status of the dumped system).

Section 1 contains the following fields:

Field Name	Length	Description
	(4)	(X'00000001') id field
ADSSRID	2	C'SR'
ADSSRCPM	4	CPU model number
ADSSRCPS	6	CPU serial number
ADSSRSTK	8	CPU timer value as stored by a Store clock instruction (binary value; not calculated in HH:MM:TH)
ADSSRSTU	22	not used (binary zeros)
ADSSRCMP	9	C'5666-301'
ADSSRREL	1	C'H'
ADSSRFEA	2	C'07'
ADSSFL01	1	X'40'if dump is from VM machine else: X'00'
ADSSFL02	1	X'80' Date and Time are in Store Clock format
ADSSRDTP	8	C'SADUMP '
remaining bytes		binary zeros

Section 2 contains offset pointers to the sections 3, 4, 5, and 6.

The symptoms in section 5 describe the status of the dumped system. Section 5 may contain the following symptoms:

MACHINE=370/E
MODE=PAGING/VM/OTHER-SYSTEM.(NO-VALID-VSE-SUPERVISOR)
MACHINE-STATUS-NOT-SAVED
LOW-CORE-NOT-ADDRESSABLE
CLOCK-NOT-AVAILABLE
ACTIVE SPACE-ID=X

Section 6 contains a 16 byte entry of binary zeros. This means that the section 6 information follows in section 6 extension records.

The Section 6 Extension Records

The first four bytes of the section 6 extension record (following the identification field of X'00000001') contain the characters C'SCT6'. This constant is followed by three fullwords of binary zeros. The following area of 2032 bytes is filled with control block locator entries (LBD entries). LBD entries will be present for the following VSE control blocks:

Control Block Name	Component Identifier	Description
LOWCORE		X'400' bytes of low core
GREGS		General Purpose Registers at time of failure
FREGS		Floating Point Registers at time of failure
PSW		PSW at time of failure
SYSCOM		System Communications Region
COMREG	BG	Partition Communications Region for BG
COMREG	Fn	Partition Communications Region for Fn
PIBTAB		Partition Information Block (Array with up to 16 entries)
PIB2TAB		Partition Information Block Extension (Array with up to 16 entries)
PUBTAB		Physical Unit Block Table
PUB2TAB		Physical Unit Block Table 2
LUBTAB	BG	Logical Unit Block Table for BG containing System and Programmer LUBs
LUBTAB	Fn	Logical Unit Block Table for Fn containing System and Programmer LUBs
ERBLOC		Error Recovery Block consisting of Error Block and first error queue entry
CHQTABLE		Channel Queue Table
CHNTAB		Channel Control Table
SAVAREA	BG	Partition save area for BG
SAVAREA	Fn	Partition save areas for Fn
SAVAREA	task-id	System task save area
TIBATAB		TIB Address Table (containing the addresses of system, main and subtask TIBs
AF-TIB	task-id	Task Information Block
AF-TCB	task-id	Task Control Block
PCB	AR	Partition Control Block for AR
PCB	BG	Partition Control Block for BG
PCB	Fn	Partition Control Block for Fn

LPT		Library Pointer table
LDT		Library Definition Table
SDT		Sublibrary Definition Table
EDT		Extent Definition Table
DDT		Device Definition Table
SDBUFFER		SDAID buffer
MESSAGE	SADUMP	Error messages (errors during taking stand-alone dump)
MESSAGE	HARDCOPY	The last 200 messages from hardcopy file

DUMP FORMATTING

Dump formatting is only provided if the stand-alone dump program writes its output to tape. These dump tapes are transferred into the VSE dump library via the onload function of Info/Analysis. To process the VSE dumps Info/Analysis needs information about the location, size and structure of the VSE control blocks.

The stand-alone dump program and the system dump routines build a symptom string with section 6 entries for some important VSE control blocks. The LBD entries in section 6 contain the storage address and the length of the control block and the control block name. This, however, is not sufficient for formatting. The terminal user wants access to selected fields of the control blocks. For that purpose additional control block templates (LBDF entries) are required.

These LBDF entries reflect the sequential fields described in the original Assembler DSECT or PL/S structure of the control block. The LBDF entries are blocked into section 6 extension records and stored into the VSE system library IJSYSRS. The name of this library member is IJBXMAP. When Info/Analysis processes a VSE dump, it loads IJBXMAP into core and matches the LBD entries of the dump with the LBDF entries of IJBXMAP.

LBDF entries for the following control blocks are contained in the IJSYSRS library member IJBXMAP:

Control block	Mapping macro	PL/S	DSECT
LOWCORE	SGLOWC	-	x
COMREG	MAPCOMR	x	-
SYSCOM	SYSCOM	x	-
PUBTAB	MAPPUB	x	-
SAVAREA	MAPSAVAR	-	x
TIB	MAPTIB	-	x
TCB	MAPTCB	-	x
PCB	MAPPCB	x	-
LPT	INLCLPT	x	-
PIBTAB	MAPPIB PIB=YES	x	-
PIB2TAB	MAPPIB PIB2=YES	x	-

GENERAL LOGIC FLOW

The following indicates the general flow of the dump utility modules whose functions are described on the following pages.

Figure 2 on page 147 shows the general flow of control through the dump utility program DOSVSDMP.
Figure 3 on page 148 shows the flow of control within the module IJBXDM5.
Figure 4 on page 149 shows the flow of control within the phase DOSVSDX7.
Figure 5 on page 150 shows the flow of control within the generated dump program.

A description of the functions of the different modules of DOSVSDMP is given in "Module Descriptions" on page 151.

Note: For text of menus 1 and 2 referred to on the following chart see "Dump Utility Menus" on page 179.

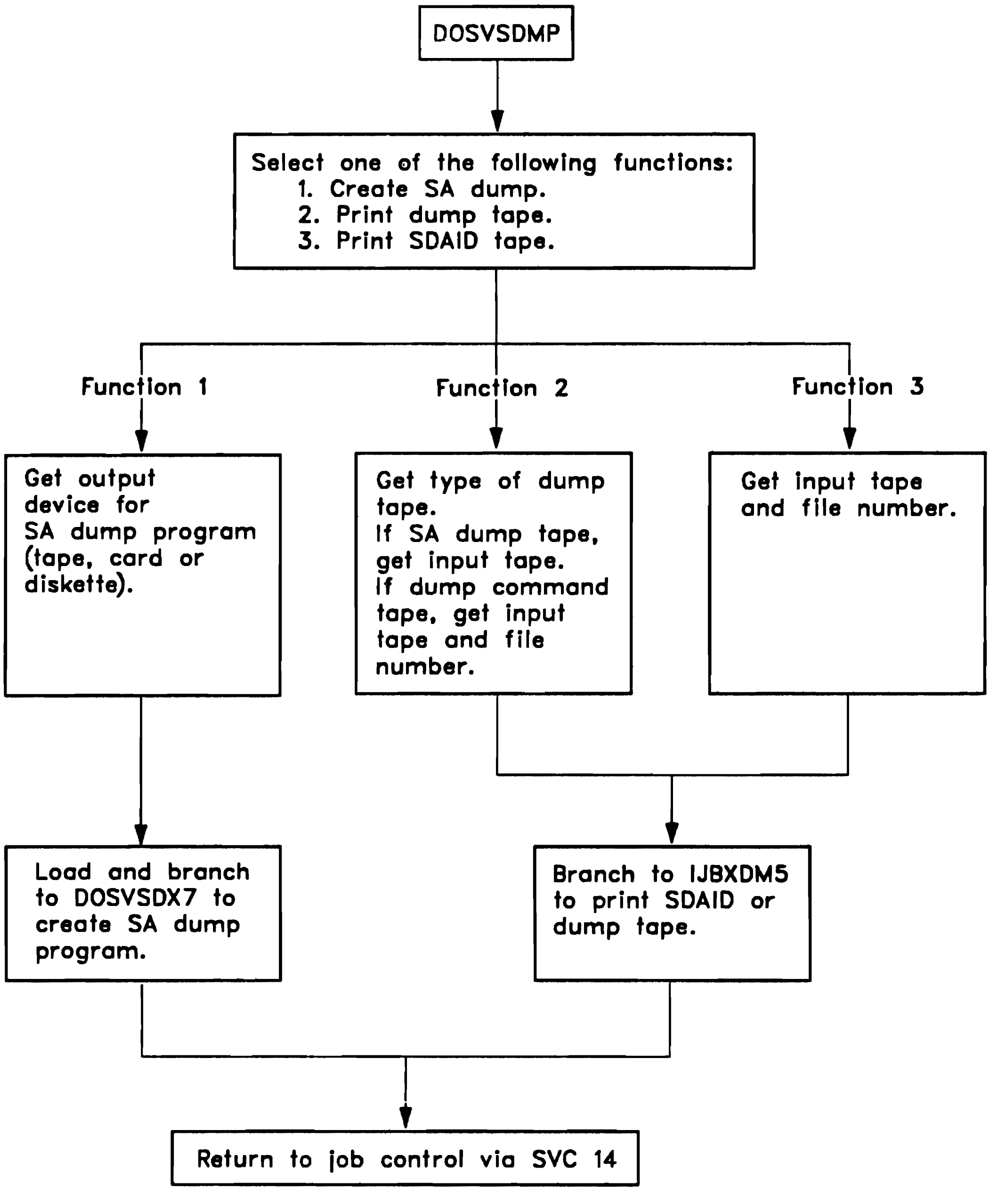

DOSVSDMP
Select one of the following functions:
1. Create SA dump.
2. Print dump tape.
3. Print SDAID tape.
Function 1
Function 2
Function 3
Get output device for SA dump program (tape, card or diskette).
Get type of dump tape. If SA dump tape, get input tape. If dump command tape, get input tape and file number.
Get input tape and file number.
Load and branch to DOSVSDX7 to create SA dump program.
Branch to IJBXDM5 to print SDAID or dump tape.
Return to job control via SVC 14

Figure 2. General Flow of DOSVSDMP

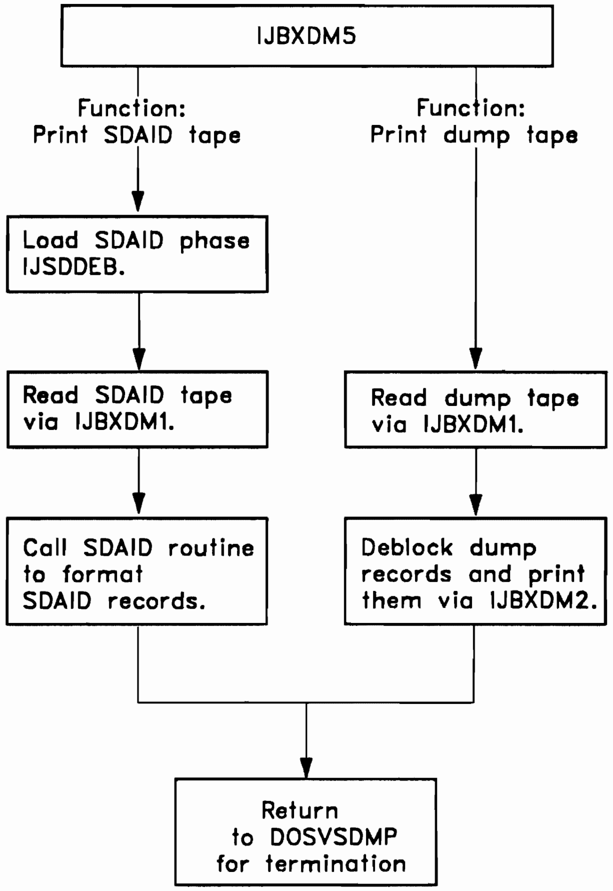

IJBXDM5
Function:
Print SDAID tape
Function:
Print dump tape
Load SDAID phase IJSDDEB.
Read SDAID tape via IJBXDM1.
Read dump tape via IJBXDM1.
Call SDAID routine to format SDAID records.
Deblock dump records and print them via IJBXDM2.
Return to DOSVSDMP for termination

Figure 3. General Flow of DOSVSDX5

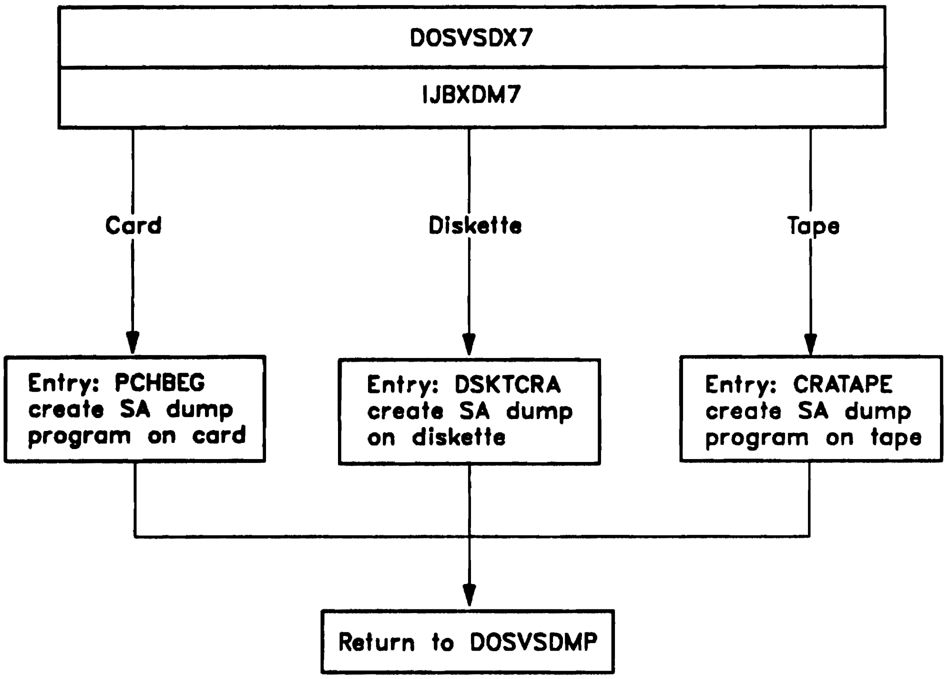

DOSVSDX7
IJBXDM7
Card
Diskette
Tape
Entry: PCHBEG create SA dump program on card
Entry: DSKTCRA create SA dump on diskette
Entry: CRATAPE create SA dump program on tape
Return to DOSVSDMP

Figure 4. General Flow of DOSVSDX7

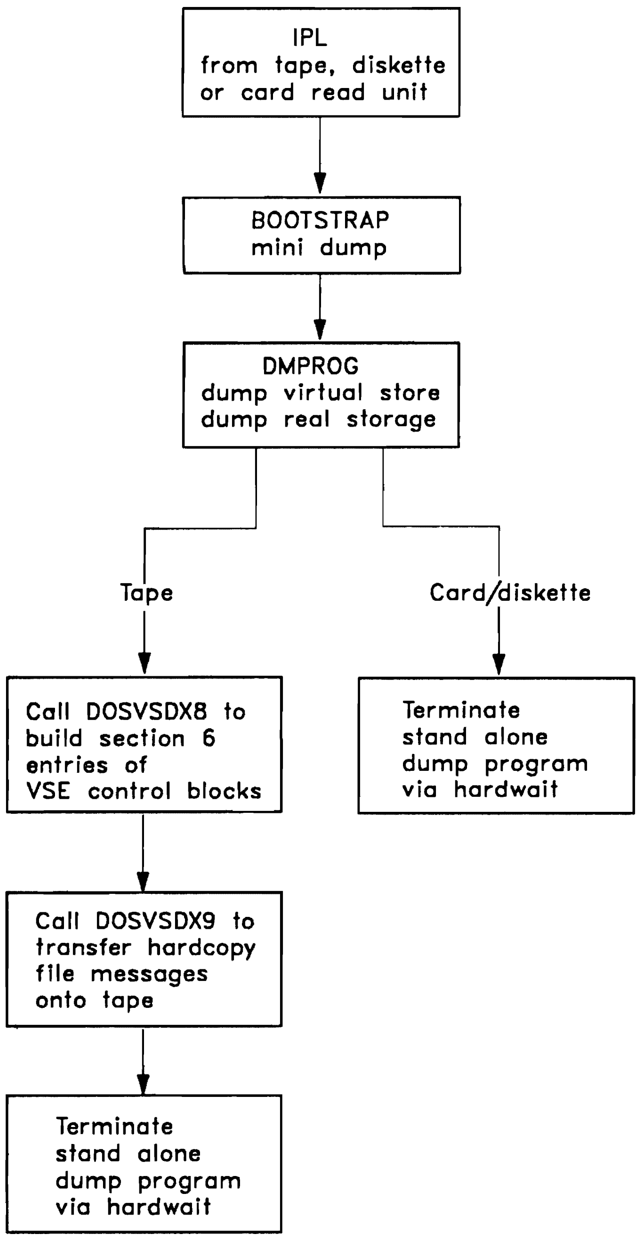


Figure 5. General Flow of DMPROG

MODULE DESCRIPTIONS

IJBXDMP: Initialization Routine

```
***  P R O L O G  *****************************************************
*
*     VSE/AF - STAND-ALONE DUMP PROGRAM
*
***********************************************************************
*
*
*     PHASE NAME: DOSVSDMP
*
*     FUNCTION: GENERATE STAND-ALONE DUMP PROGRAM ON
*               CARD, DISKETTE OR TAPE.
*               PRINT DUMP TAPES.
*
*     CALLED BY: JOB CONTROL
*
*
*     MODULE/PHASE RELATIONSHIP:
*
*               THE LINK BOOK IJBMDUXL IS USED TO LINK
*               THE PHASES OF THE STAND-ALONE DUMP PROGRAM.
*
*              PHASE DOSVSDMP,*,NOAUTO
*               INCLUDE IJBXDMP  DOSVSDMP MONITOR ROUTINE
*              PHASE DOSVSDX2,*
*               INCLUDE IJBXDM2  CONSOLE AND SYSLST I/O
*              PHASE DOSVSDX1,*
*               INCLUDE IJBXDM1  TAPE  RETRIEVAL ROUTINE
*               INCLUDE IJBXDM5  PRINT SDAID OR DUMP TAPES
*              PHASE DOSVSDX7,IJBXDM1
*               INCLUDE IJBXDM7  DUMP PROGRAM GENERATION
*              PHASE DOSVSDX8,*
*               INCLUDE IJBXDM8  SECTION 6 ENTRIES FOR CONTROL BLOCKS
*              PHASE DOSVSDX9,0
*               INCLUDE IJBXDM9  SECTION 6 ENTRIES FOR HARDCOPY MSG
*
*     MODULE GENERATION:
*
*               FOR ASSEMBLING THE ABOVE MODULES THE MACROS
*               IJBXSCT6, DMPSCTX, IJBXRC, AND IJBXLIO ARE REQUIRED.
*
***********************************************************************
```

```
************************************************************************
*
*     VSE STAND-ALONE DUMP PROGRAM - INITIALIZATION ROUTINE
*
************************************************************************
*
*     MODULE NAME: IJBXDMP
*
*     ENTRY POINT: IJBXDMP
*
*     FUNCTION: DOSVSDMP INITIALIZATION,
*               IF DOSVSDMP IS INVOKED WITH A PARAMETER,
*               CHECK THE PARAMETER, ELSE
*               PROMPT OPERATOR FOR ADDITIONAL INPUT,
*               DEPENDENT ON THE REQUIRED FUNCTION
*               TRANSFER CONTROL
*               TO IJBXDM5 (DUMP TAPE),
*               OR TO IJBXDM7 (DUMP PROGRAM CREATION)
*
*     CALLED BY:  JOB CONTROL
*
*     MODULES CALLED:
*               IJBXDM1 (FOR TAPE I/O)
*               IJBXDM2 (FOR CONSOLE AND SYSLST I/O)
*               IJBXDM5 (TO PRINT DUMP TAPES)
*               IJBXDM7 (FOR DUMP PROGRAM CREATION)
*
*     MESSAGES CAUSED:
*             4G01D  'MENU 1'
*             4G02D  'MENU 2'
*             4G04D  SPECIFY OUTPUT DEVICE IN THE FORM SYSNNN OR CUU
*             4G05D  SPECIFY INPUT TAPE IN THE FORM SYSNNN OR CUU
*             4G06D  SPECIFY PRINTER ADDRESS IN THE FORM CUU
*             4G09I  DUMP PROGRAM HAS BEEN CREATED
*             4G11I  SELECTED OPTION IS INVALID
*             4G12I  INVALID PRINTER ADDRESS SPECIFIED
*             4G13I  INVALID DEVICE SPECIFICATION
*             4G15I  PROGRAMMER LOGICAL UNIT IS NOT ASSIGNED
*             4G16I  NO FREE PROGRAMMER LOGICAL UNIT AVAILABLE
*             4G19I  SYSLST IS NOT ASSIGNED
*             4G23I  WRONG TAPE REEL
*             4G25I  DEVICE NOT AVAILABLE IN THE SYSTEM
*             4G26I  WRONG DEVICE SPECIFIED
*             4G30D  SPECIFY FILE NUMBER ON TAPE
*             4G31I  FILE NUMBER SPECIFIED INCORRECTLY
*             4G32I  DEVICE ALREADY USED
*
```

```
*     MESSAGES ISSUED: NONE.
*
*     INPUT: FOR PROMPTING MODE:
*            OPERATOR RESPONSE TO CONSOLE MENUS.
*            FOR PARAMETER INVOCATION:
*            R1 = ADDRESS OF THE POINTER TO THE PARAMETER LIST
*            R15 ≠ R1 IF A PARAMETER WAS SPECIFIED
*            FORMAT OF THE PARAMETER LIST:
*            BYTE 0 - 1: LENGTH OF THE SPECIFIED PARAMETER STRING
*            BYTE 2 ...: PARAMETER CHARACTER STRING
*
*            FORMAT OF THE PARAMETER:
*
*            PARM='CREATE DUMP TAPE=CUU'
*            PARM='CREATE DUMP TAPE=SYSNNN'
*            PARM='PRINT SDAID TAPE=CUU FILE=NNNN'
*            PARM='PRINT SDAID TAPE=SYSNNN FILE=NNNN'
*
*     OUTPUT: UPDATED INTERFACE BLOCK DMPSCTX
*
*     EXIT : TO JOB CONTROL VIA EOJ MACRO
*
*
*     REGISTERS USED:
*               REGISTER 0   WORK REGISTER (MACRO EXPANSION)
*               REGISTER 1   WORK REGISTER (MACRO EXPANSION)
*               REGISTER 2   WORK REGISTER
*               REGISTER 3   WORK REGISTER
*               REGISTER 4   WORK REGISTER
*               REGISTER 5   BASE REGISTER
*               REGISTER 6   WORK REGISTER
*               REGISTER 7   LINK REGISTER TO GETDEVIC
*               REGISTER 8   WORK REGISTER
*               REGISTER 9   WORK REGISTER
*               REGISTER 10  WORK REGISTER
*               REGISTER 11  WORK REGISTER
*               REGISTER 12  SYSLOG INPUT AREA ADDRESS
*               REGISTER 13  SAVE AREA ADDRESS FOR SUBROUTINE CALL
*               REGISTER 14  LINK REGISTER
*               REGISTER 15  WORK REGISTER
*
```

```
*     SEQUENCE OF OPERATION:
*
*     IJBXDMP:  ENTRY POINT.
*
*     BEGIN:    LOAD BASE REGISTERS.
*               FOR PROMPTING MODE:
*               ISSUE MENU 1 FOR REQUIRED FUNCTION.
*               FOR PARAMETER MODE:
*               CHECK OPERANDS FOR REQUIRED FUNCTION.
*               BRANCH TO THE APPROPRIATE ROUTINE.
*
*     CREATION: DUMP PROGRAM CREATION
*
*     STOC:     OTHER OPTIONS (PRINT DUMP AND SDAID TAPE).
*               SET OC EXIT ROUTINE.
*               ISSUE MENU 2 IF PRINT DUMP TAPE REQUESTED.
*
*     SDAIDROT: PRINT SDAID TAPE
*
*     DUMPEOJ:  CLOSE FILES AND RETURN TO EOJ.
*
*
*     SUBROUTINES:
*
*     BAL  RE,PACKRT    PACK AND CHECK A DEVICE ADDRESS
*
*     BAL  RE,FINDFILE  FIND FILE NUMBER ON TAPE.
*
*     BAL  R7,GETDEVIC  CHECK SYSNNN OR CUU
*                       (USING SUBROUTINE PACKRT)
*
*
*
********* END OF PROLOG ************************************************
```

IJBXDM1: Tape Input/Output Routine

```
*/** PROLOG  **********************************************************/
*/*                                                                   */
*/*    VSE STAND-ALONE PROGRAM - TAPE INPUT/OUTPUT ROUTINES           */
*/*                                                                   */
*/*********************************************************************/
*/*                                                                   */
*/*  MODULE IJBXDM1                                                   */
*/*                                                                   */
*/*  MODULE NAME:     IJBXDM1                                         */
*/*                                                                   */
*/*  ENTRY POINT:     IJBXDM1                                         */
*/*                                                                   */
*/*  FUNCTION :       PERFORM ALL I/O OPERATIONS FOR THE DUMP TAPE    */
*/*                                                                   */
*/*  CALLED BY  :     IJBXDMP, IJBXDM5 VIA THE MACRO IJBXLIO          */
*/*                                                                   */
*/*  MODULES CALLED:  IJBXDM2  FOR PRINTING ON SYSLST AND SYSLOG      */
*/*                                                                   */
*/*  MESSAGES CAUSED: 4G28I   DUMP FILE IS EMPTY                      */
*/*                   4G29I   SDAID FILE IS EMPTY                     */
*/*                                                                   */
*/*  MESSAGES ISSUED: NONE                                            */
*/*                                                                   */
*/*  INPUT :          DMPSCTX AS GLOBAL CONTROL BLOCK                 */
*/*  OUTPUT :         RETC000      - NORMAL RETURN                    */
*/*                                                                   */
*/*                                                                   */
*/*  EXIT-NORMAL:     VIA PLS/III RETURN STATEMENT                    */
*/*                                                                   */
*/*  EXIT-ERROR:      NONE                                            */
*/*                                                                   */
*/*   REGISTER CONVENTIONS:                                           */
*/*                   PLS/III TYPE I LINKAGE CONVENTIONS              */
*/*                   REGISTER 12 IS PRIMARY BASE REGISTER            */
*/*                                                                   */
*/*   MODULE-TYPE:    PROCEDURE                                       */
*/*                                                                   */
*/*     PROCESSOR:    SEE LISTING HEADER                              */
*/*                                                                   */
*/*                                                                   */
*/*     LINKAGE:      PLS/III CALL CONVENTIONS                        */
*/*                                                                   */
*/*  EXT.REFERENCES                                                          */
*/*                                                                          */
*/*   ROUTINES    :    NONE                                                  */
*/*                                                                          */
*/*   DATA AREAS :    DMPSCTX   COMMON DATA AREA FOR DOSVSDMP                */
*/*                                                                          */
*/*   MACROS      :    LIOCS MACROS FOR TAPE PROCESSING                      */
*/*                                                                          */
*/*                                                                          */
*/************************************************************************/
*/*                                                                          */
*/*    SEQUENCE OF OPERATION:                                                */
*/*                                                                          */
*/*    IJBXDM1  : ENTRY POINT. INITIATE DTF FOR SDAID TAPE OR                */
*/*               DUMP TAPE. ANALYZE THE REQUESTED FUNCTION.                 */
*/*               DEPENDING ON THE FUNCTION CODE DO:                         */
*/*                                                                          */
*/*    TAPOPEN  : OPEN DUMP FILE. CHECK VALIDITY OF SA AND AR DUMP,          */
*/*               E.G. THE FIRST RECORD MUST BE A SYMPTOM RECORD.            */
*/*                                                                          */
*/*    TAPCLOSE : CLOSE DUMP FILE                                            */
*/*                                                                          */
*/*    TAPNEXT  : READ NEXT DUMP RECORD                                      */
*/*                                                                          */
*/*    TAPDATA  : READ DATA RECORD                                           */
*/*                                                                          */
*/*    TAPSYM   : READ SYMPTOM RECORD                                        */
*/*                                                                          */
*/*    TAPPOS   : POSITION TO FILE START                                     */
*/*                                                                          */
*/*                                                                          */
*/****** END OF PROLOG ***************************************************/
```

IJBXDM2: Message Writer Module

```
***  P R O L O G  *****************************************************
*
*    VSE STAND-ALONE DUMP PROGRAM - MESSAGE WRITER MODULE
*
***********************************************************************
*
*  MODULE NAME:        IJBXDM2
*
*  ENTRY POINT:        IJBXDM2
*
*  FUNCTION:
*                      THIS IS THE MESSAGE WRITER PHASE
*                      OF THE STAND-ALONE DUMP PROGRAM
*                      IT IS CALLED TO DISPLAY MESSAGES
*                      ON SYSLOG OR SYSLST.
*
*  CALLED BY:          IJBXDMP
*                      IJBXDM1
*                      IJBXDM5
*
*  MESSAGES CAUSED:    NONE.
*
*  MESSAGES ISSUED:    ALL MESSAGES OF THE STAND-ALONE DUMP
*                      PROGRAM, EXCEPT THOSE MESSAGES,
*                      WHICH ARE DISPLAYED FROM THE
*                      GENERATED DUMP PROGRAM.
*
*  INPUT:   GLOBAL DATA AREA DMPSCTX
*
*  OUTPUT:  MESSAGE ON SYSLOG OR SYSLST
*
*  NORMAL EXIT:       RETURN TO CALLER
*
*  ERROR EXIT:        NONE.
*
*  REGISTERS :        R4    ADDRESS OF INTERFACE AREA DMPSCTX
*                     R5    BASE REGISTER
*                     RF    RETURN CODE
*
*  MACROS    :        LOGICAL IOCS MACROS
*
```

```
*
*  SEQUENCE OF OPERATION:
*
*           IJBXDM2 :  ESTABLISH BASE REGISTERS
*
*           BEGIN   :  DEPENDING ON THE FUNCTION CODE BRANCH TO ONE
*                      OF THE FOLLOWING ROUTINES.
*
*           OPENROUT:  OPEN SYSLST
*
*           CLOSROUT:  CLOSE SYSLST
*
*           PSYSLST :  SYSLST OUTPUT ROUTINE
*
*           MSYSLST :  DISPLAY THE REQUESTED MESSAGE ON SYSLST
*
*           RSYSLOG :  READ THE OPERATOR RESPONSE TO SELECTION MENUES
*                      FROM SYSLOG
*
*           WSYSLOG :  DISPLAY THE REQUESTED MESSAGE ON SYSLOG.
*
*           LOGLINE :  DISPLAY THE CONTENTS OF THE SYSLOG OUT AREA
*
********* END OF PROLOG ***********************************************
```

IJBXDM5: Print SDAID/ Dump Tape

```
*** P R O L O G ***********************************************************
*
*    VSE/AF - STAND-ALONE DUMP PROGRAM. PRINT SDAID/DUMP TAPE
*
***************************************************************************
*
*
* MODULE NAME:     IJBXDM5
*
* ENTRY POINT:     IJBXDM5
*
* FUNCTION:        PRINT AND FORMAT SDAID TAPES,
*                  PRINT DUMP TAPES UNFORMATTED
*
* CALLED BY:          IJBXDMP
*
* PHASES FETCHED:  IJSDDEB  (SDAID FORMATTING ROUTINE)
*                  IJSDCVT  (SDAID CONVERT ROUTINES)
*                  IJSDPWB  (BUFFER DEBLOCKING ROUTINES)
*
* MODULES CALLED:  IJBXDM1  FOR TAPE I/O
*                  IJBXDM2  FOR PRINTING ON SYSLST AND SYSLOG
*
*
* REGISTERS USED:
*                  R3       SYSLST OUTPUT AREA
*                  R4       BASE REGISTER
*                  R5       COMMON INTERFACE AREA ADDRESS
*
* MESSAGES CAUSED:     4G17I PRINTOUT CANCELED BY THE OPERATOR
*                      4G18I EXECUTION OF SDAID NOT POSSIBLE
*                      4G20I SDAID DEBLOCKING ROUTINES CANNOT BE LOADED
*
* MESSAGES ISSUED:    NONE
*
* INPUT:              SDAID TAPE OR DUMP TAPE
*                     GLOBAL DATA AREA DMPSCTX
*
* OUTPUT:             PRINT OUTPUT
*
* NORMAL EXIT:        RETURN TO CALLER
*
* ERROR  EXIT:        NONE
*
```

```
*  SEQUENCE OF OPERATION:
*
*           IJBXDM5 : INITIALIZE BASE REGISTERS
*
*           BEGIN:   DEPEND ON THE FUNCTION BYTE BRANCH
*                    TO PRTDUMP OR SDAIDRT
*
*           PRTDUMP: PRINT DUMP TAPE
*
*           SDAIDRT: PRINT SDAID TAPE
*
*
********* END OF PROLOG ************************************************
```

IJBXDM7: Create Stand-alone Dump

```
***  P R O L O G  ******************************************************
*
*    CREATE VSE STAND-ALONE DUMP
*
*    THE STAND-ALONE DUMP PROGRAM IS CREATED UNDER CONTROL OF
*    THE PHASE DOSVSDX7. THE PHASE DOSVSDX7 CONTAINS A SKELETON
*    OF THE IPL BOOTSTRAP RECORD AND OF THE STAND-ALONE DUMP
*    PROGRAM. THIS SKELETON - TAYLORED TO THE ACTUAL ENVIRONMENT -
*    IS WRITTEN ONTO CARD OR TAPE.
*
************************************************************************
*
*  PHASE NAME:      DOSVSDX7
*
*  MODULE NAME:     IJBXDM7
*
*-----------------------------------------------------------------------
*      CREATE STAND-ALONE DUMP PROGRAM  (VSE PART)
*-----------------------------------------------------------------------
*
*  ENTRY POINT:     CREATBEG  CREATE A STAND-ALONE DUMP RPOGRAM.
*                   THIS ROUTINE TRANSFERS CONTROL TO
*                   ONE OF THE FOLLOWING ROUTINES:
*
*                   PCHBEG:   GENERATES A STAND-ALONE DUMP
*                             PROGRAM ON CARD.
*                   DSKTCRA:  GENERATES A STAND-ALONE DUMP
*                             PROGRAM ON DISKETTE.
*                   CRATAPE:  GENERATES A STAND-ALONE DUMP
*                             PROGRAM ON TAPE.
*
```

```
*  CALLED BY:        IJBXDMP
*
*  PHASES FETCHED:   DOSVSDX8 ,DOSVSDX9
*                    THESE PHASES ARE WRITTEN ONTO DUMP TAPE
*
*  MODULES CALLED:   NONE
*
*  MESSAGES CAUSED:     NONE
*
*
*  INPUT:               NONE
*
*  OUTPUT:              STAND-ALONE DUMP PROGRAM FOR CARD,
*                       DISKETTE OR TAPE.
*
*  EXITS:               RETURN TO IJBXDMP FOR TERMINATION
*
*
*
*
*
*----------------------------------------------------------------------
*      PERFORM STAND-ALONE DUMP (STAND-ALONE PART)
*----------------------------------------------------------------------
*
*  FUNCTION:            THE GENERATED DUMP PROGRAM WRITES A VIRTUAL
*                       DUMP AND/OR REAL STORAGE ONTO A TAPE OR  PRINTER
*
*  CALLED BY:           THE STAND-ALONE DUMP PROGRAM IS IPLED FROM
*                       A TAPE OR CARD READ UNIT
*
*  PHASES FETCHED:      DOSVSDX8,DOSVSDX9
*                       THESE PHASES ARE FETCHED FROM DUMP TAPE
*                       FILE ONE
*
*  MODULES CALLED:      IJBXDM8  FOR CREATING SYMPTOM RECORD SECTION 6
*                                ENTRIES OF VSE CONTROL BLOCKS
*                       IJBXDM9  FOR CREATING SYMPTOM RECORD SECTION 6
*                                ENTRIES OF HC MESSAGES
*
*  MESSAGES CAUSED:     4G10I STAND-ALONE DUMP COMPLETE
*
*
*  MESSAGES ISSSUED:    4G10I STAND-ALONE DUMP COMPLETE
*
*  INPUT:               NONE
*  OUTPUT:              DUMP ON PRINTER
*                       DUMP FILE ON TAPE
*
```

```
*  NORMAL EXIT:        HARDWAIT PSW=X'000A000000CE0000'
*
*  EXIT-ERROR:   HARD WAIT STATE
*                PSW INDICATES ERROR AS FOLLOWS
*                X'000A000000CE0001' I/O ERROR ON SIO
*                X'000A000000CE0002' DEVICE NOT OPERATIONAL
*                X'000A000000CE0004' CHANNEL ERROR
*                X'000A000000CE0008' PERMANENT I/O ERROR
*                X'000A000000CE0010' I/O ERROR DURING ERROR
*                                    RECOVERY PROCEDURE
*                X'000A000000CE0020' END OF TAPE
*                X'000A000000CE0040' CONSOLE IO ERROR
*                X'000A000000CE0080' MACHINE CHECK
*
*                ABNORMAL CONDITIONS DURING THE EXECUTION OF DMPROG
*                INDICATED IN BYTE 7 OF THE HARD WAIT PSW.
*                IN THIS CASE THE DUMP ONTO TAPE IS NOT COMPLETE,
*                NO TM/TM AS EOF .
*                X'000A000000CE0100'  I/O ERROR DURING IPL
*                X'000A000000CE0200'  NO PAGE FRAME AVAILABLE DURING
*                                     IPL
*                X'000A000000CE0400'  PROGRAM CHECK DURING IPL
*                X'000A000000CE0800'  PROGRAM CHECK DURING PREPARING
*                                     VIRTUAL DUMP
*                X'000A000000CE1000'  PROGRAM CHECK DURING DUMPING
*                                     THE VIRTUAL STORAGE
*                X'00A0000000CE2000'  PROGRAM CHECK DURING SHIFTING
*                                     DMPROG AT THE END OF STORAGE
*                X'00A0000000CE4000'  PROGRAM CHECK DURING EXECUTION
*                                     OF IJBXDM8 OR IJBXDM9
*                X'00A0000000CE8000'  NO PAGE FRAMES FOR SHIFTING
*                                     THE DMPROG ( E-MODE)
*
*  DATA AREAS :         CONTROL-BLOCKS:  SYSCOM
*                                        COMREG
*                                        SMCB
*                                        PAGE TABLE
*                                        PAGE DATA SET TABLE (DPDTAB)
*                       SGLOWC     LOW CORE
*                       MAPDEVTY   DEVICE CODES
*                       MAPCCB     COMMAND CONTROL BLOCK
*                       MAPSCB     SPACE CONTROL BLOCK
*                       MAPSTE     SEGMENT TABLE
*                       CRTSAV     CRT SAVE AREA
*                       CRTGEN     CRT CONSTANT TABLE
*                       INLCLPT    LIBRARY POINTER TABLE
*                       IJBXSCT6   DUMP SYMPTOM RECORD DSECT
*                       DMPSCTX    GLOBAL DATA AREA FOR DOSVSDMP
*                       IJBXRC     DUMP RECORD LAYOUT
*  SEQUENCE OF OPERATION:
*                      - DUMP ON TAPE
*                        TAPEIPL        IPL-RECORD
*                        ALTDUMP        BOOTSTRAP MINI DUMP PROGRAM
*                        IPL3           ROUTINE IF LOW CORE NOT
*                                       ADDRESSABLE (ECPS:VSE MODE)
*                        DMPROG         DUMP OF PROCESSOR STORAGE.
*                                       FETCH PHASES DOSVSDX8 AND
*                                       DOSVSDX9. WRITE THEM TO TAPE.
*
*
*                      - /370 DUMP CARD/DISKETTE DUMP
*                        IPLCARD        IPL-CARD /IPL-RECORD
*                        BASEO          CARD/REC 2 -I
*                          .              .   .     I-  BOOTSTRAP MINI
*                          .              .   .     I   DUMP PROGRAM
*                        ENDIPL1        CARD/REC 50-I
*                        PRINTIO        PRINT ROUTINE
*                        DMPROG         DUMP VIRTUAL AND REAL
*                                       STORAGE
*
*                      - ECPS:VSE MODE CARD/DISKETTE DUMP
*                        IPLCRD         IPL-CARD/ IPL-RECORD
*                        RDIPL          CARD/REC 2 (READS MINI DUMP
*                                                         PROGRAM)
*                        EXIPL          MINI DUMP PROGRAM
*                        PRINTIO        PRINT ROUTINE
*                        DMPROG         DUMP VIRTUAL STORAGE
*
*
```

```
*-------------------------------------------------------------------*
*      OUTLAY OF STAND-ALONE DUMP TAPE                               *
*-------------------------------------------------------------------*
*
* FILE 1)
*     RECORD#  #OF BYTES    CONTENTS  STARTING AT LABEL
*       1         24          IPL1    TAPEIPL
*       2        256          IPL2    ALTDMP
*       3        256          IPL3    MAKE PAGES ADDRESSABLE
*       4       2048        VERBAL DESCRIPTION OF THE TAPE
*       5       4400        DUMPPROGRAM   DMPROG
*       6       7000        PHASE DOSVSDX8
*       7       7000        PHASE DOSVSDX9
*               TAPEMARK
* FILE 2)
*     RECORD#  #OF BYTES         CONTENTS
*       1       2048        STORAGE    0 -  7FF
*       2       2048        STORAGE  800 -  FFF
*       3       2048        STORAGE 1000 - 17FF
*       4       2048        STORAGE 1800 - 1FFF
*       5       2048        STORAGE 2000 - 27FF
*       6       2048        STORAGE 2800 - 2FFF
*       .         .
*      10       2048        STORAGE 4800 - 4FFF
*               TAPEMARK
* FILE 3)
*     RECORD#  #OF BYTES           CONTENTS
*       1       2056        SYMPTOM RECORD SECTION 1 AND 2
*       2       2056        SR RECORD SECTION 6 ENTRIES
*                           FOR STATUS INFORMATION
*       3       2056        STORAGE    0 -  7FF AS SAVED BY
*                                             MACHINE SAVE
*       4       2056        STORAGE  800 -  FFF
*       5       2056        STORAGE 1000 - 17FF
*       6       2056        STORAGE 1800 - 1FFF
*       7       2056        STORAGE 2000 - 27FF
*       8       2056        STORAGE 2800 - 2FFF
*      - -     - - -         - - - - - - - - -
*       N       2056        LAST 2K PAGE OF VIRTUAL STORAGE
*
*        -        -
*       N+1     2056        SYMPTOM RECORD SECTION
*       -         -         6 ENTRIES FOR VSE CONTROL BLOCKS
*       -         -         AND FOR THE LAST 200 MESSAGE LINES OF THE
*       -         -         OPERATOR CONSOLE GATHERED FROM THE
*       -         -         MESSAGE BUFFER AND THE HC-FILE
*       N+K     2056        (ONLY IF A VSE SUPERVISOR IS DUMPED)
*               TAPEMARK
*               TAPEMARK
```

```
********** END OF PROLOG ***********************************************
```

IJBXDM8: Create Symptom Record Section 6

```
***  P R O L O G  *****************************************************
*
*   VSE STAND-ALONE DUMP - CREATE SYMPTOM RECORD SECTION 6
*
************************************************************************
*
*  PHASE NAME:         DOSVSDX8
*
*  MODULE NAME:        IJBXDM8
*
*  ENTRY POINT:        IJBXDM8
*
*  FUNCTION:           CREATE SYMPTOM RECORD SECTION 6
*                      ENTRIES FOR VSE CONTROL BLOCKS
*
*  CALLED BY:          DOSVSDX7 (DMPROG)
*
*  MODULES CALLED:     I/O ROUTINE IN DMPROG
*
*  MESSAGES CAUSED:    NONE
*
*  MESSAGES ISSUED:    NONE
*
*  INPUT:              IN REGISTER R1 ADDRESS OF INTERFACE
*                      AREA.
*                      R1 --> *************************
*                             *  0  ADDRESS OF LPT    *
*                             *  4  NOT USED          *
*                             *  8  NOT USED          *
*                             *  C  ADR OF I/O ROUT.  *
*                             * 10  IOBASE SAVE AREA  *
*                             * 14  ADR OF I/O BUFFER*
*                             * 18  LENGTH OF BUFFER  *
*                             *************************
*                      REGISTER R13 ADDRESS OF SAVE AREA
*
*
*  OUTPUT:             SECTION 6 ENTRIES IN BUFFER
*
*  EXITS:              RETURN TO CALLER
```

```
*
*  SEQUENCE OF OPERATION:
*
*                    BEGIN:    ESTABLISH BASE REGISTERS
*                              SAVE CALLERS REGISTERS
*                    WRITEREC: WRITE SECTION 6 RECORD IN
*                              BUFFER . IF THE BUFFER IS
*                              FULL WRITE IT VIA I/O ROUTINE
*                              IN DMPROG ONTO TAPE
*                    MAKEENTR: MAKE A LBD ENTRY
*                    ARRAYEN:  MAKE A LBXA ENTRY
*
************ END OF PROLOG ************************************
```

IJBXDM9: Append Console Messages

```
*/** PROLOG  *********************************************************/
*/*
*/*  VSE STAND-ALONE PROGRAM - CREATE SYMPTOM RECORD SECTION 6 FOR
*/*                            HARD COPY FILE MESSAGES
*/*
*/********************************************************************/
*
*        PHASE :        DOSVSDX9
*
*        MODULE:        IJBXDM9
*
*        FUNCTION:      MOVE ABOUT 200 MESSAGE LINES FROM
*                       HARD COPY FILE TO THE END OF SA DUMP
*                       IN FORM OF SECTION 6 ENTRIES OF
*                       SYMPTOM RECORD
*
*        EXTERNAL SERVICES:
*                       WRTFORC  --- WRITE CURRENT HCFBUFFER
*                                    TO HARDCOPY-FILE
*                       HCFREAD  --- READ ONE RECORD FROM HCF
*                       SKIPHCF  --- SKIP ONE OR MORE LOG. REC
*                       MODHCF   --- CHANGE DIRECTION OF READ
*                       HCFUPD   --- UPDATE CURRENT DISK ADDR
*                                    IN HCFCB
*      MESSAGES CAUSED :
*             4F02I HARD COPY FILE NOT OPEN
*             4F03I INVALID EXTENTS IN HCFCB
*             4F04I INVALID CURRENT DISK ADDRESS IN HCFCB
*             4F05I HARD COPY FILE DISK NOT READY|OPERATIONAL
*             4F06I READ ERROR DURING FIRST READ FROM HARD COPY FILE
*             4F07I INTERNAL ERROR DURING POSITIONING OF MSGPTR
*             4F08I UNDEFINED ERROR
*             4F09I ERROR DURING WRITE TO HARD COPY FILE
*             4F10I DIFFERENT MSG LGTH FOUND - CONSOLE MSG LGTH USED
*             4F11I READ ERROR DURING READ FROM HARD COPY FILE
*
*        REGISTER USED:
*                       RC  BASE REGISTER
*
*                       RA  ADDR OF CRTSAV
*                       RB  ADDR OF HCFCB
*                       RD  ADDR OF SAVEAREA
*                       RF  RETURN CODES FROM HCF-SERVICE
*
*        WORK-REGISTER:
*                       R0  IOAREA OF HCF READ
*                       R2
*                       R3
*                       R4
*                       R5
*                       R6
*                       R7
*                       R8  -
*                            - FOR BRANCHING
*                       R9  -
*
*
*        INPUT:         CALLED VIA REG. 15
*                       R01  POINTS TO INTERFACE BLOCK
*                       R14  RETURN ADDRESS
*
*        RETURN CODES:  R0
*
*        EXIT NORMAL :  RETURN TO CALLER
*
*        ERROR EXITS:   NONE
*
**** END OF PROLOG *****************************************************
```

DUMP UTILITY: DATA AREA INFORMATION AND MACROS

COMMON DATA AREA DMPSCTX

The interface control blocks for the intermodule communication between the different stand-alone phases are generated by the macro DMPSCTX. The macro DMPSCTX contains the following data:

- I/O area addresses
- return information from logical I/O module (IJBXDM1)
- device type flags and dump device constants
- function codes for dump retrieval
- function codes for I/O on SYSLOG and SYSLST
- equates for symbolic addressing of messages
- operation codes for printing

Format of DMPSCTX

```
  [label] DMPSCTX DDMP = {DSECT|CSECT}
```

Keyword value CSECT: generates the common data area as CSECT

DSECT: generates the common data area as DSECT

Reference List of DMPSCTX

Defined in	Referenced in	Modified in
IJBXDMP	IJBXDMP	IJBXDMP
	IJBXDM1	IJBXDM1
	IJBXDM2	IJBXDM2
	IJBXDM5	IJBXDM5
	IJBXDM7	

DOSVSDMP MACROS

Logical I/O Macro IJBXLIO

The macro IJBXLIO provides an interface to the logical dump retrieval routine, the SYSLST routine, the SYSLOG routine and the message handler routine.

All modules which use the logical I/O macro must contain the common data area as DSECT or CSECT.

Format of IJBXLIO

```
  [label] IJBXLIO  FUNCT = { READDATA |
                             READSYM  |
                             READNEXT |
                             OPENDUMP |
                             CLOSEDUMP|
                             POSITION |
                             OPENLST  |
                             CLOSELST |
                             PRLINE   |
                             PRMSG    |
                             CONSMSG  |
                             CONSLINE |
                             READLOG }
                          [,MSG = msgnumber]
                          [,SAVEAREA = {name|(Rx)}]
```

Description of IJBXLIO

Operand	Keyword	Description
FUNCT	READNEXT	Reads next record.
	OPENDUMP	Opens the dump medium.
	CLOSEDUMP	Closes the dump medium.
	POSITION	Positions tape to file start.
	OPENLST	Opens SYSLST.
	CLOSELST	Closes SYSLST.
	PRLINE	Prints contents of print I/O area.
	PRMSG	Prints a specified message.
	CONSMSG	Writes a specified message or menu to the console.
	CONSLINE	Writes a line on SYSLOG.
	READLOG	Reads operator response from SYSLOG.
MSG	msg-number	Specifies a message to print on SYSLST or to display on SYSLOG (valid only if PRMSG or CONSMSG specified).
SAVAREA	name	Specifies the name of a register save area.
	(Rx)	Specifies a register which contains the address of a register save area. If the SAVAREA operand is omitted register 13 must point to a save area.

IJBXRC

The macro IJBXRC contains the layout of the dump record.

Format:

```
  [label] IJBXRC
```

This macro is referenced by IJBXDM5 and IJBXDM7.

IJBXSCT6

The macro IJBXSCT6 contains the layout of the symptom record.

Format:

```
  [label] IJBXSCT6
```

This macro is referenced by IJBXDM7 and IJBXDM8.

IDUMP Macro and Symptom String

To improve the quality of the problem determination aids the Symptom String/Auto Debug Architecture has been defined for software service. This architecture allows any subsystem or program detecting a problem to pass a set of error symptoms and dumping instructions to the system control program.

- All subsystems and system programs may implement error exit routines which pass error symptoms and dumping instructions via an internal dump macro (IDUMP macro) to the VSE serviceability routines. The serviceability routines include the symptoms passed via the IDUMP macro into the dump data.

IDUMP

In order to request a dump and to communicate the symptom record to the VSE system, the VSE components use the IDUMP macro.

The issuer of the IDUMP macro defines a symptom record containing all error symptoms which might be helpful for problem source identification. In addition, the user defines the area to be dumped.

FORMAT OF THE IDUMP MACRO

The format of the IDUMP macro is as follows:

```
    IDUMP {addr1|(r1)}
          ,{addr2|(r2)}
          ,{addr3|(r3)}
          [,MFG={param|(S,param)|(r4)}]
```

addr1 This parameter specifies the start address of the area in the issuer's partition that is to be dumped. If the address is lower than the partition start address, it is adjusted to partition start. If it is higher than the partition end address, no area is dumped.

addr2 This parameter specifies the end address of the area to be dumped. If it is higher than the partition end address, it is adjusted to partition end. If it is lower or equal to addr1, no area is dumped.

addr3 The address specified by this parameter points to a parameter list. This parameter list contains interface information and an address pointer to the symptom record. The parameter list and the symptom record may be located in the issuer's partition or in the system GETVIS area. If the address contains an invalid value, no user information is put into the header of the new dump member.

MFG This parameter is optional and can be used to provide an area where the other three parameters are stored to make the issuing program reentrant.

The contents of registers zero and one is changed by the IDUMP macro.

IDUMP RETURN CODES:

If a return code is requested (bit 1 of the flag byte in the parameter list is on), it is passed to the issuer of the IDUMP macro in register 15. If the mentioned flag bit is off, register 15 remains unchanged.

The following return codes may be encountered:

0	request completed successfully
4	dump library full or dump library not defined
8	library error (I/O error or OPEN/CLOSE error)
12	invalid parameter list or invalid symptom record

If the Job Control option NOSYSDMP is active only the return codes zero and twelve may be encountered.

If library problems are encountered (return codes 4 and 8) the dump fragment is purged from the dump library. The dump library will never contain incomplete dump members. In this case the system tries to print the dump on SYSLST (but only if the user does not accept a return code).

DESCRIPTION OF THE PARAMETER LIST

This area must contain information to tell VSE how to handle the IDUMP just issued. This information is not put into the dump header.

The layout of the area is as follows:

OFFSET	TYPE	LENGTH	DENOTATION
0	char	4	Type of IDUMP
4	hex	1	Flag byte 1: X'80' reserved X'40' if a return code is to be passed in register 15. other bits: reserved
5	hex	3	Address of save area for PSW and registers
8	hex	2	length of this IDUMP parameter list
10	hex	1	flag byte 2 X'80' do not add additional supervisor dump data
11	hex	1	flag byte 3
12	hex	4	reserved (X'00000000')
16	char	8	name of this dump member
24	char	8	reserved (X'4040404040404040)
32	hex	2	reserved (x'0000')
34	hex	2	length of symptom record
36	pointer	4	address of symptom record

Type of IDUMP

This field is used as a control block identification. It contains the character string 'ONLY' to distinguish the new IDUMP macro from the IDUMP macro of former releases.

Address of VSE save area.

This parameter is useful, if the IDUMP is issued from a user exit routine.

The IDUMP routine adds the user task's PSW and registers as a section 6 entry to the user supplied symptom record. Normally VSE takes this information from the running task's save area.

If the IDUMP is given from a user exit routine (e.g.an AB exit routine), then the PSW and registers of the user exit routine are often worthless for debugging. Instead the issuer of the IDUMP may want to have dumped the PSW and registers of the failing routine. If the IDUMP is issued from a special exit routine, then you may specify here the save area address of the failing routine. The format of the save area must be equal to the format of the VSE partition save area: It contains an 8-byte program name followed by the 8-byte PSW, followed by a 64-byte field containing the registers 9 to 8.

In all other cases specify binary zeros instead of a save area address.

Length of this IDUMP parameter list

This field is inserted to allow future extensions of this parameter list. Specify a halfword containing the value of 40.

Flag byte 2

Set bit zero to binary one, if supervisor data should not be added to the IDUMP. If bit zero is on, then only the following is dumped:

> the area defined by addr1 and addr2 of the IDUMP macro, and
>
> the pages containing control blocks specified in section 6 of the user supplied symptom record.

Name of this dump member

Dump member names are created by the VSE system. The generated dump name is returned to the issuer of the IDUMP macro in this eight-byte name field. Leave this field blank if you issue an IDUMP macro.

Length of symptom record

This is length of the symptom record excluding additional section-6's if any.

Address of symptom record

This is a four-byte address pointer to the area where the issuer of the IDUMP has created the symptom record. It is the responsibility of the user to supply a complete symptom record, except for section 1 which is filled in by the VSE dump routines.

The issuer of the IDUMP should specify binary zeros in section 1, except for bytes 0 and 1, that must contain the characters 'SR' to identify the start of the symptom record.

The symptom header record may be followed by one or more extension records. These extension records are used to add more section 6 entries to the dump. The block size of the header record and the section 6 extension records must not exceed 2048 bytes.

Each section 6 extension record contains a 16-byte chain field followed by the control block locating definitions.

The layout of the 16 byte-chain field is as follows:

- 4 bytes characters 'SCT6' to identify the start of section 6.
- 4 bytes of zeros or address of an additional section 6 in storage if 2K were not enough to keep the complete symptom record for this one IDUMP request.
- 4 bytes of zeros or length of next section 6.
- 4 bytes of zeros.

The address and length, if specified, must describe an area in the partition of the issuer of the IDUMP macro or in the system GETVIS area. Otherwise they are treated as zeros.

The area that the address is pointing to must contain a complete section 6. The maximum length of this section 6 is again 2K bytes. The 16 bytes at the beginning of this section 6 must also follow the rules described above. There is no limitation to the number of section 6 chained together for one IDUMP request.

SPECIFICATION OF STORAGE TO BE DUMPED

The IDUMP macro dumps the area of storage which is deliminated by the first and second IDUMP parameter. In addition VSE dumps the pages which contain control blocks specified by section 6 control block locators. However, it will only consider those addresses and lengths that are found in the section 6 entries of the type 'LBD '.

VSE will not explicitely include those addresses that are found in 'alternate header' entries, array extension entries, nor will it follow chains, cascades, etc. to locate the storage areas that are to be dumped.

Therefore, if a component describes its control blocks by other entries than 'LBD ' entries, that component should also describe the storage area to be dumped separately. This will ensure that complete dump information is available to cover these control blocks.

DUMP UTILITY: DIAGNOSTIC AIDS

MESSAGE CROSS-REFERENCE

4G01I	IJBXDMP
4G02I	IJBXDMP
4G04D	IJBXDMP
4G05D	IJBXDMP
4G06D	IJBXDMP
4G09I	IJBXDMP
4G10I	IJBXDM7
4G11I	IJBXDMP
4G12I	IJBXDMP
4G13I	IJBXDMP
4G15I	IJBXDMP
4G16I	IJBXDMP
4G17I	IJBXDM5
4G18I	IJBXDM5
4G19I	IJBXDMP
4G20I	IJBXDM5
4G23I	IJBXDMP
4G25I	IJBXDMP
4G26I	IJBXDMP
4G28I	IJBXDM1

4G29I	IJBXDM1
4G30D	IJBXDMP
4G31I	IJBXDMP
4G32I	IJBXDMP
4F02I - 4F11I	IJBXDM9

DUMP UTILITY MENUS

Menu 1

```
XX 4G01D  SELECT THE ACTIVITY YOU WANT TO PERFORM
......................................................
1 CREATE SA DUMP PROGRAM ON TAPE
2 PRINT DUMP TAPE
3 PRINT SDAID TAPE
R END DOSVSDMP PROCESSING
(ENTER ONE OPTION ONLY)
```

Menu 2

```
XX 4G02D  SPECIFY THE TYPE OF DUMP
.....................................................
1 STAND-ALONE DUMP ON TAPE
2 DUMP COMMAND ON TAPE
R END DOSVSDMP PROCESSING
(ENTER ONE OPTION ONLY)
```

HARD WAIT CODES

If the execution of the generated dump program must be terminated one of the following hard wait codes are issued by the dump program.

```
HARD WAIT STATE
PSW INDICATES ERROR AS FOLLOWS
X'000A000000CE0001' I/O ERROR ON SIO
X'000A000000CE0002' DEVICE NOT OPERATIONAL
X'000A000000CE0004' CHANNEL ERROR
X'000A000000CE0008' PERMANENT I/O ERROR
X'000A000000CE0010' I/O ERROR DURING ERROR
                    RECOVERY PROCEDURE
X'000A000000CE0020' END OF TAPE
X'000A000000CE0040' CONSOLE I/O ERROR
X'000A000000CE0080' MACHINE CHECK
X'000A000000CE0100' I/O ERROR DURING IPL
X'000A000000CE0200' NO PAGE FRAME AVAILABLE DURING IPL
X'000A000000CE0400' PROGRAM CHECK DURING IPL
X'000A000000CE0800' PROGRAM CHECK DURING PREPARING
                    VIRTUAL DUMP
X'000A000000CE1000' PROGRAM CHECK DURING DUMPING
                    THE VIRTUAL STORAGE
X'000A000000CE2000' PROGRAM CHECK DURING SHIFTING
                    DMPROG AT THE END OF STORAGE
X'000A000000CE4000' PROGRAM CHECK DURING EXECUTION
                    OF IJBXDM8 OR IJBXDM9
X'000A000000CE8000' NO PAGE FRAMES FOR SHIFTING
                    THE DMPROG ( E-MODE)
```

DUMP ANALYSIS PROGRAMS

INTRODUCTION

The VSE Dump Analysis Programs IJBXDBUG and IJBXSDA are written to do post-dump analysis for VSE stand-alone dumps.

The data produced by these programs will assist the user in recognizing errors and duplucate problems without printing a dump and without analysis work by experts.

IJBXDBUG and IJBXSDA are implemented in the form of exit routines to Info/Analysis. They will be invoked by Info/Analysis during an interactive session or when the hatch functions are executed.

INFO/ANALYSIS passes dump data to the Dump Analysis Programs. Results of the dump analysis are returned to INFO/ANALYSIS for further processing or to be displayed on the terminal.

The dump analysis routines return the analysis data in the form of LBXT entries (text extension entries). These LBXT entries are added to the dump member in the library. They will remain available for other INFO/ANALYSIS sessions.

The INFO/ANALYSIS user may request an overview list showing the names of all control blocks (LBD), text data (LBXT) and hex data (LBXX). The user may select any control block or text block out of that list to be displayed on the terminal. For batch mode the user will print the formatted dump which will include the analysis program output.

LBXT entries built by analysis routines will get a special flag to distinguish them from pre-dump analysis data. Moreover the entries built by the dump analysis routines will get the name IJBXDBUG or IJBXSDA in the component identification field to distinguish them from other post-dump analysis data.

DUMP ANALYSIS PROGRAM IJBXDBUG

IJBXDBUG extracts environmental information, status information and error information. It looks for incorrect pointer data, inconsistent system status, bound states, etc.

IJBXDBUG supplies general information, which can be extracted from all stand-alone dumps, and specific information which is only available in distinct error situations.

The following data will be supplied on all dumps:

- Service level identifier
- Supervisor id
- Supervisor name
- Date the dump was taken
- Dump type
- System status
- Current task
- Owner of LTA and transient name (if active)
- DOC Screen image buffer (if applicable)
- ASYNOC reply status (if applicable).

MESSAGES

The Dump Analysis Program does not produce error messages for errors in the dump. Any unusual condition detected in the stand-alone dump data is returned to INFO/ANALYSIS in the form of an LBXT entry. These 'messages' are available to the system analyst as dump analysis data. The Dump Analysis Program IJBXDBUG may produce error messages which indicate errors with the interface to INFO/ANALYSIS. These messages indicate internal errors in the Dump Analysis Program or INFO/ANALYSIS.

4G80I IJBXDBUG analysis output already exists for this dump.
IJBXDBUG terminated.

4G82I Dump Analysis Routine 'IJBXDBUG' completed successfully.

4G83I IJBXDBUG Call Error. Reason code: X.

4G87I Nonzero return code from INFO/ANALYSIS Dump Access.
Return code: XX. Reason code: XX.

4G88I Nonzero return code from INFO/ANALYSIS symptom record update.
Section: X. Return code: XX. Reason code: XX.

4G89I IJBXDBUG internal symptom record update error.
Invalid section number: X.

4G90I   Internal error in IJBXDBUG or dump file. More than 15 LBD's built.

These messages will be written by the analysis routine using the INFO/ANALYSIS print routine.

Additional information type messages:

IJBXDBUG cannot analyze non-VSE dumps.

IJBXDBUG cannot analyze XXXX type dumps.
(XXXX equals dump type.)

Save system data not available.

Low core not available in this dump. IJBXDBUG cannot continue.

SYSCOM not available in this dump. IJBXDBUG cannot continue.

STORE STATUS not done.

Problem program begin address is zero.

SYSCOM indicates IPL in progress.

Program old PSW indicates BC mode.

Dump indicates no free channel queue entries.

Instruction length code zero.

SYSCOM address is zero. IJBXDBUG cannot continue.

Comparing failing instruction against current running system indicates possible overlay, instruction currently: XXXXXX

Unable to locate BG COMREG.

Invalid address XXXXXX encountered during analysis.

Address of: (Area / Block name)

Address referenced 500 times.

Possible loop within control blocks.

LOGIC

This section is intended to supply information pertaining to the internal logic of IJBXDBUG.

Introduction

IJBXDBUG is called via the INFO/ANALYSIS "call exit" function of dump viewing. INFO/ANALYSIS passes a parameter list to the exit. This list supplies the exit with the addresses of the INFO/ANALYSIS Dump Access Routine, Symptom Record Access Routine, Symptom Record Update Routine, Print Routine, and other various pointers. IJBXDBUG uses these addresses for the INFO/ANALYSIS interface. When requesting a service from INFO/ANALYSIS a parameter list is passed back to INFO/ANALYSIS for that particular service. For more information reference INFO/ANALYSIS documentation.

IJBXDBUG analyzes the dump supplied by INFO/ANALYSIS by requesting portion of the dump data using the INFO/ANALYSIS Dump Access Routine. As the data is analyzed symptoms are placed in section 5 of the dump symptom record using the INFO/ANALYSIS Symptom Record Update Routine. As the section 5 symptoms are being built text data is also being built. These text data entries are passed to INFO/ANALYSIS via the Symptom Record Update Routine to be placed in section 6 of the dump. If an error occurs during section 5 update, no additional section 5 updates will be attempted. IJBXDUG will continue to build text entries for section 6. On section 5 and/or section 6 update errors, a message will be issued listing the section number, the return code, and the reason code which INFO/ANALYSIS generated.

Macros

The following macros are required to assemble IJBXDBUG:

GSXVALID	Internal macro for address validation
GSXADDR	Internal macro to get dump data from INFO/ANALYSIS
GSXSYMAC	Internal macro to pass LBD to INFO/ANALYSIS
IJBXSCT6	Symptom record DSECT's
SGLOWC	Supervisor low core area
MAPPCB	Partition control block
MAPTIB	Task information block
MAPTCB	Task control block

SYSCOM	System communication area
MAPCOMR	Partition communication region
MAPPUB	Physical unit block table
MAPPIB	Partition information block
MAPCCB	Command control block
SGCHQ	Channel queue table
CRTGEN	Display operator console table
CRTSAV	Display operator console area
ASYTAB	ASYNOC table
ORE	ASYNOC reply table
SLADCT	Label area
ERBLOC	Error block
MAPSAVAR	Save areas
MAPSVAHD	SVA header
INLCDENT	SDL entry
MAPRASL	RAS linkage area
MAPRTAB	RAS table
SDAGDT	SDAID global table

```
* ------------------------------------------------------------------ *
* GSXVALID MACRO - USED TO VALIDATE ADDRESSES                        *
* ------------------------------------------------------------------ *
*                                                                    *
* FORMAT - GSXVALID REG,FLAG,WORD1,WORD2,WORD3,WORDN                 *
*          REG   = REGISTER WHICH CONTAINS ADDRESS TO VALIDATE       *
*          FLAG  = MSG OR NOMSG                                      *
*                  MSG   = ISSUE INVALID ADDRESS MESSAGE IF FIELD    *
*                          VALID IS NON ZERO (VALID DESCRIBED BELOW).*
*                  NOMSG = DO NOT ISSUE MESSAGE ON NON ZERO VALID    *
*          WORD1 = NAME OF AREA POINTED TO BY REGISTER. UP TO 10     *
*                  WORDS CAN BE SPECIFIED. WORDS MUST BE SEPARATED   *
*                  BY COMMAS. MAXIMUM LENGTH OF WORDS IS DECIMAL 32. *
*                                                                    *
* OUTPUT - REGISTER F AND FIELD 'VALID' CONTAIN RETURN CODE          *
*          RETURN CODE = 00 ADDRESS IN SUPERVISOR                    *
*                        01 ADDRESS IN OVERLAYED PART OF SUPERVISOR  *
*                           BELOW X'10', FLOG, OR DUMPAREA (OVERAREA)*
*                        02 ADDRESS IN PARTITION AREA (PARTAREA)     *
*                        04 ADDRESS IN SVA AREA (SVAAREA)            *
*                        08 ADDRESS UNKNOWN (UNKWAREA)               *
*                        10 ADDR REF'D 500 TIMES IN A ROW (LOOPFLGA) *
*                        20 AREA REF'D 1000 TIMES IN A ROW (LOOPFLGB)*
*                        CODE 10 OR 20 MAY BE COMBINED WITH OTHER    *
*                        CODES. ALL CODES ARE IN HEX.                *
*                                                                    *
* MESSAGE = INVALID ADDRESS XXXXXX ENCOUNTERED DURING ANALYSIS.      *
*           ADDRESS OF:  (BLOCK/AREA NAME)                           *
*                                                                    *
* ------------------------------------------------------------------ *
**********************************************************************
```

```
**********************************************************************
* ------------------------------------------------------------------ *
* GSXADDR MACRO - USED TO REQUEST DATA FROM INFO/ANALYSIS (DUMP)     *
* ------------------------------------------------------------------ *
*                                                                    *
* FORMAT - GSXADDR REG,LENGTH,BUFFER                                 *
*          REG    = REGISTER WHICH CONTAINS ADDRESS OF DATA          *
*          LENGTH = LENGTH OF DATA REQUESTED.  MAXIMUM IS X'800'.    *
*          BUFFER = NAME OF BUFFER TO RETURN DATA IN OR 0            *
*                   IF 0 IS SPECIFIED THE DATA IS ONLY AVAILABLE     *
*                   UNTIL THE NEXT ENTRY TO ROUTINE GSXADDR VIA THIS *
*                   MACRO OR VIA DIRECT BRANCH.                      *
*                                                                    *
* OUTPUT - REGISTER F CONTAINS RETURN CODE                           *
*          RETURN CODE = 00 DATA AVAILABLE                           *
*                        OF DATA NOT AVAILABLE                       *
*                                                                    *
*          ADDRESS OF THE BUFFER CONTAINING THE DATA IS RETURNED IN  *
*          THE REGISTER SPECIFIED IN 'REG' PARAMETER.                *
*                                                                    *
* MESSAGE = ALL OR PART OF REQUESTED DATA NOT AVAILABLE FROM         *
*           DUMP FILE.                                               *
*           ADDRESS OF REQUESTED DATA: XXXXXX                        *
*           LENGTH OF REQUESTED DATA: XXXXXX                         *
*           ADDRESS OF:  (BLOCK/AREA NAME)                           *
*                                                                    *
*           (IF GSXADDR ISSUES THIS MESSAGE THE BLOCK/AREA NAME      *
*           IS GOTTEN FROM THE FIELD 'LASTBLK'. THIS FIELD IS SET    *
*           BY THE GSXVALID MACRO. THEREFORE, IF THE ADDRESS         *
*           REQUESTED BY GSXADDR IS NOT THE ADDRESS OF THE LAST      *
*           AREA VALIDATED BY GSXVALID, THE FIELD 'LASTBLK' SHOULD   *
*           BE CLEARED AND SET TO THE PROPER NAME JUST PRIOR TO      *
*           ISSUING GSXADDR MACRO OR BRANCHING TO GSXADDR ROUTINE.)  *
*                                                                    *
* ------------------------------------------------------------------ *
* ------------------------------------------------------------------ *
* GSXSYMAC MACRO - USED TO REQUEST SECT 6 SYMPTOM FROM INFO/ANALYSIS *
* ------------------------------------------------------------------ *
*                                                                    *
* FORMAT - GSXSYMAC TYPE,LBD,BLOCK                                   *
*          TYPE  = TYPE OF REQUEST                                   *
*                  B = SECTION 6 RECORDS                             *
*                  E = ENTIRE SYMPTOM RECORD                         *
*          LBD   = TYPE OF SECTION 6 LBD REQUESTED                   *
*          BLOCK = NAME OF SECTION 6 BLOCK REQUESTED                 *
*                                                                    *
* OUTPUT - REGISTER F CONTAINS RETURN CODE                            *
*          RETURN CODE = 00 RECORD AVAILABLE                         *
*                        OF RECORD NOT AVAILABLE                     *
*                                                                    *
*          REGISTER 1 POINTS TO THE RECORD REQUESTED.                *
*                                                                    *
* ------------------------------------------------------------------ *
```

Flow of Control

```
* ------------------------------------------------------------------ *
*                                                                    *
*         IJBXDBUG - VSE/AF 2.1.0 POST DUMP ANALYSIS MODULE          *
*                                                                    *
* ------------------------------------------------------------------ *
* ------------------------------------------------------------------ *
*                                                                    *
*  MODULE NAME:        IJBXDBUG                                      *
*                                                                    *
*  ENTRY POINT:        IJBXDBUG                                      *
*                                                                    *
*  FUNCTION:           DO POST DUMP ANALYSIS ON STAND-ALONE DUMPS AND *
*                      SYSTEM DUMPS (DUMP COMMAND) OF THE SUPERVISOR *
*                      PRODUCE A MINIMAL AMOUNT OF MEANINGFUL        *
*                      OUTPUT FOR PROBLEM DETERMINATION, PROBLEM     *
*                      SOURCE IDENTIFICATION, AND/OR PROBLEM         *
*                      RESOLUTION.                                   *
*                                                                    *
*  CALLED BY:          INFO/ANALYSIS CALL EXIT COMMAND               *
*                                                                    *
*  REGISTER USAGE:     R0 - N/A                                      *
*                      R1 - PARAMETER LIST POINTER                   *
*                      R2 - POINTER TO LOW CORE BUFFER               *
*                      R3 - BASE FOR SECTIONS MAIN, WAITFFF, ETC.    *
*                      R4 - BASE 1 FOR SUBROUTINE & CONSTANT SECTIONS *
*                      R5 - BASE 2 FOR SUBROUTINE & CONSTANT SECTIONS *
*                      R6 - WORK REGISTER                            *
*                      R7 - WORK REGISTER / SYSCOM                   *
*                      R8 - WORK REGISTER / TIB                      *
*                      R9 - WORK REGISTER                            *
*                      RA - WORK REGISTER                            *
*                      RB - WORK REGISTER                            *
*                      RC - WORK REGISTER                            *
*                      RD - SAVE AREA POINTER / WORK REGISTER        *
*                      RE - LINK REGISTER                            *
*                      RF - RETURN CODES & INFO/ANALYSIS ROUTINES    *
*                                                                    *
*  PATCH AREA LABEL:   PATCH AND PATCH2                              *
*                                                                    *
*  MESSAGES CAUSED:    BLN9003I                                      *
*                                                                    *
*  MESSAGES ISSUED:    4G80I THROUGH 4G90I                           *
*                                                                    *
*  INPUT:              R1 POINTS TO INFO/ANALYSIS PARAMETER LIST PTR  *
*                                                                     *
*  OUTPUT:             OUTPUT WILL BE IN ONE OF TWO FORMS,            *
*                      LBD ENTRIES(SECTION 5 AND/OR 6) OR PRINT.      *
*                      THE TYPE OF OUTPUT IS PRINT ONLY IF            *
*                      INFO/ANALYSIS DOES NOT SUPPLY A SYMPTOM RECORD *
*                      UPDATE ROUTINE FOR ADDING LBD'S TO SECTION 5   *
*                      AND 6 OF THE DUMP.                             *
*                                                                     *
*  LBD NAMES:          DBUGHDR - BASIC INFO (STATUS, ETC.)            *
*                      DBUGR07 - REGISTERS 0 THRU 7 FOR HARD WAITS    *
*                      DBUGR8F - REGISTERS 8 THRU F FOR HARD WAITS    *
*                      DBUGSCR - CRT SCREEN BUFFER IF APPLICABLE      *
*                                                                     *
*  MACRO'S USED:       GSXVALID - INTERNAL ADDRESS VALIDATION         *
*                      GSXADDR  - GET DUMP DATA FROM INFO/ANALYSIS    *
*                      GSXSYMAC - REQUEST SYMPTOM RECORD DATA         *
*                      IJBXSCT6 - SYMPTOM RECORD DSECTS               *
*                      MAPPING  - SEE SECTION SYSTEM OWNED DSECTS     *
*                                                                     *
*  NORMAL EXIT:        PLACE TERMINATION MESSAGE IN INFO/ANALYSIS     *
*                      MESSAGE BUFFER.                                *
*                      RETURN TO INFO/ANALYSIS VIA REGISTER 14.       *
*                                                                     *
*  ERROR EXIT:         PRINT ERROR MESSAGE AND/OR PLACE MESSAGE IN    *
*                      INFO/ANALYSIS MESSAGE BUFFER.                  *
*                      RETURN TO INFO/ANALYSIS VIA REGISTER 14.       *
*                                                                     *
* ------------------------------------------------------------------- *
* ------------------------------------------------------------------- *
*                                                                     *
*  FLOW OF CONTROL:                                                   *
*                                                                     *
*       GSXENTRY:      PROVIDE ADDRESSABILITY                         *
*                      CHECK PARAMETERS PASSED BY INFO/ANALYSIS       *
*                      MAKE SURE THAT A SUPPORTED VSE DUMP            *
*                      HAS BEEN PASSED FOR ANALYSIS.                  *
*                      ON ERRORS RETURN VIA INFOERR.                  *
*                                                                     *
*       READCORE:      READ IMPORTANT DATA AREAS AND CONTROL BLOCKS   *
*                      DO PLAUSIBILITY CHECKING                       *
*                      SET VALUES FOR ADDRESS VALIDATION              *
*                      ON PROBABLE ERRORS PROVIDE ERROR               *
*                      MESSAGE IN LBXT FORMAT.                        *
*                                                                     *
*       CKSTSTAT:      CHECK STATUS OF DUMPED SYSTEM.                 *
*                      BRANCH TO APPROPRIATE ROUTINE:                 *
*                          NOWAIT:   RUNNING SYSTEM, LOOP, ETC.       *
*                          SOFTWAIT: SOFTWAIT STATE                   *
*                          HARDWAIT: HARDWAIT FFF                     *
*                          NONFFF:   NON FFF HARDWAIT.                *
*                                                                     *
*      NOWAIT:        GIVE INFORMATION MESSAGE AND CURRENT PSW PTR   *
*                     THEN CONTINUE VIA SOFTWAIT                      *
*                                                                     *
*      SOFTWAIT:      DISPLAY STATUS OF ACTIVE NON TP DEVICES         *
*                     DISPLAY STATUS OF ACTIVE TASKS                  *
*                     ANALYZE BOUND CONDITIONS                        *
*                                                                     *
*      HARDWAIT:      DISPLAY WAIT CODE, INTERRUPT CODE,              *
*                     FAILING ADDRESS, FAILING INSTRUCTION,           *
*                     GIVE ADDITIONAL INFO IF FAILURE                 *
*                     IN PTA, LTA, CRT, RTA, SVA.                     *
*                     DISPLAY CURRENT TASK,                           *
*                     DISPLAY REGISTER VALUES AND DATA.               *
*                                                                     *
*      NONFFF:        DISPLAY WAIT CODE, CURRENT TASK,                *
*                     ADDITIONAL INFORMATION FOR SPECIFIC             *
*                     TYPE WAITS (SENSE, PHASE NOT FOUND, ETC.)       *
*                     DISPLAY REGISTER VALUES AND DATA.               *
*                                                                     *
* SUBROUTINES:                                                        *
*                                                                     *
*      CRTSCRN:       DISPLAY CRT SCREEN IMAGE BUFFER                 *
*      GSXADDR:       GET A STORAGE INTERVAL FROM INFO/ANALYSIS       *
*      GSXSYMAC:      RETRIEVE LBD ENTRIES FROM INFO/ANALYSIS         *
*      GSXSYMUP:      UPDATE LBD ENTRIES                              *
*      GSXSYMS3:      ADD A SYMPTOM TO SECTION 3                      *
*      GSXSYMS4:      ADD A SYMPTOM TO SECTION 4                      *
*      GSXSYMS5:      ADD A SYMPTOM TO SECTION 5                      *
*      REGDATA:       DISPLAY REGISTER VALUES AND DATA                *
*      FINDTASK:      FIND TASK NAME                                  *
*      SVAFAIL:       CHECK FOR FAILURE IN SVA                        *
*      APARSCAN:      SCAN PHASE FOR APAR ID (DYNNNNN)                *
*      LTAACTIV:      CHECK FOR LTA ACTIVE                            *
*      ACTTID:        FIND CURRENT TASK                               *
*      CHECKORE:      CHECK FOR REPLIES OUTSTANDING                   *
*      FDBGCMRG:      FIND BG-COMREG                                  *
*      VALADDR:       VALIDATE DUMP ADDRESS                           *
*      GSXPRT1,2,3:   PRINT DATA OR ADD DATA TO LBD                   *
*      GSXPRINT:      PRINT DATA ONLY                                 *
*      INFOCALL:      EXIT TO INFO/ANALYSIS FOR SERVICE/FUNCTION      *
*      INFOERR:       EXIT TO INFO/ANALYSIS ERROR MSG PASSED          *
*      INFOEXIT:      EXIT TO INFO/ANALYSIS AT IJBXDBUG TERMINATION  *
*                                                                     *
* ------------------------------------------------------------------- *
```

STAND-ALONE DUMP ANALYSIS ROUTINE IJBXSDA

The Info/Analysis exit routine IJBXSDA formats the SDAID buffer in a stand-alone dump, if SDAID was active at the time the stand-alone dump was taken. IJBXSDA decompresses the encoded SDAID records and converts them to readable format. The trace entries retrieved from the SDAID buffer are added as text data to the symptom record of the stand-alone dump. The exit routine may format SDAID buffers up to a size of 26K bytes. If a bigger SDAID buffer is allocated, IJBXSDA only formats the last recent 26K of the SDAID buffer. The text entries are afterwards available for an inspection on the ISPF screen or they may be printed as part of the PRINT FORMAT option. The name of the new symptom is 'SDAID00n' (n = 1 to 3), the component-id is 'IJBXSDA' and the symptom type is 'TEXT DATA'.

Module Description

```
* -------------------------------------------------------------------- *
*         IJBXSDA - VSE/AF 2.1.7 INFO/ANALYSIS EXIT ROUTINE           *
*                   FOR FORMATTING TRACE RECORDS                       *
*                   IN A STAND-ALONE DUMP                              *
* -------------------------------------------------------------------- *
* -------------------------------------------------------------------- *
*                                                                      *
*  MODULE NAME:        IJBXSDA                                         *
*                                                                      *
*  ENTRY POINT:        IJBXSDA                                         *
*                                                                      *
*  FUNCTION:           IJBXSDA FORMATS THE SDAID BUFFER IN A           *
*                      STAND-ALONE DUMP                                *
*                      (IF SDAID WAS ACTIVE AT THE TIME WHEN THE       *
*                      STAND-ALONE DUMP WAS TAKEN).                    *
*                      THIS ALLOWS TO SEE THE VERY LAST ACTIONS        *
*                      OF THE SYSTEM BEFORE THE SA-DUMP WAS TAKEN.     *
*                                                                      *
*  CALLED BY:          INFO/ANALYSIS AS AN EXIT ROUTINE                *
*                                                                      *
*  INPUT PARAMETERS:   R1 POINTS TO THE ADDRESS OF THE INFO/ANALYSIS   *
*                      PARAMETER LIST ADSPXRPL.                        *
*                                                                      *
*  INPUT:              IJBXSDA RETRIEVES ITS INPUT DATA FROM THE       *
*                      VSE DUMP LIBRARY VIA THE INFO/ANALYSIS          *
*                      INTERFACE ROUTINES.                             *
*                      THE ADDRESS AND THE LENGTH OF THE SDAID BUFFER  *
*                      ARE TAKEN FROM A CONTROL BLOCK LOCATOR ENTRY    *
*                      (LBD ENTRY) FROM WITHIN THE STAND-ALONE DUMP.   *
*                                                                      *
*  OUTPUT:             IJBXSDA BUILDS TEXT EXTENSION RECORDS (LBXT)    *
*                      AND COLLECTS THEM IN AN INTERNAL BUFFER OF 2K.  *
*                      WHEN THE BUFFER IS FULL IT IS PASSED TO         *
*                      INFO/ANALYSIS VIA THE SYMPTOM RECORD UPDATE     *
*                      INTERFACE.                                      *
*                      THE NAME OF THE FIRST LBD IS 'SDAID001',        *
*                      THE COMPONENT-ID IS 'IJBXSDA'.                  *
*                      THE SEQUENCE FIELD IS USED TO BUILD UP TO       *
*                      16 TEXT BLOCKS WITH THE SAME LBD NAME.          *
*                      IF MORE THEN 16 LBD BLOCKS ARE FILLED OTHER     *
*                      SEQUENCES OF LBD'S ARE CREATED WITH THE NAMES   *
*                      'SDAID002', 'SDAID003', ETC.                    *
```

```
*
*  EXIT:               THE PROGRAM MOVES A COMPLETION MESSAGE TO THE  *
*                      INFO/ANALYSIS MESSAGE BUFFER AND RETURNS TO    *
*                      INFO/ANALYSIS WITH A RETURN CODE OF ZERO.      *
*                                                                     *
*  ERROR EXIT:         THE PROGRAM MOVES AN ERROR MESSAGE TO THE      *
*                      INFO/ANALYSIS MESSAGE BUFFER AND RETURNS TO    *
*                      INFO/ANALYSIS WITH A NON-ZERO RETURN CODE.     *
*                                                                     *
*  RETURN CODES:        0 IF SDAID BUFFER WAS FORMATTED SUCCESSFULLY, *
*                         OR THE STAND-ALONE DUMP DID NOT CONTAIN AN  *
*                         SDAID BUFFER, OR IJBXSDA WAS ALREADY        *
*                         EXECUTED BEFORE.                            *
*                       8 ERROR DURING FREEVIS (VIA INFO/ANALYSIS),   *
*                         THE SDAID BUFFER HAS BEEN FORMATTED         *
*                         SUCCESSFULLY BUT FREEVIS FAILED.            *
*                      12 THIS CODE IS RETURNED IN SEVERE ERROR CASES *
*                         WHICH PREVENT FROM FORMATTING THE SDAID     *
*                         BUFFER.                                     *
*                                                                     *
*  MESSAGES:           MESSAGES ARE PASSED TO INFO/ANALYSIS IN A      *
*                      MESSAGE BUFFER.                                *
*                                                                     *
*                      4G18I FORMAT OF SDAID BUFFER IS INCORRECT      *
*                      4G21I PHASE IJSDDEB NOT FOUND                  *
*                      4G21I PHASE IJSDPWB NOT FOUND                  *
*                      4G21I PHASE IJSDCVT NOT FOUND                  *
*                      4G22I SDAID BUFFER WAS FORMATTED SUCCESSFULLY  *
*                      4G70I DUMP TO BE PROCESSED WAS NOT PRODUCED    *
*                            BY VSE                                   *
*                      4G71I DUMP DOES NOT CONTAIN AN SDAID BUFFER    *
*                            (R.C.=0)                                 *
*                      4G72I NO DUMP DATA FOUND FOR SDAID BUFFER      *
*                      4G73I WRONG DUMP TYPE. IJBXSDA PROCESSES       *
*                            STAND-ALONE DUMPS ONLY.                  *
*                      4G74I GETVIS FOR SDAID BUFFER FAILED           *
*                      4G75I FREEVIS FOR SDAID BUFFER FAILED          *
*                      4G76I IJBXSDA CALL ERROR. REASON CODE: X       *
*                            X=1 DUMP DOES NOT CONTAIN A SYMPTOM      *
*                                RECORD                               *
*                            X=2 NO INFO/ANALYSIS DUMP ACCESS ROUTINE *
*                            X=3 NO INFO/ANALYSIS S.-R. UPDATE        *
*                                ROUTINE                              *
*                            X=4 NO INFO/ANALYSIS S.-R. ACCESS        *
*                                ROUTINE                              *
*                            X=5 NO INFO/ANALYSIS GETVIS/FREEVIS      *
*                                ROUTINE                              *
*                      4G77I UPDATING THE SYMPTOM RECORD FAILED       *
*                            RETURN CODE: XXX, REASON CODE: XXX       *
*                      4G78I TRACE DATA FOR THIS DUMP IS ALREADY      *
*                           STORED IN DUMP LIBRARY                   *
*                                                                     *
*  PROGRAM LOGIC: PART 1: CHECK INFO/ANALYSIS INTERFACE:              *
*                         IF AN ERROR IS DETECTED THEN PREPARE AN     *
*                         ERROR MESSAGE AND RETURN TO INFO/ANALYSIS:  *
*                         MESSAGES:                                   *
*                            DUMP NOT PRODUCED BY VSE       (4G70I)   *
*                            DUMP DOES NOT CONTAIN A                  *
*                               SYMPTOM RECORD              (4G76I-1) *
*                            DUMP ACCESS ROUTINE NOT                  *
*                               AVAILABLE                   (4G76I-2) *
*                            SYMPTOM UPDATE ROUTINE NOT               *
*                               AVAILABLE                   (4G76I-3) *
*                            SYMPTOM ACCESS ROUTINE NOT               *
*                               AVAILABLE                   (4G76I-4) *
*                                                                     *
*                 PART 2: RETRIEVE LBD ENTRY 'SDBUFFER'.              *
*                         CALCULATE LENGTH OF SDAID PHASES.           *
*                         ASK INFO/ANALYSIS FOR THE REQUIRED GETVIS   *
*                         SPACE.                                      *
*                         LENGTH OF GETVIS SPACE =                    *
*                            LENGTH OF SDAID BUFFER  +                *
*                            LENGTH OF PHASE IJSDDEB +                *
*                            LENGTH OF PHASE IJSDPWB +                *
*                            LENGTH OF PHASE IJSDCVT +                *
*                                                                     *
*                         IF LBD ENTRY FOR 'SDBUFFER' NOT             *
*                            FOUND                          (4G71I)   *
*                         OR THE SDAID PHASES CANNOT BE               *
*                            LOADED                         (4G21I)   *
*                         OR NO GETVIS/FREEVIS ROUTINE                *
*                            AVAILABLE                      (4G76I-5) *
*                         OR NOT ENOUGH GETVIS SPACE                  *
*                            AVAILABLE                      (4G74I)   *
*                         THEN PREPARE AN ERROR MESSAGE AND RETURN    *
*                         TO INFO/ANALYSIS.                           *
*                                                                     *
*                 PART 3: PREPARE THE SDAID BUFFER AND LOAD THE       *
*                         REQUIRED SDAID PHASES.                      *
*                         IF AN ERROR IS DETECTED THEN FREE THE       *
*                         GETVIS SPACE, PREPARE AN ERROR MESSAGE      *
*                         AND RETURN TO INFO/ANALYSIS.                *
*                                                                     *
*                         REQUEST DUMP DATA (CONTENTS OF SDAID        *
*                         BUFFER) FROM INFO/ANALYSIS AND MOVE IT TO   *
*                         GETVIS BUFFER.                              *
*                         IF 'NO DUMP DATA FOR SDBUFFER'              *
*                                                      THEN MSG 4G72I *
*                                                                    *
*                        LOAD SDAID PHASE IJSDDEB.                   *
*                        IF 'LOAD FAILURE FOR IJSDDEB'               *
*                                                     THEN MSG 4G21I *
*                        LOAD SDAID PHASE IJSDPWB.                   *
*                        IF 'LOAD FAILURE FOR IJSDPWB'               *
*                                                     THEN MSG 4G21I *
*                        LOAD SDAID PHASE IJSDCVT.                   *
*                        IF 'LOAD FAILURE FOR IJSDCVT'               *
*                                                     THEN MSG 4G21I *
*                                                                    *
*                        STORE LOAD ADDRESS OF IJSDPWB AND IJSDCVT   *
*                        INTO THE INTERFACE AREA OF IJSDDEB.         *
*                                                                    *
*                PART 4: PROCESS THE SDAID TRACE BUFFER.             *
*                                                                    *
*                        CALL IJSDDEB (FUNCTION=INIT)                *
*                        INITIALIZE THE LBD BUFFER.                  *
*                        BUILD LBD AND LBXT HEADERS.                 *
*                                                                    *
*                        CALL IJSDDEB (FUNCTION=PRINT) FOR TEST      *
*                        THE SDAID DEBLOCKING FUNCTIONS ARE CALLED   *
*                        TO COUNT THE NUMBER OF LIBRARY BLOCKS       *
*                        REQUIRED.                                   *
*                        IF BUFFER INVALID (R.C.>0)   THEN MSG 4G18I *
*                                                                    *
*                        CALL IJSDDEB (FUNCTION=PRINT) FOR           *
*                        PRODUCTION                                  *
*                        THE FORMATTED TRACE RECORDS ARE PASSED AS   *
*                        LBXT RECORDS TO INFO/ANALYSIS.              *
*                        IF BUFFER INVALID (R.C.>0)   THEN MSG 4G18I *
*                                                                    *
*                        PROCESS THE LAST LBD BUFFER.                *
*                        PASS THE REMAINING LBXT ENTRIES TO          *
*                        INFO/ANALYSIS.                              *
*                                                                    *
*                        FREE GETVIS SPACE.                          *
*                                                                    *
*                        IF BUFFER HAS BEEN PROCESSED SUCCESSFULLY,  *
*                        THEN SET RETURN CODE TO ZERO, AND RETURN    *
*                        MESSAGE 4G22I TO INFO/ANALYSIS.             *
*                                                                    *
*  REGISTER USAGE:    R0 - WORK REGISTER (ESPECIALLY FOR SVC'S)      *
*                     R1 - WORK REGISTER (ESPECIALLY FOR SVC'S)      *
*                     R2 - WORK REGISTER                             *
*                     R3 - WORK REGISTER                             *
*                     R4 - WORK REGISTER                             *
*                     R5 - WORK REGISTER                             *
*                     R6 - RPL POINTER                               *
*                      R7 - WORK REGISTER                             *
*                      R8 - WORK REGISTER                             *
*                      R9 - WORK REGISTER                             *
*                      RA - WORK REGISTER                             *
*                      RB - RESERVED FOR SECOND BASE REGISTER         *
*                      RC - BASE REGISTER                             *
*                      RD - SAVE AREA POINTER / POINTER TO DUMP CAUSE *
*                      RE - LINK REGISTER                             *
*                      RF - RETURN CODES & INFO/ANALYSIS ROUTINES     *
*                                                                     *
*  SUBROUTINES:        (1) IJBXSDA USES THE FOLLOWING INFO/ANALYSIS   *
*                          ROUTINES:                                  *
*                                                                     *
*                             DUMP DATA ACCESS ROUTINE                *
*                             SYMPTOM RECORD ACCESS ROUTINE           *
*                             SYMPTOM RECORD UPDATE ROUTINE           *
*                             GETVIS/FREEVIS SERVICE ROUTINE          *
*                                                                     *
*                      (2) WRITELBD.                                  *
*                          THIS ROUTINE IS USED BY THE SDAID MODULE   *
*                          IJSDDEB TO PASS LBXT ENTRIES TO            *
*                          INFO/ANALYSIS.                             *
*                                                                     *
*  MODULES CALLED:     IJSDDEB (TO FORMAT THE SDAID BUFFER).          *
*                                                                     *
*                      THE ADDRESS OF AN INTERFACE AREA (DUMPIF) IS   *
*                      PASSED TO IJSDDEB IN REGISTER 1.               *
*                                                                     *
*  MACROS USED:        LOAD       LOAD A PHASE INTO GETVIS STORAGE    *
*                      MAPDNTRY   STRUCTURE OF DIRECTORY ENTRY        *
*                      GENL       GENERATE A LOAD LIST                *
*                      IJBXSTC6   SYMPTOM RECORD INTERFACE            *
*                      ADSPXRPL   ANALYSIS ROUTINE INTERFACE          *
*                                                                     *
*  PATCH LABEL AREA:   PATCH                                          *
*                                                                     *
*---------------------------------------------------------------------*
```

INFO/ANALYSIS: INTRODUCTION

This chapter contains general information describing the Info/Analysis for VSE Program Product.

PROGRAM OVERVIEW

Info/Analysis is a component of VSE/Advanced Functions Version 2 Release 1. It provides interactive and batch facilities for dump analysis, dump management, and dump symptom extraction.

Info/Analysis consists of a group of interrelated display panels and functional routines for:

- Dump data set management
- Dump symptom display
- Dump data viewing
- Offload and onload of dump data

Info/Analysis provides facilities to support the Symptom Record Architecture. These facilities include:

- Display of the problem failure symptom record
- Display of dump data using pre-defined control block and data locators and formatting descriptors
- Invocation of system and component dump analysis routines

Info/Analysis is intended to be used with the Interactive System Productivity Facility (ISPF) Program Product for full screen display and interaction. However, facilities are provided for invocation from batch.

INFO/ANALYSIS FUNCTIONAL COMPONENTS

Figure 6 is an overview of the functional components of Info/Analysis.

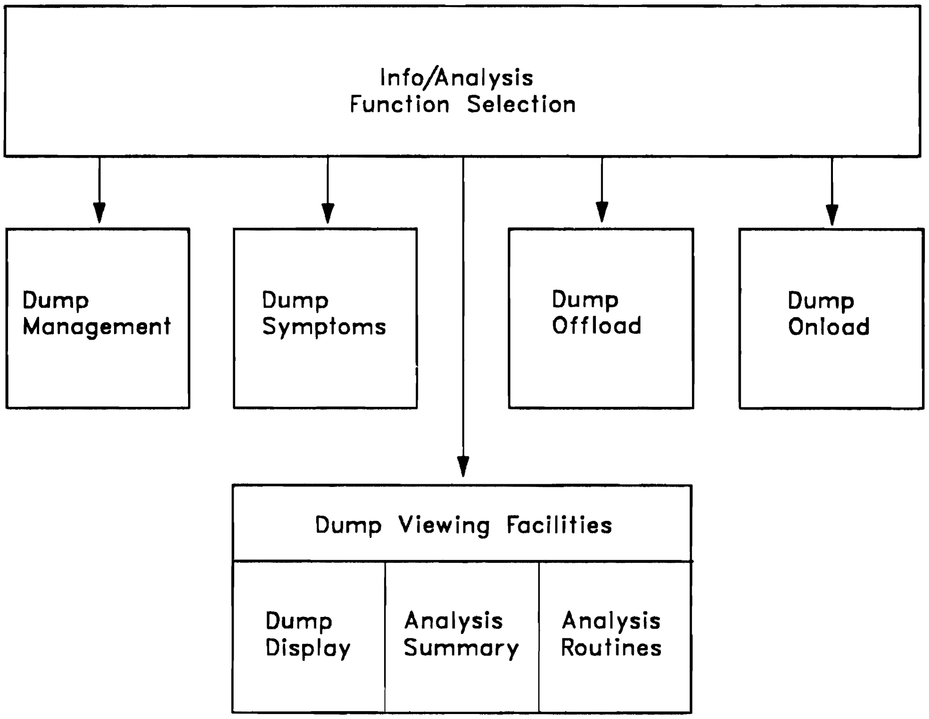


Figure 6. Info/Analysis Functional Components

With Info/Analysis, you can simplify the task of using dump data to solve software problems. Info/Analysis assists you in this task through the following functions:

- Dump data set management - to list the dumps being managed by Info/Analysis, to add or delete dumps from that list, and to delete dumps from the system.

- Dump symptom display - to display problem failure information collected by the dumping component and by subsequent analysis routines. This information may be used both locally and in an IBM maintenance data base for problem determination.

- Dump data viewing - to display dump data in hexadecimal format, to locate, format, and display control blocks and other dump data that may be pertinent to the problem, to mask out (overlay) sensitive data in specific areas of the dump, to invoke dump analysis routines, and to display the results of those routines.

- Offload of dump data - to copy a dump to tape for later retrieval or for forwarding to an IBM software support center.

- Onload of dump data - to copy an offloaded dump, a dump from another system, or a stand-alone dump to the system.

Info/Analysis processes storage dumps that result from errors within the system supervisor or within subsystem or user programs running on the system. The dumps are created by system dump and stand-alone dump utilities. Info/Analysis does not directly access the dump data. Rather, it uses system facilities to retrieve and update dump data and the symptom record. The symptom record is a collection of problem-related information stored in the dump and its extensions.

Info/Analysis uses a dump management file to maintain information about dumps. A dump must be identified in this file before it can be processed by Info/Analysis. This file is maintained using the dump management function.

Info/Analysis also uses an external routines file. This file contains a list of analysis routines that you may invoke to process dump data. The file identifies user exit routines and dump access routines called by Info/Analysis.

Info/Analysis may be used in two modes, interactive and batch, which are described below. All major functions are available in both modes; however, dump data display has more flexibility in the interactive mode, while it is preferable to use dump onload and offload in batch mode.

Interactive Mode

In interactive mode, Info/Analysis runs in a VSE/ICCF interactive partition as a dialog managed by the Interactive System Productivity Facility (ISPF) Program Product.

You communicate with Info/Analysis through a 3270-type display terminal. By responding to a sequence of panels, you may select functions and options, enter commands, and display data. Figure 7 illustrates the interactive environment of Info/Analysis.

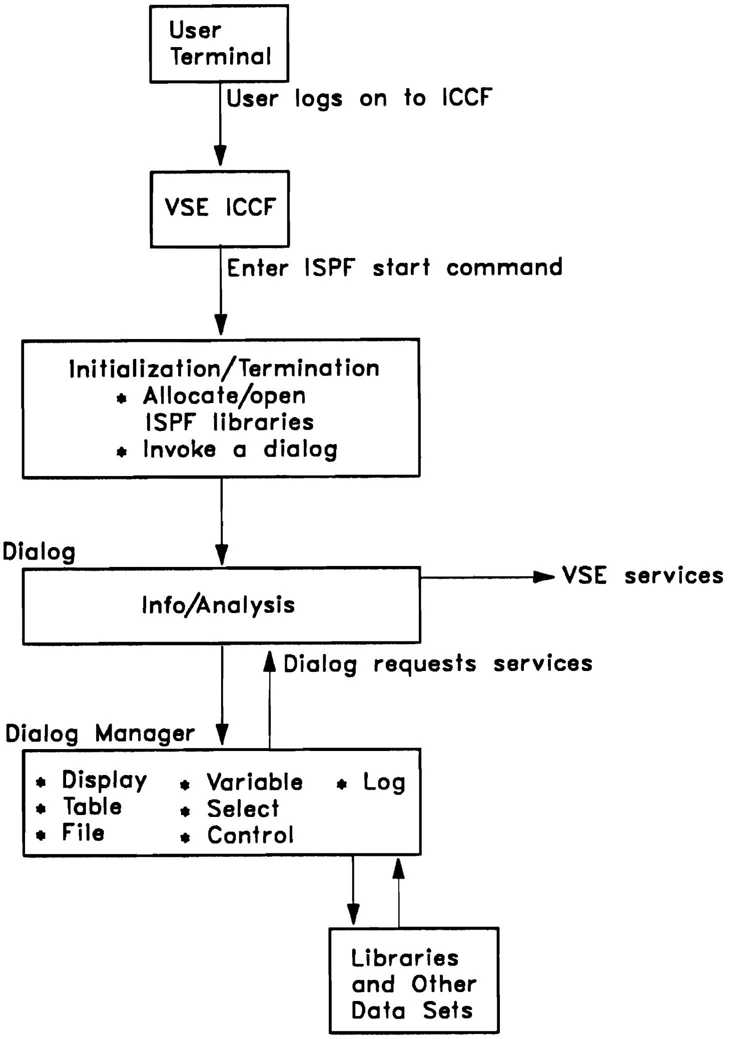


Figure 7. Info/Analysis Interactive Environment

The user first logs on to the Interactive Communication and Control Facility (ICCF), allocates the data sets used by Info/Analysis, and invokes an ICCF procedure, ISPSTART. The ISPSTART procedure invokes ISPF as a problem program in an ICCF pseudo partition by issuing the PGM keyword.

The PGM keyword causes the Info/Analysis environment to be initialized by setting up the necessary control blocks, assuring that certain files are defined, and verifies that the required VSE libraries are available. The last step of Info/Analysis initialization is to display the function selection panel, where all of the Info/Analysis functions are available for selection.

Info/Analysis may call any of the services (control, display, table services, and so on) provided by ISPF. In addition to the services available in ISPF, the Info/Analysis dialog may call any of the VSE services that are supported in an ICCF pseudo partition.

The VSE libraries used by these services must have been specified by ISPDEF statements in the ISPSTART procedure used to invoke Info/Analysis. These libraries must contain the necessary panels, messages, tables, and skeletons needed by the dialog during its execution.

When the user terminates Info/Analysis, all allocated system resources are released and control is returned to ICCF.

Batch Mode

In batch mode, Info/Analysis runs in a VSE/Advanced Functions partition or in a VSE/ICCF interactive partition. You may enter control statements in two modes:

- Line mode - from the operator console or ICCF console

- Reader mode - from the system input device or the ICCF virtual reader defined as the input device.

From a VSE/Advanced Functions partition, all output from batch operations is routed to the SYSLST device assigned to the partition. In line mode, messages are sent to the console as well as to SYSLST.

From a VSE/ICCF interactive partition, all output from batch operations is routed to the ICCF virtual printer called the print area. In line mode, messages are sent to the console as well as to the print area.

In the batch environment certain services, such as display, are not available.

PHYSICAL CHARACTERISTICS

Info/Analysis is comprised of reentrant phases only. For performance reasons, it is recommended that the phases be loaded in the shared virtual area (SVA).

INFO/ANALYSIS: TECHNICAL OVERVIEW

This chapter provides an overview of the structure of Info/Analysis as well as information on resources and security.

INFO/ANALYSIS STRUCTURE

Info/Analysis interacts with several external programs and components that perform specialized services. These programs and components are:

- ISPF
- VSE Dump Access
- BLX

Figure 8 illustrates the structure of Info/Analysis.

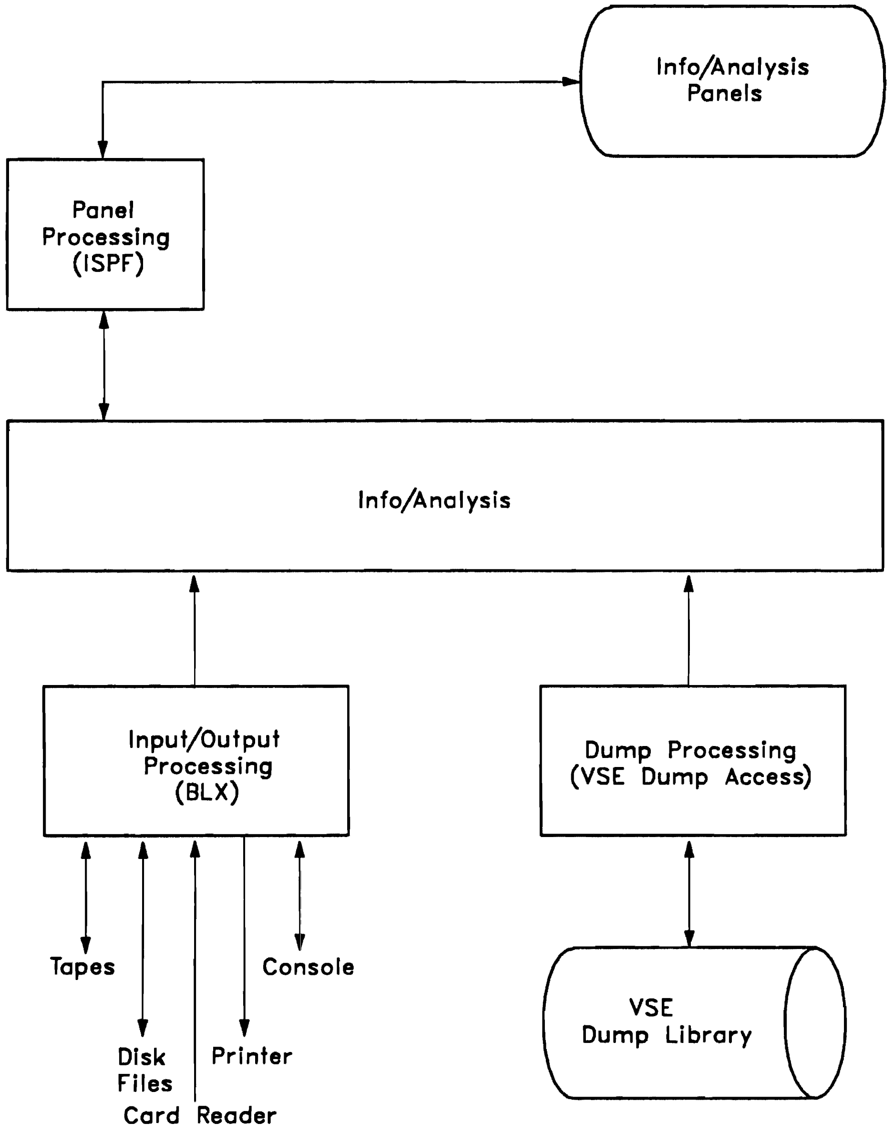


Figure 8. Info/Analysis Structure

The different external components of Info/Analysis have been isolated to facilitate future requirements and problem determination. Each of the components is implemented as a separate load module.

Each component has its own method of problem determination; therefore, when a problem does occur, it is important to identify which of these components caused the problem.

All accesses to the VSE dump library are performed by a dump access service written specifically for VSE. Control is transferred to dump access when a request is made to read, write, or update dump data.

Panel processing is handled by ISPF. Info/Analysis issues calls to ISPF routines to display panels, read panel input, process tutorial requests, write to the ISPF log and list files, and interpret the cursor position. ISPF utilizes the VSE DTFCN facilities to display line I/O terminal output. Control is transferred to ISPF when a display is requested or the dialog is completed.

All I/O operations are isolated in BLX services. These services provide system independent macros for all I/O operations. Control is transferred to BLX when I/O operations are requested.

INFO/ANALYSIS LIBRARIES AND FILES

In VSE, source statement libraries are used to store panels, messages, tables, and skeletons. The ISPDEF statement establishes the relationship between the Info/Analysis library name and the actual VSE source statement library(s).

All VSE and ISPF libraries and files must be defined to the VSE operating system by the user before Info/Analysis is initialized. Info/Analysis requires auxiliary storage space for its own programs and display panels. The current estimate is 190 tracks of 3330 disk storage for Info/Analysis, Dump Access and BLX Services.

The customer determines the allocation and size of the library for dump data storage. This determination depends on the number of dumps that the user site is prepared to maintain at any one time. Space for three dumps of 2M bytes each may be a valid initial specification, subject to change based on experience.

Info/Analysis requires a data set allocation to contain the list of dumps currently being managed. This data set is estimated to be two tracks or less of 3330 storage, depending on the number of dumps being managed. This data set contains control information for all dumps known to Info/Analysis whether they are currently stored in the VSE dump library or not. This data set is maintained by Info/Analysis and may not be modified by the user.

Info/Analysis requires a data set allocation to contain the list of post dump analysis exit routines available. This data set is estimated to be two tracks or less of 3330 storage. The precise size depends on the number of dump analysis routines available. This data set contains the name and an optional description of each routine available for use with Info/Analysis. The data set may be modified by the user with system utilities or with ISPF editor facilities. The format of each record in the data set is as follows:

- The first nonblank characters in each record of the data set are the character string 'ANEXIT'. This string indicates that this record contains the name and description of an analysis exit routine. Any other starting string of characters is ignored.

- The name of the routine follows the 'ANEXIT' string. There must be at least one blank between 'ANEXIT' and the routine name. The routine name may be one to eight characters in length, followed by at least one blank.

- An optional description may be included following the routine name and the blank character delimiter. This description is free-form text and is placed alongside the name of the routine in the displayed list. Info/Analysis separates the description from the routine name with six blank characters in the displayed list. Continuation cards are not displayed or printed.

Examples of this record are:

```
ANEXIT   IVBXSVA    ANALYZE VSE SUPERVISOR
ANEXIT IVBXSVAF FORMAT VSE SUPERVISOR CONTROL BLOCKS
```

Note that there may be more than one space between 'ANEXIT' and the routine name and between the routine name and its description. Info/Analysis uses the information in this record for display and selection of an analysis routine.

Info/Analysis requires tape volumes to onload and offload dump information.

LIBRARY AND FILE PROTECTION

Info/Analysis makes use of the VSE enqueue facilities as provided by BLX services. This use covers access to dump data sets and other auxiliary data sets maintained on external storage. Conflicts in VSE library access by Info/Analysis and the VSE librarian or by multiple Info/Analysis users are controlled through this facility.

Info/Analysis serializes read and write activities to the same file with a BLXENQ request for exclusive use using the logical device name (LNAME).

This BLXENQ is unconditional (UNCOND) with an option of WAIT. UNCOND specifies that if a previous BLXENQ has been issued for the same resource, without an intervening BLXDEQ, the task is to abend. WAIT specifies that if a resource is not immediately available, the user is to be put into a wait state until it can be enqueued. This may result in a temporary wait if the library is being written into by another Info/Analysis user. This wait is usually of short duration. If successful, this enqueue remains in effect for the entire time Info/Analysis has control of the library.

SECURITY

Info/Analysis provides minimal facilities for security of customer information. The primary facility is the ability to overlay sections of the dump that may contain customer sensitive data. Refer to the MASK command in the VSE/AF Serviceability Aids and Debugging Procedures.

SYSTEM INTEGRITY

Info/Analysis executes entirely as a nonprivileged problem program in the host environment. Therefore, no system integrity problems are anticipated.

INFO/ANALYSIS HOST SYSTEM CONTROL BLOCKS

Info/Analysis does not refer to or utilize any host system control blocks. Communication with system facilities is done through standard system interfaces such as system macros or system-provided functions (dump access).

INFO/ANALYSIS STORAGE REQUIREMENTS

Info/Analysis is expected to require approximately 400K bytes, including BLX system services. Storage requirements for other products such as ICCF and ISPF used with Info/Analysis may cause this figure to vary. Under most conditions, a partition size of 896K bytes is considered adequate; a GETVIS area of 200K is recommended.

INFO/ANALYSIS EQUIPMENT SUPPORTED

Info/Analysis operates on any configuration and CPU that meets the minimum specifications for VSE/AF Version 2 Release 1. Equipment support is also provided for those devices and systems that are supported by ISPF.

INFO/ANALYSIS: DIAGNOSTIC AIDS

This chapter contains the following information, which can help in diagnosing problems within Info/Analysis and Dump Access:

- Register usage
- Abend information
- I/O diagnostic aids
- Terminal I/O error codes
- Messages and codes
- Module/message cross-reference for Info/Analysis
- Message/module cross-reference for Info/Analysis
- Module/reason cross-reference for Info/Analysis
- Reason/module cross-reference for Info/Analysis
- Calling/called module cross-reference for Info/Analysis
- Called/calling module cross-reference for Info/Analysis
- Module/reason cross-reference for Dump Access
- Reason/module cross-reference for Dump Access
- Calling/called module cross-reference for Dump Access
- Called/calling module cross-reference for Dump Access

REGISTER USAGE

When an Info/Analysis or Dump Access program passes control to a separately compiled Info/Analysis, Dump Access, BLX, or ISPF program, the following linkage conventions are generally observed:

Register	Contents
1	Pointer to parameter list; each address in the list, in turn, points to a parameter.
2 - 12	Work registers.
13	Address of an 80-byte save area that includes the normal 72-byte save area and eight bytes of BLX control information.
14	Return address.
15	When a program is entered, contains its entry address; upon return to the invoking program, contains its return code.

Within Info/Analysis programs, the following register conventions are used:

Register	Contents
1	Involved in program linkage.
2 - 11	Used differently by each program.
12	Address of the storage containing the program and its literals (base register).
13	Address of an 80-byte save area which includes the normal 72-byte save area and eight bytes of BLX control information followed by the local dynamic storage associated with the module.
14 - 15	Involved in program linkages.

ABEND INFORMATION

When an abend occurs in Info/Analysis, information about the abend is normally provided in an internal dump (IDUMP). In batch mode, Info/Analysis always takes an IDUMP. When a dump is taken a symptom string is built and placed at the beginning of the dump. The symptom string can be displayed or printed using Info/Analysis.

The symptom string is a line of text that contains specific items in the condensed format described below. It may help an installation or an IBM representative in tracking problems because it can be easily compared to the symptom strings of known problems. The string will contain at least the following required symptoms:

- AB/abend code prefix, abend code

  The prefix is "S" for a system abend code or "U" for a user code.

- PIDS/component identifier

  If the abend occurred in Info/Analysis, or if the location of the abend cannot be identified, the Info/Analysis component id is shown.

- RIDS/routine identifier

  The Info/Analysis module where the abend occurred.

- REGS/register number, PSW-register difference

  This item shows the result of comparing the value of each general register with the address in the PSW at the time of the abend. The results are shown for the two highest numbered registers whose values are less than the PSW, but only by an amount less than 4096 bytes. The hexadecimal register number occupies the first two positions, and the hexadecimal difference occupies the last three positions.

  If no registers satisfy this condition, this item appears as "REGS/FFFFF". If the PSW address is less than 512, this item appears as "REGS/FExxx" where xxx is the hexadecimal representation of the PSW address.

Audit Considerations

The log file can be used to record all activity that took place during a session, and thus an audit trail of activity that took place prior to an abend. The log file will not be created unless specifically requested by the LOG command. The operational items logged are:

- Panel identifiers
- Output messages
- Command entries
- Data entry fields
- All selections

The ISPF LOG service is used for control of the output in the interactive operation. In batch mode, all control statements, messages, and output are routed to the SYSLST device.

The IDUMP

An IDUMP consists of the entire ICCF pseudo partition in which Info/Analysis is running. When Info/Analysis is running in batch mode, the entire batch partition is dumped. In either case, if the address of the abend is within the SVA, a phase of the SVA is also dumped.

The following information is provided by an IDUMP:

- IDUMP comment

  The comment at the top of each page identifies Info/Analysis and the name of the Info/Analysis module in which the abend occurred. The comment also contains the address of the IDUMP information area within the dump, which is described below.

- Registers

  The contents of the general registers at the time of the abend are listed at the start of the IDUMP.

- IDUMP information area

  The IDUMP information area contains a copy of the symptom string described above, followed by a 24-byte "ABEND SAVEAREA" label and an 80-byte field in partition save area format. This 80-byte field consists of the name of the abended module or dialog (8 bytes), the abend-time PSW (8 bytes), and registers 9 through 8 at abend time (64 bytes).

Decimal ("User") Abend Codes

Info/Analysis does not issue any abend codes. BLX abend codes are contained in the VSE/SP Messages and Codes Manual. Abend codes issued by other programs or components can be found in the appropriate program or component documentation.

I/O DIAGNOSTIC AIDS

All Info/Analysis library and sequential file access has been isolated within the external components. BLX codes are contained in the VSE/SP Messages and Codes Manual. Any codes issued by other programs or components can be found in the appropriate program or component documentation.

TERMINAL INPUT/OUTPUT ERROR CODES

ISPF is responsible for all terminal I/O done by Info/Analysis. Refer to the ISPF Diagnosis for any error codes.

MESSAGES AND CODES

Messages serve to communicate status information to the user. These messages pertain to error conditions and requests for verification and confirmation that may occur while using the operations and functions of Info/Analysis.

In ISPF, the messages must exist in an ISPF message data set, whereas the batch environment must be able to run without depending on the presence of ISPF. BLX and Info/Analysis message services are both utilized to accomplish this task.

BLX message services are used to construct the skeleton forms of the message. BLX message macros are used to route the output.

In the batch environment, the BLX message macro is used to route the messages to the printer or output device. In the ISPF environment, the constructed message is placed in an ISPF variable and displayed in the panel.

Each Info/Analysis message is identified by a string of eight characters called the message id. This message id has the format BLNnnnna, where:

- BLN is the Info/Analysis component identifier
- nnnn is the assigned decimal message number
- a is the action code

The first two digits of the message number identify the functional area to which the message applies, as follows:

- 09 Symptom record access
- 10 Function selection
- 20 Dump management
- 30 Dump onload
- 40 Dump offload
- 50 Dump viewing
- 60 Dump symptoms
- 70 Analysis summary and routines
- 80 Scrolling
- 90 General usage

The following action codes are used in Info/Analysis:

- D = Decision — You must choose between courses of action.
- I = Information — This message provides error or status information; you should correct the error if one is indicated.

When an action or decision is necessary, Info/Analysis usually waits until you enter an acceptable reply from the keyboard, or perform an action such as mounting a disk pack, readying a device, or placing cards in the card reader.

Variable data in the message text is indicated in lowercase. The variable that appears in the message documentation is replaced in an issued message as follows:

• addr	- address
• block	- control block name
• disp	- DISPLAY value
• dumpid	- dump name
• entry	- a command, control statement, or data entry field value
• fil	- file name
• func	- function
• jobname	- job name
• maxrec	- maximum number of records
• mod	- module
• msg	- message
• ofst	- offset
• oper	- operand
• pnl	- panel name
• rcd	- record number
• req	- request

- rsc - reason code
- rtc - return code
- rtn - external routine
- serv - service
- tbl - table
- userec - number of records currently in use

Reason codes may be used within a message to further describe the error condition that caused the message. These reason codes assist the user in identifying and correcting error situations.

More severe error conditions that the user cannot correct by altering input may cause an abend to occur. In these cases, a reason code is entered in internal control blocks before the dump is taken. These reason codes are then used during dump analysis to identify the problem.

The reason codes that may be included in Info/Analysis messages are associated with particular functions, as follows:

- 0 - 899 Dump access interface
- 900 - 999 Symptom record access
- 1000 - 1999 Function selection and batch control statements
- 2000 - 2999 Dump management
- 3000 - 3999 Dump onload
- 4000 - 4999 Dump offload
- 5000 - 5999 Dump viewing
- 6000 - 6999 Dump symptoms
- 7000 - 7499 Analysis summary
- 7500 - 7999 Analysis routines
- 8000 - 8999 Screen function
- 9000 - 9999 General

INFO/ANALYSIS MODULE/MESSAGE CROSS-REFERENCE

In the following list, object modules that use messages are listed alphabetically by module name. Each module name is followed by the message id of each message that the module issues, the possible environments for each message, the type of message (error or informational), and the variables that are inserted in the message. The possible environment modes are:

- I - Interactive
- B - Batch
- C - Console

Module Issuing	Message ID	Modes	Type	Inserts
BLNALADF	BLN0929I	I/B/C	INFO	rcd, ofst
BLNARDLO	BLN9001I	I/B/C	ERROR	
BLNARDSP	BLN7540I	I/B/C	ERROR	
BLNAREXA	BLN9001I	I/B/C	ERROR	
BLNAREXA	BLN9007I	I/B/C	ERROR	rsc
BLNAREXC	BLN7540I	I/B/C	ERROR	
BLNAREXD	BLN7515I	B/C	ERROR	rtn
BLNAREXD	BLN7520I	B/C	ERROR	
BLNAREXD	BLN7540I	I/B/C	ERROR	
BLNAREXG	BLN9002I	I/B/C	ERROR	rtc, rsc
BLNAREXL	BLN7516I	I/B/C	ERROR	rtn, rsc
BLNARFND	BLN9014I	I	INFO	entry
BLNARFND	BLN9023I	I	ERROR	
BLNARGRN	BLN9001I	I/B/C	ERROR	
BLNARINT	BLN9007I	I/B/C	ERROR	rsc
BLNARPD6	BLN9001I	I/B/C	ERROR	
BLNARPRN	BLN9001I	I/B/C	ERROR	
BLNARPRT	BLN5011I	I/B/C	ERROR	
BLNARPRT	BLN7540I	I/B/C	ERROR	
BLNARPRT	BLN9012I	I/B/C	INFO	
BLNARPRT	BLN9022I	I/B/C	ERROR	
BLNARS6U	BLN9001I	I/B/C	ERROR	
BLNARXPL	BLN9001I	I/B/C	ERROR	
BLNASADF	BLN0923I	I/B/C	INFO	
BLNASADF	BLN0930I	I/B/C	INFO	
BLNASADL	BLN0923I	I/B/C	INFO	
BLNASADL	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASADL	BLN0930I	I/B/C	INFO	
BLNASAXA	BLN0923I	I/B/C	INFO	

Module Issuing	Message ID	Modes	Type	Inserts
BLNASAXA	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASAXA	BLN0930I	I/B/C	INFO	
BLNASAXA	BLN0931I	I/B/C	INFO	
BLNASAXC	BLN0923I	I/B/C	INFO	
BLNASAXC	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASAXC	BLN0930I	I/B/C	INFO	
BLNASAXC	BLN0931I	I/B/C	INFO	
BLNASAXK	BLN0923I	I/B/C	INFO	
BLNASAXK	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASAXK	BLN0930I	I/B/C	INFO	
BLNASAXK	BLN0931I	I/B/C	INFO	
BLNASAXT	BLN0923I	I/B/C	INFO	
BLNASAXT	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASAXT	BLN0930I	I/B/C	INFO	
BLNASAXT	BLN0931I	I/B/C	INFO	
BLNASAXX	BLN0923I	I/B/C	INFO	
BLNASAXX	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASAXX	BLN0930I	I/B/C	INFO	
BLNASAXX	BLN0933I	I/B/C	INFO	
BLNASFMT	BLN0921I	I/B/C	ERROR	
BLNASGET	BLN0921I	I/B/C	ERROR	
BLNASLCB	BLN0920I	I/B/C	ERROR	
BLNASLCB	BLN0921I	I/B/C	ERROR	
BLNASLD1	BLN0921I	I/B/C	ERROR	
BLNASLD1	BLN0926I	I/B/C	ERROR	
BLNASLOC	BLN0926I	I/B/C	ERROR	
BLNASLST	BLN0921I	I/B/C	ERROR	
BLNASMAP	BLN0921I	I/B/C	ERROR	
BLNASMAP	BLN0927I	I/B/C	ERROR	
BLNASMD	BLN0923I	I/B/C	INFO	
BLNASMD	BLN0929I	I/B/C	INFO	rcd, ofst
BLNASMD	BLN0932I	I/B/C	INFO	
BLNASMD	BLN0934I	I/B/C	INFO	
BLNASMP1	BLN0921I	I/B/C	ERROR	
BLNASRSR	BLN0921I	I/B/C	ERROR	
BLNASRSR	BLN0927I	I/B/C	ERROR	
BLNASRSR	BLN0928I	I/B/C	ERROR	rsc
BLNASTRM	BLN0920I	I/B/C	ERROR	
BLNAXPD2	BLN9012I	I/B/C	INFO	
BLNAXPD2	BLN9022I	I/B/C	ERROR	
BLNAXPD3	BLN7521I	I/B/C	INFO	
BLNAXPD3	BLN7522I	I/B/C	INFO	
BLNAXPD4	BLN7522I	I/B/C	INFO	
BLNAXPD6	BLN9022I	I/B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNAXSU1	BLN7518I	I/B/C	ERROR	
BLNBABH1	BLN1007I	I/B/C	ERROR	
BLNBABH1	BLN9004I	I/B/C	ERROR	entry
BLNBABH1	BLN9019I	I/B/C	ERROR	dumpid
BLNBABH1	BLN9032I	I/B/C	ERROR	dumpid
BLNBABH2	BLN1004I	B	INFO	
BLNBADM1	BLN1007I	I/B/C	ERROR	
BLNBADM1	BLN9004I	I/B/C	ERROR	entry
BLNBADM1	BLN9016I	B/C	ERROR	
BLNBADM2	BLN1003I	B/C	ERROR	
BLNBADM2	BLN9004I	I/B/C	ERROR	entry
BLNBADS1	BLN9004I	I/B/C	ERROR	entry
BLNBADS1	BLN9016I	B/C	ERROR	
BLNBADV1	BLN9004I	I/B/C	ERROR	entry
BLNBAFLS	BLN1004I	B	INFO	
BLNBAINT	BLN1008I	B/C	INFO	
BLNBAINT	BLN9001I	I/B/C	ERROR	
BLNBAPF1	BLN4006I	I/B/C	ERROR	
BLNBAPF1	BLN9004I	I/B/C	ERROR	entry
BLNBAPF1	BLN9016I	B/C	ERROR	
BLNBAPN1	BLN9004I	I/B/C	ERROR	entry
BLNBAPN1	BLN9016I	B/C	ERROR	
BLNBARDR	BLN1004I	B	INFO	
BLNBARDR	BLN1005D	C	INFO	func
BLNBARDR	BLN1006I	B	ERROR	
BLNBARDR	BLN1009I	B/C	INFO	
BLNBARDR	BLN9008I	I/B/C	ERROR	mod, rsc
BLNCMCHD	BLN9006I	I/B/C	INFO	
BLNCMCHD	BLN9019I	I/B/C	ERROR	dumpid
BLNCMERM	BLN9002I	I/B/C	ERROR	rtc, rsc
BLNCMERM	BLN9003I	I/B/C	ERROR	msg
BLNCMERM	BLN9031I	I/B/C	ERROR	rtc, rsc
BLNCMFIL	BLN9004I	I/B/C	ERROR	entry
BLNCMHEX	BLN9004I	I/B/C	ERROR	entry
BLNCMVOL	BLN4001I	I/B/C	ERROR	
BLNCMVOL	BLN9001I	I/B/C	ERROR	
BLNCMVOL	BLN9004I	I/B/C	ERROR	entry
BLNCMXCF	BLN9004I	I/B/C	ERROR	entry
BLNCMXDE	BLN9030I	I/B/C	ERROR	req, fil, rsc
BLNDDBAS	BLN5005I	I	ERROR	addr
BLNDDCDB	BLN5005I	I	ERROR	addr
BLNDDCDB	BLN5006I	I	ERROR	entry
BLNDDCDB	BLN5007I	I	ERROR	entry
BLNDDCDB	BLN5030I	I/B/C	INFO	

Module Issuing	Message ID	Modes	Type	Inserts
BLNDDCDB	BLN9001I	I/B/C	ERROR	
BLNDDDPT	BLN9012I	I/B/C	INFO	
BLNDDFDO	BLN5013I	I/B/C	INFO	
BLNDDFDZ	BLN5030I	I/B/C	INFO	
BLNDDFFN	BLN9014I	I	INFO	entry
BLNDDFMT	BLN5009I	I	ERROR	
BLNDDFMT	BLN9001I	I/B/C	ERROR	
BLNDDFPP	BLN5030I	I/B/C	INFO	
BLNDDFPP	BLN9001I	I/B/C	ERROR	
BLNDDFPP	BLN9007I	I/B/C	ERROR	rsc
BLNDDFPP	BLN9022I	I/B/C	ERROR	
BLNDDFPT	BLN9012I	I/B/C	INFO	
BLNDDHDO	BLN5014I	I/B/C	INFO	block
BLNDDHFN	BLN9014I	I	INFO	entry
BLNDDINT	BLN9001I	I/B/C	ERROR	
BLNDDMDT	BLN5014I	I/B/C	INFO	block
BLNDDMDT	BLN9010I	I/B/C	INFO	
BLNDDNBU	BLN9001I	I/B/C	ERROR	
BLNDDPOP	BLN5011I	I/B/C	ERROR	
BLNDDPOP	BLN5022I	I/B/C	ERROR	
BLNDDPOP	BLN9004I	I/B/C	ERROR	entry
BLNDDPRT	BLN9012I	I/B/C	INFO	
BLNDDPRT	BLN9022I	I/B/C	ERROR	
BLNDDQAL	BLN5006I	I	ERROR	entry
BLNDDRDF	BLN9020I	I	INFO	
BLNDDRDF	BLN9021I	I/B/C	INFO	
BLNDDRPT	BLN5014I	I/B/C	INFO	block
BLNDDRPT	BLN9010I	I/B/C	INFO	
BLNDDRPT	BLN9012I	I/B/C	INFO	
BLNDDRPT	BLN9022I	I/B/C	ERROR	
BLNDDSYI	BLN9001I	I/B/C	ERROR	
BLNDDVFA	BLN5022I	I/B/C	ERROR	
BLNDDVFA	BLN9004I	I/B/C	ERROR	entry
BLNDDVFA	BLN9023I	I	ERROR	
BLNDDVFN	BLN9004I	I/B/C	ERROR	entry
BLNDDVFN	BLN9023I	I	ERROR	
BLNDMADU	BLN2006I	I/B/C	ERROR	
BLNDMADU	BLN9018I	I/B/C	INFO	dumpid, func
BLNDMAD1	BLN2019I	I/B/C	INFO	
BLNDMAD1	BLN9006I	I/B/C	INFO	
BLNDMCNT	BLN2007I	I/B/C	INFO	maxrec, userec
BLNDMDDL	BLN2017I	I/B/C	ERROR	
BLNDMDDL	BLN9001I	I/B/C	ERROR	
BLNDMDEL	BLN2009I	I/B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNDMDEL	BLN2017I	I/B/C	ERROR	
BLNDMDEL	BLN9018I	I/B/C	INFO	dumpid, func
BLNDMDMU	BLN2013I	I/B/C	ERROR	rsc
BLNDMDMU	BLN2014I	B/C	ERROR	
BLNDMDMU	BLN2020I	B/C	INFO	
BLNDMGNT	BLN9008I	I/B/C	ERROR	mod, rsc
BLNDMPDL	BLN2017I	I/B/C	ERROR	
BLNDMPDL	BLN9001I	I/B/C	ERROR	
BLNDMPDL	BLN9012I	I/B/C	INFO	
BLNDMRDF	BLN2013I	I/B/C	ERROR	
BLNDMUAD	BLN9004I	I/B/C	ERROR	entry
BLNDMWDF	BLN2013I	I/B/C	ERROR	
BLNDSCEN	BLN9006I	I/B/C	INFO	
BLNDSDSP	BLN9007I	I/B/C	ERROR	rsc
BLNDSGBF	BLN9001I	I/B/C	ERROR	
BLNDSPRT	BLN9007I	I/B/C	ERROR	rsc
BLNDSPRT	BLN9012I	I/B/C	INFO	
BLNDSTRM	BLN9007I	I/B/C	ERROR	rsc
BLNDSYMP	BLN9007I	I/B/C	ERROR	rsc
BLNDVKCB	BLN9001I	I/B/C	ERROR	
BLNDVSIA	BLN9001I	I/B/C	ERROR	
BLNDVSIN	BLN9007I	I/B/C	ERROR	rsc
BLNFNARC	BLN9007I	I/B/C	ERROR	rsc
BLNFNDDC	BLN9007I	I/B/C	ERROR	rsc
BLNFNDDO	BLN5002I	I	INFO	
BLNFNDDO	BLN9013I	I	ERROR	entry
BLNFNDMC	BLN9007I	I/B/C	ERROR	rsc
BLNFNDSC	BLN9007I	I/B/C	ERROR	rsc
BLNFNDVS	BLN9007I	I/B/C	ERROR	rsc
BLNFNMSK	BLN9007I	I/B/C	ERROR	rsc
BLNFNPFC	BLN9007I	I/B/C	ERROR	rsc
BLNFNPNC	BLN9007I	I/B/C	ERROR	rsc
BLNFNSMC	BLN9007I	I/B/C	ERROR	rsc
BLNLBARC	BLN9001I	I/B/C	ERROR	
BLNLBARR	BLN9001I	I/B/C	ERROR	
BLNLBCAC	BLN9001I	I/B/C	ERROR	
BLNLBCAE	BLN9001I	I/B/C	ERROR	
BLNLBCAI	BLN9001I	I/B/C	ERROR	
BLNLBCAL	BLN9001I	I/B/C	ERROR	
BLNLBCHC	BLN9001I	I/B/C	ERROR	
BLNLBCHE	BLN9001I	I/B/C	ERROR	
BLNLBCHI	BLN9001I	I/B/C	ERROR	
BLNLBCHL	BLN9001I	I/B/C	ERROR	
BLNLBCHR	BLN9001I	I/B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNLBCHW	BLN9001I	I/B/C	ERROR	
BLNLBLAL	BLN9001I	I/B/C	ERROR	
BLNLBSIM	BLN9001I	I/B/C	ERROR	
BLNLBVER	BLN7013I	I/B/C	ERROR	block, rsc
BLNMACBC	BLN1013I	I/B/C	ERROR	
BLNMACBG	BLN9002I	I/B/C	ERROR	rtc, rsc
BLNMKBMD	BLN5017I	I	INFO	
BLNMKBMD	BLN5018I	I	INFO	
BLNMKBMD	BLN9001I	I/B/C	ERROR	
BLNMKDSP	BLN9007I	I/B/C	ERROR	rsc
BLNMKINT	BLN9007I	I/B/C	ERROR	rsc
BLNMKMSK	BLN5008I	I	INFO	
BLNMKMSK	BLN5016I	I	ERROR	
BLNMKMSK	BLN9007I	I/B/C	ERROR	rsc
BLNMKQRY	BLN5030I	I/B/C	INFO	
BLNMKVMR	BLN5012I	I	ERROR	
BLNMKVMR	BLN5015I	I	ERROR	
BLNMKVMR	BLN5022I	I/B/C	ERROR	
BLNMKVMR	BLN9004I	I/B/C	ERROR	entry
BLNOFFLD	BLN4003I	I/B/C	ERROR	
BLNOFFLD	BLN4004I	I/B/C	INFO	dumpid, ERASED
BLNOFFLD	BLN4005I	I/B/C	ERROR	rsc
BLNOFFLD	BLN4007I	I/B/C	ERROR	
BLNOFFLD	BLN4010I	I/B/C	INFO	
BLNOFFLD	BLN9019I	I/B/C	ERROR	dumpid
BLNOFFLD	BLN9040I	I	INFO	jobname
BLNONEND	BLN3002I	I/B/C	ERROR	rsc
BLNONEND	BLN9018I	I/B/C	INFO	dumpid, func
BLNONINT	BLN3002I	I/B/C	INFO	rsc
BLNONINT	BLN4007I	I/B/C	ERROR	
BLNONINT	BLN9040I	I	INFO	jobname
BLNSCBKW	BLN8004I	I	ERROR	
BLNSCCRP	BLN8002I	I	ERROR	func, tbl, rtc
BLNSCDIS	BLN8004I	I	ERROR	
BLNSCDTB	BLN9001I	I/B/C	ERROR	
BLNSCFWD	BLN8004I	I	ERROR	
BLNSCLFT	BLN8004I	I	ERROR	
BLNSCPRD	BLN8001I	I	ERROR	pnl
BLNSCPRD	BLN8003I	I	ERROR	serv, rtc
BLNSCPRD	BLN8006I	I	ERROR	pnl, rtc
BLNSCPRD	BLN9008I	I/B/C	ERROR	mod, rsc
BLNSCRDT	BLN9001I	I/B/C	ERROR	
BLNSCRGH	BLN8004I	I	ERROR	
BLNSCSUB	BLN8002I	I	ERROR	func, tbl, rtc

Module Issuing	Message ID	Modes	Type	Inserts
BLNSFAMS	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFASM	BLN9004I	I/B/C	ERROR	entry
BLNSFAS1	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFAS1	BLN9017I	I	ERROR	
BLNSFBST	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFBUM	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFCAR	BLN9004I	I/B/C	ERROR	entry
BLNSFCA1	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFCA1	BLN9017I	I	ERROR	
BLNSFCMQ	BLN9011I	I	INFO	
BLNSFDAT	BLN9004I	I/B/C	ERROR	entry
BLNSFDD2	BLN5004I	I	ERROR	
BLNSFDD3	BLN5003I	I	ERROR	
BLNSFDD3	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFDID	BLN1007I	I/B/C	ERROR	
BLNSFDID	BLN3002I	I/B/C	ERROR	rsc
BLNSFDID	BLN9004I	I/B/C	ERROR	entry
BLNSFDID	BLN9019I	I/B/C	ERROR	dumpid
BLNSFDID	BLN9032I	I/B/C	ERROR	dumpid
BLNSFDMB	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFDMB	BLN9017I	I	ERROR	
BLNSFDMD	BLN2009I	I/B/C	ERROR	
BLNSFDMD	BLN9004I	I/B/C	ERROR	entry
BLNSFDMD	BLN9013I	I	ERROR	entry
BLNSFDMG	BLN9004I	I/B/C	ERROR	entry
BLNSFDMS	BLN8005I	I	ERROR	
BLNSFDMS	BLN9004I	I/B/C	ERROR	entry
BLNSFDSP	BLN9004I	I/B/C	ERROR	entry
BLNSFDST	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFDVD	BLN9004I	I/B/C	ERROR	entry
BLNSFDVO	BLN9004I	I/B/C	ERROR	entry
BLNSFDVO	BLN9013I	I	ERROR	entry
BLNSFDVS	BLN9004I	I/B/C	ERROR	entry
BLNSFFIN	BLN9014I	I	INFO	entry
BLNSFFIN	BLN9023I	I	ERROR	
BLNSFHLP	BLN9004I	I/B/C	ERROR	entry
BLNSFICB	BLN8003I	I	ERROR	serv, rtc
BLNSFIMP	BLN9005I	I	INFO	
BLNSFISP	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFLGD	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFLGD	BLN9011I	I	INFO	
BLNSFLNK	BLN9004I	I/B/C	ERROR	entry
BLNSFLN1	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFLN1	BLN9017I	I	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNSFLOC	BLN9004I	I/B/C	ERROR	entry
BLNSFLOG	BLN9004I	I/B/C	ERROR	entry
BLNSFMHH	BLN9004I	I/B/C	ERROR	entry
BLNSFMSH	BLN9004I	I/B/C	ERROR	entry
BLNSFMSH	BLN9009I	I	INFO	
BLNSFMSK	BLN5001I	I	INFO	
BLNSFMSK	BLN9004I	I/B/C	ERROR	entry
BLNSFMSS	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFMSS	BLN9017I	I	ERROR	
BLNSFNDN	BLN9004I	I/B/C	ERROR	entry
BLNSFOFL	BLN4001I	I/B/C	ERROR	
BLNSFOFL	BLN4006I	I/B/C	ERROR	
BLNSFOFL	BLN4007I	I/B/C	ERROR	
BLNSFOFL	BLN9004I	I/B/C	ERROR	entry
BLNSFOFL	BLN9013I	I	ERROR	entry
BLNSFOFL	BLN9041I	I	ERROR	entry
BLNSFONL	BLN4001I	I/B/C	ERROR	
BLNSFONL	BLN4007I	I/B/C	ERROR	
BLNSFONL	BLN9004I	I/B/C	ERROR	entry
BLNSFONL	BLN9013I	I	ERROR	entry
BLNSFONL	BLN9041I	I	ERROR	entry
BLNSFPRT	BLN8002I	I	ERROR	func, tbl, rtc
BLNSFSDC	BLN9017I	I	ERROR	
BLNSFTRM	BLN8003I	I	ERROR	serv, rtc
BLNSFVDF	BLN8003I	I	ERROR	serv, rtc
BLNSFVDF	BLN9001I	I/B/C	ERROR	
BLNSMBLE	BLN7010I	I/B/C	INFO	
BLNSMBLE	BLN9007I	I/B/C	ERROR	rsc
BLNSMBLF	BLN9001I	I/B/C	ERROR	
BLNSMBLO	BLN9001I	I/B/C	ERROR	
BLNSMBMA	BLN9001I	I/B/C	ERROR	
BLNSMBME	BLN9007I	I/B/C	ERROR	rsc
BLNSMCMO	BLN9001I	I/B/C	ERROR	
BLNSMDSP	BLN7010I	I/B/C	INFO	
BLNSMFND	BLN7010I	I/B/C	INFO	
BLNSMFND	BLN9014I	I	INFO	entry
BLNSMFND	BLN9023I	I	ERROR	
BLNSMFRE	BLN9001I	I/B/C	ERROR	
BLNSMGRE	BLN9001I	I/B/C	ERROR	
BLNSMHXA	BLN9001I	I/B/C	ERROR	
BLNSMHXA	BLN9007I	I/B/C	ERROR	rsc
BLNSMHXO	BLN9001I	I/B/C	ERROR	
BLNSMINT	BLN9007I	I/B/C	ERROR	rsc
BLNSMLKB	BLN9001I	I/B/C	ERROR	

Module Issuing	Message ID	Modes	Type	Inserts
BLNSMLKC	BLN9001I	I/B/C	ERROR	
BLNSMLKD	BLN9001I	I/B/C	ERROR	
BLNSMLKL	BLN5030I	I/B/C	INFO	
BLNSMLKL	BLN9001I	I/B/C	ERROR	
BLNSMLKO	BLN9001I	I/B/C	ERROR	
BLNSMOCL	BLN9001I	I/B/C	ERROR	
BLNSMPCD	BLN9007I	I/B/C	ERROR	rsc
BLNSMPCD	BLN9022I	I/B/C	ERROR	
BLNSMPHP	BLN9001I	I/B/C	ERROR	
BLNSMPRT	BLN5011I	I/B/C	ERROR	
BLNSMPRT	BLN9007I	I/B/C	ERROR	rsc
BLNSMPRT	BLN9012I	I/B/C	INFO	
BLNSMPSO	BLN9001I	I/B/C	ERROR	
BLNSMPSW	BLN9001I	I/B/C	ERROR	
BLNSMSEL	BLN7010I	I/B/C	INFO	
BLNSMTRM	BLN9007I	I/B/C	ERROR	rsc
BLNSMTXA	BLN9001I	I/B/C	ERROR	
BLNSMTXA	BLN9007I	I/B/C	ERROR	rsc
BLNSMTXO	BLN9001I	I/B/C	ERROR	

INFO/ANALYSIS MESSAGE/MODULE CROSS-REFERENCE

In the following list, Info/Analysis message ids are listed numerically. Each message id is followed by the name of the module that issues the message, the possible environments for each message, the type of message (error or informational), and the variables that are inserted in the message. (Refer to "Info/Analysis Module/Message Cross-Reference" on page 215 for additional information.)

Message ID	Module Issuing	Modes	Type	Inserts
BLN0920I	BLNASLCB	I/B/C	ERROR	
BLN0920I	BLNASTRM	I/B/C	ERROR	
BLN0921I	BLNASFMT	I/B/C	ERROR	
BLN0921I	BLNASGET	I/B/C	ERROR	
BLN0921I	BLNASLCB	I/B/C	ERROR	
BLN0921I	BLNASLD1	I/B/C	ERROR	
BLN0921I	BLNASLST	I/B/C	ERROR	
BLN0921I	BLNASMAP	I/B/C	ERROR	
BLN0921I	BLNASMP1	I/B/C	ERROR	
BLN0921I	BLNASRSR	I/B/C	ERROR	
BLN0923I	BLNASADF	I/B/C	INFO	
BLN0923I	BLNASADL	I/B/C	INFO	

Message ID	Module Issuing	Modes	Type	Inserts
BLN0923I	BLNASAXA	I/B/C	INFO	
BLN0923I	BLNASAXC	I/B/C	INFO	
BLN0923I	BLNASAXK	I/B/C	INFO	
BLN0923I	BLNASAXT	I/B/C	INFO	
BLN0923I	BLNASAXX	I/B/C	INFO	
BLN0923I	BLNASMD	I/B/C	INFO	
BLN0926I	BLNASLD1	I/B/C	ERROR	
BLN0926I	BLNASLOC	I/B/C	ERROR	
BLN0927I	BLNASMAP	I/B/C	ERROR	
BLN0927I	BLNASRSR	I/B/C	ERROR	
BLN0928I	BLNASRSR	I/B/C	ERROR	rsc
BLN0929I	BLNALADF	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASADL	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASAXA	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASAXC	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASAXK	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASAXT	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASAXX	I/B/C	INFO	rcd, ofst
BLN0929I	BLNASMD	I/B/C	INFO	rcd, ofst
BLN0930I	BLNASADF	I/B/C	INFO	
BLN0930I	BLNASADL	I/B/C	INFO	
BLN0930I	BLNASAXA	I/B/C	INFO	
BLN0930I	BLNASAXC	I/B/C	INFO	
BLN0930I	BLNASAXK	I/B/C	INFO	
BLN0930I	BLNASAXT	I/B/C	INFO	
BLN0930I	BLNASAXX	I/B/C	INFO	
BLN0931I	BLNASAXA	I/B/C	INFO	
BLN0931I	BLNASAXC	I/B/C	INFO	
BLN0931I	BLNASAXK	I/B/C	INFO	
BLN0931I	BLNASAXT	I/B/C	INFO	
BLN0932I	BLNASMD	I/B/C	INFO	
BLN0933I	BLNASAXX	I/B/C	INFO	
BLN0934I	BLNASMD	I/B/C	INFO	
BLN1003I	BLNBADM2	B/C	ERROR	
BLN1004I	BLNBABH2	B	INFO	
BLN1004I	BLNBAFLS	B	INFO	
BLN1004I	BLNBARDR	B	INFO	
BLN1005D	BLNBARDR	C	INFO	func
BLN1006I	BLNBARDR	B	ERROR	
BLN1007I	BLNBABH1	I/B/C	ERROR	
BLN1007I	BLNBADM1	I/B/C	ERROR	
BLN1007I	BLNSFDID	I/B/C	ERROR	
BLN1008I	BLNBAINT	B/C	INFO	
BLN1009I	BLNBARDR	B/C	INFO	

Message ID	Module Issuing	Modes	Type	Inserts
BLN1013I	BLNMACBC	I/B/C	ERROR	
BLN2006I	BLNDMADU	I/B/C	ERROR	
BLN2007I	BLNDMCNT	I/B/C	INFO	maxrec, userec
BLN2009I	BLNDMDEL	I/B/C	ERROR	
BLN2009I	BLNSFDMD	I/B/C	ERROR	
BLN2013I	BLNDMDMU	I/B/C	ERROR	rsc
BLN2013I	BLNDMRDF	I/B/C	ERROR	
BLN2013I	BLNDMWDF	I/B/C	ERROR	
BLN2014I	BLNDMDMU	B/C	ERROR	
BLN2017I	BLNDMDDL	I/B/C	ERROR	
BLN2017I	BLNDMDEL	I/B/C	ERROR	
BLN2017I	BLNDMPDL	I/B/C	ERROR	
BLN2019I	BLNDMAD1	I/B/C	INFO	
BLN2020I	BLNDMDMU	B/C	INFO	
BLN3002I	BLNONEND	I/B/C	ERROR	rsc
BLN3002I	BLNONINT	I/B/C	ERROR	rsc
BLN3002I	BLNSFDID	I/B/C	ERROR	rsc
BLN4001I	BLNCMVOL	I/B/C	ERROR	
BLN4001I	BLNSFOFL	I/B/C	ERROR	
BLN4001I	BLNSFONL	I/B/C	ERROR	
BLN4003I	BLNOFFLD	I/B/C	ERROR	
BLN4004I	BLNOFFLD	I/B/C	INFO	dumpid, ERASED
BLN4005I	BLNOFFLD	I/B/C	ERROR	rsc
BLN4006I	BLNBAPF1	I/B/C	ERROR	
BLN4006I	BLNSFOFL	I/B/C	ERROR	
BLN4007I	BLNOFFLD	I/B/C	ERROR	
BLN4007I	BLNONINT	I/B/C	ERROR	
BLN4007I	BLNSFOFL	I/B/C	ERROR	
BLN4007I	BLNSFONL	I/B/C	ERROR	
BLN4010I	BLNOFFLD	I/B/C	INFO	
BLN5001I	BLNSFMSK	I	INFO	
BLN5002I	BLNFNDDO	I	INFO	
BLN5003I	BLNSFDD3	I	ERROR	
BLN5004I	BLNSFDD2	I	ERROR	
BLN5005I	BLNDDBAS	I	ERROR	addr
BLN5005I	BLNDDCDB	I	ERROR	addr
BLN5006I	BLNDDCDB	I	ERROR	entry
BLN5006I	BLNDDQAL	I	ERROR	entry
BLN5007I	BLNDDCDB	I	ERROR	entry
BLN5008I	BLNMKMSK	I	INFO	
BLN5009I	BLNDDFMT	I	ERROR	
BLN5011I	BLNARPRT	I/B/C	ERROR	
BLN5011I	BLNDDPOP	I/B/C	ERROR	
BLN5011I	BLNSMPRT	I/B/C	ERROR	

Message ID	Module Issuing	Modes	Type	Inserts
BLN5012I	BLNMKVMR	I	ERROR	
BLN5013I	BLNDDFDO	I/B/C	INFO	
BLN5014I	BLNDDHDO	I/B/C	INFO	block
BLN5014I	BLNDDMDT	I/B/C	INFO	block
BLN5014I	BLNDDRPT	I/B/C	INFO	block
BLN5015I	BLNMKVMR	I	ERROR	
BLN5016I	BLNMKMSK	I	ERROR	
BLN5017I	BLNMKBMD	I	INFO	
BLN5018I	BLNMKBMD	I	INFO	
BLN5022I	BLNDDPOP	I/B/C	ERROR	
BLN5022I	BLNDDVFA	I/B/C	ERROR	
BLN5022I	BLNMKVMR	I/B/C	ERROR	
BLN5030I	BLNDDCDB	I/B/C	INFO	
BLN5030I	BLNDDFDZ	I/B/C	INFO	
BLN5030I	BLNDDFPP	I/B/C	INFO	
BLN5030I	BLNMKQRY	I/B/C	INFO	
BLN5030I	BLNSMLKL	I/B/C	INFO	
BLN7010I	BLNSMBLE	I/B/C	INFO	
BLN7010I	BLNSMDSP	I/B/C	INFO	
BLN7010I	BLNSMFND	I/B/C	INFO	
BLN7010I	BLNSMSEL	I/B/C	INFO	
BLN7013I	BLNLBVER	I/B/C	ERROR	block, rsc
BLN7515I	BLNAREXD	B/C	ERROR	rtn
BLN7516I	BLNAREXL	I/B/C	ERROR	rtn, rsc
BLN7518I	BLNAXSU1	I/B/C	ERROR	
BLN7520I	BLNAREXD	B/C	ERROR	
BLN7521I	BLNAXPD3	I/B/C	INFO	
BLN7522I	BLNAXPD3	I/B/C	INFO	
BLN7522I	BLNAXPD4	I/B/C		
BLN7540I	BLNAREXD	I/B/C	ERROR	
BLN7540I	BLNAREXC	I/B/C		
BLN7540I	BLNARDSP	I/B/C		
BLN7540I	BLNARPRT	I/B/C		
BLN8001I	BLNSCPRD	I	ERROR	pnl
BLN8002I	BLNSCCRP	I	ERROR	func, tbl, rtc
BLN8002I	BLNSCSUB	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFAMS	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFAS1	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFBST	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFBUM	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFCA1	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFDD3	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFDMB	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFDST	I	ERROR	func, tbl, rtc

Message ID	Module Issuing	Modes	Type	Inserts
BLN8002I	BLNSFISP	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFLGD	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFLN1	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFMSS	I	ERROR	func, tbl, rtc
BLN8002I	BLNSFPRT	I	ERROR	func, tbl, rtc
BLN8003I	BLNSCPRD	I	ERROR	serv, rtc
BLN8003I	BLNSFICB	I	ERROR	serv, rtc
BLN8003I	BLNSFTRM	I	ERROR	serv, rtc
BLN8003I	BLNSFVDF	I	ERROR	serv, rtc
BLN8004I	BLNSCBKW	I	ERROR	
BLN8004I	BLNSCDIS	I	ERROR	
BLN8004I	BLNSCFWD	I	ERROR	
BLN8004I	BLNSCLFT	I	ERROR	
BLN8004I	BLNSCRGH	I	ERROR	
BLN8005I	BLNSFDMS	I	ERROR	
BLN8006I	BLNSCPRD	I	ERROR	pnl, rtc
BLN9001I	BLNARDLO	I/B/C	ERROR	
BLN9001I	BLNAREXA	I/B/C	ERROR	
BLN9001I	BLNARGRN	I/B/C	ERROR	
BLN9001I	BLNARPRN	I/B/C	ERROR	
BLN9001I	BLNARXPL	I/B/C	ERROR	
BLN9001I	BLNARPD6	I/B/C	ERROR	
BLN9001I	BLNARS6U	I/B/C	ERROR	
BLN9001I	BLNBAINT	I/B/C	ERROR	
BLN9001I	BLNCMVOL	I/B/C	ERROR	
BLN9001I	BLNDDCDB	I/B/C	ERROR	
BLN9001I	BLNDDFMT	I/B/C	ERROR	
BLN9001I	BLNDDFPP	I/B/C	ERROR	
BLN9001I	BLNDDINT	I/B/C	ERROR	
BLN9001I	BLNDDNBU	I/B/C	ERROR	
BLN9001I	BLNDDSYI	I/B/C	ERROR	
BLN9001I	BLNDMDDL	I/B/C	ERROR	
BLN9001I	BLNDMPDL	I/B/C	ERROR	
BLN9001I	BLNDSGBF	I/B/C	ERROR	
BLN9001I	BLNDVKCB	I/B/C	ERROR	
BLN9001I	BLNDVSIA	I/B/C	ERROR	
BLN9001I	BLNLBARC	I/B/C	ERROR	
BLN9001I	BLNLBARR	I/B/C	ERROR	
BLN9001I	BLNLBCAC	I/B/C	ERROR	
BLN9001I	BLNLBCAE	I/B/C	ERROR	
BLN9001I	BLNLBCAI	I/B/C	ERROR	
BLN9001I	BLNLBCAL	I/B/C	ERROR	
BLN9001I	BLNLBCHC	I/B/C	ERROR	
BLN9001I	BLNLBCHE	I/B/C	ERROR	

Message ID	Module Issuing	Modes	Type	Inserts
BLN9001I	BLNLBCHI	I/B/C	ERROR	
BLN9001I	BLNLBCHL	I/B/C	ERROR	
BLN9001I	BLNLBCHR	I/B/C	ERROR	
BLN9001I	BLNLBCHW	I/B/C	ERROR	
BLN9001I	BLNLBLAL	I/B/C	ERROR	
BLN9001I	BLNLBSIM	I/B/C	ERROR	
BLN9001I	BLNMKBMD	I/B/C	ERROR	
BLN9001I	BLNSCDTB	I/B/C	ERROR	
BLN9001I	BLNSCRDT	I/B/C	ERROR	
BLN9001I	BLNSFVDF	I/B/C	ERROR	
BLN9001I	BLNSMBLF	I/B/C	ERROR	
BLN9001I	BLNSMBLO	I/B/C	ERROR	
BLN9001I	BLNSMBMA	I/B/C	ERROR	
BLN9001I	BLNSMCMO	I/B/C	ERROR	
BLN9001I	BLNSMFRE	I/B/C	ERROR	
BLN9001I	BLNSMGRE	I/B/C	ERROR	
BLN9001I	BLNSMHXA	I/B/C	ERROR	
BLN9001I	BLNSMHXO	I/B/C	ERROR	
BLN9001I	BLNSMLKB	I/B/C	ERROR	
BLN9001I	BLNSMLKC	I/B/C	ERROR	
BLN9001I	BLNSMLKD	I/B/C	ERROR	
BLN9001I	BLNSMLKL	I/B/C	ERROR	
BLN9001I	BLNSMLKO	I/B/C	ERROR	
BLN9001I	BLNSMOCL	I/B/C	ERROR	
BLN9001I	BLNSMPHP	I/B/C	ERROR	
BLN9001I	BLNSMPSO	I/B/C	ERROR	
BLN9001I	BLNSMPSW	I/B/C	ERROR	
BLN9001I	BLNSMTXA	I/B/C	ERROR	
BLN9001I	BLNSMTXO	I/B/C	ERROR	
BLN9002I	BLNAREXG	I/B/C	ERROR	rtc, rsc
BLN9002I	BLNCMERM	I/B/C	ERROR	rtc, rsc
BLN9002I	BLNMACBG	I/B/C	ERROR	rtc, rsc
BLN9003I	BLNCMERM	I/B/C	ERROR	msg
BLN9004I	BLNBABH1	I/B/C	ERROR	entry
BLN9004I	BLNBADM1	I/B/C	ERROR	entry
BLN9004I	BLNBADM2	I/B/C	ERROR	entry
BLN9004I	BLNBADS1	I/B/C	ERROR	entry
BLN9004I	BLNBADV1	I/B/C	ERROR	entry
BLN9004I	BLNBAPF1	I/B/C	ERROR	entry
BLN9004I	BLNBAPN1	I/B/C	ERROR	entry
BLN9004I	BLNCMFIL	I/B/C	ERROR	entry
BLN9004I	BLNCMHEX	I/B/C	ERROR	entry
BLN9004I	BLNCMVOL	I/B/C	ERROR	entry
BLN9004I	BLNCMXCF	I/B/C	ERROR	entry

Message ID	Module Issuing	Modes	Type	Inserts
BLN9004I	BLNDDPOP	I/B/C	ERROR	entry
BLN9004I	BLNDDVFA	I/B/C	ERROR	entry
BLN9004I	BLNDDVFN	I/B/C	ERROR	entry
BLN9004I	BLNDMUAD	I/B/C	ERROR	entry
BLN9004I	BLNMKVMR	I/B/C	ERROR	entry
BLN9004I	BLNSFASM	I/B/C	ERROR	entry
BLN9004I	BLNSFCAR	I/B/C	ERROR	entry
BLN9004I	BLNSFDAT	I/B/C	ERROR	entry
BLN9004I	BLNSFDID	I/B/C	ERROR	entry
BLN9004I	BLNSFDMD	I/B/C	ERROR	entry
BLN9004I	BLNSFDMG	I/B/C	ERROR	entry
BLN9004I	BLNSFDMS	I/B/C	ERROR	entry
BLN9004I	BLNSFDSP	I/B/C	ERROR	entry
BLN9004I	BLNSFDVD	I/B/C	ERROR	entry
BLN9004I	BLNSFDVO	I/B/C	ERROR	entry
BLN9004I	BLNSFDVS	I/B/C	ERROR	entry
BLN9004I	BLNSFHLP	I/B/C	ERROR	entry
BLN9004I	BLNSFLNK	I/B/C	ERROR	entry
BLN9004I	BLNSFLOC	I/B/C	ERROR	entry
BLN9004I	BLNSFLOG	I/B/C	ERROR	entry
BLN9004I	BLNSFMHH	I/B/C	ERROR	entry
BLN9004I	BLNSFMSH	I/B/C	ERROR	entry
BLN9004I	BLNSFMSK	I/B/C	ERROR	entry
BLN9004I	BLNSFNDN	I/B/C	ERROR	entry
BLN9004I	BLNSFOFL	I/B/C	ERROR	entry
BLN9004I	BLNSFONL	I/B/C	ERROR	entry
BLN9005I	BLNSFIMP	I	INFO	
BLN9006I	BLNCMCHD	I/B/C	INFO	
BLN9006I	BLNDMAD1	I/B/C	INFO	
BLN9006I	BLNDSCEN	I/B/C	INFO	
BLN9007I	BLNAREXA	I/B/C	ERROR	rsc
BLN9007I	BLNARINT	I/B/C	ERROR	rsc
BLN9007I	BLNDDFPP	I/B/C	ERROR	rsc
BLN9007I	BLNDSDSP	I/B/C	ERROR	rsc
BLN9007I	BLNDSPRT	I/B/C	ERROR	rsc
BLN9007I	BLNDSTRM	I/B/C	ERROR	rsc
BLN9007I	BLNDSYMP	I/B/C	ERROR	rsc
BLN9007I	BLNDVSIN	I/B/C	ERROR	rsc
BLN9007I	BLNFNARC	I/B/C	ERROR	rsc
BLN9007I	BLNFNDDC	I/B/C	ERROR	rsc
BLN9007I	BLNFNDMC	I/B/C	ERROR	rsc
BLN9007I	BLNFNDSC	I/B/C	ERROR	rsc
BLN9007I	BLNFNDVS	I/B/C	ERROR	rsc
BLN9007I	BLNFNMSK	I/B/C	ERROR	rsc

Message ID	Module Issuing	Modes	Type	Inserts
BLN9007I	BLNFNPFC	I/B/C	ERROR	rsc
BLN9007I	BLNFNPNC	I/B/C	ERROR	rsc
BLN9007I	BLNFNSMC	I/B/C	ERROR	rsc
BLN9007I	BLNMKDSP	I/B/C	ERROR	rsc
BLN9007I	BLNMKINT	I/B/C	ERROR	rsc
BLN9007I	BLNMKMSK	I/B/C	ERROR	rsc
BLN9007I	BLNSMBLE	I/B/C	ERROR	rsc
BLN9007I	BLNSMBME	I/B/C	ERROR	rsc
BLN9007I	BLNSMHXA	I/B/C	ERROR	rsc
BLN9007I	BLNSMINT	I/B/C	ERROR	rsc
BLN9007I	BLNSMPCD	I/B/C	ERROR	rsc
BLN9007I	BLNSMPRT	I/B/C	ERROR	rsc
BLN9007I	BLNSMTRM	I/B/C	ERROR	rsc
BLN9007I	BLNSMTXA	I/B/C	ERROR	rsc
BLN9008I	BLNBARDR	I/B/C	ERROR	mod, rsc
BLN9008I	BLNDMGNT	I/B/C	ERROR	mod, rsc
BLN9008I	BLNSCPRD	I/B/C	ERROR	mod, rsc
BLN9009I	BLNSFMSH	I	INFO	
BLN9010I	BLNDDMDT	I/B/C	INFO	
BLN9010I	BLNDDRPT	I/B/C	INFO	
BLN9011I	BLNSFCMQ	I	INFO	
BLN9011I	BLNSFLGD	I	INFO	
BLN9012I	BLNARPRT	I/B/C	INFO	
BLN9012I	BLNAXPD2	I/B/C	INFO	
BLN9012I	BLNDDDPT	I/B/C	INFO	
BLN9012I	BLNDDFPT	I/B/C	INFO	
BLN9012I	BLNDDPRT	I/B/C	INFO	
BLN9012I	BLNDDRPT	I/B/C	INFO	
BLN9012I	BLNDMPDL	I/B/C	INFO	
BLN9012I	BLNDSPRT	I/B/C	INFO	
BLN9012I	BLNSMPRT	I/B/C	INFO	
BLN9013I	BLNFNDDO	I	ERROR	entry
BLN9013I	BLNSFDMD	I	ERROR	entry
BLN9013I	BLNSFDVO	I	ERROR	entry
BLN9013I	BLNSFOFL	I	ERROR	entry
BLN9013I	BLNSFONL	I	ERROR	entry
BLN9014I	BLNARFND	I	INFO	entry
BLN9014I	BLNDDFFN	I	INFO	entry
BLN9014I	BLNDDHFN	I	INFO	entry
BLN9014I	BLNSFFIN	I	INFO	entry
BLN9014I	BLNSMFND	I	INFO	entry
BLN9016I	BLNBADM1	B/C	ERROR	
BLN9016I	BLNBADS1	B/C	ERROR	
BLN9016I	BLNBAPF1	B/C	ERROR	

Message ID	Module Issuing	Modes	Type	Inserts
BLN9016I	BLNBAPN1	B/C	ERROR	
BLN9017I	BLNSFAS1	I	ERROR	
BLN9017I	BLNSFCA1	I	ERROR	
BLN9017I	BLNSFDMB	I	ERROR	
BLN9017I	BLNSFLN1	I	ERROR	
BLN9017I	BLNSFMSS	I	ERROR	
BLN9017I	BLNSFSDC	I	ERROR	
BLN9018I	BLNDMADU	I/B/C	INFO	dumpid, func
BLN9018I	BLNDMDEL	I/B/C	INFO	dumpid, func
BLN9018I	BLNONEND	I/B/C	INFO	dumpid, func
BLN9019I	BLNBABH1	I/B/C	ERROR	dumpid
BLN9019I	BLNCMCHD	I/B/C	ERROR	dumpid
BLN9019I	BLNOFFLD	I/B/C	ERROR	dumpid
BLN9019I	BLNSFDID	I/B/C	ERROR	dumpid
BLN9020I	BLNDDRDF	I	INFO	
BLN9021I	BLNDDRDF	I/B/C	INFO	
BLN9022I	BLNARPRT	I/B/C	ERROR	
BLN9022I	BLNAXPD2	I/B/C	ERROR	
BLN9022I	BLNAXPD6	I/B/C	ERROR	
BLN9022I	BLNDDFPP	I/B/C	ERROR	
BLN9022I	BLNDDPRT	I/B/C	ERROR	
BLN9022I	BLNDDRPT	I/B/C	ERROR	
BLN9022I	BLNSMPCD	I/B/C	ERROR	
BLN9023I	BLNARFND	I	ERROR	
BLN9023I	BLNDDVFA	I	ERROR	
BLN9023I	BLNDDVFN	I	ERROR	
BLN9023I	BLNSFFIN	I	ERROR	
BLN9023I	BLNSMFND	I	ERROR	
BLN9030I	BLNCMXDE	I/B/C	ERROR	req, fil, rsc
BLN9031I	BLNCMERM	I/B/C	ERROR	rtc, rsc
BLN9032I	BLNBABH1	I/B/C	ERROR	dumpid
BLN9032I	BLNSFDID	I/B/C	ERROR	dumpid
BLN9040I	BLNOFFLD	I	INFO	jobname
BLN9040I	BLNONINT	I	INFO	jobname
BLN9041I	BLNSFOFL	I/B/C	ERROR	entry
BLN9041I	BLNSFONL	I/B/C	ERROR	entry

INFO/ANALYSIS MODULE/REASON CROSS-REFERENCE

In the following list, object modules that use reason codes are listed alphabetically by module name. Each module name is followed by the reason code that the module uses, the message id in which it may be inserted, and a description of the reason code.

Module Issuing	Reason Code	Message ID	Description
BLNADIFI	0004		Invalid dump access entry point
BLNADIFI	0008		Storage allocation failure
BLNADIFI	0012		Unable to get ADCB
BLNADIFR	0900		Unable to get ADCB
BLNADIFX	0900		Unable to get ADCB
BLNADIFX	0904		Unable to delete ADCB from BLX
BLNARDLO	7550		Storage allocation failure
BLNARDSP	7540		No analysis routines list available
BLNARDSP	7541		Initialization error
BLNAREXA	7544	BLN9007I	Unable to add ARCB to BLX
BLNAREXA	7592		Storage allocation failure
BLNAREXC	7570		Initialization error
BLNAREXG	7581	BLN9002I	Unable to delete ARCB from BLX
BLNAREXL	7582	BLN7516I	Load of analysis routine failed
BLNARFND	7567		Initialization error
BLNARFND	7568		No analysis routines list available
BLNARGRN	7503		Storage allocation failure
BLNARGRN	7504		Storage allocation failure
BLNARINT	7502	BLN9007I	Duplicate initialization
BLNARPRN	7505		Storage allocation failure
BLNARPRT	7560		Initialization not done
BLNARPRT	7567		Initialization error
BLNARXPL	7583		Storage allocation failure
BLNASADF	0923		Format error in section 6
BLNASADL	0923		Format error in section 6
BLNASAXA	0923		Format error in section 6
BLNASAXC	0923		Format error in section 6
BLNASAXK	0923		Format error in section 6
BLNASAXT	0923		Format error in section 6
BLNASAXX	0923		Format error in section 6
BLNASFMT	0921		Storage allocation failure
BLNASFSY	0925		Data not found
BLNASGET	0921		Storage allocation failure
BLNASLCB	0920		Unable to get ASCB
BLNASLCB	0921		Storage allocation failure
BLNASLD1	0921		Storage allocation failure
BLNASLD1	0925		Data not found
BLNASLD1	0926		Error in parameter list
BLNASLD6	0925		Data not found
BLNASLOC	0926		Error in parameter list
BLNASLST	0921		Storage allocation failure
BLNASLST	0925		Data not found
BLNASMAP	0921		Storage allocation failure
BLNASMAP	0927		Invalid dump name

Module Issuing	Reason Code	Message ID	Description
BLNASMD	0923		Format error in section 6
BLNASMP1	0921		Storage allocation failure
BLNASRSR	0921		Storage allocation failure
BLNASRSR	0925		Data not found
BLNASRSR	0928		Dump access error
BLNASTRM	0920		Unable to get ASCB
BLNAXDAC	0100		Invalid environment pointer
BLNAXDA1	0016		Invalid request type
BLNAXDA1	0040		Dump name changed
BLNAXDA1	0100		Invalid environment pointer
BLNAXFMT	0100		Invalid environment pointer
BLNAXFM1	0008		Storage length too small
BLNAXFM1	0016		Storage allocation failure
BLNAXFM1	0096		Unable to access ARCB
BLNAXGST	0100		Invalid environment pointer
BLNAXGS1	0096		Unable to access ARCB
BLNAXGS2	0004		Storage allocation failure
BLNAXGS2	0008		Invalid length request
BLNAXGS3	0012		Invalid free address passed
BLNAXPDS	0100		Invalid environment pointer
BLNAXPD1	0016		No output device specified
BLNAXPD1	0020		Invalid buffer length
BLNAXPD1	0096		Unable to access ARCB
BLNAXPD3	0004		Output routed to alternate device
BLNAXPD3	0096		Unable to access ARCB
BLNAXPD5	0096		Unable to access ARCB
BLNAXPD6	0096		Unable to access ARCB
BLNAXPD7	0096		Unable to access ARCB
BLNAXSAC	0100		Invalid environment pointer
BLNAXSA1	0100		Invalid environment pointer
BLNAXSA1	0926		Error in parameter list
BLNAXSUA	0024		Symptom record update error
BLNAXSUA	0092		Storage allocation failure
BLNAXSUP	0100		Invalid environment pointer
BLNAXSU1	0004		Symptom record has been modified
BLNAXSU1	0016		Symptom record format invalid
BLNAXSU1	0020		Dump name changed
BLNAXSU1	0092		Storage allocation failure
BLNAXSU1	0096		Unable to access ARCB
BLNAXS3U	0092		Storage allocation failure
BLNAXS4U	0092		Storage allocation failure
BLNAXS5U	0092		Storage allocation failure
BLNAXS6U	0024		Symptom record update error
BLNAXS6U	0092		Storage allocation failure

Module Issuing	Reason Code	Message ID	Description
BLNAXS6V	0012		Section 6 updates invalid
BLNAXS6V	0024		Symptom record update error
BLNCMCHD	0925		Data not found
BLNCMXDE	9501		BLX allocation - invalid unit
BLNCMXDE	9502		BLX allocation - extract macro error
BLNCMXDE	9503		BLX allocation - device not supported
BLNCMXDE	9504		BLX allocation - RECFM not supported
BLNCMXDE	9505		BLX allocation - blocked records not supported
BLNCMXDE	9506		BLX allocation - access mode not supported
BLNCMXDE	9507		BLX allocation - DLBL statement not supported
BLNCMXDE	9508		BLX allocation - open VTOC not supported
BLNCMXDE	9509		BLX allocation - close VTOC not supported
BLNCMXDE	9510		BLX allocation - extent VOLSER invalid
BLNCMXDE	9511		BLX allocation - extent logical unit invalid
BLNCMXDE	9512		BLX allocation - DDNAME not found
BLNCMXDE	9519		BLX free error
BLNCMXDE	9521		BLX open - permanent open error
BLNCMXDE	9522		BLX open - invalid FSEQ keyword
BLNCMXDE	9523		BLX open - data set already open
BLNCMXDE	9531		BLX close - permanent close error
BLNCMXDE	9532		BLX close - end of extent
BLNCMXDE	9541		BLX read - update not allowed
BLNCMXDE	9542		BLX read - wrong access type
BLNCMXDE	9543		BLX read - buffer length too small
BLNCMXDE	9544		BLX read - invalid relative record number
BLNCMXDE	9545		BLX read - VSAM position error
BLNCMXDE	9546		BLX read - invalid record length
BLNCMXDE	9547		BLX read - invalid key length
BLNCMXDE	9548		BLX read - VSAM MODCB verb failed
BLNCMXDE	9549		BLX read - undefined request
BLNCMXDE	9591		Unknown error condition
BLNDDBAS	5006		Invalid characters in BASE
BLNDDBAS	5007		Imbedded blanks in BASE
BLNDDCDB	5023		Storage allocation failure
BLNDDFDZ	5015		Dump data not available
BLNDDFMT	5014		Storage allocation failure
BLNDDFND	5018		Invalid find command
BLNDDFPP	5019		Storage allocation failure
BLNDDFPP	5020		Storage allocation failure
BLNDDFPP	5021	BLN9007I	Error freeing buffer chain
BLNDDFPP	5022		Error printing buffers
BLNDDINT	5002		Storage allocation failure
BLNDDINT	5003		Storage allocation failure
BLNDDNBU	5016		Storage allocation failure

Module Issuing	Reason Code	Message ID	Description
BLNDDPRT	5017		Invalid print command
BLNDDQAL	5008		Invalid characters in QUAL
BLNDDQAL	5009		Imbedded blanks in QUAL
BLNDDRDF	5010		Locator not found
BLNDDRDF	5011		Invalid base in locator
BLNDDRDF	5012		Invalid length in locator
BLNDDSYI	5004		Storage allocation failure
BLNDDVRF	5013		DISPLAY and BASE fields blank
BLNDMAD1	0925		Data not found
BLNDMDEL	2204		Dump access failure
BLNDMDEL	2208		No dump delete verification
BLNDMRDF	2004	BLN2013I	Dump management file open error
BLNDMRDF	2008	BLN2013I	Storage allocation failure
BLNDMRDF	2012	BLN2013I	Dump management file close error
BLNDMRDF	2020	BLN2013I	Dump management file read error
BLNDMRDF	2028	BLN2013I	Invalid control record in dump management file
BLNDMRD1	2024	BLN2013I	Invalid data record in dump management file
BLNDMUAI	2008		Storage allocation failure
BLNDMUDI	0925		Data not found
BLNDMUDI	2008		Storage allocation failure
BLNDMWDF	2004	BLN2013I	Dump management file open error
BLNDMWDF	2012	BLN2013I	Dump management file close error
BLNDMWDF	2016	BLN2013I	Dump management file write error
BLNDSCEN	6020		No symptom record in dump
BLNDSDSP	6004	BLN9007I	Damage to DSCB suspected
BLNDSGBF	6008		Storage allocation failed
BLNDSPRT	6004	BLN9007I	Damage to DSCB suspected
BLNDSPRT	6012		Permanent I/O error on print
BLNDSPRT	6016		BLX error
BLNDSTRM	6004	BLN9007I	Damage to DSCB suspected
BLNDSYMP	6000	BLN9007I	Unknown subfunction request
BLNDVKCP	5901		Storage allocation failure
BLNDVSIA	5921		Storage allocation failure
BLNDVSIA	5922		Storage allocation failure
BLNDVSIA	5923		Storage allocation failure
BLNDVSIA	5924		Storage allocation failure
BLNDVSIA	5925		Storage allocation failure
BLNDVSIN	5902	BLN9007I	Unable to access DVVT
BLNFNARC	7501	BLN9007I	Unknown subfunction request
BLNFNDDC	5001	BLN9007I	Unknown subfunction request
BLNFNDMC	1101	BLN9007I	Unable to access DMCB
BLNFNDMC	1103	BLN9007I	Unable to access DSCB
BLNFNDVS	5701	BLN9007I	Unknown subfunction request
BLNFNMSK	5901	BLN9007I	Unknown subfunction request

Module Issuing	Reason Code	Message ID	Description
BLNFNPFC	1101	BLN9007I	Unable to access DMCB
BLNFNPNC	1101	BLN9007I	Unable to access DMCB
BLNFNSMC	7099	BLN9007I	Unknown subfunction request
BLNLBARC	7406		Storage allocation failure
BLNLBARC	7407		Storage allocation failure
BLNLBARR	7408		Storage allocation failure
BLNLBCAC	7464		Storage allocation failure
BLNLBCAC	7477		Storage allocation failure
BLNLBCAE	7465		Storage allocation failure
BLNLBCAE	7475		Storage allocation failure
BLNLBCAI	7466		Storage allocation failure
BLNLBCAI	7476		Storage allocation failure
BLNLBCAL	7467		Storage allocation failure
BLNLBCAL	7468		Storage allocation failure
BLNLBCHC	7473		Storage allocation failure
BLNLBCHC	7483		Storage allocation failure
BLNLBCHE	7472		Storage allocation failure
BLNLBCHE	7478		Storage allocation failure
BLNLBCHI	7471		Storage allocation failure
BLNLBCHI	7481		Storage allocation failure
BLNLBCHL	7469		Storage allocation failure
BLNLBCHL	7479		Storage allocation failure
BLNLBCHR	7474		Storage allocation failure
BLNLBCHW	7470		Storage allocation failure
BLNLBLAL	7460		Storage allocation failure
BLNLBSIM	7405		Storage allocation failure
BLNLBVER	7401	BLN7013I	Unknown number of elements in array
BLNLBVER	7402	BLN7013I	Array extension type = B, no chain extension
BLNLBVER	7403	BLN7013I	Invalid array extension type field
BLNLBVER	7411	BLN7013I	Invalid end condition field
BLNLBVER	7412	BLN7013I	Invalid end condition length field
BLNLBVER	7413	BLN7013I	Invalid chain extension type field
BLNLBVER	7414	BLN7013I	Invalid chain extension address list pointer
BLNLBVER	7421	BLN7013I	Invalid hex extension length
BLNLBVER	7431	BLN7013I	Invalid text extension length
BLNLBVER	7441	BLN7013I	Keyfield length is zero
BLNLBVER	7442	BLN7013I	Invalid keyfield format field
BLNLBVER	7443	BLN7013I	Invalid keyfield type field
BLNLBVER	7451	BLN7013I	Number of entries in format descriptor is zero
BLNLBVER	7451	BLN7013I	Number of entries in linkage descriptor is zero
BLNLBVER	7452	BLN7013I	Length error in format descriptor
BLNLBVER	7452	BLN7013I	Length error in linkage descriptor
BLNLBVER	7453	BLN7013I	Invalid offset attribute type
BLNLBVER	7454	BLN7013I	Invalid sequential offset format

Module Issuing	Reason Code	Message ID	Description
BLNLBVER	7455	BLN7013I	Invalid specific offset format
BLNMACBG	1104	BLN9002I	Unable to access control blocks
BLNMKBMD	5904		Storage allocation failure
BLNMKDSP	5903	BLN9007I	Initialization error
BLNMKINT	5902	BLN9007I	Initialization error
BLNMKMSK	5905	BLN9007I	Initialization error
BLNOFFLD	4004	BLN4005I	Error deleting dump from library
BLNOFFLD	4008	BLN4005I	Error adding VOLID to dump management file
BLNOFFLD	4016	BLN4005I	Storage allocation failure
BLNOFFLD	4020	BLN4005I	No VOLID specified
BLNOFFLD	4024	BLN4005I	Dump not in dump management list
BLNOFFLD	4028	BLN4005I	Read of dump management list failed
BLNOFFLD	4030	BLN4005I	Invalid bypass-no previous offload
BLNOFFLD	4032	BLN4005I	Invalid bypass-dump altered
BLNOFFLD	4036	BLN4005I	Dump not in library
BLNOFFLD	4040	BLN4005I	Job submit failed
BLNOFOUT	4012	BLN4005I	Error writing dump to tape
BLNOFOUT	4040	BLN4005I	Tape allocate error
BLNOFOUT	4044	BLN4005I	Tape open error
BLNOFOUT	4048	BLN4005I	Tape close error
BLNOFOUT	4052	BLN4005I	Tape deallocate error
BLNOFUDM	4028	BLN4005I	Read of dump management list failed
BLNONDDT	3016	BLN3002I	Read dump tape failed
BLNONEND	3016	BLN3002I	Read dump tape failed
BLNONEND	3020	BLN3002I	Tape open error
BLNONEND	3024	BLN3002I	Tape allocate error
BLNONEND	3036	BLN3002I	Write dump management file error
BLNONINT	3020	BLN3002I	Tape open error
BLNONINT	3024	BLN3002I	Tape allocate error
BLNONINT	3032	BLN3002I	Error reading dump management file
BLNONINT	3040	BLN3002I	Error processing dump management file
BLNONINT	3044	BLN4007I	No VOLID specified
BLNONINT	3048	BLN3002I	Dump already in library
BLNONINT	3052	BLN3002I	Multi-dump not allowed
BLNONINT	3056	BLN3002I	Job submit failed
BLNONLOD	3012	BLN3002I	Error writing dump to system
BLNONMUD	3008	BLN3002I	Undefined dump type
BLNONMUD	3016	BLN3002I	Read dump tape failed
BLNONMUD	3020	BLN3002I	Tape open error
BLNONMUD	3022	BLN3002I	Tape close error
BLNONMUD	3060	BLN3002I	No operator requested dump
BLNONSCM	3016	BLN3002I	Read dump tape failed
BLNONSCM	3020	BLN3002I	Tape open error
BLNONSCM	3022	BLN3002I	Tape close error

Module Issuing	Reason Code	Message ID	Description
BLNONSCM	3028	BLN3002I	Storage allocation failure
BLNONSC1	3012	BLN3002I	Error writing dump to system
BLNONSC1	3016	BLN3002I	Read dump tape failed
BLNONTRW	3012	BLN3002I	Error writing dump to system
BLNONTRW	3016	BLN3002I	Read dump tape failed
BLNONVSE	3008	BLN3002I	Undefined dump type
BLNONVSE	3016	BLN3002I	Read dump tape failed
BLNONVSE	3020	BLN3002I	Tape open error
BLNONVSE	3022	BLN3002I	Tape close error
BLNSMBLE	0925		Data not found
BLNSMBLE	7025	BLN9007I	Internal error
BLNSMBLF	7026		Storage allocation failure
BLNSMBLO	7062		Storage allocation failure
BLNSMBMA	7021		Storage allocation failure
BLNSMBME	7004	BLN9007I	Internal error
BLNSMCMO	7024		Storage allocation failure
BLNSMDSP	7023		Initialization error
BLNSMFND	7100	BLN9007I	Initialization error
BLNSMFND	7101		Data not available on find
BLNSMFND	7102		Invalid form of find command
BLNSMFRE	7094		Storage allocation failure
BLNSMGRE	7093		Storage allocation failure
BLNSMHXA	7076		Storage allocation failure
BLNSMHXA	7078		Storage allocation failure
BLNSMHXO	7080		Storage allocation failure
BLNSMINT	7002	BLN9007I	Double initialization
BLNSMLKC	7086		Storage allocation failure
BLNSMLKD	7090		Storage allocation failure
BLNSMLKL	7082		Storage allocation failure
BLNSMLKL	7083		Control block locator not found
BLNSMLKO	7088		Starting point not found
BLNSMLKO	7089		Storage allocation failure
BLNSMOCL	7095		Storage allocation failure
BLNSMPCD	7122	BLN9007I	Internal error freeing buffers
BLNSMPHP	7018		Storage allocation failure
BLNSMPRT	7084	BLN9007I	Initialization error
BLNSMPSO	7064		Storage allocation failure
BLNSMPSW	7092		Storage allocation failure
BLNSMSEL	7110	BLN9007I	Initialization error
BLNSMTRM	7003	BLN9007I	Double termination
BLNSMTXA	7066	BLN9007I	LBD not found in symptom record
BLNSMTXA	7068		Storage allocation failure
BLNSMTXA	7070		Storage allocation failure
BLNSMTXO	7072		Storage allocation failure

INFO/ANALYSIS REASON/MODULE CROSS-REFERENCE

In the following list, Info/Analysis reason codes are listed numerically. Each reason code is followed by name of the module that issues the reason code, the message id in which it may be inserted, and a description of the reason code. (Refer to "Info/Analysis Module/Reason Cross-Reference" on page 231 for additional information).

Reason Code	Module Issuing	Message ID	Description
0004	BLNADIFI		Invalid dump access entry point
0004	BLNAXGS2		Storage allocation failure
0004	BLNAXPD3		Output routed to alternate device
0004	BLNAXSU1		Symptom record has been modified
0008	BLNADIFI		Storage allocation failure
0008	BLNAXFM1		Storage length too small
0008	BLNAXGS2		Invalid length request
0012	BLNADIFI		Unable to get ADCB
0012	BLNAXGS3		Invalid free address passed
0012	BLNAXS6V		Section 6 updates invalid
0016	BLNAXDA1		Invalid request type
0016	BLNAXFM1		Storage allocation failure
0016	BLNAXPD1		No output device specified
0016	BLNAXSU1		Symptom record format invalid
0020	BLNAXPD1		Invalid buffer length
0020	BLNAXSU1		Dump name changed
0024	BLNAXSUA		Symptom record update error
0024	BLNAXS6U		Symptom record update error
0024	BLNAXS6V		Symptom record update error
0040	BLNAXDA1		Dump name changed
0092	BLNAXSUA		Storage allocation failure
0092	BLNAXSU1		Storage allocation failure
0092	BLNAXS3U		Storage allocation failure
0092	BLNAXS4U		Storage allocation failure
0092	BLNAXS5U		Storage allocation failure
0092	BLNAXS6U		Storage allocation failure
0096	BLNAXFM1		Unable to access ARCB
0096	BLNAXGS1		Unable to access ARCB
0096	BLNAXPD1		Unable to access ARCB
0096	BLNAXPD3		Unable to access ARCB
0096	BLNAXPD5		Unable to access ARCB
0096	BLNAXPD6		Unable to access ARCB
0096	BLNAXPD7		Unable to access ARCB
0096	BLNAXSU1		Unable to access ARCB
0100	BLNAXDAC		Invalid environment pointer
0100	BLNAXFMT		Invalid environment pointer

Reason Code	Module Issuing	Message ID	Description
0100	BLNAXGST		Invalid environment pointer
0100	BLNAXPDS		Invalid environment pointer
0100	BLNAXSAC		Invalid environment pointer
0100	BLNAXSA1		Invalid environment pointer
0100	BLNAXSUP		Invalid environment pointer
0900	BLNADIFR		Unable to get ADCB
0900	BLNADIFX		Unable to get ADCB
0904	BLNADIFX		Unable to delete ADCB from BLX
0920	BLNASLCB		Unable to get ASCB
0920	BLNASTRM		Unable to get ASCB
0921	BLNASFMT		Storage allocation failure
0921	BLNASGET		Storage allocation failure
0921	BLNASLCB		Storage allocation failure
0921	BLNASLD1		Storage allocation failure
0921	BLNASLST		Storage allocation failure
0921	BLNASMAP		Storage allocation failure
0921	BLNASMP1		Storage allocation failure
0923	BLNASADF		Format error in section 6
0923	BLNASADL		Format error in section 6
0923	BLNASAXA		Format error in section 6
0923	BLNASAXC		Format error in section 6
0923	BLNASAXK		Format error in section 6
0923	BLNASAXT		Format error in section 6
0923	BLNASAXX		Format error in section 6
0923	BLNASMD		Format error in section 6
0925	BLNASFSY		Data not found
0925	BLNASLD1		Data not found
0925	BLNASLD6		Data not found
0925	BLNASLST		Data not found
0925	BLNASMAP		Data not found
0925	BLNASRSR		Data not found
0925	BLNCMCHD		Data not found
0925	BLNDMAD1		Data not found
0925	BLNDMUDI		Data not found
0925	BLNSMBLE		Data not found
0926	BLNASLD1		Error in parameter list
0926	BLNASLOC		Error in parameter list
0926	BLNAXSA1		Error in parameter list
0927	BLNASMAP		Invalid dump name
0928	BLNASRSR		Dump access error
1101	BLNFNDMC	BLN9007I	Unable to access DMCB
1101	BLNFNPFC	BLN9007I	Unable to access DMCB
1101	BLNFNPNC	BLN9007I	Unable to access DMCB
1103	BLNFNDMC	BLN9007I	Unable to access DSCB

Reason Code	Module Issuing	Message ID	Description
1104	BLNMACBG	BLN9002I	Unable to access control blocks
2004	BLNDMRDF	BLN2013I	Dump management file open error
2004	BLNDMWDF	BLN2013I	Dump management file open error
2008	BLNDMRDF	BLN2013I	Storage allocation failure
2008	BLNDMUAI		Storage allocation failure
2008	BLNDMUDI		Storage allocation failure
2012	BLNDMRDF	BLN2013I	Dump management file close error
2012	BLNDMWDF	BLN2013I	Dump management file close error
2016	BLNDMWDF	BLN2013I	Dump management file write error
2020	BLNDMRDF	BLN2013I	Dump management file read error
2024	BLNDMRD1	BLN2013I	Invalid data record in dump management file
2028	BLNDMRDF	BLN2013I	Invalid control record in dump management file
2204	BLNDMDEL		Dump access failure
2208	BLNDMDEL		No dump delete verification
3008	BLNONMUD	BLN3002I	Undefined dump type
3008	BLNONVSE	BLN3002I	Undefined dump type
3012	BLNONLOD	BLN3002I	Error writing dump to system
3012	BLNONSC1	BLN3002I	Error writing dump to system
3012	BLNONTRW	BLN3002I	Error writing dump to system
3016	BLNONDDT	BLN3002I	Read dump tape failed
3016	BLNONEND	BLN3002I	Read dump tape failed
3016	BLNONMUD	BLN3002I	Read dump tape failed
3016	BLNONSCM	BLN3002I	Read dump tape failed
3016	BLNONSC1	BLN3002I	Read dump tape failed
3016	BLNONTRW	BLN3002I	Read dump tape failed
3016	BLNONVSE	BLN3002I	Read dump tape failed
3020	BLNONEND	BLN3002I	Tape open error
3020	BLNONINT	BLN3002I	Tape open error
3020	BLNONMUD	BLN3002I	Tape open error
3020	BLNONSCM	BLN3002I	Tape open error
3020	BLNONVSE	BLN3002I	Tape open error
3022	BLNONMUD	BLN3002I	Tape close error
3022	BLNONSCM	BLN3002I	Tape close error
3022	BLNONVSE	BLN3002I	Tape close error
3024	BLNONEND	BLN3002I	Tape allocate error
3024	BLNONINT	BLN3002I	Tape allocate error
3028	BLNONSCM	BLN3002I	Storage allocation failure
3032	BLNONINT	BLN3002I	Error reading dump management file
3036	BLNONEND	BLN3002I	Write dump management file error
3040	BLNONINT	BLN3002I	Error processing dump management file
3044	BLNONINT	BLN4007I	No VOLID specified
3048	BLNONINT	BLN3002I	Dump already in library
3052	BLNONINT	BLN3002I	Multi-dump not allowed
3056	BLNONINT	BLN3002I	Job submit failed

Reason Code	Module Issuing	Message ID	Description
3060	BLNONMUD	BLN3002I	No operator requested dump
4004	BLNOFFLD	BLN4005I	Error deleting dump from library
4008	BLNOFFLD	BLN4005I	Error adding VOLID to dump management file
4012	BLNOFOUT	BLN4005I	Error writing dump to tape
4016	BLNOFFLD	BLN4005I	Storage allocation failure
4020	BLNOFFLD	BLN4005I	No VOLID specified
4024	BLNOFFLD	BLN4005I	Dump not in dump management list
4028	BLNOFFLD	BLN4005I	Read of dump management list failed
4028	BLNOFUDM	BLN4005I	Read of dump management list failed
4030	BLNOFFLD	BLN4005I	Invalid bypass-no previous offload
4032	BLNOFFLD	BLN4005I	Invalid bypass-dump altered
4036	BLNOFFLD	BLN4005I	Dump not in library
4040	BLNOFFLD	BLN4005I	Job submit failed
4040	BLNOFOUT	BLN4005I	Tape allocate error
4044	BLNOFOUT	BLN4005I	Tape open error
4048	BLNOFOUT	BLN4005I	Tape close error
4052	BLNOFOUT	BLN4005I	Tape deallocate error
5001	BLNFNDDC	BLN9007I	Unknown subfunction request
5002	BLNDDINT		Storage allocation failure
5003	BLNDDINT		Storage allocation failure
5004	BLNDDSYI		Storage allocation failure
5006	BLNDDBAS		Invalid characters in BASE
5007	BLNDDBAS		Imbedded blanks in BASE
5008	BLNDDQAL		Invalid characters in QUAL
5009	BLNDDQAL		Imbedded blanks in QUAL
5010	BLNDDRDF		Locator not found
5011	BLNDDRDF		Invalid base in locator
5012	BLNDDRDF		Invalid length in locator
5013	BLNDDVRF		DISPLAY and BASE fields blank
5014	BLNDDFMT		Storage allocation failure
5015	BLNDDFDZ		Dump data not available
5016	BLNDDNBU		Storage allocation failure
5017	BLNDDPRT		Invalid print command
5018	BLNDDFND		Invalid find command
5019	BLNDDFPP		Storage allocation failure
5020	BLNDDFPP		Storage allocation failure
5021	BLNDDFPP	BLN9007I	Error freeing buffer chain
5022	BLNDDFPP		Error printing buffers
5023	BLNDDCDB		Storage allocation failure
5701	BLNFNDVS	BLN9007I	Unknown subfunction request
5901	BLNFNMSK	BLN9007I	Unknown subfunction request
5901	BLNDVKCP		Storage allocation failure
5902	BLNDVSIN	BLN9007I	Unable to access DVVT
5902	BLNMKINT	BLN9007I	Initialization error

Reason Code	Module Issuing	Message ID	Description
5903	BLNMKDSP	BLN9007I	Initialization error
5904	BLNMKBMD		Storage allocation failure
5905	BLNMKMSK	BLN9007I	Initialization error
5921	BLNDVSIA		Storage allocation failure
5922	BLNDVSIA		Storage allocation failure
5923	BLNDVSIA		Storage allocation failure
5924	BLNDVSIA		Storage allocation failure
5925	BLNDVSIA		Storage allocation failure
6000	BLNDSYMP	BLN9007I	Unknown subfunction request
6004	BLNDSDSP	BLN9007I	Damage to DSCB suspected
6004	BLNDSPRT	BLN9007I	Damage to DSCB suspected
6004	BLNDSTRM	BLN9007I	Damage to DSCB suspected
6008	BLNDSGBF		Storage allocation failure
6012	BLNDSPRT		Permanent I/O error on print
6016	BLNDSPRT		BLX error
6020	BLNDSCEN		No symptom record in dump
7002	BLNSMINT	BLN9007I	Double initialization
7003	BLNSMTRM	BLN9007I	Double termination
7004	BLNSMBME	BLN9007I	Internal error
7018	BLNSMPHP		Storage allocation failure
7021	BLNSMBMA		Storage allocation failure
7023	BLNSMDSP		Initialization error
7024	BLNSMCMO		Storage allocation failure
7025	BLNSMBLE	BLN9007I	Internal error
7026	BLNSMBLF		Storage allocation failure
7062	BLNSMBLO		Storage allocation failure
7064	BLNSMPSO		Storage allocation failure
7066	BLNSMTXA	BLN9007I	LBD not found in symptom record
7068	BLNSMTXA		Storage allocation failure
7070	BLNSMTXA		Storage allocation failure
7072	BLNSMTXO		Storage allocation failure
7076	BLNSMHXA		Storage allocation failure
7078	BLNSMHXA		Storage allocation failure
7080	BLNSMHXO		Storage allocation failure
7082	BLNSMLKL		Storage allocation failure
7083	BLNSMLKL		Control block locator not found
7084	BLNSMPRT	BLN9007I	Initialization error
7086	BLNSMLKC		Storage allocation failure
7088	BLNSMLKO		Starting point not found
7089	BLNSMLKO		Storage allocation failure
7090	BLNSMLKD		Storage allocation failure
7092	BLNSMPSW		Storage allocation failure
7093	BLNSMGRE		Storage allocation failure
7094	BLNSMFRE		Storage allocation failure

Reason Code	Module Issuing	Message ID	Description
7095	BLNSMOCL		Storage allocation failure
7099	BLNFNSMC	BLN9007I	Unknown subfunction request
7100	BLNSMFND	BLN9007I	Initialization error
7101	BLNSMFND		Data not available on find
7102	BLNSMFND		Invalid form of find command
7110	BLNSMSEL	BLN9007I	Initialization error
7122	BLNSMPCD	BLN9007I	Internal error freeing buffers
7401	BLNLBVER	BLN7013I	Unknown number of elements in array
7402	BLNLBVER	BLN7013I	Array extension type = B, no chain extension
7403	BLNLBVER	BLN7013I	Invalid array extension type field
7405	BLNLBSIM		Storage allocation failure
7406	BLNLBARC		Storage allocation failure
7407	BLNLBARC		Storage allocation failure
7408	BLNLBARR		Storage allocation failure
7411	BLNLBVER	BLN7013I	Invalid end condition field
7412	BLNLBVER	BLN7013I	Invalid end condition length field
7413	BLNLBVER	BLN7013I	Invalid chain extension type field
7414	BLNLBVER	BLN7013I	Invalid chain extension address list pointer
7421	BLNLBVER	BLN7013I	Invalid hex extension length
7431	BLNLBVER	BLN7013I	Invalid text extension length
7441	BLNLBVER	BLN7013I	Keyfield length is zero
7442	BLNLBVER	BLN7013I	Invalid keyfield format field
7443	BLNLBVER	BLN7013I	Invalid keyfield type field
7451	BLNLBVER	BLN7013I	Number of entries in format descriptor is zero
7451	BLNLBVER	BLN7013I	Number of entries in linkage descriptor is zero
7452	BLNLBVER	BLN7013I	Length error in format descriptor
7452	BLNLBVER	BLN7013I	Length error in linkage descriptor
7453	BLNLBVER	BLN7013I	Invalid offset attribute type
7454	BLNLBVER	BLN7013I	Invalid sequential offset format
7455	BLNLBVER	BLN7013I	Invalid specific offset format
7460	BLNLBLAL		Storage allocation failure
7464	BLNLBCAC		Storage allocation failure
7465	BLNLBCAE		Storage allocation failure
7466	BLNLBCAI		Storage allocation failure
7467	BLNLBCAL		Storage allocation failure
7468	BLNLBCAL		Storage allocation failure
7469	BLNLBCHL		Storage allocation failure
7470	BLNLBCHW		Storage allocation failure
7471	BLNLBCHI		Storage allocation failure
7472	BLNLBCHE		Storage allocation failure
7473	BLNLBCHC		Storage allocation failure
7474	BLNLBCHR		Storage allocation failure
7475	BLNLBCAE		Storage allocation failure
7476	BLNLBCAI		Storage allocation failure

Reason Code	Module Issuing	Message ID	Description
7477	BLNLBCAC		Storage allocation failure
7478	BLNLBCHE		Storage allocation failure
7479	BLNLBCHL		Storage allocation failure
7481	BLNLBCHI		Storage allocation failure
7483	BLNLBCHC		Storage allocation failure
7501	BLNFNARC	BLN9007I	Unknown subfunction request
7502	BLNARINT	BLN9007I	Duplicate initialization
7503	BLNARGRN		Storage allocation failure
7504	BLNARGRN		Storage allocation failure
7505	BLNARPRN		Storage allocation failure
7540	BLNARDSP		Initialization error
7541	BLNARDSP		No analysis routines list available
7544	BLNAREXA	BLN9007I	Unable to add ARCB to BLX
7550	BLNARDLO		Storage allocation failure
7560	BLNARPRT		Initialization not done
7567	BLNARFND		Initialization error
7568	BLNARFND		No analysis routines list available
7570	BLNAREXC		Initialization error
7581	BLNAREXG	BLN9002I	Unable to delete ARCB from BLX
7582	BLNAREXG	BLN7516I	Load of analysis routine failed
7583	BLNARXPL		Storage allocation failure
7592	BLNAREXA		Storage allocation failure
9501	BLNCMXDE		BLX allocation - invalid unit
9502	BLNCMXDE		BLX allocation - extract macro error
9503	BLNCMXDE		BLX allocation - device not supported
9504	BLNCMXDE		BLX allocation - RECFM not supported
9505	BLNCMXDE		BLX allocation - blocked records not supported
9506	BLNCMXDE		BLX allocation - access mode not supported
9507	BLNCMXDE		BLX allocation - DLBL statement not supported
9508	BLNCMXDE		BLX allocation - open VTOC not supported
9509	BLNCMXDE		BLX allocation - close VTOC not supported
9510	BLNCMXDE		BLX allocation - extent VOLSER invalid
9511	BLNCMXDE		BLX allocation - extent logical unit invalid
9512	BLNCMXDE		BLX allocation - DDNAME not found
9519	BLNCMXDE		BLX free error
9521	BLNCMXDE		BLX open - permanent open error
9522	BLNCMXDE		BLX open - invalid FSEQ keyword
9523	BLNCMXDE		BLX open - data set already open
9531	BLNCMXDE		BLX close - permanent close error
9532	BLNCMXDE		BLX close - end of extent
9541	BLNCMXDE		BLX read - update not allowed
9542	BLNCMXDE		BLX read - wrong access type
9543	BLNCMXDE		BLX read - buffer length too small
9544	BLNCMXDE		BLX read - invalid relative record number

Reason Code	Module Issuing	Message ID	Description
9545	BLNCMXDE		BLX read - VSAM position error
9546	BLNCMXDE		BLX read - invalid record length
9547	BLNCMXDE		BLX read - invalid key length
9548	BLNCMXDE		BLX read - VSAM MODCB verb failed
9549	BLNCMXDE		BLX read - undefined request
9591	BLNCMXDE		Unknown error condition

INFO/ANALYSIS CALLING/CALLED MODULE CROSS-REFERENCE

This section lists, alphabetically, the names of Info/Analysis calling modules (modules that call other modules). Following each calling module name are the names of the modules that it calls. In addition, the following information on calling modules may be useful:

- Modules used to interface with other components:
  - BLNADIFR
  - BLNASLOC
  - ISPLINK
- Modules used to control dumps in Dump Management file:
  - BLNUSADM
  - BLNUSDDM
- Modules used to execute an analysis routine:
  - BLNAXGST
  - BLNAXPDS
  - BLNAXDAC
  - BLNAXFMT
  - BLNAXSUP
  - BLNAXSAC
- Modules referenced by the BLNFS001 panel:
  - BLNSFDMG

  - BLNSFDSP
  - BLNSFDVS
  - BLNSFOFL
  - BLNSFONL
  - BLNSFPNH
  - BLNSFTRM
  - BLNSFNDN

- Modules referenced by the BLNRAS service:
  - BLN6ADCB
  - BLN6ARCB
  - BLN6ASRC
  - BLN6BACB
  - BLN6CCB
  - BLN6DDCB
  - BLN6DMCB
  - BLN6DSCB
  - BLN6DVVT
  - BLN6MKCB
  - BLN6SCLC
  - BLN6SFCB
  - BLN6SMCB

- Modules referenced by the BLX service:
  - BLNCONS
  - BLNPRTS
  - BLNCOPT

  - BLNTRLX
  - BLNSTOP

- ISPF modules referenced:
  - CDP
  - CTA
  - ISPLINK

- Data set allocation models referenced:
  - BLNMODEL
  - BLNLMODL

- Link books referenced for batch and interactive respectively:
  - BLNLINK
  - BLNLKIT

- Module referenced by BLNBARDR:
  - BLNBAHLP

- Mnemonic used to denote when dump access is invoked:
  - DUMPACC

- Mnemonic used to denote when an analysis routine is invoked:
  - EXITRTN

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
BLNADIFI		BLNAREXA	BLNDVFPE	BLNAREXE	EXITRTN
BLNADIFR	BLNADIFT	BLNAREXC	BLNAREXD	BLNAREXF	BLNAREXG
BLNADIFR	DUMPACC	BLNAREXC	BLNAREXE	BLNAREXF	BLNCMERM
BLNADIFT	DUMPACC	BLNAREXD	BLNCMPRS	BLNAREXG	
BLNADIFX	BLNADIFT	BLNAREXE	BLNAREXA	BLNAREXL	
BLNARDLO		BLNAREXE	BLNAREXF	BLNARFCS	BLNARDSP
BLNARDSP	BLNARDLO	BLNAREXE	BLNAREXL	BLNARFCS	BLNCMFBC
BLNARDSP	BLNCMALN	BLNAREXE	BLNARXPL	BLNARFML	BLNARFML

Calling Module	Called Module
BLNARFND	BLNARDSP
BLNARFND	BLNARFCS
BLNARFND	BLNCMPRS
BLNARGDN	BLNARFML
BLNARGDN	BLNARPRN
BLNARGDN	BLNCMXDE
BLNARGDN	BLNMSGS
BLNARGRN	BLNARFML
BLNARGRN	BLNARGDN
BLNARGRN	BLNMSGS
BLNARINT	BLNARGRN
BLNARINT	BLNMSGS
BLNARPDI	
BLNARPRN	BLNARPDI
BLNARPRN	BLNMSGS
BLNARPRT	BLNARDLO
BLNARPRT	BLNCMALN
BLNARPRT	BLNCMFBC
BLNARPRT	BLNCMPRS
BLNARPRT	BLNCMPRT
BLNARPRT	BLNSMPLA
BLNARTRM	BLNARFML
BLNARXPL	BLNASLOC
BLNARXPL	BLNAXGS2
BLNARXPL	BLNCMERM
BLNASADF	BLNERCMB
BLNASADL	BLNERCMB
BLNASAXA	BLNERCMB
BLNASAXC	BLNERCMB
BLNASAXK	BLNERCMB
BLNASAXT	BLNERCMB
BLNASAXX	BLNASMP1
BLNASAXX	BLNERCMB
BLNASCME	BLNASGET
BLNASCME	BLNASIST
BLNASFMT	BLNASRFL
BLNASFMT	BLNASRSR
BLNASFMT	BLNERCMB
BLNASFSY	
BLNASGET	BLNERCMB
BLNASIST	BLNASGET
BLNASLCB	BLNERCMB
BLNASLD1	BLNASFSY
BLNASLD1	BLNERCMB
BLNASLD6	BLNASFMT
BLNASLOC	BLNASLCB
BLNASLOC	BLNASLD1
BLNASLOC	BLNASLD6
BLNASLOC	BLNASMAP
BLNASLOC	BLNERCMB
BLNASLST	BLNASLCB
BLNASLST	BLNASMAP
BLNASLST	BLNERCMB
BLNASMAP	BLNASMP1
BLNASMAP	BLNASMP6
BLNASMAP	BLNASRSR
BLNASMAP	BLNASTMP
BLNASMAP	BLNERCMB
BLNASMD	BLNASADF
BLNASMD	BLNASADL
BLNASMD	BLNASAXA
BLNASMD	BLNASAXC
BLNASMD	BLNASAXK
BLNASMD	BLNASAXT
BLNASMD	BLNASAXX
BLNASMD	BLNASCME
BLNASMD	BLNERCMB
BLNASMDA	BLNASCME
BLNASMP1	BLNERCMB
BLNASMP6	BLNASMD
BLNASMP6	BLNASMDA
BLNASRFL	
BLNASRSR	BLNADIFR
BLNASRSR	BLNERCMB
BLNASTMP	
BLNASTRM	BLNASLCB
BLNASTRM	BLNASTMP
BLNASTRM	BLNERCMB
BLNASUPD	BLNASLCB
BLNASUPD	BLNASUP6
BLNASUP6	BLNASMP6
BLNASUP6	BLNASRSR
BLNAXCMB	BLNAXMSG
BLNAXDAC	BLNAXDA1
BLNAXDA1	BLNADIFR
BLNAXFMT	BLNAXFM1
BLNAXFM1	BLNAXGS2
BLNAXFM1	BLNAXMSG
BLNAXFM1	BLNCMFBC
BLNAXFM1	BLNDDFMT
BLNAXFM1	BLNDDLSZ
BLNAXGST	BLNAXGS1
BLNAXGS1	BLNAXGS2
BLNAXGS1	BLNAXGS3
BLNAXGS2	
BLNAXGS3	
BLNAXMSG	BLNAXGS2
BLNAXPDS	BLNAXPD1
BLNAXPD1	BLNAXGS3
BLNAXPD1	BLNAXPD2
BLNAXPD1	BLNAXPD3
BLNAXPD2	BLNAXCMB
BLNAXPD2	BLNCMPRT
BLNAXPD3	BLNAXCMB
BLNAXPD3	BLNAXPD2
BLNAXPD3	BLNAXPD4
BLNAXPD3	BLNAXPD7
BLNAXPD4	BLNAXPD4
BLNAXPD4	BLNAXPD5
BLNAXPD4	BLNAXPD6
BLNAXPD5	BLNAXPD6
BLNAXPD6	BLNAXCMB
BLNAXPD6	BLNCMPRT
BLNAXPD7	
BLNAXSAC	BLNAXSA1
BLNAXSA1	BLNASLOC
BLNAXSA1	BLNAXGS2
BLNAXSRV	
BLNAXSUA	BLNADIFR
BLNAXSUA	BLNASUPD
BLNAXSUA	BLNAXSUA
BLNAXSUP	BLNAXSU1
BLNAXSUR	BLNAXS3U
BLNAXSUR	BLNAXS4U
BLNAXSUR	BLNAXS5U
BLNAXSU1	BLNAXCMB
BLNAXSU1	BLNAXSRV
BLNAXSU1	BLNAXSUA
BLNAXSU1	BLNAXSUR
BLNAXSU1	BLNAXS6U
BLNAXS3U	BLNAXGS2
BLNAXS3U	BLNAXGS3

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
BLNAXS3U	BLNAXS4U	BLNBAPF1	BLNFNPFC	BLNDDBAS	BLNDDSYM
BLNAXS4U	BLNAXGS2	BLNBAPN1	BLNBAMSG	BLNDDBOD	BLNCMXPF
BLNAXS4U	BLNAXGS3	BLNBAPN1	BLNBARDR	BLNDDBTE	BLNCMBUT
BLNAXS5U	BLNAXGS2	BLNBAPN1	BLNCMFIL	BLNDDBTE	BLNDDNLI
BLNAXS5U	BLNAXGS3	BLNBAPN1	BLNCMVOL	BLNDDCDB	BLNDDFMT
BLNAXS6U	BLNASUPD	BLNBAPN1	BLNFNPNC	BLNDDCDB	BLNDDLSZ
BLNAXS6U	BLNAXCMB	BLNBAPRS		BLNDDCDB	BLNDDNLI
BLNAXS6U	BLNAXS6V	BLNBARDR	BLNBAMSG	BLNDDCDB	BLNDDREQ
BLNAXS6V	BLNADIFR	BLNBARDR	BLNBAPRS	BLNDDCDB	BLNDDVRF
BLNBABH1	BLNBABH2	BLNBARDR	BLNCMLST	BLNDDDPT	BLNCMPRT
BLNBABH1	BLNBADM1	BLNBAVSE		BLNDDDPT	BLNDDCDB
BLNBABH1	BLNBADM2	BLNCMALN		BLNDDDPT	BLNDDFPP
BLNBABH1	BLNBADS1	BLNCMBUT		BLNDDDPT	BLNDDRPT
BLNBABH1	BLNBADV1	BLNCMCHD	BLNASLOC	BLNDDDSP	BLNDDCDB
BLNBABH1	BLNBAFLS	BLNCMCHD	BLNCMPRI	BLNDDEBC	BLNCMXEB
BLNBABH1	BLNBAPF1	BLNCMCHD	BLNCMTDS	BLNDDFDO	BLNCMXPF
BLNBABH1	BLNBAPN1	BLNCMDEC		BLNDDFDO	BLNDDFDZ
BLNBABH1	BLNBARDR	BLNCMERM	BLNMSGS	BLNDDFDO	BLNDDNLI
BLNBABH2	BLNBAMSG	BLNCMFBC		BLNDDFDO	BLNDVFFR
BLNBABH2	BLNBARDR	BLNCMFIL	BLNCMDEC	BLNDDFDZ	BLNCMMOB
BLNBADM1	BLNBABH2	BLNCMHEX		BLNDDFDZ	BLNDDNLI
BLNBADM1	BLNBADM2	BLNCMLST		BLNDDFDZ	BLNDDREQ
BLNBADM1	BLNBARDR	BLNCMMOB		BLNDDFFN	BLNCMFBC
BLNBADM1	BLNFNDMC	BLNCMMSP		BLNDDFMT	BLNASLOC
BLNBADM2		BLNCMPRI		BLNDDFMT	BLNCMERM
BLNBADS1	BLNBABH2	BLNCMPRO	BLNINPRT	BLNDDFMT	BLNCMFBC
BLNBADS1	BLNBARDR	BLNCMPRS		BLNDDFMT	BLNDDBTE
BLNBADS1	BLNFNDSC	BLNCMPRT	BLNCMPRO	BLNDDFMT	BLNDDFDO
BLNBADV1	BLNBABH2	BLNCMPRT	BLNINPRT	BLNDDFMT	BLNDDHDO
BLNBADV1	BLNBADV2	BLNCMTDS		BLNDDFMT	BLNLBVER
BLNBADV1	BLNBADV3	BLNCMTSZ		BLNDDFND	BLNCMFBC
BLNBADV1	BLNBARDR	BLNCMVOL		BLNDDFND	BLNDDCDB
BLNBADV1	BLNFNDDC	BLNCMXBI		BLNDDFND	BLNDDFFN
BLNBADV2	BLNFNDVS	BLNCMXCF		BLNDDFND	BLNDDHFN
BLNBADV3	BLNFNARC	BLNCMXDC		BLNDDFND	BLNDDLSZ
BLNBAFLS	BLNBAMSG	BLNCMXDE		BLNDDFND	BLNDDVFN
BLNBAFLS	BLNBARDR	BLNCMXEB		BLNDDFPD	BLNDDCDB
BLNBAINT	BLNBABH1	BLNCMXPF		BLNDDFPD	BLNDDFPP
BLNBAINT	BLNBAIPF	BLNCONS		BLNDDFPD	BLNFNSMC
BLNBAINT	BLNBAMSG	BLNCOPT	BLNCONS	BLNDDFPL	BLNDDFPD
BLNBAIPF	BLNBAVSE	BLNCOPT	BLNPRTS	BLNDDFPP	BLNCMFBC
BLNBAMSG		BLNDDBAS	BLNCMHEX	BLNDDFPP	BLNCMMOB
BLNBAPF1	BLNBARDR	BLNDDBAS	BLNCMPRS	BLNDDFPP	BLNCMPRT
BLNBAPF1	BLNCMVOL	BLNDDBAS	BLNCMXPF	BLNDDFPS	BLNDDFPL

Calling Module	Called Module
BLNDDFPS	BLNFNSMC
BLNDDFPS	BLNSMPCD
BLNDDFPT	BLNDDFPS
BLNDDFPT	BLNDVKCB
BLNDDFPT	BLNDVRCB
BLNDDFPT	BLNSMPCD
BLNDDHBL	BLNCMXPF
BLNDDHBL	BLNDDBOD
BLNDDHBL	BLNDDEBC
BLNDDHBL	BLNDDMDT
BLNDDHDO	BLNCMMOB
BLNDDHDO	BLNCMXPF
BLNDDHDO	BLNDDHBL
BLNDDHDO	BLNDDNBU
BLNDDHDO	BLNDDNLI
BLNDDHFN	BLNCMHEX
BLNDDHFN	BLNCMXPF
BLNDDHFN	BLNDDCDB
BLNDDHFN	BLNDDQRY
BLNDDHFN	BLNDDREQ
BLNDDINT	BLNDDSYI
BLNDDLSZ	
BLNDDMDT	BLNCMMOB
BLNDDMDT	BLNCMXPF
BLNDDMDT	BLNDDQRY
BLNDDMDT	BLNDDREQ
BLNDDNBU	
BLNDDNLI	BLNDDNBU
BLNDDPOP	BLNADIFR
BLNDDPOP	BLNCMHEX
BLNDDPOP	BLNCMPRS
BLNDDPRT	BLNDDDPT
BLNDDPRT	BLNDDFPT
BLNDDPRT	BLNDDPOP
BLNDDPRT	BLNDDRPT
BLNDDQAL	BLNCMHEX
BLNDDQAL	BLNCMPRS
BLNDDQAL	BLNCMXPF
BLNDDQRY	BLNADIFR
BLNDDQRY	BLNCMERM
BLNDDRDF	BLNASLOC
BLNDDRDF	BLNCMERM
BLNDDRDF	BLNCMXPF
BLNDDRDF	BLNLBVZA
BLNDDREQ	BLNADIFR
BLNDDREQ	BLNCMERM
BLNDDRPT	BLNCMFBC
BLNDDRPT	BLNCMMOB
BLNDDRPT	BLNCMPRT
BLNDDRPT	BLNCMXPF
BLNDDRPT	BLNDDFMT
BLNDDRPT	BLNDDLSZ
BLNDDRPT	BLNDDQRY
BLNDDRPT	BLNDDREQ
BLNDDSYI	BLNASLOC
BLNDDSYI	BLNCMERM
BLNDDSYI	BLNLBVER
BLNDDSYM	
BLNDDTRM	
BLNDDVFA	BLNCMHEX
BLNDDVFN	BLNCMHEX
BLNDDVFN	BLNCMPRS
BLNDDVFN	BLNDDVFA
BLNDDVRF	BLNCMXPF
BLNDDVRF	BLNDDBAS
BLNDDVRF	BLNDDQAL
BLNDDVRF	BLNDDRDF
BLNDMADU	BLNDMAD1
BLNDMADU	BLNDMCHK
BLNDMADU	BLNDMCNT
BLNDMADU	BLNDMRDF
BLNDMADU	BLNDMSRT
BLNDMADU	BLNDMWDF
BLNDMAD1	BLNASLOC
BLNDMAD1	BLNCMERM
BLNDMAD1	BLNCMTDS
BLNDMCHK	BLNDMCK1
BLNDMCHK	BLNDMCK2
BLNDMCK1	BLNDMCK2
BLNDMCK1	BLNDMCK3
BLNDMCK2	BLNADIFR
BLNDMCK2	BLNCMERM
BLNDMCK2	BLNDMAD1
BLNDMCK2	BLNDMWDF
BLNDMCK3	
BLNDMCNT	
BLNDMDDL	BLNDMCHK
BLNDMDDL	BLNDMRDF
BLNDMDDL	BLNDMSRT
BLNDMDEL	BLNADIFR
BLNDMDEL	BLNASTRM
BLNDMDEL	BLNCMERM
BLNDMDEL	BLNDMCHK
BLNDMDEL	BLNDMCNT
BLNDMDEL	BLNDMRDF
BLNDMDEL	BLNDMUDI
BLNDMDEL	BLNDMWDF
BLNDMDMU	BLNDMRDF
BLNDMDMU	BLNDMTRM
BLNDMDMU	BLNDMWDF
BLNDMGNT	BLNDMADU
BLNDMGNT	BLNDMDDL
BLNDMGNT	BLNDMDEL
BLNDMGNT	BLNDMDMU
BLNDMGNT	BLNDMPDL
BLNDMGNT	BLNDMTRM
BLNDMGNT	BLNDMUAI
BLNDMGNT	BLNOFFLD
BLNDMGNT	BLNONLOD
BLNDMPDL	BLNCMFBC
BLNDMPDL	BLNCMPRT
BLNDMPDL	BLNDMCHK
BLNDMPDL	BLNDMRDF
BLNDMPDL	BLNDMSRT
BLNDMRDF	BLNDMRD1
BLNDMRD1	
BLNDMSRT	
BLNDMTRM	
BLNDMUAB	BLNDMUAD
BLNDMUAD	BLNDMADU
BLNDMUAI	BLNDMRDF
BLNDMUAI	BLNUSADM
BLNDMUDI	BLNASLOC
BLNDMUDI	BLNCMERM
BLNDMUDI	BLNUSDDM
BLNDMWDF	
BLNDSBCL	BLNDSGBF
BLNDSBSO	BLNDSBCL
BLNDSBSO	BLNDSGBF
BLNDSCEN	BLNASLOC
BLNDSCEN	BLNCMERM
BLNDSCEN	BLNDSFED

Calling Module	Called Module
BLNDSCEN	BLNDSGBF
BLNDSCEN	BLNMSGS
BLNDSCNS	BLNASLOC
BLNDSCNS	BLNCMERM
BLNDSCNS	BLNDSBSO
BLNDSCNS	BLNDSGBF
BLNDSCOL	BLNDSCEN
BLNDSCOL	BLNDSCNS
BLNDSCOL	BLNDSCRQ
BLNDSCOL	BLNDSCST
BLNDSCRQ	BLNASLOC
BLNDSCRQ	BLNCMERM
BLNDSCRQ	BLNDSBSO
BLNDSCRQ	BLNDSGBF
BLNDSCST	BLNASLOC
BLNDSCST	BLNCMERM
BLNDSCST	BLNDSBSO
BLNDSCST	BLNDSGBF
BLNDSDSP	BLNCMFBC
BLNDSDSP	BLNDSCOL
BLNDSDSP	BLNMSGS
BLNDSFED	BLNCMTDS
BLNDSFED	BLNDSGBF
BLNDSGBF	BLNMSGS
BLNDSPRT	BLNCMFBC
BLNDSPRT	BLNCMPRT
BLNDSPRT	BLNDSCOL
BLNDSPRT	BLNDSGBF
BLNDSPRT	BLNMSGS
BLNDSTRM	BLNCMFBC
BLNDSTRM	BLNMSGS
BLNDSYMP	BLNDSDSP
BLNDSYMP	BLNDSPRT
BLNDSYMP	BLNDSTRM
BLNDSYMP	BLNMSGS
BLNDVFFR	BLNCMXBI
BLNDVFFR	BLNCMXDC
BLNDVFFR	BLNCMXEB
BLNDVFFR	BLNCMXPF
BLNDVFPE	
BLNDVKCB	BLNFNSMC
BLNDVRCB	BLNFNSMC
BLNDVRCB	BLNSMFES
BLNDVSIA	
BLNDVSIN	BLNDVFPE
BLNDVSIN	BLNDVSIA
BLNDVSIN	BLNDVSTM
BLNDVSIN	BLNFNDDO
BLNDVSTM	
BLNERCMB	
BLNFNARC	BLNARDSP
BLNFNARC	BLNAREXC
BLNFNARC	BLNARFND
BLNFNARC	BLNARINT
BLNFNARC	BLNARPRT
BLNFNARC	BLNARTRM
BLNFNARC	BLNMSGS
BLNFNDC2	
BLNFNDC3	
BLNFNDDC	BLNDDDSP
BLNFNDDC	BLNDDFND
BLNFNDDC	BLNDDINT
BLNFNDDC	BLNDDPRT
BLNFNDDC	BLNDDTRM
BLNFNDDO	
BLNFNDMC	BLNCMCHD
BLNFNDMC	BLNDMGNT
BLNFNDMC	BLNFNDC2
BLNFNDSC	BLNDSYMP
BLNFNDSC	BLNFNDC3
BLNFNDVS	BLNDVSIN
BLNFNDVS	BLNDVSTM
BLNFNMSK	BLNMKDSP
BLNFNMSK	BLNMKINT
BLNFNMSK	BLNMKMSK
BLNFNMSK	BLNMKTRM
BLNFNPC1	
BLNFNPC2	
BLNFNPFC	BLNDMGNT
BLNFNPFC	BLNFNPC2
BLNFNPNC	BLNDMGNT
BLNFNPNC	BLNFNPC1
BLNFNSMC	BLNSMDSP
BLNFNSMC	BLNSMFND
BLNFNSMC	BLNSMINT
BLNFNSMC	BLNSMPRT
BLNFNSMC	BLNSMSEL
BLNFNSMC	BLNSMTRM
BLNINIT	BLNMAIN
BLNINPRT	CDP
BLNINPRT	CTA
BLNITSPF	BLNMAIN
BLNLBARB	BLNLBARC
BLNLBARB	BLNLBCHN
BLNLBARB	BLNLBSIM
BLNLBARC	BLNLBBCI
BLNLBARR	BLNLBCHR
BLNLBARR	BLNLBCNT
BLNLBARY	BLNLBARB
BLNLBARY	BLNLBARC
BLNLBBCI	BLNADIFR
BLNLBBCI	BLNLBCEF
BLNLBBCI	BLNLBCMA
BLNLBBCI	BLNLBRCB
BLNLBBCI	BLNSMLKC
BLNLBBLR	BLNLBARR
BLNLBBLR	BLNLBARY
BLNLBBLR	BLNLBCHN
BLNLBBLR	BLNLBCHR
BLNLBBLR	BLNLBSIM
BLNLBCAC	BLNLBBCI
BLNLBCAC	BLNLBCNT
BLNLBCAC	BLNLBLAL
BLNLBCAE	BLNLBBCI
BLNLBCAE	BLNLBCEF
BLNLBCAE	BLNLBLAL
BLNLBCAI	BLNLBBCI
BLNLBCAI	BLNLBLAL
BLNLBCAL	BLNLBBCI
BLNLBCAL	BLNLBCEF
BLNLBCAL	BLNLBLAL
BLNLBCEF	
BLNLBCHC	BLNLBBCI
BLNLBCHC	BLNLBCLN
BLNLBCHC	BLNLBCNT
BLNLBCHE	BLNLBBCI
BLNLBCHE	BLNLBCEF
BLNLBCHE	BLNLBCLN
BLNLBCHI	BLNLBBCI
BLNLBCHI	BLNLBCLN
BLNLBCHL	BLNLBBCI
BLNLBCHL	BLNLBCEF

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
BLNLBCHL	BLNLBCLN	BLNMKDSP	BLNMKVMR	BLNONTRW	BLNADIFR
BLNLBCHN	BLNLBCAC	BLNMKINT		BLNONTRW	BLNCMERM
BLNLBCHN	BLNLBCAE	BLNMKMSK	BLNADIFR	BLNONVSE	
BLNLBCHN	BLNLBCAI	BLNMKMSK	BLNCMERM	BLNPRTS	BLNCMLST
BLNLBCHN	BLNLBCAL	BLNMKQRY	BLNDDREQ	BLNPRTS	BLNCMMSP
BLNLBCHN	BLNLBCHC	BLNMKSMB	BLNDDLSZ	BLNRAADB	BLNRALBD
BLNLBCHN	BLNLBCHE	BLNMKTRM		BLNRAADB	BLNRAS6S
BLNLBCHN	BLNLBCHI	BLNMKVMR	BLNCMHEX	BLNRAARB	BLNRALBD
BLNLBCHN	BLNLBCHL	BLNMKVMR	BLNCMPRS	BLNRAARB	BLNRAS6S
BLNLBCHN	BLNLBCHW	BLNMKVMR	BLNCMXPF	BLNRAASB	BLNRALBD
BLNLBCHR	BLNADIFR	BLNMSGS		BLNRAASB	BLNRAS6S
BLNLBCHR	BLNCMERM	BLNOFFLD	BLNADIFR	BLNRABAB	BLNRALBD
BLNLBCHW	BLNLBBCI	BLNOFFLD	BLNCMERM	BLNRABAB	BLNRAS6S
BLNLBCHW	BLNLBCLN	BLNOFFLD	BLNDMCHK	BLNRACCB	BLNRALBD
BLNLBCLN	BLNADIFR	BLNOFFLD	BLNDMRDF	BLNRACCB	BLNRAS6S
BLNLBCLN	BLNCMERM	BLNOFFLD	BLNOFOUT	BLNRADDB	BLNRALBD
BLNLBCMA		BLNOFFLD	BLNOFUDM	BLNRADDB	BLNRAS6S
BLNLBCNT	BLNADIFR	BLNOFFLD	BLNSUBMT	BLNRADMB	BLNRALBD
BLNLBCNT	BLNCMERM	BLNOFOUT	BLNADIFR	BLNRADMB	BLNRAS6S
BLNLBLAL	BLNADIFR	BLNOFOUT	BLNCMERM	BLNRADSB	BLNRALBD
BLNLBLAL	BLNCMERM	BLNOFUDM	BLNDMCHK	BLNRADSB	BLNRAS6S
BLNLBRCB	BLNADIFR	BLNOFUDM	BLNDMRDF	BLNRADVT	BLNRALBD
BLNLBRCB	BLNCMERM	BLNOFUDM	BLNDMWDF	BLNRADVT	BLNRAS6S
BLNLBSIM	BLNLBBCI	BLNONDDT	BLNONVSE	BLNRAEIN	
BLNLBVER		BLNONEND	BLNADIFR	BLNRALBD	
BLNLBVZA		BLNONEND	BLNCMERM	BLNRAMKB	BLNRALBD
BLNMACBC	BLNMACBG	BLNONEND	BLNDMAD1	BLNRAMKB	BLNRAS6S
BLNMACBC	BLNMAPDF	BLNONEND	BLNDMCHK	BLNRAS	BLNRASEC
BLNMACBG		BLNONEND	BLNDMRDF	BLNRASCB	BLNRALBD
BLNMAIN	BLNADIFI	BLNONEND	BLNDMSRT	BLNRASCB	BLNRAS6S
BLNMAIN	BLNBAINT	BLNONEND	BLNDMWDF	BLNRASEC	BLNRAEIN
BLNMAIN	BLNMACBC	BLNONINT	BLNDMCHK	BLNRASEC	BLNRAS4B
BLNMAIN	BLNMAOCM	BLNONINT	BLNDMRDF	BLNRASEC	BLNRAS5B
BLNMAIN	BLNRAS	BLNONINT	BLNSUBMT	BLNRASEC	BLNRAS6B
BLNMAIN	BLNSFINT	BLNONLOD	BLNONDDT	BLNRASFB	BLNRALBD
BLNMAIN	BLNTERM	BLNONLOD	BLNONEND	BLNRASFB	BLNRAS6S
BLNMAOCM		BLNONLOD	BLNONINT	BLNRASMB	BLNRALBD
BLNMAPDF		BLNONLOD	BLNONMUD	BLNRASMB	BLNRAS6S
BLNMKBMD	BLNCMMOB	BLNONLOD	BLNONSCM	BLNRAS4B	
BLNMKBMD	BLNDDFMT	BLNONLOD	BLNONTRW	BLNRAS5B	
BLNMKBMD	BLNDDREQ	BLNONMUD		BLNRAS6B	BLNRAADB
BLNMKDSP	BLNMKBMD	BLNONSCM	BLNONSC1	BLNRAS6B	BLNRAARB
BLNMKDSP	BLNMKQRY	BLNONSC1	BLNADIFR	BLNRAS6B	BLNRAASB
BLNMKDSP	BLNMKSMB	BLNONSC1	BLNCMERM	BLNRAS6B	BLNRABAB

Calling Module	Called Module
BLNRAS6B	BLNRACCB
BLNRAS6B	BLNRADDB
BLNRAS6B	BLNRADMB
BLNRAS6B	BLNRADSB
BLNRAS6B	BLNRADVT
BLNRAS6B	BLNRAMKB
BLNRAS6B	BLNRASCB
BLNRAS6B	BLNRASFB
BLNRAS6B	BLNRASMB
BLNRAS6S	
BLNSCBKW	BLNCMXDC
BLNSCBKW	BLNSCCRP
BLNSCBKW	BLNSCCTS
BLNSCBKW	BLNSCSUB
BLNSCBKW	BLNSCTOK
BLNSCBKW	ISPLINK
BLNSCCCP	BLNSCCUP
BLNSCCCP	ISPLINK
BLNSCCRP	ISPLINK
BLNSCCSC	ISPLINK
BLNSCCTS	BLNCMXDC
BLNSCCTS	ISPLINK
BLNSCCUP	
BLNSCDIS	BLNSCCSC
BLNSCDIS	BLNSCCTS
BLNSCDIS	BLNSCDRC
BLNSCDIS	BLNSCERK
BLNSCDIS	BLNSCHDL
BLNSCDIS	BLNSCOTH
BLNSCDIS	ISPLINK
BLNSCDRC	BLNSCTOK
BLNSCDRC	ISPLINK
BLNSCDTB	BLNSCDIS
BLNSCDTB	BLNSCPRD
BLNSCDTB	BLNSCRDT
BLNSCERK	
BLNSCFWD	BLNCMXDC
BLNSCFWD	BLNSCCRP
BLNSCFWD	BLNSCCTS
BLNSCFWD	BLNSCSUB
BLNSCFWD	BLNSCTOK
BLNSCFWD	ISPLINK
BLNSCHDL	BLNSCBKW
BLNSCHDL	BLNSCCUP
BLNSCHDL	BLNSCFWD
BLNSCHDL	BLNSCTOK
BLNSCHDL	ISPLINK
BLNSCLFT	BLNSCCTS
BLNSCLFT	BLNSCSUB
BLNSCLFT	BLNSCTOK
BLNSCLFT	ISPLINK
BLNSCOTH	BLNSCCCP
BLNSCOTH	BLNSCCSC
BLNSCOTH	ISPLINK
BLNSCPRD	BLNSCSUD
BLNSCPRD	ISPLINK
BLNSCRDT	BLNSCCRP
BLNSCRDT	BLNSCSUB
BLNSCRGH	BLNSCCTS
BLNSCRGH	BLNSCSUB
BLNSCRGH	BLNSCTOK
BLNSCRGH	ISPLINK
BLNSCSUB	ISPLINK
BLNSCSUD	BLNCMTSZ
BLNSCTOK	BLNCMXCF
BLNSFAMS	BLNSFBUM
BLNSFAMS	ISPLINK
BLNSFASM	BLNFNSMC
BLNSFASM	BLNSFAS1
BLNSFASM	BLNSFDIS
BLNSFASM	BLNSFDST
BLNSFASM	BLNSFMDA
BLNSFASM	ISPLINK
BLNSFAS1	BLNFNSMC
BLNSFAS1	BLNSFDAT
BLNSFAS1	BLNSFDIS
BLNSFAS1	BLNSFFRZ
BLNSFAS1	BLNSFLGD
BLNSFAS1	BLNSFLNK
BLNSFAS1	BLNSFLOC
BLNSFAS1	BLNSFRST
BLNSFAS1	ISPLINK
BLNSFBST	BLNCMFBC
BLNSFBST	ISPLINK
BLNSFBUM	BLNCMBUT
BLNSFBUM	ISPLINK
BLNSFCAR	BLNFNARC
BLNSFCAR	BLNSFCA1
BLNSFCAR	BLNSFDIS
BLNSFCAR	BLNSFDST
BLNSFCAR	BLNSFMDB
BLNSFCAR	ISPLINK
BLNSFCA1	BLNFNARC
BLNSFCA1	BLNSFDIS
BLNSFCA1	BLNSFLGD
BLNSFCA1	ISPLINK
BLNSFCDS	BLNSFCER
BLNSFCDS	BLNSFCSR
BLNSFCDS	BLNSFDMS
BLNSFCDS	BLNSFHLP
BLNSFCDS	BLNSFISP
BLNSFCDS	BLNSFLGD
BLNSFCDS	BLNSFLOG
BLNSFCDS	BLNSFPRS
BLNSFCDS	BLNSFPRT
BLNSFCDS	ISPLINK
BLNSFCER	BLNSFIMP
BLNSFCER	BLNSFSCM
BLNSFCMQ	BLNCMBUT
BLNSFCMQ	BLNSFLGD
BLNSFCSR	BLNSCDTB
BLNSFCSR	ISPLINK
BLNSFDAT	BLNFNSMC
BLNSFDAT	BLNSFDIS
BLNSFDAT	BLNSFDST
BLNSFDAT	BLNSFMDA
BLNSFDAT	ISPLINK
BLNSFDD2	BLNFNDDC
BLNSFDD2	BLNSFDD3
BLNSFDD3	BLNCMHEX
BLNSFDD3	BLNCMXPF
BLNSFDD3	ISPLINK
BLNSFDID	BLNSFDIS
BLNSFDID	BLNSFDMG
BLNSFDID	BLNSFDSP
BLNSFDID	BLNSFDVS
BLNSFDID	BLNSFFST
BLNSFDID	BLNSFLGD
BLNSFDID	BLNSFNDN
BLNSFDID	BLNSFOFL
BLNSFDID	BLNSFONL
BLNSFDID	BLNSFPNH

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
BLNSFDIS	BLNSFBST	BLNSFFIN	ISPLINK	BLNSFMHH	BLNSFDIS
BLNSFDIS	BLNSFCDS	BLNSFFRZ	BLNSFSCM	BLNSFMHH	BLNSFFRZ
BLNSFDMB	BLNSFDIS	BLNSFFST	ISPLINK	BLNSFMHH	BLNSFRST
BLNSFDMB	BLNSFDMD	BLNSFHLP	BLNSFMSH	BLNSFMSH	BLNSFDIS
BLNSFDMB	BLNSFLGD	BLNSFHLP	BLNSFPNH	BLNSFMSH	BLNSFFRZ
BLNSFDMB	BLNSFNDN	BLNSFICB	BLNSFVDF	BLNSFMSH	BLNSFIPT
BLNSFDMB	ISPLINK	BLNSFICB	ISPLINK	BLNSFMSH	BLNSFMHH
BLNSFDMD	BLNFNDMC	BLNSFIMP	BLNSFAMS	BLNSFMSH	BLNSFRST
BLNSFDMD	BLNSFDIS	BLNSFIMP	BLNSFCMQ	BLNSFMSK	BLNCMFBC
BLNSFDMD	BLNSFLGD	BLNSFINT	BLNCMTSZ	BLNSFMSK	BLNFNMSK
BLNSFDMG	BLNFNDMC	BLNSFINT	BLNSFDID	BLNSFMSK	BLNSFDIS
BLNSFDMG	BLNSFDIS	BLNSFINT	BLNSFICB	BLNSFMSK	BLNSFLGD
BLNSFDMG	BLNSFDMB	BLNSFINT	BLNSFNDN	BLNSFMSS	BLNSFDIS
BLNSFDMG	BLNSFDST	BLNSFINT	BLNSFTRM	BLNSFMSS	BLNSFLGD
BLNSFDMG	BLNSFFIN	BLNSFINT	ISPLINK	BLNSFMSS	BLNSFMSH
BLNSFDMG	BLNSFLGD	BLNSFIPT	ISPLINK	BLNSFMSS	ISPLINK
BLNSFDMG	BLNSFNDN	BLNSFISP	ISPLINK	BLNSFNDN	BLNFNDMC
BLNSFDMG	ISPLINK	BLNSFLGD	ISPLINK	BLNSFOFL	BLNFNPFC
BLNSFDMS	BLNSFDIS	BLNSFLNK	BLNFNSMC	BLNSFOFL	BLNSFDIS
BLNSFDMS	BLNSFFRZ	BLNSFLNK	BLNSFDIS	BLNSFOFL	BLNSFLGD
BLNSFDMS	BLNSFMSS	BLNSFLNK	BLNSFDST	BLNSFONL	BLNCMXCF
BLNSFDMS	BLNSFRST	BLNSFLNK	BLNSFLN1	BLNSFONL	BLNFNPNC
BLNSFDSP	BLNFNDSC	BLNSFLNK	BLNSFMDA	BLNSFONL	BLNSFDIS
BLNSFDSP	BLNSFDIS	BLNSFLNK	ISPLINK	BLNSFONL	BLNSFLGD
BLNSFDSP	BLNSFDST	BLNSFLN1	BLNFNSMC	BLNSFPNH	BLNSFFRZ
BLNSFDSP	BLNSFFIN	BLNSFLN1	BLNSFDIS	BLNSFPNH	BLNSFIPT
BLNSFDST	ISPLINK	BLNSFLN1	BLNSFDVD	BLNSFPNH	BLNSFRST
BLNSFDVD	BLNFNDDC	BLNSFLN1	BLNSFFRZ	BLNSFPRS	
BLNSFDVD	BLNSFDD2	BLNSFLN1	BLNSFLGD	BLNSFPRT	BLNSFISP
BLNSFDVD	BLNSFDIS	BLNSFLN1	BLNSFRST	BLNSFPRT	BLNSFSCM
BLNSFDVD	BLNSFDST	BLNSFLN1	ISPLINK	BLNSFRST	
BLNSFDVD	BLNSFLGD	BLNSFLOC	BLNFNSMC	BLNSFSCL	BLNCMXCF
BLNSFDVD	BLNSFMSK	BLNSFLOC	BLNSFDIS	BLNSFSCL	BLNFNDDC
BLNSFDVD	BLNSFSCL	BLNSFLOC	BLNSFDST	BLNSFSCM	BLNSFISP
BLNSFDVO	BLNFNDDO	BLNSFLOC	BLNSFLN1	BLNSFSCM	BLNSFPRS
BLNSFDVO	BLNSFDIS	BLNSFLOC	BLNSFMDA	BLNSFSCM	ISPLINK
BLNSFDVO	BLNSFLGD	BLNSFLOC	ISPLINK	BLNSFTRM	BLNSFDST
BLNSFDVS	BLNFNDVS	BLNSFLOG		BLNSFTRM	BLNSFVDL
BLNSFDVS	BLNSFASM	BLNSFMDA	BLNCMPRS	BLNSFTRM	ISPLINK
BLNSFDVS	BLNSFCAR	BLNSFMDA	BLNCMXCF	BLNSFVDF	ISPLINK
BLNSFDVS	BLNSFDIS	BLNSFMDA	BLNFNSMC	BLNSFVDL	ISPLINK
BLNSFDVS	BLNSFDVD	BLNSFMDB	BLNCMPRS	BLNSMBHX	BLNCMALN
BLNSFDVS	BLNSFDVO	BLNSFMDB	BLNCMXCF	BLNSMBHX	BLNSMHXA
BLNSFDVS	BLNSFFST	BLNSFMDB	BLNFNARC	BLNSMBHX	BLNSMHXO

Calling Module	Called Module
BLNSMBLB	BLNLBBLR
BLNSMBLB	BLNLBVER
BLNSMBLE	BLNASLOC
BLNSMBLE	BLNCMERM
BLNSMBLE	BLNSMBLF
BLNSMBLF	BLNLBVZA
BLNSMBLF	BLNSMBLH
BLNSMBLH	BLNSMBLB
BLNSMBLK	BLNCMALN
BLNSMBLK	BLNSMLKL
BLNSMBLK	BLNSMLKO
BLNSMBLL	BLNCMALN
BLNSMBLL	BLNSMBLE
BLNSMBLL	BLNSMBLO
BLNSMBLN	
BLNSMBLO	BLNSMBOL
BLNSMBLO	BLNSMLSL
BLNSMBMA	BLNSMBME
BLNSMBME	BLNSMBLN
BLNSMBME	BLNSMBMN
BLNSMBML	BLNASLST
BLNSMBML	BLNCMERM
BLNSMBML	BLNSMBMA
BLNSMBMN	
BLNSMBOK	BLNADIFR
BLNSMBOK	BLNDVFFR
BLNSMBOL	BLNCMXPF
BLNSMBOL	BLNSMBOK
BLNSMBPE	BLNCMALN
BLNSMBPE	BLNSMPSO
BLNSMBSI	BLNSMBHX
BLNSMBSI	BLNSMBLK
BLNSMBSI	BLNSMBPE
BLNSMBSI	BLNSMBTX
BLNSMBTX	BLNCMALN
BLNSMBTX	BLNSMTXA
BLNSMBTX	BLNSMTXO
BLNSMCML	BLNCMALN
BLNSMCML	BLNSMCMO
BLNSMCML	BLNSMFES
BLNSMCMO	
BLNSMDSP	BLNSMCML
BLNSMDSP	BLNSMSDC
BLNSMFCD	BLNSMFC1
BLNSMFC1	BLNSMFC1
BLNSMFES	BLNSMFCD
BLNSMFES	BLNSMFLD
BLNSMFES	BLNSMFTD
BLNSMFES	BLNSMFXD
BLNSMFLD	
BLNSMFND	BLNCMPRS
BLNSMFND	BLNSMDSP
BLNSMFND	BLNSMFN1
BLNSMFN1	BLNCMFBC
BLNSMFN1	BLNSMDSP
BLNSMFRE	BLNCMXPF
BLNSMFTD	
BLNSMFXD	
BLNSMGRE	BLNCMXPF
BLNSMHXA	BLNASLOC
BLNSMHXA	BLNCMERM
BLNSMHXA	BLNLBVER
BLNSMHXA	BLNMSGS
BLNSMHXA	BLNSMFRE
BLNSMHXA	BLNSMGRE
BLNSMHXA	BLNSMOCL
BLNSMHXA	BLNSMPSW
BLNSMHXO	
BLNSMINT	BLNSMBML
BLNSMLKB	BLNSMLKC
BLNSMLKB	BLNSMLKD
BLNSMLKB	BLNSMLKN
BLNSMLKC	BLNSMBLB
BLNSMLKD	
BLNSMLKG	BLNASLOC
BLNSMLKG	BLNCMERM
BLNSMLKG	BLNLBVZA
BLNSMLKI	BLNSMLKG
BLNSMLKL	BLNASLOC
BLNSMLKL	BLNCMERM
BLNSMLKL	BLNLBVER
BLNSMLKL	BLNSMLKB
BLNSMLKL	BLNSMLKI
BLNSMLKN	BLNASLOC
BLNSMLKN	BLNCMERM
BLNSMLKN	BLNSMLKN
BLNSMLKO	BLNSMLKP
BLNSMLKP	BLNSMBOL
BLNSMLKP	BLNSMLKP
BLNSMLSL	
BLNSMLSP	BLNSMLSP
BLNSMOCL	BLNCMXPF
BLNSMOCL	BLNDDBOD
BLNSMOCL	BLNDDEBC
BLNSMOCL	BLNDDLSZ
BLNSMPCD	BLNCMFBC
BLNSMPCD	BLNCMPRT
BLNSMPCD	BLNSMDSP
BLNSMPCD	BLNSMPHP
BLNSMPCD	BLNSMPLA
BLNSMPHP	
BLNSMPLA	
BLNSMPRT	BLNCMPRS
BLNSMPRT	BLNSMPCD
BLNSMPSO	
BLNSMPSW	BLNCMXPF
BLNSMSDC	BLNSMBLL
BLNSMSDC	BLNSMBSI
BLNSMSDC	BLNSMFES
BLNSMSEL	BLNSMSEM
BLNSMSEL	BLNSMSNI
BLNSMSEM	
BLNSMSNC	BLNCMXPF
BLNSMSNC	BLNSMLSP
BLNSMSNI	BLNSMSNC
BLNSMSNI	BLNSMSNL
BLNSMSNL	BLNCMXPF
BLNSMSNL	BLNSMLSL
BLNSMTRM	BLNSMFES
BLNSMTXA	BLNASLOC
BLNSMTXA	BLNCMBUT
BLNSMTXA	BLNCMERM
BLNSMTXA	BLNLBVER
BLNSMTXA	BLNMSGS
BLNSMTXO	
BLNSUBMT	
BLNTERM	BLNADIFX
BLNTERM	BLNASTRM
BLNTERM	BLNCMFBC
BLNUSADM	
BLNUSDDM	
CDP	

Calling Module	Called Module
CTA	
DUMPACC	
EXITRTN	
ISPLINK	

INFO/ANALYSIS CALLED/CALLING MODULE CROSS-REFERENCE

This section lists, alphabetically, the names of Info/Analysis modules that are called by other modules. Following each called module name are the names of the modules that call it.
(Refer to "Info/Analysis Calling/Called Module Cross-Reference" on page 246 for additional information.)

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNADIFI	BLNMAIN	BLNARDSP	BLNARFCS	BLNASAXK	BLNASMD
BLNADIFR	BLNASRSR	BLNARDSP	BLNARFND	BLNASAXT	BLNASMD
BLNADIFR	BLNAXDA1	BLNARDSP	BLNFNARC	BLNASAXX	BLNASMD
BLNADIFR	BLNAXSUA	BLNAREXA	BLNAREXE	BLNASCME	BLNASMD
BLNADIFR	BLNAXS6V	BLNAREXC	BLNFNARC	BLNASCME	BLNASMDA
BLNADIFR	BLNDDPOP	BLNAREXD	BLNAREXC	BLNASFMT	BLNASLD6
BLNADIFR	BLNDDQRY	BLNAREXE	BLNAREXC	BLNASFSY	BLNASLD1
BLNADIFR	BLNDDREQ	BLNAREXF	BLNAREXE	BLNASGET	BLNASCME
BLNADIFR	BLNDMCK2	BLNAREXG	BLNAREXF	BLNASGET	BLNASIST
BLNADIFR	BLNDMDEL	BLNAREXL	BLNAREXE	BLNASIST	BLNASCME
BLNADIFR	BLNLBBCI	BLNARFCS	BLNARFND	BLNASLCB	BLNASLOC
BLNADIFR	BLNLBCHR	BLNARFML	BLNARFML	BLNASLCB	BLNASLST
BLNADIFR	BLNLBCLN	BLNARFML	BLNARGDN	BLNASLCB	BLNASTRM
BLNADIFR	BLNLBCNT	BLNARFML	BLNARGRN	BLNASLCB	BLNASUPD
BLNADIFR	BLNLBLAL	BLNARFML	BLNARTRM	BLNASLD1	BLNASLOC
BLNADIFR	BLNLBRCB	BLNARFND	BLNFNARC	BLNASLD6	BLNASLOC
BLNADIFR	BLNMKMSK	BLNARGDN	BLNARGRN	BLNASLOC	BLNARXPL
BLNADIFR	BLNOFFLD	BLNARGRN	BLNARINT	BLNASLOC	BLNAXSA1
BLNADIFR	BLNOFOUT	BLNARINT	BLNFNARC	BLNASLOC	BLNCMCHD
BLNADIFR	BLNONEND	BLNARPDI	BLNARPRN	BLNASLOC	BLNDDFMT
BLNADIFR	BLNONSC1	BLNARPRN	BLNARGDN	BLNASLOC	BLNDDRDF
BLNADIFR	BLNONTRW	BLNARPRT	BLNFNARC	BLNASLOC	BLNDDSYI
BLNADIFR	BLNSMBOK	BLNARTRM	BLNFNARC	BLNASLOC	BLNDMAD1
BLNADIFT	BLNADIFR	BLNARXPL	BLNAREXE	BLNASLOC	BLNDMUDI
BLNADIFT	BLNADIFX	BLNASADF	BLNASMD	BLNASLOC	BLNDSCEN
BLNADIFX	BLNTERM	BLNASADL	BLNASMD	BLNASLOC	BLNDSCNS
BLNARDLO	BLNARDSP	BLNASAXA	BLNASMD	BLNASLOC	BLNDSCRQ
BLNARDLO	BLNARPRT	BLNASAXC	BLNASMD	BLNASLOC	BLNDSCST

Called Module	Calling Module
BLNASLOC	BLNSMBLE
BLNASLOC	BLNSMHXA
BLNASLOC	BLNSMLKG
BLNASLOC	BLNSMLKL
BLNASLOC	BLNSMLKN
BLNASLOC	BLNSMTXA
BLNASLST	BLNSMBML
BLNASMAP	BLNASLOC
BLNASMAP	BLNASLST
BLNASMD	BLNASMP6
BLNASMDA	BLNASMP6
BLNASMP1	BLNASAXX
BLNASMP1	BLNASMAP
BLNASMP6	BLNASMAP
BLNASMP6	BLNASUP6
BLNASRFL	BLNASFMT
BLNASRSR	BLNASFMT
BLNASRSR	BLNASMAP
BLNASRSR	BLNASUP6
BLNASTMP	BLNASMAP
BLNASTMP	BLNASTRM
BLNASTRM	BLNDMDEL
BLNASTRM	BLNTERM
BLNASUPD	BLNAXSUA
BLNASUPD	BLNAXS6U
BLNASUP6	BLNASUPD
BLNAXCMB	BLNAXPD2
BLNAXCMB	BLNAXPD3
BLNAXCMB	BLNAXPD6
BLNAXCMB	BLNAXSU1
BLNAXCMB	BLNAXS6U
BLNAXDAC	
BLNAXDA1	BLNAXDAC
BLNAXFMT	
BLNAXFM1	BLNAXFMT
BLNAXGST	
BLNAXGS1	BLNAXGST
BLNAXGS2	BLNARXPL
BLNAXGS2	BLNAXFM1
BLNAXGS2	BLNAXGS1
BLNAXGS2	BLNAXMSG
BLNAXGS2	BLNAXSA1
BLNAXGS2	BLNAXS3U
BLNAXGS2	BLNAXS4U
BLNAXGS2	BLNAXS5U
BLNAXGS3	BLNAXGS1
BLNAXGS3	BLNAXPD1
BLNAXGS3	BLNAXS3U
BLNAXGS3	BLNAXS4U
BLNAXGS3	BLNAXS5U
BLNAXMSG	BLNAXCMB
BLNAXMSG	BLNAXFM1
BLNAXPDS	
BLNAXPD1	BLNAXPDS
BLNAXPD2	BLNAXPD1
BLNAXPD2	BLNAXPD3
BLNAXPD3	BLNAXPD1
BLNAXPD4	BLNAXPD3
BLNAXPD4	BLNAXPD4
BLNAXPD5	BLNAXPD4
BLNAXPD6	BLNAXPD4
BLNAXPD6	BLNAXPD5
BLNAXPD7	BLNAXPD3
BLNAXSAC	
BLNAXSA1	BLNAXSAC
BLNAXSRV	BLNAXSU1
BLNAXSUA	BLNAXSUA
BLNAXSUA	BLNAXSU1
BLNAXSUP	
BLNAXSUR	BLNAXSU1
BLNAXSU1	BLNAXSUP
BLNAXS3U	BLNAXSUR
BLNAXS4U	BLNAXSUR
BLNAXS4U	BLNAXS3U
BLNAXS5U	BLNAXSUR
BLNAXS6U	BLNAXSU1
BLNAXS6V	BLNAXS6U
BLNBABH1	BLNBAINT
BLNBABH2	BLNBABH1
BLNBABH2	BLNBADM1
BLNBABH2	BLNBADS1
BLNBABH2	BLNBADV1
BLNBADM1	BLNBABH1
BLNBADM2	BLNBABH1
BLNBADM2	BLNBADM1
BLNBADS1	BLNBABH1
BLNBADV1	BLNBABH1
BLNBADV2	BLNBADV1
BLNBADV3	BLNBADV1
BLNBAFLS	BLNBABH1
BLNBAINT	BLNMAIN
BLNBAIPF	BLNBAINT
BLNBAMSG	BLNBABH2
BLNBAMSG	BLNBAFLS
BLNBAMSG	BLNBAINT
BLNBAMSG	BLNBAPN1
BLNBAMSG	BLNBARDR
BLNBAPF1	BLNBABH1
BLNBAPN1	BLNBABH1
BLNBAPRS	BLNBARDR
BLNBARDR	BLNBABH1
BLNBARDR	BLNBABH2
BLNBARDR	BLNBADM1
BLNBARDR	BLNBADS1
BLNBARDR	BLNBADV1
BLNBARDR	BLNBAFLS
BLNBARDR	BLNBAPF1
BLNBARDR	BLNBAPN1
BLNBAVSE	BLNBAIPF
BLNCMALN	BLNARDSP
BLNCMALN	BLNARPRT
BLNCMALN	BLNSMBHX
BLNCMALN	BLNSMBLK
BLNCMALN	BLNSMBLL
BLNCMALN	BLNSMBPE
BLNCMALN	BLNSMBTX
BLNCMALN	BLNSMCML
BLNCMBUT	BLNDDBTE
BLNCMBUT	BLNSFBUM
BLNCMBUT	BLNSFCMQ
BLNCMBUT	BLNSMTXA
BLNCMCHD	BLNFNDMC
BLNCMDEC	BLNCMFIL
BLNCMERM	BLNAREXF
BLNCMERM	BLNARXPL
BLNCMERM	BLNDDFMT
BLNCMERM	BLNDDQRY
BLNCMERM	BLNDDRDF
BLNCMERM	BLNDDREQ
BLNCMERM	BLNDDSYI
BLNCMERM	BLNDMAD1
BLNCMERM	BLNDMCK2

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNCMERM	BLNDMDEL	BLNCMHEX	BLNDDPOP	BLNCMVOL	BLNBAPN1
BLNCMERM	BLNDMUDI	BLNCMHEX	BLNDDQAL	BLNCMXBI	BLNDVFFR
BLNCMERM	BLNDSCEN	BLNCMHEX	BLNDDVFA	BLNCMXCF	BLNSCTOK
BLNCMERM	BLNDSCNS	BLNCMHEX	BLNDDVFN	BLNCMXCF	BLNSFMDA
BLNCMERM	BLNDSCRQ	BLNCMHEX	BLNMKVMR	BLNCMXCF	BLNSFMDB
BLNCMERM	BLNDSCST	BLNCMHEX	BLNSFDD3	BLNCMXCF	BLNSFONL
BLNCMERM	BLNLBCHR	BLNCMLST	BLNBARDR	BLNCMXCF	BLNSFSCL
BLNCMERM	BLNLBCLN	BLNCMLST	BLNPRTS	BLNCMXDC	BLNDVFFR
BLNCMERM	BLNLBCNT	BLNCMMOB	BLNDDFDZ	BLNCMXDC	BLNSCBKW
BLNCMERM	BLNLBLAL	BLNCMMOB	BLNDDFPP	BLNCMXDC	BLNSCCTS
BLNCMERM	BLNLBRCB	BLNCMMOB	BLNDDHDO	BLNCMXDC	BLNSCFWD
BLNCMERM	BLNMKMSK	BLNCMMOB	BLNDDMDT	BLNCMXDE	BLNARGDN
BLNCMERM	BLNOFFLD	BLNCMMOB	BLNDDRPT	BLNCMXEB	BLNDDEBC
BLNCMERM	BLNOFOUT	BLNCMMOB	BLNMKBMD	BLNCMXEB	BLNDVFFR
BLNCMERM	BLNONEND	BLNCMMSP	BLNPRTS	BLNCMXPF	BLNDDBAS
BLNCMERM	BLNONSC1	BLNCMPRI	BLNCMCHD	BLNCMXPF	BLNDDBOD
BLNCMERM	BLNONTRW	BLNCMPRO	BLNCMPRT	BLNCMXPF	BLNDDFDO
BLNCMERM	BLNSMBLE	BLNCMPRS	BLNAREXD	BLNCMXPF	BLNDDHBL
BLNCMERM	BLNSMBML	BLNCMPRS	BLNARFND	BLNCMXPF	BLNDDHDO
BLNCMERM	BLNSMHXA	BLNCMPRS	BLNARPRT	BLNCMXPF	BLNDDHFN
BLNCMERM	BLNSMLKG	BLNCMPRS	BLNDDBAS	BLNCMXPF	BLNDDMDT
BLNCMERM	BLNSMLKL	BLNCMPRS	BLNDDPOP	BLNCMXPF	BLNDDQAL
BLNCMERM	BLNSMLKN	BLNCMPRS	BLNDDQAL	BLNCMXPF	BLNDDRDF
BLNCMERM	BLNSMTXA	BLNCMPRS	BLNDDVFN	BLNCMXPF	BLNDDRPT
BLNCMFBC	BLNARFCS	BLNCMPRS	BLNMKVMR	BLNCMXPF	BLNDDVRF
BLNCMFBC	BLNARPRT	BLNCMPRS	BLNSFMDA	BLNCMXPF	BLNDVFFR
BLNCMFBC	BLNAXFM1	BLNCMPRS	BLNSFMDB	BLNCMXPF	BLNMKVMR
BLNCMFBC	BLNDDFFN	BLNCMPRS	BLNSMFND	BLNCMXPF	BLNSFDD3
BLNCMFBC	BLNDDFMT	BLNCMPRS	BLNSMPRT	BLNCMXPF	BLNSMBOL
BLNCMFBC	BLNDDFND	BLNCMPRT	BLNARPRT	BLNCMXPF	BLNSMFRE
BLNCMFBC	BLNDDFPP	BLNCMPRT	BLNAXPD2	BLNCMXPF	BLNSMGRE
BLNCMFBC	BLNDDRPT	BLNCMPRT	BLNAXPD6	BLNCMXPF	BLNSMOCL
BLNCMFBC	BLNDMPDL	BLNCMPRT	BLNDDDPT	BLNCMXPF	BLNSMPSW
BLNCMFBC	BLNDSDSP	BLNCMPRT	BLNDDFPP	BLNCMXPF	BLNSMSNC
BLNCMFBC	BLNDSPRT	BLNCMPRT	BLNDDRPT	BLNCMXPF	BLNSMSNL
BLNCMFBC	BLNDSTRM	BLNCMPRT	BLNDMPDL	BLNCONS	BLNCOPT
BLNCMFBC	BLNSFBST	BLNCMPRT	BLNDSPRT	BLNCOPT	
BLNCMFBC	BLNSFMSK	BLNCMPRT	BLNSMPCD	BLNDDBAS	BLNDDVRF
BLNCMFBC	BLNSMFN1	BLNCMTDS	BLNCMCHD	BLNDDBOD	BLNDDHBL
BLNCMFBC	BLNSMPCD	BLNCMTDS	BLNDMAD1	BLNDDBOD	BLNSMOCL
BLNCMFBC	BLNTERM	BLNCMTDS	BLNDSFED	BLNDDBTE	BLNDDFMT
BLNCMFIL	BLNBAPN1	BLNCMTSZ	BLNSCSUD	BLNDDCDB	BLNDDDPT
BLNCMHEX	BLNDDBAS	BLNCMTSZ	BLNSFINT	BLNDDCDB	BLNDDDSP
BLNCMHEX	BLNDDHFN	BLNCMVOL	BLNBAPF1	BLNDDCDB	BLNDDFND

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNDDCDB	BLNDDFPD	BLNDDRDF	BLNDDVRF	BLNDMRDF	BLNDMDEL
BLNDDCDB	BLNDDHFN	BLNDDREQ	BLNDDCDB	BLNDMRDF	BLNDMDMU
BLNDDDPT	BLNDDPRT	BLNDDREQ	BLNDDFDZ	BLNDMRDF	BLNDMPDL
BLNDDDSP	BLNFNDDC	BLNDDREQ	BLNDDHFN	BLNDMRDF	BLNDMUAI
BLNDDEBC	BLNDDHBL	BLNDDREQ	BLNDDMDT	BLNDMRDF	BLNOFFLD
BLNDDEBC	BLNSMOCL	BLNDDREQ	BLNDDRPT	BLNDMRDF	BLNOFUDM
BLNDDFDO	BLNDDFMT	BLNDDREQ	BLNMKBMD	BLNDMRDF	BLNONEND
BLNDDFDZ	BLNDDFDO	BLNDDREQ	BLNMKQRY	BLNDMRDF	BLNONINT
BLNDDFFN	BLNDDFND	BLNDDRPT	BLNDDDPT	BLNDMRD1	BLNDMRDF
BLNDDFMT	BLNAXFM1	BLNDDRPT	BLNDDPRT	BLNDMSRT	BLNDMADU
BLNDDFMT	BLNDDCDB	BLNDDSYI	BLNDDINT	BLNDMSRT	BLNDMDDL
BLNDDFMT	BLNDDRPT	BLNDDSYM	BLNDDBAS	BLNDMSRT	BLNDMPDL
BLNDDFMT	BLNMKBMD	BLNDDTRM	BLNFNDDC	BLNDMSRT	BLNONEND
BLNDDFND	BLNFNDDC	BLNDDVFA	BLNDDVFN	BLNDMTRM	BLNDMDMU
BLNDDFPD	BLNDDFPL	BLNDDVFN	BLNDDFND	BLNDMTRM	BLNDMGNT
BLNDDFPL	BLNDDFPS	BLNDDVRF	BLNDDCDB	BLNDMUAB	
BLNDDFPP	BLNDDDPT	BLNDMADU	BLNDMGNT	BLNDMUAD	BLNDMUAB
BLNDDFPP	BLNDDFPD	BLNDMADU	BLNDMUAD	BLNDMUAI	BLNDMGNT
BLNDDFPS	BLNDDFPT	BLNDMAD1	BLNDMADU	BLNDMUDI	BLNDMDEL
BLNDDFPT	BLNDDPRT	BLNDMAD1	BLNDMCK2	BLNDMWDF	BLNDMADU
BLNDDHBL	BLNDDHDO	BLNDMAD1	BLNONEND	BLNDMWDF	BLNDMCK2
BLNDDHDO	BLNDDFMT	BLNDMCHK	BLNDMADU	BLNDMWDF	BLNDMDEL
BLNDDHFN	BLNDDFND	BLNDMCHK	BLNDMDDL	BLNDMWDF	BLNDMDMU
BLNDDINT	BLNFNDDC	BLNDMCHK	BLNDMDEL	BLNDMWDF	BLNOFUDM
BLNDDLSZ	BLNAXFM1	BLNDMCHK	BLNDMPDL	BLNDMWDF	BLNONEND
BLNDDLSZ	BLNDDCDB	BLNDMCHK	BLNOFFLD	BLNDSBCL	BLNDSBSO
BLNDDLSZ	BLNDDFND	BLNDMCHK	BLNOFUDM	BLNDSBSO	BLNDSCNS
BLNDDLSZ	BLNDDRPT	BLNDMCHK	BLNONEND	BLNDSBSO	BLNDSCRQ
BLNDDLSZ	BLNMKSMB	BLNDMCHK	BLNONINT	BLNDSBSO	BLNDSCST
BLNDDLSZ	BLNSMOCL	BLNDMCK1	BLNDMCHK	BLNDSCEN	BLNDSCOL
BLNDDMDT	BLNDDHBL	BLNDMCK2	BLNDMCHK	BLNDSCNS	BLNDSCOL
BLNDDNBU	BLNDDHDO	BLNDMCK2	BLNDMCK1	BLNDSCOL	BLNDSDSP
BLNDDNBU	BLNDDNLI	BLNDMCK3	BLNDMCK1	BLNDSCOL	BLNDSPRT
BLNDDNLI	BLNDDBTE	BLNDMCNT	BLNDMADU	BLNDSCRQ	BLNDSCOL
BLNDDNLI	BLNDDCDB	BLNDMCNT	BLNDMDEL	BLNDSCST	BLNDSCOL
BLNDDNLI	BLNDDFDO	BLNDMDDL	BLNDMGNT	BLNDSDSP	BLNDSYMP
BLNDDNLI	BLNDDFDZ	BLNDMDEL	BLNDMGNT	BLNDSFED	BLNDSCEN
BLNDDNLI	BLNDDHDO	BLNDMDMU	BLNDMGNT	BLNDSGBF	BLNDSBCL
BLNDDPOP	BLNDDPRT	BLNDMGNT	BLNFNDMC	BLNDSGBF	BLNDSBSO
BLNDDPRT	BLNFNDDC	BLNDMGNT	BLNFNPFC	BLNDSGBF	BLNDSCEN
BLNDDQAL	BLNDDVRF	BLNDMGNT	BLNFNPNC	BLNDSGBF	BLNDSCNS
BLNDDQRY	BLNDDHFN	BLNDMPDL	BLNDMGNT	BLNDSGBF	BLNDSCRQ
BLNDDQRY	BLNDDMDT	BLNDMRDF	BLNDMADU	BLNDSGBF	BLNDSCST
BLNDDQRY	BLNDDRPT	BLNDMRDF	BLNDMDDL	BLNDSGBF	BLNDSFED

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNDSGBF	BLNDSPRT	BLNFNDMC	BLNBADM1	BLNLBBCI	BLNLBCHW
BLNDSPRT	BLNDSYMP	BLNFNDMC	BLNSFDMD	BLNLBBCI	BLNLBSIM
BLNDSTRM	BLNDSYMP	BLNFNDMC	BLNSFDMG	BLNLBBLR	BLNSMBLB
BLNDSYMP	BLNFNDSC	BLNFNDMC	BLNSFNDN	BLNLBCAC	BLNLBCHN
BLNDVFFR	BLNDDFDO	BLNFNDSC	BLNBADS1	BLNLBCAE	BLNLBCHN
BLNDVFFR	BLNSMBOK	BLNFNDSC	BLNSFDSP	BLNLBCAI	BLNLBCHN
BLNDVFPE	BLNAREXA	BLNFNDVS	BLNBADV2	BLNLBCAL	BLNLBCHN
BLNDVFPE	BLNDVSIN	BLNFNDVS	BLNSFDVS	BLNLBCEF	BLNLBBCI
BLNDVKCB	BLNDDFPT	BLNFNMSK	BLNSFMSK	BLNLBCEF	BLNLBCAE
BLNDVRCB	BLNDDFPT	BLNFNPC1	BLNFNPNC	BLNLBCEF	BLNLBCAL
BLNDVSIA	BLNDVSIN	BLNFNPC2	BLNFNPFC	BLNLBCEF	BLNLBCHE
BLNDVSIN	BLNFNDVS	BLNFNPFC	BLNBAPF1	BLNLBCEF	BLNLBCHL
BLNDVSTM	BLNDVSIN	BLNFNPFC	BLNSFOFL	BLNLBCHC	BLNLBCHN
BLNDVSTM	BLNFNDVS	BLNFNPNC	BLNBAPN1	BLNLBCHE	BLNLBCHN
BLNERCMB	BLNASADF	BLNFNPNC	BLNSFONL	BLNLBCHI	BLNLBCHN
BLNERCMB	BLNASADL	BLNFNSMC	BLNDDFPD	BLNLBCHL	BLNLBCHN
BLNERCMB	BLNASAXA	BLNFNSMC	BLNDDFPS	BLNLBCHN	BLNLBARB
BLNERCMB	BLNASAXC	BLNFNSMC	BLNDVKCB	BLNLBCHN	BLNLBBLR
BLNERCMB	BLNASAXK	BLNFNSMC	BLNDVRCB	BLNLBCHR	BLNLBARR
BLNERCMB	BLNASAXT	BLNFNSMC	BLNSFASM	BLNLBCHR	BLNLBBLR
BLNERCMB	BLNASAXX	BLNFNSMC	BLNSFAS1	BLNLBCHW	BLNLBCHN
BLNERCMB	BLNASFMT	BLNFNSMC	BLNSFDAT	BLNLBCLN	BLNLBCHC
BLNERCMB	BLNASGET	BLNFNSMC	BLNSFLNK	BLNLBCLN	BLNLBCHE
BLNERCMB	BLNASLCB	BLNFNSMC	BLNSFLN1	BLNLBCLN	BLNLBCHI
BLNERCMB	BLNASLD1	BLNFNSMC	BLNSFLOC	BLNLBCLN	BLNLBCHL
BLNERCMB	BLNASLOC	BLNFNSMC	BLNSFMDA	BLNLBCLN	BLNLBCHW
BLNERCMB	BLNASLST	BLNINIT		BLNLBCMA	BLNLBBCI
BLNERCMB	BLNASMAP	BLNINPRT	BLNCMPRO	BLNLBCNT	BLNLBARR
BLNERCMB	BLNASMD	BLNINPRT	BLNCMPRT	BLNLBCNT	BLNLBCAC
BLNERCMB	BLNASMP1	BLNITSPF		BLNLBCNT	BLNLBCHC
BLNERCMB	BLNASRSR	BLNLBARB	BLNLBARY	BLNLBLAL	BLNLBCAC
BLNERCMB	BLNASTRM	BLNLBARC	BLNLBARB	BLNLBLAL	BLNLBCAE
BLNFNARC	BLNBADV3	BLNLBARC	BLNLBARY	BLNLBLAL	BLNLBCAI
BLNFNARC	BLNSFCAR	BLNLBARR	BLNLBBLR	BLNLBLAL	BLNLBCAL
BLNFNARC	BLNSFCA1	BLNLBARY	BLNLBBLR	BLNLBRCB	BLNLBBCI
BLNFNARC	BLNSFMDB	BLNLBBCI	BLNLBARC	BLNLBSIM	BLNLBARB
BLNFNDC2	BLNFNDMC	BLNLBBCI	BLNLBCAC	BLNLBSIM	BLNLBBLR
BLNFNDC3	BLNFNDSC	BLNLBBCI	BLNLBCAE	BLNLBVER	BLNDDFMT
BLNFNDDC	BLNBADV1	BLNLBBCI	BLNLBCAI	BLNLBVER	BLNDDSYI
BLNFNDDC	BLNSFDD2	BLNLBBCI	BLNLBCAL	BLNLBVER	BLNSMBLB
BLNFNDDC	BLNSFDVD	BLNLBBCI	BLNLBCHC	BLNLBVER	BLNSMHXA
BLNFNDDC	BLNSFSCL	BLNLBBCI	BLNLBCHE	BLNLBVER	BLNSMLKL
BLNFNDDO	BLNDVSIN	BLNLBBCI	BLNLBCHI	BLNLBVER	BLNSMTXA
BLNFNDDO	BLNSFDVO	BLNLBBCI	BLNLBCHL	BLNLBVZA	BLNDDRDF

Called Module	Calling Module
BLNLBVZA	BLNSMBLF
BLNLBVZA	BLNSMLKG
BLNMACBC	BLNMAIN
BLNMACBG	BLNMACBC
BLNMAIN	BLNINIT
BLNMAIN	BLNITSPF
BLNMAOCM	BLNMAIN
BLNMAPDF	BLNMACBC
BLNMKBMD	BLNMKDSP
BLNMKDSP	BLNFNMSK
BLNMKINT	BLNFNMSK
BLNMKMSK	BLNFNMSK
BLNMKQRY	BLNMKDSP
BLNMKSMB	BLNMKDSP
BLNMKTRM	BLNFNMSK
BLNMKVMR	BLNMKDSP
BLNMSGS	BLNARGDN
BLNMSGS	BLNARGRN
BLNMSGS	BLNARINT
BLNMSGS	BLNARPRN
BLNMSGS	BLNCMERM
BLNMSGS	BLNDSCEN
BLNMSGS	BLNDSDSP
BLNMSGS	BLNDSGBF
BLNMSGS	BLNDSPRT
BLNMSGS	BLNDSTRM
BLNMSGS	BLNDSYMP
BLNMSGS	BLNFNARC
BLNMSGS	BLNSMHXA
BLNMSGS	BLNSMTXA
BLNOFFLD	BLNDMGNT
BLNOFOUT	BLNOFFLD
BLNOFUDM	BLNOFFLD
BLNONDDT	BLNONLOD
BLNONEND	BLNONLOD
BLNONINT	BLNONLOD
BLNONLOD	BLNDMGNT
BLNONLOD	
BLNONMUD	BLNONLOD
BLNONSCM	BLNONLOD
BLNONSC1	BLNONSCM
BLNONTRW	BLNONLOD
BLNONVSE	BLNONDDT
BLNPRTS	BLNCOPT
BLNRAADB	BLNRAS6B
BLNRAARB	BLNRAS6B
BLNRAASB	BLNRAS6B
BLNRABAB	BLNRAS6B
BLNRACCB	BLNRAS6B
BLNRADDB	BLNRAS6B
BLNRADMB	BLNRAS6B
BLNRADSB	BLNRAS6B
BLNRADVT	BLNRAS6B
BLNRAEIN	BLNRASEC
BLNRALBD	BLNRAADB
BLNRALBD	BLNRAARB
BLNRALBD	BLNRAASB
BLNRALBD	BLNRABAB
BLNRALBD	BLNRACCB
BLNRALBD	BLNRADDB
BLNRALBD	BLNRADMB
BLNRALBD	BLNRADSB
BLNRALBD	BLNRADVT
BLNRALBD	BLNRAMKB
BLNRALBD	BLNRASCB
BLNRALBD	BLNRASFB
BLNRALBD	BLNRASMB
BLNRAMKB	BLNRAS6B
BLNRAS	BLNMAIN
BLNRASCB	BLNRAS6B
BLNRASEC	BLNRAS
BLNRASFB	BLNRAS6B
BLNRASMB	BLNRAS6B
BLNRAS4B	BLNRASEC
BLNRAS5B	BLNRASEC
BLNRAS6B	BLNRASEC
BLNRAS6S	BLNRAADB
BLNRAS6S	BLNRAARB
BLNRAS6S	BLNRAASB
BLNRAS6S	BLNRABAB
BLNRAS6S	BLNRACCB
BLNRAS6S	BLNRADDB
BLNRAS6S	BLNRADMB
BLNRAS6S	BLNRADSB
BLNRAS6S	BLNRADVT
BLNRAS6S	BLNRAMKB
BLNRAS6S	BLNRASCB
BLNRAS6S	BLNRASFB
BLNRAS6S	BLNRASMB
BLNSCBKW	BLNSCHDL
BLNSCCCP	BLNSCOTH
BLNSCCRP	BLNSCBKW
BLNSCCRP	BLNSCFWD
BLNSCCRP	BLNSCRDT
BLNSCCSC	BLNSCDIS
BLNSCCSC	BLNSCOTH
BLNSCCTS	BLNSCBKW
BLNSCCTS	BLNSCDIS
BLNSCCTS	BLNSCFWD
BLNSCCTS	BLNSCLFT
BLNSCCTS	BLNSCRGH
BLNSCCUP	BLNSCCCP
BLNSCCUP	BLNSCHDL
BLNSCDIS	BLNSCDTB
BLNSCDRC	BLNSCDIS
BLNSCDTB	BLNSFCSR
BLNSCERK	BLNSCDIS
BLNSCFWD	BLNSCHDL
BLNSCHDL	BLNSCDIS
BLNSCLFT	
BLNSCOTH	BLNSCDIS
BLNSCPRD	BLNSCDTB
BLNSCRDT	BLNSCDTB
BLNSCRGH	
BLNSCSUB	BLNSCBKW
BLNSCSUB	BLNSCFWD
BLNSCSUB	BLNSCLFT
BLNSCSUB	BLNSCRDT
BLNSCSUB	BLNSCRGH
BLNSCSUD	BLNSCPRD
BLNSCTOK	BLNSCBKW
BLNSCTOK	BLNSCDRC
BLNSCTOK	BLNSCFWD
BLNSCTOK	BLNSCHDL
BLNSCTOK	BLNSCLFT
BLNSCTOK	BLNSCRGH
BLNSFAMS	BLNSFIMP
BLNSFASM	BLNSFDVS
BLNSFAS1	BLNSFASM
BLNSFBST	BLNSFDIS
BLNSFBUM	BLNSFAMS
BLNSFCAR	BLNSFDVS

Called Module	Calling Module	Called Module	Calling Module	Called Module	Calling Module
BLNSFCA1	BLNSFCAR	BLNSFDST	BLNSFLNK	BLNSFLOC	BLNSFAS1
BLNSFCDS	BLNSFDIS	BLNSFDST	BLNSFLOC	BLNSFLOC	
BLNSFCER	BLNSFCDS	BLNSFDST	BLNSFTRM	BLNSFLOG	BLNSFCDS
BLNSFCMQ	BLNSFIMP	BLNSFDVD	BLNSFDVS	BLNSFMDA	BLNSFASM
BLNSFCSR	BLNSFCDS	BLNSFDVD	BLNSFLN1	BLNSFMDA	BLNSFDAT
BLNSFDAT	BLNSFAS1	BLNSFDVO	BLNSFDVS	BLNSFMDA	BLNSFLNK
BLNSFDD2	BLNSFDVD	BLNSFDVS	BLNSFDID	BLNSFMDA	BLNSFLOC
BLNSFDD3	BLNSFDD2	BLNSFFIN	BLNSFDMG	BLNSFMDB	BLNSFCAR
BLNSFDID	BLNSFINT	BLNSFFIN	BLNSFDSP	BLNSFMHH	BLNSFMSH
BLNSFDID		BLNSFFRZ	BLNSFAS1	BLNSFMSH	BLNSFHLP
BLNSFDIS	BLNSFASM	BLNSFFRZ	BLNSFDMS	BLNSFMSH	BLNSFMSS
BLNSFDIS	BLNSFAS1	BLNSFFRZ	BLNSFLN1	BLNSFMSK	BLNSFDVD
BLNSFDIS	BLNSFCAR	BLNSFFRZ	BLNSFMHH	BLNSFMSS	BLNSFDMS
BLNSFDIS	BLNSFCA1	BLNSFFRZ	BLNSFMSH	BLNSFNDN	BLNSFDID
BLNSFDIS	BLNSFDAT	BLNSFFRZ	BLNSFPNH	BLNSFNDN	BLNSFDMB
BLNSFDIS	BLNSFDID	BLNSFFST	BLNSFDID	BLNSFNDN	BLNSFDMG
BLNSFDIS	BLNSFDMB	BLNSFFST	BLNSFDVS	BLNSFNDN	BLNSFINT
BLNSFDIS	BLNSFDMD	BLNSFHLP	BLNSFCDS	BLNSFOFL	BLNSFDID
BLNSFDIS	BLNSFDMG	BLNSFICB	BLNSFINT	BLNSFONL	BLNSFDID
BLNSFDIS	BLNSFDMS	BLNSFIMP	BLNSFCER	BLNSFPNH	BLNSFDID
BLNSFDIS	BLNSFDSP	BLNSFINT	BLNMAIN	BLNSFPNH	BLNSFHLP
BLNSFDIS	BLNSFDVD	BLNSFIPT	BLNSFMSH	BLNSFPRS	BLNSFCDS
BLNSFDIS	BLNSFDVO	BLNSFIPT	BLNSFPNH	BLNSFPRS	BLNSFSCM
BLNSFDIS	BLNSFDVS	BLNSFISP	BLNSFCDS	BLNSFPRT	BLNSFCDS
BLNSFDIS	BLNSFLNK	BLNSFISP	BLNSFPRT	BLNSFRST	BLNSFAS1
BLNSFDIS	BLNSFLN1	BLNSFISP	BLNSFSCM	BLNSFRST	BLNSFDMS
BLNSFDIS	BLNSFLOC	BLNSFLGD	BLNSFAS1	BLNSFRST	BLNSFLN1
BLNSFDIS	BLNSFMHH	BLNSFLGD	BLNSFCA1	BLNSFRST	BLNSFMHH
BLNSFDIS	BLNSFMSH	BLNSFLGD	BLNSFCDS	BLNSFRST	BLNSFMSH
BLNSFDIS	BLNSFMSK	BLNSFLGD	BLNSFCMQ	BLNSFRST	BLNSFPNH
BLNSFDIS	BLNSFMSS	BLNSFLGD	BLNSFDID	BLNSFSCL	BLNSFDVD
BLNSFDIS	BLNSFOFL	BLNSFLGD	BLNSFDMB	BLNSFSCM	BLNSFCER
BLNSFDIS	BLNSFONL	BLNSFLGD	BLNSFDMD	BLNSFSCM	BLNSFFRZ
BLNSFDMB	BLNSFDMG	BLNSFLGD	BLNSFDMG	BLNSFSCM	BLNSFPRT
BLNSFDMD	BLNSFDMB	BLNSFLGD	BLNSFDVD	BLNSFTRM	BLNSFINT
BLNSFDMG	BLNSFDID	BLNSFLGD	BLNSFDVO	BLNSFVDF	BLNSFICB
BLNSFDMS	BLNSFCDS	BLNSFLGD	BLNSFLN1	BLNSFVDL	BLNSFTRM
BLNSFDSP	BLNSFDID	BLNSFLGD	BLNSFMSK	BLNSMBHX	BLNSMBSI
BLNSFDST	BLNSFASM	BLNSFLGD	BLNSFMSS	BLNSMBLB	BLNSMBLH
BLNSFDST	BLNSFCAR	BLNSFLGD	BLNSFOFL	BLNSMBLB	BLNSMLKC
BLNSFDST	BLNSFDAT	BLNSFLGD	BLNSFONL	BLNSMBLE	BLNSMBLL
BLNSFDST	BLNSFDMG	BLNSFLNK	BLNSFAS1	BLNSMBLF	BLNSMBLE
BLNSFDST	BLNSFDSP	BLNSFLN1	BLNSFLNK	BLNSMBLH	BLNSMBLF
BLNSFDST	BLNSFDVD	BLNSFLN1	BLNSFLOC	BLNSMBLK	BLNSMBSI

Called Module	Calling Module
BLNSMBLL	BLNSMSDC
BLNSMBLN	BLNSMBME
BLNSMBLO	BLNSMBLL
BLNSMBMA	BLNSMBML
BLNSMBME	BLNSMBMA
BLNSMBML	BLNSMINT
BLNSMBMN	BLNSMBME
BLNSMBOK	BLNSMBOL
BLNSMBOL	BLNSMBLO
BLNSMBOL	BLNSMLKP
BLNSMBPE	BLNSMBSI
BLNSMBSI	BLNSMSDC
BLNSMBTX	BLNSMBSI
BLNSMCML	BLNSMDSP
BLNSMCMO	BLNSMCML
BLNSMDSP	BLNFNSMC
BLNSMDSP	BLNSMFND
BLNSMDSP	BLNSMFN1
BLNSMDSP	BLNSMPCD
BLNSMFCD	BLNSMFES
BLNSMFC1	BLNSMFCD
BLNSMFC1	BLNSMFC1
BLNSMFES	BLNDVRCB
BLNSMFES	BLNSMCML
BLNSMFES	BLNSMSDC
BLNSMFES	BLNSMTRM
BLNSMFLD	BLNSMFES
BLNSMFND	BLNFNSMC
BLNSMFN1	BLNSMFND
BLNSMFRE	BLNSMHXA
BLNSMFTD	BLNSMFES
BLNSMFXD	BLNSMFES
BLNSMGRE	BLNSMHXA
BLNSMHXA	BLNSMBHX
BLNSMHXO	BLNSMBHX
BLNSMINT	BLNFNSMC
BLNSMLKB	BLNSMLKL
BLNSMLKC	BLNLBBCI
BLNSMLKC	BLNSMLKB
BLNSMLKD	BLNSMLKB
BLNSMLKG	BLNSMLKI
BLNSMLKI	BLNSMLKL
BLNSMLKL	BLNSMBLK
BLNSMLKN	BLNSMLKB
BLNSMLKN	BLNSMLKN
BLNSMLKO	BLNSMBLK
BLNSMLKP	BLNSMLKO
BLNSMLKP	BLNSMLKP
BLNSMLSL	BLNSMBLO
BLNSMLSL	BLNSMSNL
BLNSMLSP	BLNSMLSP
BLNSMLSP	BLNSMSNC
BLNSMOCL	BLNSMHXA
BLNSMPCD	BLNDDFPS
BLNSMPCD	BLNDDFPT
BLNSMPCD	BLNSMPRT
BLNSMPHP	BLNSMPCD
BLNSMPLA	BLNARPRT
BLNSMPLA	BLNSMPCD
BLNSMPRT	BLNFNSMC
BLNSMPSO	BLNSMBPE
BLNSMPSW	BLNSMHXA
BLNSMSDC	BLNSMDSP
BLNSMSEL	BLNFNSMC
BLNSMSEM	BLNSMSEL
BLNSMSNC	BLNSMSNI
BLNSMSNI	BLNSMSEL
BLNSMSNL	BLNSMSNI
BLNSMTRM	BLNFNSMC
BLNSMTXA	BLNSMBTX
BLNSMTXO	BLNSMBTX
BLNSUBMT	BLNOFFLD
BLNSUBMT	BLNONINT
BLNTERM	BLNMAIN
BLNUSADM	BLNDMUAI
BLNUSDDM	BLNDMUDI
CDP	BLNINPRT
CTA	BLNINPRT
DUMPACC	BLNADIFR
DUMPACC	BLNADIFT
EXITRTN	BLNAREXE
ISPLINK	BLNSCBKW
ISPLINK	BLNSCCCP
ISPLINK	BLNSCCRP
ISPLINK	BLNSCCSC
ISPLINK	BLNSCCTS
ISPLINK	BLNSCDIS
ISPLINK	BLNSCDRC
ISPLINK	BLNSCFWD
ISPLINK	BLNSCHDL
ISPLINK	BLNSCLFT
ISPLINK	BLNSCOTH
ISPLINK	BLNSCPRD
ISPLINK	BLNSCRGH
ISPLINK	BLNSCSUB
ISPLINK	BLNSFAMS
ISPLINK	BLNSFASM
ISPLINK	BLNSFAS1
ISPLINK	BLNSFBST
ISPLINK	BLNSFBUM
ISPLINK	BLNSFCAR
ISPLINK	BLNSFCA1
ISPLINK	BLNSFCDS
ISPLINK	BLNSFCSR
ISPLINK	BLNSFDAT
ISPLINK	BLNSFDD3
ISPLINK	BLNSFDMB
ISPLINK	BLNSFDMG
ISPLINK	BLNSFDST
ISPLINK	BLNSFFIN
ISPLINK	BLNSFFST
ISPLINK	BLNSFICB
ISPLINK	BLNSFINT
ISPLINK	BLNSFIPT
ISPLINK	BLNSFISP
ISPLINK	BLNSFLGD
ISPLINK	BLNSFLNK
ISPLINK	BLNSFLN1
ISPLINK	BLNSFLOC
ISPLINK	BLNSFMSS
ISPLINK	BLNSFSCM
ISPLINK	BLNSFTRM
ISPLINK	BLNSFVDF
ISPLINK	BLNSFVDL

DUMP ACCESS MODULE/REASON CROSS-REFERENCE

In the following list, Dump Access modules are listed alphabetically by module name. Each module name is followed by the reason code that the module issues, the name of the message in which the reason code is inserted, and a description of the reason code.

The symbols under reason code have the following meaning:

- LBR - Librarian reason code
- rtc - GETVIS/FREEVIS return code

Module Issuing	Reason Code	Message ID	Description
IJBXAALC	LBR	BLN9031I	
IJBXABLD	LBR	BLN9031I	
IJBXACMP	100	BLN9002I	Invalid dump record
IJBXADRV	012	BLN9002I	Invalid base address
IJBXADRV	016	BLN9002I	Invalid request type
IJBXADRV	104	BLN9002I	No storage available
IJBXAFND	112	BLN9002I	Member not found
IJBXAFND	LBR	BLN9031I	
IJBXAFVS	800+rtc	BLN9031I	Free storage error
IJBXAGAC	116	BLN9002I	Truncation occurred
IJBXAGAC	120	BLN9002I	Exceeds member
IJBXAGAC	LBR	BLN9031I	
IJBXAGET	109	BLN9002I	Truncation occurred
IJBXAGET	110	BLN9002I	Exceeds member
IJBXAGET	LBR	BLN9031I	
IJBXAGVS	700+rtc	BLN9002I	Get storage error
IJBXAINS	124	BLN9002I	Get storage error
IJBXAINT	150	BLN9002I	Initialization failure
IJBXALCN	130	BLN9002I	No sublibrary
IJBXALCN	134	BLN9002I	Sublibrary full
IJBXALCN	138	BLN9002I	Library connect error
IJBXALCN	LBR	BLN9031I	
IJBXALDC	LBR	BLN9031I	
IJBXALNB	128	BLN9002I	Invalid dump name
IJBXAMDC	LBR	BLN9031I	
IJBXANOT	LBR	BLN9031I	
IJBXAPAC	134	BLN9002I	Sublibrary full
IJBXAPAC	LBR	BLN9031I	
IJBXAPNT	112	BLN9002I	Member not found
IJBXAPNT	LBR	BLN9031I	
IJBXAPSQ	LBR	BLN9031I	

Module Issuing	Reason Code	Message ID	Description
IJBXAQDA	028	BLN9002I	Data not found
IJBXAQDA	004	BLN9002I	Invalid mode request
IJBXARDD	028	BLN9002I	Data not found
IJBXARDD	024	BLN9002I	Partial data returned
IJBXARDD	144	BLN9002I	Invalid length
IJBXARDD	148	BLN9002I	Invalid length
IJBXARDR	004	BLN9002I	Invalid mode request
IJBXARSP	008	BLN9002I	Invalid QUAL request
IJBXARSP	028	BLN9002I	Data not found
IJBXARSR	008	BLN9002I	Invalid QUAL request
IJBXARSR	028	BLN9002I	Data not found
IJBXARWR	004	BLN9002I	Invalid mode request
IJBXASTW	LBR	BLN9031I	
IJBXATSR	144	BLN9002I	Stow error
IJBXAWUP	140	BLN9002I	Invalid length
IJBXAWUP	028	BLN9002I	Data not found

DUMP ACCESS REASON/MODULE CROSS-REFERENCE

In the following list, Dump Access reason codes are listed alphabetically. Each reason code is followed by the name of the module that issues the reason code, the message in which it is inserted, and a description of the reason code. (Refer to "Dump Access Module/Reason Cross-Reference" on page 265 for additional information.)

Reason Code	Module Issuing	Message ID	Description
LBR	IJBXAALC	BLN9031I	
LBR	IJBXABLD	BLN9031I	
LBR	IJBXAFND	BLN9031I	
LBR	IJBXAGAC	BLN9031I	
LBR	IJBXAGET	BLN9031I	
LBR	IJBXALCN	BLN9031I	
LBR	IJBXALDC	BLN9031I	
LBR	IJBXAMDC	BLN9031I	
LBR	IJBXANOT	BLN9031I	
LBR	IJBXAPAC	BLN9031I	
LBR	IJBXAPNT	BLN9031I	
LBR	IJBXAPSQ	BLN9031I	
LBR	IJBXASTW	BLN9031I	
004	IJBXAQDA	BLN9002I	Invalid mode request
004	IJBXARDR	BLN9002I	Invalid mode request
004	IJBXARWR	BLN9002I	Invalid mode request

Reason Code	Module Issuing	Message ID	Description
008	IJBXARSP	BLN9002I	Invalid QUAL request
008	IJBXARSR	BLN9002I	Invalid QUAL request
012	IJBXADRV	BLN9002I	Invalid base address
016	IJBXADRV	BLN9002I	Invalid type request
024	IJBXARDD	BLN9002I	Partial data returned
028	IJBXAQDA	BLN9002I	Data not found
028	IJBXARDD	BLN9002I	Data not found
028	IJBXARSP	BLN9002I	Data not found
028	IJBXARSR	BLN9002I	Data not found
028	IJBXAWUP	BLN9002I	Data not found
100	IJBXACMP	BLN9002I	Invalid dump record
104	IJBXADRV	BLN9002I	No storage available
109	IJBXAGET	BLN9002I	Truncation occurred
110	IJBXAGET	BLN9002I	Exceeds member
112	IJBXAFND	BLN9002I	Member not found
112	IJBXAPNT	BLN9002I	Member not found
116	IJBXAGAC	BLN9002I	Truncation occurred
120	IJBXAGAC	BLN9002I	Exceeds member
124	IJBXAINS	BLN9002I	Get storage error
128	IJBXALNB	BLN9002I	Invalid dump name
130	IJBXALCN	BLN9002I	No sublibrary
134	IJBXALCN	BLN9002I	Sublibrary full
134	IJBXAPAC	BLN9002I	Sublibrary full
138	IJBXALCN	BLN9002I	Library connect error
140	IJBXAWUP	BLN9002I	Invalid length
144	IJBXARDD	BLN9002I	Invalid length
144	IJBXATSR	BLN9002I	Stow error
148	IJBXARDD	BLN9002I	Invalid length
150	IJBXAINT	BLN9002I	Initialization failure
700+rtc	IJBXAGVS	BLN9002I	Get storage error
800+rtc	IJBXAFVS	BLN9002I	Free storage error

DUMP ACCESS CALLING/CALLED MODULE CROSS-REFERENCE

This section lists, alphabetically, the names of Dump Access calling modules (modules that call other modules). Following each calling module name are the names of the modules that it calls. In addition, the following information on calling modules may be useful:

- Link book referenced:
  - IJBXALNK

Calling Module	Called Module	Calling Module	Called Module	Calling Module	Called Module
IJBXAALC	IJBXAFVS	IJBXAEMC	IJBXAMDC	IJBXAMES	IJBXAGMP
IJBXAALC	IJBXAGVS	IJBXAEMC	IJBXAPAC	IJBXAMES	IJBXASME
IJBXAALC	IJBXALDC	IJBXAEMC	IJBXASTW	IJBXANOT	
IJBXAALC	IJBXAMDC	IJBXAFND	IJBXAFVS	IJBXAPAC	
IJBXAAMP	IJBXAALC	IJBXAFND	IJBXAGVS	IJBXAPNT	IJBXAFVS
IJBXAAMP	IJBXACMP	IJBXAFND	IJBXALDC	IJBXAPNT	IJBXAGVS
IJBXAAMP	IJBXAFND	IJBXAFND	IJBXAMDC	IJBXAPNT	IJBXALDC
IJBXAAMP	IJBXAFVS	IJBXAFVS		IJBXAPNT	IJBXAMDC
IJBXAAMP	IJBXAGET	IJBXAGAC		IJBXAPSQ	
IJBXAAMP	IJBXALDC	IJBXAGET	IJBXANOT	IJBXAQDA	IJBXAINT
IJBXAAMP	IJBXAMDC	IJBXAGET		IJBXARBM	
IJBXAAMP	IJBXASTW	IJBXAGMA		IJBXARBS	
IJBXAAMP	IJBXAWMR	IJBXAGMP	IJBXAGVS	IJBXARBU	IJBXAGAC
IJBXAAVL	IJBXABLD	IJBXAGVS		IJBXARBU	IJBXARBM
IJBXAAVL	IJBXAFVS	IJBXAIND	IJBXALCN	IJBXARBU	IJBXARBS
IJBXAAVL	IJBXAGVS	IJBXAIND	IJBXALNB	IJBXARDA	IJBXAMAP
IJBXAAVL	IJBXALCN	IJBXAIND	IJBXAMCN	IJBXARDA	IJBXARBU
IJBXAAVL	IJBXALDC	IJBXAINE	IJBXACMP	IJBXARDD	IJBXARDA
IJBXAAVL	IJBXALNB	IJBXAINE	IJBXAFND	IJBXARDR	IJBXARDD
IJBXAAVL	IJBXAMDC	IJBXAINE	IJBXAGET	IJBXARDR	IJBXARSP
IJBXABLD		IJBXAINE	IJBXAGVS	IJBXARDR	IJBXARSR
IJBXACMP	IJBXAMES	IJBXAINF	IJBXACMP	IJBXARMP	IJBXAGET
IJBXACMP		IJBXAINF	IJBXAGET	IJBXARMP	IJBXAGVS
IJBXADEL	IJBXAFND	IJBXAINF	IJBXAGVS	IJBXARSP	IJBXARBU
IJBXADEL	IJBXAFVS	IJBXAINF	IJBXAPNT	IJBXARSR	IJBXARBU
IJBXADEL	IJBXAINT	IJBXAINM	IJBXAAMP	IJBXARWR	IJBXAINT
IJBXADEL	IJBXALCN	IJBXAINM	IJBXAFND	IJBXARWR	IJBXARDR
IJBXADEL	IJBXALDC	IJBXAINM	IJBXARMP	IJBXARWR	IJBXAWAP
IJBXADEL	IJBXALNB	IJBXAINS	IJBXAGVS	IJBXARWR	IJBXAWUP
IJBXADEL	IJBXAMCN	IJBXAINT	IJBXAIND	IJBXASME	
IJBXADEL	IJBXAMDC	IJBXAINT	IJBXAINE	IJBXASTW	
IJBXADEL	IJBXASTW	IJBXAINT	IJBXAINF	IJBXATDD	IJBXAFVS
IJBXADRV	IJBXAAVL	IJBXAINT	IJBXAINM	IJBXATDD	IJBXALDC
IJBXADRV	IJBXADEL	IJBXAINT	IJBXAINS	IJBXATDD	IJBXAMDC
IJBXADRV	IJBXAGMA	IJBXALCN	IJBXAFVS	IJBXATDD	IJBXASTW
IJBXADRV	IJBXAGVS	IJBXALCN	IJBXAGVS	IJBXATDD	IJBXATMD
IJBXADRV	IJBXAINT	IJBXALDC		IJBXATMD	IJBXAFVS
IJVXADRV	IJBXAQDA	IJBXALNB		IJBXATMD	IJBXALDC
IJVXADRV	IJBXARWR	IJBXAMAP		IJBXATMD	IJBXAMDC
IJVXADRV	IJBXATRM	IJBXAMCN	IJBXAFND	IJBXATMD	IJBXASTW
IJBXAEMC	IJBXAALC	IJBXAMCN	IJBXAFVS	IJBXATMD	IJBXAWMR
IJBXAEMC	IJBXAFVS	IJBXAMCN	IJBXAGET	IJBXATRM	IJBXATDD
IJBXAEMC	IJBXAINE	IJBXAMCN	IJBXAGVS	IJBXATRM	IJBXATSR
IJBXAEMC	IJBXALDC	IJBXAMDC		IJBXATSR	IJBXAFVS

Calling Module	Called Module
IJBXAWAP	IJBXACMP
IJBXAWAP	IJBXAEMC
IJBXAWAP	IJBXAPAC
IJBXAWAP	IJBXAWNM
IJBXAWDR	IJBXAFVS
IJBXAWDR	IJBXAPAC
IJBXAWMR	IJBXAFVS
IJBXAWMR	IJBXAPSQ
IJBXAWMR	IJBXASME
IJBXAWNM	IJBXAALC
IJBXAWNM	IJBXACMP
IJBXAWNM	IJBXAGVS
IJBXAWNM	IJBXAPAC
IJBXAWUP	IJBXAGVS
IJBXAWUP	IJBXARDA
IJBXAWUP	IJBXAWDR

DUMP ACCESS CALLED/CALLING MODULE CROSS-REFERENCE

This section lists, alphabetically, the names of Dump Access modules that are called by other modules. Following each called module name are the names of the modules that call it.
(Refer to "Dump Access Calling/Called Module Cross-Reference" on page 267 for additional information.)

Called Module	Calling Module
IJBXAALC	IJBXAAMP
IJBXAALC	IJBXAEMC
IJBXAALC	IJBXAWNM
IJBXAAMP	IJBXAINM
IJBXAAVL	IJBXADRV
IJBXABLD	IJBXAAVL
IJBXACMP	IJBXAAMP
IJBXACMP	IJBXAINE
IJBXACMP	IJBXAINF
IJBXACMP	IJBXAWAP
IJBXACMP	IJBXAWNM
IJBXADEL	IJBXADRV
IJBXADRV	
IJBXAEMC	IJBXAWAP
IJBXAFND	IJBXAAMP
IJBXAFND	IJBXADEL
IJBXAFND	IJBXAINE
IJBXAFND	IJBXAINM
IJBXAFND	IJBXAMCN
IJBXAFVS	IJBXAALC
IJBXAFVS	IJBXAAMP
IJBXAFVS	IJBXAAVL
IJBXAFVS	IJBXADEL
IJBXAFVS	IJBXAEMC
IJBXAFVS	IJBXAFND
IJBXAFVS	IJBXALCN
IJBXAFVS	IJBXAMCN
IJBXAFVS	IJBXAPNT
IJBXAFVS	IJBXATDD
IJBXAFVS	IJBXATMD
IJBXAFVS	IJBXATSR
IJBXAFVS	IJBXAWDR
IJBXAFVS	IJBXAWMR
IJBXAGAC	IJBXARBU
IJBXAGET	IJBXAAMP
IJBXAGET	IJBXAINE
IJBXAGET	IJBXAINF
IJBXAGET	IJBXAMCN
IJBXAGET	IJBXARMP
IJBXAGMA	IJBXADRV
IJBXAGMP	IJBXAMES
IJBXAGVS	IJBXAALC
IJBXAGVS	IJBXAAVL
IJBXAGVS	IJBXADRV
IJBXAGVS	IJBXAFND
IJBXAGVS	IJBXAGMP
IJBXAGVS	IJBXAINE
IJBXAGVS	IJBXAINF

Called Module	Calling Module
IJBXAGVS	IJBXAINS
IJBXAGVS	IJBXALCN
IJBXAGVS	IJBXAMCN
IJBXAGVS	IJBXAPNT
IJBXAGVS	IJBXARMP
IJBXAGVS	IJBXAWNM
IJBXAGVS	IJBXAWUP
IJBXAIND	IJBXAINT
IJBXAINE	IJBXAEMC
IJBXAINE	IJBXAINT
IJBXAINF	IJBXAINT
IJBXAINM	IJBXAINT
IJBXAINS	IJBXAINT
IJBXAINT	IJBXADEL
IJBXAINT	IJBXADRV
IJBXAINT	IJBXAQDA
IJBXAINT	IJBXADEL
IJBXAINT	IJBXADRV
IJBXAINT	IJBXAQDA
IJBXAINT	IJBXARWR
IJBXALCN	IJBXAAVL
IJBXALCN	IJBXADEL
IJBXALCN	IJBXAIND
IJBXALDC	IJBXAALC
IJBXALDC	IJBXAAMP
IJBXALDC	IJBXAAVL
IJBXALDC	IJBXADEL
IJBXALDC	IJBXAEMC
IJBXALDC	IJBXAFND
IJBXALDC	IJBXAPNT
IJBXALDC	IJBXATDD
IJBXALDC	IJBXATMD
IJBXALNB	IJBXAAVL
IJBXALNB	IJBXADEL
IJBXALNB	IJBXAIND
IJBXAMAP	IJBXARDA
IJBXAMCN	IJBXADEL
IJBXAMCN	IJBXAIND
IJBXAMDC	IJBXAALC
IJBXAMDC	IJBXAAMP
IJBXAMDC	IJBXAAVL
IJBXAMDC	IJBXADEL
IJBXAMDC	IJBXAEMC
IJBXAMDC	IJBXAFND
IJBXAMDC	IJBXAPNT
IJBXAMDC	IJBXATDD
IJBXAMDC	IJBXATMD
IJBXAMES	IJBXACMP
IJBXANOT	IJBXAGET
IJBXAPAC	IJBXAEMC
IJBXAPAC	IJBXAWAP
IJBXAPAC	IJBXAWDR
IJBXAPAC	IJBXAWNM
IJBXAPNT	IJBXAINF
IJBXAPSQ	IJBXAWMR
IJBXAQDA	IJBXADRV
IJBXARBM	IJBXARBU
IJBXARBS	IJBXARBU
IJBXARBU	IJBXARDA
IJBXARBU	IJBXARSP
IJBXARBU	IJBXARSR
IJBXARDA	IJBXARDD
IJBXARDA	IJBXAWUP
IJBXARDD	IJBXARDR
IJBXARDR	IJBXARWR
IJBXARMP	IJBXAINM
IJBXARSP	IJBXARDR
IJBXARSR	IJBXARDR
IJBXARWR	IJBXADRV
IJBXASME	IJBXAMES
IJBXASME	IJBXAWMR
IJBXASTW	IJBXAAMP
IJBXASTW	IJBXADEL
IJBXASTW	IJBXAEMC
IJBXASTW	IJBXATDD
IJBXASTW	IJBXATMD
IJBXATDD	IJBXATRM
IJBXATMD	IJBXATDD
IJBXATRM	IJBXADRV
IJBXATSR	IJBXATRM
IJBXAWAP	IJBXARWR
IJBXAWDR	IJBXAWUP
IJBXAWMR	IJBXAAMP
IJBXAWMR	IJBXATMD
IJBXAWNM	IJBXAWAP
IJBXAWUP	IJBXARWR

PARSER: INTRODUCTION

The parser (phase name IJPARSER), a table-driven routine, checks the validity of all items in a given command. It does this by first checking the verb against a command directory. If the verb is valid, the parser checks the specified operands against a syntax table ("Command Table") of all possible keyword or positional parameters associated with that verb. Any command syntax item that the parser finds to be valid is changed by the parser to a standardized internal format (SIF) for further processing by one of various VSE/Advanced Functions components, such as SDAID, which presented the command.

A syntactic error in any item of the command - a parameter, delimiter, or separator - results in an error message which is passed to the calling program's message routine. The operator may usually reenter the corrected command line.

The parser is capable of analyzing any command that conforms to syntax rules for parameters as follows:

- Use parameters that represent values, one per parameter. Examples:

  ```
  TRACE BR
  TRACE PAGING
  ```

  BR and PAGING represent values for further processing.

- Use keywords that indicate the type of value which is being specified. Example:

  ```
  TRACE STORAGE PATT=E2C4C1C9C4
  ```

  PATT is a keyword which indicates a pattern value, and E2C4C1C9C4 is the value specified by the user.

- Indicate a list by using parentheses when more than one value is specified for a keyword. Example:

  ```
  TRACE SVC=(0,1C 2A)
  ```

  Specification 0,1C 2A enclosed in parentheses is a list of values.

- Ensure that alternatives on the same parameter level are unique.

- Enter range specifications in the form value :value.

- Indicate a continuation on the subsequent line of input by a dash (-) preceded by a delimiter and followed by a blank in the current line.

- Use a semicolon as a command-end indicator if the next command is to be started on the same line.

- Separate parameters (those representing a value as well as those using a keyword) from each other by a valid delimiter: a comma, one or more blanks, or a comma plus one or more blanks.

- Separate values within a list from each other by a valid delimiter.

- Enclose alphameric character strings within a pair of apostrophes. Quoted strings may also have special characters.

Keyword prompting may be used for entering the operands of a command. The parser originated messages prompt the user for the required control information.

ENVIRONMENT AND INVOCATION

For each command to be analyzed, the parser needs a command table (CT) in virtual storage. The CT must be available to the parser when it receives control. The program calling the parser must also furnish certain information (as well as receive output) via the parser control block; for example:

- Command input area pointer,

- Output area pointer (for messages),

- Command Directory pointer,

- (if provided), check exit and I/O interface routine addresses,

- SIF and value entries from processed commands.

The parser, although documented as an independent set of routines, functions only as part of the program which uses the parser's services. The complete parser control block contents are listed in the Data Area section, later in this chapter. For register conventions, see the following section.

PARSER: PROGRAM DESIGN AND ORGANIZATION INFORMATION

The general flow of control for command checking by the parser is illustrated in Figure 9 on page 275. In addition to the entry routine Parser (phase IJPARSER), subroutines are included in the following modules:

Module IJPANXTP: Subroutine NEXTPARM
IJPADIRS: DIRSERV
IJPAVALD: VALIDITY
IJPADEFT: DEFAULT
IJPASIFN: SIFENTRY
IJPAPMSG: PMSGLIB

The purpose of each is given in Figure 9 on page 275. A more detailed diagram of each is given later in this chapter.

COMMAND STRUCTURE AND THE COMMAND TABLE

The Command Table (CT) associated with each command verb indicates the required structure of the command; that is, the required sequence of keyword and positional parameters, in addition to other information. The command table formats the storage area (the size depends on each individual command) into four distinct areas:

Header Area
Node Area
Parameter Area
Value Area

- The Header area contains, for instance:
  - the command verb, abbreviation and number
  - environment requirements, mode flags, check exit information
  - pointers to associated Node, Parameter, and Value areas

- The Node area contains, for instance:
  - a block of pointers for each parameter that may occur (see below)
- Parameter area contains, for instance:
  - parameter number, flags, attributes, repetition range
  - pointers to prompting text, to default values, to error messages
- The Value area contains, for instance:
  - texts of prompting and error messages

For more details, see Figure 11 on page 279.

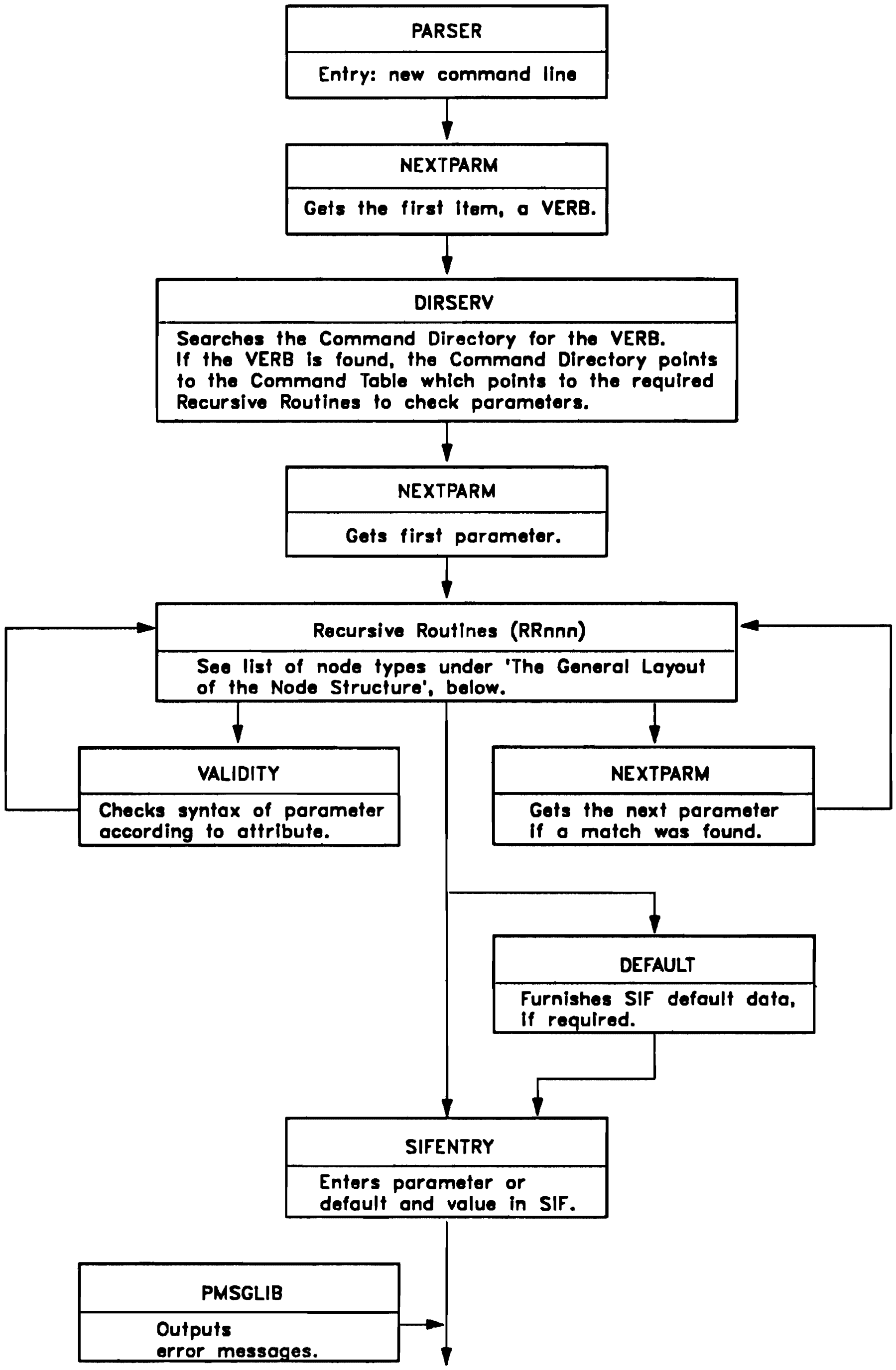

PARSER
Entry: new command line
NEXTPARM
Gets the first item, a VERB.
DIRSERV
Searches the Command Directory for the VERB. If the VERB is found, the Command Directory points to the Command Table which points to the required Recursive Routines to check parameters.
NEXTPARM
Gets first parameter.
Recursive Routines (RRnnn)
See list of node types under 'The General Layout of the Node Structure', below.
VALIDITY
Checks syntax of parameter according to attribute.
NEXTPARM
Gets the next parameter if a match was found.
DEFAULT
Furnishes SIF default data, if required.
SIFENTRY
Enters parameter or default and value in SIF.
PMSGLIB
Outputs error messages.

Figure 9. General Flow of Control for Command Checking

The General Layout of the Node Structure

The node structure part of the CT contains entries which represent the structure of the command. These entries (nodes) are used to establish the processing sequence of a given command. A node is of fixed format and contains the following fields:

1. Node type
2. Offset for parameter description
3. Same level pointer
4. Lower level pointer

0	type
2	param ptr
4	SL ptr
6	LL ptr

Node Type	Abbreviation	SIF Entry
Positional list	RRPLIST	no
Keyword list	RRKLIST	no
Command termination	RRTERM	no
Positional node	RRPOSN	yes
Positional - or	RRPOR	no
Positional alternative	RRPALT	yes
Positional repetition	RRPREP	yes
Positional alternative with repetition	RRPAREP	yes
Keyword - or	RRKOR	no
Keyword alternative	RRKALT	yes
Keyword	RRKEYW	yes
Single keyword	RRSNGL	yes
Keyword repetition	RRKREP	no

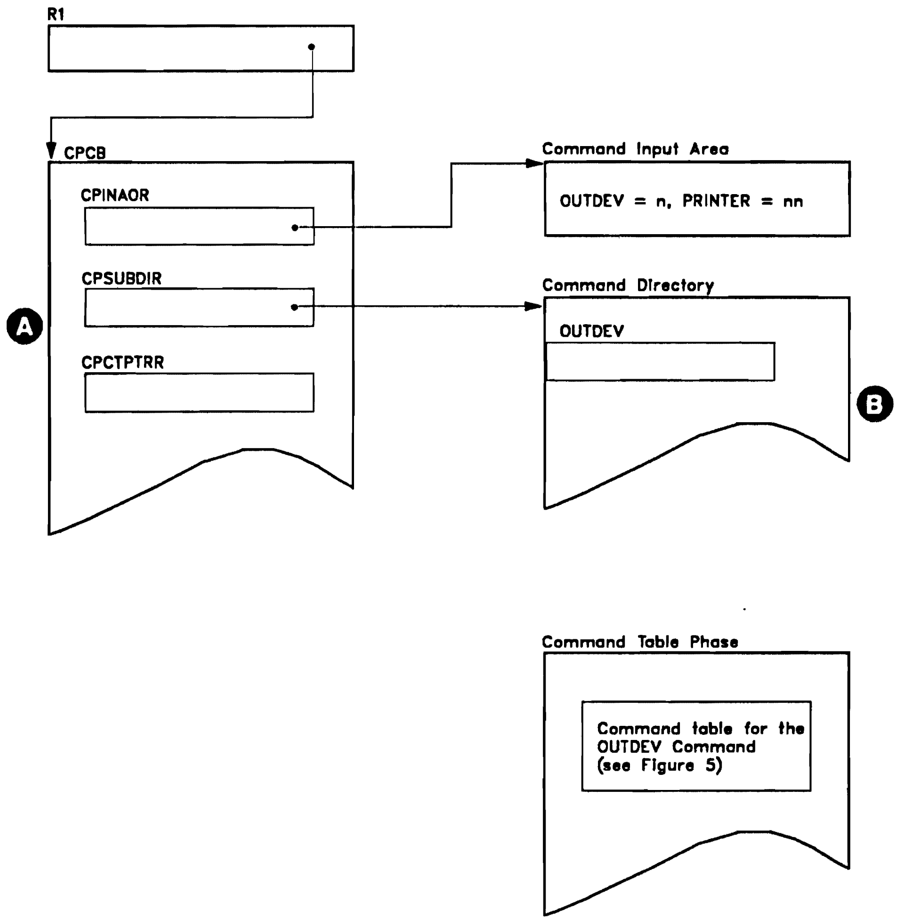
The node type name; i.e., KALT, is the entry point name of the routine RRKALT handling this parameter (keyword alternative). The offset to the parameter; i.e., P1, (see Figure 11 on page 279) is the label of the parameter definition in the command table. P3, for example, describes the parameter PRINTER. The same level and lower level pointers define the logical connection between the various parameters. Alternatives are identified by an alternative number i.e. P3 A1 identifies the keyword printer, P3 A2 the keyword TAPE, and so on.

To illustrate the logic flow in the parser, the OUTDEV command from the SDAID program has the following syntax:

```
OUTDEV ──┬──────────┬──┬─Printer=cuu─┬──;
         │          │  │             │
         └─Buffer=n─┘  └─Tape=cuu────┘
```

For an explanation of the syntax, see the manual VSE/Advanced Functions Diagnosis: Service Aids.

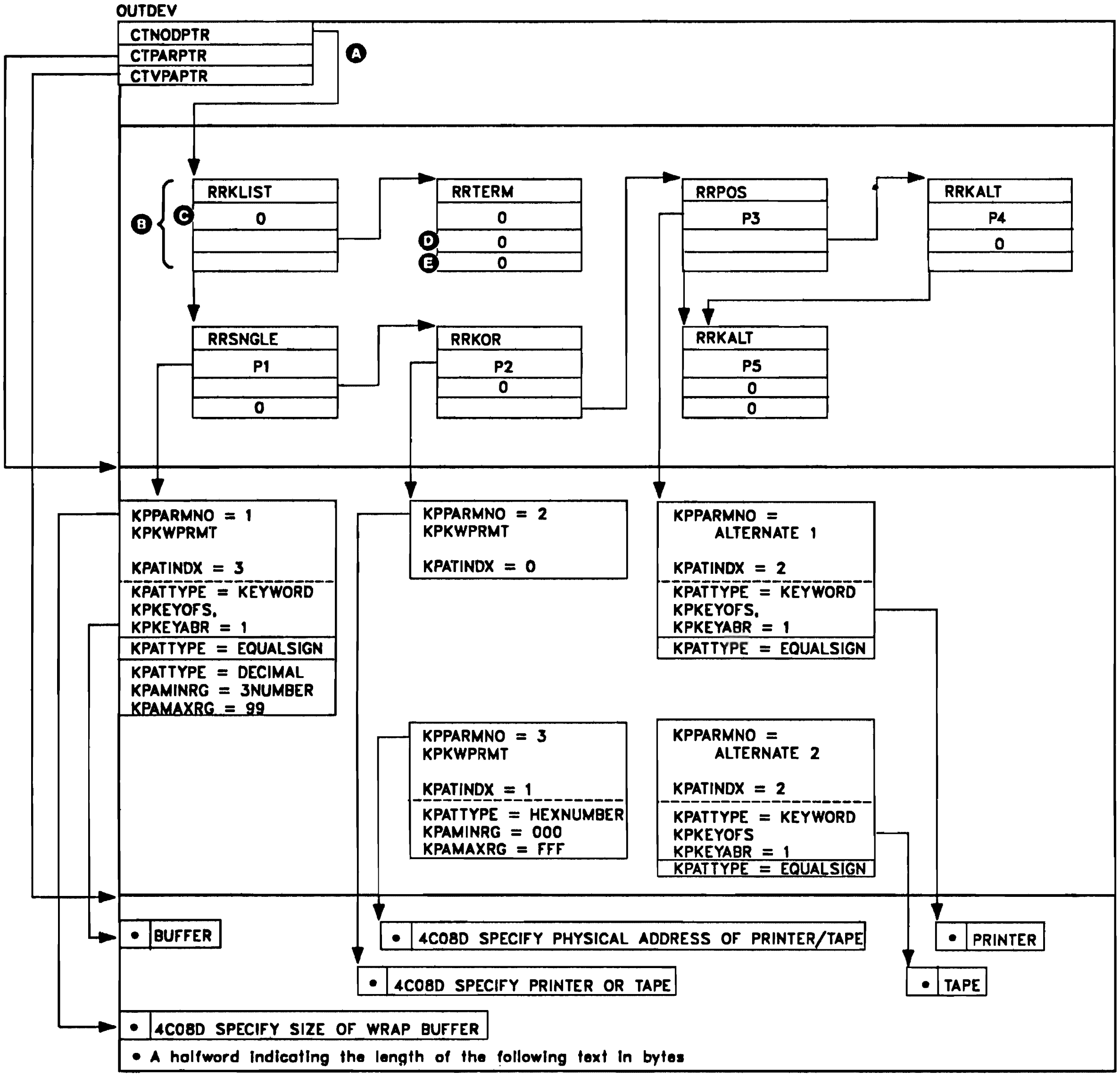
The tables used by the parser for analyzing a command are made available through register 1, which contains a pointer to the Command Processor Control Block (CPCB). This is illustrated in Figure 12 on page 305. Figure 11 on page 279, an expansion of the command table shown in Figure 10 on page 278 illustrates how the parser makes use of recursive routines to analyze the various parameters of a given command; in this case, the OUTDEV command from the SDAID program. This is further illustrated by examples in the following section.



A Command processing control block. Fields of the block via dummy section CPCBGENP.

B The entry for the command indicates the offset into the command table phase.

Figure 10. Table and Control Block Structure when the Parser Receives Control.



A Fields within the command table header are interpreted via the dummy section CTHEADER.

B Parameter nodes. Fields of these nodes are interpreted via the dummy section CTSTRE.

C CTPARMOF — Parameter entry offset. Together with the address in the header field CTPARPTR.

D CTSAMELV — Same level pointer. It points to the next parameter node of same level in the list of nodes for the command currently being processed.

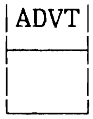
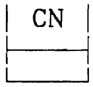
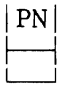
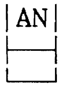
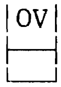
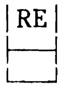
E CTLOVELV — Lower level pointer. It points to the first or only parameter at the next or only lower level of nodes for the command currently being processed.

Figure 11. Control Flow Taken by the Parser for the OUTDEV (SDAID) Command

LOGIC FLOW OF THE PARSER

SIF Construction

The SIF consists of a fixed format string, called Standard Internal Text (SIT) and a related table with all variable length information, called Value Table (VT). The SIT contains an entry for every parameter in the command. An entry is composed of two or three fields. The following field types are used:

Field	Content	Representation
ADVT	Address of Value Table	
CN	Command Number	
PN	Parameter Number	
AN	Alternative Number	
OV	Offset into Value Table	
RE	Repetition Count	

The SIT starts with address of the value table and the command number. The command number is unique within the system. It consists of an identifier for the component (2 bytes) followed by a sequence number (2 bytes). The parameters of a command are identified by a parameter number. The parameter numbers are unique within one command only. They are entered into the PN field while the SIT is under construction. Alternatives of a parameter are identified by an alternative number. These numbers are unique within one parameter only. The fields contained in a SIT entry for a specific parameter depend on the parameter type. The following entries are generated:

Parameter Type	Fields	Representation
- single keyword parameter (RRSNGL)	PN,OV	│PN│OV│
- positional parm w/o alternative (RRPOS)		
- keyword alternatives (RRKALT)	PN,AN	│PN│AN│
- positional alternatives (RRPALT)	PN,AN,OV	│PN│AN│OV│
-- positional repetition (RRPREP)	PN,RE,OV	│PN│RE│OV│
- positional alternative with repetition (RRPAREP)	PN,AN,RE,OV	│PN│AN│RE│OV│

SIT entries for keyword repetitions are built by repetitively using the entries defined above according to the parameter type. The VT is constructed from all variable information entered with the command. Variable information may be identifiers, strings, hexadecimal or decimal numbers; etc. To find the variable information belonging to a parameter, the address of the VT (ADVT field in the SIT) and the offset (OV field in the SIT) are added. The result is a pointer to the appropriate VT entry. A value table entry consists of the following fields:

Field	Content	Representation
LE	Length of Value	LE
VALUE	Variable Value	VALUE

After addressability to the first node (in the example RRKLIST) has been established, control is given up as defined by the type field; i.e., RRKLIST, RRSNGL. After one node is completed, the path defined by the lower level pointer is taken and after that the path defined by the same level pointer. (The only exceptions are the alternative nodes RRKALT, RRPALT, and RRPAREP. Here control is passed first to the same level, then to the lower level pointer.)

Examples:

These examples show the flow within the parser and the SIF produced from the given commands.

Note: The asterisk (*) and arrow (--->) indicate the nodes which enter information into the SIF and what is entered.

1. OUTDEV   BUFFER=5   PRINTER=00E;

```
Parser ENTRY
RRKLIST
RRSNGL*---------> Parameter Number, Offset into Value Table
RRKOR
RRKALT*---------> Parameter Number, Alternate Number
RRPOS*----------> Parameter Number, Offset into Value Table
RRTERM
```

SIT

Addr of VT		PN	OV	PN	AN	PN	OV
.	CMD NUM	P1	0	P2	A1	P3	3

VT

LE		Value	LE		Value		
0	1	5	0	3	0	0	E

2. OUTDEV BUF=22 T=230;

```
Parser ENTRY
RRKLIST
RRSNGL*---------> Parameter Number, Offset into VT
RRKOR
RRKALT
RRKALT*---------> Parameter Number, Alternate Number
RRPOS*----------> Parameter Number, Offset into VT
RRTERM
```

SIT

Addr of VT		PN	OV	PN	AN	PN	OV
.	CMD NUM	P1	0	P2	A2	P3	4

VT

LE		Value		LE		Value		
0	2	2	2	0	3	2	3	0

```
3.  OUTDEV   PR=1FE;
    Parser ENTRY
    RRKLIST
    RRSNGL
    RRKOR
    RRKALT*-----------> Parameter Number, Alternate Number
    RRPOS*------------> Parameter Number, Offset into VT
    RRTERM
```

SIT

Addr of VT		PN	AN	PN	OV
.	CMD NUM	P2	A1	P3	0

VT

LE		Value		
0	3	1	F	E

Note: P is a parameter number, A is an alternative number, and each square represents one byte.

PARSER: DATA AREA INFORMATION

COMMAND TABLE HEADER STRUCTURE

```
 2   CTNODPTR PTR(15),       00     ADDRESS OF NODE STRUCTURE
 2   CTPARPTR PTR(15),       02     ADDRESS OF PARAMETERS
 2   CTVPAPTR PTR(15),       04     ADDRESS OF VALUE PART

 2   CTCOMAND CHAR(12),      06     COMMAND NAME
 2   CTABBREV CHAR(6),       12     COMMAND NAME ABBREVIATION
 2   CTCMDNR  CHAR(4),       18     COMMAND NUMBER

 2   CTENVLST,               1C     ENVIRONMENT CHECK
  3   CTENVFLG   PTR(32),    1C     ENVIRONMENT ATTRIBUTE FLAG
  3   CTENVATR   PTR(32),    20     ENVIRONMENT CHECK ATTRIBUTE
   4   CTENVMTS  PTR(16),    20     ENVIRONMENT SPECIFICATION FLAGS

 2   CTCEPEXI ,              24     Reserved for future use
  3   CTCEPFLG   PTR(32),    24
  3   CTCEPATR   PTR(31),    28
   4   CTCEPRTC  PTR(8),     28
   4   CTCEPID   PTR(8),     29
   4   CTCEPPHS  FIXED(15),  2A
    5   CTCEPADR FIXED(15),  2A
     6  CTCEPOFF FIXED(15),  2A
      8 CTCEPRID FIXED(8),   2A

 2   CTCHKEXI,               2C     CHECK ROUTING INFO FOR COMMAND
  3   CTCHKFLG   PTR(32),    2C     CHECK ATTRIBUTE FLAG
  3   CTCHKATR   PTR(31),    30     CHECK ROUTING ATTRIBUTE
   4   CTCHKRTC  PTR(8),     30     ROUTE CODE
   4   CTCHKID   PTR(8),     31     IDENTIFICATION BYTE
   4   CTCHKPHS  FIXED(15),  32     OFFSET TO PHASE NAME
    5   CTCHKADR FIXED(15),  32     OFFSET TO BRANCH ADDRESS
     6  CTCHKOFF FIXED(15),  32     POINT OFFSET OF ROUTINE

 2   CTMODE     BIT(16),     34     MODE SPECIFICATION
  4    CTNEWM   BIT(1),      34     ON IF NEW MODE
  4    CTCURM   BIT(1),      34     ON IF CURRENT MODE TERMINATES
  4    CTGDVAL  BIT(1),      34     ON IF COMMAND GENERAL VALID
  4    CTMDVAL  BIT(1),      34     ON IF COMMAND
                                    IN MODE DIRECTORY VALID
  4    CTSYNCH  BIT(1),      34     ON IF SYNTAX-CHECK
  4    CTNISUB  BIT(1),      34     ON IF NOT IN SUBSET
```

```
    4    CTSKIP   BIT(1),      34     ON IF SKIP
    4    CTDATA   BIT(1),      34     ON IF DATA
    4    CTCOLMN  BIT(1);      34     ON IF COLUMN
```

COMMAND TABLE NODE STRUCTURE ENTRY

```
   2   CTTYPE  BIT(16),        00     NODE TYPE
    4  CTCNTL  BIT(8),         00     CONTROL FIELD
     6 CTSIF   BIT(1),         00     ON FOR SIF ENTRY
     6 CTNOSUB BIT(1),         00     ON FOR NOSUBSET
    4  CTTYP   BIT(8),         01     NODE TYPE
   2   CTPARMOF PTR(15),       02     PARAMETER ENTRY OFFSET
   2   CTSAMELV PTR(15),       04     OFFSET SAME LEVEL
   2   CTLOWELV PTR(15);       06     OFFSET LOWER LEVEL

 DCL PLIST  BIT(8) CONSTANT('00'X),        POSITIONAL LIST
     KLIST  BIT(8) CONSTANT('04'X),        KEYWORD LIST
     TERM   BIT(8) CONSTANT('08'X),        COMMAND TERMINATION
     POSN   BIT(8) CONSTANT('0C'X),        POSIT. NODE
     POR    BIT(8) CONSTANT('10'X),        POSITIONAL - OR
     PALT   BIT(8) CONSTANT('14'X),        POSITIONAL ALTERNATIVE
     PREP   BIT(8) CONSTANT('18'X),        POSITIONAL REPETITION
     PAREP  BIT(8) CONSTANT('1C'X),        POS. ALT. WITH REPETITION
     KOR    BIT(8) CONSTANT('20'X),        KEYWORD - OR
     KALT   BIT(8) CONSTANT('24'X),        KEYWORD ALTERNATIVE
     KEYW   BIT(8) CONSTANT('28'X),        KEYWORD
     SNGL   BIT(8) CONSTANT('2C'X),        SINGLE KEYWORD
     KREP   BIT(8) CONSTANT('30'X);        KEYWORD REPETITION
```

THIS STRUCTURE DEFINES THE LAYOUT OF THE KEYWORD AND POSITIONAL PARAMETERS OF THE PARSER

```
 2      KPPARMNO  FIXED(15),     00 PARAMETER NUMBER DEFINED BY CC
 2      KPEXCNTL  PTR  (16),     02 NODE CONTROL FLAGS
  4      KPDFLTCD PTR  (8),      02 DEFAULT ID
  4      KPCEP    BIT  (1),      02 ON IF CEP EXIT IN CURRENT COPY
  4      KPCHK    BIT  (1),      02 ON IF CHECK EXIT IN CURRENT COPY
  4      KPSTPOS  BIT  (1),      02 ON IF SIT POSITION IN CURRENT COPY
  4      KPERR    BIT  (1),      02 ON IF ERROR MSG DEFINED IN KPERRMSG
  4      KPPRMTFL BIT  (1),      02 ON IF PROMPT ANSWER DEFINED

 2      KPPROMPT  CHAR (6) BDY(HWORD),      PROMPT INFORMATION
  4      KPKWPRMT FIXED(15),     04 OFFSET TO KEYWORD PROMPT MESSAGE
 4       KPFLPRMT FIXED(15),     06 OFFSET TO FULL PROMPT MESSAGE
 4       KPPRMTOF FIXED(15),     08 OFFSET TO PROMPT ANSWER

2        KPDEFOFS FIXED(15),     0A OFFSET TO DEFAULT VALUE

2       KPREPRNG  FIXED(31),     0C REPETION RANGE
 4       KPMINRNG FIXED(15),     0C MINIMUM RANGE IF NOT PRESENT MIN=1
  6      KPREPID  PTR  (8),      0C DEFINES EXISTENCE OR *
 4       KPMAXRNG FIXED(15),     0E MAXIMUM RANGE MUST BE DEFINED

2       KPERRMSG  FIXED(15),     10 OFFSET TO ERROR MESSAGE IF PRESENT

2       KPATINDX  FIXED(15),     12 INDEX DEFINES MAX VECTOR ENTRIES

2       KPCTPEND  CHAR (0);      14 END INDICATOR
```

STRUCTURE DEFINES THE ATTRIBUTE PART OF THE PARAMETER

```
1    KPATRIBT(*) BASED(ADDR(KPCTPEND)),
                                    THE FOLLOWING DATA IS USED IN THE
                                    VALIDITY ROUTINE
 2     KPATRID  PTR  (32),       14 ATTRIBUTE IDENTIFICATION FLAGS
  4     KPATTYPE PTR(8),         14 ATTRIBUTE TYPE
  4     KPATFLGS PTR(8),         15 ATTRIBUTE FLAGS
   6     KPPAIR   BIT  (1),      15 ON IF PAIR
   6     KPCONCAT BIT  (1),      15 ON IF CONCAT
   6     KPCOMPNT BIT  (1),      15 ON IF MSHP COMPONENT FEATURE NUMBER
   6     *        BIT  (3),      15 RESERVED
   6     KPVPOF1  BIT  (1),      15 ON IF 1ST HW IN KPPARM CONTAINS
                                     OFFSET IN VALUE PART
   6     KPVPOF2  BIT  (1),      15 ON IF 2ND HW IN KPPARM CONTAINS
                                     OFFSET IN VALUE PART
  4     KPRESRVD PTR(15),        16 RESERVED
 2     KPPARM   PTR  (32),       18 OVERLAY AREA
 2     KPDUMEND CHAR (0);        1C END INDICATOR OF ATTRIBUTE LIST

DCL  KPNULL   BIT (8) CONSTANT('00'X),     DUMMY ATTRIBUTE
     KPKEYWRD BIT (8) CONSTANT('04'X),     KEYWORD
     KPLITERL BIT (8) CONSTANT('08'X),     LITERAL
     KPEQUAL  BIT (8) CONSTANT('0C'X),     ITEM '=' SIGN
     KPIDENT  BIT (8) CONSTANT('10'X),     IDENTIFIER
     KPALPHA  BIT (8) CONSTANT('14'X),     ALPHA CHAR STRING
     KPSTRING BIT (8) CONSTANT('18'X),     STRING
     KPQUOTST BIT (8) CONSTANT('1C'X),     QUOTED STRING
     KPBITSTR BIT (8) CONSTANT('20'X),     BIT STRING
     KPDECNUM BIT (8) CONSTANT('24'X),     DECIMAL NUMBER
     KPHEXNUM BIT (8) CONSTANT('28'X),     HEX NUMBER
     KPCHKEXT BIT (8) CONSTANT('2C'X),     CHECK EXIT
     KPCEPEXT BIT (8) CONSTANT('30'X),     CEP EXIT
     KPENV    BIT (8) CONSTANT('34'X),     ENVIRONMENT ATTRIBUTE
     KPSEP    BIT (8) CONSTANT('38'X),     SEPARATOR
     KPLSKIP  BIT (8) CONSTANT('3C'X),     LINE SKIP
     KPSYSKIP BIT (8) CONSTANT('40'X),     SYSIN | SYSRDR SKIP
     KPCOLUMN BIT (8) CONSTANT('44'X),     COLUMN SPECIFIED
     KPSITPOS BIT (8) CONSTANT('48'X);     SIT POSITION SPECIFIED
```

USED IF ITEM IS A LITERAL OR FOR A RANGE DEFINITION

```
 1    KPATLIT(*) DEFINED(KPATRIBT),
  2     *        PTR,               14
  2     KPATRCHK FIXED(32),         18 IF KPLITERL = ON  THEN OFFSET TO
                                       LITERAL AND SHORTEST ABREV. ELSE
                                       TWO HALF WORDS FOR RANGE DEFINITION
   4      KPAMINRG FIXED(16),       18 MINIMUM RANGE. IF KPAMINR = 0 A
                                       MINIMUM RANGE OF 1 IS ASSUMED.
    6     KPLTOFFT FIXED(16),       18 OFFSET TO LITERAL.KPLITERL = ON
   4      KPAMAXRG FIXED(16),       1A MAXIMUM RANGE DEF. MUST EXIST
    6     KPLTRUNC FIXED(8);        1A NO.OF CHARS OF SHORTEST TRUNCATION
                                       USED ONLY IF KPLITERL = ON
```

USED FOR THE CEP EXIT DEFINITION

```
 1       KPATCEP(*)  DEFINED(KPATRIBT),
  2      *           PTR,           14
  2      KPCEPEXI    PTR  (32),     18 CEP  EXIT LOCATION ID
   3      KPCEPRTC PTR  (8),        18 ROUTE CODE
   3      KPCEPID  PTR  (8),        19 IDENTIFICATION BYTE
   3      KPCEPPHS FIXED(15),       1A OFFSET TO PHASE NAME OF ROUTINE
    4     KPCEPADR FIXED(15),       1A OFFSET TO BRANCH ADDRESS OF ROUTINE
     6    KPCEPOFF FIXED(15),       1A POINT OFFSET OF ROUTINE
      8   KPCEPRID FIXED(8);        1A ROUTE ID OR *
```

USED FOR THE CHECK EXIT INFORMATION

```
 1       KPATCHCK(*)  DEFINED(KPATRIBT),
  2      *           PTR,           14
 2      KPCHKEXI  PTR  (32),     18 CHECK EXIT LOCATION ID
  4      KPCHKRTC PTR  (8),      18 ROUTE CODE
  4      KPCHKID  PTR  (8),      19 IDENTIFICATION BYTE
  4      KPCHKPHS FIXED(15),     1A OFFSET TO PHASE NAME OF ROUTINE
    8    KPCHKADR FIXED(15),     1A OFFSET TO BRANCH ADDRESS OF ROUTINE
     9   KPCHKOFF FIXED(15);     1A POINT OFFSET OF ROUTINE
```

USED FOR THE KEYWORD ITEM

```
1       KPATKWD(*)  DEFINED(KPATRIBT),
 2      *         PTR,           14
 2      KPKYWRD   FIXED(31),     18 KEYWORD IDENTIFIER
  4      KPKEYOFS FIXED(15),     18 OFFSET TO KEYWORD
  4      KPKEYABR PTR  (8);      18 NO.OF CHARS OF SHORTEST ABBREVIATION
```

USED FOR COLUMN INFO

```
1       KPATCOL(*)  DEFINED(KPATRIBT),
 2      *         PTR,           14
 2      KPCOLMN   FIXED(31),     18
  4      KPCOLNUM FIXED(15),     18 COLUMN NUMBER ONLY FOR POSITIONAL
  4      KPCOLRST FIXED(15);     18 UNUSED
```

USED FOR THE LINE SKIP INFO

```
1       KPATLSK(*)   DEFINED(KPATRIBT),
 2      *         PTR,           14
 2      KPLINSKP  FIXED(31),     18
  4      KPNOLIN  FIXED(15),     18 NUMBER OF LINES TO BE SKIPPED
  4      KPLRST   FIXED(15);     18 UNUSED
```

USED FOR THE SIT POSITION INFO

```
1       KPATSITP(*)  DEFINED(KPATRIBT),
 2      *         PTR,           14
 2      KPSITP    FIXED(31),     18
  4      KPPOSIT  FIXED(15),     18 POSITION OF NEXT ENTRY IN SIT
  4      KPSITRST FIXED(15);     18 UNUSED
```

THIS STRUCTURE DEFINES THE DATA TO BE WRITTEN INTO THE SIT AND THE VALUE TABLE DEFINED WITHIN THE CPCB.

THIS STRUCTURE MUST BE BASED ON REG1.

```
 2     SITENTRY    BDY(HWORD),      00
  4    SITINFO     PTR   (16),      00 SIT TYPE CONTROL INFORMATION
   6   SITCNTL     BIT   (8),       00 CONTROL INFO FOR THIS ENTRY
   6   SITTYPE     BIT   (8),       01 DEFINES THE NODE OF THE ITEM
  4    SITPNUM     PTR   (16),      02 PARAMETER NUMBER
   6   PNPARM      BIT   (1),       02 ON IF PARAMETER
   6   PNDEFLT     BIT   (1),       02 ON IF DEFAULT VALUE USED
   6   PNPAIR      BIT   (1),       02 ON IF PARAM VALUE HAS PAIR ATTRIB
  4    SITREST     BDY(HWORD),      04 REST OF SIF ENTRY ELEMENTS

1      SNGLPOS     DEFINED (SITREST),  04 POSITIONAL & SINGLE NODES
 2     POSOV       FIXED (15),         04 OFFSET INTO VALUE TABLE

1      KEYKALT     DEFINED (SITREST),  04 KEY &KEYWRD ALTERNAT NODES
 2     KEYAN       FIXED (15),         04 ALTERNATIVE NUMBER
  6    *           BIT(1),             04 ALWAYS ZERO
  6    ANDEFLT     BIT(1),             04 ON IF DEFAULT=@ AND PARM OMITTED

1      PALTS       DEFINED  (SITREST), 04 POSITIONAL ALTERNATIVE
 2     PALAN       FIXED (15),         04 ALTERNATIVE NUMBER
 2     PALOV       FIXED (15),         06 OFFSET INTO VALUE TABLE

1      PREPS       DEFINED (SITREST),  04 POSITIONAL WITH REPETITIONS
 2     PRERE       FIXED (15),         04 REPETITION COUNT
 2     PREOV       FIXED (15),         06 OFFSET INTO VALUE TABLE

1      PAREPS      DEFINED (SITREST),  04 POS ALTERN WITH REPETITIONS
 2     PARAN       FIXED (15),         04 ALTERNATIVE NUMBER
 2     PARRE       FIXED (15),         06 REPETITION COUNT
 2     PAROV       FIXED (15);         08 OFFSET INTO VALUE TABLE
```

THE FOLLOWING CONSTANT DEFINITIONS WILL BE USED TO INTERPRET THE POSITIONAL AND KEYWORD PARAMETERS

USED TO IDENTIFY CEP AND CHECK EXIT ROUTE CODES AND HOW TO INTERPRET THE CEP AND CHECK ENTRY POINTS

```
CDFETCH   BIT(8) CONSTANT('01'X),     EXIT IS VIA FETCH
CDPOINT   BIT(8) CONSTANT('02'X),     EXIT IS VIA POINT
CDBRANCH  BIT(8) CONSTANT('03'X),     EXIT IS VIA BRANCH
CDRETURN  BIT(8) CONSTANT('04'X),     EXIT VIA RETURN ONLY IF CEP
```

USED TO INTERPRET THE DEFAULT DEFINITION

```
CDDEFNO   BIT(8) CONSTANT('00'X),     DEFAULT = NO
CDDEFSIF  BIT(8) CONSTANT('04'X),     DEFAULT = SIF (DEF = YES, SIF)
CDDEF@    BIT(8) CONSTANT('08'X),     DEF = @ (DEF = YES, NOSIF)
CDDEFYES  BIT(8) CONSTANT('0C'X),     DEFAULT = YES (SIF FROM CT)
```

USED TO INTERPRET THE REPETITION FIELD

```
CDREPNO   BIT(8) CONSTANT('FF'X),     NO REPETITION DEFINED
CDREPAST  BIT(8) CONSTANT('5C'X);     REP = *
```

IJPACPCB - COMMAND ANALYSER CONTROL BLOCK

PARAMETER PART

```
 2   CPCBGENP   BOUNDARY(WORD),            NOTE: LEVEL 2 RESTRICTED

  3   CPPARM     BOUNDARY(WORD),
```

CPCB IDENTIFIER

```
   4   CPID       CHAR(4),                 CPCB IDENTIFIER   ('CPCB')
```

CPCB STATUS FLAGS

```
   4   CPSTFLGS,                           STATUS FLAGS AREA
    6  CPINSTAT   BIT(8),
     8 CPINCOMP   BIT(1),                  BUFFER COMPLETELY PROCESSED
                                           IF ON. SET BY NEXTPARM
     8 CPLNEEND   BIT(1),                  ON IF LINE END. COMMAND
                                           CONTINUATION REQUESTED.
     8 CPSIFFLG   BIT(1),                  SIF IS READY. SET BY
                                           SIFENTRY.
     8 CPCMDC     BIT(1),                  COMMAND CANCELLED IF ON.
                                           SET BY NEXTPARM
     8 CPPRMPT1   BIT(1),                  PROMPTING IN EFFECT IF ON.
                                           SET BY NEXTPARM. RESET BY
                                           RRTERM.
     8 CPNOPRMT   BIT(1),                  NO PROMPTING ALLOWED IF ON
     8 CPCLUPLP   BIT(1),                  CLEANUP LOOP FOR EXCESSIVE
    6  *          BIT(24),                 RESERVED
```

INPUT BUFFER CONTROL

```
   4   CPIN,                               INPUT INFORMATION
    6  CPINADR    PTR,                     ADDRESS OF INPUT AREA
    6  CPINSIZ    FIXED(15),               SIZE OF INPUT AREA
    6  CPINLEN    FIXED(15),               LINE LENGTH OF INPUT
    6  CPINSEQ    PTR(15),                 BEGIN OF SEQUENCE FIELD
    6  CPINBEG    PTR(15),                 BEGIN OF COMMAND INPUT
```

MESSAGE BUFFER CONTROL

```
   4   CPOUT,                              MESSAGE INFORMATION
    6  CPOUTADR   PTR,                     ADDRESS OF MESSAGE AREA
    6  CPOUTSIZ   FIXED(15),               SIZE OF MESSAGE AREA
    6  CPOUTLEN   FIXED(15),               LENGTH OF MESSAGE
```

COMMAND NUMBER

```
4   CPCMDNO    CHAR(4),                COMMAND/SIF NUMBER
```

SIF CONTROL

```
4   CPSITVT,                           SIF (SIT/VT) INFORMATION
 6  CPSITADR   PTR,                    ADDRESS OF SIT
 6  CPVTADR    PTR,                    ADDRESS OF VALUE TABLE
 6  CPSITLEN   FIXED(15),              LENGTH OF SIT
 6  CPVTLEN    FIXED(15),              LENGTH OF VALUE TABLE
 6  CPXWRK     PTR,                    PTR TO EXTERNALLY DEFINED
                                       VALUE TABLE WORK AREA
 6  CPXWLEN    FIXED(15),              LENGTH OF EXTERNAL WORK AREA
4   CPSITCNT   FIXED(15),              COUNTER FOR PERMUTED PARMS
```

ROUTING CONTROL

```
4   CPROUTE    ,                       ROUTING INFORMATION
 5  CPCEPRTC   PTR(8),                 ROUTING CODE
 5  *          PTR(8),                 RESERVED
 5  *          FIXED(15),              RESERVED
 5  CPCEPPHN   CHAR(8) BDY(WORD),      PHASENAME OF CEP
  6  CPCEPADR  PTR(31),                ADDRESS OF CEP
   7  CPCEPOFF  FIXED(15),             OFFSET IN BRANCHTAB TO CEP
    8  CPCEPRID  BIT(8),               ROUTING ID OR *
```

I/O INTERFACE CONTROL

```
4   CPIO,                              I/O INTERFACE SPECIFICATION
 6  CPIOPTR    PTR,                    POINTER TO I/O INTERFACE.
                                       SET BY ENVIRONMENT.

 6  CPIOANC    PTR,                    I/O INTERFACE DYNAMIC WORK-
                                       AREA ANCHOR IN MULTITHREAD

 6  CPIOTYPE   BIT(8),                 INDICATE READ AND/OR WRITE
  7  CPIOTYPW  BIT(1),                          WRITE
  7  CPIOTYPR  BIT(1),                          READ
  7  CPIOFSET  BIT(1),                 ON: IOF CALLED
 6  CPIODEV    BIT(8),                 DEVICE: SYSLOG, SYSLST,
                                       OR USER'S TERMINAL

  7  CPIODSYS  BIT(3),

   8 CPIODLOG  BIT(1),                         SYSLOG
   8 CPIODLST  BIT(1),                         SYSLST
   8 CPIODRDR  BIT(1),                 ON: IF INPUT FROM READER
                                       OFF: INPUT FROM TERMINAL
 6  *          BIT(16),                RESERVED
 6  CPIOCNTL   PTR,                    SAVE AREA FOR RET NODE FOR
                                       CONT LINE PROCESSING

 6  * (1)      PTR,                    RESERVED
```

GENERAL COMMAND SET CONTROL

```
 4   CPGENCMD,                              GENERAL COMMAND SET
  6  CPGENDIR   PTR,                        ADDR OF GENERAL DIRECTORY
  6  CPGENOFS   PTR,                        CEP EXITS, POINT=OFFSET.
                                            ADDR OF CEP EXT ADDR LIST.
  6  CPGENANC   PTR,                        CEP DYNAMIC WORKAREA ANCHOR
                                            IN MULTITASKING/MULTITHREAD
  6  CPGENCOF   PTR,                        CHK EXITS, POINT = OFFSET
                                            ADDR OF CHK EXT ADDR LIST
```

SUBCOMMAND SET CONTROL

```
 4   CPSUBCMD,                              SUBCOMMAND SET FOR ONE MODE
  6  CPSUBDIR   PTR,                        ADDR OF MODE SUBDIRECTORY
  6  CPSUBOFS   PTR,                        CEP EXITS, POINT=OFFSET.
                                            ADDR OF CEP EXT ADDR LIST.
  6  CPSUBANC   PTR,                        CEP DYNAMIC WORKAREA ANCHOR
                                            IN MULTITASKING/MULTITHREAD
  6  CPSUBCH    BIT(8),                     CHAIN TO GENERAL COMMANDS
  6  *          BIT(24),                    RESERVED
```

CHECK EXIT CONTROL

```
 4   CPCHECK    CHAR(12) BDY(WORD),         CHECK EXIT INFO
  5  CPCHKRTC   PTR(8),                     ROUTING CODE
  5  *          PTR(8),                     RESERVED
  5  *          FIXED(15),                  RESERVED
  5  CPCHKPHN   CHAR(8) BDY(WORD),          PHASENAME OF CHECK EXIT RT
   6  CPCHKADR  PTR(31),                    ADDRESS OF CHECK EXIT RT
    7  CPCHKOFF  FIXED(15),                 OFFSET IN BRANCHTAB TO CHK
 4   CPCHKSP    PTR,                        PTR TO PREPARED SIT ENTRY
 4   CPCHKVP    PTR,                        PTR TO PREPARED VT ENTRY
 4   CPCHKVL    PTR(15),                    LENGTH OF VT ENTRY
 4   CPCHKFLG   PTR(16),                    FLAGS
  6  CPCHKMSG   BIT(1),                     ON IF CHK EXIT RETURNS MSG
  6  CPCHKSET   BIT(1),                     ON: CHECK EXIT TAKEN
  6  CPCHKEC    BIT(1),                     ON: SEMANTIC ERROR CORRECT
  6  CPCHKPE1   BIT(1),                     ON: SEMANTIC ERROR CORRECT
                                                FOR FIRST PAIR ELEMENT.
  6  CPCHKPE2   BIT(1),                     ON: SEMANTIC ERROR CORRECT
                                                FOR SECOND PAIR ELEM.
  6  CPCHKSVE   BIT(1),                     SEVERE ERROR ISSUED IN A
                                            CHECK EXIT ROUTINE. COMMAND
                                            IS CANCELLED.
```

OPTIONS CONTROL

```
   4   CPOPTION   BIT(32),                 OPTIONS FOR CA
    6  CPINPUT    BIT(1),                  ON IF INP MODE DURING EDIT
    6  CPSCAN     BIT(1),                  ON IF SYNTAX CHK IN ED INPT
    6  CPDEBUG    BIT(1),                  ON IF DBG MODE TRACE
    6  *          BIT(29),
   4   * (1)      PTR,                     RESERVED
```

ENVIRONMENT INFORMATION

```
   4   CPENV,                              ENVIRONMENT INFORMATION
    6  CPENVID    BIT(16),                 ENVIRONMENT ID
    6  CPENVPAR   BIT(16),                 ENVIRONMENT PARAMETERS
```

IJPARSER INTERNAL CPCB PART

```
  3   CPINTRNL    BDY(WORD),
```

INTERNAL INTERFACE

```
   4   CPINTIF,
    6  CPCTPTR    PTR  (31),               CT-PTR
    6  CPPHASNM   CHAR (8),                PHASE NAME
    6  CPREGSAV,                           REGISTER SAVE-AREA
     8  CPREGSA1  PTR,
     8  CPNXTPIP  PTR,                     SAVE AREA POINTER TO ITEM
     8  CPNXTPIL  PTR,                     SAVE AREA LENGTH OF ITEM
    6  CPPARMNO   FIXED(31),               CURRENT PARAMETER NUMBER
    6  CPATTCNT   FIXED(15),               NOS OF ATTRIBUTES PROCESSED
    6  CPIFFLG    PTR(16),                 INTERNAL FLAGS
     8  CPDSESW   BIT(1),                  DIRSERV END SWITCH
     8  CPPRMTPA  BIT(1),                  CURRENT PARAMETER PROVIDED
                                           AS PROMPT REPLY IF ON. NO
                                           PROMPT OR SUCCEEDING ITEMS
                                           IN REPLY IF OFF. SET AND
                                           RESET BY NEXTPARM.
     8  CPPRMTAC  BIT(1),                  MORE ITEMS IN PROMPT REPLY
                                           IF ON. LAST ITEM IN REPLY
                                           IF OFF. SET/RESET NEXTPARM

     8  CPDFLTEN  BIT(1),                  ON IF DEFAULT ENTRY MADE
                                           SET BY DEFAULT
                                           RESET BY NEXTPARM
     8  CPFLSHPM  BIT(1),                  ON IF ERROR ROUTINE FLUSHES
```

```
                                                REST PARAMETERS.
                                                SET BY ERROR.
                                                RESET BY NEXTPARM.
    8  CPKREPER  BIT(1),                        IF ON. ERROR IN A KEYWORD
                                                REPETITION LIST
    8  CPMDPOS   BIT(1),                        ON IF CONT LINE IN POSPARM
    8  CPPRMLIL  BIT(1),                        ON IF LAST ITEM IN PROMPT LN
    8  CPKREPMD  BIT(1),                        ON IF PROCESSING IN KREP MOD
    8  CPCLCNT   BIT(1),                        ON IF SUCCESSIVE CONT.LINES
    8  CPPRMMOD  BIT(1),                        ON IF ANALYSER IS OPERATING
                                                IN PERMUTATION MODE.
    8  CPKRCNLN  BIT(1),                        CONTINUATION LINE IN PROCESS
```

VALIDITY INTERNAL AREA

```
  4   CPVALAR,
   6  CPVALMIN   FIXED(15),                     MINIMUM STRING LENGTH
   6  *          FIXED(15),
   6  CPVALMM    CHAR(32),                      WORKFIELD FOR MIN AND MAX
   6  CPVALIT    CHAR(LENGTH(CPVALMM)),           WRKFLD FOR ITEM TRANSLATE
  4   *          PTR,                           RESERVED
```

STACK ENTRY PREPARATION AREA

```
  4   CPSTKENT,                                 STACK ENTRIES
   6  CPPRMFLG   PTR  (32),                     PERMUTATION CONTROL FLAGS
    8 CPPRMLOP   BIT  (1),                      ON: PERMUTATION IN EFFECT
                                                    IN CURRENT KLIST TREE
   6  CPCTSTRE   PTR  (31),                     PTR TO CURRENT STRING ENTRY
   6  CPRETURN   PTR  (31),                     RETURN ADDRESS
   6  CPREPCNT   FIXED(15),                     REPETITION COUNTER
   6  CPNDTYPE   PTR  (16),                     TYPE OF LAST PROCESSED NODE
    8 CPCNTL     BIT  (8),                      TYPE CONTROL INFO
    8 CPTYPE     BIT  (8),                      TYPE DEFINITION

   6  CPCURSIT   PTR  (31),                     SIT POSITION DEFINED FOR
                                                THE CURRENT KLIST NODE.
   6  CPCURIP    PTR  (31),                     POINTS TO THE CURRENT ITEM
                                                WHICH HAS A MISMATCH.
  4   CPRETCD2   PTR  (31),                     SAVE AREA FOR RETURN CODE
                                                AFTER PROMPT
```

EXTENDED ATTRIBUTE INTERFACE AREA

```
  4   CPEXATR,
   6  CPATFLGS   PTR  (16),                     FLAG AREA
    8 CPPAIR     BIT  (1),                      ON IF PAIR ATTR.PROCESSED
```

```
     8 CPCHKEXT   BIT  (1),              ON IF CHECK EXIT PROCESSED
     8 CPSECERR   BIT  (1),              ON IF ERROR IN 2ND PAIR ELE
     8 CPNOCLN    BIT  (1),              ON IF COLON MISSING |INCOMPL
     8 CPAIRERR   BIT  (1),              ERROR IN PAIR DETECTED.
     8 CPCOMPNT   BIT  (1),              ON IF MSHP COMPONENT FNUM
    6  CPINDX     PTR  (15),             INDEX INTO ATTRIB ARRAY
    6  CPOUTPTR   PTR  (15),             PTR INTO CPOUTBUF
    6  *          PTR  (15),
    6  CPRETCD    PTR  (31),             RETURN CODE SAVE AREA
    6  CPR7       PTR  (31),             POINTER TO CT PARAMETER PART
    6  CPR8       PTR  (31),             POINTER TO CT VALUE PART
    6  CPR9       PTR  (31),             POINTER TO CURRENT STRUCTURE
                                         ENTRY.
    6  CPMDINAR   PTR  (31),             SAVE AREA FOR CPINADR
    6  CPMDINLN   PTR  (31),             SAVE AREA FOR CPINLEN
    6  CPMDINSZ   PTR  (31),             SAVE AREA FOR CPINSIZ
    6  CPMDINSQ   PTR  (31),             SAVE AREA FOR CPINSEQ
    6  CPSECLNG   FIXED(15),             LENGTH OF SECOND PAIR ELEMT
    6  *          FIXED(15),             RESERVED
    6  CPSECPRM   PTR  (31),             PTR TO 2ND ELEMENT IN ERROR
    6  CPCLNPTR   PTR  (31),             PTR TO 1ST PAIR ELEMENT
    6  CPCLNLNG   PTR  (15),             LENGTH OF 1ST PAIR ELEMENT
   4   *          PTR  (15),             DUMMY SPACE FOR FUTURE USE
```

NEXTPARM INTERNAL AREA

```
   4   CPNEXTPA,
    6  CPSWITCH   PTR  (16),             AREA FOR LEVEL INFO ETC.
     8  CPCMDEND  BIT  (1),              ON IF COMMAND END
     8  CPVERBS   BIT  (1),              VERB SEARCH REQUESTED BY
                                         CAENTRY. RESET BY NEXTPARM.
     8  CPSBLEND  BIT  (1),              ON IF SUBLIST END
     8  CPNPIRES  BIT  (1),              ON IF NO NEW ITEM FROM NP
     8  CPRETPMK  BIT  (1),              ON IF '?' IS TO BE RETURNED
                                         OFF GET PROMPT TEXT
    6  CPLITEM    FIXED(15),             LENGTH OF ITEM
    6  CPPITEM    PTR  (31),             PTR TO ITEM IN I/O AREA
```

SIT INTERFACE AREA

```
   4   CPSITIF   CHAR(10) BDY(WORD),     SIFDATA BASED ON THIS AREA
   4   *         CHAR(2) BDY(HWORD),     PADDING
   4   *         PTR,                    RESERVED
```

SIF ENTRY INTERNAL AREA

```
 4   CPSIFENT,
  6  CPSITEF    PTR,                       NEXT FREE ENTRY IN SIT AREA
  6  CPSITEND   PTR,                       END OF SIT
  6  CPVTEF     PTR,                       NEXT FREE ENTRY IN VT AREA
  6  CPVTEND    PTR,                       END OF VT
  6  CPSENTFL   BIT(8),                    STATUS FLAGS FOR SIFENTRY
   8 CPNXTCMD   BIT(1),                    PROCESSING OF NEXT COMMAND
                                           BEGINS. SET BY CAENTRY.
                                           RESET BY SIFENTRY.
  6  *          BIT(24),                   RESERVED


 4   CPSTPCTL,                             SIT PREPARATION AREA CONTRL
  6  CPSTPBEG   PTR,                       BEGIN OF SIT PREP AREA
  6  CPSTPNXF   PTR,                       NEXT FREE ENTRY IN SIT PREP
  6  CPSTPEND   PTR,                       END OF SIT PREP AREA
```

LINE BEGIN STATUS SAVE AREA

```
 4   CPSVELB,                              LINE BEGIN STATUS SAVE AREA
  6  CPLBCSTK   PTR,                       CURRENT STACK ENTRY
  6  CPLBCSTR   PTR,                       CURRENT STRUCTURE ENTRY
  6  CPLBSITE   PTR,                       NEXT FREE SIT ENTRY
  6  CPLBVTEF   PTR,                       NEXT FREE VT ENTRY
  6  CPLBRETN   PTR,                       RETURN POINT WITH FULL LINE
  6  CPLBFLGS   BIT(16),                   STATUS FLAGS
   8 CPLBLNEE   BIT(1),                    CONTINUATION LINE RECEIVED
  6  *          BIT(16),
 4   CPSEMPPT   PTR,                       CONTAINS PTR TO PARM AREA
                                           OF ENTRY KALT AND PALT.
                                           SET IN KOR & POR NODES.
                                           RESET IN ERROR.
 4   CPREGSA2   PTR,                       SAVE AREA FOR RET FROM PRMT
 4   CPERRETR   PTR,                       CONTAINS THE RETURN POINT
                                           TO WHICH THE ERROR CORRECT.
                                           TRANSFER CONTROL.
```

ERROR INTERFACE AREA

```
 4   CPERRIF,
  6  CPPBPTR   PTR,                        BEGIN OF PARAMETER TYPING
  6  CPERCLMN  FIXED(15),                  COLUMN WHERE ERROR OCCURRED
  6  CPERSVRT  FIXED(15),                  ERROR SEVERITY
```

```
   6  CPMSGID    CHAR(9) BDY(WORD),      MESSAGE ID FOR ERROR
    8 CPERMSG    CHAR(4) BDY(WORD),      CONTAINS ERROR CODE
    8 CPMSG      CHAR(5) BDY(WORD),      MESSAGE INDEX
   6  *          CHAR(3),                RESERVED
   6  CPRRBGIP  PTR,                     SAVE ERROR RESTART STATUS
   6  CPRRBGIL  PTR(15),                 OF RECURSIVE ROUTINES
   6  *         PTR(15),                 RESERVED
   6  CPERIRP   PTR,                     REST PARMS POINTER
   6  CPERREP   PTR,                     END OF REPLY POINTER
   6  CPERIRL   PTR(15),                 LENGTH OF REST PARMS IN INP
   6  CPERRL    PTR(15),                 LENGTH OF CORRECTION REPLY
   6  CPINSPTR  PTR,                     PTR TO PARMLIST FOR MSGLIB
```

STACK CONTROL AREA

```
 3  CPSTKCTL,
  4   CPBEGSTK   PTR  (31),              BEGIN OF STACK
  4   CPENDSTK   PTR  (31),              END OF STACK
  4   CPCURSTK   PTR  (31),              CURRENT STACK ENTRY
```

AREA FOR SIT PREPARATION

```
 3  CPSITPRP   CHAR(100),                SIT PREPARATION TABLE
```

AREA FOR SIF & STACK

```
 3  CPSIF      CHAR(200),                AREA FOR SIT ENTRIES

 3  CPVTBL     CHAR(300),                AREA FOR VALUE TABLE ENTRIES

 3  CPSTACK    CHAR(16*LENGTH(CPSTKENT)),    AREA FOR STACK
```

INTERNAL SAVE AREA

```
 3  CPINTSV    CHAR(72)  BDY(WORD),      INTERNAL SAVEAREA
 3  CPSAVE     CHAR(72),                 CA SAVEAREA
```

IJPARSER WORK AREA

```
 3  CPWORK     CHAR(100),                VT PREP AREA FOR PAREP, PREP
```

INTERFACE BETWEEN MESSAGELIB MODULE AND ERROR ROUTINE

```
  3  CPMESSGL   CHAR(50) BDY(WORD),
   4 CPLGMSG1  FIXED(31),                  LENGTH OF FIRST INSERT VAL
    6 CPZERINS   CHAR(2)  BDY(WORD),       END INDICATOR
   4 CPINSRT1   CHAR(20) BDY(WORD),        FIRST INSERT VALUE
   4 CPLGMSG2  FIXED(31),                  LENGTH OF SECOND INSERT VAL
    6 CPONEINS   CHAR(2)  BDY(WORD),       END INDICATOR
   4 CPINSRT2   CHAR(20) BDY(WORD),        SECOND INSERT VALUE
   4 CPTWOINS   CHAR(2),                   INSERT END INDICATOR

 2  CPREST      CHAR(2048 - LENGTH(CPCBGENP)),

 2  CPEND       CHAR(0);                   DUMMY END OF CPCB
```

END OF CPCB

PARSER: DIAGNOSTIC AIDS

MESSAGE CROSS-REFERENCE

The following table shows the parser messages and the modules which issue the messages. All messages are printed by the message module IJPAPMSG.

4D02I	IJPARS	IJPADIRS			
4D03I	IJPARS	IJPANXTP			
4D04I	IJPANXTP				
4D05I	IJPANXTP				
4D06I	IJPANXTP				
4D07A	IJPANXTP				
4D08I	IJPANXTP				
4D09I	IJPANXTP				
4D10I	IJPARS	IJPANXTP	IJPADEFT	IJPASIFN	IJPAVALD
4D12I	IJPARS				
4D13I	IJPARS				
4D14I	IJPARS				
4D15I	IJPARS				
4D16I	IJPARS				
4D17I	IJPARS				
4D18I	IJPARS				
4D19I	IJPARS	IJPASIFN			
4D20I	IJPARS				
4D21I	IJPARS				
4D22I	IJPARS				
4D23I	IJPARS				
4D24I	IJPARS				
4D25I	IJPARS				

LSERV: INTRODUCTION

The Label Service (LSERV) program consists of one module: IJBLSERV.

This program displays on SYSLST the contents of the label information area. This label information is written into that area by job control based on the following job control statements:

- For disk label information: DLBL, EXTENT

- For tape label information: TLBL

For detailed information about the internal organization of the label area, see the Diagnosis Reference: Initial Program Load and Job Control.

The format of the label information records is described in Figure 12 on page 305 and Figure 13 on page 306. LSERV assumes that format.

The label subarea in which the records are written depends on the user-specified job control OPTION statement. LSERV is executed by the // EXEC LSERV job control statement from either SYSLOG or SYSRDR. See the publication VSE/Advanced Functions Serviceability Aids and Debugging Procedures for a description of how to use LSERV. If LSERV runs real, the partition must have at least 8K bytes of real storage allocated.

LSERV: PROGRAM DESIGN AND ORGANIZATION INFORMATION

IJBLSERV - LSERV PROGRAM

Phase name: LSERV

Called by: User (EXEC LSERV)

Phases called: none

Data areas used:	SYSCOM	System Communication Region
	SLA	System Label Area
	AVRADR	Volume information area

Messages issued: (on SYSLST): WARNING: UPDATING IN PROGRESS FOR THIS AREA;
DATA NOT AVAILABLE OR OBSOLETE.

External input: Label information in SLA (System Label Area)

External output: Listing of all labels

Exits: EOJ (SVC 14)

Entry: IJBLSERV

Function Description: LSERV gets the label information from symbolic label access routines. The LABEL macro, which is described in the Diagnosis Reference: IPL and Job Control, is used for this purpose. Two functions may be performed by LSERV:

1. All label subareas are located sequentially to see if they contain any label information.

2. If there is label information, the label records are requested sequentially from symbolic label access routines.

All required control information for symbolic label access routines is contained in the LABEL PARAMETER LIST (LPL).

If a defined label subarea contains no label information records, a message is printed indicating that no records are present in this area; if a defined label subarea contains only data-secured file label information records, nothing is printed. Otherwise, the label information records (except for data-secured files) are formatted and printed on SYSLST. For examples of LSERV output, see VSE/Advanced Functions Serviceability Aids and Debugging Procedures.

```
Internal
Routines                                  Function
----------------------------------------------------------------------

IJBLSERV                                  Open SYSLST.
   |
   |--->FRSTDATA                          Print header line.
   |
   |--->GETGRP                            Print group header.
   |      |
   |      +--->GETGRPB                    Print 'update in progress' if
   |                                        label group not closed.
   |                                      Print 'NONE' if label group empty.
   |
   |--->GETLAB                            Get next label by LABEL macro.
   |      |
   |      |--->PRTHDR                     Print header line if new page.
   |      |
   |      +--->OUTPUT                     Common output routine.
   |
   |--->PRINTDK1                          Disk label output routine.
   |      |
   |      |--->PROEXTNT                   Extent information processing.
   |      |
   |      |--->CHECKEX                    Disk label processing exit routine.
   |      |
   |      +--->ADDLBLO                    Additional DLBL record processing.
   |
   |--->TAPLAB                            Tape label processing routine.
   |
   +--->EOJJT                             End of job routine.
```

LSERV: DATA AREA INFORMATION

Label Information Record Formats

TAPE	Field		Length	Offset
			decimal	hex.
Key	1	Key Field	8	0
	2	Internal Sequence Number (Not Used)	1	8
Data	1	File Name	7	9
	2	Not Used	1	10
	3	File Identifier	17	11
	4	File Serial Number	6	22
	5	Volume Sequence Number	4	28
	6	File Sequence Number	4	2C
	7	Generation Number	4	30
	8	Version Nr. of Generation	2	34
	9	Creation Date	6	36
	10	Expiration Date	6	3C
	11	File Security Number	1	42
	12	Block Count	6	43
	13	System Code	13	49
	14	Flag Bytes	2	56

Figure 12. Format of the Label Information Records for Tape

DISK	Field		Length decimal	Offset hex
Key	1	File Name	7	0
	2	Internal Sequence Number	1	7
Data	1	DLBL-Extent Indicator	1	8
	2	File Name	7	9
	3	DA-IS Indicators	1	F
	4	File Identifier	44	10
	5	Format Identifier	1	3C
	6	File Serial Number	6	3D
	7	Volume Sequence Number	2	43
	8	Creation Date	3	45
	9	Expiration Date	3	48
	10	Retention Period	2	4B
	11	Open Code	1	4D
	12	System Code (see Note 1)	13	4E
	13	Volume Serial Number	6	5B
	14	Extent Type	1	61
	15	Extent Sequence Number	1	62
	16	Extent Lower Limit	4	63
	17	Extent Upper Limit	4	67
	18	Logical Unit Address	2	6B
	19	Flag Bytes/Reserved	2	6D
		Another extent if present for DA or IS files		

Figure 13. Format of the Label Information Records for Disk

Notes:

This field is processed only when VSAM is used or when the BLKSIZE parameter of the DLBL statement is used. For BAM BLKSIZE parameter the contents of field 12 are:

12	Not used	11	4E
	blocksize	2	59

For sequential disk files, a complete block is repeated for each additional extent. For direct access or ISAM files, only fields 13 through 19 are repeated for each extent.

When VSAM is used, the contents of field 12 are:

12	Owning Catalog Filename	7	4E
	Not used	2	55
	Buffer space to be allocated	4	57

For VSAM files, there may be an additional Label Information Record following the VSAM Label Record:

DISK (VSAM only)	Field		Length (hex)	Offset
Key	1	File Name	7	0
	2	Int. Seq. Number	1	7
Data	1	File Name	1	7
	2	Disposition	1	9
	3	Records	8	C
	4	Record Size	4	14

Figure 14. Format of the Additional Label Information Record

INDEX

IBM Virtual Storage
Extended/Advanced Functions
Diagnosis Reference: Serviceability Aids
Order No. LY33-9115-01

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comments are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address: ______________________________
____
____
____

What is your occupation? ______________________________

Number of latest Newsletter associated with this publication: ______________________________

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

LY33-9115-01

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

No Postage
necessary
if mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

Postage will be paid by addressee:

International Business Machines Corp.
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold and Tape | Please Do Not Staple | Fold and Tape

IBM®

IBM Virtual Storage
Extended/Advanced Functions
(File No. S370/4300-37)
Printed in U.S.A.

LY33-9115-01