SH24-5002-4
File No. S370/4300-50

Data Language/I Disk Operating System/ Virtual Storage (DL/I DOS/VS) Diagnostic Guide

Program Product

Program Number 5746-XX1

IBM

Fifth Edition (December 1983)

This edition, SH24-5002-4, is a revision of SH24-5002-3 and applies to Version 1, Release 7 (Version 1.7) of Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters. Before using this publication in connection with the operation of IBM systems, consult the latest edition of IBM System/370 and 4300 Processors Bibliography, GC20-0001, for editions that are applicable and current.

Summary of Changes

For a list of changes, see page iii.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

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

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

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

IBM Corporation
Dept. 812BP
1133 Westchester Avenue
White Plains, NY, 10604 U.S.A.

or

IBM Deutschland GmbH
Dept. 3282
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

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

© Copyright International Business Machines Corporation 1977, 1978, 1979, 1980, 1981, 1983

SUMMARY OF CHANGES

SUMMARY OF CHANGES FOR SH24-5002-4, VERSION 1.7

This edition replaces SH24-5002-3. It applies to Version 1, Release 7 (Version 1.7) of DL/I DOS/VS. Functional enhancements include:

- Interactive Utility Generation Facility
- IMF Enhancements
- Documentation Aid Facility
- IMF Adaptation to ISPF
- Utilities Operational Improvements
- MPS Restart Facility

Miscellaneous additions, improvements, and corrections also appear throughout this manual.

SUMMARY OF CHANGES FOR SH24-5002-3, VERSION 1.6

This edition replaces SH24-5002-2 and TNL SN24-5677. It applies to Version 1, Release 6 (Version 1.6) of DL/I DOS/VS.

In addition to information relating to functional enhancements introduced in Version 1.6, major changes to the prior edition include:

- Removal of DL/I Trace Facility information from Chapter 2 to form a new chapter, Chapter 6. In Chapter 6, the existing DL/I Trace Facility information is reorganized and new information added to describe:
  - user exits and to provide samples of user exit routines.
  - enhancement to the DL/I Trace Print Utility that permits selective printing of desired trace records.
- Reorganization of message-related information in Chapter 3 to eliminate redundancy with DL/I DOS/VS Messages and Codes and to improve the usefulness of Chapter 3 information.

Miscellaneous additions, improvements, and corrections also appear throughout this manual.

SUMMARY OF CHANGES FOR SH24-5002-2 UPDATED BY SN24-5677

Technical Newsletter SN24-5677 contains information about the use of the Interactive Macro Facility with DL/I DOS/VS Version 1, Release 5. Changes in the use of the DL/I Trace Facility and the Trace Print Utility, a description of a patch area, and various editorial changes are also included.

SUMMARY OF CHANGES FOR SH24-5002-2, VERSION 1.5

This edition is a major revision of SH24-5002-1. It contains changes for the support of DL/I Version 1.5 (Release 5) of the IBM System/370 Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) Program Number 5746-XX1.

Included in this revision is information relating to DL/I control block relationships, online trace and the trace print utility, field level sensitivity debugging, fixed block architecture support, module identification, online test program DLZMDLI0, additional ABEND message explanations, and various editorial changes to improve the usefulness of this publication.

PREFACE

The purpose of this publication is to assist in analyzing and isolating DL/I production or testing problems. It is intended for installation personnel responsible for system programming and system maintenance, and the IBM Programming Support Representative (PSR).

This publication contains the following six chapters and appendixes:

- Chapter 1: General Information, is a collection of high-level information intended as a general review of DL/I job control statements, control blocks, calls, data base organization, and access methods.
- Chapter 2: Diagnostic Aids, discusses DL/I control flow, scheduling errors, online trace, logical retrieve, and online wait state debugging; and offers suggestions to assist you in isolating the problem.
- Chapter 3: Interpreting and Debugging Dumps, describes the type of dumps DL/I produces and suggests possible paths to follow in finding the cause of the dump through analysis of selected messages.
- Chapter 4: Module and Program Aids, addresses problems associated with various request handlers, initialization/termination, utility, and test programs. Included are major causes for program failure, things the user must do when using these programs, and suggestions for the user to check to determine the cause of a program failure.
- Chapter 5: Interactive Facilities, describes the Interactive Macro Facility (IMF) and Interactive Generation facility (IUG), their modules, created tables, and error conditions.
- Chapter 6: DL/I Trace Facility and Trace Print Utility, describes using the DL/I Trace Facility, its operands and parameters, in problem determination.
- Appendix A. MPS Diagnostic Aids
- Appendix B. MPS XECB-Waitlist Dependence and Main Routine Flow
- Appendix C. Segment Intent List
- Appendix D. Online Test Program - DLZMDLI0
- Appendix E. Reporting DL/I Problems, identifies documentation requirements for reporting problems and for submitting APARs.

The assumed level of skill in this publication is a thorough understanding of VSE/Advanced Functions, and CICS/VS if DL/I is used in an online environment.

Many figures in this manual show a concept and, therefore, do not show displacements. Figure 1-7 on page 1-9 is an example. You can obtain the displacements for fields, etc., described in these figures from other sources. The DL/I DOS/VS Logic Manual, Volume 1 is one such source. Other figures include displacements because they show specific data relationships. Figure 2-8 on page 2-14 is an example. Where stated, displacements are given in decimal and hexadecimal, 14 (E), respectively.

RELATED PUBLICATIONS

DL/I DOS/VS Messages and Codes, SH12-5414

DL/I DOS/VS Logic Manual, Volume 1, LY12-5016

DL/I DOS/VS Logic Manual, Volume 2, LY24-5215

DL/I DOS/VS Data Base Administration, SH24-5011

DL/I DOS/VS Resource Definition and Utilities, SH24-5021

DL/I DOS/VS Interactive Resource Definition and Utilities, SH24-5029

DL/I DOS/VS Recovery/Restart Guide, SH24-5030

OTHER DL/I PUBLICATIONS

DL/I DOS/VS Library Guide and Master Index, GH24-5008

DL/I DOS/VS Application Programming: High Level Programming Interface, SH24-5009

DL/I DOS/VS Application Programming: CALL and RQDLI Interfaces, SH12-5411

DL/I DOS/VS Low-Level Code/Continuity Check Program, Program Reference and Operations Manual, SH20-9046

DL/I DOS/VS Guide For New Users, SH24-5001

FOR RELATED VSE PUBLICATIONS

VSE/Advanced Functions System Control Statements, SC33-6095
VSE/Advanced Functions Service Aids, SC33-6099

FOR RELATED CICS/VS PUBLICATIONS

CICS/DOS/VS Installation and Operations Guide, SC33-0070

CICS/VS Application Programmer's Reference Manual (Macro Level), SC33-0079

CICS/VS Customization Guide, SC33-0131

CICS/VS Resource Definition Guide, SC33-0149

FOR RELATED VSE/VSAM PUBLICATIONS

VSE/VSAM VSAM Logic, Volume 1: Catalog Management, LY24-5191

VSE/VSAM VSAM Logic, Volume 2: Record Management, LY24-5192

These and other DL/I DOS/VS publications are described in the IBM System/370 and 4300 Processors Bibliography, GC20-0001.

CONTENTS

FIGURES

CHAPTER 1. GENERAL INFORMATION

This chapter reviews:

- Job control statements
- UPSI byte bit settings
- Partition and control block relationships
- DBD generation control statements
- PSB generation control statements
- Creating and maintaining application control blocks
- DL/I call paths between DL/I and the application program
- HD and HS data base organizations and access methods

References to sources of additional information are provided throughout the chapter.

JOB CONTROL STATEMENTS AND PARAMETERS

Data Language/I (DL/I) is executed as a standard VSE/Advanced Functions application program. Figure 1-3 on page 1-4 illustrates the job control statements required for a typical batch application program. Refer to VSE/Advanced Functions System Control Statements for detailed information concerning job control statements.

UPSI BYTE SETTINGS FOR EXECUTION CONTROL

Several execution-time functions can be controlled by the UPSI byte. The settings are shown in Figure 1-1 for online and in Figure 1-2 on page 1-3 for batch.

Bits 0 - 2	Reserved for CICS/DOS/VS.
Bits 3 - 5	Available subject to Note.
Bit 6 = 0 = 1	DL/I log function service. DL/I system log function inactive.
Bit 7 = 0 = 1	DL/I system log function on CICS/VS system journal. DL/I system log function on DL/I system log device (SYS011-LOGOUT).

Note: For further information, see CICS/DOS/VS Installation and Operations Guide.

Figure 1-1. UPSI Byte (Online)

DL/I PARAMETERS

The DL/I parameter information shown in Figure 1-4 on page 1-5 follows the EXEC statement -- see Figure 1-3. The parameter information begins in column 1, and it must be entered from either SYSIPT or SYSLOG.

Parameters can be entered from SYSIPT and SYSLOG. However, continuation statements, if required, can only be entered from SYSIPT. Continuation statements are not permitted from SYSLOG.

Continuation is indicated by a non-blank character in column 72 of the statement being continued. The parameter statement can be stopped in or before column 71 and be continued in a continuation statement.

Disk logging is not supported in the online or MPS environments. If program isolation is active, the user must select CICS/VS journal services for writing log information. Parameter information may be any of the forms shown in Figure 1-4 on page 1-5.

Bit 0 = 0 = 1	Read parameter information via SYSIPT. Read parameter information via SYSLOG.
Bits 1 - 4	Available for use by the application program.
Bit 5 = 0 = 1 (Note 3)	Storage dump produced on abnormal task termination if STXIT active (bit 7 = 0). No storage dump produced. (// OPTION NODUMP must be specified in the JCL.)
Bit 6 = 0 = 1 (Note 1)	All data base modifications written to DL/I system log tape. DL/I system log function inactive.
Bit 7 = 0 = 1 (Notes 1,3)	STXIT linkage to DL/I for abnormal task termination (Note 2.) STXIT inactive. Note that no data bases will be closed. (Note 4.)

Notes:

1. Bits 6 and 7 not used in MPS batch.
2. For reload or reload/restart, bit 7 is always set to 0 by DL/I.
3. If bit 7 = 0 and bit 5 = 1, you will not get a dump.
4. For DLZUDMP0, if bit 7 = 1, tape is not rewound.
   Other UPSI bits do not apply

Figure 1-2. UPSI Byte (Batch and MPS Batch)

// JOB		
// UPSI	x0000xxx (Note 1)	The x values identify the desired DL/I function.

// ASSGN
// TLBL
 	
or
 	
// DLBL
// EXTENT
(Note 2)
	SYS011,cuu
	LOGOUT
	
	DSKLOG1 or 2
	extent file
		These statements define the output log file. If tape is used, it must contain standard labels. If disk is used, it must be a previously defined VSAM file(s). LOGOUT is the symbolic name for tape logging. DSKLOG1 and DSKLOG2 are the symbolic filenames for disk log files. If only one file is used, it must be DSKLOG1. If two files are used, two sets of DLBL statements must be supplied .
		
		If no output log file is desired, bit 6 of UPSI must be set to one and the LOG parameter must be omitted from the DL/I parameter statement.

// ASSGN
// DLBL
// EXTENT
or
// TLBL
(tape)
(Note 2)
	SYSnnn,cuu
	filename
	extent data
	
	SYSnnn,cuu
	filename
		These statements define a data base data set. Statements must be presented for each data set referenced by every data base (DBD) generation which, in turn, is referenced by the PSB, where: nn=logical unit assignment, filename is the symbolic name of the data set (VSAM ACB-name or SAM DTF-name) to be processed. It must be the same as the DD1, DD2, or OVFLW parameter of the DATASET statement for the data base (DBD generation). (ASSGN and EXTENT statements are no longer required for VSAM data sets.) These statements are not required for an MPS batch job.

// EXEC	DLZRRC00, SIZE=nnnK (Note 3)	DL/I initialization module. Refer to DL/I DOS/VS Data Base Administration for storage requirements.
DL/I parameter information (see Figure 1-4).		

Notes:

1. Bits 6 and 7 are not used in MPS batch.
2. These statements are not used for MPS batch.
3. DLZRRC00 is replaced by DLZMPI00 for MPS batch.

Figure 1-3. DL/I Job Control Statements (Batch)

For data base reorganization or logical relationship resolution utility execution:

```
ULU,progname[,dbdname]    (Notes 1, 2)
```

For data base reformatting through HD unload utility execution:

```
PLU,DLZURGU0,psbname
```

For data base data set recovery utility execution:

```
UDR,progname,dbdname [,{buf}] [,HDBFR=] [,HSBFR=] [,TRACE=]
                      {1  }
```

For data base backout utility execution:

```
DLI,DLZBACK0,psbname,[{buf}] [,HDBFR=] [,HSBFR=] [,TRACE=] [,ASLOG=] [,LOG=({TAPE  },{PAUSE  })]
                     {1  }                                             {DISK1 }{NOPAUSE}
                                                                       {DISK2 }
```

For application programs:

```
{DLI}progname,psbname,[{buf}] [,HDBFR={bufno} [,dbdname1,dbdname1,...]] [,...]
{DLR}                 {1  }           {32  }
   [,HSBFR={indno},{ksdsbuf},[{esdsbuf}],dbdname3] [,...] [,TRACE=modname] [,ASLOG=YES]
           {3   } {2     }   {2     }
   [,LOG=({TAPE  },{PAUSE  })]    (Note 3)
          {DISK1 }{NOPAUSE}
          {DISK2 }
```

Where:

buf — specifies the number of data base subpools required for this execution; it can be a numeric value of 1 to 255. If omitted, 1 is assumed. If the number of HDBFR statements is greater than buf, this operand is overridden.

If no buffer pool control options are specified, a subpool consists of 32 fixed-length buffers. The buffer size is usually consistent with the VSAM data base control interval size and may be 512 bytes or any multiple of 512 bytes up to 4096 bytes. The total amount of space for buffers is determined at DL/I system initialization and is based on the value specified in buf or BFRPOOL. This is the number of data bases, and size of the VSAM control intervals. A data base is assigned a subpool containing buffers that are equal to or greater than the size of the data base control interval.

dbdname — specifies a 1 to 7 alphameric character name of the DBD for the data base to be accessed by the utility program, if required. This parameter is not required for the utilities:

- DLZURPR0 — Data base preorganization
- DLZURGS0 — Data base scan
- DLZURGP0 — Data base prefix update

HDBFR — describes one DL/I subpool. (Note 4)

- bufno — specifies the number of buffers to be allocated for this subpool and is a numeric value of 2 to 32. If omitted for a specific subpool, 32 is assumed.
- dbdname1,dbdname2, . . . , — specifies the names of DBDs that are allocated to this subpool.

The number of subpools allocated is always greater than or equal to the number of HDBFR operands. If the number of subpools specified by the user is smaller than the number of HDBFR operands, the number of subpools is increased by the system to the number of HDBFR operands. If there are data bases that are not assigned to a subpool by HDBFR operands and there are no unassigned subpools available, one additional subpool with 32 buffers is allocated and all the remaining data bases are assigned to it. Its buffer size is the largest one required by these data bases. If the number of subpools specified is greater than the number of HDBFR operands, the remaining data bases are assigned to the additional subpools. If there are still subpools available, one is used, if required, for delete work areas.

HSBFR —defines VSAM buffer allocation for HISAM, SHISAM, and INDEX data bases. (Note 4)

- indno — specifies the number of index buffers for KSDS.
- ksdsbuf — specifies the number of data buffers for a KSDS.
- esdsbuf — specifies the number of data buffers for an ESDS (applies to HISAM only).
- dbdname3 — is the name of the HISAM, SHISAM, or INDEX DBD referenced by the application program.

Figure 1-4 (Part 1 of 2). DL/I Parameter Information

LOG – specifies the type of logging to be used. (Notes 4, 5)

- DISK1 – indicates the log records are to be written on one disk extent with the filename DSKLOG1.
- DISK2 – indicates that the log records are to be written on two disk extents. If one disk extent becomes full, the extent is closed and the other extent is used. DSKLOG1 is used first then DSKLOG2. If DSKLOG2 becomes full, logging switches back to DSKLOG1 and continues to repeat the sequence.
- NOPAUSE – indicates that reusing a log extent or switching log extents is done without notifying the operator.
- PAUSE – indicates that before reusing the only disk extent (DISK1) or before switching to the next extent (DISK2), the operator is notified and the partition waits for the operator's reply. PAUSE is the default if the second option in the LOG parameter is omitted.
- TAPE – indicates the log records are to be written to a tape device. It is the default if the LOG parameter is omitted.

progname – specifies a 1 to 8 alphameric character name of the application program or utility to be executed.

psbname – specifies a 1 to 7 alphameric character name of the PSB indicated in the PSB generation and referenced by the application program.

TRACE – indicates that tracing is active during execution. (Note 4)

ASLOG=YES – specifies that asynchronous logging is to be used. (Note 4)

Notes:

1. For reload restarts ULR replaces ULU.
2. For HD reorganization unload and HD reorganization reload, buf, HDBFR=, HSBFR=, and TRACE= are optional entries.
3. DLR is the function code used for the MPS Restart facility. If it is specified on a batch DL/I parameter statement, it is treated the same as DLI.
4. They are ignored in MPS.
5. The UPSI byte (bit 6 = 0) must be set to indicate DL/I logging is required.

Figure 1-4 (Part 2 of 2). DL/I Parameter Information

PARTITION AND CONTROL BLOCK RELATIONSHIP

The DL/I DOS/VS Logic Manual, Volume 1, describes the DL/I partition in a batch environment and illustrates the relationship between the control blocks. This description may be found in "Section 5: Data Areas".

DBD GENERATION

A data base description (DBD) is the DL/I control block that describes the layout data structure and physical storage organization of a data base. Each data base must have a DBD associated with it. All of the information about the data base is available in its DBD. From this pool of information other DL/I control blocks are built. Many errors result from a bad DBDGEN. DBD generation control statement information can be obtained from the DL/I DOS/VS Reference Summary: System Programming, SX24-5104.

Confirming What Has Been Generated

The DOS/VSE Assembler language listing of the DBD macro expansion created during DBD generation contains a code confirming what options have been generated for each segment in the data base. For example, this code -- when interpreted -- can tell you which pointer options were generated, whether physical child pointers have been reserved in the segment's parent prefix, and how many physical children are related to the segment.

The code for each segment of the DBD generation is contained in the output of the segment table as segment flags. The segment flags appear in the output as an Assembler language define constant (DC) statement. The constant is defined as four decimal numbers followed by the comment SEGMENT FLAGS. Each number represents a hexadecimal byte. To interpret the code, convert the first three bytes to binary representation as shown in the following example. An explanation of each byte is given in Figure 1-5.

Byte	Converted Value	Description
0		**Pointer Positions Generated**
	1...	Logical child counter
	.1..	Physical twin forward
	.11.	Physical twin forward and backward
	...1	Physical parent
	.... 1...	Logical twin forward
	.... 11..	Logical twin forward and backward
	.... ..1.	Logical parent
	1	No twin
1		**Segment Processing Rules**
	10..	Insert physical
	01..	Insert virtual
	11..	Insert logical
	..10	Delete physical
	..01	Delete virtual
	..11	Delete logical
	.... 10..	Replace physical
	.... 01..	Replace virtual
	.... 11..	Replace logical
	10	Insert nonsequential first
	01	Insert nonsequential last
	11	Insert nonsequential here
2		
	.xxx .xxx	Reserved
	1...	Segment is paired
	.... 1...	Segment's parent has two physical child pointers
3		
	0-254	Number of physical children of this segment

Figure 1-5. DBD Generation Segment Flag Codes

EXAMPLE: Output from DBD generation contains the following statement for a particular segment:

```
DC    AL1(96,169,8,13)  SEGMENT FLAGS.
```

Convert the values to binary and decimal representations:

Byte 0	Byte 1	Byte 2	Byte 3
96	169	8	13
01100000	10101001	00001000	13

Figure 1-5 shows these values indicate the following for the segment:

Byte 0 Segment has physical twin forward and backward pointers.

Byte 1 Segment has physical insert, delete, and replace rules, and the rule for placement on the physical twin chain is last.

Byte 2 Two 4-byte fields are reserved for physical child pointers in the parent of this segment.

Byte 3 This segment is the parent of 13 physical child types.

PSB GENERATION

Before an application program can be executed under DL/I, it is necessary to describe that program and its use of logical data structures through a program specification block (PSB) generation. This PSB is used as input to the application control blocks creation and maintenance utility program in conjunction with one or more DBDs to create internal DL/I control blocks.

PSB generation control statement information can be obtained from the DL/I DOS/VS Reference Summary: System Programming, SX24-5104.

APPLICATION CONTROL BLOCK CREATION AND MAINTENANCE

Before the DL/I system can use the program and data base descriptions created by the PSB and DBD generation utility programs, they must be merged and expanded to an internal format. This merge and expansion is called block building or ACBGEN. Figure 1-6 illustrates the input and output of the application control blocks creation and maintenance utility program.

This utility is composed of eight modules. A complete description of these may be found in the DL/I DOS/VS Logic Manual, Volume 1.

The following points should be noted:

- A PSB will always be generated for each PSB in the BUILD statement. The generated PSB name is the returned PSB name padded with at-signs (@) to seven characters and ending with a "P".
- Creation of DMBs referenced by the PSB may be either suppressed, unconditional, or done only if it is not already present in the core image library. The generated DMB name is the returned DMB name padded with at-signs (@) to seven characters and ending with a "D".

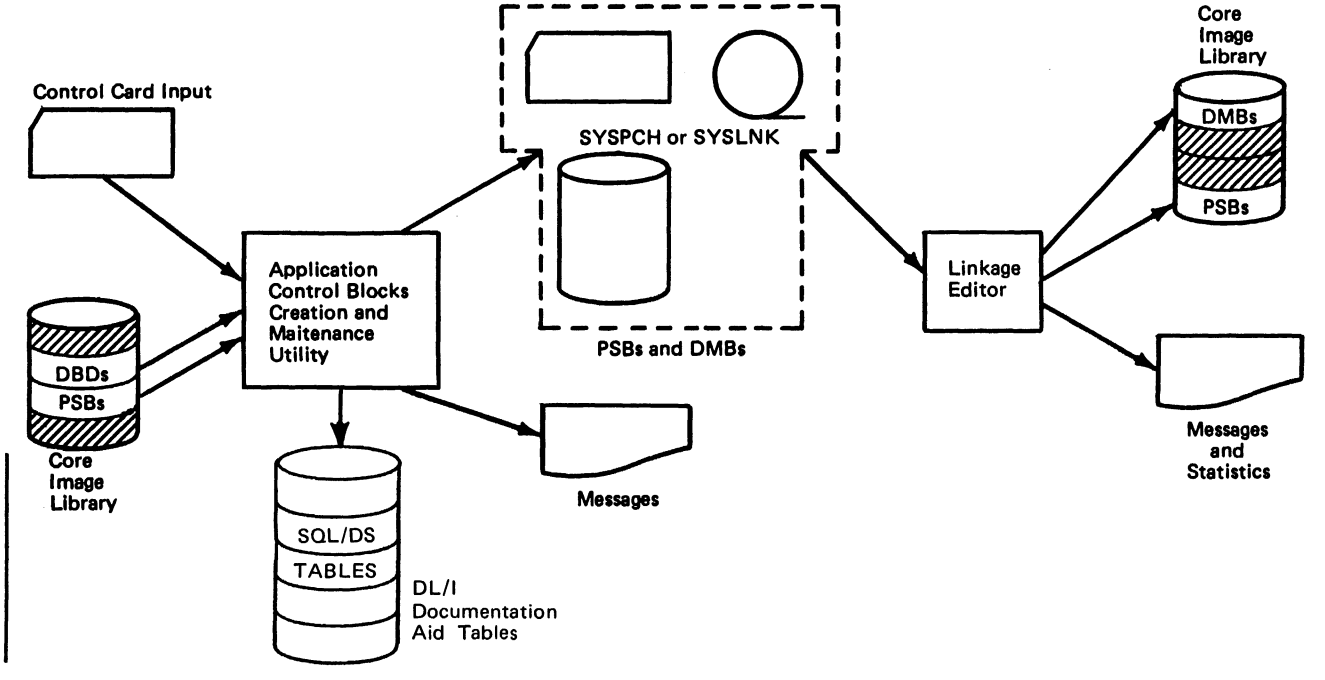


Figure 1-6. Application Control Blocks Creation and Maintenance Utility Program

- A utility program PSB will be built for each DMB which is created. The generated PSB name is the returned PSB name padded with at-signs (@) to seven characters and ending with a "U".
- If the DL/I Documentation Aid facility is selected, DBD and PSB information is collected and written to SQL/DS tables.

Note: The linkage between these modules is by standard format save area.

DL/I ONLINE WITH CICS/VS

The DL/I nucleus is loaded during CICS/VS initialization. The DL/I nucleus contains tables and DL/I-CICS/VS interface modules.

SYSGEN PARAMETERS

SYSGEN parameters for DL/I-CICS/VS are shown in Figure 1-7 . Additional information may be found in:

- CICS/DOS/VS Installation and Operations Guide
- DL/I DOS/VS Resource Definition and Utilities

The operation of DLZNUCxx is described in "Online Operation Programs" on page 4-19.

Also, during CICS/VS initialization, the DL/I initialization module (DLZOLI00) is attached. When this has initialized DL/I, it returns to CICS/VS initialization (see Figure 4-13 on page 4-20). DLZOLI00 is referred to as DFHSIDL by CICS/VS initialization overlay supervisor.

The third online module invoked by CICS/VS is DLZSTP00. It is called during CICS/VS termination.

```
CICS/VS DFHSIT *DL/I = xx        Defines which DLZNUCxx CICS/VS will load.
               *DL/I = YES       Allows the following in CICS initialization:

                                  (1) Load DLZNUCxx
                                  (2) Load DFHSIDL (DL/I initialization loaded
                                      by CICS/VS overlay supervisor)
        DFHFCT *ACCMETH = DL1
               *OPEN    = INITIAL  Data base opened by DLZOLI00.
                     or = DEFERRED DB opened dynamically by CALLDLI
                                   with STRT functions (not by CSMT).
        DFHSG     *TYPE = INITIAL  Cause DL/I services to be included in
                  --DLI = YES      CICS/VS programs.
        DFHPLT    *TYPE = ENTRY    DLZSTP00 is entered at CICS termination
                PROGRAM = DLZSTP00 to purge/close DBs. This entry is required
                                   in part 2 of the PLT.
DL/I    DLZACT *VARIOUS PARAMs     Provides the following:
                                    (1) Suffix for DLZNUCxx
                                    (2) Maxtask bufferpools values
                                    (3) Application program and PSB linkage
                                   DLZACT generation produces the DL/I DLZNUCxx
                                   (see Figure 1-8)
```

Figure 1-7. DL/I-CICS/VS SYSGEN Parameters

DL/I CONTROL BLOCK RELATIONSHIP

The relationship between the DL/I online control blocks is identical with that in a batch system except for the additional online control blocks and their physical location. These control blocks are contained in the nucleus (DLZNUCxx), are assembled as part of ACTGEN, and are discussed below. Refer to Figure "DL/I Control Block Relationship" in the DL/I DOS/VS Logic Manual, Volume 1, and Figure 1-8 on page 1-10.

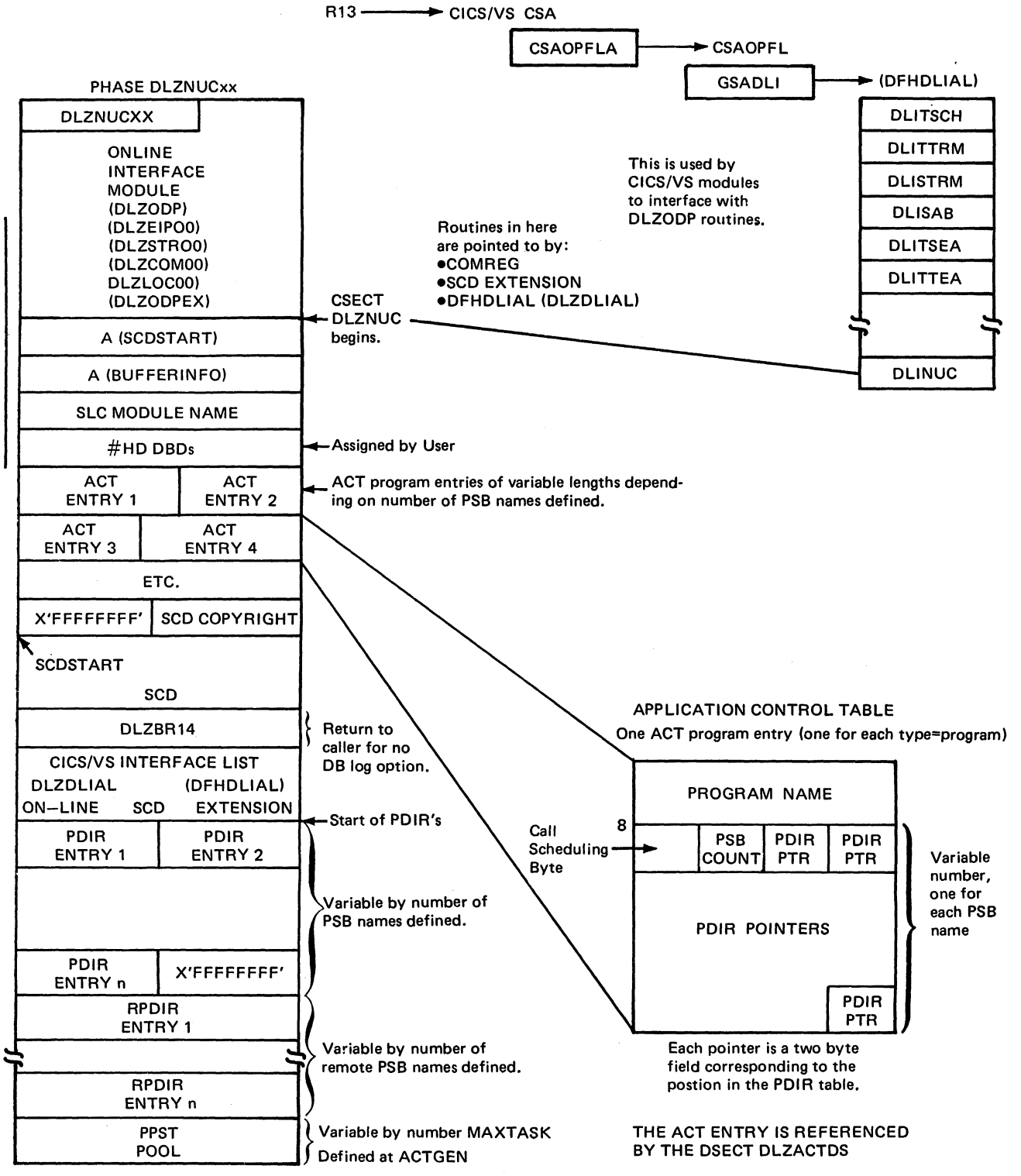


Figure 1-8. General Layout of Control Blocks in the DL/I Nucleus

The application control table (ACT) is used primarily by DL/I online at CICS/VS initialization time to verify and load all referenced PSBs and DMBs. It is used in scheduling to verify program types and by default scheduling to acquire the first PSB name.

The system contents directory (SCD) contains copyright information and the online extension which is composed of:

- Logger ECB.
- System ENQ and function ECBs.
- Address of the system password.
- PPST (PST Prefix) chain address (forward and backward, active and free).
- Address of the VSAM asynchronous exit routine (DLZOVSEX).

The active PPSTs are chained to the TCA of the task. See Figure 1-9 on page 1-12 for a description of active PPST chaining.

The PSTs are acquired by DL/I at task scheduling time by use of a CICS/VS GETMAIN. They reside in CICS/VS dynamic storage and are chained off the TCA of the task. The segment intent list (one per PCB per physical data base) is used to ensure that the segment PROCOPT intents requested against a physical segment do not conflict. (Refer to Appendix C for more information.)

APPLICATION CONTROL TABLE GENERATION (ONLINE ONLY)

The ACT is the major control block used by online DL/I. It contains the name of programs that may use DL/I facilities. It also contains all PSB names that these programs will use, all PPSTs, the SCD, and the PDIR.

The CSECT into which the ACT is assembled is established by means of the DLZACT TYPE=INITIAL statement. This must be the first statement in the ACT generation:

```
DLZACT  TYPE=INITIAL,
        SUFFIX=xx
```

This is followed by:

```
DLZACT TYPE=CONFIG,
       MAXTASK=nnn,
       CMAXTSK=nnn,
       BFRPOOL=nnn,
       PASS=password,
       SLC=phname,
       PI=xxx,
       REMOTE=xxx
```

where:

MAXTASK

causes generation of PPSTs equal to nnn. This is the maximum number of tasks that can be handled in an online system. Values of 1 to 255 are permissible.

CMAXTSK

specifies the number of transactions that can be handled in an online system. It may be less than, or equal to, MAXTASK. This value can be changed dynamically using the CMXT system call function.

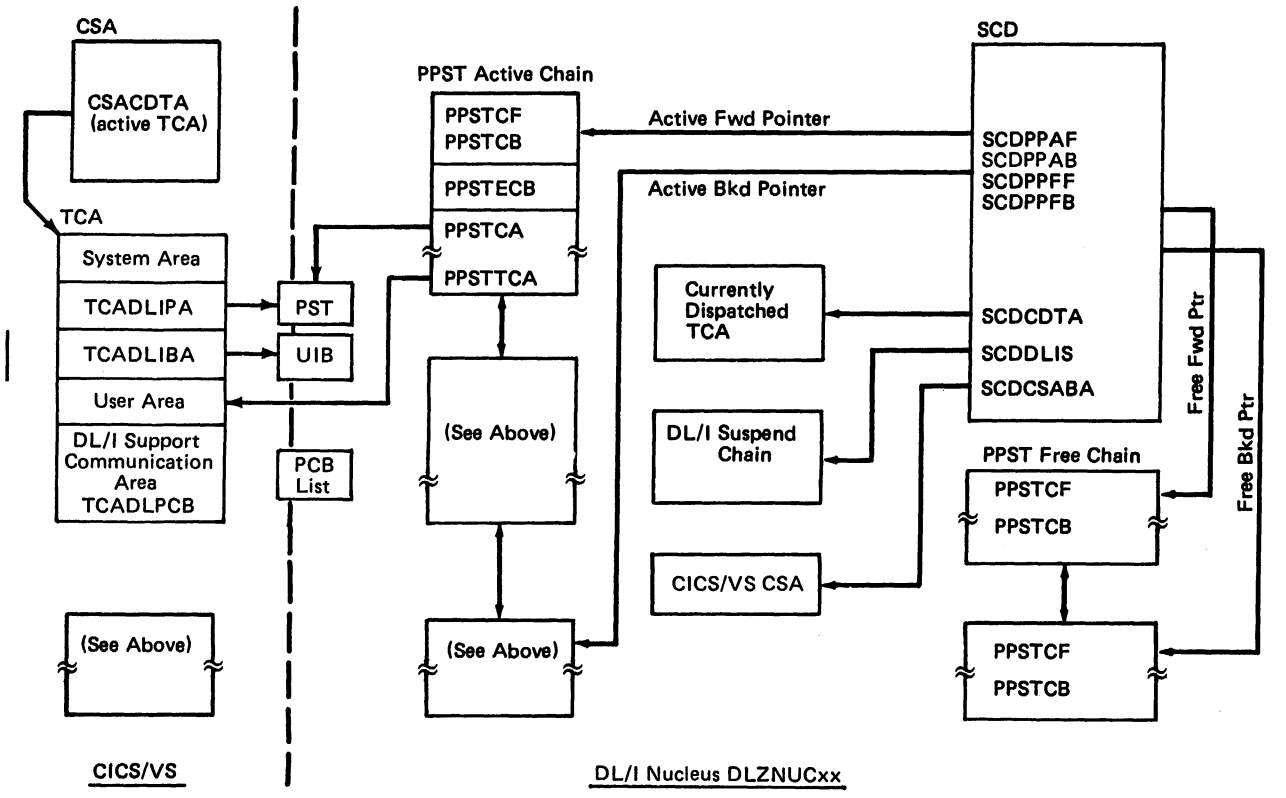


Figure 1-9. Chaining of Active PPST

BFRPOOL

specifies the number of subpools (0 to 255) to be acquired by online initialization. If 0 is specified, initialization will ask for number of buffers.

PASS

specifies special name (1 to 8 characters) that allows a transaction to issue system calls, i.e., STRT, STOP, CMXT. The default is 'DLZPASS1'.

SLC

specifies the phase name of the storage layout control table to be used.

PI

specifies the inclusion or exclusion of program isolation.

Note: PI=NO must be specified when an RPSB with local component has been specified (DLZACT TYPE=RPSB LNAME operand).

REMOTE

enables PSBs defined in the local system's DL/I nucleus to be accessed from remote systems through the CICS/VS DFHMIR mirror program.

The next required statement describes the logical connection between a CICS/VS application program and a DL/I data base. There can be 1 to 255 DLZACT TYPE=PROGRAM statements in an ACT generation:

```
DLZACT  TYPE=PROGRAM,
        PGMNAME=name,
        PSBNAME=(name,name,...)
        [,CONT=YES]
```

where:

PGMNAME

specifies the name of the application program which is authorized to schedule a DL/I PSB. This should be the same as that defined in the PPT.

PSBNAME

specifies the PSBNAME(s) associated with this task. 1 to 7 characters are permitted. The first PSBNAME is the default PSBNAME for this application. (There cannot be more than 4095 PSBNAMES defined in the entire ACT generation.)

CONT=YES

specifies that the next DLZACT statement defines more PSB names for the same program that is, it will contain only the PSBNAME= operand and possibly CONT=YES again. This operand has been introduced to circumvent the Assembler restriction of allowing not more than 255 characters in a parameter sublist.

In MPS, a TYPE=PROGRAM statement must be provided for the Batch Partition Controller, DLZBPC00, specifying all PSB names that will be used by any batch jobs running under MPS.

If CICS/VS intersystem communication support is to be used to access a data base resident on another CPU, a DLZACT TYPE=RPSB statement must immediately follow the last DLZACT TYPE=PROGRAM statement. The format of this statement is:

```
DLZACT TYPE=RPSB,
       PSB=psbname,
      [LNAME=localname,]
       RNAME=remotename,
       SYSID=systemid,
     [,LANG=pli]
```

where:

PSB

specifies the name of a PSB specified on any previous DLZACT TYPE=PROGRAM statement. It must not be the same name specified by either LNAME or RNAME.

LNAME

specifies the name of the PSB in the local system that is to be used with the PSB in the remote system to produce a remote PSB with a local component.

RNAME

specifies the name given to the PSB in the remote system -- remotename cannot be defined as an RPSB with a local component in the remote system.

SYSID

Specifies the four-character identifier of the CICS/VS system to which the remote PSB and associated data bases are assigned.

LANG

specifies that the remote PSB is to be used by an MPS application program written in PL/I (pli may be coded as PL/I, PLI, PL/1, or PL1).

Intersystem communication support is invoked when the DL/I program request handler (PRH) detects a remote PSB during a scheduling call.

When generating a DL/I nucleus to process requests from another system, a DLZACT TYPE=PROGRAM statement must be supplied for a mirror transaction, PSBNAME=DFHMIR. The mirror transaction issues DL/I calls on behalf of the other system. Therefore, the PSBNAME parameter must include all PSB names which can be referenced by any other system. In order to simplify nucleus generation for processing requests from other systems, an optional parameter, REMOTE=YES, can be specified on the DLZACT TYPE=CONFIG statement.

The next (optional) statement specifies buffer pool control options and is equivalent to the parameter statement keywords in batch:

```
DLZACT TYPE=BUFFER,
       {HDBFR=(bufno[,dbdname1,dbdname2,...])[,HSBFR=...] }
       {HSBFR=(indno,ksdsbuf,[esdsbuf],dbname)[,HDBFR=...]}
```

where:

HDBFR

describes one DL/I subpool:

- bufno specifies the number of buffers to be allocated for this subpool and is a numeric value from 2 to 32. If omitted for a specific subpool, 32 is assumed.
- dbdname1,dbdname2,... specify the names of DBDs that are to be allocated to this subpool. :

HSBFR

defines VSAM buffer allocation for HISAM and INDEX data bases.

- indno specifies the number of index buffers for a KSDS (default is 3).
- ksdsbuf specifies the number of data buffers for a KSDS (default is 2).
- esdsbuf specifies the number of data buffers for the ESDS (default is 2) -- applies to HISAM only
- dbdname is the name of the HISAM or INDEX DBD referenced by the application program.

The last statement must be:

```
DLZACT  TYPE=FINAL
```

This signifies the end of ACT generation.

The ACT generation is run as a standard VSE system job and requires the following job control statements:

```
//  JOB NUCGEN
//  OPTION CATAL
//  EXEC ASSEMBLY
    DLZACT TYPE=INITIAL
    DLZACT TYPE=CONFIG
    DLZACT TYPE=PROGRAM
          .
          .
          .
          .
    DLZACT TYPE=FINAL
/*
    ENTRY DLZNUC    *REQUIRED
//  EXEC  LNKEDT
/&
```

For a full discussion of DLZACT generation, see DL/I DOS/VS Resource Definition and Utilities.

DL/I CONTROL BLOCK RELATIONSHIP

You can locate the active DL/I task and the DL/I control blocks from the SCD. The SCD is the major control block in the DL/I system. The location of the SCD in the DL/I nucleus is shown in Figure 1-8 on page 1-10 . The relationships of the various DL/I control blocks are shown in Figure 1-10 on page 1-16 . To locate the offset for an address or pointer, refer to the appropriate control block description in the DL/I DOS/VS Logic Manual, Volume 1.

DL/I REMOTE DATA BASES

Using CICS/VS intersystem communication support, DL/I application programs can access data bases that are resident on another CPU, referred to as a remote CPU. The system on which the application program is executing is referred to as the local system. The system programmer, when generating the DL/I nucleus for the local system, specifies the location of the remote data base(s) by giving the name of the remote PSB in a DLZACT TYPE=RPSB statement. A remote PSB, and its related DMBs and data bases, must all reside on the same system.

This support enables the DL/I user to define and maintain data bases where the most activity for that data base occurs and still provide access to the data base from an interconnected system. This support also enables users to centralize their data bases while distributing the processing of this data to other interconnected systems.

Most DL/I application programs are not affected by intersystem communication support. Those affected are:

- MPS batch application programs which issue their own PSB scheduling calls receive an abnormal return code (X'08') in the UIBFCTR and return code (X'09') in the UIBDLTR (or corresponding TCA fields if not using the extended call interface).
- It is possible for a DL/I call in a remote system to have originated from an MPS batch partition. If program isolation is not used in the remote system, any task in scheduling conflict with the MPS-originated task might wait a long time because it is not recognized as an MPS-originated task in the remote system.
- Application programs that use the CALL interface and Boolean operators are restricted regarding the length of a segment search argument (SSA) that can be transmitted to a remote DL/I system. The maximum size SSA that can be transmitted is 304 bytes. Boolean SSAs with a length greater than 304 bytes will be rejected in the remote system causing the DL/I call to fail.

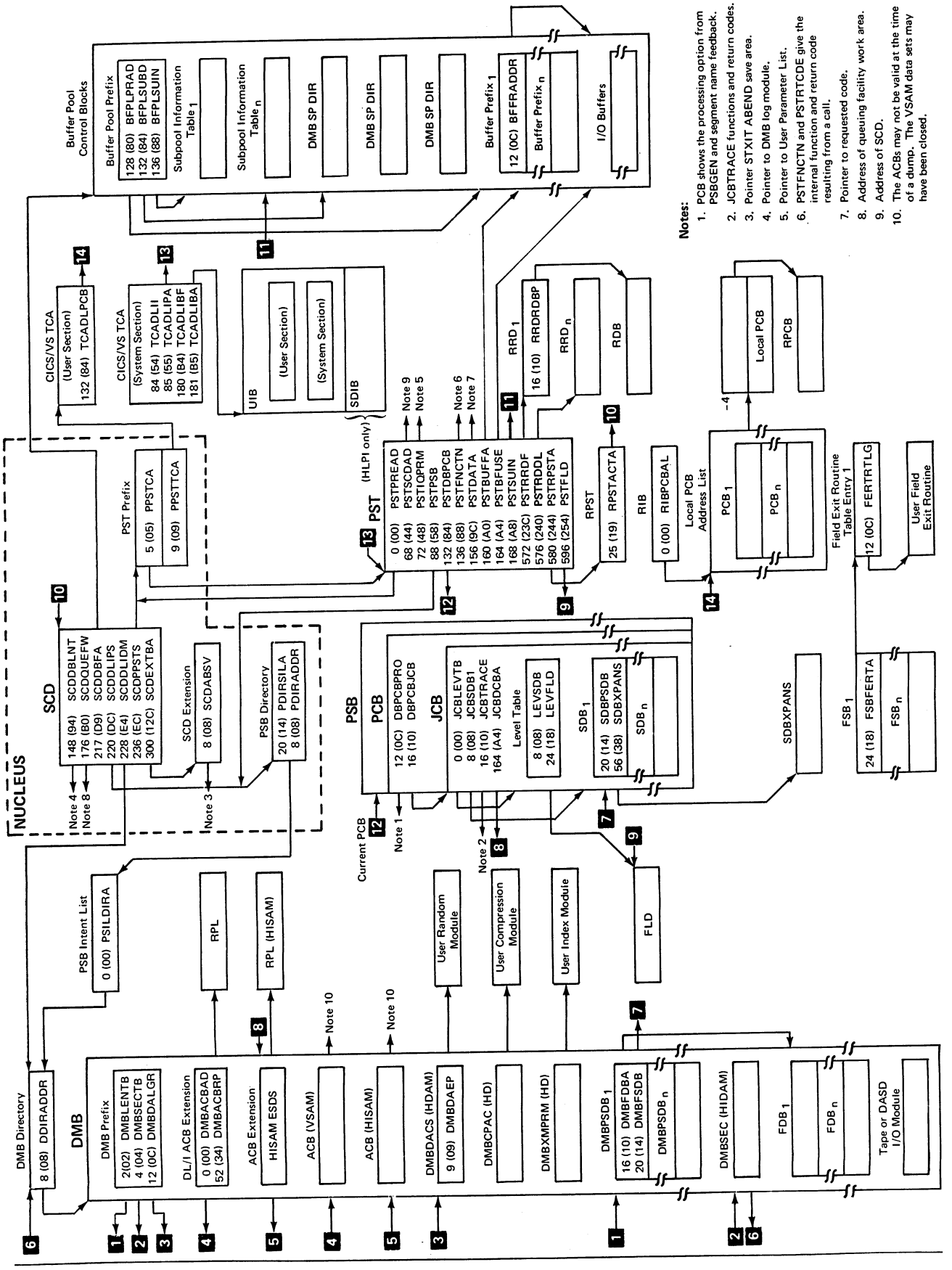


Figure 1-10. DL/I Control Block Relationship

Extended Remote PSB

Extended Remote PSB permits definition of an RPSB (DLZACT TYPE=RPSB) as a concatenation of a single local PSB and a single remote PSB. This allows DL/I online and MPS-batch application programs simultaneous access to DL/I data bases located on two DL/I systems within the same logical unit of work.

This capability is particularly applicable where:

- one or more satellite CICS/VS systems with DL/I DOS/VS are connected to a single central CICS/VS system with DL/I DOS/VS or IMS/VS.
- most of the data required by satellite application programs is resident on the local satellite system, with some remaining data resident on the remote central system.

Conditions:

- distribution of a single DL/I data base across multiple systems is not supported; therefore, remote data bases must be wholly resident on a single remote system.
- distribution of logical relationships or secondary indexes across multiple systems is not supported.
- program isolation (PI) must be used in the DLZACT generations of the local system.
- use of an extended RPSB as the remote component of an extended remote PSB is not supported; this is an example of what is called a "chained mirror" configuration.

Other considerations:

- the DFHPCT TYPE=ENTRY statement for the CICS/VS mirror transaction, DFHMIR, in each remote system should have a DTIMOUT value specified to guard against resource deadlocks.

INTERFACES

Local DL/I to CICS/VS ISC

The address of the CICS/DOS/VS intersystem communication interface module (DFHISP) is provided to DL/I in CSA field CSADISAC. On entry to this module:

- Register 1 contains the address of the parameter list.
- Register 13 contains the address of the CSA.
- Register 14 contains the DL/I return address.

Register 1 points to one of the following parameter lists:

- Scheduling call
  - Address of X'00'
  - Address of RIB (remote interface block)
  - Address of user's scheduling call parameter list
  - Address of PDIR entry
- Data base call
  - Address of X'04'
  - Address of RIB
  - Address of user's data base call parameter list
  - Address of PDIR entry
- Termination call

- Address of X'08'
- Address of RIB

CICS/VS ISC to Remote DL/I PRH

CICS/VS ISC invokes the DL/I program request handler in the remote system through the ISC mirror transaction, DFHMIR. The mirror transaction issues calls to DL/I through the Assembler language interface, if required. The mirror transaction must be defined to CICS/VS as not being eligible for automatic restart after an abnormal termination. If an IOAREA address parameter is included in the user's call parameter list, the IOAREA address is set to zero before issuing the DL/I calls to prevent duplicate data moves.

Remote DL/I PRH to CICS/VS ISC

Since the IOAREA is set to zero before the DL/I call is issued, the DL/I program request handler does not move data into the ISC IOAREA. Instead, it is moved by ISC. The address of the data is in PSBIOAWK and the length of the data is in PSTSEGL of the mirror transaction's PST.

CICS/VS ISC to Local DL/I

On scheduling calls, PSBIOASZ is returned to the local system and stored in RPCBMIOS. This is the amount of data that is transmitted from the user's IOAREA on an ISRT call.

In addition, a local copy of each PCB and its remote PCB (RPCB) are generated by ISC. The addresses of these PCBs are in the PCB address list returned to the user. For each MPS scheduling call indicated by a flag in RIBISCO, ISC generates a local PSB I/O
workarea and stores its address in RIBIOAWK.

On inbound data base retrieve calls, CICS/VS ISC must update RIBSEGL with the length of the data so that the MPS program request handler can move data to the user's IOAREA.

DL/I CALLS

The following functions (on a segment basis) are performed by DL/I:

- Get Unique (Hold) - GU, GHU
- Get Next (Hold) - GN, GHN
- Get Next within Parent (HOLD) - GNP, GHNP
- Insert - ISRT
- Delete - DLET
- Replace - REPL

The following shows calls issued by COBOL, PL/I, RPG II, and Assembler:

```
COBOL       CALL 'CBLTDLI' USING[parmcnt,]
            function[,psbname[,uibparm]]

PL/I        CALL PLITDLI (parmcnt,
            function[psbname[,uibparm]])

RPG II (uses the RQDLI command)
            GU          RQDLI[file-name]  13

Assembler   CALL ASMTDLI,([parmcnt,]
            function[,[psbname][,uibparm]])
```

Note: CALLDLI is used for online; for batch or MPS use CALL in Assembler statement.

DL/I COMMANDS FOR HLPI

The following shows the syntax of HLPI commands used in COBOL or PL/I application programs:

```
trigger function [option with arguments] command-delimiter
```

The following is an example of a command as it would appear in a COBOL program:

```
EXECUTE DLI GET NEXT USING PCB(1) KEYFEEDBACK(SKILLCOD)
                           FEEDBACKLEN(10) SEGMENT(SKILL)
                           INTO(IOAREA) WHERE(CODE=ARTIST) END-EXEC
```

trigger — function — options with arguments — command-delimiter

INITIALIZE Call	This call is generated by the translator at the start of every PL/I and COBOL external procedure. It is not generated at secondary entry points. CALL DLZEI01(arg0,user-dib); CALL 'DFHEI1' USING DFHEIU0 DLZDIB
SCHEDULE Command	CALL DLZEI01(arg0,user-dib,psbname); If psbname is not specified, the translator passes '*' in its place.
TERMINATE Command	CALL DLZEI01(arg0,user-dib);
CHECKPOINT Command	CALL DLZEI01(arg0,user-dib,checkpoint-id);

DATA BASE COMMANDS

One DL/I CALL statement is generated for each segment specified in the DL/I HLPI command. The format of each generated call is as follows:

```
CALL DLZEI102(arg-zero,
              user-dib,
              pcb-index,
              segment name,
              segment I/O area,
              length of I/O area,
              offset of destination parent,
              operators,
              field name
              field value,
              length of field value);
```

Notes on data base calls:

- Omitted arguments are indicated by dummy arguments having no defined values.
- The argument existence bits and/or the flagword determine which arguments are dummies.
- If trailing arguments are dummies, the argument list may be shortened, but there is no guarantee of the X'80' bit for last argument.

- The pcb-index is only passed on data base calls. If not specified, it defaults to one. All subsequent calls generated for the command have the same pcb index as the first call.

- The length of the I/O area argument is the value (or default) of the SEGLENGTH option. It may be a dummy in COBOL. If the path calls are used, the area must be long enough to hold the longest concatenated segment (that is, all segments involved in each path call).

- The arithmetic values (pcb-index, length of I/O areas, offset, number of expressions, length of field value) are all halfword binary.

- Number of expressions, operators, field name, field value, and length of field value are all omitted if WHERE is not specified.

- Number of expressions is always 1 for DL/I DOS/VS (when present).

- The first call generated by the translator for each DL/I HLPI command contains information about the object segment. If additional segments are specified in the command, calls are then generated for these segments in the order in which they appear in the command.

- operators

```
::=DCL 1 OPPAIRS(#pairs),
           2 KFY-OP CHAR(2),     /* 2-BYTE EBCDIC FORMAT.*/
                                 /* SEE THE DL/I APRM    */
                                 /* FOR POSSIBLE VALUES. */
           2 BOOL-OP CHAR(1),    /* 1-BYTE EBCDIC FORMAT.*/
                                 /* BLANK ON LAST PAIR   */
                                 /* AND ALWAYS BLANK FOR */
                                 /* VSE.                 */
     #pairs = number of fields (1 for present DL/I DOS/VS)
```

Figure 1-11 through Figure 1-16 illustrate the control flow in each of the operating environments for each of the HLPI host languages. The arrows between the boxes show the direction of flow; the circled numbers near the arrows show the sequence of flow.

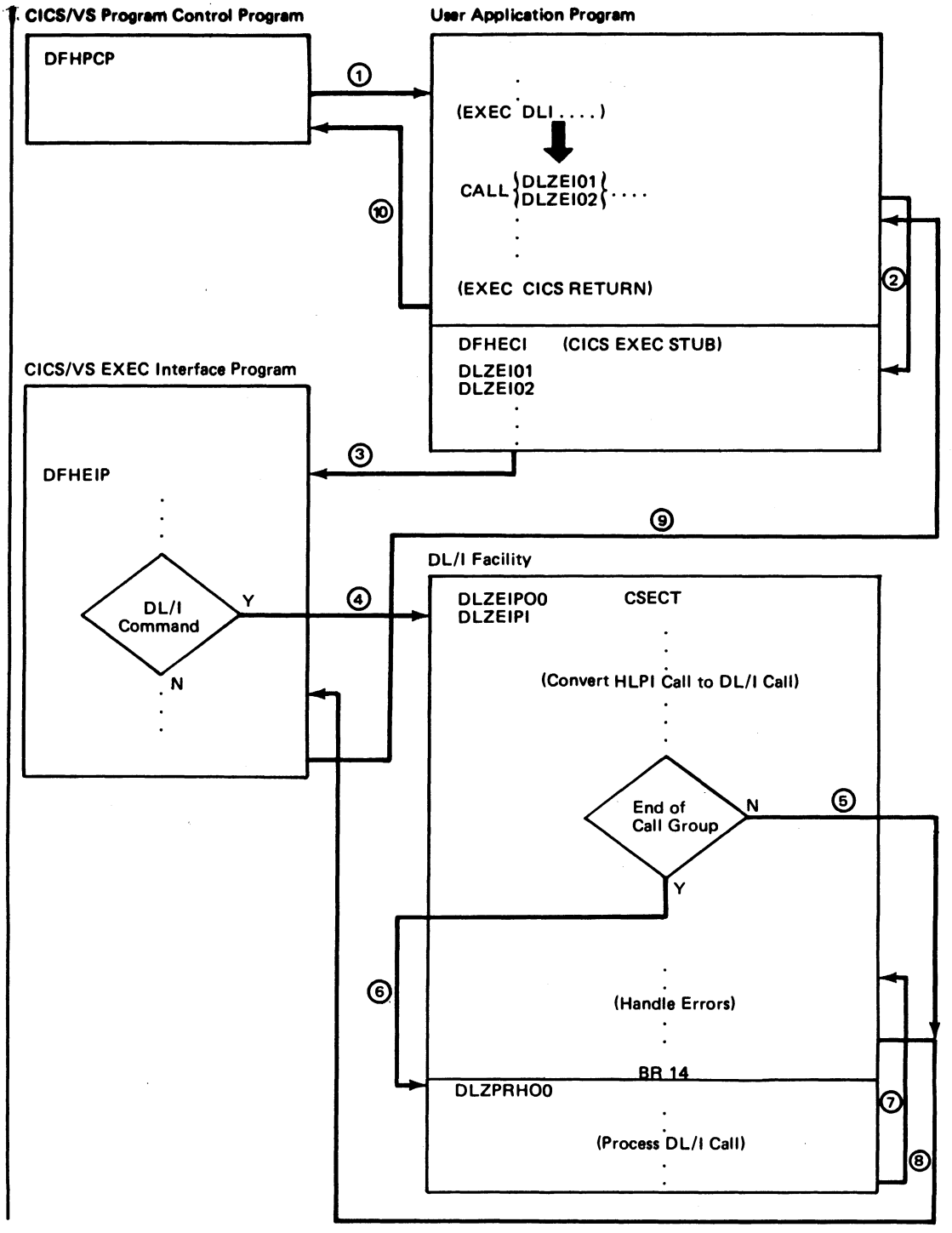


Figure 1-11. PL/I Online Control Flow (CICS/VS)

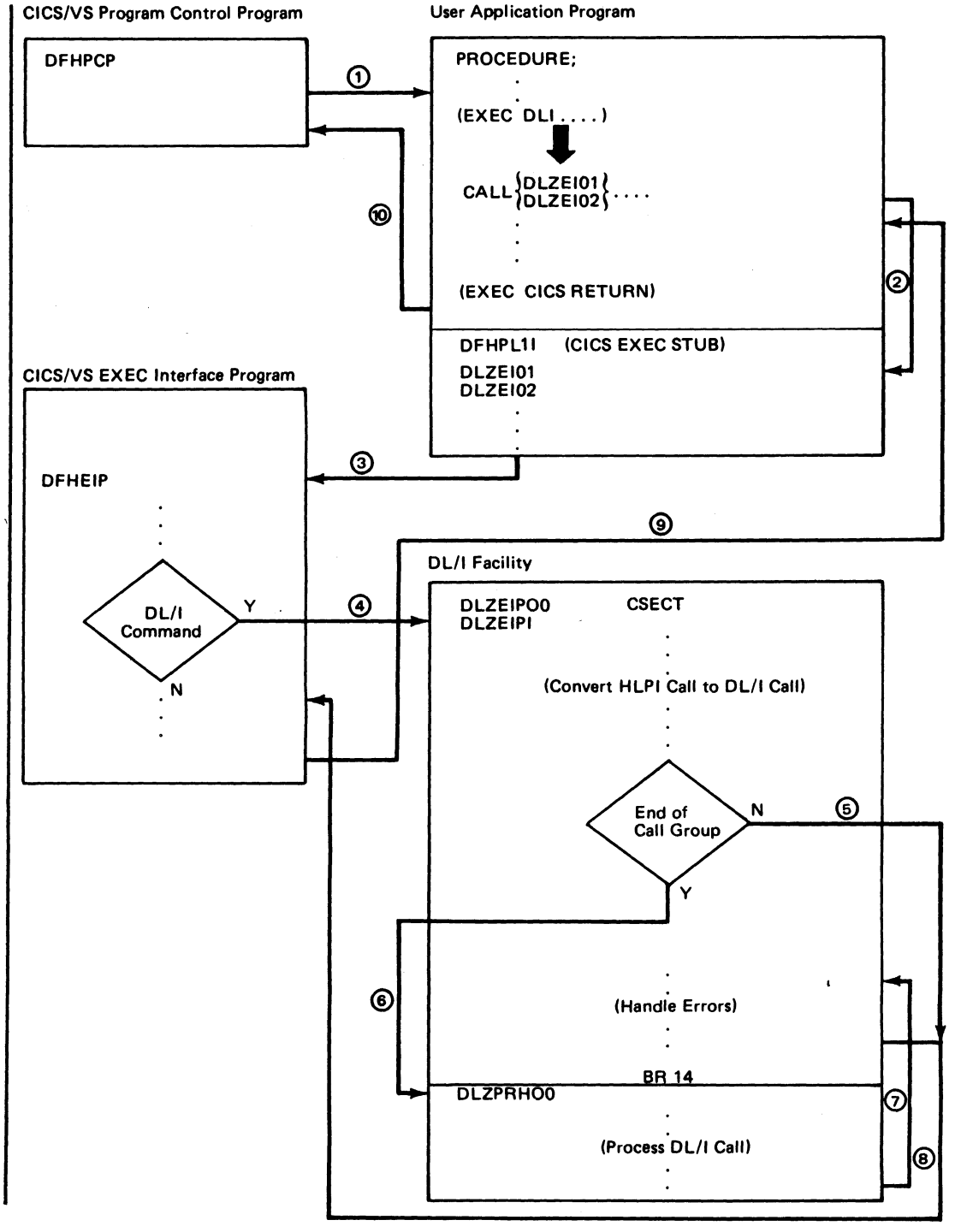


Figure 1-12. COBOL Online Control Flow (CICS/VS)

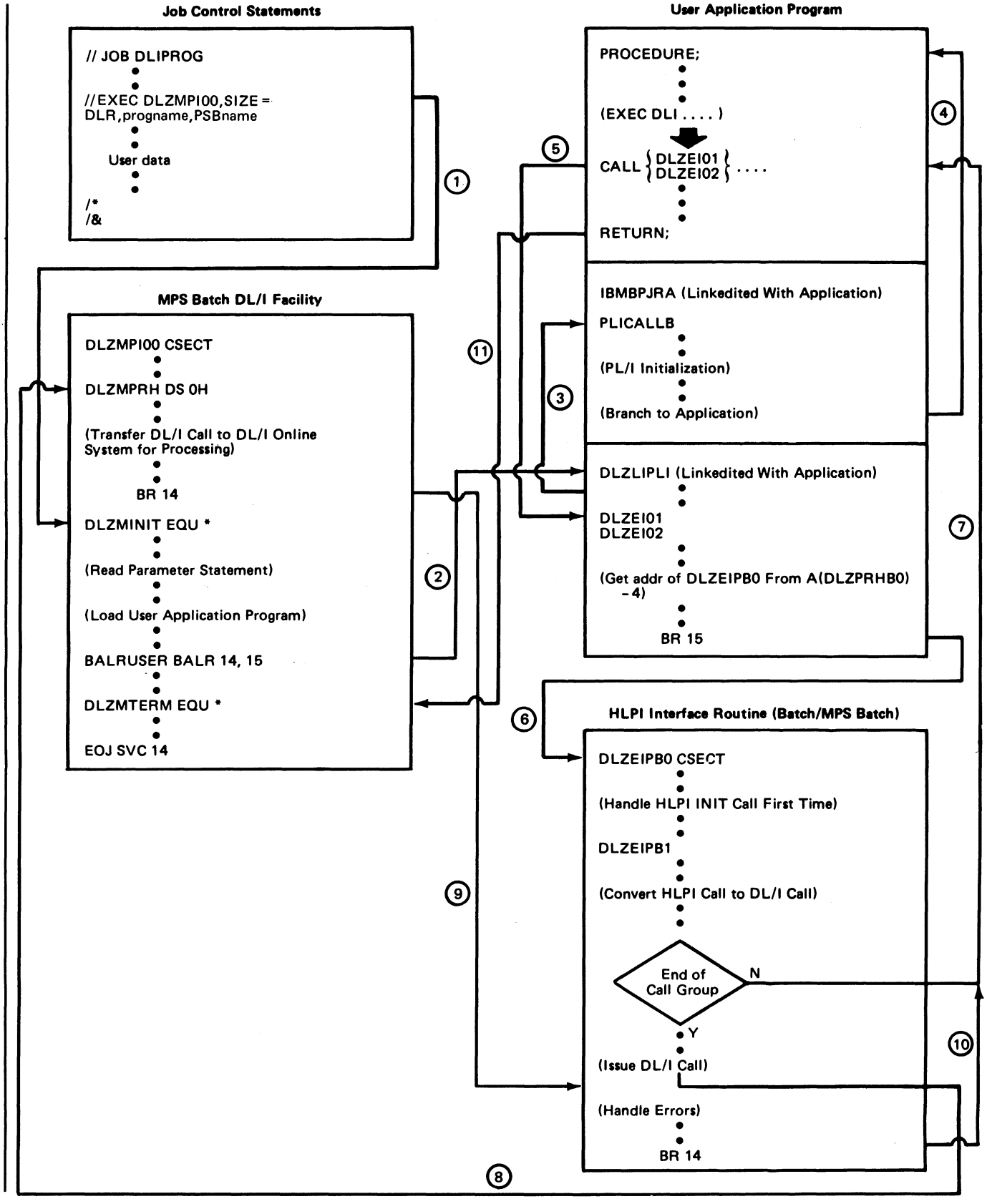


Figure 1-13. PL/I MPS Batch Control Flow

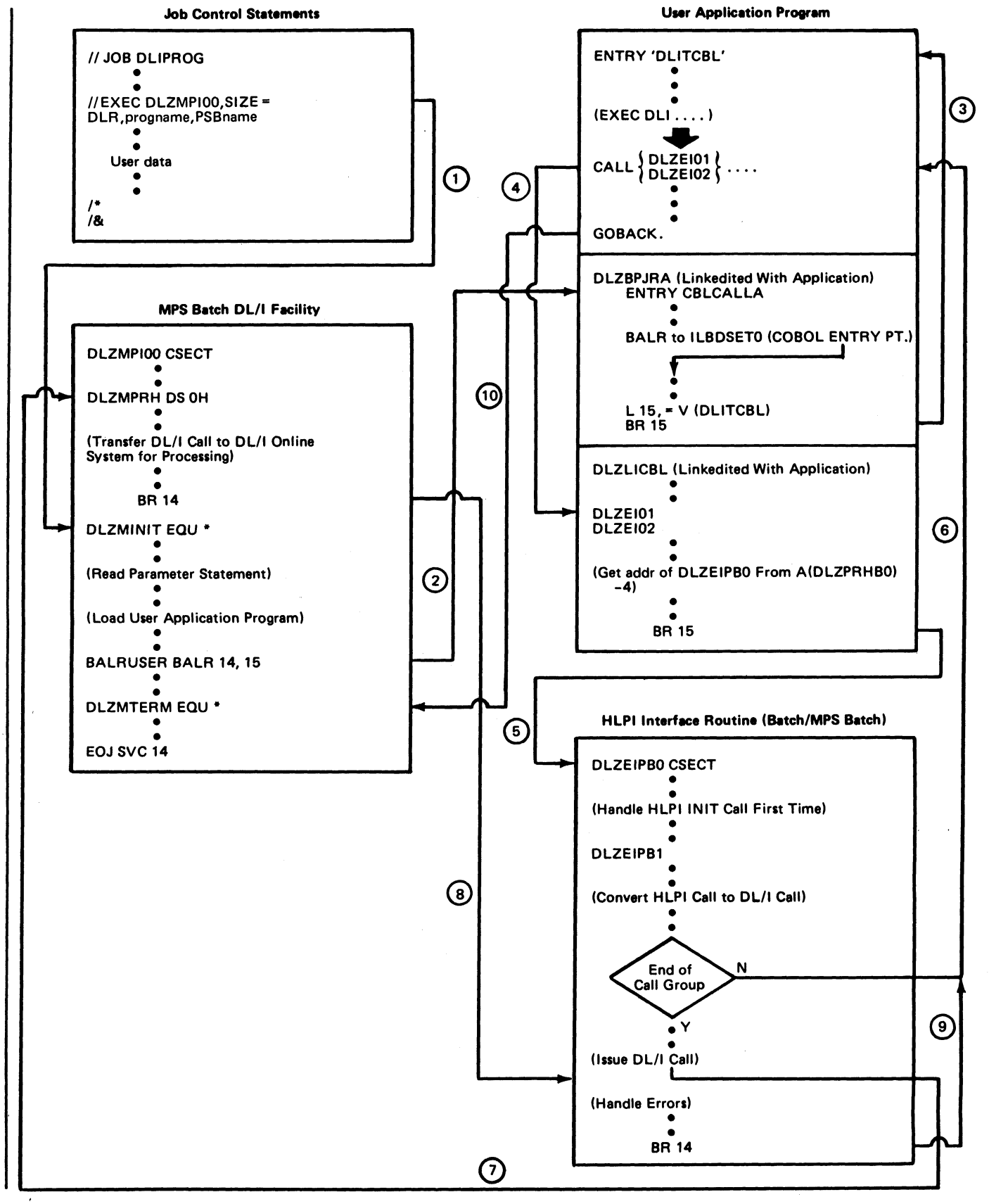


Figure 1-14. COBOL MPS Batch Control Flow

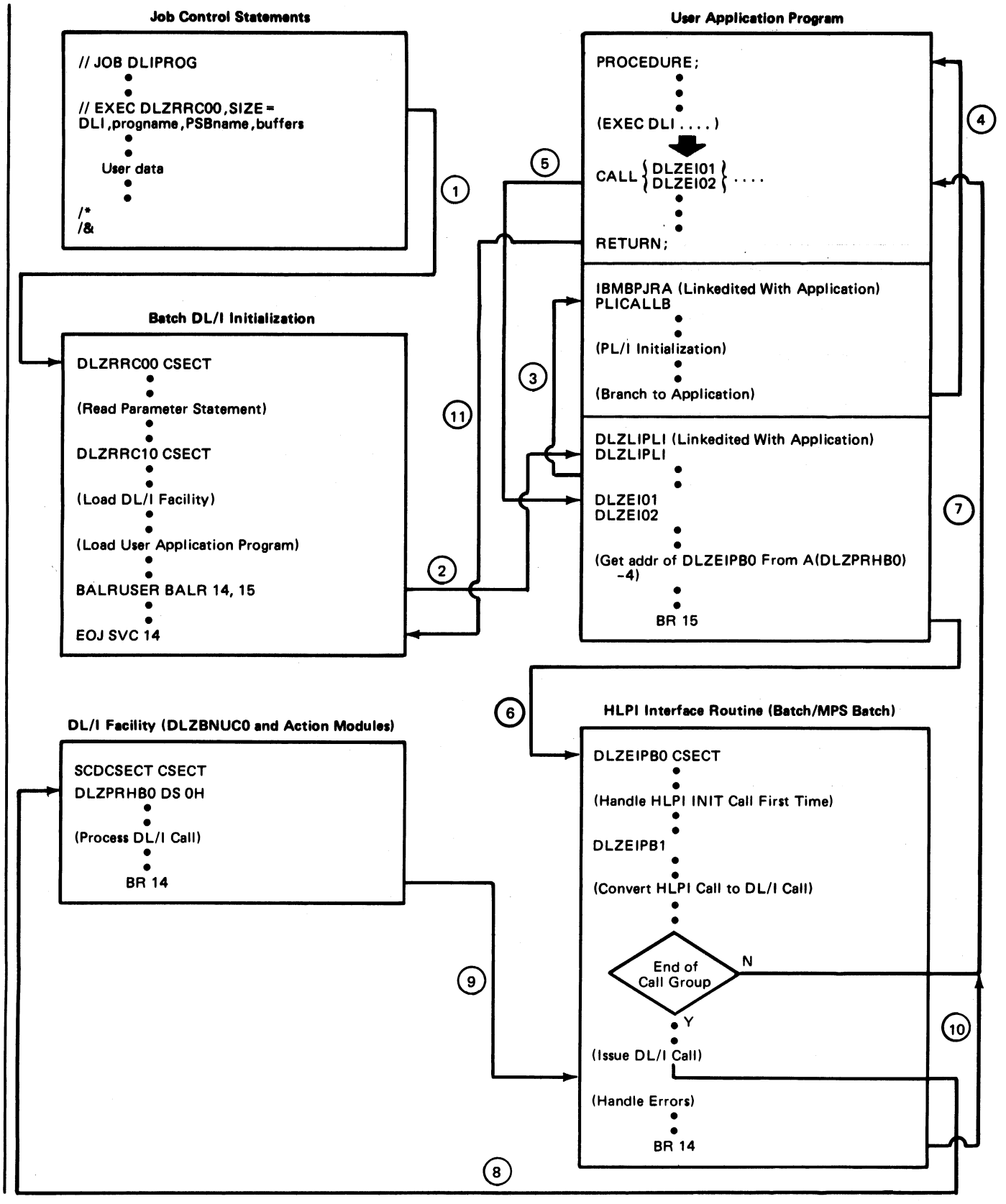


Figure 1-15. PL/I Batch Control Flow

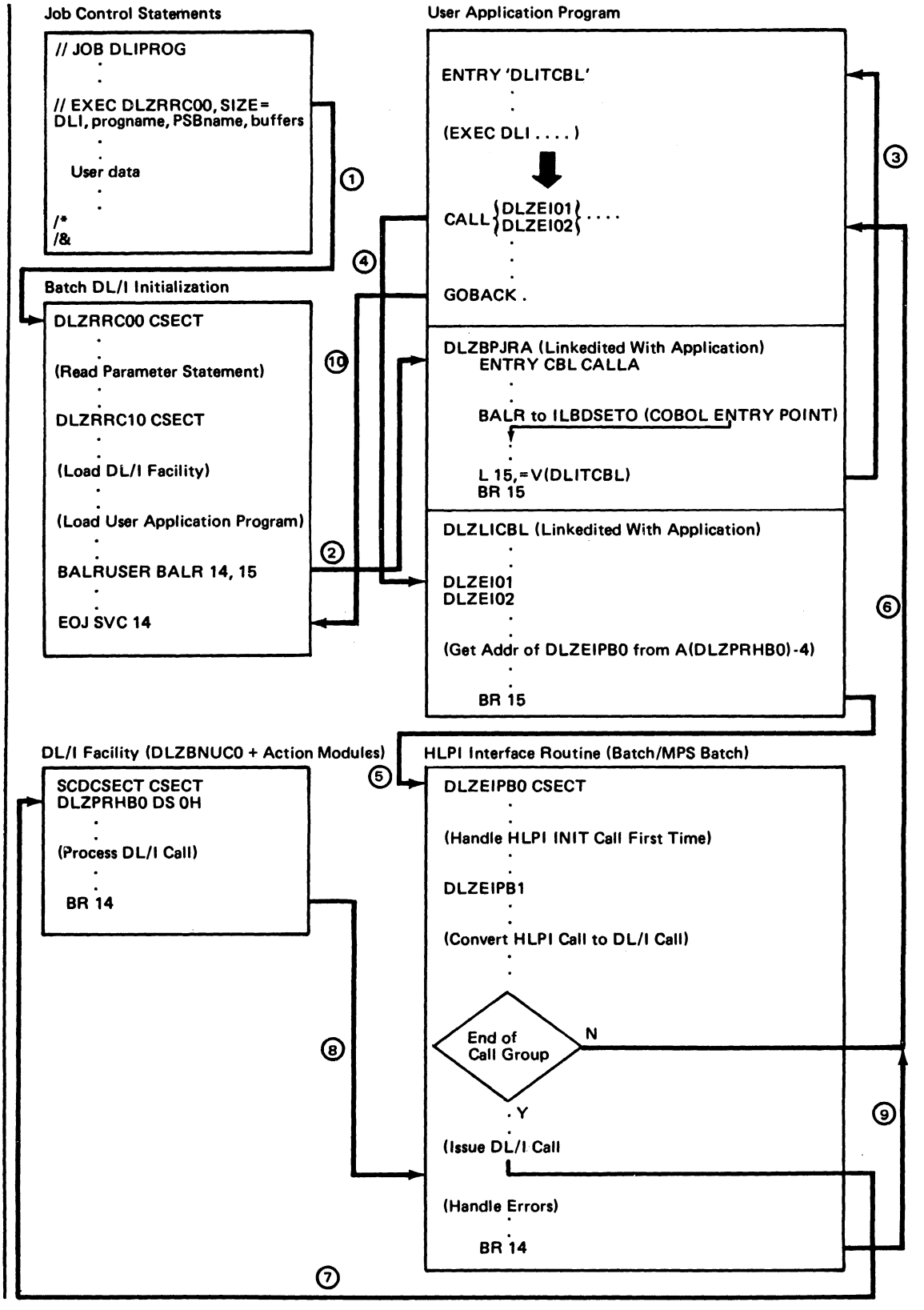


Figure 1-16. COBOL Batch Control Flow

Figure 1-17 through Figure 1-21 provide overviews of paths taken for various service requests from the application program to DL/I. The purpose of the paths is not to describe the sequence of events but to show the relationships between modules during the requests. The arrows depict the direction of the transfer of control and the return path.

GENERAL CALL PATH

Figure 1-17 on page 1-28 illustrates the general outline of a service request to DL/I and the return to the application program. The actual action module used depends upon the type of function requested.

DELETE/REPLACE PATH

Figure 1-18 on page 1-29 illustrates the path taken after a DLET or REPL request is processed by the DL/I call analyzer.

INSERT PATH

Figure 1-19 on page 1-30 illustrates the path taken after an ISRT request is formatted by the program request handler.

LOAD PATH

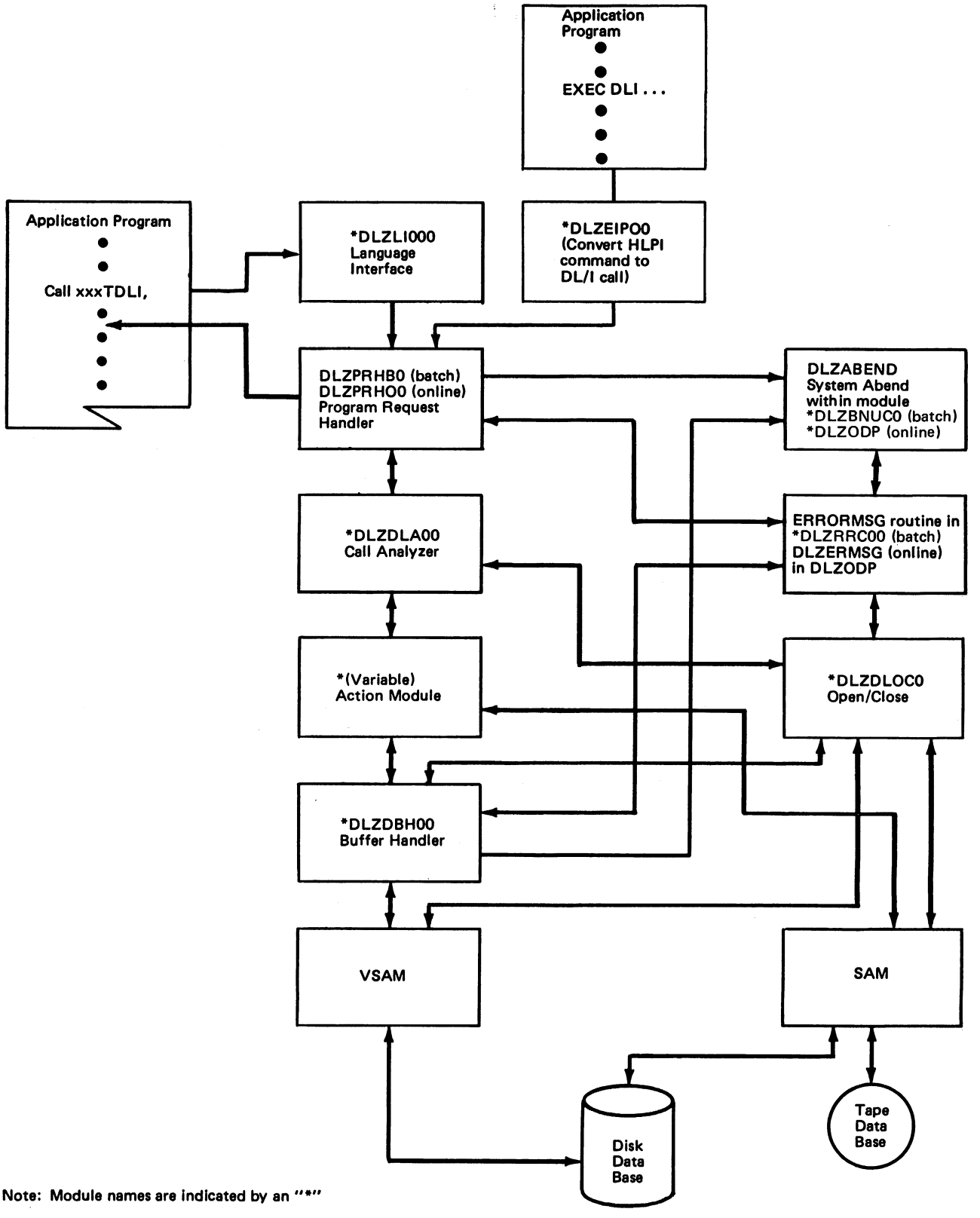
Figure 1-20 on page 1-31 illustrates the path taken after an ISRT (load type) request is formatted by the program request handler.

RETRIEVE PATH

Figure 1-21 on page 1-32 illustrates the path taken after a GET type request is processed by the DL/I call analyzer.

SEGMENT SEARCH ARGUMENT

Figure 1-22 on page 1-32 shows the format of an segment search argument (SSA).



Note: Module names are indicated by an "*"

Figure 1-17. General Service Request Path

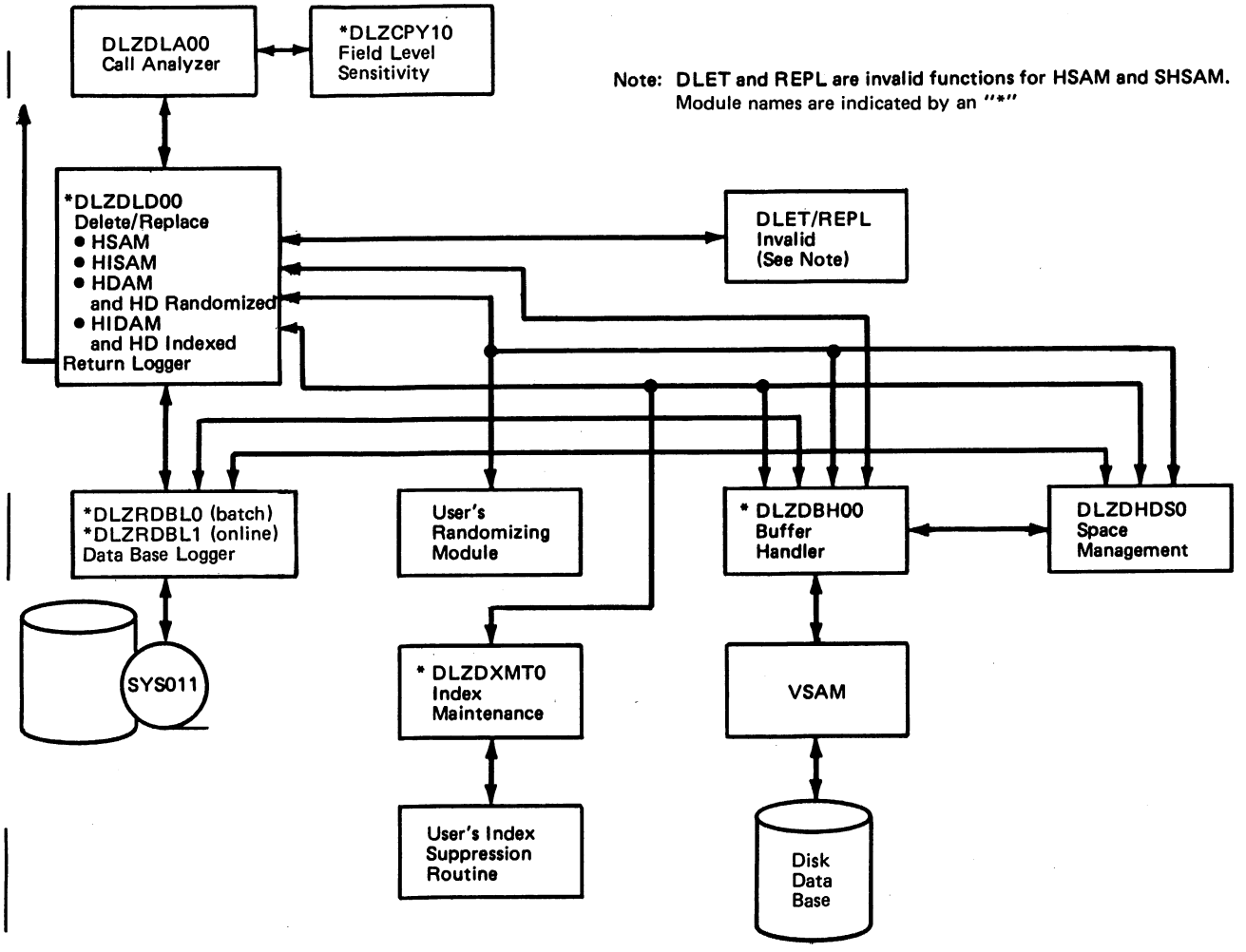
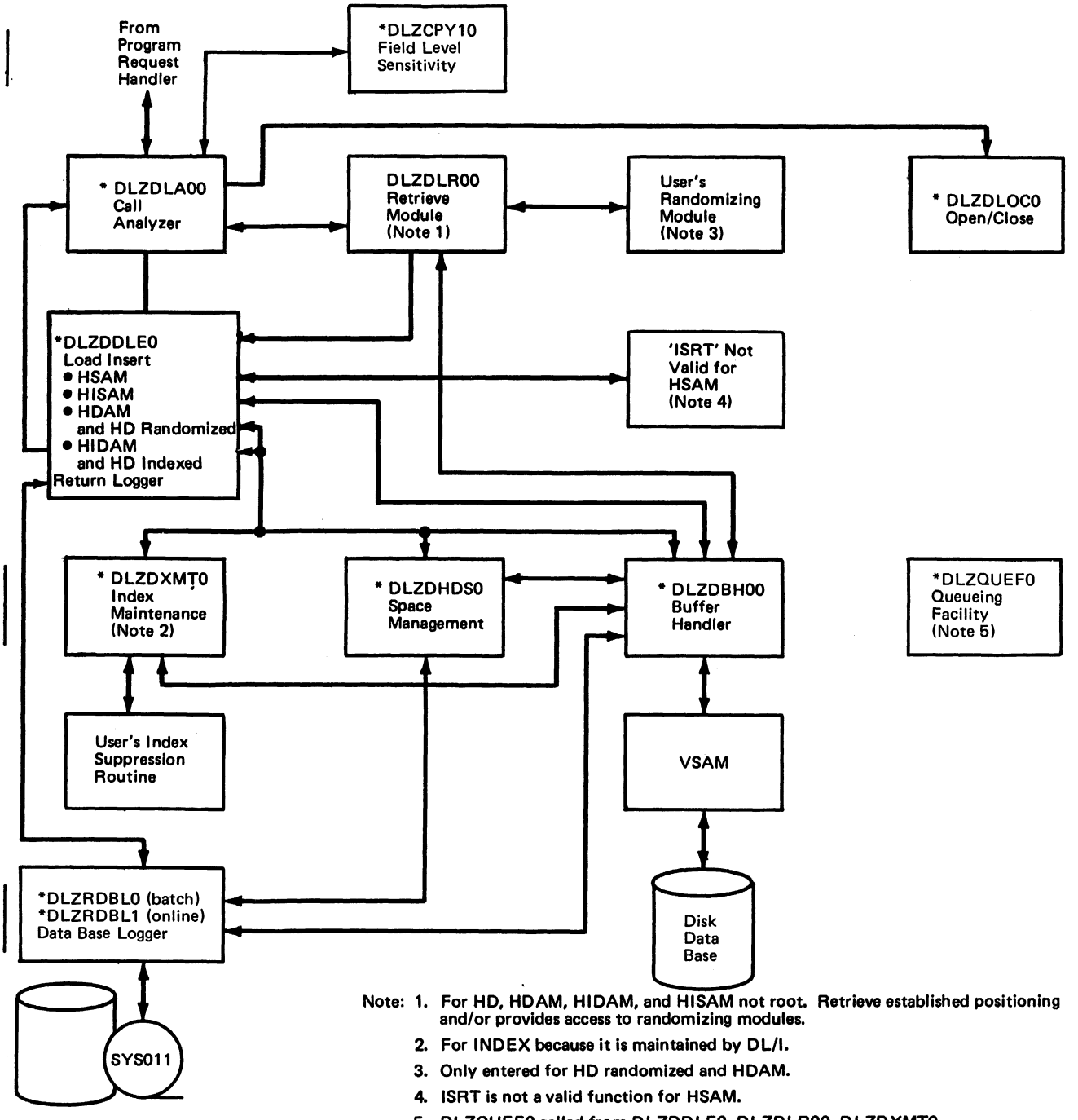
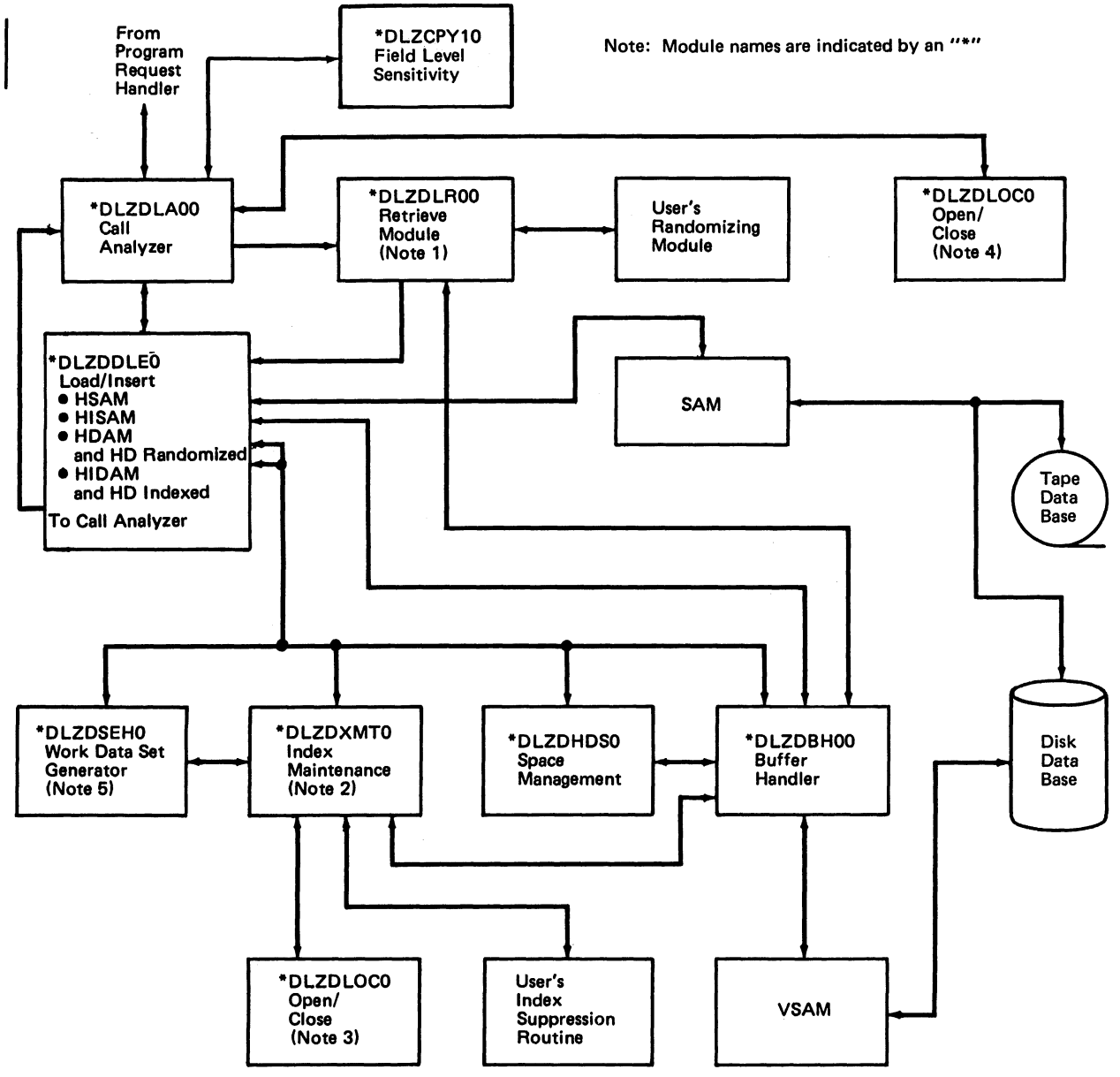


Figure 1-18. Delete/Replace Path



Note:
1. For HD, HDAM, HIDAM, and HISAM not root. Retrieve established positioning and/or provides access to randomizing modules.
2. For INDEX because it is maintained by DL/I.
3. Only entered for HD randomized and HDAM.
4. ISRT is not a valid function for HSAM.
5. DLZQUEF0 called from DLZDDLE0, DLZDLR00, DLZDXMT0.
6. Module names are indicated by an "*"

Figure 1-19. Insert Path



Note: 1. Only for HD randomized or HDAM root. Retrieve is entered for positioning and access to randomizing routine.
2. Only for primary or secondary index which is maintained by DL/I.
3. INDEX is opened at this time.
4. All files except INDEX.
5. Creates work file for secondary index and/or logical relationships.

Figure 1-20. Load Path

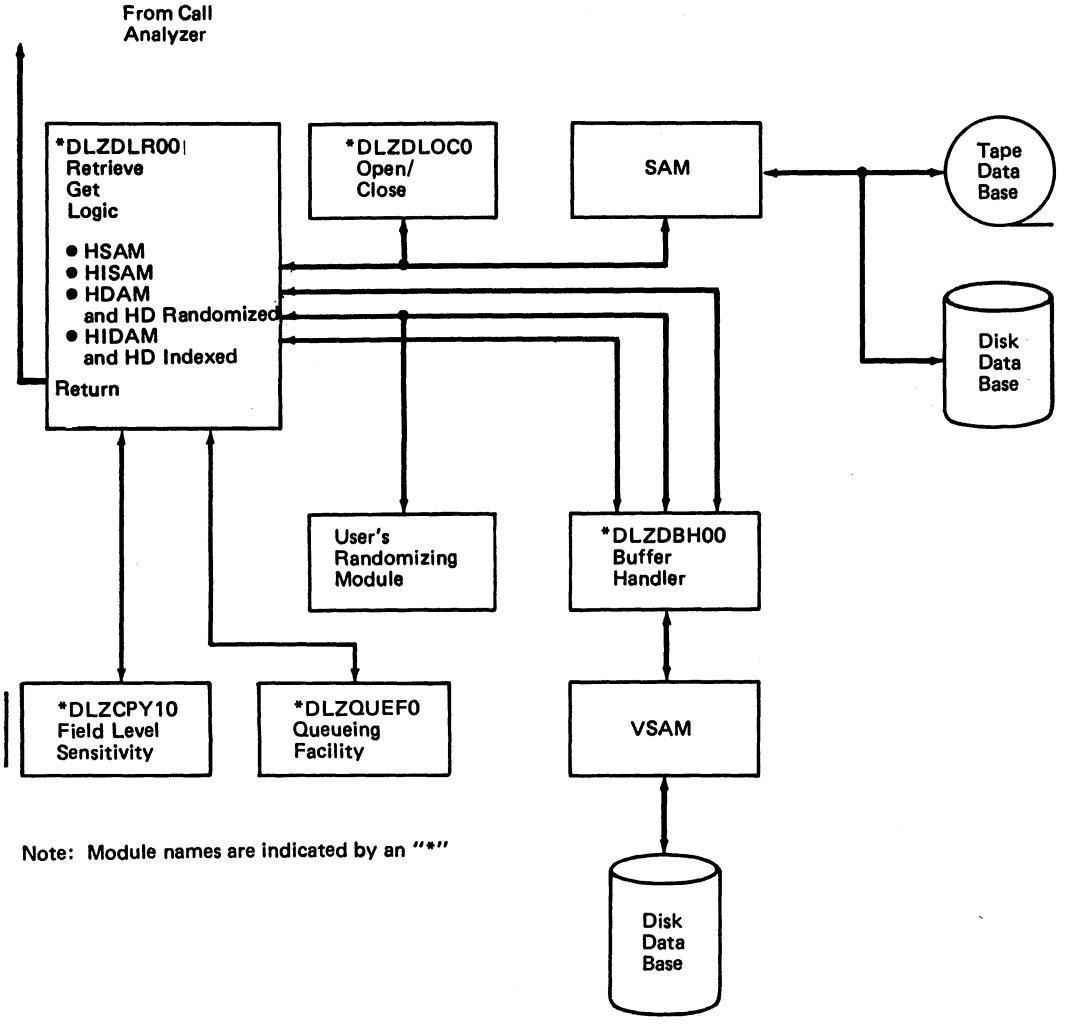


Figure 1-21. Retrieve Path

	(optional)			(optional)												
				BOOLEAN STATEMENT												
elements	Segment Name	Command Codes		Begin Qualif.	Qualification Statement No. 1			Boolean Operator	Qualification Statement No. 2			Boolean Operator	Qualification Statement No. n			End Qualif.
contents	Name of Segment type	*	Code Char-acters	'('	Field Name	R O	Compar. Value	'&' ',' or '+' 'I'	Field Name	R O	Compar. Value	'&' ',' or '+' 'I'	Field Name	R O	Compar. Value	')'
no. of bytes	8	1	Var.	1	8	2	1 to 255	1	8	2	1 to 255	1	8	2	1 to 255	1

Figure 1-22. Segment Search Argument (SSA) Format

DL/I DSECT ASSEMBLY

The job stream shown in Figure 1-23 assembles the DSECTs used by DL/I into one listing. This job stream makes available the latest change level of the DSECTs. Since no usable code is generated, errors may occur at the end of this assembly. It does not mean there are errors in the DSECTs. If a DSECT is missing in the assembly, either it has been deleted, is missing from the system, or has been obsoleted by a new release.

```
         1         2         3       7
12345678901234567890123456789       0123
// JOB DSECT ASSEMBLY
// OPTION NODECK
// EXEC ASSEMBLY
       DLZIDLI     PSTBASE=0          X
                SCDBASE=0,            X
                PDRBASE=0,            X
                DDRBASE=0,            X
                PSBBASE=0,            X
                DPCBASE=0,            X
                DSGBASE=0,            X
                JCBBASE=0,            X
                LEVBASE=0,            X
                SDBBASE-0,            X
                FDBBASE=0,            X
                DMBBASE=0,            X
                DBLBASE=0,            X
                FUNBASE=0,            X
                RPLBASE=0,            X
                EXLBASE=0,            X
                FSBBASE=0,            X
                FRTBASE=0,            X
                FERBASE=0,            X
                PDCBASE=0,            X
                CTGBASE=0,            X
                ACTBASE=0,            X
                AMDBASE=0,            X
                FLDBASE=0
```

```
         1         2         3
12345678901234567890123456789
   DLZIDBD
   DLZBFFR
   DLZBFPL
   DLZRIB
   DLZRPCB
   DLZRPST
   DLZRPDIR
   DLZPSIL
   DLZXMTWA
   DLZSBIF
   DLZURGUF
   DLZUSTAT
   DLZCKPT
   DLZUCREC
   DLZUCHDR
   DLZURHDR
   DLZUDHDR
   DLZUCOLD
   DLZUCUMC
   COPY DLZSYSDS (ONLINE ONLY)
   DLZTWAB  (MPS)
   DLZXTAB  (MPS)
   DLZMPCPT (MPS)
   DLZQFDST
   DLZDTF
   DLZUIB
   DLZPATH
   DLZSSAP
   DLZSSAX
   DLZSSA
   DLZEIPL
   DLZDIB
   DLZSDIB
   END
/*
/&
```

Figure 1-23. DL/I DSECT Assembly

DATA BASE ORGANIZATION AND ACCESS METHODS

There are two data base organizations in DL/I: direct (referred to as hierarchical direct) and sequential (referred to as hierarchical sequential).

DIRECT ORGANIZATION

The following three access methods are part of the direct organization:

- HD: hierarchical direct -- primary access randomized or primary access indexed
- HDAM: hierarchical direct access method (primary access randomized).
- HIDAM: hierarchical indexed direct access method (primary access indexed).

Some of the characteristics of the hierarchical direct organization are:

- Segments of a data base record are related by direct addresses.
- Direct addresses are controlled and maintained by DL/I.
- All segments are stored in a VSAM ESDS.

The HD, HDAM, and HIDAM data base record structures are illustrated by Figure 1-24 through Figure 1-27 .

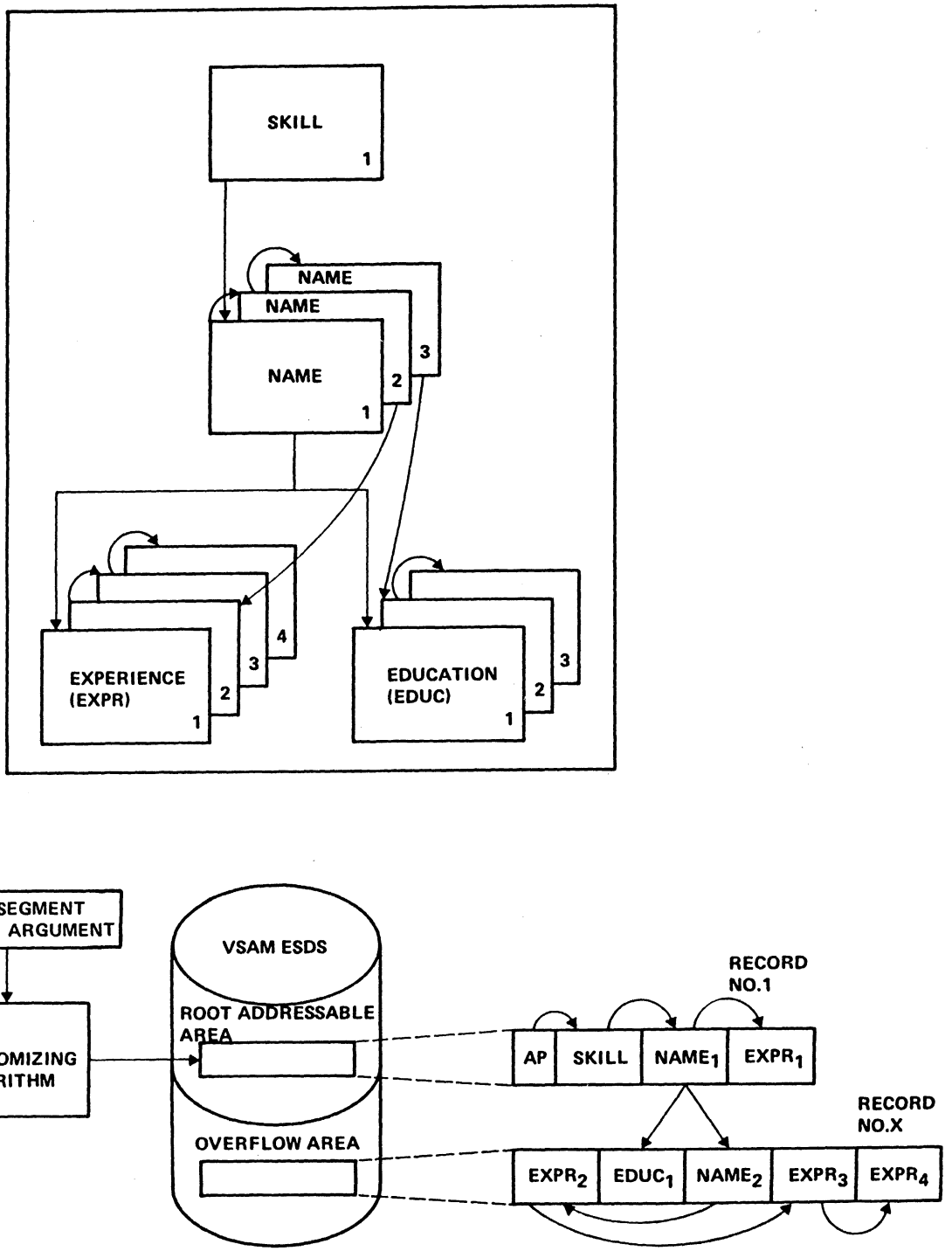


Figure 1-24. HD Randomized and HDAM Physical Storage of Logical Data Structure

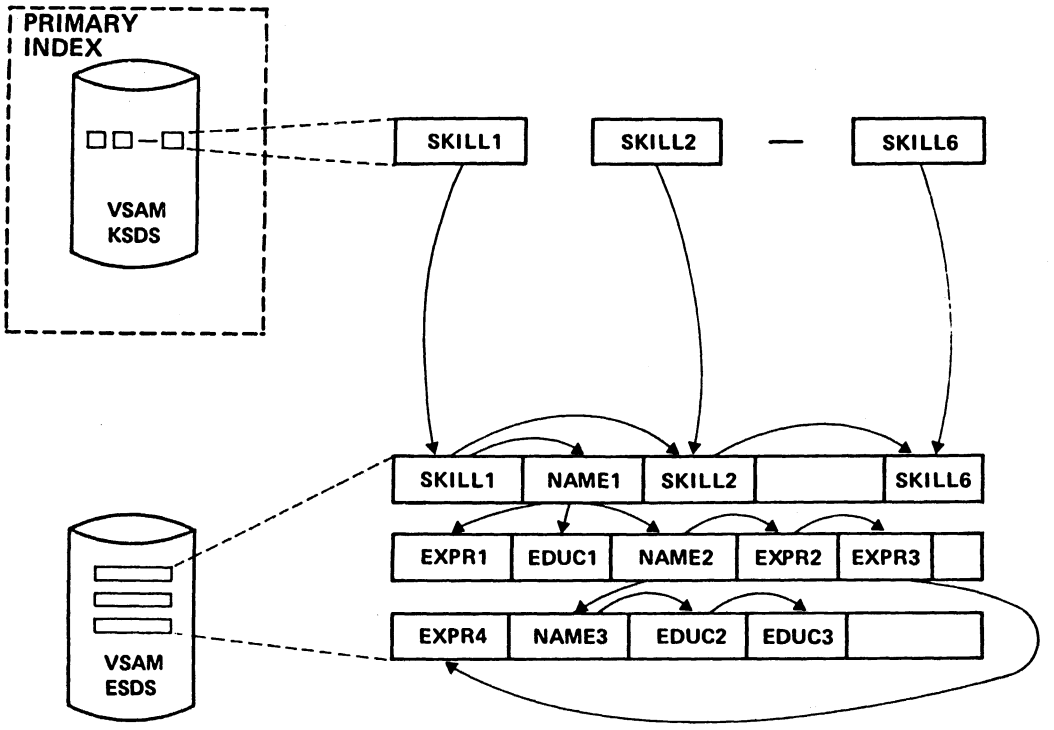


Figure 1-25. HD Indexed and HIDAM Physical Storage of Logical Data Structure

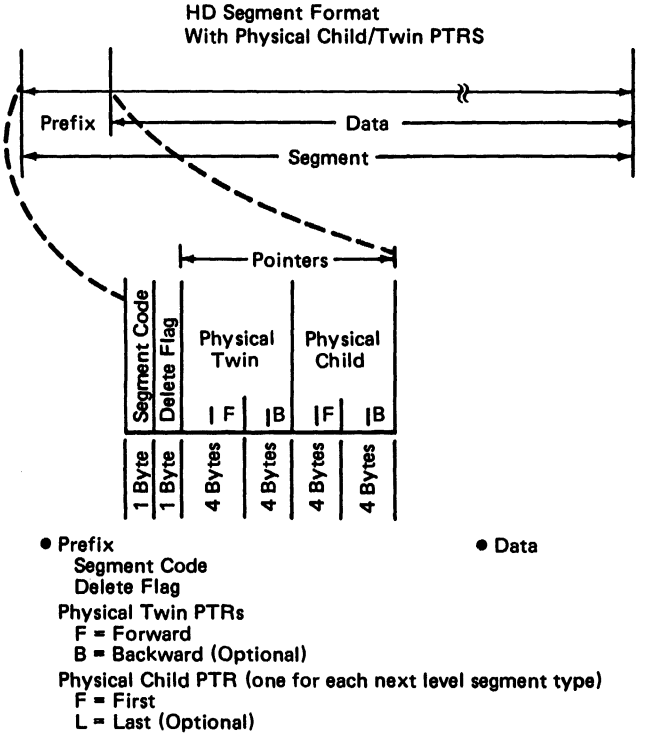


Figure 1-26. Hierarchical Direct Segment Format with Physical Child/Physical Twin Pointers

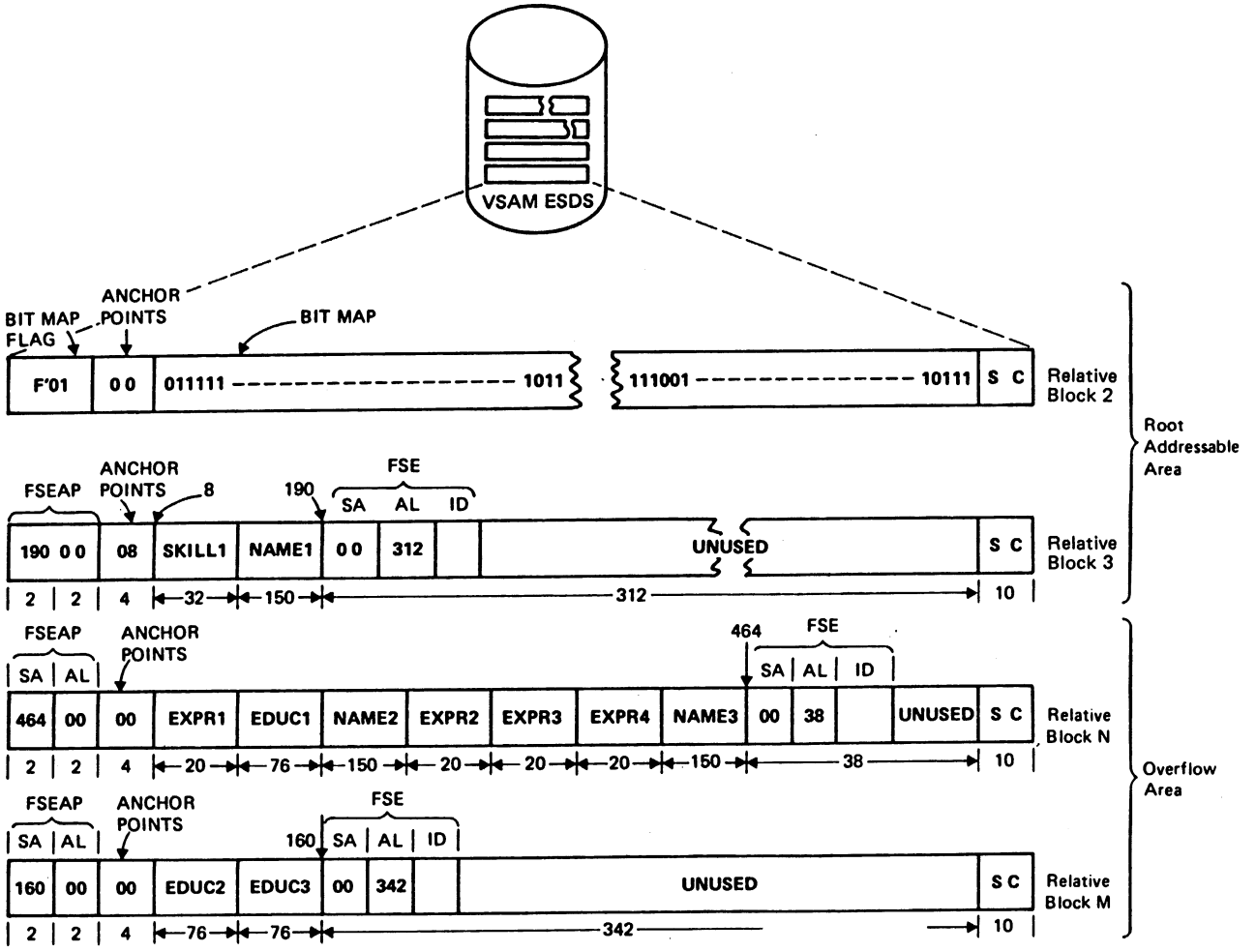


Figure 1-27. One HD Randomized (or HDAM) Data Base Record in Storage

HIERARCHICAL DIRECT DATA FORMATS

Direct address pointers within the prefix of a segment, the anchor point(s) of a block (control interval), and pointers in the prefix of segments of an index data base relating to segments in an HD indexed or HIDAM data base are constructed by the following formula:

pointer value & (block size) X (block number) & (offset within block)

The control fields used in managing entry sequenced data sets for HDAM (see Figure 1-27) and HIDAM data bases are:

- Free space anchor point
- Free space element
- Anchor point area
- Bit map control interval

Anchor Point Area

For an HD randomized or an HDAM data base, the user specifies the number of four-byte direct address root anchor points

desired in each control interval. For each anchor point specified, four bytes of space are reserved in the anchor point area of each control interval.

For an HD indexed or an HIDAM data base, the anchor point area of each control interval contains four bytes of space for one anchor point when only forward physical twin pointers are specified for the root segment type. When forward and backward physical twin pointers are specified, the root anchor point area in each control interval contains binary zeros.

You should always specify PTR=TWINBWD for the root segment of an HD indexed or an HIDAM data base. The PTF and PTB pointers that are then provided allow sequential processing across data base records without intervening references to the index data base. See DL/I DOS/VS Data Base Administration for more information on HD pointer options.in

Bit Map

The second control interval of the entry sequenced data set in a HDAM or HIDAM data base is used for a bit map. In the bit map, each bit is used to describe whether or not space is available in a particular control interval. The first bit corresponds to the interval the bit map itself is in, and each consecutive bit corresponds to the next consecutive interval in the data set. When the bit value is one, it indicates that the associated block has sufficient space remaining to store an occurrence of the longest segment type defined in this data base. When the bit value is zero, sufficient space is not available for an occurrence of the longest segment type defined in the data base. The first bit map in a data set contains n bits that describe space availability in the next n consecutive control intervals in the data set. After the first bit map, another bit map is stored at every nth control interval to describe space availability in the remaining control intervals in the data set.

Bit Map Block

A bit map control interval starts with a two-byte free space anchor point field. The field always contains zeros in a bit map control interval since no space is available in those bit map control intervals for segments. The next two bytes contain a bit map flag. A low-order one in the second two bytes of a control interval identifies that control interval as containing a bit map. The same two bytes in all other control intervals in a data set contain zeros. The next n bytes of a bit map control interval are for root anchor points. Following the anchor point area, the remainder of the control interval is a bit map.

Free Space Anchor Point

Each control interval in an entry sequenced data set starts with a four-byte free space anchor point field (FSEAP). The field contains, in the first two bytes, the offset in bytes to the first free space element in that control interval. If the control interval is a bit map block, the second two bytes contain a flag. Otherwise, the second two bytes of the field contain zeros.

Free Space Element

A free space element (FSE) identifies each area of free space in a control interval that is eight bytes or more in length. To identify each area of free space, a FSE occupies the first eight bytes of each area of free space. The fields in each FSE are:

- Free space chain pointer field (SA) -- This 2-byte field contains, in bytes, the offset to the next free space element in the control interval, from the beginning of the control interval. If zero, this is the last FSE.

- Available length (AL) -- This 2-byte field contains, in bytes, the length of the vacant space that this free space element identifies. The value in the available length field includes the length of the free space element itself.

- Task-ID field (ID) -- This 4-byte field contains the task-ID of the program that freed the space that is identified by this free space element. The task-ID enables a given program to free and reuse the same space during a given scheduling without contending for that space with other programs. This field is not used in a batch system.

  The task-ID consists of a one-byte calendar date followed by a three-byte cumulative task counter for the day.

PL/I Special Considerations

Since the application program is treated as a called program, the initialization of the PL/I program is done through the entry point PLICALLB. This entry point requires a special parameter list in addition to the PSB address normally passed by DL/I. The entries in this list include:

- A pointer to the normal parameter list that contains the address of the PSB.

- A pointer to a field that contains the amount of storage available for the dynamic storage area (DSA) which is calculated by DLZRRC00 (DLZMPI00 in MPS).

- A pointer to a field that contains the address where the DSA can be started.

To locate this parameter list in the dump, look for the names of utilities DLZURGL0, DLZURGP0, DLZURGS0, and DLZURPR0. About 40 fullwords later are two fullwords of X'FF's; following the X'FF's is the PL/I information.

In DLZMPI00, the list is located immediately in front of the AB save area, which is identified by the constant 'AB REASON CODE'.

For PL/I programs only, all pointers to PCBs actually point to dope vector tables, which point to PCBs.

SEQUENTIAL ORGANIZATION

The following four access methods are part of the sequential organization:

- Simple hierarchical sequential access method (simple HSAM).

- Hierarchical sequential access method (HSAM).

- Simple hierarchical indexed sequential access method (simple HISAM).

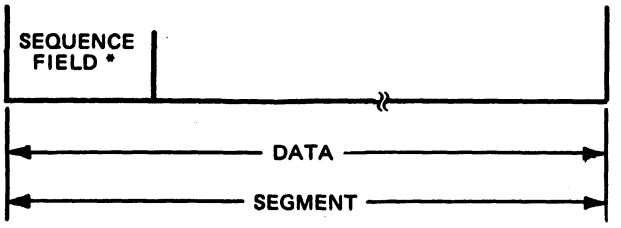
- Hierarchical indexed sequential access method (HISAM).

Some characteristics of the hierarchical sequential organization are:

- Segments of a data base record are related by physical adjacency.
- All segments in a logical record belong to the same data base record.
- Logical records are related by relative byte pointers.

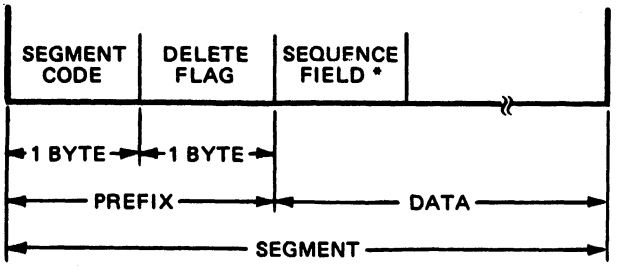
- Root only data base defined in DBD generation by ACCESS=SHISAM or ACCESS=SHSAM is a standard VSAM or SAM data set.

The segment format of a simple HSAM or simple HISAM data base is illustrated in Figure 1-28. HSAM data contains a segment code and prefix with each segment. Simple HSAM contains only root segments and no segment code or prefix. Figure 1-29 illustrates the segment format of a HSAM or HISAM data base. Figure 1-30 on page 1-41 illustrates the HISAM physical storage of a logical data structure. The HSAM physical storage of a logical data structure is illustrated in Figure 1-31 on page 1-42 .



* The sequence field may be in any position in the root segment.

Figure 1-28. Segment Format of a Simple HSAM or Simple HISAM Data Base



* The sequence field may be in any position in the segment.

Figure 1-29. Segment Format of a HSAM or HISAM Data Base

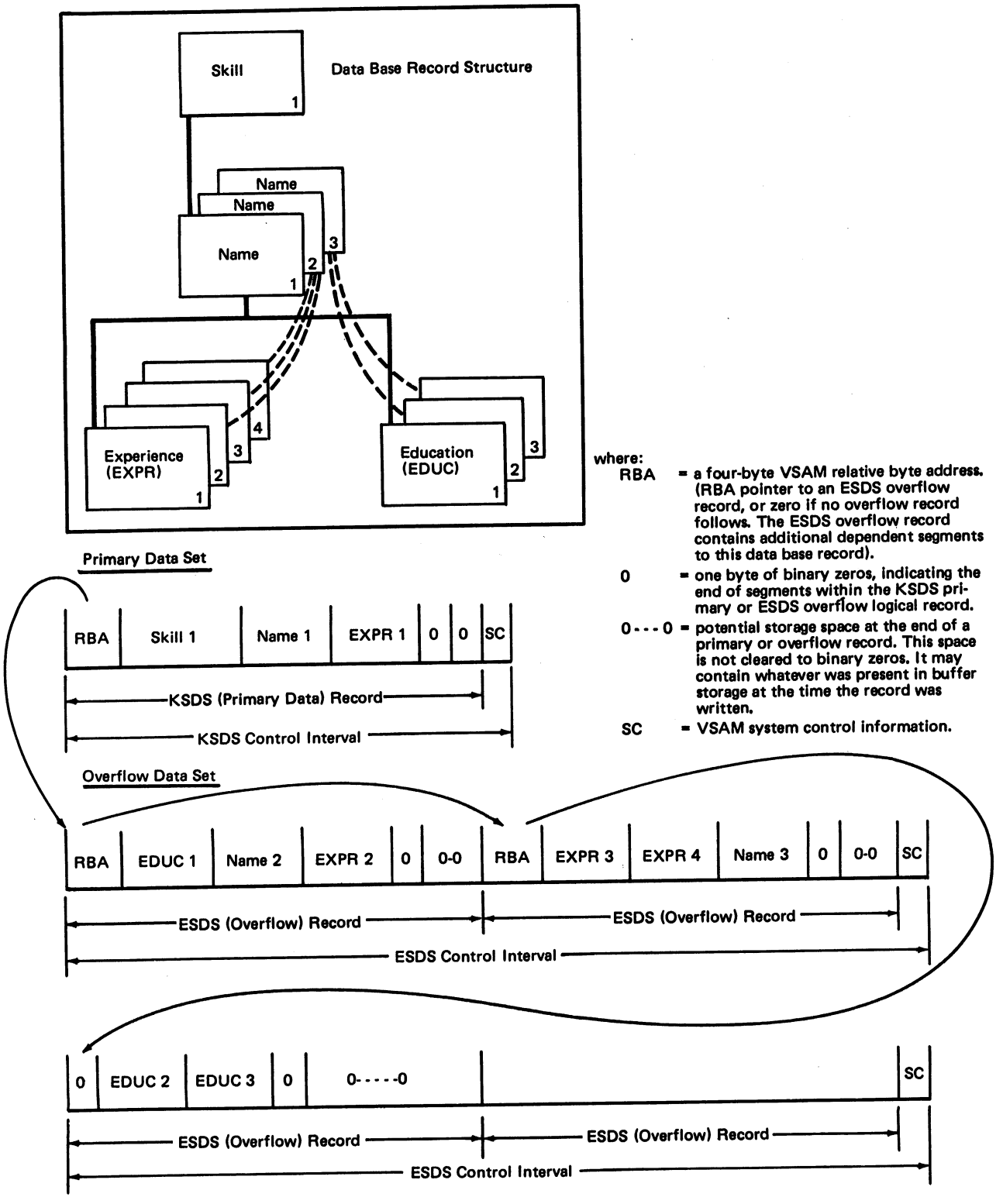


Figure 1-30. HISAM Physical Storage of a Logical Data Structure

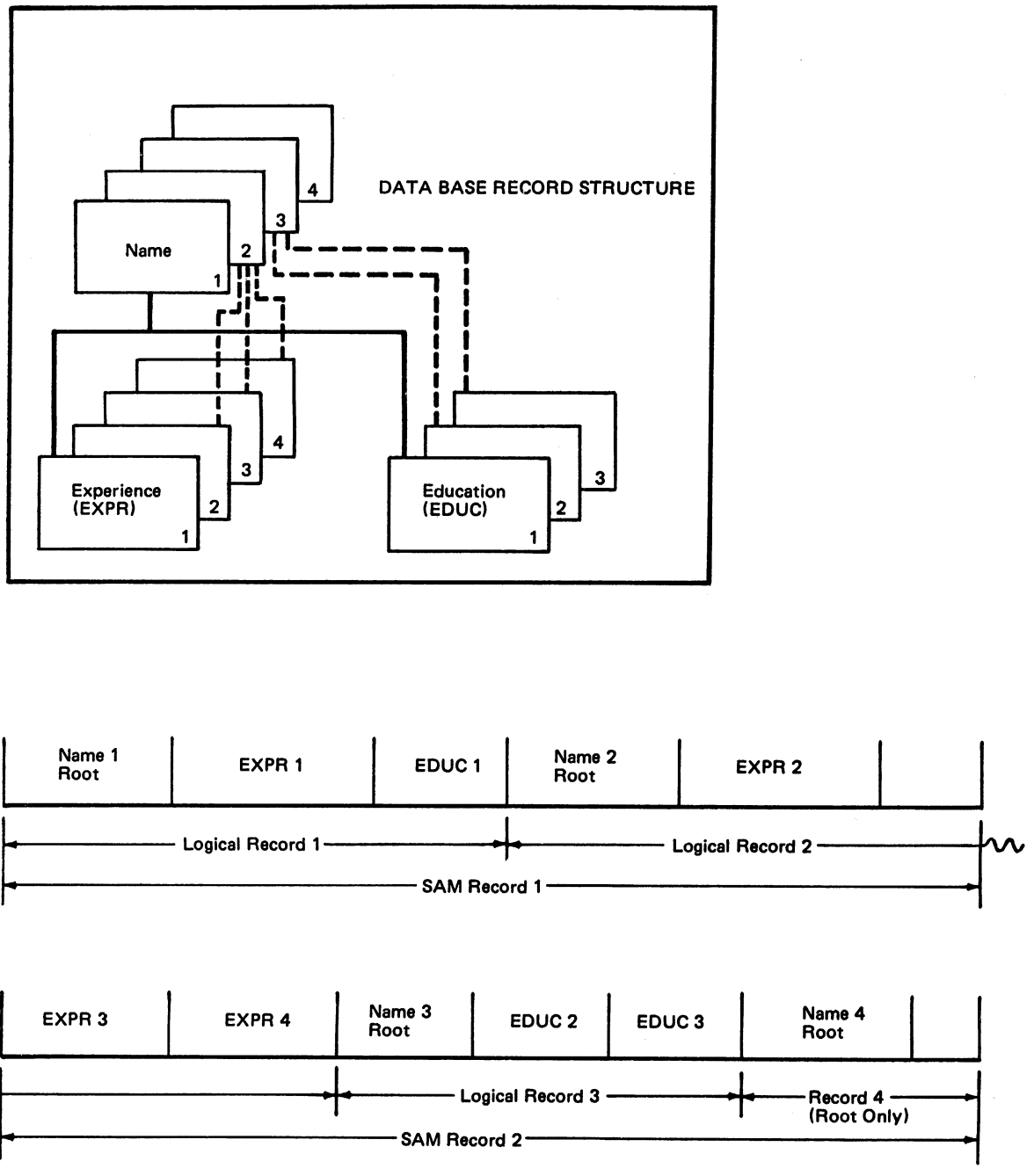
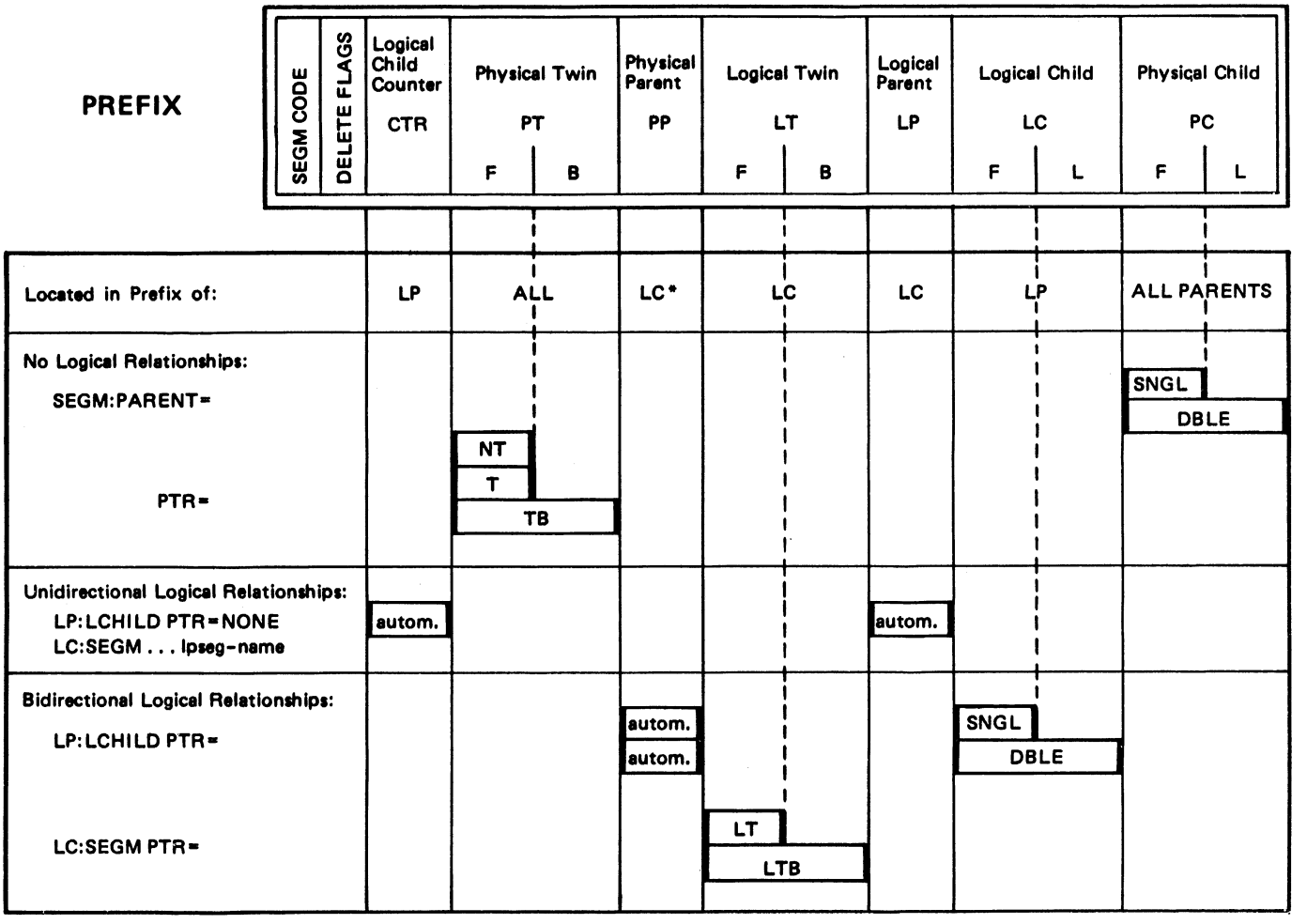


Figure 1-31. HSAM Physical Storage of a Logical Data Structure

POINTERS FOR HD DATA BASES

Figure 1-32 summarizes the physical and logical pointers for HD organization data bases.



* The physical parent pointer is automatically inserted in all segments in a physical path from a logical child, a logical parent, and/or an indexed segment up to, but not including, its root segment. This is to allow for inversion of the data structure.

Figure 1-32. Segment Prefix for HD Organization and Logical Relationships

CHAPTER 2. DIAGNOSTIC AIDS

This chapter is a review of and provides diagnostic aids for problems in:

- Control flow
- Initialization
- DL/I commands and calls
- Data locations
- Scheduling errors
- Common user errors
- Online wait/suspend state

Included are discussions of:

- Tracing control flow through use of register save areas
- JCB trace for tracing DL/ I calls
- Online trace for correcting online scheduling errors
- Retrieve subroutine trace for analysis of retrieve problems

CONTROL FLOW

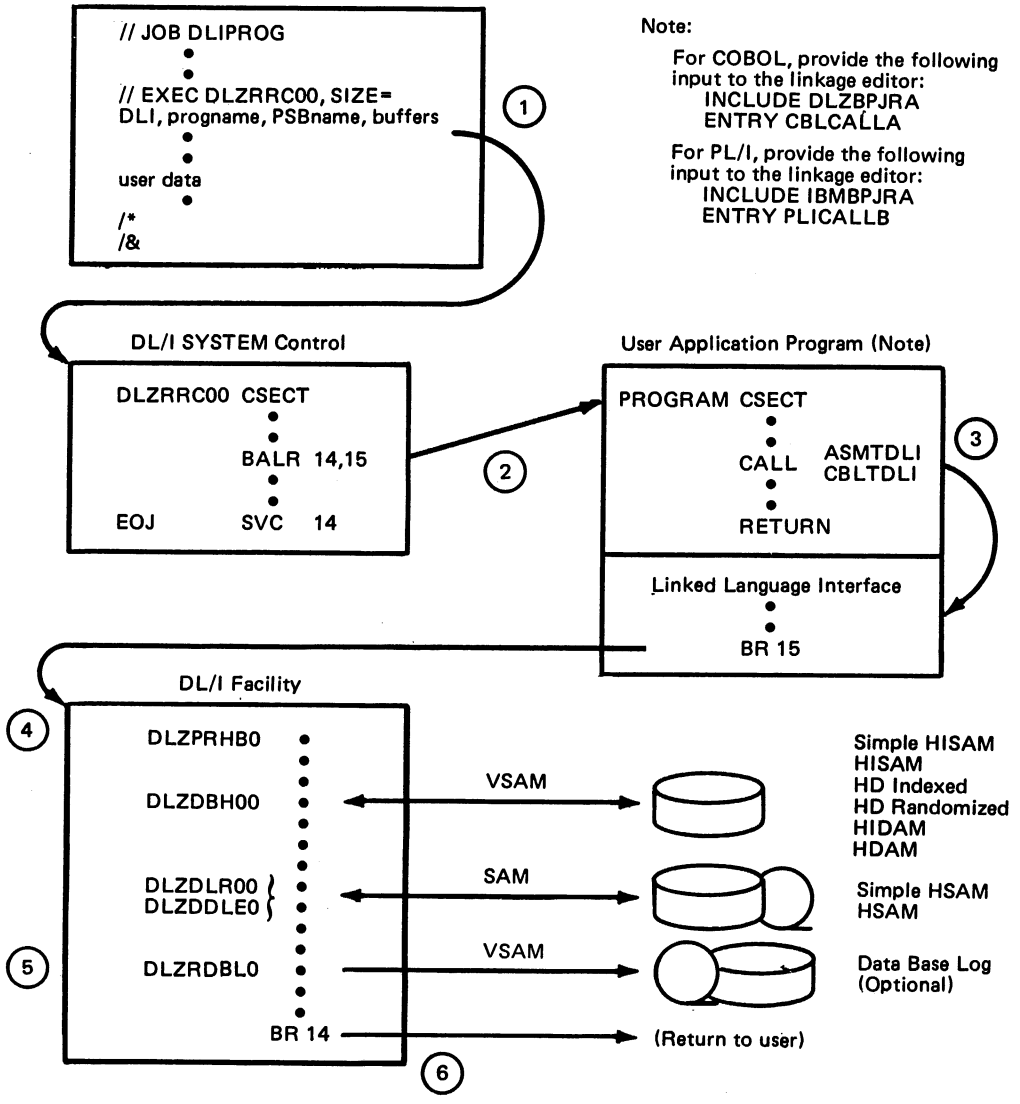


Figure 2-1. DL/I Batch Control Flow

Figure 2-1 is a general control flow for DL/I with six checkpoints. In this chapter, the following information is discussed using this figure as a reference:

- Aids at initialization
- Aids for common user errors
- Aids during a DL/I call
- Aids to isolate the problem in DL/I.

Appendix B shows the "DL/I MPS XECB-Waitlist Dependence and Main Routine Flow".

BATCH INITIALIZATION AIDS

During batch initialization, when a problem occurs between checkpoint 1 and checkpoint 2 in Figure 2-1, consider the following programming aids:

- Load and initialize nucleus and control blocks. As each control block is loaded or initialized, beginning with the

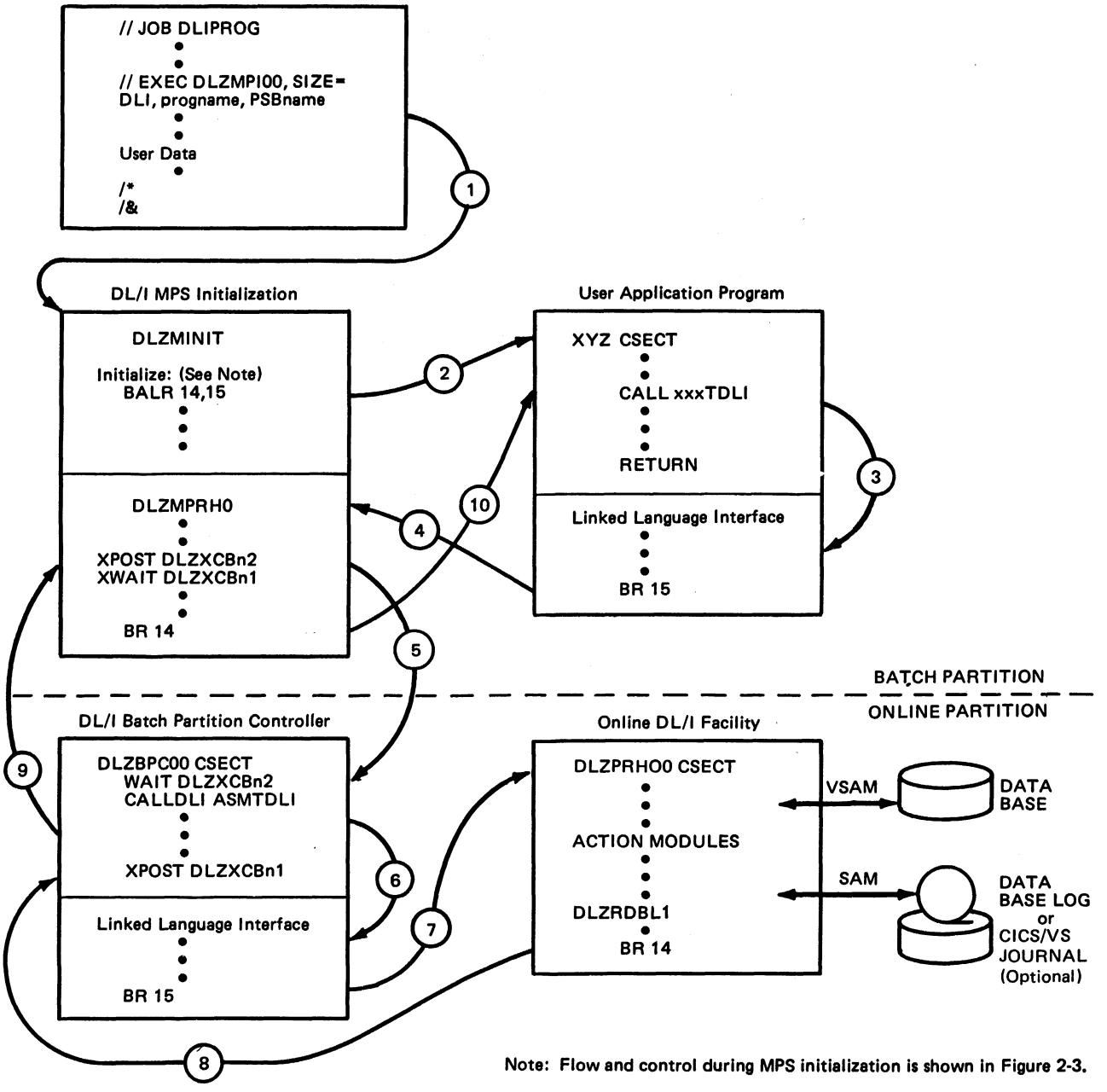


Figure 2-2. DL/I MPS Batch Control Flow

nucleus, a value is placed in field SCDDLIUP of the SCD. This value is the next available location after the control block. It can also be an indicator as to how much initialization has been accomplished.

- By using the entry and exit points in the listings, a point can be found to cause a dump or place a PDUMP macro to help trouble shoot this module.
- The DL/I DOS/VS Logic Manual, Volume 1, gives descriptions of the control flow.

MPS BATCH INITIALIZATION AIDS

When a problem occurs during MPS batch initialization, consider using the following programming aids (refer to Figure 2-2 on page 2-3 and Figure 2-3 on page 2-5):

- DLZMINIT reads the input parameter statement and checks it for validity. It then loads the user's program, after which it determines what to pass to online to use as a partition identifier in messages by checking the PIK in the BG COMREG.
- After saving the program name and PSB name for use by online, XECB DLZXCB02 is checked using the XECBTAB macro. A value, n, for this partition is obtained from the MPC partition table and is put into each XECBTAB macro having a name with an 'n' value in it. An XECB, DLZXCBn1, is defined in the batch partition for communicating with the online partition. The start partition XECB in online, DLZXCB02, is posted. This lets the online partition know that there is an MPS batch job ready to run. DLZXCBn3 is checked using the XECBTAB macro. If initialization online was successful, the XECB has been defined.
- If the MPS Restart facility is used, message DLZ124I is issued at initialization if the CICS/VS journal is not active or if the Temporary Storage Control facility is not available.
- When the online partition completes its initialization, the batch routine finishes other initialization routines, and goes to the user program.

ONLINE INITIALIZATION AIDS

During online initialization, when a problem occurs in Figure 1-7 on page 1-9 , consider using the following programming aids:

- The online DL/I system is initialized as part of CICS/VS initialization. CICS/VS initialization loads module DFHSIDL. By using Figure 1-7 on page 1-9, Figure 4-12 on page 4-18 , and Figure 4-13 on page 4-20 , you should be able to determine which section of DL/I was loaded last.
- Informational messages are issued during online initialization. Message DLZ118I provides information on the version and level of DL/I currently running; message DLZ113I indicates the status of program isolation (active or inactive); and message DLZ119I indicates the status of CICS/VS and DL/I logging.
- Using the entry and exit points in the listings, a point can be found to cause a dump or place a PDUMP macro to aid in trouble analysis.
- The DL/I DOS/VS Logic Manual, Volume 1, gives a detailed description of the control flow.

BATCH AND MPS BATCH USER ERROR AIDS

Errors which occur between checkpoint 2 and checkpoint 3 in Figure 2-1 on page 2-2 generally are user errors. The following is a summary, by groups, of some commonly misinterpreted points. Symptoms of these errors do not necessarily occur in the group.

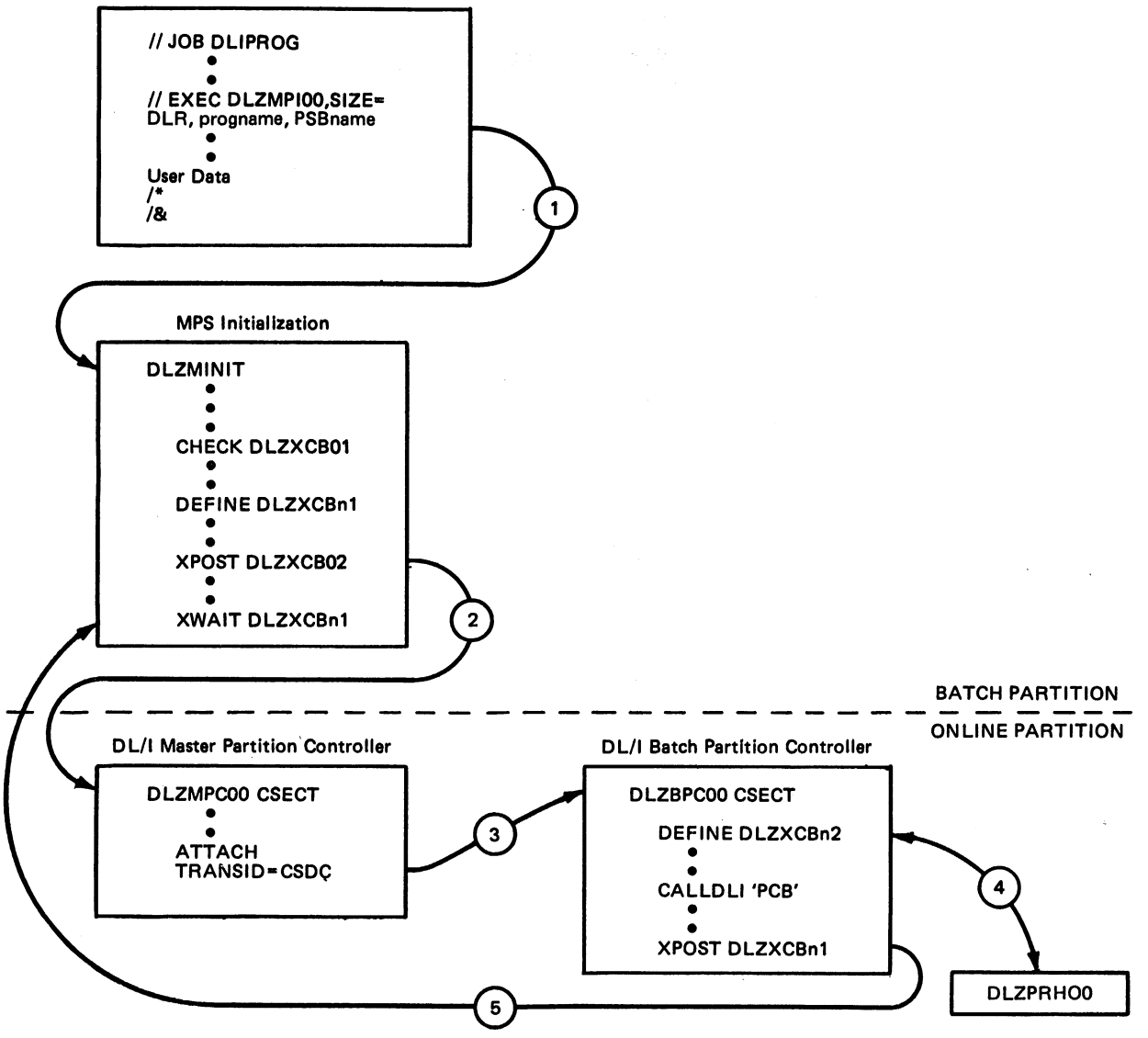


Figure 2-3. DL/I MPS Initialization

Job Control Errors

The following relationships exist for DLBL information between VSAM DELETE (or DEFINE) and DL/I execution:

- For VSAM:

  // DLBL nnnn,'mmmm',,VSAM

  where:

  nnnn is the name specified in FILE (nnnn) parameter of the VSAM DEFINE command.
  mmmm is user convention (during DEFINE CLUSTER, VSAM ignores this entry)

- For DL/I:

  // DLBL aaaa,'bbbb',,VSAM

  where:

aaaa is the name specified in DD1=aaaa of DBD generation -- aaaa must match nnnn specified for VSAM.

bbbb is the name specified in CLUSTER, NAME (bbbb) of VSAM's DELETE or DEFINE.

Note: For MPS batch, DLBL is not required for data bases, it is specified in the online partition.

Failure to specify proper UPSI bit settings result in being unable to recover from abnormal termination. MPS will force STXIT processing regardless of UPSI byte setting.

If the SIZE parameter is not specified in the EXEC statement, a program check occurs.

If the SIZE parameter value is too small to include the application program, the results are unpredictable.

DBD Generation Errors

During DL/I execution, message DLZ025I occurs if the DBD generated LRECL does not match the VSAM DEFINE RECORDSIZE (xxx yyy), where xxx is the average length of records to be written, and yyy is the maximum length of records to be written.

Also, the DBD generation control interval size must be the same as the size defined to VSAM.

During DL/I execution, message DLZ831I occurs if the DBDGEN specified an FBA device for the data base and a CKD device is found or a non-zero return code from GETVIS, SHOWCAT, and GETVCE. Space Management does not have the track and cylinder information it needs to continue. If a KSDS does not have an even record size, message DLZ855I results.

Application Control Block Creation and Maintenance Utility Errors

If a change is made to a DBD, the corresponding DMB must be rebuilt. This is done automatically if DMB=YES is included in the BUILD statement(s) for the utility.

If DMB=YES was not included in the BUILD statement(s), the first step is to delete the old DMB from the core image library.

If this is not done, the existing DMB is not replaced with the new information and can result in a variety of symptoms during DL/I execution such as, record mismatch, file not open, file does not exist, etc. The following is an example of the job control statements for deleting an old DMB from the core image library:

```
// JOB JOBNAME
// EXEC MAINT
    DELETC xxx@@@@D
/*
/&
```

where:

xxx is the user's DBD core image library name.
@@@@ is used to extend the DBD name to a length of seven bytes, if necessary.

The blocks must then be rebuilt specifying every PSB that references the changed DBD.

Application Program Errors

The following errors occur primarily in Assembler language programs:

- If the high-order bit is not cleared in the PCB address passed to the user's program on implicit calls, this field will be the delimiter of the parameters passed. The remaining parameters are ignored and results in message DLZ476I.
- The parameter count is something other than an unsigned fullword binary number, possibly a halfword. This results in message DLZ260I.
- Program check if language interface module (DLZLI000) is not autolinked to the user's application program when link edited to the core image library or it was not included via an INCLUDE statement at that time.
- Various symptoms if registers are not saved on entry or a save area is not provided and pointed to by register 13 when a CALL is issued.
- Register 1 not pointing to the parameter address list when issuing a DL/I call.
- A PCB request is issued for a PSB other than the one specified in the parameter statement (MPS).

The following errors occur when the MPS Restart facility is active:

- An invalid PCB parameter is specified in a DL/I call. This results in message DLZ132I.
- A SCHEDULE or TERMINATE command is issued. The TERMINATE command cannot be issued because it causes a CICS/VS sync point, which destroys synchronization between the data base and application program. Because a SCHEDULE command cannot be issued in MPS batch without first issuing a TERMINATE command, neither command is allowed and, if issued, results in message DLZ133I. (Note that the SCHEDULE and TERMINATE commands correspond to the PCB and TERM call functions.)
- If a VSE CHKPT is not taken before a DL/I CHKPT command, or the VSE CHKPT failed, or a Temporary Storage error occurred during DL/I CHKPT processing, no DL/I checkpoint is taken and the status code, XR, is issued.
- If a buffer write-out error occurs during a DL/I checkpoint, message DLZ131I is issued. This happens when the TSQ (used by MPS Restart to maintain checkpoint IDs) was updated before the DL/I checkpoint was attempted. When this occurs, the TSQ cannot be committed and made available to other tasks.

HD Randomizing Module Errors

The randomizing module cannot be written in PL/I, COBOL, or RPG; it must be written in Assembler language.

Operation Errors

If VSAM files are not closed properly on abnormal terminations, data base open errors may occur. The following should be checked:

- How the UPSI byte was used,
- Whether VSAM VERIFY and data base backout utility program was used,
- Whether VSAM DELETE, DEFINE and data base data set recovery utility programs were used.

Message DLZ082I occurs if the batch XECB cannot be defined (MPS).

Message DLZ083I indicates MPC is not active (MPS).

ONLINE USER ERROR AIDS

Job Control Errors

- The size parameter is not specified in the EXEC statement.
- Failure to include ENTRY DLZNUC statement in the ACT LNKEDT caused a program check during initialization. Register 2 contains characters 'DLZN'.
- No ACT LNKEDT.

ACT Generation Errors

- User tried to define more than 4095 programs.
- User tried to define more than 4095 PSBs.
- No DLZACT TYPE=FINAL macro or END statement.
- User tried to define duplicate program name.
- User tried to define duplicate PSB names in TYPE=PROGRAM macro.
- User tried to define more than 4095 DBD names in TYPE=BUFFER in the HDBFR or the HSBFR macros.
- User tried to define duplicate DBD names.
- User failed to define ACT entry for DLZBPC00 or DFHMIR.

Application Program Errors

Same as batch, plus:

- User did not issue a PCB call before trying to access the data base(s).
- Long running transaction ties up the system because user failed to issue 'TERM' call.
- User made PSB call with invalid PSB name (no matching entry in ACT).

ONLINE REMOTE DATA BASE AIDS

Most DL/I application programs run without modification on a DL/I system with ISC (intersystem communication) support[1]. To migrate from a system where all PSBs are defined locally to a system where some or all of the PSBs (and their associated data bases) are defined on a remote system, do the following:

- Generate a new DL/I nucleus for the local system using DLZACT TYPE=RPSB statements to define PSBs existing on the remote system that are referenced by application programs executing on the local system.
- Generate a new DL/I nucleus for the remote system which includes an ACT entry for DFHMIR if DL/I calls from another system are to be accepted.
- Do not issue PSB scheduling calls from an MPS batch application program.
- If accessing a remote PSB on an MVS system with an 8-character PSB name, RNAME must be used on a DLZACT TYPE=RPSB statement to specify the 8-character name.
- If a pseudo-abend occurs in the remote system, the local system is notified by a CICS/VS abend code. A pseudo-abend message is not written on the local system.

---

[1] The maximum length of SSAs transmitted to a remote system by application programs using Boolean operators is 304 bytes.

When a problem occurs between checkpoint 3 and checkpoint 4 in Figure 2-1 on page 2-2 (batch), or Figure 2-2 on page 2-3 (MPS), consider the following programming aids:

The exit from the user's application program is accomplished via the CALL macro. At the time of the call, the user must have provided the address of a save area in register 13, and a pointer to a list of addresses which point to the parameters being passed to DL/I in register 14. The call is made to the name CBLTDLI, ASMTDLI, RPGDLI, or PLITDLI, depending on the language used by the application programmer. This causes the generation of an external reference at compile time. This external reference is resolved when the module is link edited via autolink or by the statement INCLUDE DLZLI000.

This module, entered at one of its four entry points, establishes which programming language is used in register 0, saves the user's registers, picks up the address of the program request handler from the COMREG+X'10' that was stored there at initialization, and passes control to the program request handler.

The program request handler, which is part of the nucleus, establishes the parameters in the PST field PSTLIPRM before control is passed to the DL/I call analyzer. The parameters are established according to the type of address list passed, either an explicit or implicit list. Two examples follow:

EXAMPLE 1:

```
GPR1 00090000 (Explicit) contains
              pointer to user call
     00090004 (Implicit) parameter list
```

At the location specified in register 1, either of two formats of the call list may be found. It is easy to determine the format of the call list by interrogating the first parameter.

Storage Address	Storage Contents	
090000	00090100	(Count address)
090004	00090104	(Function address)
090008	0009A000	(PCB address)
09000C	00090600	(I/O area address)
090010	80090108	(SSA address)
090100	00000004	(Parameter count)
090104	C7D54040	GNbb

If register 1 contains the count address, X'00090000', the list is considered explicit; if it contains the function address, X'00090004', it is implicit. By checking the contents of the storage location designated by the first parameter in the list, you will find a binary fullword representation of the list length (explicit) or a field containing alphameric characters representing the function (implicit).

The program request handler tests bits 0 and 1 of the high-order byte of this field to determine the type of list. Therefore, values of X'C0' and higher are considered implicit notations and values of X'BF' and lower are explicit.

In implicit notation, a 1 in bit 0 of the high-order byte in the call list, stops the list scan and established the parameter count.

If the high-order byte of the last PCB address pointer is not cleared before storing it in the call list and implicit notation is used, the scan will be stopped prematurely and an error occurs.

EXAMPLE 2: When control is passed from the program request handler to the DL/I call analyzer, the values shown in Figure 2-4 are established.

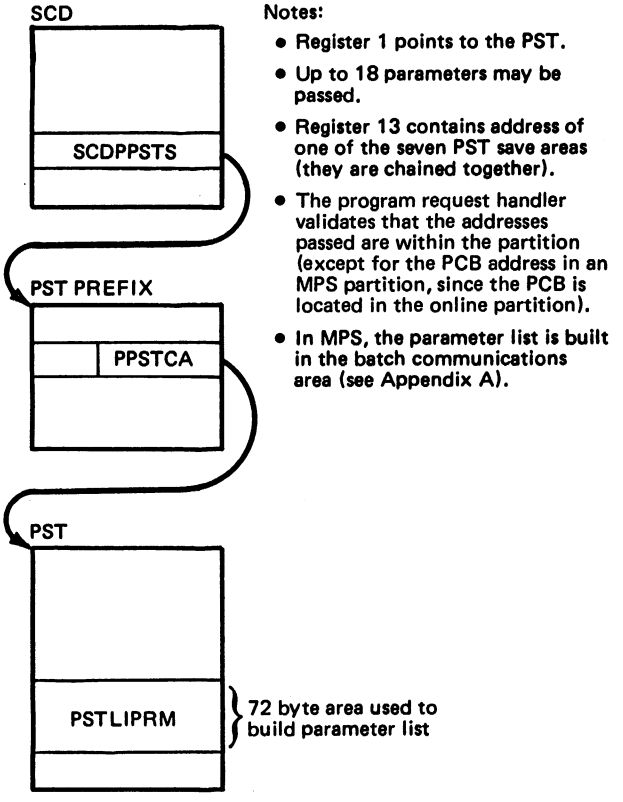


Figure 2-4. DL/I Call

DATA LOCATION AIDS

This section shows the user how to find the data buffer when an error occurs between checkpoint 4 and checkpoint 5 in Figure 2-1 on page 2-2. This section is divided into the following two parts:

- How to find the data in the DL/I buffer pool.
- How to find the data in the VSAM data area.

How to Find Data in the DL/I Buffer Pool

When processing a HDAM or HIDAM data base, the DL/I buffer pool is used, and the ESDS data is managed in the pool by the DL/I buffer handler. Note the following points:

- The buffer pool control block prefix (BFPL) contains the statistical information concerning buffer pool activity.
- One or more subpools may exist as specified by the user. The default is one. There is one subpool information table (SBIF) for each subpool.
- There are between 2 and 32 buffers per subpool (default is 32 buffers; the number can be changed via the HDBFR parameter). There is one buffer prefix (BFPR) per buffer.

- The start of each subpool is page aligned and may be 512 bytes or any multiple of 512 bytes up to 4096 bytes. However, the control interval processed in this buffer may be smaller.

Figure 2-5 illustrates how to find the data buffer in the DL/I buffer pool.

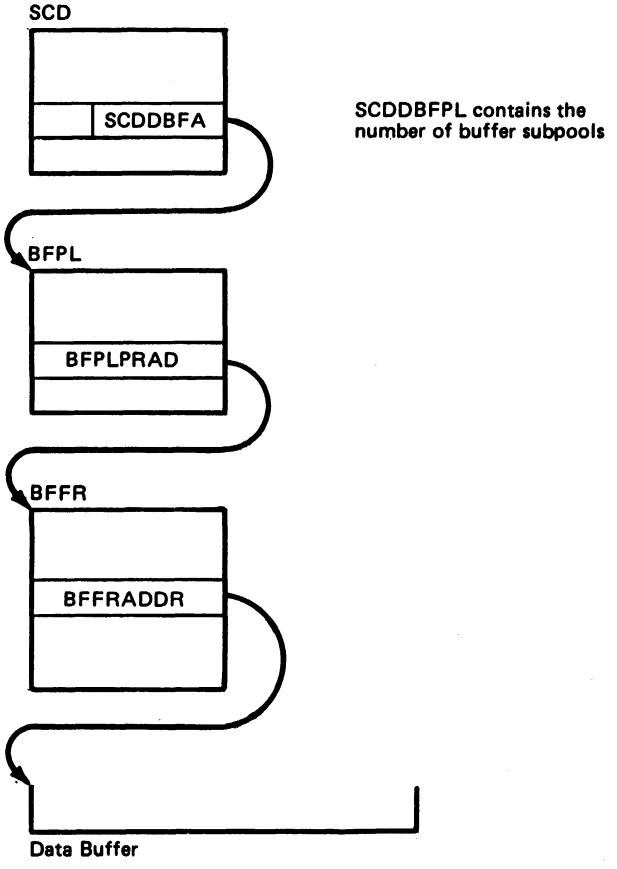


Figure 2-5. How to Find the Data Buffer in the DL/I Buffer Pool

How to Find Data in the VSAM Data Area

When processing a HISAM index data base or a secondary index, the DL/I buffer pool is not used (other than for load or insert work space). Data buffers are maintained by VSAM for the HIDAM and HISAM KSDS and the HISAM ESDS. Figure 2-6 on page 2-12 illustrates how to find the KSDS data buffer in the VSAM data area.

Note the following:

- To obtain the data buffer for the ESDS, locate the ESDS ACB Extension and continue as in Figure 2-6.
- To obtain another DMB, scan the DDIR until the desired DMB is found, and continue as in Figure 2-3 on page 2-5. The DMB name appears as the user-specified DBD name extended by at-signs (@) to a length of seven characters, if necessary. The eighth character is 'D'.

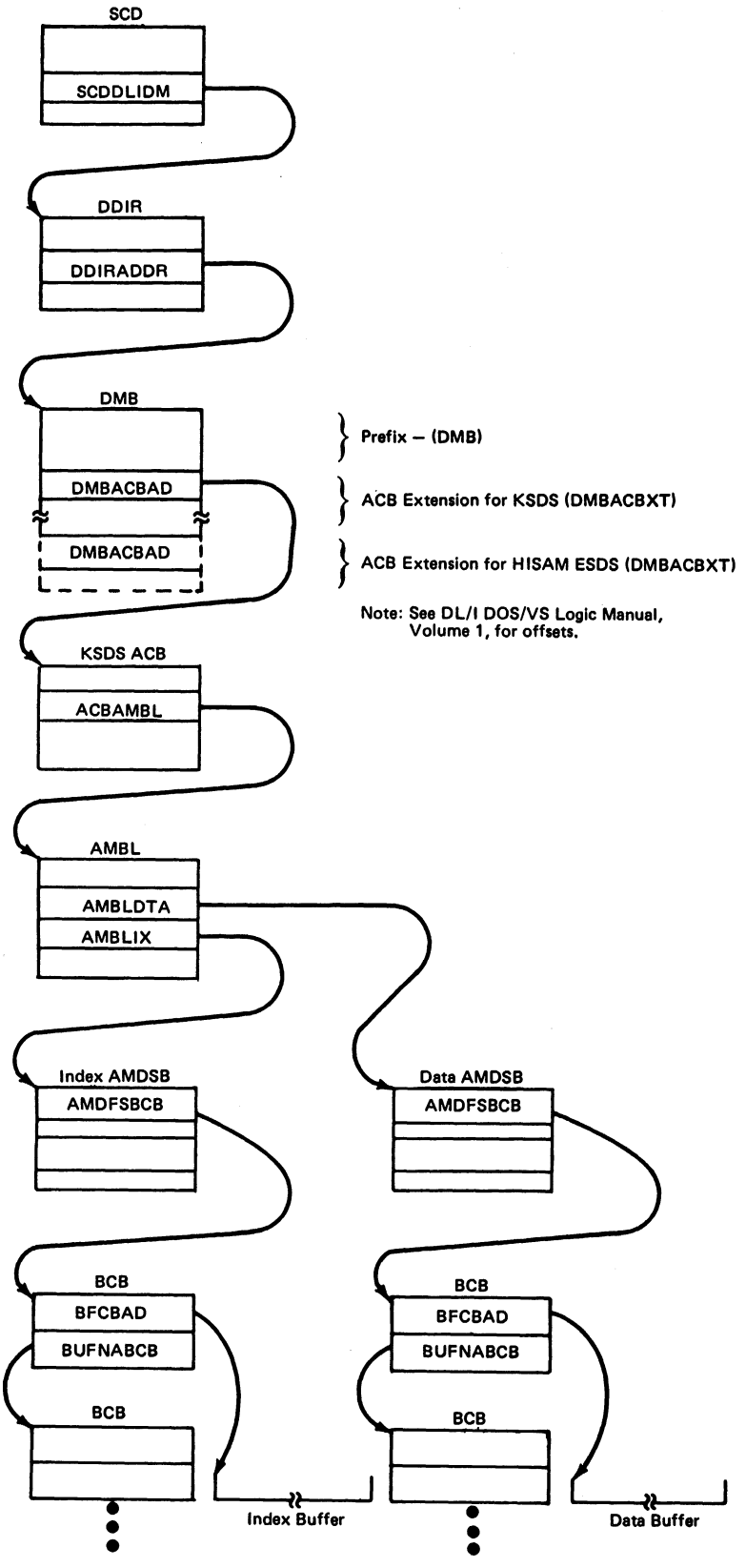


Figure 2-6. How to Find the Data Buffer in the VSAM Data Area

ONLINE EXIT CONDITIONS

By using the TCAFCTR and TCADLTR fields in the TCA, the user can determine the exit conditions of a DL/I call. The DL/I system places response codes in the fields as follows:

TCAFCTR	TCADLTR	Meaning
00	00	Normal exit
01	00	Invalid request (state already exists). For STRT/STOP calls, the data base was previously started or stopped. For TSTR calls, tracing was already active or tracing is not active.
02	00	For STRT/STOP calls, the DMB failed to initialize. For TSTR calls, the load of the trace module failed.
03	00	For STRT/STOP calls, the TESTCB failed.
04	00	Trace initialization failed.
08	00	Not scheduled for system call. Invalid CMAX value (CMXT). For STRT/STOP calls, the DMB was not in the directory.
08	01	PSB not in directory (PDIR).
08	03	Task already scheduled.
08	05	PSB initialization error.
08	06	PSB not in program ACT entry.
08	07	Term call when not scheduled.
08	08	Data base call when not scheduled.
08	09	Illegal MPS scheduling call.
08	FF	DL/I not active.
0C	01	Data base not opened or stopped.
0C	02	Scheduling conflict with MPS task.
xx	xx	Undefined return code.

The DL/I system uses the TCADLII field to control task scheduling as follows:

X'80' - The task is initially scheduled.
X'40' - The task is scheduled for a DL/I function.

The values of the flag bits in the TCADLII field used by DL/I are:

X'40' - DL/I task scheduled
X'20' - DL/I termination required
X'10' - First time for DTB
X'08' - System task indicator
X'04' - Task suspended by DL/I
X'02' - PST storage acquired for this task

The values of the flag bits in the TCA field (TCADLIBF) are:

X'80' - UIB acquired for user
X'20' - System DIB acquired

listing.

For details about the various trace entries associated with online scheduling, refer to Chapter 6, DL/I Trace Facility.

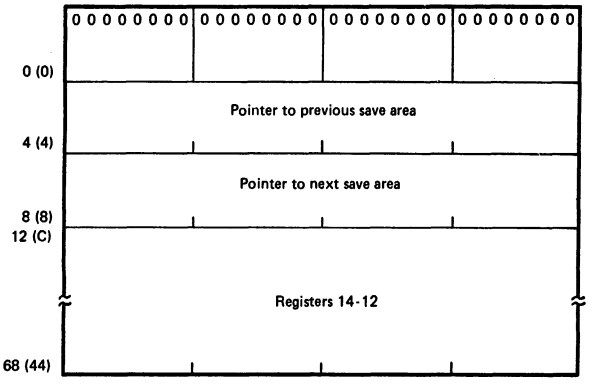
The following suggestions may help in debugging:

- The transaction must be present in the PPT of the CICS/VS system.
- There must be a corresponding program name entry in the ACT for DL/I online.
- All PSBs that the task wishes to use must be defined in the ACT.

Refer to DL/I DOS/VS Resource Definition and Utilities for details on ACTGEN.

TRACING CONTROL FLOW BY PST REGISTER SAVE AREA

DL/I provides seven save areas (see Figure 2-7 for format) in the PST control block for use in processing a DL/I call (see Figure 2-8).



2-7

Figure 2-7. DL/I Save Area Format

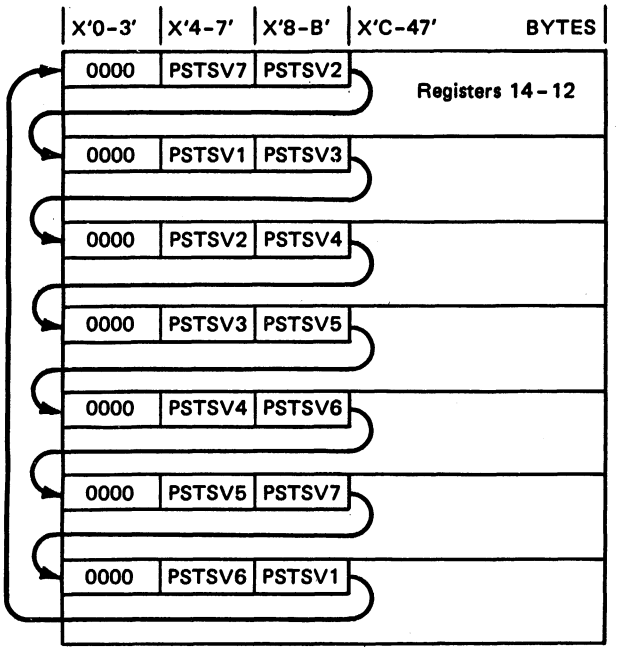


Figure 2-8. DL/I Register Save Areas

There is no way to determine exactly which module used each save area. However, register 12 usually is the base register for DL/I action modules. Register 15 usually points to the next module called, and register 14 usually points to the caller. For online interface modules like DLZODP, register 10 is usually the module base register. Registers 12 and 13 usually contain the CICS/VS TCA and CSA addresses. The save areas are located at PSTSV1. For batch applications, the first save area (PSTSV1) contains registers for one of the three routines. The calling module can be identified by register 14. The three routines are:

- Program Request Handler

  R1 PST (for Version 1.2; see R9)
  R2 label "SCD" in SCD

R3 user save area
R4 end of user's parameter list (R1 in the user's save area points to the start of the user parameter list).
R9 PST
R12 base register for program request handler
R14 return point in program request handler.

- STXIT ABEND Routine

R2 label "SCD" in SCD
R9 PST
R12 base register for STXIT ABEND routine
R14 return point in STXIT ABEND routine.

- User Completion Entry

R2 label "SCD" in SCD
R4 log I/O area
R9 PST
R10 base register for batch initialization.
R11 base register for batch initialization.
R12 base register for batch initialization.

JCB TRACE

The JCB trace table (see Figure 2-9) lists the last eight DL/I calls showing the last byte of the internal call analyzer function code and the last byte of the DL/I status code. (Also see "Interpreting a Dump After an Abend Message" on page 3-7.) The table is shifted left by two bytes.

JCBPREVF is loaded from JCBPRESF, and JCBPREVR is loaded from DBPCBSTC + 1.

DL/I status codes are summarized in DL/I DOS/VS Messages and Codes. Internal function codes are as follows:

Code Meaning

01 'GHU' Get Hold Unique
'GU' Get Unique
03 'GHN' Get Hold Next
'GN' Get Next
04 'GHNP' Get Hold Next Within Parent
'GNP' Get Next Within Parent
21 'REPL' Replace
22 'DLET' Delete
41 'ISRT' Insert

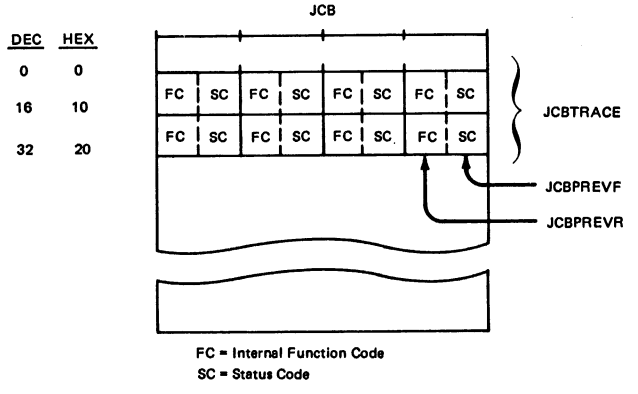


Figure 2-9. JCB Trace Table Format

These calls require a PCB and are traced in JCBTRACE. Only calls requiring a PCB are put in the trace table.

ONLINE TRACE FACILITIES

DL/I provides two methods for tracing activity in an online environment. The DL/I Trace Facility (see Chapter 6) can be used in either a batch or online environment. It traces DL/I internal areas while the various action modules are executing. The Online Trace explained here deals mainly with the external parameters for scheduling a DL/I call.

Online trace entries are provided to assist the user when errors occur in online. The entries are 32 bytes long and are in the format shown in Figure 2-10 . There are two 32-byte entries for scheduling and data base calls; one 32-byte entry for a termination call.

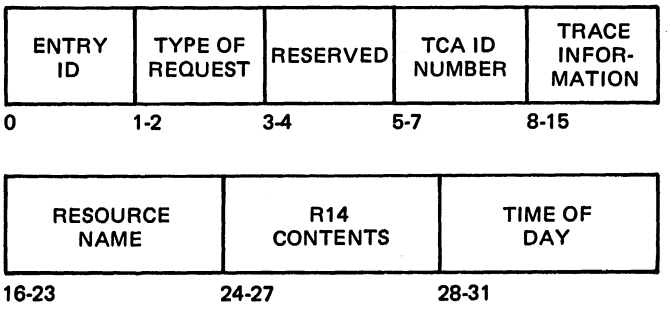


Figure 2-10. Online Trace Entry

ENTRY ID FIELD: Online trace uses an entry ID of X'F8'.

TYPE OF REQUEST FIELD: Three type-of-request codes are available:

- S(X'E2') - This type of request occurs at the completion of the scheduling call.
- D(X'C4') - This type of request occurs at the completion of the data base call.
- T(X'E3') - This type of request occurs at the completion of the termination call.

RESERVED: This 2-byte field is reserved for future use.

TCA ID NUMBER: The three-byte TCA ID number field contains the CICS/VS transaction ID (packed decimal 1-99999)

TRACE INFORMATION: The 8-byte trace information field contains the unique information as determined by the type of request. The trace information field information is described in the following sections. The S(X'E2') and D(X'C4') type of request entries place two entries in the trace table. The T(X'E3') type of request places one entry in the trace table.

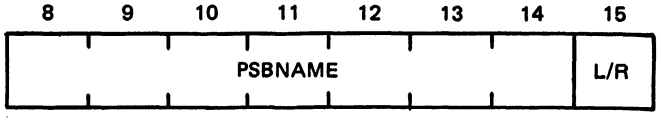
RESOURCE NAME: The 8-byte resource name field contains the name of the resource being used. This field is blank for DL/I.

R14 CONTENTS: The 4-byte R14 contents field contains the address of a DL/I control block or module. Contents of this field are unique to each type of request. The details for each type are shown below.

TIME OF DAY: This unsigned binary integer field represents time-of-day in units of 32 microseconds.

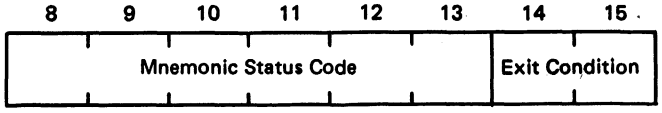
S(X'E2') Type of Request (Scheduling Call)

The trace information field for the first entry for the S(X'E2') type of request contains the PSBNAME and the L/R (local data base/remote data base) indicator.



PSBNAME = Name of the PSB being scheduled
L/R = Type of data base
' ' = local data base
'+' = remote data base
'*' = local and remote data base

The trace information field for the second entry for the S(X'E2') type of request contains the mnemonic status code and exit condition.

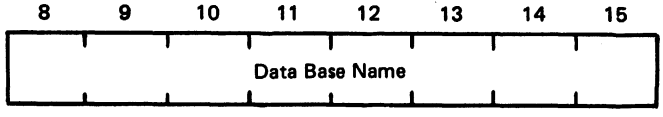


Mnemonic Status Code	Exit Condition	Meaning
SUCESS	00 00	Successfully scheduled
NO PSB	08 01	PSB not in directory (PDIR)
RESCHD	08 03	Task already scheduled
PSBERR	08 05	PSB initialization error
PSBATH	08 06	PSB not in program ACT entry
RETERM	08 07	Task already terminated
NOSCHD	08 08	Task not scheduled
ILEGAL	08 09	Illegal MPS scheduling call
DLIDWN	08 FF	DL/I not active
NOTOPN	0C 01	Data base not opened or stopped
MPSCFL	0C 02	Scheduling conflict with MPS task
UNDEFN	xx xx	Undefined return code

Exit Condition = See "Online Exit Conditions" for a complete list of exit condition codes.

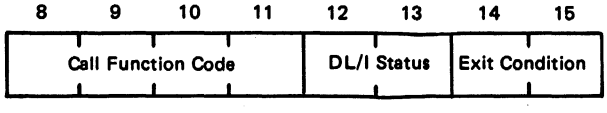
D(X'C4') Type Of Request (Data Base Call)

The trace information field for the first entry of the D(X'C4') type of request contains the data base name.



Data Base Name = Name of the DMB being accessed

The trace information field for the second entry of the D(X'C4') type of request contains the call function code, DL/I status, and exit condition.



Call Function Code = A complete list of call function codes is available in DL/I DOS/VS Application Programming: High Level Programming Interface and DL/I DOS/VS Application Programming: CALL and RQDLI Interfaces.

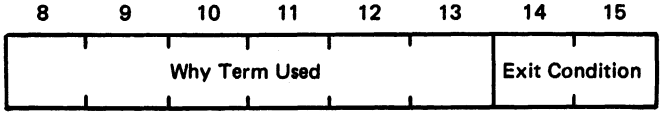
DL/I Status = A complete list of DL/I status codes is available in DL/I DOS/VS Messages and Codes.

Exit Condition = See "S(X'E2') Type of Request" for a list of exit condition codes.

Note: If a data base call is issued when the task is not scheduled or if any DL/I call is issued when DL/I is not active, the data base name is replaced by '*ERROR**' and the PCB status is replaced by '=>'. In the first case, the DL/I status code is X'0808'. In the second case, the DL/I status code is X'08FF'.

T(X'E3') Type Of Request (Termination Call)

The trace information field of the T(X'E3') Type of request contains the why-term-used code and exit conditions.



Code	Description
ABEND	The task was abnormally ended while being scheduled.
CANCEL	The task was cancelled by operator intervention.
DEADLK	The task was abnormally ended by DL/I to resolve a program isolation scheduling deadlock.
GETVIS	The task was abnormally ended by DL/I due to a lack of space to hold program isolation queue elements.
RETERM	A TERM call was issued when the task was not scheduled.
SYNCPT	Termination was caused by a CICS/VS sync point call.
USER	User issued a DL/I 'TERM' or 'T' call.

Exit Condition = See "S(X'E2') Type of Request" for a list of exit condition codes.

Register 14 Contents Field

Based upon the type of request, the following information is contained in the Register 14 Contents field:

'S' Type of Request:

Entry	Description
#1	Address of DL/I System Contents Directory (SCD)
#2	Address of CICS/VS TCA for task

'D' Type of Request:

Entry	Description
#1	Address of DL/I call parameter list
#2	Address of PCB

'T' Type of Request:

Entry	Description
#1	Address of DL/I Program Request Handler

ONLINE APPLICATION PROGRAM INTERFACE

Figure 2-11 on page 2-19 shows the paths taken for various calls from the application program through the online interface to DL/I. The prime purpose of the paths is not to describe the

sequence of events but to show the relationship between modules during calls. The arrows in the figures depict the direction of the transfer of control and return path.

From the point of exit to the DL/I call analyzer, the flow is the same as in a DL/I batch system. Refer to "Batch Initialization/Termination (DLZRRC00)" in this chapter for batch flow.

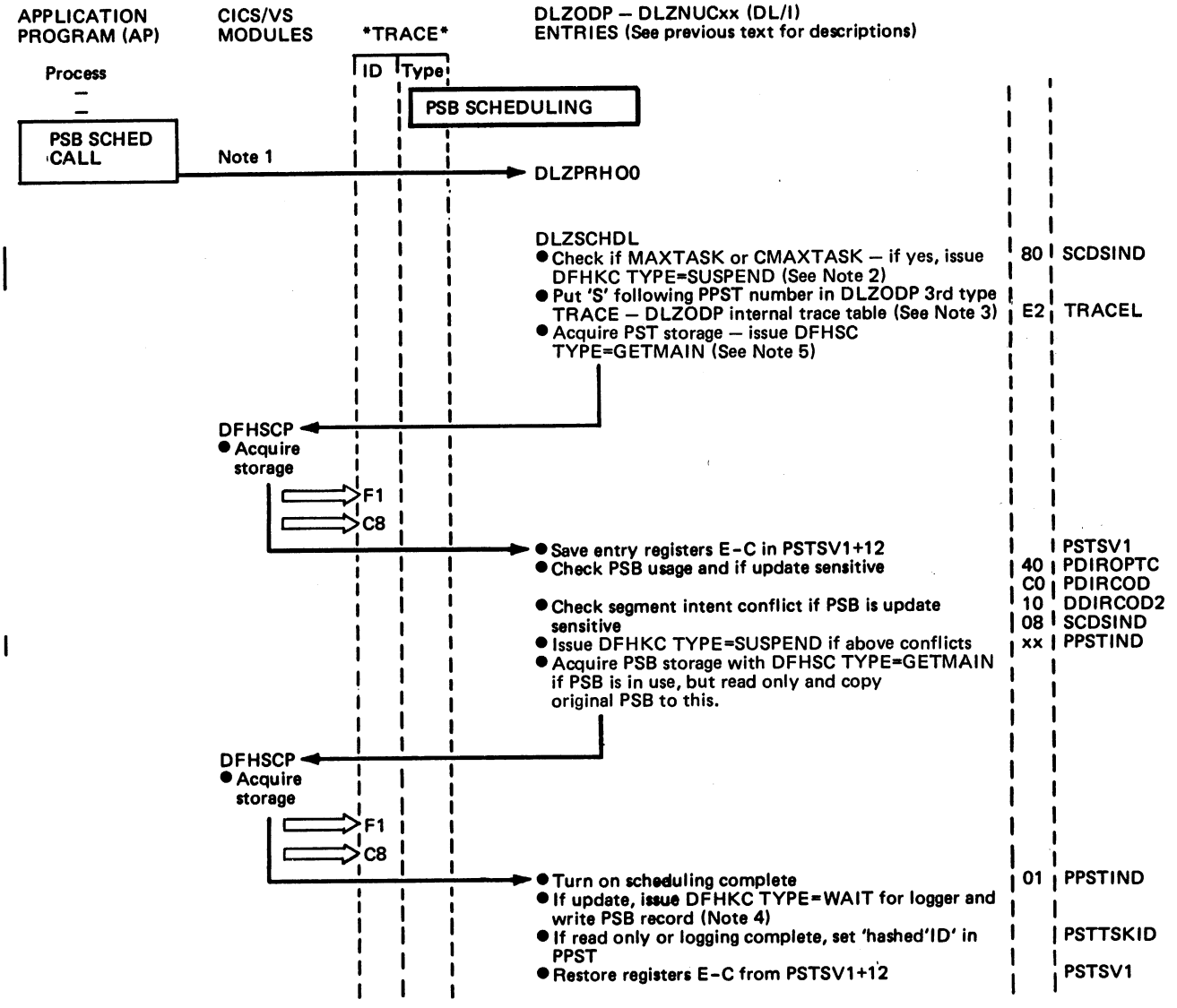


Figure 2-11 (Part 1 of 3). Call Processing Overview

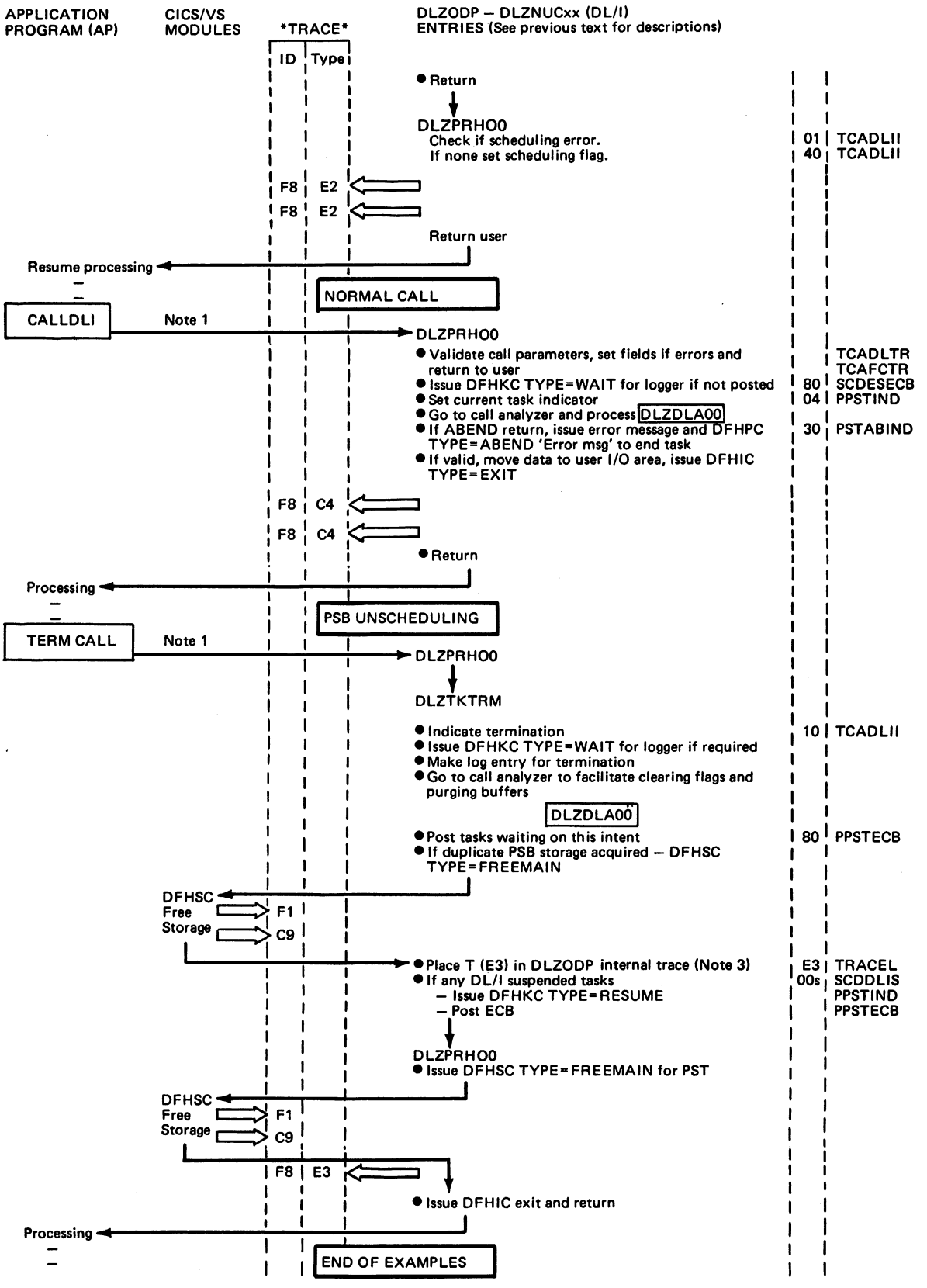


Figure 2-11 (Part 2 of 3). Call Processing Overview

Notes:

1. All communication of application (AP) and program request handler is via language interface. This is link edited to AP by the CALL macro expansion and it saves registers before branching to PRH (address in VSE COMREG).
2. Before DL/I issues CICS/VS DFHKC "SUSPEND", the task is placed on a DL/I suspend chain (address in SCDDLIS).
   Termination routine DLZTKTRM checks this list and issues DFHKC "RESUME" if any task suspended earlier.
   If suspended task is "STALL PURGED" by CICS/VS, then DFHKC enters DLIOIO through DFHDLIAL (in DLZNUC) to ensure that the task is removed from the suspend chain accordingly.
3. Within DLZNUCxx, module DLZODP, there is a trace for PSB calls and term calls indicating the PPST number.
4. When the task is put into DFHKC "WAIT", then PPSTIND is set to reason and PPSTECB is turned off. Return after wait causes retry of operation that caused wait.
5. Task could be suspended by CICS/VS if GETMAIN fails.

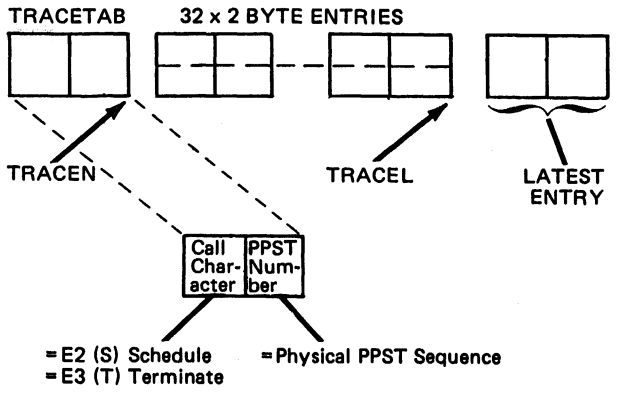


Figure 2-11 (Part 3 of 3). Call Processing Overview

Refer to Figure 2-11 on page 2-19 for scheduling calls, terminating calls, and DL/I data base calls. Refer to the HIPO diagrams in the DL/I DOS/VS Logic Manual, Volume 2, for system scheduling calls (SYSTEMDL), and for system calls (STRT, STOP, CMXT). The CALLDLI macro is used to establish a call to the online interface.

CALLDLI establishes the parameter list and save area and issues a call to ASMTDLI. Refer to the section "Code Generation for a CALLDLI" in the DL/I DOS/VS Logic Manual, Volume 1, for a breakdown of CALLDLI.

The CBLTDLI and PLITDLI interface should be used for COBOL and PL/I programs and the RQDLI command for RPGII programs as in batch.

The techniques used to interface CICS/VS, CICS/VS transactions, and DL/I are described in Figure 2-11 on page 2-19.

The CALLDLI macro generates the following code:

- A CICS/VS GETMAIN for:
  - User save area (RSA)
  - Parameter storage (CICS/VS GETMAIN)
- MOVE for parameters pointers:
  - Move the user parameter list from the call list to parameter storage.
- SAVE for CSA register (register 13) at RSA+8.
- Load parameter register (register 1).
- Load entry point of language interface.
- Transfer control of language interface.

RETRIEVE SUBROUTINE TRACE

Module DLZRLNKD is issued whenever branching to any subroutine within retrieve is necessary. Entry to a subroutine is possible via labels:

```
DLZRENT 1
DLZRENT 2
```

Exit from a subroutine is taken via label:

```
DLZREXT
```

All necessary information to branch to or to return from a subroutine is stored in part of the PST beginning at displacement X'160' (PSTDLROM) as follows:

Fullword at X'160' (PSTDLROM) points behind the last valid entry in this area.

Starting at displacement X'164' (PSTDLROM+4) each pair of fullwords contain registers 12 and 14 at entry to the desired subroutine where register 12 is the old base register and register 14 is the link register in the calling routine.

Register 12 can be compared with entries in subroutine entry table at the end of the retrieve module which can be recognized by locating the characters SET* at the end of the retrieve module.

(Refer to DLZDLRA0 for link sequence and to DLZRLNKD for the branch and return mechanism, the layout of the subroutine entry table, the subroutine explanation DSECT and the subroutine call matrix).

LOGICAL RETRIEVE - DLZDLR00 MACROS

Logical retrieve consists of 60 subroutines totaling approximately 25,000 bytes. Each call depends on positioning achieved by previous calls. Therefore, the call path differs, and problem determination in logical retrieve becomes highly complex. Any possibility of the problem lying outside logical retrieve should be exhausted first (see "Online Wait State Debugging Aids" on page 2-27). In order to debug logical retrieve problems the following points should be known:

- The last segment successfully retrieved (level, key/RBA).
- Call and resulting function requested.
- The segment expected (level, key/RBA).
- The segment (if any) obtained (level, key/RBA).

You should use the HIPO charts in the DL/I DOS/VS Logic Manual, Volume 2, to follow the control flow between the major subroutines. Linkage between subroutines is always via the DLZRLNK macro. The maintenance function of macro DLZRLNK M (see "DLZRLNK M Macro" on page 2-24) may be used to PDUMP control blocks on entry/exit of subroutines. This should enable you to ascertain if the correct logical path was followed and, if not, at which subroutine the change occurred. Alternatively, by checking important control block values in each PDUMP, it should become apparent which subroutine was responsible for a failure.

INTERNAL MACROS

The following internal macros are used in logical retrieve and it may be helpful to understand their functions before using the listings/microfiche for DLZDLR00.

DLZRCLL Macro

This macro issues the DLZRLNK C linkage macro to transfer control to another logical retrieve subroutine. It has the following format:

```
DLZRCLL name(,return)
```

where:

name — is the name of the subroutine to be called.

return — if:

- blank — return to the next sequential instruction in the subroutine issuing the call after processing new subroutine.
- label — return to label within subroutine issuing the call after processing new subroutine, or return to a label of another subroutine processing subroutine.
- DLZREXT — do not store return registers in PST; therefore, next return call will be to the caller of the subroutine issuing the call.

To determine which microfiche card or listing to use, turn to the DL/I DOS/VS Logic Manual, Volume 1, "Section 4. Directory". The name shown on the microfiche cards or listings is the relocatable module name. The subroutine names are listed as entry points.

EXAMPLE 1:

```
DLZYENT  DLZRHDR
            .
            .
            .
         DLZRCLL DLZYSTC,YSTARTA
            .
            .
            .
YSTARTA  MVC....
```

Processing is transferred from subroutine DLZYENT to subroutine DLZYSTC (microfiche card DLZDLKD0). On exit processing to subroutine DLZYENT at label YSTARTA.

EXAMPLE 2:

```
DLZLTW   DLZRHDR
            .
            .
            .
         DLZRCLL DLZPCHK
         MVC....
            .
            .
            .
DLZPCHK  DLZRHDR
            .
            .
            .
         DLZRCLL DLZSETL,DLZREXT
            .
            .
            .
```

Processing is transferred from subroutine DLZLTW (microfiche card DLZDLRC0) to DLZPCHK (microfiche card DLZDLRE0) by the first call. The second call transfers processing from DLZPCHK to DLZSETL. On exit from DLZSETL processing is transferred to the MVC instruction in DLZLTW.

DLZREXT Macro

This macro generates an unconditional branch to label DLZREXT in the DLZRLNK linkage macro. The previous subroutines base and return registers (R12 and R14) are loaded from the PST and a branch to R14 is performed.

DLZRHDR Macro

DLZRHDR release information.

```
Name: Name of retrieved CSECT.
Date: Release information.
```

where release information is:

```
V = version number
R = release number
I = future expansion
P = latest PTF applied.
```

This macro issues the DLZRLNK D linkage macro to create all DSECTs and USING statements necessary to process any logical retrieve subroutine. It is followed by a description of the function of the subroutine.

DLZRTLR Macro

This macro appears at the end of each subroutine and generates drop statements for every register.

LINKAGE MACROS (DLZRLNK N)

DLZRLNK C Macro

This macro generates the statements necessary for linkage between logical retrieve subroutines storing return and base registers in the field pointed to by PSTLSVPT.

DLZRLNK D Macro

This macro generates all DSECTs and establishes addressability necessary to process all logical retrieve subroutines.

DLZRLNK M Macro

This macro is only supplied in source form but may be useful for debugging logical problems. The macro may be assembled with the required options and logical retrieve relinked (including DLZDLR). Alternatively a dummy version may be assembled and "zapped" to activate when required.

EXAMPLE:

```
DLZRLNK M,release information              c
        (name1,option1),(namen,optionn),   c
        NUMBER = num,                      c
        CONEK  = field,                    c
        LIMIT  = (lim1,lim2),              c
        SKIP   = sk
```

where:

- release information -- is:

  V = version number

R = release number
I = future expansion
P = latest PTF

- name1,namen -- are names of subroutines requiring dump

- option1,optionn -- are:

  E = PDUMP on entry to subroutine
  X = PDUMP on exit from subroutine
  J = DBPCB JCB included in PDUMP
  P = PST included in PDUMP
  B = section buffer pool included in PDUMP (see LIMIT). Default is EXJPB. General registers and save areas are always included.

- num -- is total number of PDUMPS required, default = 100.

- field -- is a left bound character string (maximum length 40 bytes) is compared to the same number of bytes in key feedback area. PDUMP occurs when a match is obtained on entry/exit in specified subroutine.

- lim1,lim2 -- are buffer dump limits:

  lim1 number of bytes prior to present buffer address (PDUMP lower limit).
  lim2 number of bytes after present buffer address (PDUMP upper limit), default = 200,400.

- sk -- is number of times, conditions for PDUMP match before PDUMP is taken.

Note: R14 of the PDUMP shows the current entry point of the subroutine. There is no ID on the PDUMP. R14 must be used to ascertain which PDUMP it is.

The following steps should be followed to generate support that may be activated through PDZAP:

To generate:

- Assemble the macros with the following parameters:

```
DLZRLNK M,release information       c
        (DLZCLRP,EXJPB),            c
        (DLZWIPE,EXJPB),            c
        (DLZMOVA,EXJPB),            c
        (DLZMOVB,EXJPB),            c
        (DLZDELT,EXJPB),            c
        (DLZPSDB,EXJPB),            c
        NUMBER = 100,               c
        CONCK  = ABCDEF...(DL40),   c
        LIMIT  = (200,400),         c
        SKIP   = 10
```

  This generates the necessary tables in the phase.

- Use the assembly listing and PDZAP to zero out the generated parameters. (See "To Activate", the second bulleted item for format).

- Find label DLZREXT in the listing and change the following instructions:

```
From          To

B MNTENTRY  BR R12           07FC0000
B MNTEXIT   L  R1,PSTLSVPT   58107160
```

  (The source code is given for checking the displacement into the PST of PSTLSVPT for user's PTF level, etc.).

To activate:

- Reset instructions around label DLZREXT.
- Insert the required parameters into tables:

Field-name	Bytes	Bits	Usage
MNTNUMBR	2		Total number of PDUMPS.
MNTABLE	2	0-9	Ten leftmost bits of VCONST address for this subroutine.
		10-11	10=PDUMP on exit from subroutine. 11=PDUMP on entry and exit.
		12	Reserved
		13	PCB JCB dumped.
		14	PST dumped.
		15	Section of buffer pool dumped.

Note: These fields are always consecutive. For VCONST addresses, see DLZRLNK M listing - subroutine entry table.

MNTCONCL	2	Length of compare argument.
MNTCONCK	(40max)	Left justified compare argument.

These fields are always consecutive:

MNTB	2	Number of bytes to present buffer address.
MNTF	2	Number of bytes after present buffer address.
MSTSKIP	2	Number of PDUMPS initially skipped.

LOGICAL RETRIEVE DEBUGGING AIDS

- To obtain a dump at the beginning of logical retrieve, the DLZRLNK M linkage macro may be used specifying DLZNOOP in the name operand for a dump on entry. This should help to ascertain if the problem is caused by an error prior to logical retrieve.
  - Check user parameter list (see message DLZ261I in Chapter 3).
  - Input to logical retrieve (see "Retrieve DLZDLR00" HIPO diagram in the DL/I DOS/VS Logic Manual, Volume 2).
- To obtain a dump on exit from logical retrieve/entry to buffer handler, the DLZRLNK M linkage macro may be used specifying DLZBH in the name operand.

  Note: request a dump on entry because exit from DLZBH is not via DLZRLNK.

  See "BUFFER HANDLER (DLZDBH0n)" on page 4-2 for debugging aids.

  To reduce the chance of an APAR being closed RETURN, as much of the following information as possible should be sent:

  - Storage dump.
  - Retrieve trace (see "Retrieve Subroutine Trace" on page 2-22)
  - Call sequence: Preferably DL/I test program call cards, which produce the error or application program including call before failure occurred.
  - DBDGEN, etc.
  - Data Base Dump of affected record(s).
  - Log file(s).
  - System console sheet
  - EREP printout.
  - LISTCAT ALL.
  - Application listing(s).
  - User recovery technique narrative.
  - Data Base Description (dependence of segments).

- The following register usage convention is used during the majority of the subroutines:

  R4 Level table for current level
  R5 SDB for current segment type
  R6 Present buffer address

  Note: At the beginning of logical retrieve these values will apply to the last processed and will later be updated.

  R7 PST
  R11 Base register for linkage module (DLZRLNK)
  R12 Base register for current subroutine
  R14 Return address

- The following control block fields contain information which may be of use during debugging. Care should be taken to establish whether their contents refer to the present request or the previous call.

  KEEPIT EQU JCBSTOR7 JCB

  If KEEPIT equals X'01' logical retrieve is processing in DLZLTW attempting to use previous position to satisfy this call.

  KEEPIT+1 EQU JCBSTOR7+1 JCB

  If KEEPIT+1 equals X'00', processing is for GN call. If KEEPIT+1 equals X'01', processing is for GU or ISRT call.

  JCBLEV1C JCB

  Contains a pointer to the lowest level table successfully processed for current request.

  Note: If KEEPIT = X'01', JCBLEV1C will point at level table applicable to segment accessed.

  LEVF1 LEV TAB

  There is one level table for each level defined in the DBDGEN. The contents of LEVF1 may be used to determine how far processing has progressed in the hierarchical path. All later levels are set to LEVEMPTY. This shows positioning for levels higher than the target level and may be useful in solving "wrong record returned" or hierarchical return code problems. Once the target level is found, LEVEMPTY in LEVF1 is turned off.

  CURTTR EQU JCBSTOR5 JCB

  On return from the buffer handler PSTBYTNM (current RBA) is set in CURTTR. For a no record found condition, CURTTR is set to zeros.

  PSTLSVPT SAVE AREA PST

  The address of this label in the PST points to the next 8 byte save area to use for logical retrieve linkage (R12 - R14).

  Note: Once processing returns to the caller, the save area is reused; therefore, it cannot be used to show the flow through logical retrieve.

ONLINE WAIT STATE DEBUGGING AIDS

Figure 2-12 on page 2-28 suggests steps for online wait state debugging.

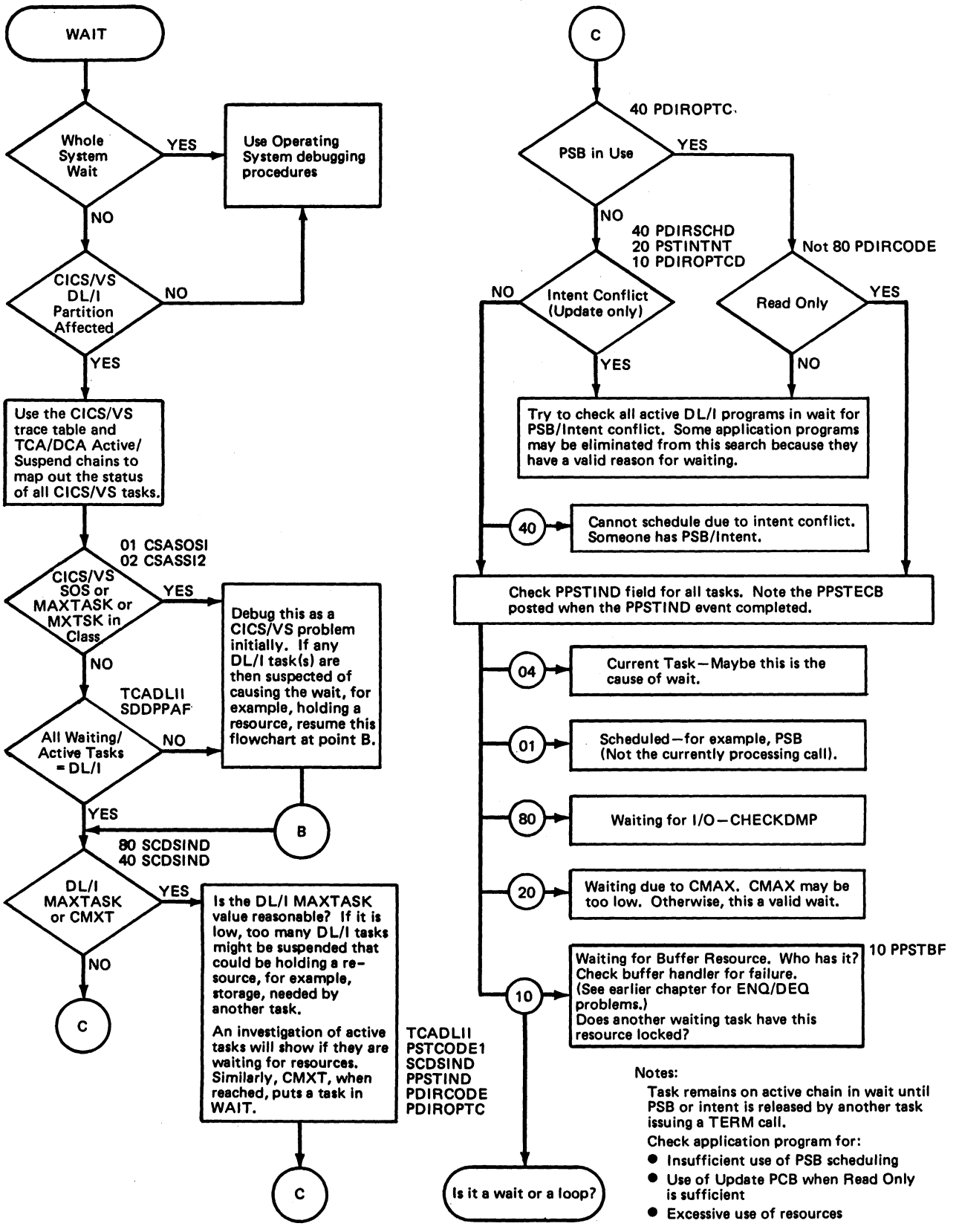


Figure 2-12. Online Wait State Debugging

HIGH LEVEL LANGUAGE DEBUGGING

High Level Language (HLL) Debugging allows diagnostic information to be supplied by the high level language (PL/I) and by DL/I. The HLL support is for batch and MPS batch execution of DL/I.

This support is activated whenever there is a program check. A STXIT PC routine gets control when:

- DL/I has STXIT support requested (UPSI bit 7 = 0) for batch
- MPS batch is being used.

In case of a program check, if the error was from the application program or the PL/I compiler, the saved information about the PL/I error routine is examined, the return address to DL/I is modified in the user save area to point to a special entry into the DL/I error handling code and a branch is taken to PL/I to handle the error. When PL/I completes its error handling routine, it branches back to DL/I at the modified address. Then DL/I performs its normal error processing. Since PL/I issues an EXIT PC macro within its code, DL/I must not issue this macro. If the error was within DL/I code, there is no branch to PL/I.

FIELD LEVEL SENSITIVITY DEBUGGING

Field level sensitivity allows the user to specify the fields from the physical definition that are to be included in his version of the segment. The user's view need not be in the same sequence as the physical view. DL/I maps the fields from the physical segment into the user's view.

In addition to the basic support, the following items are available to the user of field level sensitivity:

- Virtual fields
- Automatic data format conversion
- User field exit routine
- Dynamic segment expansion
- No sequence field order restrictions

Suggestions:

- In the PSB generation, the SENFLD statement follows the PCB and SENSEG macro statement.
- In the PSB generation, the VIRFLD statement follows the PCB and SENSEG macro statements. This error condition may be indicated by MNOTE VFLD100 or VFLD110.
- Only the following data conversions are supported:
  - C to C (length conversion only)
  - F to H, P, X, or Z
  - H to F, P, X, or Z
  - P to F, H, X, or Z
  - X to F, H, P, or Z
  - Z to F, H, P, or X

  Conversions not supported may be indicated by error message DLZ943I during ACB Generation.
- The maximum value for F, H, and X data fields is 2,147,483,647 the minimum value is -2,147,483,648.
- Conversion limits for packed decimal and zoned decimal is 16 characters (± 9999999999999999).
- Length conversions for F, H, P, X, or Z data fields are padded to the left with zeros. Truncation occurs from the

left. Truncation of significant digits results in the field being set to the maximum (minimum if negative) value, and the return of a 'KA' status code.

- Length conversions for C to C data fields are padded on the right with blanks. Truncation occurs on the right; significant digit truncation results in a 'KB' status code. This condition may be indicated by MNOTE VFLD270.

- An invalid zoned or packed decimal field results in the null value being provided for the converted field and a 'KC' status code.

- Non-supported field conversions are allowed by ACB generation only if a field exit routine is supplied. Non-supported field conversions encountered during field access result in a 'KD' status code.

- A 'C' data field in the physical segment will not be converted to a different field type in the user's view. Length adjustment is performed. An incorrect status code which is not reset by the user's exit routine results in immediate termination of the request.

- The START=position parameter in the SENFLD macro statement must be numeric and in the range of 1 through 32,767 or alphameric and contain the name of a previously defined field. An error condition may be indicated by MNOTE SFLD150 or SFLD155.

- The BYTES=n parameter in the SENFLD macro statement must be numeric in the range of 1 through 256. An error condition may be indicated by MNOTE SFLD130 or SFLD190.

- The NAME=fldname parameter in the SENFLD macro statement must be 1 to 8 characters with the first character alphabetic and the remaining characters alphanumeric. An error condition may be indicated by MNOTE SFLD140.

- The RTNAME=prog parameter in the SENFLD macro statement must be 1 to 8 characters with the first character alphabetic and the remaining characters alphanumeric. An error condition may be indicated by MNOTE SFLD210.

- The START=position parameter in the VIRFLD macro statement must be numeric in the range of 1-32,767 or alphanumeric and contain the name of previously defined field. An error condition may be indicated by MNOTE VFLD150 or VFLD155.

- The BYTES=n parameter in the VIRFLD macro statement must be numeric and in the range of 1-256. An error condition may be indicated by MNOTE VFLD190.

- Do not specify the BYTES=n parameter in the VIRFLD macro statement with floating-point field types D, E, or L. An error condition may be indicated by MNOTE VFLD180.

- The NAME=fldname parameter in the VIRFLD macro statement must be 1 to 8 characters with the first character alphabetic and the remaining characters alphanumeric. An error condition may be indicated by MNOTE VFLD140.

- The RTNAME=prog parameter in the VIRFLD macro statement must be 1 to 8 characters with the first character alphabetic and the remaining characters alphanumeric. An error condition may be indicated by MNOTE VFLD210.

- The TYPE=t parameter in the VIRFLD macro statement must be numeric. An error condition may be indicated by MNOTE VFLD260.

- SSA information should be in the format of the application's view of the field.

- Sensitivity must be specified for any sequence fields in the segment if the field sensitivity for a segment is specified in conjunction with insert sensitivity. An error condition may be indicated by message DLZ942I.
- Sensitivity for inserting bi-directional logical children requires sensitivity to both physical and logical twin sequence fields. An error condition may be indicated by message DLZ942I.
- The application program must be sensitive to the entire concatenated key if insert sensitivity is specified for a logical child. An error condition may be indicated by message DLZ938I.
- If a field and subfields of a field appear in both the physical and application view of a segment, the subfields must be subfields in both views. The order of a subfield may change.
- If a field overlaps either end of another field in either the physical or application view, the intersection must be defined by subfields.
- When the user specifies a routine to be given control whenever a field is referenced, the routine must be cataloged in the core image library. In addition to the normal conventions for registers 14 and 15, register 1 points to the parameter list shown in Figure 2-13.

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	1	FERPEC		Entry code (GET = FERPGET, PUT = FERPPUT)
1	1	1	FERPFNCT		Function code (RETRIEVE = FERPRET, INSERT = FERPINS, REPLACE = FERPREP)
2	2	1	FERPCSC		Conversion status code: blank = OK A = numeric B = character truncation error C = format error D = type conflict
4	4	4	FERPDSA		Physical segment address (if variable length, it points to the two-byte length field)
8	8	2	FERPPSL		Physical segment length
10	A	2	FERPPFL		Physical field length
12	C	4	FERPPFA		Physical field address
16	10	4	FERPUSA		User's segment address
20	14	2	FERPUSL		User's segment length
22	16	2	FERPUFL		User's field length
24	18	4	FERPUFA		User's field address
28	1C	4	FERPFSBA		FSB address
32	20	32	FERPUWA		User's work area

Figure 2-13. Field Level Sensitivity Parameter List

FIXED BLOCK ARCHITECTURE SUPPORT

Fixed block architecture support allows the user's DL/I data base and log files to reside on an FBA device. DL/I addresses the data base by block number rather than the familiar CKD (count-key-data) disk addressing of track and cylinder.

The functions impacted by fixed block architecture support are:

- DBDGEN
- Space management
- Utilities (DLZDVCE macro)

Suggestions:

- The SCAN=blks parameter on the DATASET macro statement may be any integer from 0 to 32,767. The default is equivalent to three cylinders on a CKD device.
- The DEVICE=FBA parameter on the DATASET macro statement must be used to specify an FBA device. An error condition occurs if an FBA device is specified and the data base resides on a CKD device. This error may be indicated by error message DLZ831I. Either mount the data base on an FBA device and rerun the job or regenerate and recatalog the DBDGEN specifying the correct DEVICE= parameter and then rerun the job.

PATCH AREA

A 128-byte patch area can be used by any module with addressability to the SCD. If operating in batch mode, the patch area is in the module DLZBNUC0. If operating in online mode, the patch area is in the online nucleus (DLZNUCxx) being used.

The patch area immediately follows the SCD, at displacement SCDLNGTH from label DLZSCD. The start of the patch area can be located by looking for the characters "* PATCH AREA *".

ASSEMBLY OF DL/I DOS/VS MODULES

Assembler error message IPK182 -- ALIGNMENT ERROR IN OPERAND 2 ... may be issued during assembly of some DL/I DOS/VS modules; for example, modules DLZDLOC0 (open/close) and DLZTPRT0 (trace print utility). **This message can be ignored**, the assembly is complete.

The message indicates an RX (i.e., register-and-indexed storage) operation which does not require fullword alignment on S/370 and successor systems.

LINKAGE EDITOR WARNINGS

The linkage editor identifies unresolved external references that result in unresolved address constants during link editing DLZDBH00 or DLZACT (application control table, ACT). The latter is the method of link editing the DL/I online nucleus (DLZODP, DLZNUCxx, and DLZNUC).

The unresolved external references are caused by WXTRNs (weak external references) which, unlike the EXTERN, do not cause an automatic library search for the module that contains the external symbol. A WXTRN is resolved only if the external symbol specified is defined in one of the modules already involved in the link edit.

The following unresolved external references are expected and cause no operational error:

Module	External Reference
DLZDBH00	WXTRN ONLTRAC
DLZODP	WXTRN ONLTRAC

Notes:

1. There may be other references to the ONLTRAC symbol, but they should not cause any operational problems.

2. Additional Linkage Editor warning messages may appear when linking the DL/I code. The link edit process should continue. Check the output (SYSLST). See the Program Installation Directory for additional information.

CHAPTER 3. INTERPRETING AND DEBUGGING DUMPS

This chapter provides:

- A review of ABEND messages
- The names of the module(s) that issue the ABEND messages
- An explanation of the error (if needed)
- Suggested paths to follow in locating the reason for a dump after an ABEND message

Pictorial layouts are used for:

- Locating save areas
- Showing contents of save areas
- Locating parameter lists
- Locating DMBs
- Locating the buffer pool and buffers
- Showing parent/child relationship
- Checking return codes

DL/I DUMPS

DL/I DOS/VS produces three types of dumps:

- **Unformatted dumps** display register contents and the contents of a section of storage at the time of failure.
- **Formatted dumps** provide additional information by:
  - locating DL/I control blocks in storage
  - dumping each control block separately
  - identifying each block with a control block heading
- **Problem determination dumps** are unformatted dumps that are identified by a special header and written to a special dump data set. The header and dump information are used by problem determination programs such as the Interactive Problem Control System (IPCS) to produce formatted dumps, problem analysis reports, etc. Problem determination can be performed offline to minimize the impact of dump analysis on system performance.

DL/I DUMP CONTROl

DL/I dumps are taken in the event of a DL/I:

- online task failure
- online system failure
- batch or MPS batch application failure
- batch utility failure

where failure is an abnormal ending of a job or task.

The following table shows the type of dump that can be produced for each type of failure.

	Unformatted	Formatted	Problem Determ.
Online task	-	X	X
Online system	-	X	X
Batch/MPS application	-	X	X
Batch utility	X	X	X

ONLINE FORMATTED TASK DUMP

Online formatted task dumps are produced by DLZFTDP0. This program dumps DL/I system control blocks, including:

- system contents directory (SCD) and SCD extension
- PST prefix table (PPST)
- buffer pool prefix
- subpool information table

DLZFTDP0 also dumps DL/I task control blocks, including:

- ACT entry for current program
- non-empty buffers in subpool
- PSB scheduled by task
- PCBs, JCBs, SDBs, level tables, etc., of the scheduled PSB

Invoking An Online Formatted Task Dump

A formatted dump is automatically invoked for online task failures if the formatted task dump program, DLZFTDP0, is link edit-

ed with the DL/I online nucleus. If DLZFTDP0 is not part of the DL/I nucleus, formatted dumps are not taken. DL/I does not provide a user interface to suppress online task dumps.

ONLINE FORMATTED SYSTEM DUMP

Online formatted system dumps are produced by DLZFSDP0. This program is sensitive to the DL/I environment it is used in. For example, in addition to the standard DL/I control blocks, this program dumps intersystem communication control blocks if CICS/VS intersystem communication is being used by an online DL/I system.

Invoking An Online Formatted System Dump

A formatted dump is automatically invoked for online system failures if the formatted system dump program, DLZFSDP0, is in the core image library and if no SYSDMP is in effect for the partition. If DLZFSDP0 is not in the library, formatted system dumps are not taken. DL/I does not provide a user interface to suppress online system dumps.

BATCH AND MPS BATCH APPLICATION DUMPS

Formatted batch and MPS batch application dumps are produced by DLZFSDP0.

Invoking Batch and MPS Batch Application Dumps

DL/I dumps for batch and MPS batch applications are requested through bit settings in the UPSI byte:

Bit 5 =0 Storage dump on set exit (STXIT) abnormal termination if STXIT active (that is, bit 7 = 0).

=1 No storage dump on set exit (STXIT) abnormal termination.

When using the UPSI byte for application dumps, the SYSDMP option must not be invoked.

Note: STXIT linkage to DL/I for abnormal termination is always active under MPS.

BATCH UTILITY DUMPS

DL/I utilities that use DL/I (e.g., HD Reload and Data Base Backout) produce formatted dumps. DL/I utilities that do not use DL/I produce unformatted dumps.

PROBLEM DETERMINATION DUMPS

Problem determination dumps are selected through job control statements (see VSE/Advanced Functions System Control Statements). The following job control statements tell DL/I to take unformatted problem determination dumps.

```
// OPTION SYSDMP
```

This requests the system to write the output of any system dump that might be caused by the next, or any subsequent, job or job step in the particular partition to SYSDMP. If the system dump is a DL/I dump, it will be written in problem determination format.

```
// OPTION SYSDMP
```

This requests the system to write the output of any system dump that might be caused by the next, or any subsequent, job step of the current job. If the system dump is a DL/I dump, it will be written in problem determination format.

PROCESSING DUMP OUTPUT

Unformatted and formatted dumps for batch and MPS batch application programs can be printed on the line printer to which SYSLST is assigned. These dumps can also be written onto magnetic tape or disk and later printed using the VSE program, DOSVSDMP (refer to VSE/Advanced Functions Service Aids).

Formatted dumps for online task and system failures are written to the the CICS/VS dump data set, DFHDMPA or DFHDMPB. These dumps can be printed using the CICS/VS Dump Utility Program, DFHDUP (see CICS/VS Installation and Operations Guide).

Problem determination dumps are written into the dump file(s) on SYSDMP (refer to VSE/Advanced Functions Service Aids).

MODULE IDENTIFICATION

Most DL/I modules can be identified in storage or in a storage dump by information contained in the beginning of the module. Modules that are formatted by the DLZID macro contain DL/I identification information in sixteen bytes near the beginning of the module.

This information makes it easier to tell if a module is back level, has been modified to add a new function, has been modified after shipment, or has any PTF or APAR fixes included. Any alterations of the fields could result in delays in solving future problems.

The contents of these sixteen bytes are:

Offset Dec(Hex)		Length (Bytes)	Meaning
0	0	8	Module name
8	8	3	Last module change
8	8	1	Version
9	9	1	Release
10	A	1	Modification level
11	B	1	PTF number
12	C	1	Base code indicator
13	D	3	Current DL/I level
13	D	1	Version
14	E	1	Release
15	F	1	Modification level

Where:

Module name: The one to eight character name assigned to the module. This name is usually the same name that appears on the microfiche for this module.

Last module change: The version and release level that was current at the time this module was last changed by programming development. This generally means that new function and/or APAR fixes were included in this module. This is given in the form; version, release, modification level. (For example: 160 for Version 1, Release 6, Modification 0.)

PTF number: The latest PTF number that affects this module. This number is inserted by the change team when creating a PTF tape.

Base code indicator: This is a character constant of 'I' in all modules as shipped by IBM on the PID tape. For all modules assembled in the field, this character is 'U' to indicate that the module was updated in some way after shipment. The purpose of this is to alert anyone trouble shooting a problem to the fact that this module was altered in some way.

Current DL/I level: This is the version, release, and release modification level that is current at the time this module was prepared for shipment. This value is updated to the current level of each release regardless if the module was changed or not for this release.

SCD - THE KEY TO A DL/I DUMP

The SCD is the key to a DL/I storage dump. It is located at the beginning of the DL/I partition in batch. The first 96 bytes of the SCD contain the copyright information, thus making it easy to identify it. The SCD contains pointers to the major control blocks and entry point addresses for the primary DL/I modules (refer to Figure 1-10 on page 1-16). Note that when the DL/I system log function is inactive (UPSI bit 6 = 1), the entry point to the data base logger contains the address of a branch register (BR 14) instruction.

You can obtain the address of the SCD by issuing a GSCD (get SCD) call. The format of the GSCD call in Assembler Language is:

```
LA      1,GSCDPARM
CALL    ASMTDLI
```

where:

```
GSCDPARM  DC   A(GSCD)        FUNCTION CODE ADDRESS
PCBADDR   DC   A(pcb-addr)    PCB ADDRESS - NOT USED BUT MUST BE VAL
          DC   X'80'
          DC   AL3(IOAREA)    INPUT/OUTPUT AREA ADDRESS
GSCD      DC   CL4'GSCD'      FUNCTION CODE
IOAREA    DC   CL8''          INPUT/OUTPUT AREA
```

Upon return, the first four bytes in IOAREA contain the SCD address plus 96 (X'60'). The second four bytes contain the address of the partition specification table (PST).

The GSCD call may be issued in a PL/I program by using the standard DL/I call format with three parameters (excluding the count):

1. 'GSCD' as the function code,
2. a valid PCB, and
3. an I/O area of at least eight bytes.

Although the GSCD call may be issued in COBOL, the addresses returned in the I/O area would be unusable in a COBOL program. In this case, it is recommended that an Assembler Language subroutine be used to issue the GSCD call and process the returned addresses.

Note: The GSCD call is not supported by the High Level Programming Interface (HLPI).

Normally, no status codes are returned with the GSCD call. The only exception is when the call is issued by an online task through intersystem communication support while the task is to a PSB located on a remote system. In this case, a status code of 'AD' is returned in the PCB, indicating an invalid function code was specified.

ONLINE ASRA ABEND DEBUGGING

ASRA abends are caused by application oriented program checks that are intercepted by CICS/VS to prevent bringing down the network. When the CICS/VS program check handler gains control, checks are made for the type of CICS/VS task in control when the program check occurred. If the task which abended was running under other than a system TCA, the assumption is made that it is a user task and the ASRA is the result. Since DL/I runs as an extension of a user task, a program check anywhere in DL/I shows up as an ASRA abend.

Some useful service aids are:

- CICS/VS trace
- CICS/VS formatted dump

The minimum documentation needs are:

- CICS/VS supplied dump
- Application program listing
- SVA map
- Extended trace

A problem analysis approach to online ASRA ABEND debugging is:

1. Interrogate the dump for the PSW and the register contents at entry to abend. The PSW contains the exception code and instruction address of the exception.

2. If the instruction address is within the application (user task), proceed to step four. If the instruction address is outside the application but within the CICS/VS partition, proceed to step eight. If the address is outside the partition, this usually indicates that the module causing the problem is located in the SVA. Proceed to the next step for SVA discussion.

3. If the instruction address is located within the SVA, obtain a map of the SVA or a dump of the SVA. Locate the closest module entry point that precedes the program check address. Subtract the entry point address from the exception address to obtain the displacement. At this point, the microfiche listings of the module identified should be interrogated to determine the instruction flow and control blocks involved in the failure. Proceed to step ten.

4. The dump provided by CICS/VS contains a section which is the application. Locate the failing instruction and cross-reference to the assembly listing of the application. Determine what the instruction flow and control blocks involved were at the failure.

5. The failure could be due to user failure to follow specified CICS/VS or DL/I coding conventions. Verify that all calls and parameter lists, work areas, and CICS/VS facilities are correct as per the application programmer guides for both products.

6. Layout the CICS/VS trace entries for this task from the dump and establish the sequences of scheduling, DL/I calls for data, and termination. This can aid in determining if the failure only occurs with a specific call sequence.

7. If it is determined that the exception was caused by bad DL/I pointers or control blocks, the control blocks formatted by the dump program may aid in verification of the diagnosis.

8. If the failing instruction address is within the CICS/VS partition but not within the application, the program check is most probably within a DL/I system module such as buffer

handler, online nucleus module, or the retrieve module. Determining which module the exception occurred in may be achieved by the following procedure:

- Using the CICS/VS dump, locate the SCD. Using the logic manual or a listing of the SCD DSECT, locate the fields which point to the DL/I system module's entry points.
- Locate the entry point closest to, and preceding, the program check address. Using the SCD DSECT locate the corresponding field. The comments indicate the name of the module pointed to by this field (i.e., retrieve, buffer handler, etc.).
- Now subtract the entry point address from the program check address to obtain the displacement within the module of the failing instruction.

9. Using the CICS/VS trace, establish the sequence of schedule, data, and termination calls issued. What was the current call at the time of the failure?

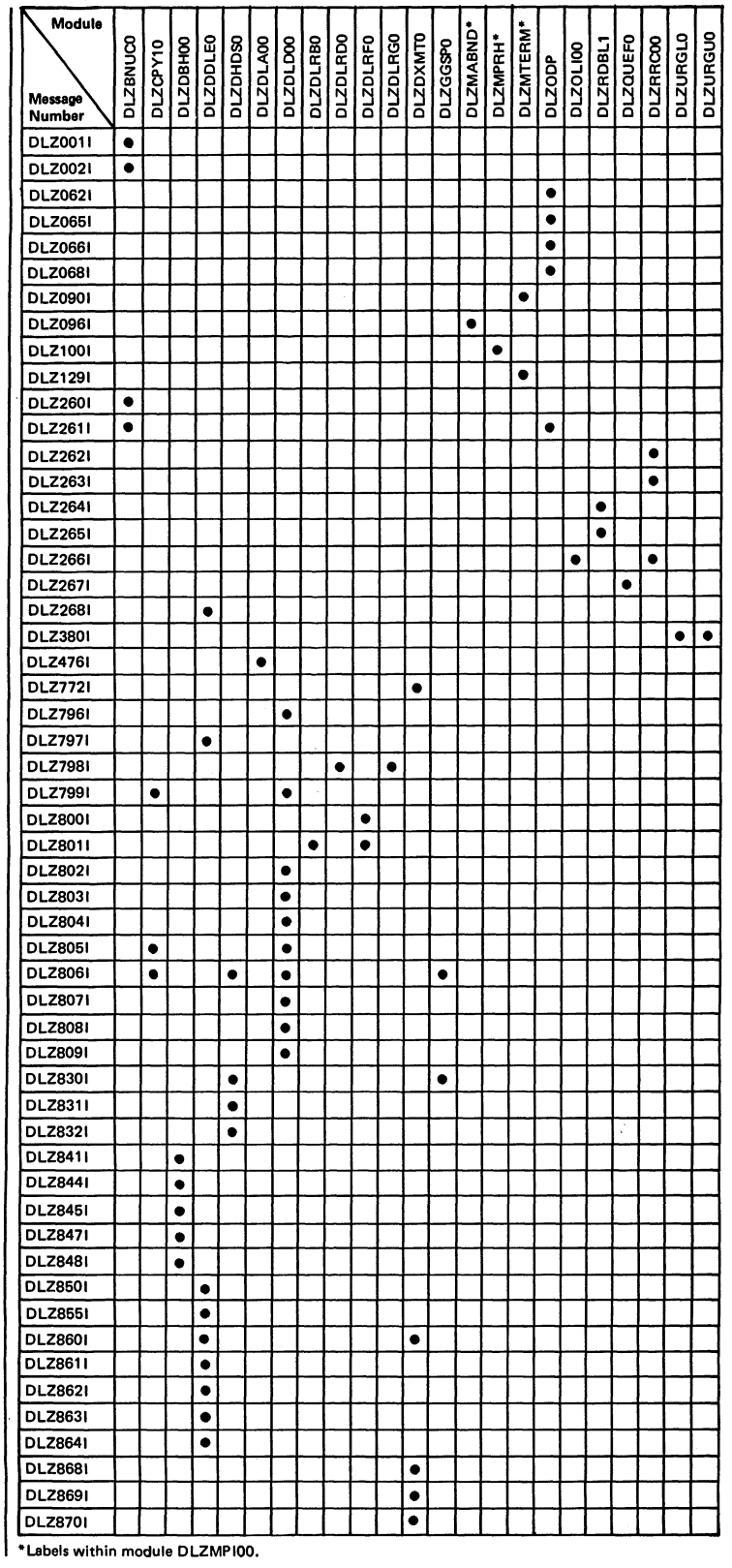
10. At this point the failing module and current call have been determined. Make a RETAIN search at this time.

INTERPRETING A DUMP AFTER AN ABEND MESSAGE

Figure 3-1 on page 3-8 summarizes the ABEND messages and the modules that originate the messages. Additional information that can aid in determining the error that caused most of these messages is located in "Additional Diagnostic Aids".

Note: If you expect an abend, be sure that you have the correct UPSI bit setting: bit 5 = 0, bit 7 = 0 (for batch).

The DL/I DOS/VS Messages and Codes manual explains the cause(s) for the message and suggests action for correcting the message condition. It should be referenced when any message is being interrogated. If a message appears in Figure 3-1 on page 3-8 but not in the following "Additional Diagnostic Aids" section, DL/I DOS/VS Messages and Codes contains all the appropriate information and helps.



Message Number \ Module	DLZBNUC0	DLZCPY10	DLZDBH00	DLZDDLE0	DLZDHDS0	DLZDLA00	DLZDLD00	DLZDLRB0	DLZDLRD0	DLZDLRF0	DLZDLRG0	DLZDXMT0	DLZGGSP0	DLZMABND*	DLZMPRH*	DLZMTERM*	DLZODP	DLZOLI00	DLZRDBL1	DLZQUEF0	DLZRRC00	DLZURGL0	DLZURGU0
DLZ0011	●																						
DLZ0021	●																						
DLZ0621																	●						
DLZ0651																	●						
DLZ0661																	●						
DLZ0681																	●						
DLZ0901																●							
DLZ0961														●									
DLZ1001															●								
DLZ1291																●							
DLZ2601	●																						
DLZ2611	●																●						
DLZ2621																					●		
DLZ2631																					●		
DLZ2641																			●				
DLZ2651																			●				
DLZ2661																		●			●		
DLZ2671																				●			
DLZ2681				●																			
DLZ3801																						●	●
DLZ4761						●																	
DLZ7721												●											
DLZ7961							●																
DLZ7971				●																			
DLZ7981									●		●												
DLZ7991		●					●																
DLZ8001										●													
DLZ8011								●		●													
DLZ8021							●																
DLZ8031							●																
DLZ8041							●																
DLZ8051		●					●																
DLZ8061		●			●		●						●										
DLZ8071							●																
DLZ8081							●																
DLZ8091							●																
DLZ8301					●								●										
DLZ8311					●																		
DLZ8321					●																		
DLZ8411			●																				
DLZ8441			●																				
DLZ8451			●																				
DLZ8471			●																				
DLZ8481			●																				
DLZ8501				●																			
DLZ8551				●																			
DLZ8601				●								●											
DLZ8611				●																			
DLZ8621				●																			
DLZ8631				●																			
DLZ8641				●																			
DLZ8681												●											
DLZ8691												●											
DLZ8701												●											

*Labels within module DLZMPI00.

Figure 3-1. Abend Message/Module Summary

ADDITIONAL DIAGNOSTIC AIDS

This section identifies the DL/I abend messages which cause storage dumps and contains additional diagnostic information that may be helpful in locating the problem.

DLZ002I - DLZBNUC0

If no abend message preceded the DLZ001I and DLZ002I messages, the abend routine gets control from VSE system supervisor.

See the message descriptions for abend messages on the following pages and in DL/I DOS/VS Messages And Codes for suggestions. Any reference to the PST fields mentioned should be located in the abend save area rather than in the PST itself.

The pseudo abend save area is the same as the PC save area. Pseudo abend means no system detected abend occurred, but DL/I detected an error which caused the program to terminate. The abend handling routine is the same for both types of abend, "System Abends" and "DL/I Abends".

DLZ096I - DLZMABND

The register save areas (PSW followed by registers 0-15) are immediately preceded in the dump of module DLZMPI00 by the eye-catcher, 'AB SAVE' or 'PC SAVE'. The one-byte reason code for AB STXIT entry is preceded by 'AB REASON CODE'.

It is possible for an addressing program check or a VSE invalid address abend to occur when the batch partition tries to access data in the online partition after the online partition terminated and its address space was redefined (the addresses used by batch are no longer valid). This condition cannot be detected by DL/I. The following is a list of labels in DLZMPI00 at or after which the online partition is accessed:

```
DLZMINIT: XWAIT1
          XECBCHK2
DLZMPRH:  XWAIT0
          PRHERR3
          ERR0
```

Termination of the online partition has no impact on the validity of any of any data moved into the batch partition because after each access a check is issued to ensure that online is still active.

DLZ100I - DLZMPRH

The error is detected after label XWAIT0 because bit MPCERR in the MPCFLAG in the partition table entry is on. The error flag is set by DLZODP01 (task termination) after label BPCLNUP on abnormal termination of the BPC.

The message is issued at label PRHERR1.

A program check probably occurred in either the BPC or a DL/I action module. See "Online ASRA ABEND Debugging" for additional information.

Note: A dump is produced when you receive a code of 02 with this message. No dump is produced when you receive a code of 01.

DLZ261I - DLZBNUC0, DLZODP01

This message can be caused by several user generated errors:

- Failing to code the language parameter of the PSB which causes DL/I initialization failures such as insufficient storage messages.
- In Assembler Language, the cause can usually be traced to the user having set up a bad pointer in the call list. The pointer should point to a valid DL/I call function (GN, GU, ISRT, REPL, etc.) or to an area within the range of the user's application for IOAREA and SSA, or the pointer to the PCB may be invalid. If the list is invalid, the application program may have incorrectly established addressing to the field in error.
- In high level languages, a common error is that the user does not link edit the DLZBPJRA (COBOL) or IBMPJRA (PL/I) module correctly. The user either forgets to include it or fails to add the entry statement.

The last fullword in the list may not have an X'80' in the high-order byte if implicit length is used.

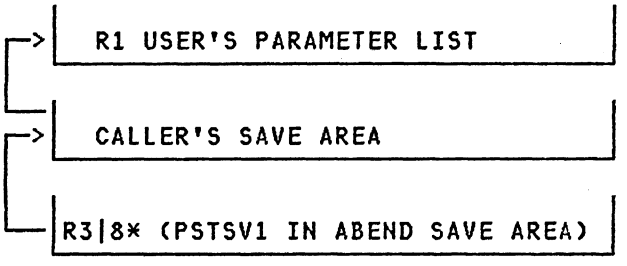
The validity of the parameters should be checked.

Incorrect control blocks may have been used.

The SSA should be checked for incorrect parameters.

The following is true for batch operations:

- If message DLZ001I is printed with this message (indicating batch operation), register 3 points to the caller's save area and register 1 in the caller's save area points to the parameter list (see Figure 3-2).
- If message DLZ001I is not issued with this message (indicating online operation), register 8 points to the caller's save area and register 1 in the caller's save area points to the parameter list (see Figure 3-2).



* R3 for batch operation,
R8 for online operation.

Figure 3-2. Finding the User's Parameter List

DLZ262I - DLZRRC00

Go to module DLZRRC00. Start at label GREATPRO and check the pointers used. This routine detects the error.

The error is indicated in routine PROCSEC.

The pseudo abend save area is not loaded by DLZRRC00, but the DUMP macro is used after writing the message. The message writer saves and reloads registers 14-12.

DLZ263I - DLZRRC00

The error is indicated in routine PROCSEC.

The pseudo abend save area is not loaded by DLZRRC00, but the DUMP macro is used after writing the message.

The message writer saves and reloads registers 14-12.

Locate the user's parameter list (see Figure 3-2 on page 3-10) then locate the PCB address in the parameter list.

Locate the DMBs (see Figure 3-3).

Ensure that the DMB and PSB are at compatible levels. (This can be done by redoing an ACBGEN.)

Consider an application control blocks utility error.

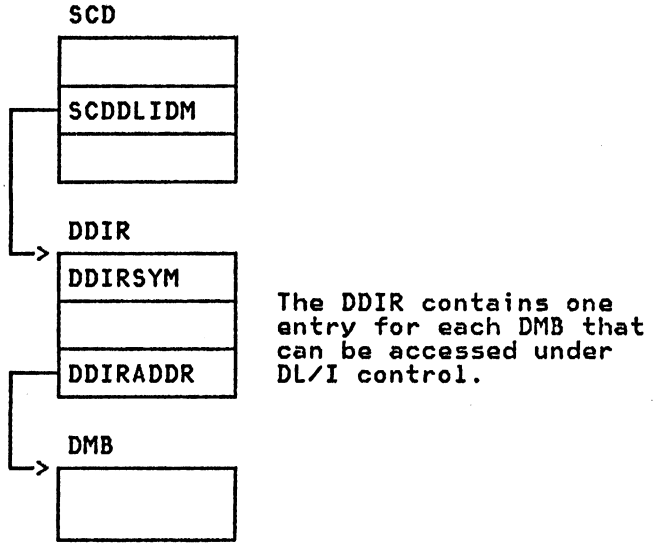


Figure 3-3. Locating DMBs

DLZ264I - DLZRDBL1

Possible error conditions are:

- Contradictions between application program and application control blocks.
- Erroneous control information in data bases.
- Destroyed DL/I control blocks or buffers.
- Undetected program errors in DL/I.

DLZ265I - DLZRDBL1

Possible error conditions are:

- Contradictions between application program and application control blocks.
- Erroneous control information in data bases.
- Destroyed DL/I control blocks or buffers.
- Undetected program errors in DL/I.

DLZ266I - DLZRRC00, DLZOLI00

Possible error conditions are:

- Contradictions between application program and application control blocks.
- Erroneous control information in data bases.
- Destroyed DL/I control blocks or buffers.
- Undetected program errors in DL/I.

DLZ267I - DLZQUEF0

Possible error conditions are:

- Contradictions between application program and application control blocks.
- Erroneous control information in data bases.
- Destroyed DL/I control blocks or buffers.
- Undetected program errors in DL/I.
- Back level SCD resulting in a call of the wrong module.
- Error in the calling routine resulting in the wrong function code.

DLZ268I - DLZDDLE0

The error is detected below label CKPTRS.

The message is issued at label DELERR.

The contents of the registers in the pseudo abend save area are:

```
Register  1 = Address of PST.
Register  4 = Address of LSDB.
Register  5 = Address of PSDB.
Register  6 = Address of JCB.
Register 12 = Entry point to DLZDDLE0.
```

It is important to determine the reason for this condition before continuing processing of the data base because continued processing could result in more damage to the data base.

DLZ380I - DLZURGL0, DLZURGU0

For HD Unload, the status code is checked after label CKRETCDE and the message is issued at label NOSEGFND.

For HD Reload, the message is issued under two conditions:

- Nonblank status code on GU called for root at label CALLIT.
- Status code other than blank, GA, GB, or GK at GN call when sequentially reading data base to determine the last segment written at label GNCALL.

Ensure that the correct unload tape is provided as input to the restart.

For reload restart, ensure that the entered checkpoint number is not beyond the last checkpoint processed in the previous reload run.

DLZ476I - DLZDLA00

The error is detected at label TESTPCB0 in routine TESTPCB or in routine VALIDCK3.

The message is issued at label ERR0 or ERR22.

Ensure that the language specified in the PSB is the same as that used in the application program.

Locate the PCB address list. Use the contents of register 1 when it was passed to the user on entry to the application program. Get the PCB address(es) from the PCB address list.

Check the addresses in the list. X'80' should be in the left-most byte of the last entry (this entry should not be zero).

Locate the user's parameter list (see Figure 3-2 on page 3-10). X'80' should not be in the left most byte of the function address. The pseudo abend save area is loaded.

Return is taken to caller.

DLZ772I - DLZDXMT0

The error is detected after label LKEY1C.

The message is issued at label LABND772.

Check the pointers and fields used in the routine starting with label LGETKEY for correct relationship.

The pseudo abend save area is loaded by the LABEND routine.

Return is taken to the caller.

For batch processing refer to "DL/I Partition and Control Block Relationship" in the DL/I DOS/VS Logic Manual, Volume 1 to find the label fields listed above. For online processing refer to "DL/I Online With CICS/VS" to find the label fields listed above.

DLZ796I - DLZDLD00

The error is detected at label:

DOWN8	- segment code not the one expected, parent pointer in child does not point to parent.
FNDPRPT	- root segment not on root anchor point chain, dependent segment not on physical twin chain.
UPDPT	- segment code not the one expected.

The message is issued at label ABEND796.

Register 1 = Address of PST.
Register 2 = Address of pseudo abend save area.
Register 5 = Address of DMBPSDB.
Register 8 = Address of segment.

Use the linkage registers in the pseudo abend save area to locate the error detection point which indicates the exact error condition. Check the buffer and control blocks involved to determine the cause of the error.

DLZ797I - DLZDDLE0

The error is detected:

- after labels

  POSCOK
  CONTIN66
  PIBYPSCC

- in routine UPPREFIX after labels

  UPPREFIX
  UPPRE2
  UPPRE3
  UPPRE4

The message is issued at label ABEND797.

The contents of the registers in the pseudo-abend save area are:

Register 1 = Address of PST.
Register 4 = Address of SDB.
Register 6 = Address of JCB.
Register 12 = Entry point of DLZDDLE0.

If the segment code from the segment just accessed does not match the expected segment code in the SDB:

- Register 10 = address of the bad segment code if the error is detected after labels CONTIN66 or PIBYPSCC or in routine UPPREFIX.

  or

- Register 14 = address of the bad segment code if the error is detected after label POSCOK.

If sequence errors are detected in routine UPPREFIX:

- Register 10 = Address of sequence field in buffer area.
- Register 14 = Address of sequence field in user area.

It is important to stop processing the invalid data base until the problem is found and fixed. Continued processing can result in more damage to the data base.

DLZ798I - DLZDLRD0, DLZDLRG0

The error is detected in DLZDLRD0 subroutine DLZRETK0 at label RETHTST or in DLZDLRG0 subroutine DLZPOST0 at label POSTABND.

To locate the segment which contains the invalid RBA:

- If the error is detected in subroutine DLZRETK0, the invalid RBA is in the field SDBPOSN (X'34') pointed to by register 5. If the field SDBPOSC (X'30') is zero, the RBA is in the prefix of the parent (the address of the SDB is in field SDBTARG (X'21') and the address of the RBA of the parent is in field SDBPOSC).

  Note: If R5 points to the root segment, the bad RBA is in the index for a HIDAM data base or the RAP for a HDAM data base. If SDBPOSC is not zero, the RBA is the physical twin pointer in the current segment (the address of the SDB is in register 5).

- If the error is detected in subroutine DLZPOST0, the invalid RBA is in the field SDBPOSN (X'34') pointed to by register 5. If the field SDBPOSC (X'30') is zero, the RBA is in the prefix of the parent (the address of the SDB is in field SDBPARA (X'18') and the address of the RBA of the parent is in field SDBPOSC).

  Note: If R5 points to the root segment, the bad RBA is in the index for a HIDAM data base or the RAP for a HDAM data

base. If SDBPOSC is not zero, the RBA is the physical twin pointer in the current segment (the address of the SDB is in register 5).

The cause of the bad RBA can be one of many reasons. Some of the more common causes are as follows:

- The usual error here is that the user has changed the DBD and has added or deleted pointers without doing a unload/reload.
- DL/I action modules caused an overlay of the prefix field. Since there are many millions of combinations of physical and logical data bases, there are many combinations of paths through the DL/I modules, routines, and subroutines. Occasionally one of these combinations causes an incorrect execution of the logic which could result in bad data base pointers.
- An unrecoverable system error (such as a power failure) occurred, and the installation did not perform recovery procedures correctly. An example of this would be an online system doing updates/additions to a data base when a system failure condition occurred. The installation should perform a data base back-out using the log tape. This removes the effects of unfinished updates or additions.
- Multipartition concurrent access of a data base. The use of two DL/I partitions to access/update a data base without using MPS can cause prefix and control field problems. The data base must be accessed from only one partition at a time. The use of VSAM share option 3 or 4 is a major cause of this message.
- The VSAM component may be at fault. A physical or logical VSAM error may have been incorrectly handled by VSAM. Since DL/I is simply a VSAM application program, care should be taken that VSAM is performing correctly (i.e. passing status information, correctly interpreting a data base condition).
- Failure to supply a GOBACK (COBOL) or RETURN (PL/I) statement can cause this message to be issued.

The contents of the registers in the pseudo abend save area are:

R3 = Address of JCB.
R4 = Address of level table.
R5 = Address of SDB.
R6 = Address of segment in DL/I buffer.
R7 = Address of PST.
R8 = Address of DSG.
R12 = Address of subroutine which detected the error.

Field JDBPRESF (X'3F') contains the code of the failing call.

Field JCBSTOR5 (X'54') contains the RBA of the segment in error.

It is important to stop processing of the invalid data base until the problem is found and fixed. Continued processing can result in more damage to the data base.

DLZ799I - DLZDLD00, DLZCPY10

The error is detected in routine COMPRESS.

The message is issued at label ABEND799.

The pseudo abend save area is loaded by routine ABENDCOM.

R5 = Address of DMBPSDB.
R6 = New segment length.
R14 = Segment source address.
R15 = Segment destination address.

Check DMBPSDB field DMBSGMX which contains the maximum length against R6.

An error may exist in user's compression routine.

The maximum length in the DBD generation for this segment may be too small.

DLZ800I - DLZDLRF0

The error is detected in routine COMPRTN.

The message is issued at label PSABND.

The pseudo abend save area is loaded by routine PSABND.

Return is taken to analyzer.

R8 = DMBSGMX maximum length.

R3 + X'6A' 2 bytes current length.

R3 + X'74' 4 bytes segment data address.

Check register 8 which contains the maximum length against R3+X'6A' which points to the current segment length.

An error may exist in user's compression routine.

The maximum length in the DBD generation for this segment may be too small.

DLZ801I - DLZDLRB0, DLZDLRF0

This message identifies a data base related error. DL/I issues this abend when an invalid RBA is detected (either DL/I or VSAM).

A possible cause is the use of the processing option PROCOPT=GO(P) with limited data base sharing.

The error is detected on return from buffer handler with R15 not zero.

The message is issued in DLZDLRB0 subroutine DLZBH at label ISMINERA or in the DLZDLRF0 subroutine DLZALTS at label XMINERA.

The RBA is usually contained within the segment prefix. The segment prefix contains RBA values for the various pointers which were defined in the DBD by the user. An invalid RBA is one which points beyond the end of the "High Used" portion of the data base.

DLZDLR00 received a bad return code from a call to the buffer handler.

Refer to the simplified flow of the two subroutines to get the logic which decides to abend (see Figure 3-4 on page 3-18 for subroutine DLZBH, or Figure 3-5 on page 3-19 for subroutine DLZALTS).

The pseudo abend save area is loaded by routine ISMINERA or XMINERA.

The contents of the registers in the pseudo abend save area are:

R2 = DBPCB (user's PCB).
R3 = JCB prefix
R4 = Level table address for segment.
R5 = SDB for segment attempting to access.
R7 = PST
R9 = RBA in error.

R14 = points to the subroutine which detects the error.

Locate the PST fields in the pseudo abend save area:

PSTFNCTN - callers function to buffer handler.
PSTRTCDE - return code from buffer handler.
PSTBYTNM - contains the RBA that was passed to VSAM and caused the incorrect return code. Bit 0 of byte 0 indicates the following for HISAM:

0 = KSDS
1 = ESDS

Locate the VSAM RPL (see Figure 3-6 on page 3-20). Check for fields RPLFDBK1 and RPLFDBK2.

Note: The fields are valid only if the files are not closed. Use DL/I TRACE VSAMINTF.

Locate the SDB and LEVEL table entries for the segment processed. The following labels contain:

Label	Disp	Func	Contents
SDBPOSP	X'2C'	HD	RBA of previous twin, 0 if none
		HS	0
SDBPOSC	X'30'	HD	RBA of current segment
		HISAM	RBA of current segment
		HSAM	Number of current LRECL
SDBPOSN	X'34'	HD	RBA of next twin, 0 if none
		HS	Offset to segment in LRECL
LEVTTR	X'04'	HD	RBA of the data base position satisfying the call.
LEVSEGOF	X'02'	HD	0, or if
		HS	Offset to the data base segment within the LRECL satisfying the call.

Note: LEVTTR and LEVSEGOF should contain the position of the previous call or the current call. SDBPOSP, SDBPOSC and SDBPOSN always reflect the newest (may be internal temporary) status whether or not the call is satisfied by the corresponding segment.

Locate the JCBTRACE field (trace of last seven calls) in the JCB, see "JCB Trace" on page 2-15.

Locate the buffers. The following labels contain:

Label	Disp	Func	Contents
JCBSTOR4	X'50'		Main storage address of the last segment obtained.
JCBSTOR5	X'54'	HISAM	RBA of the last LRECL requested.
		HD	RBA of the last segment requested.
		HSAM	RBA of the last record number requested.

The most common user error causing this abend is that the user changes his physical description of the data base (DMB generation) without reloading the data base.

Another cause for this abend could be that an unrecoverable system failure occurred and the user failed to run the necessary recovery utilities or ran an incorrect recovery. This incorrect recovery could be due to log tapes missing or image copy not correct.

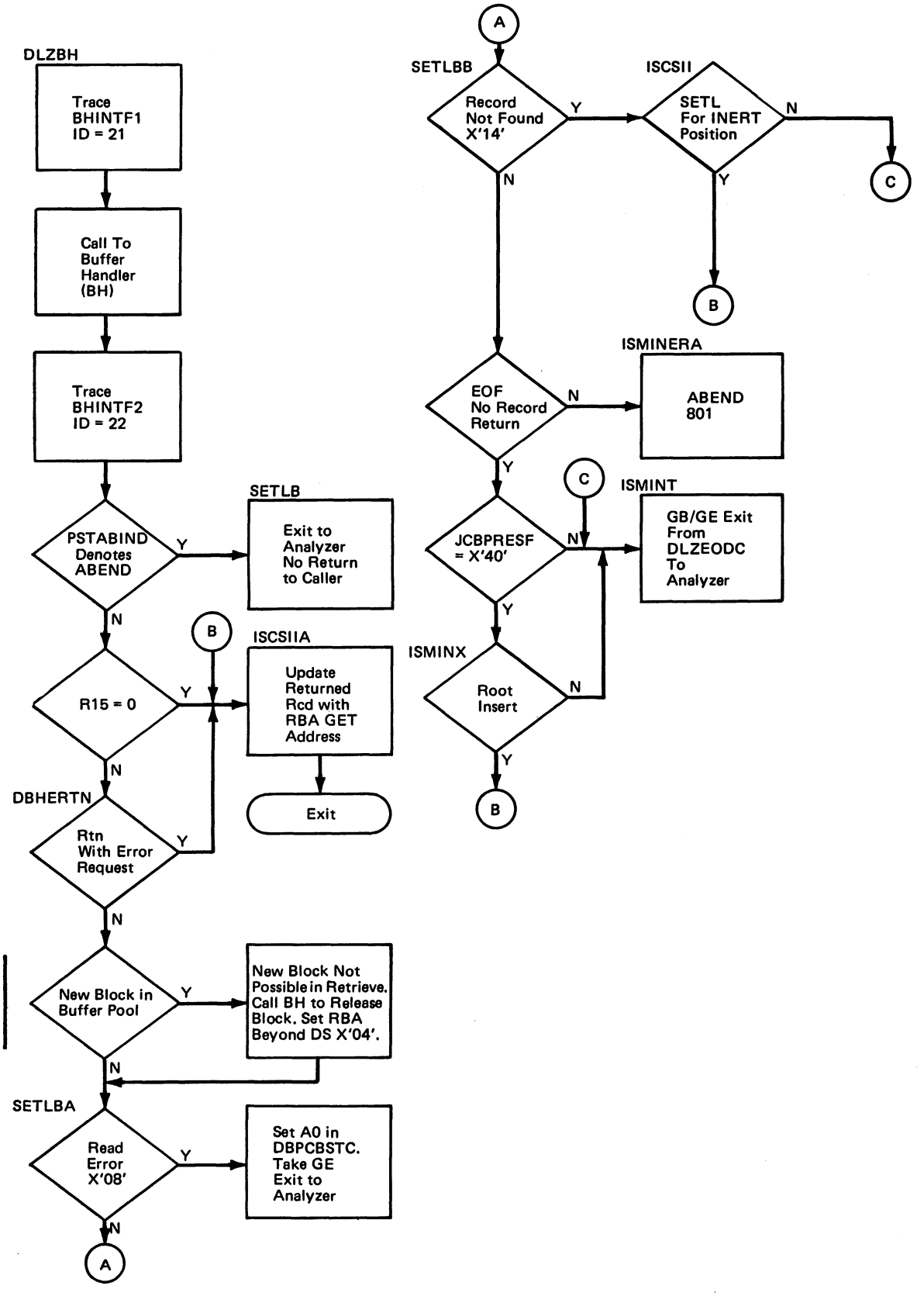


Figure 3-4. Simplified Return Code Checking for Subroutine DLZBH

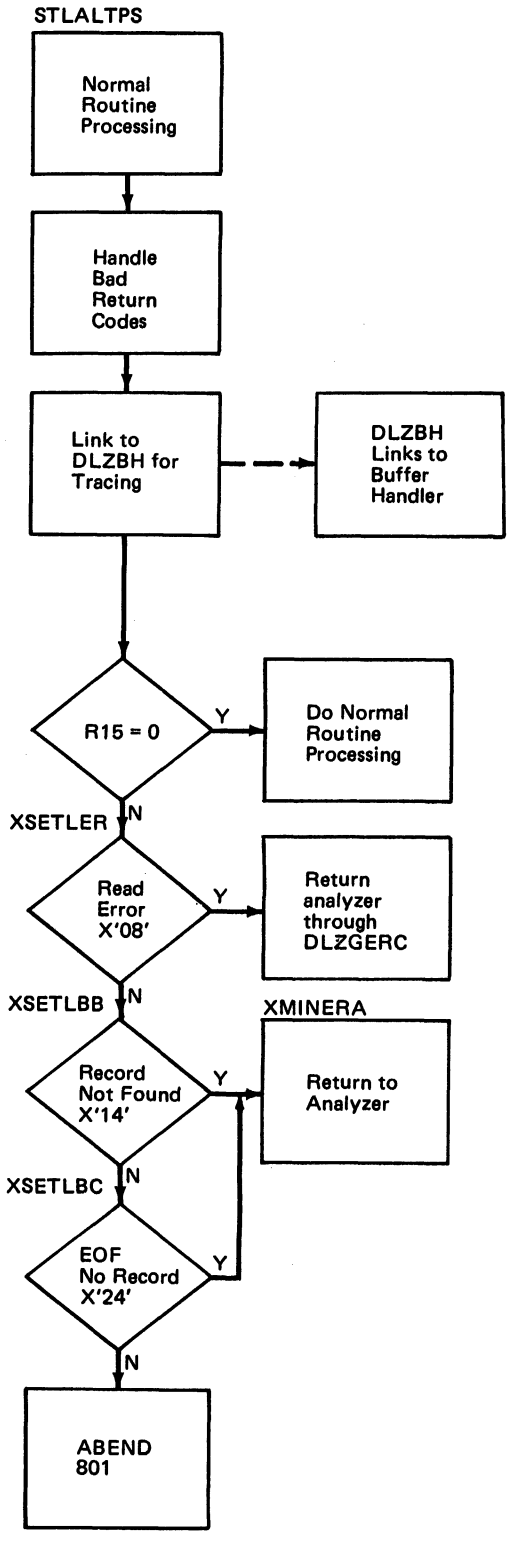


Figure 3-5. Simplified Return Code Checking for Subroutine DLZALTS

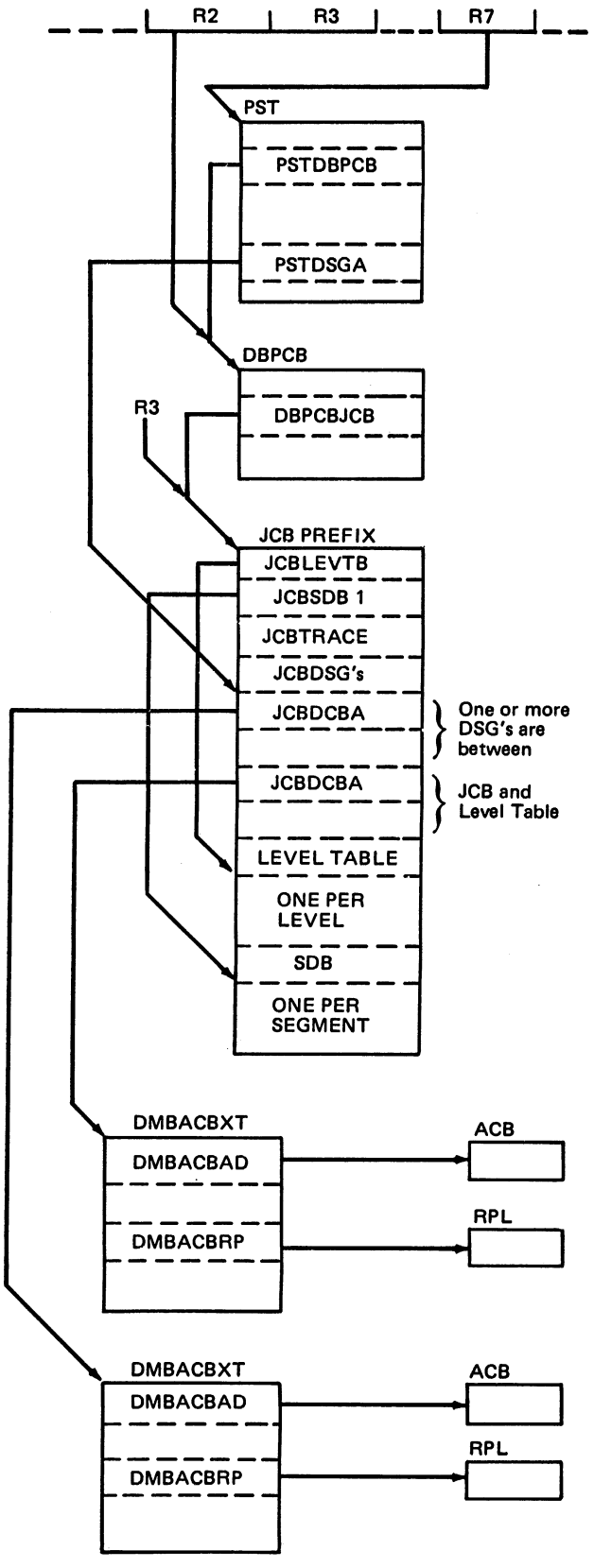


Figure 3-6. Locating ACBs and RPLs

The cause of the bad RBA can be one of many reasons. Some of the more common causes are as follows:

- The user changed the DBD between loads/reloads. The usual error here is that the user added or deleted pointers from his DBD without doing a load/reload.
- DL/I action modules caused an overlay of the prefix field. Since there are many millions of combinations of physical and logical data bases, there are many combinations of paths through the DL/I modules, routines, and subroutines. Occasionally one of these combinations causes an incorrect execution of the logic which could result in bad data base pointers.
- An unrecoverable system error (such as a power failure) occurred, and the installation did not perform recovery procedures correctly. An example of this would be an online system doing updates/additions to a data base when a system failure condition occurred. The installation should perform a data base back-out using the log tape. This removes the effects of unfinished updates or additions.
- Multipartition concurrent access of a data base. The use of two DL/I partitions to access/update a data base without using MPS can cause prefix and control field problems. The data base must be accessed from only one partition at a time. The use of VSAM share option 3 or 4 is a major cause of this message.
- The VSAM component may be at fault. A physical or logical VSAM error may have been incorrectly handled by VSAM. Since DL/I is simply a VSAM application program, care should be taken that VSAM is performing correctly (i.e. passing status information, correctly interpreting a data base condition).
- Failure to supply a GOBACK (COBOL) or RETURN (PL/I) statement can cause this message to be issued.

Use DL/I Test Programs for reproducing the error. Use the DL/I trace facility to trace the VSAM interface (VSAMINTF) and buffer handler interface (BHINTF).

All necessary information can be found in these trace outputs.

Use LISTCAT to list the catalog entry for the cluster.

Check the RBA indicated as invalid against the high-used RBA in the data component.

Find the segment whose pointer has the wrong RBA.

Print the part of the data base containing the error.

DLZ802I - DLZDLD00

The error is detected at label REPVLS81 after return from DLZDHDS0.

The message is issued at label ERR802.

The pseudo abend save area is loaded by routine ABENDCOM.

DLZ803I - DLZDLD00

The error is detected after label:

NWDMB6A after a link to routine GETPTR.
TOLTPTR after a link to routine GETPTR.
TOLTPTR1 after a link to routine GETBYTE.
TOLPCTR1 when the counter in the prefix of the logical parent was negative.

The message is issued at label ABEND803.

The pseudo abend save area is loaded by routine ABENDC.

R1 PST address.
R5 Address DMBPSDB (segment description of segment just worked on).

Use the contents of the registers in the pseudo abend save area, to find the point which detects the error, then check the fields used.

DLZ804I - DLZDLD00

The error is detected in routine:

FREEWKAS
NWDMB34C
OPENDB

The message is issued at label ABEND804.

The pseudo abend save area is loaded by routine ABENDCOM.

Locate the buffer pool and buffers (see Figure 3-7). Verify that all buffers (2 to 32 possible) in the subpool are full. If not, or if only one data base is used, the error is probably in the buffer handler. If more than one data base is used, the job should be rerun with an additional subpool specified in the DL/I parameter statement.

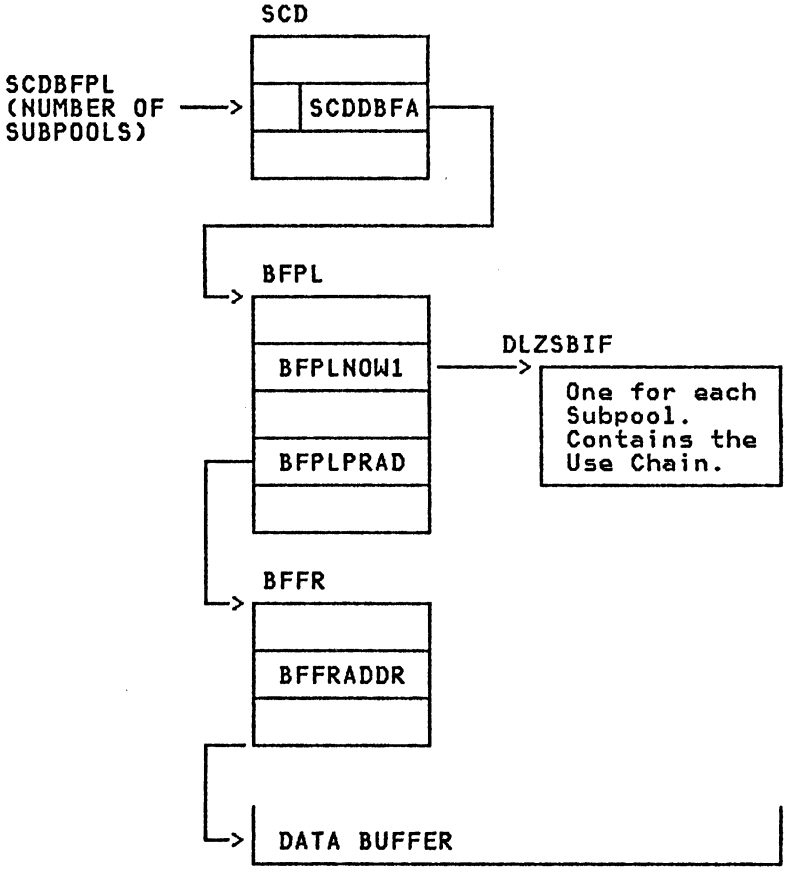


Figure 3-7. Locating the Buffer Pool and Buffers

DLZ806I - DLZDLD00, DLZCPY10, DLZDHDS0

The error is detected on non-zero return codes in register 15 on a return from the buffer handler.

The message is issued at label ABEND806.

The pseudo abend save area is loaded by routine ABENDCOM.

R1 = PST address.
R2 = The pseudo abend save area address.
R5 = DMBPSDB.

Locate the buffer pool and buffers (see Figure 3-7 on page 3-22). Verify that all buffers (2 to 32 possible) in the subpool are full. If not, or if only one data base is used, the error is probably in the buffer handler. If more than one data base is used, the job should be rerun with an additional subpool specified in the DL/I parameter statement.

DLZ807I - DLZDLD00

The error is detected behind label:

NWDMB4X3
NWDMB5A4
UPDPTIP

The message is issued at label ABEND807.

The pseudo abend save area is loaded by routine ABENDCOM.

Locate the buffer pool and buffers (see Figure 3-7 on page 3-22). Verify that all buffers (2 to 32 possible) in the subpool are full. If not, or if only one data base is used, the error is probably in the buffer handler. If more than one data base is used, the job should be rerun with an additional subpool specified in the DL/I parameter statement.

DLZ808I - DLZDLD00

An invalid control block exists.

The error is detected on scanning the data base and control blocks for correct relationship.

Tests are made against the following fields:

DMBORG in the DMB prefix.
DMBSCDE in the DMBSEC.
DMBPTR in the DMBPSDB.
DMBSECTB in the DMB.
DMBLENTB in the DMB.
DMBPSC in the DMB.

The error is detected in routines or at the following labels:

DELTHD
CALCPAR
FNLPSEC1
FLPSLC
TOLP1A1
TOLP2
TOLPCTR
DELT14
DELT15
DELT16

The message is issued at label ABEND808.

The pseudo abend save area is loaded by routine ABENDCOM.

Return is taken to caller with:

R1 = PST
R3 = SDB
R4 = DMB prefix
R5 = DMBPSDB current
R6 = DMBSEC

Check the DMB to determine if the offsets in fields DMBLENTB and DMBSECTB are correct.

Check the relationship between the blocks used in other routines.

Check the blocks against the generated DBD.

DLZ830I - DLZDHDS0

This message is for HD, HDAM, or HIDAM.

The error is detected in routine SRCHBLK at label NRMLRUN.

The message is issued after a return from routine SRCHBLK.

The search for free space in a block (CI) detects that the block cannot contain the largest segment specified in the DBD generation.

The block pointed to by the bit map was not large enough.

Note: In the bit map we have a bit position for every block in the ESDS part of a DB. When the bit value is one, it indicates that the associated block has sufficient space remaining to store the longest segment type defined in this data base.

The pseudo abend save area is loaded by the error indicating module.

An exit is taken directly to the DLZABEND abend routine.

Check the DBD generation.

DLZ831I - DLZDHDS0

Space management cannot continue because it does not have the track and cylinder capacities to calculate the number of control intervals per track and cylinder to be inserted into the DMB for space management's use. DBDGEN sets an FBA indicator in the DBD which signals DL/I space management to issue the GETVCE macro the first time it is entered to obtain the device characteristics.

Ensure that the data base specified resides on an FBA device and rerun the job, or regenerate and recatalog the DBDGEN specifying the proper device and rerun the job.

DLZ832I - DLZDHDS0

The error is detected in routine SRCHBLK at label CTNUSRCH.

Check to ensure that the control interval being searched is not control interval zero.

Look at the control interval in the buffer to determine the invalid FSE. Register2 points to the buffer and Register 3 contains the offset to the valid FSE.

DLZ841I - DLZDBH00

This error is issued by module DLZDCI00.

This message is also issued for errors (non-zero return codes) from GETVIS, SHOWCAT, and GETVCE.

The error is detected in the FINDBUF routine.

The message is issued at label ALPEWRER.

The pseudo abend save area is loaded.

If this message appears without another message pointing to an I/O error, check the number of subpools specified in the parameter statement.

Locate the buffer prefixes in DLZBFFR (32 per subpool) and check the BFFRSW field for permanent write errors. In this case, other messages should have occurred before this abend. If online, check the master terminal destination.

Locate the buffer pool and buffers (see Figure 3-7 on page 3-22). Verify that all buffers (2-32 possible) in the subpool are full. If not, or if only one data base is used, the error is probably in the buffer handler. If more than one data base is used, the job be should rerun with an additional subpool specified in the DL/I parameter statement.

DLZ844I - DLZDBH00

The error is detected in routine DETIOERR based on a non-zero return code for VSAM.

The message is issued in the routine DETIOERR.

The pseudo abend save area is loaded.

Another DL/I message should have occurred before this one. If online, check master terminal destination.

DLZ845I - DLZDBH00

The error detected was in routine LOCATE or SCANFSE. It is usually caused by a bad free space element (FSE).

The message is issued at label MOV845.

The pseudo abend save area is loaded.

The following registers contain:

R5 - address of the buffer
R6 - address of the FSE
R10 - address of the BFFR

This is an unexpected condition. Locate the buffer prefixes DLZBFFR (32 per subpool), field BFFRCIID, and check against the PST field PSTBLKNM. Something probably was destroyed.

Locate the buffer pool and buffers (see Figure 3-7 on page 3-22).

Locate field BFFRCIRB (pointed to by register 10 since it is the first field in BFFR) and compare it with field PSTBLKNM. If they are not equal, then the buffer indicated by PSTBLKMN can not be found. If they are equal, go to address pointed to by Register 10 plus X'0C'. This will the buffer holding the control interval from the data base. Follow the FSE chain (see Figure 1-27 on page 1-37) and see if you have an FSE whose available length field is larger than the CI size.

DLZ847I - DLZDBH00

An error is detected in the TESTWRT routine.

The message is issued below label TESTWRT in the LOCATE routine.

The pseudo abend save area is loaded.

For batch processing refer to "DL/I Partition and Control Block Relationship" in the DL/I DOS/VS Logic Manual, Volume 1, to find the fields used.

For online processing refer to "DL/I Online With CICS/VS" on page 1-9 to find the fields used.

DLZ848I - DLZDBH00

The error was detected in routine GBSPC behind label WRONSIZW.

Locate the PST field PSTBYTNM and the subpool information table(s) field SUBBSIZ. Compare the fields. Note that the value in PSTBYTNM is decreased by one and then shifted right by nine.

The pseudo abend save area is loaded.

For batch processing refer to "DL/I Partition and Control Block Relationship" in the DL/I DOS/VS Logic Manual, Volume 1, to find the fields used.

For online processing refer to "DL/I Online With CICS/VS" on page 1-9 to find the fields used.

DLZ850I - DLZDDLE0

The error indicated and detected was in routine DFSDHIL0.

The pseudo abend save area is loaded.

Locate the JCB using the pseudo abend save area, register 6, and check the field JCBPRESF for a correct function code.

DLZ855I - DLZDDLE0

An error was detected in routine WRITEOLD behind label KNNDONEZ.

The message is issued at label CATERROR in the WRITEOLD routine.

The pseudo abend save area is loaded.

DLZ860I - DLZDDLE0, DLZDXMT0

An error is issued at label LABND860 in the DLZDXMT0 module or an error was detected behind label GOTOBUFF in the DLZDDLE0 routine.

Module DLZDDLE0 detected and issued an error in routine GOTOBUFF after label RETBUFF1.

Locate the pseudo abend save area (see message DLZ002I). Register 12 in the pseudo abend save area points to the module which detected the bad return code.

Check the PSTFNCTN and PSTRTCDE fields in the pseudo abend save area.

A bad insert position in the SDBs from retrieve (DLZDLR00).

The data base has bad pointers.

DLZ861I - DLZDDLE0

An error is detected at label ABEND861 or an error was detected in routine HIISRTR.

The registers are saved in the pseudo abend save area by the RETURNER routine. To locate the pseudo abend save area, see message DLZ002I.

Use the registers in the pseudo abend save area to find the following fields:

R1 = PST
R4 = SDB
R6 = JCB
R9 = Address of the bad segment code
R11 = Insert address
R12 = Entry point DLZDDLE0.

Check compared registers to see if they are loaded correctly by using routine HIISRTR.

The DBD generation or VSAM DEFINE command changed the length of the segments and data was not reloaded.

A bad insert position in the SDBs from retrieve (DLZRLR00).

DLZ862I - DLZDDLE0

The error was detected in routine CKKEYLP and issued at label ABEND862. The registers are saved in the pseudo abend save area by the RETURNER routine. Figure 3-8 shows a possible example of a relation described in the message.

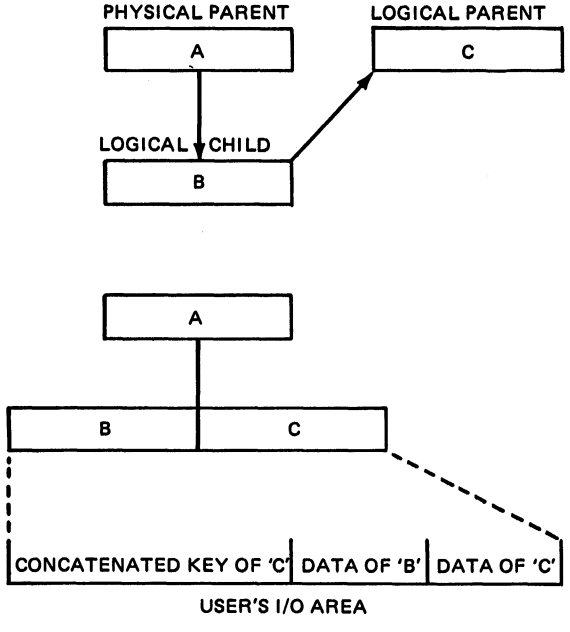


Figure 3-8. Parent/Child Relationship

DLZ863I - DLZDDLE0

An error was detected after label NOLLFLD and issued at label ABEND863.

The registers in the PSTSV3 DL/I Abend Save Area are as follows:

R1 = PST address
R2 = Segment source address in the user's I/O area.
R4 = LSDB address
R5 = PSDB address
R10 = Compressed segment address
R12 = Entry point DLZDDLE0.

DLZ864I - DLZDDLE0

The error was detected in routine HIISNXLV at label CALLERR1.

The pseudo abend save area is loaded.

DLZ868I - DLZDXMT0

This message is issued for two reasons:

1. Writing X'40' records on a work data set causes an I/O error. The error is detected and indicated in the LWORKTAP routine.

   Note: In this case, there must have been another error message previously issued.

2. Bad control blocks were found for a secondary index. An error is indicated at label LABENDXX. The error was detected in the LFINDSDB routine. The pseudo abend save area is loaded by the LABEND routine.

Find the error routine.

Check for a DL/I I/O error message before this message. If yes, the first explanation describes the cause.

CHAPTER 4. MODULE AND PROGRAM AIDS

This chapter is a summary of the modules and utility programs and:

- Their mode of operation
- Their access methods
- Their save areas
- Suggestions for possible reasons for their failure

and a description of a test program with:

- Control statement formats
- Sample job control statements
- Suggestions for usage

Pictorial layouts are used for:

- Utility program data flow
- Overview of call processing
- CSECTS and their entry points and labels

BATCH INITIALIZATION/TERMINATION (DLZRRC00)

Section 3 "Program Organization" of the DL/I DOS/VS Logic Manual, Volume 1, and the HIPO charts in the DL/I DOS/VS Logic Manual, Volume 2, give a description of the functions of DLZRRC00 and may be used to determine how much initialization or termination was completed.

Notes:

1. The user must save initialization registers in the save area pointed to by R13 on entry to the application program. This save area is located at label PCC0SAVE in module DLZRRC00. The registers are reloaded from the save area when terminating by the RETURN statement. R14 points to the termination entry point in DLZRRC00 module.

2. The STXIT AB save area overlays the DLZRRC00 module at label DLZPINIT.

   To find the save area, see the notes for message DLZ002I in the DL/I DOS/VS Messages and Codes manual.

BUFFER HANDLER (DLZDBH0N)

Section 3 "Program Organization" of the DL/I DOS/VS Logic Manual, Volume 1, and the HIPO charts in the DL/I DOS/VS Logic Manual, Volume 2, give a description of the functions of the buffer handler which consists of three modules:

DLZDBH00

DLZDBH02

DLZDBH03

The resulting phase controls all I/O except HSAM and SHSAM which is controlled by the action modules. The first module is responsible for ESDS processing using DL/I buffers, the second for processing using VSAM buffers, and the third for some internal and housekeeping routines.

Linkage is accomplished by:

TABLE - (PSTFNCTN x 4) = ENTRY POINT

For all functions the buffer handler is entered at DLZDBH00. The function requested in the PST is decoded and used as a displacement into a table of entry points. If the required routine is not in the first module the address of the required module is loaded and the process is repeated on entry.

The first module also contains a table (ROUTAB) of entry points for internal routines. This is accessed by a displacement in R4.

Save areas for entry registers are stored in the current PST save area (see Figure 2-11 on page 2-19) unless the buffer handler is called directly from a utility. Internal flow is by a BALR 14,15 instruction. Return registers are saved in the PST at label PSTRETRE with register 9 pointing to the current entry.

Suggestions:

- Verify the input to the buffer handler. The DL/I DOS/VS Logic Manual, Volume 1, contains a chart showing the input for each function.

  Note:

  PSTBYTNM = VSAM RBA, and
  PSTBLKNM = DL/I relative block number (RBA).

- Check the parameter statement for specification and assignment of buffers.
- Use LISTCAT to verify correct information (high-used RBA, etc.) passed to DL/I (DMB) at open/close time.

DATA BASE RECOVERY UTILITIES

Before attempting to debug utility programs, it is recommended the appropriate sections of the DL/I DOS/VS logic manuals. and the DL/I DOS/VS Resource Definition and Utilities or DL/I DOS/VS Interactive Resource Definition and Utilities be read. They give a description of how the utility programs work.

There are two major causes of utility program problems:

- The first is the lack of storage or GETVIS area. DL/I DOS/VS Data Base Administration should be consulted and, where applicable, the program rerun before further debugging.
- The second common cause of failure is changes are made to system access methods and not to DL/I. For this reason, the access methods used in each utility are listed separately.

DLZBACK0 - DATA BASE BACKOUT UTILITY

Module DLZBACK0 reads the log file backwards with undefined record length and passes the record read to DLZRDBC0 which updates the segment. A data flow for this utility is shown in Figure 4-1 on page 4-4.

Mode of Operation:

DL/I application program.

Access Methods:

- Input = SAM read backwards.
- Output = Buffer handler using PSTFNCTN.

Save Area:

Initialization registers are stored in PCC0SAVE in modules DLZRRC00, DLZBACK0, and DLZRDBC0.

Suggestions:

- Check that the data base was closed properly. If not, run the VSAM Verify program.
- Online changes not backed out. These are backed out only if the online system failed. If the application program failed but the online system did not fail, an X'07' type termination record is written to the log during termination and the application program's changes are ignored.

DLZLOGP0 - LOG PRINT UTILITY

The log print utility enables you to print the contents of DL/I log files to help you recover from system failures. Used in conjunction with the DLZBACK0 - Data Base Backout Utility, the information printed may be used to determine which data bases to backout or recover in case backout fails. A data flow for this utility is shown in Figure 4-2 on page 4-4.

Mode of Operation:

Application program; it does not make any DL/I calls.

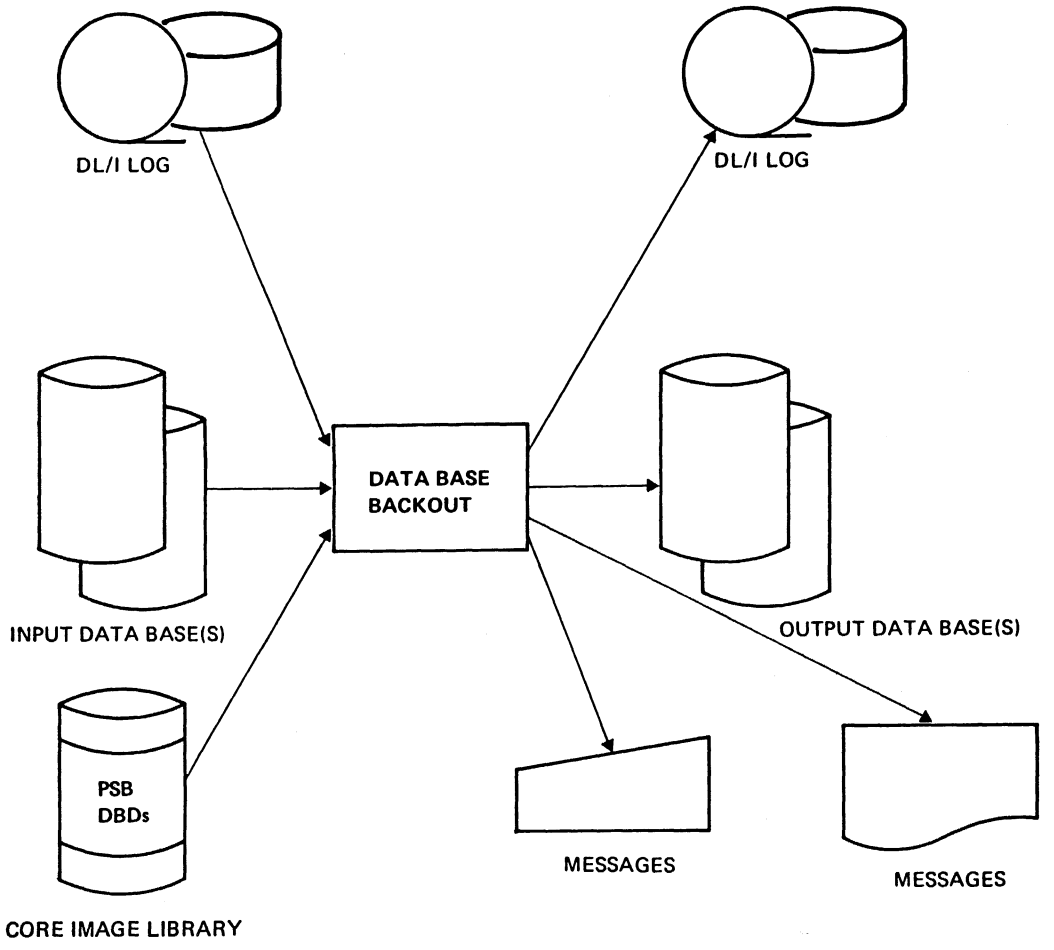


Figure 4-1. Data Base Backout Utility Program

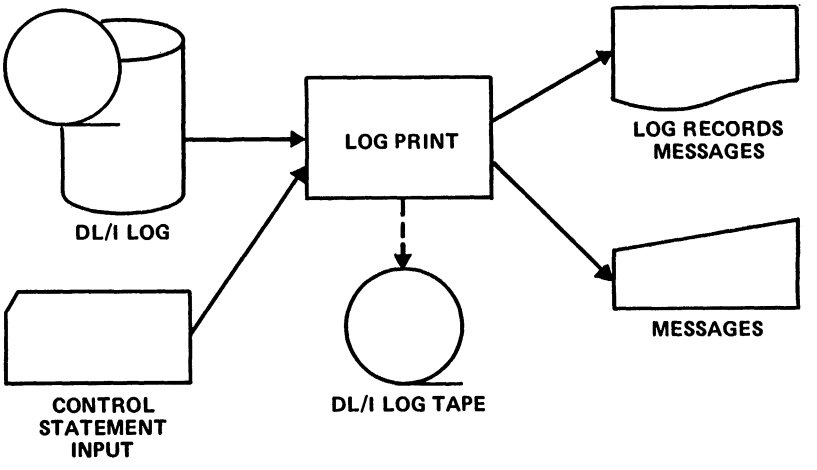


Figure 4-2. Log Print Utility Program

Access Methods:

- Input =
  - Standard labeled tape log file.
  - Unlabeled tape log file.
  - VSAM disk log file.

- SAM disk log file.

- Output =
  - Log records on SYSLST.
  - Information messages on SYSLST.
  - Error messages on SYSLST.
  - Error messages on SYSLOG.
  - Standard labeled tape log file (optional).

DLZUCUM0 - DATA BASE CHANGE ACCUMULATION UTILITY

A data flow for this utility is shown in Figure 4-3 on page 4-6.

Mode of Operation:

Application program using DOS/VS Sort/Merge.

Access Methods:

- Input = SAM
- Output = SAM

Save Area:

At label SAVEREG in module DLZUCUM0. Sort/Merge registers are saved on exit at label SAVEEX15 in module DLZUC150, and at label SAVEEX35 in module DLZUC350. A common DSECT, CUMCONST, holds the constants used by all modules.

Suggestions:

- Because of Sort/Merge and GETVIS usage, the size parameter and partition size is critical and should be checked first.
- Output is grouped in the GETVIS area by "like id". If LLC/CC is used where many records have a "like id", increase the GETVIS area to the maximum.
- Message 0P47A may appear on SYSLOG at open time of the second log tape, if the first tape is still rewinding. This clears when the new log tape is made ready.

DLZUDMP0 - DATA BASE DATA SET IMAGE COPY UTILITY

An error in this utility is most likely caused by VSAM. Processing is done by the VSAM GET macro. See the VSE/VSAM logic manuals for debugging. This utility backs up the individual VSAM data sets; for example, the KSDS (primary index) or ESDS (HIDAM data base). Also, data sets for a data base must be backed up at the same time without intervening updates. A data flow for this utility program is shown in Figure 4-4 on page 4-7.

Mode of Operation:

Application program.

Access Methods:

- Input = VSAM
- Output = SAM

Save Area:

At label SAVEREG in module DLZUDMP0.

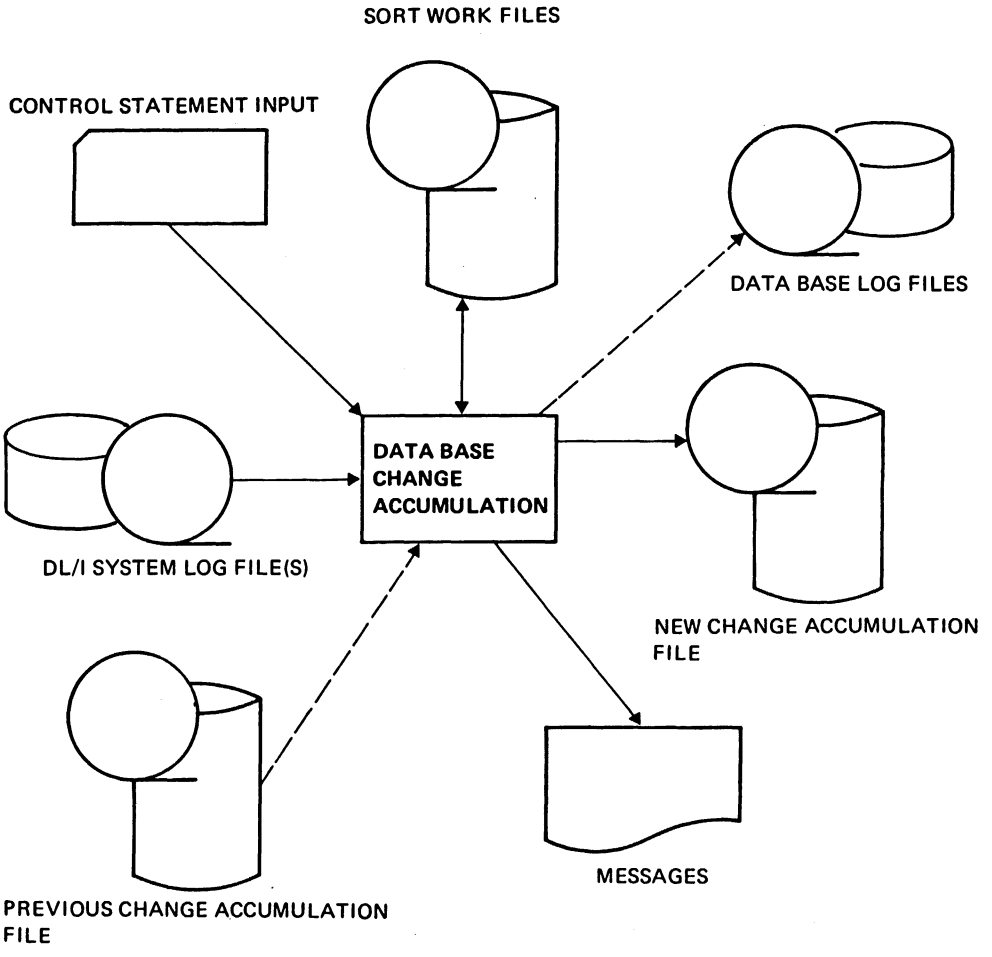


Figure 4-3. Data Base Change Accumulation Utility Program

Suggestions:

- Check that the DMB is accurate because it is used to generate the VSAM control blocks.
- Debug as a VSAM problem.

DLZURDB0 - DATA BASE DATA SET RECOVERY UTILITY

This is a two phase utility, either of which may be run separately. The first phase merges the image copy (DUMPIN) with the previous change accumulation (CUMIN). The second phase then merges any new log tapes producing an updated data base. A data flow for this utility program is shown in Figure 4-5 on page 4-8.

Mode of Operation:

DL/I application program.

Access Methods:

- Input = SAM
- Output = Buffer handler using PSTFNCTN.

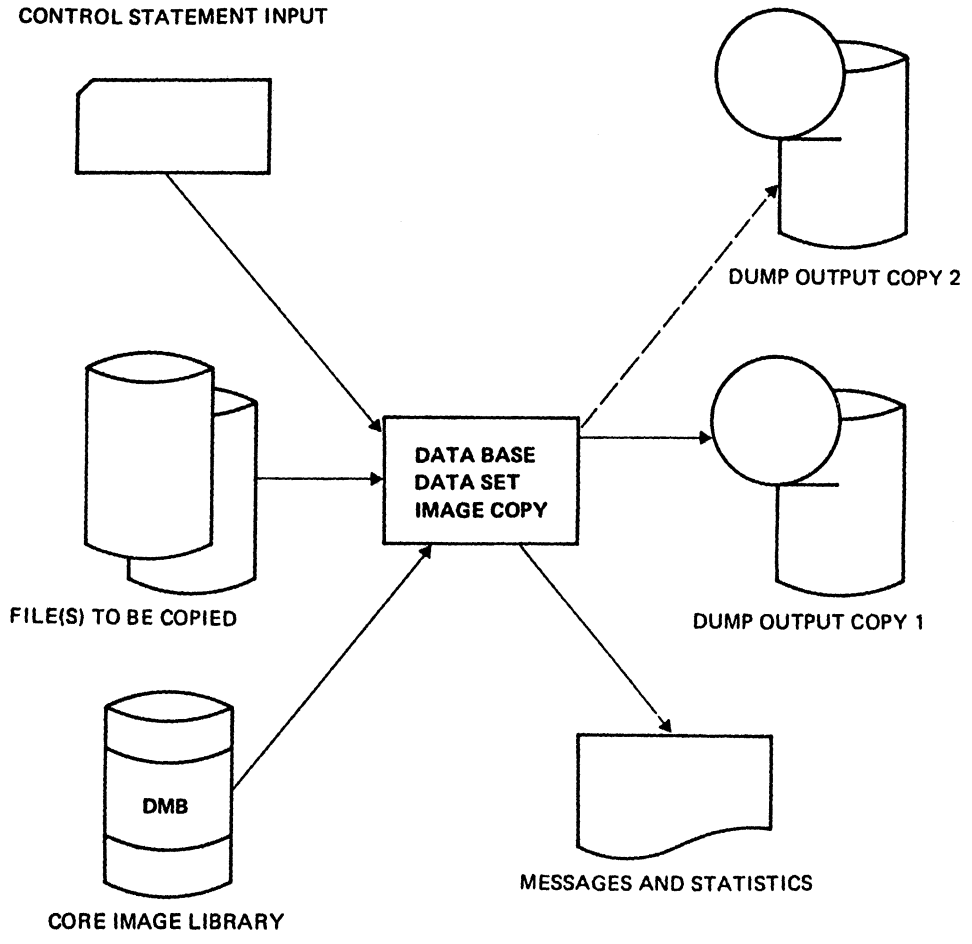


Figure 4-4. Data Base Data Set Image Copy Utility Program

Save Area:

Initialization registers are saved at label PCC0SAVE in module DLZRRC00. On entry to the buffer handler, the registers are saved at label SAVEREG in module DLZURDB0.

Suggestions:

- SYS013 is unassigned, an I/O error occurred (possibly causing an open error), or log tapes are not used or are not mounted,
- If HISAM unload output was used as DUMPIN, check that the reload was performed immediately after the unload or changes that were made will occur in the DUMPIN and the log files.
- If receiving a primary index data base, check that the DBD name specified on the UDR statement is that of the HIDAM data base, since only one utility PSB is created for HIDAM. This error results in job cancellation because a utility program PSB (dbdname) does not exist.

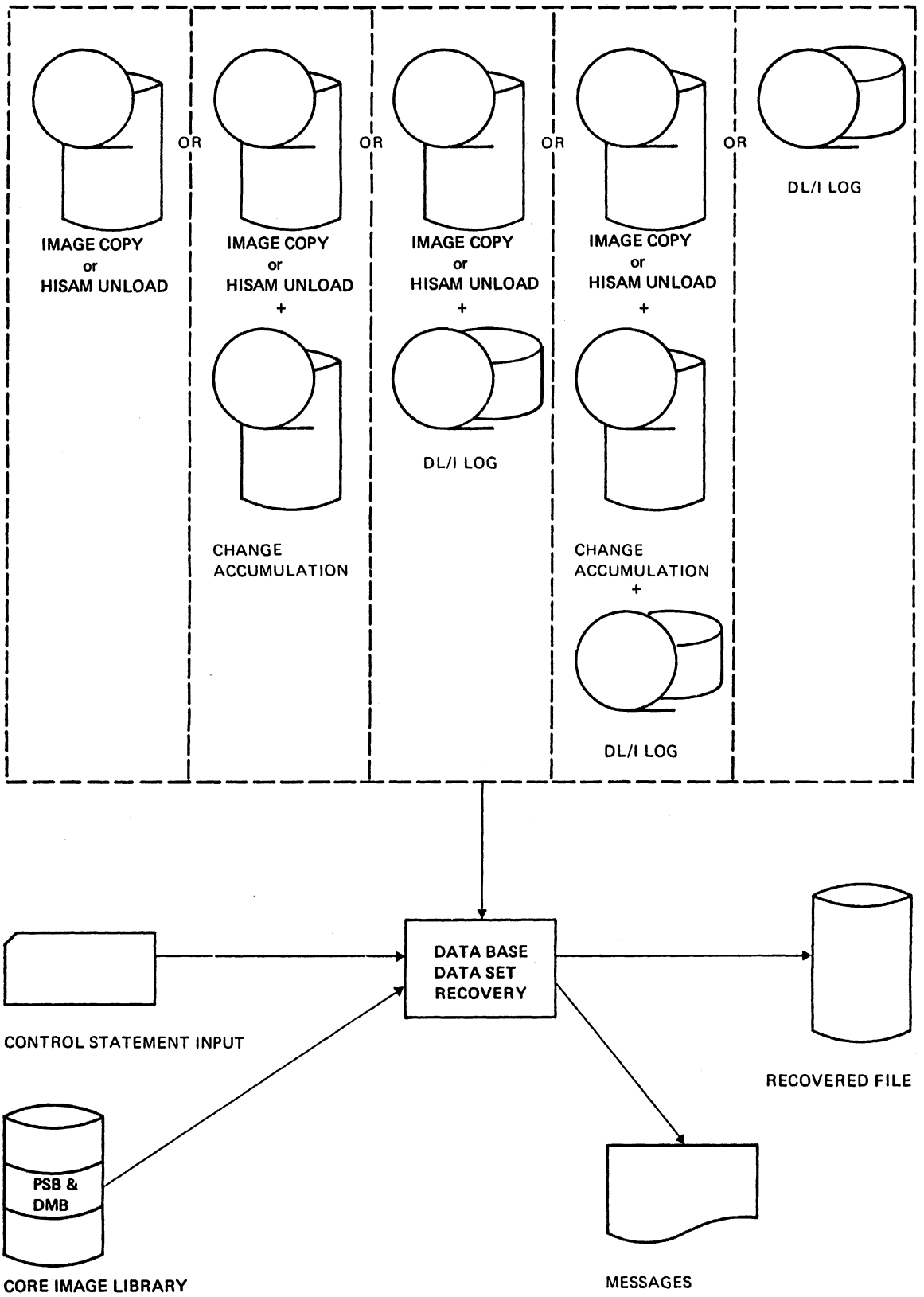


Figure 4-5. Data Base Data Set Recovery Utility Program

DATA BASE REORGANIZATION UTILITIES

DLZURGL0 - HD REORGANIZATION RELOAD UTILITY

A data flow for this utility program is shown in Figure 4-6 on page 4-10.

Mode of Operation:

DL/I application program.

Access Methods:

- Input = DIMOD GET
- Output = DL/I ASRT
- On restart: Input also = DL/I GU and GN (for partially reloaded data base).

Save Area:

At initialization the registers are stored at label PCC0SAVE in module DLZRRC00, and the DLZURGL0 module save area is at label SAVEREGS.

DLZURGU0 - HD REORGANIZATION UNLOAD UTILITY

This utility may be used to make changes to a data base for HD and HISAM organization. The data base is unloaded, the DBD is redefined and then the data base is reloaded. There are certain restrictions that should be studied carefully (see "HD Reorganization Reload Utility" in the DL/I DOS/VS Resource Definition and Utilities or DL/I DOS/VS Interactive Resource Definition and Utilities. A data flow for this utility program is shown in Figure 4-7 on page 4-11.

Mode of Operation:

DL/I application program.

Access Methods:

- Input = DL/I GN
- Output = DIMOD PUT

The output is a sequential file containing each active segment, together with 34 bytes of utility prefix information, arranged in hierarchical sequence. The output records, which are used by the HD reorganization reload utility, are blocked internally with a block size of 6446 bytes for tape, or 6454 bytes for disk.

Save Area:

At initialization registers are stored at label PCC0SAVE in module DLZRRC00, and the save area for module DLZURGU0 is at label REGSAVE.

Suggestions:

- To verify the input data base, run the test program (see "Test Program - DLZDLTXX" on page 4-23) issuing GN calls. If it fails, rerun with a compare statement for the failing call forcing a dump.
- A sequence check verifies the sequence of the segments within the data base. If a sequence error is encountered, message DLZ400I SEQUENCE ERROR IN DATA BASE, is issued. This implies a critical data base problem. The sequence error

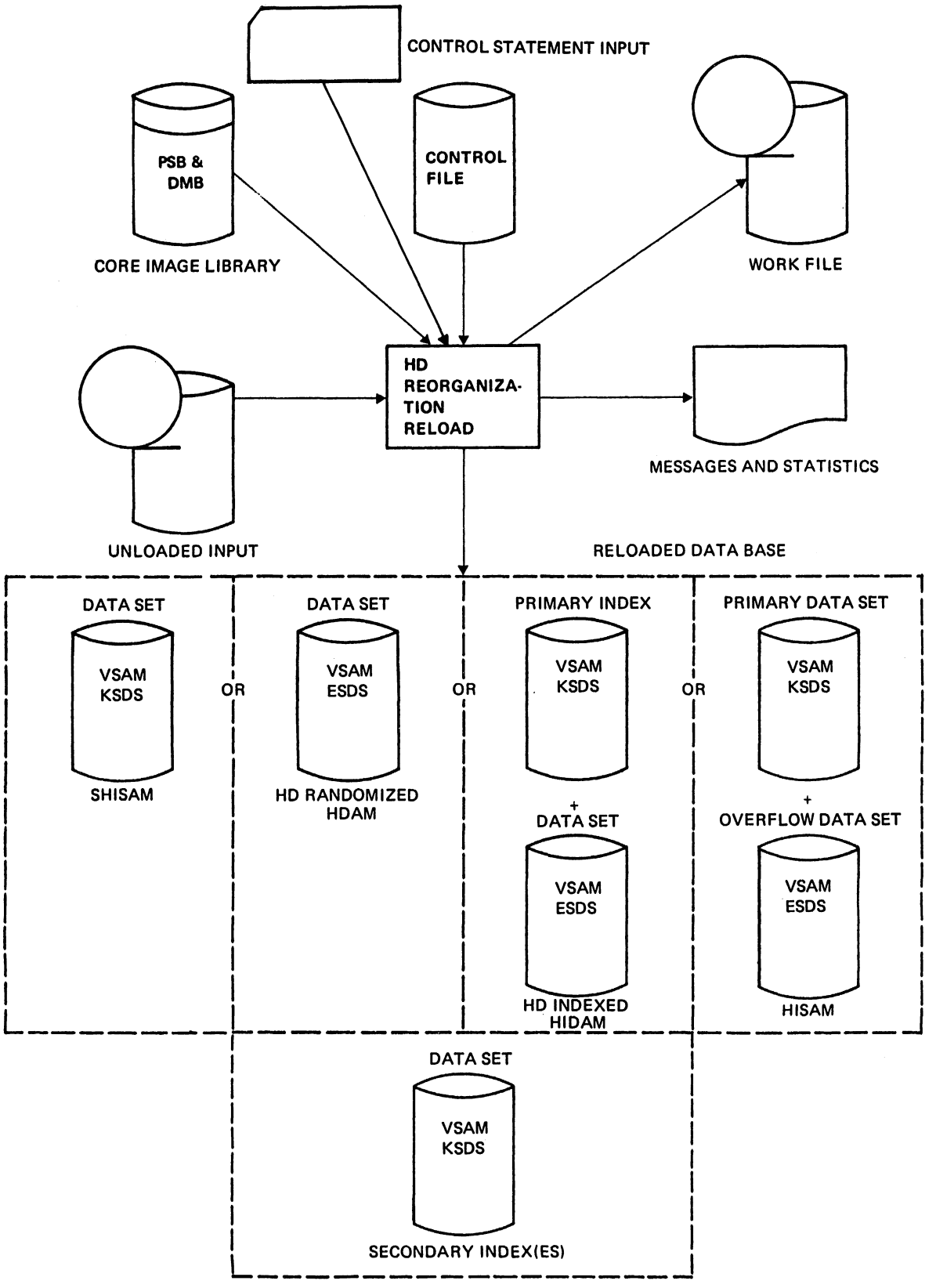


Figure 4-6. HD Reorganization Reload Utility Program

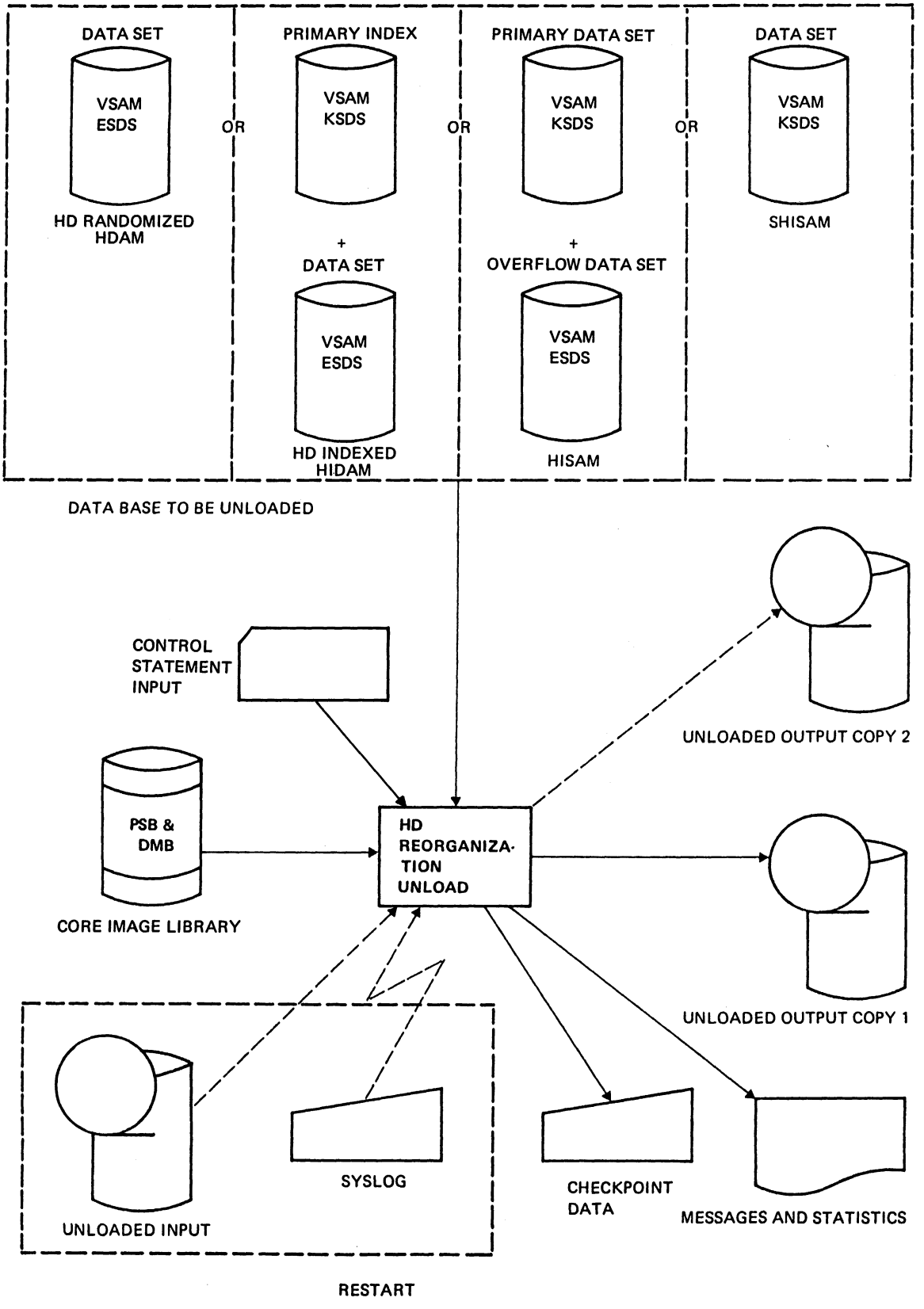


Figure 4-7. HD Reorganization Unload Utility Program

should be corrected and the job resubmitted. A procedure that can be used to locate the segment that caused the sequence error in the dump is described in the following "HD Unload Sequence Checking Procedure".

- If structural changes are intended, check that the original DMB is used for the unload.
- Check DL/I DOS/VS Data Base Administration for storage requirements. Insufficient storage area may result in a GE return code.
- If the checkpoint restart facility was used, check that the data base was restored to its original status before rerunning.

HD Unload Sequence Checking Procedure

During an HD unload of a DL/I data base, the DL/I HD unload utility sequence checks the segments as they are retrieved from the data base. The HD unload utility checks for two segment error conditions:

- Within each data base record, it checks to see that the physical segment codes occur in ascending sequence.
- Within sequence fields, it checks to see that the sequence fields occur in ascending sequence and that no duplicate sequence fields exist if the DBD specifies that the sequence fields must be unique.

In the event that the HD unload utility detects a sequence error for either of these conditions it issues message "DLZ400I SEQUENCE ERROR IN DATA BASE" and cancels the job with a JDUMP.

The following procedure can be used to locate the segment that caused the sequence error in the dump:

1. Locate the literal '********SEQUENCE ERROR REGISTERS*' in the dump. Immediately following this literal is a 16 fullword register save area. The registers are saved in the sequence R0 to R15. Register 6 points to the I/O area that contains the segment that caused the sequence error. Figure 4-8 on page 4-13 is the IOAREA DSECT format.

   In addition to the information found in the IOAREA DSECT (Figure 4-8 on page 4-13), the following information may be found:

   - At label SEQPCSV is the physical segment code for the previous segment read.
   - The value of the key of the previous segment read can be found at the label STET.

   By comparing these fields to those in the IOAREA DSECT you can determine what condition caused the sequence error.

2. Register 4, as saved in the sequence error register save area, points to the PCB used by the HD Unload Utility. The key feedback area of the PCB should help you identify the data base record containing the segment that caused the sequence error.

One way to circumvent the sequence error is to use a user written unload program that skips retrieving specific segments based on control card input. After the good segments have been unloaded and the data base reloaded, the missing information can be restored by inserting it with a user program.

Offset Dec(Hex)		Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	1	RGUSEGLV		Segment code of this segment
1	1	1	RGUHSDF		Delete flag used by HSAM flags. HSAM has the following special format -- DB: HSAMDFLG EQU X'00'
2	2	2	RGUHDRLN		Length of header portion of the record
4	4	2	RGUSEGLN		Length of data portion (IOSEG) of the record
6	6	8	RGUSEGNM		Segment name
14	E	1	RGUSEGDF		Segment delete flag
15	F	4	RGUPFCTR		Prefix counter field
19	13	4	IOTWFOR		Logical twin forward pointer
23	17	4	IOTWBACK		Logical twin backward pointer
27	1B	4	IOPAR		Logical parent pointer
31	1F	4	IOOLD		Old location of record
35	23		IOSEG		Variable length data field

Figure 4-8. IOAREA DSECT Format

PARTIAL DATA BASE REORGANIZATION UTILITY

The partial data base reorganization utility (see Figure 4-9 on page 4-14) reorganizes a user-selected range of HD, HIDAM, or HDAM data base records into a designated target area within the data base.

For HD indexed and HIDAM data bases, the range is defined by a set of low-high key values in the primary index data set. For HD randomized and HDAM data bases, the range is defined by a set of low-high relative block numbers in the root addressable area.

PART1 (DLZPRCT1) performs pre-reorganization functions to check for user errors without using the data base.

PART2 (DLZPRCT2) does the actual reorganization of the data base. It is run as a batch job and requires use of the data base.

Note: Structural changes to the data base cannot be made with the partial data base reorganization utility.

The IUP (installed user program) DL/I DOS/VS Space Management Utilities, Program Number 5796-PKF, can be used to create a report to assist in identifying ranges that need reorganization and selecting a target area.

DLZURRL0 - HISAM REORGANIZATION RELOAD UTILITY

The header record read from the unloaded data base is used to provide the information for generation of the VSAM control blocks. Then records are read sequentially and processed for

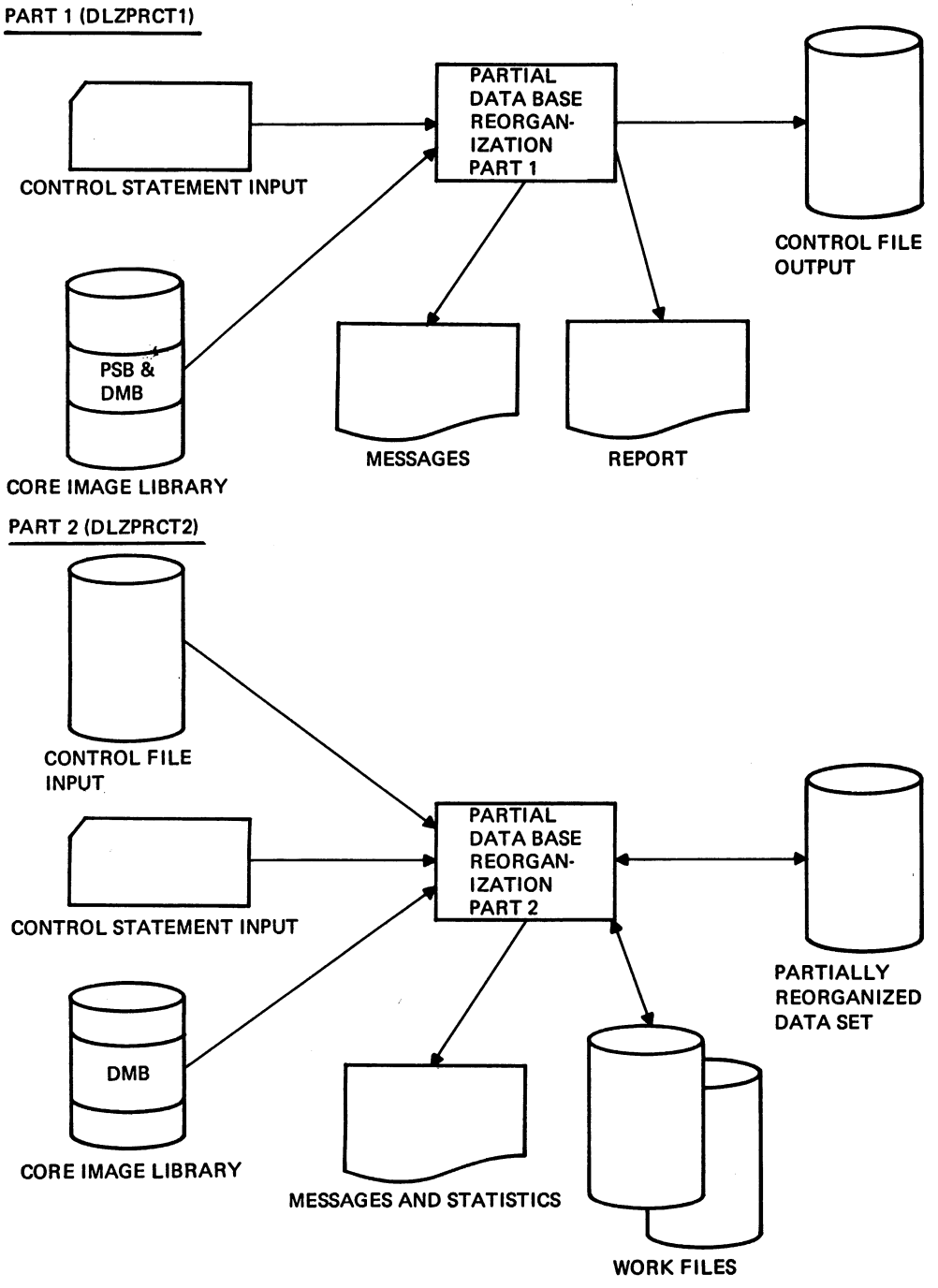


Figure 4-9. Partial Data Base Reorganization Utility

statistical information before being written to the output data bases. A data flow for this utility program is shown in Figure 4-10.

Mode of Operation:

Application program.

Access Methods:

- Input = SAM

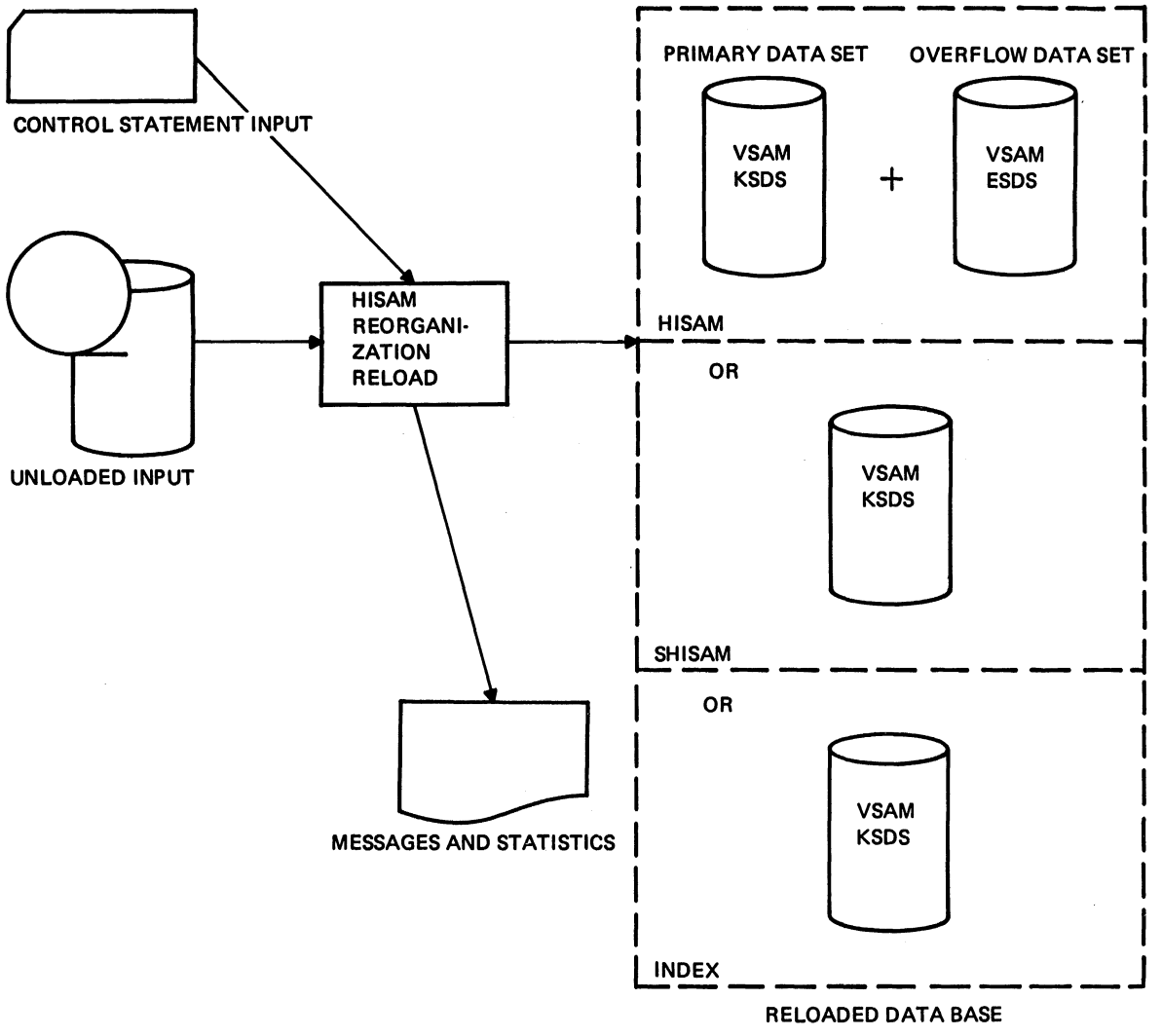


Figure 4-10. HISAM Reorganization Reload Utility Program

- Output = VSAM

Save Area:

At label SAVEREG in module DLZURRL0.

Suggestions:

- Check the header record on the unloaded data base. The problem may be in unloading.
- Check the mandatory VSAM DELETE and DEFINE runs prior to the last successful reload.
- Using LISTCAT, check the catalog entry for this data base.
- Check that no structured changes were made to the DMB. If there are changes, HD unload/reload must be run.

DLZURUL0 - HISAM REORGANIZATION UNLOAD UTILITY

The DMB is loaded from the core image library and a short segment table is built using the information in the SDBs. This is used to create the VSAM control blocks. A data flow for this utility program is shown in Figure 4-11.

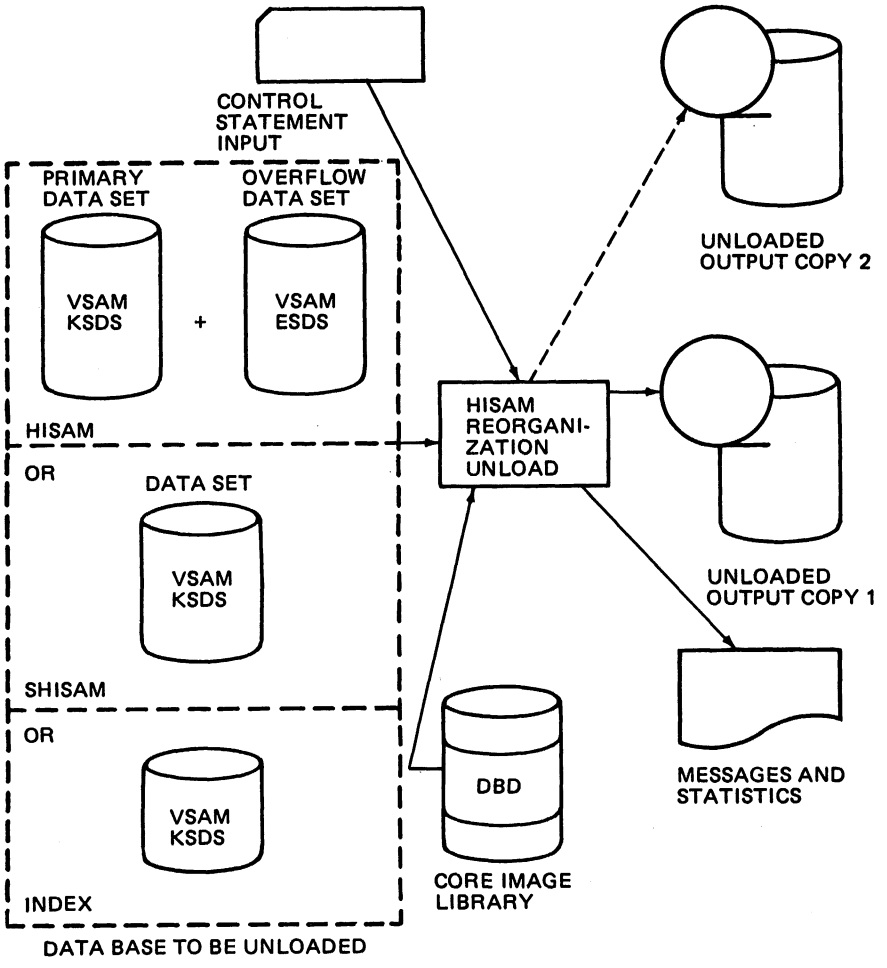


Figure 4-11. HISAM Reorganization Unload Utility Program

Mode of Operation:

Application program.

Access Methods:

- Input = VSAM
- Output = SAM

Save Area:

At label SAVEREG in module DLZURUL0.

Suggestions:

- Check the DBDGEN because it is used to create VSAM control blocks. Bad SDBs may result in bad or lost data because the output is written when the buffer is full and not when a new root is found.

- Records are obtained by VSAM and not by DL/I. Logically deleted records are also obtained. Check any previous delete run for errors.
- The DL/I test program (see "Test Program - DLZDLTXX" on page 4-23) may be used to find the physical position of the last record written to the output data base. A print of the input data base from this point may be useful in case of corruption.

  Note: This must be done with a print program and not with DL/I because you may need to look at logically deleted records.

LOGICAL RELATIONSHIP UTILITIES

Figure 4-12 on page 4-18 shows the necessary programs required for physical reorganization and/or logical relationship resolution processing during initial load or reorganization of an HD, HDAM, or HIDAM data base.

DLZDSEH0 - WORKFILE GENERATOR UTILITY

This program is called by HD reload and the scan utility to create workfile records showing the old and new location of the segments involved in logical relationships. The interface between module DLZDSEH0 and its caller depends on the function required. The DL/I DOS/VS Logic Manual, Volume 1, gives the details of register usage and return codes under the heading "Interfaces = DLZDSEH0".

DLZURPR0 - DATA BASE PREREORGANIZATION UTILITY

This module uses the DMB for the data bases requested by control statements to decide if prefix resolution is necessary. If it is, a control file entry is created that describes which prefix pointers need resolution. It also creates the control statements necessary for the scan utility.

Mode of Operation:

DL/I application program.

Save Area:

At initialization registers are stored at label PCC0SAVE in module DLZRRC00, and the DLZURPR0 save area is at label REGSAVE.

DLZURGS0 - DATA BASE SCAN UTILITY

This program issues the GN calls with a qualified segment name according to the information in the control file. Using the information in the LEVTAB, a branch is made to the buffer handler with PSTFNCTN = X'02' and the physical location of the segment is passed to the work file generator.

Mode of Operation:

DL/I application program.

DLZURG10 - DATA BASE PREFIX RESOLUTION UTILITY

The first phase of this utility program uses the DOS/VS Sort/Merge program to group records from load, reorganization, or scan together which refer to the same logical parent. The second phase, also using Sort/Merge, sorts the records into physical locations within the data base(s).

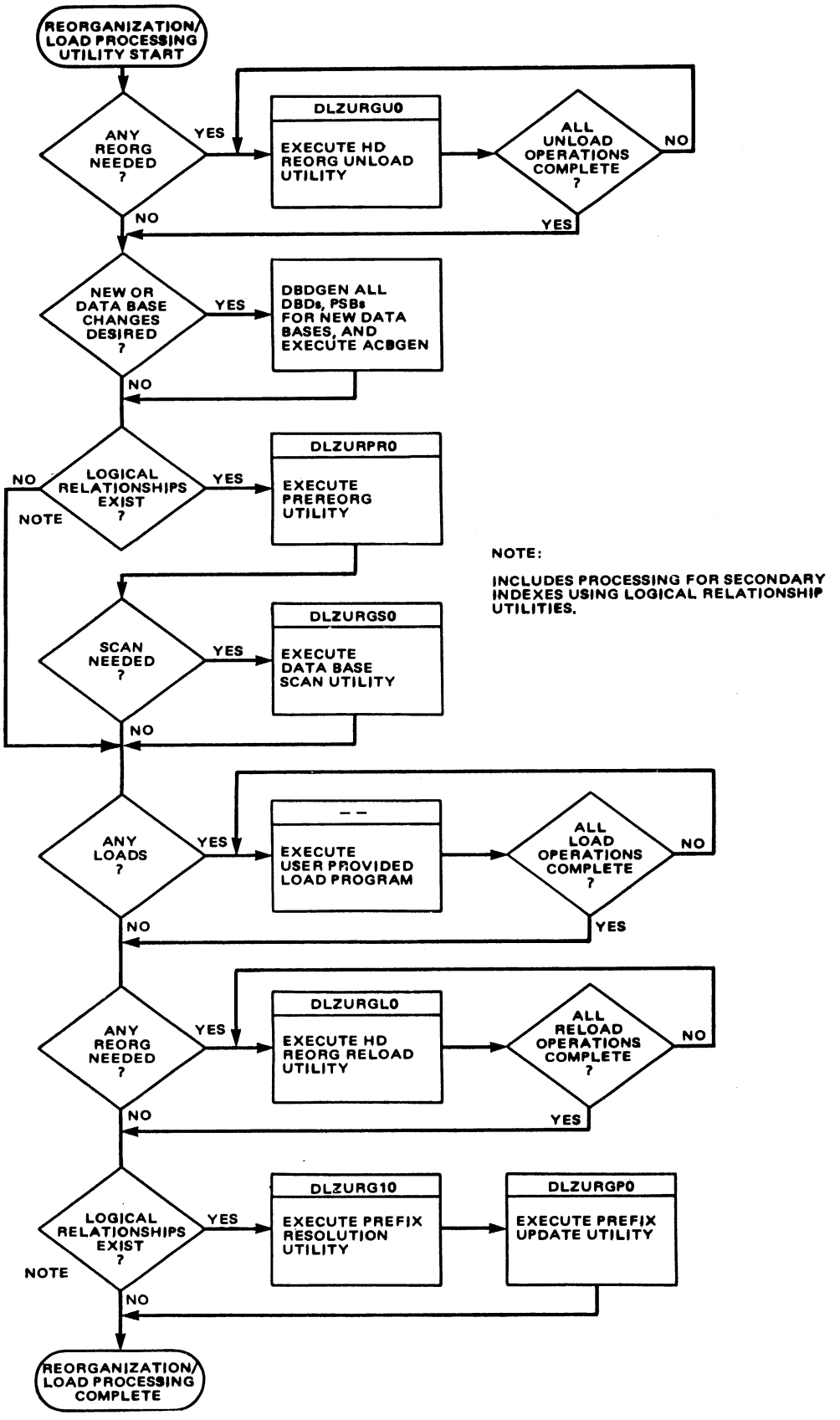


Figure 4-12. Reorganization/Load Flowchart

Mode of Operation:

Application program; it does make any DL/I calls.

Save Area:

At label SAVEREG in module DLZURG10, and at save area label EXITSAVE for exits 15 and 35.

DLZURGP0 - DATA BASE PREFIX UPDATE UTILITY

Records are obtained from the workfile with SAM. The original record is obtained from the data base by branching to the buffer handler with PSTFNCTN = X'06' (locate and mark buffer altered). The prefix is then altered to the updated value. For new records (load mode) a DL/I ISRT call is issued.

Mode of Operation:

DL/I application program.

MPS BATCH INITIALIZATION, TERMINATION AND PROGRAM REQUEST HANDLER - DLZMPI00

Section 3 "Program Organization" of the DL/I DOS/VS Logic, Volume 1, and the HIPO charts in the DL/I DOS/VS Logic, Volume 2, give a description of the functions of DLZMPI00. The module is made up of the following five routines:

DLZMINIT - MPS batch initialization

DLZMMSG - MPS batch message writer

DLZMABND - MPS batch abend routine

DLZMPRH - MPS batch program request handler

DLZMTERM - MPS batch termination

Note: DLZMPI00 is the only DL/I component residing in the MPS batch partition.

ONLINE OPERATION PROGRAMS

DLZODP - ONLINE INTERFACE

This module is loaded during CICS/VS initialization when DL/I is requested in DFHSIT. It is part of the phase DLZNUCxx and the dump identifier is +DLZNUCXX+. DLZODP has the CSECTs and entry points shown in Figure 4-13 on page 4-20.

DLZOLI00 - ONLINE INITIALIZATION

Module DLZOLI00 is loaded as DFHSIDL during CICS/VS initialization. At this time DLZNUCXX (the DL/I nucleus) is already loaded by CICS/VS and it contains the ACT that is referenced by DLZOLI00. The module has two CSECTs: DLIOLI00 with a dump identifier of DLZOLI00 and DLIOLI10 without a dump identifier. DLZOLI00 (DFHSIDL) has the labels and operations shown in Figure 4-15 on page 4-22 .

DLZPRH00 - ONLINE PROGRAM REQUEST HANDLER

This module handles the following DL/I calls:

CSECT	ENTRY	DUMP ID	NOTE
DLZODP	DLZODP00	DLZODP[1]	Dummy initial scheduling routine (not executed).
	DLZSCHDL	--	DL/I task scheduling.
	DLZODP03	--[1]	Entered from DFHSRP, it causes an abend exit path through DLZODP02.
	DLZODP02	DLZODP02[1]	Entered from CICS/VS system termination for DL/I normal termination (and abnormal system termination following DLZODP03).
	DLZODP04		Entered from the local PSB scheduling routine to write a CICS/VS Start of Task record beforethe PSB schedule record is written to the CICS/VS journal
	DLZODP07		Entered from DFHPCP on abnormal task termination before STXIT routine is given control to ensure that DL/I can cancel outstanding operations before STXIT routine gets control.
	DLZODP06		Entered from DFHPCP on abnormal task termination before dynamic transaction backout. Routine invokes IDUMP or formatted task dump routine, DLZFTDP0.
DLZODP01	DLZTKTRM	DLZODP01[1]	Entered on normal or abnormal task termination from the program request handler (DLZPRH00), from DFHPCP when TCADLITE=1, and from DFHKCP when task is stall purged.
	DLZTKBAD		Entered from program request handler on PSB scheduling error.
	DLZODP05	DLZODP05	Entered from task termination routine after termination record is written to system log.
DLZPRH00	DLZPRH00	DLZPRH00[2]	Entered from the language interface for DL/I call and from DLZEIP00 for online HLPI command
	DLZABND0	--[3]	Online abend routine.
			System call processing routine (PROCSYS).
DLZOLT00	--	DLZOLT00	Builds trace entry for scheduling call.
	DLZOLT01		Builds trace entry for data base call.
	DLZOLT02		Builds trace entry for termination call.
DLZOWAIT	--	DLZOWAIT[3]	Issues CICS DFHKC TYPE=WAIT for DL/I tasks and gives ABEND D062 for these tasks if DL/I abends (see SCDIND2) during wait. Also contains CMF hooks interface routine for PI queueing facility.
	DLZSUSP	--	Common suspend routine for DL/I online modules.
	DLZRESUM	--	Common resume routine for DL/I online modules.
	DLZCMFHK	--	CMF Hook interface for program isolation queueing facility.

[1] Address in DFHDLIAL -- This is part of DLZNUCxx and provides addressas of DL/I routines for CICS/DOS/VS.

[2] Address in COMREG.

[3] Address in SCD.

Figure 4-13 (Part 1 of 2). CSECTs and Entry points for DLZODP

CSECT	ENTRY	DUMP ID	NOTE
DLZOVSEX	DLZOVSEX	DLZOVSEX[4]	VSAM EXCP exit routine to issue CICS DFHKC TYPE=WAIT to ensure only task waits and not the whole partition.
DLZERMSG	--	DLZERMSG	Issues CICS DFHTD TYPE=PUT for error messages (DESTINATION=CSMT) and EXCP to SYSLOG.
DLZODP10	DLZODP10	DLZODP10	Common get storage routine for DL/I online modules.
	DLZODP11	DLZODP11	Common free storage routine for DL/I online modules.

[4] Address in SCD.

Figure 4-13 (Part 2 of 2). CSECTs and Entry points for DLZODP

- PSB scheduling - Allocates a PSB for a task (see Figure 2-11 on page 2-19 for an example).
- Normal - Performs data base operation.
- Termination - Frees PSB from task.
- CMXT - Changes current MAXTASK value.
- STRT - Opens data base (FCT=DEFERRED).
- STOP - Closes data base.
- TSTR - Starts DL/I trace.
- TSTP - Stop DL/I trace.

Figure 4-14 shows the CICS/VS call interface application program interface to DL/I.

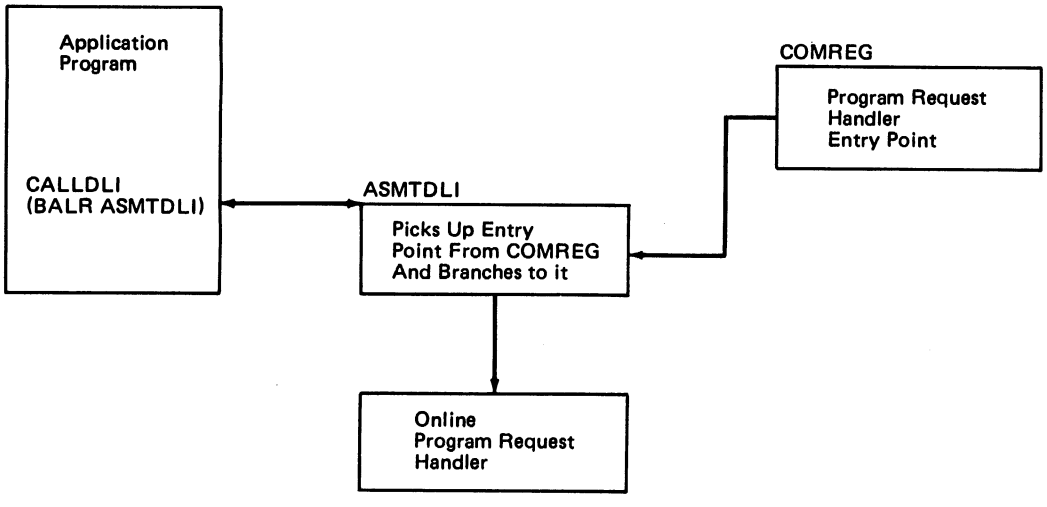


Figure 4-14. CICS/VS Call Interface Application Program - DL/I Interface

Label	Operation	Table/PGM	Flag	Field	Message
DLIOLI00	• Store CICS/VS Registers 0 - 15. • Save original storage remaining. • Check CSA OPT Feature List for DL/I.	DLZOLI DLZOLI CSAOPFL		CICSAVAR INITSZSV, INITUPSV CSADLI	
NONUCER	If 00 address, issue messages and return to CICS/VS after loading dummy DL/I.	DLZOLI			DLZ052I DLZ064I
NUCFOUND	• Save UPSI. • SAVE IWAIT address. • COMPARE ACT and PPT entries.	SCD SCD DLZOLI	 80	SCDSIND SCDIWAIT INITSW	
PSBLOAD	• Store 00s in PST at initial upper area – Set flag. • Store SCD address.	SCD PPST PST	 80 	SCDCWRK PPSTECB PSTSCDAD	
PSBILUP	• Using PDIR entries, load PSBs. – Any missing, set flags. – Issue message and continue.	 DLZOLI PDIR DLZOLI	 40 01 	 INITSW PDIROPTC 	 DLZ044I
PDIRNPCB	• Check PSBs. – If errors, issue message and continue.	 DLZOLI DLZOLI DLZOLI			 DLZ071I DLZ043I DLZ042I
PDIRPSMV	• Move Intent List.				
DDIRINIT	• Initialize DDIR. • If no FCT, issue message, load dummy DL/I and return to CICS/VS. • If FCT entry, issue FFs message and continue. • If FCT indicates INIT. DB opening, set INIT. OPEN flag.	 DLZOLI DLZOLI DLZOLI FCT DDIR	 01 20	 FCTDSOPN DDIRCODE	 DLZ045I DLZ054I DLZ046I
DDIRFOND	• If no DDIR entry, issue message, load dummy DL/I and return to CICS/VS.	DLZOLI			DLZ049I
DMBLLUP	• Load DMBs in sequence. – If not found, issue message and continue. – If wrong message level, issue message and continue. • Load randomizing module, if required. – If errors, issue message and continue. • If no valid DMBs found, issue message, load dummy DL/I, and return to CICS/VS.	DLZOLI DLZOLI DLZOLI			DLZ047I DLZ072I DLZ048I DLZ049I
DLZCPI00	• Build buffers in ascending CI size using value in PSTWRK3 (+1 for deletes). – If value X'00', issue message, continue as noted in reply. – If any problems with buffers, issue message and cancel partition.	PST DLZOLI DLZOLI		PSTWRK3	 DLZ060I DLZ262I
DLILOAD	• Check if DB logger is suppressed. – If yes, then zero logger entry in load sensitivity table to prevent loading. • Check if CICS/VS journaling. – If yes, then change DL/I logger name from DLZRDBL0 to DLZRDBL1. • Load action modules. – If none are found, issue message, load dummy DL/I and return to CICS/VS. • If OK, post "No Wait".	SCD DLZOLI SCD DLZOLI DLZOLI SCD	02 00s 01 80	SCDSIND LDSNSLGF SCDSIND NUCLODLG SCDESECB	 DLZ055I

Figure 4-15 (Part 1 of 2). Online Initialization

Label	Operation	Table/PGM	Flag	Field	Message
DLILOAD (cont'd.)	• If DB logging, set log open flag. • If no CICS/VS journaling, then open log DTF and attach logger. If attach is unsuccessful, issue message and close log DTF.	SCD DLZOLI	80	SCDDBLDP	DLZ061I
PCCORET	• Relocate PSBs after CICS/VS GETMAIN for storage after "Ensave". — If problem, issue message and continue.	SIPCOM DLZOLI		ENDSAVE	 DLZ056I
DMBOPENA	• Check if DB is to be opened. — If not, set flag. • Check if INIT is successful. — If not, reset OPEN bit.	DDIR DDIR DDIR DDIR	20 40 01 DF	DDIRCODE DDIRCODE DDIRCOD2 DDIRCODE	
DMBSCNX	• Branch to DLZDLOC0 to open DBs with OPEN ALL and continue. • Check DDIRCODE for OPEN ALL results. — If not X'20', X'00', or X'40', set flag, issue message and continue. — If equal X'40', check VSAM return code and issue message if not 00s. — If equal X'20' or X'00', continue.	 DDIR DDIR DLZOLI DDIR DLZOLI	 04 00	 DDIRCODE DDIRCODE DDIRVSRT	 DLZ057I DLZ057I
DMBSCNK	• Issue VSAM MODCB				
M113BLT	• Check Program Isolation status and build write message.	SCD	40	SCDDBMPS	DLZ113I
EXITOVL	• Check switch — If not 00, issue message and return to CICS/VS. — If 00, everything is OK, issue message	DLZOLI DLZOLI DLZOLI		INITSW	 DLZ054I DLZ053I
M119BLT	• Check status of DL/I logging and write appropriate status message	SCD		SCDSIND DLIATTFG	DLZ119I
INITRET	• Restore CICS/VS registers. • Return to CICS/VS.	DLZOLI		CICSAVAR	

Figure 4-15 (Part 2 of 2). Online Initialization

TEST PROGRAM - DLZDLTXX

The primary purpose of this program is to provide a facility to test DL/I by issuing calls based on control statements and, optionally, to compare the results with anticipated results. This program is, basically, an application program that issues calls to DL/I based on control statement information. It provides the facility to issue any call against any DL/I data base. The compare control statement can be used to compare the results of a call with anticipated results.

This program may also be used as a general purpose data base program; however, the control statement language does not lend itself to executing large volumes of calls. It is useful as a debugging aid, since it has the facility to display DL/I control blocks, and provides an easy method of executing any call against any data base.

The program is included in the released system because of its usefulness as a debugging aid. However, it is neither supported nor maintained for customer use so that the development group may be free to modify or expand it to satisfy their own requirements for testing. Secondly, it was not designed for general utility use; therefore, it is not ideally suited for that purpose.

GENERAL DESCRIPTION

The DL/I test program is basically a control statement processor. There are four basic control statements which are: status, comment, call, and compare. Additionally there are special control statements which will be discussed separately later in this section.

- The status statement is used to establish print options and to control which PCB is used when the PSB calls are issued. The call to be issued is provided in the call statement.
- The compare statement is optional and is used to tell the program what the expected results of this call should be in the data base PCB and/or in the user input/output area. Various print and display options are available, based on whether the results of the call agree with the data in the compare statement.
- The call statement is used to specify the DL/I call type and optionally SSA qualification. Boolean operators may be used to further qualify the segment to be located.
- The comment statements are optional. As the name implies, they contain only comments. There are two types of comments: conditional and unconditional.

The general sequence of operation is to read call statements until a noncontinued call statement is read. The DL/I call is issued based on data in the call statements, and another control statement is read. If a compare statement is read, this program compares what is in the compare statement with the corresponding field in the PCB, or, if it is a data compare statement, with the data in the user input/output area. The comments, call, compare, PCB, input/output area, and compare data are printed if requested. If any control statement other than a compare statement is read after a call is issued, the results of the prior call are printed first, and the new control statement is processed.

The DLZDLTxx program can accept control statements from either of two sources. This is controlled by the assignment of logical unit SYS010. To read the control statements from SYSIPT, SYS010 must be unassigned. Hence a //ASSGN SYS010,UA in the jobstream will cause DLZDLTXX to read the control statements from SYSIPT. It is also possible to create a disk with testcase jobstreams that are used frequently, and use them as input to DLZDLTXX. This must be a FIXBLK, LRECL = 80, BLKSIZE = 1600 file, and the name on the DLBL statement must be PRINPUT. The following is an example of the Job Control needed to read such a file:

```
// ASSGN SYS010,x'230'
// DLBL PRINPUT,'DISK.INPUT'
// EXTENT SYS010
```

BASIC CONTROL STATEMENT FORMATS

In the statement formats shown below, a $ indicates those fields which are normally filled in. The absence of a $ indicates that the field can normally be left blank.

Status Statement Format

$Col 01	S, identifies this as a status statement.
Col 02	Not used.
$Col 03	Print comments option. • Blank, do not print comments: • 1, always print comments. • 2, print comments only if compare is done and is not equal.
Col 04	Not used.
$Col 05	Print call option - same format as comments option.

Col 06	Not used.
Col 07	Print compare option - same as comments option.
Col 08	Blank.
$Col 09	Print PCB option - same as comments option.
Col 10	Not used.
$Col 11	Print segment option - same as comment option.
Col 12	Task Timing Option
Col 13	Real Timing Option • Blank, no timing. • 1, Time each call with standard timer. • 2, Time each call with high resolution timer.
Col 14,15	Additional print option override. Additional print options are normally established from columns 3 and 4 of the compare statements. See "Compare Statement Format (PCB Compare)" in this chapter for details. If non-blank, this overrides columns 3 and 4 of all compare statements.
Col 16-23	This field identifies which PCB in the PSB is used for the subsequent calls. The possibilities are: • blank, the first PCB is used if this is the first status statement; otherwise, the current PCB is used. • DBDNAME, the first PCB found that had this dbdname specified in the DBDNAME= operand is selected. • seven blanks followed by a number; the number indicates the number of the PCB in the PSB. See columns 25-28 for further information.
Col 24	Print options. • 1, do not use print options in this statement. • 2, do not print this status statement. • 3, do not print this status statement or use print options. • Blank, use print options and print this statement.
Col 25-28	PCB processing options and print this statement. Used only when two PCBs have the same name but different processing options. If non-blank, it is used in addition to the DBD name in col 16-23 to select the PCB in the PSB.

If no status statement is read, the default PCB is the first data base PCB in the PSB and the print options are 2 and blank. New status statements may occur at any point in the input stream, to change either the data base to be referenced or the options.

Comment Statement Format

Unconditional Comments:

Col 01	U, identifies this as an unconditional comment statement.
Col 02-80	Comments. Any number of unconditional comments are allowed; they are printed when read. Time and date are printed with each unconditional comment statement.

Conditional Comments:

Col 01	T, identifies this as a conditional comment statement.
Col 02-80	Comments. Up to five conditional comment statements per call are allowed. No continuation in column 72 is required. Printing is conditional and is based on the status statement. No printing is performed before the related calls and optional compare are encountered. However, printing takes place prior to the printing of the next call.

Call Statement Format

$Col 01	L, identifies this as a call statement.
Col 02	Not used.
Col 03	SSA level (optional). The SSA level is normally blank in which case the first call statement fills SSA 1. Each following call statement fills the next lower SSA.
Col 04	Statement format. • U, data starting at column is unformatted, no blanks separate the fields. When '*D' is specified for a pathcall, a 'U' must be specified. • V, Data statement with variable length segment. • M, Second or subsequent segment of concatenated variable length segments. • P, Multiple segments of a path call - data is fixed length.
Col 05-08	Number of times to repeat this call (optional). Normally blank, but if filled, it must be right justified with leading zeros. The identical call is repeated equal to the number in columns 5-8.
$Col 10-13	Call function. The call function is only required on the first SSA of the call.
Col 14,15	Not used.
$Col 16-23	SSA segment name. The segment name is not filled for unqualified calls.
Col 24	Not used.
$Col 25	(, if segment is qualified.
$Col 26-33	SSA field name.
Col 34	Not used.
$Col 35,36	Operator or operators.
Col 37	Not used.
$Col 38-xx	Field value.
$Col xx+1	), end character, or Boolean operator
Col 72	Non-blank if more SSAs.

If the call contains multiple SSAs, put a non-blank character in column 72 of the prior call statement and put the next SSA in the next statement, using the same format. Column 1 and column 10-13 are not required.

If the field value extends past column 71, put a non-blank character in column 72 and CONT in columns 10-13 of the next statement. Continue the field value, starting in column 16. Maximum field value is 256 bytes. Handle Boolean qualifications that extend beyond one statement the same way.

Maximum number of levels for this program is the DL/I limit of 15.

On INSERT or REPLACE calls, data must follow the last (non-continued) call statement with an L in column 1, DATA in columns 10-13, and the segment data in column 16-71. Data may be continued with a non-blank character in column 72 and data starting in column 16 of the next statement. Maximum length of segments is 1500 bytes this may be changed by assembling the program with the field labeled USERSEG altered.

Note: On INSERT calls, the last SSA should have the segment name only with no qualification and may not be continued.

An alternate format for the call statement is available by putting a U in column 4, in which case the exact SSA begins in column 16 with no intervening blanks in columns 24, 34, and 37.

This program arbitrarily calculates the length of the segment displayed on the REPL or ISRT calls as the number of data statements read times 56 (column 72 -16 = 56). However, DL/I uses the proper length.

This program does not check for errors in the call, hence invalid functions, segments, fields, operators, or field lengths are not detected.

Compare Statement Format (PCB Compare)

Col 01	E, identifies this is a compare statement.
Col 02	Blank or H. An H indicates hold compare statement, see below for details.
Col 03	Option requested if compare results are not equal. • 1, request print of I/O buffers. • 2, request DUMP of entire partition. • 4, request PDUMP of DL/I blocks. • 8, abort this step, go to end of job. Combinations, such as 5, 7, or 9 are valid.
Col 04	Buffer printing options. • Blank, print only buffers used in call when printing I/O buffers. Buffer ID is in LEVTTR. • P, print entire buffer pool when printing I/O buffers
Col 05,06	Segment level.
Col 07	Not used.
Col 08,09	Status code.
Col 10	Not used.
Col 11-18	Segment name.
Col 19	Not used.
Col 20-22	Length of feedback key.
Col 23	Blank.
Col 24-XX	Concatenated key feedback.
Col 72	Non-blank to continue key feedback statement.

The compare statement is optional and is normally used only to perform regression testing of known data bases or to call for the printing of DL/I control blocks.

Any fields left blank are not compared to the corresponding fields in the PCB. Blanks are a valid status code; therefore, if the user does not wish to compare a status code, XX should be inserted in columns 08 and 09. To accept any valid status code, that is blank, GA, or GK, put OK in columns 8 and 9. To execute the same compare after each call, put an H in column 2. This simulates that compare being read after each call. This technique is particularly handy when loading - when the blank status code must be checked for blank only. After the compare is done, the current control statement type is E in column 1, so the next control statement read must have its type in column 1 or it will default to E.

The hold compare statement remains in effect until another compare statement is read. If a new compare statement is read, two compares are performed for the preceding call, since the hold compare and optional printing are performed before reading the new compare statement.

SPECIAL CONTROL STATEMENT FORMATS

The special formats shown below, can be used to achieve special testing environments.

Punch Statement Format

$Col 01-03	CTL, identifies this statement type.
$Col 10-14	PUNCH, further identifies this as a punch statement controlling punch output.
Col 15	Blank
$Col 16	Start of keyword parameter controlling various options (parameters separated by commas): • PCBL, produce the full PCB compare statement. • PCBs, produce the PCB compare statement, dropping the key feedback if it exceeds one statement. • DATAL, produce the complete data compare statements. • DATAS, produce only one data compare statement. • OTHER, reproduce all control statements except for compare control statements.

- START=, starting sequence number to be given in columns 73-80. Eight numeric characters must follow the START= parameter, with no leading or trailing zero truncated.
- INCR=, increment to be added to the sequence number of each statement. Four numeric characters must follow the INCR= parameter with no leading or trailing zeros truncated.

PI Support Testing

The following four statements are for testing PI support and are used only when running MPS.

Statement 1

Col 01-02	PI, start PI processing.
Col 10-nn	Task identifier.

Statement 2

Col 01-04	WAIT, wait on the XECB of the task identified in columns 11 to nn.
Col 10	Return code character.
Col 11-nn	Task identifier.

Statement 3

Col 01-04	POST, post the XECB of this task.
Col 10	Return code given on the post.

Statement 4

Col 01-04	SYNC, to synchronize the running of two or more tasks.
Col 10	Return code to be specified (optional).
Col 11-nn	Identifier of the task to synchronize with (required).

Special call statements

Statement 1

Col 01	L, identifies this as a call statement.
Col 10	STAT, print the current buffer pool statistics

Statement 2

Col 01	L, identifies this as a call statement.
Col 05-08	number of times to repeat series of calls
Col 10-13	STAK or END

The purpose of this is to issue multiple iterations of a group of DL/I calls. All control statements between the 'STAK' and 'END' statement are saved in memory and and executed as many times as indicated in columns five through eight of the 'STAK' statement.

Other special statements

Statement 1

Col 01	WTO, puts the message in the remainder of statement on the system console.
Col 10	Message text to a maximum of sixty characters.

Statement 2

Col 01 LOOP, causes program to loop at this point. The current PSW is displayed on the console. To restart Load the PSW to the current PSW + 4.

Statement 3

Col 01 CHKP, causes a DL/I checkpoint call.
Col 10 Eight character checkpoint ID.

Statement 4

Col 01 ABEND, cause program to abnormally terminate and provide a DUMP.

Statement 5

Col 01 TWAIT, issues an uninterruptable WAIT.

Statement 6

Col 01 N or ., disregard this statement.

SUGGESTIONS FOR USING TEST PROGRAM DLZDLTXX

To Display a Data Base: The following sequence of control statements may be used:

```
          1         2
12345678901234567890 12
S 1 2 2 2 1    DBDNAME  Display comments
                        and segment.
L         GN            Do one, get next.
EH8    OK               Hold Compare, GA,
                        GK, OK, terminate
                        on GB.
L   9999 GN             Do 9,999 get next
                        calls.
```

To Load a Data Base: This program normally is used to load very small data bases, since all the call and data statements must be provided to the program rather than data being generated. Of course, it could be used to load large volume data bases if the control statements were generated as a sequential data set.

Following is an example of a job that is used to load a DL/I test database.

```
         1         2         3         4 |       5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345
// JOB TESTEX LOAD DATA BASE
// ASSGN SYS005,3330,VOL=DLIDBP,SHR
// UPSI 00000011
// ASSGN SYS010,UA
// DLBL SHI0B01,'DLI.CHI0B01',,VSAM
// EXTENT SYS005,DLIDBP
// DLBL SSI0B51,'DLI.CSI0B51',,VSAM
// EXTENT SYS005,DLIDBP
// DLBL SSI0B52,'DLI.CSI0B52',,VSAM
// EXTENT SYS005,DLIDBP
// DLBL SSI0B53,'DLI.CSI0B53',,VSAM
// EXTENT SYS005,DLIDBP
// DLBL SHI0BI1,'DLI.CHI0BI1',,VSAM
// EXTENT SYS005,DLIDBP
// EXEC DLZRRC00,SIZE=250K
DLI,DLZDLTXX,PHILB01,1
T • LDB0301 • LOAD DB FOR PHDLB01
S 2 2 2 2 2            12
L         ISRT  A1
L         DATA  A1020M1980M2500
E 5 01     A1        005 A1020
L         ISRT  AA
L         DATA  AA0201    ' M30S192080
E 5 02     AA        011 A1020AA0201
L         ISRT  AAA
L         DATA  XXROOT-020  AAA02012   ;▪  98098
E 5 03     AAA       019 A1020AA0201AAA02012
L         ISRT  AAA
L         DATA  XXROOT-020  AAA02013   ;┼  98096
E 5 03     AAA       019 A1020AA0201AAA02013
                     •
                     •
                     •
L         ISRT  A1
L         DATA  A1999M199999999
E 5 01     A1        005 A1999
L         ISRT  AA
L         DATA  AA9999RRRR 999S199999
E 5 02     AA        011 A1999AA9999
/•
/&
```

To Perform Regression Testing: This program is well suited for performing regression testing. By using a known data base, calls can be issued and the results compared to expected results using compare statements. The program can determine whether DL/I calls are being executed correctly. By making the print options of the status statement all 2s, only those test cases not satisfied properly will be displayed.

To Use As A Debugging Aid: When debugging, the DL/I blocks are usually required. By using the compare statements, the blocks may be displayed whenever they are required. Sometimes the blocks are needed even though the call is executed correctly, for instance, for the call before a failing call. In those cases, an extra incorrect compare statement may be inserted to cause the blocks to be displayed even though the call is executed correctly.

To Verify How A Call Is Executed: Since it is very easy to execute a particular call, this program can be used to verify how a particular call is handled. This can be valuable when DL/I is suspected of not operating correctly in a specific situation. The calls which are suspected of not being executed properly can be issued using this program and the results examined.

Sample DL/I Test Program Job Control Statements: Sample DL/I test program job control statements are shown in Figure 4-16 .

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890123
// JOB JSHISA14 GET CALLS FOR SHISAM1
// UPSI 00000010
// ASSGN SYS010,UA INDICATE NO DISK INPUT
// DLBL IJSYSCT,'AMASTCAT'
// EXTENT SYSCAT,USR992,1,0,20,20
// ASSGN SYS005,X'232'
// DLBL SHISAM1,'CSHISAM1'
// EXTENT SYS005,USR992
// EXEC DLZRRC00,SIZE=250K
DLI,DLZDLTXX,PSHIS1G
S 1 2 2 2 2
U DO GETS FOR SIMPLE HISAM
L         GU    A1111111 (A111111K  = A060000111)
L         GN
E 7 01     A1111111 010 A06000511
L         GN
E 7 01     A1111111 010 A06000611
L         GN
E 7 01     A1111111 010 A06000811
L         GN
E 7 01     A1111111 010 A06000911
L         GN
E 7 01     A1111111 010 A060001211
L         GN
E 7 01     A1111111 010 A060001411
L         GN
E 7 01     A1111111 010 A060002011
L         GN
E 7 01     A1111111 010 A060002311
L         GN
E 7 01     A1111111 010 A060002911
L         GN
E       GB
T DO GHN FOR ALL
L         GHU   A1111111 (2222222  =  ROOTXXX)
E 7 01     A1111111 010 A06000111
L         GHN
E 7 01     A1111111 010 A06000511
L  U      GN    A1111111 (A111111K  > A060000611&A111111K  < A060000911)
E  7 01    A1111111 010 A060000811
L         GU    A1111111 (11111111 =  4TH)
E          A0600000811
L         GNP
E       AD
L         GU    A1111111 (11111111 = 11TH)
E       GE
L         GU    A1111111 (11111111 = 8TH)
E       AC
L         GU
E          A060000111
L         GU    A1111111 (A111111K NG A060000111)
E       AJ
L         BAD   A1111111 (A111111K =  A060000111)
E       AD
T END OF GET CALLS
/*
```

Figure 4-16. Sample DL/I Test Program Job Control Statements

CHAPTER 5. INTERACTIVE FACILITIES

DL/I provides two interactive facilities: the Interactive Macro Facility (IMF) and Interactive Utility Generation Facility (IUG). IMF and UGF offer easy-to-use interactive procedures that let you perform resource definition and utility functions at a 3270-type terminal.

This manual assumes that you are familiar with the Interactive System Productivity Facility (ISPF) program product (5668-960). This program product establishes the DL/I definitional environment, displays IMF and IUG panels, and provides the interface to the user's VSE, VM/VSE, or VM/CMS system. IMF and IUG use ISPF conventions. See VSE System IPO/Extended Planning Guide, GC20-1875, for ISPF documentation.

IMF and IUG are invoked through the DL/I Interactive Functions menu, DLZ0. The relationship between the IMF and IUG interactive functions is shown in Figure 5-1.

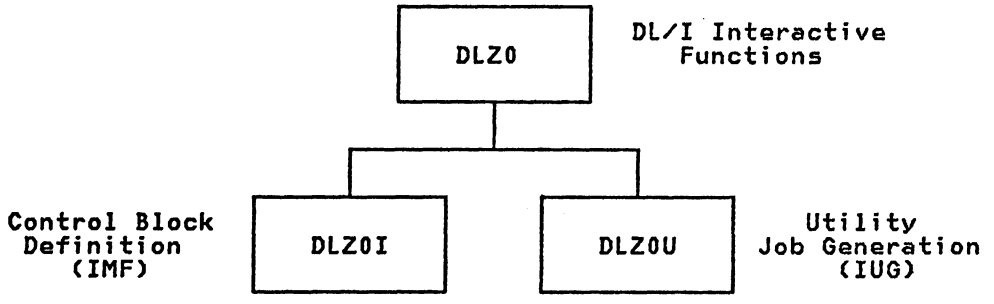


Figure 5-1. DL/I Interactive Functions

This chapter is divided into two sections. The Interactive Macro Facility (IMF) is described in the first part; the Interactive Utility Generation Facility (IUG) is described in the second.

INTERACTIVE MACRO FACILITY (IMF)

This section contains:

- An overview of IMF
- Module flow diagrams for DBD, PSB, and ACT data gathering, and ELIAS-I DBD and PSB table migration
- Default tables which supply initial values for DBD, PSB, and ACT tables
- Formats of the DBD, PSB, and ACT tables built by IMF and used as repositories for user-entered data for the most recently defined control block
- An example of a VSE system job stream for generating a DBD control block.

IMF OVERVIEW

IMF offers easy-to-use interactive procedures that let you create, modify, and delete DL/I control block definitions (DBD, PSB, and ACT) at a 3270-type terminal. Through a series of formatted display panels, IMF prompts you to enter required information about the control blocks interactively. IMF also allows you to migrate (to IMF format) DL/I control blocks that were initially defined using the ELIAS program product.

IMF works with several components: Interactive System Productivity Facility (ISPF), VSE/ICCF, and the VSE system as shown in Figure 5-2 ; also, VM and CMS can be optionally used.

Many IMF panel data entry items are initialized with information from the default table or with the data that was saved in the definition table the last time it was displayed. The 3270 keyboard is used to modify the IMF panel data entry items. When the required information is entered in an IMF panel on the 3270 terminal, pressing the ENTER key causes the information to be transferred to the appropriate definition table and the next panel to be displayed. When the DBD, PSB, or ACT definition is complete, the information that was accumulated in the definition tables is used to generate DBD, PSB, or ACT assembler input. DL/I control blocks are created from the assembly.

When IMF panels are displayed, some of the data entry fields will have values already entered. These values are obtained from the default table for the DBD, PSB, or ACT. (Refer to the description of the DBD, PSB, or ACT default tables later in this section.) If these default values are not what you wanted, new values can be entered in the IMF panel location. Entering information in one of these IMF panel locations causes the default table to be updated.

Skeletons are preliminary generated output information. During generation, they are used in creating the generated utility output.

The ISPF error log is used to contain information about abnormal conditions.

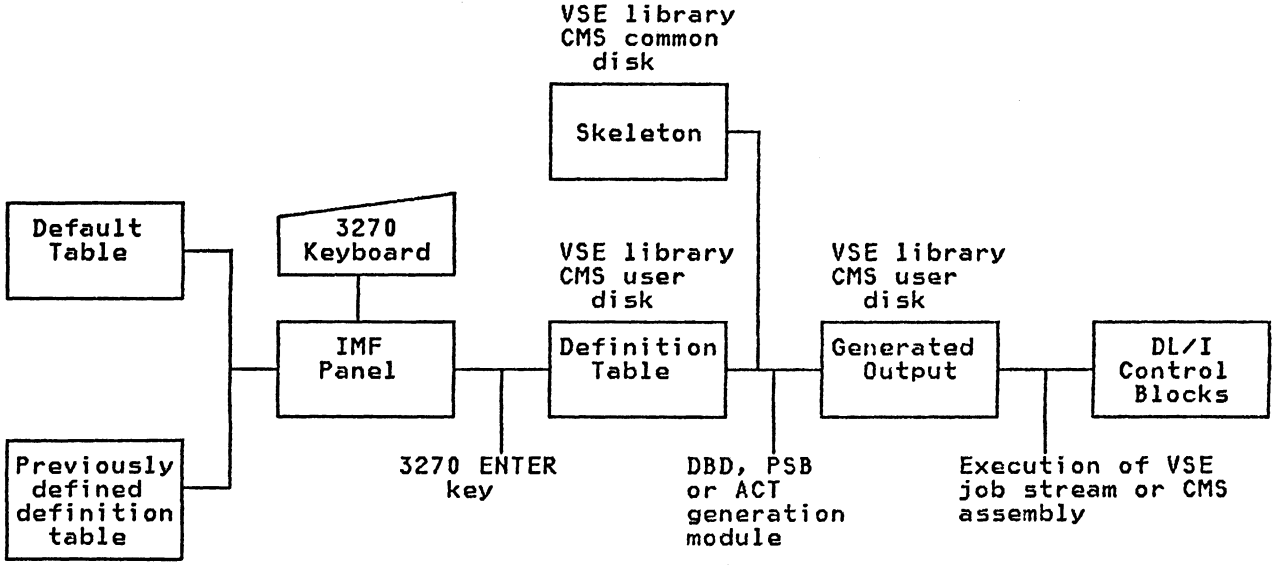


Figure 5-2. IMF Overview

IMF MODULES

Figure 5-3 through Figure 5-6 show how the IMF modules are related and give the module name for all routines. The first name in a box is the module name. The name(s) below the module name is the name of the panel(s) displayed by that module. When modules are listed in columns, the top module calls only one module in the column for a given definition. For example, in Figure 5-3, the top module (DLZ0DBD) calls either DLZ0SE1, DLZ0SE2, or DSZ0SE3, depending on the type of SEGM statement being defined which is determined by the ACCESS parameter in the DBD macro.

DBD Module Flow

Figure 5-3 shows control flow for DBD generation.

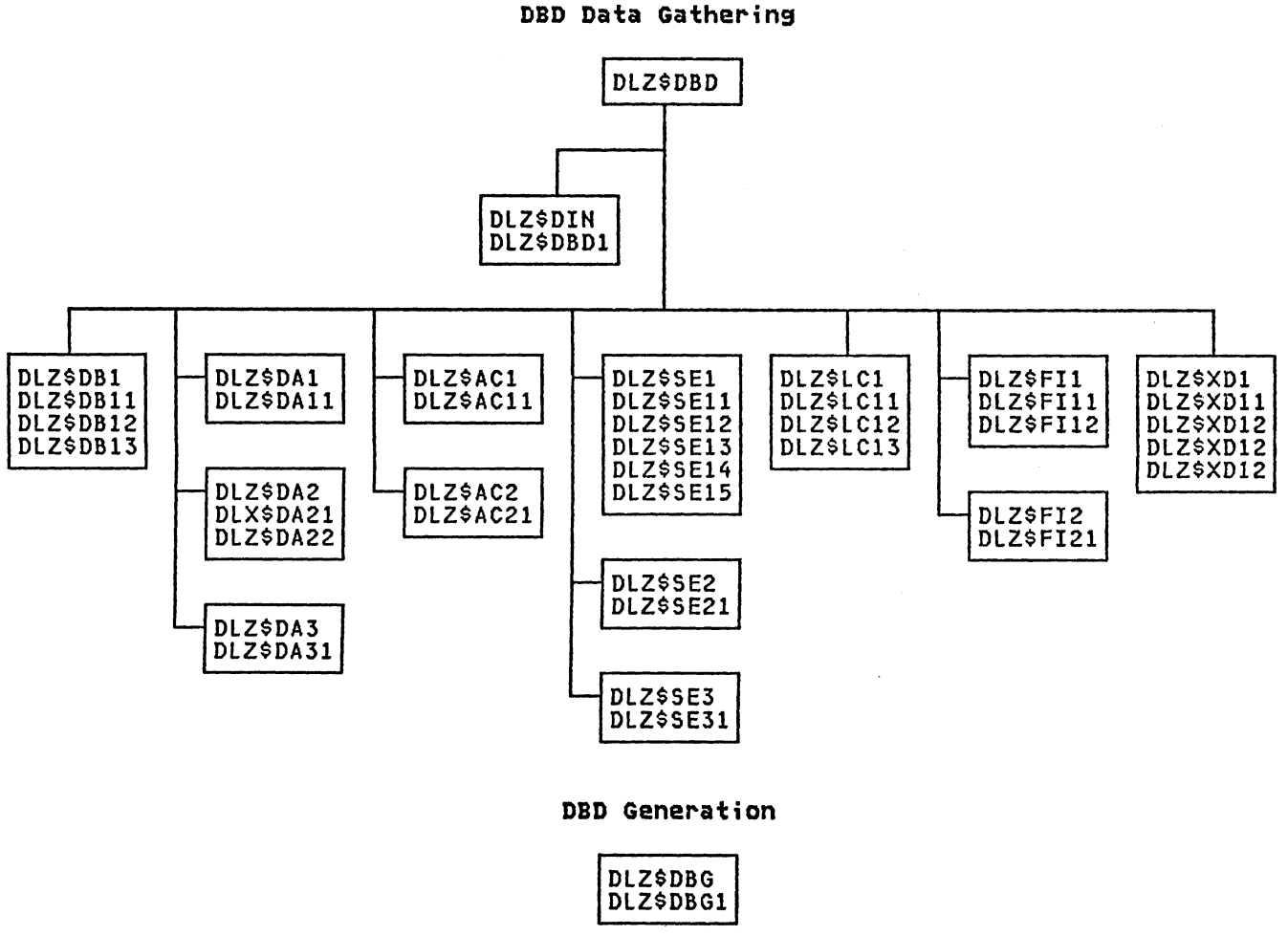


Figure 5-3 (Part 1 of 2). DBD Module Flow

Module	Function
DLZ$DBD	Top function for DBD Data Gathering
DLZ$DIN	DBD Initialization subroutine
DLZ$DB1	Function for DBD statement
DLZ$DA1	Function for DATASET statement for HD
DLZ$DA2	Function for DATASET statement for (S)HISAM and INDEX
DLZ$DA3	Function for DATASET statement for (S)HSAM
DLZ$AC1	Function for ACCESS statement for primary access
DLZ$AC2	Function for ACCESS statement for secondary index
DLZ$SE1	Function for SEGM statement for HD
DLZ$SE2	Function for SEGM statement for HS and INDEX
DLZ$SE3	Function for SEGM statement for LOGICAL
DLZ$LC1	Function for LCHILD statement
DLZ$FI1	Function for FIELD statement for HD
DLZ$FI2	Function for FIELD statement for HS and INDEX
DLZ$XD1	Function for XDFLD statement
DLZ$DBG	Function for DBDGEN

Skeleton	Function
DLZ$DBGA	Skeleton for DBD statement
DLZ$DBGB	Skeleton for DATASET statement
DLZ$DBGC	Skeleton for SEGM statement
DLZ$DBGD	Skeleton for LCHILD statement
DLZ$DBGE	Skeleton for FIELD statement
DLZ$DBGF	Skeleton for XDFLD statement
DLZ$DBGG	Skeleton for ACCESS statement
DLZ$DBGX	Skeleton for starting job control statements
DLZ$DBGY	Skeleton for ending job control statements

Figure 5-3 (Part 2 of 2). DBD Module Flow

PHASE GROUPING: All DBD modules except DLZ0DBG are called by phase DLZ0DBD. The DLZ0DBG module is in the phase DLZ0DBG.

SAVE AREAS: When returning from the initialization routine, information is returned in a save area called DLZ0SVIR. The information is:

Name - name of the DBD being defined

VAS=0 - if new data is being gathered

=1 - if data is being modified

SKP=0 - start at first macro

=1 - start after first macro

V2 - name of the macro if not starting at the first

ACCESS - the access method

The top module uses the save area DLZ0SAVE to pass information to the subroutine. The information is:

NAME - name of the DBD being defined

VAS=0 - if new data is being gathered

=1 - if data is being modified

ACCESS - the access method

ROOT - tells SEGM that it is a root segment

The subroutine uses the save area DLZ0SAVE to pass information to the top module. The information is:

MACRO - the name of the next macro

NEXT - an indication of what to do next

ACCESS - the access method

CREATING TABLES: The top module creates if necessary and updates the default table. It also saves the information in the definition table. The initialization subroutine creates the definition table and creates or updates the LIFO list.

PSB Module Flow

Figure 5-4 shows control flow for PSB generation.

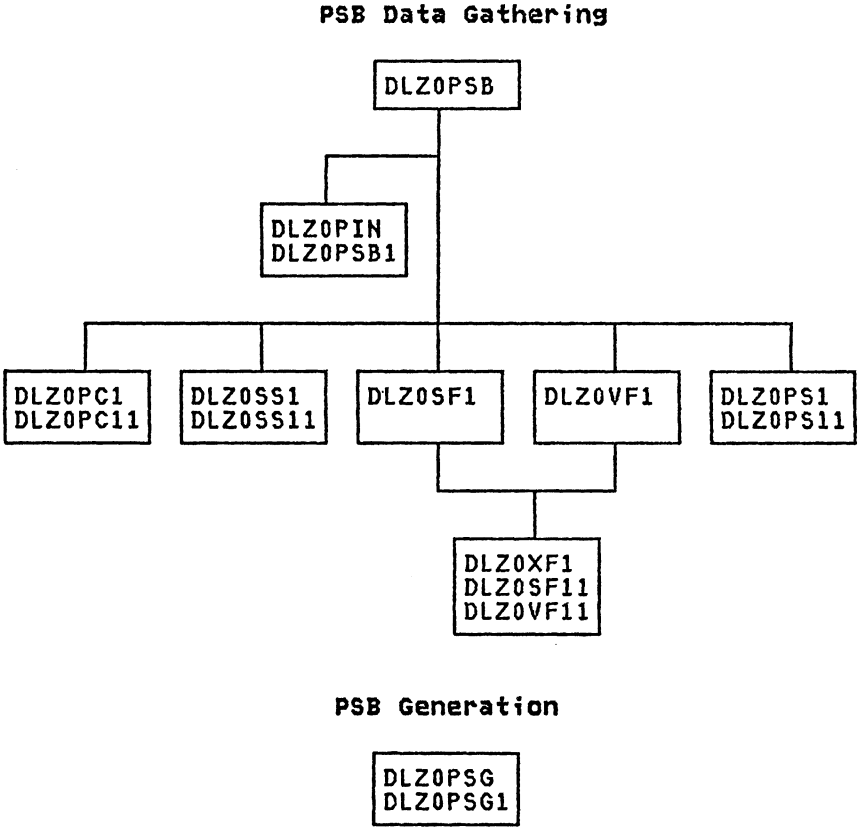


Figure 5-4 (Part 1 of 2). PSB Module Flow

Module	Function
DLZOPSB	Top function for PSB Data Gathering
DLZOPIN	PSB Initialization subroutine
DLZOPC1	Function for PCB statement
DLZOSS1	Function for SENSEG statement
DLZOSF1	Function for SENFLD statement
DLZOVF1	Function for VIRFLD statement
DLZOXF1	Common routine for DLZOSF1 and DLZOVF1
DLZOPS1	Function for PSBGEN statement
DLZOPSG	Function for PSBGEN

Skeleton	Function
DLZOPSGA	Skeleton for PCB statement
DLZOPSGB	Skeleton for SENSEG statement
DLZOPSGC	Skeleton for SENFLD statement
DLZOPSGD	Skeleton for VIRFLD statement
DLZOPSGE	Skeleton for PSBGEN statement and ending JCL
DLZOPSGF	Skeleton for Block Builder and ending job control statements
DLZOPSGX	Skeleton for starting job control statements

Figure 5-4 (Part 2 of 2). PSB Module Flow

PHASE GROUPING: All PSB modules except DLZOPSG are called by phase DLZOPSB. The DLZOPSG module is in the phase DLZOPSG.

SAVE AREAS: When returning from the initialization routine, information is returned in a save area called DLZOSVIR. The information is:

Name - name of the PSB being defined

VAS=0 - if new data is being gathered

=1 - if data is being modified

SKP=0 - start at first macro

=1 - start after first macro

V2 - name of the macro if not starting at the first

The top module uses the save area DLZOSAVE to pass information to the subroutine. The information is:

NAME name of the PSB being defined

VAS=0 - if new data is being gathered

=1 - if data is being modified

The subroutine uses the save area DLZOSAVE to pass information to the top module. The information is:

MACRO - the name of the next macro

NEXT - an indication of what to do next

CREATING TABLES: The top module creates if necessary and updates the default table. It also saves the information in the definition table. The initialization subroutine creates the definition table and creates or updates the LIFO list.

ACT Module Flow

Figure 5-5 on page 5-8 shows control flow for ACT generation.

PHASE GROUPING: All ACT modules except DLZ0ACG are called by phase DLZ0ACT. The DLZ0ACG module is in the phase DLZ0ACG.

SAVE AREAS: When returning from the initialization routine, information is returned in a save area called DLZ0SVIR. The information is:

Name - name of the ACT being defined

VAS=0 - if new data is being gathered

=1 - if data is being modified

SKP=0 - start at first macro

=1 - start after first macro

V2 - name of the macro if not starting at the first

The top module uses the save area DLZ0SAVE to pass information to the subroutine. The information is:

NAME - name of the ACT being defined

VAS=0 - if new data is being gathered

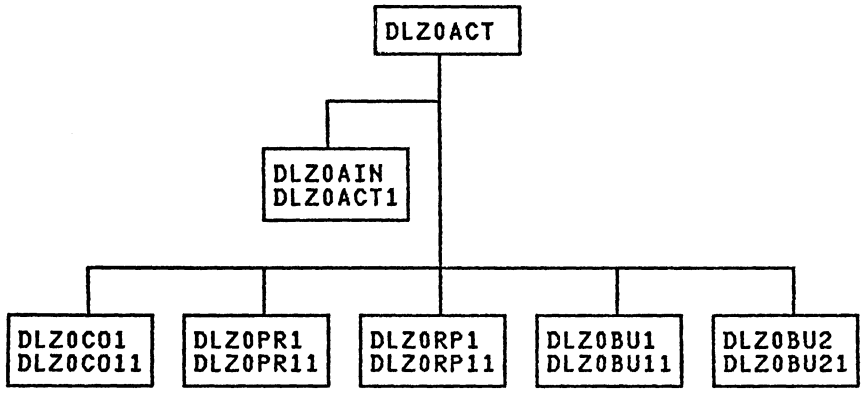
=1 - if data is being modified

The subroutine uses the save area DLZ0SAVE to pass information to the top module. The information is:

MACRO - the name of the next macro

NEXT - an indication of what to do next

ACT Data Gathering



ACT Generation

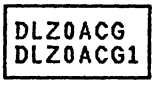


Figure 5-5 (Part 1 of 2). ACT Module Flow

Module	Function
DLZ0ACT	Top function for ACT Data Gathering
DLZ0AIN	ACT Initialization subroutine
DLZ0CO1	Function for TYPE=CONFIG statement
DLZ0PR1	Function for TYPE=PROGRAM statement
DLZ0RP1	Function for TYPE=RPSB statement
DLZ0BU1	Function for TYPE=BUFFER statement for HDBFR
DLZ0BU2	Function for TYPE=BUFFER statement for HSBFR
DLZ0ACG	Function for ACTGEN

Skeleton	Function
DLZ0ACGA	Skeleton for TYPE=INITIAL statement
DLZ0ACGB	Skeleton for TYPE=CONFIG statement
DLZ0ACGC	Skeleton for TYPE=PROGRAM statement
DLZ0ACGD	Skeleton for TYPE=RPSB statement
DLZ0ACGE	Skeleton for TYPE=BUFFER statement for HDBFR
DLZ0ACGF	Skeleton for TYPE=BUFFER statement for HSBFR
DLZ0ACGX	Skeleton for starting job control statements
DLZ0ACGY	Skeleton for ending job control statements

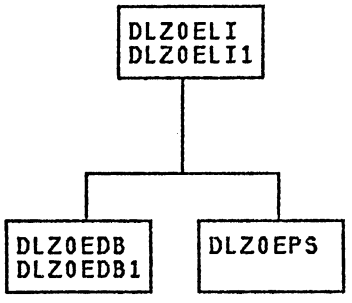
Figure 5-5 (Part 2 of 2). ACT Module Flow

CREATING TABLES: The top module creates if necessary and updates the default table. It also saves the information in the definition table. The initialization subroutine creates the definition table and creates or updates the LIFO list.

ELIAS Table Migration Module Flow

Figure 5-6 shows control flow for migrating ELIAS tables.

ELIAS Table Migration



Module	Function
DLZ0ELI	Top function for ELIAS to IMF migration
DLZ0EDB	Function for migrating DBDs
DLZ0EPS	Function for migrating PSBs

Figure 5-6. ELIAS Table Migration Module Flow

IMF CREATED TABLES

IMF creates and maintains several tables for DBDs, PSBs, and ACTs. A DBD definition table, PSB definition table, or ACT definition table is created for each DBD, PSB, or ACT that is defined. In VSE/Advanced Functions, a job stream is set up by the generation procedure to assemble and link-edit the DBD, PSB, or ACT into the core image library. In CMS, the generated output is the input for assembly of the control block. A DBD LIFO

list, PSB LIFO list, or ACT LIFO list is created and maintained by IMF that contains the names of the DBDs, PSBs, or ACTs that are defined using IMF and not deleted through IMF. A DBD default list, PSB default list, or ACT default list is updated each time a DBD, PSB, or ACT is displayed using IMF.

Additional information about the creation of these tables and examples of the tables are given in the "Doing a DBDGEN", "Doing a PSBGEN", or "Doing a ACTGEN" sections of DL/I DOS/VS Interactive Resource Definition and Utilities.

You can display the IMF created tables by using an editor program. The tables are stored as members in the ISPF table output library defined by the label ISPTABL.

DBD Tables

You can display your DBD Definition Table by using an editor program. The member name is the dbdname that you assigned to the table. The letter keys in Figure 5-7 and Figure 5-8 show the relationship of the parameters specified in the macros and their locations in the definition table.

Figure 5-7 shows the relationship of the macro (V2) and the parameters (V3 through V26) to the variables in the definition table. Although there is space for 26 variables in each macro, only those variables shown in the figure are valid.

Variable		Macros and Parameters							
NAME	A	dbdname -							
V2	B	DBD	DATASET	SEGM	LCHILD	H	FIELD	XDFLD	ACCESS
V3	C	access	device	name	name	I	name	name	segm
V4	D	rmnane	dd1	parent	db	J	seq	segment	rmrtn
V5	E	anchor	dd2	pcptr	pointer	K	mult	nullval	ref
V6	F	rbn	ovflw	lpseg	index	L	bytes	extrtn	seqval
V7	G	bytes	block	lpdb	rules	M	start	srch1	supval
V8			block2	pointer	pair	N	type	srch2	suprtn
V9			record	ltptr				srch3	seqfld1
V10			record2	bytes1				srch4	seqfld2
V11			scan	bytes1				srch5	seqfld3
V12			frspc1	comprt1				subseq1	seqfld4
V13			frspc2	comprt2				subseq2	seqfld5
V14			devaddr1	rules				subseq3	seqseg
V15			devaddr2	where				subseq4	
V16				source1				subseq5	
V17				source2				ddata1	
V18				source3				ddata2	
V19				source4				ddata3	
V20								ddata4	
V21								ddata5	
V22									
V23									
V24									
V25									
V26									

Figure 5-7. DBD Definition Table Variables

Figure 5-8 on page 5-11 is an example of a job stream set up for DBDGEN and saved in a file for submission to VSE/POWER by the user. The VSE/ICCF SUBMIT function is used to send the job stream to VSE/POWER. For CMS, set up macro libraries (MACLIB) and assemble the generated macro. Another example of a DBD generated output appears in the section "Doing a DBDGEN" in DL/I DOS/VS Interactive Resource Definition and Utilities. You can display the generated output by using the system's editor pro-

gram. The filename is the DBD name followed by an at-sign (@). The filename for the job stream shown in Figure 5-8 on page 5-11 is PAYRLDB@.

```
                    A
* $$ JOB JNM=PAYRLDB@,CLASS=0
// JOB PAYRLDB DBD GENERATION
// OPTION CATAL  A
// EXEC ASSEMBLY
     DBD                D     E  F   G                                    *
      B          RMNAME=(RANDMOD2,1,300,620),                             *
                 NAME=PAYRLDB,  A                                         *
                 ACCESS=HDAM   C
     DATASET                                                              *
                 DD1=PAYROLL,                                             *
                 BLOCK=(1024),                                            *
                 SCAN=3,                                                  *
                 DEVICE=3330
     SEGM                                                                 *
                 NAME=NAME,                                               *
                 BYTES=150,                                               *
                 POINTER=TB,                                              *
                 PARENT=0
     FIELD                                                                *
       H         BYTES=60,   L                                            *
                 START=1,   M                                             *
                 TYPE=C,   N                                              *
                 NAME=(EMPLOYEE,SEQ,U)
     FIELD                    I     J  K                                  *
                 BYTES=15,                                                *
                 START=61,                                                *
                 TYPE=C,                                                  *
                 NAME=MANNBR
     FIELD                                                                *
                 BYTES=75,                                                *
                 START=76,                                                *
                 TYPE=C,                                                  *
                 NAME=ADDR
     SEGM                                                                 *
                 NAME=ADDRESS,                                            *
                 POINTER=TB,                                              *
                 PARENT=((NAME,DBLE))
     FIELD                                                                *
                 BYTES=100,                                               *
                 START=1,                                                 *
                 TYPE=C,                                                  *
                 NAME=HOMEADDR
     FIELD                                                                *
                 BYTES=100,                                               *
                 START=101,                                               *
                 TYPE=C,                                                  *
                 NAME=COMAILOC
     FIELD                                                                *
                 BYTES=80,                                                *
                 START=COMAILOC,                                          *
                 TYPE=C,                                                  *
                 NAME=DEPTNAME
     FIELD                                                                *
                 BYTES=10,                                                *
                 NAME=DEPTNO
     FIELD                                                                *
                 BYTES=10,                                                *
                 NAME=BLDGNO
```

Figure 5-8 (Part 1 of 2). Example of a VSE Environment DBD Job Stream

```
    SEGM                                                              •
                NAME=PAYROLL,                                         •
                BYTES=100,                                            •
                POINTER=TB,                                           •
                PARENT=((NAME,DBLE))
    FIELD                                                             •
                BYTES=15,                                             •
                START=51,                                             •
                TYPE=P,                                               •
                NAME=HOURS
    FIELD                                                             •
                BYTES=15,                                             •
                START=1,                                              •
                TYPE=P,                                               •
                NAME=BASICPAY
    DBDGEN
    FINISH
    END
/•
// EXEC LNKEDT
/&
• $$ EOJ
```

Figure 5-8 (Part 2 of 2). Example of a VSE Environment DBD Job Stream

Figure 5-9 shows the relationship of the parameters of the macro to the variables in the default table. Although there is space for 17 variables only those variables shown in the figure are valid. All default locations for a DBD are updated each time a default is changed.

Variable	Macro	Parameter	Initial Value
D1	DBD	ACCESS	none
D2	DBD	ANCHOR	none
D3	DATASET	DEVICE	none
D4	DATASET	SCAN	none
D5	DATASET	FRSCP1	none
D6	DATASET	FRSPC2	none
D7	SEGM	POINTER1	TB
D8	SEGM	COMP-INIT	none
D9	SEGM	RULES	LLL
D10	SEGM	WHERE	LAST
D11	DATASET	DEVIC5-HSAM	none
D12	GEN	ENVIRON	1
D13	LCHILD	RULES	LAST
D14	DBD	IMSCOMP	NO
D15			
D16			
D17			

Figure 5-9. DBD Default Table Variables

PSB Tables

The definition table and generated output for a PSB are similar to the ones created for the DBD. Figure 5-10 on page 5-13 shows the relationship of the macro (V2) and the parameters (V3 through V8) to the variables in the definition table. Although there is space for 8 variables in each macro, only those variables shown in the figure are valid. You can display the definition table by using the system's editor program. The filename is the psbname. An example of a PSB job stream appears in the section "Doing a PSBGEN" in DL/I DOS/VS Interactive Resource

Definition and Utilities. You can display the generated output by using the system's editor program. The filename is the PSB name followed by an at-sign (@).

A LIFO list of all PSBs defined using IMF and not deleted through IMF is created similar to the DBD LIFO list. An example of a PSB LIFO list appears in the section "Doing a PSBGEN" in DL/I DOS/VS Interactive Resource Definition and Utilities. You can display the LIFO list by using the system's editor program. The filename is DLZOTPS1.

A PSB default table with information updated to the value of the last defined PSB is maintained by IMF. The default table is similar to the one created for the DBD. Figure 5-11 shows the relationship of the parameters of the macro to the variables in the default table. Although there is space for 8 variables only those variables shown in the figure are valid. ALL default locations for a macro are updated each time the macro using that parameter is displayed. You can display the default table by using the system's editor program. The member name is DLZOTDP.

Variable	Macros and Parameters				
NAME	psbname - - - - - - - - - - - - - - - - - -				
V2	PCB	SENSEG	SENFLD	VIRFLD	PSBGEN
V3	dbdname	name	name	name	lang
V4	procopt	parent	bytes	bytes	
V5	keylen	procopt	start	start	
V6	pos		type	type	
V7	procseq		rtname	rtname	
V8			replace	value	

Figure 5-10. PSB Definition Table Variables

Variable	Macro	Parameter	Initial Value
D1	PCB	POS	S
D2	PSBGEN	LANG	none
D3	GEN	ENVIR	1
D4	GEN	DMB	none
D5			
D6			
D7			
D8			

Figure 5-11. PSB Default Table Variables

ACT Tables

The definition table and generated output for an ACT are similar to the ones created for the DBD. Figure 5-12 on page 5-14 shows the relationship of the macro (V2) and the parameters (V3 through V17) to the variables in the definition table. Although there is space for 17 variables in each macro, only those variables shown in the figure are valid. You can display the definition table by using the system's editor program. The filename is DLZNUCxx, where xx is the suffix. An example of an ACT job stream appears in the section "Doing a ACTGEN" in DL/I DOS/VS

Interactive Resource Definition and Utilities. You can display the generated output by using an editor program. The member name is DLZ followed by the suffix and two at-signs (DLZxx@@, where xx is the suffix).

Variable	Macros and Parameters					
NAME	actname -					
					BUFFER	
V2	INITIAL	CONFIG	PROGRAM	RPSB	HDBFR	HSBFR
V3		maxtask	pgmname	psb	bufno	indno
V4		cmaxtsk	actname1	sysid	subpool	ksdsbuf
V5		bfrpool	actname2	rname	dbdname1	esdsbuf
V6		pass	actname3	lang	dbdname2	dbdname
V7		slc	actname4	name	dbdname3	
V8		pi	actname5		dbdname4	
V9		remote			bdbname5	
V10						
V11						
V12						
V13						
V14						
V15						
V16						
V17						

Figure 5-12. ACT Definition Table Variables

A LIFO list of all ACTs defined using IMF and not deleted through IMF is created similar to the DBD LIFO list. An example of a ACT LIFO list appears in the section "Doing a ACTGEN" in DL/I DOS/VS Interactive Resource Definition and Utilities. You can display the LIFO list by using an editor program. The filename is DLZOTAC1.

An ACT default table with information updated to the value of the last defined ACT is maintained by IMF. The default table is similar to the one created for the DBD. Figure 5-13 on page 5-15 shows the relationship of the parameters of the macro to the variables in the default table. Although there is space for 17 variables only those variables shown in the figure are valid. All default locations for a macro are updated each time the macro using that parameter is displayed. You can display the default table by using an editor program. The filename is DLZOTDA.

Variable	Macro	Parameter	Initial Value
D1	CONFIG	MAXTASK	10
D2	CONFIG	CMAXTSK	10
D3	CONFIG	PASS	DLZPASS1
D4	CONFIG	PI	YES
D5	CONFIG	REMOTE	NO
D6	CONFIG	BFRPOOL	none
D7	BUFFER	HDBFR	32
D8	GEN	ENVIRON	1
D9			
D10			
D11			
D12			
D13			
D14			
D15			
D16			
D17			

Figure 5-13. ACT Default Table Variables

Migrated ELIAS Tables

An ELIAS table (either DBD or PSB) is first reformatted by IMF to create a temporary table called DLZ0@. The data is then saved in an IMF table and the temporary table deleted. The name of the new IMF table is the same as the ELIAS name for the particular table. This causes the ELIAS job control skeleton to be overlaid in ICCF. However, the IMF table and the ELIAS table exist at the same time because ELIAS adds a suffix to the DBD and PSB names (D and P, respectively), when saving data in the table.

ELIAS tables migrated by IMF have the same format as a table initially created by IMF.

IMF TABLES SUMMARY

The tables that are created by IMF can be displayed to verify their content. The tables are displayed by using the an editor program. Figure 5-14 on page 5-16 shows the filename and filetype of the IMF tables and gives a brief description of the tables.

Note: The table input and output libraries must both have the same names.

Member Name			Description
DBD	PSB	ACT	
dbdname	psbname	DLZNUCxx[1],[2]	Definition table for DBD, PSB or ACT that is created at DBD, PSB, or ACT definition. This member is the input for generation.
DLZOTDD	DLZOTDP	DLZOTDA[2]	Default table that is created or updated at DBD, PSB, or ACT definition.
DLZOTDB1	DLZOTPS1	DLZOTAC1[2]	LIFO list - includes names of defined DBD, PSB, or ACT that were created or updated at DBD, PSB, or ACT definition
DBDNAME@	PSBNAME@	DLZxx@@[1],[3]	Generated output created at DBD, PSB, or ACT generation time.

[1] xx is the suffix.
[2] The member names in the first three rows are in the MACLIB defined for ISPTLIB and ISPTABL.
[3] The member names in the last row are stored in the MACLIB defined for ISPFILE.

Figure 5-14. Displaying IMF Tables

ERROR CONDITIONS

IMF Error Detecting

If information having incorrect syntax (such as a value outside the acceptable range of choices) is entered, an error message appears on the data entry panel so that the information can be corrected before it is entered into the definition table. The error messages are described in the "IMF Error Messages" section of DL/I DOS/VS Interactive Resource Definition and Utilities. Because IMF does not check for all other possible error conditions, the output listings should be carefully checked, the errors corrected, and the output regenerated before attempting to use the IMF panel defined files for defining the data base.

Abnormal Ends

If the system abnormally ends after data has been entered for at least one macro but before the user entered "STOP" as the next thing to do, the definition table may have one row in it with the name variable containing "|". This is a position keeper and it will be deleted the next time the definition table is specified. It does no harm and will not affect the generated output.

INTERACTIVE UTILITY GENERATION FACILITY (IUG)

This section contains:

- An overview of IUG
- Module flow diagrams for utility job stream data gathering, ISQL EXTRACT DEFINEs, and associated IUG functions
- Default tables which supply initial values for utility tables
- CONTROL table information built by IUG and used as repositories for user-entered data for the most recently defined utility
- Examples of a VSE system job stream as generated by IUG for each DL/I utility

IUG OVERVIEW

IUG provides the capability to interactively generate job streams for any of the DL/I utilities at a 3270-type terminal. IUG also allows you to generate job streams for ISQL EXTRACT DEFINEs and to review or modify information previously provided for utility job stream generation.

Like IMF, IUG works with several components: Interactive System Productivity Facility (ISPF), VSE/ICCF, and the VSE system as shown in Figure 5-15; also, VM and CMS can be optionally used.

Many IUG panel data entry items are initialized with information from the default table or the data that was saved in a job stream table the last time it was displayed. The 3270 keyboard is used to modify the IUG panel data entry items. When the required information is entered in an IUG panel on the 3270 terminal, pressing the ENTER key causes the information to be transferred to the appropriate job stream data table and the next panel to be displayed. When the utility job stream generation process is complete, the job control statements, in proper execution order, are contained in VSE or CMS files.

When the IUG panels are displayed, some of the data entry fields will have values already entered. These values are obtained from the default table for each utility. (Refer to the descriptions of the utility default tables later in this section.) If these default values are not what you wanted, new values can be entered in the IUG panel location. Entering information in one of these IUG panel locations causes the default table to be updated.

Skeletons are preliminary generated output information. During generation, they are used in creating the generated utility output.

The ISPF error log is used to contain information about abnormal conditions.

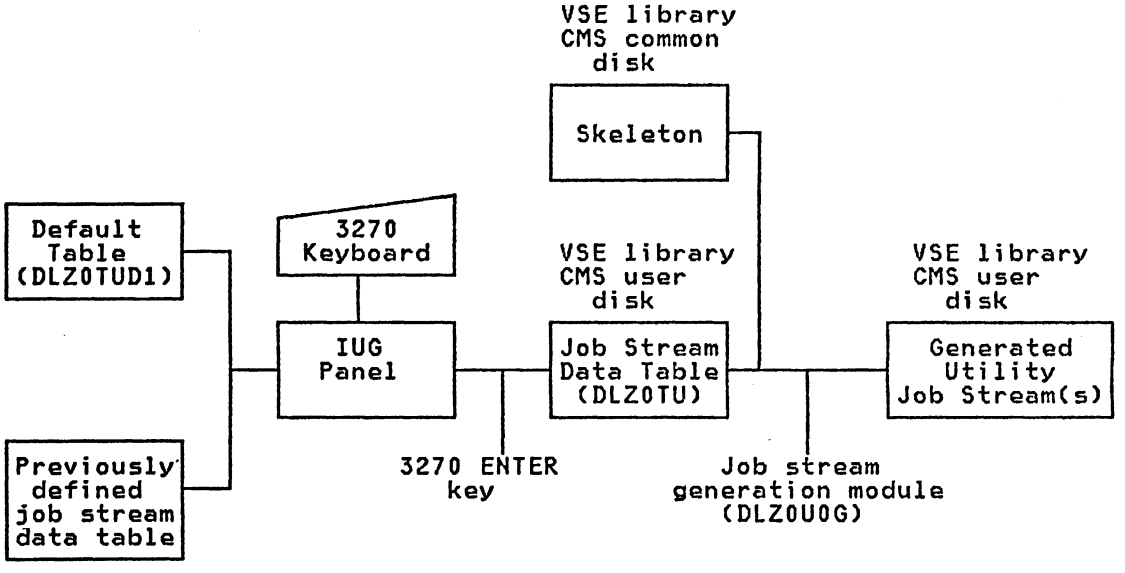


Figure 5-15. IUG Overview

IUG MODULES

Figure 5-16 through Figure 5-32 show how the IUG modules are related and give the module name for all routines. For each functional area, the initial selection menu is also shown to illustrate the relationship between logical IUG activities. The first name in a box is the module name. The name(s) below the module name is the name of the panel(s) (8 characters) displayed by that module. When modules are listed in columns, the control module calls only one module in the column for a given activity. For example, in Figure 5-18, the control module (DLZOU1 or DLZOU2) calls either DLZOU0D, DLZOU0T (to display panel DLZOU0T1) , or DLZOU0T (to display menu panel DLZOU0T3), depending on the type of device on which the WORKFIL resides.

Reorganization/Load Module Flow

Figure 5-16 shows the functions that can be initiated through the Reorganization or Load options of IUG. Job streams will be generated for one or more of these utilities depending upon the option specified on the IUG Reorganization or Load selection menu. Each of these functions is described in the following sections.

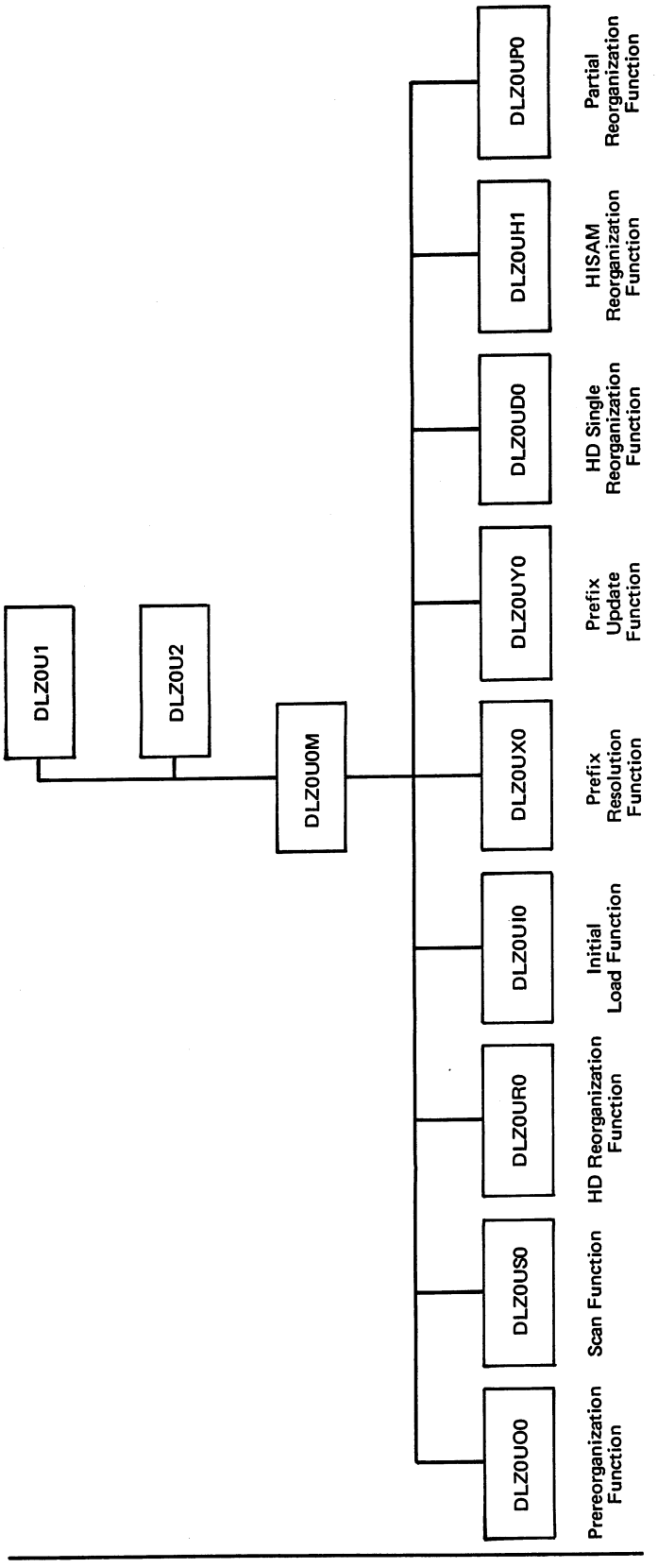


Figure 5-16. Reorganization/Load Utility Functions

PREREORGANIZATION FUNCTION: Figure 5-17 shows the control flow for generating a preorganization utility job stream.

Module	Function
DLZ0U1	Reorganization selection menu
DLZ0U2	Load selection menu
DLZ0U00	Function for prereorganization
DLZ0U0D	Function for CONTROL file DLBL
Skeleton	**Function**
DLZ0U00A	Skeleton for DBD information
DLZ0U00B	Skeleton for STAT information

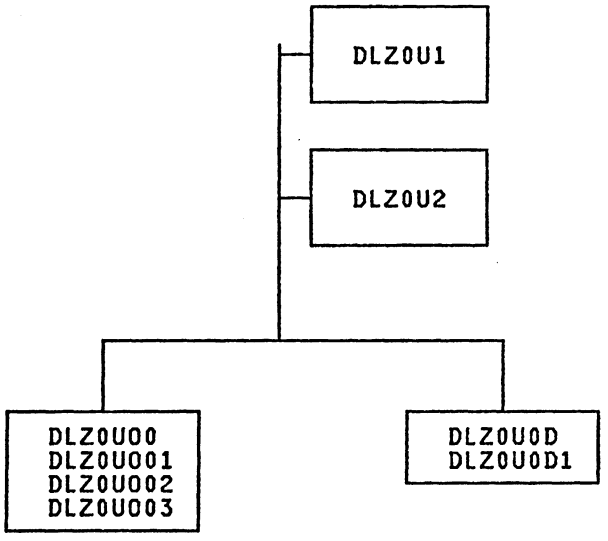


Figure 5-17. Prereorganization Function

SCAN FUNCTION: Figure 5-18 shows the control flow for generating a scan utility job stream.

Module	Function
DLZOU1	Reorganization selection menu
DLZOU2	Load selection menu
DLZOUS0	Function for scan
DLZOU0B	Function for DBR, DBS information
DLZOU0D	Function for disk WORKFIL data
DLZOU0T	Function for tape WORKFIL data
DLZOU0A	Function for tape ASSGN
Skeleton	**Function**
DLZOUS0A	Skeleton for scan utility

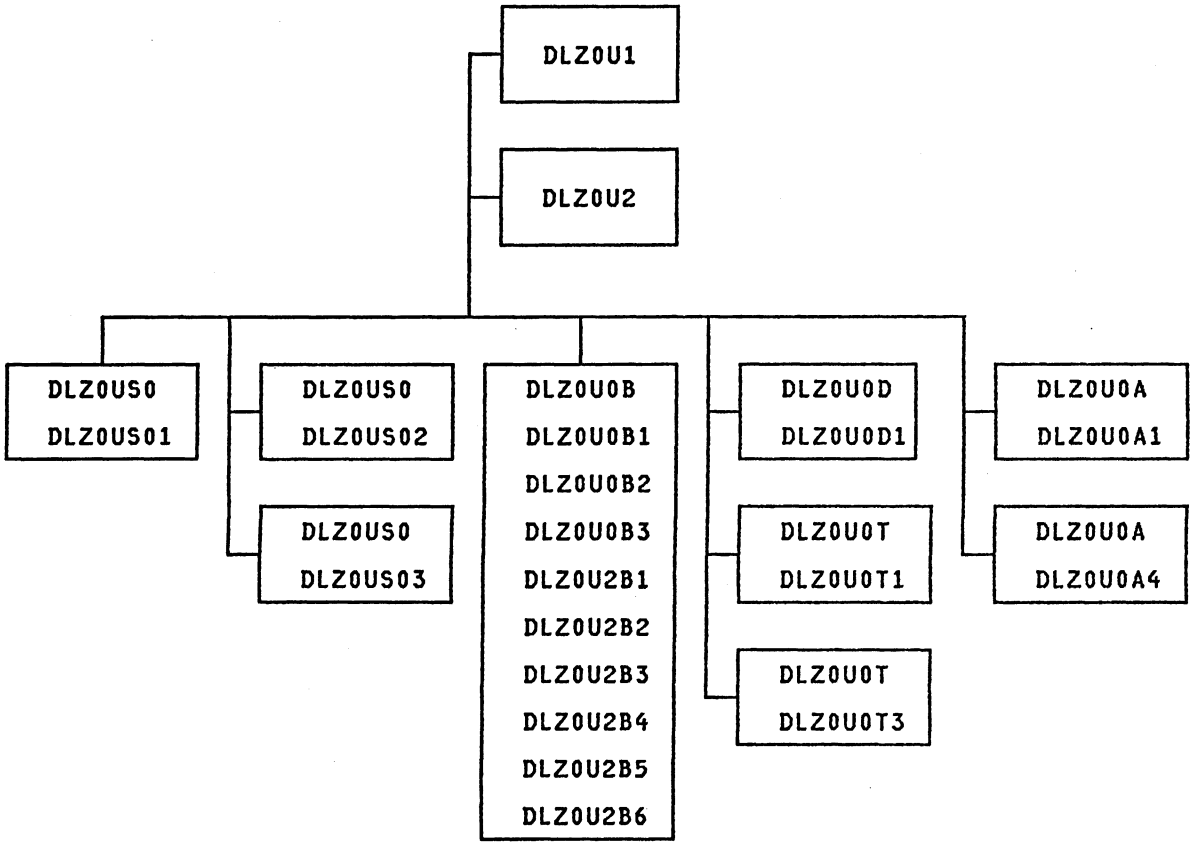


Figure 5-18. Scan Function

HD REORGANIZATION FUNCTION: Figure 5-19 on page 5-24 shows the control flow for generating an HD reorganization utility job stream.

Module	Function
DLZOU1	Reorganization selection menu
DLZOU2	Load selection menu
DLZOUR0	Top function for HD reorganization
DLZOU0B	Function for data base information
DLZOUR1	Function for HD Unload
DLZOU0D	Function for disk HDUNLD1 data
DLZOU0T	Function for tape HDUNLD1 data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk HDUNLD2 data
DLZOU0T	Function for tape HDUNLD2 data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk RESTART data
DLZOU0T	Function for tape RESTART data
DLZOU0A	Function for tape ASSGN data
DLZOU0R	Function for DLZRRC00 information
DLZOUR2	Function for HD Reload
DLZOU0D	Function for disk WORKFIL data
DLZOU0T	Function for tape WORKFIL data
DLZOU0A	Function for tape ASSGN data
DLZOU0R	Function for DLZRRC00 information

Skeleton	Function
DLZOUD0A	Skeleton for HD Unload/Reload

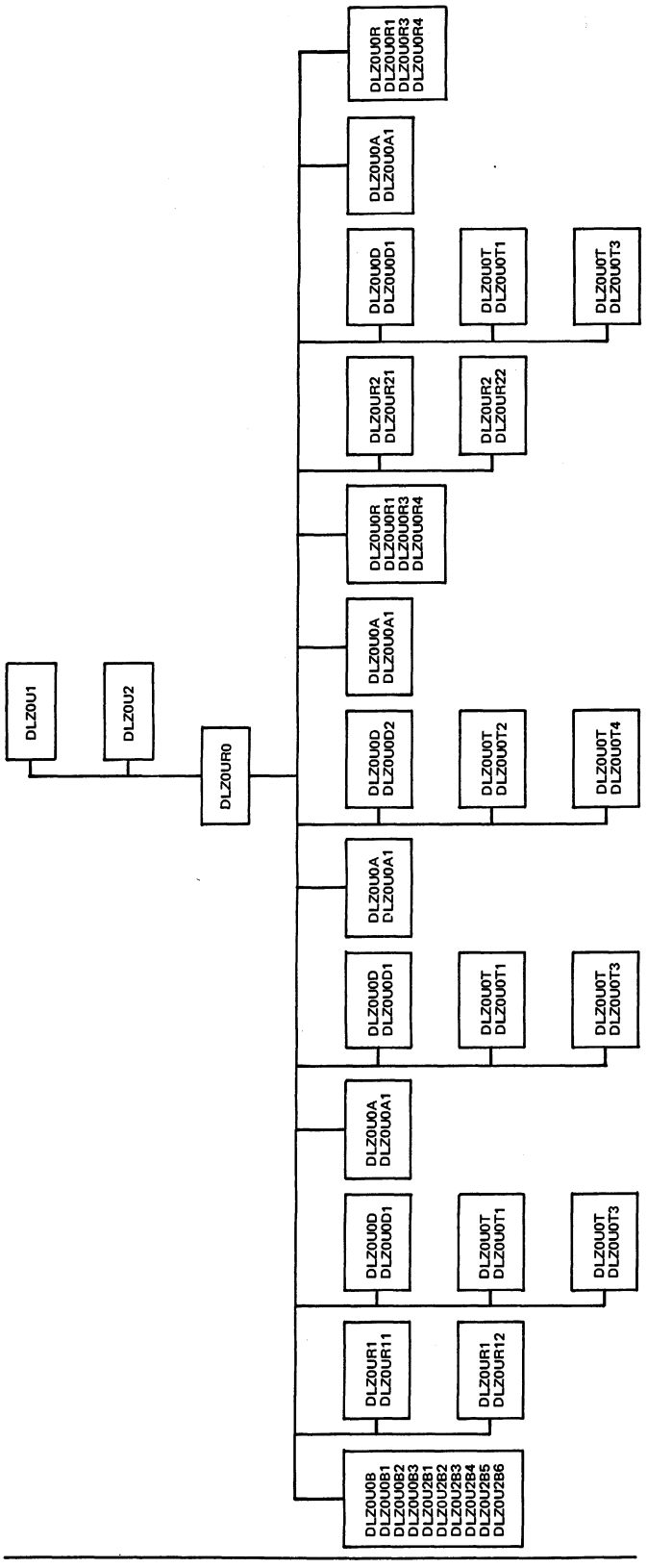


Figure 5-19. HD Reorganization Function

INITIAL LOAD FUNCTION: Figure 5-20 shows the control flow for generating an initial load utility job stream.

Module	Function
DLZOU1	Reorganiztion selection menu
DLZOU2	Load selection menu
DLZOUI0	Function for initial load
DLZOU0A	Function for tape ASSGN data
DLZOU0T	Function for tape input data
DLZOUD	Function for disk WORKFIL data
DLZOU0T	Function for tape WORKFIL data
DLZOU0A	Function for tape ASSGN data
DLZOU0B	Function for data base information
DLZOU0R	Function for DLZRRC00 information

Skeleton	Function
No skeletons are used	

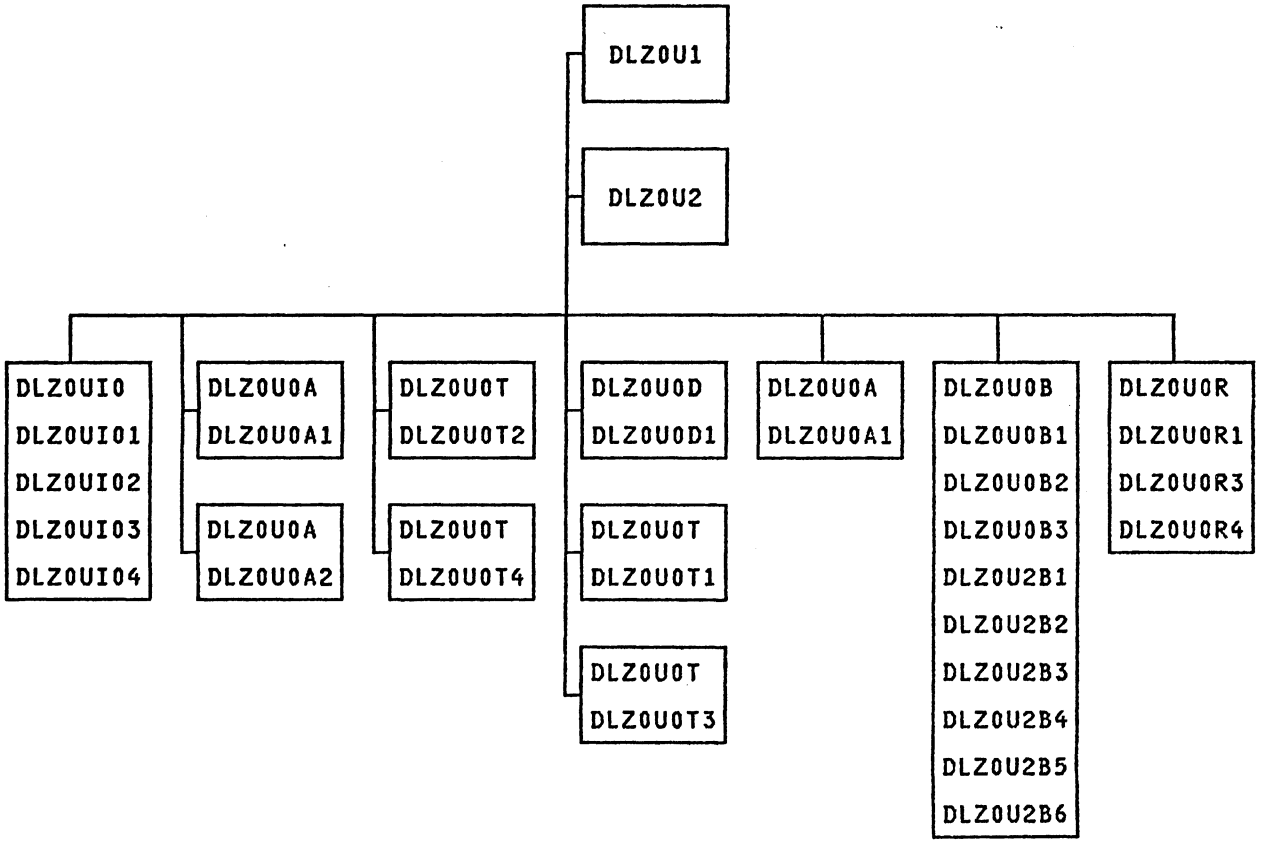


Figure 5-20. Initial Load Function

PREFIX RESOLUTION FUNCTION: Figure 5-21 on page 5-27 shows the control flow for generating a prefix resolution utility job stream.

Module	Function
DLZOU1	Reorganization selection menu
DLZOU2	Load selection menu
DLZOUX0	Function for prefix resolution
DLZOU0S	Function for sort work files
DLZOU0N	Function for save sort data
DLZOU0A	Function for WRKIN ASSGN data
DLZOU0D	Function for disk INTRMED data
DLZOU0T	Function for tape INTRMED data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk WORKFIL data
DLZOU0T	Function for tape WORKFIL data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk INDXWRK data
DLZOU0T	Function for tape INDXWRK data
DLZOU0A	Function for tape ASSGN data

Skeleton	Function
DLZOUX0A	Skeleton for prefix resolution utility

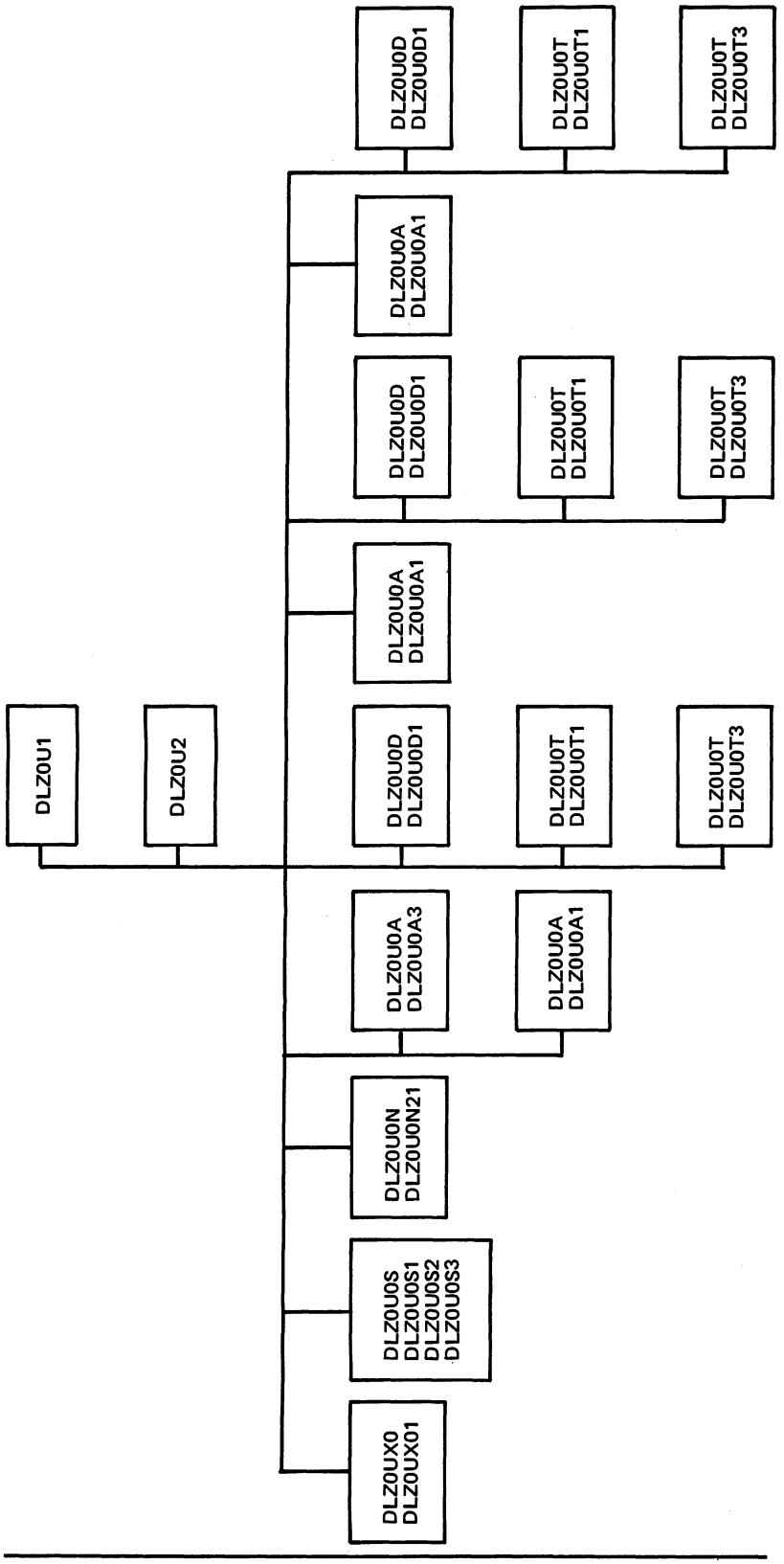


Figure 5-21. Prefix Resolution Function

PREFIX UPDATE FUNCTION: Figure 5-22 shows the control flow for generating a prefix update utility job stream.

Module	Function
DLZ0U1	Reorganization selection menu
DLZ0U2	Load selection menu
DLZ0UY0	Function for prefix update
DLZ0U0B	Function for data base information
DLZ0U0A	Function for tape ASSGN data
DLZ0U0T	Function for tape INDXWRK data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0T	Function for tape WORKFIL data

Skeleton	Function
DLZ0UY0A	Skeleton for prefix update utility

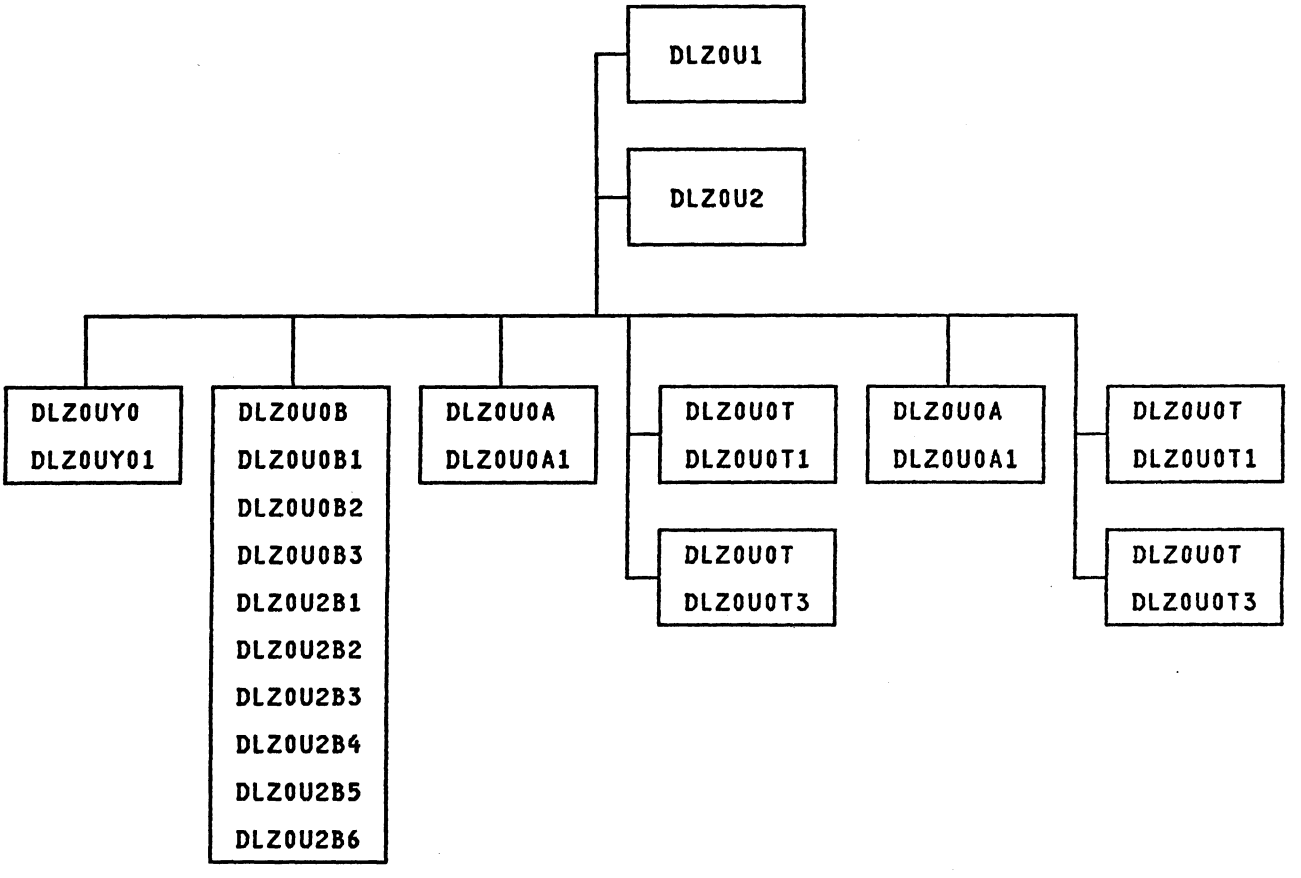


Figure 5-22. Prefix Update Function

HD SINGLE REORGANIZATION FUNCTION: Figure 5-23 on page 5-30 shows the control flow for generating an HD single reorganization utility job stream.

Module	Function
DLZ0U1	Reorganization selection menu
DLZ0U2	Load selection menu
DLZ0UD0	Top function for HD single reorganization
DLZ0U0N	Function for job stream name
DLZ0U0B	Function for data base information
DLZ0UD1	Function for HD single unload
DLZ0U0D	Function for disk RESTART data
DLZ0U0T	Function for tape RESTART data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0D	Function for disk HDUNLD1 data
DLZ0U0T	Function for tape HDUNLD1 data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0D	Function for disk HDUNLD2 data
DLZ0U0T	Function for tape HDUNLD2 data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0R	Function for DLZRRC00 information
DLZ0UD2	Function for HD single reload
DLZ0U0D	Function for disk HDUNLD1 data
DLZ0U0T	Function for tape HDUNLD1 data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0R	Function for DLZRRC00 information

Skeleton	Function
DLZ0UD0A	Skeleton for HD unload/reload

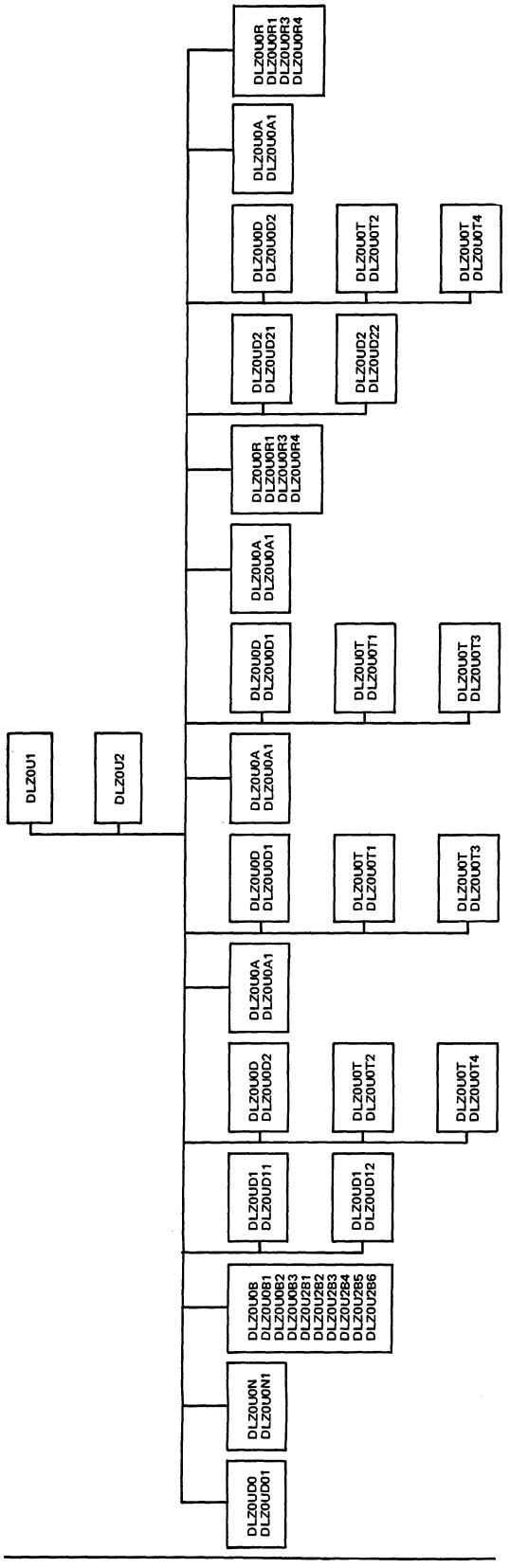
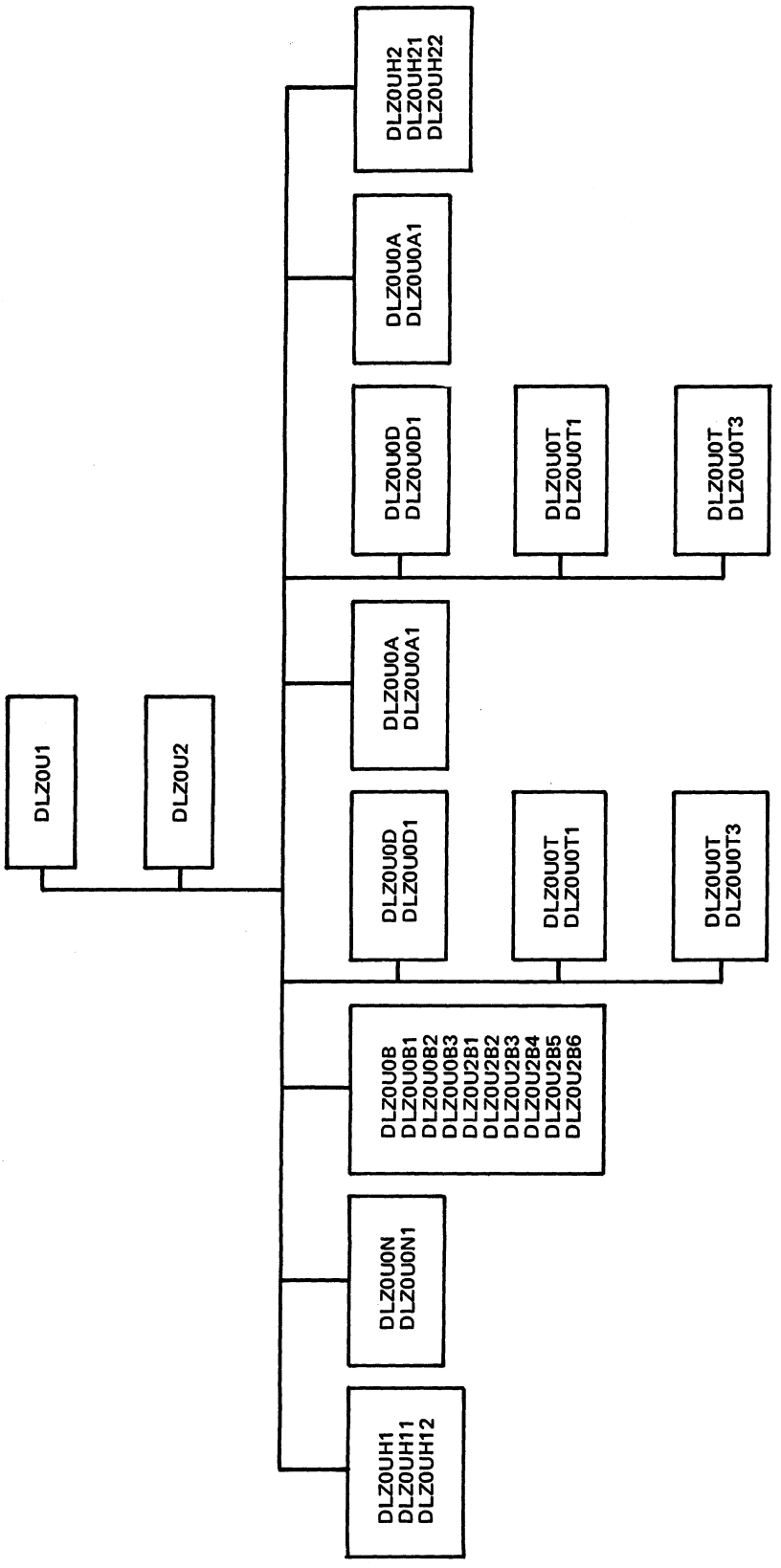


Figure 5-23. HD Single Reorganization Function

HISAM REORGANIZATION FUNCTION: Figure 5-24 on page 5-32 shows the control flow for generating a HISAM reorganization utility job stream.

Module	Function
DLZOU1	Reorganization selection menu
DLZOU2	Load selection menu
DLZOUH1	Top function for Hisam reorganization
DLZOU0N	Function for job stream name
DLZOU0B	Function for data base information
DLZOU0D	Function for disk Copy 1 data
DLZOU0T	Function for tape Copy 1 data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk Copy 2 data
DLZOU0T	Function for tape Copy 2 data
DLZOU0A	Function for tape ASSGN data
DLZOUH2	Function for Hisam reload

Skeleton	Function
DLZOUH1A	Skeleton for HISAM unload utility
DLZOUH2A	Skeleton for HISAM reload utility

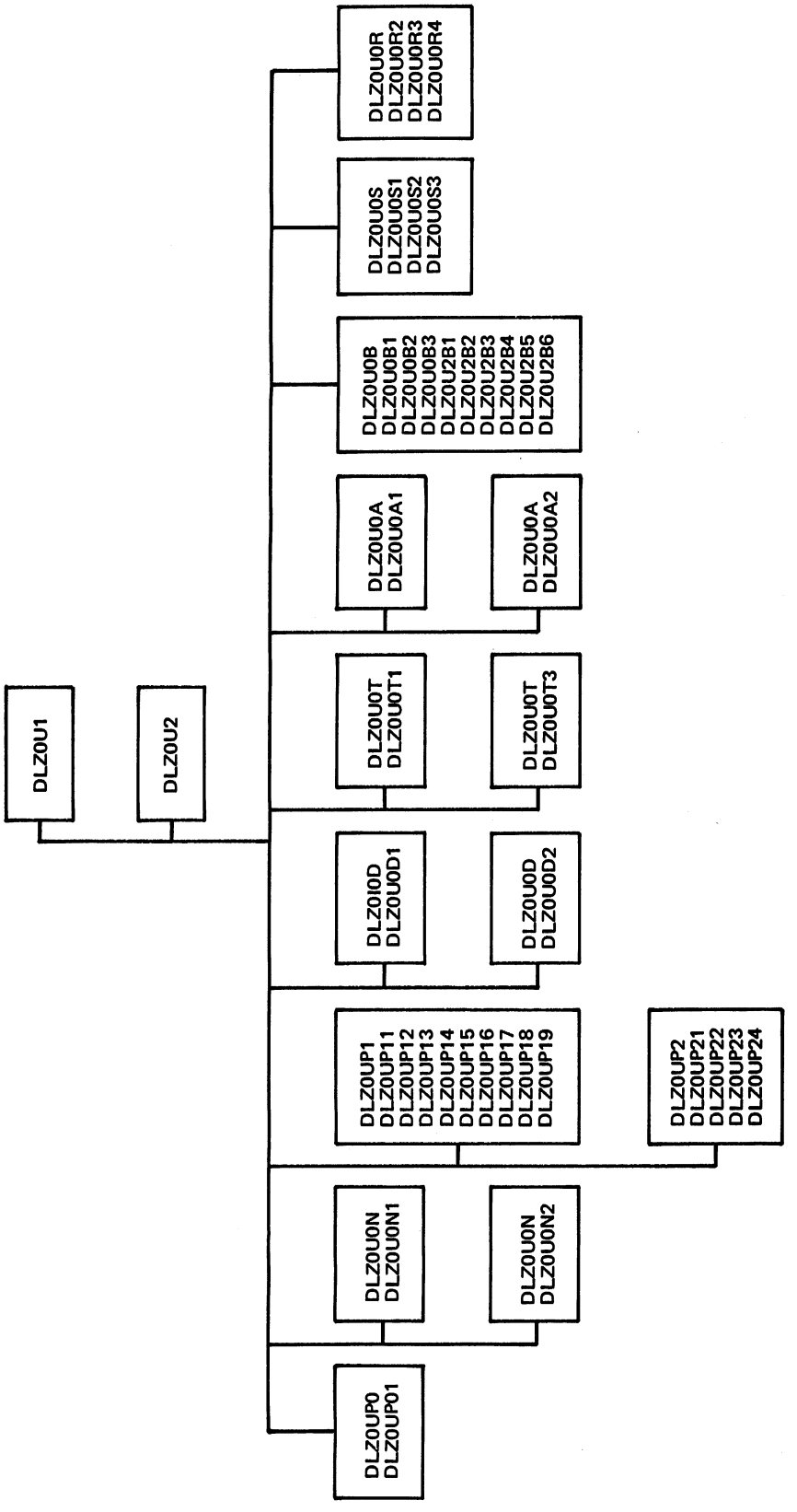


| Figure 5-24. HISAM Reorganization Function

PARTIAL REORGANIZATION FUNCTION: Figure 5-25 on page 5-34 shows the control flow for generating a partial reorganization utility job stream.

Module	Function
DLZOU1	Reorganization selection menu
DLZOU2	Load selection menu
DLZOUP0	Top function for partial reorganization
DLZOU0N	Function for job stream name
DLZOUP1	Function for partial reorganization, Part 1
DLZOU0D	Function for CONTROL data
DLZOU0A	Function for PSB ASSGN data
DLZOUP2	Function for partial reorganization, Part 2
DLZOU0N	Function for save work file data
DLZOU0D	Function for CONTROL data
DLZOU0B	Function for data base information
DLZOU0S	Function for sort work files
DLZOU0R	Function for DLZRRC00 information
DLZOU0D	Function for disk log data
DLZOU0T	Function for tape log data
DLZOU0A	Function for tape ASSGN data

Skeleton	Function
DLZOUP1A	Skeleton for Part 1 parameter data
DLZOUP1B	Skeleton for Part 1 FROMAREA data
DLZOUP1C	Skeleton for Part 1 HEX KEYRANGE data
DLZOUP1D	Skeleton for Part 1 CHARACTER KEYRANGE data
DLZOUP1E	Skeleton for Part 1 TOAREA data
DLZOUP1F	Skeleton for Part 1 PSBGEN data
DLZOUP2A	Skeleton for Part 2 parameter data
DLZOUP2B	Skeleton for Part 2 parameter data



| Figure 5-25. Partial Reorganization Function

Recovery Module Flow

Figure 5-26 on page 5-36 shows the functions that can be initiated through the Recovery option of IUG. Each of these functions is described in the following sections.

CHANGE ACCUMULATION FUNCTION: Figure 5-27 on page 5-37 shows the control flow for generating a change accumulation utility job stream.

Module	Function
DLZ0U3	Recovery selection menu
DLZ0UA0	Function for change accumulation
DLZ0U0N	Function for job stream name
DLZ0U0S	Function for sort work files
DLZ0UA1	Function for detail activity
DLZ0U0D	Function for disk CUMOUT data
DLZ0U0T	Function for tape CUMOUT data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0D	Function for disk LOGOUT data
DLZ0U0T	Function for tape LOGOUT data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0D	Function for disk CUMIN data
DLZ0U0T	Function for tape CUMIN data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0L	Function for LOGIN data
DLZ0U0D	Function for LOGIN for DL/I disk log
DLZ0U0T	Function for LOGIN for DL/I tape log
DLZ0U0A	Function for tape ASSGN data

Skeleton	Function
DLZ0UA0A	Skeleton for ID control statement
DLZ0UA0B	Skeleton for DB0 or DB1 control statement
DLZ0UA0C	Skeleton for AI, AO, or LO control statement

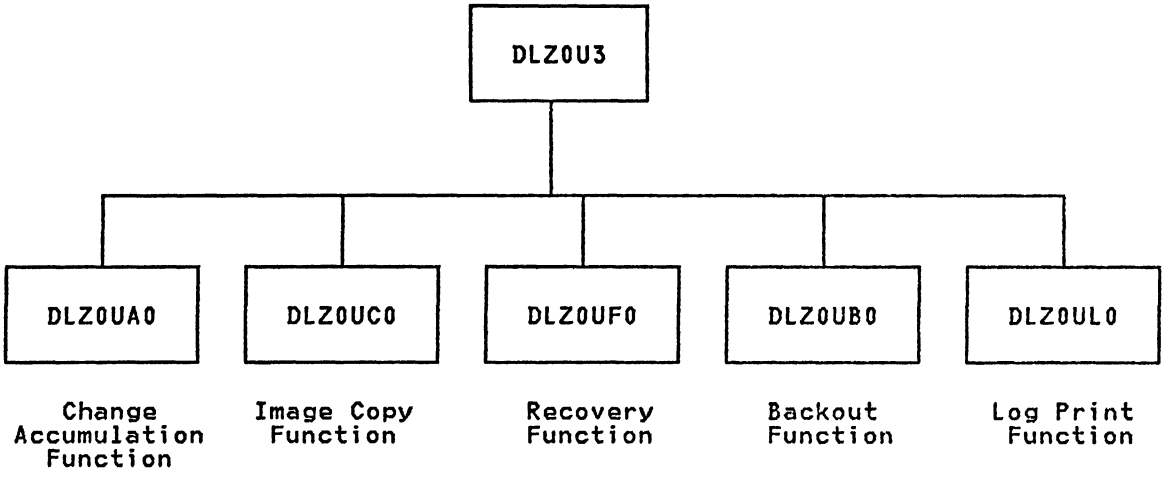


Figure 5-26. Recovery Selection Menu

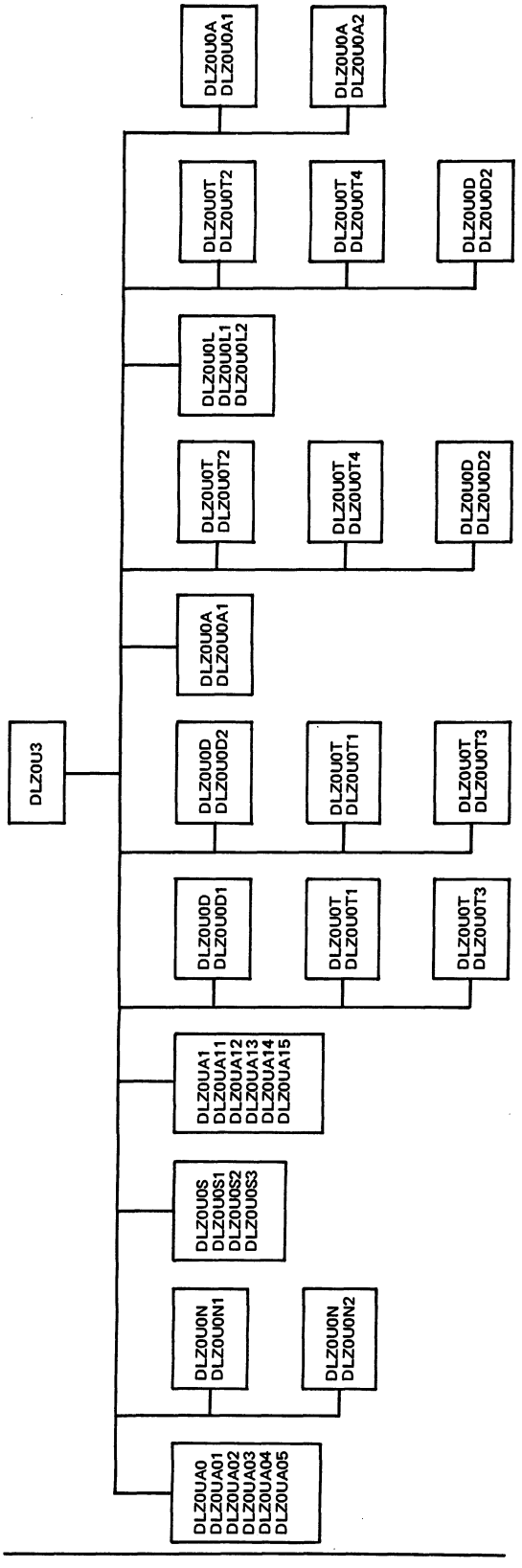


Figure 5-27. Change Accumulation Function

IMAGE COPY FUNCTION: Figure 5-28 shows the control flow for generating an image copy utility job stream.

Module	Function
DLZ0U3	Recovery selection menu
DLZ0UC0	Function for image copy
DLZ0U0N	Function for job stream name
DLZ0U0B	Function for data base information
DLZ0U0T	Function for tape COPY 1 data
DLZ0U0D	Function for disk COPY 1 data
DLZ0U0A	Function for tape ASSGN data
DLZ0U0T	Function for tape COPY 2 data
DLZ0U0D	Function for disk COPY 2 data
DLZ0U0A	Function for tape ASSGN data

Skeleton	Function
DLZ0UC0A	Skeleton for image copy utility

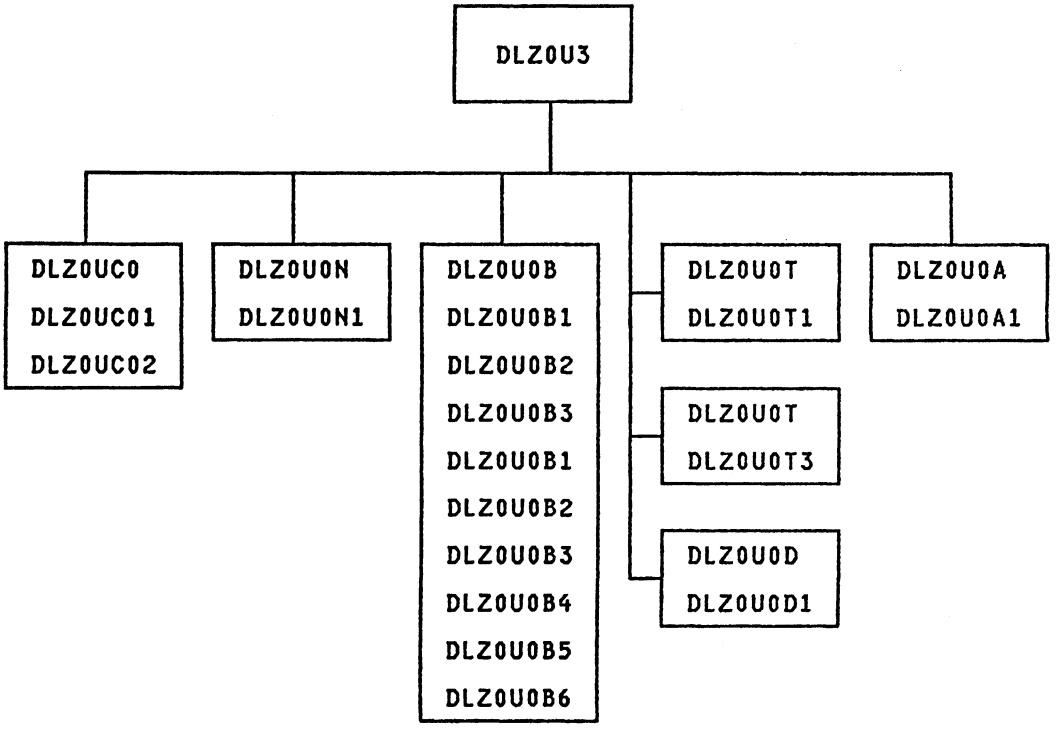


Figure 5-28. Image Copy Function

RECOVERY FUNCTION: Figure 5-29 on page 5-40 shows the control flow for generating a recovery utility job stream.

Module	Function
DLZOU3	Recovery selection menu
DLZOUF0	Function for recovery
DLZOU0N	Function for job stream name
DLZOU0B	Function for data base information
DLZOU0D	Function for disk DUMPIN data
DLZOU0T	Function for tape DUMPIN data
DLZOU0A	Function for tape ASSGN data
DLZOU0D	Function for disk CUMIN data
DLZOU0T	Function for tape CUMIN data
DLZOU0A	Function for tape ASSGN data
DLZOU0L	Function for LOGIN data
DLZOU0D	Function for LOGIN for DL/I disk log
DLZOU0T	Function for LOGIN for DL/I tape log
DLZOU0A	Function for tape ASSGN data
DLZOU0R	Function for DLZRRC00 data

Skeleton	Function
DLZOUF0A	Skeleton for recovery utility
DLZOUA0C	Skeleton for AI and DI control statements

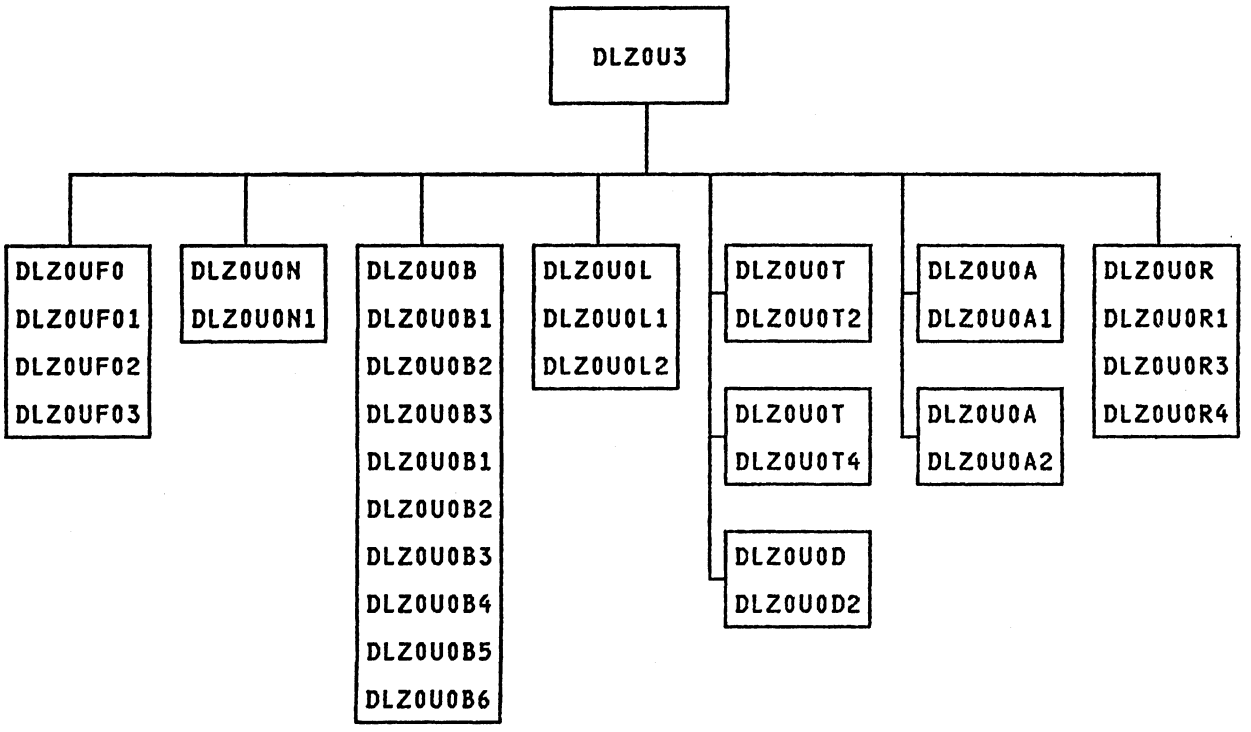


Figure 5-29. Recovery Function

BACKOUT FUNCTION: Figure 5-30 on page 5-42 shows the control flow for generating a backout utility job stream.

Module	Function
DLZOU3	Recovery selection menu
DLZOUB0	Function for backout
DLZOU0N	Function for job stream name
DLZOU0B	Function for data base information
DLZOU0L	Function for LOGIN data
DLZOU0T	Function for LOGIN for DL/I tape log
DLZOU0D	Function for LOGIN for DL/I disk log
DLZOU0A	Function for tape ASSGN data
DLZOU0R	Function for DLZRRC00 data
DLZOU0D	Function for disk log data (DSKLOG1, DSKLOG2)
DLZOU0T	Function for tape log data (LOGOUT)
DLZOU0A	Function for tape ASSGN data

Skeleton	Function
No skeletons are used	

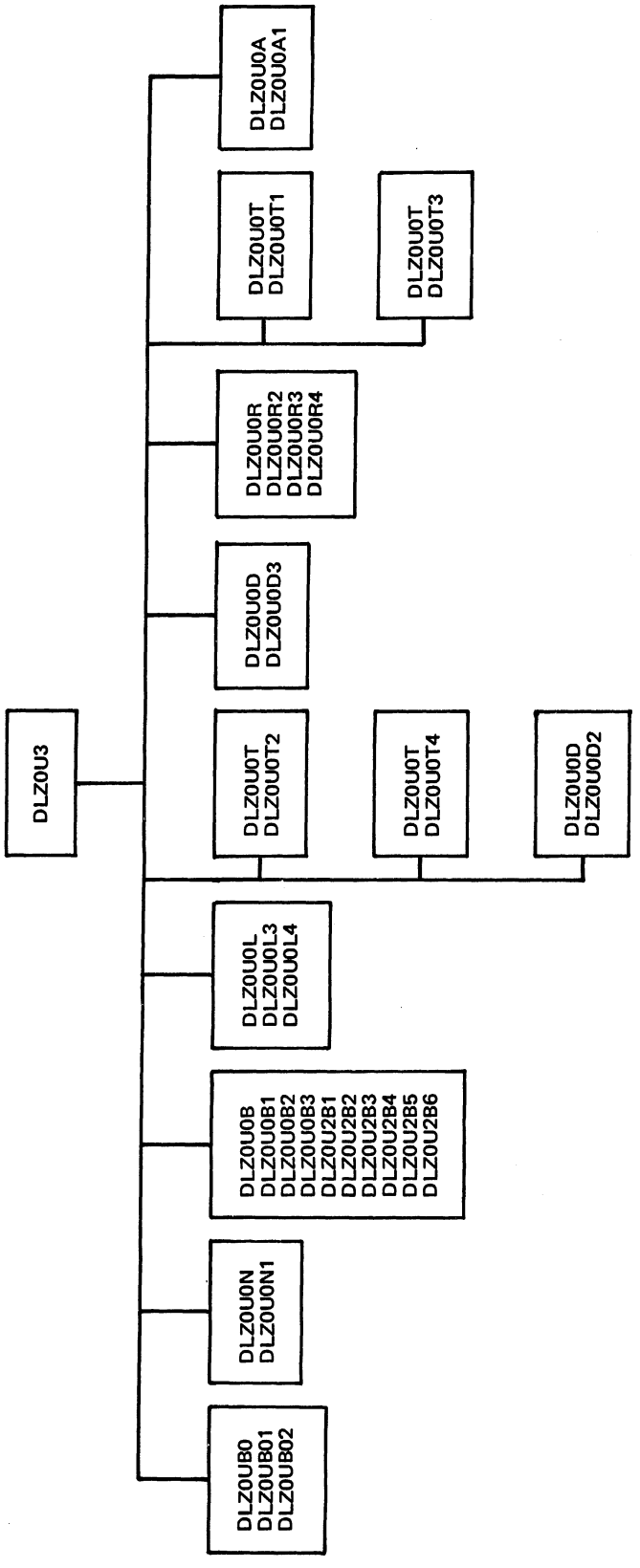


Figure 5-30. Backout Function

LOG PRINT FUNCTION: Figure 5-31 shows the control flow for generating a log print utility job stream.

Module	Function
DLZ0U3	Recovery selection menu
DLZ0U4	Problem Determination selection menu
DLZ0UL0	Function for log print
DLZ0U0N	Function for job stream name
DLZ0U0L	Function for LOGIN data
DLZ0U0T	Function for LOGIN for DL/I tape log
DLZ0U0A	Function for tape ASSGN data
DLZ0U0D	Function for LOGIN for DL/I disk log
DLZ0U0T	Function for tape log data (LOGOUT)
DLZ0U0A	Function for tape ASSGN data

Skeleton	Function
DLZ0UL0A	Skeleton for LO control statement
DLZ0UL0B	Skeleton for LS control statement

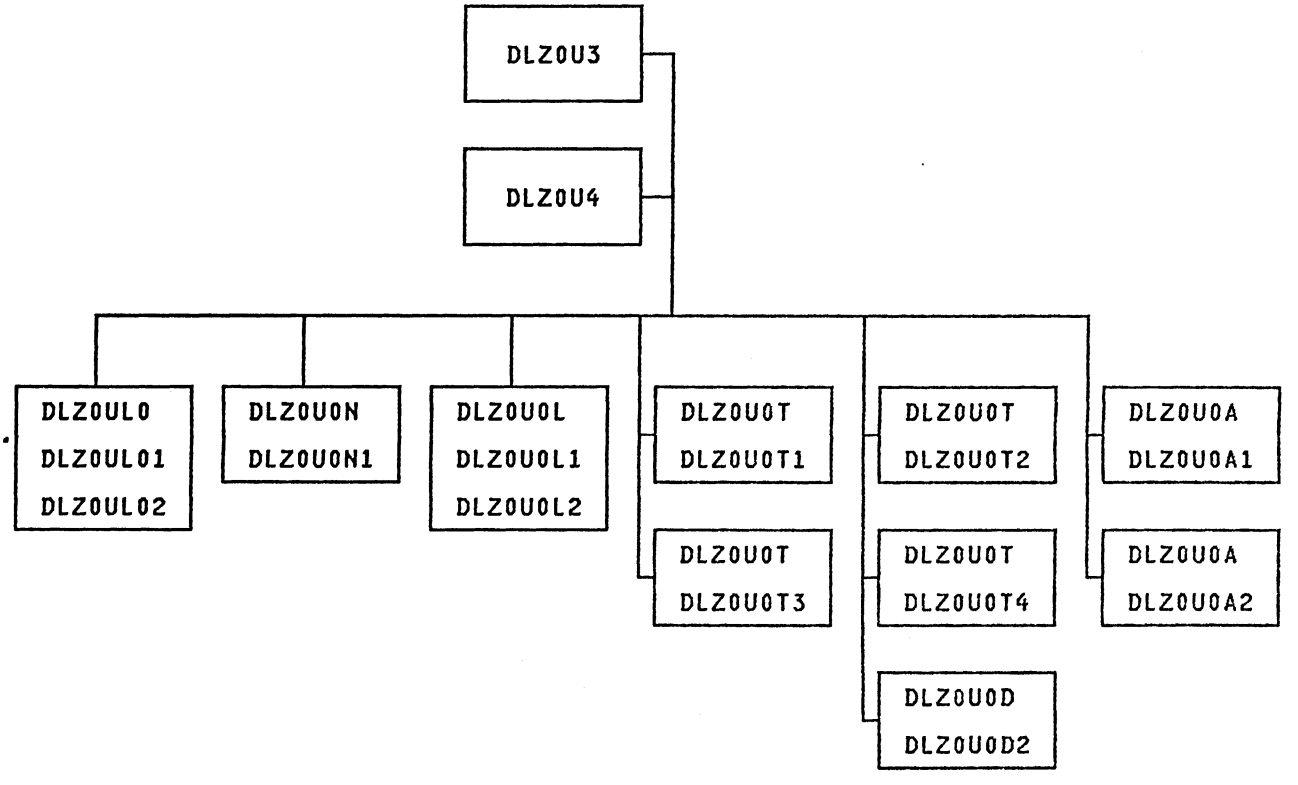


Figure 5-31. Log Print Function

Problem Determination Module Flow

Figure 5-32 shows the functions that can be initiated through the Problem Determination option of IUG. The log print function is described in the previous section, "Recovery Module Flow". Trace print is described in the following section.

TRACE PRINT FUNCTION: Figure 5-33 on page 5-45 shows the control flow for generating a trace print utility job stream.

Module	Function
DLZOU4	Problem determination selection menu
DLZOUTO	Function for trace print
DLZOUON	Function for job stream name
DLZOUOA	Function for tape ASSGN data
DLZOUOA	Function for disk ASSGN data

Skeleton	Function
DLZOUTOA	Skeleton for TI control statement
DLZOUTOB	Skeleton for TO control statement parameters
DLZOUTOC	Skeleton for additional TO control statement parameters

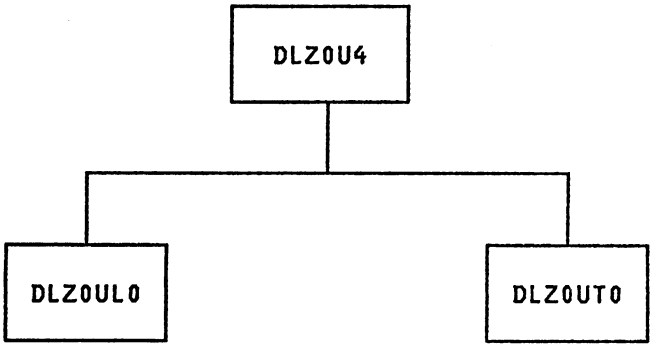


Figure 5-32. Problem Determination Selection Menu

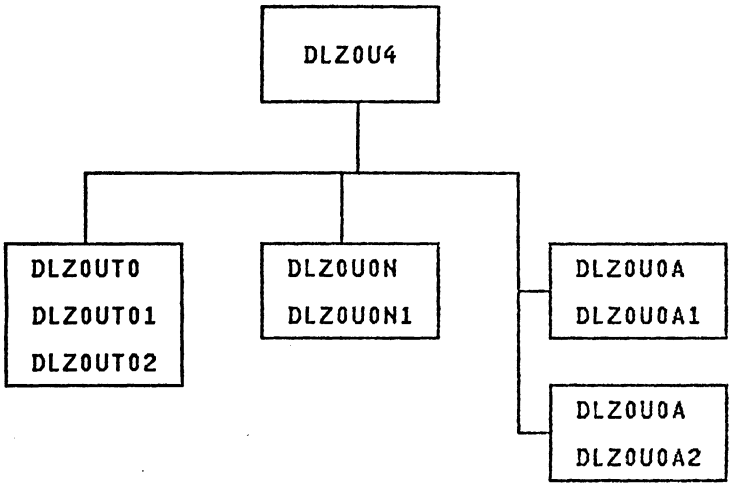


Figure 5-33. Trace Print Function

Other IUG Functions

The ISQL EXTRACT DEFINEs function consists of a single module, DLZ0UE0, and a single data entry panel, DLZ0UE01.

The RESUME INTERRUPTED JOB option is an automatic function that is invoked when the appropriate selection is made on the IUG primary selection menu.

The Data Base Label Information function consists of three modules, and the control flow is shown in Figure 5-34 below.

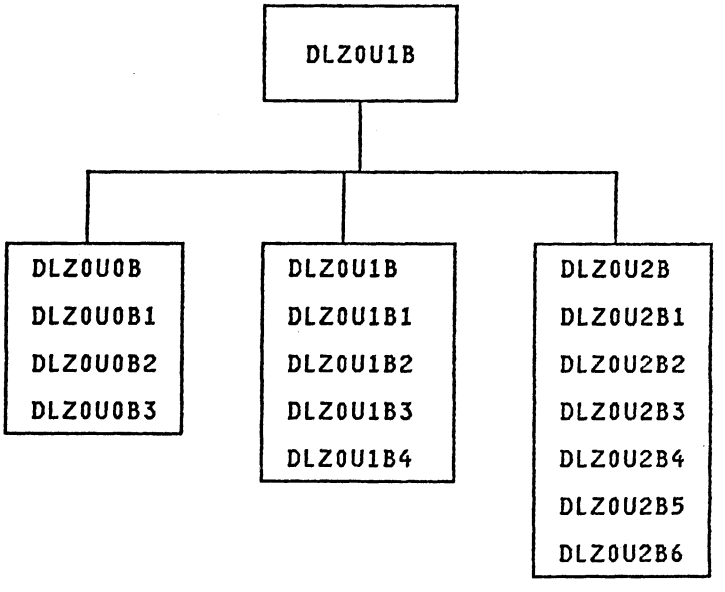


Figure 5-34. Data Base Label Information Function

Module	Function
DLZ0U1B	Data Base Label Information menu
DLZ0U0B	Function for data base DLBL maintainer from function
DLZ0U1B	Function for data base DLBL maintainer from menu
DLZ0U2B	Function for data base DLBL maintainer from function and menu

The Delete or Display Data option allows you to look at, or delete, data maintained by IUG in one of three tables. The tables are:

- CONTROL - These tables are automatically added after the successful generation of a job stream for HD Reorganization with logical relationship utilities.
- SORT - These tables are created when the user requests their creation (for later use).
- PART - These partial reorganization work file tables are also created when requested by the user.

This function consists of a single module, DLZ0U0P, and four display panels, DLZ0U0P1, DLZ0U0P2, DLZ0U0P3, and DLZ0U0P4.

IUG CREATED TABLES

IUG creates and maintains several tables for the generation of utility job streams. These tables include:

- Default table - The default table contains all default values that are to be used when generating a specified utility job stream.
- CONTROL table - The CONTROL table contains all information collected for a specified utility job stream.
- SORT table - The SORT table is a user-specified table that contains sort information that can be used by more than one utility or that can be used in subsequent generations of utility job streams.
- PART table - The PART table is a user-specified table that contains partial reorganization work file information. Like the SORT table data, this information can be used for multiple utilities or multiple job stream generations.

Information contained in the CONTROL, SORT, and PART tables can be displayed or deleted through IUG by selecting the Delete or Display option on the IUG primary selection menu.

IUG GENERATED JOB STREAMS

The job control statements and parameter information that is generated by IUG for each of the DL/I utilities are shown in the following sections.

Preorganization Utility

```
***** CONTROL              CONTROL FILE - used by SCAN, WORKFILE GEN,
                                         and PREFIX RESOLUTION
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                  *
               RECORDS=(25,25),RECSIZE=1600,CAT=usercat,               *
               DISP=(NEW,KEEP)
// EXTENT ,volid1

 ASSGN SYSPCH - only if used OPTIONS=PUNCH - for SCAN
** SYSPCH NOT SUPPORTED

// EXEC DLZRRC00,SIZE=

ULU,DLZURPR0

DBIL= DBD names to be initially loaded - not needed for index

DBR=  DBD names to be reorganized - not needed for index

OPTIONS=(NOPUNCH,STAT,SUMM)
```

Scan Utility

```
***** DATA BASES          for each DB including indexes and LR
// DLBL,DD1name,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

***** RESTART             for restart - same as WORKFIL from previous
                                         interrupted SCAN
// ASSGN SYS010,cuu
// TLBL RESTART,'fileid.17.bytes',YYDDD,serial
        or
// DLBL RESTART,'user.entered.fileid.44.bytes',,VSAM,                  *
               CAT=usercat,DISP=(OLD,KEEP)

***** WORKFIL             output work file - used by PREFIX RESOLUTION
// ASSGN SYS013,cuu
// TLBL WORKFIL,'fileid.17.bytes'YYDDD,serial
        or
// DLBL WORKFIL,'user.entered.fileid.44.bytes',,VSAM,                  *
               RECORDS=(100,100),RECSIZE=6448,CAT=usercat,             *
               DISP=(NEW,KEEP)
// EXTENT ,volid1

***** CONTROL                from PREREORG
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                  *
               CAT=usercat,DISP=(OLD,KEEP)

// EXEC DLZRRC00,SIZE=

ULU,DLZURGS0

DBS=dbdname,segment name - each must be 8 bytes
DBS NOT SUPPORTED - always use CONTROL file from Prereorganization

CHKPT=NO/nnnnn          checkpoint after specified number of records
                           on WORKFIL

RSTRT=NO/nnnnn          restart at specified number from data on
                           RESTART file
```

HD Reorganization Unload Utility

```
••••• DATA BASES              for each DB including indexes and LR
                              2 for HISAM
// DLBL,DDlname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• RESTART                 to RESTART previous unload
// ASSGN SYS010,IGN
          or
// ASSGN SYS010,cuu
// TLBL RESTART,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL RESTART,'user.entered.fileid.44.bytes',,VSAM,                 •
               CAT=usercat,DISP=(OLD,KEEP)

••••• UNLOAD 1                first UNLOAD file - TLBL has SL
// ASSGN SYS011,cuu
// TLBL HDUNLD1,'fileid.17.bytes'YYDDD,serial,volseq,fileseq,genn,vn
          or
// DLBL HDUNLD1,'user.entered.fileid.44.bytes',,VSAM,                 •
               RECORDS=(100,100),RECSIZE=6448,CAT=usercat,           •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• UNLOAD 2               second UNLOAD file
// ASSGN SYS012,cuu
// TLBL HDUNLD2,'fileid.17.bytes',YYDDD,serial,volseq,fileseq,genn,vn
          or
// DLBL HDUNLD2,'user.entered.fileid.44.bytes',,VSAM,                 •
               RECORDS=(100,100),RECSIZE=6448,CAT=usercat,           •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

// EXEC DLZRRC00,SIZE=

ULU,DLZRRC00,dbname,buf,HDBFR,HSBFR,TRACE
PLU          psbname

REW=U/N/R                REWIND option

CHKPT=5000/nnnnn/NO      CHECKPOINT option
```

HD Reorganization Reload Utility

```
••••• DATA BASES              for each DB including indexes and LR
                              2 for HISAM
// DLBL,DDlname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• UNLOAD Input
// ASSGN SYS011,cuu
// TLBL HDUNLD1,'fileid.17.bytes'YYDDD,serial,volseq,fileseq,genn,vn
        or
// DLBL HDUNLD1,'user.entered.fileid.44.bytes',,VSAM,                 •
              CAT=usercat,DISP=(OLD,KEEP)

••••• RESTART             for restart - same as WORKFIL from previous
                                        interrupted Reload
// ASSGN SYS010,cuu
// TLBL RSTFILE,'fileid.17.bytes',YYDDD,serial
        or
// DLBL RSTFILE,'user.entered.fileid.44.bytes',,VSAM,                 •
              CAT=usercat,DISP=(OLD,KEEP)

••••• WORKFIL             output work file - used by PREFIX RESOLUTION
                          required for LR and, (in IUG) for SI
// ASSGN SYS013,cuu
// TLBL WORKFIL,'fileid.17.bytes'YYDDD,serial
        or
// DLBL WORKFIL,'user.entered.fileid.44.bytes',,VSAM,                 •
              RECORDS=(10,10),RECSIZE=6446,CAT=usercat,              •
              DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• CONTROL                from PREREORG - required for LR and SI
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                 •
              CAT=usercat,DISP=(OLD,KEEP)

// EXEC DLZRRC00,SIZE=

ULU,DLZURGL0,dbdname,buf,HDBFR,HSBFR,TRACE
ULR

REW=U/N/R

BLDINDEX=NO
         Use single data base reorg option to build SI automatically
         Use full logical relationship option to build SI using LR
           and always specify BLDINDEX=NO
```

Initial Load Utility

```
••••• User Input File     If reading data from a file
                          If on Tape, SAM ESDS or SAM
// ASSGN SYSnnn,cuu
   and if labeled
// TLBL filenam,'fileid.17.bytes'YYDDD,serial,volseq,fileseq,genn,vn
         or
// DLBL filenam,'user.entered.fileid.44.bytes',,VSAM,                 •
              CAT=usercat,DISP=(OLD,KEEP)

••••• WORKFIL             output work file - used by PREFIX RESOLUTION
                          required for LR and SI using LR utilities
// ASSGN SYS013,cuu
// TLBL WORKFIL,'fileid.17.bytes'YYDDD,serial
         or
// DLBL WORKFIL,'user.entered.fileid.44.bytes',,VSAM,                 •
              RECORDS=(10,10),RECSIZE=6446,CAT=usercat,              •
              DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• CONTROL            from PREREORG - required for LR and SI
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                 •
              CAT=usercat,DISP=(OLD,KEEP)

••••• DATA BASES          for each DB including indexes and LR
                          2 for HISAM
// DLBL,DD1name,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat
        OVFLWname

// EXEC DLZRRC00,SIZE=

DLI,LOADPGM,LOADPSB,buf,HDBFR,HSBFR,TRACE
```

Prefix Resolution Utility

```
••••• Sort Work          Up to 8 sort work files
// DLBL SORTWKn,'DOS.WORKFILE.SYS.SORTWKn',                          •
               VSAM,RECORDS=(70,70),RECSIZE=1024,CAT=usercat,       •
               DISP=(NEW,DELETE)
// EXTENT ,volid

••••• INTRMED            intermediate file - Sort data between Sort
                         calls
// ASSGN SYS010,cuu
// TLBL INTRMED,'fileid.17.bytes'YYDDD,serial
          or
// DLBL INTRMED,'user.entered.fileid.44.bytes',,VSAM,                •
               RECORDS=(70,70),RECSIZE=1024,CAT=usercat,            •
               DISP=(NEW,KEEP)
// EXTENT ,volid1
// EXTENT ,volid2
// EXTENT ,volid3

••••• WRKINnn            Input work files from INITIAL LOAD, RELOAD,
                         and SCAN - nn=01-99
// ASSGN SYS013,cuu
// TLBL WRKINnn,'fileid.17.bytes'YYDDD,serial
          or
// DLBL WRKINnn,'user.entered.fileid.44.bytes',,VSAM,                •
               CAT=usercat,DISP=(OLD,DELETE)

••••• WORKFIL            Sorted LR output - Used by PREFIX UPDATE
// ASSGN SYS011,cuu
// TLBL WORKFIL,'fileid.17.bytes'YYDDD,serial
          or
// DLBL WORKFIL,'user.entered.fileid.44.bytes',,VSAM,                •
               RECORDS=(25,25),RECSIZE=3192,CAT=usercat,            •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• INDXWRK            Sorted SI output - Used by PREFIX UPDATE
// ASSGN SYS014,cuu
// TLBL INDXWRK,'fileid.17.bytes'YYDDD,serial
          or
// DLBL INDWRKL,'user.entered.fileid.44.bytes',,VSAM,                •
               RECORDS=(25,25),RECSIZE=3192,CAT=usercat,            •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• CONTROL          from PREREORG - required for LR and SI
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                •
               CAT=usercat,DISP=(OLD,DELETE)

// EXEC DLZRRC00,SIZE=

1   10          15              21 22           25
R   L,I,blank   Num Sort Work   Num Input Work  DUMP
```

Prefix Update Utility

```
••••• DATA BASES          For each DB including indexes and LR
// DLBL DDlname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• WORKFIL             Work file from PREFIX RESOLUTION
// ASSGN SYS011,cuu
// TLBL WORKFIL,'fileid.17.bytes'YYDDD,serial
        or
// DLBL WORKFIL,'user.entered.fileid.44.bytes',,VSAM,                    •
               CAT=usercat,DISP=(OLD,DELETE)                            •
// EXTENT ,volid

••••• INDXWRK             Work file from PREFIX RESOLUTION
// ASSGN SYS014,cuu
// TLBL INDXWRK,'fileid.17.bytes'YYDDD,serial
        or
// DLBL INDWRKL,'user.entered.fileid.44.bytes',,VSAM,                    •
               CAT=usercat,DISP=(OLD,DELETE)                            •
// EXTENT ,volid

// EXEC DLZRRC00,SIZE=

ULU,DLZURGP0

1   10
U   L,I,blank
```

HISAM Reorganization Unload Utility

```
••••• DATA BASES          for each DB to be unloaded - KSDS and ESDS
// DLBL,DDlname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• UNLOAD 1            First UNLOAD file - TLBL has SL
// ASSGN SYS011,cuu
// TLBL DLZ1D01,'fileid.17.bytes'YYDDD,serial,volseq,fileseq,genn,vn
        or
// DLBL DLZ1D01,'user.entered.fileid.44.bytes',,VSAM,                    •
               RECORDS=(100,100),RECSIZE=4112,CAT=usercat,              •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• UNLOAD 2          Second UNLOAD filefor first data base
// ASSGN SYS012,cuu
// TLBL DLZ2D01,'fileid.17.bytes',YYDDD,serial,volseq,fileseq,genn,vn
        or
// DLBL DLZ2D01,'user.entered.fileid.44.bytes',,VSAM,                    •
               RECORDS=(100,100),RECSIZE=4112,CAT=usercat,              •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

// EXEC DLZRRC00,SIZE=

1 2 3     4       13      22      31      40
R n U,N,R DBDname DDlname DLZ1D01 DLZ2D01 comments
```

HISAM Reorganization Reload Utility

```
••••• DATA BASES          for each DB to be reloaded - KSDS and ESDS
// DLBL,DD1name,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• RELOAD              UNLOAD file
// ASSGN SYS011,cuu
// TLBL DLZ1D01,'fileid.17.bytes'YYDDD,serial,volseq,fileseq,genn,vn
        or
// DLBL DLZ1D01,'user.entered.fileid.44.bytes',,VSAM,                  •
               CAT=usercat,DISP=(OLD,KEEP)
// EXEC DLZRRC00,SIZE=

1 3     22      40
L U,N,R DLZ1D01 comments
```

Partial Reorganization, Part 1, Utility

```
••••• CONTROL            CONTROL FILE - used by Part 2
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                    •
               RECORDS=40,RECSIZE=6000,CAT=usercat,                    •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

If a PSBGEN will be done, the following job control statements
will also be generated:
a)
// DLBL IJSYSPH,'PART 1 PSB',0
// EXTENT SYSPCH,,,,tr,num          Disk with cuu specified
ASSGN SYSPCH,cuu
                                         or
b)
// DLBL IJSYSPH,'PART 1 PSB',0
// EXTENT SYSPCH,volid,,,tr,num    Disk with volid specified
ASSGN SYSPCH,DISK,VOL=volid,SHR
            devtype
                                        or
c)
// TLBL IJSYSPH,'PART 1 PSB'       Labeled Tape

ASSGN SYSPCH,(cuu,cuu,cuu,cuu,cuu),ss
                                        or
ASSGN SYSPCH,TAPE,VOL=volid        Tape with volid specified
            devtype

••••• Main part of Partial Reorg Part 1 job stream is here •••••
// EXEC DLZPRCT1,SIZE=AUTO

DBNAME=dbdname

KEYRANGE=(low,high)
              EOD
FROMAREA=(lowblock,highblock)
                   EOD
TOAREA=(DD1name,lowblock,highblock)
                EOD  or  EOD
PSB=psbname
                           ALL
SORTOPT='STORAGE=nK,PRINT=CRITICAL,DIAG'
                           NONE

// DLBL IJSYSIN,'PART 1 PSB'      Disk
// EXTENT SYSIPT
     or
// TLBL IJSYSIN,'Part 1 PSB'  Labeled Tape

ASSGN SYSIPT,SYSPCH          Assign SYSIPT to SYSPCH device
CLOSE SYSPCH,00d             Close and set to user standard address
// EXEC PROC=procname        Optional PROC
// OPTION CATAL
// EXEC ASSEMBLY             PSBGEN
// EXEC LNKEDT
CLOSE SYSIPT,00c             Close and set to user standard address
// EXEC DLZUACB0,SIZE=AUTO   Block Build
    BUILD PSB=PSBname,OUT=LINK,DMB=YES    DMB=YES if user requested
/•
// EXEC LNKEDT
/&
```

Partial Reorganization, Part 2, Utility

```
••••• CONTROL           CONTROL FILE - created by Part 1
// DLBL CONTROL,'user.entered.fileid.44.bytes',,VSAM,                    •
               CAT=usercat,DISP=(OLD,DELETE)

••••• DATA BASES         For each DB including indexes and LR
// DLBL,DD1name,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat

••••• PRWRKF1            Intermediate workfile 1
// DLBL PRWRKF1,,0,VSAM,RECORDS=380,RECSIZE=3080,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF2            Intermediate workfile 2
// DLBL PRWRKF2,,0,VSAM,RECORDS=900,RECSIZE=180,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF3            Intermediate workfile 3
// DLBL PRWRKF3,,0,VSAM,RECORDS=900,RECSIZE=180,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF4            Intermediate workfile 4
// DLBL PRWRKF4,,0,VSAM,RECORDS=200,RECSIZE=1000,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF5            Intermediate workfile 5
// DLBL PRWRKF5,,0,VSAM,RECORDS=200,RECSIZE=1000,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF6            Intermediate workfile 6
// DLBL PRWRKF6,,0,VSAM,RECORDS=80,RECSIZE=3008,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF7            Intermediate workfile 7
// DLBL PRWRKF7,,0,VSAM,RECORDS=110,RECSIZE=1800,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF8            Intermediate workfile 8
// DLBL PRWRKF8,,0,VSAM,RECORDS=80,RECSIZE=3008,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid

••••• PRWRKF9            Intermediate workfile 9
// DLBL PRWRKF9,,0,VSAM,RECORDS=80,RECSIZE=2620,CAT=usercat,DISP=(NEW,KEEP)
// EXTENT ,volid
••••• SORT WORK          3 sort work files
// DLBL SORTWKn,'DOS.WORKFILE.SYS.SORTWKn',                             •
               VSAM,RECORDS=(10000),RECSIZE=80,CAT=usercat,DISP=(NEW,DELETE)
// EXTENT ,volid

// TLBL LOGOUT,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
         or
// DLBL DSKLOG1,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat         •
    and (optionally)
// DLBL DSKLOG2,'user.entered.fileid.44.bytes',,VSAM,CAT=usercatd

// EXEC DLZRRC00,SIZE=

DLI,DLZPRCT2,psbname

DBNAME=dbdname    same name as in Part 1

SCANSEG=(dbdname1,segname1)
SCANSEG=(dbdname1,segname2)
SCANSEG=(dbdname2,segname1)
```

Change Accumulation Utility

```
••••• Sort Work              Up to 8 sort work files
// DLBL SORTWKn,'DOS.WORKFILE.SYS.SORTWKn',                          •
               VSAM,RECORDS=(10000),RECSIZE=80,CAT=usercat,         •
               DISP=(NEW,DELETE)
// EXTENT ,volid
••••• CUMOUT                 - SAM ESDS file or Tape
// ASSGN SYS010,cuu or
                IGN
// TLBL CUMOUT,'fileid.17.bytes',YYDDD,serial
          or
// DLBL CUMOUT,'user.entered.fileid.44.bytes',,VSAM,                 •
               RECORDS=1500,RECSIZE=80,CAT=usercat,                 •
               DISP=(NEW,KEEP)
// EXTENT ,volid
••••• CUMIN                  - SAM ESDS or Tape
// ASSGN SYS011,cuu or
                IGN
// TLBL CUMIN,'fileid.17.bytes',YYDDD,serial
          or
// DLBL CUMIN,'user.entered.fileid.44.bytes',,VSAM,                  •
               CAT=usercat,DISP=(OLD,KEEP)
      (should user be asked?)  or   DELETE
••••• LOGOUT                 - VSAM file
// ASSGN SYS012,cuu or
                IGN
// TLBL LOGOUT,'fileid.17.bytes',YYDDD,serial
          or
// DLBL LOGOUT,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat,     •
               DISP=(NEW,KEEP)
      (must be preDEFINEd VSAM file with this ID)
••••• LOGIN                  - labeled/unlabeled tape, VSAM, SAM
// ASSGN SYSnnn,cuu
   and if labeled
// TLBL LOGINnn,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL LOGINnn,'user.entered.fileid.44.bytes',,VSAM,                •
               CAT=usercat,DISP=(OLD,KEEP)
          or
// ASSGN SYSnnn,cuu
// DLBL LOGINnn,'user.entered.fileid.44.bytes'
// EXTENT SYSnn,volser        - volser only if specified on DLZOUOA2

// EXEC DLZUCUMO,SIZE=

1   11-13          31-33    41    51-52
ID  Number of DBn  Max Key  Sort  Number of input logs

1    4-10     12-20
DBO  DBDname  yydddhhmm
      •ALL
      •OTHER

1    4-10     12-20
DB1  DBDname  yydddhhmm
      •ALL
      •OTHER

1   11-16  18     21        31-35
LI  SYSnn  U,N,R  L,U,V,S   bufsize

1   11-16   18
LO  SYS012  N,R,U

1   11-16   18
AI  SYS011  U,N,R

1   11-16   18
AO  SYS010  U,N,R
```

Image Copy Utility

```
••••• DATA BASES              For each DB to be dumped
// DLBL,DD1name,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat
     or OVFLWnm

••••• OUTPUT 1                First OUTPUT file - TLBL has SL
// ASSGN SYS011,cuu
// TLBL DLZ1D01,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL DLZ1D01,'user.entered.fileid.44.bytes',,VSAM,                  •
               RECORDS=(100,100),RECSIZE=4112,CAT=usercat,            •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

••••• OUTPUT 2                Second OUTPUT file
// ASSGN SYS012,cuu
// TLBL DLZ2D01,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL DLZ2D01,'user.entered.fileid.44.bytes',,VSAM,                  •
               RECORDS=(100,100),RECSIZE=4112,CAT=usercat,            •
               DISP=(NEW,KEEP)
// EXTENT ,volid1

// EXEC DLZUDMP0,SIZE=

1  2  4         13         22         30      31         39      40
D  n  DBDname   DD1name    DLZ1D01    U,N,R   DLZ2D01    U,N,R   comments
                OVFLWname
```

Recovery Utility

```
••••• DUMP Input              Input from Image Copy or Hisam Unload
// ASSGN SYS011,cuu
// TLBL DUMPIN,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL DUMPIN,'user.entered.fileid.44.bytes',,VSAM,                 •
               CAT=usercat,DISP=(OLD,KEEP)

••••• CUMIN                   Accumulated change file
// ASSGN SYS012,cuu or
                IGN
// TLBL CUMIN,'fileid.17.bytes',YYDDD,serial
          or
// DLBL CUMIN,'user.entered.fileid.44.bytes',,VSAM,                  •
               CAT=usercat,DISP=(OLD,KEEP)

••••• LOGIN                   labeled/unlabeled tape, VSAM, SAM
// ASSGN SYSnnn,cuuor
IGN
   and if labeled
// TLBL LOGINnn,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL LOGINnn,'user.entered.fileid.44.bytes',,VSAM,                 •
               CAT=usercat,DISP=(OLD,KEEP)
          or
// ASSGN SYSnnn,cuu
// DLBL LOGINnn,'user.entered.fileid.44.bytes'
// EXTENT SYSnn,volser        - volser only if specified on DLZOU0A2

••••• DATA BASES              for each DB, 2 for HISAM
// DLBL,DDlname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat
     or OVFLWnm

// EXEC DLZUCUM0,SIZE=
UDR,DLZURDB0,dbdname

1  4         13        22-23                  25
S  DBDname   DDlname   Number-of-log-files    comments

1   11-16    18      21        31-35
LI  SYSnnn   U,N,R   L,U,V,S   bufsize

1   11-16    18
DI  SYS011   U,N,R

1   11-16    18
AI  SYS012   U,N,R
```

Backout Utility

```
••••• LOGIN                    - labeled/unlabeled tape, VSAM
// ASSGN SYSnnn,cuu
   and if labeled
// TLBL LOGINnn,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL LOGINnn,'user.entered.fileid.44.bytes',,VSAM,                 •
               CAT=usercat,DISP=(OLD,KEEP)

••••• LOGOUT                   labeled tape or VSAM
// ASSGN SYSnnn,cuu
// TLBL LOGOUT,'fileid.17.bytes',YYDDD,serial,volseq,fileseq,genn,vn
          or
// DLBL DSKLOG1,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat      •
     and (optionally)
// DLBL DSKLOG2,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat      •

••••• DATA BASES               for each DB, 2 for HISAM
// DLBL,DDIname,'user.entered.fileid.44.bytes',,VSAM,CAT=usercat
     or OVFLWnm

// EXEC DLZRRC00,SIZE=

DLI,DLZBACK0,PSBname,buf,hdbfr=,hsbfr=,log=,trace=

1   11-16  18     21       31-35
LI  SYSnn  U,N,R  L,U,V,S  bufsize
```

Log Print Utility

```
••••• LOGIN                    - labeled/unlabeled tape, VSAM, SAM
// ASSGN SYSnnn,cuu
   and if labeled
// TLBL LOGINnn,'fileid.17.bytes',YYDDD,serial,volseq,genn,vn
          or
// DLBL LOGINnn,'user.entered.fileid.44.bytes',,VSAM,                 •
               CAT=usercat,DISP=(OLD,KEEP)
          or
// ASSGN SYSnnn,cuu
// DLBL LOGINnn,'user.entered.fileid.44.bytes'
// EXTENT SYSnn,volser         - volser only if specified on DLZOU0A2

••••• LOGOUT                   labeled tape
// ASSGN SYS012,cuu
// TLBL LOGOUT,'fileid.17.bytes',YYDDD,serial,volseq,fileseq,genn,vn

// EXEC DLZOGP0,SIZE=

1   11-16  18     21       31-35
LI  SYSnn  U,N,R  L,U,V,S  bufsize

1   4-10     13-21       23-31       33-39    41-44  46-50  52
LO  PSBname  start       end         KEYWORD  COPY   CICSD  N,R,U
             yydddhhmm   yydddhhmm

1   4-10     12-16     18-25
LS  DBDname  CICS ID   RBN
             nnnnn     xxxxxxxx
```

Trace Print Utility

```
••••• TAPEIN                  unlabeled tape
// ASSGN SYS013,cuu
// TLBL TAPEIN

       or

••••• DISKIN                  SAM Disk File
// ASSGN SYS013,cuu
// DLBL DISKIN,'user.entered.file.id.44.bytes'

// EXEC DLZTPRT0,SIZE=

1   11      31
TI  SYS013  bufsize

TO TASKID=abcde
TO MODULE=fghi
TO TYPECALL=(abcd,efgh,ijkl,mnop,qrst) - parentheses only if
                                         more than 1
TO CALLNUM=(999,11111)
```

Extract Define Utility

Multi Partition

```
• $$ JOB JNM=jobname
// JOB jobname
// EXEC DLZEXDFP,SIZE=AUTO
EXTRACT PSBNAME=psbname                                          •
PCBNAME=pcbname                                                  •
DLIPROC=procname                                                 •
REPLACE,                                                         •
USERID=userid/password
EXTRACT PSBNAME=(psbname,pcbnumber)                              •
PCBNAME=pcbname                                                  •
DLIPROC=procname                                                 •
USERID=userid/password
/•
/&
• $$ EOJ
```

Single Partition

```
• $$ JOB JNM=jobname
// JOB jobname
// EXEC ARISOLDS,SIZE=AUTO,PARM='SYSMODE=S,PROGNAME=DLZEXDFP'
EXTRACT PSBNAME=(psbname,pcbnumber)                              •
PCBNAME=pcbname                                                  •
DLIPROC=procname                                                 •
REPLACE,                                                         •
USERID=userid/password
EXTRACT PSBNAME=(psbname,pcbnumber)                              •
PCBNAME=pcbname                                                  •
DLIPROC=procname                                                 •
USERID=userid/password
/•
/&
• $$ EOJ
```

ERROR CONDITIONS

IUG Error Detecting

If information having incorrect syntax (such as a value outside the acceptable range of choices) is entered, an error message appears on the data entry panel so that the information can be corrected before it is entered into the CONTROL table. The error messages are described in the "Error Messages" section of DL/I DOS/VS Interactive Resource Definition and Utilities. Because IUG does not check for all possible error conditions, the output listings should be carefully checked, the errors corrected, and the output regenerated before attempting to use the IUG panel defined job streams.

Abnormal Ends

If the system abnormally ends after data has been entered for a utility but before job stream generation has completed, you can resume processing from the point of interruption. This can be done by selecting the "Resume Interrupted Processing" option on the IUG primary selection menu.

CHAPTER 6. DL/I TRACE FACILITY AND TRACE PRINT UTILITY

This chapter covers:

- using the DL/I Trace Facility and its operands and parameters to assist you in problem determination.
- using the Trace Print Utility to print desired trace entries from files created by the DL/I Trace Facility.

Pictorial layouts are used for:

- Data
- Parameter lists
- Trace entry header
- DSECT formats

with explanations of their content.

The OPTION operands list the long and short format and the trace points.

DL/I TRACE FACILITY

DL/I offers a trace facility for use as a diagnostic tool. This facility consists of three major areas.

1. The trace module which is assembled from the DLZTRACE and DLZTRENT macros which are optionally link edited with the DLZTRPP0 module.

2. The user exit which is optionally supplied by the user by writing his own routine or modifying the sample user exit (DLZTXIT0). Macro DLZTUPRM provides a DSECT of the user exit parameter list.

3. The trace print utility, consisting of modules DLZTPRT0 and DLZTPRM0, which is used to print the trace output that was directed to tape or disk.

Each of these areas are discussed in detail in the following text.

Macro DLZTRCAL is inserted into specific DL/I modules to link to the trace module. Code generated by this macro tests if the trace type specified for that point has been initialized. If so, it branches to the trace module with a pointer to the parameter list that identifies the trace point. The trace module, which was loaded by DL/I initialization, then determines if tracing is to take place and, if so, records the required information.

The DL/I trace is composed of the following types of traces, any combination of which may be selected by the option parameter of the DLZTRACE macro:

- Trace users calls (USERCALL)
- Trace module flow (MODTRACE)
- Retrieve JCB values (RETRIEVE)
- Current position in data base (CURRPOS)
- Exit/return to VSAM (VSAMINTF)
- Exit/return to buffer handler(BHINTF)
- Requests to index maintenance (INDEXTRC)
- Requests to online buffer handler (ONLINEBH)
- Program isolation queueing calls (PITRACE)

These may be further limited by specific data base, key range, call number, function PSB, segment type, etc., or by use of a user exit.

THE TRACE MODULE

DEFINING THE TRACE FACILITY

The trace facility is defined by specifying the desired operands in the DLZTRACE macro. You may choose the type(s) of trace(s) desired, where you want the tracing to occur, and the output designation. If more direct control is needed, you can provide a user exit.

To use the DL/I trace facility, a trace module must be in the core image library. This module must be assembled using the DLZTRACE macro. You can assemble a number of link trace modules, each with a different set of options, using a unique phasename for each module.

DLZTRACE MACRO

The options and selectable parameters that are available for the DLZTRACE macro are shown in Figure 6-1 on page 6-4.

Note: If you choose to specify call conditions that are to govern the trace, there are three mutually exclusive ways of specifying the CALLCON operand, as illustrated in Figure 6-1. You must select one of the alternatives in combination with your selection of the remaining operands shown. The next section describes the various call conditions you can establish through selection of CALLCON parameters.

CALLCON

States the call conditions that must be satisfied before any tracing occurs during the call. Five parameters are available for defining the call conditions.

- CALLNUM - the relative call occurrence, the first DL/I call being number 1. The value entries represent the call numbers and must be decimal self-defining terms. ro1 and ro2 represent relational operators. If a range of calls is desired, all five parameters must be specified. Value1 must be less than value2. ro1 and ro2 must be LE or LT.

  If only one comparison value is required, the first two parameters (value1 and ro1) must be omitted and ro2 can be EQ, GE, GT, LE, or LT.

  If CALLNUM is specified, none of the other four CALLCON parameters are permitted and any STRTKEY/STOPKEY specification is ignored.

- KEYFDBK - the current value of the key feedback area in the user's PCB at the beginning of a call. The key value(s) can be expressed in either character or hexadecimal notation as indicated by the leading C or X respectively. The key itself must be enclosed in single quotes. The length of this operand value is limited by the DOS/VSE Assembler to 256 bytes.

  If a range of keys is desired, all five parameters must be coded. ro1 and ro2 must be LE or LT. Checks are not performed on the key values.

  If one key comparison value is required, the first two parameters (key1 and ro1) must be omitted. ro2 can be EQ, GE, GT, LE, or LT.

  If this parameter list is specified, none of the other four CALLCON parameters are permitted and any STRTKEY/STOPKEY specification is ignored.

- PSBNAME - the name of the PSB used by the task issuing the call. This is meaningful only in an online environment. The name specified must be from 1 to 7 characters in length. No check is made to determine whether the name is that of a valid PSB.

- DBPCBDBD - the name of the DBD directly referenced by the PCB used for the call (the value of the DBDNAME operand in the PCB macro of PSBGEN). The name must be from 1 to 7 characters in length. No check is made to determine if it is the name of a valid DBD.

- CALLFUNC - the call function specified for this call in the user's parameter list. The functions (G, GH, and UPD) are valid and result in tracing the named call functions:

  ```
  G   -- GHN, GHNP, GHU, GN, GNP, GU
  GH  -- GHN, GHNP, GHU
  UPD -- DLET, ISRT, REPL
  ```

  Also, any one of the actual DL/I call types (GHN, GN, DLET, ISRT, etc.) may be specified in place of G, GH, or UDP.

Note: When specifying the call conditions in the DLZTRACE macro, select one form of the CALLCON operand. See "CALLCON" for explanation of parameters.

```
DLZTRACE [CALLCON=/([value1,ro1,]CALLNUM,ro2,value2),]
                                  OR
         [CALLCON= ([key1,ro1,]KEYFDBK,ro2,key2),]
                                  OR
         [CALLCON=[((PSBNAME,EQ,psbname),]
                   [(DBPCBDBD,EQ,dbdname),]
                   [(CALLFUNC,EQ,func   ),])]
         [,STRTKEY=key]
         [,STOPKEY={key}]
                   {nn }
                              AND/OR
         [,]OPTION=(┌USERCALL ┐
                    │,MODTRACE│
                    │,RETRIEVE│
                    │,CURRPOS │
                    │,VSAMINTF│
                    │,BHINTF  │
                    │,INDEXTRC│
                    │,ONLINEBH│
                    └,PITRACE ┘)
         [,TYPETRC={FULL}]
                   {SHORT}
         [,TRCECON=┌(DDIRSYM,EQ,dmbname)          ┐]
                   │,(SDBPHYCD,EQ,physcode)       │
                   └,([rba1,ro1,]PSTBYTNM,ro2,rba2┘
         [,{OUTPUT={(INCORE,{32})}]
                   {        {nn} }
                   {(CICS,{32})  }
                   {      {nn}   }
                   { SYSLST      }
         [,ENVIRON={ONLINE}]
                   {BATCH }
         [,TRACSIZ={256}]
                   {nn }
         [,USREXIT=entrypt]
```

Figure 6-1. DLZTRACE Macro Options and Parameters

The CALLFUNC, DBPCBDBD, and PSBNAME parameter lists may be combined. All of the named CALLCON parameter conditions must be satisfied before the decision is made to trace the call.

STRTKEY

Specifies the first call to be traced in a sequence of calls. It is ignored if either the CALLNUM or KEYFDBK parameter lists are specified in the CALLCON operand.

If any or all of the CALLFUNC, DBPCBDBD, or PSBNAME parameter lists are specified in the CALLCON operand, these conditions must be satisfied before the STRTKEY value is compared. If the key value specified is equal to the value in the PCB key feedback area at the beginning of a call, tracing is activated for the call and all succeeding calls until a STOPKEY condition is satisfied or until DL/I terminates.

The key value may be either character or hexadecimal format as indicated by the C or X respectively. The key value itself must be enclosed in single quotes. The length of the operand value is limited to 256 bytes by the DOS/VSE Assembler.

STOPKEY

Used to stop call tracing that was started by the STRTKEY condition being fulfilled. It is ignored if CALLNUM or KEYFDBK

parameters of the CALLCON operand are specified or if the STRTKEY operand is omitted. Every call after the one that started the sequence is checked to determine if the STOPKEY condition is met. While the STRTKEY condition is operative, CALLCON condition checks are not performed. Once the STOPKEY condition is satisfied, the STRTKEY condition is again checked.

The value of the operand may be a key value in character or hexadecimal notation as, indicated by the C or X respectively. The key value itself must be enclosed in single quotes. This value is compared to the contents of the key feedback area in the PCB at the beginning of a call.

A decimal self-defining term, which indicates the number of calls to be traced from the first call that satisfies the STRTKEY conditions, may also be specified.

OPTION

May be specified with one or more parameters. Each option has a predefined set of fields or tables that are traced at predefined points in the DL/I code.

The options selected cause the corresponding trace points within the DL/I action modules to be eligible for activation based on any call condition parameters, as already described. When the trace module is called from one of these activated trace points, tracing occurs as defined for that option. A description of the OPTION parameters is shown in Figure 6-2 on page 6-6. For a detailed explanation of the exact fields to be traced (the format of the trace entry created and the trace points), see the section "Trace Entry Format" on page 6-9.

Note: For OPTION=USERCALL, a trace size (TRACSIZ) of 4096 should be specified to avoid truncation of trace entries.

TYPETRC

Enables the user to shorten the amount of information traced. For each option, a subset or short form is defined. For the description of the "full" and "short" traces as defined for each option, refer to the trace entry format descriptions in "Trace Entry Format" on page 6-9. The default is the full trace definition.

TRCECON

Allows the user to eliminate tracing for any active trace points based on internal DL/I conditions. Unless the specified conditions are satisfied at a particular trace point, tracing does not occur. The first processing step at every activated trace point is to check these conditions which must be satisfied before tracing is performed. One or more of the following parameters may be specified:

- DDIRSYM - the current name of the data base as indicated via PSTDSGA is compared with the specified DMB name. The name must be 8 bytes in length and end with the letter 'D'.
- SDBPHYCD - the current segment code as found in SDBPHYCD of the SDB corresponding to the level indicated in JCBLEV1C is compared to the user value. physcode represents a segment code and must be of the form X'nn' where nn is the hexadecimal value of the physical code.
- PSTBYTNM - the current value of PSTBYTNM is compared as indicated in this parameter. A range of RBAs or a comparison of one value may be specified. rba1 and rba2 represent relative byte addresses. The RBA values can be from 1 to 8

Options	What Traced	Type ID	Trace			
			Module	CSECT	Label	ID
USERCALL interface to DL/I	DBPCB, SSA, I/O area	01 02	DLZDLA00		GETJCB RETURN2	01 02
MODTRACE action module trace	Time	10	ALL DL/I ACTION MODULES			
RETRIEVE for get calls	JCB (some fields)	20	DLZDLRC0 DLZDLRA0	DLZSSA DLZTAG DLZLTW DLZEODC DLZGER	SSAEVALB SSAEVALM BKUPLP UNQLA SSAEVALH LSWON TAG4C LTWSSART LTWSSACB LTWSSAG LTWSSAF LTWSSAD	31 32 33 34 35 36 41 51 52 53 54 55 101 102
CURRPOS current position information	LEVTAB (active), SDBs with SDBPOSC	30	DLZDLA00		RETURN2	02
VSAMINTF interface to VSAM	Request to VSAM, VSAM return information	50	DLZDBH02		DOVGET DOVPUT DETIOERR CONTPUT	01/03 02 04 05
BHINTF action module interface to buffer handler	Request to buffer handler. Buffer handler return information	61/62	DLZDLRB0 DLZDLD00 DLZDDLE0 DLZDHDS0 DLZGGSP0 DLZFRSP0 DLZRRHM0 DLZRCHP0 DLZRCHB0 DLZDXMT0 DLZCPY10	DLZBH	DLZBH BHCALL GOTOFUNC FIXBTMP NOBKUT GTBASEBK BRAD01 BTMPLIED BRAD04 NOITDONT XIT FIXBTMP BRAD01 LOCAN HIORBOTH BTCHONLY XIT GOTOBUFF BHCALL	21/22 02/03 04/05 06/07 08/09 10/11 12/13 14/15 16/17 18/19 20/21 22/23 24/25 26/27 28/29 30/31 32/33 34/35 02/03
INDEXTRC index procedure	Request to index maintenance	70	DLZDXMT0			01
ONLINEBH online trace	PST prefix, BFFR prefix	80	DLZDBH00 DLZDBH02		PSEUDINT AFTW	06 08
PITRACE program isolation trace	Queueing CALL parameters	90	DLZQUEF0		QENQDEQ QRETURN2 QWAIT 1 QWAIT2	01 02 03 04

Figure 6-2. OPTION Parameter Description

hexadecimal characters and are coded without an X or single quotes. The values are right-justified if there are fewer than 8 characters.

If a range is desired, all five parameters must be specified. ro1 and ro2 must be LE or LT.

If a range is not desired, the first two parameters (rba1 and ro1) must be omitted. ro2 can be EQ, GE, GT, LE, or LT.

OUTPUT

Controls the destination of the tracing results. There are three possibilities:

- INCORE - the tracing entries are kept in a table in virtual storage. If the table becomes full, a wrap-around condition occurs and the oldest entry is overlaid. 'nn' specifies the approximate number of entries to be kept in the table. It can be a decimal value between 1 and 32,767. This number is multiplied by the sum of the TRACSIZ value and the header size to determine the size of the table. The default is 32. When this default and the default of 256 for the TRACSIZ operand is used, a table size of approximately 8K results which for most options actually provides space for 200 to 300 entries. If the calculated INCORE table size is not available in virtual storage, half of the amount is requested repeatedly until space is obtained, or insufficient space for at least four entries is available. If no space is available, no tracing is performed.

  When the INCORE option is chosen, the resulting table is built from the bottom up. That is, the first entry is placed in the high address space. The header portion is then followed by the data portion of the entry. Pointers to the INCORE table are found in the SCD extension at labels:

  ```
  SCDETRTB = current entry
  SCDETRTE = trace table end + 1
  SCDETRTS = trace table start
  ```

- CICS/VS - the CICS/VS option is similar to the INCORE option, except that the trace entries are placed in the table at the low address end. When the table becomes full, it is written out to a CICS/VS extrapartition dataset before wrap-around occurs (see "Defining the Extrapartition Dataset"). This provides a means of tracing in an online environment without being restricted to the size of an incore table. The output can be directed to either a tape or a disk file and at a later time printed by the trace print utility (DLZTPRT0). Tracing will not take place while I/O is in process. If an error is passed back, a message is issued and tracing is disabled. The task continues to run but without tracing.

  The maximum incore table size for this option is 32763 bytes, and is calculated as explained under INCORE.

  The user is responsible to ensure that the trace records will fit on the specified device. The record size is determined by the size of the incore table and the BLKSIZE specified by the Destination Control Table (DCT). The incore table size may be reduced dynamically during execution if sufficient space is not available for the size requested. To increase performance and keep the number of I/O operations to a minimum, the user should provide enough GETVIS space to provide for as large a table as the device will accept and is practical.

  Because the record format is VARBLK it may be possible to fit several incore tables into one record. This will also limit the number of I/O operations in the extrapartition dataset.

  The CICS/VS device must be opened before it can be used for storing tracing entries (see "Opening the CICS/VS Device" on page 6-29).

- SYSLST - each trace entry is printed on SYSLST at the time it is created. This option may not be specified in an online environment, unless the system is in a debug mode with only one DL/I task active at one time.

If the formatted dump option and the INCORE option have both been selected, the formatted dump routine invokes the trace table print routine to format and print the latest 10 entries of this table.

ENVIRON

Specifies the type of environment that the trace program will be assembled and executed in.

- ONLINE - indicates that CICS/VS macros and modules are available for the assembly and execution of the trace program. This is the default.
- BATCH - indicates that the trace program is to be assembled and run in a purely batch environment, and the use of CICS/VS macros and modules is prohibited.

TRACSIZ

Specifies the maximum decimal number of bytes (nn) required to build the largest trace entry for the options selected. The size of the trace entry header need not be included. Refer to the format description of the individual trace entries in the section "Trace Entries Format" for information on how to determine the size. The value specified must be between 8 and 4096 bytes. The default is 256 bytes which for most cases should be sufficient.

If the INCORE or CICS/VS parameter of the OUTPUT operand is specified, this number is used to calculate the size of the table. Only that space which is required to build the trace entry is actually used. If the resulting table size is not large enough to hold one particular entry, that entry is omitted from the table.

If the SYSLST option is selected, the TRACSIZ value is the size of the work area used to create each trace entry. Any trace entry which results in a length larger than this size is not printed.

Beginning with DL/I Release 1.7, the trace facility formats the header portion of the entry and prints the following message:

••••• THIS ENTRY WAS TRUNCATED BECAUSE OF INSUFFICIENT SPACE TO BUILD TRACE ENTRY •••••

If the information in the header is not sufficient to diagnose the problem, reassemble the trace module with a larger TRACSIZ.

If the third user exit option is used to create trace entries, the value of the TRACSIZ operand is used to acquire work space for the user exit routine in which to build its trace entry.

USREXIT

Because it is impractical to create a generalized trace module that can track all situations that might arise in any user environment, the DL/I trace facility provides a user exit to allow you to perform and control tracing of specific conditions.

The user exit allows you to influence three major decisions made by the DLZTRACE module: These are:

1. Is the current call eligible for tracing?
2. Is the current trace point activated?
3. What information should be traced at this selected trace point?

The entry point name of the user exit routine must be specified in the DLZTRACE macro in the USREXIT operand. The exit routine, which should be written in Assembler Language, need not be reentrant. It must not issue any supervisor or I/O macros.

The exit routine must be assembled and cataloged into the relocatable library. When the DLZTRACE module is link-edited, this module must be included via AUTOLINK or an INCLUDE statement.

The trace module performs as indicated in the macro parameters. If an optional user exit routine is specified, there are three points at which it can be given control.

The exit routine programmer is responsible for knowing what fields or table entries are valid at the time the exit routine receives control. Therefore, a good knowledge of the internal logic of DL/I is necessary to use the trace user exit functions.

See "User Exit" on page 6-30 for more information on the user exit and samples of coding user exit routines.

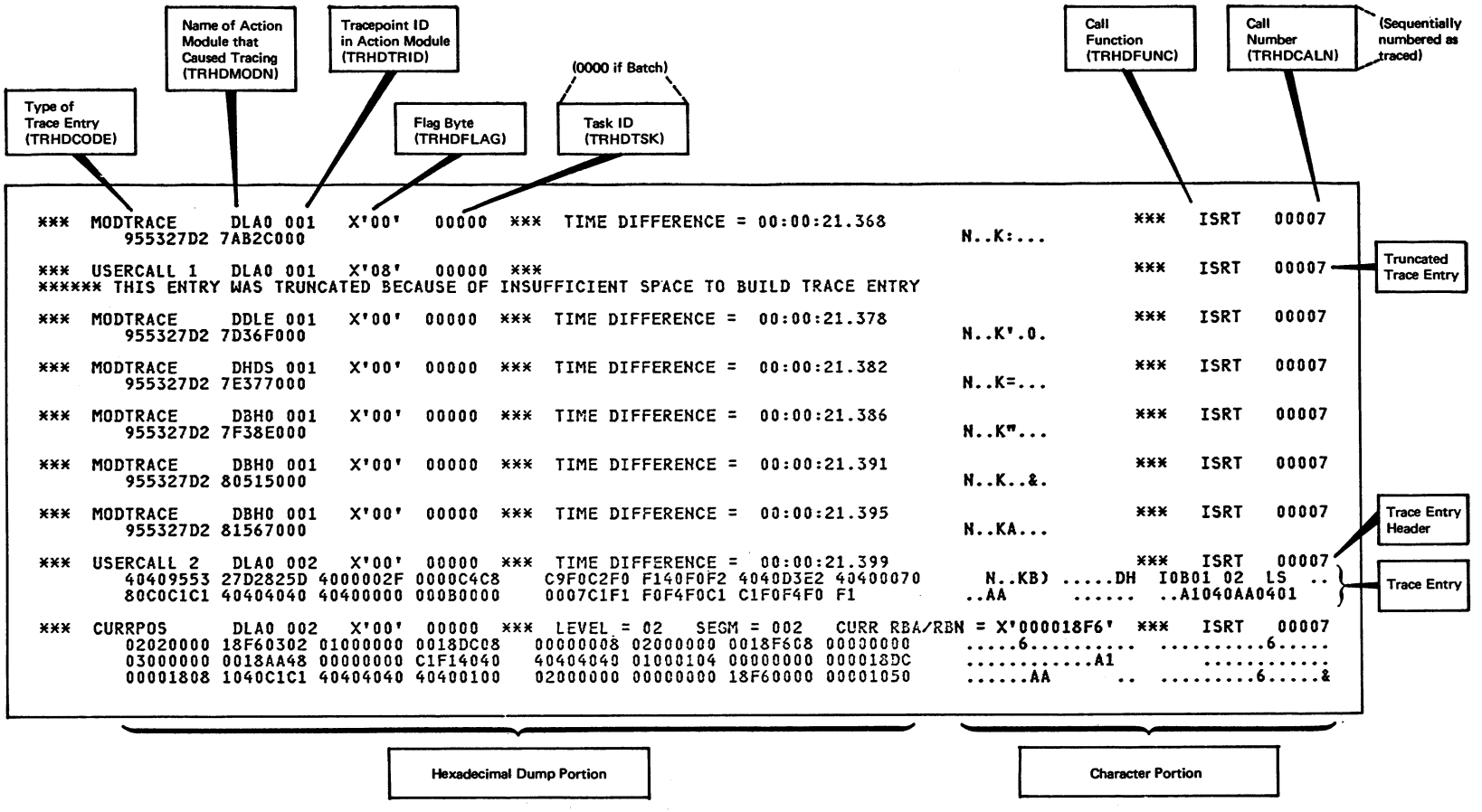
TRACE ENTRY FORMAT

Each entry in the trace entry is of variable length. Boundary alignments are not guaranteed for any field. The macro DLZTRENT can be used to generate DSECTs for the trace entries. The contents of the trace entry vary according to the option(s) selected.

The trace entry header (TRACEHDR) DSECT is usable if the trace runs in core wrap around mode. The header line for SYSLIST output is composed from TRACEHDR. The header DSECT is in macro DLZTRENT and may be used when writing user routines.

The contents of the header line and the format of the trace entry header (TRACEHDR) is shown in Figure 6-3 on page 6-10.

Figure 6-3. Sample DL/I Trace Output



Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	2	TRHDELEN		Length of the trace entry (including the header)
2 2	2	TRHDCALN		Call number relative to 1
4 4	4	TRHDMODN		Name of the action module that caused the trace
8 8	3	TRHDTSK		Packed CICS task ID if running online
11 B	1	TRHDTRID		Trace point ID in the action module
12 C	1	TRHDFUNC		Code for user's call function
			00	= GU
			01	= GN
			02	= GNP
			03	= GHU
			04	= GHN
			05	= GHNP
			06	= ISRT
			07	= DLET
			08	= REPL
			09	= CKPT
			0A	= PCB
			0B	= UNLD
			0C	= GSCD
			0D	= TERM
13 D	1	TRHDCODE		Type of trace entry
		TRHDUC1	X'01'	USERCALL - type 1 (start of call)
		TRHDUC2	X'02'	USERCALL - type 2 (end of call)
		TRHDAMOD	X'10'	MODTRACE
		TRHDRETR	X'20'	RETRIEVE
		TRHDCPOS	X'30'	CURRPOS
		TRHDVSAM	X'50'	VSAMINTF
		TRHDBH1	X'61'	BHINTF - type 1 (before call)
		TRHDBH2	X'62'	BHINTF - type 1 (after call)
		TRHDINDX	X'70'	INDEXTRC
		TRHDOLBH	X'80'	ONLINEBH
		TRHDPITR	X'90'	PITRACE
			X'A0'	••USER•• - Traced by user
14 E	1	TRHDFLAG		Flag byte
		TRHDSHRT	X'80'	This trace entry is the short form
		TRHDUEX1	X'40'	User exit 1 reversed decision
		TRHDUEX2	X'20'	User exit 2 reversed decision
		TRHDUEX3	X'10'	User exit performing tracing
		TRHDLN	X'08'	Trace entry truncated

Note:
The remainder of each entry depends on the option being traced.

Figure 6-4. Trace Entry Header Format

USERCALL Type 1 (X'01')

The trace point is DLZDLA00 (Call Analyzer) - at the beginning of user call validation. (Some user errors could be detected before this trace point, in which case no tracing takes place).

The call function code in the trace entry serves as the short form.

The format of the TRUSRCL1 DSECT is:

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	8	TRU1TIME		Time the call was made (from STCK instruction)
8	8	2	TRU1PCBL		Length of DBPCB including key feedback
10	A	2	TRU1IOLN		Length of I/O area (0 if not traced)
12	C	2	TRU1SSAL		Length of SSAs (0 if none)
14	E	Var	TRU1PCB		DBPCB with key feedback
--	-	Var	TRU1IOAR[1]		I/O area contents (not traced for GET call.) I/O tracing is performed for only DLET, ISRT, or REPL calls. The length of the I/O area traced is the largest of: • Longest segment length (maximum if variable) • Longest concatenated segment length • Longest path call length
--	-	Var	TRU1SSA		SSAs. If more than one SSA is present, at least one blank separates them.

[1] This field is not part of the short form.

USERCALL Type 2 (X'02')

The trace point is DLZDLA00 (Call Analyzer) - at the end of processing before control is given to PRH.

I/O area tracing is only performed when data is returned to the user. The length of the I/O area is the length of the data returned to the user.

The format of the TRUSRCL2 DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	2	TRU2STC		Returned status code (DBPCBSTC)
(-------- End of short form --------)				
2 2	8	TRU2TIME		Time at the end of the call
10 A	2	TRU2PCBL		Length of DBPCB
12 C	2	TRU2IOLN		Length of I/O area (0 if not traced)
14 E	Var	TRU2PCB		DBPCB with key feedback
-- -	Var	TRU2IOAR		I/O area contents

MODTRACE (X'10')

The trace points are:

- DLZDLA00 (Call Analyzer) - on entry after CKCALL.
- DLZDLR00 (Retrieve) - on entry.
- DLZDLD00 (Delete/Replace) - on entry.
- DLZDDLE0 (Load/Insert) - on entry.
- DLZDBH00 (Buffer Handler) - on entry.
- DLZDHDS0 (Space Management) - on entry.
- DLZDXMT0 (Index Maintenance) - on entry.
- In buffer handler before VSAM call (module ID is 'VSAM').
- DLZCPY10 (FLS copy) on entry.

The format of the TRMODTRC DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	4	TRMDTIME		Time at trace point (from STCK instruction)

RETRIEVE (X'20')

The trace points are:

- DLZTAG - when completing a level.
- DLZSSA - when completing a level.
- DLZLTW - when accepting a level.
- DLZEODC - end of data base condition.
- DLZGER - not found condition.

The format of the TREETRV DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	1	TRRTJCD		JCB flags (JCBCODE)
			X'80'	On log child-log parent insert, log parent is present
			X'40'	Deferred delete required
			X'20'	Retrieved deleted segments
1 1	1	TRRTJL1C		Level number pointed to by JCBLEV1C
2 2	4	TRRTLVT		Input for DLZSKPG (JCBLVT...)
6 6	4	TRRTBGBF		Retrieve work field (BEGBUF)
10 A	4	TRRTCTTR		Retrieve work field (CURTTR)
14 E	4	TRRTPRSW		Retrieve work field (PROCSW)
18 12	4	TRRTKPIT		Retrieve work field (KEEPIT)
22 16	1	TRRTINDG		VL-UDC flags (DSGINDG)
23 17	1	(reserved)		
24 18	68	TRRTREGS		Contents of registers 14 - 12

CURRPOS (X'30')

The trace point is DLZDLA00 (Call Analyzer) - at the exit from the call analyzer.

The format of the TRCURPOS DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	1	TRCPCLEV		Current level number (LEVLEV)
1 1	1	TRCPCPC		Current segment code (LEVPC)
2 2	4	TRCPLTTR		Current position (LEVTTR)(RBA or RBN)
(-------- End of short form --------)				
6 6	1	TRCPNOLV		Number of level entries in TRCPLEVL
7 7	1	TRCPNSDB		Number of SDB entries in TRCPSDBS

The first three fields of the TRCURPOS DSECT contain information for the current level only (as found via JCBLEV1C). The level table information is traced for every active level plus 1. TRCPNOLV contains the number of levels traced.

The format of the level entries found beginning in TRCPLEVL is shown as follows for the TRLEVEL DSECT:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 (0)	1	TRLVLEV		Level number (LEVLEV)
1 (1)	2	TRLVSGOF		Offset to segment (LEVSEGOF)
3 (3)	4	TRLVTTR		Current RBA or RBN (LEVTTR)
7 (7)	1	TRLVF1		Flag byte LEVF1
			X'80'	Segment at this level is newly deleted
			X'40'	This level table entry is empty
			X'20'	Segment at this level holds status
			X'10'	Segment at this level in heirarchial path
			X'08'	Segment at this level moved to user
			X'04'	Segment is last of type for parent
			X'02'	Segment is first of type for parent
			X'01'	Last level table for PCB
8 (8)	1	TRLVF2		Flag byte LEVF2
			X'80'	Used by retrieve
			X'40'	Level has not found position for higher level
			X'20'	EOD flag
			X'10'	LEVTAB has been modified
			X'08'	Used by retrieve
			X'04'	Used by retrieve
			X'02'	Used by retrieve
			X'01'	Used by retrieve
9 (9)	2	TRLVUSOF		Offset to segment in I/O area (LEVUSEOF)
11 (B)	1	TRLVF3		Flag byte LEVF3
			X'08'	Pseudo SSA filling gap
			X'04'	At least one member qualified on data
			X'02'	Every Boolean set has at least one key field

In the TRLEVEL DSECT for each level, except the last, information from the corresponding SDB is traced. In addition, if the corresponding SDB has a target (nonzero SDBTARG) and the target has a current position (nonzero SBDPOSC), the target is also traced. This means that for one level, one or more SDBs may be traced. Each target SDB recurs in the list immediately after its parent SDB. A target SDB is easily identified because it has no name (TRSDBSYM).

The format of the TRSDB DSECT is:

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	8	TRSDBSYM		Segment name (SDBSYM)
8	8	1	TRSDBF3		Flag byte SDBF3
				X'80'	Read only
				X'40'	Insert
				X'20'	Replace
				X'10'	Delete
				X'08'	Key only
				X'04'	Path only
				X'02'	Exclusive
				X'01'	Load
9	9	1	TRSDBF4		Flag byte SDBF4
				X'40'	Secondary index is main processing sequence
				X'10'	Field is destination parent
				X'04'	CI split HISAM KSDS
				X'02'	Position lost
				X'01'	Temporary SW for replace. Data changed.
				X'01'	Field in logical child
10	A	1	TRSDBPC		Physical code (SDBPHYCD)
11	B	1	TRSDBTFG		Target relationship code (SDBTFLG)
				X'C0'	Segment is physical parent of target of SDBPARA
				X'80'	Segment is physical parent of SDBPARA
				X'40'	Segment is physical child of target of SDBPARA
				X'20'	Segment points to logical parent with physical key
				X'10'	SDB is a generated SDB or a SDB for a physical pair
				X'08'	Segment points to a physical parent
				X'04'	Segment is retrieved via index
				X'02'	Segment points to logical child
				X'01'	Segment points to logical parent
12	C	4	TRSDBPRV		Previous position (SDBPOSP)
16	10	4	TRSDBCUR		Current position (SDBPOSC)
20	14	4	TRSDBNXT		Next position (SDBPOSN)
24	18	1	TRSDBORG		Data base organization (SDBORGN)
				X'44'	This segment is root of index
				X'20'	This segment is HDAM
				X'10'	This segment is HIDAM
				X'05'	This segment is SHISAM
				X'04'	This segment is HISAM
				X'02'	This segment is HSAM
				X'01'	This segment is SHSAM
25	19	1	TRSDBPT		Physical pointer flag (SDBPTDS)
				X'80'	This logical parent segment has a counter
				X'40'	This segment has a physical twin forward pointer
				X'20'	This segment has a physical twin backward pointer
				X'10'	This segment has a physical parent pointer
				X'08'	This segment has a logical twin forward pointer
				X'04'	This segment has a logical twin backward pointer
				X'02'	This segment has a logical parent pointer

VSAMINTF (X'50')

The trace point is DLZDBH00 (Buffer Handler) - after VSAM GET or VSAM PUT

The format of the TRVSAMIF DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	1	TRVSREQ		Request (RPLREQ)
			X'24'	Put locate
			X'20'	Verify
			X'1C'	FORCIO
			X'18'	ENDREQ
			X'14'	Check
			X'10'	Insert request
			X'0C'	Update request
			X'0C'	Put request
			X'08'	Erase request
			X'04'	Get request
			X'00'	Point request
1 1	1	TRVSFBK1		Return code = (RPLFDBK1)
			X'0C'	Physical error
			X'08'	Logical error
			X'04'	Concurrent request on same RPL
			X'00'	No error
2 2	1	TRVSFBK3		Return code = (RPLFDBK3)
			No error (RPLFDBK1 = 0)	
			X'04'	EOV called during request
			Logical errors (RPLFDBK1 = 8)	
			X'04'	End of data set reached
			X'08'	Duplicate record
			X'0C'	Sequence error
			X'10'	No record found
			X'14'	Data ALR in exclusive control
			X'18'	Volume is not mounted
			X'1C'	Data set cannot be extended
			X'20'	Invalid RBA specified
			X'24'	No key range specified for record
			X'28'	Insufficient virtual storage
			X'2C'	User buffers too small
			X'40'	PLH in use (no string available)
			X'44'	Access type not requested at open
			X'48'	Keyed request for ESDS
			X'4C'	Address or CNB insert for KSDS
			X'50'	Invalid erase request
			X'54'	Invalid specification of locate mode
			X'58'	Positioning error
			X'5C'	No get UPD issued
			X'60'	Key change for update
			X'64'	Length change for address update
			X'68'	Invalid or conflicting RPL option specified
			X'6C'	Improper record length specified
			X'70'	Improper generic key length specification
			X'74'	Invalid request during data set loading

(continued on next page)

TRVSAMIF DSECT (continued)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				Physical errors (RPLFDBK1 = 12)
			X'04'	Read error in data set
			X'08'	Read error in index set
			X'0C'	Read error in sequence set
			X'10'	Write error in data set
			X'14'	Write error in index set
			X'18'	Write error in sequence set
		(-------- End of short form --------)		
3 (3)	1	TRVSLARG		Length of TRVSARG
4 (4)	4	TRVSRBA		RBA returned (RPLRBA)
8 (8)	4	TRVSAREA		Pointer area (RPLAREA)
12 (C)	8	TRVSTIME		Clock time call completed
20 (14)	2	TRVSOPCD		Option codes (RPLOPTCD)
			The first option code byte equates	
			X'80'	Keyed access
			X'40'	Addressed access
			X'20'	Sequential
			X'10'	Direct processing
			X'08'	Asynchronous
			X'04'	Skip sequential access
			X'02'	CINV access (by RBA)
			X'01'	Update
			The second option code byte equates	
			X'80'	Search key greater than or equal to
			X'40'	Generic key request
			X'20'	Note string position
			X'10'	No update
			X'08'	Locate mode
			X'04'	User buffers
22 (16)	Var	TRVSARG		RBA or key requested (from RPLARG)

BHINTF Type 1 (X'61')

The trace points are:

- DLZDLR00 (Retrieve) - before call to buffer handler.
- DLZDLD00 (Delete/Replace) - before call to buffer handler.
- DLZDDLE0 (Load/Insert) - before call to buffer handler.
- DLZDXMT0 (Index Maintenance) - before call to buffer handler.
- DLZDHDS0 (Space Management) - before call to buffer handler.
- DLZCPY10 (FLS Copy) - before call to buffer handler.

The format of the TRBHT0 DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 (0)	1	TRBHTFUN		Caller's request (PSTFNCTN). Equates for buffer handler (DLZDBH00) function codes:
			X'01'	Locate control interval with CI RBA
			X'02'	If HD, locate a data CI RBA pointing to a segment. If HISAM or HIDAM INDEX read a record by RBA from a KSDS.If HISAM, read a record by RBA from an ESDS.
			X'03'	Get a buffer space
			X'04'	Free buffer space
			X'04'	Mark buffer empty
			X'05'	If HD, mark a buffer containing data altered. If HISAM or HIDAM INDEX,write a record by RBA to a KSDS. If HISAM, write a record by RBA to an ESDS.
			X'06'	Locate a CI with an RBA pointing to a segment and mark buffer altered
			X'07'	Purge all buffers altered by a task
			X'08'	Write a new record to HISAM ESDS
			X'09'	Get a record from KSDS equal or high
			X'0A'	Erase a record in a KSDS
			X'0B'	Get the next record from KSDS
			X'0C'	Get the first record of a KSDS
			X'0D'	Insert a record into KSDS by key
			X'0E'	Insert a record sequentially into KSDS
(-------- End of short form --------)				
1 (1)	1	TRBHTKYL		Length of the key at TRBHTKEY
2 (2)	2	TRBHTDMB		DMB number (PSTDMBNM)
4 (4)	4	TRBHTBLK		Relative block number (PSTBLKNM)
8 (8)	4	TRBHTBYT		RBA or RBN (PSTBYTNM)
12 (C)	4	TRBHTDAT		Address of data in buffer (PSTDATA)
16 (10)	4	TRBHTDSG		Address of the DSG portion of the JCB (PSTDSGA)
20 (14)	Var	TRBHTKEY		Key if PSTFNCTN = PSTSTLEQ

Note: If PSTFNCTN = PSTSTLEQ, PSTBYTNM contains the address of the key to search for.

BHINTF Type 2 (X'62')

The trace points are:

- DLZDLR00 (Retrieve) - after call to buffer handler.
- DLZDLD00 (Delete/Replace) - after call to buffer handler.
- DLZDDLE0 (Load/Insert) - after call to buffer handler.
- DLZDHDS0 (Space Management) - after call to buffer handler.
- DLZDXMT0 (Index Maintenance) - after call to buffer handler.
- DLZCPY10 (FLS Copy) - after call to buffer handler.

The format of the TRBHFROM DSECT is:

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	1	TRBHFRC		Buffer handler return code (PSTRTCDE)
				X'00'	No errors
				X'04'	RBA is beyond the end of the data set
				X'08'	I/O error
				X'08'	Permanent read error
				X'0C'	No space in data set for additions
				X'10'	An illegal call was made
				X'14'	No record found (retrieve by key)
				X'18'	New block was created in buffer pool
				X'1C'	Not enough space in buffer pool
				X'20'	Size of requested buffer exceeds size of buffers in any subpool
				X'24'	End of data set. No record returned
				X'28'	Key or RBA is higher than highest key or RBA in data set
				X'2C'	End of data set reached on a request issued by open
(-------- End of short form --------)					
1	1	2	TRBHFDMB		DMB number (PSTDMBNM)
3	3	2	TRBHFOFF		Offset to RBA from PSTDATA (PSTOFFST)
5	5	4	TRBHFBLK		Block number (PSTBLKNM)
9	9	4	TRBHFBYT		RBA (PSTBYTNM)
13	D	4	TRBHTDAT		Address of data in buffer (PSTDATA)
17	11	4	TRBHFBFA		Address of the buffer prefix (PSTBUFFA)

INDEXTRC (X'70')

The trace point is DLZDXMT0 (Index Maintenance) - at the beginning.

The format of the TRINDEX DSECT is:

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0 0	1	TRIXFNCT		Request to index maintenance (PSTFNCTN)
			X'A0'	Delete
			X'A1'	Replace
			X'A2'	Insert
			X'A3'	Unload
1 1	1	TRIXWKT4		Special request (PSTWRKT4 - 1st byte)
			X'01'	Normal delete
			X'02'	Delete primary index only
			X'03'	Do not delete primary index
			X'04'	Do not delete HIDAM index
(-------- End of short form --------)				
2 2	2	TRIXDMB		DMB number (PSTDMBNM)
4 4	4	TRIXBYT		ISS RBA (PSTBYTNM)
8 8	1	TRIXSC		Segment code of ISS (DMBSC)

ONLINEBH (X'80')

The trace points are:

- PSEUDINT routine - before WAIT is issued for a task because The interlock detection matrix is full.
- ISSWAIT routine - before WAIT is issued for a task because it requested a buffer owned by another task.
- AFTW routine - before WAIT is issued for a task because it needed a data base already in use (ACB busy).

The format of the TRONLINE DSECT is:

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	1	TROLIND		Schedule and dispatch indicator (PPSTIND)
				X'80'	Waiting for I/O
				X'40'	Cannot schedule due to segment intent conflict
				X'20'	Cannot schedule due to task count
				X'10'	Task enqueued by buffer handler
				X'04'	Current task
				X'01'	Task scheduled
1	1	1	TROLID		Task ID (PPSTID)
(-------- End of short form --------)					
2	2	2	TROLECB		ECB (PPSTECB)
4	4	2	TROLEXCI		Existing CI (PPSTEXCI + 2) enqueue/dequeue pointer
6	6	2	TROLPECI		Pending CI (PPSTPECI + 2) enqueue/dequeue pointer
8	8	2	TROLSUPO		Subpool space (PPSTSUPO + 2) enqueue/dequeue pointer
10	A	2	TROLMATR		Matrix space (PPSTMATR + 2) enqueue/dequeue pointer
12	C	2	TROLCHAI		Chain field for pending CI (PPSTCHAI + 2)
14	E	2	TROLPST		PST pointer for BFFRPST enqueue/dequeue
16	10	2	TROLNPST		PST pointer for BFFRNPST enqueue/dequeue
18	12	6	TROLCIID		CI identification (BLKNUM-DMBNM) (BFFRCIID)
24	18	1	TROLSW		Switch
				X'80'	Buffer on write chain
				X'40'	Buffer being written
				X'20'	Buffer being read
				X'10'	Buffer empty
				X'08'	Buffer waiting for PRED being written
				X'04'	Buffer has permanent write error
				X'02'	Existing CI ID enqueued
				X'01'	Pending CI ID enqueued
25	19	1	TROLFUNC		Caller's request (PSTFNCTN)
26	1A	4	TROLBLKN		Block number (PSTBLKNM)

PITRACE (X'90')

The trace points are:

- DLZQUEF0 (Queueing Facility) - on entry.
- In QWAIT - if a wait was required.
- In QWAIT - if this task was selected for termination due to a deadlock condition.
- DLZQUEF0 - on exit, if a wait was required.

The format of the TRPITRC DSECT is:

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	4	TRPIRBA		RBA (PSTWRK2)
4	4	2	TRPIDMB		DMB number (PSTWRK2 + 4)
6	6	1	TRPIACB		ACB (PSTWRK2 + 6)
7	7	1	TRPIPSC		Physical segment code (PSTWRK2 + 7)
8	8	1	TRPIFNC		Caller's function (PSTFNCTN). Equates for program isolation (DLZDBH00) function code are:
				X'00'	Enqueue
				X'04	Verify
				X'08'	Dequeue
				X'0C'	Purge
9	9	1	TRPIQLEV		Current level (PSTQLEV)
				X'00'	Read only level
				X'04	Update level
				X'08'	Exclusive level
10	A	1	TRPITKID		Task ID (PPSTID)
11	B	1	TRPIRTCD		Program isolation return code (PSTRTCDE)
				X'01'	Wait was required
				X'02'	Other owners present
				X'04'	Terminated due to deadlock
				X'08'	Terminated due to bad call
				X'10'	Terminated due to insufficient storage
				X'40'	Secondary index updated
				X'80'	Primary index updated
12	C	4	TRPIPSTP		PST pointer
16	10	4	TRPIRTAD		Return address

TRACING IN THE BATCH ENVIRONMENT

The DL/I trace is activated if TRACE=modname is specified in the DL/I parameter statement as:

```
DLI,pgmname,psbname,buff=,[TRACE=modname]
```

The named module is loaded from the core image library by batch initialization. Tracing is performed for the program execution according to the parameters specified in the generation of DLZTRACE.

Which Calls to Trace

Determine which calls are to be traced. The decision will be one of the types indicated below. Find the type and then note which trace macro operands should be coded. Refer to Figure 6-1 on page 6-4 for detailed syntax specification information. Only DL/I data base calls can be traced; PCB, TERM, UNID, GSCD, and online system calls cannot be traced.

- If all calls are to be traced, do not code the CALLCON, STRTKEY, STOPKEY, or TRCECON operands.
- If specific calls that have similar characteristics are to be traced, but are not necessarily sequentially issued to DL/I, determine the class they fall into and code the indicated CALLCON operand parameter as shown in Figure 6-5 on page 6-26.
- If a range of calls are to be traced (for example, each of 10 successive calls) identified by either:
  - The relative call numbers of the first of the range and the last. For example, if the 6th through the 15th calls are to be traced, in the CALLCON operand, specify:

    ```
    (6,LE,CALLNUM,LE,15)
    ```

  - The key feedback value in the PCB at the start of the first call of the range. Code the STRTKEY operand. In addition, the beginning of the range can be further qualified by the DBPCBDBD and/or CALLFUNC parameter lists of the CALLCON operand. To identify the last call of the range, specify the STOPKEY operand. In STOPKEY, specify either the number of calls to be traced, or the value of the PCB key feedback area at the end of the last call to be traced.
- If one specific call is to be traced, identify this call in the same manner as the first call of a range as described above.

  If the problem being traced does not occur with easily identifiable calls, but with certain internal DL/I conditions, another method is available to define when tracing should occur. The following specifications are possible:

  - Trace at certain points within DL/I execution whenever a specific physical file is being processed. For example, to perform tracing when the secondary index data base SINDEX@D is being used to service a call, specify:

    ```
    TRCECON=(DDIRSYM,EQ,SINDEX@D)
    ```

  - Trace those calls that work with a particular segment type. For example, to perform tracing when the physical code of the current SDB (as found using JCBLEV1C and LEVSDB) is equal to X'0A' at the activated trace points, specify:

    ```
    TRCECON=(SDBPHYCD,EQ,X'0A')
    ```

Similar Characteristic	Parameter List of CALLCON Operand
A similar value in the key feedback area of the PCB at the beginning of each call.	Code the KEYFDBK parameter list.
A specific PCB identified by by the name of the DBD it references (the DBDNAME operand of the PCB macro in PSBGEN).	Code the DBPCBDBD parameter list.
A similar call function used by the calls. This option allows groups of calls to be identified, for example: all get calls, all hold calls, or all update calls.	Code the CALLFUNC parameter list.
A combination of a specific PCB used in making the calls and the call function.	Code the DBPCBDBD and CALLFUNC parameter lists.

Figure 6-5. Batch Environment Tracing

- Trace those calls in which the RBA is PSTBYTNM is a specified value. For example, specify:

  TRCECON=(00004120,PSTBYTNM,LE,5530)

Combinations of the above operands are possible. For example, the following instructions can be given to the trace module: trace if DDIRSYM is DMBNAM@D and the segment in that data base has code X'03' and the current value in PSTBYTNM is greater than 00001280.

The parameter values are checked during the execution of the call. At some trace points, the values may not be meaningful. For example, at the very beginning of a user call the PSTBYTNM value may be zero, so no tracing occurs, unless the zero value happened to satisfy the PSTBYTNM value(s) specified in the tracing macro.

What Information to Trace

There are several ways to define what information should be traced at certain points of the DL/I execution. The selection of one or more of these options should be based on the type of problem that has occurred. For example:

- If the problem has not been isolated as occurring within DL/I, or if the user call receives unexpected return information, the USERCALL option should be used to trace the input the application program is giving to DL/I and the resulting output. The problem can also be further investigated by specifying CURRPOS which gives the user's current position within the data base.
- If the problem is isolated as occurring within the retrieve module (DLZDLR00) or concerns insert positioning, the RETRIEVE option provides information concerning the events of the call.

- If the data base does not appear to have been updated as the user requested, the BHINTF and/or VSAMINTF options can be used to trace exactly which calls have been made to the DL/I buffer handler (DLZDBH00) and/or to VSAM. The return status in each case is given.

- If the secondary (or primary) indexes appear to have been incorrectly updated, specify the INDEXTRC option. This provides information concerning the updates to index data bases.

- If it is necessary to trace the path the call takes through DL/I, the MODTRACE option lists the time of entry to each module.

If the situation does not fit any of the above and assuming that many calls need not be traced, all of the options (except ONLINEBH) can be specified. Some options such as RETRIEVE and CURRPOS, could result in large trace listings if many calls are traced. The CALLCON, STRTKEY/STOPKEY, and/or TRCECON operands may be used to omit the unnecessary or uninteresting calls. In addition, a SHORT trace form is available for all options.

Batch Trace Output

The user should decide which storage medium to use for the tracing output. The following possibilities are available:

- The SYSLST parameter of the OUTPUT operand causes each trace entry to be printed on SYSLST as it is being created. Thus, whenever a DL/I action module call to the trace facility is made, the information is printed on SYSLST before control is returned to the calling module. This option is the most useful for a debugging situation when no abnormal termination dump is expected.

- The INCORE option of the OUTPUT operand builds a table of user-specified size to hold all trace entries. This table is never written onto any output device, except if a storage dump is produced. When the table is filled with entries, a wrap-around condition occurs and the oldest entry in the table is overlaid. A table large enough to hold at least one entry must be defined.

Figure 6-6 on page 6-28 is a sample job. Many of the possible options are used. Output is directed to SYSLST. A selection of special calls is not done.

TRACING IN THE ONLINE ENVIRONMENT

The situation in an online environment is different from that in batch; therefore, the purpose of tracing may also be different. There are two major reasons to trace online DL/I calls:

1. To collect information continually in case a future error occurs by means of a general trace running continuously.

2. To aid in debugging an online application program or to trace a known system failure by means of a specific trace.

Tracing Control

In the online environment, several trace modules may be loaded at initialization by entering their names into the CICS/VS PPT. Actual selection of the trace module to be used is done by the TSTR system call.

To control tracing in the online environment, two system calls are available. After the defined tracing modules have been link-edited into the core image library, a user-written application program can issue the TSTR (trace start) system call naming

```
Col 1                                                          Col 72
// JOB ASSEMBLY TRACEMOD
// OPTION CATAL,NODECK
       PHASE TRACEALL,*
// EXEC ASSEMBLY,SIZE=60K
       DLZTRACE OPTION=(USERCALL,MODTRACE,RETRIEVE,CURRPOS,          *
             VSAMINTF,BHINTF,INDEXTRC),OUTPUT=SYSLST
       END
/*
// EXEC LNKEDT
/&
```

Figure 6-6. DLZTRACE Sample Job

the phase name of the desired trace module. This causes the tracing module to be loaded and activated. Tracing then begins with the next call to DL/I that satisfies the user-specified conditions.

To disable the tracing facility, the TSTP (trace stop) system call is used. If the trace entries are being accumulated INCORE when TSTP is issued, the space for the table is released and is no longer available.

See the DL/I DOS/VS Data Base Administration manual for details of the formats and return conditions of the TSTR and TSTP calls, along with examples for using the calls.

General Trace

It is assumed that the user wants to trace all DL/I calls, and the CALLCON, STRTKEY/STOPKEY and TRCECON operands should be omitted from the DLZTRACE macro generation.

The OPTION parameter selected depends on the suspected problem:

- If the timing seems to be the problem, in other words, tasks seem to be "suspended", the ONLINEBH option should be selected.
- If bad data base updates have occurred, the BHINTF and/or VSAMINTF options should be selected.
- The other options are available, but probably do not help solve a general problem.

To limit the amount of information traced at each trace point, the SHORT parameter of the TYPETRC operand may be specified. Performance degradation is to be expected when the tracing facility is activated.

Debug

If a known problem occurs on demand and can be isolated to a particular application program, the tracing facility can be used in a way similar to the batch environment. Refer to "Which Calls to Trace" in "Tracing in the Batch Environment" in this section for instructions and examples. All of the batch information applies equally to the online environment with the following additional function.

In defining which calls are eligible to be traced in an online situation, an additional parameter list may be specified to limit the trace to only those calls made by a certain PSB. The operand CALLCON should be coded in the DLZTRACE generation with the parameters (PSBNAME,EQ,YOURPSB). In addition, the DBPCBDBD and CALLFUNC parameters may also be specified to limit the calls to be traced. This means, for example, that the following instructions can be given to the tracing facility:

- Trace only those calls made by PSBNAME1.
- Trace the PCB in that PSB identified by DBDNAME2 whose call function is UPD (DLET, ISRT, or RFPL).

Trace Output

In the normal online environment -- that is, when not debugging -- the trace output cannot be on SYSLST. For this environment either the INCORE or CICS/VS option may be used.

If the INCORE option is used, it is important to indicate a table size large enough to hold all of the needed trace entries. The DL/I formatted dump program prints the latest 10 entries when a DL/I or CICS/VS abnormal termination occurs. Because the remainder of the trace table is not transferred to an output storage device, it is necessary to take a storage dump to obtain it.

If the CICS/VS option is chosen, the incore trace table is written to a CICS/VS extrapartition data set each time the incore trace table is filled. This facilitates printing the trace entries at a later time by using the trace print utility program.

If the online system is in a debug environment, the SYSLST output option may be used. This means that only one task making DL/I calls can be executing at any one time. If more than one task is executing, unpredictable results occur because no provision is made to force single-threading of trace calls due to SYSLST I/O.

DEFINING THE EXTRAPARTITION DATASET

It is necessary to define the CICS/VS extrapartition dataset used by the trace facility when OUTPUT=CICS is specified in the DLZTRACE macro. CICS/VS controls the output to this dataset. The dataset can be defined to reside on several different device types. However, only one disk and one tape device are supported by the trace print utility. Since different disk devices have different capacities which affect the maximum size of the incore table, and because it is possible to run out of space rapidly, it is advisable to use tape as the device to hold the records. If a disk device is used and the disk file becomes full, the trace program is disabled and tracing ends. If a tape device is used and the tape becomes full, another volume can be loaded and tracing resumed. Unlabeled tapes must be used, because the trace print utility does not support labeled tapes.

Example

Note: The blocksize specification on the DCT must be at least 4 bytes larger than the trace table. See beginning of DLZTRACE assembly for trace table size.

Figure 6-7 on page 6-30 is an example of defining the destination control table (DCT) for CICS/VS. Also, an entry must be made in the processing program table (PPT) for the trace program on DFHTRNxx, where xx is the TRNSUFX=xx parameter from the DCT. Additional information can be located in CICS/VS System Programmers Reference Manual listed in the Preface.

Opening the CICS/VS Device

It is necessary to open the extrapartition dataset before tracing can begin when OUTPUT=CICS is specified in the DLZTRACE macro. One way to open the CICS/VS device is to issue a CSMT transaction as follows:

```
CSMT OPE,TRANSD,DESTID=DLIT..xx
```

```
DFHDCT  TYPE=INITIAL,TRNSUFX=TR,SUFFIX=XX,

DFHDCT  TYPE=SDSCI,DSCNAME=TAPEIN,RECFORM=VARBLK,SUFFIX=TR,
        TYPEFILE=OUTPUT,BLKSIZE=32763,DEVICE=TAPE,DEVADDR=SYS013

DFHDCT  TYPE=EXTRA,DESTID=DLIT,OPEN=DEFERRED,RESIDNT=NO

DFHDCT  TYPE=FINAL
```

Figure 6-7. Defining the Destination Control Table (DCT) for CICS/VS

where:

xx - is the two character suffix that was specified on the TRNSUFX= parameter of DFHDCT TYPE=INITIAL.

Another way to open the extrapartition dataset is with the DFHOC macro call. The macro can be included with the transaction that starts the trace function.

USER EXIT

USER EXIT DESCRIPTION

The three user exits are:

EXIT 1: The user receives control after the trace module determines whether to activate or deactivate all selected trace points for this call. The determination is based on the specifications in the CALLCON and/or STRTKEY/STOPKEY operands.

Input parameter list:

- TREXPST contains PST address.
- TREXNUM contains 0.
- TREXFLAG has flag TREXFDEC on if the trace module is to trace the call; off, if it is not to trace the call.

Output:

- No change to parameter list.
- R15 = 0 use trace module decision.
- R15 = 4 reverse trace module decision.

EXIT 2: The user receives control after the determination has been made to trace at this point. Any TRCECON parameters have already been checked. This exit is only invoked if the result of the call condition test (including any user exit 1 decision) was to trace. If the trace point has been defined to process more than one option, user exits 2 and 3 are given only one exit each. If the decision at user exit 2 is not to trace, then no tracing occurs at all. If user exit 3 performs tracing, no other tracing is performed at the trace point.

Input parameter list:

- TREXPST contains PST address.
- TREXCALL contains trace point parameter list address.
- TREXNUM contains 4.
- TREXFLAG has flag TREXFDEC on if the trace module decided to trace at this point; off, if it decided not to trace.

Output:

- No change to parameter list.
- R15 = 0 use trace module decision.
- R15 = 4 reverse trace module decision.

EXIT 3: The user exit receives control before any actual tracing is performed for the specified trace point. If the trace point has been defined to process more than one option, user exits 2 and 3 are given only one exit each. If the decision at user exit 2 is not to trace, no tracing occurs at all. If user exit 3 performs tracing, no other tracing is performed at the trace point.

Input parameter list:

- TREXPST contains PST address.
- TREXCALL contains the trace point parameter list address.
- TREXADAT contains the address of the area to be used by the user exit to build the trace entry.
- TREXLDAT contains the length of TREXADAT as specified in the TRACSIZ operand of DLZTRACE.
- TREXNUM contains 8.

Output:

- R15 = 0 trace module should do tracing.
- R15 = 4 exit routine has done tracing.
- R15 = 8 exit routine has done tracing and wants to do more.
- If R15 return code is 4 or 8, then TREXLDAT must contain the length of the trace entry built.

The user exit routine is responsible for using only the size of the user area that is allotted in TREXLDAT at entry to the routine. If the user performed the trace with the SYSLST option in operation, the trace module prints the standard header followed by the unformatted trace data in both character and hexadecimal format.

USER EXIT INTERFACE

The interface to the user exit routine is the same for all three exit points. The input registers are:

- R1 = parameter list address.
- R13 = trace module save area address.
- R14 = trace module return address.
- R15 = user routine entry point address.

On entry, the user routine must first store the trace module registers in its save area, then set up its own register save area in register 13 and chain it off the input save area. Before the user exit routine returns control to the trace module, all registers except register 15 must be restored from the trace module save area.

The output register contents must be:

- R1 = parameter list address.
- R13 = trace module save area address.
- R15 = return code.

The exact meaning of the return code and the parameter list values are described with each of the user exits. Only those values that are mentioned are available or have meaning for the particular exit. The parameter list contains both input information for the user exit routine and can contain some output information.

USER EXIT PARAMETER LIST FORMAT

A macro, DLZTUPRM, is supplied to provide addressability to the user exit parameter list. Figure 6-8 shows the format of the user exit parameter list DSECT, TRUSRPRM, and Figure 6-9 shows the format of the trace point parameter list DSECT, TRCPLIST.

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	4	TREXPST		Current PST address
4	4	4	TREXCALL		Address of trace point parameter list (TRCPLIST) for exits 2 and 3
8	8	4	TREXADAT		Trace data area for user (exit 3)
12	C	2	TREXLDAT		Length of data area (exit 3)
14	E	1	TREXNUM		Present user exit as follows:
				0	=exit 1
				4	=exit 2
				8	=exit 3
15	F	1	TREXFLAG		Flag byte
			TREXFDEC	X'80'	Trace is active

Figure 6-8. TRUSRPRM DSECT - User Exit Parameter List

Offset Dec	Offset (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	4	TRCMODNM		Calling module name
4	4	1	TRCMODID		Trace point id
5	5	1	TRCNOPT		Number of options following
6	6	1	TRCTOPT		First option

Figure 6-9. TRCPLIST DSECT - Trace Point Parameter List

SAMPLE TRACE USER EXIT ROUTINES (DLZTXIT0)

Module DLZTXIT0 provides examples of how to code a user exit for the DL/I Trace Facility. You may either copy the module and modify the copy to suit your needs, or you can modify module DLZTXIT0 itself.

You execute DLZTXIT0 (or your copy of the module) by specifying the module name in the USEREXIT= parameter of the DLZTRACE macro. The parameter can be placed anywhere in the macro statement and is coded in the following way:

```
DLZTRACE ...,
         USEREXIT=DLZTXIT0,
         ...
```

Two routines are coded for each of the three exits; however, only the first example for each exit will execute. You must modify the branch instructions within the sample user exit module if you want to execute the alternate examples.

Note: DLZTXIT0 provides only samples of user exit routines and may not perform the exact operations that you want. You must provide the user exit routines for specific applications.

DLZTXIT0 - Module Description

DLZTXIT0 REGISTER CONVENTIONS:

R2 = PST address
R3 = SCD address
R12 = base register
R13 = register savearea
R14 = return address
R15 = entry point/return code

ENTRY POINT: DLZTXIT0

INPUT:

Trace User Exit Parameter List
Trace Point Parameter List
DL/I Call Parameter List
Various control blocks and data areas

OUTPUT: A user trace entry

EXITS:

Normal - return to Trace Facility
Error - return to Trace Facility

EXTERNAL REFERENCES:

Data areas:
Trace User Exit Parameter List
Trace Point Parameter List
DL/I Call Parameter List

Control Blocks:
PST
SCD
DDIR
DMB

MACROS:

DLZID
DLZIDLI
DLZSCD
DLZSPST
DLZTUPRM
DLZQUATE

INITIALIZATION: This routine is entered for all user exits. It establishes the necessary addressability and saves the registers from the DL/I Trace Facility. It checks the exit number in the parameter list and branches to the proper exit routine.

Note: The branch instructions at label EXITNUM are initially set up for executing the first example of each user exit; that is, EXIT1A, EXIT2A, and EXIT3A. You must change the addresses to EXIT1B, EXIT2B, or EXIT3B to execute the alternate examples of the respective exits.

EXIT ROUTINES:

Exit 1A: Checks the DL/I call parameter list to see if the cur-

rent call is a replace call. If so, tracing is activated; of not, tracing is turned off.

Exit 1B: Activates Trace if the logger has been loaded in a DL/I Version 1.6 or later system. If not, tracing is deactivated.

Exit 2A: Checks if tracing point is at id X'01' in the call analyzer before allowing tracing.

Exit 2B: Activates tracing if the current access method being processed is HIDAM. If not, tracing is deactivated.

Exit 3A: Traces the PPST and indicates to Trace that the user has done the tracing.

Exit 3B: Traces the first 50 bytes of the log workarea if 50 or more records are logged.

RETURN TO TRACE ROUTINE: This routine determines if tracing is active or not and takes the appropriate action. It indicates the action taken in register 15 and returns control to the DL/I Trace Facility. Register 15 contains:

0 = for Exit 1: accept current trace status
= for Exit 2: accept current trace status
= for Exit 3: do normal trace

4 = for Exit 1: reverse current trace status
= for Exit 2: reverse current trace status
= for Exit 3: user did tracing

TRACE PRINT UTILITY

The trace print utility allows the user to print trace entries from tape or disk input files. The DL/I Trace Facility creates these files as a CICS/VS extrapartition dataset when DLZTRACE OUTPUT=CICS is specified. The format of the printed trace entry is the same as that produced when DLZTRACE OUTPUT=SYSLST is specified. See "Trace Entry Format" on page 6-9for a description of the trace output formats.

The trace print utility program runs as a VSE program and does not make any DL/I calls.

The input and output to the program is shown in Figure 6-10 on page 6-35. The input of the trace entries can be from either tape or disk. Labelled tapes should not be used as input. If used, the trace print utility will reject them. The trace print utility decides the input source. The control statement is read from SYSIN. The formatted trace output is directed to SYSLST. Most messages are written to both SYSLST and SYSLOG unless it would be of no value to one or the other.

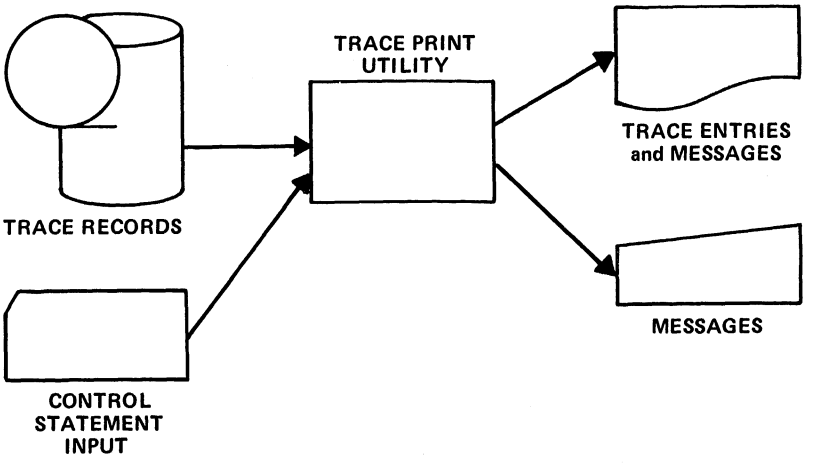


Figure 6-10. Trace Print Utility Program

The input record format is VARBLK. It consists of a two-byte long field, followed by two bytes of zeros, followed by the unformatted data from the incore trace table. The maximum record size is determined by the size of the incore table used by the trace program and the BLKSIZE specified by the DCT. When the DLZTRACE macro is assembled, an MNOTE, which gives the incore table size, is printed at the beginning of the assembled output listing. See the description of the INCORE option in "OUTPUT" for information about calculating the incore table size.

If the DCT blocksize is large enough to hold multiple incore tables the trace program will block them and the trace print utility will unblock them.

The output is to SYSLST with a record length of 121 characters. The first character is used for carriage control.

JOB CONTROL STATEMENT REQUIREMENTS

The trace print utility executes as a VSE application program and requires the job control statements shown in Figure 6-11.

// ASSGN	SYSnnn,cuu	These statements define the input trace entry file(s) to be printed.

// TLBL (tape)
or
// DLBL (dasd)
// EXTENT
	TAPEIN
	
	DISKIN
	extent data
		SYS013 is the default logical unit unless otherwise specified on the TI control statement. The file names are fixed. This name must be specified by the DSCNAME= parameter on the DFHDCT TYPE=SDSCI macro. (See heading "Defining The Extra-partition Dataset" in this chapter).

// EXEC	DLZTPRT0 SIZE=nnnK	Trace print utility.

Figure 6-11. Job Control Statements for Trace Print Utility

The VSE job control statements may contain additional information. More detailed information concerning job control statements is available in VSE/Advanced Functions System Control Statements listed in the Preface.

CONTROL STATEMENT REQUIREMENTS

The Trace Print Utility uses the TI control statement to allow some flexibility in defining the input. This statement allows you to specify the logical unit and buffer size to be used for the trace entry input. Only one TI (trace input) statement is allowed. The parameters are read from left to right. This means that the logical unit will be used if it is valid even though the buffer size on the same TI statement may be invalid. If neither TI nor TO (trace output) control statements are used, a /* statement should be placed in the job stream to indicate their absence. If no TI statement is read, the default values are assumed.

The format of the TI control statement is:

1	11	31
TI	SYSnnn	nnnnn

Column	Description
1	Columns 1 and 2 must contain the characters TI.
11	Columns 11 - 16 define the logical unit used as input. It must be in the format SYSnnn, where nnn is a valid assigned logical unit. If omitted, SYS013 is assumed.
31	Columns 31 - 35 define the buffer size to be used for the file. The buffer size must be large enough to hold the input record but no larger than 32767. If omitted, 32767 is assumed. Note: Two buffers will be acquired by the trace print utility. Sufficient GETVIS space must be available.

To use the selective output of the recorded trace entries, you must include the TO statement(s). A TO statement may contain one or more parameter entries: however, each TO (trace output) statement parameter can be entered only once. If more than one parameter is included in a statement, each parameter must be separated by a comma. The first entry may start in any column after column 3. A blank after any parameter stops the scan on the TO statement. This means that comments may be included in the TO statement following a blank.

The TO statement is optional. If it is omitted, the trace print utility will print all the trace entries. The TO statement may contain one or more options. It is possible to include multiple TO statements. Because options and comments are allowed in columns 4 through 80, no continuation character is used. In fact, a statement ending with a comma followed by a blank is flagged as an error. You may enter each option as a separate TO statement, but you can not continue an option over more than one statement.

The TO statement format is:

```
col 1 and 2    = TO
col 3          = blank
col 4 thru 80 = options
```

The valid TO options are:

```
[TASKID=task
[,MODULE=id]
[,TYPECALL={GU                          }]
           {GN                          }
           {GNP                         }
           {GHU                         }
           {GHN                         }
           {GHNP                        }
           {ISRT                        }
           {DLET                        }
           {REPL                        }
           {GET                         }
           {UPDT                        }
           {MISC                        }
           {(xxx,xxxx,xxxx,xxxx,xxxx)}
[,CALLNUM=([start][,stop])]
```

where:

task — a 1 to 5 digit taskid from the CICS/VS TCA (online only)

id — characters 4 through 7 of the module name

GU — DL/I command/call type

GN — " "

GNP — " "

GHU — " "

GHN — " "

GHNP — " "

ISRT — " "

DLET — " "

REPL — " "

GET — print all get type trace entries

UPDT — print all update type trace entries (i.e.., ISRT, DLET, REPL)

MISC — print trace entries that had an unknown trace command/call type.

xxxx — up to five of the above type command/call entries in a list

start — the number of the command/call at which printing is to begin. If omitted, printing begins at the start of the input. Valid range is 1 - 65535.

Note: These numbers begin at one, with the first DL/I command/call, and are sequentially numbered to the end of the trace.

stop — the number of the command/call at which printing is terminated. If omitted, printing ends at the end-of-file on input. See note under start number.

Any or all of the above options may be specified. However, you should note that an 'AND' relationship exists between the various options, and that an 'OR' relationship exists within the TYPECALL list if it is used. In other words, if three separate options are specified, each one must be satisfied for each entry before that entry will be printed. However, if TYPECALL is

specified using the list format, for example, TYPECALL=(GET,DLET), then only one of the types must be present to satisfy the TYPECALL condition.

Examples

EXAMPLE 1: This example prints the trace entries from a tape file using all default values. The output is directed to SYSLST which is assumed to have been permanently assigned.

```
// JOB    PRTTRC1
// ASSGN  SYS013,X'180'
// TLBL   TAPEIN
// EXEC   DLZTPRT0,SIZE=100K
/*
/&
```

EXAMPLE 2: This example prints the trace entries from a disk file. A TI statement is provided to override the defaults.

```
// JOB    PRTTRC2
// ASSGN  SYS015,X'230'
// DLBL   DISKIN,'TRACE.INPUT'
// EXTENT SYS015,DIIVOL
// EXEC   DLZTPRT0,SIZE=100K
TI        SYS015              4095
/*
/&
```

EXAMPLE 3: This example shows the use of the trace print utility to print REPL, DLET, ISRT, and invalid type command/call trace records that were recorded while task number 10 was in module DLZDHDS0. The trace records are in 4095-byte blocks on tape mounted on device 181 which is assigned to logical unit SYS020.

```
// JOB   PRTTRC3
// ASSGN SYS020,181
// TLBL  TAPEIN
// EXEC  DLZTPRT0,SIZE=AUTO
TI        SYS002                4095
TO TASKID=10,MODULE=DHDS   SELECT DLZDHDS0 ENTRIES FROM TASK 10
TO TYPECALL=(UPDT,MISC)
/*
/&
```

Additional examples of the TO options are:

`TO TYPECALL=DLET`	to print only delete commands/calls
`TO TYPECALL=(GET,ISRT)`	to print all get and insert commands/calls
`TO CALLNUM=(50,200)`	to print only commands/calls from number 50 to number 200
`TO CALLNUM=(100)`	to print all commands/calls after number 100
`TO CALLNUM=(,75)`	to print all commands/calls up to the number 75

TRACE INVOCATION MACRO FOR ACTION MODULES

The macro DLZTRCAL is inserted into the DL/I action modules at defined points. This macro expansion first checks if tracing is enabled. This can be determined by checking a byte in the SCD which indicates if the trace module is loaded and if this trace point is activated. If tracing is not enabled, normal processing continues. If tracing is enabled, a parameter list is passed to the tracing module with the following information:

Offset Dec(Hex)		Length	Field/Flag Name	Flag Code(Hex)	Meaning
0	0	4	TRCMODNM		Calling module name
4	4	1	TRCMODID		Trace point ID
5	5	1	TRCNOPT		Number of options following
6	6	1	TRCTOPT		Trace option for this point (more than one is possible)

The trace module then makes the final check as to whether or not tracing should be performed at this point. This includes checking the TRCECON operand values and abiding by the results of any user exit routine. After processing, control is returned to the calling module.

A special expansion of DLZTRCAL is used to request the tracing module to evaluate any call-tracing conditions as specified in CALLCON, STRTKEY, or STOPKEY. This is not a trace point and no tracing can occur. This macro is placed before the first trace point in the call analyzer.

DLZTRCAL is also used to request the tracing function to purge any buffers and to free acquired storage. This is done by the online program request handler, when a TSTP call is received, and by system termination.

The operands of DLZTRCAL are:

- TYPE=TRACE - is the default to indicate that tracing occurs at a predefined trace point.
- CKCALL - is a special invocation to indicate the trace module should check the call conditions to determine if the call should be traced.
- START - is a special invocation of the trace module that causes it to initialize itself.
- STOP - is a special invocation to stop tracing.
- CALLER=cccc - is the identification of the calling module. This should be characters 4-7 of the official name and must be specified if TYPE=TRACE.
- ID=nn - is a decimal number from 1 to 255 and uniquely identifies the trace point. It is required if TYPE=TRACE.
- OPTION=USERCAL1, USERCAL2, MODTRACE, RETRIEVE, CURRPOS, VSAMINTF, BHINTF1, BHINTF2, INDEXTRC, ONLINEBH, or PITRACE - is the trace option serviced at this trace point. One is required if TYPE=TRACE, but more than one may be specified.
- PSTREG=REG - is the number or symbolic name of the register that contains the PST address. If the entry is omitted, 1 is assumed. If TYPE=START or STOP, the register contains the SCD address.

Before DLZTRCAL is executed, addressability is required to the SCD.

- If TYPE=TRACE, no registers are changed.
- If TYPE=CKCALL or STOP, the contents of register 15 are destroyed by this macro.
- If TYPE=START, register 15 contains a return code:

0 = no error, tracing is initialized.
4 = GETVIS failure, SIZE parameter omitted from EXEC statement.
8 = GETVIS failure, program is executing in real mode.
12 = GETVIS failure, no storage available.

- If TYPE=START, the entry point address of the trace module must be in register 15 before the macro is executed.

APPENDIX A. MPS DIAGNOSTIC AIDS

XECBS USED IN MPS

MPS requires a number of XECBs equal to three times the number of active MPS batch partitions, plus three additional XECBs. For example, if a maximum of 6 MPS batch partitions could be currently active, MPS would require (3 X 6) + 3, or 21 XECBs. MPS limits the number of concurrent batch partitions (including pseudo-partitions) to 12.

The naming conventions for partition-related XECBs are:

DLZXCBny

where:

n is an alphabetic character identifying the MPCPT entry associated with the batch partition.

y ranges from 1-3 and specifies the XECB type as follows:

1 - is the batch partition's XECB XPOSTed by BPC after completion of initialization and after completion of each call; it is also posted by the MPC if the BPC ATTACH fails, or on abnormal termination of the BPC.

2 - is the BPC's XECB XPOSTed by the batch program request handler each time a DL/I call is to be processed and by batch termination at EOJ time.

3 - is the BPC's ABEND XECB XPOSTed by the batch abend processor, DLZMABND, encountering an abend situation.

In addition to these partition related XECBs, there are three general XECBs used by MPC:

- DLZXCB00, the MPS stop XECB, posted by the stop transaction program, DLZMSTP0.
- DLZXCB01, the partition stop XECB, posted by the BPC on termination, or by task termination of a BPC.
- DLZXCB02, the batch partition start XECB, XPOSTed by the batch initialization routine.

Figure A-1 on page A-2 shows a module/usage cross reference for all XECBs.

XECB MACRO RETURN CODES

The following section discusses the return codes for the various XECB macros, as they appear in messages DLZ082I and DLZ084I, giving some suggestions on what could cause these return codes.

XECBTAB TYPE=DEFINE

X'04' The XECB has already been defined.

Suggestions: Make sure no other programs in the system use the DL/I reserved XECB names.

X'08' No room in the supervisor table.

DIAGNOSTIC GUIDE
MODULE CROSS REFERENCE FOR XECBs

XECB Name	Defining Routine *Label	Waiting Routine *Label	Type of Wait	Posting Routine *Label	Type of Post	Deleting Routine *Label	Reason for Posting the XECB
DLZXCB00	DLZMPC00 *XECBDFN0	DLZMPC00 *MPCWAIT	CICS/VS WAITM	DLZMSTP0 *DLZMSTP0	OI	DLZMPC00 *XECBDL00 *XECBDLT0 *MPCABEXT *MPCSYSTM	Notify MPC that MPS is to stop.
DLZXCB01	DLZMPC00 *XECBDFN1	DLZMPC00 *MPCWAIT	CICS/VS WAITM	DLZBPC00 *BPCEXIT2 DLZMPC00 *MPCECBOT *MPCBPCNT *MPCBPCA1	 OI OI OI OI	DLZMPC00 *XECBDL1 *XECBDLT1 *MPCABEXT *MPCSYSTM	Notify MPC that BPC (batch partition) has terminated.
DLZXCB02	DLZMPC00 *XECBDFN2	DLZMPC00 *MPCWAIT	CICS/VS WAITM	DLZMINIT *XPOST1	 XPOST	DLZMPC00 *XECBSDEL *MPCABEXT *MPCSYSTM	Notify MPC that batch partition (BPC) is to be started.
DLZXCBn1	DLZMINIT *XECBDEF1	DLZMINIT *XWAIT1 DLZMPRH *XWAIT0 DLZMTERM *XWAIT2	XWAIT XWAIT XWAIT	DLZMPC00 *XECBABN *MPCBPCF2 *MPCBPCNT *XECBFN1 *MPCSYSC1 DLZBPC00 *BPCSCHOK *BPCPRHR *DELXECB *BPCDFERR	 XPOST XPOST XPOST XPOST XPOST XPOST XPOST XPOST XPOST	DLZMINIT *XECBDEL1	1) Notify batch that DEFINE for DLZXCBx3 XECB failed. 2) Notify batch that BPC ATTACH failed. 3) Notify batch that EOJ has been recognized. 4) Notify batch that BPC has abended. 5) Notify batch that CICS/VS has terminated. 6) Notify batch that scheduling call complete. 7) Notify batch that call processing is complete. 8) Notify batch that DLZXCBn2 DEFINE failed.
DLZXCBn2	DLZBPC00 *XECBDFN	DLZBPC00 *BPCWAIT	CICS/VS WAITM	DLZMPRH *XPOST0 DLZMTERM *XPOST2 DLZMPC00 *CHKXECB	 XPOST XPOST OI	DLZBPC00 *XECBDEL *DELXECB DLZMPC00 *XECBDN2 *MPCSYSD2	1) Notify BPC that call is to be processed. 2) Notify BPC that EOJ is to be processed. 3) Notify BPC that MPC is abending.
DLZXCBn3	DLZMPC00 *XECBABN	DLZMPC00 *MPCWAIT DLZBPC00 *BPCWAIT	CICS/VS WAITM CICS/VS WAITM	DLZMPC00 *XECBTCH *XECBATCH DLZMABND *XPOST3	 OI OI XPOST	DLZMPC00 *XECBDELA *XECBDEL3 *DLT3XECB *MPCSYSD3 *XECBDLN3	1) Notify BPC that batch has disappeared. 2) Notify MPC that: • BPC attach failure recognized. • BPC abnormal termination recognized. Notify BPC that batch abend in progress.

Figure A-1. Module/Usage Cross Reference for XECBs

XECBTAB TYPE=CHECK

X'04' XECB not found in the table.

Suggestion: There are two possible reasons for this return code:

1. The XECB has not been defined. This situation is expected to occur in batch initialization, in which case it means that MPS has not been activated in the online partition. Message DLZ089I results from this case.

2. The XECB was there, but has been deleted in the meantime. If the failing module is DLZMPRH, probably the online partition was terminated, thus causing the XECBs to be deleted.

If the failing module is DLZBPC00, the batch partition terminated after having posted the MPC.

Check for appropriate messages from the online partition.

XECBTAB TYPE=DELETE

X'04' XECB was not in the table.

X'08' Issuing program is not the owner of this XECB.

Suggestion: Make sure no other programs in the system use DL/I reserved XECB names.

XWAIT

X'04' XECB not found in the table.

Under VSE, this return code is also given to the waiting task when the XECB was deleted. This can only occur when the waiting task is not the owner of the XECB. In MPS, the owner of an XECB is always the waiting task; therefore, this return code is logically impossible in MPS. Its occurrence indicates a problem in the VSE/Advanced Functions supervisor.

X'08' Communication with the other task using this XECB has been broken. Some task issued an XECBTAB TYPE=DELETALL.

X'0D' Access code violation: task not allowed to issue XWAIT. This code is not expected to occur in MPS.

X'0E' See X'0D'.

XPOST

X'04' XECB was not found in the table.

Suggestion: This return code indicates that the related partition owning the XECB was terminated.

Check for appropriate messages from the related partition.

X'0D' Access code violation: task not allowed to issue XPOST.

Suggestion: Check for any program outside DL/I that may have used DL/I reserved names.

X'0E' See X'0D'.

HOW TO LOCATE XECB TABLE ENTRIES

There is no direct pointer to locate the supervisor's XECB table in a dump. However, there are some pointers saved in DL/I modules that can be used to find the table. These pointers are:

Module	Field	Contains table entry address for XECB
DLZMPI00	N02ENTRY	DLZXCB02
	N1ENTRY	DLZXCBn1
	N2ENTRY	DLZXCBn2
	N3ENTRY	DLZXCBn3
DLZBPC00	TWAN1PTR	DLZXCBn1

If none of these pointers is available or valid, obtain a SYSGEN listing of the supervisor and locate label XECBTAB which is the beginning of the table. The halfword labeled XECBHDR contains the number of entries generated.

MPS DATA AREAS

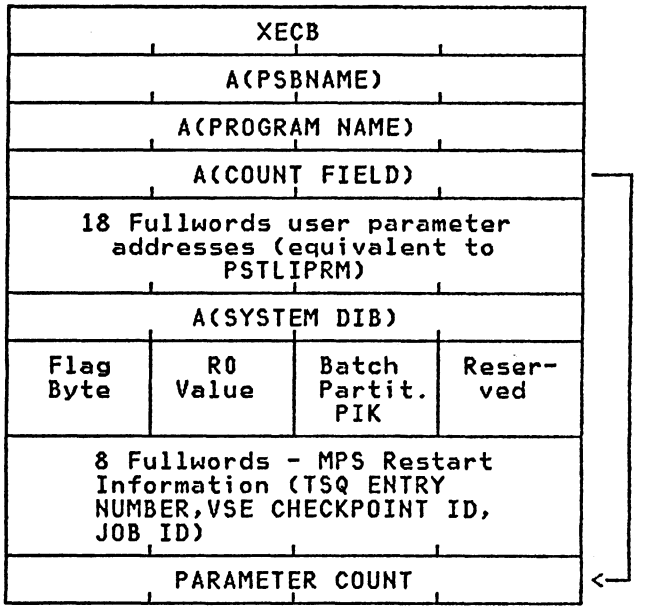
The following control blocks are used for communication between batch and online partitions:

BATCH COMMUNICATIONS AREA

Module DLZMPI00

Label DLZXCBN1

The batch communications area (see Figure A-2) is accessed by DLZBPC00. The address is obtained via XECBTAB TYPE=CHECK for DLZXCBn1.



Flag byte: X'01' EOJ
X'02' PL/I
X'04' MPS restart active
X'08' Checkpoint/restart call
X'10' Checkpoint not verified - restart processing continued

JOB ID = Jobname - 8 bytes
Partition ID - 2 bytes
Start Date - 8 bytes
End date - 8 bytes

Figure A-2. Batch Communications Area

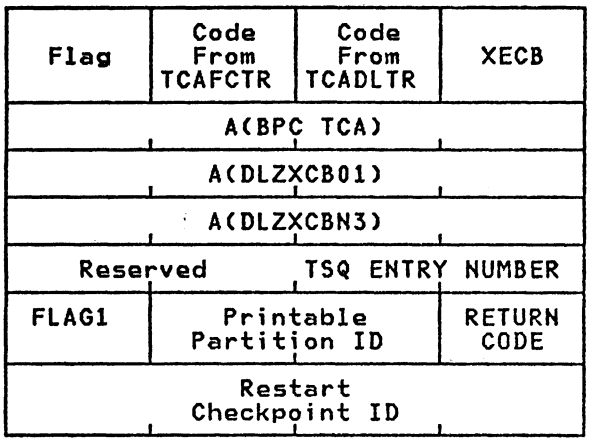
MASTER PARTITION CONTROLLER PARTITION TABLE

The master partition controller partition table (see Figure A-3 on page A-5) is the major MPS control block.

Module DLZMPC00

Label MPCPARTB (located in Transaction Work Area).

Pointed to by SCDMPCPT in the SCD. There is one 28 byte entry for each batch partition. One entry is mapped by DSECT DLZMPCPT.



Flag: X'80' Partition active
X'40' Error on call or attach
X'20' Stop transaction request issued
X'10' Partition to stop
X'08' MPC to wait for ABEND XECB

Flag1: X'80' Cancel BPC at stop transaction if batch not active
X'40' Checkpoint verification not possible during restart
X'20' Wrong checkpoint or restart
X'10' At least one combined checkpoint issued
X'08' Job ABEND - Reinitialize TSQ if combined checkpoint not taken
X'04' MPS restart active

Return Code: 0 MPS restart ok
1 CICS/VS Journaling not active
2 Temporary storage not available or error

Figure A-3. Master Partition Controller Partition Table

BATCH WAIT STATES

There are certain conditions under which MPS is no longer able to communicate with a waiting batch partition, thus, leaving it in a wait state.

- Scheduling Conflict or MAXTASK

  A BPC may have been suspended due to scheduling conflict, or the MAXTASK, or CMAXTASK condition being detected. This results in the batch partition also waiting. This is a normal condition. It can be avoided by not running conflicting batch and/or online tasks concurrently.

- Uncontrolled BPC Abend

  If the BPC abends at a point in time where it has not successfully scheduled a PSB (e.g. after unsuccessful initial scheduling), CICS/VS will not take the DL/I task termination exit. Thus, the batch partition XECB will not be posted and, consequently, it will never leave its wait state. It will have to be cancelled in this case. Check for any previous BPC abends or use CSMT to detect missing BPCs.

- BPC Attach failure

  If the ATTACH for the BPC failed for any reason other than insufficient storage, the MPC will not be aware of this situation, thus leaving the batch partition in its wait state.

If CSMT TASKL shows that a BPC which is expected to be active is not, check for any CICS/VS initialization messages disabling transaction CSDC and for message DFH2009 at the time the MPC tried to attach the BPC. The reason for the ATTACH failure could be one of the following:

- DLZBPC00 was not defined in the PPT.
- CSDC was not defined in the PCT.
- DLZBPC00 was not in the core image library.

TEMPORARY STORAGE ERRORS

A temporary storage queue (TSQ) entry table exists for each MPS batch job using MPS Restart. The format for the table is shown in Figure A-4.

Module DLZTSQ00

FLAG BYTE	"A"	"Ø"	Reserved
Reserved			
Checkpoint ID			
JOB ID			

JOB ID = Jobname - 8 bytes
Partition ID - 2 bytes
Start Date - 8 bytes
End Date - 8 bytes

Temporary storage errors that may occur when running MPS batch applications using MPS Restart include the following:

- TSQ access error

  If message DLZ123I is issued, an unexpected response code was returned while using CICS/VS Temporary Storage Control to access the TSQ. The name of the routine in which the error occurred is provided in the message.

  DLZBPC00 - The message text identifies the partition identifier of the batch partition served by the BPC, the action in which the TSQ was involved (GETQ, PUTQ, or PURGE), and the response code.

  DLZMPC00 - If a partition identifier is specified, then the error occurred in the BPC abnormal termination routine located in DLZMPC00. If no partition identifier is specified, the error occurred during start or stop partition processing in MPC. In either case, the TSQ activity (GETQ, PUTQ, or PURGE) and a response code are also identified.

  The meaning of the response code can be obtained from the CICS/VS Application Programmer's Reference Manual (Macro Level).

- TSQ does not exist or is not available

  If message DLZ125I is issued, the MPS Restart facility was unable to verify that the checkpoint ID used on the VSE RSTRT job control statement corresponded to the last successful combined checkpoint for the job being restarted. This is caused when the temporary storage queue is not found or has been modified so that the checkpoint ID is no longer available.

This situation may result from purging the TSQ using the CSDP transaction.

If a temporary storage error occurs, or if there is some other problem with the TSQ, the CICS/VS Command Interpreter can be used to directly access the TSQ.

DIAGNOSTIC AIDS FOR MPS MESSAGES

This section contains additional diagnostic aids for MPS messages that may be helpful in locating your problem.

DLZ082I - DLZBPC00, DLZMINIT, DLZMPC00, DLZMPRH

If the return code is X'00' with a CHECK macro, the check itself was successful but the XECB address returned was different from the one obtained from a previous check. This indicates that the XECB has been deleted and redefined. This condition is detected in DLZMPI00 after labels CHECKSUB, XWAIT1, NOTPLI20, and XECBCHK2.

Ensure that after an immediate CICS/VS shutdown all MPS batch partitions also terminate before restarting CICS/VS with MPS.

DLZ083I - DLZMSTR0

The error is detected at the ATTACH macro in DLZMSTR0.

The message is issued at label NOATTACH.

The macro uses the COND=YES option (an internal interface) to receive a return code instead of being abended. A return code of X'31' means there is insufficient storage to service the ATTACH.

Ensure that sufficient storage is available in the CICS/VS partition.

There are several other error conditions possible at the ATTACH which give a CICS/VS message without notifying the start program.

DLZ084I - DLZBPC00, DLZMINIT, DLZMPC00, DLZMPRH

If the macro is XPOST and the return code is X'04', one of a pair of partitions terminated without being able to communicate to the other partition. Check for a corresponding messages from the associated partition.

If the macro is XWAIT, the indication is a VSE system problem or interference by other programs using DL/I reserved XECB names.

See "MPS Control Blocks and XECB Usage" for additional information.

DLZ085I - DLZMINIT

The error is first detected in DLZMPC00 at label MPCBPCFL.

Byte MPCFLAG in the partition table entry is set to X'40' to indicate an ATTACH failure. The batch partition's XECB is posted.

Batch initialization, after being posted, detects the error after label XECBCHK3 and writes the message at label ERR1.

Two reasons for a bad CICS/VS ATTACH return code are insufficient storage and the maximum tasks within class limit has been reached.

There are several other possible errors at the ATTACH which give a CICS/VS message rather than a message to the MPC. In these cases, the MPC does not know about the failure, and the batch partition remains in its wait state. See "Batch Wait States" for additional information.

DLZ090I - DLZMTERM

At normal termination time, DLZMTERM sets an EOJ indicator in the batch communications area and XPOSTs the BPC. In order to keep this indicator in storage until the BPC has checked it, DLZMTERM issues another XWAIT to be posted from the BPC's termination routine.

If the error occurs on XPOST, the CICS/VS system or the BPC terminated before batch starts termination processing. Check for appropriate online messages.

If the error occurs on XWAIT, the indication is a VSE system problem or interference by other programs using DL/I reserved XECB names.

See "MPS Control Blocks and XECB Usage" for additional information.

DLZ095I - DLZMINIT

After the scheduling call, DLZBPC00 moves the TCA return code to the partition table entry and sets an error flag. After being posted, batch initialization checks the error code at label XWAIT1 and issues the message at label ERR1.

If the return code is X'0C02', an intent conflict with another batch partition was detected. The user should not run conflicting batch and/or online tasks concurrently.

No return code is given if the batch partition is in conflict with a running online task or a batch task issued its own PCB call. In this case, the BPC is put into a WAIT state since the task is expected to release the PSB shortly.

See "Exit Conditions" for a complete list of the scheduling errors.

DLZ097I - DLZMSTR0

The message is issued at label NODLI. DLZMSTR0 uses the DL/I interface list to determine if DL/I is active.

If the pointer to the interface list (CSADLI in the CSA optional features list) is zero, DL/I is not in the system.

When online initialization decides to use the dummy nucleus, it sets up the DL/I interface list so that all fields point to the dummy nucleus. Therefore, if task termination and system termination addresses are equal, DLZMSTR0 assumes a dummy nucleus is loaded. Because any MPS batch partition with a bad scheduling return code, as set by the dummy nucleus, would abend immediately, the MPS start request is rejected.

DLZ100I - DLZMPRH

The error is detected after label XWAIT0 because bit MPCERR in the MPCFLAG in the partition table entry is on. The error flag is set by DLZODPQ1 (task termination) after label BPCLNUP on abnormal termination of the BPC.

The message is issued at label PRHERR1.

A program check probably occurred in either the BPC or a DL/I action module. See "Online ASRA ABEND Debugging" for additional information.

Note: A dump is produced when you receive a code of 02 with this message. No dump is produced when you receive a code of 01.

DLZ102I - DLZMPRH

The TCA code is set by the online program request handler and is detected after label XWAIT0 in DLZMPRH.

The message is issued at label PRHERR2.

The BPC terminates with message DLZ103I.

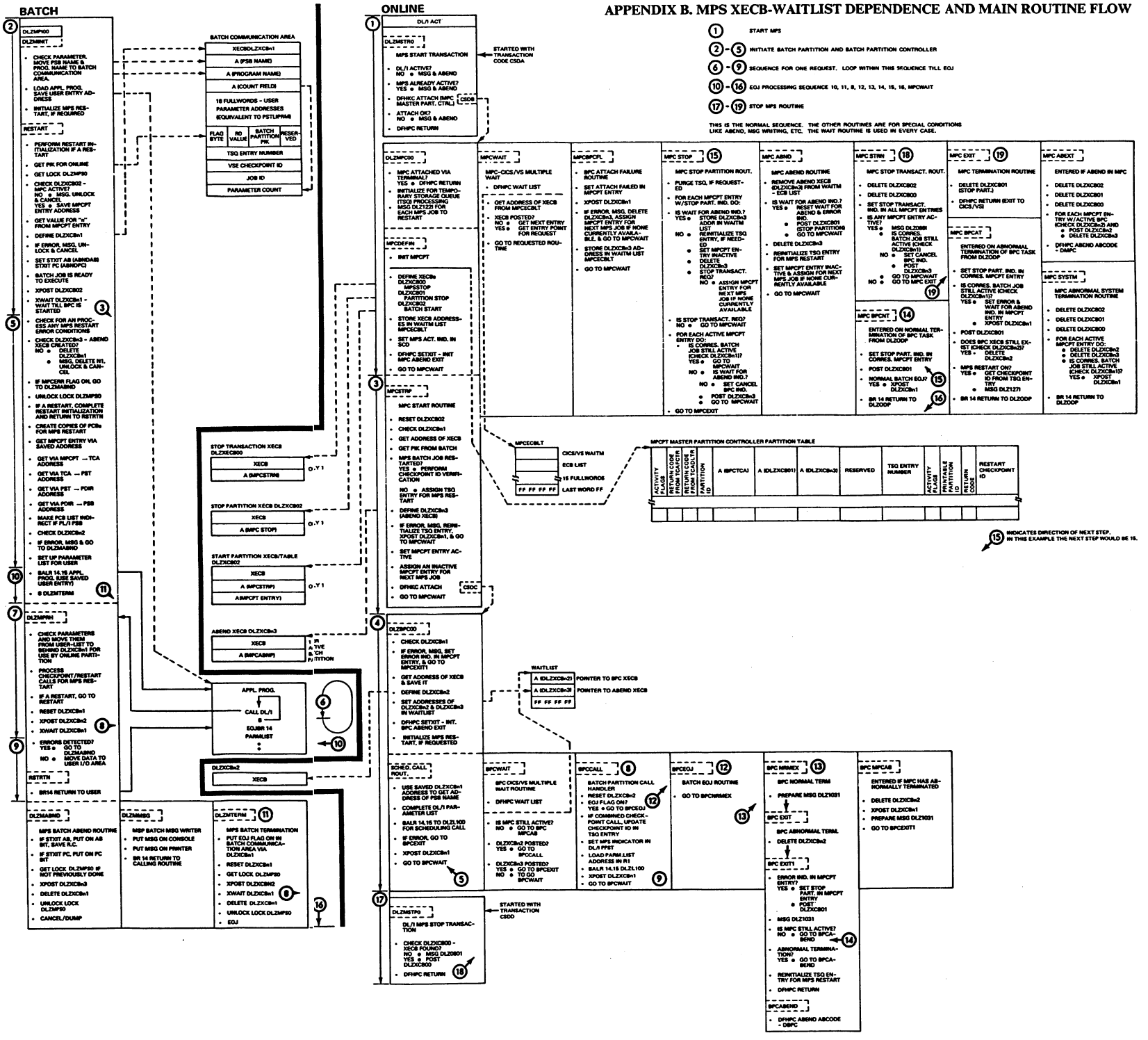
Locate the user parameter list using the batch communications area to determine the type of call (refer to "MPS Control Blocks and XECB Usage" for additional information).

If the call was PCB, see "Exit Conditions" for a complete list of the scheduling errors.

If the call was TERM, the user probably issued two TERM calls in a row.

If the call was a data base or system call, the user probably failed to issue a PCB call after the previous TERM call.

APPENDIX B. MPS XECB-WAITLIST DEPENDENCE AND MAIN ROUTINE FLOW



APPENDIX C. SEGMENT INTENT LIST

CICS/VS application programs that issue DL/I calls are scheduled for access to DL/I data bases by the DL/I online processor. Scheduling is by PSB intent -- that is, according to the manner in which a user wishes to access segments of the physical data structures (DBDs) contained within his logical data structure (PSB). When the user defines his logical data structure (PSB generation), he specifies the processing options to be used on the sensitive segments for each data base PCB contained within his PSB. He does this by specifying the PROCOPT= parameter in each PCB statement and optionally in the SENSEG statements for each PCB.

When the DL/I system is initialized, a PSB segment intent list entry is created for each data base referenced by the PSB. When the application program requests to be scheduled for DL/I access, scheduling is performed only if the PSB intent scheduling rules (see Figure C-1 on page C-2) are satisfied for every data base accessed by the application program (PSB). If any rules are violated, the application program is not scheduled. The following conclusions can be drawn from the figure:

- An application program can guarantee private access to a data base by specifying exclusive control.
- If an application program accesses only HDAM or HIDAM data bases and sensitivity is read only, the application program is always scheduled (unless another application program has previously specified exclusive control).

PSB SEGMENT INTENT		RULES			
PSB Executing	PSB to be scheduled	HDAM	HIDAM	HISAM	Root Only HISAM
Root I	G	Yes	Yes	No	Yes
Root D	G	Yes	Yes	Yes	Yes
Root R	G	Yes	Yes	Yes	Yes
Dependent I,R, or D	G	Yes	Yes	Yes	---
I,R, or D	Different segment type: I, R, or D	Yes	Yes	No	---
I,R, or D	Same segment type: I, R, or D	No	No	No	No
E	G,I,R,D, or E	No	No	No	No
G,I,R,D	E	No	No	No	No
G	Root I	Yes	Yes	No	No
G	Root D	Yes	Yes	Yes	No

Legend

G - Read only or get sensitivity
I - Insert sensitivity
R - Replace sensitivity
D - Delete sensitivity
E - Exclusive control

Figure C-1. PSB Intent Scheduling Rules

APPENDIX D. ONLINE TEST PROGRAM - DLZMDLI0

CAUTION
This program is included in the released system because of its usefulness as a debugging aid; however, it is neither supported nor maintained for customer use. It may be modified or expanded by IBM at any time without prior notice.

The primary purpose of this program is to provide an online entry-informational facility. DLZMDLI will only support system calls, and a few other special calls developed for testing purposes. Data base processing calls are not supported.

CICS/VS basic mapping support (BMS) is used to write to and read from a 3270 display screen (required for DLZMDLI). Entries are required in the CICS/VS Processing Program Table (PPT) for the program DLZMDLI0 and its map set, DLZMDLM.

```
DFHPPT TYPE=ENTRY,
       PROGRAM=DLZMDLI0,
       RELOAD=YES

DFHPPT TYPE=ENTRY,PROGRAM=DLZMDLM
```

An entry in the CICS/VS program control table (PCT) is also required for the DLZMDLI0 program.

```
DFHPCT TYPE=ENTRY,TRANSID=MDLI,
       PROGRAM=DLZMDLI0,
       TWASIZE=1200,DTB=YES
```

After entering the transaction id, MDLI, Screen 1 (Figure D-1) is displayed.

With this screen the user is given the opportunity to schedule a PSB of his choice. If you do not wish to schedule a PSB, press EOB and Screen 2 (Figure D-2 on page D-2) is displayed.

```
                    DL/I ON-LINE TEST PROGRAM
PRESS ENTER TO DISPLAY DL/I SYSTEM CALLS SUPPORTED BY THIS PROGRAM
ENTER PSBNAME BELOW IF YOU WANT TO SCHEDULE A CERTAIN PSB
_

MDLI WILL ONLY SUPPORT SYSTEM CALLS  NO DATA BASE PROCESSING CALLS
```

Figure D-1. Screen 1

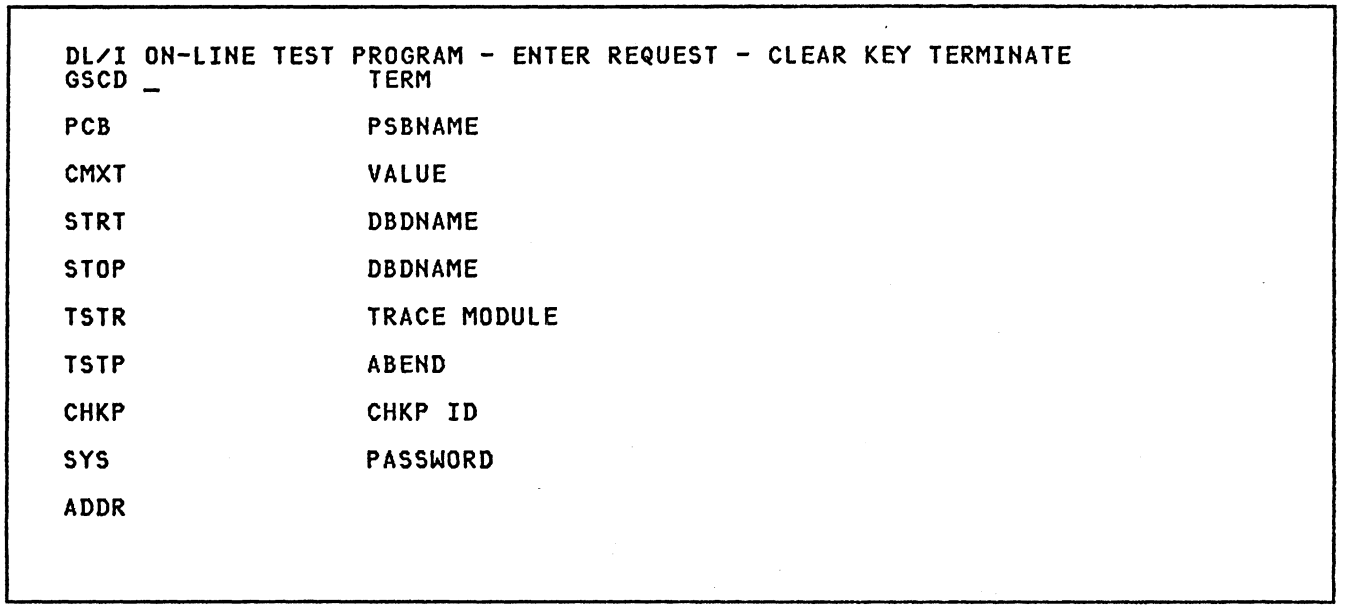


Figure D-2. Screen 2

Some of the calls listed on the Screen 2 must be entered in pairs. That is, any character in the column 1 position (e.g. an 'x' after PCB), followed in column 2 by the specific data requested, (namely a PSBNAME). GSCD, TERM, TSTP, and ADDR have no second operand. An 'x', or some other character, is all that is required. Detailed instructions follow. See SYS call explanation before executing other calls. All results will be displayed on Screen 3 (Figure D-3 on page D-3).

Screen 2 accepts system and other special calls:

GSCD retrieves the system contents directory address. A PSB must be scheduled to receive a valid address.
TERM terminates the scheduled PSB. Only one PSB may be scheduled at a given time.
PCB is used to schedule a PSB if one was not scheduled on Screen 1 or if a TERM call was executed and another PSB is to be scheduled. PSBNAME must contain the name of the PSB to be scheduled.
CMXT is the first of the system calls. It adjusts the current maximum task value. The new VALUE along with the old CMXT value will be displayed on Screen 3. See SYS call before executing.
STRT starts and opens a DBD. The DBDNAME must be supplied in column 2. See SYS call before executing.
STOP stops and closes a DBD. The DBDNAME must be supplied in column 2. See SYS call before executing.
TSTR starts a trace program. TRACE MODULE must contain the name of the trace program. See SYS call before executing.
TSTP stops a trace program. See SYS call before executing.
ABEND is not supported.
CHKP issues a DL/I checkpoint call with the associated CHKPID. A PSB must be scheduled before issuing this call.
SYS call identifies the password associated with the Application Control Table (ACT). If no PASSWORD was specified the default password, DLZPASS1, is used. SYS call must be executed before system calls - CMXT, STRT, STOP, TSTR, and TSTP - will execute.
ADDR returns a list of module names and the addresses of the modules found in the SCD.

As a result of choosing a function from screen 2, screen 3 is displayed.

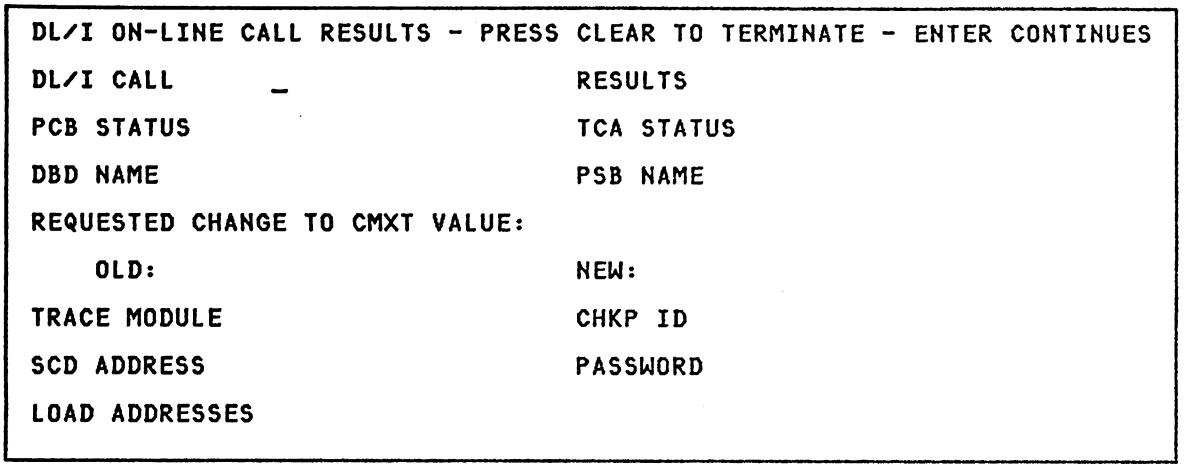


Figure D-3. Screen 3

A more detailed description of the DL/I system calls can be found in the DL/I DOS/VS Data Base Administration, under "Controlling the DL/I Online System Environment" in Chapter 10.

Use of the system calls should be kept to a minimum, and, if possible, be restricted to one program since these calls are specific to DL/I DOS/VS and are not supported by IMS/VS. Unrestricted use could, therefore, make future migration to CICS/OS/VS with DL/I interface more difficult.

TCA Status Codes:

Code	Meaning
08 01	PSB Not in Directory (PDIR)
08 03	Task Already Scheduled
08 05	PSB Initialization Error
08 06	PSB Not in Programs ACT Entry
08 07	TERM Call When Not Scheduled
08 08	Data Base Call When Not Scheduled
08 09	Illegal MPS Scheduling Call
08 FF	DL/I Not Active
0C 01	Data Base Not Opened or Stopped
0C 02	Online Scheduling Conflict With MPS Task
01 00	Invalid Request (State Already Exists)
02 00	Load of Trace Module Failed (TSTR) DMB Failed to Initialize (STRT/STOP)
03 00	TESTCB Failed (STRT/STOP)
04 00	Trace Initialization Failed
08 00	System Interface in Use ('SYSTEMDL' SCHD) Invalid CMAX Value (CMXT) DMB Not in Directory (STRT/STOP)

APPENDIX E. REPORTING DL/I PROBLEMS

GENERAL DOCUMENTATION REQUIREMENTS

The following items of information should be known or readily available when reporting any DL/I problems to the IBM Support Center:

1. Component identification number and release level. For example, the component identification number for DL/I DOS/VS Version 1, Release 7 is:

   5746-XX-100

   and the release level is 170.

2. Problem description, that is, type of failure, and failing function and module, if known.

3. Problem history, that is, modification level change, PTF application, APAR application, data base crash, etc.

4. Listing of program involved.

5. Pertinent dump and console log.

Should it become necessary to do so, the following documentation requirements must be fulfilled when submitting an APAR (authorized program analysis report).

APAR REQUIREMENTS

Materials which are necessary to recreate a problem must be submitted in machine-readable format. Change teams may request other information in addition to the data submitted with the APAR (authorized program analysis report). The actual data submitted should include the indicated items shown in Figure E-1 on page E-2 , but should not be limited to only those items.

Data Submitted	ABEND/Program Check	Access Problem	Data Base Related Error	Loops	Pseudo-ABEND	Utility/Intermediate Data Base Problems	Waits
ACT generation (1)	O				O		O
CICS/VS trace table (1)	E			E	O		E
Console sheet	E	E	E	E	E	E	E
DBDs	E	E	E	O	E	E	
DLZTRACE output		E	E	O	E		
DL/I ABEND dump	E				E		
DL/I Log Print	O	O	E		O	E	O
DL/I test program control cards/output (2)		E	E	E	E		
Explain any unusual events	E	E	E	E	E	E	E
Full partition dump at failing point	E	E	E	E		E	E
JCL listing of failing job		E		E	E	E	E
Listing of affected modules, if modified	E	E	E	E	E	E	E
Listing of application program	O	O	E				O
Log print before backout (3)						E	
Log print after backout (3)						E	
Module name in which problem occurred	E			E		E	
Printout of affected records	O	E	E	O	E	E	
PSBs	E	E	E	O	E		E
SDAID output	O			E			
Transaction dump (1)	E		E		E		
User modifications, PTFs installed	E	E	E	E	E	E	E
VSAM DEFINE	E	E	E		O	O	
VSAM LISTCAT	E	O	E		O	O	

Notes:

(1) If online problem

(2) Include the DL/I test program only if it shows the error conditions

(3) If backout problem

O = optional

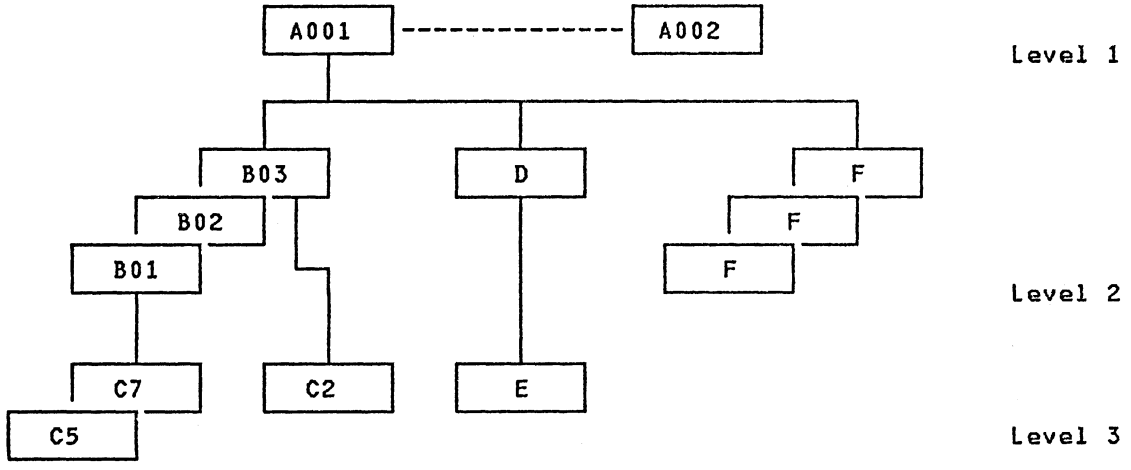
E = essential

Figure E-1. APAR Required Information

GLOSSARY

To improve the readability and your understanding of this and other DL/I DOS/VS publications, the significant and important abbreviations, acronyms, and terms relevant to DL/I DOS/VS are defined in this glossary.

Some of the definitions refer to the representative DL/I hierarchical structure shown in Figure E-2.



- Each block represents a segment.
- The segment names are A through F.
- The numbers represent the sequence fields (keys). If no number is present, the segment has no key.
- The lines connecting the segment blocks show the hierarchical paths.

Figure E-2. Representative DL/I Hierarchical Structure

A

ACB. (1) Application control blocks (DL/I). (2) Access method control block (VSAM).

ACBGEN. Application control block generation.

access method control block (ACB). A control block that links a program to a VSAM data set.

access method services. A multifunction utility program that defines VSAM data sets (or files) and allocates space for them, and lists data set records and catalog entries.

ACT. Application control table.

addressed direct access. In systems with VSAM, the retrieval or storage of a data record identified by its relative byte address, independent of the record's location relative to the previously retrieved or stored record. (See also keyed direct access, addressed sequential access, keyed sequential access, and relative byte address).

addressed sequential access. The retrieval or storage of a VSAM data record relative to the previously retrieved or stored record. (See also keyed sequential access, addressed direct access, and keyed direct access).

aggregate. See data aggregate.

anchor point (AP). See root anchor point.

application control blocks. The control blocks created from the output of DBDGEN and PSBGEN, e.g., a DMB of an internal PSB created by the ACB utility program.

application control block generation (ACBGEN). The process by which application control blocks are created.

application control table (ACT). A DL/I online table describing those CICS/VS application programs that utilize DL/I.

argument. (1) (ISO)[1] An independent variable. (2) (ISO)[1] Any value of an independent variable. (3) Information, such as , names, constants, or variable values included within the parentheses in a DL/I command.

attribute. A property of an entity expressing a value. Synonymous with field.

B

backout. The process of removing all the data base updates performed by an application program that has terminated abnormally. See also dynamic backout.

batch checkpoint/restart. The facility that enables batch processing programs to synchronize checkpoints and to be restarted at a user-specified checkpoint.

batch processing. A processing environment in which data base transactions requested by applications are accumulated and then processed periodically against a data base.

Boolean operator. (ISO)[1] A operator each of the operands of which and the result of which take one of two values.

business process. A defined function of a business enterprise usually interrelated through information requirements with other business processes. For example, personnel management is the business process responsible for employee welfare from pre-hire through retirement. It is related to the accounting business process through payroll.

C

CA. Control area.

call. (1) (ISO)[1] The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (2) (ISO)[1] In computer programming, to execute a call. (3) The instruction in the COBOL, PL/I, or Assembler program that requests DL/I services. For RPG II, see RQDLI command. See also command.

checkpoint. A time at which significant system information is written on the system log, and optionally, the system shut down.

child. Synonymous with child segment.

child segment. A segment one or more levels below the segment which is its parent, but with a direct path back up to the parent. Depending on the structure of the data base, a parent may have many children; however, a child has only one parent segment. Referring to Figure E-2 on page X-1:

- all the B, C, D, E, and F segments are children of A-001.
- C-5 and C-7 are children of B-01 (and A-001), but not children of the other B segments.
- B-02 has no children.

See also logical child and physical child.

CI. Control interval.

combined checkpoint. IN MPS batch processing, a VSE checkpoint followed immediately by a DL/I CHKP command. The two checkpoints together form one logical checkpoint, and allow the use of the VSE restart facility to restart MPS batch programs.

command. The statement in DL/I High Level Programming Interface (HLPI) that requests services for application programs written in COBOL or PL/I. See also call.

command code. An optional addition to the SSA that provides specification of a function variation applicable to the call function.

concatenated key. The key constructed to access a particular segment. It consists of the key fields, including that of the root segment and successive children down to the accessed segment.

control area (CA). A collection of control intervals. Used by VSAM to distribute free space.

control interval (CI). (1) A fixed length amount of auxiliary storage space in which VSAM stores records and distributes free space. (2) The unit of information transmitted to or from auxiliary storage by VSAM.

D

DACT. direct algorithm communication table

---

[1] International Organization for Standardization, Technical Committee 97/Subcommittee 1

data aggregate. A group of data elements that describe a particular entity. Synonymous with segment. See also data element.

data base (DB). (1) (ISO)[1] A set of data, part of the whole of another set of data, and consisting of at least one file, that is sufficient for a given purpose or for a given data processing system. (2) A collection of data records comprised of one or more data sets. (3) A collection of interrelated or independent data items stored together without unnecessary redundancy to serve one or more applications. See physical data base and logical data base.

data base administration (DBA). The tasks associated with defining the rules by which data is accessed and stored. The typical tasks of data base administration are outlined in the DL/I DOS/VS Data Base Administration, SH24-5011.

data base administrator (DBA). A person in an installation who has the responsibility (full or part time) for technically supporting the use of DL/I.

data base description (DBD). A description of the physical characteristics of a DL/I data base. One DBD is generated and cataloged in a core image library for each data base that is used in the installation. It defines the structure, segment keys, physical organization, names, access method, devices, etc., of the data base.

data base integrity. The protection of data items in a data base while they are available to any application program. This includes the isolation of the effects of concurrent updates to a data base by two or more application programs.

data base organization. The physical arrangement of related data on a storage device. DL/I data base organizations are hierarchical direct (HD) and hierarchical sequential (HS). See hierarchical direct organization and hierarchical sequential organization.

data base record. A collection of DL/I data elements called segments hierarchically related to a single root segment.

Referring to Figure E-2 on page X-1: A-001, B-01,C-5, C-7, B-02, B-03, C-2, D, E, F,F,F constitute a data base record.

data base reorganization. The process of unloading and reloading a data base to optimize physical segment adjacency, or to modify the DBD.

data communication (DC). A program that provides terminal communications and automatic scheduling of application programs based on terminal input. For example, CICS/DOS/VS.

data dictionary. (1) A centralized repository of information about data, such as its meaning, relationship to other data, usage, and format. (2) A program to assist in effectively planning, controlling, and evaluating the collection, storage, and use of data. For example, DOS/VS DB/DC Data Dictionary.

data element. The smallest unit of data that can be referred to. Synonymous with field. See also data aggregate.

data field. Synonymous with field.

data independence. (1) The concept of separating the definitions of logical and physical data such that application programs do not depend on where or how physical units of data are stored. (2) The reduction of application program modification in data storage structure and access strategy.

data management block (DMB). The data management block is created from a DBD by the application control blocks creation and maintenance utility, link edited, and cataloged to a core image library. The DMB describes all physical characteristics of a data base. Before an application program using DL/I facilities can be run, one DMB for each data base accessed, plus a PSB for the program itself, must be cataloged in a core image library. The DMBs and the associated PSB are automatically loaded into main storage from the core image library at the beginning of the application program execution (their loading is controlled by the parameter information supplied to DL/I at the beginning of program execution).

data management block (DMB) directory (DDIR). The entries of this directory point to the DMBs known in the DL/I system.

data set. A named organized collection of logically related records. They may be organized sequentially, as in the case of VSE SAM, or in key entry sequence, as in the case of VSE/VSAM. Synonymous with file.

data set group (DSG). A control block linking together a data base with the data sets comprising this DL/I data base.

DB. Data base.

DBA. (1) Data base administration. (2) Data base administrator.

DBD. Data base description.

DBDGEN. Data base definition generation -- the process by which a DBD is created.

DB/DC. Data base/data communication.

DC. Data communication.

DDIR. DMB directory

dependent segment. A DL/I segment that relies on at least the root segment (and other dependent segments) for its full hierarchical meaning. Synonymous with child segment.

destination parent. The physical or logical parent segment reached by the logical child path.

device independence. The concept of writing application programs such that they do not depend on the physical characteristics of the device on which data is stored.

DIB. DL/I interface block.

direct access. The retrieval or storage of a VSAM data record independent of the record's location relative to the previously retrieved or stored record. (See also address direct access and keyed direct access). Contrast with sequential access.

direct algorithm communication table (DACT). Contained in the DMB in DL/I. Provides all the information needed by a HDAM randomizing routine.

distributed data. The ability of DL/I application programs to access a data base that is resident on another processor.

distributed free space. See free space.

DL/I interface block (DIB). Variables automatically defined in an application program using HLPI to receive information passed to the program by DL/I during execution. Contrast with PCB mask.

DMB. Data management block.

DSG. Data set group.

DTF. Define the file -- a control block that connects a program to a data set.

dynamic backout. A process that automatically cancels all activities performed by an application program that terminates abnormally.

E

entity. A item about which information is stored. It has properties that can be recorded. Information about an entity is a record.

entry sequenced data set (ESDS). A VSAM data set whose records are physically in the same order as they were put in the data set. It is processed by addressed direct access or addressed sequential access and has no index. New records are added at the end of the data set.

ESDS. Entry sequenced data set

exclusive intent. The scheduling intent type that prevents an application program from being scheduled concurrently with another application program. See scheduling intent.

F

FDB. field description block

field. (1) (ISO)[1] In a record, a specified area used for a particular category of data, for example, in which a salary rate is recorded. (2) a unique or nonunique area (as defined during DBDGEN) within a segment that is the smallest unit of data that can be referred to. (3) Any designated portion of a segment. (4) See also key field.

field description block (FDB). With each PSDB, there is a list of one or more FDBs describing the fields of a DL/I segment.

field level descriptor (FLD). Contains an entry for each field description in the SSAs of the DL/I call. The entries are used and updated in processing the SSAs of the call.

field level sensitivity. The ability of an application program to access data at the field level. See sensitivity.

file. (ISO)[1] A set of related records treated as a unit. See also data set.

FLD. field level descriptor

forward. Movement in a direction from the beginning of the data base to the end of the data base, accessing each record in ascending root key sequence, and accessing the dependent segments of each root segment from top to bottom and from left to right. Referring to Figure E-2 on page X-1 , forward accessing of all segments shown would be in the following sequence:

A-001, B-01, C-5, C-07, B-02, B-03, C-2, D, E, F, F, F, A-002.

free space. Space available in a VSAM data set for inserting new records. The space is distributed throughout a key sequenced data set (KSDS) or left at the end of an entry sequenced data set (ESDS). Synonymous with distributed free space.

free space anchor point. A fullword at the beginning of a control interval pointing to the first free space element in this CI.

free space element. In HD data bases, the portions of direct access storage not occupied by DL/I segments are called and marked as free space elements.

FSA. free space anchor point

FSE. free space element

H

HD. Hierarchical direct

HDAM. Hierarchical direct access method

HIDAM. Hierarchical indexed direct access method

HIDAM index. A data base that consists of logical DL/I records each containing an image of the key field of a HIDAM root segment. A HIDAM index data base consists of one VSAM KSDS (keyed sequenced data set).

hierarchical direct access method (HDAM). Provides for direct access to a DL/I data base in the HD organization. Segments are stored in VSAM control intervals and are referenced by a relative byte address. Root segments are accessed through a randomizing routine. An HDAM data base consists of one VSAM entry sequenced data set (ESDS).

hierarchical direct organization. An organization of DL/I segments of a data base that are related by direct addresses and may be accessed through an HD randomizing routine or an index.

hierarchical indexed direct access method (HIDAM). Provides for indexed access to a DL/I data base in the HD organization. Segments are stored in VSAM control intervals and are referenced by a relative byte address. Root segments are accessed through a HIDAM index data base. A HIDAM data base consists of one VSAM Entry Sequenced Data Set (ESDS).

hierarchical indexed sequential access method (HISAM). Provides for indexed access to a DL/I data base. A HISAM data base consists of one VSAM key sequenced data set (KSDS) and one VSAM entry sequenced data set (ESDS).

hierarchic sequence. The sequence of segment occurrences in a data base record defined by traversing the hierarchy from top to bottom, front to back, and left to right.

hierarchical sequential access method (HSAM). The segments of a DL/I HSAM physical data base record are arranged in sequential order with the root segments followed by the dependent segments. HSAM data bases are accessed by the VSE sequential access method (SAM).

hierarchical sequential organization. An organization of DL/I segments of a data base that are related by physical adjacency.

hierarchy. (1) An arrangement of data segments beginning with the root segment and proceeding downward to dependent segments. (2) A "tree" structure.

high level programming interface (HLPI). A DL/I facility providing services to application programs written in either COBOL or PL/I Optimizer language through commands.

HISAM. Hierarchical indexed sequential access method

HLPI. High level programming interface

HS. Hierarchical sequential

HSAM. Hierarchical sequential access method

I

IMF. interactive macro facility

index data base. An ordered collection of DL/I index entries (segments) consisting of a key and a pointer used by VSAM to sequence and locate the records of a key sequenced data set (KSDS). Organized as a balanced tree of levels of index.

index data set. Synonymous with index data base.

index pointer segment. The segment that contains the data and pointers used to index the index target segments.

index record. A system-created collection of VSAM index entries that are in collating sequence by the key in each of the entries.

index segment. The segment in the index data base that contains a pointer to the segment containing data (the indexed segment). Synonymous with index pointer segment.

index set. The set of VSAM index levels above the sequence set. An entry in a record in one of these levels contains the highest key entered in an index record in the next lower level and a pointer that indicates the record's physical location.

index source segment. The segment containing the data from which the indexing segment is built.

index target segment. The segment pointed to by a secondary index entry, that is, by an index pointer segment.

indexed segment. A segment that is located by an index. Synonymous with index target segment.

interactive macro facility (IMF). IMF offers easy-to-use interactive procedures that let the user create, modify, and delete DL/I control blocks at a 3270-type terminal.

interactive productivity facility (IPF). The Interactive Productivity Facility is a tool designed to allow users with minimal knowledge of computers to become proficient in either or both the management and use of the computer system.

interactive system productivity facility (ISPF). A dialog manager for interactive applications. It provides control and services to support execution of the dialogs.

interactive utility generation facility (IUG). IUG offers easy-to-use interactive procedures that let the user generate job streams for each of the DL/I utilities at a 3270-type terminal.

intersection data. Any user data in a logical child segment that does not include the logical parent's concatenated key.

inverted file. In information retrieval, a method of organizing a cross-index file in which a key identifies a record. The items pertinent to that key are indicated.

IPF. interactive productivity facility

ISPF. interactive system productivity facility

IUG. interactive utility generation facility

J

JCB. job control block

job control block (JCB). Part of an internal DL/I PSB. One for each PCB. Contains work space and trace information.

K

key. (1) (ISO)[1] One or more characters within a set of data that contains information about that set, including its identification. (2) The field in a segment used to store segment occurrences in sequential order. (3) A field used to search for a segment. See primary key and secondary key. (4) Synonymous with key field and sequence field.

> **Note:** A segment may or may not have a key, that is, a sequence field. All root segments, except for HSAM and simple HSAM data bases, must have keys. DL/I insures that multiple segments of the same type that have keys are maintained in strict ascending sequence by key. The key may be located anywhere within a segment; it must be in the same location in all segments of the same type within a data base. The maximum sizes for keys are 236 alphameric characters for root segments, and 255 for all dependent segments. Keys provide a convenient way to retrieve a specific occurrence of a segment type, maintain the uniqueness and sequential integrity of multiples of the same segment type, and determine under which segment of a group of multiples new dependent segments are to be inserted. Keys should normally be prescribed for all segment types; the exception being if there will never be multiples of a particular type or if a particular segment type will never have dependents.

key field. The field in a segment used to store segment occurrences in sequential ascending order. A key field is also a search field. Synonymous with key and sequence field.

key sequenced data set (KSDS). A VSAM file whose records are loaded in key sequence and controlled by an index. See also keyed direct access and keyed sequential access.

keyed direct access. The retrieval or storage of a data record by use of an index that relates the record's key to its physical location in the VSAM data set, independent of the record's location relative to the previously retrieved or stored record. See also addressed direct access, keyed sequential access, and addressed sequential access.

keyed sequential access. The retrieval or storage of a VSAM data record in its collating sequence relative to the previously retrieved or stored record, by the use of an index that specifies the collating sequence of the records by key. See also addressed sequential access, keyed direct access, and keyed sequential access.

KSDS. Key sequenced data set.

L

level. (1) (ISO)[1] The degree of subordination of an item in a hierarchic arrangement. (2) Level is the depth in

the hierarchical structure at which a segment is located. Roots are always the highest level and the segments at the bottom of the structure are the lowest level. The maximum number of levels in a a DL/I data base is 15. For purposes of documentation and reference, the levels are numbered from 1 to 15, with the root segments being level number 1. Referring to Figure E-2 on page X-1:

- Three levels are shown.
- The A segments (roots) are at the highest level (level 1).
- The C and E segments are at the lowest level (level 3).

level table. Part of an internal DL/I PSB. Contains an entry for each hierarchical level of a DL/I call and is used and updated in processing the SSAs of a call.

local system. (1) A specific system in a multisystem environment. Contrast with remote system. (2) The system in a multisystem environment on which the application program is executing. The local application may process data from data bases located on both the same (local) system and and another (remote) system.

local view. A description of the data that an individual business process requires. See system view.

logical. When used in reference to DL/I components, logical means that the component is treated according to the rules of DL/I rather than physically as it may exist, or as it may be organized, on a physical storage device. For example, a logical DL/I record (a root segment and all of its dependent segments grouped) might be contained on several physical records or blocks on a storage device; and because of prior insertions and deletions, the segments might be in different physical sequence than that by which they are retrieved logically for the application program by DL/I.

logical child. A pointer segment that establishes an access path between its physical parent and its logical parent. It is a physical child of its physical parent; it is a logical child of its logical parent. See also logical parent and logical relationship.

logical data base. A data base composed of one or more physical data bases representing a hierarchical structure derived from relationships between data segments that can be different from the physical structure.

logical data base record. (1) A set of hierarchically related segments of one or more segment types. As viewed by the application program, the logical data base record is always a hierarchic tree structure of segments. (2) All of the segments that exist hierarchically dependent on a given root segment, and that root segment.

logical data structure. A hierarchic structure of segments that is not based solely on the physical relationship of the segments. See also logical relationships.

logical parent. The segment a logical child points to. A logical parent segment can also be a physical parent. See also logical child and logical relationship.

logical relationship. A user defined path between two segments; that is, between logical parent and logical child, which is independent of any physical path. Logical relationships can be defined between segments in the same physical data base hierarchy or in different hierarchies.

logical twin. All occurrences of one type of logical child with a common logical parent. Contrast with physical twin. See also twin segment.

M

MPS. Multiple partition support

MPS restart facility. The capability to restart an MPS batch job when a system or application program failure occurs. This facility uses the VSE checkpoint/restart with the DL/I checkpoint facility.

multiple partition support (MPS). Multiple partition support provides a centralized data base facility to permit multiple applications in different partitions to access DL/I data bases concurrently. MPS follows normal DL/I online conventions in that two programs cannot both update the same segment type in a data base concurrently. However, two or more programs can retrieve from a data base while another program updates it. If one program has exclusive use of a data base, no other program can update it or retrieve from it.

multiple SSA. A series of segment search arguments (SSAs) included in a DL/I call to identify a specific segment or path. See also segment search argument.

O

object segment. The segment at the lowest hierarchical level specified in a particular command. See also path call.

online. A operating environment in which DL/I is used with CICS/DOS/VS (or another data communication program) to permit end-users of application programs to access and store information in a data base through terminals.

option. A command keyword used to qualify the requested function.

P

parent. Synonymous with parent segment.

parent segment. (1) A segment that has one or more dependent segments. Contrast with child. (2) A parent is the opposite of a child, or dependent segment, in that dependent segments exist directly beneath it at lower levels. A parent may also itself be a child. Referring to Figure E-2 on page X-1:

- A-001 is the parent of all B, C, D, E, and F segments.
- D is a parent of E; yet a child of A.
- B-02 is not a parent.
- None of the level-3 segments are parents.

parentage. Establishment in a program of a particular parent as the formal beginning point for the use of the GNP (get next in parent) or GHNP (get hold next in parent) functions. Parentage can only be established by issuing successful GU, GHU, GN, or GHN commands/calls.

partition specification table (PST). Used for communication between DL/I modules when serving an application program call. Contains pointers, parameters and work space. There is one PST for batch and one PST for each DL/I task in an online environment.

partition specification table (PST) prefix (PPST). Logical (but physically separated) part of the PST.

path. The chain of segments within a record that leads to the currently retrieved segment. The formal path contains only one segment occurrence from each level from the root down to the segment for which the path exists. The exact path for each retrieved segment is returned in the PCB by means of the following of its nine fields:

Field 2 Segment hierarchy level indicator

Field 6 Segment name feedback area

Field 7 Length of key feedback area

Field 9 Key feedback area, containing the concatenated keys in the path

Referring to Figure E-2 on page X-1:

- The path to C-5 is A-001, B-01.
- The path to C-7 is the same as the path to C-5.
- There is no path to A-002 because it is a root segment.

path call. (1) The retrieval or insertion of multiple segments in a hierarchical path in a single call, by using the D command code and multiple SSAs. (2) The retrieval, replacement, or insertion of data for multiple segments in a hierarchical path in a single command, by using the FROM or INTO options specifying an I/O area for each parent segment desired. The object segment is always retrieved, replaced, or inserted.

PCB. Program communication block

PCB mask. A skeleton data base PCB in the application program by which the program views a hierarchical structure and into which DL/I returns the results of the application's calls.

PDIR. PSB directory

physical child. A segment type that is dependent on a segment type defined at the next higher level in the data base hierarchy. All segment types, except the root segment, are physical children because each is dependent on at least the root segment. See also child segment.

physical data base. An ordered set of physical data base records.

physical data base record. A physical set of hierarchically related segments of one or more segment types.

physical data structure. A hierarchy representing the arrangement of segment types in a physical data base.

physical parent. A segment that has a dependent segment type at the next lower level in the physical data base hierarchy. See also parent.

physical segment. The smallest unit of accessible data.

physical segment description block (PSDB). Part of a DMB, one per DL/I segment, describing this segment's physical attributes.

physical twins. All occurrences of a single physical child segment type that

have the same (single occurrence) physical parent segment type. Contrast with logical twins. See also twin segment..

PI. Program isolation

pointer. A physical or symbolic identifier of a unique target.

position pointer. For most call functions a position pointer exists before, during, and after the completion of the function. The pointer indicates the next segment in the data base that can be retrieved sequentially. It is normally set by the successful completion of all call functions.

Referring to Figure E-2 on page X-1:

- If A-001 has just been retrieved, it points to B-01.
- If a new segment C-6 has just been inserted, it points to C-7.
- If the D segment has been deleted (E will be deleted along with it), it points to the first F segment.
- If the last F segment has just been retrieved, it points to A-002.

During PSB generation it is possible to specify either single or multiple positioning.

PPST. PST prefix

primary key. The data element, or combination of data elements, within a data aggregate that uniquely identifies an occurrence of that data aggregate. See key and secondary key.

program communication block (PCB). Every data base accessed in an application program has a PCB associated with it. The PCB actually exists in DL/I and its fields are accessed by the application program by defining their names within the application program as follows:

- COBOL - The PCB names are defined in the linkage section.
- PL/I - The PCB names are defined under a pointer variable.
- RPGII - The PCB names are automatically generated by the translator, or may be defined by the user.
- Assembler - The PCB names are defined in a DSECT.

There are nine fields in a PCB:

1. Data base name
2. Segment hierarchy level indicator
3. DL/I results status code
4. DL/I processing options
5. Reserved for DL/I
6. Segment name feedback area
7. Length of key feedback area
8. Number of sensitive segments
9. Key feedback area

program isolation (PI). A facility that isolates all data base activity of an application program from all other application programs active in the system until that application program commits, by reaching a synchronization point, that the data it has modified or created is valid.

This concept makes it possible to dynamically backout the data base activities of an application program that terminates abnormally without affecting the integrity of the data bases controlled by DL/I. It does not affect the activity performed by other application programs processing concurrently in the system.

program specification block (PSB). A PSB is generated for each application program that uses DL/I facilities. The PSB is associated with the application program for which it was generated and contains a PCB for each data base that is to be accessed by the program. Once it is generated, the PSB is cataloged in a core image library, and subsequently processed by a utility along with the associated DBDs to produce the updated PSB and DMBs; all of these are cataloged in a core image library for subsequent use by the application program during execution.

program specification block (PSB) directory (PDIR). Each entry of this DL/I directory points to a PSB. (Only one in batch).

program specification block (PSB) segment intent list (PSIL). With each PSB, there is a PSIL showing in form of a bit mask what sensitivity the application program has for which segments. Scheduling in an online environment is controlled through this list.

PSB. Program specification block

PSBGEN. PSB generation -- the process by which a program specification block is created.

PSDB. physical segment description block

PSIL. PSB segment intent list

PST. partition specification table

Q

qualified call. A DL/I call that contains at least one segment search argument (SSA). See also segment search argument.

qualified segment selection. The identification of a specific occurrence of a given segment type in a command, by using the WHERE option in the command for the desired segment. Contrast with qualified SSA.

qualified SSA. A qualified segment search argument contains both a segment name that identifies the specific segment type, and segment qualification that identifies the unique segment within the type for which the call function is to be performed. See also segment search argument and multiple SSA.

R

RAP. Root anchor point.

RBA. Relative byte address

read-only intent. The scheduling intent type that allows a program to be scheduled with any number of other programs except those with exclusive intent. See scheduling intent.

record. A data base record is made up of at least a unique root segment, and all of its dependent segments. See data base record.

relative byte address (RBA). The displacement of a stored record or control interval from the beginning of the storage space allocated to the VSAM data set to which it belongs.

remote system. In a multisystem environment, the system containing the data base that is being used by an application program resident on another (local) system. Contrast with local system.

root anchor point (RAP). A DL/I pointer in an HDAM control interval that points to a root segment or a chain of root segments.

root segment. The highest level (level 1) segment in a record. A root segment must have a key unless the organization is HSAM or simple HSAM. The sequence of the root segments constitutes the fundamental sequence of the data base. There can be only one root segment per record. Dependent segments cannot exist without a parent root segment; but a root segment can exist without any dependent segments.

RQDLI command. The instruction in the RPGII program used to request DL/I services.

S

SAM. sequential access method

SCD. system contents directory

scheduling intent. An application program attribute defined in the PSB that specifies how the program should be scheduled if multiple programs are contending for scheduling. See exclusive intent, read-only intent, and update intent.

SDB. segment description block

search field. In a given DL/I call, a field that is referred to by one or more segment search arguments (SSAs).

secondary index. Secondary indexes can be used to establish alternate entries to physical or logical data bases for application programs. They can also be processed as data bases themselves. See also secondary index data base.

secondary index data base. An index used to establish accessibility to a physical or logical data base by a path different from the one provided by the data base definition. It contains index pointer segments.

secondary key. A data element, or combination of data elements, within a data aggregate that identifies -- and is used to locate -- those occurrences of the aggregate that have a property named by the key. See key and primary key.

secondary lists. Expansions of DL/I PSDBs to describe or establish relations between segments or data bases (such as the HIDAM index relationship).

segment. A segment is a group of similar or related data that can be accessed by the application program with one I/O function call. There may be a number of segments of the same type within a record.

segment description block (SDB). Sometimes also called LSDB. Description of the logical view an application has of a DL/I segment. One for each sensitive segment. SDBs are contained in the internal PSB.

segment name. A segment name is assigned to each segment type. Segment names for the different segment types must be unique within a data base. The segment name is used by the application programmer when constructing a qualified or unqualified SSA prior to issuing a call for a specific segment. Synonymous with segment type.

segment occurrence. One instance of a set of similar segments.

segment search argument (SSA). Describes the segment type, or specific segment within a segment type,

that is to be operated on by a DL/I call. See also multiple SSA, qualified SSA, and unqualified SSA.

segment selection. The specifying of parent and object segments by name in a command. Selection may be either qualified or unqualified. Contrast with segment search argument.

segment type. Different segment types may have different lengths, but within each single type, all segments must be the same length (unless variable length segments have been specified by DBA).

Referring to Figure E-2 on page X-1, there are six different types of segments: A through F. Synonymous with segment name.

sensitivity. (1) A DL/I capability that ensures that only data segments or fields predefined as "sensitive" are available for use by a particular application program. The sensitivity concept also provides a degree of control over data security, inasmuch as users can be prevented from accessing particular segments or fields from a logical data base. (2) Sensitivity to the various segments and fields that constitute a data base is controlled, on a program-by-program basis, when the PSB for each program is generated. For example, a program is said to be sensitive to a segment type when it can access that segment type. When a program is not sensitive to a particular segment type, it appears to the program as if that segment type does not exist at all in the data base. Segment sensitivity applies to types of segments, not to specific segments within a type, and to all segment types in the path to the lowest level sensitive segment type.

sequence field. Synonymous with key field.

sequence set. The lowest level of a VSAM index. It immediately controls the order of records in a key sequenced data set (KSDS). A sequence set entry contains the key of the highest keyed record stored in a control interval of the data set and a pointer to the control interval's physical location. A sequence set record also contains a pointer to the physical location of each free control interval in the fan-out of the record.

sequential access. The retrieval or storage of a VSAM or SAM data record in either its physical order or its collating sequence relative to the previously retrieved or stored record. (Also see addressed sequential access and keyed sequential access).

sequential access method (SAM). The data management used to access HSAM data bases.

sequential processing. Processing or searching through the segments in a data base in a forward direction. See also forward.

simple HISAM. A hierarchical indexed sequential access method data base containing only one segment type.

source segment. A segment containing the data used to construct the secondary index pointer segment. See also secondary index data base.

SSA. segment search argument

status code. Each DL/I request for service returns a status code that reflects the exact result of the operation. The first operation that a program should perform immediately following a DL/I request is to test the status code to ensure that the function requested was successful. Following a command, the status code is returned in the DIB at the label DIBSTAT. Following a call, the status code is returned in field 3 of the PCB.

sync(h) point. Synonymous with synchronization point.

synchronization point. A logical point in time during the execution of a application program where the changes made to the data bases by the program are committed and will not be backed out if the program subsequently terminates abnormally. Synonymous with sync point or synch point.

A synchronization point is created by:

- a DL/I CHECKPOINT command or CHKP call
- a DL/I TERMINATE command or TERM call
- a CICS/VS synch point request
- an end of task (online) or an end of program (MPS-batch).

system contents directory (SCD). Shows entry points of all major modules of the DL/I system and other system-related information.

system view. A conceptual data structure that integrates the individual data structures associated with local views into an optimum arrangement for physical implementation as a data base. See local view.

T

transaction. A specific set of input data that triggers the execution of a specific process or job.

twin segments. All child segments of the same segment type that have a par-

ticular instance of the same parent type. See also physical twins and logical twins.

twins. Synonymous with twin segments.

U

unqualified call. A DL/I call that does not contain a segment search argument.

unqualified segment selection. The identification of a given segment type in a command without specifying a particular occurrence of that segment type (without using the WHERE option). As a general rule, unqualified segment selection retrieves the first occurrence of the specified segment type. Contrast with unqualified SSA.

unqualified SSA. An unqualified SSA contains only a segment name that identifies the specific type of segment for which the I/O function is to be performed. As a general rule, the use of an unqualified SSA retrieves the first occurrence of the specified type of segment. See also segment search argument

update intent. The scheduling intent type that permits application programs to be scheduled with any number of other programs except those with exclusive intent. See scheduling intent.

UPSI. User program switch indicator. A switch (or flag) byte in the VSE Partition Communication Region that can be set using the // UPSI job control statement. Programs can test the settings of the UPSI switches to control function and/or execution flow.

V

virtual storage access method (VSAM). The data management used to access HISAM, HD randomized, HD indexed, HDAM, and HIDAM.

VSAM. virtual storage access method

A

B

E

F

G

H

I

J

K

L

M

O

P

R

S

T

DL/I DOS/VS
Diagnostic Guide
Order No. SH24-5002-4

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

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.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? _____

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp 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.)

Reader's Comment Form

Cut or Fold Along Line

DL/I DOS/VS Diagnostic Guide (File No. S370/4300-50) SH24-5002-4

Fold and Tape Please Do Not Staple Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

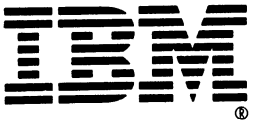
State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

SH24-5002-4

DL/I DOS/VS Diagnostic Guide (File No. S370/4300-50) SH24-5002-4



SH24-5002-04