LY12-5016-6
File No. S370/4300-50

Program Product

Data Language/I Disk Operating System/ Virtual Storage (DL/I DOS/VS) Logic Manual, Volume 1

Program Number 5746-XX1

Seventh Edition (June 1981)

This edition applies to Version 1, Release 6 (Version 1.6) of IBM System/370 Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1. It supersedes LY12-5016-5.

This edition, LY12-5016-6, in conjunction with DL/I DOS/VS Logic Manual, Volume 2, LY24-5215 is a major revision of LY12-5016-5.

Summary of Amendments

For a list of changes, see page iii. Changes and additions are indicated by a vertical line to the left of the change.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

Publications are not stocked at the address given 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 Programming Publications, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatsoever. You may, of course, continue to use the information you supply.

DL/I VERSION 1.6

This version of DL/I provides system changes and functional enhancements such as:

Limited Data Sharing (Read Only)

This function supports sharing of data bases between DL/I subsystems in one host or across hosts. One subsystem with update capability and multiple read-only subsystems can execute concurrently. This function does not guarantee data consistency for the read-only subsystem.

MPS Under Interactive Computing and Control Facility (ICCF).

DL/I MPS allows multiple MPS batch jobs to run in a single DOS/VSE partition.

Boolean Qualification Statements

Boolean logic qualification decreases the application program logic necessary for complex data retrieval. The user specifies multiple qualification statements to perform Boolean logic qualification for each segment. Boolean AND and OR operators logically relate the qualification statements.

ACCESS Macro

The new ACCESS macro allows the user to specify on one statement all of the necessary parameters to define an access point to an HD data base. The ACCESS macro automatically generates the definition of any required index data base DBDs.

Selective Unload

With selective unload, the user can reformat data using Field Level Sensitivity and Segment Sensitivity. The user can also add new fields for an application program and move a subset of a data base to another location for faster processing.

Current Position Trace Entry Addition

This function adds two fields (SDBORGN and SDBPTDS) to the current position trace entry. These fields specify the data base organization and physical pointers for the segment.

DL/I Trace Point Utility Improvement

This enhancement provides a means of selecting which trace entries print from a file created by DL/I Trace with OUTPUT=CICS. This function reduces the amount of output generated by the Trace Print Utility.

Rewind Option for Reorganization Utilities

This support adds an option to the HISAM and HD reorganization unload and reload utilities to allow the user to not rewind input and output tapes, or to select rewind only without having the tapes unloaded. This enables the user to reorganize multiple data bases without having to mount a new tape for each data base reorganized.

Seperate Index Reorganization

With this function, the user can now reorganize an index data base separately by using the HISAM unload and relcad utilities.

Partial Data Base Reorganization Utility

This utility reorganizes a user-selected range of HIDAM or HDAM data base records into a designated target area within a data base. This minimizes the time a data base is offline for reorganization.

Run and Buffer Statistics

This facility reports statistics for certain run and buffer events that are currently collected by DL/I, but not formatted or displayed. The data base administrator or system programmer uses the statistics in selecting parameters for system tuning.

Extended Remote PSB

This support enables CICS/VS applications to process both local and remote DL/I data bases within the same CICS/VS logical unit of work. To application programs, a concatination of PCBs from local and remote PSBs appear as a single PSB containing views of both local and remote data bases.

HLPI ICR Intigration

This release recoganizes the content of the logic manual by dividing the book into two volumes. Volume 1 includes all of the information included in prior editions except Section 2, Method of Operation. Section 2 is now in Volume 2.

DL/I Version 1.5

This version of DL/I provides system changes and functional enhancements such as:

Field Level Sensitivity

This function makes it possible for the user to specify only those fields in the physical definition of a given segment that are to be included in his application's view of that segment, while remaining insensitive to the other fields in the segment.

Extended Logical Relationships

The restriction of only one logical relationship per logical path has been removed. The user may now define as many logical relationships as he needs to satisfy his requirements.

Unique Segment Support

It is possible for the user to specify that only one occurrence of a particular segment type is allowed under a particular parent.

Selective Log Print

It is possible for the user to selectively print data from the log, using the log print utility, by specifying a DBD name, CICS task ID, or relative block number.

DL/I FBA Device Support ICR

Technical Newsletter LN24-5614 documents the following from the FBA device support Independent Component Release (ICR):

FBA Device Support

This support makes it possible for data bases and utility work files to reside on Fixed Block Architecture devices.

DL/I Version 1.4

This version of DL/I provides system changes and functional enhancements such as:

RPG II Support

Application programs written in RPG II can now access DL/I data bases in a manner similar to programs written in COBOL, PL/I, and Assembler language.

Prefix Resolution Improvement

The prefix resolution utility now passes an actual maximum record length, instead of a maximum possible record length, to the DOS/VS or DOS sort/merge program.

Extended DL/I Call Interface

This support, along with CICS/VS high level language support, eliminates the need for application programs to reference internal CICS/VS control blocks. A new parameter has been added to the PCB call to obtain the address of the DL/I User Interface Block. This control block contains the information previously returned in the TCA.

This enhancement is required for application programs written in RPG II. It may also be used in programs written in COBOL, PL/I, and Assembler.

Intersystem Communication

CICS/VS intersystem communication support enables DL/I application programs to access a data base that is resident on another CPU.

High Level Language Debugging for PL/I

This support for PL/I allows diagnostic information to be supplied by both PL/I and DL/I. It is designed for only batch and MPS batch execution of DL/I, and does not require any changes to the PL/I code.

Performance Improvements

Performance improvements have been made to image copy, the batch partition controller, the HD unload utility, the log buffer and log print utility; and program isolation.

This manual is to be used with the program listings for DL/I DOS/VS. It discusses the internal operation of the DL/I system as an application program under DOS/VS. It is intended for use by persons involved in program maintenance and by system programmers who are altering the program design.

DL/I DOS/VS is a data management control system that assists the user in creating, accessing, and maintaining large common data bases. In conjunction with the Customer Information Control System (CICS/VS), DL/I DOS/VS can be used in an online teleprocessing environment.

Readers of this manual must be thoroughly familar with the use of DOS/VS, and of CICS/VS, if DL/I DOS/VS is to be used in the online or multiple partition support (MPS) environment.

Because DL/I DOS/VS is a functional subset of the IBM Information Management System/Virtual Storage (IMS/VS), some specific IMS or OS terms are used in this manual. These terms are used to allow easy reference to the documentation of the related systems.

This manual is divided into seven sections.

Section 1: Introduction: Summarizes DL/I DOS/VS giving general information about the purpose of system control modules, DL/I facility modules, MPS modules, and utility modules.

Section 2: Method of Operation: Contains HIPO diagrams that describe the DL/I modules. The diagrams include cross-references to labels in the program listings. See Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) Logic Manual, Volume 2 Order No. LY24-5215.

Section 3: Program Organization: This section provides descriptive information about the DL/I modules and major routines.

SECTION 4: Directory: Lists DL/I module, entry point, and control section names with cross-references to Section 2: Method of Operation.

Section 5: Data Areas: Describes the data areas used by DL/I. Field and flag names for each data area are also listed alphabetically.

Section 6: Diagnostic Aids: Gives information that may be helpful in locating specific program listings.

Section 7: Appendixes: Contains information about LLC/CC in DL/I, DBD generation, PSB generation and DL/I macros.

An index is also included.

Note: In this publication, the system and component name DOS/VS should be read as DOS/VSE unless that name explicitly refers to DOS/VS release 34 or an earlier DOS/VS release.

Related Publications

DL/I DOS/VS General Information Manual, GH20-1246
DL/I DOS/VS Application Programming: CALL and RQDLI, SH12-5411

DL/I DOS/VS Data Base Administration, SH24-5011
DL/I DOS/VS Resource Definition and Utilities, SH24-5021
DL/I DOS/VS Messages and Codes,SH12-5414
DL/I DOS/VS Guide for New Users, SH24-5001
DL/I DOS/VS Diagnostic Guide, SH24-5002
DL/I DOS/VS Logic Manual Volume 2, LY24-5215

For DOS/VS messages and return codes:

DOS/VSE Messages, GC33-5379
DOS/VSE Macro User's Guide, GC24-5139
DOS/VSE Macro Reference, GC24-5140
Using VSE/VSAM Commands and Macros, SC24-5144
VSE/VSAM Messages and Codes, SC24-5146

Users employing DL/I DOS/VS in an online environment should have access to the following CICS/VS publications:

CICS/VS System Programmer's Reference Manual, SC33-0069
CICS/VS Application Programmer's Reference Manual (Macro Level), SC33-0079
CICS/VS System Application Design Guide, SC33-0068
CICS/VS System Programmer's Guide (DOS/VS), SC33-0070.

FIGURES

Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS, hereafter referred to as DL/I) is a data management control system that assists the user in creating, accessing, and maintaining large common data bases. In conjunction with the Customer Information Control System (CICS/DOS/VS), DL/I can be used in an online teleprocessing environment. Also in conjunction with CICS/VS, DL/I provides a centralized data facility, multiple partition support (MPS), which controls concurrent access to data bases from multiple batch partitions.

Section I summarizes and describes the following:

- DL/I Batch System
- DL/I Online Processor
- DL/I Facility Modules
- Multiple Partition Support (MPS)
- DL/I Utilities

DL/I BATCH SYSTEM

The DL/I batch system executes as an application program in a virtual storage environment under DOS/VS. The DOS/VS partition in which the DL/I batch system executes is composed of the elements shown in Figure 1-1. These are:

- The system control facility
- The DL/I facility
- The DOS/VS VSAM and SAM data management modules
- The user application program

The major components of the DL/I system are the system control facility and the DL/I facility. The system control facility receives control from DOS/VS job control, initializes the DL/I batch system, and interfaces between DL/I and the user application program. The DL/I facility interfaces with the DOS/VS VSAM and SAM data management modules when performing the data base call function requested by the user application.

The system control facility is divided into three functional areas (see Figure 1-2):

- Batch initialization
- Language interface
- Program request handler.

Batch initialization is responsible for:

- Initial interface with DOS/VS job management

- Analysis and validity checking of DL/I parameter information

- Loading the batch nucleus.

- Loading the DL/I application control blocks (PSB and DMBs) and relocating the control block addresses.

- Creation of the PSB intent list and the DMB directory (DDIR).

- Acquiring and formatting storage for the buffer pool control blocks and their related I/O buffers.

- Loading the DL/I facility modules.

- Loading the application program and passing control to it.

The language interface provides communication between the application program and the program request handler. This module is link-edited with the application program and provides a common interface for DL/I calls written in PL/I, COBOL, RPG II, or Assembler language.

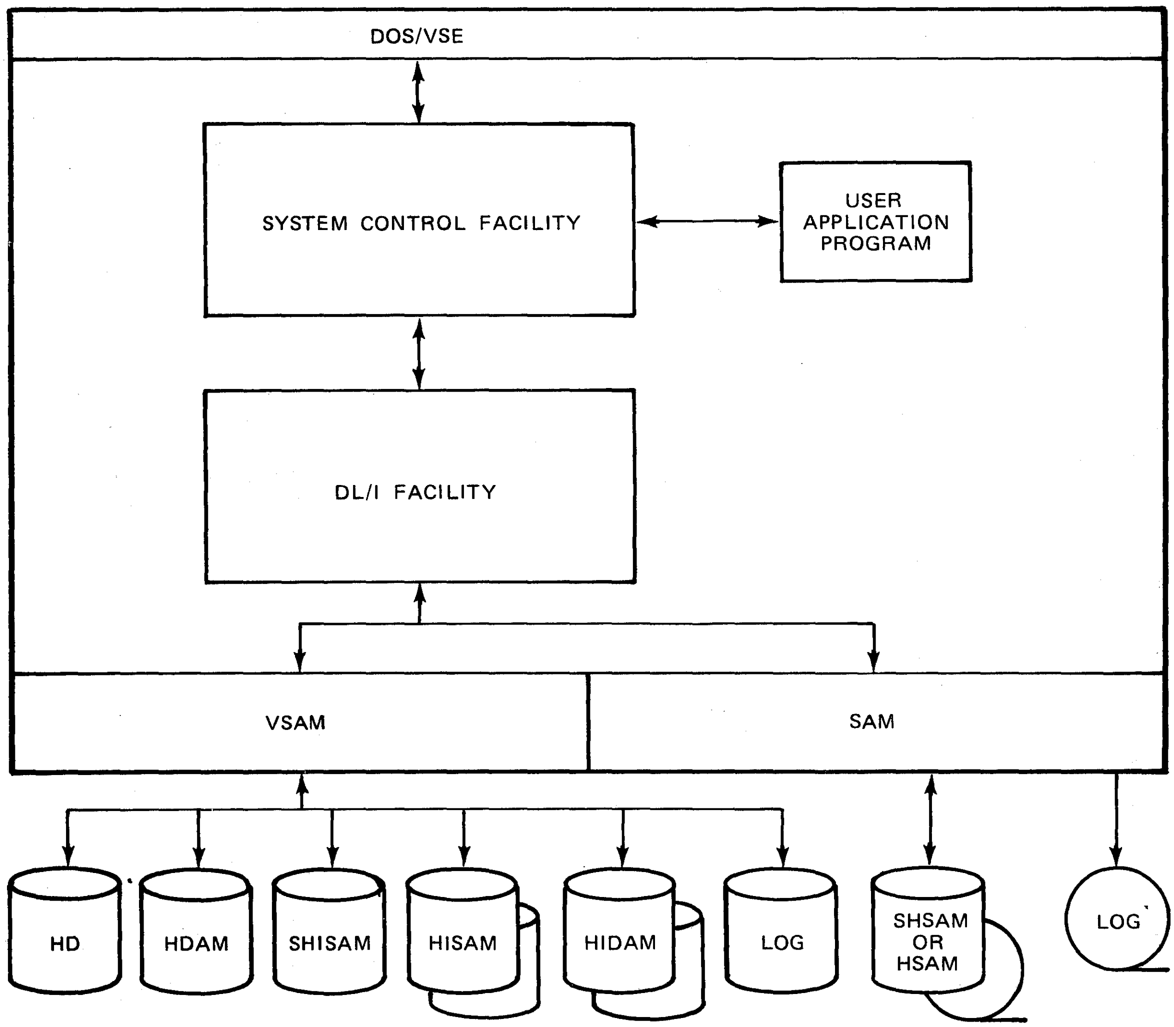


Figure 1-1. Elements of a DL/I DOS/VS Batch Partition

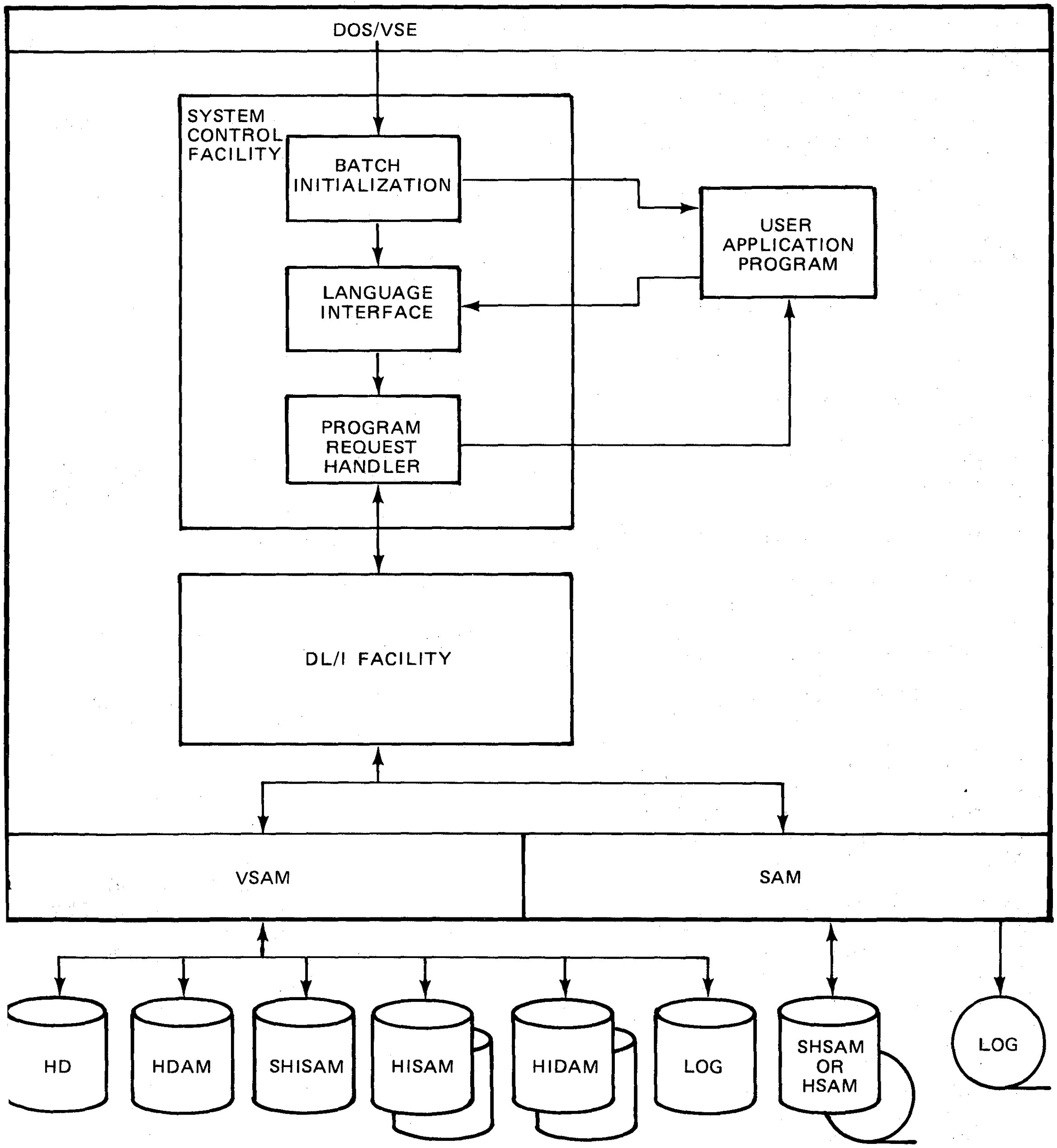


Figure 1-2. System Control Facility Relationships

The program request handler receives the DL/I call from the user application program via the language interface. It performs the following functions:

- Checks validity and, if necessary, reformats the caller's parameter lists and submits them to the DL/I facility.
- Accepts parameter lists from the DL/I facility and moves data to the user's work area, if required.
- Returns control directly to the user application program.

See Section 3 for a detailed description of each of these modules.

DL/I ONLINE PROCESSOR

In an online environment, the DL/I system executes within the CICS/VS partition. CICS/VS provides exit interfaces to DL/I for the following:

- DL/I system initialization during CICS/VS initialization.
- DL/I system termination during CICS/VS termination.
- DL/I user task scheduling of DL/I resources before an application program accesses DL/I.
- DL/I user task completion and return of DL/I resources after the application program has issued a CICS/VS synchronization point (SYNCPOINT) command or has completed DL/I processing.

When the user application program issues a DL/I call, control passes to the online language interface module and the program request handler. The program request handler validates the call and passes it to the DL/I facility. The DL/I facility invokes CICS/VS services through the online interface for such functions as transaction and storage management. On completion of the DL/I call, the DL/I facility returns control to the user application program via the program request handler. The program request handler also interfaces with CICS/VS for any functions performed externally to DL/I.

DL/I FACILITY MODULES

The functions of data base creation, access, maintenance, and reorganization are accomplished by the DL/I facility (see Figure 1-3). The DL/I call is passed from the system control facility to the DL/I call analyzer, which is the focal point of the DL/I facility. The type of call is analyzed (DL/I call, pseudo call, or internal call resulting from a DL/I call), and control is passed to the appropriate action module to process the call.

The action modules of the DL/I facility, together with their major functions, are listed below:

- Open/Close Module
  - Open DL/I data bases
  - Close DL/I data bases
  - Interface with data base logger to write data set open record to log file
- Delete/Replace Module
  - Delete a segment of a DL/I data base in conjunction with the buffer handler
  - Replace a segment of a DL/I data base in conjunction with the buffer handler
  - Interface with data base logger to record changes on log file
  - Interface with space management for HDAM and HIDAM data bases
  - Interface with index maintenance for data bases with indexes
- Load/Insert Module
  - Load segments into a DL/I data base in conjunction with the buffer handler
  - Insert segments into a DL/I data base in conjunction with the buffer handler
  - Interface with data base logger to record changes on log file
  - Interface with space management for HDAM and HIDAM data bases
  - Interface with index maintenance for data bases with indexes
  - Issue I/O for HSAM and Simple HSAM data bases
- Retrieve Module
  - Retrieve a segment of a DL/I data base in conjunction with the buffer handler
  - Perform data base positioning for load/insert
  - Issue I/O for HSAM and Simple HSAM data bases
- Index Maintenance
  - Maintain any indexes for HDAM or HIDAM data bases in conjunction with the buffer handler
  - Interface with data base logger to record changes on log file
- Space Management
  - Allocate and maintain free space on DASD in conjunction with the buffer handler for storage of DL/I segments for HDAM and HIDAM data bases
  - Interface with data base logger to record changes on log file

- Buffer Handler
  - For HDAM or HIDAM data base, satisfy requests for segments or records from data currently available in the buffer pool
  - Issue I/O to VSAM for HDAM or HIDAM data base requests that cannot be satisfied from the buffer pool
  - Issue I/O to VSAM for all HISAM, Simple HISAM, and Index data base requests
- Data Base Logger
  - Record all data base modifications on the DL/I log tape using DOS/VS SAM or disk log using VSAM, or CICS Journal
- Queuing Facility
  - Provide support for contention control at the segment and record level
  - Provide deadlock detection and resolution.
- Field Level Sensitivity Copy Module
  - Provide user view/physical view conversion for field level sensitivity.

See Section 3 for a detailed description of the modules.

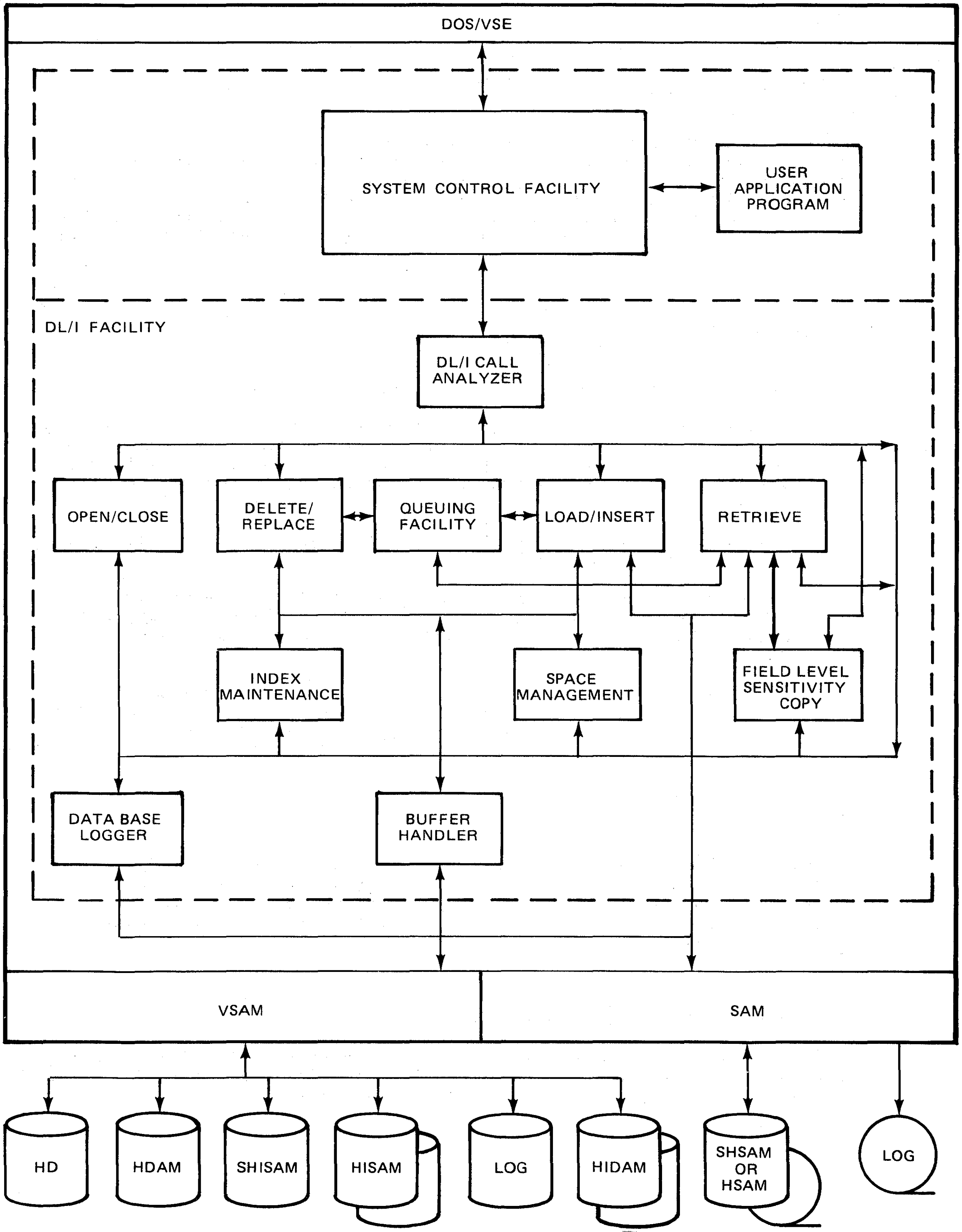


Figure 1-3. DL/I Facility Relationships

MULTIPLE PARTITION SUPPORT (MPS)

DL/I enables batch application programs executing in different partitions to access online data bases concurrent with online applications. This capability is called multiple partition support (MPS). For example, MPS permits online applications to issue inquiries to a data base while a batch program updates the data base. MPS uses the DL/I resources and the multitasking facilities of DL/I and CICS/VS.

DL/I UTILITIES

The DL/I utility modules are categorized as follows:

- Application control blocks creation and maintenance: this utility program is used to merge and expand into an internal format the control blocks created by the DBD and PSB generation utilities. The control blocks created by this utility are used by the DL/I system.
- Data base recovery: this is a set of utility programs employed to reconstruct a data base.
- Data base reorganization: this is a set of utility programs employed to reorganize a data base. Use of these programs reduces direct access storage requirements by compacting data and thus reducing data base access time.
- Data base logical relationship resolution: this is a set of utility programs employed to update pointer information when data bases involved in logical relationships and/or secondary index relationships are initially loaded or reorganized.

HLPI INTERFACE MODULES

The HLPI interface modules, DLZEIPO0, DLZEIPB0, and DLZEIPB1 build DL/I calls from data provided in calls generated from EXEC DLI commands by the CICS EXEC translator. After the HLPI interface modules build the DL/I calls, they pass the calls to the Program Request Handler for execution by DL/I.

LANGUAGE INTERFACE MODULES

There are two language interface modules used with batch and MPS HLPI programs. They are the COBOL language interface module (DLZLICBL) and the PL/I language interface module (DLZLIPLI).

SECTION 2: METHOD OF OPERATION

This section contains HIPO (Hierarchy, plus Input, Process, Output) diagrams and is included in Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) Logis Manual, Volume 2, Order Number LY24-5215.

SECTION 3: PROGRAM ORGANIZATION

This section contains descriptions of the DL/I modules and their major routines.

SYSTEM CONTROL MODULES

DLZRRC00 - BATCH INITIALIZATION - Part 1

The responsibilities of this module are to:

- Read required PARM information from SYSIPT or SYSLOG based on the UPSI byte setting.
- Determine load address for batch nucleus module (DLZBNUC0).
- Provide a DL/I message subroutine (ERRORMSG).
- Branch to region control interface (DLZRRC10).

Entry Interface - DLZRRC00

DLZRRC00 receives control from DOS/VS job control

Exit Interface

DLZRRC00 passes control through branch to region control interface (DLZRRC10).

Register Contents

R7	Address of ERRORMSG
R10	Entry point address of DLZRRC10

Entry Interface - ERRORMSG

ERRORMSG receives control through BALR from DL/I modules

Register Contents

R1	PST address or parameter list address
R13	Save area address
R14	Return address
R15	Entry point address (DLZERRMS)

Exit Interface - Calling Module

Passes control through branch on register 14

DLZRRC10 - REGION CONTROL/INITIALIZATION - Part 2

This routine receives control from the DL/I initialization Part 1 routine and continues batch initialization. Its responsibilities are:

- Save input parameters
- Load batch nucleus module (DLZBNUC0)
- Establish SCD and PST addressability
- Invoke parameter analysis (DLZRRA00)

* Branch to application program control module (DLZPCC00)

Entry Interface - DLZRRC10

Receives control through branch from DLZRRC00

Register Contents

R7	Address of ERRORMSG
R10	Entry point address

Exit Interface - Parameter Analysis

Passes control through fall through to DLZRRA00

Register Contents:

R2	Address of SCD
R9	Address of PST
R13	Save area address

DLZRRA00 - USER PARAMETER ANALYSIS

This routine checks the positional parameters for valid length and contents when first entered. Invalid parameters cause DL/I to issue an error message and abnormally end. There is an entry at NXTPORT (just before buffers are to be allocated) to check keyword parameters. Errors cause DL/I to issue an error message and abnormally end.

Layout and Description of PARM Field

	xxx,aaaaaaaa,bbbbbbb,ccc,keyword operands
xxx	PARM identifier in columns 1-3. DLI = Data base program to be executed. UDR = Data base recovery utility to be executed. ULU = Data base reorganization or logical relationship resolution program to be executed. ULR = HD reorganization reload utility to be restarted from checkpoint record. PLU = Selective Unload
aaaaaaaa	One- to eight-character name of the application program to be executed.
bbbbbbb	One- to seven-character name of the program specification block (PSB) as specified in the PSB generation. If PARM is UDR, ULU, or ULR, one- to seven-character name of the data base description (DBD) as specified in the DBD generation.
ccc	Number of data base buffer sub-pools required for job execution.
keyword operands	HDBFR, HSBFR, ASLOG, LOG, and TRACE

Entry Interface

Receives control from DLZRRC10

Entry Register Contents

- When entered at DLZRRA00:
  R2 Pointer to SCD (not used)
  R9 PST address
  R13 Save area address (not used)

- When entered at NXTPORT:
  R6 Pointer to first subpool information table
  R8 SCD address

Exit Interface

- From DLZRRA00 entry: Passes control by fall through to DLZPCC00

- From NXTPORT entry: Passes control by branch to PRMSRET

Exit Register Contents:

- From DLZRRA00 entry:
  R2 SCD address
  R9 PST address

R13   Save address

- From NXTPORT entry:
  R2   SCD address
  R6   Pointer to last subpool information table
  R9   PST address
  R13   Save area address

DLZPCC00 - APPLICATION PROGRAM CONTROL

This routine is used only in the batch partitions. It performs some functions analogous to those performed by the CICS scheduler in the online control program. It is responsible for the following functions:

- Initializing the storage management routine
- Invoking the application control blocks loader/relocator (DLZPINIT)
- Invoking the control program initialization routine
- Loading the application program
- Initializing the PL/I region (if PL/I)
- Invoking the application program
- Issuing an unload call in behalf of the application program upon termination
- Writing the application program termination record on the DL/I log
- Closing the DL/I log.

Data Areas Used

PST
SCD
DDIR
DMB
SDB
PSIL

Entry Interface

Receives control by fall through from DLZRRA00

Entry Register Contents

R2   SCD address
R9   PST address
R13   Save area address

Exit Interface

- Passes control through BAL to DLZPINIT (entry point in DLZDBLM0)
- Passes control through BAL to application program
- Passes control through BAL to call analyzer (DLZDLA00)

- Passes control through BAL to data base logger DLZRDBL0)
- Passes control to DOS/VS supervisor by issuing an SVC 14 normal EOJ supervisor call.

Exit Register Contents

- From exit to DLZPINIT:

R2	SCD address
R9	PST address
R14	Return address

- From exit to application program:

R1	Address of PCB address list
R13	Save area address
R14	Return address
R15	Entry point

- From exit to DLZDLA00:

R1	PST address
R13	Save area address
R14	Return address
R15	Entry address of call analyzer (obtained from SCD at label SCDDLICT)

- From exit to DLZRDBL0:

R1	PST address
R13	Save area address
R14	Return address
R15	Entry point of log write-only routine (obtained from SCD at label SCDREENT) or, Entry point of force write routine (obtained from SCD at label SCDDBLFW) or, Entry point of logger close routine (obtained from SCD at label SCDDBLCL)

DLZDBLM0 - APPLICATION CONTROL BLOCKS LOAD AND RELOCATE

This routine performs the functions of loading and relocating DL/I application control blocks. Once the blocks are loaded and offsets resolved to actual addresses, the SDBs in the PCBs are connected to the appropriate PSDBs in the DMBs. The JCB data sets in the data base are connected to the appropriate ACBs in the DMBs, and control is returned to the calling routine.

For 'DLI' or 'PLU' execution, the PSB name extracted from the PARM card is moved to the PSB directory and the PSB is loaded. The address of the PSB segment intent list and the PSB are stored in the PSB directory. The index work area (if required) is allocated and addresses are resolved. Next the intent list is scanned and the DMB directory is constructed from it. The DMB directory entries are scanned and the DMBLOADR subroutine (see below) is called to load and relocate the DMBs in the directory. Upon completion, the SDBs are connected to their corresponding PSDBs, the JCB DSGs are connected to their ACBs, and return is made to the caller.

For the following utilities there is no PSB name in the parameter information:

DLZURPR0 - Data base prereorganization
DLZURGS0 - Data base scan
DLZURGP0 - Data base prefix update

These utilities perform dynamic block loading using the DLZBLKLD macro.

The DMBLOADR subroutine performs the loading and relocation of DMBs. The DMB directory is accessed and the DMB name extracted from it. A load is issued for the DMB and, if HDAM, the randomizing module extracted from the DMB is loaded. Next, the DMB directory entry is updated with a buffer size indication. For HD, this value is the control interval size of the data set; for HISAM, it is the logical record size. Then all offsets are relocated to addresses, and control is passed to DLZCPI00.

Entry Register Contents:

R2	SCD address
R9	PST address
R13	Address of one of a set of prechained save areas
R14	Return address

Exit Register Contents

Same as entry register contents

DLZCPI00 - BATCH CONTROL PROGRAM INITIALIZATION

This routine receives control from the application control blocks load and relocate routine and completes the intialization of the DL/I batch system. It is responsible for:

- Allocation of the buffer pool
- Formatting the buffer pool prefix, one or more subpool prefixes, and the buffer prefixes
- Loading all required DL/I action modules
- Initializing the SCD
- Opening the DL/I log
- Writing the application program scheduling record on the DL/I log

Entry Interface - DLZCPI00

Receives control by fall through from routine DLZDBLM0.

Entry Register Contents:

R2	SCD address
R9	PST address
R13	Save area address

Exit Interface

Returns to DLZPCC00

Exit Register Contents

R9	PST address
R2	SCD address
R14	Return address

DLZLI000 - LANGUAGE INTERFACE

The language interface provides communication between the application program and the program request handler. A copy of this module is link edited with user application programs.

The language interface has responsibility for:

- Storing the user's registers in the save area provided.
- Providing a specific entry for Assembler, COBOL, RPG II, and PL/I application programs.
- Locating the entry point of the program request handler.
- Passing control to the program request handler

Entry Interface - DLZLI000

Receives control through branch from application program

Entry Register Contents:

R1	Call parameter list of implicit or explicit format
R13	Save area address
R14	Return address
R15	Entry point

Exit Interface

Passes control to program request handler through branch from DLZLI000

Exit Register Contents:

R0	Language identifier code
R1	Parameter list
R2-14	As entered from application program
R15	Entry point of program request handler

DLZLICBL - DL/I DOS/VS HLPI BATCH/MPS COBOL LANGUAGE INTERFACE

This module obtains the entry point address of and passes control to DLZEIPB0.

Control Blocks - DLZLICBL

EIPL - EIP parameter list

Normal Entry Point

The entry points to this module are:

DLZEI01 - Data base calls
DLZEI02 - All other calls
DLZEI03 - Reserved
DLZEI04 - Reserved

Register Contents on Entry

R13 - Register savearea address

DLZLIPLI - DL/I DOS/VS HLPI BATCH/MPS PL/I LANGUAGE INTERFACE

This module has two routines; An initialization routine with an entry point DLZLIPLI and a processing routine with an entry point DLZEI0x.

Entry point DLZLIPLI is entered before the application program gets control. It finds the entry point address of PLICALLB and passes control to it. This is done to enable the PL/I HLPI application program to use non-PL/I PSBs.

DLZEI0x performs the same functions as DLZLICBL (see DLZLICBL for details).

CONTROL BLOCKS - DLZLIPLI

- EIPL - EIP parameter list

Normal Entry Points

The normal entry points to this module are:

DLZLIPLI - From DL/I initialization
DLZEI01 - All other calls
DLZEI02 - Data base calls
DLZEI03 - Reserved
DLZEI04 - Reserved

Register Contents on Entry

R13 - Register savearea address

DLZPRHB0 - PROGRAM REQUEST HANDLER

The interface between the application program and the DL/I batch or control program is managed by the program request handler routine (DLZPRHB0) in module DLZBNUC0. It accepts parameters passed to it by the language interface module (DLZLI000), or the HLPI batch EXEC interface program, DLZEIPB1. It validates these parameters and passes a parameter list to the call analyzer.

The program request handler accepts three call list formats: implicit direct, explicit direct, and explicit indirect. COBOL and Assembler-language programs may use either the implicit direct or explicit direct call list formats. Since special provisions are made for PL/I in handling the explicit indirect call list, it may be used only by PL/I language programs.

The first parameter (argument 0) of the DL/I CALL determines whether the list is explicit or implicit. If the argument contains the address of the parameter count (count of the number of arguments that follow), this list is an explicit list. If the argument contains the address of the DL/I CALL function, this list is an implicit list.

The responsibilities of this routine are to:

- Verify parameter list addresses aligned and within the dynamic area of the machine
- Reformat explicit parameter lists to implicit prior to submission
- Reset PL/I STXIT PC processing
- Provide caller's parameter list to the call analyzer
- Return data to application program work areas
- Maintain PL/I variable-length character string dope vector
- Identify abnormal termination condition
- Return directly to application program
- Write checkpoint message if checkpoint issued

Data Areas Used

PPST
PST
SCD

Entry Interface

Receives control through branch from language interface (DLZLI000)

Entry Register Contents

R0	Language indicator Bit X'01' ON if PL/I, OFF for other languages. Bit X'02' ON if HLPI, Off if call interface
R1	Parameter list address (in application program format)
R13	Save area address
R14	Return (to application program)
R15	Entry point address

Exit Interfaces

- Passes control through branch to call analyzer (DLZDLA00)
- Passes control through branch to error message writer (ERRORMSG)
- Passes control through branch to abend processor (DLZABEND)
- Passes control through branch to application program

Exit Register Contents

- From exit to DLZDLA00:

R1	PST address
R13	Save area address
R14	Return address
R15	Entry point of call analyzer (obtained from SCD) at label SCDDLICT)

- From exit to ERRORMSG:

R1	PST address
R13	Save area address (PSTSV1)
R14	Return address
R15	Entry point of error message writer (obtained from SCD at label SCDERRMS)

- From exit to DLZABEND:

R15	entry point to DLZABEND

- From exit to application program:

R2 - R12	Restored to contents upon entry from application program to language interface module (DLZLI000)
R14	Application program return address

DLZABEND - STXIT ABEND

Abnormal terminations invoked through the DOS/VS STXIT or terminations requested by DL/I action modules are handled by DLZABEND. Responsibilities are as follows:

- Close the DL/I log.
- Issue an UNLD call to write the last records for Simple HSAM, HSAM, Simple HISAM and HISAM or write all buffers altered by the user. The UNLD call also closes the data base.
- If a dump is requested, write a formatted dump of DL/I control blocks.
- Cancel the partition.

Entry Interfaces

- Receives control through DOS/VS STXIT PC interface or STXIT AB interface
- Receives control through branch from program request handler (DLZPRHB0)
- Receives control through branch from DL/I action modules (including a special entry from the buffer handler)

Exit Interfaces

- Passes control through branch to data base logger (DLZRDBL0)
- Passes control through branch to call analyzer (DLZDLA00)
- Passes control through SVC 6 (CANCEL) or SVC 2 ($$BJDUMP) to DOS/VS

Exit Register Contents

- From exit to DLZRDBL0:

R1	PST address
R13	Save area address (PSTSV1)
R14	Return address
R15	Entry point of logger force write routine (obtained from SCD at label SCDDBLEW) or, Entry point of logger close routine (obtained from SCD at label SCDDBLCL)

- From exit to DLZDLA00:

R1	PST address
R13	Save area address
R14	Return address
R15	Entry address of call analyzer (obtained from SCD at label SCDDLICT)

DLZIWAIT - DL/I IWAIT

This module receives control when a DL/I action module requires DOS/VS wait linkage.

Entry Interface

Receives control through BALR from a DL/I action module

Entry Register Contents:

R2	Address of event control block
R14	Return address of caller
R15	Entry point of DLZIWAIT

Exit Interface

- Passes control through SVC 7 (WAIT) to DOS/VS.
- Passes control through branch on register 14 to the calling program.

DLZSTRB0 - BATCH FIELD LEVEL DESCRIPTOR (FLD) STORAGE MANAGER

This module frees the current field level descriptor storage, increases storage requirements for FLD by 128 bytes, and acquires the storage for the new FLD entries.

Interface

This module interfaces with the following module:

DLZDLA00 - Call analyzer

Control Blocks DLZSTRB0

- PPST - PST prefix
- PST - Partial specification table
- SCD - System contents directory

Normal Entry Point

The only entry point to this module is DLZSTRBO

Register Contents on Entry

R1 - PST address
R13 - Current register savearea address

DLZSTROO - ONLINE FIELD LEVEL DESCRIPTOR (FLD) STORAGE MANAGER

This module frees the current field level descriptor storage, increases storage requirements for FLD by 128 bytes, and acquires the storage for the new FLD entries.

Interface

This module interfaces with the following modules:

CLZDLA00 - Call analyzer

Control Blocks - DLZSTRO0

- CSA
- TCA
- PPST - PST prefix
- PST - Partial specification table
- SCD - System contents directory

Normal Entry Point

The normal entry point to this module is DLZSTRO0.

Register Contents on Entry

R1 - PST address
R13 - Current register savearea address

ONLINE DL/I PROCESSOR MODULES

Before attempting to use the information concerning DL/I processor modules, you should be familiar with the Customer Information Control System/Virtual Storage (CICS/VS). References to the prerequisite publications are contained in the preface to this manual.

The online DL/I processor modules DLZOLI00 and DLZODP provide services in a CICS/VS-DL/I environment as follows:

a. DL/I system initialization
b. DL/I user task scheduling
c. Processing DL/I calls (online program request handler)
d. DL/I user task completion
e. DL/I normal system termination
f. DL/I abnormal system termination
g. DL/I online message writer
h. DL/I-VSAM-CICS synchronization via VSAM 'EXCP' Exit.

DLZOLI00 - ONLINE INITIALIZATION

In order to process DL/I applications in an online environment, a DL/I online nucleus must first be generated. The DL/I online nucleus generation procedure is described in DL/I DOS/VS Resource Definition and Utilities. The result of the procedure described in the publication is a DL/I online nucleus CSECT.

The generated nucleus, which is link-edited into a DOS/VS core image library, consists of a system contents directory (SCD), a table of partition specifications table prefixes (PPST), a PSB directory entry for each PSB specified, a remote PSB directory entry for each remote PSB specified, and an application control table (ACT).

The application control table (ACT) is used by DL/I online at CICS initialization to verify and load all PSBs and DMBs that can be referenced online. The ACT is used during scheduling to determine whether an online transaction is to use DL/I. It is also used by DL/I default scheduling to acquire a PSB to use with a DL/I application program if none was explicitly specified.

The ACT is produced from parameters specified in the following DLZACT macro instructions:

```
DLZACT TYPE=INITIAL
DLZACT TYPE=CONFIG
DLZACT TYPE=PROGRAM
DLZACT TYPE=RPSB
DLZACT TYPE=BUFFER
DLZACT TYPE=FINAL
```

Each ACT program entry is generated from the DLZACT TYPE=PROGRAM statement. These statements define to DL/I which application programs can use DL/I online. They also define which PSB names can be used by each of the application programs. There is one ACT program entry for each DLZACT TYPE=PROGRAM statement used to generate the online nucleus. See the format of the application control table (ACT) in Figure 3-1.

A.	B.	C.	
4	8	2	bytes

A. Buffer pool information address or 0
B. Storage layout control table name or 0
C. Number of HD DBDs in HDBFR operand

Generated from: DLZACT TYPE = PROGRAM

Program entry '1'

D.	E.	F.	G.	• • •	G.	
8	1	2	2		2	bytes

D. ACTNM	ACT program entry name
E. ACTIND	Entry indicator byte: X'80' Program is a DL/I program X'40' Program name not in CICS PPT X'30' ABEND option bit X'02' Program is deferred-scheduled
F. ACTPCNT	Count of PDIR (PSB) pointers for this program
G. ACTPPTR	PDIR pointer(s). ACTPCNT indicates how many pointer are included here before the start of the next ACT entry.

Program entry 'n'

A maximum of 4095 DLZACT TYPE = PROGRAM statements and a maximum of 4095 unique entries (an entry consisting of program name and one PSBNAME) may occur in one ACT generation.

H.	
4	bytes

H. Delimiter (FF FF FF FF) indicating end of program entries

Generated from: DLZACT TYPE = BUFFER

HDBFR entry (subpool '1')

I.	J.	• • •	J.	K.	
2	8		8	2	bytes

I. Length of entry
J. DBD name
K. Number of buffers

HDBFR entry (subpool 'n')

HSBFR entry (DBD #1)

L.	M.	N.	O.	P.	
2	8	2	2	2	bytes

L. FF 00
M. DBD name
N. Number of index buffers
O. Number of KSDS buffers
P. Number of ESDS buffers

HSBFR entry (DBD #n)

Q.	
4	bytes

Q. Delimiter (FF FF FF FF)

Figure 3-1. Application Control Table (ACT) Format

DL/I initialization is performed during CICS/VS initialization just after loading the CICS/VS nucleus. The DL/I online nucleus module has been loaded by CICS/VS in the same manner as a CICS/VS nucleus module, and its address is placed in the CICS/VS CSA optional features list.

Nucleus and Table Initialization

DL/I verifies the presence of the online nucleus by checking the CICS/VS optional features list DL/I entry for a non-zero value. Once verified, the program request handler entry point is moved to the COMREG using the MVCOM macro. Next, the application control table (ACT) is located and an indicator is set in each corresponding PPT entry for all application programs which will use DL/I. Each PSB name in the ACT is eight characters in length. Each PSB name is padded with @'s, if required, to make it seven characters long, and a P to make it eight characters long.

Next the PSB segment intent list is built. This is accomplished by loading each PSB defined in the ACT, except those defined as remote PSBs, in ascending address space in the low end of the partition and moving the intent list, which is appended to the front of the PSB, to an entry in the PSB segment intent list table. The length of the PSB plus the length of the index work area, if required, are used to calculate how much storage to reserve. The segment intent list is overlaid during this process because its information is redundant. The PSB directory entry for each PSB is initialized with the address of the intent list, the PSB's storage address, and the amount of storage required.

The DMB directory is constructed. One DMB directory entry is created for each unique data base (DMB) defined in the PSB intent list entries. DMB names are eight characters in length and consist of the DBD generation name extended to seven characters by at-signs (@) if necessary. The eighth character is D. At this time, a validity check is performed to ensure that all required DMBs, defined by the PSB intent list, have been defined in the CICS/VS file control table (FCT). If any are missing, a message is written on the system console and the operator is given the option to continue or cancel. If initialization is to continue, PSBs which require the omitted DMB(s) are flagged to indicate this condition. Application programs which use these PSBs are not scheduled.

Initialization continues with the loading of all DMBs specified in the DMB directory. As each DMB is loaded, the corresponding entry in the DMB directory is initialized. A test is then made for HDAM and the defined randomizing routine is loaded. As the DMBs are loaded, they are initialized. After all DMBs have been loaded and initialized, the size of the buffer pool is determined. The size of the pool is based on a user-supplied parameter which defines the number of subpools, the control interval size of each VSAM data set, and the HDBFR subparameter, which tells how many buffers will be in a subpool.

After the pool size is determined, the required address space is reserved. Then the buffer pool prefix in the online nucleus is initialized. Next the subpool prefixes are created and initialized. There are 2-32 prefixes for each subpool.

Load Action Modules

Upon completion of initialization of the buffer pool and prefixes, the DL/I action modules are loaded. As the modules are loaded, their corresponding entry points are moved to the SCD. The modules are loaded in the following standard sequence if not otherwise specified by a storage layout control table:

DLZDBH00	-	Buffer handler
DLZDLR00	-	Retrieve
DLZDLA00	-	Call analyzer
DLZRDBL0	-	Data base logger
DLZDLD00	-	Delete/Replace
DLZDDLE0	-	Load/Insert
DLZDHDS0	-	Space management
DLZDXMT0	-	Index maintenance
DLZDLOC0	-	Open/Close
DLZQUEF0	-	Program Isolation ENQ/DEQ module
DLZQUEFW	-	Program Isolation ENQ/DEQ work area
DLZCPY10	-	Field Level Sensitivity Copy

Initialize PSBs

Upon completion of the loading of the action modules, initialization moves the specified PSBs using information stored in the PSB directory entries. After each PSB is moved, it is initialized and its corresponding PSB directory entry filled in.

Attach Logger

If data base logging has been specified by the user, the logger I/O module is initialized and attached. If the log module fails to attach, the data base log is closed and no logging takes place.

Open Data Bases

The final step of initialization is the opening of the data bases. The DMB directory is scanned for DMB's that failed during initialization and the open initial attribute is reset for any found. Next the data bases are opened via an 'open all' call to the DL/I Open/Close module. All modules indicating open initial in the DDIR are opened by Open/Close at this time.

Upon completion of the open processing, the IWAIT routine address is restored and control is returned to CICS initialization.

DLZODP

Task Prescheduling

DL/I task prescheduling is initiated when a task receives control on a Transfer Control (XCTL). The CICS/VS Program Control Program (PCP) examines the DL/I user bit in the CICS/VS PPT entry. If the bit is set and the task is not already scheduled, CICS/VS branches to DL/I prescheduling routine, DLZODP00. An indicator is set in the CICS/VS task control area (TCA) and control is returned to the CICS/VS PCP.

PSB Scheduling

A DL/I call or HLPI SCHEDULE command initiates PSB scheduling. The call function code is 'PCB' and the call contains the name of the PSB to be executed. The call is passed to the online program request handler via a language interface module and a scheduling validity check is made. If the call is valid, the parameter list is checked for a User Interface Block (UIB) pointer parameter. If specified, a UIB will be used for returning return code and PCB address list information to the application program. Upon completion, control is returned to the application program through the program request handler and the language interface. If the call is invalid, a two byte error return code is stored in the UIB or CICS/VS TCA and control is returned directly to the application program. For an HLPI command, the task abnormally terminates with a DLZ037I message indicating why the PSB was not scheduled if the call could not be completed.

If the 'PCB' call is made to schedule the system interface, the password is tested against the user generated one in the nucleus and the interface is tested for availability. A PST and dummy DSG are acquired for the caller, the task is marked as a system task, and control is returned to the user.

The caller provides the name of the PSB to be scheduled or optionally if the caller omits the PSB name in the call list, the first PSB name in this program's ACT entry is provided as default.

Task Scheduling

This subroutine determines whether DL/I can support another task and creates an entry in the PST prefix area for this task.

The SCD maximum task indicator is tested. If it is on, the task cannot be scheduled, the SCD suspended task counter is incremented by one, and an indicator is turned on in the SCD. A CICS/VS SUSPEND macro is issued to suspend this task.

If the SCD maximum task indicator is off, an available PST prefix entry is located and initialized for this task. The DL/I task accumulator is incremented by one and a test is made to determine whether the number of DL/I tasks now equals the maximum allowed. If yes, the SCD maximum task indicator is set. Next the SCD current maximum task indicator is tested. If on, the task cannot be scheduled immediately, and the subroutine issues a CICS/VS SUSPEND macro to suspend the task. The SCD current maximum task indicator is set if the scheduling of the task causes the current maximum task value to be reached. Control is passed to the task scheduling subroutine. If a remote PSB is to be scheduled, control is passed to the remote scheduling subroutine which transfers the request to the remote system.

PST storage is acquired from CICS/VS Storage Management and the storage address is saved in the assigned PST prefix. Task Scheduling consists of formatting the save area chains and storing the address of the assigned PST prefix. Control is passed to the PSB scheduling routine, DLZCOM00.

Local PSB Scheduling

This subroutine determines the segment intent of the PSB being scheduled and ensures that no more than one task is scheduled to update the same segment type(s) in the same data base unless program isolation is active. For retrieve sensitive only PSBs or update sensitive PSBs with program isolation active, a duplicate PSB is created if a prior task was scheduled with the same PSB. If the task cannot be scheduled, a CICS/VS SUSPEND is issued to suspend the task. If not in use, but retrieve sensitive only, the in-use indicator is set and controlis passed to PSB initialization. If neither of the above is true, the PSB segment intent list entry must be scanned. If program isolation is not active and the PSB is not retrieve only sensitive, the PSB segment intent list entry must be scanned.

The segment intent list for this PSB is located from the PSB directory entry. This list defines all segments in the data base(s) used by this PSB and also defines the PSB's sensitivity to them. The segment intent list entry is compared to the segment intent list entries of all scheduled PSBs. If no intent conflict is detected, the PSB initialization subroutine is called. Otherwise a CICS/VS SUSPEND is issued for the task. Upon completion of a successful segment intent scan, the PSB initialization subroutine is called.

If it is necessary to provide duplicate copy(s) of PSBs, this routine acquires storage for the copy and moves the original copy to it. Addresses in the duplicate are adjusted correspondingly and a duplicate PSB directory entry is created. The level table(s) are then reset and control passed to the PSB initialization subroutine.

PSB Initialization

PSB initialization consists of inserting the SDBs in the PSB into the SDB chain. The PSB is located from its PSB directory entry, and the address of the PCB address list is stored in the CICS TCA. Each PCB is located and the JCB pointer is used to obtain the address of the start of the SDBs for that PCB (JCBSDB1). Each JCB is accessed and the SDB chain pointers in the SDB and the PSDB in the DMB are updated. This process continues for all SDBs defined in the PSB.

The address of the assigned PST is obtained from the PST prefix and stored in the PSB. Using this address, the PSB directory entry address is stored in the PST. The "DL/I is scheduled" indicator in the PST prefix is set. If the PSB indicates the user is update sensitive, a call is made to the DL/I data base logger module (DLZRDBL0) or CICS journal interface routine (DLZDRBL1) to write an application program scheduling record (X'08'). Control is then returned to the calling routine.

Remote PSB Scheduling

This routine builds a scheduling call parameter list and passes it to the CICS/VS ISC interface routine, DFHISP. The call format is again transformed and routed by CICS/VS to the remote system that was defined in the corresponding DL/I online nucleus RPSB definition. The scheduling call is then executed on the remote system by a CICS/VS mirror program, DFHMIR. The results of the scheduling call is returned to the local system by CICS/VS. If the scheduling call was successful, CICS/VS also returns the addresses of local copies of the PCBs acquired in the remote system.

DLZPRHO0 - ONLINE PROGRAM REQUEST HANDLER

DL/I online calls are made in the same format as batch calls except that CALLDLI is used instead of CALL for Assembler language. The user issues a call instruction, passing parameters in the call list, and provides a register save area address in register 13. Communication of the results of the call is also identical to the batch system. It should be noted that although the format of the call instruction for online is the same as in batch, storage used by DL/I to process the call (i.e., register save area, all data items in the call list, I/O area) must be acquired from CICS/VS dynamic storage due to the re-enterability requirements of application programs which run under CICS/VS.

DL/I HLPI commands are translated into calls to the DL/I EXEC interface routine DLZEIPO0. This routine builds standard DL/I calls from HLPI commands

Language Interface Module

Although the language interface is not part of module DLZODP, it is involved in call processing. The language interface module is link-edited to each application program via the call instruction. The module has two entry points; one for Assembler, COBOL, and RPG II; and the other for PL/I. The first function performed at either entry point is to save the user's registers. Then a language indicator is set, the entry point to the program request handler is acquired from the DOS/VS COMREG, and a branch is taken to the program request handler.

For HLPI, CICS/VS EXEC stubs, DFHECI for COBOL, and DFHPL1I for PL/I, are used instead of the DL/I language interface module.

Program Request Handler

This routine is responsible for communication to and from the DL/I action modules and the user. It establishes the necessary table addressability for the action modules, and formats and validity checks the call list. It also moves the requested data to the user's I/O area and returns control to the application program.

The program request handler validates the DL/I call parameters before passing the call on to the next routine. For scheduling calls, control is first given to the task scheduling subroutine and then to the common PSB scheduling routine, DLZCOM00. For data base calls, control is given to the common data base call subroutine, DLZCOM01. This subroutine routes local calls to the call analyzer and remote calls to the remote data base call subroutine, DLZISC01.

The DL/I action modules process the local calls and return control to the program request handler through the call analyzer. A test is made in the program request handler to determine whether a pseudo-ABEND condition exists. If it does, a CICS/VS task ABEND macro is issued with an ABEND code indicating the reason. If an ABEND is not required, a test is made to determine whether the call requires data to be moved back to the user. The data is moved to the user's I/O area if required. The user's registers saved by the language interface are restored and control passed back to the calling application program.

Processing of the system calls 'CMXT', 'STRT', 'STOP', 'TSTR', and 'TSTP' is accomplished in the program request handler code. If these functions are identified in the call list a direct branch is taken to the appropriate routine.

IWAIT Routine

The IWAIT routine is entered from the DL/I buffer handler (DLZDBH00) or from other modules whenever an I/O wait or resource enqueue wait must be issued. The following processing occurs:

- Registers 14 through 12 and 13 are saved.
- Registers 12 and 13 are initialized with the CICS/VS CSA and currently dispatched TCA.
- A CICS/VS WAIT to CICS/VS Task Control Management is issued.
- Upon return, registers 14 through 12 and 13 are restored.
- Return is to the calling module via register 14.

DLZODP01 - TASK TERMINATION

DL/I task termination is entered by the CICS/VS PCP when a user's task scheduled by DL/I returns through CICS/VS Program Management, issues a CICS/VS sync point, or when the application program issues a DL/I 'TERM' call. This routine is responsible for purging any buffers altered by this task, calling the data base logger to write the application program termination record (X'07'), releasing any system resources owned by this task, and resuming tasks which were marked as not scheduled.

Task Termination

Task termination first determines whether this task was scheduled to use a remote PSB. If it was, control is given to the remote termination call subroutine. This subroutine issues a CICS/VS sync point call which causes DL/I programs processing calls on behalf of the local application program to be terminated. Next, task termination determines whether this task was assigned a PST prefix. If not, this task must have been stall-purged by CICS/VS and was originally suspended by the task scheduling module. In this case the suspended count accumulator is decremented and the task's TCA removed from the DL/I suspended task chain. Control is then returned to CICS/VS Program Management. If the task terminates abnormally, its DL/I control blocks are dumped by DFHDC.

If this task was assigned a PST prefix, a test is made to determine whether the task was scheduled. If not, the task was stall-purged by CICS/VS. This means this task was suspended by a CICS/VS Storage Management attempt to acquire either PST or PSB storage. If it was due to PST storage acquisition, the assigned PST prefix is cleared and put back on the free chain and the system resource allocation routine is entered. If it was due to PSB storage acquisition, the PSB directory entry is cleared, PST storage is freed, and the PST prefix is inserted in the free chain. Control is then passed to the system resource allocation routine.

If the task was scheduled and active, normal task termination proceeds. First a DL/I internal 'TERM' call is issued to the call analyzer (DLZDLA00). This call causes the analyzer to reset the level table(s) in the PSB. If update sensitive, the buffer handler (DLZDBH00) is called to write out all buffers altered by this task. Next the PSB directory entry is tested for update sensitivity. If indicated, the data base logger (DLZRDBL0 or DLZRDBL1, if CICS journal is in use) is called to write the application program termination record (X'07'). If the task had update sensitivity, the PST prefixes are scanned and any waiting for scheduling because of segment intent conflict are 'RESUMED'.

Next the PSB directory entry is released. For update sensitivity PSBs, this involves resetting the "user scheduled" indicator. For retrieve only, a test is made to determine whether this was a duplicate PSB. If so, the storage acquired for the PSB is freed and the duplicate PSB directory entry is cleared. Control passes to the system resource allocation routine.

If the system call interface is active the DDIR entries for the terminating PSB are checked for the waiting for close indicator. If the indicator is on and the use count of the DMB is now zero, the system task is resumed.

System Resource Allocation

This routine is responsible for determining whether any tasks are waiting to be scheduled and, if so, for taking the proper action to cause them to be scheduled. First the DL/I suspended task counter is tested. If nonzero, the first task on the DL/I suspend chain is located and a CICS/VS RESUME macro is issued. The suspend chain is then updated by removing the task's TCA from it, the suspended task counter is decremented, and, if zero, the maximum task indicator is reset. Next the DL/I task counter is decremented. If the task count is less than the current maximum task value, the current maximum task indicator is reset and PST prefixes which were 'WAITING' due to this condition are 'POSTED' complete. Control is then returned to the CICS/VS PCP.

DLZODP02 - DL/I NORMAL SYSTEM TERMINATION

The following processing occurs prior to CICS/VS termination.

- DL/I system termination (DLZODP02) is entered from the DL/I linkage module DLZSTP00, as specified in the CICS/VS pre-termination processing list section of the program list table (PLT).
- The DL/I log DTF is located and a DOS/VS CLOSE is issued for the DL/I log.
- DL/I system termination is re-entered by CICS/VS System Termination Program.
- A DL/I CLOSE call is issued to the DL/I Open/Close module (DLZDLOC0) to close all data sets for all DMBs in the system.
- Return is made to the CICS/VS via the DL/I linkage module.

DLZODP03 - DL/I ABNORMAL SYSTEM TERMINATION

The DL/I abnormal system termination routine is entered from CICS/VS when the DL/I partition is to be terminated abnormally. The following processing occurs:

- The DL/I control blocks are dumped.
- Return is made to the calling CICS/VS program.

DLZODP04 - PSB SCHEDULING START-OF-TASK RECORD ROUTINE

This routine issues CICS/VS DFHJC macros to write a CICS/VS Start-of-Task record to the CICS journal.

This routine is entered from DLZCOM00 on successful completion of a PSB scheduling call for a local data base.

This routine is not called if a PSB with read-only intent is scheduled. If a CICS/VS Start-of-Task record was previously written for the current CICS/VS logical unit of work, this routine returns control to its caller without writing the Start-of-Task record.

DLZODP05 - TASK TERMINATION SYNC POINT ROUTINE

This routine issues a CICS/VS DFHSP macro to force a CICS/VS sync point to be taken when a DL/I PSB termination or DL/I checkpoint call is being processed. For TERM calls, this routine is entered from the DL/I Task Termination Routine, DLZODP01. For CHKP calls, it is entered from DL/I Online Common Data Base Routine, DLZCOM01.

The sync point macro is not issued when DLZODP01 and subsequently, DLZODP05, is entered from the CICS/VS sync point program, DFHSPP, while processing a CICS/VS sync point.

DLZODP06 - ABNORMAL TASK TERMINATION DUMP ENTRY

This routine is entered from DFHPCP on abnormal task termination before dynamic transaction backout is performed by CICS/VS. This routine determines whether a DL/I formatted or DOS/VS IDUMP should be taken and gives control to the appropriate dump routine.

DLZODP07 - ABNORMAL TASK TERMINATION I/O CHECK ENTRY

This routine is entered from DFHPCP on abnormal task termination before SETEXIT check is made. This routine checks for and cancels any DL/I I/O requests that had not completed when the task was terminated.

DLZODP10 - COMMON GET STORAGE ROUTINE FOR DL/I ONLINE MODULES

This routine gets storage for CICS/VS (up to the maximum GETMAIN size) or DOS/VSE (for requests beyond the maximum CICS/VS GETMAIN size) on behalf of various DL/I online routines. This routine adjusts the requested storage size and address to allow for the CICS/VS Storage Accounting Area and its own storage accounting area.

DLZODP11 - DL/I ONLINE COMMON FREE STORAGE ROUTINE

This routine returns storage obtained by using DLZODP10.

DLZERMSG - DL/I ONLINE MESSAGE WRITER

The following processing occurs:

- The DL/I error code is extracted from the active PST or from a parameter list pointed to by register 1.
- CICS/VS storage is acquired.
- The appropriate DL/I message is created and logged to the destination CSMT via CICS/VS Transient Data Management and to the operator's console.
- Return is made to the calling routine.

If CICS/VS storage cannot be acquired or an error occurs while writing to transient data, an indicator is placed in the TCA and return is made to the calling routine.

DLZOVSEX - VSAM EXCP EXIT PROCESSOR

The EXCP exit processor receives control directly from VSAM after each SVC 0 resulting from a GET or PUT call from the buffer handler. DL/I checks the ECB for completion of the I/O request. If the request is incomplete the CICS/VS environment is re-established and a CICS/VS task control wait is issued in behalf of the current task. If the ECB was previously posted or the event completion has caused the task to be removed from the wait condition, control is returned directly to VSAM via register 14.

DLZFSDP0 - DL/I FORMATTED SYSTEM DUMP PROGRAM

The batch and online nucleus programs use this module to dump DL/I control blocks.

Entry Interface - DLZFSDP0

This module interfaces with DLBNUC0 in batch and DLZODP02 in online.

Exit Interface

This module returns control to caller.

Register contents on Entry

R1 - SCD address
R13 - Save area address
R14 - Caller return address
R15 - Module entry point address

Control blocks

ACB	PDCA
ACT	PDIR
BFFR	PPST
BFPL	PST
DDIR	RIB
DIB	RPCB
DMB	RPDIR
EIPL	SBIF
FERT	SCD
FSB	SDIB
PCB	UIB

DLZFTDP0 - DL/I FORMATTED TASK DUMP PROGRAM

This module formats DL/I task control blocks and writes them to CICS/VS dump data sets whenever this module is linkedited with the online nucleus and an application program scheduled to a DL/I data base ABEND.

If the DL/I system terminates abnormally without the CICS/VS system abnormally terminating, this module executes for each DL/I task active at DL/I ABEND.

Entry Interface - DLZFTDP0

This module interfaces with DLZODP06.

Exit Interface

This module returns control to DLZODP06.

Register Contents on Entry

- R6 - System TCA address
- R12 - User TCA address
- R13 - CSA address
- R14 - Caller return address
- R15 - Module entry point address

Control Blocks

ACB	PPST
ACT	PSIL
BFFR	PST
BFPL	RIB
CSA	RPCB
DDIR	RPDIR
DIB	RPST
DMB	SBIF
FERT	SCB
FSB	SDIB
PCB	TCA
PDCA	UIB
PDIR	

DLZDLA00 - CALL ANALYZER

The call analyzer module is used for initiation of all data base calls. Under normal circumstances, it receives control from the DL/I common data base call routine (DLZCOM01) in the CICS-DL/I region or from the batch application program request handler (DLZPRHB0). It receives control from application program control (DLZPCC00) at termination of a DL/I batch partition or online task termination (DLZODP01) in a CICS-DL/I region.

For internal DL/I calls to update an index data base, this module (DLZDLA00) receives control from the index maintenance module (DLZDXMT0).

The call types handled by the call analyzer module can be divided into two groups: (1) normal data base calls, and (2) special control calls, which are sometimes referred to as 'pseudo' calls. The special calls are GSCD, get SCD address; TERM, write all buffers altered by that user; and UNLD, write last records for simple HSAM, HSAM, simple HISAM, and HISAM load or write all HDAM and HIDAM data base buffers altered by that user and close all data sets in the system. In the online environment, GSCD calls are processed by DLZCOM01 and passed to the call analyzer module.

The primary responsibilities of the call analyzer are:

- Test the first parameter in the call list for a valid four-character function and encode this into a one-byte function code.
- Test the second parameter in the call list for a valid PCB address and store the PCB address in the PST.
- Store the third parameter in the call list in the PST. This is the user's I/O area address.
- Verify the format of all segment search arguments (SSAs) in the call list and fill in the corresponding level table entry for the SSA in the call.
- Do required checking based on call type and SSAs.
- Test for field level sensitivity when processing SSAs and set on bit if present. Call DLZCPY10 to map user's view to physical view if necessary.
- Do sequence checking when loading a data base.
- Pass control to the proper action module to process the call.

If a data base call requires the VSAM control blocks or SAM DTF representing the files within a data base to be opened, the analyzer calls upon the DL/I open/close module (DLZDLOC0) to perform the data management open for all files which may be needed for that PCB. The DL/I open/close module is called when the UNLD call is received to close all DL/I data bases opened in the batch partition.

During normal processing of the SSA, when an SDB has been located for the segment, a test of the SDB will be made to determine if field level sensitivity has been specified (bit SDBFSB set on in field SDBXFL). If it has, an indicator will be set in the JCB, signifying that at least one segment has field level sensitivity (bit JCBFLS set on in field JCBLVT).

When processing a qualified SSA, a check is made to determine if field level sensitivity has been specified for the segment. If it has, the FSB chain is scanned to see if the field name exists. If the field name does not exist or if the FSB is not flagged as an allowable field, a return code of 'AK' (invalid field name in call) is stored in the PCB and return is made to the caller.

If the field name is found and it is an allowable field, then qualification is set in the level table based on information in the FSB (qualification on data or key).

When the Call Analyzer determines that at least one segment has field level sensitivity, it will no longer do the processing to determine the offset of the segment in the user's I/O area (entry in LEVUSEOF will not be initialized by the Call Analyzer).

Prior to calling the insert, replace, or retrieve (only if called on behalf of insert) action modules, if the field level sensitivity indicator has been set in the JCB, the Call Analyzer will exit to DLZCPY10 to map the user's view to the physical view. At this point,the field level sensitivity indicator in the JCB will be reset. Any error passback from DLZCPY10 will be detected and exit will be taken to the Program Request Handler.

The field level sensitivity indicator will also be reset if an error is detected while processing the SSAs.

<u>Control Blocks - DLZDLA00</u>

PST
PDIR
PSB
DDIR
DMB
PCB
JCB
Level table
SDB
FDB
FSB

<u>Register Contents</u>

R1 = PST address
R13 = Save area address
R14 = Return address
R15 = Entry point address

<u>Interfaces - DLZDLA00</u>

Receives control from DLZPCC00, DLZODP00, and DLZPRHB0.

Passes control to DLZDLR00, DLZDLD00, DLZDDLE0 (DL/I action modules):

These modules need not save the analyzer's registers. They can return to the analyzer's entry point plus an offset stored in the SCD.

Call to DLZDLOC0 - DL/I open/close:

PSTFNCTN has open function
PSTDBPCB has address of the PCB

Call to DLZDBH00 - buffer handler:

PSTFNCTN is PSTPGUSR (X'07')

Call to DLZCPY10 - field level sensitivity copy

DLZDLOC0 - OPEN/CLOSE MODULE

The function of module DLZDLOC0 is to open and close the DL/I data bases in either the CICS online control region or the batch partition. DOS/VS open/close macros are used to open and close data sets. DLZDLOC0 opens/closes VSAM ACBs for all data base organizations besides HSAM and simple HSAM, where DTFs are used. For simplicity the term ACB is used in the following description where ACB or DTF would be correct. For a HISAM data base with all functions, except for PSTOCDCB, both the KSDS and ESDS are opened/closed.

The PSTFNCTN byte in the PST determines the type of operation to be performed by DLZDLOC0.

- PSTOCDCB (X'10') - Only one ACB is opened/closed. It is located by DSG address (PSTDSGA).
- PSTOCPCB (X'02') - For PROCOPT = L or LS one data base is opened.

  For PROCOPT ≠ L or LS:

  All SDBs of that PCB are scanned and all referenced data bases are opened, that is, index data bases and logically related data bases are opened/closed with this call.
- PSTOCDSG (X'40') - One or two (HISAM) data bases are opened/closed.

  The ACB is located by DSG address (PSTDSGA).
- PSTOCALL (X'04')
  - For open:

    All ACBs specified for initial opening are opened (CICS online control region only)
  - For close:

    All ACBs in the system are closed.
- PSTOCDMB (X'01') - The ACBs of one DMB are opened/closed. The DMB directory address is passed in register 2.

DLZDLOC0 compares the following values specified in DBD generation with the VSAM catalog entries for a data base:

- Control interval size
- Key length (KSDS)

- Relative key position (KSDS)
- Highest RBA used in the data base based on the PROCOPT. For example, PROCOPT=L requires an empty data base (high RBA=0), while a data base must contain data if PROCOPT≠L (high RBA>0).

For HISAM, HIDAM, and HDAM data bases, the first control interval of the VSAM ESDS is reserved for the DL/I control record. DLZDLOC0 maintains this record.

- If PROCOPT=L or LS, space is acquired for one control interval and the DL/I control record is constructed. The buffer handler (DLZDBH00) is called to write the DL/I control record.

An open record, code X'2F', is written to the log file whenever a data base is opened. If the open call is successful, bit zero (JCBOPEN) of the JCBORGN byte equals one (PCB call); and bit zero (PSTOCBAD) of the PSTFNCTN byte equals zero.

All PSDBs of a DMB are scanned for variable length segments with the edit/compression routine. All edit/compression routines that have 'INIT' specified are called after "open" and before "close".

Register Contents

R1	-	PST address
R2	-	DDIR address if it is a close DMB call
R13	-	Save area address
R14	-	Return address
R15	-	Entry point address

Control Blocks - DLZDLOC0

- DL/I control record - DLZREC0
- PSTFNCTN field of the PST:

Bit	Value	Meaning
1	1	Process DSG
2	1	Open for load
3	1	Process specific ACB
4	0	Close call
	1	Open call
5	1	Open/close all DMBs
6	1	Open/close a PCB
7	1	Open/close a DMB

DLZDLD00 - DELETE/REPLACE

This module performs the logical actions involved in replacing or deleting segments in a DL/I data base for all organizations, except HSAM (which has no delete or replace).

The replace function checks to ensure that the key field of the segment was not inadvertently altered and that the replace rules were not violated. If the segment to be replaced is indexed, this module interfaces with the DL/I index maintenance module (DLZDXMT0).

The first check made upon entry is a key check of the contents of the PCB key feedback area to the key of the segment in the user's I/O area. If there are any changes, a 'DA' status code results. Next the segment is retrieved and the sequence fields are checked for any changes. If any changes occurred, a 'DA' status code again results. Then the remainder of the data is checked for changes. If there were no changes, a blank status code is returned. If there were changes, the data is replaced.

If the segment to be replaced is in an HDAM or HIDAM data base and the segment is variable length, the segment and its prefix may be separated. The separation of data is determined by the min-byte value of DBDGEN and the current size of the segment. Also in this regard, if the segment was previously separated from its prefix prior to a replace call, the replace will attempt to rejoin data and prefix.

The delete function for a HISAM data base reads the segment to be deleted. If the organization is simple HISAM, the buffer handler is called to issue a VSAM ERASE. Otherwise, the segment is deleted by setting the HISAM segment delete bit. In addition, if this is the root segment, the record delete bit is also set.

The delete function for HDAM or HIDAM data bases includes a check to ensure that delete rules stated for the DMB will not be violated. If logically related segments with a physical delete rule exist in the data base within the physical hierarchy starting with the segment to be deleted, a scan is made of all the segments to ensure that they include no segment which has not been logically deleted.

A scan of the data base from the point of deletion is performed. During this scan, each segment is accessed twice: once on the way 'down', and again on the way 'up'. While scanning 'down', any segment in a logical relationship is inspected to determine its eligibility for deletion and to terminate as many logical relationships as possible. In some cases (for example, the last logical child for a logical parent which has already been deleted through its physical path), the deletion of all, or a portion of, the logically related data base record is required. In this case, the delete action is expanded to perform the total delete function (except for the checking) for the new data base record. Then the scan of the original data base record is continued at the point of exit.

When scanning 'up', an interface with index maintenance (DLZDXMT0) is made if the segment is indexed. Physical pointers are adjusted to bypass any removable segments (HDAM or HIDAM segments which are no longer required) whose space is released by interfacing with the space management module, DLZDHDS0. For nonremovable segments (segments required to remain because of existing logical relationships), a logical delete bit is set to indicate the status of the segment.

A work area is obtained from the DL/I buffer pool to maintain the concatenated key and position of segments in the data base record(s) being scanned during delete or for calls to index maintenance during replace.

Delete/Replace Work Space Acquisition and the Work Space Prefix

DLZDLD00 acquires space to build work area(s) from DLZDBH00 (buffer handler) via a PSTGBSPC call. The calculated minimum size required is indicated in PSTBYTNM. If the space is available, the buffer handler returns the address of the selected buffer in PSTDATA and its size in PSTWRK1.

The first section of the work space contains a prefix whose format and contents are described in Section 5. Immediately following is the work area containing information concerning the segment to be deleted (or the index source segment to be replaced), its physical data base (HIDAM or HDAM), and other segments in that data base record.

If a second work area is needed because of logically related segments and the space remaining in the current work space is large enough, the next work area will be allocated in the same work space (buffer) immediately following the previous work area. Forward and backward chains are maintained. If the remaining space is not large enough, another buffer is obtained from the buffer handler and chained to and from the previous work space.

Except in the case of an error condition, work areas are freed in the reverse order in which they were allocated. When the work area freed was the first one in the work space, the buffer is freed via a PSTFBSPC call to the buffer handler.

Segment Delete Codes

Segment delete codes utilized in the second byte of the prefix of each DL/I segment:

1...	This segment has been deleted (HISAM only).
.1..	This data base record has been deleted (HISAM only).
..1.	This segment has been processed by delete.
...1	This variable-length segment has its data separated from the prefix.
.... x...	Reserved
.... .1..	This segment is no longer required by its physical parent.
.... ..1.	This segment is no longer required by its logical parent.
....1	This segment has been removed from its logical twin chain.
1111 1111	This segment contains the separated data of a variable-length segment.

Interfaces - DLZDLD00

This module interfaces with the following modules:

DLZDBH00
DLZDHDS0
DLZRDBI0
DLZDXMT0
DLZQUEF0

Control Blocks - DLZDLD00

- Delete workspace prefix
- Delete work area.

Register Contents at Entry

R1 Contains the address of the PST
R13 Points to the current save area
R14 Contains the DL/I analyze call function module (DFSDLA00) return point
R15 Contains the module entry point

Register Contents at Exit

R1 Contains the PST address
R13 Points to the current save area
R14 Contains the DL/I analyze call function module (DFSDLA00) return point
R15 Contains a return code (0)

Register Contents on ABEND - in the SCD ABEND Save Area

R1 -	PST address
R2 -	SCD address
R3 -	SDB address
R4 -	DMB address
R5 -	PSDB address
R6/R10	Work registers
R11 -	Base - (subroutine CSECT)
R12 -	Base (main CSECT)
R13 -	Current save area
R14/R15	- Work registers

DLZDDLE0 - LOAD/INSERT MODULE

The function of DLZDDLE0 is to load HDAM, HIDAM, Simple HISAM, HISAM, Simple HSAM, and HSAM data bases (in batch only) and insert segments into HDAM, HIDAM, Simple HISAM, and HISAM data bases.

DLZDDLE0 is entered from the DL/I call analyzer (DLZDLA00) on load requests for HIDAM, Simple HISAM, HISAM, HSAM, and Simple HSAM segments, HDAM dependent segments, and insert requests for Simple HISAM and HISAM roots. It is also entered from the retrieve module (DLZDLR00) on load requests for HDAM root segments, and insert requests for HDAM, HIDAM, and HISAM dependent segments.

The module performs the following functions:

A. HDAM/HIDAM load/insert -

1. Normal segment:

- Positioning: retrieve positions for inserting and loading of HDAM roots. For all other loading, DLZDDLE0 simulates retrieve positioning.

- Space for new segment is acquired using the space management module, DLZDHDS0.

- The segment is moved from the user's I/O area to the buffer.

- Prefix pointers are updated.

- Actual write is performed by the buffer handler using VSAM.
- Prefix pointers of twins and parents are updated.
- The data base logger (DLZRDBL0) is called to write the new segment and the updated prefixes.
- If the segment is an index source segment, index maintenance (DLZDXMT0) is called.
- Exit is to the call analyzer.

2. Concatenated segment:

- If the destination parent already exists, and the insert rule is physical or logical: same as normal segment.
- If the destination parent exists and the insert rule is virtual: the logical child segment is inserted as for a normal segment, data of destination parent are replaced afterwards.
- If the destination parent does not exist and the rule is not physical, the destination parent is inserted as for a normal segment; afterwards the logical child is inserted as a normal segment.

B. HISAM and simple HISAM load

- Main storage for a logical record for key sequenced data set (KSDS) and for entry sequenced data set (ESDS) is acquired from the buffer handler.
- The root and all dependent segments that fit into one logical record are written to the KSDS, using the buffer handler. The remaining dependent segments are moved to one or more records of the ESDS.
- Pointers to those records are inserted.

C. HISAM and simple HISAM root insert

- A key equal to or greater than the request is made to the buffer handler. If the key exists and the delete bit is flagged (HISAM), the space is reused; otherwise a II status code is returned. If the key does not exist, main storage is acquired from the buffer handler and the new record is built and then inserted by VSAM through the buffer handler.
- Old (if deleted) and new records are logged.

D. HISAM dependent segment insert

- If the segment fits into the record for which retrieve (DLZDLR00) has positioned, it is inserted by shifting the segments beyond the insert point to the right. If the segment does not fit into the record, a new ESDS record is built. The segment and shifted data are inserted into the new record. If the shifted data does not fit into the record, a second new ESDS record is created.
- Pointers to the new records are created.
- Old and new records are logged.

E. HSAM and simple HSAM load

- The I/O areas allocated by batch initialization are used to move the segments from the user area. PUT locate is executed, whenever one I/O area is filled.

Blocks and Tables - DLZDDLE0

PST
DDIR
DMB
PCB
JCB
Level table
SDB
FDB
SCD

Registers on Entry and to All Called Modules

R1 = PST

Interfaces - DLZDDLE0

This module calls the following modules:

DLZRDBL0	- Data base logger
DLZDBH00	- Buffer handler
DLZDHDS0	- Space management
DLZDXMT0	- Index maintenance
DLZQUEF0	- Queuing Facility

Status Codes - DLZDDLE0

II
AO
IX
LB

DLZDXMT0 - INDEX MAINTENANCE

The function of this module is to load - insert - delete the index pointer segment of a HIDAM data base and to load - insert - delete - replace the index pointer segment for secondary indexes of a HDAM or HIDAM data base.

Abbreviations used throughout the module are:

ISS	Index source segment
XDS	Index target segment (indexed segment)
XNS	Index pointer segment (indexing segment)

The following major functions are performed:

ALL CALLS

- Save PST information in XMAINT work area

LOAD
INSERT

* Build index pointer segment in work area

  For primary indexes - take key from user I/O area. For secondary indexes - construct segment from SRCH, SUBSEQ and DDATA fields. For /CK fields use PCB-key feedback area or read parents of ISS using SDBPOSC or PP pointers. Call user suppression routine, if needed.

* Build temporary blocks SDB, JCB, DSG

INSERT

* Build call list and SSA
* Call analyzer
* Take next index relationship of this ISS

LOAD

* Open data base, if necessary, or work data set
* Call buffer handler to write index record or write work data set for secondary index
* Take next index relationship of this ISS

UNLD

* Write FF-key record to all index data bases belonging to this data base

DLET

* Call buffer handler to get old ISS
* Construct the old index pointer segment
* For /CK fields take CONCAT key from DLET work area
* Call user exit routine, to check for suppression
* Build temporary blocks
* Log POINTER CHANGE and DEL.BYTE CHANGE
* Call buffer handler to change index
* Take next index entry

REPL

* First part = DLET
* Second part = ISRT

ALL CALLS

* Restore PST
* Return to calling module

Entries:

Receives control from DLZDDLE0 (load/insert) and DLZDLD00 (delete/replace)

Register Contents

```
R1        =   PST address
R14       =   Return address
R15       =   Start address

PSTWRK1       LSDB of ISS for ISRT, ASTR, REPL calls
                  LSDB of ROOT for UNLD CALL
                  PSDB of ISS for DLFT call
PSTFNCTN          'A0'    Delete
                  'A1'    Replace
                  'A2'    Insert
                  'A3'    Unload
PSTBYTNM          RBA of index source segment
```

Interface to called modules:

1. DLZDLA00 (analyzer)
   Called for insert, not load mode

   PSTIQPRM points to internal call list
   Segment name*X(keyvalue) is used as SSA

2. DLZDBH00 (buffer handler)
   ```
   PSTFNCTN:   PSTMSPUT load HIDAM index
               PSTBYLCT get index target segment again
               PSTSTLEQ get index pointer segment
               PSTPUTKY index of HIDAM data base
               PSTBFALT update index of HIDAM data base

   PSTBYTNM:   RBA of segment
       or      Pointer to key to be inserted
   ```

3. DLZDLOC0 (open/close)
   ```
   R2:      Address of DDIR
   PSTFNCTN:   PSTOCOPN + PSTOCLD + PSTOCDMB
               PSTOCOPN + PSTOCDMB
               PSTOCCLS + PSTOCDMB
   ```

4. DLZRDBL0 (logger)
   ```
   PSTWRK1:    DBLLGDLT             logical delete
               DBLNDXC + DBLCMC     XMAINT chain maintenance
   PSTWRK2:    Old segment code and old delete byte
               Old RBA pointer
   PSTOFFST:   Offset to new segment code
               Offset to new RBA pointer
   PSTBYTNM:   RBA of record
   ```

5. DLZDSEH0 (work data set module)

   Is called at entry point - 12 to open work file.
   Return is to BALR if open not successful,
   to BALR + 4 if open successful.

6. DLZQUEF0 (queueing facility)
   Called to do any program isolation queueing necessary

Exits:

Back to calling module.

Control Blocks - DLZDXMT0

- Index work area - DLZXMTWA
- SSA for the XMAINT call to the analyzer.

DLZDLR00 - RETRIEVE

The DL/I retrieve module is responsible for retrieval of all segments, independent of physical data base organization. When an application program requests the retrieval of a segment, this module (DLZDLR00) gains control from the DL/I call analyzer, DLZDLA00. The analyzer has validity-checked the parameters in the application program's retrieval request. The analyzer has also placed this parameter information for retrieval in the DL/I control blocks.

Based upon this information, the retrieve module calls the DL/I buffer handler module, DLZDBH00, which controls physical I/O operations, to read the block containing the desired segment. Once the desired block exists in the data base buffer pool, its presence is made known to the retrieve module.

It is the responsibility of the retrieve module to "deblock" segments within the block. Once the desired segment is located, the retrieve module places the location and length of the segment in the PST control block associated with the application making the retrieve request and returns to the DL/I call analyzer. Once a particular segment within a data base is retrieved for a particular application program, "position" is established within the data base for the application program. This "position" is subsequently used to move sequentially through the data base if the application program issues GN and GNP calls.

If the block containing the segment to be retrieved already exists in the data base buffer pool, the request from the retrieve module to the buffer handler results only in the address of the desired data being returned to the retrieve module. No physical I/O is performed. In the case of HISAM, if a retrieve request involves inspection of several segments within a record, the retrieve module requests only the first of these from the buffer handler and finds the remaining segments itself, utilizing position information. Positioning information for each application program and each data base is maintained in the DL/I control blocks which are an extension of the PCB (that is, JCB, LEVVTAB, and LSDB).

In addition to servicing all data base retrieval requests, the retrieve module performs "positioning" functions for all segment insertion. In this case, the retrieve module receives control from the DL/I call analyzer module on an insert call. Prior to the insertion of a new segment occurrence, DL/I must insure that the segment does not already exist in the data base. It is the responsibility of the retrieve module to retrieve the block where the segment to be inserted may already exist. If the segment does not already exist in the data base, the block retrieved is normally used for segment insertion. Once the desired physical block is retrieved and positioning for segment insertion within the block is established, control is passed to the DL/I load/insert module, DLZDDLE0. If the data base organization is Simple HSAM or HSAM, the retrieve module performs the I/O (Get/Put) rather than calling the buffer handler.

HIDAM root retrieval by key (qualified GU, GN), results in two buffer handling requests. The first retrieves the index segment as any HISAM root. The second uses the RBA of the HIDAM root in the index segment to get the corresponding root segment. The position of the index segment is saved in a special SDB.

Retrieval of segments addressed by secondary indexes is performed in the same manner, as far as possible, as the retrieval of a HIDAM primary root segment. (The SDBs are generated so that the index looks like a primary index and the index target segment like a HIDAM primary root.) The most important differences are:

- The layout of the index pointer segment is user dependent and is different from that of a primary index.
- The sequence field of a secondary index is not necessarily part of the target segment and may be in a dependent segment.

Variable length segments are handled by the routine VLRT which provides an exit to a user routine to handle any necessary data expansion after calling the normal buffer handler interface (SETL).

Retrieval of logically related segments requires special handling. The retrieved segment (the concatenated segment) consists of the logical child (that is the concatenated key and the intersection data) and the physical or logical parent (destination parent). Since the SDBs always reflect the user's view of the data base, the same program logic is used whether the segment to be concatenated to the logical child is a physical or a logical parent. The concatenated key of the destination parent is constructed using the physical or the logical parent pointer of the logical child and the physical parent pointer of the destination parent. For ISRT calls the concatenated key in front of the input data is used to position on the destination parent. All positions on the physical path to the destination parent and on the twin chain of the destination parent are maintained.

Command Codes Affecting Retrieval

D - The segment data is moved when the level table is updated and not at return to the analyzer.

L - The segment skip routine is employed to skip to the last occurrence.

T - The RBA specified in the SSA is moved to the next position pointer location in the appropriate SDB and an unqualified GN is performed.

F - For a GN (GNP) call, the same logic is employed to retrieve the first occurrence as for a GU call.

Module Layout - DLZDLR00

This module consists of 60 subroutines, a main entry routine (DLZDLR0), a main exit routine (DLZDLR1), and a general linkage and maintenance support routine (DLZKLNKD), each of which is preceded by a description in the form input - processing - output. The subroutines are linked using macro DLZRLNK and the following macros (refer to the comments in the DLZRLNK source program listing):

DLZRHDR - First macro of a subroutine; generates DSECTs, EQU, and module identification.

DLZRTLR - Last macro of a subroutine.

DLZRCLL - Generates code to transfer control to a subroutine using DLZRLNK.

DLZREXT - Generates code to return control to a calling subroutine using DLZRLNK.

The module is supplied as eight files. The first seven, DLZDLRA0 to DLZDLRG0, contain the subroutines and the eighth, DLZDLNKD, contains the linkage and maintenance support routine that is generated using the macro DLZRLNK. The second file, DLZDLRA0, also contains the routines DLZDLR0 and DLZDLR1. The distribution of the subroutines within the CSECTs contained in the files DLZDLRA0 to DLZDLRG0 is arbitrary and can be changed at will, necessitating only that the affected CSECTs be reassembled.

Maintenance Support - DLZDLR00

The module DLZRLNKD contains facilities to dynamically dump control blocks and I/O buffer sections. The extent and frequency of the dumping is controlled by DLZRLNK macro parameters or control fields in the PST as described in the DLZRLNK source program listing.

Interfaces - DLZDLR00

This module interfaces with the following modules:

DLZDDLE0 - Load/insert
DLZDBH00 - Buffer handler
DLZQUEF0 - Queuing Facility

Register Contents on Entry and Return

R0 = SCD
R1 = PST
R2 = PCB

Register Contents During Execution

R0 =	Work
R1 =	Work
R2 =	Work, PCB
R3 =	JCB
R4 =	LEVTAB
R5 =	SDB
R6 =	Segment address
R7 =	PST
R8 =	DSG part of JCB
R9 =	Byte or record location of SEGM in data base
R10=	Work, FLD
R11=	Base register for linkage routine DLZRLNKD
R12=	Base register
R13=	Save area
R14=	Work
R15=	Work

DLZDHDS0 - HD SPACE MANAGEMENT

Module DLZDHDS0 allocates and maintains free space on direct access storage devices for storage of DL/I segments in the hierarchical direct organizations (HDAM and HIDAM). This space is managed through the use of free space elements (FSEs) in each block of each data set of a data base and a bit map. The bit map describes blocks that have at least one FSE which can contain the largest segment in the data set. There is one bit map per data set consisting of one or more blocks distributed equidistant over the data set.

Module DLZDHDS0 consists of CSECTs which perform the following functions:

DLZDHD00 contains the entry point for the combined module. It saves registers, initializes the work words in the PST, and branches to the appropriate module.

DLZGGSP0 consists of a 'driver' for all subfunctions that may be invoked to find space. It uses one byte of the work space to control invocation. This CSECT also controls formatting for HDAM when the root anchor point is beyond the current end of the data set and formatting of new bit map blocks, if necessary.

DLZFRSP0 returns to free space the space occupied by a segment being deleted. It logs the deletion of the segment and updates the bit map if required.

DLZRCHB0 searches the block passed to it for an FSE that satisfies the current request. If none is found, control returns to the calling module. If the request can be satisfied, the return is directly to the invoker of DLZDHDS0.

DLZRRHP0 searches the DL/I buffer pool for a block in the range passed to it. If one is found, module DLZRCHB0 is called to search it. If the block is rejected, the search continues to the end of the pool, and control is returned to DLZGGSP0. To avoid changing the position of buffers on the buffer pool use chain, online and batch are treated differently. In a batch environment, the buffer to be searched is passed to DLZRCHB0 and may be used without being requested from the buffer handler. In a DL/I online environment, the buffer is passed to DLZRCHB0. If the request can be satisfied from it, the buffer is then requested from DLZDBH00 and again passed to DLZRCHB0 for actual alteration.

DLZRRHM0 searches the bit map for a bit that is a one and is also in the specified range. If one is found, its corresponding block number is returned to DLZGTSP0. If all bits are zero, PSTNOSPC is returned to DLZGGSP0. The map search functions include creation and formatting of new bit map blocks, if necessary. To further proximity of space for related segments, whenever possible, the search within a given range is done from the center to the outer ends of that range in both directions at the same time.

DLZLMCL0 calculates search limits for DLZGGSP0. A switch is used to determine the appropriate limit - track, control area, delta control areas. The limits of the previous scan are used to break the range into two subranges. This prevents the re-requesting of blocks that were rejected during earlier scans.

DLZMPLC0	determines the block number for the bit map block appropriate to the block number passed to it. It also determines the relative bit position in the bit map block of the block number passed to it.
DLZMMUD0	turns the appropriate bit ON or OFF according to the entry point involved. The log is also called to reflect the change.
DLZDCI00	tests to see if the device containing the data base is actually an FBA device if it was specified as such, and, if it is, calculates the CIs per track and per cylinder and the scan value in cylinders equivalent to the number of FBA blocks specified during DBD generation. These values are stored in the DMB for later use.

Interfaces - DLZDHDS0

The following modules are called by DLZDHDS0:

DLZDBH00 - Buffer handler
DLZRDBL0 - Data base logger

Calling Sequence

```
R1       PST address
         PSTDSGA     DSG address for appropriate file (all calls)
         PSTFNCTN
                     PSTGTSPC    01  Get space
                     PSTFRSPC    02  Free space
                     PSTBTMPF    03  Turn off bit in bit map
                     PSTGTRAP    04  Get space close to root anchor
                                     point
         PSTRBN      RBN of segment to get space close to - PSTGTSPC
                     RBN of segment to be deleted - PSTFRSPC
                     BBBR - PSTGTRAP
                     where BBB = relative block number,
                     R = root anchor point number
         PSTBLKNM    Block number whose bit is to be turned off -
                     PSTBTMPF
R5       DMBPSDB     Address of PSDB of subject segment
R14      Return point
R15      Entry point - DLZDHDS0
```

On Return

```
PSTRTCDE - PSTCALOK  Space obtained; RBN is in PSTRBN
                       - PSTGTSPC, PSTGTRAP
                     Space freed - PSTFRSPC
         - PSTBTMPF  Space obtained.  After insert, call
                     DLZDHDS0 to adjust bit map.
R15      - 0         For above return codes.
         - 4         Error has occurred; check PSTRTCDE
PSTRTCDE - PSTGTDS   The RBN to get close to does not exist
         - PSTNOSPC  DLZDHDS0 could not find space in data
                     set -  PSTGTSPC, PSTGTRAP
         - PSTIOERR  See DLZDBH00
           PSTNPLSP  See DLZDBH00
```

DLZDBH00 - DB BUFFER HANDLER

The primary functions of module DLZDBH00 are:

1. To satisfy requests for buffer space for the processing of the data blocks of HD data bases. For Simple HISAM and HISAM data bases and for the index of HIDAM data bases, the VSAM buffer management is used.

2. To issue I/O requests to VSAM whenever data must be read or written. Thus, the buffer handler provides an interface between the DL/I action modules and VSAM data sets.

3. Whenever possible, to satisfy requests for data base segments and or records from data currently available in its buffer pool without issuing an I/O request. For this purpose, data is retained in the pool as long as possible. Various features such as use chains and alteration flags are employed so that a centralized buffer management is facilitated for concurrent use by all application programs.

The buffer handler satisfies the following requests as indicated by PSTFNCTN:

1. For processing HDAM, HIDAM, or HISAM ESDS:

Symbol Function	Hex Function	Description
PSTBYLCT	02	If the request is issued for an HDAM or HIDAM data base, the buffer handler retrieves the control interval whose relative byte number is stored in PSTBYTNM. The relative byte number in PSTBYTNM is first converted to a VSAM control interval number and an offset within the control interval. If this control interval is not in the buffer pool, buffer space is obtained in the buffer pool, the buffer which will be used is written, and the control interval is read into this buffer by a VSAM get call. If the requested control interval is already in the buffer pool, no read is done and the address of the buffer containing this control interval is passed back to the caller. If the request is issued for a HISAM ESDS data base, the buffer handler only issues the proper VSAM call for retrieving the record identified by the RBA which has been passed to the buffer handler in PSTBYTNM.
PSTBKLCT	01	The same as PSTBYLCT for an HDAM or HIDAM data base except that a VSAM control interval number is passed to the buffer handler in PSTBLKNM.

PSTBYALT	06	A locate relative byte number (refer to PSTBYLCT) is done first and then the buffer which contains the control interval is marked as altered by this specific user.
PSTBFALT	05	If the request has been issued for an HDAM or HIDAM data base, the buffer whose prefix address is stored in PSTBUFFA is marked altered. If, however, the request applies to a HISAM ESDS, the proper VSAM call is issued to write the record immediately.
PSTGBSPC	03	A buffer with the length specified in PSTBYTNM (possibly rounded to the next multiple of 512 bytes) is provided to the caller.
PSTFBSPC	04	A buffer identified by a DMB number, ACB number, and control interval number in PSTDMBNM, PSTACBNM, and PSTBLKNM is freed, that is, it is marked empty and put on the bottom of the use chain.
PSTPGUSR	07	All the buffers which have been modified by a specific user are written. All nonreusable buffers held by this user are marked empty and put to the bottom of the use chain. The bit representing this user is turned off in the user mask of all permanent write error blocks. If the purge request is on behalf of a CHKP function-call, all DMBs are scanned for index data bases and ENDREQs are issued to ensure that all VSAM buffers are written to the data bases.
PSTBFMPT	04	All buffers of one data base or certain buffers of a data base are marked empty and put on the bottom of the use chain.
PSTWRITE	08	A logical record is added to a HISAM ESDS.

2. For processing HIDAM index, Simple HISAM or HISAM KSDS:

(a) Accessed by VSAM RBA

Symbol Function	Hex Function	Description
PSTBYLCT	02	Retrieve the VSAM KSDS record by the RBA which is in PSTBYTNM.
PSTBFALT	05	Write the VSAM KSDS record by the RBA which is in PSTBYTNM.

PSTERASE	0A	Delete the VSAM KSDS record identified by the RBA which is in PSTBYTNM.

(b) Accessed by key

Symbol Function	Hex Function	Description
PSTSTLEQ	09	Retrieve the VSAM KSDS record whose key is equal to or greater than the key whose address is stored in PSTBYTNM.
PSTGETNX	0B	Retrieve the next sequential VSAM KSDS record.
PSTSTLBG	0C	Retrieve the first VSAM KSDS record in a data base.
PSTPUTKY	0D	Insert a record by key directly into a VSAM KSDS.
PSTMSPUT	0E	Insert a record which is in ascending key order into a VSAM KSDS.

The buffers which are used for satisfying these requests are provided by VSAM buffer management. The buffer handler provides VSAM control blocks (ACB, EXLST, and RPL) to VSAM data management when issuing the required VSAM action macro.

The module DLZDBH00 consists of three CSECTs:

DLZDBH00 Contains the code for the functions

- PSTBYLCT
- PSTBKLCT
- PSTBYALT
- PSTBFALT
- PSTGBSPC
- Maintenance of write chain and use chain

DLZDBH02 Contains the code for the functions

- PSTSTLEQ
- PSTGETNX
- PSTSTLBG
- PSTPUTKY
- PSTMSPUT
- PSTERASE
- PSTWRITE

Additionally, this CSECT contains the code required for preparing and issuing of VSAM calls and for processing feedback information by VSAM.

DLZDBH03 Contains code for the functions

- PSTFBSPC
- PSTBFMPT
- PSTPGUSR

In addition, this CSECT contains the subroutines for providing an enqueue/dequeue function.

Write Chain

The new control intervals of a HIDAM or HDAM data base are chained together on a write chain in ascending order of their control interval numbers. If one of the buffers on the write chain has to be written, all buffers on the chain are written.

There is a write chain for every data base. It is maintained by storing the prefix numbers of the prefixes of the next higher and the next lower buffers in bytes 18 and 19 of the prefix. A bit switch in byte 7 of the prefix (X'80') is on if a buffer is on a write chain.

Use Chain

All buffers are chained together in the order of their usage. This use chain is physically separated from the buffer prefixes and consists of one-byte elements containing relative numbers of prefixes. The order of the buffers on the use chain is indicated by the physical order of these use chain elements.

There is one use chain area per subpool. Each use chain area has a maximum of 32 entries. The maintenance of the use chain involves putting a use chain element on the bottom or on the top of the use chain as follows. The contents of the use chain element which is to be moved are saved. Then all use chain elements located behind the element to be put on top, or located before the element to be put on the bottom, are moved to the address which is one byte lower than the load address (or one byte higher if an element is placed at the bottom). The saved element is then stored at the top or the bottom of the chain.

ENQ/DEQ Subroutines

Since transactions in an online environment may be processed in multi-thread mode, the buffer handler may have to synchronize and/or delay requests for buffers and/or buffer space. This is accomplished in two subroutines which perform ENQ/DEQ type functions and an interlock check. The following fields are used by the ENQ/DEQ routine:

Function	Label	Control block
ENQ/DEQ existing control interval (CI) ID	BFFRPST	Buffer prefix
	PPSTEXCI	PST prefix
ENQ/DEQ pending CI ID	BFFRNPST	Buffer prefix
	PPSTPECI	PST prefix
	PPSTCHAI	PST prefix
ENQ/DEQ subpool	SUBNQFI	Subpool information table
	SUBNQLA	Subpool information table
	PPSTSUPO	PST prefix
ENQ/DEQ matrix	BFPLPSIL	Buffer pool prefix
	BFPLFSIF	Buffer pool prefix
	BFPLPSIL	Buffer pool prefix
	PPSTMATR	PST prefix

For interlock detection, the ENQ/DEQ routines use the contents of the following buffer pool prefix fields:

BFPLINMA interlock detection matrix
BFPLINW1 work areas
BFPLINW2

The ENQ/DEQ routines use the following fields in the buffer pool prefix as work space:

BFPLNQW1
BFPLNQW2

Normally, the resources to be enqueued are the existing contents of a buffer (existing CI ID) or planned contents of a buffer (pending CI ID). Under certain circumstances, other resources may be enqueued.

Enqueuing of a resource consists of the following steps.

If the resource is available:

1. Store the PST ID into a field of the resource reserved for this purpose (that is, BFFRPST, BFFRNPST, SUBNQF1, BFLPSIF).

2. Store the resource ID (for example, the buffer number) into a field in the PST reserved for this purpose (that is, PPSTEXCI, PPSTPECI, PPSTSUPO, PPSTMATR).

3. Indicate successful ENQ with a return code of 4 and return to caller.

If the resource is not available:

1. Find a position for the current PST in the interlock detection matrix.

2. Indicate by an appropriate entry that this PST is waiting and for which task.

3. Check whether this waiting would cause an interlock.

4. If no interlock possible:

   a. Chain with appropriate chain fields the current PST behind the last PST already waiting for this resource.

   b. Return with a return code of 8 to indicate that a wait condition exists.

5. If an interlock would occur if the current PST were to attempt to wait on this resource:

   a. Remove the entry made in 2 above from the interlock detection matrix.

   b. Indicate with a return code of 12 that an interlock would occur and return.

Dequeuing of a resource consists of the following steps.

1. Remove the resource ID from the appropriate field in the current PST.

2. Remove the PST ID from the appropriate field in the resource.

3. If the PST chain fields indicate that no other PST was waiting on this resource, return to caller.

4. If another PST was waiting on this resource:

   a. Move the waiting PST ID into the resource and remove the corresponding wait indication from the interlock detection matrix.

   b. Post the waiting PSTs and unchain the current PST.

   c. If, because of 4.a, certain rows and columns in the interlock detection matrix are free now, make these available for use by other PSTs and post those (see description of action taken on pseudo-interlock conditions).

d. Return to caller.

For performance reasons, resources contain, in addition to the owning PST's ID, the ID of the last PST in the wait chain for this resource. These IDs are also maintained by the ENQ/DEQ routines.

The interlock detection matrix consists of a pair of eight-bit matrices. The first bit matrix indicates for up to eight PSTs which PST is waiting on which other PST. Rows and columns are dynamically allocated to PSTs as required. A one-bit in the appropriate row and column indicates a wait condition. The second bit matrix is the transpose of the first. An imminent interlock is detected by some simple logical operations executed against those two matrices. In the event that eight PSTs are occupying this matrix when further PSTs request service involving a wait condition, a code of 16, indicating pseudo-interlock, is returned and no enqueuing takes place.

The following types of ENQ requests may occur:

ENQ existing CI ID	When a task either wants to write a buffer or wants to get posted when reading into or writing a buffer is finished.
ENQ pending CI ID	When a task wants to reuse a buffer in the buffer pool or when a task wants to get posted when the creation of a pending (i.e., new) CI is finished.
ENQ subpool	When there is currently no buffer prefix in a subpool allowing a pending CI ID.
ENQ matrix	When a task wants to ENQ on a resource currently held by another task and no free row/column in the interlock detection matrix is available.

The following action is taken by the main routine of the buffer handler on a return code (RC) indicating nonsuccessful ENQ.

Condition	RC	Issue
Wait	8	Issue IWAIT macro.
Interlock	12	Dequeue all resources held by this PST and retry the current DL/I request.
Pseudo	16	Dequeue all resources held by this PST and enqueue on interlock detection matrix. This causes a wait condition. Issue IWAIT. Upon post, dequeue matrix and retry current DL/I request.

Control Blocks - DLZDBH00

PST
PPST
DDIR
DMB
DSG
SCD
BFPL
BFFR
SBIF

Interfaces - DLZDBH00

DLZDBH00 uses the PST for communication from and to the calling modules and for work space. The DSG is used to obtain the DMB number and ACB number of the data set which applies during a request. The address of the buffer pool prefix is obtained from the SCD. The address of the buffer prefix area is obtained from the buffer pool prefix. VSAM is invoked for all I/O.

In order to make sure that writing of log information is always ahead of updating a data base, the buffer handler may branch to a specific entry point of DLZRDBL0 or DLZRDBL1. (Refer to the description in the paragraph about DLZRDBL0 and DLZRDBL1.)

DLZDBH00 issues the RELPAG macro for buffers that are marked empty.

Buffer Handler Functions and Required Fields

The following chart illustrates which fields must be supplied to the buffer handler (input) for each specific function and which fields are filled in by the buffer handler (output) on completion of the function.

1. Function used to access a HIDAM or HDAM data base

Function	Input		Output	
	Field	Contents	Field	Contents
PSTBYLCT	PSTBYTNM	Relative byte number of desired segment	PSTDATA	Core address of desired segment
			PSTOFFST	Offset of segment from beginning of control interval
PSTBKLCT	PSTBLKNM	RBA of desired segment	PSTDATA	Core address of desired segment
PSTBYALT		See PSTBYLCT		See PSTBYLCT
PSTBFALT	PSTBUFFA	Address of buffer prefix which is to be marked altered		
PSTGBSPC	PSTBYTNM	Number of desired bytes	PSTDATA	Address of provided buffer
PSTFBSPC/ PSTBFMPT	PSTDMBNM PSTACBNM PSTBLKNM	DMB ACB Control interval RBA All or part of buffer identifier may be processed.		
PSTPGUSR	PSTDMBNM PSTACBNM PSTBLKNM PPSTID	DMB ACB Control interval RBA User identifier Any or all of these may be passed.		

2. Functions used to access a HISAM ESDS

Function	Input		Output	
	Field	Contents	Field	Contents
PSTBYLCT	PSTBYTNM	RBA of the logical record to be read	PSTDATA	Address of the record within the buffer
PSTBFALT	PSTBYTNM	RBA of the logical record to be written		
PSTWRITE	PSTDATA	Address of work area containing the logical record	PSTBLKNM	RBA of the record added to the ESDS as calculated by VSAM
	PSTBUFFA	Prefix Address		

3. Functions used to access a KSDS by key (Simple HISAM, HISAM or HIDAM index)

Function	Input		Output	
	Field	Contents	Field	Contents
PSTSTLEQ	PSTBYTNM	Address of the field which contains search argument	PSTBYTNM	RBA of the logical record retrieved
			PSTDATA	Core address of record
PSTSTLBG			PSTBYTNM	RBA of the logical record retrieved
			PSTDATA	Core address of record
PSTGETNX			PSTBYTNM	RBA of the logical record retrieved
			PSTDATA	Core address of record
PSTPUTKY	PSTDATA	Address of work area containing the logical record		
	PSTBUFFA	Prefix address		
PSTMSPUT	PSTDATA	Address of work area containing the logical record		
	PSTBUFFA	Prefix address		

4. Functions used to access a KSDS by RBA (HISAM or HIDAM index)

Function	Input		Output	
	Field	Contents	Field	Contents
PSTBYLCT	PSTBYTNM	RBA of the logical record to be retrieved	PSTDATA	Address of the record within the buffer
PSTBFALT	PSTBYTNM	RBA of the logical record to be written		
	PSTDATA	Address of record within the buffer		
PSTERASE	PSTBYTNM	RBA of the logical record to be erased		

Calling Sequence

R0 - SCD address
R1 - PST address
R14 - Return address to caller
R15 - Address of DLZDBH00

Fields Required (Independent of Function)

PSTFNCTN Hexadecimal code for desired function

PSTDSGA Address of associated DSG needed for: PSTBYLCT, PSTBKLCT, PSTBYALT

PSTBLKNM Identification of desired block needed for: PSTBKLCT, PSTBFALT, PSTFBSPC

PSTDMBNM Number of associated DMB needed for: PSTBKLCT, PSTBFALT, PSTFBSPC, PSTGBSPC

PSTACBNM — Number of associated ACB needed for: PSTBKLCT, PSTBFALT, PSTFBSPC, PSTGBSPC

PSTBYTNM — PSTBYLCT/PSTBYALT - relative byte address of desired segment - relative record number of HISAM ESDS (high-order byte = X'80')

PSTGBSPC - fullword size of requested space

PSTBUFFA — Address of buffer prefix for block to be marked 'altered' - PSTBFALT

DSGDMBNO — DMB number of the referenced data base

DSGDCBNO — ACB number of the referenced data set

On Return

R15	0	Request satisfied
	4	Warning or error condition

Fields Returned (Independent of Function)

PSTOFFST — Offset from PSTDATA back to first byte of block

PSTDMBNM — DMB number

PSTACBNM — ACB number

PSTDATA — Address of first byte of requested segment, record, or space

PSTBUFFA — Address of buffer prefix

PSTNUMRD — Number of reads done during this call

PSTNUMWT — Number of writes done during this call

PSTCLRWT

Bit 0	This caller waited during request
1-8	Reserved

PSTRTCDE

Return Code Function	Hex Function	Description
PSTCLOK	00	No error occurred during this request.
PSTGTDS	04	Record, CI, or segment requested is more than one CI beyond the end of the data set - returned on PSTBKLCT, PSTBYLCT, PSTBYALT
PSTIOERR	08	Requested CI, record, or segment could not be read successfully on a PSTBKLCT, PSTBYLCT, or PSTBYALT call or could not be written successfully on a PSTPUTKY, PSTMSPUT, PSTWRITE, or PSTBFALT call.

PSTNOSPC	0C	An out of space condition occurred on the data set DASD while processing this request.
PSTBDCAL	10	The byte at PSTFNCTN is not a valid function or the DMB/ACB/BLKID in the PST do not match corresponding fields pointed to in PSTBUFFA for a PSTBFALT call.
PSTNOTFD	14	A PSTSTLEQ call has been issued for a record whose key is higher than the highest key in the data set.
PSTNWBLK	18	The requested CI, record, or segment will go in the CI, one greater than the current end of the data set. Space has been allocated in the pool to hold the new CI. The address is at PSTDATA.
PSTNPLSP	1C	The pool does not contain enough space to satisfy the request.
PSTWROSI	20	A request (GBSPC) was issued for a buffer size which exceeds the highest buffer size handled by any subpool.
PSTENDDA	24	The end of data set has been reached on a PSTGETNX call.
PSTBYEND	28	A request has been issued with a key or RBA higher than the highest key or RBA in the data set.
PSTEOD	2C	End of data set has been reached on a request by DLZDLOC0.
PSTINLD	34	Invalid request during data set loading.

DLZRDBL0 - DB LOGGER

The data base logger module logs the modifications made to a data base. These data base log records are written to the system log. This module is invoked by several of the DL/I modules associated with data base modifications.

The logging of data base modifications, additions, and deletions is done on a physical basis to facilitate a quick recovery procedure. Only calls that actually cause a change to be made to a data base are logged. Two sets of information are logged for each modification - a before set and an after set.

The before information is that required by the data base backout utility. It is used to back out a partially completed update series and to restore a data base to some prior point in time.

The after information is that required by the data base recovery routines to restore the data base from a previous backup copy.

There are five basic types of data base log records.

1. POINTER maintenance record
   When a segment is deleted or inserted and it causes a change in any of the pointers in other segments, each pointer is logged separately as a POINTER maintenance record. A POINTER maintenance record is indicated by bits 1, 2, and 3 of the DLOGFLG2 field of the log record being set to zero.

2. PHYSICAL INSERT record
   When a segment is physically added to the data base, a PHYSICAL INSERT record is written. This type of record is indicated by a one in bit 1 of the DLOGFLG2 field.

3. PHYSICAL DELETE record
   When a segment is physically removed from the data base, a PHYSICAL DELETE record is written. This type of record is indicated by a one in bit 2 of the DLOGFLG2 field.

4. PHYSICAL REPLACE record
   When a segment in a data base is modified, a PHYSICAL REPLACE record is written. This type of record is indicated by a one in bit 3 of the DLOGFLG2 field.

5. LOGICAL DELETE record
   When a DLET call is issued but the segment is not physically removed from the data base, a LOGICAL DELETE record is written. Only the segment code and delete bytes are logged. A logical delete record is indicated by bits 1 and 2 of the DLOGFLG2 field being set to a one.

In addition to data base log records, the data base logger module also uses:

- Application program termination records
- Application program scheduling records
- File open records
- Checkpoint records

The layout for these records is shown in Section 5 of this manual.

Record types 1, 2, 3, and 5 contain the before and after information in the same record and have a log code of X'50'. Type 4 requires two records. The after record has a log code of X'50'; the before record has a log code of X'51'. Additionally, if a physical insert reuses space of a deleted record, log records X'50' and X'51' are written.

If the change is an insert or a delete, the before and after are part of the same record. On an insert, the new segment, including the prefix, is logged as the change data. On a delete, the old segment and prefix are the change data. In HD, both insert and delete cause changes to the free space elements (FSEs) within a block. The new FSEs and their offsets are logged following the change data and a count of the changes is placed in bits 4 through 7 of the DLOGFLG1 field.

The information needed to create the log record is retrieved from the various DL/I blocks. A small amount of additional information is passed as parameters from the DL/I action modules.

The data base log tape format is undefined records (UNDEF). The block size is 1024 bytes. Maximum record length is 512 bytes. If a segment cannot be logged into one record, it is internally spanned over two or more log records. The first record is logged with a data length adjusted to match the data it contains. The offset for the second record is incremented by the length of the first, and the second is written as a separate segment. The adjusting of data length and offset continues until the entire segment is written.

The data base disk log uses VSAM with a CI size of 1024. The user buffer facility is used to ensure that the log records are written immediately. The disk log record format is compatible with the tape log record.

Control Blocks - DLZRDBL0

- Data base log record
- Application program termination record
- Application program scheduling record
- File open record.

Register Contents

R1 - PST address
R13 - Save area
R14 - Return address
R15 - Entry point address.

High-order byte of PSTWRK1 field in PST:

Bit	Value	Definition
0	1	Index maintenance call
1-3	000	Chain maintenance call
	001	Physical replace
	010	Physical delete
	100	Physical insert
	110	Logical delete
	111	Reserved
4	1	Last change for this user call
5	0	One FSE (physical delete or insert)
	1	Two FSEs
6	1	Old copy of physical replace
7	1	New block log call
4&6	1-1	No data - end of user call

PSTWRK1 - Physical SDB address (except new block call)
- Data length (low halfword) if new block call

PSTWRK2, PSTWRK3, PSTWRK4 - Old data on pointer maintenance and logical delete calls. FSE data on physical insert and delete calls.

Before a data base block is updated (that is, before the buffer handler issues the put for an updated block), the associated log information is first written to the log tape or disk in the following manner.

After issuing a put to write a log block to the log tape or disk, the log module updates the count of written log blocks in the field SCDLOCOU.

When the log module processes a log call, in which a data base buffer is involved, the current count of written log records is stored from SCDLOCOU into byte 7 of the buffer prefix in the case of HD, or into the field DMBACBLC in the ACB extension in the case of HISAM and HIDAM index.

Before issuing any put for updating a data base block, the buffer handler compares the value stored in the buffer prefix (HD) or in the ACB extension (HISAM, HIDAM INDEX) with the current value in SCDLOCOU. If the two values are unequal, the log information associated with the data base update has already been written out. If the two values, however, are equal, the buffer handler branches to entry point WRIAHEAD of DLZRDBL0 to force the current contents of the log I/O area to be written out immediately. If, however, asynchronous logging was requested by the user, the count comparison is bypassed, that is, no "write ahead" logging takes place.

Logging in the Online System

In the online system the put for the log blocks is issued in a separate, asynchronous subtask, which is attached at system initialization time. This subtask is a separate CSECT within the log module DLZRDBL0.

The purpose for this is to avoid losing tasks when the end of volume condition is encountered on the log tape.

The communication between the asynchronous log subtask, the logger, and the DL/I online nucleus (DLZODP) is achieved by using three ECBs as follows:

1. System ECB (SCDESECB, in SCD extension), which is used for the communication between the log module (DLZRDBL0) and DLZODP00.

2. Log I/O ECB (SCDELECB, in the SCD extension), which is used for the communication between the log module and the asynchronous log subtask.

3. Private ECB (fullword in the log subtask CSECT), which is used for the communication between the asynchronous log subtask and the log module during the end of the I/O operation that was initiated by the log subtask.

Figure 3-2 shows the events which take place when a PUT for a log block becomes necessary in an online environment.

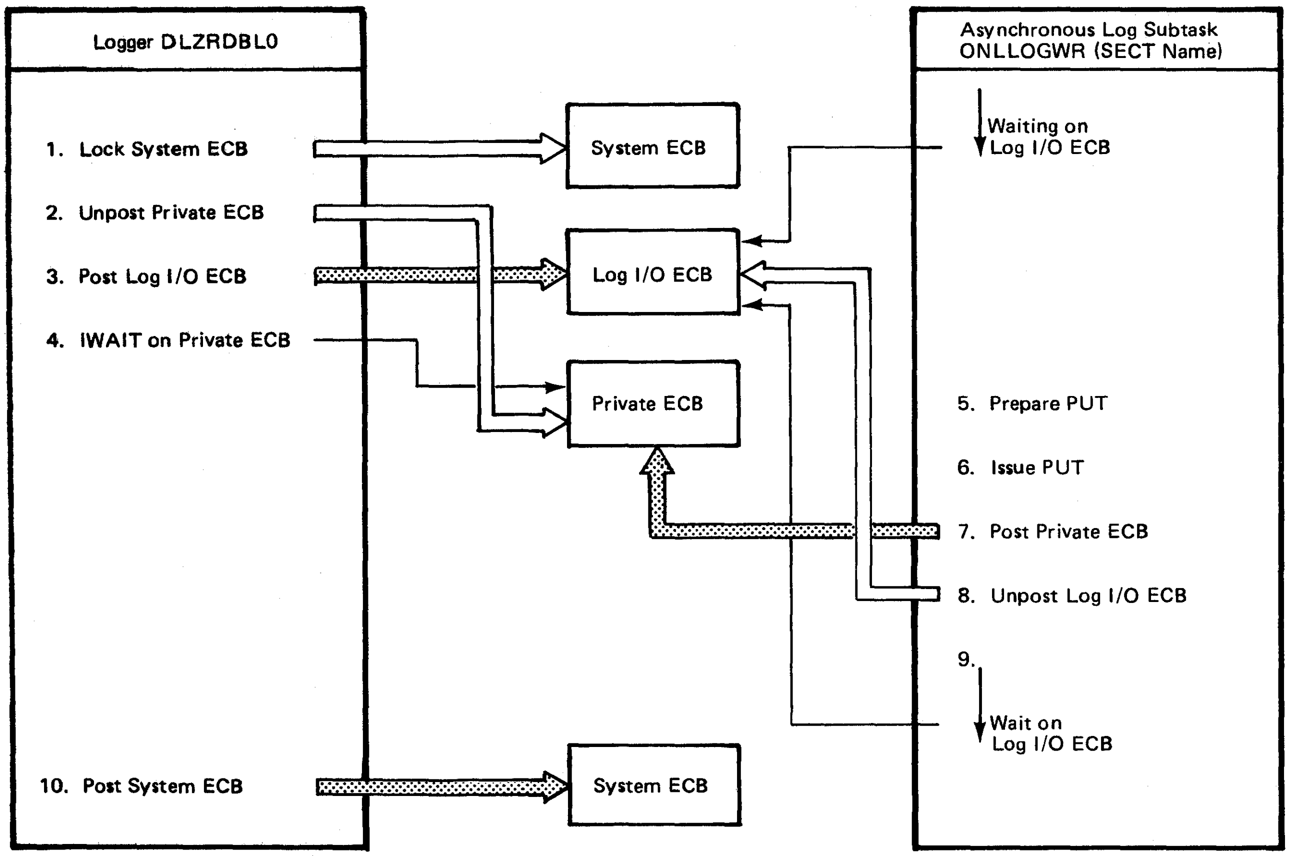


Figure 3-2. Online Log Block Put Operation

The relationship between all modules involved in the asynchronous log writing is as follows:

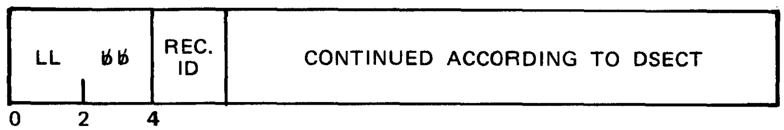
	DLZODP00 PRH Schedul.Rout TERMIN.Rout MESSAGE Rout IWAIT Rout EXCPAD Rout	DLZOLI00	DLZRDBL0	ONLLOGWR
System ECB	Checks system ECB, if LOG subtask is active: 1 Before a call is processed (PRH branches to analyzer) 2 When a log request will be issued 3 Before branching back into a task after control was given up		When PUT has to be issued, unpost system ECB --- After log subtask is finished, post system ECB	
Log I/O ECB		Attach asynchronous log subtask	When PUT has to be issued, post log I/O ECB, get log subtask started	Waiting on log I/O ECB --- After put is finished, unpost log I/O ECB
Private ECB			When put has to be issued, lock private ECB (I/O is active) IWAIT on private ECB	After put, posts private ECB

DLZRDBL1 - CICS JOURNAL LOGGER

Logging in the online system can also be done by using the journaling feature of CICS. That means the DL/I log information as described about module DLZRDBL0 will go on the same file as any CICS journal information.

This is possible because CICS uses different journal record IDs than DL/I (DL/I uses X'07', X'08', X'2F', X'50', X'51'). Any DL/I utility which uses a journal tape will check the record ID and process only those records, which have record IDs used by DL/I.

The general structure of DL/I log records, CICS journal records and CICS journal blocks is shown in Figures 3-3, 3-4, and 3-5, respectively.



Note: DL/I Log Records are described in detail in Section 5 under the heading "Data Base Log Records."

Figure 3-3. DL/I Log Record

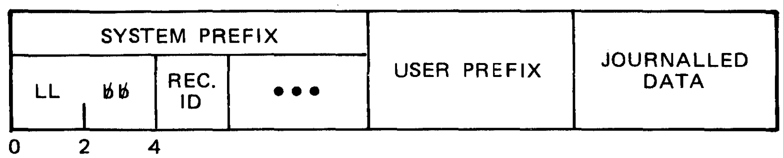


Figure 3-4. CICS Journal Record

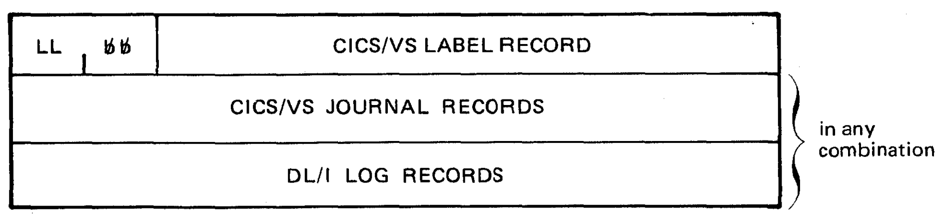


Figure 3-5. Layout of a Journal Block

If the user requests logging by CICS journaling (UPSI bits 6 and 7 = 0), DLZOLI00 loads module DLZRDBL1 instead of the standard log module DLZRDBL0. This module provides the following services:

- Build and write open records for each data base that has been opened. DFHJC TYPE=WRITE is issued to CICS.
- Build and write log records on request by the action modules. DFHJC TYPE=WRITE is issued.
- Write log records built by the sched/term. routine. DFHJC TYPE=WRITE is issued.
- Initiate a physical put to the journal tape on request of the buffer handler. DFHJC TYPE=WAIT is issued.

Before a journal call is issued to CICS, DLZRDBL1 checks if the task which is going to write a journal record already owns a JCA. If it does not, a GET JCA call is issued prior to issuing the DFHJC call.

Since DLZRDBL1 is not reentrant, no task can be allowed to enter this module while log I/O is being processed.

DLZRDBL1 unposts an ECB (SCDESECB) prior to any physical I/O. In various parts of DLZODP this ECB is checked, and, if it is locked, a CICS wait is issued before control is passed to any action module.

When log information is written by using CICS journaling, the writing of log information is always ahead of updating the associated data base blocks. The scheme used is the same as with standard logging, the only difference being that the value for the number of written journal blocks (CICS BCN) is not manipulated by the log module but is taken out of the JCT.

Control Blocks Addressed

- Data base log record
- Application program termination record
- Application program scheduling record
- File open record

DLZQUEFO - QUEUING FACILITY

The DL/I queuing facility module provides resource contention control exclusively for the requirements of program isolation (PI).

Program isolation supports resource contention control at the segment level (for HDAM/HIDAM data bases) and at the record level (for HISAM data base). Module DLZQUEFO provides the control through enqueue/dequeue mechanisms using a unique 7-byte resource identifier:

Bytes 1-4	-	a relative byte address (RBA) associated with the resource
Bytes 5-6	-	the DMB number
Byte 7	-	the ACB number

The RBAs used are:

For segment level resources - RBA of the segment

For record level resources - RBA+1 of the root segment

For variable length segments where data separation has occurred, the segment is considered a single entity with an ID based on the RBA of the prefix.

The queuing facility module will automatically update the RBA portion of the resource ID in the event of a VSAM CI or CA split (HISAM only). The module also contains a deadlock detection routine and will resolve the deadlock by terminating one of the tasks involved.

Three basic control blocks are used to accomplish the enqueue/dequeue function:

1. PST/PPST - used to identify the task.
2. RDB - used to describe a particular resource.
3. RRD - used to describe a particular task's request (either satisfied or pending) for a resource.

As shown in Figure 3-6, the RDBs are chained together, both forward and backward, to one of several queue heads located in the QWA (queuing facility work area). Note that the queue heads have only a forward pointer. The proper queue head is determined by hashing the resource ID and using the results as an index to the table of queue headers.

There is one RDB for each resource, no matter how many tasks (maximum of 255) have enqueued it. The RRBs are forward and backward chained on two queues, one from the RDB and one from the PST for the requesting task. There is one RRD for each resource a task has or is requesting.

On entry to module DLZQUEF0, register 1 contains the PST address and register 15 contains the entry point address (high-order byte contains 'FLAG' if specified). The function requested (enqueue, dequeue, verify, or purge) is contained in the PSTFNCTN field of the PST. If the requested function is enqueue, dequeue, or verify, the PSTQLEV and PSTWRK2 fields also are initialized in the PST. These fields contain the queue request level (read-only, update, or exclusive) and the address of the resource ID, respectively. See Appendix D for the macros used to request a specific function.

Enqueue and verify function are essentially the same and are, therefore, processed by the same routines. The only difference between them is that the user is not the owner of the resource at the return from a verify request.

Three conditions can be present for the processing of the enqueue and verify function:

1. The resource is not currently enqueued (no RDB exists) and is therefore, available. In this case, if the requested function is enqueue, the user is queued as owning the resource and control is returned to the caller. If the requested function is verify, processing is complete.

2. The resource is currently enqueued, but is available at the requested level. In this case, if the request was for an enqueue, the user is queued as an owner at that level and control is returned to the caller.

3. The resource is not available. In this case the user is queued as waiting for the resource, deadlock detection is performed, and a CICS SUSPEND is issued pending the availability of the resource.

   When the wait is satisfied and if the request was for an enqueue, control is returned to the user. If, however, the request was for a verify, the user is first dequeued (see dequeue function) as owner of the specified level before he is given control.

Dequeue function processing first determines if the resource is currently owned by the requestor. If it is not, the request is ignored. If it is, the enqueue count at the specified level is decremented. If all levels are now zero, task ownership is relinquished, and any waiting tasks that may now own the resource are promoted. If FLAG was specified, it is set for all waiting tasks.

If the enqueue count goes to zero and it was the highest level, but lower levels still exist, the ownership level is lowered and any waiting tasks that may now own the resource are promoted.

Purge function processing searches the chain of RRDs queued off the specified PST for a task and unconditionally relinquishes ownership for all resources encountered. Any waiting tasks that may now own the resource are promoted.

On return from module DLZQUEF0, return codes are set in register 15 and in the PSTRTCDE in the PST.

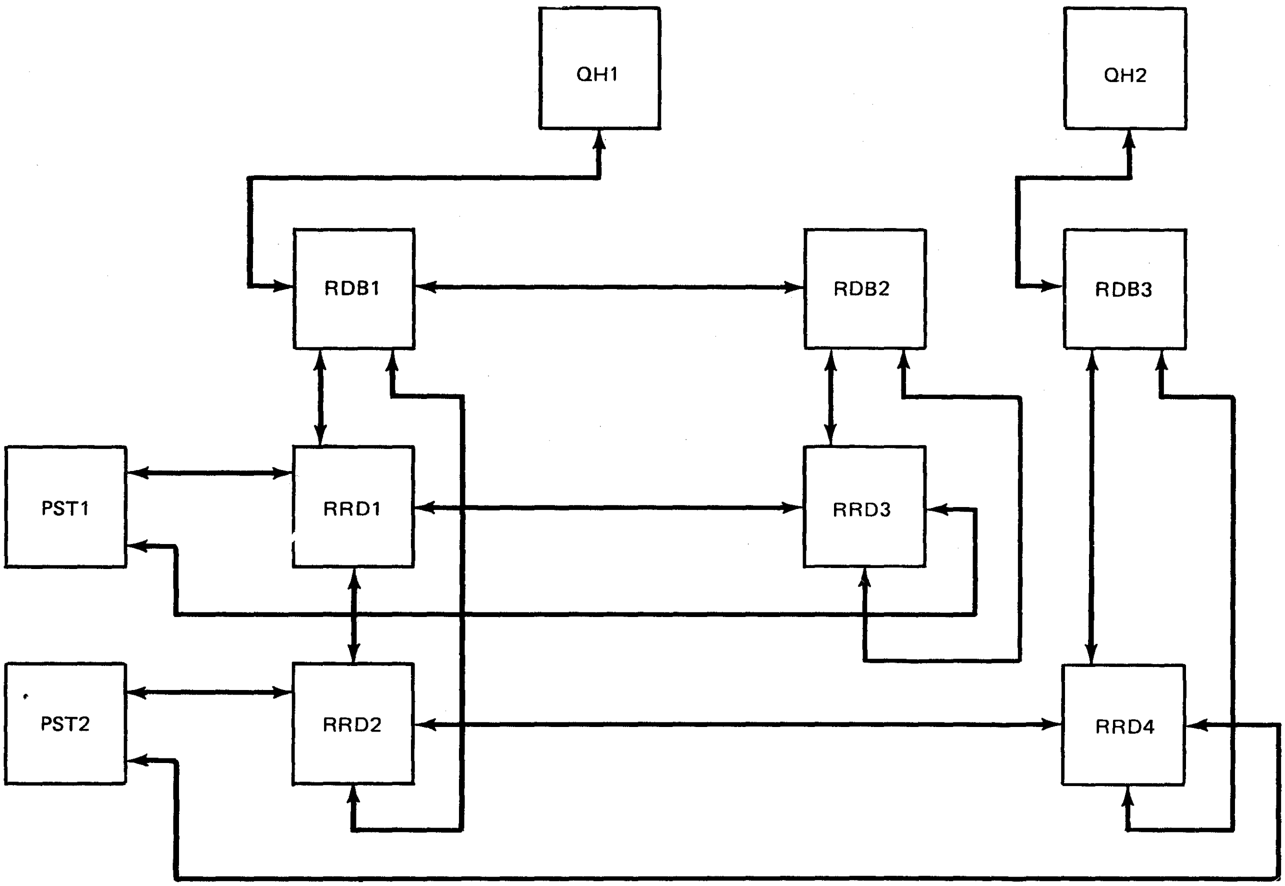


Figure 3-6. Enqueue/Dequeue Control Block Relationships

The following table identifies the mainline routines and the functional subroutines of the queuing facility module:

Mainline Routines

Routine	Function
QENQDEQ	Common Entry Logic
QRETURN	Common Exit Logic
QENQVER	Enqueue/Verify Mainline
QNRENQ	New Resource Enqueue/Verify
QERENQ	Existing Resource Enqueue/Verify
QREENQ	Re-enqueue or Verify of Resource Already Owned
QDEQ	Dequeue Mainline
QDEQVER	Dequeue Specific RRD
QRELRSC	Relinquish Ownership of Resource
QPUR	Dequeue all Resource for a Task
DLZJRNAD	Update Routine for RBA on CI or CA Split

Functional Subroutines

Routine	Function
QLOCRDB	Locate RDB or Position on Chain
QLOCRRD	Locate RRD or Position on Chain
QBLDRDB	Build, Initialize, and Chain RDB
QBLDRRD	Build, Initialize, and Chain RRD
QUCFRDB	Unchain and Free RDB
QDASOWN	Define Task as Owner of Resource
QWAIT	Wait for Ownership of Resource
QLOCNPO	Locate New Prime Owner
QPNOWCM	Promote New Owners, Do Wait Chain Updates
QPFLAGP	Pass Flag Parameters To Waiting Tasks
QDLKDTN	Detect and Resolve Deadlocks
QDLKRSV	Resolve Deadlocks
QGETBLK	Get 24-Byte Block from Free Chain
QRETBLK	Return 24-Byte Block from Free Chain

Data Areas Used

SCD
PPST
PST
RDB
RRD
QWA

Entry Points

QENQDEQ - General entry point for request to enqueue, dequeue, or verify a resource, or to purge enqueues for a task.

DLZJRNAD - Entry point to update the RBA portion of any resource IDs as required due to data movement during a VSAM CI or CA split (HISAM only).

DLZCPY10 - FIELD LEVEL SENSITIVITY COPY

DLZCPY10 has two entry points: DLZCPY10 and DLZSEGCV.

The function of DLZCPY10 is to map the user view of a segment into its physical view for DL/I ISRT and REPL calls, in support of field level sensitivity. On a path call, DLZCPY10 maps the segment at each level of the path. If a level in the path is not field sensitive, the segment at that level is moved without modification. DLZCPY10 is invoked by Call Analyzer (DLZDLA00).

The function of DLZSEGCV is to convert a segment from either the physical view to the user view, or the user view to the physical view. DLZSEGCV is invoked by DLZCPY10 to convert ISRT and REPL calls from user view to physical view. DLZSEGCV is invoked by Retrieve (DLZDLR00) to convert Get calls from physical view to user view. DLZSEGCV is also invoked by Retrieve to convert SSA values from user view to physical view.

Interfaces - DLZCPY10

This module interfaces with the following module:

DLZDBH00

Register Contents at Entry

R1	=	PST address (DLZCPY10)
		FER address (DLZSEGCV)
R5	=	SDB address (DLZSEGCV)
R13	=	Save area address
R14	=	Return address
R15	=	Entry point address (DLZCPY10)
		Addr(DLZCPY10)+4 - (DLZSEGCV)

Control Blocks - DLZCPY10

SDB	PSB
SDB Exp.	PCB
FSB	JCB
FER	LEV
FERT	PSDB
PST	FDB
SCD	SEC
PDIR	DDIR

MPS CONTROL MODULES

DLZMSTR0 - START MPS TRANSACTION

This module is invoked by the user via a specific transaction code (CSDA) to start multiple partition support (MPS). The responsibilities of this module are to:

- Check if the DL/I nucleus is loaded.
- Check if MPS is already active.
- Attach the master partition controller (DLZMPC00).

Control Blocks Addressed

CSA-Common System Area (CICS/VS)
SCD-System Contents Directory

Register Contents
R13 Contains CSA address

DLZMPC00 - MASTER PARTITION CONTROLLER (MPC)

The master partition controller (MPC) is attached by the start transaction module (DLZMSTR0).

The functions performed by the master partition controller are:

- Initialize the MPC partition table (DLZMPCPT).
- Define some of the XECBs required for cross partition communication.
- Process all start batch partition controller (BPC) requests and attach a BPC for a specific batch partition.
- Process all stop partition requests.
- Process the abend condition if the batch partition controller attach fails.
- Process the stop transaction request to terminate MPS.
- Return control to CICS/VS after all activity is completed.

Control Blocks Addressed

MPCPT	MPC Partition Table
SYSCOM	System Communication Region
CSA	Common System Area (CICS/VS)
SCD	System Contents Directory
MPCECBLT	CICS ECB Pointer List
TCA	Task Control Area

Register Contents
R12 Contains TCA address (at entry)
R13 Contains CSA address (at entry)

Macros Used

```
DFHKC     TYPE=WAIT
DFHKC     TYPE=ATTACH
DFHPC     TYPE=ABEND
DFHPC     TYPE=SETXIT
DFHPC     TYPE=RETURN
XECBTAB   TYPE=CHECK
XECBTAB   TYPE=DEFINE
XECBTAB   TYPE=DELETE
XPOST
```

DLZBPC00 - BATCH PARTITION CONTROLLER (BPC)

The batch partition controller (BPC) is attached by the master partition controller (MPC) when a start request has been made by a batch partition. The functions performed by the batch partition controller are:

- Define XECB for cross partition communication with the MPS batch initialization (DLZMINIT), MPS batch program request handler (DLZMPRH), and MPS batch termination (DLZMTERM).
- Issue the DL/I scheduling call on behalf of the batch partition.
- Process all DL/I calls on behalf of the batch partition.
- Process ABEND conditions occurring in the batch partition.
- Return control to CICS/VS for normal and abnormal conditions

This module must be link-edited with the language interface module, DLZLI000.

Control Blocks Addressed

MPCPT	MPC Partition Table
TCA	Transaction Control Area
TWA	Transaction Work Area
PST	Partition Specification Table
PPST	Prefix PST
DLZXCB1	DL/I Parameter List

Register Contents

R12 Contains TCA address (at entry)
R13 Contains CSA address (at entry)

Macros Used

```
DFHKC    TYPE=WAIT
DFHKC    TYPE=ATTACH
DFHKC    TYPE=RETURN
XECBTAB  TYPE=CHECK
XECBTAB  TYPE=DEFINE
XECBTAB  TYPE=DELETE
XPOST
```

DLZMPI00 - MPS BATCH

The MPS batch module is made up of the following five routines:
1. MPS Batch Initialization (DLZMINIT)
2. MPS Batch Termination (DLZMTERM)
3. MPS Batch Program Request Handler (DLZMPRH)
4. MPS Batch Abend (DLZMABND)
5. MPS Batch Message Writer (DLZMMSG)

A separate description for each routine is given in the following text.

MPS Batch Initialization - DLZMINIT

This is one of five routines that make up module DLZMPI00 to support the batch part of MPS.

DLZMINIT reads the input parameter statement and checks it for validity. It then loads the user's program. Next, it determines what to use as a partition identifier by checking the PIK in the COMREG. This value is used in online messages. The value for 'n' in XECB names is found in the partition table entry pointed to in the area following XECB DLZXCB02, and is put into each XECBTAB macro issued.

After saving the program name and PSB name for use by online, an XECB, DLZXCBn1, is defined in the batch partition for communicating with the online partition. The online partition XECB, DLZXCB02, is XPOSTed. This lets the online partition know that there is an MPS batch job ready to run.

When the online partition completes its initialization, the batch routine sets up STXIT routines, finishes other initialization activities, and goes to the user program.

DLZMINIT is entered by DOS/VS job control at the start of the job.

Control Blocks Addressed

MPCPT	MPC Partition Table
TCA	Transaction Control Area
PST	Partition Specification Table
COMREG	Communication Region
XCB1	XECB DLZXCBn1 and data following it
DTFs for	SYSLST, SYSLOG, and SYSIPT
STXIT AB	Savearea
STXIT PC	Savearea
XECBs	DLZXCB02, DLZXCBn2, DLZXCBn3

Register Contents (at Entry to Other Routines)

- User Program
  - R1 PCB list if not PL/I; or a pointer to a list containing the following if PL/I:
    - address of PCB list
    - address of location containing size of dynamic storage
    - address of start of dynamic storage
  - R13 Save area
  - R14 Return address
  - R15 Entry address

- Message Writer (DLZMMSG)
  - R14 Return Address

- ABEND Routine (DLZMABND)
  - No special register values

Macros Used

XECBTAB TYPE=DEFINE
XECBTAB TYPE=DELETE
XECBTAB TYPE=CHECK
XPOST
XWAIT
OPEN
CLOSE
EXTRACT
GET
GETVIS
PUT
CANCEL
STXIT PC
STXIT AB
MVCOM
COMRG
LOAD
LOCK
UNLOCK

MPS Batch Termination - DLZMTERM

This is one of five routines that make up module DLZMPI00 to support the batch part of MPS.

The MPS batch termination routine is entered when the user program finishes. It tells the online partition to do termination activity, deletes its own XECB, and ends the job.

Control Blocks Addressed

XCB1 XECB DLZXCBn1 and the data following it

Register Contents

Registers have the same values at entry as when MPS batch initialization (DLZMINIT) completed.

Macros Used
XPOST
XWAIT
EOJ
LOCK
UNLOCK
XECBTAB TYPE=DELETE

MPS Batch Program Request Handler -DLZMPRH

This is one of five routines that make up module DLZMPI00 to support the batch part of MPS.

The MPS batch program request handler routine is entered on each call to DL/I made by the user program. The user call list is validated and set up for the online partition to use. Then the online partition is notified by an XPOST of XECB DLZXCBN2. When the call is complete, data is moved to the user's I/O area.

Control Blocks Addressed

MPCPT	MPC Partition Table
TCA	Transaction Control Area
PST	Partition Specification Table
XCB1	XECB DLZXCB1

Register Contents

- At entry:
    R0   Bit X'01' ON if PL/I, OFF if not PL/I
         Bit X'02' ON if HLPI, OFF if call interface
    R1   If PL/I, points to list of pointers to parameters;
         if not PL/I, points to list of parameters
    R13 Save area
    R14 Return address
    R15 Entry address

- Message Writer (DLZMMSG)
    R14 Return address

Macros Used

GETFLD
STXIT PC
XPOST
XWAIT
XECBTAB TYPE=CHECK

MPS Batch ABEND - DLZMABND

This is one of five routines that make up module DLZMPI00 to support the batch part of MPS.

The MPS batch abend routine has four entries:

1. External routine
2. PC STXIT
3. AB STXIT
4. Other MPS batch routines that cause abnormal termination.

The first entry initializes registers and then joins the main path. The next two each identify which way the ABEND routine was entered. They then issue an error message. Then the fourth entry joins them as the online partition is notified. All entries delete the batch XECB and cancel or dump.

When an abnormal termination situation has occurred, DLZMABND is entered by:

- DLZMINIT
- DLZMTERM
- DLZMPRH

Control Block Addressed

STXIT AB Save area
STXIT PC Save area

Register Contents

- At entry
  No special values except base registers initialized

- Message Writer (DLZMMSG)
  R14 Return address

Exits

JDUMP   If dump requested
CANCEL   If no dump requested

Entry Points

External routine	Abnormal end for separately assembled routine
STXIT AB	If abnormal end entered by DOS/VS
STXIT PC	If program check determined by DOS/VS
XPOST Entry	Other abnormal end when BPC must be notified

Macros Used

DLZIDUMP
LOCK
UNLOCK
XPOST
XECBTAB TYPE=DELETE
JDUMP
CANCEL

MPS Batch Message Writer - DLZMMSG

This is one of five routines that make up module DLZMPI00 to support the batch part of MPS.

There are two entries:

- From external routines
- From routines within DLZMPI00

The MPS batch message writer routine handles all messages issued by the MPS batch partition. At entry, a parameter list is set up. The first parameter is always a pointer to the message number. Other parameters, if any, are as needed for the message.

When a message is to be written to SYSLOG and/or SYSLST, the DLZMMSG routine is entered by:

- DLZMINIT
- DLZMTERM
- DLZMPRH
- DLZMABND
- External routines

Control Blocks Addressed

DTFs for SYSLOG and SYSLST

Register Contents

- At entry:
  R14 Return address
  Base registers already initialized except for external routine entry, which initializes registers before joining mainline

- At entry to message table (DLZMMSGT):

R1	Points to parameter list
R4	Base register for DLZMMSGT
R5	Address of where message is to be placed
R7	Length of message set up before calling DLZMMSGT; after call, R7 has total message length
R9	Points to PST (for checkpoint message DLZ105I)
R10	Second base register for DLZMMSGT

Exits

To calling routine via branch register 14

Macros Used

PUT

DLZMSTPO - STOP MPS TRANSACTION

This module is invoked when a user wants to stop MPS. The user inputs a specific transaction code (CSDD) defined to initiate the stop transaction processing. The module then posts the particular XECB that causes the MPC to end the MPS environment.

After the post, the MPC allows batch jobs already executing to complete, but will not allow any new ones to start.

This transaction should be started before CICS/VS non-immediate shutdown is initiated.

Macros Used

XECBTAB TYPE=CHECK

DLZBACK0 - BATCH BACKOUT INTERFACE

The batch backout interface module reads and validates any 'II' control statements from SYSIPT. A log input specification table describing each log file to be processed is created. The module then reads the DL/I log files and passes the data base log records to the data base backout module (DLZRDBC0) for processing.

By reading the log files in a backward mode, this module is able to process the data base records in reverse sequence without using an intermediate work data set. When a block is read in, it is searched and the sequence field located at the end of each logical record is replaced by the length of that logical record. With the length thus in the back of a record as well as in the front, it is deblocked and spanned.

The interface process includes the following record types:

X'07' - Application program termination record
X'08' - Application program scheduling record
X'41' - Checkpoint record
X'50' - Data base log record
X'51' - Data base log record

The batch backout utility is executed under DL/I control as an application program. Processing of module DLZBACK0 is as follows:

1. Control is received from DL/I initialization and the PSB name is obtained from the parameter data.

2. The log file is opened to be read backward.

3. The log file is read backward and records bypassed until the first data base log record for the PSB is obtained.

4. An application program termination record (X'07') for the PSB indicates no backout necessary, the message "BACKOUT COMPLETE" is issued at SYSLOG, the log is closed, and the job is terminated.

5. Data base log records (X'50' and X'51') are passed to module DLZRDBC0 to be processed against the appropriate data base. Processing terminates when an application program scheduling record or a checkpoint record is read, the message "BACKOUT COMPLETE" is issued at SYSLOG, the log is closed, and the job is terminated.

If end of file is reached on the log (i.e., the header record is read), it is closed. If more log files are to be processed, the above process is repeated starting at step 2. Multiple log files must be processed in reverse order of their creation. When all log files are processed, a "BACKOUT COMPLETE" message is issued and the job step is terminated. The job is terminated by returning control to DL/I which purges all buffers, closes all DMBs, and closes the output log file.

Register Contents on Entry

R1 = PSB list address
R13 = Save area
R14 = Return
R15 = Entry point

Control Blocks - DLZBACK0

Application program scheduling record
Application program termination record
Checkpoint record
Data base log record
DMB
PDIR
PSB
PST
SCD

External Modules Called

DLZRDBC0 - Called to interface with DL/I and perform backout.

Record and Message Formats - DLZBACK0

All messages are sent to the SYSLOG and SYSLST devices. The messages are contained in module DLZBACM0.

DLZRDBC0 - DB CHANGE BACKOUT

This module receives control from DLZBACK0 with a log record to process. It calls open/close (DLZDLOC0) to open the DMB specified in the record unless the data base is already open. The buffer handler (DLZDBH00) is called to retrieve the KSDS or ESDS block as indicated by the key or the ESDS relative block number or relative byte address.

The data in the buffer is replaced with the 'old' information in the log, thereby nullifying the offending programs update. In the case of HD, when a physical delete or insert record is processed, space management (DLZDHDS0) is called to update the free space elements and bit map, if necessary and to build the input data for the data base logger. DLZRDBL0 is called to record the changes made to the data base.

The buffer handler is then called again to mark that buffer altered and control is returned to DLZBACK0.

Register Contents and Control Blocks on Entry

```
    R1   = PST address
    R13  = Save area
    R14  = Return
    R15  = Entry point
PSTSCDAD =  SCD address
ADDRLOG  =  Address of data base log record within DLZBACK0
PSTDGU & PSTDGN must be zero on initial entry
```

Control Blocks - DLZRDBC0

Data base log record
DDIR
DMB
DSG
PCB
PDIR
PSB
PST

SCD

External Modules Called

DLZDBH00 - Called to read a data base record and to mark the buffer altered
DLZDHDS0 - Called to free or reserve space in an HDAM or HIDAM record
DLZDLOC0 - Called to open data base
DLZRDBL0 - Called to log backout modifications to data base

Interface with External Modules

All modules expect R14 + R15 to contain return address + module entry point address.

DLZDLOC0

R1 = address of PST
R2 = address of DDIR entry for DMB to be opened

PSTDSGA = address of DSG to open
PSTFNCTN = PSTCCDMB + PSTOCOPN
SCDCWRK = address of normal log record work area

DLZDBH00

R1 = address of PST

PSTBLKNM = RBN if HD ESDS
PSTACBNO = 1
PSTDMBNO = 1
PSTBYTNM = RBA if HISAM ESDS or address of key if KSDS
PSTFNCTN = desired function

DLZDHDS0

R1 = address of PST
R5 = address of PSDB of segment

PSTOFFST = offset to segment from beginning of block
PSTCODE1 = indicates backout in control (for logger)
PSTFNCTN = PSTFRSPC + X'80' (to show backout in control)

DLZRDBL0

R0 = SCD address
R1 = PST address

PSTCODE1 = PSTINTNT + PSTSCHED to indicate backout calling
PSTDATA = address of data in buffer
SCDCWRK = address of backout log work area containing the control information for this log record

Register Contents on Exit

All registers are restored with the exception of register 15 which contains a return code. If this code is non-zero, DLZBACK0 will print and type the appropriate error message.

Error Codes and Handling - DLZRDBC0

All error codes are passed to DLZBACK0 in register 15.

DLZURDB0 - DB DATA SET RECOVERY

The data base data set recovery utility module DLZURDB0 is executed under DL/I control as an application program. Control is passed to DLZURDB0 from DL/I initialization. This module is comprised of two independent but logically related functions. The first consists of an image dump and a change accumulation processor. The PCB address is saved, and a GSCD call is issued to obtain the PST address. Control is passed to DLZURCC0 to read and process control statements from SYSIPT. From information saved by DLZURCC0, a DMB is loaded from the Core Image Library to obtain the physical characteristics of the data set to be recovered. The DL/I open/close routine (DLZDLOC0) is called to open the output ACB and the input file is opened. Then the program enters a dump/cum data merge routine. This routine selects a dump record, merges any accumulated changes from the cum data set, and a call is made to the buffer handler (DLZDBH00) to write the new record to the output data set. Upon completion, a partial or completely recovered data set may exist. If no additional changes are to be applied through log files, the program calls the DL/I open/close routine (DLZDLOC0) to close the output ACB and terminates.

If additional changes are to be applied from log files, the program enters the second function. This routine opens the logs, scans the log to find a record that applies to this data set, and merges the data from the log to the data set record. Upon completion, the routine does post-processing and a recovered data set then exists.

The operation of this routine depends on certain DL/I functions to process the logs. The log is scanned for a matching data base/data set name record. When one is encountered, the record ID, either a key of a KSDS record or a relative block number of an ESDS record is saved, and a call is made to the buffer handler (DLZDBH00) requesting that the record be retrieved. Upon successful return, the log record data is merged with the returned record, and a call is made to the buffer handler requesting that the record be marked as altered to cause rewriting. The records from the log are thus processed until an end of file is encountered on the log input. At this time, a call is made to the buffer handler requesting that all altered buffers be purged, that is, that all records that have been altered be rewritten. The program then calls the DL/I open/close routine (DLZDLOC0) to close the output ACB, and the program terminates.

Blocks and Tables - DLZURDB0

This module utilizes certain DL/I blocks, including the PST, DSG, DMB, DMB directory, SDB, PCB, JCB, and SCD. Additionally, several record formats are used as follows:

1. HISAM reorganization header and data records. See HISAM reorganization unload (module DLZURUL0) for details.

2. Data base image dump header and data records. See data base data set image copy module (DLZUDMP0) for details.

3. Accumulated change CUM header and data records. See change accumulation module (DLZUCUM0) for details.

4. Data base change log records.

Normal Entry Points

The only entry point to this module is DLZURDB0.

Register On Entry

R1 = pointer to fullword containing address of PCB

Registers On Exit

All registers are restored to entry conditions.

Modules Called by DLZURDB0

The recovery control statement processor (DLZURCC0) is called to read and validate any input control statements.

R1 = pointer to recovery common area

The DL/I open routine (DLZDLOC0) is called to open a specific ACB.

R1 = pointer to PST

The DL/I buffer handler (DLZDBH00) is called to retrieve and write a specific record, mark a buffer altered, and purge (rewrite) all altered buffers.

R1 = pointer to PST

The DL/I close routine (DLZDLOC0) is called to close a specific VSAM ACB.

R1 = pointer to PST

Error Codes and Handling - DLZURDB0

All codes are in the form of messages. The module DLZRDBM0 contains all error messages issued by the Data Base Data Set Recovery Utility.

DLZURCC0 - Recovery Control Statement Processor

This module reads and validates the input control statements from SYSIPT. The 'S' control statement describes the data base to be recovered. The 'LI' control statements describe the log files to be processed. Information from these statements is saved in the recovery common area for use by DLZURDB0.

Normal Entry Point

The only entry point to this module is DLZURCC0.

Registers on Entry

R1 = pointer to recovery common area.

Registers on Exit

All registers are restored to entry conditions except R15, which contains a return code (see below).

Error Codes and Handling

Messages are issued to SYSLST and SYSLOG for any invalid control statements. On return to DLZURDB0, R15 is set as follows:

R15 = 0 - No errors
R15 = 4 - No input control statements
R15 = 8 - Input control statement error

DLZUDMP0 - DB DATA SET IMAGE DUMP

The data base data set image copy utility module DLZUDMP0 is executed as a standard DOS/VS application program and creates a backup copy of a specific data base data set. Input may be either a KSDS (HISAM, Simple HISAM, or HIDAM INDEX) or an ESDS (HISAM, HIDAM, or HDAM). The output is used as input to the data base data set recovery utility. Processing is as follows:

1. A control card is read from SYSIPT and preliminary validity checking is performed on various fields. The input card defines the data base/file to be dumped, the dump output symbolic filenames, and the number of output copies to be created.

2. The device type is determined for each output file specified and the file(s) are opened.

3. The DMB is loaded from a core image library to obtain the physical characteristics of the data base file to be dumped.

4. A header record is written to the output file. This record contains information necessary to allow the use of the image dump file by the data base data set recovery utility.

5. The input file is opened.

6. Input segments are read sequentially, an 8-byte prefix is added to identify the segment, and the logical record (prefix + segment) is blocked and written to the output file.

7. After all segments have been copied (EOF), the input and output files are closed.

8. Output statistics for the file are written to SYSLST.

9. Processing continues from step 1 until there are no more input cards, at which time the program terminates.

Control Blocks - DLZUDMP0

- Dump record prefix
- Dump header record.

Error Codes and Handling - DLZUDMP0

All error codes are in the form of messages to SYSLST and SYSLOG. All the messages used by the DB Data Set Image Dump Utility are contained in module DLZDMPM0; a read-only CSECT.

DLZUCUM0 - DB CHANGE ACCUMULATION UTILITY

The data base change accumulation utility module DLZUCUM0 is executed as a standard DOS/VS application program. DLZUCUM0 controls the overall operation of the Data Base Change Accumulation Utility. First, the control card processor module (DLZUCCT0) is called to read the input stream. Upon its return, the PROCFLAG switch is tested. If records are to be passed to sort, the sort parameter list is formatted, including a sort Exit 15 (DLZUC150) and the sort Exit 35 (DLZUC350). The sort program is then loaded, and this module (DLZUCUM0) waits for it to terminate. Upon termination, a completion code is tested and appropriate messages are provided as output. If records are not to be sorted, that is, no DB0 type control cards were read, the module calls the Exit 15 module (DLZUC150) to create the new log file. If error are encountered by any of the four processing modules, control is passed to the common error routine DLZUCER0.

Control Blocks - DLZUCUM0

- Data base name table, containing the data base names and the address of the date/time table for this entry.
- Data/time table
- Accumulation header record
- Accumulation record

Normal Entry Point

The main entry point to this module is DLZUCUM0. DLZERRTN is an entry point used by DLZUC150 on any error condition.

Entry Conditions

This is the main module which controls the overall operation of the Data Base Change Accumulation Utility program.

Control information is passed from module to module by means of an externally referenced table contained in DLZUCUM0.

DLZUCER0 - Common Error Routine

This module is the common error routine. Control may be passed to it from any of the four processing modules. It addresses a message from the message module (DLZCUMM0), depending on parameters passed to it, and prints a message to the SYSLST and SYSLOG devices. If the passed parameters indicate a multi-part message, it does not write the message on the first entry. Instead, it passes the last-used position in the output buffer back to the caller to allow the caller to insert special data in the messages. On the second entry to this routine, the message is written. All messages issued by the DB Change

Accumulation Utility are contained in module DLZCUMM0. It is a read-only module.

Normal Entry Point

The only entry to this module is DLZUCER0.

Entry Conditions

This module is entered to output all error messages.

Register Contents on Entry

R1 contains a message number. R2 is negative if this is a multi-part message. (R2 points to last byte of message on second entry of multi-part message.)

Register Contents on Exit

All registers are restored to entry conditions except R2, which points to last byte of message on first entry return of multi-part message.

DLZUCCT0 - Control Card Processor

This module is the control card processor. It reads the control card input stream, checks the cards for validity, and constructs the data base name table and the date/time table if data base names are supplied. It also constructs the log input specification table describing the input log file(s).

Normal Entry Point

The only entry to this module is DLZUCCT0.

Entry Conditions

This module is entered to process the control card input stream.

Register Contents on Exit

All registers are restored to entry conditions.

DLZUC150 - Sort Exit 15

This module is the sort Exit 15 routine. It reads the log input records, checks the purge date if applicable, and determines the disposition of the record. If the record matches an entry in the data base name table, the date/time table is searched and the appropriate purge date and time are compared. If the record is before the purge date, the program returns to read another record. If the record is not purged, the routing is determined from the table and written either to sort or to the new log. A table of DMB names and purge dates is prepared for Exit 35.

Normal Entry Point

This module is entered at DLZUEX15 if no records are to be accumulated, and at DLZUC150 by sort.

Entry Conditions

This module is entered to read input logs and disperse records to new log or sort. R1 contains the address of the parameter list from sort or a dummy list if control was received from DLZUCUM0.

Register Contents on Exit

All registers are restored.

DLZUC350 - Sort Exit 35

This module is the sort Exit 35 routine. It receives all records from sort. If an old accumulated data set is supplied, a record is read from the data set and a record is retrieved from sort. The data base name and file identification of the records are compared. All input cum records are purge-checked according to the date/time, if any, specified on DB0 card(s). If the old cum input is low, it is written to the new cum data set. If the records are equal, the data from the sort record is merged to the old cum record, unless purged, and another record is obtained from sort. This sequence continues until an unequal condition is detected, at which point the record is written to the new cum data set. If the old cum is high, records from sort are combined and written to the new cum data set until the compare condition changes. This process continues until both the sort and the old cum records are exhausted.

Normal Entry Point

This module is entered at DLZUEX35 by sort.

Register Contents on Entry

Register 1 contains the address of the sort Exit 35 parameter list.

Entry Conditions

This module is entered by sort to dispose of all sorted records.

Register Contents on Exit

All registers are restored to entry conditions, with the sort parameter list updated as needed.

DLZLOGP0 - LOG PRINT UTILITY

The log print utility module (DLZLOGP0) is executed as a standard DOS/VS application program and prints the contents of DL/I log files. Input log files may be either tape or disk. Optionally, the utility can create an output log tape suitable as input to the backout utility module (DLZBACK0). Processing of the log print utility is as follows:

1. Module DLZLPCC0 is called to process input control statements.

2. If requested, the output log tape file is opened.

3. The DLZDVCE macro is issued to determine the log device type, and the log file is opened.

4. The log records are read and deblocked, and the record types are checked to see if valid DL/I record.

5. The log records are printed to SYSLST in either keyword format or dump format.

6. If requested, log records are written to output log tape.

7. The input log file is closed. If more input log files were specified, processing continues from Step 3.

8. If requested, the output log file is closed.

9. Informational statistics are written to SYSLST and the program terminates.

Error Codes and Handling

All error codes are in the form of messages written to SYSLST and SYSLOG. All the messages used by the log print utility are contained in module DLZLGPM0.

DLZLPCC0 - Log Print Control Statement Processor

This module is called by DLZLOGP0 to read and process input control statements. The control statements are read from SYSIPT and validity checking is performed. Valid control statement types are: 'LO', 'LS', and 'LI'. Information from the control statements is saved in the log print common area.

Normal Entry Point

This module is entered at DLZLPCC0 by DLZLOGP0.

Register Contents on Entry

Register 1 points to the log print common area.
Register 9 points to the next available print line buffer.

Entry Conditions

This module is entered by DLZLOGP0 to read and process input control statements.

Register Contents on Exit

All registers are restored to entry conditions except register 9, which is updated to point to the next available print line buffer.

Error Codes and Handling

All error codes are in the form of messages written to SYSLST and SYSLOG. All the messages used by the log print utility are contained in module DLZLGPM0.

DLZURUL0 - HS DB UNLOAD

The HISAM reorganization unload module DLZURUL0 is executed as a standard DOS/VS application program. A control card specifying the data base name, data set name, and output symbolic unit name is read. The DBD specified is loaded, and a short segment table is constructed. This table consists of the first eight bytes of each segment table entry in the DBD. This includes, among other things, the segment physical code and the segment length. The size of the prefix, as described for each segment type, is added to the segment length and entered in the table. This length is later used to move the segment from the input area to the output area.

Next, the input and output data sets are opened. A header record containing information about the data base data sets is constructed, and a statistics record is written. The first KSDS record is then read and the root segment is checked to determine whether the deleted flag is on (no prefix if Simple HISAM). If it is on, the total segment chain for that root is ignored, and the next root is processed. If the root is not deleted, it is moved to the output area, and the first dependent segment, if present, is processed. If the dependent segment is not deleted, it is moved to the output area, and the next segment is processed. This continues until the complete dependent segment chain for this root, including any overflow dependent segments on the ESDS, have been processed. If the segment is deleted, each succeeding segment that is a child of the deleted segment is also deleted. The first segment that is not a child of the deleted segment causes the normal segment processing to be resumed. The last record written is a statistics record which includes information needed for audit trail. The output data set now contains the reorganized KSDS and ESDS logical records in physical sequential format (only KSDS if Simple HISAM or INDEX). An image of the KSDS record containing a root segment and dependent segment is followed by images of the ESDS records containing overflow dependent segments for the root segment. A chain pointer in the KSDS contains the correct relative byte address of the next ESDS record containing overflow dependent segments. If more than one ESDS record is needed to contain overflow dependent segments, they follow in sequence and chain pointers are maintained in the records.

Error message handling is accomplished in the following manner: When a routine within module DLZURUL0 requires an error message to be generated, a number is loaded into R1. This number corresponds to a message in the message CSECT (DLZRULM0). The routine then branches to a common routine which outputs the message. The number passed in R1 is multiplied by 4 and added to the start of the message CSECT (DLZRULM0). At that offset, a fullword containing the length of the message and the offset to the start of message text is obtained. These values are used to move the message to an output buffer. DLZRULM0 is a read-only module containing all error messages issued by module DLZURUL0.

Control Blocks - DLZURUL0

- Short segment table
- Output data record
- Output header record
- Statistics record.

Errcr Codes and Handling - DLZURUL0

All error codes are in the form cf error messages.

Sample Description of HISAM Reorganized Format

Assume a HISAM data base which consists of a single root segment and dependent segments in the hierarchical format shown in Figure 3-7.

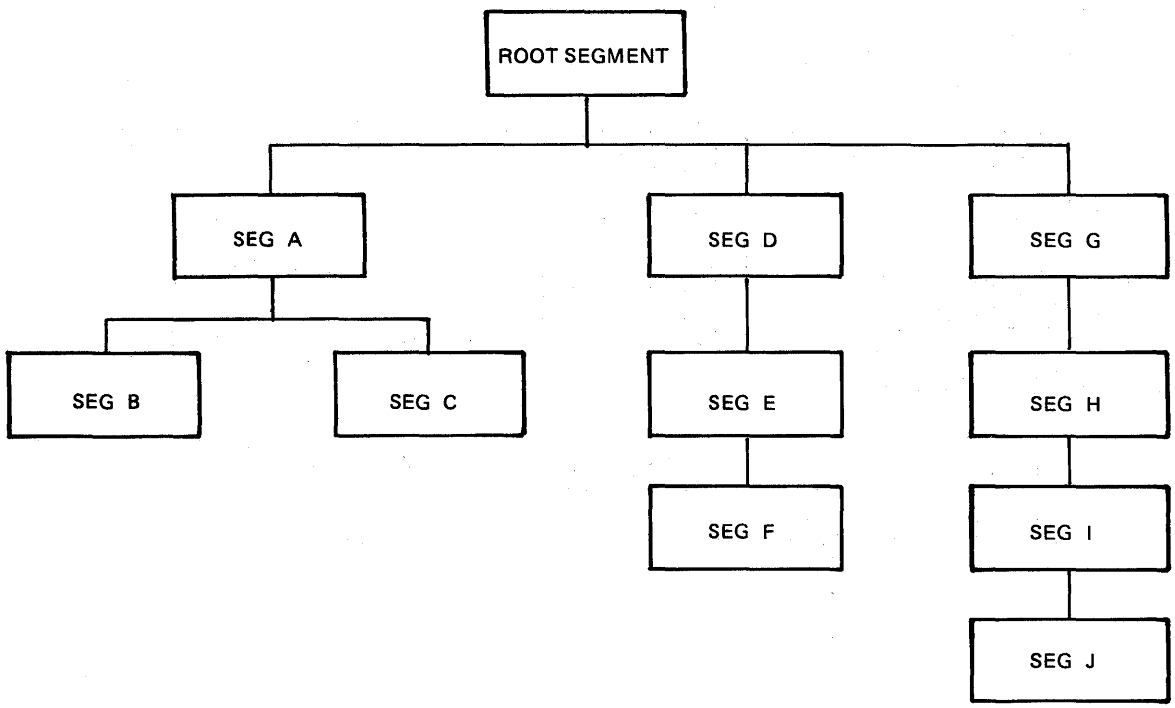


Figure 3-7. HISAM Data Base with One Root Segment

The input for the HISAM Reorganization Unload Utility appears as shown in Figure 3-8.

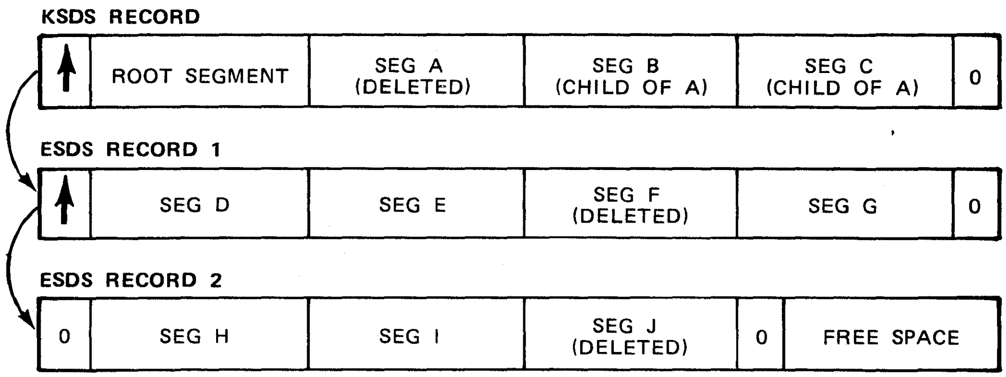


Figure 3-8. Input for HISAM Reorganization Unload Utility

Given this input, the HISAM Reorganizaticn Unload Utility provides the output shcwn in Figure 3-9.

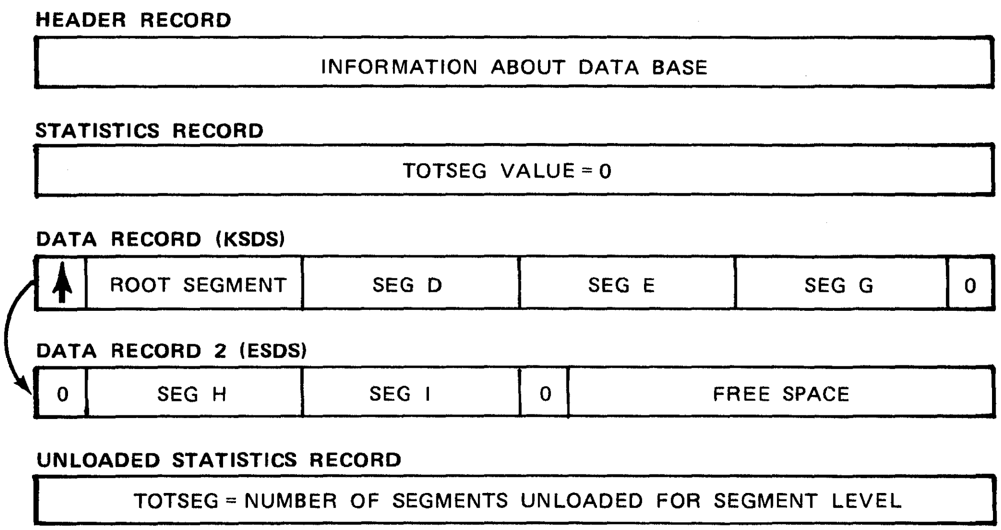


Figure 3-9. HISAM Reorganization Unload Utility Output

Note: A second ESDS record is unnecessary because space occupied by deleted segments is reclaimed.

DLZURRL0 - HS DB RELOAD

The HISAM reorganization reload module DLZURRL0 is executed as a standard DOS/VS application program and is used to reload a reorganized HISAM data base data set group. The input to the program consists of a reorganized dump of the key sequenced data set (KSDS) and entry sequenced data set (ESDS) created by the HISAM Reorganization Unload Utility program. Processing is as follows:

1. A control card, which contains the filename of the input file containing the HISAM data base to be reloaded, is read. The input file is opened and the header record is read.

2. The output KSDS and ESDS ACBs are generated using the information contained in the header record and the KSDS and ESDS are opened (only KSDS if Simple HISAM or INDEX).

3. The statistics record is read and the statistics table initialized.

4. Records are read sequentially from the input file. These records are images of KSDS and ESDS records.

5. KSDS records are written to the output KSDS using VSAM keyed sequential (mass) insert.

6. ESDS logical records are written to the output ESDS using VSAM addressed sequential insert.

7. After all data records have been processed, the last input statistics record is read, and a statistics report is printed, comparing segments unloaded/reloaded.

8. The files are closed.

All error messages issued by the HS DB reload utility are contained in module DLZRRLM0. It is a read-only module.

Control Blocks - DLZURRL0

- Header record
- Input data record

DLZURGU0 - HD DB UNLOAD

The HD reorganization unload module DLZURGU0 is executed under control of the DL/I system as an application program and is used to unload a data base by issuing DL/I calls. One or two files may be created and output may be to tape or DASD. The module contains two processing modes - "normal" and "restart".

Normal processing, after module DLZURGU0 receives control from DL/I, is as follows:

1. The PCB address is saved and a GSCD call is issued to obtain the PST address. The PST allows the program to access the DL/I control blocks needed to construct the prefix portion of the output record. This prefix, as described below, is used by the HD Reorganization Reload Utility.

2. The number of outputs (one or two) and output device type (tape or DASD) are determined.

3. Storage is obtained for the statistics table.

4. Each output file is opened.

5. The statistics tables, which have been initialized for all data base segment types, are written to the output file(s).

6. A Get Next (GN) call is issued for the first (or succeeding) segment.

7. The statistics table for the segment type is updated.

8. The segment is combined with the segment prefix to form an output logical record. The output logical records are blocked and written.

9. Whenever a checkpoint interval is reached (first root segment after 5000 segments have been processed or as specified on CHKPT parameter), a checkpoint record is written to the output file.

The current statistics are part of the checkpoint record. To insure the checkpoint record is physically written, a dummy checkpoint is also written to output. Additionally a message containing the ID of the checkpoint record is written to SYSLOG.

10. Processing continues at step 6 until end of file is encountered.

11. At end of file, the statistics table totals are written, the output file(s) is closed, and the program returns control to DL/I.

Restart processing, after module DLZURGU0 receives control from DL/I, is as follows:

1. Steps 1 - 4 of "normal processing" are performed.

2. The restart (RESTART) input file is opened. This is either the output1 (HDUNLD1) or output2 (HDUNLD2) file from the previously terminated job execution.

3. A message is issued to SYSLOG requesting the checkpoint record number (ID) at which to restart. The number is validated.

4. All records, including the requested checkpoint record, of the RESTART file are copied to the output file(s).

5. A Get Unique (GU) call is issued for the checkpointed root segment to establish positioning. If the RBA is available for the root segment, it is placed in the SSA with an internal "*T" command code; otherwise the segment's key is placed in the SSA and an internal "*C" (key retrieve) command code call is issued. The statistics table is initialized with the checkpointed statistics record.

6. Steps 6 - 11 of "normal processing" are performed.

Control Blocks - DLZURGU0

- Output record containing segment prefix
- SSA for GU call by RBA
- SSA for GU call by key
- Output table record
- Checkpoint record.

Interfaces - DLZURGU0

This module interfaces with DL/I through the DL/I language interface module DLZLI000 at entry point ASMTDLI and by fast path interface to retrieve.

Error Codes and Handling - DLZURGU0

All errors are indicated by error messages. All messages issued by the HD DB unload utility are contained in module DLZRGUM0. It is a read-only module.

DLZURGL0 - HD DB RELOAD

The HD reorganization reload utility (DLZURGL0) is loaded under DL/I control as an application program. It reloads a data base under control of DL/I. Input to the module consists of a sequential dump data set of logical records created by the HD reorganization unload utility (DLZURGU0). A logical record consists of a segment prefix and a segment.

During the reload, a message is issued each time a checkpoint record is encountered (approximately every 5000 segments or as specified by user on unload). This message is the same in content and format as that issued during unload when the checkpoint record was created, and identifies the checkpoint by number. If the reload facility fails, a restart capability called 'Reload Restart' allows restarting from a checkpoint record.

After module DLZURGL0 receives control from DL/I initialization, processing is as follows:

1. The PCB address is saved, and a GSCD call is issued to obtain the PST address.
2. The input device type is determined and the data set is opened.
3. If restarting, obtain checkpoint restart number from operator and locate checkpoint record. The data base is then positioned (GU call) and the end of data is found (GN calls).
4. An input record is read (segment), and a DL/I call list is constructed.
5. A DL/I Insert (ASRT) call is issued for the segment.
6. After all segments have been processed, the last statistics table record is read and a comparative statistics report is written.
7. The input data set is closed, and the program returns control to DL/I.

Blocks and Tables

Input record

Interfaces - DLZURGL0

This module interfaces with the DL/I routines through the DL/I language interface module DLZLI000 at entry point ASMTDLI.

Error Codes and Handling - DLZURGL0

All error conditions are indicated by error messages. All messages issued by the HD DB reload utility are contained in module DLZRGLM0. It is a read-only module.

PARTIAL DATA BASE REORGANIZATION UTILITY

DLZPRCT1 - PART 1 CONTROL

The Part 1 Control module initializes the environment for Part 1 then cotrols the order of execution for Part 1 processing.

Initially this module acquires storage for the data base table (DBT), segment table (SGT), action table (ACT), and range table (RGT). The common area (COMAREA) is part of this module and is not dynamically acquired.

Next the Part 1 Control module loads the Part 1 service modules and their entry points in COMAREA.

The final processing by this module links the Part 1 action modules to the sequence defined by the linklist table. As each linked to module returns, its return code is checked. Part 1 processing ends when the return code exceeds the maximum value allowed for that module, which is an error condition, or Part 1 successfully completes. In this case the return code is zero.

The highest return code that the Part 1 Control module encounters is the return code for the Part 1 Control processing.

Interface

This module interfaces with the following modules:

DLZPRERR - Message writer
DLZPRWFM - Work file manager
DLZPRABC - Action table build
DLZPRCLN - Cleanup
DLZPRDBD - DBD analysis
DLZPRPAR - Parameter analysis
DLZPRPSB - PSB source generator
DLZPRREP - PART1 report writer

Control blocks - DLZPRCT1

- ACT - Action table
- DBT - Data base table
- SGT - Segment table

Normal Entry Point

The only entry point to this module is DLZPRCT1.

Register Contents of Entry

Standard register conventions are used for linkage to this module.

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRABC - ACTION TABLE BUILD

This module analyzes logical relationships in the prime and related data bases. It builds entries in the action table (ACT). The action table entries indicate the necessary actions for reorganized segments and for segments that are related to reorganized segments.

Interface

This module interfaces with the following module:

DLZPRERR - Message writer

Control blocks - DLZPRABC

- CCMAREA - common area

Normal Entry Point

the only entry point to this module is DLZPRABC.

Register Contents on Entry

R8 - Addressability for ACT
R9 - Addressability for DBT
R10 - Addressability for SGT
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the sameas on entry except R15, which contains the return code.

DLZPRCLN - PART 1 CLEANUP

This module writes the tables created in part one to the communication data set for subsequent use in part two. The tables are written in the following order:

1. Common area
2. Data base table
3. Segment table
4. Range table

Control blocks - DLZPRCLN

- COMAREA - Common area

Normal Entry Point

The only entry point to this module is DLZPRCLN.

Register contents on Entry

Standard register conventions are used for linkage to this module.

R8 - Communication data set DTF
R9 - Internal linkage address
R11 - Common area
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRDBD - DBD ANALYSIS

This module analyzes the DBD that is to be used in data base partial reorganization. The module uses the characteristics of the prime and any related DBDs to build the data base table (DBT). It enters information about data sets in the dataset table3 in COMAREA. DLZPRDBD uses the characteristics of and relationships between segments in the DBDs to build the segment table (SGT).

Interface

This module interfaces with the following module:

DLZPRERR - Message writer

Control blocks - DLZPRDBD

- COMAREA - common area

Normal Entry Point

The only entry point to this module is DLZPRDBD.

Register Contents on Entry

R2 - Addressability for SGT
R3 - Addressability for TGT
R4 - Addressability for DBT
R5 - Second base register
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRPSB - PROGRAM SPECIFICATION BLOCK SOURCE GENERATOR

This module creates a PSB source deck if the partial reorganization input parameter PSB= specifies input to Part 1. Because it is not necessary to process all of the segments in the data base, a PSB source deck specifies only the sensitive segments. The information used to create this source deck is taken from the partial reorganization table created in Part 1 Control. It is the user's responsibility to run the PSBGEN and ACBGEN for this PSB prior to Part 2 Processing.

Interface

This module interfaces with the following modules:

DLZPRERR - Message writer
DLZPRWFM - Work file manager

Normal Entry Point

The only entry point to this module is DLZPRPSB.

Register Contents on Entry

R2 - Addressability for DBT
R6 - Addressability for SGT
R10 - File control block
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRREP - PART 1 REPORT WRITER

This module creates a report based on Part 1 processing for the data base that is going to be partially reorganized. The information used to create the report is extracted from the range table, data base table, and the segment table.

Interface

This module interfaces with the following module:

DLZPRWFM - Work file manager

Normal Entry Point

The only entry point to this module is DLZPRREP.

Register Contents on Entry

R2 - Addressability for RGT and SGT
R3 - Addressability for DBT
R8 - BAL register
R10 - File control block
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRCT2 - PART 2 CONTROL

This module first loads the service modules. Then it restores the common area and the tables that were built during Part 1 Control processing from the DLZPRCOM dataset. Finally, this module establishes linkage to each Part 2 phase.

Interface

This module interfaces with the following modules:

DLZPRERR - Message writer
DLZPRPAR - Parameter analysis
DLZPRUPD - Update prefix
DLZPRSTC - Sort control
DLZPRURC - Unload/reload control

Control blocks - DLZPRCT2

- COMAREA - Common area
- DBT - Data base table

Normal Entry Point

The only entry point to this module is DLZPRCT2.

Register Contents on Entry

R10 - File control block
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRPAR - PARAMETER ANALYSIS

This module analyzes input control statements and generates data in the common area (COMAREA), segment table (SGT), action table (ACT), and the range table (RGT).

Interface

This module interfaces with the following modules:

DLZPRWFM - Work file manager
DLZPRERR - Message writer

Control blocks - DLZPRPAR

- DBT - Data base table
- SGT - Segment table
- ACT - Action table

Normal Entry Point

The only entry point to this module is DLZPRPAR.

Register Contents on Entry

R1 - Parameters
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRSCC - SCAN CONTROL

This module scans segments of a data base as indicated in the data base table and action table in order to produce K records for SORT1 and T records for SORT3. K record types represent segments with unidirectional pointers to segments which may have moved during reorganization. T record types represent segments in secondary index data bases with non-unique key values from the source segment. T records are provided with a relative record number based on the number of times the key of the index value is duplicated.

Interface

This module interfaces with the following modules:

ASMTDLI - DL/I interface
DLZPRERR - Message writer
DLZPRDLI - DL/I service
DLZPRWFM - Work file manager

Normal Entry Point

The only entry point to this module is DLZPRSCC.

Register Contents on Entry

R11 - Addressability for COMAREA
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRUPD - UPDATE PREFIX

This module adds, deletes, and updates segments and indexes according to the input data work records and index work records from workfile 3 and workfile 5, respectively. This module processes each data base in physical order until all changes are complete.

Interface

This module interfaces with the following modules:

ASMTDLI - DL/I interface
DLZPRERR - Message writer
DLZPRDLI - DL/I service
DLZPRWFM - Work file manager
DLZPRSTW - Statistical writer

Normal Entry Point

The only entry point to this module is DLZPRUPD.

Register Contents on Entry

R11 - Addressability for COMAREA
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRSTC - SORT CONTROL

This module contains four routines, SORT1 through SORT4. These routines arrange data work records for prefix update (DLZPRUPD). Each routine invokes DOS/VS sort passing parameters which includes the addresses of sort exits 15 and 35. The sort exits perform the processing required by SORT1, SORT2, SORT3, and SORT4.

SORT1 and SORT2 process data work records exclusively. The input to SORT1 is from RELOAD and SCAN. The input to SORT2 is from RELOAD and SORT1. Together these routines save the new relative byte address (RBA) of the segment moved in the associated work records and arranges them in physical sequence as they exist in the data bases.

SORT3 and SORT4 process index work records exclusively. The input to SORT3 is from RELOAD and SCAN. The input to SORT4 is from the DL/I index maintenance file and SORT3. Together these routines eliminate index work records that are not involved in update, convert the DL/I index maintenance records into partial reorganization format, and arrange the index work records in physical sequence.

Interface

This module interfaces with the following modules:

DLZPRERR - Message writer
DLZPRWFM - Work file manager

Normal Entry Point

The only entry point to this module is DLZPRSTC.

Register Contents on Entry

R11 - Addressability for COMAREA
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRURC - UNLOAD/RELOAD CONTROL

This module performs the unload and reload of segments within user specified ranges. DLZPRURC frees the spaces previously occupied by the unload segments. It then inserts the segments into the user specified target area. The inserted segment's prefix carries forward the logical pointers, counters, and delete byte.

As physical changes occur in the data base during the process, this module records them on the data base log data set. DLZPRURC gathers unload and reload statistics for reports during the processing. Finally, it creates work records for update depending on actions defined in the action table for reload.

Interface

This module interfaces with the following modules:

ASMTDLI - DL/I interface
DLZPRWFM - Work file manager
DLZPRERR - Message writer
DLZPRDLI - DL/I service

Control blocks - DLZPRURC

- CCMAREA - Common area
- FCB - File control block
- DBT - Data base table
- SGT - Segment table
- ACT - Action table
- RGT - Range table

Normal Entry Point

The only entry point to this module is DLZPRURC.

Register Contents on Entry

R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRWFM - WORK FILE MANAGER

This module provides open, close, input, and output operations for VSAM and SAM files used in data base partial reorganization.

Interface

This module interfaces with the following modules:

ASMTDLI - DL/I interface
DLZPRERR - Message writer

Control blocks - DLZPRWFM

- COMAREA - Common area
- FCB - File control block

Normal Entry Point

The only entry point to this module is DLZPRWFM.

Register Contents on Entry

R6 - Addressability for XWR
R8 - Addressability for FILECB
R9 - Addressability for DWR
R10 - Addressability for DBPCB
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRDLI - DL/I SERVICES

This module is the interface with DL/I DOS/VS when the function required cannot be accomplished by any of the calls documented in the DL/I DOS/VS reference manuals. Examples of such functions are:

- Retrieval of information from DL/I DOS/VS blocks
- Direct interface with the DL/I DOS/VS buffer handler
- Direct request to log changed prefix data

To make use of this module, the caller must:

1. Complete any pre-requisite for the service needed
2. Set the code for the service needed in COMCIREQ
3. Enter this module by a BALR 14,15

Interface

This module interfaces with the following modules:

DLZDBH00 - Buffer handler
DLZPRERR - Message writer
DLZFRSP0 - Space management
DLZRDBL0 - Data base logger

Control blocks - DLZPRDLI

- COMAREA - Common area
- FCB - File control block
- DBT - Data base table
- SGT - Segment table
- ACT - Action table
- RGT - Range table

Normal Entry Point

The only entry point to this module is DLZPRDLI.

Register Contents on Entry

R3 - Addressability for DDIR, DMBDACS
R5 - Addressability for JCB
R6 - SGT, SCD, LEV, SDB, PSDB
R7 - DBT, DMB
R8 - Data base PCB
R9 - PST
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRSTW - STATISTICAL WRITER

This module is used to produce statistical reports for UNLOAD, RELOAD, and SCAN in Part 2 Control.

The report created for UNLOAD consists of range unload statistics, block range statistics, and block changes by data base record.

The report created for RELOAD consists of range reload statistics and block range statistics.

The report created for SCAN consists of a scanned segment count for each affected data base record.

Interface

This module interfaces with the following modules:

DLZPRERR - Message writer
DLZPRWFM - Work file manager

Control blocks - DLZPRSTW

- ACT - Action table
- DBT - Data base table
- SGT - Segment table
- RGT - Range table
- COMAREA - Common area

Normal Entry Point

The only entry point to this module is DLZPRSTW.

Register Contents on Entry

R1 - Parameters, File control base register
R6 - Print line base register
R7 - Addressability for ACT, RGT
R8 - Addressability for SGT
R9 - Addressability for DBT
R10 - Program base register
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

DLZPRERR - ERROR MESSAGES

This module formats and sends messages to SYSLST.

Based on the message number passed to this module by the caller, the text of the message is retrieved from the message table located in this module. If the message has a variable data field, the variable data passed by the caller is inserted in the message text.

If an invalid message number is passed by the caller, the message number is printed with text that indicates it is an invalid message number.

Control blocks - DLZPRERR

- COMAREA - Common area
- FCB - File control block

Normal Entry Point

The only entry point to this module is DLZPRERR.

Register Contents on Entry

R1 - Parameters
R3 - Addressability for SYSPRINT DCB
R5 - FCB File control block base register
R8 - Message table base register
R9 - Message buffer base register
R11 - Addressability for COMAREA
R12 - Program base register
R13 - Save area address
R14 - Return address
R15 - Entry point address

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return code.

HIGH LEVEL PROGRAM INTERFACE

DLZEIPB0 - DL/I Batch/MPS EXEC Interface

This module can be logically divided into two parts: an initialization routine with the entry point of DLZEIPI and a non-INIT call handling routine starting at label DLZEIPO.

All CICS/VS application programs which issue SL/I HLPI statements execute a translator-generated DL/I HLPI INIT call on entry to that program. This INIT call results in passing control to entry point DLZEIPI.

The CICS EXEC Interface Program, DFHEIP calls DLZEIPI, which first checks to see if this is a DL/I HLPI INIT call. If not, it passes control to the call handling routine, DLZEIPO. Initialization continues with the checking of TCA to see if storage for UIB/SDIB has been obtained. If not, DLZEIPO issues CICS GETMAIN to acquire storage for these control blocks. If storage was already acquired, DLZEIPI checks the integrity of SDIB. Following acquisition of storage for UIB and SDIB, DLZEIPI returns control to the caller.

On entry, DLZEIPO determines if the call is a data base call. If so, it determines which PCB in the PCB list to use. It also checks the following:

- If data transfer is to take place
- If segment name has been specified
- If the call is a replace call with a previous get path call

DLZEIPO then does the following:

- Acquires storage for SSA
- Establishes the correct command code
- Builds field qualifications
- Sets up the correct SSA for use by the DL/I Program Request Handler, DLZPRH00

After DLZEIPO finishes building SSA, it calculates the required I/O area size and builds a common I/O area for path calls. DLZEIPO passes control to DLZPRH00.

DLZEIPO also processes scheduling calls, termination calls, and checkpoint calls.

Interface

This module interfaces with the following modules:

DLZBNUC0 - Batch Nucleus
DLZRRC00 - Batch Initialization
DLZMPI00 - MPS Batch

Control blocks - DLZEIPB0

- DIB - User DL/I interface block
- SDIB - System DL/I interface
- EIPL - EIP parameter list
- HLPI - HLPI parameter list address
- ARGO - ARGO parameter list
- PATH - Path header control block

Normal Entry Point

The only entry point to this module is DLZEIPI.

Register Contents on Entry

R1 - HLPI parameter list address
R2 - System DIB address
R3 - ARGO parameter list address
R6 - EIP parameter list address
R8 - User DIB address
R11 - Base register
R13 - Save area address

DLZEIPB1 - BATCH/MPS EXEC INTERFACE PROGRAM

This module handles all DL/I HLPI calls except the HLPI EXEC translator generated INIT call. It performs the same functions as routine DLZEIP0 in DLZEIP00.

DLZEIPB1 passes control to DL/I Program Request Handler (DLZPRHB0) for batch of (DLZMPRH) for MPS.

There are differences between DLZEIPB1 and DLZEIP0 because of the different environments. First, DLZEIPB1 makes use of DOS/VS storage control GETVIS or FREEVIS instead of CICS/VS storage control. Secondly, DLZEIPB1 uses its own data structure DLZEIPL instead of CICS/VS TCA fields for obtaining the PCB address list.

Interface

This module interfaces with the following modules:

DLZBNUC0 - Batch Nucleus
DLZRRC00 - Batch Initialization
DLZMPI00 - MPS Batch

Control blocks - DLZEIPB0

- DIB - User DL/I interface block
- SDIB - System DL/I interface block
- EIPL - EIP parameter list
- ARG0 - ARG0 parameter list
- HLPI - HLPI parameter list address
- PATH - PATH header control block
- SSAP - PATH SSA appendage

Normal Entry Point

The only entry point to this module is DLZEIP0.

Register Contents on Entry

R1 - HLPI parameter list address
R2 - System DIB address
R3 - ARG0 parameter list address
R6 - EIP parameter list address
R13 - Register save area address
R14 - Caller's return address
R15 - Entry point address

DLZEIPO0 - DL/I ONLINE EXEC INTERFACE

DLZEIPB0 is the online interface routine that connects the user application program to the online program request handler. It performs the combined function of its batch environment counterpart DLZEIPB0 and DLZEIPB1. DLZEIPO0 builds data base calls to online program request handler according to the HLPI command syntax.

Interface

This module interfaces with the following module:

DLZODP - Online nucleus

Control blocks - DLZEIPO0

- DIB - User DL/I interface block
- SDIB - System DL/I interface block
- EIPL - EIP parameter list
- ARG0 - ARG0 parameter list
- HLPI - HLPI parameter list address
- PATH - PATH header control block
- SSAP - PATH SSA appendage

Normal Entry Point

The only entry point to this module is DLZEIP1

Register Contents on Entry

R1 - HLPI parameter list address
R7 - CICS CSA address
R13 - Register save area address
R14 - Caller's return address
R15 - Entry point address

APPLICATION CONTROL BLOCKS CREATION AND MAINTENANCE

DLZUACB0 - ACB CREATION AND MAINTENANCE

The application control blocks creation and maintenance utility creates the internal control blocks required by the DL/I application program. Using the PSB and DBDs as input, this utility creates DL/I internal format control blocks as output. These output control blocks must be link edited into the DOS/VS Core Image Library, either private or system, as specified by the user. These blocks contain information about the data bases and the programs which use them. They describe some device and media characteristics, the stored data structures, and the logical data structures as seen by both the system and application The program accepts control card input to determine what functions are required.

The logic flow is as follows: The control card input stream is processed and each card is syntax-checked. A sorted list of requested blocks is built in main storage Each PSB name specified on the control card is inserted into the list.

Each name on the constructed build list is then passed to the application control block have blocks constructed. Addresses are relocated relative to zero and the completed blocks are written to a SYSPCH or SYSLNK data set.

Blocks and Tables - DLZUACB0

Program control parameter block
PST
SCD
PDIR

Interfaces - DLZUACB0

This module interfaces with the following modules:

DLZUSCH0 - Called to create and search sorted PSB lists
DLZLBLM0 - Called to format prebuilt messages
DLZDLBL0 - Called to build and output control blocks for a PSB

Register Contents

R0-R1	= PARM registers
R2-R8	= Work registers
R9	= Pointer to PST
R10-R11	= Work registers
R13	= Pointer to save area and primary base register
R14-R15	= Operating system linkage registers

DLZUSCH0 - ACB MAINTENANCE BINARY SEARCH/INSERT

The function of module DLZUSCH0 is to create and search sorted lists in dynamic (GETVIS) storage using the binary search technique. Any number of lists may be created simultaneously (subject only to the limit of available storage). A list entry may be any length from 1 to 256 bytes. The key or sequence field may also be from 1 to 256 bytes in length and may be located anywhere in the list entry. The only restriction on keys is that they must consist of a single contiguous string of bytes within the list entry.

The number of entries in any list is limited only by available storage. However, since this routine physically moves data in storage to make room for new entries, it becomes less efficient as the number of entries increases. For large numbers of items, it might be best to consider sorting the entries in the conventional fashion.

This module is called by DLZUACB0 to build and maintain the list of PSBs to be processed.

Operation

I. The following interface is used to initiate a new list:

```
L 15,=V(DLZUSCH0)
LA 1,PARMS
BALR 14,15
```

where PARMS is a 3-word list whose contents are as follows:

```
Word 1 = length of the list entry
Word 2 = offset from the beginning of the list
         entry to the key/sequence field
Word 3 = length of the key/sequence field
```

On return, register 1 contains the location of the new list control block. (This location must be submitted to the search routine on all subsequent search or insert calls for this list.)

II. The following interface is used to insert an entry into a list:

```
L 15,=V(INSRCH)
LA 1,INPARMS
BALR 14,15
```

where INPARMS is the location of a two-word list whose contents are:

```
Word 1 = address of the list control block
Word 2 = address of the list entry to be
         inserted
```

On return from INSRCH, register 15 contains zero if the entry was successfully inserted, and register 1 contains the location at which the insert was made.

If the entry was not inserted (because a duplicate was found), register 15 contains 8, and register 1 contains the location of the duplicate entry.

III. The following interface is used to locate an entry in a list created by INSRCH:

```
L 15,=V(LOCSRCH)
LA 1,LOCPARMS
BALR 14,15
```

where LOCPARMS is the location of a two-word list whose contents are:

```
Word 1 = address of the list control block
Word 2 = address of the search argument (key)
```

On return from LOCSRCH, register 15 contains zero if an entry containing the search argument in its key field was found, and register 1 contains the location of this entry.

If no entry was found, Register 15 contains 4 and register 1 remains as it was on entry to LOCSRCH.

IV. The following interface is used to delete all storage obtained by OPENSRCH and INSRCH for a given list:

```
L 15,=V(CLOSESCH)
L 1,LOCPARMS
BALR 14,15
```

where LOCPARMS contains the location of the list control block for the list to be deleted.

Control Blocks - DLZUSCH0

- List control block
- Sorted list block.

Programming Note

If some number of entries have been placed in a list through repeated calls to INSRCH, they can be retrieved in sorted order by locating the first block by way of CHAINLOC and all subsequent blocks by way of their CHAIN fields. The entries are in order (low to high logical sequence) with the lowest entry in block 1 entry 1, next in block 1 entry 2, etc., with the highest entry located in the last-used slot in the last block.

DLZLBLM0 - ACB Generation Error Message Handler

This module is used to contain, select, and format error messages for the ACB generation facility. Given a message number in register one, the module will select the matching message and format it by inserting an arbitrary number of additional character strings addressed by specified registers. The 'PRTMSG' routine in module DLZUACB0 is called to print the message. Control is returned to the caller.

Register Contents on Entry - DLZLBLM0

```
R1  - Message number
R13 - Save area
R14 - Return address
R15 - Entry point
```

Additionally, any registers are passed that have been defined to contain pointers to character strings to be inserted into the message. These are generally (but not always) registers 5, 6, and 7.

External Routines Called - DLZLBLM0

PRTMSG - Entry point to the print routine in module DLZUACB0.

DLZDLBL0, DLZDLBL1, DLZDLBL2, DLZDLBL3 - ACB BUILDER

These four modules are jointly responsible for building all the control blocks for a given PSB and its associated DBDs, and for outputting them to either SYSPCH or SYSLNK in a format that allows LINKing them into the DOS/VS core image library.

The first module, DLZDLBL0, loads the specified PSB and builds the PCBs and SDBs for segments identified via SENSEG statements at PSBGEN time. It then passes control to module DLZDLBL1.

Module DLZDLBL1 loads the DBDs for all referenced data bases and builds the associated DMBs (for all but logical DBDs). It then processes the SDBs associated with each DBD, copying any required information from the physical definitions and building any required generated SDBs. Control is given to module DLZDLBL2 when all DBDs have been processed.

Module DLZDLBL2 finishes the processing of the SDBs. It acquires and builds the intent list, including propagation of intent, and initializes any field level sensitivity control blocks required. The PCB is moved to its proper location and the JCB, level table, and DSGs are built. Control is passed to module DLZDLBL3.

The last module, DLZDLBL3, builds the index maintenance PCB if one is required, performs some additional clean-up, and packages and outputs the DMBs and the PSB to either SYSLNK or SYSPCH. If a utility PSB is required, module DLZDPSB0 is called to build it, and module DLZDLBL0 is re-called at entry PSBPASS to initialize it.

Interfaces - DLZDLBL0 - DLZDLBL3

These modules interface with the following modules:

DLZDPSB0	-	Called to build a utility PSB
DLZLBLM0	-	Called to format and write error message

Register Contents on Entry

R1	-	PST address
R13	-	Save area address
R14	-	Return address
R15	-	Entry point address

Register Contents on Exit

All registers are restored. The return code appears in PSTERCOD of the PST.

PSTERCOD = 0	Valid return
PSTERCOD ≠ 0	Errors encountered

DLZDPSB0 - UTILITY PSB BUILDER

This module is called by the application control blocks builder module (DLZDLBL0) to dynamically construct a special utility PSB from a specific DBD. The created PSB is in PSBGEN format. A GETVIS is issued to obtain storage necessary to create the PSB. The created PSB is sensitive to all segments for the data base.

Register Content on Entry

R1 - Address of parameter list
R13 - Save area address
R14 - Return address of DLZDLBL0
R15 - Entry point

The parameter list consists of a DBD address and a PSB address.

Registers on Exit

All registers are restored except R15 which contains a return code passed to DLZDLBL0.

R15 = 0	Valid return
R15 ≠ 0	Errors encountered

DLZURPRO - PREREORGANIZATION

The purpose of this module is to examine input control cards provided by the user, and, based upon the information contained in DL/I control blocks, to generate a control data set for use by other programs concerned with the resolution of logical and index relationships.

The input control cards for this program indicate the names of data bases that a user wishes to initially load or to reorganize. The control blocks for each segment of each data base listed on an input control card are examined. For each logical relationship in which a segment participates, a prefix resolution check is performed. This check consists of generating a bit map reflecting the prefix fields involved in the logical relationship, and then checking the bit map against a table that indicates the fields which must be resolved for the types of data bases in which the logical parent and the logical child reside. For purposes of the prefix resolution check, the type of data base is considered to mean an initially loaded data base, a reorganized data base, or another data base (not reorganized or loaded, but logically related to a data base that is reorganized or loaded). If the bit map and the table entry match yields a nonzero value, prefix fields must be resolved in either or both the logical parent and logical child.

If prefix fields must be resolved, a control list entry is built for the logical parent and/or the logical child. This control list entry indicates the fields to be resolved, the work data set record format options to use, etc. As control data set list entries are built, each record is calculated to determine a maximum record length. The largest size is saved and put into field LESRTSZE when the control data set is written. The prefix resolution utility (DLZURG10) reads this value and passes it to SORT.

After generating the control list, the data bases to be scanned, loaded, or reorganized are listed. The scan list is punched if requested. The control list is then written to the control data set.

Control Blocks - DLZURPRO

- Control file consisting of one or more records, each with a pointer to the next block of control file and an area containing one or more control list entries.
- List entry.
- Secondary list entry.

Interfaces - DLZURPRO

The interface with the reorganization message module (DLZURGM0) is through the tables provided in that module. See the description of that module for table format.

The interface with batch initialization to load the required blocks dynamically is accomplished with the DLZBLKLD macro.

Error Codes and Handling - DLZURPR0

This program audits all input control cards and verifies the consistency of DL/I control blocks. Any errors encountered cause one or more messages to be generated. Refer to DL/I DOS/VS Messages and Codes for details.

DLZURGS0 - DB SCAN

This module searches one or more data bases for all segments that are involved in logical relationships. For each such segment, DLZURGS0 generates one or more output records, depending upon the relationships in which that segment is involved. The output work data set of this program serves as one of the inputs to the prefix resolution utility.

This program scans data bases as indicated either by scan control cards or by the control data set generated by the prereorganization program. If scan control cards are present, they are checked for consistency with the DL/I control blocks. Data base scanning is done by segment type for HDAM and HIDAM data bases. If scan control cards are provided for segments in an HDAM or a HIDAM data base, work data set records are generated only for those segments listed on scan control cards.

After the segments are read into core, control is passed to the work data set generator module (DLZDSEH0). DLZDSEH0 generates any necessary output work data set records based upon information contained in the control data set. It then returns control to this program (DLZURGS0).

Interfaces - DLZURGS0

Module DLZURGS0 interfaces with the reorganization message module (DLZURGM0) through the tables provided in that module. See the description of that module for table format.

The interface with the work data set generator module (DLZDSEH0) is as described in the documentation for that module.

The interface with the buffer handler module (DLZDBH00) is as described in the documentation for that module. The buffer handler module is used to directly access records in a data base.

The interface with batch initialization to load the required blocks needed for processing is accomplished with the DLZBLKLD macro.

Error Codes and Handling - DLZURGS0

This program audits all input control cards and verifies the consistency of DL/I control blocks with the control data set. Any errors encountered cause one or more messages to be generated. Refer to DL/I DOS/VS Messages and Codes.

ABENDs - DLZURGS0

If an input card is read with "ABEND" in columns 1-5, a dump (PDUMP) will be taken if an error condition is detected. This should always be done on a rerun of this utility if an APAR is to be submitted because of an error return code.

DLZDSEH0 - WORKFILE GENERATOR

This module generates the work file records that are required to resolve logical and/or index relationships after one or more data bases have been initially loaded or reorganized. This program is used by the HD reload (DLZURGL0) and scan (DLZURGS0) utility programs provided by DL/I DOS/VS. It is also called automatically by internal DL/I modules (DLZDDLE0 and DLZDXMT0) when a data base is initially loaded by a user-written program.

The general operation of this program consists of creating one or more work file records for each segment that is initially loaded, reloaded, or scanned, if that segment is involved in at least one logical or index relationship. The work file records reflect the new location of each segment and, if the data base is being reloaded, its old location. Each work file record also contains related information that indicates the data bases and segments involved in the logical or index relationship described by the record, their old pointer values, etc.

This program generates all work file records that are used as input by the data base prefix resolution module (DLZURG10). The format of each output record generated by this program (DLZDSEH0) is as described for input of the data base prefix resolution module (DLZURG10).

This module contains a CSECT which is also used by scan (DLZURGS0) and index maintenance (DLZDXMT0) to open the work file DTF. Within this routine is a subroutine (FINDDTF) which is also used by scan to determine the correct DTF (disk or tape) to use for a given file depending on the assignment for it.

DLZDSEH0 is loaded by batch initialization when the PROCOPT is 'load' or when HD reload or scan are to be executed. The primary entry point address is found in SCDDSEH0. The DL/I termination routine will close the work data set.

Interfaces - DLZDSEH0

The first seven fullwords of the CSECT contain information to be used by the modules which interface with DLZDSEH0. These words concern the work data set and entry points or addresses needed by scan (DLZURGS0).

Displ. from Entry Point DLZDSEH0	Contents
-28	Base address of this module
-24	Address of LPLCSV - information needed by scan
-20	Address of TEST - entry point when called by scan
-16	Address of FINDDTF - a subroutine used by scan
-12	Address of OPENWORK - entry point of routine to open WORKFIL file

-8	Address of work area available to build output record
-4	Address of opened work file DTF. If this field is zero, the file is not open.

- When invoked during initial data base load or during data base reorganization, the following interface is used:

Entry Point

DLZBEGIN (Address found in SCDDSEHO)

Register Contents

R1 - PST
R13 - Save area
R14 - Return address
R15 - Entry point address

Control Blocks

JCBPRESF - Operation type (FUNCASRT or FUNCISRT)
PSTWRK1 - SDB address

Exit

Return to calling program with a return code in register 15. The values are:

0 (X'0')	=	Successful completion
4 (X'4')	=	WORKFIL could not be opened (IGN was specified). This is not an error condition if the user does not wish to create a work file.
8 (X'8')	=	Sort field size exceeded
12 (X'C')	=	GETVIS error occurred
16 (X'10')	=	Invalid DL/I control blocks
20 (X'14')	=	Length of PCB key feedback area is zero
24 (X'18')	=	I/O error occurred on WORKFIL or CONTROL data set.
28 (X'1C')	=	CONTROL or WORKFIL data set could not be opened (invalid or unassigned device)

- When the OPENWORK routine is called by scan (DLZURGS0) or index maintenance (DLZDXMT0), the following interface is used:

Entry Point

OPENWORK

Register Contents

R13 - Caller's save area address
R14 - Return address
R15 - Entry point address.

Exit

All registers are restored to entry condition. Return is made to the address in R14 plus the displacement 0 if an unknown or invalid device is specified or 4 if WORKFIL is successfully opened.

- When invoked during a data base scan, the following interface is used:

Entry Point

TEST

Register Contents

```
R3   -   Location for prefix parameter list area for segment just read
R5   -   Secondary list entry
R6   -   PSDB
R7   -   SDB
R9   -   PCB
R10  -   PST
R11  -   Location of DTF for work data set (must be open)
R12  -   Base address for DLZDSEHO
R13  -   Save area for use by DLZDSEHO
R15  -   Entry point TEST
```

Control Blocks

```
PSTWRK1 Byte 0     - Operation type (FUNCIHPS)
        Byte 1-3 -  SDB address
```

Exit

Return to calling program with return code in register 15 as for entry point DLZBEGIN.

- When the FINDDTF routine is invoked by scan, the following interface is used:

Entry Point

FINDDTF

Register Contents

```
R0   -   System logical unit number in hex
R2   -   Address of disk DTF
R3   -   Address of tape DTF (or 0, if not an option)
R13  -   Caller's save area address
R14  -   Return address
R15  -   Entry point of FINDDTF
```

Exit

Register 15 - address of chosen DTF

All other registers are restored to entry conditions. Return is made to the address in R14 plus the displacement 0 if an unknown or invalid device specified or 4 if successful completion. When error return to R14+0 is made, R15 is zero if IGN was specified, or nonzero otherwise.

DLZURG10 - PREFIX RESOLUTION

This module accumulates the information generated on work data sets during the load and/or reorganization of one or more data bases. It produces an output data set that contains the prefix information needed to complete the logical and/or index relationships defined for the data base(s).

Operation of this program centers around at least one and possibly two, phases of the DOS Sort/Merge program execution. In the first phase, the Sort/Merge program is attached by this program. All work data set records generated during data base initial load, reorganization, or scan are input to the sort program. All input records are sorted such that all work data set records associated with a given occurrence of a logical parent follow the work data set record describing that logical parent. On exit from the first phase sort, this program has available the information needed to resolve the logical parent pointers that reside in logical children, the counter field and logical child pointers in the logical parent, and the logical twin pointers in the logical child (if a sequence field is carried in the work data set record). Any unnecessary records are dropped before entering the second sort phase. The second phase of this program is not executed if only index relationships need to be resolved.

In the second phase of this program, the Sort/Merge program is again attached. In this sort execution, the output records from phase one are sorted according to data base name and physical location within data base of each segment that must be updated by the prefix update program. On exit from the second phase sort, any remaining logical twin pointers are resolved, and further accumulation of logical parent counter fields is performed. Any records not actually necessary to update a data base are dropped at this time.

This program uses the control data set generated by the prereorganization program to govern its general operation. That is, the lists in the control data set indicate prefix fields to be resolved, etc. The pre-reorganization utility also calculates the maximum record length for SORT records and stores the size in the control data set (LESRTSZE). The prefix resolution utility reads this value and passes it to SORT.

Control Blocks - DLZURG10

- Input work file record - DLZURWF1
- Output work file record - DLZURWF3

Error Codes and Handling - DLZURG10

This program audits all input work data set records for consistency and for correspondence with the control list provided with the control data set. Any errors encountered cause one or more messages to be generated. Refer to the DL/I DOS/VS Messages and Codes

DLZURGP0 - PREFIX UPDATE

This module reads the input work data set provided by the data base prefix resolution module, reads the data base segment indicated by each record of the input work data set, and applies the prefix changes indicated by the work data set record to the segment read into main storage.

The input work data set is sorted in data base and segment physical location order by the data base prefix resolution module (DFSURG10) to afford most efficient update of each data base by this module. The format of each input record read by this program is as described for output of the data base prefix resolution module.

One or more input work data set records may be present for each segment that participates in logical or index relationships. The records are successively applied to the prefix of each segment affected, and the updated segment is written to its storage device. The prefix fields updated by this program include the logical parent, logical twin, and logical child pointer fields, and the counter fields associated with logical parents.

Interfaces - DLZURGP0

The interface with the reorganization message module (DLZURGM0) is through the tables provided in that module. See the description of that module for table format.

The interface with the language interface module (DLZLI000) is as described in the documentation for that module. The DL/I "ISRT" and "GHU" calls are issued by this program.

The interface with the buffer handler module (DLZDBH00) is as described in the documentation for that module. The buffer handler module is used to directly access records in a data base.

The interface with batch initialization to load the required blocks dynamically is accomplished with the DLZBLKLD macro.

Error Codes and Handling - DLZURGP0

This program audits all input work data set records for consistency with data base control blocks, checks all data base update operations, and checks input control card information. Any errors encountered cause one or more messages to be generated. Refer to the DL/I DOS/VS Messages and Codes.

DLZURGM0 - DB REORGANIZATION MESSAGE

This module contains messages used by the following utilities: preorganization (DLZURPR0), scan (DLZURGS0), prefix resolution (DLZURG10), and prefix update (DLZURGP0). The module consists of the two tables defined below.

Control Blocks - DLZURGM0

1. Message Length and Offset Table

   One 4-byte table entry exists for each message. Each 4-byte entry contains the message length and offset.

2. Message Table

   One variable-length entry is present for each message. Each entry contains the text of the message. The length is found in the message length and offset table.

Interfaces - DLZURGM0

This module contains messages that are used by the following modules:

DLZURPR0 (prereorganization)
DLZURGS0 (scan)
DLZURG10 (prefix resolution)
DLZURGP0 (prefix update)

TRACE PRINT UTILITY

DLZTPRT0 - TRACE PRINT UTILITY

The Trace Print Utility is used to format and print trace entries previously written to a tape or disk by the CICS/VS extra partition dataset facility. The format of the output records on SYSLST is the same as those written directly to SYSLST by the Trace Facility. Trace Print Utility processing is as follows:

1. The utility opens the reader (SYSIN), printer (SYSLST), and console log (SYSLOG).

2. A read is issued to SYSIN, looking for a TI statement. If present, the fields on the statement are validated and saved. Further reads are issued to SYSIN until EOF is returned. All statements read from SYSIN are recorded on SYSLST .

3. When End-of-File is reached on SYSIN, the reader is closed.

4. A GETVIS is issued to acquire sufficient storage for two trace input buffers. The buffer size will either be the default of 32763 bytes, or the size specified on the TI statement.

5. The device assigned for trace input is then checked by the DLZDVCE macro routine. If the device is a valid tape or disk, the corresponding DTF is modified and the file opened for input.

6. Trace records are then read from the input file until End-of-File is returned.

7. Trace entries are processed from the input buffer one at a time until all of the entries in the record are printed. If selective output was specified by using a TO statement, each entry is checked against the desired selection. If the entry passes the selection test, it is printed. If it does not pass the test, it is ignored. When the last entry of the record is processed, control is returned to the read routine.

8. Any errors detected will be written to SYSLST and/or SYSLOG. If no errors are detected, a message indicating successful completion is written.

DL/I RUN AND BUFFER STATISTICS

DLZSTTL - DL/I Run and Buffer Statistics

The run and buffer statistics function captures online (including MPS) DL/I system statistics and writes them to the extra-partition CSSL. This data is cumulative for the current invocation of CICS/VS and automatically printed during CICS/VS shutdown.

Interfaces

This module interfaces with the following modules:

CSAPCNAC - CICS/VS program control routine
CSASCNAC - CICS/VS storage control routine
CSATDNAC - CICS/VS transient data control routine

Control blocks - DLZPRCT1

- CICS/VS - CSA
- CICS/VS - TCA
- DL/I - SCD
- DL/I - BFFL
- DL/I - SBIF

Normal Entry Point

The only entry point to this module is DLZPRCT1.

Register Contents on Entry

R1 - RPL address
R2 - STTLPUT subroutine linkage
R3 - STTLCNFG loop control
R5 - DLZSBIF base register
R6 - DLZBFPL base register
R8 - DFHTCTTE base register
R9 - DFHTIOA base register
R10 - DFHTDOA base register
R11 - DLSSTTL base register
R12 - DFHTCADS base register
R13 - DFHCSADS base register
R14 - External link

Register Contents on Exit

All registers are the same as on entry except R15, which contains the return address.

This table gives the following information for all DL/I DOS/VS modules:

- CORE IMAGE LIBRARY

  The name of the DL/I DOS/VS phase residing in the core image library.

- CSECT(S)/ENTRY POINT(S)

  The CSECTs that comprise each PHASE. Any indented name under a CSECT is an entry point within that CSECT. If the indented name is preceded by '*', it designates a routine within the CSECT and may, or may not, appear on the link-edit map. Unreferenced entry points have been omitted.

- RELOCATABLE LIBRARY

  The name(s) of the module(s) in the relocatable library that are needed for linkage editing.

- SOURCE LIBRARY

  The name(s) of the module(s) in the source statement library. For each module, source code listings are available on microfiche (under the module name).

- STORAGE ID

  The storage ID for the applicable modules. This is located near the beginning address of each module and is usually followed by the version, release level, and latest PTF number applied.

- SUPPLEMENTARY INFORMATION

  The entry SVA means the module concerned is eligible to be loaded into the shared virtual area (SVA). Any other entry in this column is the entry point name that must be present on the END card when assembling this module, for example, END DLZBEGIN. • FIGURE REFERENCE

  The figure number shown after the module name refers to the figure number of the module's HIPO diagram in Section 2: Method of Operation, Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) Logic Manual, Volume 2, Order No. Ly24-5215.

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
SYSTEM CONTROL MODULES					
** Batch Initialization ** (See Figure 2-3)					
DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRCST
	*ERRORMSG				
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
	DLZRDR				
	DLZCONSL				
	DLZRRC10				
	*DLZRRA00				
	*DLZPCC00				
	*DLZDBLM0				
	*LOADDMBS				
	*PCBROUT				
	*DLZCPI00				
	*DMBLOADR				
** Batch Nucleus ** (See Figure 2-4)					
DLZBNUC0	SCDCSECT	DLZBNUC0	DLZBNUC0	DLZBNUC0	
	SCDSTART				
	*DLZIWAIT				
	*DLZPRHB0				
	*DLZABEND				
	*DLZEIP1	DLZEIPB0	DLZEIPB0	DLZEIPB0	
** Online Initialization ** (See Figure 2-5)					
DFHSIDL	DLZOLI00	DLZOLI00	DLZOLI00	DLZOLI00	
	*DLZCPI00		DLZOLI00	DLZOLI00	
** Online Nucleus ** (See Figure 2-6)					
	DLZEIPO0				
DLZNUCxx	DLZODP	DLZODP	DLZODP	DLZNUCxx	DLZODP
	DLZODP00				
	DLZSCHDL				
	DLZODP03				
	DLZODP02			DLZODP02	
	DLZODP04			DLZODP04	
	DLZODP07				
	DLZODP06				
	DLZODP01			DLZODP01	
	DLZTKTRM				
	*DLZTKBAD				
	*TRMSUSPA				
	DLZODP05			DLZODP05	
	DLZPRHO0			DLZPRHO0	
	DLZABND0				
	DLZOLT00			DLZOLT00	
	DLZOLT02				
	DLZOLT01				
	DLZOWAIT			DLZOWAIT	
	DLZOVSEX			DLZOVSEX	
	DLZERMSG			DLZERMSG	
	DLZODP10			DLZODP10	
	DLZODP11			DLZODP11	
	DLZEIPI	DLZEIPO0	DLZEIPO0	DLZEIPO0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	DLZSTRO0	DLZSTRO0	DLZSTRO0	DLZSTRO0	
	DLZCOM00	DLZCOM00	DLZCOM00	DLZCOM))	
	DLZCOM01				
	DLZLOC00	DLZLOC00	DLZLOC00	DLZLOC00	
	DLZLOC01				
	DLZODPEX			DLZODPEX	
	DLZNUC				
	SCDSTART				
	DLZEIPL				
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
	DLZFTDP0	DLZFTDP0	DLZFTDP0	DLZFTDP0	
	DLZISC00	DLZISC00	DLZISC00	DLZISC00	
	DLZISC02				
	DLZISC01				
	DLZISC03				

Note: xx is the result of ACT generation.

** DL/I Online System Termination ** (See Figure 2-7)

DLZSTP00	DLZSTP00	DLZSTP00	DLZSTP00		

DL/I FACILITY MODULES

** Call Analyzer ** (See Figure 2-8)

DLZDLA00	DLZDLA00	DLZDLA00	DLZDLA00	DLZDLA00	SVA DLZEPDLA
	DLZDLA01	DLZDLA01	DLZDLA01	DLZDLA01	

** Retrieve ** (See Figure 2-9)

DLZDLR00	DLZDLR00	DLZDLRA0	DLZDLRA0	DLZDLRA0	SVA
	DLZDLR10				
	DLZRETN0				
	DLZEODC0				
	DLZGERC0				
	DLZGER0				
	DLZGETS0				
	DLZCLRP0	DLZDLRB0	DLZDLRB0	DLZDLRB0	
	DLZWIPE0				
	DLZMOVA0				
	DLZMOVB0				
	DLZDELT0				
	DLZPSDB0				
	DLZHUNT0				
	DLZSETL0				
	DLZBH0				
	DLZSSDB0				
	DLZNOOP0				
	DLZCONC0				
	DLZSSA0	DLZDLRC0	DLZDLRC0	DLZDLRC0	
	DLZTAG0				
	DLZLTW0				
	DLZNOSS0				
	DLZHIDA0	DLZDLRE0	DLZDLRE0	DLZDLRE0	
	DLZHDAM0				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	DLZHISA0				
	DLZSTLA0				
	DLZSTLG0				
	DLZUPDT0				
	DLZKDTE0				
	DLZPCHK0				
	DLZISRT0	DLZDLRF0	DLZDLRF0	DLZDLRF0	
	DLZVLRT0				
	DLZAREJ0				
	DLZVLCH0				
	DLZXDFT0	DLZFLD0			
	DLZHSAM0				
	DLZALTS0				
	DLZLOGR0	DLZDLRD0	DLZDLRD0	DLZDLRD0	
	DLZRETK0				
	DLZRETI0				
	DLZKDRK0				
	DLZKDTL0				
	DLZUPDC0				
	DLZUPDL0				
	DLZAPST0				
	DLZYENT0				
	DLZYSTC0				
	DLZYEND0				
	DLZDEQ0				
	DLZPOST0	DLZDLRG0	DLZDLRG0	DLZDLRG0	
	DLZSKPG0				
	DLZSKPS0				
	DLZSKPD0				
	DLZSKPE0				
	DLZRLNKD	DLZRLNKD	DLZRLNKD	DLZRLNKD	

** Load/Insert ** (See Figure 2-10)

DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	SVA
	HDROUTIN				
	HSROUTIN				

** Delete/Replace ** (See Figure 2-11)

DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	SVA DELREPEP
	DLZDLDS0				
	DLZDLDD0				
	DLZDLDA0				
	DLZDLDR0				

** Index Maintenance ** (See Figure 2-12)

DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	SVA DLZDXMT0

** HD Space Management ** (See Figure 2-13)

DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	SVA
	DLZGGSPC	DLZGGSP0	DLZGGSP0	DLZGGSP0	
	DLZRRTRN				
	DLZFRSPC	DLZFRSP0	DLZFRSP0	DLZFRSP0	
	DLZLLCLC	DLZLLCL0	DLZLLCL0	DLZLLCL0	
	DLZMMLCT	DLZMMLC0	DLZMMLC0	DLZMMLC0	
	DLZRRHPL	DLZRCHP0	DLZRCHP0	DLZRCHP0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	DLZRCHBK	DLZRCHB0	DLZRCHB0	DLZRCHB0	
	DLZRCBK2				
	DLZMMUDT	DLZMMUD0	DLZMMUD0	DLZMMUD0	
	DLZMMOFF				
	DLZMMON				
	DLZRRHMP	DLZRRHM0	DLZRRHM0	DLZRRHM0	
	DFSRLO30	DLZDHDS0	DLZDHD00		
	*SNAPDCB				
	*SNPSW				
	*SNPCNT				
	DLZDCI00	DLZDCI00	DLZDCI00	DLZDCI00	

** Open/Close ** (See Figure 2-14)

DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	

** DB Buffer Handler ** (See Figure 2-15)

DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	SVA
	DLZEBH00				DLZEBH00
	*MAINROUT				
	ROULINK				
	*PREPENQ				
	*PREPDEQ				
	*ABEXIT				
	*BOTTOUSE				
	*ALLDEQ				
	*BFFERREL				
	*RETURN				
	DLZDBH02	DLZDBH02	DLZDBH02	DLZDBH02	
	*WRITE				
	*READ				
	*HSREAD				
	*HSWRITE				
	*LOWRITE				
	*PUTKY				
	*MSPUT				
	*STLEQ				
	*STLBG				
	*GETNX				
	DETIOERR				
	*TSTPST1				
	DLZDBH03	DLZDBH03	DLZDBH03	DLZDBH03	
	*ENQ				
	*DEQ				
	*CONVADNR				
	*MRKEMPT				
	*PGUSR				
	*CONVNARD				

** DB Logger ** (See Figure 2-16)

DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0
	DLZIDBL0				
	IOFILA1				
	LOGOUT				
	LSCDADDR				
	IJFUZZZN	IJFUZZZN			
	IJFUZZZZ				
	IJ2N00nn				
	ONLLOGWR	DLZRDBL0	DLZRDBL0		

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
(DLZRDBL0)	SAVE				
	PRIVECB				
** CICS/VS Journal Logger ** (See Figure 2-17)					
DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1
	DLZIDBL0				
** Queuing Facility ** (See Figure 2-23)					
DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0
DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW
** Field Level Sensitivity Copy ** (See Figure 2-40)					
DLZCPY10	DLZCPY10	DLZCPY10	DLZCPY10	DLZCPY10	SVA
	DLZSEGCV			DLZSEGCV	

MPS CONTROL MODULES

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
** Start Transaction ** (See Figure 2-18)					
DLZMSTR0	DLZMSTR0	DLZMSTR0	DLZMSTR0	DLZMSTR0	
** Master Partition Controller ** (See Figure 2-19)					
DLZMPC00	DLZMPC00	DLZMPC00	DLZMPC00	DLZMPC00	
** Batch Partition Controller ** (See Figure 2-20)					
DLZBPC00	DLZBPC00	DLZBPC00	DLZBPC00	DLZBPC00	
	DLZLI000				
** MPS Batch ** (See Figure 2-21)					
DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	DLZMINIT
	*DLZMPRH				
	DLZMINIT				
	*DLZMTERM				
	*DLZMMSG				
	*DLZMABND				
	DLZCONSL				
	DLZDIMOD				
	*DLZEIPI	DLZEIPB0	DLZEIPB0	DLZEIPB0	
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
** Stop Transaction ** (See Figure 2-22)					
DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	

DATA BASE RECOVERY UTILITIES

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
** DB Data Set Image Copy ** (See Figure 2-25)					
DLZUDMP0	DLZUDMP0	DLZUDMP0	DLZUDMP0	DLZUDMP0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	IJ2Mnnnn	DLZUDMP0	DLZUDMP0		
	DLZDMPM0	DLZDMPM0	DLZDMPM0		
	IJJFCBZD	IJJFCBZD			
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				

** DB Change Accumulation ** (See Figure 2-26)

DLZUCUM0	DLZUCUM0	DLZUCUM0	DLZUCUM0	DLZUCUM0	
	DLZERRTN				
	DLZUSPKL				
	DLZWORK#				
	DLZPRNT				
	DLZSLOG				
	DLZUCONS				
	DLZUCCT0	DLZUCCT0	DLZUCCT0	DLZUCCT0	
	DLZUC150	DLZUC150	DLZUC150	DLZUC150	
	DLZUEX15				
	DLZUC350	DLZUC350	DLZUC350	DLZUC350	
	DLZUEX35				
	DLZUCER0	DLZUCER0	DLZUCER0	DLZUCER0	
	DLZCUMM0	DLZCUMM0	DLZCUMM0	DLZCUMM0	
	IJFSZZWN	IJFSZZWN			
	IJFVZZWZ				
	IJFSZZWZ				
	IJGQICZZ	IJGQICZZ			
	IJGQIZZZ				
	IJGQOCZZ	IJGQOCZZ			
	IJGQOZZZ				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJ2Mnnnn	DLZUCUM0	DLZUCUM0		
	IJFUZZZZ	IJFUZZZZ			
	IJGUIZZZ	IJGUIZZZ			

** DB Data Set Recovery ** (See Figure 2-27)

DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0
	DLZURCC0	DLZURCC0	DLZURCC0	DLZURCC0	DLZURCC0
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CDLTDLI				
	DLZRDBM0	DLZRDBM0	DLZRDBM0	DLZRDBM0	
	IJJFCBID	IJJFCBID			
	IJJFCBZD				
	IJJFCIID				
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJ2Mnnnn	DLZURDB0	DLZURBD0		
	IJFUZZZN	IJFUZZZN			
	IJGUICZZ	IJGUICZZ			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				

** DB Change Backout ** (See Figure 2-28)

DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0	
	READAREA				
	IJ2Mnnnn				
	DLZRDBC0	DLZRDBC0	DLZRDBC0	DLZRDBC0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	DLZBACM0	DLZBACM0	DLZBACM0	DLZBACM0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	IJFUBZZZ	IJFUBZZZ			
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				

** Log Print Utility ** (See Figure 2-39)

DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGPE
	DLZLGPCN				
	DLZLGPMT				
	DLZLPCC0	DLZLPCC0	DLZLPCC0	DLZLPCC0	DLZLPCC0
	DLZLGPM0	DLZLGPM0	DLZLGPM0		DLZLGPM0
	IJJFCBID	IJJFCBID			
	IJJFCIID				
	IJFUZZZN	IJFUZZZN			
	IJGUICZZ	IJGUICZZ			

DATA BASE REORGANIZATION UTILITIES

** HS DB Unload ** (See Figure 2-29)

DLZURUL0	DLZURUL0	DLZURUL0	DLZURUL0	DLZURUL0	
	DLZRULM0	DLZRULM0	DLZRULM0		
	IJJFCBZD	IJJFCBZD			
	IJFVZZWN	IJFVZZWN			
	DLZCONSL				

** HS DB Reload ** (See Figure 2-30)

DLZURRL0	DLZURRL0	DLZURRL0	DLZURRL0	DLZURRL0	
	DLZRRLM0	DLZRRLM0	DLZRRLM0		
	IJJFCBZD	IJJFCBZD			
	IJFVZZWN	IJFVZZWN			
	IJFVZZWZ				
	DLZCONSL				

** HD DB Unload ** (See Figure 2-31)

DLZURGU0	DLZURGU0	DLZURGU0	DLZURGU0	DLZURGU0	
	DLZCONSL				
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CBLTDLI				
	DLZRGUM0	DLZRGUM0	DLZRGUM0		
	IJJFCBZD	IJJFCBZD			
	IJFUZZZN	IJFUZZZN			
	IJGUOCZZ	IJGUOCZZ			
	IJGUICZZ	IJGUICZZ			

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF

** HD DB Reload ** (See Figure 2-32)

DLZURGL0	DLZURGL0	DLZURGL0	DLZURGL0	DLZURGL0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CBLTDLI				
	DLZRGLM0	DLZRGLM0	DLZRGLM0		
	IJJFCBZD	IJJFCBZD			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				

ACB UTILITY

** ACB Creation ** (See Figure 2-33)

DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	
	PRTMSG				
	DLZDLBL0	DLZDLBL0	DLZDLBL0	DLZDLBL0	
	PSBPASS				
	DLZDLBL4				
	DLZDLBL1	DLZDLBL1	DLZDLBL1	DLZDLBL1	
	DLZDLBL2	DLZDLBL2	DLZDLBL2	DLZDLBL2	
	DLZDLBL3	DLZDLBL3	DLZDLBL3	DLZDLBL3	
	FREESTOR				
	IJSYSLN				
	PCHDTF				
	DLZLBLM0	DLZLBLM0	DLZLBLM0	DLZLBLM0	
	DLZUSCH0	DLZUSCH0	DLZUSCH0	DLZUSCH0	
	INSRCH				
	CLOSESCH				
	DLZDPSB0	DLZDPSB0	DLZDPSB0	DLZDPSB0	
	IJJCPD1N	IJJCPD1N			
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				

DB LOGICAL RELATIONSHIP UTILITIES

** Prereorganization ** (See Figure 2-34)

DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	DLZURGM0	DLZURGM0	DLZURGM0		
	IJJFCBZD	IJJFCBZD			
	IJGFOCZZ	IJGFOCZZ			

** DB Scan ** (See Figure 2-35)

DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	
	DLZCONSL				
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJGFICZZ	IJGFICZZ			

** Prefix Resolution ** (See Figure 2-36)

DLZURG10	DLZURG10	DLZURG10	DLZURG10	DLZURG10	
	DLZURGM0	DLZURGM0	DLZURGM0		
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJGFICZZ	IJGFICZZ			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJFVZZWN				
	IJFFZZZN	IJFFZZZN			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				
	DLZX15S1	DLZURG10	DLZURG10		
	DLZX15S2				
	DLZX35S1				
	DLZX35S2				

** Prefix Update ** (See Figure 2-37)

DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	CBLTDLI				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				

** Work File Generator ** (See Figure 2-38)

DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZBEGIN
	DLZBEGIN				
	OPENWORK				
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJGFICZZ	IJGFICZZ			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				

DIAGNOSTIC AND TEST MODULES

** System Formatted Dump **

DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0	
		DLZTRPR0	DLZTRPR0	DLZTRPR0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
** DL/I Tracing Facility **					
user chosen	DLZTRACE	user chosen	DLZTRACE	DLZTRACE	
	DLZTRPR0	DLZTRPR0	DLZTRPR0	DLZTRPR0	
	IJJFCBIC	IJJFCBIC			
** DL/I Test Program - Batch **					
DLZDLTXX	DLITCBL	DLZDLTXX	DLZDLTXX	DLZDLTXX	
	DLZSNAP				
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CBLTDLI				
	IJGFIZZZ	IJGFIZZZ			
	IJJFCBID	IJJFCBID			
	IJJFCIID				
** DL/I Test Program - Online **					
DLZDLTXY	DLITCBL	DLZDLTXY	DLZDLTXY	DLZDLTXY	
	DLZSNAP				
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CBLTDLI				
	IJGFIZZZ	IJGFIZZZ			
	IJJFCBID	IJJFCBID			
	IJJFCIID				
** Online Task Formatted Dump **					
DLZFTDP0	DLZFTDP0	DLZFTDP0	DLZFTDP0	DLZFTDP0	
** Run and Buffer Statistics ** (See Figure 2-42)					
DLZSTTL	DLZSTTL	DLZSTTL	DLZSTTL	DLZSTTL	
** Trace Print Utility ** (See Figure 2-41)					
DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRTE
	DLZTPRM0	DLZTPRM0	DLZTPRM0		DLZTPRM0
	IJJFCBIC				
	IJJFCIZD	IJJFCIZD			
	IJFVZZZZ	IJFVZZZZ			
	IJGVIEZZ	IJGVIEZZ			
	IJ2M0021	IJ2M0021			
** HD Partial Reorganization Utility ** (See Figure 2-43)					
DLZPRABC	DLZPRABC	DLZPRABC	DLZPRABC	DLZPRABC	
DLZPRCLN	DLZPRCLN	DLZPRCLN	DLZPRCLN	DLZPRCLN	
DLZPRCT1	DLZPRCT1	DLZPRCT1	DLZPRCT1	DLZPRCT1	
	COMAREA				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
DLZPRCT2	DLZPRCT2	DLZPRCT2	DLZPRCT2	DLZPRCT2	
	WORK1				
	COMAREA				
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	CBLTDLI				
	PLITDLI				
	RPGTDLI				
	IJJFCBZD	IJJFCBZD			

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	STORAGE ID	SUPPL INF
-------	--------	-------	-------	-----	-----
	IJJFCIZD				
DLZPRDBD	DLZPRDBD	DLZPRDBD	DLZPRDBD	DLZPRDBD	
DLZPRDLI	DLZPRDLI	DLZPRDLI	DLZPRDLI	DLZPRDLI	
DLZPRERR	DLZPRERR	DLZPRERR	DLZPRERR	DLZPRERR	
DLZPRPAR	DLZPRPAR	DLZPRPAR	DLZPRPAR	DLZPRPAR	
DLZPRPSB	DLZPRPSB	DLZPRPSB	DLZPRPSB	DLZPRPSB	
DLZPRREP	DLZPRREP	DLZPRREP	DLZPRREP	DLZPRREP	
DLZPRSCC	DLZPRSCC	DLZPRSCC	DLZPRSCC	DLZPRSCC	
DLZPRSTC	DLZPRSTC	DLZPRSTC	DLZPRSTC	DLZPRSTC	
DLZPRSTW	DLZPRSTW	DLZPRSTW	DLZPRSTW	DLZPRSTW	
DLZPRUPD	DLZPRUPD	DLZPRUPD	DLZPRUPD	DLZPRUPD	
DLZPRURC	DLZPRURC	DLZPRURC	DLZPRURC	DLZPRURC	
DLZPRWFM	DLZPRWFM	DLZPRWFM	DLZPRWFM	DLZPRWFM	

This section describes the major data areas used by DL/I DOS/VS. The description of each data area generally includes:

- Its DSECT name.
- The symbolic names of the fields and flags.
- The displacement of each field, in both decimal and hexadecimal.
- The length of each field.
- An alphabetic listing of all field and flag names (flags are indicated by asterisks).
- The hexadecimal code of each flag.

The data areas are documented in alphabetical order as listed in the Contents of this publication.

This section also describes the DL/I partition in a batch environment and illustrates the relationship of the DL/I control blocks. In addition, the description and general structure is given for the data management block (DMB), the program specification block (PSB), and the DL/I buffer pool control blocks.

THE DL/I PARTITION AND CONTROL BLOCK RELATIONSHIP

The following text describes the DL/I partition in a batch environment and illustrates the relationship of the DL/I control blocks described in this section.

THE DL/I BATCH PARTITION

Figure 5-1 is a map of main storage in the DL/I DOS/VS batch partition. Storage is allocated from the bottom or lowest storage address to the top or highest storage address of the partition. The eight areas in the DL/I batch partition are as follows:

- Area 1 contains the DL/I nucleus. The SCD is the first control block in the nucleus and contains the DL/I copyright information. This block also contains the entry point address for every module in the DL/I system. The PST prefix, PST, and PSB directory (PDIR) are in this area. There is one entry in the PSB directory (PDIR).

- Area 2 contains the DL/I program request handler, DLZPRHB0, which is loaded during DL/I initialization. It is part of the batch nucleus module (DLZBNUC0).

- Area 3 contains the PSB intent list, PSB, and one DMB directory (DDIR) for each DMB referenced by the PSB. The DMB directory is created dynamically during DL/I initialization.

- Area 4 contains DMBs loaded from the DOS/VS Core Image Library by the DL/I Batch Initialization module. Randomizing modules are loaded after the DMBs for HDAM. They are followed by VSAM control blocks, index management modules if secondary indexes are used, and by segment compression modules if variable length segments are used.

- Area 5 contains the DL/I buffer pool control blocks. These blocks are created dynamically. There are one buffer pool prefix, one subpool information table for each subpool specified, one DMB subpool directory entry for each DMB, and 2-32 buffer prefixes for each subpool specified.

- Area 6 contains the DL/I I/O buffers which comprise the buffer pool. There are 2-32 buffers for each subpool specified. Each subpool is aligned on a 2K page boundary.

- Area 7 contains the DL/I action modules and the user trace module if requested.

- Area 8 contains the user batch application program.

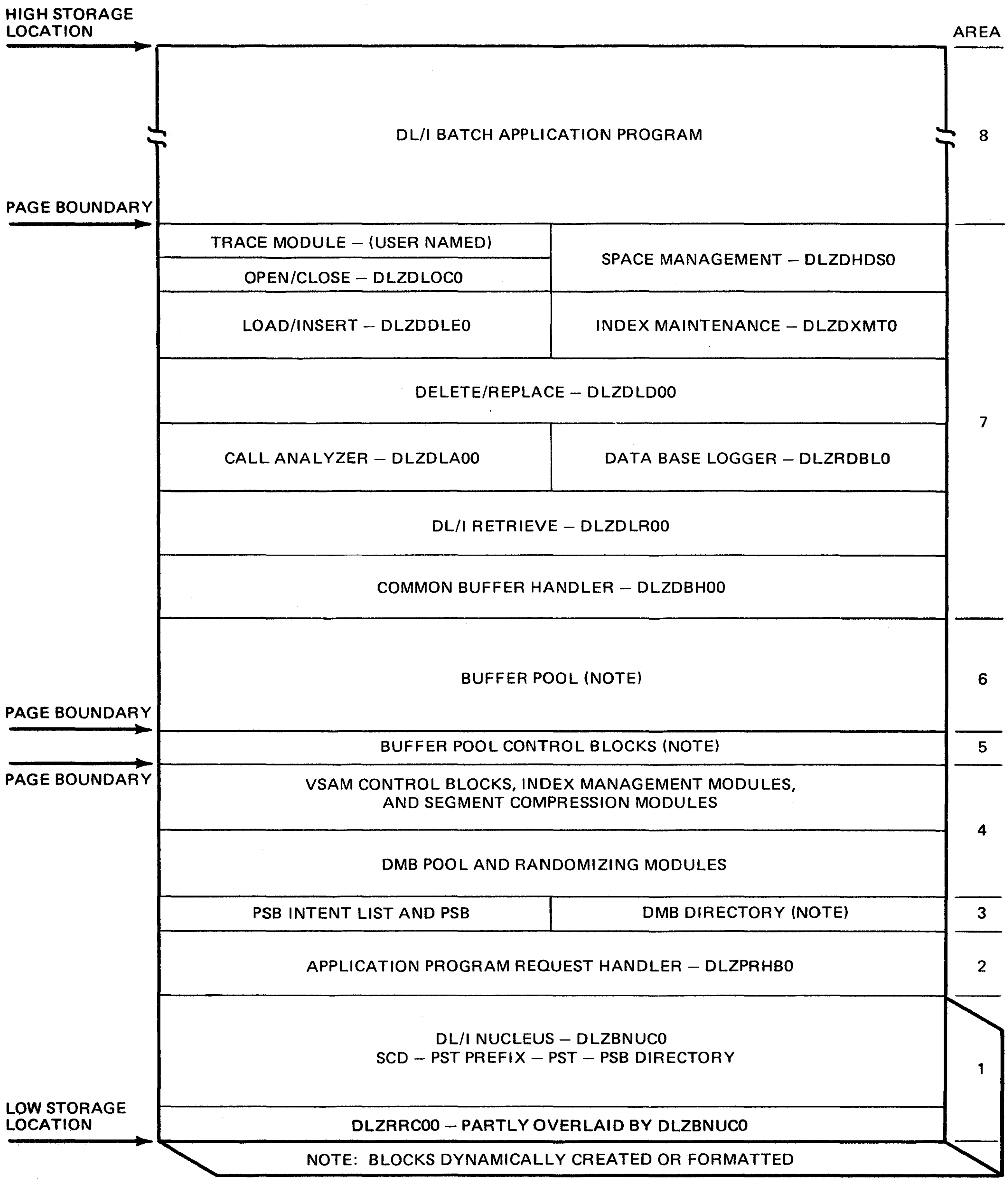


Figure 5-1. Map of Main Storage in the DL/I Batch Partition

DL/I CONTROL BLOCK RELATIONSHIP

The purpose of this section is to show the relationships of the various DL/I control blocks and provide a means by which the user can quickly find his way to these control blocks. The following discussion references Figure 5-2.

The SCD is the major control block in the DL/I system. It is located at the beginning of the DL/I nucleus. The SCD contains DL/I copyright information, entry point addresses of the DL/I logic module, and pointers to the following DL/I control blocks:

- The buffer pool prefix, which is the first block of the buffer pool control blocks.
- The first PSB directory from which the first PSB and PSB intent list may be obtained. In a batch system, there is only one PSB directory.
- The first DMB directory. There is one DMB directory for each DMB referenced by the PCBs.
- The first PST prefix from which the first PST may be obtained. There is only one PST prefix in a batch system.

The PST, including the PST prefix, functionally relates the control blocks for DL/I and represents the batch or CICS/DOS/VS - DL/I online task being served by DL/I. The PST is the dispatching block and is the only parameter passed when calling another module. The address of the PST is contained in the PST prefix. The following pointers are available in the PST:

- Caller's (user program) parameter list
- SCD
- PSB directory for the task
- PCB currently being accessed
- I/O buffer to be used for the data base call (used by the buffer handler)
- Subpool information table assigned to the data base (used by the buffer handler)
- Buffer prefix which points to the I/O buffer containing the segment for the call (used by the buffer handler)

There is one PSB directory entry and one PSB for each program that may be accessed by DL/I. In a CICS/DOS/VS - DL/I online environment, the maximum is 255; in batch, there can be only one. The PSB directory contains address pointers to the PSB and the PSB intent list.

The PSB intent list is a variable-length control block and contains an entry for each DMB referenced by the PSB. Each entry contains the address of the DMB.

The PSB contains prefix information and one or more PCBs. For each PCB there is a JCB, which is made up of the following: JCB prefix, level table, and one or more SDBs. The PCB points to the JCB. The JCB contains working storage for the program's use of that data base and points to the level table. The JCB also points to the SDB for the root segment and the VSAM ACB for the data base (KSDS ACB if HISAM).

The level table contains working storage for DL/I to store its positioning data for each level of the data base. The level table points to the current level SDB.

The SDB describes the user's logical use of the sensitive segment. There is one SDB for each segment to which the user is sensitive. Each SDB points to the corresponding PSDB in the DMB.

The DMB directory contains the address of the DMB. Each DMB contains a prefix, one ACB extension for each data set in the DMB (two if HISAM), one PSDB for each physical segment type, and one FDB for each field defined for a segment. In addition, there is one direct algorithm communication table (DMBDACS) if HDAM is used, and secondary list entries if HIDAM or HDAM with index or original relationships is used.

The DMB prefix contains:

- A two-byte relative offset to the first PSDB
- A two-byte relative offset to the end of the last PSDB+1, which is either the first secondary list entry (HIDAM) or the first FDB
- A four-byte pointer to DMBDACS if HDAM

The ACB extension contains information about the data set as well as an address pointer to the VSAM ACB and RPL for the data set.

Each PSDB contains:

- A pointer to the first FDB for the segment
- A pointer to the SDB for the active PCB which is sensitive to this segment type. If more than one PCB is sensitive to this segment type, the address of the SDB for the next PCB is contained in the active PSDB.

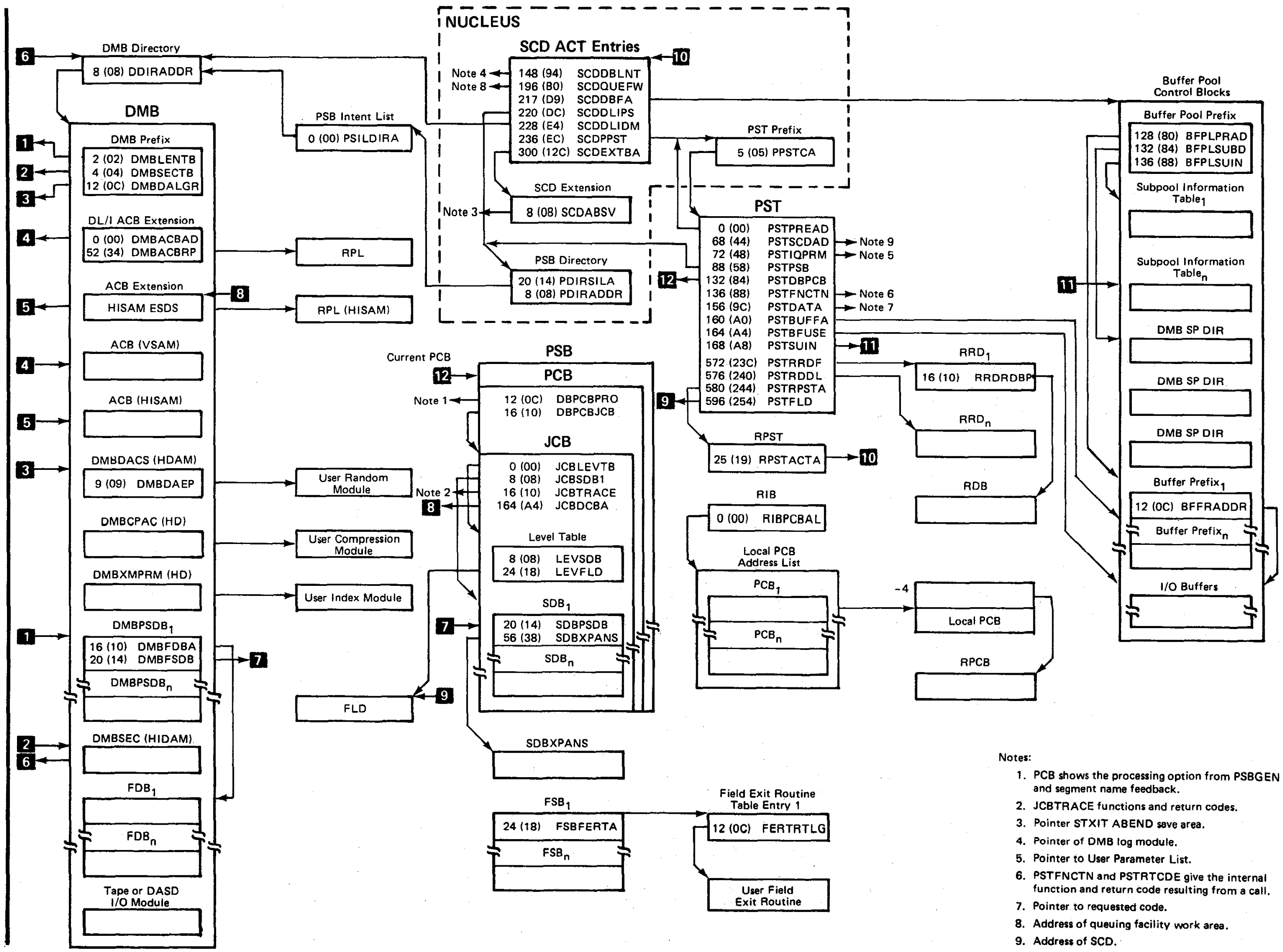
The DMBDACS contains the address of the user's randomizing routine; most of the secondary list entries point to the DMB directory for the described index or logically related data base.

The following items may be obtained from the buffer pool prefix:

- The first subpool information table (immediately following the buffer pool prefix)
- An address pointer to the first buffer prefix
- An address pointer to the first DMB subpool directory entry

The buffer prefix contains an address pointer to the I/O buffer which it references.

Figure 5-2. DL/I Control Block Relationships



Notes:

1. PCB shows the processing option from PSBGEN and segment name feedback.
2. JCBTRACE functions and return codes.
3. Pointer STXIT ABEND save area.
4. Pointer of DMB log module.
5. Pointer to User Parameter List.
6. PSTFNCTN and PSTRTCDE give the internal function and return code resulting from a call.
7. Pointer to requested code.
8. Address of queuing facility work area.
9. Address of SCD.

DATA MANAGEMENT BLOCK - DMB

A skeleton DMB is created during DBD generation (DBDGEN) as part of the DBD. The DMB consists primarily of a description of each segment contained in the data base and information concerning the physical data base description. This is contained in ACB extensions or, in the case of HSAM, in DTFs. The DBD is loaded into storage by the DL/I application control blocks creation and maintenance utility, which builds the DMB from the DBD created by DBDGEN. The DMB is then cataloged and link edited into a core image library. The DMB is moved to its execution-time location in the DMB pool by the application control blocks load and relocate module (DLZDBLM0).

The DMB consists of the following sections:

- A prefix section containing primarily offsets to subsections of the DMB
- An ACB extension. For an HISAM organizaton, there is a pair of ACB extensions for each data base; a KSDS ACB and an ESDS ACB. If the data base contains only root segments, only the KSDS ACB extension is created. The ACBs are generated only when the blocks are loaded for execution by DLZDBLM0 from the information in the ACB extensions.
- A DTF extension if SHSAM or HSAM for input and output file
- A direct algorithm communication table if HDAM
- A compression section for each compressable segment
- An index maintenance parameter section for each secondary exit routine
- A physical segment description block
- A secondary list to describe indexed fields or logical relationships.
- Field description blocks describing each field in each segment
- A tape or DASD I/O module if SHSAM or HSAM. This module is included by the ACB utility.

GENERAL STRUCTURE

The general structure of the DMB is shown in Figure 5-3.

Each DMB section is shown as a separate data area in Section 5 of this PLM, For the data area layout, see:

DMB PREFIX	DSECT Name: DMB	DMB — DMB Prefix
ACB EXTENSION	DSECT Name: DMBACBXT	ACB — ACB Extension
DTF EXTENSION	DSECT Name: DMBDTFXT	
DIRECT ALGORITHM COMMUNICATION TABLE	DSECT Name: DMBDACS	DACS — HDAM Randomizing Routine Interface Table
COMPRESSION SECTION	DSECT Name: DMBCPAC	CPAC — HDAM/HIDAM Variable Length Segment Compression/Expansion Routine
INDEX MAINTENANCE PARAMETERS	DSECT Name: DMBXMPRM	XMPRM — HDAM/HIDAM User Secondary Index Suppression Routine Interface Table
PHYSICAL SEGMENT DESCRIPTION BLOCK	DSECT Name: DMBPSDB	PSDB — Physical Segment Description Block
SECONDARY LIST	DSECT Name: DMBSEC	SEC — Secondary List
FIELD DESCRIPTION BLOCK	DSECT Name: FDB	FDB — Field Description Block
Tape or DASD I/O Module		

Figure 5-3. General Structure of DMB

PROGRAM SPECIFICATION BLOCK - PSB

A PSB must be created for every user program which will run under DL/I control. The PSB is created in "skeleton" format (principally PCBs only) by PSBGEN. The PSB must be cataloged and link edited into the Core Image Library. The PSB is loaded into main storage by the DL/I Application Control Blocks Creation and Maintenance Utility program and expanded and completed by this utility. The expansion is performed by segment definition in the DBD representing the associated data base. The expanded PSB is link edited into the Core Image Library. The PSB is moved to its execution-time location in the PSB pool by the application control blocks load and relocate module (DLZDBLM0). In expanded final format, the PSB consists of the following parts in the order specified:

1. PSB prefix - of which the most important part is the variable-length PSB list: the address list of the PCBs in the PSB. A dope vector table follows the PSB prefix for PL/I programs.

2. A variable number of data base PCBs. For each data base PCB there is a JCB (job control block) consisting of the following parts:

   - JCB prefix

   - DSG (data set group) table. This table contains entries describing the data bases specifically used for this PCB. There are entries for all logically connected data bases, all primary HIDAM indexes, and a secondary index if used as the processing sequence.

   - Level table. This table provides memory of the last DL/I CALL.

   - SDB (segment description block). This block contains an entry for each segment to which the user has declared himself sensitive in the PCB. The SDB entry describes the sensitive segment.

   - Work area for index maintenance, variable-length segment support, or miscellaneous function. These are allocated only when required (if any user PCB directly or indirectly refers to an index data base).

   - PSB work areas; of variable length depending on the requirements of the PCBs.

GENERAL STRUCTURE

The general structure of the PSB is shown in Figure 5-4.

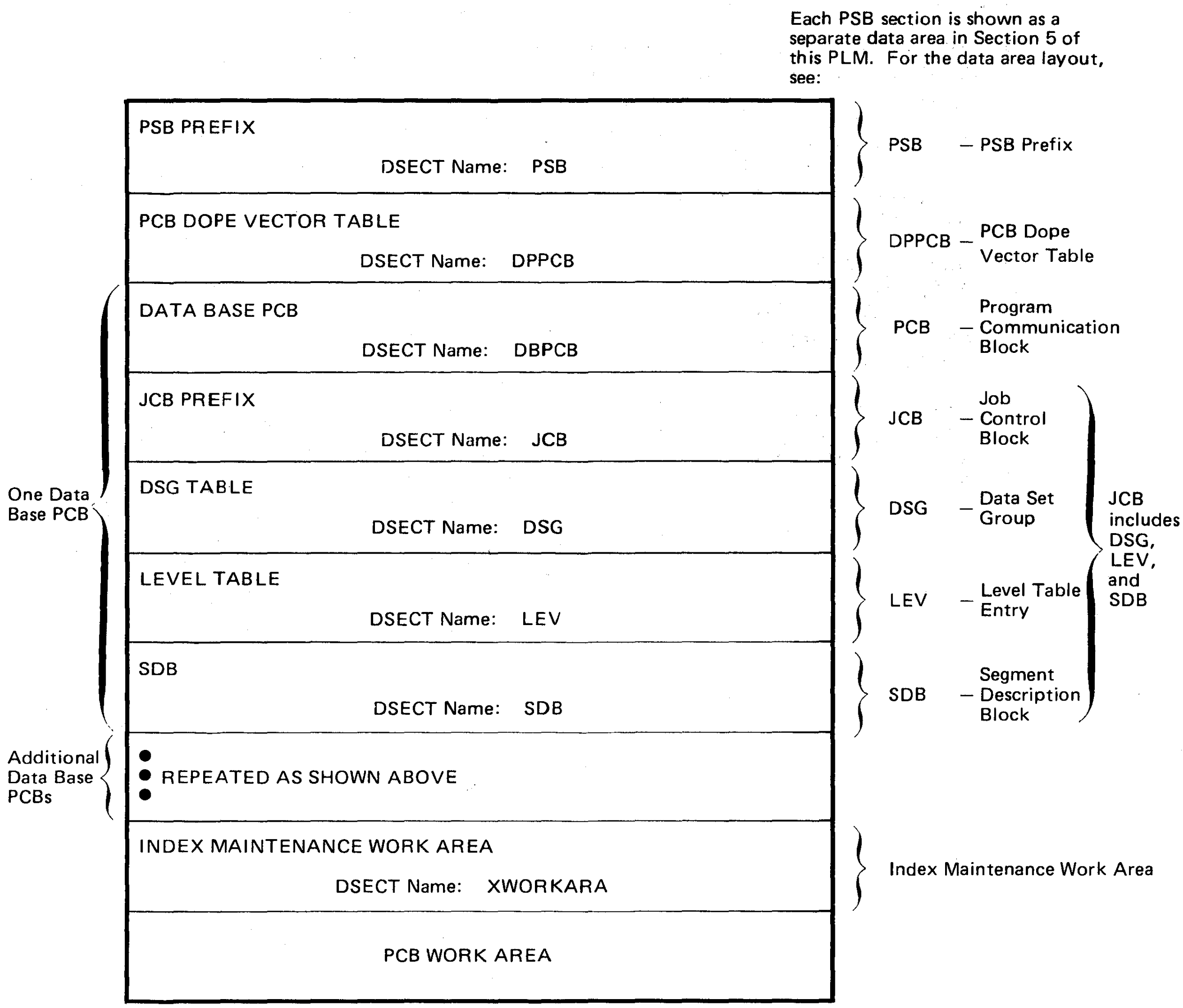


Figure 5-4. General Structure of PSB.

DL/I BUFFER POOL CONTROL BLOCKS

The DL/I buffer pool control blocks provide the control information to manage the entire buffer pool for the DL/I task. The buffer pool control blocks are as follows:

- Buffer Pool Control Block Prefix - This control block contains the statistics and other control information for the entire buffer pool.
- Subpool Information Table - This control block contains information for a specific subpool, including the size of the buffers in the subpool. There is one subpool information table for each subpool allocated.
- DMB Subpool Directory - This control block contains a one-byte subpool number relative to zero for each HDAM or HIDAM data base allocated. The DMB sequence number is used as an offset intc the DMB directory and allows a DMB to be identified with a specific subpool.
- Buffer Prefix Control Block - This control block contains key information about the contents of a specific buffer in a subpool. There is one buffer prefix control block for each buffer. Each subpool contains 2-32 buffers.

GENERAL STRUCTURE

The general structure of the DL/I buffer pool control blocks is shown in Figure 5-5.

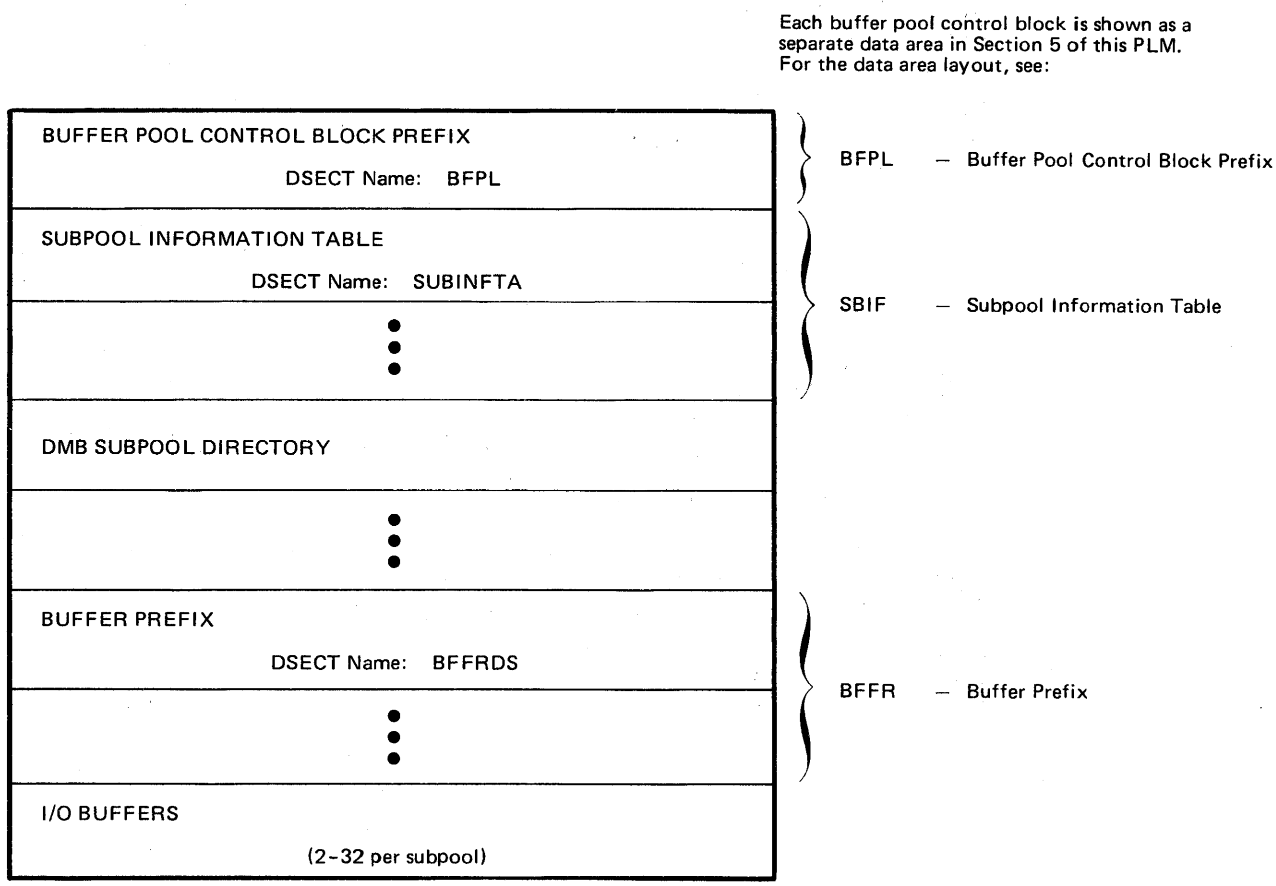


Figure 5-5. General Structure of DL/I Buffer Pool Control Blocks

ACBXT - ACB EXTENSION

DSECT Name: DMBACBXT

The ACB extension is described as part of the general structure and description of the data management block (DMB). The information in ACBXT is repeated for each data set in the DMB.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBACBAD	0(00)	
DMBACBAP	7(07)	
DMBACBDL	6(06)	
DMBACBEX	68(44)	
DMBACBLC	56(38)	
DMBACBLN	80(50)	
DMBACBMN	10(0A)	
DMBACBMX	8(08)	
DMBACBND	80(50)	
DMBACBNM	60(3C)	
DMBACBRP	52(34)	
DMBACBST	0(00)	
DMBACLN0	60(3C)	
*DMBBESDS	46(2E)	40
DMBBFACT	44(2C)	
DMBCICYL	28(1C)	
DMBCINV	4(04)	
*DMBCISPL	35(23)	80
DMBCITRK	30(1E)	
DMBDTFIN	0(00)	(See DTF extension at end of ACBXT)
DMBDTFOT	4(04)	(See DTF extension at end of ACBXT)
DMBECB	12(0C)	
DMBFBASN	72(48)	
*DMBFBA	46(2E)	20
DMBFRSPC	58(3A)	
DMBFRSP1	59(3B)	
DMBHIBLK	16(10)	
DMBHIRBA	36(24)	
DMBINDO	46(2E)	
*DMBIGNOR	34(22)	40
*DMBKEY	46(2E)	80
DMBKEYLE	31(1F)	
DMBLRECL	42(2A)	
*DMBNUSE	34(22)	20
DMBOFLGS	34(22)	
*DMBOPEN	34(22)	10
*DMBPSEQ	35(23)	10
*DMBPUTKY	34(22)	08
DMBRBASN	20(14)	
DMBRKP	32(20)	
DMBRLBLK	24(18)	
DMBSPLCT	48(30)	
DMBVSBFR	40(28)	
DMBVSFLG	35(23)	
*DMBWCHK	46(2E)	08

RECORD LAYOUT - ACBXT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	DMBACBST		Start of ACB extension
0(00)	4	DMBACBAD		Address of corresponding ACB
4(04)	2	DMBCINV		Control interval size
6(06)	1	DMBACBDL		Delta cylinders to scan
7(07)	1	DMBACBAP		Number of root anchor points per control interval (HDAM)
8(08)	2	DMBACBMX		Length of the largest segment in data set
10(0A)	2	DMBACBMN		Length of the smallest segment in data set
12(0C)	4	DMBECB		VSAM ACB event control block (ECB) used by buffer handler (DLZDBH00)
16(10)	4	DMBHIBLK		Highest control interval RBA
20(14)	4	DMBRBASN		RBA of last logical record assigned (HISAM) or relative block number of last control interval assigned (HD). During batch initialization the high-order byte is the buffer size (control interval size/512) indicator
24(18)	4	DMBRLBLK		Relative block number of last control interval written (HD)
28(1C)	2	DMBCICYL		Number of control intervals per cylinder
30(1E)	1	DMBCITRK		Number of control intervals per track
31(1F)	1	DMBKEYLE		Key length of KSDS
32(20)	2	DMBRKP		Relative key position
34(22)	1	DMBOFLGS		Open flags
		DMBIGNOR	40	IGN was specified for workfile on load
		DMBNUSE	20	ACB does not have resolved secondary index entries; workfile must be used

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		DMBOPEN	10	The corresponding ACB is open
		DMBPUTKY	08	Simulate not load mode to VSAM
35(23)	1	DMBVSFLG		Flags
		DMBCISPL	80	Control interval split occurred
		DMBPSEQ	10	Sequential processing is possible for this KSDS
36(24)	4	DMBHIRBA		Highest RBA in present range of extents (HIDAM ESDS only)
40(28)	2	DMBVSBFR		Number of buffers to be used
42(2A)	2	DMBLRECL		Logical record length
44(2C)	2	DMBBFACT		Blocking factor
46(2E)	1	DMBINDO		Permanent indicators
		DMBWCHK	08	Write check option
		DMBFBA	20	FBA device suport
		DMBBESDS	40	Blocked ESDS
		DMBKEY	80	Data set contains keys (Simple HISAM and SHISAM)
47(2F)	1			**Reserved**
48(30)	4	DMBSPLCT		Control interval split count
52(34)	4	DMBACBRP		Address of RPL for this ACB
56(38)	2	DMBACBLC		Log count (HISAM only)
58(3A)	1	DMBFRSPC		Distributed free space parameter
59(3B)	1	DMBFRSP1		Second free space parameter
60(3C)	8	DMBACBNM		Data set name as in ACB
		DMBACLN0		Length of version 1.0
68(44)	4	DMBACBEX		Address of exit list for this ACB
72(48)	2	DMBFBASN		FBA scan value
74(4A)	6			**Reserved**
80(50)	2	DMBACBND		End of ACB extension
		DBMACBLN		Length of ACB extension (DMBACBND minus DMBACBST)

Note: HSAM DMBs have the following DTF extension.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
DSECT Name: DMBDTFXT				
0(00)	4	DMBDTFIN		Address of HSAM input DTF
4(04)	4	DMBDTFOT		Address of HSAM output DTF

ACT - PARTIAL REORGANIZATION ACTION TABLE

DSECT Name: ACT

This DSECT describes one action to be taken by either RELOAD or SCAN. It also defines the action to be taken by UPDATE when the record created by RELOAD or SCAN is read back. It is built by the action table builder and is used by RELOAD, SCAN and UPDATE phases in step 2. Its address is held in the common area field (COMAACT).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
ACTCROW	1(01)	
ACTCRTYP	0(00)	
ACTCSDS	5(05)	
ACTGDEST	4(04)	
ACTGOPTN	3(03)	
ACTLLEN	24(18)	20
ACTOANXT	22(16)	
ACTOCHFD	16(10)	
ACTOCNXT	18(12)	
ACTOPRMV	12(0C)	
ACTOPUPD	14(0E)	
ACTOSGT	6(06)	
ACTOSUPD	8(08)	
ACTOSZID	10(0A)	
ACTOTEST	20(14)	
ACTQOPT2	3(03)	
ACTQSRT1	4(04)	
ACTQSRT2	4(04)	
ACTQSRT3	4(04)	
ACTQSRT4	4(04)	
ACTSTART	0(00)	

RECORD LAYOUT - ACT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	ACTCRTYP		Action record type
1(01)	1	ACTCROW		Action row number
2(02)	1			** Reserved **
3(03)	1	ACTGOPTN		Optional action identifier
		ACTQOPT2	80	Option with two
4(04)	1	ACTGDEST		Destination indicator flags
		ACTQSRT1	80	Record goes to sort 1
		ACTQSRT2	40	Record goes to sort 2
		ACTQSRT3	20	Record goes to sort 3

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		ACTQSRT4	10	Record goes to sort 4
5(05)	1	ACTCSDS		Data set of moved segment for sort 1
6(06)	2	ACTOSGT		Offset in SGT from which this is chained
8(08)	2	ACTOSUPD		Offset in SGT for segment to be updated
10(0A)	2	ACTOSZID		OFFSET in SGT for z segment in physical pair
12(0C)	2	ACTOPRMV		Offset in prefix of pointer to be extracted
14(0E)	2	ACTOPUPD		Offset in prefix of pointer to be updated
16(10)	2	ACTOCHED		Offset in prefix of chain head pointer
18(12)	2	ACTOCNXT		Offset in prefix of next in chain pointer
20(14)	2	ACTOTEST		Offset to be tested for zero or non-xero
22(16)	2	ACTOANXT		Offset in ACT of next action
24(18)	4(2)			** Reserved **
		ACTLLEN		(*-ACTSTART) length of an action table entry

ARGO - HLPI ARGO PARAMETERS

DSECT Name: HLPI

The DSECT describes the fields contained in the DL/I HLPI ARGO Interface Parameter list.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*APPLPLI	4 (04)	02
ARGO	0 (00)	
ARGOCCOD	25 (19)	
ARGOFLG1	2 (02)	
ARGOFLG2	3 (03)	
ARGOFLG3	4 (04)	
ARGOFNCD	1 (01)	
ARGOFNID	0 (00)	
ARGOOPTS	24 (18)	
ARGORELN	6 (06)	
ARGORMGR	8 (08)	
ARGOSOPT	27 (1B)	
ARGOSTMT	16 (10)	
ARGOTOTN	7 (07)	
*CCFIRST	25 (19)	80
*CCINFORM	25 (19)	10
*CCLAST	25 (19)	40
*CCLOCKED	25 (19)	20
*CHKPCALL	1 (01)	08
*DLETCALL	1 (01)	16
DLZARGO	0 (00)	
*GNCALL	1 (01)	0C
*GNPCALL	1 (01)	10
*GUCALL	1 (01)	0A
*INITCALL	1 (01)	02
*ISRTCALL	1 (01)	12
*LOADCALL	1 (01)	18
*OPTFLDL	27 (1B)	10
*OPTOFF	27 (1B)	02
*OPTSEGL	27 (1B)	80
*OPTSEGM	27 (1B)	04
*OPTVAR	27 (1B)	08
*OPTWHERE	27 (1B)	40
*REPLCALL	1 (01)	14
*SCHDCALL	1 (01)	04
*TERMCALL	1 (01)	06
*USINGPCB	24 (18)	40

RECORD LAYOUT - ARGO

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	ARGO		
0(00)	1	ARGOFNID		ARGO ID X'00'
1(01)	1	ARGOFNCD		Function code
		INITCALL	02	Initialize call
		SCHDCALL	04	Schedule call
		TERMCALL	06	Termination call
		CHKPCALL	08	Checkpoint call
		GUCALL	0A	Get unique call
		GNCALL	0C	Get next call
			0E	** Reserved **
		GNPCALL	10	Get next in parent call
		ISRTCALL	12	Insert call
		REPLCALL	14	Replace call
		DLETCALL	16	Delete call
		LOADCALL	18	Load call
2(02)	1	ARGOFLG1		Argument flag 1
3(03)	1	ARGOFLG2		Argument flag 2
4(04)	1	ARGOFLG3		Argument flag 3
		APPLPLI	02	Application program is PL/I
5(05)	1			** Reserved **
6(06)	1	ARGORELN		Relative number of this call
7(07)	1	ARGOTOTN		Total number of calls in this statement
8(08)	8	ARGORMGR		Resource manager's ID
16(10)	8	ARGOSTMT		Statement identifier
24(18)	1	ARGOOPTS		Statement level options
		USINGPCB	40	Using PCB
25(19)	1	ARGOCCOD		Command codes
		CCFIRST	80	First
		CCLAST	40	Last
		CCLOCKED	20	Locked
		CCINFROM	10	Into or from
26(1A)	1			** Reserved **
27(1B)	1	ARGOSOPT		Segment options
		OPTSEGL	80	SEGLENGTH specified or default
		OPTWHERE	40	Where
			20	Boolean where (IMS only)
		OPTFLDL	10	Field length specified or default
		OPTVAR	08	Variable

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		OPTSEGM	04	Segment name present
		OPTOFF	02	Offset specified
28(1C)	1			** Reserved **

BFFR - BUFFER PREFIX

DSECT Name: BFFRDS

The buffer prefix is described as part of the general structure and description of the DL/I buffer pool control blocks. There is one buffer prefix for each buffer allocated.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
BFFRADDR	12(0C)	
BFFRCIID	0(00)	
BFFRCIRB	0(00)	
BFFRDCB	6(06)	
BFFRDMB	4(04)	
*BFFREXNQ	7(07)	02
BFFRHOLE	30(1E)	
*BFFRLAST	27(1B)	01
BFFRLEN	32(20)	
*BFFRLOCK	27(1B)	40
BFFRLOCU	10(0A)	
*BFFRMT	7(07)	10
BFFRNACB	26(1A)	
BFFRNCII	20(14)	
BFFRNCID	20(14)	
BFFRNDMB	24(18)	
*BFFRNORU	27(1B)	80
BFFRNPSF	28(1C)	
BFFRNPSL	29(1D)	
BFFRNPST	28(1C)	
*BFFRPNNQ	7(07)	01
*BFFRPRED	7(07)	08
BFFRPST	8(08)	
BFFRPSTF	8(08)	
BFFRPSTL	9(09)	
*BFFRREAD	7(07)	20
*BFFRREL	27(1B)	08
BFFRSW	7(07)	
BFFRSW1	27(1B)	
BFFRUSCT	12(0C)	
BFFRUSID	16(10)	
BFFRWCBW	19(13)	
BFFRWCFW	18(12)	
*BFFRWCH	7(07)	80
*BFFRWERR	7(07)	04
*BFFRWRT	7(07)	40

RECORD LAYOUT - BFFR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	7	BFFRCIID		Control Interval identifier
0(00)	4	BFFRCIRB		Control Interval RBA
4(04)	2	BFFRDMB		DMB Number
6(06)	1	BFFRDCB		ACB Number
7(07)	1	BFFRSW		Flags
		BFFRWCH	80	Buffer on write chain
		BFFRWRT	40	Buffer being written
		BFFRREAD	20	Buffer being read
		BFFRMT	10	Buffer empty
		BFFRPRED	08	Buffer waiting for predecessor being written
		BFFRWERR	04	Buffer has permanent write error
		BFFREXNQ	02	Existing CI ID enqueued
		BFFRPNNQ	01	Pending CI ID enqueued
8(08)	2	BFFRPST		PST prefix numbers for enqueue/dequeue
8(08)	1	BFFRPSTF		PST prefix number of the controlling task
9(09)	1	BFFRPSTL		PST prefix number of the last task in the chain of waiting tasks
10(0A)	2	BFFRLOCU		Log count
12(0C)	1	BFFRUSCT		Use count
12(0C)	4	BFFRADDR		Address of buffer
16(10)	2	BFFRUSID		ID of the users who altered this buffer
18(12)	1	BFFRWCFW		Next lower buffer on the write chain
19(13)	1	BFFRWCBW		Next higher buffer on the write chain
20(14)	7	BFFRNCID		New control interval identifier
20(14)	4	BFFRNCII		New control interval RBA
24(18)	2	BFFRNDMB		New DMB number
26(1A)	1	BFFRNACB		New ACB number

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
27(1B)	1	BFFRSW1		Flags
		BFFRNORU	80	Buffer is not reusable
		BFFRLOCK	40	Buffer locked by logger
		BFFRREL	08	Buffer is released
		BFFRLAST	01	Last buffer prefix for this subpool
28(1C)	2	BFFRNPST		PST prefix numbers for enqueue/dequeue
28(1C)	1	BFFRNPSF		PST prefix number of task that enqueued on new CI ID and is first in the chain
29(1D)	1	BFFRNPSL		PST prefix number of task that enqueued on new CI ID and is last in the chain
30(1E)	2	BFFRHOLE		Length of largest space available in the buffer
32(20)		BFFRLEN		Length of buffer prefix

BFPL - BUFFER POOL CONTROL BLOCK PREFIX

DSECT Name: BFPL

The BFPL is described as part of the general structure and description of DL/I buffer pool control blocks. There is one buffer pool control block prefix that contains information for the entire buffer pool.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
BFPL	0(00)	
BFPLALTR	28(1C)	
BFPLBKWT	36(24)	
BFPLCHBK	48(30)	
BFPLCHWT	44(2C)	
BFPLCOUT	62(3E)	
*BFPLEXCI	64(40)	00
BFPLID	0(00)	
BFPLIGET	56(38)	
BFPLINCO	96(60)	
BFPLINMA	72(48)	
BFPLINPL	20(14)	
BFPLINRO	88(58)	
BFPLINW1	88(58)	
BFPLINW2	104(68)	
BFPLISTL	52(34)	
BFPLLEN	136(88)	
BFPLNQW1	64(40)	
BFPLNQW2	68(44)	
BFPLNWBK	40(28)	
BFPLOSWT	32(20)	
*BFPLPECI	64(40)	04
BFPLPRAD	128(80)	
BFPLPSIF	124(7C)	
BFPLPSIL	125(7D)	
BFPLPSI1	120(78)	
BFPLRDCT	24(18)	
BFPLROCO	63(3F)	
BFPLRQCT	16(10)	
BFPLSUBD	132(84)	
BFPLSUIN	136(88)	
*BFPLSUPO	64(40)	08
*BFPLSW00	68(44)	00
*BFPLSW80	68(44)	80
BFPLWERR	60(3C)	
BFPLWERT	61(3D)	

RECORD LAYOUT - BFPL

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)		BFPL		
0(00)	4	BFPLID		Buffer pool control block ID (BFPL)
4(04)	12			** Reserved **
16(10)	4	BFPLRQCT		Number of requests received by the buffer handler
20(14)	4	BFPLINPL		Number of requests satisfied from buffer pool
24(18)	4	BFPLRDCT		Number of read requests issued
28(1C)	4	BFPLALTR		Number of buffer alter requests received
32(20)	4	BFPLOSWT		Number of writes issued
36(24)	4	BFPLBKWT		Number of blocks written
40(28)	4	BFPLNWBK		Number of new blocks created in pool
44(2C)	4	BFPLCHWT		Number of chained writes issued
48(30)	4	BFPLCHBK		Number of blocks written on write chain
52(34)	4	BFPLISTL		Number of retrieves by key calls
56(38)	4	BFPLIGET		Number of GN calls received
60(3C)	1	BFPLWERR		Number of permanent write error buffers in pool
61(3D)	1	BFPLWERT		Largest number of write error buffers ever in pool
62(3E)	1	BFPLCOUT		Number of rows/columns in matrix currently in use
63(3F)	1	BFPLROCO		Mask showing available rows/columns in matrix

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
64(40)	4	BFPLNQW1		ENQ/DEQ workarea 1. Byte 0 indicates the following:
		BFPLEXCI	00	ENQ/DEQ existing CI code
		BFPLPECI	04	ENQ/DEQ pending CI code
		BFPLSUPO	08	ENQ/DEQ subpool code Bytes 1-3 contain a pointer to the PST prefix numbers of the first and last task waiting for the resource
68(44)	4	BFPLNQW2		ENQ/DEQ workarea 2
		BFPLSW00	00	Mask to turn off wait switch
		BFPLSW80	80	Task waiting for matrix space
72(48)	16	BFPLINMA		Interlock detection matrix
88(58)	16	BFPLINW1		Interlock detection workarea 1
88(58)	8	BFPLINRO		
96(60)	8	BFPLINCO		
104(68)	16	BFPLINW2		Interlock detection workarea 2
120(78)	4	BFPLPSI1		Pointer to the PST prefix numbers of the first and last task waiting for matrix space
124(7C)	1	BFPLPSIF		PST prefix number of the first task waiting for matrix space
125(7D)	1	BFPLPSIL		PST prefix number of the last task waiting for matrix space
126(7E)	2			** Reserved **
128(80)	4	BFPLPRAD		Beginning address of the buffer prefix area
132(84)	4	BFPLSUBD		Beginning address of the DMB subpool directory
136(88)	0	BFPLSUIN	88	Beginning of the subpool information table entries
136(88)		BFPLLEN	88	Length of the buffer pool control block prefix

COM - COMMON AREA

DSECT Name: COM

This CSECT/DSECT describes the common area used by partial reorganization. The common area is assembled as a CSECT in the Part1 and Part2 control modules. In all other modules it is used as a DSECT. The common area is made up of the following sections:

1. General address section
2. Switch and data section
3. DL/I address section
4. File section
5. Checkpoint section

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
COMAACT	76 (04C)	
COMABUFH	200 (0C8)	
COMACHKD	752 (2F0)	
COMACHKP	36 (024)	
COMACHXR	724 (2D4)	
COMACOM	16 (010)	
COMADBD	44 (02C)	
COMADBT	52 (034)	
COMADDIR	192 (0C0)	
COMADLI	224 (0E0)	
COMADLII	32 (020)	
COMAERRS	24 (018)	
COMAFILE	28 (01C)	
COMAFL25	764 (2FC)	
COMAGBUF	756 (2F4)	
COMAIOWK	736 (2E0)	
COMAIPCB	728 (2D8)	
COMALMXS	732 (2DC)	
COMALOG	196 (0C4)	
COMAMSGN	158 (09E)	
COMAPLST	740 (2E4)	
COMAPMCT	720 (2D0)	
COMAPREF	208 (0D0)	
COMAPST	188 (0BC)	
COMAREA	0 (000)	
COMARGT	84 (054)	
COMASCD	184 (0B8)	
COMASGT	64 (040)	
COMASIOA	20 (014)	
COMASMGR	204 (0CC)	
COMASTWR	40 (028)	
COMAVTXT	164 (0A4)	
COMCDDNM	784 (310)	
COMCID	0 (000)	
COMCIREQ	128 (080)	
COMCPROC	704 (2C0)	
COMCPSBN	120 (078)	
COMCSDIA	154 (09A)	
COMCSMSG	145 (091)	
COMCSSIZ	136 (088)	
COMCSTEC	132 (084)	
COMCTRAC	876 (36C)	
COMFACTL	72 (048)	
COMFACTM	80 (050)	
COMFCKID	700 (2BC)	
COMFCOML	12 (00C)	
COMFCXPL	744 (2E8)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
COMFDBTL	48 (030)	
COMFDBTM	56 (038)	
COMFFCBL	760 (2F8)	
COMFLCKD	748 (2E8)	
COMFPMCT	768 (300)	
COMFRETC	116 (074)	
COMFSGTL	60 (03C)	
COMFSGTM	68 (044)	
COMFWRK1	168 (0A8)	
COMFWRK2	172 (0AC)	
COMFWRK3	176 (0B0)	
COMFWRK4	180 (0B4)	
COMGOUT	129 (081)	
COMGPART	131 (083)	
COMGPHAS	705 (2C1)	
COMGPHS2	710 (2C6)	
COMGUCTL	130 (082)	
COMHEADC	987 (3DB)	
COMHEADD	1010 (3F2)	
COMHEADP	1025 (401)	
COMHEADR	908 (38C)	
COMHEADV	951 (3B7)	
COMHKYLN	112 (070)	
COMHLACT	102 (066)	
COMHLDBT	90 (05A)	
COMHLRGT	106 (06A)	
COMHLSGT	94 (05E)	
COMHMXPR	98 (062)	
COMHMXSG	110 (06E)	
COMHNACT	104 (068)	
COMHNDBT	92 (05C)	
COMHNRGT	108 (06C)	
COMHNSGT	96 (060)	
COMHNSGX	100 (064)	
COMHPAGE	1029 (405)	
COMLCXPL	772 (304)	
*COMLDSGT	792 (318)	09
COMLLEN	908 (38C)	
COMLRGT	88 (058)	
COMOCRGT	706 (2C2)	
COMODBSN	708 (2C4)	
COMPAGEM	1031 (407)	
*COMQBFAL	128 (080)	0D
*COMQBKLC	128 (080)	03
*COMQBMOF	128 (080)	0C
*COMQBMON	128 (080)	0B
*COMQBYAL	128 (080)	05
*COMQBYLC	128 (080)	04
COMQCKND		2D0
*COMQCRAP	128 (080)	15
*COMQDUNQ	130 (082)	80
*COMQFREE	128 (080)	07
*COMQGNDX	128 (080)	16
*COMQGPRE	128 (080)	01
*COMQGRBA	128 (080)	06
*COMQINBF	710 (2C6)	80
*COMQINTR	128 (080)	13
*COMQINTU	128 (080)	11
*COMQINT2	128 (080)	10
*COMQIPRE	128 (080)	02
*COMQLNEW	128 (080)	0A
*COMQLOLD	128 (080)	09

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*COMQNDBR	130 (082)	10
*COMQNPSB	129 (081)	01
*COMQOPTN	705 (2C1)	02
*COMQPHSC	704 (2C0)	02
*COMQPHSO	704 (2C0)	03
*COMQPHUD	704 (2C0)	04
*COMQPHUR	704 (2C0)	01
*COMQRIP	131 (083)	02
*COMQRKEY	128 (080)	08
*COMQRSTR	128 (080)	14
*COMQRSTU	128 (080)	12
*COMQSALL	129 (081)	80
*COMQSCAN	705 (2C1)	80
*COMQSNON	129 (081)	20
*COMQSPOF	130 (082)	20
*COMQSPUS	130 (082)	40
*COMQSRT1	705 (2C1)	40
*COMQSRT2	705 (2C1)	20
*COMQSRT3	705 (2C1)	10
*COMQSRT4	705 (2C1)	08
*COMQSTP1	131 (083)	80
*COMQSTP2	131 (083)	40
*COMQSUMM	129 (081)	40
*COMQULHB	128 (080)	0E
*COMQUPDT	705 (2C1)	04
*COMQUPIX	705 (2C1)	01
*COMQXRMA	128 (080)	0F
COMRHIPT	220 (0DC)	
COMRLOPT	216 (0D8)	
COMRLSEG	212 (0D4)	
COMSFL01	232 (0E8)	
COMSFL02	268 (10C)	
COMSFL03	304 (130)	
COMSFL04	340 (154)	
COMSFL05	376 (178)	
COMSFL06	412 (19C)	
COMSFL07	448 (1C0)	
COMSFL08	484 (1E4)	
COMSFL09	520 (208)	
COMSFL10	556 (22C)	
COMSFL11	592 (250)	
COMSFL12	628 (274)	
COMSFL13	664 (298)	
COMSTART	0 (000)	
COMXBR14	114 (072)	
COMXDGID	792 (318)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	0	COMAREA		
0(000)	4	COMSTART		
0(000)	1	COMCID		Identifier
G E N E R A L A D D R E S S S E C T I O N				
12(00C)	4	COMFCOML		Length of common
16(010)	4	COMACOM		Address of common

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
20(014)	4	COMASIOA		Address of an I/O area for GU, GN calls
24(018)	4	COMAERRS		Entry point for error message writer
28(01C)	4	COMAFILE		Entry point of file manager
32(020)	4	COMADLII		Entry point of DL/I interface module
36(024)	4	COMACHKP		Entry point of checkpoint processor
40(028)	4	COMASTWR		Entry point of statistics writer
44(2C)	4	COMADBD		Address of data base block
48(30)	4	COMFDBTL		Length of data base table (DBT)
52(34)	4	COMADBT		Address of DBT
56(38)	4	COMFDBTM		Maximum size of DBT
60(3C)	4	COMFSGTL		Length of segment table (SGT)
64(40)	4	COMASGT		Address of SGT
68(44)	4	COMFSGTM		Maximum size of SGT
72(48)	4	COMFACTL		Length of action table (ACT)
76(4C)	4	COMAACT		Address of ACT
80(50)	4	COMFACTM		Maximum size of ACT
84(54)	4	COMARGT		Address of RGT
88(58)	2	COMLRGT		Length of range table (RGT)
90(5A)	2	COMHLDBT		Length of a DBT entry
92(5C)	2	COMHNDBT		Number of DBT entries
94(5E)	2	COMHLSGT		Length of a SGT entry
96(60)	2	COMHNSGT		Number of SGT entries
98(62)	2	COMHMXPR		Length of longest prefix in data base #1
100(64)	2	COMHNSGX		Number of SGX entries
102(66)	2	COMHLACT		Length of an ACT entry

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
104(68)	2	COMHNACT		Number of ACT entries
106(6A)	2	COMHLRGT		Length of an RGT entry
108(6C)	2	COMHNRGT		Number of RGT entries
110(6E)	2	COMHMXSG		Length of data part of longest segment
112(70)	2	COMHKYLN		Length of current HIDAM KEY
SWITCH AND DATA SECTION				
114(72)	2	COMXBR14		A BR 14 instruction
116(74)	4	COMFRETC		Level of most severe error to date
120(78)	8	COMCPSBN		Name to be given to generated PSB
128(80)	1	COMCIREQ		DLI common services request code
		COMQGPRE	01	Get prefix address of last segment retrieved
		COMQIPRE	02	Get prefix address of last segment inserted
		COMQBKLC	03	Locate block
		COMQBYLC	04	Byte locate
		COMQBYAL	05	Locate byte for updating
		COMQGRBA	06	Get RBA of last segment retrieved/inserted
		COMQFREE	07	Free space occupied by a segment
		COMQRKEY	08	Find key of HDAM root at block N
		COMQLOLD	09	Log data before change
		COMQLNEW	0A	Log data after change
		COMQBMON	0B	Turn bit maps on
		COMQBMOF	0C	Turn bit maps off
		COMQBFAL	0D	Mark buffer altered
		COMQULHB	0E	Set LO and HI block number for unload
		COMQXRMA	0F	Swap randomizer entry points
		COMQINT2	10	Initialize for part 2
		COMQINTU	11	Initialize for unload
		COMQRSTU	12	Reset after unload
		COMQINTR	13	Initialize for reload
		COMQRSTR	14	Reset after reload
		COMQCRAP	15	Clear HDAM root anchor point
		COMQGNDX	16	Retrieve an index record
129(81)	1	COMGOUT		Output control switches
		COMQSALL	80	Full statistics required
		COMQSUMM	40	Summary of statistics required t+0
		COMQSNON	20	No statistics to be produced
		COMQNPSB	01	No PSB to be generated

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
130(82)	1	COMGUCTL		Update process control switches
		COMQDUNQ	80	Q record update is complete
		COMQSPUS	40	Spill is in use
		COMQSPOF	20	Spill overflow has unprocessed records
		COMQNDBR	10	No database record in HDAM range
131(83)	1	COMGPART		Part in progress indicator
		COMQSTP1	80	Part 1 is in progress
		COMQSTP2	40	Part 2 is in progress
		COMQRIP	02	RESTART in progress
132(84)	4	COMCSTEC		SORT technique to be used
136(88)	9	COMCSSIZ		Main storage to be used by SORT
145(91)	9	COMCSMSG		SORT output message level
154(9A)	4	COMCSDIA		SORT diagnostic option
158(9E)	3	COMAMSGN		Error message numner to be printed
164(A4)	4	COMAVTXT		Address of variable text for message
168(A8)	4	COMFWRK1		First work word
172(AC)	4	COMFWRK2		Second work word
176(B0)	4	COMFWRK3		Third work word
180(B4)	4	COMFWRK4		Fourth work word
	D L / I	A D D R E S S	S E C T I O N	
184(B8)	4	COMASCD		Address of system contents directory (SCD)
188(BC)	4	COMAPST		Address of partition specification block
192(C0)	4	COMADDIR		Address of data base directory
196(C4)	4	COMALOG		Address of data base change logger
200(C8)	4	COMABUFH		Address of buffer handler router
204(CC)	4	COMASMGR		Address of space manager
208(D0)	4	COMAPREF		Address of prefix of last segment retrieved

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
212(D4)	4	COMRLSEG		RBA of last segment retrieved
216(D8)	4	COMRLOPT		Value of root PTB pointer at start of range
220(DC)	4	COMRHIPT		Value of root PTF pointer at end of range
224(E0)	4	COMADLI		Address of ASMTDLI
228(E4)	4			** Reserved **

F I L E   S E C T I O N

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
232(0E8)	36	COMSFL01		FCB for PRWRKF1
268(10C)	36	COMSFL02		FCB for PRWRKF2
304(130)	36	COMSFL03		FCB for PRWRKF3
340(154)	36	COMSFL04		FCB for PRWRKF4
376(178)	36	COMSFL05		FCB for PRWRKF5
412(19C)	36	COMSFL06		FCB for PRWRKF6
448(1C0)	36	COMSFL07		FCB for PRWRKF7
484(1E4)	36	COMSFL08		FCB for PRWRKF8
520(208)	36	COMSFL09		FCB for PRWRKF9
556(22C)	36	COMSFL10		FCB for PRWRKFA
592(250)	36	COMSFL11		FCB for SYSPRINT
628(274)	36	COMSFL12		FCB for SYSPUNCH
664(298)	36	COMSFL13		FCB for SYSIN

C H E C K P O I N T   S E C T I O N

Contains switches and data to be checkpointed and recovered during restart. Also includes the parameter list of user areas to be checkpointed for DL/I.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
700(2BC)	4	COMFCKID		ID of last DL/I checkpoint taken
704(2C0)	1	COMCPROC		PART2 phase in process indicator
		COMQPHUR	01	UNLOAD/RELOAD in progress
		COMQPHSC	02	SCAN in progress
		COMQPHSO	03	SORT in progress
		COMQPHUD	04	UPDATE in progress
705(2C1)	1	COMGPHAS		Phase GO/NOGO switches
		COMQSCAN	80	SCAN required
		COMQSRT1	40	SORT 1 required
		COMQSRT2	20	SORT 2 required

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		COMQSRT3	10	SORT 3 required
		COMQSRT4	08	SORT 4 required
		COMQUPDT	04	Update required
		COMQOPTN	02	option selection required
		COMQUPIX	01	Only index update required
706(2C2)	2	COMOCRGT		RGT offset for range being processed
708(2C4)	2	COMODBSN		DBT offset for DB being scanned
710(2C6)	1	COMGPHS2		restart flags
		COMQINBF	80	record in buffer for update
711(2C7)	1			** Reserved **
712(2C8)	8			** Reserved **
		COMQCKND		"*" end of area to be checkpointed

From COMFCKID to here is checkpoint data to be restored by DL/I extended restart. The fields that follow are the list of areas to checkpoint and recover. This list is passed to DL/I.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
720(2D0)	4	COMAPMCT		> parameter count
724(2D4)	4	COMACHXR		> EBCDIC function code (CHKP OR XRST)
728(2D8)	4	COMAIPCB		> I/O PCB
732(2DC)	4	COMALMXS		> Fullword value of COMHMXSG or 2K
736(2E0)	4	COMAIOWK		> 12 Byte work area
740(2E4)	4	COMAPLST		> Lengths and addresses to be checkpointed
744(2E8)	4	COMFCXPL		Length of checkpoint list
748(2EC)	4	COMFLCKD		Length of common checkpoint data
752(2F0)	4	COMACHKD		> checkpoint area origin
756(2F4)	4	COMAGBUF		> origin of combined GSAM I/O areas
760(2F8)	4	COMFFCBL		Length of PRWRKF2,3,4,5
764(2FC)	4	COMAFL25		> FCBS for PRWRKF2,3,4,5
768(300)	4	COMFPMCT		FW parameter count list
772(304)	4	COMLCXPL		Equate for end of parameter list
772(304)	12			** Reserved **

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
End of checkpoint restart parameter list				
D A T A S E T G R O U P T A B L E				
784(310)	4			
784(310)	8	COMCDDNM		DDNAME for a data set group to reorganize
792(318)	1	COMXDGID		DL/I data set group ID code
		COMLDSGT	09	"*-COMCDDNM" length of a DSG table entry
793(319)	729			Space for 9 more DSG entries
874(36A)	2			** Reserved **
876(36C)	16	COMCTRAC		Trace of last 16 requests to DLI services
892(37C)	16			** Reserved **
		COMLLEN		"*-COMSTART" length of common
P R I N T H E A D E R L I N E				
900(384)	121	COMHEADR		
908(38C)	43			
951(3B7)	36	COMHEADV		
987(3DB)	23	COMHEADC		
1010(3F2)	15	COMHEADD		
1025(401)	4	COMHEADP		
1029(405)	2	COMHPAGE		Page number packed
1031(407)	4	COMPAGEM		

CPAC - HDAM/HIDAM VARIABLE LENGTH SEGMENT COMPRESSION/EXPANSION ROUTINE INTERFACE TABLE

DSECT Name: DMBCPAC

This table is described as part of the general structure and description of the data management block (DMB). There is one entry for each compressible segment in the DMB.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBCPCNM	0(00)	
DMBCPCSG	8(08)	
DMBCPEP	16(10)	
DMBCPFLG	20(14)	
*DMBCPKEY	20(14)	02
DMBCPLNG	26(1A)	
*DMBCPNIT	20(14)	01
DMBCPRES	28(1C)	
*DMBCPSEQ	20(14)	08
DMBCPSGL	24(18)	
DMBCPSQF	21(15)	
DMBCPSQL	22(16)	
*DMBCPVLR	20(14)	04

RECORD LAYOUT - CPAC

Offset Dec (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBCPCNM		Segment Name
8(08)	4	DMBCPCSG		Compression routine name
16(10)	4	DMBCPEP		Entry point of compression routine
20(14)	1	DMBCPFLG		Flag byte
		DMBCPSEQ	08	Segment has a sequence field defined
		DMBCPVLR	04	Segment is variable
length				
		DMBCPKEY	02	Segment has key compression option
		DMBCPNIT	01	Initialization and termination processing required
21(15)	1	DMBCPSQF		Length of key field minus 1
22(16)	2	DMBCPSQL		Offset to sequence field
24(18)	2	DMBCPSGL		Maximum segment length
26(1A)	2	DMBCPLNG		Total length of CSECT - fixed lengths, constants, plus user data
28(1C)	4	DMBCPRES		Reserved for
intialization				

DACS - HDAM RANDOMIZING ROUTINE INTERFACE TABLE

DSECT Name: DMBDACS

The HDAM randomizing routine interface table is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBDABLK	16(10)	
DMBDABYC	24(18)	
DMBDABYM	20(14)	
DMBDACP	28(1C)	
DMBDAEP	9(09)	
DMBDAKL	8(08)	
DMBDANME	0(00)	
DMBDARAP	14(0E)	
DMBDASZE	12(0C)	

RECORD LAYOUT - DACS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBDANME		Name of address conversion algorithm load module
8(08)	1	DMBDAKL		Root Key length minus 1
9(09)	3	DMBDAEP		Entry point to conversion module
12(0C)	2	DMBDASZE		Size of this DSECT
14(0E)	2	DMBDARAP		Number of root anchor pointers per block
16(10)	4	DMBDABLK		Number of highest block directly addressable
20(14)	4	DMBDABYM		Maximum number of bytes per root before overflow outside of directly addressable area
24(18)	4	DMBDABYC		Current number of bytes consecutively inserted or loaded under root
28(1C)	4	DMBDACP		Result of last address conversion

DBT - DATA BASE TABLE

DSECT Name: DBT

This DSECT describes the data bases needed for the partial reorganization process. It is built during the DBD analysis phase and used by all subsequent phases in PART1 and PART2. Its address is held in the common area field (COMADBT).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DBTADBD	8 (008)	
DBTADMB	36 (024)	
DBTAJCB	16 (010)	
DBTAPCB	12 (00C)	
DBTASJCB	28 (01C)	
DBTASPCB	24 (018)	
DBTCID	48 (030)	
DBTCKEY	50 (032)	
DBTCNAME	0 (000)	
DBTFRASZ	40 (028)	
DBTGFLAG	49 (031)	
DBTHDMBN	46 (02E)	
DBTHRAPB	44 (02C)	
DBTHSIZ1	58 (03A)	
DBTLLEN	588	
DBTOSGT	78 (04E)	
DBTQHDAM	49 (031)	08
DBTQHIDM	49 (031)	04
DBTQHISM	49 (031)	10
DBTQSCAN	49 (031)	80
DBTQSOPT	49 (031)	40
DBTQVSAM	49 (031)	20
DBTQXPRI	49 (031)	02
DBTQXSEC	49 (031)	01

RECORD LAYOUT - DBT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	DBTSTART		
0(000)	8	DBTCNAME		Data base name
8(008)	4	DBTADBD		Address of loaded DMB or DBD
12(00C)	4	DBTAPCB		Address of primary PCB
16(010)	4	DBTAJCB		Address of JCB for primary PCB

20(014)	4			** Reserved **
24(018)	4	DBTASPCB		Address of second PCB for SCAN
28(01C)	4	DBTASJCB		Address of JCB for secondary PCB
32(020)	4			** Reserved **
36(024)	4	DBTADMB		Address of DMB
40(028)	4	DBTFRASZ		Number of blocks in root addressable area
44(02C)	2	DBTHRAPB		Number of root anchor points per block
46(02E)	2	DBTHDMBN		DMB number for this data base
48(030)	1	DBTCID		Data base internal ID
49(031)	1	DBTFLAG		DBT flag byte
		DBTQSCAN	80	Scan required, nct completed
		DBTQSOPT	40	Optional scan required
		DBTQVSAM	20	Access method is VSAM
		DBTQHISM	10	Entry is for HISAM data base
		DBTQHDAM	08	Entry is for HDAM data base
		DBTQHIDM	04	Entry is for HIDAM data base
		DBTQXPRI	02	Entry is for HIDAM prime index part
		DBTQXSEC	01	Entry is for secondary index data base
50(032)	8	DBTCKEY		Name of key field for root segment
58(03A)	3	DBTHSIZ1		Block sizs for first data set group
60(03C)	18			Block sizes for 9 more data set groups
78(04E)	510	DBTOSGT		Offsets in SGT for segments in this data base
588(24C)	4			Force fullword alignment
		DBTLLEN		*-DBTSTART length of a DBT entry

DDIR - DMB DIRECTORY

DSECT Name: DLZDDIR

The DMB directory contains an entry for every DMB (data management block) that can be accessed under DL/I control. The DMB directory is part of the DL/I nucleus and is created during DL/I system definition for online processing. The start address of the directory (SCDDLIDM) and entry length (SCDDLIDL) are contained in the system contents directory (SCD).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DDIRADDR	8 (08)	
*DDIRBAD	19 (13)	01
DDIRCNT	12 (0C)	
DDIRCODE	18 (12)	
DDIRCOD2	19 (13)	
DDIRDMBL	13 (0D)	
*DDIREXCL	19 (13)	10
*DDIREXSD	19 (13)	08
*DDIRGRP	19 (13)	02
*DDIRHSAM	19 (13)	20
*DDIRINOP	18 (12)	20
*DDIRKBRQ	18 (12)	10
DDIRLEN	24 (18)	
*DDIRNDMB	19 (13)	80
*DDIRNOSC	18 (12)	04
*DDIRNOUP	18 (12)	01
*DDIRNRAN	19 (13)	40
DDIRNUMB	16 (10)	
*DDIROPEN	18 (12)	40
DDIRPPST	21 (15)	
*DDIRSECL	18 (12)	80
DDIRSYM	0 (00)	
DDIRVSRT	20 (14)	
*DDIRWAIT	18 (12)	08
*DDIR1GRP	19 (13)	04

RECORD LAYOUT - DDIR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	DDIR		Label to establish entry address
0(00)	8	DDIRSYM		DMB name - converted from DBDNAME supplied during DBDGEN
8(08)	4	DDIRADDR		DMB address
12(0C)	1	DDIRCNT		Number of users scheduled for this DMB
13(0D)	3	DDIRDMBL		Storage required for this DMB
16(10)	2	DDIRNUMB		DMB number of this DMB
18(12)	1	DDIRCODE		DMB code
		DDIRSECL	80	Security locked
		DDIROPEN	40	At least one ACB is opened
		DDIRINOP	20	DMB to be opened during online initialization or during start call
		DDIRKBRQ	10	Buffer pool space required for this KSDS
		DDIRWAIT	08	System task waiting for zero DDIRCNT
		DDIRNOSC	04	Do not schedule this DMB because it is stopped
		DDIRNOUP	01	No PSBs referencing the DMB with other than GO or GOP PROCOPT were loaded
19(13)	1	DDIRCOD2		DMB code byte 2
		DDIRNDMB	80	DMB not present in library
		DDIRNRAN	40	Requested randomizing module not present in library
		DDIRHSAM	20	This DMB for HSAM
		DDIREXCL	10	This DMB being used exclusively
		DDIREXSD	08	Exclusive control required for scheduling
		DDIR1GRP	04	DMB first in shared index
		DDIRGRP	02	DMB belongs to shared index
		DDIRBAD	01	DMB initialization failed
20(14)	1	DDIRVSRT		R15 VSAM return code
21(15)	3	DDIRPPST		PPST address in DMB is used exclusively
24(18)		DDIRLEN		Length of one DDIR entry

DIB - DL/I INTERFACE BLOCK

DSECT Name: DIB

This DSECT describes the HLPI DL/I system interface block fields.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DIBCNTAD	144 (090)	
DIBCOUNT	100 (064)	
DIBEIPAD	8 (008)	
DIBGPATH	110 (06E)	40
DLBHLPIA	140 (08C)	
DLBID	0 (000)	
DIBIO	88 (058)	
DIBIOSIZ	92 (05C)	
DIBLUDIB	136 (088)	
DIBMSG	124 (07C)	
DIBMSGRC	132 (094)	
DIBMSGSC	128 (080)	
DIBNOPCB	112 (070)	
DIBPARM	144 (090)	
DIBPARMA	184 (0B8)	
DIBPARMB	188 (0BC)	
DIBPARMC	192 (0C0)	
DIBPARMD	196 (0C4)	
DIBPARME	200 (0C8)	
DIBPARMF	204 (0CC)	
DIBPARMG	208 (0D0)	
DIBPARMH	212 (0D4)	
DIBPARMI	216 (0D8)	
DIBPARM1	148 (094)	
DIBPARM2	152 (098)	
DIBPARM3	156 (09C)	
DIBPARM4	160 (0A0)	
DIBPARM5	164 (0A4)	
DIBPARM6	168 (0A8)	
DIBPARM7	172 (0AC)	
DIBPARM8	176 (0B0)	
DIBPARM9	180 (0B4)	
DIBPATHC	108 (06C)	
DIBPATHP	120 (078)	
DIBPCBAD	84 (054)	
DIBPCBNO	116 (074)	
DIBPRCNT	104 (068)	
DIBPSIZE	96 (060)	
DIBPSPLI	110 (06E)	80
DIBRBKWD	12 (00C)	10
DIBREGSV	12 (00C)	
DIBRERC	12 (00C)	18
DIBRTNCD	114 (072)	
DIBS	0 (000)	
DIBSFLAG	110 (06E)	
DIBSLEN	220 (0DC)	DC
DIBSSAS	160 (0A0)	
DIBTOTN	111 (06F)	
DLZSDIB	0 (000)	

RECORD LAYOUT - DIB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	0	DLZSDIB		
0(000)	4	DIBS		System DIB
0(000)	8	DIBID		DIB identifier 'DLZSDIB'
8(008)	4	DIBEIPID		EXEC interface program Address (batch/MPS only
12(00C)	72	DIBREGSV		Register save area
		DIBRBKWD		"DIBREGSV+4" SAVEAREA backward pointer
		DIBRERC		"DIBREGSV+12" Savearea for registers 14 through 12
84(054)	4	DIBPCBAD		PCB address list address
88(058)	4	DIBIO		Address of EIP common I/O area
92(05C)	4	DIBIOSIZ		Size of EIP common I/O area
96(060)	4	DIBPSIZE		I/O area size required on call
100(064)	4	DIBCOUNT		DL/I call parameter count
104(068)	4	DIBPRCNT		Previous get path call DL/I parameter count
108(06C)	8	DIBPATHC		Data transfer segment count
110(06C)	4	DIBSFLAG		Flag byte
		DIBPSPLI	80	PSB generated for PL/I program (online only)
		DIBGPATH	40	Previous call was a get path call
111(06F)	1	DIBTOTN		Number of calls on previous get path call
112(070)	2	DIBNOPCB		Maximum PCB index
114(072)	2	DIBRTNCD		Falling return code
116(074)	2	DIBPCBNO		PSB number for current call
118(076)	2			** Reserved **
120(078)	4	DIBPATHP		Address of path call header control blocks
124(07C)	4	DIBMSG		Address of message number

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
128(080)	4	DIBMSGSC		Address of DL/I status code
132(084)	4	DIBMSGRC		Address of failing return code address of statement identifier
136(088)	4	DIBLUDIB		Address of last user DIB
140(08C)	4	DIBHLPIA		Address of HLPI parameter list
144(090) list	4	DIBPARM		Start of call parameter
144(090)	4	DIBCNTAD		Address of parameter count
148(094)	4	DIBPARM1		PARM 1 = A(function)
152(098)	4	DIBPARM2		PARM 2 = A(PCB)
156(09C)	4	DIBPARM3		PARM 3 = A(IOAREA)
160(0A0)	4	DIBSSAS		Start of SSAS
160(0A0)	4	DIBPARM4		PARM 4 = A(SSA1)
164(0A4)	4	DIBPARM5		PARM 5 = A(SSA2)
168(0A8)	4	DIBPARM6		PARM 6 = A(SSA3)
172(0AC)	4	DIBPARM7		PARM 7 = A(SSA4)
176(0B0)	4	DIBPARM8		PARM 8 = A(SSA5)
180(0B4)	4	DIBPARM9		PARM 9 = A(SSA6)
184(0B8)	4	DIBPARMA		PARM 10 = A(SSA7)
188(0BC)	4	DIBPARMB		PARM 11 = A(SSA8)
192(0C0)	4	DIBPARMC		PARM 12 = A(SSA9)
196(0C4)	4	DIBPARMD		PARM 13 = A(SSA10)
200(0C8)	4	DIBPARME		PARM 14 = A(SSA11)
204(0CC)	4	DIBPARMF		PARM 15 = A(SSA12)
208(0D0)	4	DIBPARMG		PARM 16 = A(SSA13)
212(0D8)	4	DIBPARMH		PARM 17 = A(SSA14)
216(0DC)	4	DIBPARMI		PARM 18 = A(SSA15)
220(0DC)	4			Length if fullword multiple
		DIBSLEN		"*-DIBS" Length of system

DMB - DMB PREFIX

DSECT Name: DMB

The DMB prefix is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBDALGR	12(0C)	
*DMBHD	6(06)	06
*DMBHI	6(06)	07
*DMBHSAM	6(06)	05
*DMBIMSC	10(0A)	80
*DMBISAM1	6(06)	02
DMBLDDCB	7(07)	
DMBLENTB	2(02)	
*DMBNDEX	6(06)	08
DMBNREF	12(0C)	
DMBORG	6(06)	
DMBPDATA	8(08)	
DMBPFLG	10(0A)	
DMBPPRLN	16(10)	
DMBPPRND	16(10)	
DMBSECTB	4(04)	
*DMBSHIS	6(06)	01
DMBSIZE	0(00)	
*DMBSSAM	6(06)	04
*DMBV11	0(00)	80

RECORD LAYOUT - DMB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	2	DMBSIZE		DMB size
		DMBV11	80	DL/I version 1.1 or later
2(02)	2	DMBLENTB		Offset from DMB to first PSDB (DMBPSDB)
4(04)	2	DMBSECTB		Offset from DMB to end of PSDBs + 1
6(06)	1	DMBORG		DMB organization
		DMBSHIS	01	Simple HISAM
		DMBISAM1	02	HISAM
		DMBSSAM	04	Simple HSAM
		DMBHSAM	05	HSAM
		DMBHD	06	HDAM
		DMBHI	07	HIDAM
		DMBNDEX	08	Index data base
7(07)	1	DMBLDDCB		ACB number (minus 1) of sequential data set used to write index records on data base load
8(08)	2	DMBPDATA		Length of system data in index data base (protected)
10(0A)	1	DMBPFLG		Flag byte
		DMBIMSC	80	IMS compitability required
11(0B)	1			** Reserved **
12(0C)	1	DMBNREF		Number of entries in external reference table
12(0C)	4	DMBDALGR		Address of direct algorithm communication table if HDAM (DMBDACS); LRECL number if HSAM
16(10)		DMBPPRND		End + 1 of DMB prefix. This is also the address of the first ACB extension (DMBACBXT)
16(10)		DMBPPRLN		Length of DMB prefix (DMBPPRND minus DMB)

DPPCB - PCB DOPE VECTOR TABLE

DSECT Name: DPPCB

The PCB dope vector table is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DPPCBDBD	0(00)	
DPPCBJCB	32(20)	
DPPCBKFD	52(34)	
DPPCBLEV	8(08)	
DPPCBLKY	44(2C)	
DPPCBPRO	28(18)	
DPPCBSFD	36(24)	
DPPCBSTC	16(10)	
DPPCPNSS	48(30)	

RECORD LAYOUT - DPPCB

Offset Dec(Hex)	Length	Field Name	Meaning
0(00)	4	DPPCBDBD	The address of the location that contains DBPCBDBD
4(04)	2	Maximum Length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
6(06)	2	Current length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
8(08)	4	DPPCBLEV	The address of the location that contains DBPCBLEV
12(0C)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit

Offset Dec(Hex)	Length	Field Name	Meaning
14(0E)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
16(10)	4	DPPCBSTC	The address of the location that contains DBPCBSTC
20(14)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
22(16)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
24(18)	4	DPPCBPRO	The address of the location that contains DBPCBPRO
28(1C)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
30(1E)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
32(20)	4	DPPCBJCB	The address of the location that contains DBPCBJCB
36(24)	4	DPPCBSFD	The address of the location that contains DBPCBSFD
40(28)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
42(2A)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string

Offset Dec(Hex)	Length	Field Name	Meaning
44(2C)	4	DPPCBLKY	The address of the location that contains DBPCBLKY
48(30)	4	DPPCPNSS	The address of the location that contains DBPCBNSS
52(34)	4	DPPCBKFD	The address of the location that contains DBPCBKFD
56(38)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
58(3A)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string

DSG - DATA SET GROUP

DSECT Name: DSG

The DSG is described as part of the general structure and description of the program specification block (PSB).

Note: With the exception of the first three characters of each field/flag name (DSG instead of JCB) the layout of the data set group is identical to the layout of the 'DSG Section' of the job control block (JCB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*DSGBLDEL	15(0F)	80
DSGBOFF	12(0C)	
*DSGCOMMD	16(10)	02
*DSGCONST	15(0F)	20
*DSGDATX	16(10)	40
DSGDCBA	0(00)	
DSGDCBNO	6(06)	
DSGDMBNO	4(04)	
DSGDSGLN	28(1C)	
*DSGDSOHD	7(07)	20
*DSGDSOHI	7(07)	10
*DSGDSOHS	7(07)	02
*DSGDSOH1	7(07)	04
*DSGDSOLS	7(07)	80
*DSGDSORI	7(07)	44
*DSGDSOUP	7(07)	01
*DSGDUPS	15(0F)	08
*DSGHDULD	15(0F)	40
DSGHSADD	8(08)	
*DSGHSWLR	15(0F)	01
DSGINDA	7(07)	
DSGINDB	14(0E)	
DSGINDC	15(0F)	
DSGINDG	16(10)	
DSGLROOT	24(18)	
DSGNOSAM	20(14)	
*DSGPADKY	15(0F)	10
*DSGPREM	16(10)	80
*DSGRETD	16(10)	04
*DSGVL	16(10)	08
*DSGXP	16(10)	10

RECORD LAYOUT - DSG

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	DSGDCBA		Address of the ACB extension for this data set (KSDS ACB extension if HISAM)
4(04)	2	DSGDMBNO		DMB number for this DSG
6(06)	1	DSGDCBNO		ACB number of ACB in DMB (KSDS ACB number if HISAM)
7(07)	1	DSGINDA		JCB indicators
		DSGDSOLS	80	This is last DSG in JCB
		DSGDSORI	44	Data set group is root in index
		DSGDSOHD	20	Data set group is HDAM
		DSGDSOHI	10	Data set group is HIDAM
		DSGDSOH1	04	Data set group is HISAM or simple HISAM
		DSGDSOHS	02	Data set group is HSAM or simple HSAM
		DSGDSOUP	01	Data set group is SHSAM or SHISAM
8(08)	4	DSGHSADD		HSAM I/O area after open
12(0C)	2	DSGBOFF		HSAM block size
14(0E)	1	DSGINDB		(Not used in DL/I DOS/VS)
15(0F)	1	DSGINDC		JCB indicators
		DSGBLDEL	80	Delete/replace DSG
		DSGHDULD	40	HD unload is running
		DSGCONST	20	Index data set contains constant
		DSGPADKY	10	Search argument not equal to key length
		DSGDUPS	08	Nonunique secondary index keys
		DSGHSWLR	01	HSAM wrong length record
16(10)	1	DSGINDG		DSG indicators - retrieve's variable length flags
		DSGPREM	80	Segment prefix moved to work area
		DSGDATX	40	Segment completely expanded
		DSGXP	10	Force complete segment expansion
		DSGVL	08	The variable length routine has been entered for segment
		DSGRETD	04	Data return call
		DSGCOMMD	02	Path return call
17(11)	3			**Reserved**
20(14)	4	DSGNOSAM		Retrieve's HSAM ID

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
24(18)	4	DSGLROOT		RBA of current root
28(1C)		DSGDSGLN		Length of each DSG section of JCB

DWR - DATA WORK RECORD

DSECT Name: DWR

This DSECT has the following uses:

1. Record the old and new location of a segment.
2. Record the location and old value of a pointer that may have to be updated.

These records are created by RELOAD and SCAN. The same format is used by UPDATE for its spill table and file.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DWRCDSG	11(00B)	
DWRCRDB	12(00C)	
DWRCRDSG	13(00D)	
DWRCSKEY	12(00C)	
DWRCSORT	11(00B)	
DWRCTYPE	10(00A)	
DWRLLEN	14(00E)	12
DWROACT	8(008)	
DWRRCOMP	4(004)	
DWRRMOVE	0(000)	
DWRRUPDT	14(00E)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	DWRSTART		
0(00) segment	4	DWRRMOVE		New RBA of a moved
4(04)	4	DWRRCOMP		Old RBA of a segment for compare
8(08)	2	DWROACT		Offset in ACT that built this record
10(00A)	1	DWRCTYPE		Record type code
11(00B)	1	DWRCSORT		Minor sort key
11(00B)	1	DWRCDSG		Data set group of moved segment in K record
12(00C)	1	DWRCSKEY		Update sort key: DB, ID, DSG, RBA
12(00C)	1	DWRCRDB		Data base ID of segment to be updated
13(00D)	4	DWRCRDSG		Data set group ID of segment to be updated
14(00E)	4	DWRRUPDT		RBA of segment to be to be updated
		DWRLLEN	12	*-DWRSTART

EIPL - EXEC INTERFACE PROGRAM PARAMETER LIST

DSECT Name: EIPL

This DSECT describes the DL/I HLPI interface program parameter list fields.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
EIPABEND	8 (008)	
EIPEPB0	4 (004)	
EIPERMSG	0 (000)	
EIPFLAG	28 (01C)	
*EIPLLEN	29 (01D)	20
*EIPMPS	28 (01C)	20
EIPPARML	0 (000)	
EIPPCBL	12 (00C)	
EIPPLILN	16 (010)	
*EIPPLIPG	28 (01C)	40
*EIPPLIPS	28 (01C)	80
EIPPLISA	20 (014	
EIPSDIB	24 (018)	
*EIPSPCLN	20 (014)	0C

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	EIPPARML		DL/I-EIP parameter list
0(000)	4	EIPERMSG		Address of DL/I message Routine (Online/Batch/MPS)

The following fields are used only in batch/MPS environment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
4(004)	4	EIPEPB0		"V(DLZEIPI)" address of EXEC interface program (DLZEIPB0)
8(008)	4	EIPABEND		Address of DL/I ABEND routine

NOTE: The following fields must remain in the following order:

1. Address of the PCB list
2. Pointer to length of initial storage area
3. Address of initial storage area

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
12(00C)	4	EIPPCBL		Address of PCB list
16(010)	4	EIPPLILN		Pointer to length of initial storage area

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
20(014)	4	EIPPLISA		Address of initial storage area
		EIPSPCLN		"-*EIPPCBL" length of PL/I parameter list
24(018)	4	EIPSDIB		Address of system DIB
28(01C)	1	EIPFLAG		Flag byte
		EIPPLIPS	80	PSB generated for PL/I program
		EIPMPS	20	MPS environment
29(01D)	3			** Reserved **
		EIPLLEN		"*-EIPPARML" EIP parameter list length

FCB - FILE CONTROL BLOCK

DSECT Name: FILECB

This DSECT describes the fields used to control one file used by the partial reorganization utility . It is passed as a parameter to the work file manager.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
FCBABUF	4 (004)	
FCBADTF	0 (000)	
FCBAEOD	8 (008)	
FCBFRECT	12 (00C)	
FCBGREQU	31 (01F)	
FCBGSTAT	30 (01E)	
FCBHBLKS	26 (01A)	12
FCBHLLRL	20 (014)	
FCBHLREC	24 (018)	
FCBLLEN	31 (01F)	20
FCBOCREC	28 (01C)	
FCBQCLOS	31 (01F)	08
FCBQGET	31 (01F)	20
FCBQINPT	30 (01E)	80
FCBQOPNI	31 (01F)	80
FCBQOPNO	31 (01F)	40
FCBQOUTP	30 (01E)	40
FCBQPUT	30 (01E)	10
FCBSTART	0 (000)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	FCBSTART		
0(00)	4	FCBADTF		Address of the DTF for this file
4(04)	4	FCBABUF		Address of the current record
8(08)	2	FCBAEOD		Address of the end of data routine
12(00C)	4	FCBFRECT		Number of records read or written
16(010)	4			** Reserved **
20(014)	2	FCBHLLRL		Last logical record length
22(016)	2			Unused
24(018)	2	FCBHLREC		Logical record length

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
26(01A)	2	FCBHBLKS		Physical block size
28(01C)	2	FCBOCREC		Offset of current record in block
30(01E)	1	FCBGSTAT		File status flag
		FCBQINPT	80	File is in input mode
		FCBQOUTP	40	File is in output mode
31(01F)	1	FCBGREQU		Request flags
		FCBQOPNI	80	Open file for input
		FCBQOPNO	40	Open file for output
		FCBQGET	20	Get next record
		FCBQPUT	10	Put a record
		FCBCLOS	08	Close the file
		FCBLLEN		*-FCBSTART" Length of a FCB entry

FDB - FIELD DESCRIPTION BLOCK

DSECT Name: FDB

The field description block (FDB) is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*FDBCHAR	10 (0A)	03	
FDBDCENF	10 (0A)		
FDBEND	12 (0C)		(See XDFLD fields)
*FDBEQOK	10 (0A)	20	
FDBFLENG	11 (0B)		
*FDBFP	10 (0A)	04	
*FDBHEX	10 (0A)	01	
*FDBKEY	10 (0A)	40	
*FDBLAST	10 (0A)	80	
FDBLEN	11 (0B)		(See DFLD fields)
FDBOFFCK	8 (08)		(See /CK fields)
FDBOFFST	8 (08)		
*FDBPACK	10 (0A)	02	
*FDBSPEC	10 (0A)	10	
FDBSYMBL	0 (00)		
FDBSYSLN	10 (0A)		(See /CK fields
FDBSYSNM	0 (00)		(See /CK fields)
*FDBTYPE	10 (0A)	07	
*FDBXDCON	10 (0A)	08	(See XDFLD fields)
*FDBXDEQ	10 (0A)	01	(See XDFLD fields)
FDBXDFLG	10 (0A)		(See XDFLD fields)
FDBXDLEN	12 (0C)		(See XDFLD fields)
*FDBXDLST	10 (0A)	80	(See XDFLD fields)
FDBXDNM	0 (00)		(See XDFLD fields)
FDBXDSEC	8 (08)		(See XDFLD fields)
*FDBXDSPC	10 (0A)	10	
*FDBXDSSQ	10 (0A)	04	(See XDFLD fields)
*FDBXDSSS	10 (0A)	20	
*FDBXDSYM	10 (0A)	40	
*FDBZD	10 (0A)	07	

RECORD LAYOUT - FDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FDBSYMBL		Symbolic name
8(08)	2	FDBOFFST		Field offset from segment beginning
10(0A)	1	FDBDCENF		Flags
		FDBLAST	80	Last FDB for this segment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		FDBKEY	40	This is segment's sequence field
		FDBEQOK	20	Duplicate sequence fields allowed
		FDBSPEC	10	Special FDB (XDFLD, /CK, or /SK)
		FDBTYPE	07	Field format bits
		FDBZD	07	Field is zoned decimal
		FDBFP	04	Field is floating point
		FDBPACK	02	Field is packed decimal
		FDBHEX	01	Field is hexadecimal
		FDBCHAR	03	Field is character
11(0B)	1	FDBFLENG		Executable field length
This describes the /CK system-related field				
0(00	3	FDBSYSNM		Constant '/CK'
3(03)	5			Remainder of field name
8(08)	2	FDBOFFCK		Offset from beginning of concatenated key
10(0A)	2	FDBSYSLN		Bits 0-3 = X'0001'; Bits 4-15 = length minus 1
This describes the XDFLD				
0(00)	8	FDBXDNM		FDB Name
8(08)	2	FDBXDSEC		Offset to secondary list for this index
10(0A)	1	FDBXDFLG		Flags
		FDBXDLST	80	Last FDB
		FDBXDSYM	40	Pointer is symbolic
		FDBXDSSS	20	Pointer is contained in SOURCE/SUBSEQ data
		FDBXDSPC	10	Special FDB
		FDBXDCON	08	Constant present
		FDBXDSSQ	04	SUBSEQ present
		FDBXDEQ	01	Index segment same as index source segment
11(0B)	1	FDBXDLEN		Length of search field
12(0C)		FDBEND		End of FDB entry
12(0C)		FDBLEN		Length of FDB entry (FDBEND minus FDBSYMBL)

FER - FIELD EXIT ROUTINE INTERFACE LIST

DSECT Name: FER

The FER (Field Exit Routine Interface List) is used to pass information to the named user-written exit routine whenever a designated field is to be processed.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Code (Char)
FERPSCS	2 (002)	
*FERPCSCT	2 (002)	B
*FERPCSFE	2 (002)	C
*FERPCSNT	2 (002)	A
*FERPCSOK	2 (002)	
*FERPCSTC	2 (002)	D
FERPEC	0 (000)	
FERPFNCT	1 (001)	
FERPFSBA	28 (01C)	
*FERPGET	0 (000)	G
*FERPINS	1 (001)	I
FERPLEN	80 (050)	
FERPPFA	12 (00C)	
FERPPFL	10 (00A)	
FERPPSA	4 (004)	
*FERPPUT	0 (000)	P
*FERPREP	1 (001)	R
*FERPRET	1 (001)	G
*FERPSSA	1 (001)	S
FERPUFA	24 (018)	
FERPUFL	22 (016)	
FERPUSA	16 (010)	
FERPUWA	32 (020)	
*FERPXDF	1 (001)	X

RECORD LAYOUT - FER

Offset Dec(Hex)	Length	Field/Flag Name	Code (Char)	Meaning
0(00)	1	FERPEC		Entry code
		FERPGET	G	Get function
		FERPPUT	P	Put function
1(01)	1	FERPFNCT		Function code
		FERPRET	G	Retrieve segment conversion
		FERPINS	I	Insert
		FERPREP	R	Replace
		FERPSSA	S	Retrieve SSA conversion
		FERPXDF	X	Retrieve SSA conversion for XDFLD
2(02)	1	FERPCSC		Conversion status code
		FERPCSOK		OK
		FERPCSNT	A	Numeric truncation error
		FERPCSCT	B	Character truncation
error				
		FERPCSFE	C	Format error
		FERPCSTC	D	Type conflict
3(03)	1			**Reserved**
4(04)	4	FERPPSA		Physical segment address (if variable length, points to two byte length field)
8(08)	2			**Reserved**
10(0A)	2	FERPPFL		Physical field length (zero if virtual field)
12(0C)	4	FERPPFA		Physical field address (zero if virtual field)
16(10)	4	FERPUSA		User segment address
20(14)	2			**Reserved**
22(16)	2	FERPUFL		User field length
24(18)	4	FERPUFA		User field address
28(1C)	4	FERPFSBA		FSB address
32(20)	48	FERPUWA		User work area
80(50)	0	FERPLEN		Length of field exit routine interface list

FERT - FIELD EXIT ROUTINE TABLE

DSECT Name: FERT

The FERT (Field Exit Routine Table) is used to hold information about a user-written exit routine.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code (Hex)
*FERTDUMP	20 (014)	80
FERTFLAG	20 (014)	
FERTLEN	24 (018)	
FERTNAME	0 (000)	
FERTPRES	16 (010)	
FERTRTEP	8 (008)	
FERTRLTG	12 (00C)	

RECORD LAYOUT - FERT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FERTNAME		Module name
8(08)	4	FERTRTEP		Module entry point
12(0C)	4	FERTRTLG		Module length
16(10)	4	FERTPRES		Pointer to next FERT entry
20(14)	1	FERTFLAG		
		FERTDUMP	80	Control block dumped
21(15)	3			**Reserved**
24(18)	0	FERTLEN		Length of field exit routine table

FLD - FIELD LEVEL DESCRIPTOR

DSECT Name: FLD

The FLD (Field Level Descriptor) block is used to hold information about fields, operators and connectors.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code (Hex)
*FLDDATA1	0(00)	80
*FLDDPAR	0(00)	08
FLDEND	5(05)	08
FLDFLENG	5(05)	
FLDF1	0(00)	
*FLDKEY1	0(00)	44
*FLDLCH	0(00)	04
*FLDMBR	1(01)	
FLDLENG	5(05)	
*FLDMEMAD	1(01)	04
*FLDMEMAQ	1(01)	80
*FLDMEMGT	1(01)	20
*FLDMEMLT	1(01)	40
*FLDMEMNE	1(01)	60
*FLDMEMOR	1(01)	02
*FLDMEMRP	1(01)	01
*FLDNOCOV	0(00)	20
*FLDNXYSM	0(00)	10
FLDSSAOF	2(02)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	FLDF1		Field flags
		FLDDATA	80	Field qualified on data
		FLDKEY1	40	Field qualified on key
		FLDNOCOV	20	No conversion for this
field				
		FLDNXTSM	10	Next field is the same
		FLDDPAR	08	Field destination parent
		FLDLCH	04	Field in logical child
1(01)	1	FLDMBR		Encode operators/connectors
		FLDMEMEQ	80	Operator has = sign
		FLDMEMLT	40	Operator has < sign
		FLDMEMGT	20	Operator has > sign
		FLDMEMNE	60	Operator is not equal
		FLDMEMAD	04	AND connector
		FLDMEMOR	02	OR connector
		FLDMEMRP	01	Right parenthesis
2(02)	2	FLDSSAOF		Offset to value area in SSA for this field
4(04) field	1	FLDFLENG		Executable length of
5(05)	3			** Reserved **
		FLDEND		End of field level descriptor
		FLDLENG		Length of each FLD entry (FLDEND-FLD)

FSB - FIELD SENSITIVITY BLOCK

DSECT Name: FSB

The FSB (Field Sensitivity Block) is used to hold information about a field which has been defined with a SENFLD statement during PSBGEN.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
FSBCHAIN	28 (01C)	
*FSBCHAR	10 (00A)	03
*FSBCR	11 (00B)	20
*FSBDPF	10 (00A)	10
*FSBEQOk	10 (00A)	20
FSBFDBP	0 (000)	
*FSBFER	16 (010)	20
FSBFERTA	24 (018)	
FSBFLAG	11 (00B)	
FSBFLDNM	0 (000)	
*FSBFP	10 (00A)	04
*FSBHEX	10 (00A)	01
*FSBIV	16 (010)	40
FSBIVA	20 (014)	
*FSBKEY	10 (00A)	40
*FSBLAST	10 (00A)	80
FSBLEN	32 (020)	
*FSBNR	16 (010)	08
*FSBOVF	11 (00B)	40
*FSBPACK	10 (00A)	02
FSBPCHA	4 (004)	
FSBPWYAD	6 (006)	
FSBPVLEN	12 (00C)	
FSBPVLOK	8 (008)	
FSBPVTYP	10 (00A)	
*FSBSSA	11 (00B)	80
*FSBTYPE	10 (00A)	07
*FSBUCHAR	16 (010)	03
*FSBUFP	16 (10)	04
*FSBUHEX	16 (10)	01
*FSBUPACK	16 (10)	02
FSBUVLEN	18 (12)	
FSBUVLOC	14 (0E)	
FSBUVTYP	16 (10)	
*FSBUZD	16 (10)	07
*FSBVF	16 (10)	10
*FSBZD	1 0(0A)	07

RECORD LAYOUT - FSB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FSBFLDNM		Field name
0(00)	4	FSBFDBP		FDB address (ACBGEN only)
4(04)	2	FSBPCHA		Physical view chain pointer (ACBGEN only)
6(06)	2	FSBPHYAD		Field physical adjustment factor (ACBGEN only)
8(08)	2	FSBPVLOC		Displacement in physical segment
10(0A)	1	FSBPVTYP		Physical field type
		FSBLAST	80	Last FSB
		FSBKEY	40	Sequence field
		FSBEQOK	20	Duplicate sequence allowed
		FSBDPF	10	Field is in destination parent
		FSBTYPE	07	Field format bits
		FSBZD	07	Field format is zoned decimal
		FSBFP	04	Field format is floating point
		FSBCHAR	03	Field format is character
		FSBPACK	02	Field format is packed decimal
		FSBHEX	01	Field format is binary
11(0B)	1	FSBFLAG		Flags
		FSBSSA	80	Field may be used in an SSA
		FSBOVF	40	Field has subfields
		FSBCR	20	Conversion required
12(0C)	2	FSBPVLEN		Physical field length (executable)
14(0E)	2	FSBUVLOC		Field displacement in user's view
16(10)	1	FSBUVTYP		User's field type
		FSBIV	40	Initial value specified
		FSBFER	20	Field exit routine specified
		FSBVF	10	Field is virtual
		FSBNR	08	Replace prohibited
		FSBUZD	07	User field format is zoned decimal
		FSBUFP	04	User field format is floating point
		FSBUCHAR	03	User field format is character
		FSBUPACK	02	User field format is packed decimal
		FSBUHEX	01	User field format is binary

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
17(11)	1			**Reserved**
18(12)	2	FSBUVLEN		User's field length (executable)
20(14)	4	FSBIVA		Pointer to specified initial value
24(18)	4	FSBFERTA		Field exit routine table entry address
28(1C)	4	FSBCHAIN		Chain pointer for ACBGEN
32(20)	0	FSBLEN		Length of FSB entry

HLPIL - HIGH LEVEL PROGRAM INTERFACE PARAMETER LIST

DSECT Name: HLPIL

This DSECT describes the fields contained in the HLPI parameter list.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code (Hex)
DLZHLPIL	0 (000)	
HLPIARGO	0 (000)	
HLPICKID	8 (008)	
HLPIDIBP	4 (004)	
HLPIFLDN	36 (024)	
HLPIFLDV	40 (028)	
HLPILFLD	44 (02C)	
HLPILIOA	20 (014)	
HLPINFLD	28 (01C)	
HLPIOFST	24 (018)	
HLPIOPER	32 (020)	
HLPIPCBI	8 (008)	
HLPIPSBN	8 (008)	
HLPISEGN	12 (00C)	
HLPISIOA	16 (010)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	HLPIARGO		Address of ARGO parameter list
4(04)	4	HLPIDIBP		Address of user DIB
8(08)	4	HLPIPSBN		Pointer to PSBNAME for scheduling call
8(08)	4	HLPICKID		Pointer to checkpoint ID for checkpoint call
8(08)	4	HLPIPCBI		Pointer to PCB index number1
12(0C)	4	HLPISEGN		Pointer to segment name
16(10)	4	HLPISIOA		Address of the segment I/O area
20(14)	4	HLPILIOA		Pointer to the length of the I/O area
24(18)	4	HLPIOFST		Pointer to the length of the variable destination parent
28(1C)	4	HLPINFLD		Pointer to the number of fields (always 1 for DL/I)
32(20)	4	HLPIOPER		Pointer to the relational operators
36(24)	4	HLPIFLDN		Pointer to field name
40(28)	4	HLPIFLDV		Pointer to field value
44(2C)	4	HLPILFLD		Pointer to length of the field value

JCB - JOB CONTROL BLOCK

DSECT Name: JCB

The JCB is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBALD	42 (2A)	40
*JCBALLEX	64 (40)	04
*JCBALQD	42 (2A)	80
*JCBBLDEL	179 (B3)	80
JCBBOFF	176 (B0)	
*JCBCCALL	41 (029)	04
JCBCODE	60 (03C)	
*JCBCOMMD	180 (0B4)	02
*JCBCONST	179 (0B3)	20
*JCBDATX	180 (0B4)	40
JCBDCBA	164 (0A4)	
JCBDCBNO	170 (0AA)	
*JCBDEFDL	60 (03C)	40
*JCBDLET	148 (094)	02
JCBDMBNO	168 (0A8)	
*JCBDOPI	64 (040)	08
JCBDSGLN	188 (0BC)	
*JCBDSOHD	171 (0AB)	20
*JCBDSOHI	171 (0AB)	10
*JCBDSOHS	171 (0AB)	02
*JCBDSOH1	171 (0AB)	04
*JCBDSOLS	171 (0AB)	80
*JCBDSORI	171 (0AB)	44
*JCBDSOUP	171 (0AB)	01
*JCBDUPS	179 (0B3)	08
*JCBFLS	64 (040)	01
*JCBFNSL	43 (02B)	80
*JCBHDULD	179 (0B3)	40
JCBHSADD	172 (0AC)	
*JCBHSWLR	179 (0B3)	01
JCBINDA	171 (0AB)	
JCBINDB	178 (0B2)	
JCBINDC	179 (0B3)	
JCBINDG	180 (0B4)	
*JCBISRT	148 (094)	01
*JCBKEYX	180 (0B4)	20
JCBLEVND	4 (004	
JCBLEVTB	0 (000)	
JCBLEV1C	32 (020)	
JCBLROOT	188 (0BC)	
*JCBLSSAQ	40 (028)	02
JCBLVC	65 (041)	
JCBLVT	64 (040)	
*JCBLV1C	41 (029)	01
JCBMKYL	38 (026)	
*JCBMLPOS	60 (03C)	08
*JCBMSSA	40 (028)	10

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBMUSSA	40 (028)	08
*JCBNODEQ	148 (094)	80
JCBNOSAM	184 (0B8)	
*JCBNSSA	40 (028)	80
*JCBNTFD	148 (094)	08
*JCBOPEN	61 (03D)	80
*JCBORGHD	61 (03D)	20
*JCBORGHI	61 (03D)	10
*JCBORGHS	61 (03D)	02
*JCBORGH1	61 (03D)	04
JCBORGN	61 (03D)	
*JCBORGRI	61 (03D)	44
*JCBORGSH	61 (03D)	05
*JCBORGSS	61 (03D)	01
*JCBPADKY	179 (0B3)	10
JCBPC	66 (042)	
*JCBPCHK	148 (094)	20
JCBPOP	67 (043)	
*JCBPPENQ	148 (094)	10
*JCBPREM	180 (0B4)	80
JCBPRESF	63 (03F)	
JCBPREVF	30 (01E)	
JCBPREVR	31 (01F)	
JCBPRLEN	188 (0BC)	
*JCBQAUQ	40 (028)	04
*JCBQFLP	43 (02B)	40
*JCBQSAD	42 (02A)	20
*JCBQSSA	40 (028)	40
*JCBRAP	148 (094)	40
*JCBRDREQ	60 (03C)	01
JCBRES1	40 (028)	
JCBRES2	44 (2C)	
JCBRES3	48 (30)	
JCBRES4	52 (34)	
JCBRES5	56 (38)	
JCBRES11	40 (28)	
JCBRES12	41 (29)	
JCBRES13	42 (2A)	
JCBRES14	43 (2B)	
*JCBRETD	180 (B4)	04
*JCBRETDL	60 (3C)	20
*JCBRTIST	60 (3C)	02
JCBRWKF	62 (3E)	
JCBSDBND	12 (0C)	
JCBSDB1	8 (08)	
*JCBSGRET	60 (3C)	04
JCBSIZE	36 (24)	
*JCBSKPG	148 (94)	04
JCBSTOR1	68 (44)	
JCBSTOR2	72 (48)	
JCBSTOR3	76 (4C)	
JCBSTOR4	80 (50)	
JCBSTOR5	84 (54)	
JCBSTOR6	88 (58)	
JCBSTOR7	92 (5C)	
JCBSTOR8	96 (60)	
*JCBSWAP	179 (B3)	01
*JCBTAREX	60 (3C)	10
*JCBTARPR	60 (3C)	80
*JCBTCALL	41 (29)	02
JCBTRACE	16 (10)	
*JCBTSKF	43 (2B)	01

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBUQSSA	40 (28)	20
*JCBVL	180 (B4)	08
JCBWKR0	100 (64)	
JCBWKR1	104 (68)	
JCBWKR2	108 (6C)	
JCBWKR3	112 (70)	
JCBWRK4	116(74)	
JCBWKR5	120(78)	
JCBWKR6	124(7C)	
JCBWKR7	128(80)	
JCBWKR8	132(84)	
JCBWKR9	136(88)	
JCBWKR10	140(8C)	
JCBWKR11	144(90)	
JCBWKR12	148(94)	
JCBWKR13	152(98)	
JCBWKR14	156(9C)	
JCBWKR15	160(A0)	
JCBWK12A	148(94)	
JCBWK12B	149(95)	
*JCBXP	180(B4)	10

RECORD LAYOUT - JCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	JCBLEVTB		Address of level table
4(04)	4	JCBLEVND		Address of end of level table + 1
8(08)	4	JCBSDB1		Address of first SDB entry (roots)
12(0C) 1	4	JCBSDBND		Address of end of SDBs +
16(10)	14	JCBTRACE		Prior 7 functions followed by return code

DL/I FUNCTION CODES

The following calls require a PCB and will be traced in JCBTRACE. Any call not requiring a PCB is not put in the trace table. However, the function code appears in JCBPREVF or JCBPREVR.

Name	Code(Hex)	Meaning
FUNCGU	01	'GU' Get Unique
FUNCGHU	01	'GHU' Get Hold Unique
FUNCGN	03	'GN' Get Next
FUNCHHN	03	'GHN' Get Hold Next
FUNCGNP	04	'GNP' Get Next Within Parent
FUNCGHNP	04	'GHNP' Get Hold Next Within Parent
FUNCDRTY	20	Delete/Replace
FUNCREPL	21	'REPL' Replace
FUNCDLET	22	'DLET' Delete
FUNCISTY	40	'ISRT' Insert
FUNCISRT	41	Insert
FUNCASRT	42	DL/I Utility Insert

The following codes must have a PCB

FUNCCHKP	85	'CHKP' checkpoint
FUNCPCBM	90	PCB Call for MPS

The following codes do not require a PCB

FUNCUNLD	A0	'UNLD' Unload Call
FUNCGSCD	A1	'GSCD' Get SCD Call
FUNCTERM	A3	'TERM' Termination Call

DL/I FUNCTION TYPES

FUNCGNTY	80	Get Next Type
FUNCGUTY	40	Get Unique Type
FUNCPATY	20	Parent Type
FUNCHOTY	08	Hold Type

30(1E)	1	JCBPREVF		Prior function

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
31(1F)	1	JCBPREVR		Prior return code (right byte)
32(20)	4	JCBLEV1C		Address of first level table entry in call; Address of lowest level table entry succesfully processed by retrieve
36(24)	2	JCBSIZE		PCB plus JCB size
38(26)	2	JCBMKYL		Maximum length of key feedback area
40(28)	4	JCBRES1		Call characteristics set by call analyzer
40(28)	1	JCBRES11		First flag byte
		JCBNSSA	80	No SSAs
		JCBQSSA	40	Qualified SSAs
		JCBUQSSA	20	Unqualified SSAs
		JCBMSSA	10	Multiple SSAs
		JCBMUSSA	08	Multiple unqualified SSAs
		JCBQAUQ	04	Qualified SSA after an unqualified SSA
		JCBLSSAQ	02	Last SSA qualified
41(29)	1	JCBRES12		Second flag byte
		JCBCCALL	04	Call has C command code
		JCBTCALL	02	Call has T command code
		JCBLV1C	01	JCBLEV1C has been filled on this call
42(2A)	1	JCBRES13		Third flag byte
		JCBALQD	80	Any level qualified on data
		JCBALD	40	Any level had D command code
		JCBQSAD	20	Qualified SSA follows D command code
43(2B)	1	JCBRES14		Fourth flag byte
		JCBFNSL	80	Field is not in sublist
		JCBQFLP	40	Qualification field is in logical parent
		JCBTSKF	01	This set has a key field
44(2C)	4	JCBRES2		Action modules work area
48(30)	4	JCBRES3		Action Modules work area
52(34)	4	JCBRES4		Action Modules work area
56(38)	4	JCBRES5		Action modules work area
60(3C)	1	JCBCODE		Inter-module communications switch
		JCBTARPR	80	DLZPOST update twin pointers only
		JCBDEFDL	40	Re-insert of a deleted segment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		JCBRETDL	20	Return deleted segment for HD unload
		JCBTAREX	10	Reposition for GN (no SSA) with multiple positioning
		JCBMLPOS	08	Retrieve keeping multiple positions
		JCBSGRET	04	Used in positioning after not found
		JCBRTIST	02	Retrieve positioning for insert
		JCBRDREQ	01	DLZSKPG start at next occurence of segment
61(3D)	1	JCBORGN		Open switch and composite organization of all SDBs in the JCB
		JCBOPEN	80	Open done for all data sets in the JCB
		JCBORGRI	44	Organization is root of index
		JCBORGHD	20	Organization is HDAM
		JCBORGHI	10	Organization is HIDAM
		JCBORGSH	05	Organization is simple HISAM
		JCBORGH1	04	Organization is HISAM
		JCBORGHS	02	Organization is HSAM
		JCBORGSS	01	Organization is simple HSAM
62(3E)	1	JCBRWKF		Retrieve's working function
63(3F)	1	JCBPRESF		Present coded function (see DL/I Function Codes)
64(40)	1	JCBLVT		Switches used in accessing segments via DLZSKPG routine
		JCBDOPI	08	Program isolation is to be done for associated PCB
		JCBALLEX	04	All sensitive segments have exclusive intent
		JCBNMFDB	02	Field name not found in FDB
		JCBFLS	01	At least one segment has field level sensitivity (used by call analyzer)
65(41)	1	JCBLVC		Level of segment being searched for by retrieve
66(42)	1	JCBPC		Physical code of segment being searched for by retrieve
67(43)	1	JCBPOP		Parent level for within parent calls

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
68(44)	4	JCBSTOR1		Insert's use across I/O or calls
72(48)	4	JCBSTOR2		Insert's use across I/O or calls
76(4C)	4	JCBSTOR3		Insert's use across I/O or calls
80(50)	4	JCBSTOR4		Address of last segment read - referenced by label BEGBUF in retrieve
84(54)	4	JCBSTOR5		Current segment RBA - referenced by label CURTTR in retrieve
88(58)	4	JCBSTOR6		Retrieve"s use across I/O or calls
92(5C)	4	JCBSTOR7		Contains switches for positive check phase - referenced by label KEEPIT in retrieve
96(60)	4	JCBSTOR8		Work area for retrieve
100(64)	4	JCBWKR0		Action modules work area
104(68)	4	JCBWKR1		Action modules work area
108(6C)	4	JCBWKR2		Action modules work area
112(70)	4	JCBWKR3		Action modules work area
116(74)	4	JCBWKR4		Action modules work area
120(78)	4	JCBWKR5		Action modules work area
124(7C)	4	JCBWKR6		Action modules work area
128(80)	4	JCBWKR7		Action modules work area
132(84)	4	JCBWKR8		Action modules work area
136(88)	4	JCBWKR9		Action modules work area
140(8C)	4	JCBWKR10		Action modules work area
144(90)	4	JCBWKR11		Action modules work area
148(94)	4	JCBWKR12		Action modules work area
148(94)	4	JCBWK12A		Program isolation switches (retrieve only)
		JCBNODEQ	80	No dequeue processing; all level table entries empty after CHKP, TERM, etc.
		JCBRAP	40	Root anchor pointer enqueued (HDAM only)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		JCBPCHK	20	DLZPCHK calling DLZPOST (enqueue not required)
		JCBPPENQ	10	DLZKDTL enqueued on physical parent searching on data field
		JCBNTFD	08	DLZPCHK processing not found condition
		JCBSKPG	04	DLZDEQ should release all outstanding enqueues
		JCBDLET	02	ENQ/DEQ required in DLZPCHK due to delete
		JCBISRT	01	Indicates DLZHIDA or DLZHDAM is accessing destination parent during a logical child insert
149(95)	3	JCBWK12B		Action modules work area
152(98)	4	JCBWKR13		Action modules work area
156(9C)	4	JCBWKR14		Action modules work area
160(A0)	4	JCBWKR15		Action modules work area
Start of each DSG section of JCB				
164(A4)	4	JCBDCBA		Address of the ACB extension for this data set (KSDS ACB extension if HISAM)
168(A8)	2	JCBDMBNO		DMB number for this DSG
170(AA)	1	JCBDCBNO		ACB number of ACB in DMB (KSDS ACB number if HISAM)
171(AB)	1	JCBINDA		JCB Indicators
		JCBDSCLS	80	This last DSG in JCB
		JCBDSORI	44	Data set group is root in index
		JCBDSOHD	20	Data set group is HDAM
		JCBDSOHI	10	Data set group is HIDAM
		JCBDSOH1	04	Data set group is HISAM or simple HISAM
		JCBDSOHS	02	Data set group is HSAM or simple HSAM
		JCBDSOUP	01	Data set group is SHSAM or SHISAM
172(AC)	4	JCBIRECA		
172(AC)	4	JCBHSADD		HSAM I/O area after open
172(AC)	4	JCBTTR		
176(B0)	2			HSAM block size
178(B2)	1	JCBINDB		(Not used in DL/I DOS/VS)
179(B3)	1	JCBINDC		JCB indicators

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		JCBBLDEL	80	This DSG belongs to delete/replace
		JCBHDULD	40	HD unload is running
		JCBCONST	20	Index data set contains constant
		JCBPADKY	10	Search argument not equal to key length
		JCBDUPS	08	Non-unique secondary index keys
		JCBHSWLR	01	HSAM wrong length record
180(B4)	1	JCBINDG		JCB indicators - retrieve variable length flags
		JCBPREM	80	Segment prefix moved to work area
		JCBDATX	40	Segment completely
expanded				
		JCBXP	10	Force complete segment expansion
		JCBVL	08	The variable length routine has been entered for segment
		JCBRETD	04	Data return call
		JCBCOMMD	02	Path return call
181(B5)	3			**Reserved**
184(B8)	4	JCBNOSAM		Retrieve HSAM's ID
188(BC)	4	JCBLROOT		RBA of current root
		JCBPRLEN		Length of JCB prefix
		JCBDSGLN		Length of each DSG section of JCB

LEV - LEVEL TABLE ENTRY

DSECT Name: LEV

The level table entry is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*LEVCDB	13 (0D)	80
*LEVCOMMC	18 (12)	40
*LEVCOMMD	19 (13)	04
*LEVCOMMF	19 (13)	20
*LEVCOMML	19 (13)	10
*LEVCOMMN	19 (13)	02
*LEVCOMMQ	19 (13)	01
*LEVCOMMT	18 (12)	80
*LEVCOMMU	18 (12)	01
*LEVCOMMV	18 (12)	02
*LEVCOMMX	18 (12)	20
*LEVCONT	13 (0D)	08
*LEVDATA	12 (0C)	08
*LEVDATA1	17 (11)	04
*LEVDLET	12 (0C)	80
*LEVEMPTY	12 (0C)	40
LEVEND	36 (24)	
*LEVEOD	13 (0D)	20
LEVFLD	24 (18)	
LEVF1	12 (0C)	
LEVF2	13 (0D)	
LEVF3	17 (11)	
LEVF4	18 (12)	
LEVF5	19 (13)	
*LEVHELD	12 (0C)	20
*LEVHIER	12 (0C)	10
*LEVISRT	17 (11)	80
*LEVKEY1	17 (11)	02
*LEVLAST	12 (0C)	01
LEVLEN	36 (24)	
LEVLEV	0 (00)	
*LEVLSW	13 (0D)	02
*LEVMEMAC	20 (14)	08
LEVMEMBR	20 (14)	
*LEVNDB	13 (0D)	01
*LEVNFPOS	13 (0D)	40
*LEVNOCOV	17 (11)	01
LEVNUPC	16 (10)	
LEVNUSDB	28 (1C)	
LEVPC	1 (01)	
*LEVPFRST	12 (0C)	02
*LEVPLAST	12 (0C)	04
*LEVPSUDO	17 (11)	08
LEVSDB	8 (08)	
LEVSEGOF	2 (02)	
LEVSSA	32 (20)	
*LEVSTOP	13 (0D)	04

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
LEVTTR	4 (04)	
LEVUSEOF	14 (0E)	

RECORD LAYOUT - LEV

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	LEVLEV		Level number
1(01)	1	LEVPC		Current segment physical code

Note: This portion of the level table, once set by retrieve/insert, is never cleared to zeros; it is only changed as needed.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
2(02)	2	LEVSEGOF		Segment's physical code offset from start of record (relative offset to segment from start of buffer)
4(04)	4	LEVTTR		Relative byte address
8(08)	4	LEVSDB		SDB entry address for current segment physical code in this entry
12(0C)	1	LEVF1		Flags
		LEVDLET	80	Segment at this level newly deleted
		LEVEMPTY	40	This level table entry empty
		LEVHELD	20	Segment at this level in hold status
		LEVHIER	10	Segment at this level in hierarchic path (HISAM only)
		LEVDATA	08	Segment at this level moved to user
		LEVPLAST	04	Segment is last of type for parent
		LEVPFRST	02	Segment is first of type for parent
		LEVLAST	01	This is the last level table for PCB
13(0D)	1	LEVF2		Flags
		LEVCDB	80	Verify enques required in data base of current segment
		LEVNFPOS	40	Level has not found position for higher level
		LEVEOD	20	EOD flag
			10	** Reserved **
		LEVCONT	08	The SSA at this level allows retrieve to obtain

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				the next sequential segment
		LEVSTOP	04	Used to determine the setting of LEVCONT by retrieve
		LEVLSW	02	Used by retrieve
		LEVNDB	01	Verify enques required in destination parents data base
14(0E)	2	LEVUSEOF		Offset of segment in user I/O area (PSTUSER)

Note: Fields LEVNUPC through LEVSSA describe the SSA set by the call analyzer for this entry.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
16(10)	1	LEVNUPC		Physical code of requested segment
17(11)	1	LEVF3		Flags
		LEVISRT	80	Inserting at this level (set by retrieve)
		LEVHOLD	10	At least one Boolean expression in range at this level
		LEVPSUDO	08	This is a pseudo SSA filling gap
		LEVDATA1	04	At least one member qualified on data
		LEVKEY1	02	Every Boolean set has at least one key field
		LEVNOCOV	01	No conversion to be done for this segment
18(12)	1	LEVF4		Flags
		LEVCOMMT	80	T command code - retrieve by direct address
		LEVCOMMC	40	C command code - qualifier is concatenated key
		LEVCOMMX	20	X command code - index maintenance internal call
		LEVCOMMV	02	V command code - maintain existing position at all levels
		LEVCOMMU	01	U command code - maintain existing position at this level
19(13)	1	LEVF5		Flags
		LEVCOMMF	20	F command code - get first of segment type
		LEVCOMML	10	L command code - get last of segment type
		LEVCOMMD	04	D command code - transfer data this level
		LEVCOMMN	02	N command code - do not replace this level
		LEVCOMMQ	01	Q command code - enqueue segment at this level read only

Offset Dec (Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
20(14)	1	LEVMEMBR		Switch for each member
			80	** Reserved **
			40	** Reserved **
			20	** Reserved **
			10	** Reserved **
		LEVMEMAC	08	This member in use - (unqualified in only bit)
			04	** Reserved **
			02	** Reserved **
			01	** Reserved **
21(15)	3			** Reserved **
24(18)	4	LEVFLD		Pointer to first field entry (SSA unqualified if zeros)
28(1C)	4	LEVNUSDB		SSAs SDB address
32(20)	4	LEVSSA		SSAs left parenthesis position address
36(24)		LEVEND		End of level table entry
36(24)		LEVLEN		Length of level table entry (LEVEND minus LEVLEV)

MPC - START PARTITION DLZXCB02

PARAMETER LIST MAPPING

DSECT Name: MPCSPART

The MPCSPART maps the start partition XECB parameter list.

RECORD LAYOUT - MPCSPART

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	MPCXECBS		DLZXCB0Z XECB
4(04)	4	MPCSPRO		Address of start partition processing routine in DLZMPC00
8(08)	4	MPCPTABE		Address of next partition table entry to be used for batch partition entry

MPCPT - MPC PARTITION TABLE

The Master Partition Controller (MPC) partition table is used to pass control information when processing batch partition application programs under multiple partition support (MPS). The MPC partition table resides in the transaction work area. There is one entry for every partition that is system generated.

Field Name	Length (bytes)	Description
MPCPARTB	340	Contains one 28 byte entry (see MPC Partition Table entry) for each batch partition allowed to run concurrently. The last entry is delimited by a full-word of X'FF'.
MPCECBLT	4 (per entry)	This is the CICS WAITM ECB list. It contains one entry for each: • DLZXCB00 (Stop Transaction XECB) - used to stop MPS • DLZXCB01 (Stop Partition XECB) - posted by BPC when it stops • DLZXCB02 (Start partition XECB) - defined by MPS. Used by batch initialization to notify MPC to start the BPC • DLZXCBn3 (ABEND XECB) - Used for ABEND handling Note: n is the partition identifier. The last entry is delimited by a fullword of X'FF'.

MPC PARTITION TABLE ENTRY

DSECT Name: MPCPT

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*MPCABWT	0(00)	08
MPCAXECB	12(0C)	
*MPCCNBPC	20(14)	80
MPCDELIM	0(00)	
*MPCERR	0(00)	40
MPCFLAG	0(00)	
MPCFLAG1	20(14)	
*MPCNPTE	28(1C)	12
*MPCPACT	0(00)	80
MPCPID	3(03)	
MPCPIDHX	21(15)	
*MPCPSTP	0(00)	10
MPCPTLN	28(1C)	
MPCRC1	1(01)	
MPCRC2	2(02)	
MPCSXECB	8(08)	
MPCTCA	4(04)	
*MPCTSTP	0(00)	20

RECORD LAYOUT - MPC

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	MPCDELIM		MPCPT delimiter field
0(00)	1	MPCFLAG		MPC activity flags
		MPCPACT	80	Partition active indicator
		MPCERR	40	Error condition encountered on DL/I scheduling call, or BPC attach failure
		MPCTSTP	20	Stop transaction indicator
		MPCPSTP	10	Stop partition indicator
		MPCABWT	08	MPC should wait on the ABEND XECB in the event of a BPC ABEND
1(01)	1	MPCRC1		Error return code from TCAFCTR
2(02)	1	MPCRC2		Error return code from TCADLTR
3(03)	1	MPCPID		XECB identifier generated by DLZMPC00
4(04)	4	MPCTCA		Address of TCA
8(08)	4	MPCSXECB		Address of stop partition XECB (DLZXCB01)
12(0C)	4	MPCAXECB		Address of partition ABEND XECB (DLZXCBn3)
16(10)	4	Unnamed		**Reserved**
20(14)	1	MPCFLAG1		MPC activity flags
		MPCCNBPC	80	Cancel BPC at stop transaction when MPS batch partition is not active.
21(15)	2	MPCPIDHX		Partition identifier
23(17)	1	Unnamed		** Reserved **
24(18)	4	Unnamed		** Reserved **
28(1C)		MPCPTLN		Length of partition table entry
28(1C)		MPCNPTE		Number of partition table entries defined by DLZMPC00

PATH - PATH HEADER CONTROL BLOCK

DSECT Name: PATH

This DSECT describes the fields for DL/I HLPI PATH header control block.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PATHCALL	8(008)	80
PATHFLAG	8(008)	
*PATHLEN	10(00A)	0C
PATHPRCT	4(004)	
PATHSSAP	0(000)	
PATHTOTN	9(009)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	PATHSSAP		Address of PATH SSA appendage
4(004)	4	PATHPRCT		Previous get PATH call DL/I parameter count
8(008)	1	PATHFLAG		Flag byte
		PATHCALL	80	Previous call was a GET PATH call
9(009)	1	PATHTOTN		Number of calls on previous GET PATH call
10(00A)	2			** Reserved **
		PATHLEN		"*-PATHSSAP" length of PATH header control block

PCB - PROGRAM COMMUNICATION BLOCK

DSECT Name: DBPCB

The data management PCB (program communication block) is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DBPCBAE	9 (09)	01
DBPCBDBD	0 (00)	
DBPCBGO	9 (09)	02
DBPCBJCB	16 (10)	
DBPCBKFD	36 (24)	
DBPCBLEV	8 (08)	
DBPCBLE1	8 (08)	
DBPCBLE2	9 (09)	
DBPCBLKY	28 (1C)	
DBPCBMKL	28 (1C)	
DBPCBNSS	32 (20)	
DBPCBPRO	12 (0C)	
DBPCBSFD	20 (14)	
DBPCBSTC	10 (0A)	
*DBPCBTKW	16 (10)	80

RECORD LAYOUT - PCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DBPCBDBD		DBD Name
8(08)	2	DBPCBLEV		Level feedback

The following fields are used for communication from PSBGEN tc ACBGEN only.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
8(08)	1	DBPCBLE1		Level feedback flag byte one
9(09)	1	DBPCBLE2		Level feedback flag byte two
		DBPCBGO	02	GO of GOP PROCOPT
		DBPCBAE	01	Suppress program isolaticn
10(0A)	2	DBPCBSTC		Status codes
12(0C)	4	DBPCBPRO		DL/I processing cptions
16(10)	4	DBPCBJCB		JCB address
		DBPCBTKW	80	Another task waiting for resource owned by this task
20(14)	8	DBPCBSFD		Segment name feedback
28(1C)	4	DBPCBLKY		Maximum length of key feedback area
28(1C)	4	DBPCBMKL		Current length of key feedback area
32(20)	4	DBPCBNSS		Number of sensitive segments in the PCB
36(24)	Var	DBPCBKFD		Key feedback area

PDCA - PROBLEM DETERMINATION CONTROL AREA

DSECT Name: PDCA

The PDCA (Problem Determination Control Area) is used to hold miscellaneous data used in problem determination.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PDCACPAC	0(00)	
PDCADECB	16(10)	
PDCAEND	32(20)	
PDCAFERT	8(08)	
PDCAFLAG	12(0C)	
PDCAMSG	13(0D)	
*PDCASTOP	12(0C)	
PDCAXPRM	4(04)	
PDCBPCAT	24(18)	
PDCBPCNT	20(14)	
PDCSYSTT	28(1C)	

RECORD LAYOUT - PDCA

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	PDCACPAC		Variable length segment compression routine list pointer
4(04)	4	PDCAXPRM		Secondary index suppression routine list pointer
8(08)	4	PDCAFERT		Field exit routine list
12(0C)	1	PDCAFLAG		PDCA flag byte
		PDCASTOP	80	Stop saving messages
13(0D)	3	PDCAMSG		ABEND code
16(10)	4	PDCADECB		Online formatted dump ECB
** MPS TERMINATION CLEANUP ROUTINE ADDRESSES **				
20(14)	4	PDCBPCNT		Address of DLZBPC00 normal terminaticn MPS cleanup routine in DLZMPC00
24(18)	4	PDCBPCAT		Address of DLZBPC00 abnormal termination MPS cleanup routine in DLZMPC00
28(1C)	4	PDCSYSTT		Address of system abnormal termination MPS cleanup routine in DLZMPC00
32(20)		PDCAEND		End of problem determination control area

PDIR - PSB DIRECTORY

DSECT Name: DLZPDIR

The PSB directory contains an entry for every PSB (program specification block) that may run under DL/I control. The PSB directory is part of the DL/I nucleus and is created during DL/I system definition for online processing. The start address of the PSB directory (SCDDLIPS) and the entry length (SCDDLIPL)are contained in the SCD (system contents directory).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PDIRADDR	8 (08)	
*PDIRBAD	19 (13)	01
*PDIRBPLI	19 (13)	08
PDIRCODE	18 (12)	
*PDIRDELT	18 (12)	02
*PDIRDUPL	18 (12)	10
PDIREMOT	24 (18)	
*PDIREXC	18 (12)	40
PDIRLEN	28 (1C)	
*PDIRMPLI	18 (12)	08
*PDIRNOSC	19 (13)	80
*PDIRNTNT	19 (13)	10
PDIROPTC	19 (13)	
*PDIRPLI	18 (12)	20
PDIRPSBL	12 (0C)	
*PDIRREM	19 (13)	20
*PDIRSCHD	19 (13)	40
PDIRSILA	20 (14)	
PDIRSYM	0 (00)	
*PDIRTFAL	18 (12)	01
*PDIRUPD	18 (12)	80
*PDIRXPSB	19 (13)	04
*PDIRZWA	16 (10)	

RECORD LAYOUT - PDIR

Offset Dec (Hex)	Length	Field/Flag Name	Flag Code (Hex)	Meaning
0(00)	0	PDIR		Label used to establish address
0(00)	8	PDIRSYM		PSB execution name - converted from name supplied during PSBGEN
8(08)	4	PDIRADDR		PSB address (contains 0 for remote PSB)
12(0C)	4	PDIRPSBL		Storage required for PSB
16(10)	2	PDIRZWA		Storage required for index workarea
18(12)	1	PDIRCODE		PSB code byte
		PDIRUPD	80	This PSB is update sensitive
		PDIREXC	40	This PSB requires DMB exclusive control
		PDIRPLI	20	This PSB for PL/I
		PDIRDUPL	10	This PSB is duplicate
		PDIRMPLI	08	MPS batch application language is PL/I
		PDIRDELT	02	This PSB is delete sensitive
		PDIRTFAL	01	PSDB-SDB chaining error detected during online task termination
19(13)	1	PDIROPTC		PSB scheduling codes
		PDIRNOSC	80	Do not schedule this PSB
		PDIRSCHD	40	This PSB is scheduled
		PDIRREM	20	This PSB is remote
		PDIRNTNT	10	This PSB is waiting for intent
		PDIRBPLI	08	** Reserved **
		PDIRXPSB	04	Remote PSB with local component
		PDIRBAD	01	PSB initialization failed
20(14)	4	PDIRSILA		Address of PSB segment intent list
24(18)	4	PDIREMOT		Address of RPDIR entry for this remote PSB
28(1C) length		PDIRLEN		PSB directory entry

PPST - PST PREFIX

DSECT Name: DLZPPST

The PST prefix contains data required for user task scheduling in a CICS/VS online environment. It also contains a section used by buffer handler for enqueue/dequeue information and another section used for online segment intent scheduling. The PST prefix is logically part of the PST (partition specification table). However, in order to operate more efficiently in a virtual storage environment, all PST prefixes (one for batch) are organized so that they are physically located in one contiguous area.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PPST	0 (00)	
*PPSTA	4 (04)	01
*PPSTACT	4 (04)	04
*PPSTBF	4 (04)	10
PPSTCA	5 (05)	
PPSTCB	1 (01)	
PPSTCF	0 (00)	
PPSTCHAI	28 (1C)	
PPSTCW	3 (03)	
PPSTECB	2 (02)	
PPSTEND	32 (20)	
PPSTEXCI	12 (0C)	
PPSTID	8 (08)	
PPSTIND	4 (04)	
*PPSTIO	4 (04)	80
PPSTLEN	32 (20)	(See segment intent scheduling section)
PPSTMATR	24 (18)	
*PPSTMPS	4 (04)	08
*PPSTMSDL	4 (04)	02
PPSTPECI	16 (10)	
PPSTPDIR	12 (0C)	(See segment intent scheduling section)
*PPSTSI	4 (04)	40
PPSTSUPO	20 (14)	
*PPSTTC	4 (04)	20
PPSTTCA	9 (09)	
PPSTTSKP	16 (10)	(See segment intent scheduling secion)

RECORD LAYOUT - PPST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	PPST		
0(00)	1	PPSTCF		Prefix chain forward pointer
1(01)	1	PPSTCB		Prefix chain backward pointer
2(02)	1	PPSTECB		POST/WAIT byte of PST ECB
3(03)	1	PPSTCW		PST prefix program isolation wait chain
4(04)	1	PPSTIND		Task schedule and dispatch indicators
		PPSTIO	80	Waiting for I/O
		PPSTSI	40	Cannot schedule due to segment intent conflict
		PPSTTC	20	Cannot schedule - task count limit exceeded
		PPSTBF	10	Task enqueued by buffer handler
		PPSTMPS	08	Indicates MPS task
		PPSTACT	04	This is current task
		PPSTMSDL	02	Scheduled by BPC
		PPSTA	01	Task is scheduled
5(05)	3	PPSTCA		Address of PST
8(08)	1	PPSTID		Task ID
9(09)	3	PPSTTCA		Task TCA address
This section used by buffer handler for enqueue/dequeue				
12(0C)	4	PPSTEXCI		Enqueue/dequeue pointers for existing control interval: Byte 0-1 = buffer number Byte 2-3 = PPST number of task next in chain
16(10)	4	PPSTPECI		Enqueue/dequeue pointers for pending control interval: Byte 0-1 = buffer number Byte 2-3 = PPST number of task next in chain
20(14)	4	PPSTSUPO		Enqueue/dequeue pointers for subpool space: Byte 0-1 = subpool
number				
				Byte 2-3 = PPST number of task next in chain

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
24(18)	4	PPSTMATR		Enqueue/dequeue pointers for interlock detection matrix space: Byte 0-1 = X'00' Byte 2-3 = PPST number of task next in chain
28(1C)	4	PPSTCHAI		Enqueue/dequeue pending control interval chain field pointers: Byte 0-1 = buffer number Byte 2-3 = PPST number of task next in chain
32(20)		PPSTEND		End of prefix DSECT

This section used for online segment intent scheduling

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
12(0C)	4	PPSTPDIR		Task PDIR entry address
16(10)	1	PPSTTSKP		Task dispatching priority
32(20)		PPSTLEN		Length of PST prefix

PSB - PSB Prefix

DSECT Name: PSB

The PSB prefix is described as part of the general structure and description of the program specification block (PSB)

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSB	0 (00)	
PSBCODE	29 (1D)	
PSBDBOFF	34 (22)	
*PSBFLS	29 (1D)	01
PSBFRTA	0 (00)	
PSBINDEX	28 (1C)	
PSBIOASZ	1 (01)	
PSBIOAWK	18 (24)	
PSBLIST	36 (24)	
*PSBLOGDB	29 (1D)	02
PSBNDXWK	20 (14)	
PSBNPLI	29 (1D)	20
*PSBPLI	29 (1D)	10
PSBPST	12 (0C)	
PSBSEGWK	8 (08)	
PSBSIZE	30 (1E)	
PSBTPOFF	32 (20)	
PSBVMID	0 (00)	
*PSBV11	0 (00)	01
PSBXIOWK	4 (04)	
PSBXPCB	16 (10)	

RECORD LAYOUT - PSB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	PSB		
0(00)	1	PSBVMID		DOS DL/I version ID
		PSBV11	01	Version 1.1 or later
0(00)	4	PSBFRTA		Field exit routine address. If no entries in table, low order 3 bytes = 0 (used cnly during initialization)
1(01)	3	PSBIOASZ		Size of the PSB I/O work area whose address is in PSBIOAWK. This field contains a 16-bit logical number.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
4(04)	4	PSBXIOWK		Address of index I/O work area or user's version of a segment built by retrieve
8(08)	4	PSBSEGWK		Address of variable length segment work area
12(0C)	4	PSBPST		PST address if PSB is scheduled or active
16(10)	4	PSBXPCB		Address of index PCB
20(14)	4	PSBNDXWK		Address of index maintenance work area or pointer to the field exit parameter list
24(18)	4	PSBIOAWK		Address of I/O work area
28(1C)	1	PSBINDEX		(Not used in DL/I DOS/VS)
29(1D)	1	PSBCODE		PSB flags
		PSBNPLI	20	PSB is for non-PL/I language
		PSBPLI	10	PL/I is source language
		PSBFLS	01	PSB contains field sensitive segment
		PSBLOGDB	02	PSB retrieves a logical data base
30(1E)	2	PSBSIZE		PSB size
32(20)	2	PSBTPOFF		(Not used in DL/I DOS/VS)
34(22)	2	PSBDBOFF		Offset from the PSBLIST to first DB PCB
36(24)	Var	PSBLIST		Beginning of PCB list. Note: this field is a list of fullword pointers containing PCB addresses. Last PCB address word has byte 0, bit 0 = 1. List may contain a maximum of 64 addresses. For PL/I programs these pointers are to the dope Vector Tables in which the first word is a pointer to the associated PCB.

PSDB - PHYSICAL SEGMENT DESCRIPTION BLOCK

DSECT Name: DMBPSDB

The PSDB is described as part of the general structure and description of the data management block (DMB)

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBCKL	14 (0E)	
*DMBCPT	24 (18)	04
*DMBCPTIT	24 (18)	01
*DMBCPTKY	24 (18)	02
*DMCCTR	7 (07)	80
DMBDCB	6 (06)	
DMBDL	10 (0A)	
DMBDLT	13 (0D)	
*DMBDRL	13 (0D)	03
*DMBDRP	13 (0D)	02
*DMBDRV	13 (0D)	01
*DMBEX	16 (10)	80
DMBFDBA	16 (10)	
DMBFLAG	32 (20)	
DMBFSDB	20 (14)	
*DMBIFST	12 (0C)	10
*DMBIHERE	12 (0C)	30
*DMBILST	12 (0C)	20
*DMBIRL	12 (0C)	03
*DMBIRP	12 (0C)	02
*DMBIRV	12 (0C)	01
DMBISRT	12 (0C)	
*DMBLCEX	32 (20)	20
DMBLEV	2 (02)	
*DMBLP	7 (07)	02
*DMBLPEX	32 (20)	40
DMBLST	32 (20)	
*DMBLTBK	7 (07)	04
*DMBLTFD	7 (07)	08
*DMBNXEX	32 (20)	10
*DMBPI	24 (18)	80
DMBPLEM	36 (24)	
*DMBPP	7 (07)	10
DMBPPBK	5 (05)	
DMBPPFD	4 (04)	
DMBPRSZ	8 (08)	
DMBPSC	1 (01)	
DMBPSDBN	36 (24)	
*DMBPTBK	7 (07)	20
*DMBPTFD	7 (07)	40
DMBPTR	7 (07)	
*DMBRRL	13 (0D)	0C
*DMBRRP	13 (0D)	08
*DMBRRV	13 (0D)	04
DMBSC	0 (00)	
DMBSCTAB	25 (19)	
DMBSGMN	28 (1C)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBSGMX	30 (1E)	
*DMBUP	16 (10)	40
DMBUSE	16 (10)	
DMBVLDFG	24 (18)	
*DMBVIS	24 (18)	04
*DMBXDES	32 (20)	04
DMBXNULL	3 (03)	
*DMBXPROT	12 (0C)	80

RECORD LAYOUT - PSDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	DMBSC		Segment code
			01	Root segment code
1(01)	1	DMBPSC		Parent's segment code
2(02)	1	DMBLEV		Segment level
3(03)	1	DMBXNULL		(Not used in DL/I DOS/VS)
4(04)	1	DMBPPFD		Pointer number in parent to first occurrence of segment for parent
5(05)	1	DMBPPBK		Pointer number in parent to last occurrence of segment for parent
6(06)	1	DMBDCB		ACB number
7(07)	1	DMBPTR		Prefix flags
		DMBCTR	80	Counter present
		DMBPTFD	40	Segment has physical twin forward pointer
		DMBPTBK	20	Segment has physical twin backward pointer
		DMBPP	10	Segment has physical parent pointer
		DMBLTFD	08	Segment has logical twin forward pointer
		DMBLTBK	04	Segment has logical twin backward pointer
		DMBLP	02	Segment has logical parent pointer
8(08)	2	DMBPRSZ		Prefix length of segment
10(0A)	2	DMBDL		Data length of segment
12(0C)	1	DMBISRT		Insert rules
		DMBXPROT	80	System data in index is protected
		DMBIHERE	30	If no key field, insert at current position
		DMBILST	20	If no key field, insert after existing segment
		DMBIFST	10	If no key field, insert before existing segment
		DMBIRL	03	Insert rule is logical
		DMBIRP	02	Insert rule is physical
		DMBIRV	01	Insert rule is virtual
13(0D)	1	DMBDLT		Delete/replace rules
		DMBRRL	0C	Replace rule is logical
		DMBRRP	08	Replace rule is physical
		DMBRRV	04	Replace rule is virtual
		DMBDRL	03	Delete rule is logical
		DMBDRP	02	Delete rule is physical
		DMBDRV	01	Delete rule is virtual

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
14(0E)	2	DMBCKL		Concatenated key length of parent of this segment
16(10)	1	DMBUSE		Code Byte
		DMBEX	80	This PSDB in use exclusively
		DMBUP	40	This PSDB in use for update. Bits 2-7 contain a count of read-only users
16(10)	4	DMBFDBA		Address of FDBs for this segment
20(14)	4	DMBFSDB		Address of first SDB for this segment
24(18)	1	DMBVLDFG		Variable length data flag
		DMBPI	80	Program isolation should be done for this segment
		DMBCPT	08	Segment has compression routine
		DMBVLS	04	Segment is variable
length				
		DMBCPTKY	02	Compression routine has key expand routine
		DMBCPTIT	01	Compression routine has intialization processing
25(19)	3	DMBSCTAB		Address of segment compaction table
28(1C)	2	DMBSGMN		If variable length segment; minimum length of segment
30(1E)	2	DMBSGMX		If variable length segment; maximum length of segment
32(20)	1	DMBFLAG		Secondary list flag
		DMBLPEX	40	A logical parent exists (segment is a logical child)
		DMBLCEX	20	One or more logical children exists (segment is a logical parent)
		DMBNXEX	10	One or more indexes exist
		DMBXDEX	04	An indexed segment exists
32(20)	4	DMBLST		Address of secondary list for this segment
36(24)		DMBPSDBN		End of one PSDB entry
36(24)		DMBPLEN		Length of each PSDB entry (DMBPSDBN minus DMBSC)

PSIL - PSB SEGMENT INTENT LIST

DSECT Name: DLZPSIL

The PSB segment intent list is pointed to from the PSB directory and is a list of all the DMBs which may be used by that PSB (program).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PSILBFRI	8 (08)	20
*PSILDBEX	8 (08)	80
*PSILDBUP	8 (08)	40
PSILDIRA	0 (00)	
PSILDIRN	4 (04)	
PSILDMBN	0 (00)	
PSILGOPO	8 (08)	10
PSILLNGH	9 (09)	
PSILNOPI	8 (08)	04
PSILNREF	8 (08)	10
PSILNTNT	8 (08)	
PSILSEGD	10 (0A)	

RECORD LAYOUT - PSIL

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	PSILDMBN		DMB name for this list entry - overlaid during initialization
0(00)	4	PSILDIRA		Address of DMB directory entry - resolved during initialization
4(04)	4	PSILDIRN		DMB number of this DMB
8(08)	1	PSILNTNT		Segment intent descriptor byte
		PSILDBEX	80	PSB contains a PCB which requires exclusive control for this DMB
		PSILDBUP	40	PSB contains a PCB which is update sensitive
		PSILBFRI	20	Buffer pool space required for this KSDS
		PSILGOPO	10	PSB references are all 'GO'
		PSILNREF	08	Data base is not referenced
		PSILNOPI	04	No translate for PI
9(09)	1	PSILLNGH		Length of this entry in list
10(0A)	Var	PSILSEGD		Segment intent bits.

Each segment in the DMB pointed to by an intent list entry is described by 2-bit fields begining at PSILSEGD. There is a list entry for each DMB referenced in the associated PSB. The two bits represent the PSB's sensitivity to each PSDB and have the following meanings:

Bit	Meaning
00	PSB not sensitive to the segment
01	PSB read-only sensitive
10	PSB is update sensitive
11	PSB requires exclusive control (HISAM root insert)

1111 1111	PSILNS	"X'FF'" mask used to test four segments for no sensitivity
1.1. 1.1.	PSILRO	"X'AA'" mask used to test four segments for update

The bits are allocated to segments in the following manner:

	BYTE 1								BYTE 2							
BIT	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SEGMENT	4		3		2		1		8		7		6		5	

12(0C)	4	PSILEND	End of PSB intent list indicator

PST - PARTITION SPECIFICATION TABLE

DSECT Name: DLZPST

One partition specification table (PST) exists for each task in an online or batch processing partition. All DL/I resources allocated to the task can be located through the PST. The PST also contains pointers to the task I/O area and any segments currently associated with the task.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*DBLCMC	440 (1B8)	00
*DBLFSE1	440 (1B8)	00
*DBLFSE2	440 (1B8)	04
*DBLLASTC	440 (1B8)	08
*DBLLGDLT	440 (1B8)	60
*DBLNDXC	440 (1B8)	80
*DBLNEWBL	440 (1B8)	01
*DBLNTCR	440 (1B8)	70
*DBLOOPS	440 (1B8)	0A
*DBLPHYD	440 (1B8)	20
*DBLPHYL	440 (1B8)	40
*DBLPHYR	440 (1B8)	10
*DBLPHYRO	440 (1B8)	02
PSTABIND	72 (048)	
PSTACBNM	150 (096)	
PSTACCT	92 (05C)	
*PSTBATCH	468 (1D4)	80
*PSTBDCAL	137 (089)	10
*PSTBFALT	136 (088)	05
*PSTBFMPT	136 (088)	04
PSTBFUSE	164 (0A4)	
*PSTBKLCT	136 (088)	01
PSTBLKNM	144 (090)	
*PSTBTMPF	136 (088)	03
PSTBUFFA	160 (0A0)	
*PSTBYALT	136 (088)	06
*PSTBYEND	137 (089)	28
*PSTBYLCT	136 (088)	02
PSTBYTNM	152 (098)	
*PSTCANLI	487 (1E7)	40
*PSTCHKP	469 (1D5)	04
*PSTCIOAF	178 (0B2)	04
*PSTCLOK	137 (089)	00
PSTCLRWT	258 (102)	
PSTCNVB	479 (1Df)	
PSTCODE1	68 (044)	
PSTCPLN	184 (0B8)	
PSTCTGFL	224 (0E0)	
PSTCTGL1	248 (0F8)	
PSTCTGL2	251 (0FB)	
PSTCTGNM	184 (0B8)	
PSTCTGPL	184 (0B8)	
PSTCTGRT	252 (0FC)	
PSTCTGWK	248 (0F8)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSTCURWA	344 (158)	
PSTCWKLN	252 (0FC)	
PSTDATA	156 (09C)	
PSTDBPCB	132 (084)	
PSTDCHKP	128 (080)	
PSTDDLET	120 (078)	
PSTDELTA	616 (268)	
PSTDGHN	108 (06C)	
PSTDGHNP	112 (070)	
PSTDGHU	104 (068)	
PSTDGN	96 (060)	
PSTDGNP	100 (064)	
PSTDGU	92 (05C)	
PSTDISRT	116 (074)	
PSTDLIWA	44 (02C)	
PSTDLIWB	48 (030)	
PSTDLIWC	52 (034)	
PSTDLIWD	56 (038)	
PSTDLIWE	60 (03C)	
PSTDLIWF	64 (040)	
PSTDLIW0	4 (004)	
PSTDLIW1	8 (008)	
PSTDLIW2	12 (00C)	
PSTDLIW3	16 (010)	
PSTDLIW4	20 (014)	
PSTDLIW5	24 (018)	
PSTDLIW6	28 (01C)	
PSTDLIW7	32 (020)	
PSTDLIW8	36 (024)	
PSTDLIW9	40 (028)	
PSTDLROM	352 (160)	
PSTDLTWA	348 (15C)	
PSTDMBNM	148 (094)	
PSTDREPL	124 (07C)	
PSTDSGA	140 (08C)	
*PSTDUMPI	487 (1E7)	80
*PSTENDDA	137 (089)	24
*PSTEOD	137 (089)	2C
*PSTERASE	136 (088)	0A
PSTERCD1	470 (1D6)	
PSTERCD2	471 (1D7)	
PSTERCOD	470 (1D6)	
PSTERDT1	472 (1D8)	
PSTERDT2	479 (1DF)	
PSTERIND	487 (1E7)	
*PSTERMSP	72 (048)	80
*PSTEXPAD	258 (102)	40
*PSTFBSPC	136 (088)	04
PSTFLD	596 (254)	
PSTFLDAL	604 (25C)	
PSTFLDC	608 (260)	
PSTFLDE	604 (25C)	
PSTFLDN	600 (258)	
PSTFNCTN	136 (088)	
*PSTFRBLK	137 (089)	30
*PSTFRSPC	136 (088)	02
*PSTGBSPC	136 (088)	03
*PSTGETNX	136 (088)	0B
*PSTGTDS	136 (088)	04
*PSTGTRAP	136 (088)	04
*PSTGTSPC	136 (088)	01
PSTGVPL	236 (0EC)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSTGVWKL	236 (0EC)	
*PSTHDWIP	178 (0B0)	08
*PSTHISES	178 (0B0)	80
*PSTHISMR	470 (1D6)	10
*PSTHLPI	469 (1D5)	02
*PSTINLD	137 (089)	34
*PSTINTNT	68 (044)	40
*PSTIOERR	137 (089)	08
PSTIQPRM	72 (048)	
*PSTIWAIT	258 (102)	80
PSTLIPRM	488 (1E8)	
PSTLNGTH	1144 (478)	
*PSTLODU	468 (1D4)	40
*PSTLODUH	468 (1D4)	20
*PSTLOGIP	068 (044)	02
PSTLOGQ	440 (1B8)	
PSTLOGWA	436 (1B4)	
PSTMI	76 (04C)	
PSTMROCO	181 (0B5)	
*PSTMSPUT	136 (088)	0E
*PSTNOERR	180 (0B4)	40
PSTNORO	568 (238)	
*PSTNOSPC	137 (089)	0C
*PSTNOTFD	137 (089)	14
*PSTNPLSP	137 (089)	1C
PSTNUMRO	256 (100)	
PSTNUMWT	257 (101)	
*PSTNWBLK	137 (089)	18
*PSTOCALL	136 (088)	04
*PSTOCBAD	136 (088)	80
*PSTOCCLS	136 (088)	00
*PSTOCDCB	136 (088)	10
*PSTOCDMB	136 (088)	01
*PSTOCDSG	136 (088)	40
*PSTOCLD	136 (088)	20
*PSTOCOPN	136 (088)	08
*PSTOCPCB	136 (088)	02
PSTOFFST	138 (08A)	
*PSTOLTW	68 (044)	04
PSTOPEN	620 (267)	
*PSTOPEN1	620 (26C)	40
*PSTPERQC	178 (0BC)	FF
PSTPCPGM	452 (1C4)	
PSTPCPSB	460 (1CC)	
PSTPCT1	468 (1D4)	
PSTPCT2	469 (1D5)	
*PSTPGUSR	136 (088)	07
*PSTPIPIU	137 (089)	80
*PSTPISIU	137 (089)	40
*PSTPLI	469 (1D5)	01
PSTPLIPR	560 (230)	
PSTPOSEL	180 (0B4)	
PSTPREAD	00 (00)	
PSTPREAR	172 (0AC)	
PSTPRTGT	612 (264)	
*PSTPRVWT	68 (044)	08
PSTPSB	88 (058)	
*PSTPUTKY	136 (088)	0D
*PSTQDEQ	136 (088)	08
*PSTQENQ	136 (088)	08
PSTQLEV	572 (23C)	
*PSTQLEXC	572 (23C)	08

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PSTQLRO	572 (23C)	00
*PSTQLUPD	572 (23C)	04
*PSTQPUR	136 (088)	0C
*PSTQRBDC	137 (089)	08
*PSTQRDDL	137 (089)	04
*PSTQRNSE	137 (089)	10
*PSTQROOP	137 (089)	02
*PSTQRWR	137 (089)	01
*PSTQVER	136 (088)	04
PSTRAEND	444 (1BC)	
PSTREAL	206 (0CE)	
*PSTRDERR	137 (089)	08
PSTRETRE	224 (0E0)	
PSTRPSTA	580 (244)	
PSTRRDF	572 (23C)	
PSTRRDL	576 (240)	
PSTRTCDE	137 (089)	
*PSTSABND	72 (048)	20
PSTSAVRE	184 (0B8)	
PSTSAVTR	584 (248)	
*PSTSCALL	68 (044)	80
PSTSCDAD	68 (044)	
*PSTSCHED	68 (044)	10
PSTSDATA	206 (0CE)	
PSTSEG	84 (054)	
PSTSEGL	80 (050)	
*PSTSMRQ	178 (0B2)	02
PSTSPL	212 (0D4)	
*PSTSTLBG	136 (088)	0C
*PSTSTLEQ	136 (088)	09
PSTSUBNM	176 (0B0)	
PSTSUIN	168 (0A8)	
PSTSV1	640 (280)	
PSTSV2	712 (2C8)	
PSTSV3	784 (310)	
PSTSV4	856 (358)	
PSTSV5	928 (3A0)	
PSTSV6	1000 (3E8)	
PSTSV7	1072 (430)	
PSTSWI	178 (0B2)	
PSTSWKAR	184 (0B8)	
PSTSWKL	206 (0CE)	
*PSTTABND	72 (048)	10
PSTTSKID	260 (104)	
*PSTUDR	468 (1D4)	04
PSTUIB	276 (114)	
*PSTULU	468 (1D4)	02
PSTUSER	76 (04C)	
*PSTUSM	468 (1D4)	01
*PSTUST	468 (1D4)	08
PSTVLSR	250 (0FA)	
PSTVSL	206 (0CE)	
*PSTWABUF	144 (090)	80
*PSTWRITE	136 (088)	08
PSTWRKD1	316 (13C)	
PSTWRKD2	320 (140)	
PSTWRKD3	324 (144)	
PSTWRKD4	328 (148)	
PSTWRKD5	332 (14C)	
PSTWRKD6	336 (150)	
PSTWRKD7	340 (154)	
PSTWRKT1	296 (128)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSTWRKT2	300 (12C)	
PSTWRKT3	304 (130)	
PSTWRKT4	308 (134)	
PSTWRKT5	312 (138)	
PSTWRK1	280 (118)	
PSTWRK2	284 (11C)	
PSTWRK3	288 (120)	
PSTWRK4	292 (124)	
*PSTWROSI	137 (089)	20
*PSTXCONM	469 (1D5)	80
*PSTXMDLT	136 (088)	A0
*PSTXMISR	136 (088)	A2
*PSTXMRPL	136 (088)	A1
*PSTXMUNL	136 (088)	A3
*PSTXPRTM	469 (1D5)	40
PSTXPSV1	264 (108)	
PSTXPSV2	268 (10C)	
PSTXPSV3	272 (110)	

RECORD LAYOUT - PST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000) prefix	4	PSTPREAD		Address of this PST
4(004)	4	PSTDLIW0		Action modules work area HD unload (DLZURGU0) return address for retrieve
(008)	4	PSTDLIW1		Action modules work area
12(00C)	4	PSTDLIW2		Action modules work area
16(010)	4	PSTDLIW3		Action modules work area
20(014)	4	PSTDLIW4		Action modules work area
24(018)	4	PSTDLIW5		Action modules work area
28(01C)	4	PSTDLIW6		Action modules work area
32(020)	4	PSTDLIW7		Action modules work area
36(024)	4	PSTDLIW8		Action modules work area
40(028)	4	PSTDLIW9		Action modules work area
44(02C)	4	PSTDLIWA		Action modules work area
48(030)	4	PSTDLIWB		Action modules work area
52(034)	4	PSTDLIWC		Action modules work area
56(038)	4	PSTDLIWD		Action modules work area
60(03C)	4	PSTDLIWE		Action modules work area
64(040)	4	PSTDLIWF		Action modules work area
USER CALL PROCESSING SECTION				
68(044)	1	PSTCODE1		
		PSTSCALL	80	PST for system call
		PSTINTNT	40	Cannot schedule, intent not satisfied
		PSTSCHED	10	OK to complete scheduling
		PSTPRVWT	08	Logger private wait indicator
		PSTOLTW	04	Another task waiting for resource owned by this task. Note: If PSTINTNT and PSTSCHED are both set, DL/I backout is in control.
		PSTLOGIP	02	Log I/O in progress
68(044)	4	PSTSCDAD		Address of SCD
72(048)	4	PSTABIND		Task/system ABEND indicator
		PSTERMSP	80	PUT error message indicator

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTSABND	20	System ABEND indicator
bit				
		PSTTABND	10	Task ABEND indicator bit
72(048)	4	PSTIQPRM		Address of caller's parameter list
76(04C)	4	PSTMI		Return segment indicator
76(04C) area	4	PSTUSER		Address of user's I/O
80(050)	4	PSTSEGL		Retrieved segment length
84(054)	4	PSTSEG		Retrieved segment address
88(058)	4	PSTPSB		PDIR entry address
USER TASK STATISTICS				
92(05C)	4	PSTACCT		
92(05C)	4	PSTDGU		Number of GU calls issued
96(060)	4	PSTDGN		Number of GN calls issued
100(064)	4	PSTDGNP		Number of GNP calls issued
104(068)	4	PSTDGHU		Number of GHU calls issued
108(06C)	4	PSTDGHN		Number of GHN calls issued
112(070)	4	PSTDGHNP		Number of GHNP calls issued
116(074)	4	PSTDISRT		Number of ISRT calls issued
120(078)	4	PSTDDLET		Number of DLET calls issued
124(07C)	4	PSTDREPL		Number of REPL calls issued
128(080)	4	PSTDCHKP		Number of CHKP calls issued
ACTION MODULES SECTION				
132(084)	4	PSTDBPCB		Address of current PCB
136(088)	1	PSTFNCTN		Function codes
EQUATES FOR BUFFER HANDLER FUNCTION CODES				
		PSTBKLCT	01	Locate relative block number
		PSTBYLCT	02	If HD, locate relative byte number. If HISAM or

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				HIDAM INDEX, read a record by RBA from a KSDS. If HISAM, read a record by RBA from an ESDS.
		PSTGBSPC	03	Get buffer space
		PSTFBSPC	04	Free buffer space
		PSTBFMPT	04	Mark buffers enpty
		PSTBFALT	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
		PSTBYALT	06	Locate a relative byte number and mark buffer altered
		PSTPGUSR	07	Purge all buffers altered by a task
		PSTWRITE	08	Write a new record to HISAM ESDs
		PSTSTLEQ	09	Read a record by key from a KSDS
		PSTERASE	0A	Erase a record in a KSDS
		PSTGETNX	0B	Read the next record in a KSDS
		PSTSTLBG	0C	Read the record containing the first root in a KSDS
		PSTPUTKY	0D	Insert a record by key into a KSDS
		PSTMSPUT	0E	Insert record(s) sequentially into a KSDS

EQUATES FOR OPEN/CLOSE FUNCTION CODES

		PSTOCDMB	01	Close DMB. Address of DMB in R2
		PSTOCPCB	02	Close PCB. Address of PCB in R2
		PSTOCALL	04	Close all DMBs
		PSTOCCLS	00	Close call. Bit 4 = 0
		PSTOCOPN	08	Open call. Bit 4 = 1
		PSTOCDCB	10	Open/close the DMB in PSTDCBNM. DSG address in PSTDSGA
		PSTOCLD	20	Open for load
		PSTOCDSG	40	Open the DSG in PSTDSGA
		PSTOCBAD	80	Open unsuccessful

EQUATES FOR SPACE MANAGEMENT FUNCTION CODES

			80	Backout in control
		PSTGTSPC	01	Get space for segment. R5 contains pointer to PSDB
		PSTFRSPC	02	Free space for segment. R5 contains pointer to PSDB
		PSTBTMPF	03	Do bit map update

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTGTRAP	04	Get space close to RAP in PSTBYTNM

EQUATES FOR INDEX MAINTENANCE FUNCTION CODES

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTXMDLT	A0	Perform index maintenance for segment to be deleted
		PSTXMRPL	A1	Perform index maintenance for segment to be replaced
		PSTXMISR	A2	Perform index maintenance for segment to be inserted
		PSTXMUNL	A3	Perform index maintenance for segment to be unloaded

EQUATES FOR PROGRAM ISOLATION FUNCTION CODES

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTQENQ	00	Enqueue (Queueing facility)
		PSTQVER	04	Verify (Queueing facility)
		PSTQDEQ	08	Dequeue (Queueing facility)
		PSTQPUR	0C	Purge (Queueing facility)
137(089)	1	PSTRTCDE		Return codes

EQUATES FOR BUFFER HANDLER RETURN CODES

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTCLOK	00	No error occurred
		PSTGTDS	04	RBN is beyond the end of the data set
		PSTIOERR	08	I/O error
		PSTRDERR	08	Permanent read error
		PSTNOSPC	0C	No space for adds
		PSTBDCAL	10	Illegal call
		PSTNOTFD	14	No record found (retrieve by key)
		PSTNWBLK	18	New block was created in the buffer pool
		PSTNPLSP	1C	Insufficient space in the buffer pool
		PSTWROSI	20	Size of requested buffer exceeds the size of buffers in any subpool
		PSTENDDA	24	End of data set. No record returned
		PSTBYEND	28	Key or RBA higher than the highest key cr RBA in the data set
		PSTEOD	2C	End of data set reached on a request issued by open
		PSTINLD	34	Invalid request during data set loading

SPACE MANAGEMENT RETURN CODES

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTFRBLK	30	Block not used due to distributed free space parameter
		PSTBTMPF	03	Bit map update required
EQUATES FOR PROGRAM ISOLATION RETURN CODES				
		PSTQRWR	01	Wait was required
		PSTQROOP	02	Other owners present
		PSTQRDDL	04	Terminated due to deadlock
		PSTQRBDC	08	Terminated due to bad call
		PSTQRNSE	10	Terminated. Insufficient storage
		PSTPISIU	40	Secondary index updated
		PSTPIPIU	80	Primary index updated
138(08A)	2	PSTOFFST		Offset to PSTDATA from start of buffer
140(08C)	4	PSTDSGA		Address of DSG portion of the JCB
144(090)	4	PSTBLKNM		Relative block number
		PSTWABUF	80	Buffer is being used as a workarea and is not associated with an RBA
148(094)	2	PSTDMBNM		DMB number
150(096)	1	PSTACBNM		ACB number
151(097)	1			**Reserved**
152(098)	4	PSTBYTNM		RBA or relative record number. High order byte contains X'80' if request is for HISAM ESDS
156(09C)	4	PSTDATA		Address of requested data
160(0A0)	4	PSTBUFFA		Address of buffer prefix
BUFFER HANDLER AND SPACE MANAGEMENT SECTION				
164(0A4)	4	PSTBFUSE		Address of the buffer prefix to be used
168(0A8)	4	PSTSUIN		Address of the subpool information table to be used
172(0AC)	4	PSTPREAR		Beginning address of the buffer prefix area for the subpool information table used
176(0B0)	2	PSTSUBNM		Subpool number used during this call
178(0B2)	2	PSTSWI		Work space

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTHDWIP	08	HD write in progress
		PSTCIOAF	04	CI in overflow area full
		PSTHISES	80	HISAM ESDS is being processed
		PSTSMRQ	02	Request made to the buffer handler by space management
		PSTPBRQC	FF	Purge buffer request completed
180(0B4)	1	PSTPOSEL		Count for position of use chain element
		PSTNOERR	40	No error message
181(0B5)	1	PSTMROCO		Number of the row/column in the interlock detection matrix currently used by this task
182(0B6)	2			**Reserved**
184(0B8)	40	PSTSAVRE		Work area used by buffer handler when processing a request

THIS AREA IS USED BY DLZDCI00 FOR SHOWCAT AND GETVCE FOR FBA SUPPORT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
184(0B8)	40	PSTSWKAR		SHOWCAT work area used by Space Management DLZGGSP0 and Open/Close DLZDLOC0
206(0CE)		PSTSDATA		Location of needed data returned by SHOWCAT
		PSTRBAL		RBA data length (equated to 4)
		PSTVSL		Volume serial number length (equated to 6)
		PSTSWKL		Length of SHOWCAT work area (equated to 64)
250(0FA)		PSTVLSR		Volume serial number save area
212(0D4)		PSTSPL		SHOWCAT parameter list
236(0EC)		PSTGVPL		GETVCE parameter list
		PSTGVWKL		Length of GETVCE work area (equated to 52)

THE FOLLOWING FIELDS ARE USED BY DL/I OPEN/CLOSE (DLZDLOC0) AND SPACE MANAGEMENT (DLZDHDS0) FOR VSAM CATALOG PARAMETER LIST WHEN PROCESSING AN OUT-OF-SPACE CONDITION FOR HIDAM DATA BASE

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
184(0B8)	40	PSTCTGPL		Area used as the VSAM catalog parameter list (CTGPL) by DLZGGSP0 and DLZDLOC0 to do locate
		PSTCPLN		Length of CTGPL block (equated to 40)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTCTGNM		Number of CTGFL entries (equated to 1)
224(0E0)	32	PSTRETRE		Buffer handler subroutine linkage register (R14) save area when procssing a request

THE FOLLOWING FIELDS ARE USED BY OPEN/CLOSE (DLZDLOC0) AND SPACE MANAGEMENT (DLZDHDS0) FOR VSAM FIELD PARAMETER LIST WHEN PROCESSING AN OUT-OF-SPACE CONDITION FOR HIDAM DATA BASE

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
224(0E0)	24	PSTCTGFL		Area used as the VSAM field parameter list (CTGFL) by DLZGGSP0 and DLZDLOC0 to do locate
248(0F8)	8	PSTCTGWK		VSAM catalog work area
248(0F8)	3	PSTCTGL1		Catalog work area length 1
251(0FB)	1	PSTCTGL2		Catalog work area length 2
252(0FC)	4	PSTCTGRT		VSAM catalog return area for HI-RBA
		PSTCWKLN		Length of catalog work area (equated to 8)

BUFFER HANDLER STATISTICS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
256(100)	1	PSTNUMRO		Number of blocks read on this call
257(101)	1	PSTNUMWT		Number of writes issued on this call
258(102)	1	PSTCLRWT		Buffer handler switch
		PSTIWAIT	80	IWAIT issued during this call
259(103)	1			**Reserved**
260(104)	4	PSTTSKID		Hashed task ID. High-order byte, binary date. Low-order three bytes, assigned in ascending sequence

THE FOLLOWING FIELDS ARE USED AS SAVE AREAS SO THAT THE DMB ECB CAN BE POSTED IF THE TASK IS CANCELED WHILE WAITING FOR I/O COMPLETION

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
264(108)	4	PSTXPSV1		User VSAM save area address
268(10C)	4	PSTXPSV2		EXCPAD return address
272(110)	4	PSTXPSV3		EXCPAD parameter list address
276(114)	4	PSTUIB		Address of UIB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
PST WORK AREAS				
280(118)	4	PSTWRK1		PSTWRKn are work words for
284(11C)	4	PSTWRK2		buffer handler (DLZDBH00)
288(120)	4	PSTWRK3		and data base logger.
292(124)	4	PSTWRK4		
296(128)	4	PSTWRKT1		PSTWRKn is work space
300(12C)	4	PSTWRKT2		preserved across calls
304(130)	4	PSTWRKT3		to the buffer handler.
308(134)	4	PSTWRKT4		
312(138)	4	PSTWRKT5		
THE HIGH-ORDER BYTE OF PSTWRKT4 IS USED TO PASS THE FOLLOWING FUNCTION CODES TO INDEX MAINTENANCE				
			04	Reinsert index
			03	Secondary indexes only
			02	Primary indexes only
			01	Both primary and secondary indexes
316(13C)	4	PSTWRKD1		PSTWRKDn is work space for
320(140)	4	PSTWRKD2		use by DELETE/REPLACE,
324(144)	4	PSTWRKD3		FLD Storage Manager,
328(148)	4	PSTWRKD4		RETRIEVE, and LOAD/INSERT.
332(14C)	4	PSTWRKD5		
336(150)	4	PSTWRKD6		
340(154)	4	PSTWRKD7		
344(158)	4	PSTCURWA		Current delete work area
348(15C)	4	PSTDLTWA		First delete work area address
352(160)	84	PSTDLROM		Save and maintenance work area for retrieve
DATA BASE LOG SECTION				
436(1B4)	4	PSTLOGWA		Work area address for log O/P
440(1B8)	4	PSTLOGQ		Address of reuse queue QCB in pool

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
DATA BASE LOG USE OF PSTWRK1				
		PSTWRK1		Physical SDB address. If new block, low-order 2 bytes are call ccunt. High-order byte used for function code
DATA BASE LOG FUNCTION CODES				
		DBLNDXC	80	Index maintenance call
		DBLCMC	00	Bits 1-3 = 0, chain maintenance call
		DBLNTCR	70	Counter maintenance
		DBLLGDLT	60	Delete byte maintenance
		DBLPHYI	40	Insert
		DBLPHYD	20	Physical delete
		DBLPHYR	10	Replace
		DBLOOPS	0A	No data. End of user call
		DBLLASTC	08	Last change for user call
		DBLFSE1	00	Bit 5 = 0, one FSE (if bits 1 or 2 on)
		DBLFSE2	04	Two FSEs (if bits 1 or 2 on)
		DBLPHYRO	02	Old copy of a replace
		DBLNEWBL	01	New block log call
DATA BASE LOG USE OF PSTWRK2 - PSTWRK4				
Chain maintenance - Old copy of chain pointer (4 bytes). Insert/Delete - Offset and new FSEs (6 or 12 bytes)				
444(1BC)	4	PSTRAEND		End of root addressable area used by space manager
448(1C0)	4			**Reserved**
PARTITION/TASK INFORMATION				
452(1C4)	8	PSTPCPGM		Application program name. If batch UDR, ULR,or ULU; DBD name
460(1CC)	8	PSTPCPSB		PSB name
468(1D4)	1	PSTPCT1		Partition/task option
		PSTBATCH	80	PST is in batch partition
		PSTLODU	40	Load utility
		PSTLODUH	20	Load HDAM DB
		PSTHISMR	10	HISAM data base recovery in process
		PSTUST	08	Statistics utility
		PSTUDR	04	Data base recovery utility
		PSTULU	02	Data base load/unload utility
		PSTUSM	01	Security maintenance utility

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
469(1D5)	1	PSTPCT2		Program options/information overlaid on every call to the batch program request handler
		PSTXCONM	80	Exclude console message
		PSTXPRTM	40	Exclude printer message
		PSTCHKP	04	User checkpoint call successful
		PSTHLPI	02	Application program using HLPI
		PSTPLI	01	User program is PL/I
470(1D6)	1	PSTERCOD		Error message codes
470(1D6)	1	PSTERCD1		Error message code byte one
471(1D7)	1	PSTERCD2		Error message code byte two
472(1D8)	7	PSTERDT1		Error message data for ACB or DTF name
479(1DF)	6	PSTCNVB		Doubleword for HD randomizing module
479(1DF)	6	PSTERDT2		Variable error message data
487(1E7)	1	PSTERIND		Error routine indicator
		PSTDUMPI	80	Issue dump after error message put
		PSTCANLI	40	Issue cancel after error message put
488(1E8)	72	PSTLIPRM		Area to build user parameter list and register save area for MPS start and stop calls
560(230)	8	PSTPLIPR		PL/I region STXIT processor
568(238)	4	PSTNORO		Number of owned resources
572(23C)	0	PSTQLEV		Queue request level
		PSTQLRO	00	Read only level
		PSTQLUPD	04	Update level
		PSTQLEXC	08	Exclusive level
572(23C)	4	PSTRRDF		Pointer to first RRD
576(240)	4	PSTRRDL		Pointer to last RRD
580(244)	4	PSTRPSTA		Remote PST (RPST) address. Contains 0 if not scheduled to a remote PSB.
584(248)	12	PSTSAVTR		Trace save area - used only if output = CICS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
596(254)	4	PSTFLD		Start of field level descriptor block entries
600(258)	4	PSTFLDN		Next available field level descriptor entry
604(25C)	4	PSTFLDE		Field level descriptor area end address plus one
		PSTFLDAL	128	Initial field level descriptor area length
608(260)	4	PSTFLDC		Current field level descriptor entry for this level (retrieve)
PARTIAL REORGANIZATION CONTROL FIELDS				
612(264)	4	PSTPRTGT		Pointer to partial reorganization target table
616(268)	4	PSTDELTA		Partial reorganization HDAM RBA block number change
620(26C)	1	PSTOPEN		Flag byte
		PSTOPEN1	40	Open for partial reorganization
621(2C1)	3			** Reserved **
624(270)	16			** Reserved **
REGISTER SAVE AREA				
640(280)	72	PSTSV1		PSTSV1 through PSTSV7 are seven register save areas required for processing DL/I user calls. The convention used in storing registers in these save areas is to begin with R14 and end with R12; that is, R14, R15, R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, and R12.
712(2C8)	72	PSTSV2		
784(310)	72	PSTSV3		
856(358)	72	PSTSV4		
928(3A0)	72	PSTSV5		
1000(3E8)	72	PSTSV6		
1072(430)	72	PSTSV7		
1144(478)		PSTLNGTH		Length of PST (PSTLNGTH-PST)

QWA - QUEUING FACILITY WORK AREA

DSECT Name: DLZQWA

The QWA contains information used by the queuing facility module to build control blocks and RDB queue headers. It also contains information used to locate the proper RDB for a particular resource ID.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
QWACPP	20 (14)	
*QWADDDF	24 (18)	01
QWAFLG1	24 (18)	
QWAFLG2	25 (19)	
QWAFLG3	26 (1A)	
QWAFLG4	27 (1B)	
QWAFPP	16 (10)	
QWAHMLT	36 (24)	
QWANOQH	32 (20)	
*QWANPOF	24 (18)	02
QWARDBQH	40 (28)	
QWAWFLD	28 (1C)	

RECORD LAYOUT - QWA

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	16			Module ID
16(10)	4	QWAFPP		First page pointer for free block management
20(14)	4	QWACPP		Current page pointer for free block management
24(18)	1	QWAFLG1		First flag byte
		QWADDDF	01	Do deadlock detection
		QWANPOF	02	New prime owner exists
25(19)	1	QWAFLG2		Second flag byte
26(1A)	1	QWAFLG3		Third flag byte
27(1B)	1	QWAFLG4		Fourth flag byte
28(1C)	4	QWAWFLD		Work field 1
32(20)	4	QWANOQH		Number of queue heads
36(24)	4	QWAHMLT		Hashing Multiplier
40(28)	4	QWARDBQH		RDB chain queue headers(one fullword entry)

RDB - RESOURCE DESCRIPTOR BLOCK

DSECT Name: DLZRDB

The RDB (Resource Descriptor Block) is used to describe a rescurce for which enqueues are outstanding. In addition, it acts as an anchor for the chains of RRDs (Resource Request Descriptors) that describe the current queue requests for the resource.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
RDB	0 (00)	
RDBLEN	24 (18)	
RDBMAXL	8 (08)	
RDBNOWN	12 (0C)	
RDBPOID	0 (00)	
RDBRDBB	4 (04)	
RDBRDBF	0 (00)	
RDBRID	16 (10)	
RDBRRDF	8 (08)	
RDBRRDL	12 (0C)	
RDBUCID	4 (04)	

RECORD LAYOUT - RDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)		RDB		
0(00)	1	RDBPOID		Primary owner PST prefix number
0(00)	4	RDBRDBF		RDB forward chain pointer
4(04)	1	RDBUOID		Update owner PST prefix number
4(04)	4	RDBRDBB		RDB backward chain pointer
8(08)	1	RDBMAXL		Top enqueue level of current owners
8(08)	4	RDBRRDF		Pointer to first RRD
12(0C)	1	RDBNOWN		Current number of owners
12(0C)	4	RDBRRDL		Pointer to last RRD
16(10)	7	RDBRID		Resource ID (described in Section 3)
23(17)	1			**Reserved**
24(18)		RDBLEN		Length of RDB

RGT - RANGE TABLE

DSECT Name: RGT

This DSECT describes one range of keys or blocks to be reorganized. The range table is part of the common area. There are ten RGT entries available. They are completed by parameter analysis from data supplied by the user in his control cards. This control block is used by the partial reorganization utility.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
RGTFBKHI	4 (004)	
RGTFBKLO	0 (000)	
RGTFLEN	90 (05A)	5C
RGTFTHI1	12 (00C)	
RGTFTHI2	20 (014)	
RGTFTHI3	28 (01C)	
RGTFTHI4	36 (024)	
RGTFTHI5	44 (02C)	
RGTFTHI6	52 (034)	
RGTFTHI8	68 (044)	
RGTFTHI9	76 (04C)	
RGTFTH10	84 (054)	
RGTFTLO1	8 (008)	
RGTFTLO2	16 (010)	
RGTFTLO3	24 (018)	
RGTFTLO4	32 (020)	
RGTFTLO5	40 (028)	
RGTFTLO6	48 (030)	
RGTFTLO7	56 (038)	
RGTFTLO8	64 (040)	
RGTFTLO9	72 (048)	
RGTFTL10	80 (050)	
RGTKEYAR	90 (05A)	5C
RGTKIND1	88 (058)	
RGTKIND2	89 (059)	
RGTSTART	0 (000)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	RGTSTART		
0(000)	4	RGTFBKLO		First block in range to be reorganized
4(004)	4	RGTFBKHI		Last block in range to be reorganized
8(008)	4	RGTFTLO1		First block in data set group 1 for reload to use
12(00C)	4	RGTFTHI1		Last block in data set group 1 for reload to use
16(010)	4	RGTFTLO2		Same as for data set group 2
20(014)	4	RGTFTHI2		Same as for data set group 2
24(018)	4	RGTFTLO3		
28(01C)	4	RGTFTHI3		
32(020)	4	RGTFTLO4		
36(024)	4	RGTFTHI4		
40(028)	4	RGTFTLO5		
44(02C)	4	RGTFTHI5		
48(030)	4	RGTFTLO6		
52(034)	4	RGTFTHI6		
56(038)	4	RGTFTLO7		
60(03C)	4	RGTFTHI7		
64(040)	4	RGTFTLO8		
68(044)	4	RGTFTHI8		
72(048)	4	RGTFTLO9		
76(04C)	4	RGTFTHI9		
80(050)		RGTFTL10		First block in data set group 10 for relcad to use
84(054)		RGTFTH10		Last block in data set group 10 for reload to use
88(058)		RGTKIND1		Key range format indicator 1 (C or X)
89(059)		RGTKIND2		Key range format indicator 2 (C or X)
90(05A)				** Reserved **
		RGTFLEN		"*-RGTSTART" Length of a RGT entry

RIB - REMOTE INTERFACE BLOCK

DSECT Name: DLZRIB

This DSECT describes remote interface block fields. The RIB is used by DL/I for CICS/VS intersystem communication (ISC) support. It defines fields passed between CICS/VS and DL/I.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*RIBBUFAL	18 (12)	40
*RIBCALL	20 (14)	40
RIBCHAIN	4 (04)	
RIBCHKP	18 (20)	20
RIBDLTR	22 (16)	
RIBFCTR	21 (15)	
*RIBFUNC	20 (14)	80
RIBHLPI	19 (13)	40
RIBINDEX	16 (10)	
RIBIOAWK	8 (08)	
RIBIOLEN	24 (18)	
RIBISC	18 (12)	
RIBISCI	20 (14)	
RIBISCO	19 (13)	
*RIBLEN		1C
*RIBLNKNA	20 (14)	20
*RIBLNKSH	20 (14)	10
RIBNOSTT	20 (14)	08
RIBPCBAL	0 (00)	
*RIBPCBM	18 (12)	80
RIBRSET	23 (17)	
*RIBSYNC	19 (13)	80
RIBUPPER	12 (0C)	

RECORD LAYOUT - RIB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RIB		Start of RIB DSECT. This control block follows immediately after the RPST
0(00)	4	RIBPCBAL		Local PCB address list
4(04)	4	RIBCHAIN		Remote PSB storage chain
8(08)	4	RIBIOAWK		Local PSB I/O work area
12(0C)	4	RIBUPPER		Highest address of caller partition
16(10)	2	RIBINDEX		PCB index number
18(12)	1	RIBISC		ISC scheduling duration flags:
		RIBPCBM	80	PCBM scheduling call issued
		RIBBUFAL	40	RIBIOAWK buffer allocated
		RIBCHKP	20	DL/I checkpoint call in progress
19(13)	1	RIBISCO		ISC outbound flags:
		RIBHLPI	40	DL/I HLPI command with SSA and I/O lengths provided
		RIBSYNC	80	Synchronization point issued
20(14)	1	RIBISCI		ISC inbound flags:
		RIBFUNC	80	Function string invalid
		RIBCALL	40	User call parameter list invalid
		RIBLNKNA	20	Link does not exist
		RIBLNKSH	10	Link is out of service
		RIBNOSTT	08	CICS not counting DL/I calls
21(15)	1	RIBFCTR		ISC response code
22(16)	1	RIBDLTR		Additional response information
23(17)	0	RIBRSET	04	Length of function dependent flags
23(17)	1			** Reserved **
24(18)	4	RIBIOLEN		I/O area length for HLPI data base command
		RIBLEN	1C	Length of RIB

RPCB - REMOTE PCB

DSECT Name: DLZRPCB

This DSECT describes remote PCB fields. The RPCB is an extension of PCB local storage used by DL/I for CICS/VS intersystem communication (ISC) support. RPCBs exist only while a task is scheduled for a data base that is located on some other system. In this case, the address of the RPCB is located four bytes ahead of the PCB.

RECORD LAYOUT - RPCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPCB		Start of RPCB DSECT.
0(00)	4	RPCBMIOS		Maximum PCB I/O area size.
4(04)	4	RPCBSEGL		Length of last retrieve.
8(08)	1	RPCBFLAG		Flag byte:
		RPCBPATH	80	Previous get hold path call.
9(09)	3	Unnamed		**Reserved**
12(0C)		RPCBLEN		Length of RPCB

RPDIR - REMOTE PSB DIRECTORY

DSECT Name: DLZRPDIR

This DSECT describes remote PSB directory fields. The RPDIR is an extension of the PDIR. It contains PSB information used by DL/I for CICS/VS intersystem communication (ISC) support.

RECORD LAYOUT - RPDIR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPDIR		Start of RPDIR DSECT
0(00)	4	RPDIRSYS		System name on which remote PSB is defined
4(04)	8	RPDIRPSB		Name of PSB to use on remote system
12(0C)	2	RPDIRLOC		Optional local PSB PDIR pointer
14(0E)	1	RPDIRFLG		Flag byte
		RPDIREXT		
		RPDIRORD	01	Local PCBs follow remote
16(10)		RPDIRLEN		Length of RPDIR

RPST - REMOTE PST

DSECT Name: DLZRPST

This DSECT describes remote PST fields. The RPST is an extension of task local storage used by DLZODP for CICS/VS intersystem communication (ISC) support.

RECORD LAYOUT - RPST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPST		Start of RPST DSECT
0(00)	4	RPSTISC1		ISC parameter 1
4(04)	4	RPSTISC2		ISC parameter 2
8(08)	4	RPSTISC3		ISC parameter 3
12(0C)	4	RPSTISC4		ISC parameter 4
16(10)	4	RPSTISC5		ISC parameter 5
20(14)	4	RPSTISC6		ISC parameter 6
24(18)	1	RPSTATUS		Flag byte
25(19)	3	RPSTACTA		Program's ACT entry address
28(1C)	4	RPSTRPSB		Remote PSB PDIR entry address
32(20)	4	RPSTRPCB		Remote PSB PCB address list address
36(24)	4	RPSTXPSB		Local PSB PDIR entry address
40(28)	4	RPSTXPCB		Local PSB PCB address list address
44(2C)	0	RPSTACCT		Remote call statistics
44(2C)	4	RPSTGU		Number of GU calls issued
48(30)	4	RPSTGN		Number of GN calls issued
52(34)	4	RPSTGNP		Number of GNP calls issued
56(38)	4	RPSTGHU		Number of GHU calls issued
60(3C)	4	RPSTGHN		Number of GHN calls issued

64(40)	4	RPSTGHNP	Number of GHNP calls issued
68(44)	4	RPSTISRT	Number of ISRT calls issued
72(48)	4	RPSTDLET	Number of DLET calls issued
76(4C)	4	RPSTREPL	Number of REPL calls issued
80(50)	4	RPSTCHKP	Number of CHKP calls issued
		RPSTSTLN	Length of remote status section (*-RPSTACCT)
		RPST	Length of RPST (*-RPST)

RRD - RESOURCE REQUEST DESCRIPTOR

DSECT Name: DLZRRD

The RRD (Resource Request Descriptor) is used to maintain a record of all the requests by one task for a particular resource and their current status.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
RRD	0 (00)	
RRDFLAG	16 (10)	
RRDLEN	18 (24)	
RRDMAXL	12 (0C)	
RRDNQEX	8 (08)	
RRDNQRO	0 (00)	
RRDNQUP	4 (04)	
*RRDOWNF	16 (10)	01
*RRDPOWNF	16 (10)	04
RRDPSTP	20 (14)	
RRDPSTQB	4 (04)	
RRDPSTQF	0 (00)	
RRDRDBP	16 (10)	
RRDRDBQB	12 (0C)	
RRDRDBQF	8 (08)	
*RRDWAITF	16 (10)	02

RECORD LAYOUT - RRD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)		RRD		
0(00)	1	RRDNQRO		Number of read-only ownerships for task
0(00)	4	RRDPSTQF		PST queue forward pointer; next RRD for task
4(04)	1	RRDNQUP		Number of exclusive (update) ownerships for task
4(04)	4	RRDPSTQB		PST queue backward pointer; prior RRD for task
8(08)	1	RRDNQEX		Number of exclusive ownerships for task
8(08)	4	RRDRDBQF		RDB queue forward pointer; next RRD for resource
12(0C)	1	RRDMAXL		Current maximum ownership level for resource by task
12(0C)	4	RRDRDBQB		RDB queue backward pointer; prior RRD for resource
16(10)	1	RRDFLAG		Flag byte
		RRDOWNF	01	PST owns resource
		RRDWAITF	02	PST is waiting for resource
		RRDPOWNF	04	PST is prime owner of resource
16(10)	4	RRDRDBP		RDB address for resource
20(14)	4	RRDPSTP		PST address for task
24(18)	4	RRDLEN		Length of RRD

SBIF - SUBPOOL INFORMATION TABLE

DSECT Name: SUBINFTA

The subpool information table is described as part of the general structure and description of DL/I buffer pool control blocks. There is one subpool information table for each subpool allocated.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SUBBFHD	3 (03)	
SUBBFNO	2 (02)	
SUBBFSIZ	44 (2C)	
SUBDMBCT	45 (2D)	
SUBDUMP	3 (03)	40
*SUBFRSV	3 (03)	80
SUBLEN	46 (2E)	
SUBNQFI	0 (00)	
SUBNQLA	1 (01)	
SUBUCHAI	8 (08)	
SUBUCPRE	4 (04)	
SUBUCSUF	40 (28)	
SUBUSCHA	4 (04)	

RECORD LAYOUT - SBIF

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	SUBNQFI		PST prefix number of first task in chain for enqueue subpool
1(01)	1	SUBNQLA		PST prefix number of last task in chain for enqueue subpool
2(02)	1	SUBBFNO		Number of buffers in this subpool
3(03)	1	SUBBFHD		HDBFR indicator
		SUBFRSV	80	DMB assigned to this subpool by HDBFR parameter
		SUBDUMP	40	Buffers associated with subpool dump
4(04)	4	SUBUSCHA		Buffer use chain
4(04)	4	SUBUCPRE		Accumulated number of buffers in preceeding subpools
8(08)	32	SUBUCHAI		Buffer use chain
40(28)	4	SUBUCSUF		(Not used in DL/I DOS/VS)
44(2C)	1	SUBBFSIZ		Size of the buffers in this subpool: X'01' = 512 bytes X'02' = 1024 bytes X'03' = 1536 bytes X'04' = 2048 bytes X'05' = 2560 bytes X'06' = 3072 bytes X'07' = 3584 bytes X'08' = 4096 bytes
45(2D)	1	SUBDMBCT		Number of DMBs assigned
46(2E)	0	SUBLEN		Length of subpool information table

SCD - SYSTEM CONTENTS DIRECTORY

DSECT Name: DLZSCD

The DL/I SCD (System Contents Directory) is produced during DL/I system definition for online CICS/VS-DL/I. The SCD is preassembled as part of the DL/I nucleus in the batch DL/I system. The SCD contains major entry pointers for all DL/I facilities.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
CPYRITE	0 (000)	
SCD	96 (060)	
SCDABEND	200 (0C8)	
SCDABSAV	288 (120)	
SCDACTBA	264 (108)	
SCDASE	196 (0C4)	
SCDATSKC	106 (06A)	
SCDBFPL	216 (0D8)	
SCDBKWRK	352 (160)	
SCDCDTA	268 (10C)	
SCDCMTCT	384 (180)	
*SCDCMTI	284 (11C)	40
SCDCMXT	104 (068)	
SCDCOMRG	124 (07C)	
SCDCPY10	180 (0B4)	
SCDCSABA	276 (114)	
SCDCWRK	336 (150)	
SCDCWRKL	340 (154)	
SCDDATE	98 (062)	
*SCDDBASL	346 (15A)	02
SCDDBFA	217 (0D9)	
SCDDBFPL	216 (D8)	
SCDDBLAS	324 (144)	
*SCDDBLCJ	346 (15A)	20
SCDDBLCL	320 (140)	
*SCDDBLD2	346 (15A)	10
SCDDBLFW	316 (13C)	
SCDDBMPS	304 (130)	
SCDDBLNT	148 (094)	
*SCDDBLO	346 (15A)	80
SCDDBLOP	346 (15A)	
*SCDDBLOR	346 (15A)	40
*SCDDBLSP	346 (15A)	08
SCDDBLSV	328 (148)	
*SCDDBLTD	346 (15A)	20
SCDDBLWO	332 (14C)	
SCDDDBH0	136 (088)	
*SCDDELT	284 (11C)	20
SCDDHDS0	160 (0A0)	
*SCDDLARE	144 (090)	28
SCDDLICL	168 (0A8)	
SCDDLICT	144 (090)	
SCDDLIDL	232 (0E8)	
SCDDLIDM	228 (0E4)	
SCDDLIDN	234 (0EA)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SCDDLIDR	152 (098)	
SCDDLIIN	156 (09C)	
SCDDLIM	97 (061)	
SCDDLIPL	224 (0E0)	
SCDDLIPN	226 (0E2)	
SCDDLIPS	220 (0DC)	
SCDDLIRE	140 (08C)	
SCDDLIS	272 (110)	
SCDDLIUP	276 (114)	
SCDDLIV	96 (060)	
SCDDLOCT	380 (17C)	
SCDDSEH0	172 (0AC)	
SCDDXMT0	164 (0A4)	
SCDERRMS	192 (0C0)	
SCDEXTBA	300 (12C)	
SCDFLPC	244 (0F4)	
*SCDFLSAV	244 (0F4)	40
*SCDHLRE	284 (11C)	08
SCDIWAIT	188 (0BC)	
*SCDLIPLI	244 (0F4)	80
SCDLNGTH	520 (208)	
SCDLOCOU	348 (15C)	
SCDLOWER	108 (06C)	
SCDLOWID	120 (078)	
SCDLSTAD	292 (124)	
SCDMPCPT	296 (128)	
*SCDMTI	284 (11C)	80
SCDMXTSK	102 (066)	
*SCDNABND	284 (11C)	01
SCDNAVID	116 (074)	
*SCDNDMP	284 (11C)	04
*SCDNJNL	284 (11C)	01
*SCDNLOGI	284 (11C)	02
SCDNTWC	286 (11E)	
SCDPATCH	392 (188)	
SCDPDUP	388 (184)	
*SCDPI	304 (130)	40
SCDPPAB	248 (0F8)	
SCDPPAF	244 (0F4)	
SCDPPFB	256 (100)	
SCDPPFF	252 (0FC)	
SCDPPSTL	240 (0F0)	
SCDPPSTN	242 (0F2)	
SCDPPSTS	236 (0EC)	
SCDPRHED	132 (084)	
SCDPSTLN	260 (104)	
*SCDQFJRN	172 (0AC)	08
*SCDQFSDC	172 (0AC)	04
SCDQUEFW	176 (0B0)	
SCDQUEF0	172 (0AC)	
SCDREENT	312 (138)	
*SCDRELOD	285 (11D)	08
SCDREPLN	344 (158)	
*SCDRLABN	285 (11D)	04
*SCDRLRST	285 (11D)	10
*SCDRPSB	304 (130)	20
SCDSEQ	342 (156)	
SCDSIND	284 (11C)	
SCDSIND2	285 (11D)	
*SCDSOPLG	285 (11D)	01
SCDSPCNT	282 (11A)	
SCDSTR00	208 (0D4)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*SCDSYACT	285 (11D)	40
*SCDSYINT	285 (11D)	02
*SCDSYSAB	285 (11D)	80
*SCDSYWAT	285 (11D)	20
*SCDTAMOD	368 (170)	40
*SCDTBHCL	368 (170)	02
*SCDTCPOS	368 (170)	10
*SCDTINDX	368 (170)	01
SCDTKCNT	280 (118)	
SCDTKTRM	204 (0CC)	
*SCDTOLBH	369 (171)	80
*SCDTPITR	369 (171)	40
SCDTRACE	356 (164)	
SCDTRCNM	360 (168)	
*SCDTRETR	368 (170)	20
SCDTRFL1	368 (170)	
SCDTRFL2	369 (171)	
SCDTSKCR	372 (174)	
*SCDTUSER	368 (170)	80
*SCDTVSAM	368 (170)	04
*SCDTWFI	284 (11C)	08
*SCDUPD	284 (11C)	10
SCDUPPER	112 (070)	
SCDUSAVE	244 (0F4)	
SCDWAIT	262 (106)	
*SCDXECB	304 (130)	80

RECORD LAYOUT - SCD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	96	CPYRITE		Reserved for copyright information
96(60)	0	SCD		Start of addressable SCD
SYSTEM CONFIGURATION SECTION				
96(60)	1	SCDDLIV		DL/I version number
97(61)	1	SCDDLIM		DL/I release level
98(62)	4	SCDDATE		System date - Julian
102(66)	2	SCDMXTSK		DL/I minimum task count - online
104(68)	2	SCDCMXT		DL/I current maximum task - online
106(6A)	2	SCDATSKC		Active DL/I task counter - online
108(6C)	4	SCDLOWER		Partition lower boundary; address pointer to addressable part of the SCD (batch only)
112(70)	4	SCDUPPER		Partition upper boundary address
116(74)	4	SCDNAVID		Next available task ID
120(78)	4	SCDLOWID		Lowest task ID
124(7C)	4	SCDCOMRG		COMREG address
128(80)	4			**Reserved**
ACTION MODULE ENTRY POINT ADDRESSES				
132(84)	4	SCDPRHED		Entry point of program request handler: Batch = DLZPRHB0 Online = DLZPRH00
136(88)	4	SCDDDBH0		Entry point of buffer handler (DLZDBH00)
140(8C)	4	SCDDLIRE		Entry point of retrieve (DLZDLR00)
144(90)	4	SCDDLICT		Entry point of call analyzer (DLZDLA00)
		SCDDLARE	28	Offset to entry point on return to call analyzer
148(94)	4	SCDDBLNT		Entry point of data base log module (DLZRDBL0) = entry point of log

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				initialization until after initialization
152(98)	4	SCDDLID2		Entry point of delete/replace (DLZDLD00)
156(9C)	4	SCDDLIIN		Entry point of load/insert for retrieve (DLZDDLE0)
160(A0)	4	SCDDHDS0		Entry point of space management (DLZDHDS0)
164(A4)	4	SCDDXMT0		Entry point of index maintenance (DLZDXMT0)
168(A8)	4	SCDDLICL		Entry point of open/close (DLZDLOC0)
172(AC)	4	SCDDSEH0		Entry point of routine to create work files for batch only (DLZDSEH0)
172(AC)	4	SCDQUEF0		Entry point of enqueue/dequeue module for program isolation - online only (DLZQUEF0)
		SCDQFSDC	04	Displacement to SCD address field in DLZQUEF0
		SCDQFJRN	08	Displacement to JRNAD exit address field in DLZQUEF0
176(B0)	4	SCDQUEFW		Enqueue/dequeue work area
180(B4)	4	SCDCPY10		Entry point for field level sensitivity expansion routine (DLZCPY10)
184(B8)	4			**Reserved**
188(BC)	4	SCDIWAIT		Entry point of IWAIT routine: Batch = DLZIWAIT Online = DLZOWAIT
192(C0)	4	SCDERRMS		Entry point of error message routine: Batch = ERRORMSG Online = DLZERMSG
196(C4)	4	SCDASE		Entry point of online schedule and termination (DLZSCHDL)
200(C8)	4	SCDABEND		Entry point of DL/I ABEND routine: Batch = DLZABEND Online = DLZABND0

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
204(CC)	4	SCDTKTRM		Entry point of online task termination for program request handler (DLZTKTRM)
208(D0)	4	SCDSTR00		Entry point of FLD storage manager (Batch=DLZSTRB0) (Online=DLZSTR00)
212(D4)	4			** Reserved **
SYSTEM CONTROL BLOCK SECTION				
216(D8)	0	SCDDBFPL		Label for buffer handler
216(D8)	1	SCDBFPL		Number of buffer subpools
217(D9)	3	SCDDBFA		Address of buffer pool control block prefix (DLZBFPL)
220(DC)	4	SCDDLIPS		Address of PSB directory (DLZPDIR)
224(E0)	2	SCDDLIPL		Length of PDIR entries
226(E2)	2	SCDDLIPN		Number of PDIR entries
228(E4)	4	SCDDLIDM		Address of DMB directory (DLZDDIR)
232(E8)	2	SCDDLIDL		Length of DDIR entries
234(EA)	2	SCDDLIDN		Number of DDIR entries
236(EC)	4	SCDPPSTS		Address of PST prefix entries (DLZPPST)
240(F0)	2	SCDPPSTL		Length of PPST entries
242(F2)	2	SCDPPSTN		Number of PPST entries
244(F4)	4	SCDPPAF		Online forward PST prefix active pointer
244(F4)	4	SCDUSAVE		Used for MPS or batch. Contains address of user savearea where DL/I registers are saved.
244(F4)	1	SCDFLPC		Flag byte (used for MPS or batch):
		SCDLIPLI	80	0 = Currently executing in DL/I code (or in a user program that is not written in PL/I). 1 = Currently executing in PL/I code.
		SCDFLSAV	40	0 = User savearea used for STXIT PC.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				1 = DL/I savearea used for STXIT PC.
248(F8)	4	SCDPPAB		Online backward PST prefix active pointer
252(FC)	4	SCDPPFF		Online forward PST prefix free pointer (DLZPPSTF)
256(100)	4	SCDPPFB		Online backward PST prefix free pointer (DLZPPSTE)
260(104)	2	SCDPSTLN		Length of PST
262(106)	2	SCDWAIT		Number of tasks waiting for CMAX
264(108)	4	SCDACTBA		Address of online application program control table (DLZACTBA)
268(10C)	4	SCDCDTA		Address of current online dispatched task's TCA
272(110)	4	SCDDLIS		Address of first online task suspended
276(114)	4	SCDDLIUP		Address of batch DL/I upper boundary
276(114)	4	SCDCSABA		Address of online CICS CSA
280(118)	2	SCDTKCNT		Count of DL/I tasks assigned PPST
282(11A)	2	SCDSPCNT		Count of suspended tasks due to maximum task
284(11C)	1	SCDSIND		System indicator
		SCDMTI	80	DL/I Maximum task indicator
		SCDCMTI	40	DL/I current maximum task indicator
		SCDDELT	20	Online indicator for PSB has delete sensitivity
		SCDUPD	10	Online indicator for PSB has update sensitivity
		SCDTWFI	08	Task waiting for segment intent
		SCDHLRE	08	High level language reentry indicator STXIT
		SCDNDMP	04	No dump at ABEND
		SCDNLOGI	02	No data base logging to be done
		SCDNABND	01	Batch - no STXIT ABEND to be issued
		SCDNJNL	01	Online - no CICS journal in use
285(11D)	1	SCDSIND2		System flags

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		SCDSYSAB	80	System ABEND online
		SCDSYACT	40	System task active
		SCDSYWAT	20	System task waiting
		SCDRLRST	10	HD reload/restart
		SCDRELOD	08	HD reload utility
		SCDRLABN	04	HD reload or reload/restart ABEND is in process
		SCDSYINT	02	Initialization bit
		SCDSOPLG	01	Open records written to CICS journal
286(11E)	2	SCDNTWC		Segment intent wait counter
288(120)	4	SCDABSAV		Pointer to pseudo ABEND save area (DLZABSAV)
292(124)	4	SCDLSTAD		Address of CICS interface address list (DLZDLIAL)
296(128)	4	SCDMPCPT		Address of MPC partition table
300(12C)	4	SCDEXTBA		Pointer to SCD extension
304(130)	1	SCDDBMPS		Flag Byte
		SCDXECB	80	XECBs defined by MPC
		SCDPI	40	Program isolation active.
		SCDRPSB	20	Remote PSB defined.
305(131)	1			**Reserved**
306(132)	2			**Reserved**
308(134)	4			**Reserved**
DATA BASE CHANGE LOG SECTION				
312(138)	4	SCDREENT		Entry point of log write only
316(13C)	4	SCDDBLFW		Entry point of log force write
320(140)	4	SCDDBLCL		Entry point of log close routine
324(144)	4	SCDDBLAS		Entry point of asynchronous log
328(148)	4	SCDDBLSV		Entry point of log save area
332(14C)	4	SCDDBLWO		Entry point of write log open record
336(150)	4	SCDCWRK		Address of DB log work area
340(154)	2	SCDCWRKL		Length of DB log work area

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
342(156)	2	SCDSEQ		DB log sequence number
344(158)	2	SCDREPLN		Length of DB log prefix
346(15A)	1	SCDDBLOP		Data base log option byte
		SCDDBLO	80	DB log is open
		SCDDBLOR	40	DB log open required
		SCDDBLTD	20	Disk logging used
		SCDDBLD2	10	Two disk extents used
		SCDDBLSP	08	Pause before extent switch
		SCDDBLCJ	04	CICS journal in use
		SCDDBASL	02	DB asynchronous log required
347(15B)	1			**Reserved**
348(15C)	2	SCDLOCOU		Current log count
350(15E)	2			**Reserved**
352(160)	4	SCDBKWRK		Backout log workarea pointer.
TRACE SECTION				
356(164)	4	SCDTRACE		Entry point of trace module if present
360(168)	8	SCDTRCNM		Name of trace module
368(170)	1	SCDTRFL1		Trace option byte 1
		SCDTUSER	80	User call interface
		SCDTAMOD	40	Action module trace
		SCDTRETR	20	Retrieve (for GET calls)
		SCDTCPOS	10	Current position information
		SCDTVSAM	04	VSAM interface
		SCDTBHCL	02	Buffer handler interface
		SCDTINDX	01	Requests to index maintenance
369(171)	1	SCDTRFL2		Trace option byte 2
		SCDTOLBH	80	Online trace
		SCDTPITR	40	Program isolation trace
370(172)	2			**Reserved**
STATISTICS SECTION (Online only)				
372(174)	8	SCDTSKCT		Total number of PSB scheduling calls
380(17C)	4	SCDDLOCT		Program isolation deadlock occurrence count
384(180)	4	SCDCMTCT		Number of times at current maximum task
388(184)	4	SCDPDUP		Number of duplicate PSBs created

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
392(188)	128	SCDPATCH		DL/I patch area
520(208)		SCDLNGTH		Length of SCD

SCDEXT - SCD EXTENSION

DSECT Name: SCDEXTDS

The SCD extension is generated in the same manner as the SCD (system contents directory) and is a logical extension of it.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SCDAPSTR	24 (18)	Batch usage
SCDEABEX	4 (04)	Batch usage
SCDEABSV	8 (08)	Batch usage
SCDEFECB	8 (08)	Online usage
SCDEIDNX	24 (18)	Online usage
SCDEIDST	20 (14)	Online usage
SCDEIDWK	28 (1C)	Online usage
SCDELECB	0 (00)	Online usage
SCDEMSGT	32 (20)	Online usage
SCDEPASS	16 (10)	Online usage
SCDEPCEX	12 (0C)	Batch usage
SCDEREEN	0 (00)	Batch usage
SCDESECB	4 (04)	Online usage
SCDETRAN	16 (10)	Batch usage
SCDETRSV	20 (14)	Batch usage
SCDETRTB	36 (24)	Online usage and batch usage
SCDETRTE	40 (28)	Online usage and batch usage
SCDETRTS	44 (2C)	Online usage and batch usage
SCDEVSEX	12 (0C)	Online usage
SCDEXLEN	52 (34)	Online usage

RECORD LAYOUT - SCDEXT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
Online Usage of the SCD Extension				
0(00)	4	SCDELECB		Logger I/O ECB
4(04)	4	SCDESECB		System enqueue ECB
8(08)	4	SCDEFECB		System function call ECB
12(0C)	4	SCDEVSEX		Address of VSAM EXCP exit (DLZOVSEX)
16(10)	4	SCDEPASS		Address of system password (DLZPASS)
20(14)	4	SCDEIDST		Address of first PPST ID assigned (DLZIDLST)
24(18)	4	SCDEIDNX		Address of last active PPST ID (DLZIDLST)
28(1C)	4	SCDEIDWK		Address of PPST search table (DLZIDWRK)
32(20)	4	SCDEMSGT		Address of online message module (DLZMMSGT)
36(24)	4	SCDETRTB		Current entry in incore table
40(28)	4	SCDETRTE		End address +1 of trace table
44(2C)	4	SCDETRTS		Start address of trace table
48(30)	4			**Reserved**
52(34)		SCDEXLEN		Length of SCD extension
Batch Usage of SCD Extension				
0(00)	4	SCDEREEN		Address of utility block call entry point
4(04)	4	SCDEABEX		Address of STXIT ABEND routine (DLZAABND)
8(08)	4	SCDEABSV		Address of STXIT ABEND save area
12(0C)	4	SCDEPCEX		Address of STXIT PC routine (DLZPABND)
16(10)	4	SCDETRAN		Address of ABTERM transient area
20(14)	4	SCDETRSV		Address of transient save area

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
24(18)	4	SCDAPSTR		Application program start address
28(1C)	8			(Not used in batch)
36(24)	4	SCDETRTB		Current entry in incore table
40(28)	4	SCDETRTE		End address +1 of trace table
44(2C)	4	SCDETRTS		Start address of trace table
48(30)	4			**Reserved**

SDB - SEGMENT DESCRIPTION BLOCK

DSECT Name: SDB

The segment description block (SDB) is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*SDBALTSC	11 (0B)	20
*SDBALTSQ	11 (0B)	40
*SDBCISP	11 (0B)	04
*SDBCTR	37 (25)	80
*SDBDCHG	11 (0B)	01
SDBDDIR	12 (0C)	
SDBDSGA	28 (1C)	
SDBEND	60 (3C)	
*SDBFLS	56 (38)	02
SDBF3	10 (0A)	
SDBF4	11 (0B)	
*SDBGEN	32 (20)	10
SDBKEYFD	40 (28)	
SDBKEYLN	24 (18)	
SDBLEN	60 (3C)	
SDBLEVEL	8 (08)	
*SDBLP	37 (25)	02
*SDBLTPK	37 (25)	04
*SDBLTFD	37 (25)	08
SDBLTN	0 (00)	
SDBLTP	0 (00)	
SDBNSDB	16 (10)	
*SDBORGHD	9 (09)	20
*SDBORGHI	9 (09)	10
*SDBORGHS	9 (09)	02
*SDBORGH1	9 (09)	04
SDBORGN	9 (09)	
*SDBORGRI	9 (09)	44
*SDBORGSH	9 (09)	05
*SDBORGSS	9 (09)	01
SDBPARA	24 (18)	
SDBPCB	39 (27)	
SDBPCF	38 (26)	
*SDBPCTSP	32 (20)	40
SDBPHYCD	12 (0C)	
SDBPOSC	48 (30)	
*SDBPOSL	11 (0B)	02
SDBPOSN	52 (34)	
SDBPOSP	44 (2C)	
*SDBPP	37 (25)	10
*SDBPPST	32 (20)	80
*SDBPPTSP	32 (20)	C0
SDBPSDB	20 (14)	
*SDBPTB	37 (25)	20
SDBPTDS	37 (25)	
*SDBPTF	37 (25)	40

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
SDBPTNO	36 (24)		
*SDBSEND	10 (0A)	10	
*SDBSENG	10 (0A)	80	
*SDBSENI	10 (0A)	40	
*SDBSENK	10 (0A)	08	
*SDBSENL	10 (0A)	01	
*SDBSENP	10 (0A)	04	
*SDBSENR	10 (0A)	20	
*SDBSENX	10 (0A)	02	
*SDBSLC	32 (20)	02	
*SDBSLP	32 (20)	01	
*SDBSNX	32 (20)	04	
*SDBSPP	32 (20)	08	
SDBSYM	0 (00)		
SDBTARG	33 (21)		
SDBTFLG	32 (20)		
SDBXFFSB	16 (10)		(See SDBXP block at end of SDB)
SDBXFISL	6 (06)		(See SDBXP block at end of SDB)
SDBXFL	56 (38)		
SDBXFLAG	12 (0C)		(See SDBXP block at end of SDB)
SDBXFLEN	16 (10)		(See SDBXP block at end of SDB)
SDBXFLN	2 (02)		(See SDBXP block at end of SDB)
SDBXFNB	1 (01)		(See SDBXP block at end of SDB)
*SDBXFNR	12 (0C)	80	(See SDBXP block at end of SDB)
SDBXFSBP	8 (08)		(See SDBXP block at end of SDB)
SDBXFUSL	4 (04)		(See SDBXP block at end of SDB)
SDBXPANS	56 (38)		
SDBXPASF	16 (10)		(See SDBXP block at end of SDB)
SDBXPEND	20 (14)		(See SDBXP block at end of SDB)
SDBXPFDB	0 (00)		(See SDBXP block at end of SDB)
*SDBXPFS	0 (00)	02	(See SDBXP block at end of SDB)
SDBXPMSK	4 (04)		(See SDBXP block at end of SDB)
*SDBXPRES	56 (38)	01	
*SDBXPSI	0 (00)	01	(See SDBXP block at end of SDB)
SDBXPSZ	20 (14)		(See SDBXP block at end of SDB)
SDBXPTYP	0 (00)		(See SDBXP block at end of SDB)
SDBXSQLN	14 (0E)		(See SDBXP block at end of SDB)
SDBXSQOF	12 (0C)		(See SDBXP block at end of SDB)
SDBXWMSK	8 (08)		(See SDBXP block at end of SDB)

RECORD LAYOUT - SDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	SDBSYM		Segment symbolic name
0(00)	4	SDBLTP		Prior segment on logical twin chain
0(00)	4	SDBLTN		Next segment on logical twin chain
8(08)	1	SDBLEVEL		Level of this segment (logical)
9(09)	1	SDBORGN		Organization of data base containing segment
		SDBORGRI	44	This segment is root of index
		SDBORGHD	20	This segment is in a HDAM organization
		SDBORGHI	10	This segment is in a HIDAM organization
		SDBORGSH	05	This segment is in a simple HISAM organization
		SDBORGH1	04	This segment is in a HISAM organization
		SDBORGHS	02	This segment is in an HSAM organization
		SDBORGSS	01	This segment is in a simple HSAM organization
10(0A)	1	SDBF3		Call sensitivity
		SDBSENG	80	Sensitivity is read only
		SDBSENI	40	Sensitivity is insert
		SDBSENR	20	Sensitivity is replace
		SDBSEND	10	Sensitivity is delete
		SDBSENK	08	Sensitivity is key only
		SDBSENP	04	Sensitivity is path only
		SDBSENX	02	Sensitivity is exclusive
		SDBSENL	01	Sensitivity is load
11(0B)	1	SDBF4		Code byte
		SDBALTSQ	40	Secondary index is main processing sequence
		SDBALTSC	20	Secondary index search fields require conversion
		unnamed	10	** Reserved **
		SDBCISP	04	Control interval split occurred in HISAM KSDS
		SDBPOSL	02	Position lost
		SDBDCHG	01	Temporary switch for replace; data changed
12(0C)	1	SDBPHYCD		Segment code
12(0C)	4	SDBDDIR		DMB directory address
16(10)	4	SDBNSDB		Next SDB for this PSDB
20(14)	4	SDBPSDB		Address of PSDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
24(18)	1	SDBKEYLN		Executable key length of key field
24(18)	4	SDBPARA		Parent SDB (address of PCB for root SDB) or address of prior SDB on 'SDBTARG' chain for generated SDBs (SDBGEN on in SDBTFLG)
28(1C)	4	SDBDSGA		Address of data set group section of JCB for data set containing segment
32(20)	1	SDBTFLG		Logical relationship code
		SDBPPTSP	C0	Segment is physical parent of target of SDBPARA
		SDBPPSP	80	Segment is physical parent of SDBPARA
		SDBPCTSP	40	Segment is physical child of target of SDBPARA
		SDBGEN	10	This SDB is a generated SDB
		SDBSPP	08	Segment is a virtual logical child
		SDBSNX	04	Segment is retrieved via index
		SDBSLC	02	(See bit flag 0001 0010)
		SDBSLP	01	Segment is a logical child

SDBTFLG Bit Flags

1xx0 xxxx	Inverted structure - The segment logically above this one is below it in the physical data base hierarchy. The segment logically above this one is represented by the SDB pointed to in SDBPARA. If SDBPARA points to a SDB for a logical child, this segment could be physically above either the logical child or its destination parent. A generated SDB pointed to by SDBTARG in the logical child's SDB represents the destination parent.
x1x0 xxxx	Logical relation - The segment represented by the SDB pointed to by SDBPARA is a logical child and this segment is either the physical parent or a physical child of its destination parent.
10x0 xxxx	This segment is the physical parent of the segment represented by the SDB identified as SDBPARA.
11x0 xxxx	The segment represented by the SDB pointed to in SDBPARA is a logical child and this segment is the physical

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
			parent of its destination parent (SDBTARG).	
		01x0 xxxx	The segment represented by the SDB pointed to in SDBPARA is a logical child and this segment is a physical child of its destination parent.	
		xxx0 1xxx	This segment is the logical child in a virtual logical child concatenated segment and SDBTARG point to the logical child's physical parent.	
		xxx0 xxx1	This segment is the logical child in a normal concatenated segment and SDBTARG points to the logical parent.	
		xxx1 xxxx	SDB is a generated SDB.	
		0001 0010	SDB is a generated SDB for an index. If SDBTARG is non-zero, it points to the generated SDB for the index target.	
		0001 0110	SDB is a generated SDB for a HIDAM root segment. SDBTARG points to the SDB for the primary index segment.	
33(21)	3	SDBTARG		Address of the logically related segments SDB
36(24)	1	SDBPTNO		Pointer number of first physical pointer
37(25)	1	SDBPTDS		Physical pointer flag
		SDBCTR	80	This logical parent segment has a counter
		SDBPTF	40	This segment has a physical twin forward pointer
		SDBPTB	20	This segment has a physical twin backward pointer
		SDBPP	10	This segment has a physical parent pointer
		SDBLTFD	08	This segment has a logical twin forward pointer
		SDBLTBK	04	This segment has a logical twin backward pointer
		SDBLP	02	This segment has a logical parent pointer
38(26)	1	SDBPCF		Pointer number in parent to first occurrence of this segment type
39(27)	1	SDBPCB		Pointer number in parent to last occurrence of this segment type

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
40(28)	4	SDBKEYFD		The address within DBPCBKFD for key this segment. In generated SDB for logical destination parent: Byte 0 = physical segment code of logical child Bytes 1-3 = logical child's PSDB address In generated SDB for physical destination parent: Byte 0 = Physical segment code of virtual logical child Bytes 1-3 = virtual logical child's PSDB address
44(2C)	4	SDBPOSP		Previous position
48(30)	4	SDBPOSC		Current position. X'80' in high-order byte = position lost, in conjunction with SDBPOSL in SDBF4
52(34)	4	SDBPOSN		Next position (current position in generated SDBs)
56(38)	1	SDBXFL		SDB expansion flag
		SDBXPRES	01	SDB expansion for secondary index processing sequence is present. (Secondary index is main processing sequence.)
		SDBFLS	02	Segment has field level sensitivity
56(38)	4	SDBXPANS		SDB expansion address
60(3C)		SDBEND		End of SDB entry
60(3C)		SDBLEN		Length of each SDB (SDBEND minus SDBSYM)

SDB EXPANSION BLOCK

DSECT Name: SDBXP

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning

This block is present if indicated in SDB; see field SDBXFL, flag SDBXPRES.

0(00)	1	SDBXPTYP		SDB expansion type
		SDBXPSI	01	SDB expansion is for secondary index
		SDBXPFS	02	SDB expansion is for field sensitivity
0(00)	3	SDBXPFDB		Address of secondary index sequence field FDB
4(04)	4	SDBXPMSK		Mask of XDFLD FDBs allowed in SSAs
8(08)	4	SDBXWMSK		Work area reserved for open/close
12(0C)	2	SDBXSQOF		Offset from DBPCBKFD to SUBSEQ area (0 if area not present)
14(0E)	2	SDBXSQLN		Length of SUBSEQ field(s) minus 1
16(10)	4	SDBXPASF		Alternate sequence FSB pointer
20(14)		SDBXPEND		End of SDB expansion block entry
20(14)		SDBXPSZ		Length of one SDB expansion block entry (SDBXPEND minus SDBXP)

SDB EXPANSION BLOCK FOR FIELD SENSITIVITY

1(01)	1	SDBXFNB		Number of FSBs
2(02)	2	SDBXFLN		Length of expansion block
4(04)	2	SDBXFUSL		Length of segment in user's view
6(06)	2	SDBXFISL		Insert length of segment
8(08)	4	SDBXFSBP		ACBGEN - first FSB address
12(0C)	1	SDBXFLAG		Flags
		SDBXFNR	80	At least one NOREPL rule
13(0D)	3			**Reserved**
16(10)	0	SDBXFEND		End of SDB expansion block entry
16(10)	0	SDBXFLEN		Length of one SDB expansion block

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
16(10)	0	SDBXFFSB		Start of first FSB

SEC - SECONDARY LIST

DSECT Name: DMBSEC

The secondary list is described as part of the general structure and description of the DMB. The labels in SEC vary with the type of secondary index entry. See the field description listed by code type in the record layout.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*DMBEXIT	1 (01)	02	(See Code 40)
*DMBEXLOD	1 (01)	04	(See Code 40)
*DMBEXTRN	0 (00)	40	
DMBFDFLG	1 (01)		(See Code 04)
DMBFDOFF	6 (06)		(See Code 04)
*DMBFDONE	1 (01)	10	(See Code 04)
*DMBFDUSE	1 (01)	01	(See Code 04)
*DMBINDXD	0 (00)	44	
DMBIPSDB	8 (08)		(See Code 64)
DMBISSOF	2 (02)		(See Code 64)
DMBISSSC	8 (08)		(See Code 64)
DMBNBYTE	4 (04)		(See Code 40)
*DMBNXISS	0 (00)	60	
*DMBNXXDS	0 (00)	64	
DMBSCDE	0 (00)		
DMBSECDB	4 (04)		(See Code 01)
DMBSECLN	16 (10)		(See Code 64)
DMBSECND	16 (10)		(See Code 64)
DMBSECNM	8 (08)		(See Code 01)
DMBSECSC	4 (04)		(See Code 01)
DMBSFCEN	12 (0C)		(See Code 08)
DMBSFD	2 (02)		(See Code 01)
DMBSFLEN	13 (0D)		(See Code 08)
DMBSFLG	1 (01)		(See Code 01)
DMBSFLG1	1 (01)		(See Code 40)
DMBSFNAM	2 (02)		(See Code 08)
DMBSFOFF	10 (0A)		(See Code 08)
DMBSFPSC	1 (01)		(See Code 08)
DMBSKYLN	1 (01)		(See Code 60)
*DMBSLC	0 (00)	02	
*DMBSLCF	0 (00)	08	
DMBSLCFL	2 (02)		(See Code 02)
DMBSLCIR	1 (01)		(See Code 02)
*DMBSLP	0 (00)	01	
*DMBSND	0 (00)	80	
*DMBSNULL	1 (01)	01	(See Code 40)
DMBSOFF	2 (02)		(See Code 44)
*DMBSOURC	0 (00)	20	
*DMBSRCH	0 (00)	04	
*DMBSUBSQ	0 (00)	24	
*DMBSYMN1	1 (01)	04	(See Code 04)
DMBSYMOF	14 (0E)		(See Code 44)
*DMBSYM1	1 (01)	08	(See Code 04)
*DMBSYSFD	1 (01)	02	(See Code 04)
*DMBVKY	1 (01)	C'V'	(See Code 01)

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*DMBXDCON	12 (0C)	08	(See Code 44)
*DMBXDEQ	12 (0C)	01	(See Code 44)
DMBXDFLG	12 (0C)		(See Code 44)
*DMBXDLST	12 (0C)	80	(See Code 44)
DMBXDPAD	13 (0D)		(See Code 44)
DMBXDSC	8 (08)		(See Code 44)
DMBXDSDB	4 (04)		(See Code 44)
*DMBXDSPC	12 (0C)	10	(See Code 44)
DMBXDSSC	4 (04)		(See Code 44)
*DMBXDSSQ	12 (0C)	04	(See Code 44)
*DMBXDSSS	12 (0C)	20	(See Code 44)
*DMBXDSYM	12 (0C)	40	(See Code 44)
DMBXITAD	4 (04)		(See Code 40)
DMBXNSDB	4 (04)		(See Code 60)
DMBXNSSC	4 (04)		(See Code 60)
DMBXPSDB	8 (08)		(See Code 44)
DMBXSOFF	14 (0E)		(See Code 08)

RECORD LAYOUT - SEC

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	DMBSCDE		Code byte
		DMBSLP	01	Secondary list describes a logical parent
		DMBSLC	02	Secondary list describes a logical child
		DMBSRCH	04	Secondary list describes index search field(s)
		DMBSLCF	08	Secondary list describes logical twin sequence field
		DMBSOURC	20	Secondary list describes index DDATA field(s)
		DMBSUBSQ	24	Secondary list describes index SUBSEQ field(s)
		DMBEXTRN	40	Secondary list describes index user exit routine
		DMBINDXD	44	Secondary list describes index target segment as seen from index pointer segment
		DMBNXISS	60	Secondary list describes index relationship as seen from index source segment
		DMBNXXDS	64	Secondary list describes index relationship as seen from index target segment. This list is not present if ISS=TARGET
		DMBSND	80	Last entry in secondary list

THE FOLLOWING FIELDS ARE LISTED BY CODE TYPE

CODE 01 - DESCRIBES LOGICAL PARENT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
1(01)	1	DMBSFLG		
		DMBVKY	C'V'	Key of logical parent is virtual
2(02)	2	DMBSFD		Logical parent key length
4(04)	1	DMBSECSC		Segment code of referenced segment
4(04)	4	DMBSECDB		DDIR address of referenced data base
8(08)	8	DMBSECNM		Segment name of referenced segment

CODE 02 - DESCRIBES LOGICAL CHILD

1(01)	1	DMBSLCIR		Logical twin sequence insert rule
2(02)	2	DMBSLCFL		Number of first and last logical child pointers in logical parent prefix

Remaining fields are same as Code 01.

CODE 04 - DESCRIBES INDEX SEARCH FIELDS

1(01)	5	DMBFDFLG		Five 1-byte flags associated with the following FDB offsets
		DMBSYM1	08	First part of symbolic pointer
		DMBSYMN1	04	Not first part of symbolic pointer (middle or last)
		DMBSYSFD	02	This slot for system-related field
		DMBFDUSE	01	This slot in use
		DMBFDONE	10	This entry processed by block builder
6(06)	10	DMBFDOFF		Offset to FDB from first FDB of ISS if this slot is in use. Otherwise, zero.

CODE 08 - DESCRIBES LOGICAL TWIN SEQUENCE FIELD

1(01)	1	DMBSFPSC		Virtual logical child physical segment code
2(02)	8	DMBSFNAM		FDB field name
10(0A) segment	2	DMBSFOFF		Offset to field in
12(0C)	1	DMBSFCEN		Code byte (same as FDBDCENF in FDB)
13(0D)	1	DMBSFLEN		Executable field length

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
14(0E)	2	DMBXSOFF		Offset to field in indexed segment

CODE 20 - DESCRIBES DDATA FIELD

Same fields as Code 04

CODE 24 - DESCRIBES SUBSEQ FIELD

Same fields as Code 04

CODE 40 - DESCRIBES INDEX EXIT ROUTINE

1(01)	1	DMBSFLG1		Flag byte
		DMBSNULL	01	Null field present
		DMBEXIT	02	Exit routine present
		DMBEXLOD	04	Exit routine has been
loaded				
2(02)	2			***Reserved***
4(04)	4	DMBNBYTE		If index field equals this byte, bypass indexing
4(04)	4	DMBXITAD		Address of index maintenance parameter CSECT
8(08)	8			***Reserved***

CODE 44 - DESCRIBES INDEX TARGET SEGMENT

1(01)	1	DMBSKYLN		Executable length of key
2(02)	2	DMBSOFF		Offset to PSDB address pointer of index target segment
4(04)	4	DMBXDSSC		Segment code of index target segment
4(04)	4	DMBXDSDB		DDIR address of index target segment
8(08)	4	DMBXDSC		Segment code of index target segment
8(08)	4	DMBXPSDB		PSDB address of index target segment
12(0C)	1	DMBXDFLG		Code byte from associated FDB
		DMBXDLST	80	Last FDB in list
		DMBXDSYM	40	Index pointer is symbolic
		DMBXDSSS	20	Pointer contained in source/subseq data
		DMBXDSPC	10	Special FDB for secondary index
		DMBXDCON	08	Constant present
		DMBXDSSQ	04	SUBSEQ present
		DMBXDSOR	02	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		DMBXDEQ	01	XDS=ISS
13(0D)	1	DMBXDPAD		Padding constant
14(0E)	2	DMBSYMOF		Offset to symbolic pointer indexing segment

CODE 60 - DESCRIBES INDEX FROM ISS

1(01)	3			Same as code 44
4(04)	1	DMBXNSSC		Segment code of index pointer segment
4(04)	4	DMBXNSDB		DDIR address of index

Remaining fields same as Code 44

CODE 64 - DESCRIBES INDEX FROM INDEX TARGET

1(01)	1			Same as code 44
2(02)	2	DMBISSOF		Offset to Code 60 from start of ISS seccndary list
4(04)	4			Same as code 60
8(08)	1	DMBISSSC		Segment code of index source segment
8(08)	4	DMBIPSDB		PSDB address of index source segment
12(0C)	1			Same as code 44
16(10)		DMBSECND		End of each secondary list entry
16(10)		DMBSECLN		Length of each secondary list entry

SGT - SEGMENT TABLE

DSECT Name: SGT

This DSECT describes the segments used by the partial reorganization process. It is built during the DBD analysis phase and used by all subsequent phases in PART1 and PART2. Its address is held in the common area field (COMASGT). Associated with the SGT is the segment extension table (SGX).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SGTCDS	61 (03D)	
SGTCLEV	62 (03E)	
SGTCNAME	0 (000)	
SGTCSC	60 (03C)	
SGTFCNT1	16 (010)	
SGTFCNT2	20 (014)	
SGTFCNT3	24 (018)	
SGTFCNT4	28 (01C)	
SGTFCNT5	32 (020)	
SGTFCNT6	36 (024)	
SGTGATR1	64 (040)	
SGTGATR2	65 (041)	
SGTGATR3	66 (042)	
SGTGATR4	67 (043)	
SGTHDLEN	48 (030)	
SGTHKLEN	44 (02C)	
SGTHPLEN	46 (02E)	
*SGTLLEN	104 (068)	6C
SGTODBT	40 (028)	
SGTOKEY	42 (02A)	
SGTOPCF	56 (038)	
SGTORACT	50 (032)	
SGTOSACT	52 (034)	
SGTOSIBL	58 (03A)	
*SGTQDRCT	65 (041)	04
*SGTQDSEN	67 (043)	40
*SGTQHB	64 (040)	02
*SGTQHIDR	64 (040)	40
*SGTQHIER	64 (040)	04
*SGTQKSEN	67 (043)	80
*SGTQLC	65 (041)	80
*SGTQLCL	65 (041)	01
*SGTQLP	66 (042)	40
*SGTQLTB	65 (041)	02
*SGTQMOVE	64 (040)	80
*SGTQNOLT	66 (042)	80
*SGTQNPRO	67 (043)	20
*SGTQPCL	64 (040)	08
*SGTQPP	64 (040)	20
*SGTQPPR	65 (041)	10
*SGTQPTB	64 (040)	10
*SGTQPTF	66 (042)	01
*SGTQSCAN	67 (043)	01
*SGTQSOPT	67 (043)	02

*SGTQSYM	65 (041)	08
*SGTQUNID	65 (041)	40
*SGTQVPR	65 (041)	20
*SGTQVRLN	64 (040)	01
*SGTQXDRT	66 (042)	08
*SGTQXSX	66 (042)	02
SGTRNEW	12 (00C)	
SGTROLD	8 (008)	
SGTSTART	0 (000)	
*SGTUNIQ	66 (042)	04
SGXFBLK	104 (068)	
SGXOCTR	68 (044)	
SGXOHB	84 (054)	
SGXOHIER	82 (052)	
SGXOLCF	88 (058)	
SGXOLCWK	102 (066)	
SGXOLP	80 (050)	
SGXOLTB	78 (04E)	
SGXOLTF	76 (04C)	
SGXOPAIR	94 (05E)	
SGXOPCF	86 (056)	
SGXOPCWK	100 (064)	
SGXOPP	74 (04A)	
SGXOPTB	72 (048)	
SGXOPTF	70 (046)	
SGXOSLP	92 (05C)	
SGXOSPP	90 (05A)	
SGXOSRCE	98 (062)	
SGXOTARG	96 (060)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	SGTSTART		
0(000)	8	SGTCNAME		Segment name un/reloded
12(00C)	4	SGTRNEW		New RBA of last segment reloaded
16(010)	4	SGTFCNT1		Statistical counter
20(014)	4	SGTFCNT2		Statistical counter
24(018)	4	SGTFCNT3		Statistical counter
28(01C)	4	SGTFCNT4		Statistical counter
32(020)	4	SGTFCNT5		Statistical counter
36(024)	4	SGTFCNT6		Statistical counter
40(028)	2	SGTODBT		Offset to DBT entry for this segments DB
42(02A)	2	SGTOKEY		Segment key start POS root only
44(02C)	2	SGTHKLEN		Segment key length roots only
46(02E)	2	SGTHPLEN		Segment prefix length

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
48(030)	2	SGTHDLEN		Segment data length maximum if variable
50(032)	2	SGTORACT		Offset in ACT to first reload action
52(032)	2	SGTOSACT		Offset in ACT to first scan action
54(036)	2			Spare offset field
56(038)	2	SGTOPCF		Offset in SGT to first physical child of this segment
58(03A)	2	SGTOSIBL		Offset in TGT to next SESIBLING segment
60(03C)	1	SGTCSC		DL/I segment code
61(03D)	1	SGTCDS		DL/I data set code
62(03E)	1	SGTCLEV		DL/I level code
63(03F)	1			** Reserved **
64(040)	1	SGTGATR1		Segment physical attributes
		SGTQMOVE	80	Segment to be moved for reorganization
		SGTQHIDR	40	Segment is HIDAM root
		SGTQPP	20	Segment has PP pointer
		SGTQPTB	10	Segment has PTB pointer
		SGTQPCL	08	Segments parent has PCL pointer to this
		SGTQHIER	04	Segment has hierarchic pointers
		SGTQHB	02	Segment has hierarchic backward pointer
		SGTQVRLN	01	Segment is variable length
65(041)	1	SGTGATR2		Segment logical attributes
		SGTQLC	80	Segment is a logical child
		SGTQUNID	40	Segment is logical child unidirectional relation
		SGTQVPR	20	Segment has virtual pair
		SGTQPPR	10	Segment has physical pair
		SGTQSYM	08	Segment has only symbolic pointer to logical parent
		SGTQDRCT	04	Segment has direct pointer to logical parent
		SGTQLTB	02	Segment has LTB pointer
		SGTQLCL	01	Segments logical parent has LCL pointer to this
66(042)	1	SGTGATR3		Segment logical and index attributes
		SGTQNOLT	80	Virtually paired with no logical twin pointers

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		SGTQLP	40	Segment is a logical parent
		SGTQXDRT	08	Segment is index segment with direct pointer
		SGTUNIQ	04	Segment is in a unique index
		SGTQXSX	02	Segment is index segment with SX field
		SGTQPTF	01	Segment has a PTF pointer
67(043)	1	SGTGATR4		Segment PSB attributes
		SGTQKSEN	80	Key only sensitivity required
		SGTQDSEN	40	Data sensitivity required
		SGTQNPRO	20	Segment not processed used to reach physical child
		SGTQSOPT	02	Scan is option for this segment
		SGTQSCAN	01	This segment will be scanned

S E G M E N T   E X T E N S I O N   T A B L E

This part of the DSECT is for additional information about the segments used by the partial reorganization process. It contains offsets needed to create the action table (ACT). It is created during the DBD analysis phase.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
68(044)	2	SGXOCTR		Offset in prefix of log
70(046)	2	SGXOPTF		Offset in prefix of PTF REL counter pointer
72(048)	2	SGXOPTB		Offset in prefix of PTB pointer
74(04A)	2	SGXOPP		Offset in prefix of PP pointer
76(04C)	2	SGXOLTF		Offset in prefix of LTF pointer
78(04E)	2	SGXOLTB		Offset in prefix of LTB pointer
80(050)	2	SGXOLP		Offset in prefix of logical parent pointer
82(052)	2	SGXOHIER		Offset in prefix of hier pointer
84(054)	2	SGXOHB		Offset in prefix of hier back pointer
86(056)	2	SGXOPCF		Offset in segments physical parent of PCF to this segment
88(058)	2	SGXOLCF		Offset in segments logical parent of LCF to this segment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
90(05A)	2	SGXOSPP		Offset in SGT of physical parent
92(05C)	2	SGXOSLP		Offset in SGT of logical parent
94(05E)	2	SGXOPAIR		Offset in SGT of physical pair
96(060)	2	SGXOTARG		Offset in SGT of target of this segment
98(062)	2	SGXOSRCE		Offset in SGT of source of this segment
100(064)	2	SGXOPCWK		Work area to hold offset to first physical child
102(066)	2	SGXOLCWK		Work area to hold offset to first logical child pointer
104(068)	4	SGXFBLK		Last block un/reloaded used in PART2
		SGTLLEN		"*-SGTSTART" length of a SGT entry

SSAP - SEGMENT SEARCH APPENDAGE

DSECT Name: SSAP

This DSECT describes the fields contained in the DL/I HLPI Segment Search Argument get path call appendage.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SSAAPLN	14 (0E)	08
SSAP	0 (00)	
SSAPDATT	8 (08)	40
SSAPFLAG	8 (08)	
SSAPIOA	9 (09)	
SSAPLEN	14 (0E)	10
SSAPLIOA	12 (0C)	
SSAPPROC	8 (08)	20
SSAPSEGM	0 (00)	
SSAPSGOF	14 (0E)	
SSAPSTOR	14 (0E)	F0
SSAPVARL	8 (08)	80

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	8	SSAPSEGM		Segment name
8(008)	1	SSAPFLAG		SSA flag
		SSAPVARL	80	Variable length segment
		SSAPDATT	40	Data to be transferred
		SSAPPROC	20	Segment already processed
9(09)	3	SSAPIOA		Address of I/O area for this segment
12(00C)	2	SSAPLIOA		Length of the I/O area for this segment
14(00E)	2	SSAPSGOF		Offset to length of the destination parent
		SSAPLEN	10	"*-SSAPSEGM" length of SSA appendage
		SSAPSTOR	F0	"SSAPLEN*15" length for required number of SSA appendages
		SSAAPLN	08	"*-SSAPFLAG" length of appendage information

STA - STATISTICS TABLE

DSECT Name: STA

This layout describes the fields used for gathering statistics by the partial reorganization utility. The fields are initialized and incremented by UNLOAD and RELOAD. The data is referenced by the statistics writer when formatting statistical reports.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
STATONTR	0 (000)	
STBLBK40	2 (002)	
STBLBH41	82 (052)	
STBLCT	0 (000)	
STHASHS	88 (058)	
STHDOV	212 (0D4)	
STLOHICT	92 (05C)	
STMXBL	84 (054)	
STNDCNT	216 (0D8)	
STRG	216 (0D8)	
STROV	172 (0AC)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4			
0(00)	2	STBLCT		Block count
2(002)	80	STBLBK40		Counters for blocks 1 to 40
82(052)	2	STBLBK41		Counter for blocks over 40
84(054)	2	STMXBL		Maximum number of blocks this range
88(058)	4	STHASHS		Number of blocks over 40
92(05C)	80	STLOHICT		10 pairs of low-high block numbers
172(0AC)	40	STROV		For reload
212(0D4)	4	STHDOV		HDAM roots in overflow
216(0D8)	2	STNDCNT		
0(000)	216	STATCNTR		Length statistic counters
216(0D8)	2	STRG		Range counter for statistics

DLZTWAB - TRANSACTION WORK AREA

DSECT Name: DLZTWAB

The DLZTWAB macro provides the mapping for the batch partition controller's transaction work area. The information is used for communication with:

- DL/I task termination
- CICS/VS
- Batch partition
- Sheduling MPS batch jobs
- Online message module

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
TWABEND	202 (CA)	
TWABPC	0 (00)	
TWABPCID	4 (04)	
*TWABPCOK	0 (00)	80
TWABPCSV	76 (4C)	
TWABPSCD	56 (38)	
TWACALL	40 (28)	
TWACOND	192 (C0)	
*TWAEOJSW	0 (00)	40
TWAMPCE	5 (05)	
TWAMPCPT	1 (01)	
TWAMPSFG	0 (00)	
TWAMPSID	180 (B4)	
TWAMSG	148 (94)	
TWAMSGID	152 (98)	
TWAMSGNO	148 (94)	
TWAMSG01	156 (9C)	
TWAMSG02	160 (A0)	
TWAMSG03	164 (A4)	
TWAMSG04	168 (A8)	
TWAN1PTR	32 (20)	
TWAPARMC	36 (24)	
TWAPSBDL	55 (37)	
TWAPSBN	44 (2C)	
TWAPSBNM	48 (30)	
TWAPSW	172 (AC)	
TWARCODE	190 (BE)	
TWASCHDC	36 (24)	
TWAWLIST	8 (08)	
TWAXCBDL	16 (10)	
TWAXCBN1	24 (18)	
TWAXCBN2	20 (14)	
TWAXCB2	8 (08)	
TWAXCB3	12 (0C)	
TWAXNAME	182 (B6)	

RECORD LAYOUT - DLZTWAB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
THE FOLLOWING FIELDS ARE USED FOR COMMUNICATING WITH THE DL/I TASK TERMINATION ROUTINE				
0(00)	0	TWABPC		Start of TWABPC
0(00)	1	TWAMPSFG		BPC flag byte:
		TWABPCOK	80	BPC abnormal termination processing completed
		TWAEOJSW	40	EOJ processing reached for MPS batch partition
1(01)	3	TWAMPCPT		Address of MPC partition table
4(04)	1	TWABPCID		Batch partition XECB identifier
5(05)	3	TWAMPCE		Address of specific MPC partition table entry
THE FOLLOWING IS THE BATCH PARTITION CONTROLLER'S CICS/VS WAITM ECB LIST, DELIMITER, AND XECB				
8(08)	0	TWAWLIST		Start of TWAWLIST
8(08)	4	TWAXCB2		Pointer to BPC's XECB (DLZXCBn2)
12(0C)	4	TWAXCB3		Pointer to ABEND XECB (DLZXCBn3)
16(10)	4	TWAXCBDL		ECB list delimiter ('FFFFFFFF')
20(14)	4	TWAXCBN2		XECB for BPC
THE FOLLOWING FIELDS ARE USED FOR COMMUNICATION WITH THE BATCH PARTITION				
24(18)	8	TWAXCBN1		XECB name for batch initialization (DLZXCBn1)
32(20)	4	TWAN1PTR		XECBTAB table entry address for batch initialization's XECB (DLZXCBn1)
THE FOLLOWING FIELDS ARE USED FOR THE BATCH PARTITION CONTROLLER'S DL/I SCHEDULING CALL PARAMETER LIST AND THE PSBNAME TO BE SCHEDULED				
36(24)	0	TWASCHDC		Start of TWASCHDC
36(24)	4	TWAPARMC		Pointer to parameter count
40(28)	4	TWACALL		Pointer to call function
44(2C)	4	TWAPSBN		Pointer to PSB name

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
48(30)	7	TWAPSBNM		PSB name (PSBNAME)
55(37)	1	TWAPSBDL		PSB name delimiter
THE FOLLOWING FIELD CONTAINS THE SCD ADDRESS				
56(38)	4	TWABPSCD		Start of TWAPIDTE
56(38)	40			** Reserved **
BATCH PARTITION CONTROLLER REGISTER SAVE AREA				
76(4C)	72	TWABPCSV		BPC register save area (18 fullwords)
THE FOLLOWING ARE THE PARAMETER LIST POINTERS, PARAMETERS, AND MESSAGE FILLERS PASSED TO THE DL/I ONLINE MESSAGE MODULE (DLZERMSG) FOR ALL BPC MESSAGES				
148(94)	0	TWAMSG		Start of TWAMSG
148(94)	4	TWAMSGNO		Message number pointer for all BPC messages
152(98)	4	TWAMSGID		Partition ID pointer (for messages DLZ082I, DLZ084I, and DLZ103I) BPC module ID pointer (for message DLZ104I)
156(9C)	4	TWAMSG01		Module name pointer (for messages DLZ082I and DLZ084I) Termination condition pointer and delimiter (for message DLZ103I) CICS ABEND code pointer and delimiter (for message DLZ104I)
160(A0)	4	TWAMSG02		XECBTAB TYPE= pointer (for messages DLZ082I and DLZ084I) PSW pointer and delimiter (for message DLZ104I)
164(A4)	4	TWAMSG03		XECBTAB XECB=XECBname pointer (for messages DLZ082I and DLZ084I)
168(A8)	4	TWAMSG04		Return code pointer and delimiter (for messages DLZ082I and DLZ084I)
172(AC)	8	TWAPSW		Program interrupt PSW

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
180(B4)	2	TWAMPSID		Batch partitiion ID of the form BG, F1, F2, . . .
182(B6)	8	TWAXNAME		XECBTAB XECB=XECBname (DLZXCBnn)
190(BE)	2	TWARCODE		Return code
192(C0)	10	TWACOND		BPC termination condition (abnormally or normally)
202(CA)	4	TWABEND		CICS ABEND completion list entry

UIB - USER INTERFACE BLOCK

DSECT Name: DLIUIB

This control block is used by extended DL/I call interface support (along with CICS/VS high-level language support). This section contains scheduling and system call status information returned to the user. (Prior to Version 1.4, this information was returned to the user in the TCA.)

RECORD LAYOUT - UIB (USER SECTION)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	UIB		Start of UIB DSECT
0(00)	4	UIBPCBAL		PCB address list
4(04)	2	UIBRCODE		DL/I return codes
4(04)	1	UIBFCTR		Return code
5(05)	1	UIBDLTR		Additional information
6(06)	2	Unnamed		** Reserved **
8(08)		UIBLEN		Length of UIB (for Assembler language only)

UIB - USER INTERFACE BLOCK

DSECT Name: DLZUIB

The user section of this control block is used by extended DL/I call interface support (along with CICS/VS high-level language support). This section contains scheduling and system call status information returned to the user. (Prior to Version 1.4, this information was returned to the user in the TCA.) A system section of the UIB follows the user section. It is used by DL/I as task-local storage. Unlike PST storage, UIB storage is not released at scheduling termination.

RECORD LAYOUT - UIB (USER SECTION)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	UIB		Start of UIB DSECT
0(00)	4	UIBPCBAL		PCB address list
4(04)	2	UIBRCODE		DL/I return codes
4(04)	1	UIBFCTR		Return code
5(05)	1	UIBDLTR		Additional information
6(06)	2	Unnamed		** Reserved **
8(08)		UIBLEN		Length of UIB (for Assembler language only)

RECORD LAYOUT - UIB (SYSTEM SECTION)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
8(08)	64	UIBREGSV		Register save area
72(48)	8	UIBPSB		PSB name on scheduling call
80(50)	4	UIBFUNC		Call function type
84(54)	4	UIBSDIB		System DIB address
88(58)	1	UIBFLAG1		UIB Flag
		UIBSCHD	01	Scheduling call
		UIBDB	02	Data base call
		UIBTERM	04	Term call
		UIBMPS	08	UIB acquired for MPS task

		UIBXRPSB	20	Remote with local PSB scheduled
		UIBHLPI	40	HLPI command level program
		UIBREMOT	80	PSB on remote system
89(59)	1	UIBFLAG2		
		UIBDUMP	80	Task dump taken
90(5A)	1	UIBTYPSB		Type of PSB ' ' = Local '+' = Remote '*' = Local and remote
91(5B)	1	UIBRSTAT		Local and remote status
		UIBXBGUN	80	XPSB scheduled call in progress
		UIBXLOC	40	Local PSB scheduled
		UIBXREM	20	Remote PSB scheduled
		UIBXSTOR	10	PCB list storage acquired
		UIBXUNSC	08	Local PSB unscheduled
92(5C)	4	UIBPST		Task PST address
96(60)	4	UIBSUSP		Task suspend chain pointer
100(64)	4	UIBIPCBA		Internal address of PCB address list
104(68)	2	UIBICODE		Initial DL/I return code
106(6A)	1			** Reserved **
107(6B)	3	UIBTSKID		CICS/VS task ID
110(6E)	4	UIBMSGPM		Message parameter list
114(72)	4	UIBMSGP2		Second message parameter
118(76)		UIBMSGP3		Third message parameter
122(7A)	4	UIBWORK		Work area
126(7E)	72	UIBTRCSV		DLZOLT00 register save area
		UIBSLEN	C6	Length of user and system UIB

XMPRM - HDAM/HIDAM USER SECONDARY INDEX SUPPRESSION ROUTINE INTERFACE TABLE

DSECT Name: DMBXMPRM

This table is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBXMPLN	28 (1C)	
DMBXMRES	32 (20)	
DMBXMSGN	0 (00)	
DMBXMXDN	8 (08)	
DMBXMXEP	24 (18)	
DMBXMXNM	16 (10)	

RECORD LAYOUT - XMPRM

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBXMSGN		Name of indexed segment
8(08)	8	DMBXMXDN		Name of XDFLD
16(10)	8	DMBXMXNM		Name of user exit routine
24(18)	4	DMBXMXEP		Entry point of user exit routine
28(1C)	2	DMBXMPLN		Length of index maintenance parameters
30(1E)	2			** Reserved **
32(20)	4	DMBXMRES		Reserved for initialization

XWR - INDEX WORK RECORD

DSECT Name: XWR

This DSECT describes an index work record that is created by the partial reorganization utility while performing pointer maintenance.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
XWRCKEY	22 (016)	
XWRCRDB	20 (014)	
XWRCRDSG	21 (015)	
XWRCTYPE	18 (012)	
XWRFSEQ	12 (00C)	
XWRGFLAG	19 (013)	
XWRHLL	0 (000)	
XWRH00	2 (002)	
XWRLFIX	21 (015)	16
XWROACT	16 (010)	
XWRRCOMP	8 (008)	
XWRRMOVE	4 (004)	
XWRSTART	0 (000)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	XWRSTART		
0(000)	2	XWRHLL		VLR length control field
2(002)	2	XWRH00		VLR control binary zeros
4(004)	4	XWRRMOVE		New RBA of a moved segment
8(008)	4	XWRRCOMP		Old RBA of a segment for compare
12(00C)	4	XWRFSEQ		Record sequence number for nonunique index
16(010)	2	XWROACT		Offset in ACT that built this record
18(012)	1	XWRCTYPE		Record type code
19(013)	1	XWRGFLAG		Processing option flags
20(014)	1	XWRCRDB		Data base ID of segment to be updated
21(015)	1	XWRCRDSG		Data set group ID of segment to be updated
		XWRLFIX		"*-XWRSTART" length of fixed part of record
22(016)	1	XWRCKEY		Key of segment to be updated

RECORD LAYOUTS

The rest of this section provides layouts and field descriptions for the following records:

Accumulation Header Record
Accumulation Record
Application Program Scheduling Record
Application Program Termination Record
Checkpoint Log Record
Checkpoint Record
Control Data Set
Data Base Log Record
Data Record (Input)
Data Record (Output)
Date/Time Table
Delete Work Area
Delete Work Space Prefix
DL/I Control Record
Dump Header Record
Dump Record Prefix
File Open Record
Header Record (Input)
Header Record (Output)
Index Maintenance Work Area
List Control Block
Output Record Containing Segment Prefix
Output Table Record
Short Segment Table
Sorted List Block
SSA for GU Call by Key
SSA for GU Call by RBA
SSA for the XMAINT Call to the Analyzer
Statistics Record
Description of Variable Output
Work File 1
Description of Variable Input
Work File 3

ACCUMULATION HEADER RECORD

This record is used by modules DLZUC350 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	HLENGTH	2	Length of cum header record
2	2	HSPACE	2	Zeros
4	4	HCODE	1	Header record ID X'25'
5	5	HFLG	1	Type of data set X'02' VSAM ESDS X'04' VSAM KSDS
6	6	HLRECL	2	Record length
8	8	HORG	1	Prefix organization code
9	9	HPURGDT	7	Purge date/time for data base data set
9	9	HPURDATE	3	Purge date for data base data set -YYDDDF
C	12	HPURTIME	4	Purge time for data base data set -HHMMSS0F
10	16	HDDNAME	8	Data set symbolic filename
18	24	HDBNAME	8	Data base name
20	32	HDSID	1	Data set ID
21	33	HDATE	3	Run date - YYDDDF
24	36	HTIME	4	Run time - HHMMSS0F
28	40	HSEQ	2	Zeros
2A	42	HBLKSIZE	2	Zeros

ACCUMULATION RECORD

This record is used by modules DLZUC350 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	CLENGTH	2	Length of cum record
2	2	CSPACE	2	Zeros
4	4	CCODE	1	X'50' record identifier
5	5	CFLG	1	Type of data set/entry X'01' VSAM KSDS/Entry was VSAM ERASED X'02' VSAM ESDS X'04' VSAM KSDS
6	6	CIDLN	2	Length of CDATAID field
8	8	CDBNAME	8	Data base name

10	16	CDSID	1	Data set ID
11	17	CDATE	3	Date - YYDDDF
14	20	CTIME	4	Time - HHMMSSOF
18	24	CSEQ	2	Sequence number
1A	26	CCOUNT	2	Number of data elements in CDATA
1C	28	CDATAID	Var	KSDS prime key or ESDS RBN
		CDATAOL	Var	One or more 4 byte data elements: bytes 0-1 - offset into data set record bytes 2-3 - length of corresponding CDATASEG
		CDATASEG	Var	One or more segment data entries to be moved into data set record.

APPLICATION PROGRAM SCHEDULING RECORD

This record is used by modules DLZRDBL0, DLZRDBL1, DLZLOGP0, and DLZBACK0.

Hex	Dec	Name	Ln	Description
0	0	LENGTH	2	Length of record
2	2	SPACE	2	Binary zero
4	4	LOGFLAG	1	Record type code - X'08'
5	5	SCHDCODE	1	Task ID
8	8	PSBNAME	8	PSB name
E	14	CICSID	3	Packed CICS Transaction ID (online only)

APPLICATION PROGRAM TERMINATION RECORD

This record is used by modules DLZRDBL0, DLZRDBL1, DLZLOGP0, and DLZBACK0.

Hex	Dec	Name	Ln	Description
0	0	PLENGTH	2	Halfword binary length of logical record
2	2	PSPACE	2	Halfword reserved for system use (binary zero)
4	4	ALLOGFLG	1	Identifies this logical record as application program termination record; value is X'07'
5	5	ALPSBNAM	8	PSB name
D	13	ALID	1	TASK ID
E	14	TSKSTAT	40	10 fullwords of Accounting from PSTACCT (online only)
36	54	CICSID	3	Packed CICS transaction I.D. (online only)

CHECKPOINT LOG RECORD

Checkpoint log records are used to restart a job near its point of failure. The records are created and written on the DL/I log (if data base logging is active) if requested by the user via checkpoint calls (CHKP). Each log record contains a user-supplied unique checkpoint identification passed with the CHKP call.

In case of a job failure in a batch environment, the backout utility can be run to backout data base changes occurring since the last checkpoint record was written. For MPS and/or online tasks with CICS/VS dynamic transaction backout active, backout is performed automatically to the last checkpoint when a task fails.

Hex	Dec	Name	Ln	Description
0	0	CHKPLEN	2	Length of log record
2	2	CHKPSPC	2	Blanks/zeros
4	4	CHKPCODE	1	Log record ID
		CHKPLRID	41	Checkpoint Log record ID
5	5	CHKPPSB	8	Checkpoint PSB name
D	13	CHKPID	8	User checkpoint ID
15	21	CHKPRLEN		Length of checkpoint log record

CHECKPOINT RECORD

This DSECT (RCHKREC) defines the format of the checkpoint records within the unloaded data base for HD reorganization unload/reload utilities.

Hex	Dec	Name	Ln	Description
0	0	RCHKPTID	1	Identifies checkpoint record; Always X'00'
1	1	RCHKNAME	6	Constant for checkpoint record; Always C'CHKPNT'
7	7	RCHKNUM	4	Checkpoint number; 1-9999 (decimal)
B	11		1	Comma, for message to SYSLOG and SYSLST
C	12	RCHKVOL1	6	If tape, file serial number of output volume one at checkpoint time. If DASD - ******.
12	18		1	Comma, for message to SYSLOG and SYSLST
13	19	RCHKVOL2	6	If tape, file serial number of output volume two at checkpoint time. If DASD - ******.
19	25		1	Comma, for message to SYSLOG and SYSLST
1A	26	RCKSEGNM	8	Segment name of root segment in process

				at checkpoint time
22	34		4	Reserved for future use
26	38	RCHKRECL	2	Length of I/O area needed for GU call at restart time
28	40	RCHKPOSC	4	RBN of current record, if HD organization
2C	44	RCHKPTNR	1	Number of checkpoint records (1 or 2)
2D	45	RCHKEYLN	1	Key length of current segment, if HISAM
2E	46	RCKEYVAL	236	Segment sequence field value, if HISAM
11A	282	Reserved	12	Reserved
126	294	RCHKSEG	4	Total number of segments unloaded
12A	298	RCHKROOT	4	Total number of root segments unloaded
12E	302	RCHKREND	Var	Statistics table

Notes:
- Dummy checkpoint record does not contain statistics table.
- Checkpoint message written to SYSLOG and SYSLST consists of message prefix DLZ381I followed by bytes 1 - 34 of the checkpoint record.

CONTROL DATA SET

Macro DLZUCDS0 contains the DSECT defining format of a control list entry. One or more list entries may be contained in the control list. The control list may spread over one or more control list blocks.

Control Information and Identifier

Hex	Dec	Name	Ln	Description
0	0	LECELCNT	2	Number of 1600 byte records in control data set
2	2	LELSTLOC	2	Displacement to next entry
4	4	LECDSID	20	Identifier: ' CONTROL DATA SET '.
18	24	LEFLG4	1	Flag byte 4:
		FLAG Name	Hex Code	Meaning
		LESTAT	80	Statistics to be provided
		LESUMM	40	Give summary for message DLZ978I
19	25	Unnamed	1	**Reserved**
1A	26	LESRTSZE	2	Maximum work file record length used as SORT size parameter by prefix resolution utility (DLZURG10).

Data Base List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same level)
4	4	LENAME	8	DBD name.
C	12	LESLPTR	4	List entry sublist pointer (to list at next lower level)
10	16	LECRNO	2	Input control card number
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan method option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

Segment List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same level)
4	4	LENAME	8	Logical parent segment name.
C	12	LESLPTR	4	List entry sublist pointer (to list at next lower level)
10	16	LECRNO	2	Input control card number
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan method option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

Hex	Dec	Name	Ln	Description
14	20	LEPSDB	4	PSDB for segment entry
18	24	LELSDB	4	LSDB for segment entry

Secondary List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same lavel)

4	4	LENAME	8	Referenced data base name.
C	12	LEFDLP	2	Length of logical parent concatenated key.
E	14	LEFLG3	1	Flag byte 3:

Flag Name	Hex Code	Meaning
LET23	80	Use type 20/30 records.
LELCSQ	40	Use logical child sequence field.
LENLC	20	No logical child found for logical parent.
LELPCK	02	Use logical parent concatenated key.
LELPOA	01	Use logical parent old address.

F	15	Unnamed	1	**Reserved**
10	16	LEFDLC	2	Position of logical child pointers in prefix
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan mehtod option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

14	20	LELCSC	1	Logical child's segment code
15	21	LEFLG2	1	Flage byte 2:

Flag Name	Hex Code	Meaning
LECTR	80	Update counter
LELCF	40	Update logical child forward pointer
LELCL	20	Update logical child last pointer
LELP	10	Update logical parent pointer
LELTF	08	Update logical twin forward pointer
LELTB	04	Update logical twin backward pointer
LECUS	02	Counter used this logical child

17	23	Unnamed	2	**Reserved**

DATA BASE LOG RECORD

Note: If CICS journaling is used, see Section 3 under the heading "CICS Journal Logger" for additional information.

This record is used by modules DLZRDBL0, DLZRDBL1, DLZBACK0, DLZLOGP0, DLZURDB0, DLZUC150, and DLZUC350.

Hex	Dec	Name	Ln	Description
0	0	DLENGTH	2	Length of record

2	2	DSPACE	2	Zero
4	4	DLOGCODE	1	Log record ID X'50' = Data base log record X'51' = Old copy of a replaced segment
5	5	DLOGFLG1	1	Bits 0-3 Task ID 4-7 Count of FSE records present
6	6	DLOGFLG2	1	Bits 0=1 Index maintenance record 1-3=001 Physical replace =010 Physical delete =100 Physical insert =110 Logical delete =000 POINTER maintenance record =111 Counter Maintenance 4=1 Last record of a change group 5=0 ESDS data set =1 KSDS data set 6=0 HS organization =1 HD organization 7=1 New block call
7	7	DLOGFLG3	1	Bits 0=1 REPL call 1=1 DLET call 2=1 ISRT call 3&4=00 Modification by control region =01 Modification by message or batch message program =10 Modification by batch program 5=1 Record written by backout 6=1 First log record of a segment 7=1 Last log record of a segment
8	8	DIDLN	2	Length of DDATAID field
A	10	DOFFSET	2	Data offset from beginning of block
C	12	DDATALN	2	Length of DDATA field
E	14	DCCODE	2	DL/I completion code
10	16	DPGMNAME	8	PSB name
18	24	DDBDNAME	8	Data base name from the DMB
20	32	DDSID	1	File identification within the DMB
21	33	DDATE	3	Date - YYDDDF
24	36	DTIME	4	Time - HHMMSS0F
28	40	DSEQ	2	Sequence stamp
2A	42	DDATAID	Var	KSDS - KSDS prime key ESDS - Relative block number

POINTER maintenance record (DDATALN is set to H'4')

DDATA	4	New pointer value
	4	Old pointer value

LOGICAL DELETE record (DDATALN is set to H'2')

DDATA	2	Segment code and new delete byte
	2	Segment code and old delete byte

PHYSICAL INSERT record (DDATALN is set to segment length)

DDATA	V*	New segment data
DFSEOFF	2	Offset to FSE
DFSE	4	New FSE value If more than one FSE changes, DFSEOFF and DFSE are repeated for each additional one.

PHYSICAL DELETE record (DDATALN is set to segment length)

DDATA	V*	Old segment data
DFSEOFF	2	Offset to FSE
DFSE	4	New FSE value If more than one FSE changes, DFSEOFF and DFSE are repeated for each additional one.

PHYSICAL REPLACE record (DDATALN is set to segment length)

DDATA	V*	Old segment data - DLOGCODE = X'51'
		New segment data - DLOGCODE = X'50'

V* = varies with segment length

DCOUNTER The last four bytes of every log record contain the log record sequence number. Numbers are incremented by one. The sequence number of the first record is one.

DATA RECORD (INPUT)

This record is used as input to module DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	4	ESDS RBA identifier; unused if KSDS
4	4	DSIDIN	1	Character I if KSDS; O if ESDS
5	5	Unnamed	3	Reserved
8	8	Unnamed	Var	KSDS or ESDS physical record image. The first four bytes contain the VSAM relative byte address (RBA) of the next ESDS record containing overflow dependent segments for the root segment. The RBA is zero if no (more) ESDS records follow. The last byte of the data record contains a special physical code X'0'. If the data base contains only HISAM root segments and ACCESS=SHISAM, the physical code and RBA do not exist.

DATA RECORD (OUTPUT)

This output record is used by module DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	CONTOUT	4	ESDS RBA identifier; unused if KSDS
4	4	DSIDOUT	1	Character I if KSDS; O if ESDS
5	5	BLNKDOUT	1	(Not used)
6	6	DSRECLN	2	Record size + prefix length
8	8	DATA	Var	KSDS or ESDS physical record image. The first four bytes contain the VSAM relative byte address (RBA) of the next ESDS record containing overflow dependent segments for the root segment. The RBA is zero if no (more) ESDS records follow. The last byte of the data record contains a special physical code X'0'. If the data base contains only HISAM root segments and ACCESS=SHISAM, the physical code and RBA do not exist.

DATE/TIME TABLE

This record is used by modules DLZUCCT0 and DLZUC150.

Hex	Dec	Name	Ln	Description
0	0	TABFLAG1	1	Blank. Used as table delimiter
1	1	TABFLAG2	1	Contains a 0 or 1 to denote routing

for the data base in this table

2	2	TABFLAG3	1	Contains flags as follows: Name — Bit — Meaning TABF3N — 0 — Record to LOGOUT if 1 TABF3DT — 1 — Purge date specified
3	3	TABFLAG4	1	Reserved for future use
4	4	TABFLAG5	4	Reserved for future use
8	8	TABFLAG6	8	Contains date/time, if specified

DELETE WORK AREA

This record is used by module DLZDLD00.

Hex	Dec	Name	Ln	Description
0	0	DLTRSCID	7	Resource ID for PI queuing (must be first in WKA)
0	0	DLTRSCRB	4	RBA portion of resource ID
4	4	DLTCHN	8	Chain (prior content PSTWRKD1-2)
4	4	DLTPWAID	4	ID of current work area; DMB number, ACB number, and work area sequence number
4	4	DLTRSCID	3	DMB/ACB number part of resource ID
4	4	DLTDMBNO	2	DMB number
8	8	Unnamed	4	Prior scan exit address (PSTWRKD2)
C	12	DLTWANXT	4	Address of next WKA
10	16	DLTWASW	1	Switch

Flag Name	Hex Code	Meaning
DLTWSBEG	01	First work area in work space
DLTERFLG	02	R-O record flag required
DLTLRFLG	04	R-O record flag required due to LP LC counter update
DLTVRFLG	08	Verifies are required
DLTSCFLG	10	Pre-scan was done
DLTIMFLG	20	Index maintenance was done

10	16	DLTWAPRI	4	Address of prior WKA
14	20	DLTDMB	4	DMB address of this WKA
18	24	DLTSPSDB	4	Scan start PSDB
1C	28	DLTLPSDB	4	Scan end PSDB
20	32	DLTSLEV	2	Level at which scan started
22	34	DLTTEMPH	2	Half word temporary save area
24	36	DLTESECL	4	Secondary list address causing exit

28	40	DLTEDMB	4	Exit DMB address
2C	44	DLTEPSDB	4	Prior DMB's PSDB (exit point)
30	48	DLTERBN	4	Exit RBN
34	52	DLTLPKOF	2	Offset from DLTWA to concatenated key
36	54	DLTWASZ	2	Length of this work area
38	56	DLTMID	36	'Middle' of WKA
38	56	DLTPLT	4	Save area for prior L/C on twin chain
3C	60	DLTCLT	4	Save area for current L/C on twin chain
40	64	DLTNLT	4	Save area for next L/C on twin chain
44	68	DLTTEMP1	4	Working register save area (R6)
48	72	DLTTEMP2	4	Working register save area (R7)
4C	76	DLTTEMP3	4	Working register save area (R8)
50	80	DLTTEMP4	4	Working register save area (R9)
54	84	DLTLEVEL	8	Level information beginning
54	84	DLTRFLG	1	Flag byte

Flag Name	Hex Code	Meaning
DLTSVPP	01	Save segment and parents
DLTSVPC	02	Save segment and physical children
DLTLDO	03	Logical delete only
DLTKEYSW	04	Key stored for this level
DLTTEFLG	08	Temporary lock enqueue was done

54	84	DLTPSDB	4	Current PSDB this level
58	88	DLTRBN	4	RBN of segment this level
5C	92	DLTLEVLN	8	Length of level information entry
64	100	DLTMIDLN	36	Length of last half work area
88	136	DLTWALN	92	Length of basic delete work area

DELETE WORK SPACE PREFIX

This record is used by module DLZDLD00.

Hex	Dec	Name	Ln	Description
0	0	DLTBLKNM	4	Block number of buffer (from PSTBLKNM)
4	4	DLTBUFFA	4	Address of buffer prefix (from PSTBUFFA)
8	8	DLTNXTWS	4	Address of next work space

C	12	DLTPRIWS	4	Address of prior work space
10	16	DLTSIZWS	4	Usable size of this space
14	14		4	Reserved

DL/I CONTROL RECORD

This record is used by module DLZDLOC0.

Hex	Dec	Name	Ln	Description
0	0	RECDATCR	3	Creation date - YYDDDF
3	3	RECTIMCR	5	Creation time - HHMMSSTH0F
8	8	RECDATRE	3	Recovery date - YYDDDF
B	11	RECTIMRE	5	Recovery time - HHMMSSTH0F
10	16	RECDATER	3	Reserved
13	19	RECTIMER	5	Reserved
18	24	RECNXRBA	4	Not used
1C	28	RECDOS	3	DL/I component code (DLZ)
1E	31	RECVERS	3	Version and release level
22	34	RECPTF	2	PTF number
24	36	RECLKSDS	4	KSDS record length (HISAM only)
28	40	RECLESDS	4	ESDS record length
2C	44	RECORGAN	1	Data base organization

Name	Character	Meaning
RECHDAM	D	HDAM
RECHIDAM	I	HIDAM
RECHISAM	S	HISAM

2D	45		Var	Reserved to end of control interval

DUMP HEADER RECORD

This record is used by modules DLZUDMP0 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	DHSAMCTL	1	Reserved for future use
1	1	DUMPID	1	Character D
2	2	DCBNOOUT	2	Reserved for future use
4	4	DUMPDBDN	8	Name of the DMB devised from the Data Base Description (DBD)
C	12	DIDDNOUT	8	Contains the name of the key sequenced data set if this is dump of a KSDS

				data set
14	20	DDATEOUT	4	Julian date in packed decimal - 00YYDDDF
18	24	DTIMEOUT	4	Time in packed decimal - HHMMSS0F
1C	28	DODDNOUT	8	Contains the name of the entry sequenced data set if this is dump of an ESDS data set
24	36	DIBLKOUT	2	Contains KSDS control interval size if this is dump of KSDS data set
26	38	DIRECOUT	2	Contains KSDS record length if dump of KSDS data set
28	40	DOBLKOUT	2	Contains ESDS control interval size if this is dump of ESDS data set
2A	42	DORECOUT	2	Contains ESDS record length if dump of ESDS
2C	44	DKEYLEN	2	Contains KSDS key length if dump of KSDS
2E	46	DKEYPOS	2	Contains KSDS relative key positive if dump of KSDS
30	48	DDBDORG	1	Data set organization code

DUMP RECORD PREFIX

This record is used by module DLZUDMP0.

Hex	Dec	Name	Ln	Description
0	0	COUNTOUT	4	ESDS RBA identifier; record count if KSDS
4	4	DSIDOUT	1	Character I if KSDS; O if ESDS
5	5	Reserved	1	Reserved for future use
6	6	DSRECLN	2	Record size + prefix length
8	8	DATA	Var	Physical record image

FILE OPEN RECORD

This record is used by modules DLZRDBL0, DLZRDBL1, DLZLOGP0, DLZUC150, and DLZUC350.

Hex	Dec	Name	Ln	Description
0	0	DLENGTH	2	Length of record
2	2	DSPACE1	2	Binary zero
4	4	DLOGCODE	1	Record type code - X'2F'
5	5	DLOGFLG1	2	Data set organization X'00' = ESDS X'04' = KSDS

7	7	DSPACE2	9	Binary zero
10	16	DPGMNAME	8	Data set filename (ACB)
18	24	DDBDNAME	8	DMB name
20	32	DDSID	1	DSGACBNO (2 if HISAM ESDS; otherwise 1)
21	33	DDATE	3	Binary zero
24	36	DTIME	4	Binary zero
28	40	DCOUNT2F	4	Log record sequence number

HEADER RECORD (INPUT)

This record is used as input for module DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	1	X'FF' header/statistic record identifier
1	1	IDIN	1	Character R
2	2	RECLNOUT	2	Size of output record, including prefix
4	4	DBDNAME	8	Name of the DMB derived from the Data Base Description (DBD)
C	12	DDNAMEI	8	Name of key sequenced data set (KSDS)
14	20	Unnamed	4	Julian date in packed decimal-00YYDDDF
18	24	Unnamed	4	Time in packed decimal-HHMMSSOF
1C	28	DDNAMEO	8	Name of entry sequenced data set (ESDS)
24	36	BLKSIZEI	2	KSDS record length * number of records/control interval
26	38	LRECLI	2	KSDS record length
28	40	BLKSIZEO	2	ESDS record length * number of records/control interval
2A	42	LRECLO	2	ESDS record length
2C	44	Unnamed	1	0; (Not used)
2D	45	KEYLENGI	1	KSDS key length
2E	46	KEYPOSI	2	KSDS relative key position

HEADER RECORD (OUTPUT)

This record is used by module DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	HSAMCTRL	1	X'FF' header/statistic record identifier

1	1	IDOUT	1	Character R
2	2	RECLNOUT	2	Size of output record, including prefix
4	4	DBDOUT	8	Name of the DMB derived from the Data Base Description (DBD)
C	12	IDDNOUT	8	Name of key sequenced data set (KSDS)
14	20	DATEOUT	4	Julian date in packed decimal-00YYDDDF
18	24	TIMEOUT	4	Time in packed decimal-HHMMSSOF
1C	28	ODDNOUT	8	Name of entry sequenced data set (ESDS)
24	36	IBLKSOUT	2	KSDS record length * number of records/control interval
26	38	ILRECOUT	2	KSDS record length
28	40	OBLKSOUT	2	ESDS record length * number of records/control interval
2A	42	OLRECOUT	2	ESDS record length
2C	44	IKEYLENG	2	KSDS key length
2E	46	IKEYPOS	2	KSDS relative key position

INDEX MAINTENANCE WORK AREA

This record is used by module DLZDMXT0.

Hex	Dec	Name	Ln	Description
0	0	XSAVDSGA	4	Save location for caller's DSG
4	4	XSAVPCB	4	Save location for caller's PCB
8	8	XSAVUSER	4	Save location for caller's I/O area
C	12	XSAVIQPR	4	For caller's call list address
10	16	XPHYSPP	4	Save location for physical parent pointer.
14	20	XWORKPCB	4	Save location for XMAINTs PCB
18	24	XWORKSAA	4	Address of SSA built by DLZDXMT0
1C	28	XWORKFNC	4	XMAINTs function code for call
20	32	XDPSDBAD	4	Address of PSDB of indexed segment
24	36	XDSECLST	4	Secondary list of indexed segment
28	40	XDRID	8	Indexed segment ID for enqueue
28	40	XDRBAPTR	4	RBA of indexed segment
2C	44	XDDMBACB	4	DMB and ACB numbers of indexed segment

30	48	XNRID	8	Indexing segment ID for enqueue
30	48	XNRBAPTR	4	RBA of indexing segment
34	52	XNDMBACB	4	DMB and ACB numbers of indexing segment
38	56	XSPSDBAD	4	PSDB of index source segment
3C	60	XSSECLST	4	Secondary list of index source segment
40	64	XSRBAPTR	4	RBA of index source segment
44	68	XNPSDBAD	4	Address of PSDB of indexing segment
48	72	XDSDBAD	4	Index target segment SDB address
4C	76	XSSDBAD	4	Index source segment SDB address
50	80	XPROT	2	Length of protected data
52	82	XRPREFIX	2	Record prefix length
54	84	XSPREFIX	2	Segment prefix length
56	86	XNSEGLEN	2	Length of indexing segment
58	88	XNKEYLEN	2	Sequence field length of index pointer segment
5C	92	STACK1	4	Return address for first level subroutine
60	96	STACK2	4	Return address for second level subroutine
64	100	STACK3	4	Return address for third level subroutine
68	104	XSAVSTC	1	Save status code
69	105		1	*Reserved*
6A	106	XCALLFUN	1	Call attributes byte

Flag Name	Hex Code	Meaning
ISLOAD	80	Load mode
ISASRT	40	ASRT call
ISDLET	20	DLET call
ISISRT	10	ISRT call
ISREPL	08	Function is replace
ISUNLD	02	UNLD call

6B	107	XTSWIT1	1	Temporary switch

Flag Name	Hex Code	Meaning
XNOSUPR	80	No suppression for this index
XOLDSUPR	40	Old segment was suppressed
XPTRONLY	20	PTR to XDS only, no

		CONCAT key
XISPRIM	10	A primary index was found
XNULLFLD	01	Null value suppression
XEXITRT	02	Exit routine for suppression
XDATACHN	04	XNS changed in a replace call

6E	110	XWORKPUT	2	Begin of record for load

(The rest of this record starts on a fullword boundary)

70	112	XWORKUSR	0	XMAINTs I/O area for call
70	112	XWORKDUM	2	Reserved
72	114	XWORKSEG	0	Start of segment
72	114	XWORKCD	1	Segment code

Flag Name	Hex Code	Meaning
XNSEGC01	01	Segment code of indexing segement

73	115	XWORKDEL	1	Delete byte in indexing segment
74	116	XWORKPTR	4	Pointer in indexing segment
78	120	XWORKKEY	VAR	Area for key in indexing segment

(The SSA for the XMAINT call to the analyzer is created behind the key)

LIST CONTROL BLOCK

This record is used by module DLZUSCH0.

Hex	Dec	Name	Ln	Description
1C	28	ENTLNGTH	2	The length, in bytes, of each entry in the list
1E	30	COMPLOC	2	The offset from the beginning of each entry to the key field
20	32	COMPLNG	2	The length of the key field
22	34	NUMENT	2	The current number of entries in the list
24	36	CHAINLOC	4	The location of the first of a chain of core blocks containing sorted list entries
28	40	CHBACK	4	The location of the last block in the chain
2C	44	ENTBLKSZ	4	The size of each core block used for list entries (includes the chaining fields).

This value is calculated as follows: ENTBLKSZ = 16*ENTLNGTH+8

30	48	LASTLO, LASTHI,	12	Work areas used by INSRCH and LOCSRCH

LASTMD
ENTLOC

OUTPUT RECORD CONTAINING SEGMENT PREFIX

This DSECT (IOAREA) defines the format of the unloaded data base records used by the HD reorganization unload/reload utilities.

Hex	Dec	Name	Ln	Description
0	0	RGUSEGLV	1	Segment code for this segment
1	1	RGUHSDF	1	HSAM delete flag; always X'80' to denote HD Reorganization Unload Utility
2	2	RGUHDRLN	2	Length of header portion of record
4	4	RGUSEGLN	2	Length of data portion of reccrd
6	6	RGUSEGNM	8	Segment name
E	14	RGUSEGDF	1	Delete flag of segment
F	15	RGUPFCTR	4	Counter field of prefix
13	19	IOTWFOR	4	Logical twin forward pointer
17	23	IOTWBACK	4	Logical twin backward pointer
1B	27	IOPAR	4	Logical parent pointer
1F	31	IOOLD	4	Old location of record
23	35	IOSEG	Var	Variable-length data field

OUTPUT TABLE RECORD

This DSECT (DLZUSTAT) defines the format of the statistics table within the unloaded data base for HD reorganization unload/reload utilities.

Hex	Dec	Name	Ln	Description
0	0	RGUSEGLV	1	Always X'00'
1	1	RGUHSDF	1	X'80' for first table reccrd and checkpoint table record X'90' for last table record
2	2	RGUHDRLN	2	Length
4	4	RGUSEGLN	Var	A table containing one entry for each segment type.

Field Description of RGUSEGLN

Hex	Dec	Name	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	SMIMCHLD	4	Maximum immediate children

C	12	SAIMCHLD	4	Average immediate children
10	16	WKIMCHLD	4	Working entry for above
14	20	SMSBCHLD	4	Maximum subordinate children
18	24	SASBCHLD	4	Average subordinate children
1C	28	WKSBCHLD	4	Working entry for above
20	32	TSEGTYPE	4	Total segments for this type
24	36	SEGLEVEL	1	Segment level
25	37	SEGPHYCD	1	Segment physical code
26	38	TABLEND	2	Table end indicator (X'80')
26	38	TSEGLEN	2	Segment length including prefix
28	40	STATABSZ		Length of each table entry

SHORT SEGMENT TABLE

This record is used by module DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	SEGMDSN0	1	Data set number (not used by DLZURUL0)
1	1	SEGMCODE	1	Physical segment code
2	2	PARSEGCD	1	Physical code of this segment's parent
3	3	SEGMLEVL	1	Segment hierarchical level
4	4	Unnamed	2	Number of logical children and fields (not used by DLZURUL0)
6	6	SEGMLENG	2	Segment length, including prefix

SORTED LIST BLOCK

This record is used by module DLZUSCH0.

Hex	Dec	Name	Ln	Description
0	0	ENCNT	1	The count minus one of the current number of entries in this block (currently, the maximum value for count is 16)
1	1	CHAIN	3	The location of the next sorted list block in the chain. In the last block, this field contains binary zeros.
4	4	BKCHAIN	4	The location of the preceding sorted list block in the chain. In the first block on the chain, this field

				contains the location of the CHAINLOC field in the list control block.
8	8	ENTRIES	Var	Up to 16 full entries in sorted order.

Note: All blocks are the same size regardless of the number of entries contained. Unused space at the end of a block is not zeroed.

SSA FOR GU CALL BY KEY

This record is used by module DLZURGU0.

Hex	Dec	Name	Ln	Description
0	0	KEYSEGNM	8	Name of segment to be retrieved
8	8	KEYCODE	2	'*C' - command code
A	10	KLEFTPAR	1	'(' - left parenthesis
B	11	KEY	1-236	key to be retrieved
-	-	KRITEPAR	2	')' - right parenthesis

SSA FOR GU CALL BY RBA

This record is used by module DLZURGU0.

Hex	Dec	Name	Ln	Description
0	0	RBASEGNM	8	Name of segment to be retrieved
8	8	RBACODE	2	'*T' - command code
A	10	RLEFTPAR	1	'(' - left parenthesis
B	11	RBA	4	RBA to be retrieved
F	15	RRITEPAR	1	')' - right parenthesis

SSA FOR THE XMAINT CALL TO THE ANALYZER

This record is used by module DLZDXMT0.

Hex	Dec	Name	Ln	Description
0	0	XSEGNAME	8	Name of index pointer segment
8	8	XCOMMCOD	2	'*X' - command code
A	10	XLEFTPAR	1	'(' - left parenthesis
B	11	XKEYVALU	VAR	Key value followed by right parenthesis ')'

STATISTICS RECORD

This record is used by modules DLZURUL0 and DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	1	X'FF' header/statistics record identifier
1	1	Unnamed	1	Character S
2	2	Unnamed	2	Number of segment types in data set group (16 bytes per segment type)
4	4	Unnamed	8	Name of the DMB derived from the DBD
C	12	Unnamed	8	KSDS filename
14	20	Unnamed	8	ESDS filename
1C	28	Unnamed	Var	A 16-byte table entry for each segment type in the data base

DESCRIPTION OF VARIABLE LENGTH LAST FIELD OF STATISTICS RECORD WHEN USED AS OUTPUT FOR DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	TSEGTYPE	4	Total number of segments unloaded
C	12	SEGLEV	1	Segment level
D	13	SEGPCD	1	Segment physical code
E	14	TSEGLN	2	Segment length, including prefix

DESCRIPTION OF VARIABLE LENGTH LAST FIELD OF STATISTICS RECORD WHEN USED AS INPUT FOR DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	TOTSEG	4	Total number of segments unloaded
C	12	SEGLEV	1	Segment level
D	13	SEGPCD	1	Segment physical code
E	14	SEGLN	2	Segment length, including prefix

WORK FILE 1

This record is used as the input file for DLZURG10.

Hex	Dec	Name	Ln	Description
0	0	ALENGTH	2	Length of work file 1 record
2	2	ASPACE	2	Two bytes of zeros
4	4	ALTYPE	1	Type of input record

Flag Name	Hex Code	Meaning

		ATYPE00	00	Type 00 record
		ATYPE01	01	Type 01 record
		ATYPE02	02	Type 02 record
		ATYPE03	03	Type 03 record
		ATYPE10	10	Type 10 record
		ATYPE20	20	Type 20 record
		ATYPE30	30	Type 30 record
		ATYPE40	40	Type 40 record

DL/I Record Type	Use
00	Generated once for each use of a segment as a logical parent
10	Generated once for each use of a segment as a logical child.
20	Generated when a segment used as a logical child contains logical twin forward pointers and when the logical twin chain cannot be resolved by using the logical child's sequence field.
30	Generated when a segment used as a logical child contains logical twin backward pointers and when the logical twin chain cannot be resolved by using the logical child's sequence field.
40	Generated once for each time a segment is indexed

5 5 ALFLAG1 1 Flag 1

Flag Name	Hex Code	Meaning
AL1LOAD	80	Set to 1 if ISRT; set to 0 if ASRT
AL1SEQ	40	Set to 1 if sequence field is present
AL1SCAN	20	Set to 1 if record produced by scan program (DLZURGS0)
AL1LPCK	10	Set to 1 if logical parent concatenated key is prsent
AL1SQUN	08	Sequence field is unique
AL1SEQA	04	Set to 1 if root sequence field is present

				AL1CONST 02 Constant present in key
				AL1SYMB 01 For type 40 record; pointer is symbolic
				AL1T23 01 Set to 1 if logical twin pointers are to be resolved by type 20 and 30 records
6	6	ALFLAG2	1	Executable length of sequence field, if present
7	7	ALFLAG3	1	Executable length of indexed field, if present, or executable length of logical parent concatenated key, if present
8	8	ALEVTTR	4	Value of LEVTTR after BYLCT
C	12	ALPDBNAM	8	Data base of logical parent
14	20	ALPSEQ	1	Segment code of logical parent
15	21	ALPCKEY	4	Logical parent's concatenated key
15	21	ALPOADDR	4	Logical parent's old address
19	25	ALCDBNAM	8	Data base of logical child
21	33	ALCSEG	1	Segment code of logical child
FOR TYPE 00 AND 01 RECORDS				
22	34	ALCFL	4	Old value of logical child first or logical child last pointer
26	38	ALT0001	1	X'00' or X'01'
27	39	ALPLSGOF	2	Value of logical parent's LEVSEGOF after BYLCT
29	41	ALCCTR	4	Old value of counter field
2D	45	ALPDCB	1	DCB NUMBER FOR LP
(TYPE 01 RECORD ENDS HERE)				
2E	46	ALPSEQA	1	Sequence field and length for root of segment
FOR TYPE 02 RECORDS				
22	34	ALCOAD	4	Logical child old address
26	38	ALT02	1	X'02'
FOR TYPE 10, 20, AND 30 RECORDS				
22	34	ALFIL	1	X'FF'

23	35	ALCSEQ	4	Logical child sequence field
23	35	ALCM	4	If LC has LT pointers and a non-unique sequence field and is being reloaded, ALCM contains the following: For Type 10 - LC's old address For Type 20 - LC's old LT forward pointer For Type 30 - LC's old LT backward pointer Otherwise, ALCM contains the value of LEVSEGOF, with high order bit set to one
27	39	ALT123	1	X'10', or X'20', or X'30'
28	40	ALCDCB	1	DCB number for LC
29	41	ALCSEQA	1	Sequence field and length for root of segment
FOR TYPE 40 RECORDS				
8	8	AILCOA	4	Logical child old address
C	12	AIDBNAM	8	Index data base name
14	20	AIFLDVAL	1	Indexed field value (variable length)
14	20	AISC	1	Index segment's segment code
15	21	AISEQ	1	Index segment's sequence code (if second level and present)
15	21	AISEGN	8	Index segment's name (For level 2 index segments)
15	21	AIFLDN	8	Indexed field name (For level 1 index segments)
1D	29	AISDBN	8	Indexed segment's data base name
25	37	AISSC	1	Indexed segment's segment code
26	38	AILCNA	4	Logical child new address
2A	42	AIDATA	1	Indexed segment data (for source fields)
FOR TYPE 40 RECORD USED AS SSA AND I/O AREA				
9	9	AISSFN	8	Index segment name or field name
11	17	AISSAID	3	SSA ID and command code
14	20	AISFLDV	1	Indexed segment's indexed field value (variable length)
14	20	AISSEQ	1	Index segment's sequence field value (variable length)
21	33	AXSC	1	Segment code of indexed segment

22	34	AXDDIR	3	DDIR address of indexed data base
25	37	AXLCNA	4	Logical child new address
29	41	AXDATA	1	Index source data

WORK FILE 3

This record is the output file from DLZURG10 and is used as the input file for DLZURGP0.

Hex	Dec	Name	Ln	Description
0	0	CLENGTH	2	Length of work file record
2	2	CSPACE	2	Zeros
4	4	CTYPE	1	Work file record type

Flag Name	Hex Code	Meaning
CTYPE0	00	Type 00 record
CTYPE01	01	Type 01 record
CTYPE1	10	Type 10 record
CTYPE2	20	Type 20 record
CTYPE3	30	Type 30 record
CTYPE4	40	Type 40 record

5	5	CFLAG1	1	Origin of record

Flag Name	Hex Code	Meaning
CF1LOAD	80	Flag on-initial load; Flag off-reorganization
CF1SCAN	20	Record produced by scan
CFILPCK	10	Logical parent concatenated key if present
CF1SEQA	04	Set to 1 if root sequence field present
CF1T0F	02	Set to 1 if matching type 10 record found
CF1T23	01	Set to 1 if logical twin pointer is to be resolved by type 20 and 30 records

FIELDS IN TYPE 0 RECORD

6	6	CLCDBN0	8	Logical child data base name
E	14	CLCSEGN0	1	Logical child segment code
F	15	CLPSEGN0	1	Logical parent segment code
10	16	CLCFRST	4	Logical child first pointer
14	20	CLCDLST	4	Logical child last counter
18	24	CLCDCNT	4	Logical child delta counter
1C	28	CLPDBN0	8	Logical parent data base name

FIELDS IN TYPE 1 RECORD

6	6	CLPDBN1	8	Logical parent data base name
E	14	CLPSEGN1	1	Logical parent segment code
F	15	CLCSEGN1	1	Logical child segment code
10	16	CLTFWD	4	Logical twin forward pointer
14	20	CLTBKWD	4	Logical twin backward pointer
18	24	CLPNWAD1	4	Logical parent new address
1C	28	CLCDBN1	8	Logical child data base name
24	36	CDCB	1	DCB number
25	37	CFIL	1	
26	38	CLEVTTR	4	Contents of LEVTTR after BYLCT
2A	42	CLEVSGOF	2	Contents of LEVSEGOF after BYLCT (high order bit of CLEVSGOF is set to 1 if segment is not in HD)
2C	44	CLCCNT	4	Old value of counter field
30	48	CLSEQ	1	Root sequence field

SECTION 6: DIAGNOSTIC AIDS

This section contains two tables that cross-reference DL/I messages and DL/I status codes with the module(s) that originate them.

Additional diagnostic information can be found in the DL/I DOS/VS Diagnostic Guide, SH24-5002.

SYSTEM MESSAGE/MODULE CROSS REFERENCE

This table cross-references message numbers (in numeric order) with the module(s) that can cause that message to be issued. In addition, if the message is described in the module HIPO diagram in Section 2, the HIPO figure number is also shown. The modules are described in Section 3 of this publication. The messages are described in Chapter 1 of "DL/I DOS/VS Messages and Codes".

Message Number	Module	Figure Number
DLZ000I	DLZRRG00	
DLZ001I	DLZBNUC0	2-4.2
DLZ002I	DLZBNUC0	2-4.2
DLZ003I	DLZDDLE0	
DLZ004I	DLZDBH02	
	DLZRDBL0	2-16.7
DLZ005I	DLZDBH02	
DLZ006I	DLZOLI00	2-5.4
DLZ007I	DLZDSEH0	2-38
	DLZDXMT0	
DLZ008I	DLZRRC00	2-3.9
DLZ009I	DLZRRC00	2-3.8
DLZ010A	DLZRRC00	
	DLZMPI00	2-21.1
DLZ011I	DLZRRC00	2-3.2
DLZ012I	DLZMPI00	2-21.1
	DLZRRC00	2-3.4, 2-3.7, 2-3.9
DLZ013I	DLZOLI00	2-5.3
DLZ014A	DLZRRC00	
	DLZMPI00	2-21.1
DLZ015I	DLZRRC00	2-3.3, 2-3.9
DLZ016I	DLZDLOC0	
DLZ017I	DLZRRC00	2-3.7
DLZ018I	DLZRRC00	2-3.7
DLZ019I	DLZRRC00	2-3.9
DLZ020I	DLZDLOC0	2-14.1
	DLZRDBL0	2-16.1
DLZ021I	DLZDLOC0	
	DLZRDBL0	2-16.6
DLZ022I	DLZDLOC0	
DLZ023I	DLZDLOC0	2-14.1
DLZ024I	DLZDLOC0	
DLZ025I	DLZDLOC0	2-14.1
DLZ026I	DLZRRC00	2-3.8
DLZ027I	DLZDLOC0	2-14.1
DLZ028I	DLZDLOC0	2-14.1
DLZ029I	DLZOLI00	2-5.3, 2-5.9
DLZ030I	DLZOLI00	2-5.8
DLZ031I	DLZOLI00	2-5.1
DLZ032A	DLZOLI00	2-5.4
	DLZRDBL1	
DLZ033I	DLZISC00	2-6.15
DLZ034I	DLZOLI00	2-5.1
DLZ037I	DLZEIPB0	2-45.4, 2-46.21
	DLZEIPB0	
	DLZEIP00	
DLZ038I	DLZEIPB0	2-45.4, 2-45.6
	DLZEIPB1	
	DLZMPI00	

Message Number	Module	Figure Number
	DLZRRC00	2-3.4
DLZ039I	DLZOLI00	
DLZ040A	DLZOLI00	
DLZ041I	DLZOLI00	
DLZ042I	DLZOLI00	2-5.2
DLZ043I	DLZOLI00	2-5.2
DLZ044I	DLZOLI00	2-5.2
DLZ045I	DLZOLI00	2-5.3
DLZ046I	DLZOLI00	2-5.3
DLZ047I	DLZOLI00	2-5.3
DLZ048I	DLZOLI00	2-5.3
DLZ049I	DLZOLI00	2-5.3
DLZ050I	DLZOLI00	2-5.1
DLZ051I	DLZOLI00	2-5.1
DLZ052I	DLZOLI00	2-5.5
DLZ053I	DLZOLI00	2-5.5
DLZ054I	DLZOLI00	2-5.5
DLZ055I	DLZOLI00	2-5.4
DLZ056I	DLZOLI00	2-5.4
DLZ057I	DLZOLI00	2-5.5
DLZ058I	DLZOLI00	2-5.6, 2-5.7
	DLZRRC00	
DLZ060I	DLZOLI00	2-5.9
DLZ061A	DLZOLI00	2-5.9
DLZ062I	DLZODP	2-6.10
DLZ063I	DLZODP	2-6.2
DLZ064I	DLZOLI00	2-5.1
DLZ065I	DLZODP	2-6.2
DLZ066I	DLZODP	2-6.2
DLZ067I	DLZODP	2-6.2
DLZ068I	DLZODP	2-6.2
DLZ069I	DLZODP	2-6.2
DLZ070I	DLZODP	2-6.2
DLZ071I	DLZOLI00	2-5.2
DLZ072I	DLZOLI00	2-5.3
DLZ073I	DLZOLI00	2-5.3
DLZ074I	DLZOLI00	2-5.3
DLZ075I	DLZRRC00	2-3.9
DLZ076A	DLZRDBL0	2-16.7
DLZ077I	DLZRDBL0	2-16.1, 2-16.7
DLZ078I	DLZRRC00	2-3.9
DLZ079I	DLZRDBL0	2-16.7
DLZ080I	DLZMSTP0	2-22
DLZ081I	DLZMPI00	2-21.1
DLZ082I	DLZBPC00	2-20.1, 2-20.5
	DLZMPC00	2-19.2, 2-19.4, 2-19.5, 2-19.7, 2-19.8
	DLZMPI00	2-21.1, 2-21.3
DLZ083I	DLZMSTR0	2-18
DLZ084I	DLZBPC00	2-20.2, 2-24.4
	DLZMPC00	2-19.4
	DLZMPI00	2-21.1, 2-21.3
DLZ085I	DLZMPI00	2-21.1
DLZ086I	DLZMPC00	2-19.7
DLZ087A	DLZMPI00	2-21.1
DLZ088I	DLZMPI00	2-19.1
DLZ089I	DLZMPI00	2-21.1
DLZ090I	DLZMPI00	2-21.2
DLZ091I	DLZMPI00	2-21.3
DLZ092I	DLZMPI00	2-21.3
DLZ093I	DLZMPC00	2-19.2

Message Number	Module	Figure Number
DLZ094I	DLZMPC00	2-19.1, 2-19.8
DLZ095I	DLZMPI00	2-21.1
DLZ096I	DLZMPI00	2-21.5
DLZ097I	DLZMSTR0	2-18
DLZ098I	DLZMPI00	2-21.3
DLZ099I	DLZMPI00	2-21.1
DLZ100I	DLZMPI00	2-21.3
DLZ101I	DLZMSTR0	2-18
DLZ102I	DLZMPI00	2-21.3
DLZ103I	DLZBPC00	2-20.5
DLZ104I	DLZMPC00	2-19.9
	DLZBPC00	2-20.6
DLZ105I	DLZRRC00	
	DLZBNUC0	2-4.1
	DLZMPI00	
	DLZISC00	2-6.15
DLZ106I	DLZQUEF0	
DLZ108I	DLZQUEF0	
DLZ120I	DLZTRACE	
DLZ260I	DLZBNUC0	2-4.1
	DLZODP	2-6.6, 2-6.10
DLZ261I	DLZBNUC0	2-4.1
	DLZODP	2-6.6, 2-6.10
DLZ262I	DLZRRC00	2-3.8
	DLZOLI00	2-5.9
DLZ263I	DLZRRC00	2-3.7
DLZ264I	DLZRDBL1	
DLZ266I	DLZRRC00	2-3.7
	DLZOLI00	2-5.3
DLZ267I	DLZQUEF0	2-23
DLZ268I	DLZDDLE0	
DLZ280I	DLZSTTL	2-42
DLZ281I	DLZSTTL	2-42
DLZ282I	DLZSTTL	2-42
DLZ301I	DLZUDMP0	
	DLZURDB0	
	DLZURGL0	2-32
	DLZURGU0	2-31
	DLZURRL0	
	DLZUC350	
	DLZURUL0	
DLZ302I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZURCC0	2-27.1
DLZ303I	DLZUDMP0	2-25
	DLZURUL0	2-29
DLZ304I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURCC0	2-27.1
DLZ305I	DLZUDMP0	
	DLZURDB0	
	DLZURUL0	
DLZ306I	DLZURUL0	
	DLZURDB0	
	DLZUDMP0	
DLZ307I	DLZURUL0	2-29
	DLZUDMP0	2-25
	DLZURRL0	2-30
	DLZURCC0	2-27.1

Message Number	Module	Figure Number
DLZ308I	DLZUDMP0	2-25
	DLZURUL0	2-29
DLZ309I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZRDBL0	
DLZ310I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZRDBL0	
	DLZURCC0	2-27.1
DLZ311I	DLZURRL0	
	DLZURGU0	2-31
	DLZURGL0	2-32
	DLZLOGP0	
	DLZTPRT0	
DLZ312I	DLZURDB0	
DLZ313I	DLZURDB0	
DLZ314I	DLZURDB0	
DLZ315I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ316I	DLZURDB0	
	DLZUDMP0	
DLZ317I	DLZURDB0	
DLZ318A	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ319I	DLZURUL0	
	DLZURGU0	
	DLZUDMP0	
	DLZURGL0	2-32
	DLZURDB0	
	DLZURRL0	
DLZ320I	DLZURUL0	
	DLZURGU0	
	DLZUDMP0	
DLZ321I	DLZURUL0	
	DLZUDMP0	
	DLZURRL0	
DLZ322I	DLZURDB0	
DLZ323I	DLZURDB0	
DLZ324I	DLZURDB0	
DLZ325I	DLZURDB0	
DLZ326I	DLZURDB0	
DLZ327I	DLZURDB0	
DLZ328I	DLZURDB0	
DLZ329I	DLZURGU0	2-31
DLZ330I	DLZURDB0	
DLZ331I	DLZURDB0	
DLZ332I	DLZURDB0	
DLZ333I	DLZURDB0	
DLZ334I	DLZURDB0	
DLZ335I	DLZURDB0	
DLZ336I	DLZURDB0	
DLZ337I	DLZURDB0	
DLZ338I	DLZURDB0	
DLZ339I	DLZURDB0	
DLZ340I	DLZURDB0	
DLZ341I	DLZURDB0	
DLZ342I	DLZBACK0	
	DLZLPCC0	

Message Number	Module	Figure Number
	DLZURCC0	2-27.1
	DLZUCCT0	
DLZ343I	DLZURDB0	
DLZ344I	DLZURRL0	2-30
	DLZURUL0	
DLZ345I	DLZURGU0	2-31
	DLZUDMP0	
	DLZURUL0	
DLZ346I	DLZURGU0	
DLZ347I	DLZURGU0	2-31
DLZ348I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ349I	DLZURGU0	2-31
DLZ350I	DLZUDMP0	
DLZ351I	DLZURGL0	2-32
DLZ352I	DLZURGU0	2-31
DLZ353I	DLZURRL0	
DLZ354I	DLZURGL0	2-32
DLZ355I	DLZURGL0	2-32
DLZ356I	DLZURRL0	
DLZ357I	DLZURUL0	
	DLZUDMP0	
DLZ358I	DLZURUL0	
DLZ359I	DLZURGU0	2-31
DLZ360I	DLZUCCT0	
DLZ361I	DLZUCCT0	
DLZ362I	DLZUCCT0	
DLZ363I	DLZUCCT0	
DLZ364I	DLZUCCT0	
DLZ365I	DLZUCCT0	
DLZ366I	DLZUCCT0	
DLZ367I	DLZUCCT0	
DLZ368I	DLZURGL0	2-31
	DLZURGU0	2-32
DLZ369I	DLZUCCT0	
	DLZUC150	
DLZ370I	DLZURGL0	2-32
DLZ371I	DLZUC150	
DLZ372I	DLZURCC0	2-27.1
	DLZLPCC0	
	DLZBACK0	
	DLZUCCT0	
DLZ373I	DLZUC350	
DLZ374I	DLZUC150	
	DLZUC350	
DLZ375I	DLZUC350	
DLZ376I	DLZURGL0	2-32
DLZ377I	DLZURGU0	
DLZ378I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ379I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ380I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ381I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ382I	DLZURUL0	
DLZ383I	DLZURUL0	
DLZ384I	DLZUCUM0	
DLZ385I	DLZUCUM0	

Message Number	Module	Figure Number
DLZ386I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ387I	DLZURGL0	
DLZ389I	DLZURGL0	2-32
	DLZURRL0	
DLZ390I	DLZUC150	
	DLZLOGP0	
DLZ391I	DLZUDMP0	
	DLZURDB0	
	DLZURUL0	
	DLZURRL0	
	DLZBACK0	
	DLZLOGP0	
	DLZUC150	
	DLZUC350	
	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
	DLZURGP0	
	DLZUCCT0	
	DLZTPRT0	
DLZ392I	DLZURUL0	
	DLZURGU0	2-31
	DLZURRL0	
DLZ393I	DLZURRL0	
DLZ394I	DLZURRL0	
	DLZURDB0	
DLZ395I	DLZBACK0	
DLZ396I	DLZRDBC0	
DLZ397I	DLZRDBC0	
DLZ398I	DLZRDBC0	
DLZ399I	DLZRDBC0	
DLZ400I	DLZURGU0	2-31
DLZ401I	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	
DLZ402I	DLZBACK0	
	DLZURDB0	
	DLZUC150	
DLZ404I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ405I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ406I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ407I	DLZLPCC0	
	DLZTPRT0	
	DLZURCC0	
DLZ408I	DLZLPCC0	
DLZ409I	DLZLPCC0	
DLZ410I	DLZLPCC0	
DLZ411I	DLZLPCC0	
DLZ412I	DLZLPCC0	
DLZ413I	DLZLPCC0	

Message Number	Module	Figure Number
DLZ414I	DLZLPCC0	
	DLZURCC0	
	DLZTPRT0	
DLZ415I	DLZLPCC0	
	DLZURCC0	
DLZ416I	DLZLOGP0	2-39.1
DLZ417I	DLZLOGP0	
DLZ418I	DLZLOGP0	
DLZ419I	DLZLOGP0	
DLZ420I	DLZLOGP0	
DLZ421I	DLZLOGP0	
DLZ422I	DLZLOGP0	
DLZ423I	DLZLOGP0	
DLZ424I	DLZLOGP0	
DLZ425I	DLZLOGP0	
DLZ426I	DLZLPCC0	
DLZ427I	DLZLOGP0	
DLZ428I	DLZLOGP0	
DLZ429I	DLZLOGP0	
DLZ430I	DLZLPCC0	
DLZ431I	DLZLPCC0	
DLZ432I	DLZLPCC0	
DLZ433I	DLZLPCC0	
DLZ434I	DLZLPCC0	
DLZ440I	DLZTPRT0	
DLZ441I	DLZTPRT0	
DLZ442I	DLZTPRT0	
DLZ443I	DLZTPRT0	
DLZ444I	DLZTPRT0	
DLZ445I	DLZTPRT0	
DLZ446I	DLZTPRT0	
DLZ447I	DLZTPRT0	
DLZ448I	DLZTPRT0	
DLZ449I	DLZTPRT0	
DLZ450I	DLZTPRT0	
DLZ451I	DLZTPRT0	
DLZ452I	DLZTPRT0	
DLZ453I	DLZTPRT0	
DLZ454I	DLZTPRT0	
DLZ476I	DLZDLA00	
	DLZODP	2-6.16
DLZ570I	DLZDLBL3	
	DLZUABC0	2-33.12
DLZ571I	DLZUACB0	2-33
DLZ572I	DLZDLBL0	
	DLZDLBL1	
DLZ573I	DLZDLBL0	
	DLZDLBL1	
DLZ583I	DLZUACB0	
DLZ584I	DLZUACB0	
DLZ585I	DLZUACB0	
DLZ587I	DLZUACB0	2-33
DLZ588I	DLZUACB0	2-33
DLZ589I	DLZUACB0	2-33
DLZ600I	DLZPRCT2	
DLZ602I	DLZPRPAR	2-43.8
DLZ603I	DLZPRPAR	2-43.8
DLZ604I	DLZPRPAR	2-43.8
	DLZPRDBD	2-43.4
DLZ605I	DLZPRPAR	2-43.8

Message Number	Module	Figure Number
DLZ606I	DLZPRPAR	2-43.8
DLZ608I	DLZPRPAR	2-43.8
DLZ609I	DLZPRPAR	2-43.8
DLZ610I	DLZPRPAR	2-43.8
DLZ611I	DLZPRPAR	2-43.8
DLZ612I	DLZPRPAR	2-43.8
	DLZPRDBD	2-43.4
DLZ613I	DLZPRDBD	2-43.4
	DLZPRPAR	2-43.8
DLZ614I	DLZPRDBD	2-43.4
	DLZPRPAR	2-43.8
DLZ615I	DLZPRDBD	2-43.4
	DLZPRABC	2-43.2
	DLZPRDL1	2-43.14
	DLZPRWFM	
DLZ616I	DLZPRDBD	2-43.4
DLZ617I	DLZPRDBD	2-43.4
DLZ618I	DLZPRDBD	2-43.4
DLZ623I	DLZPRABC	2-43-2
DLZ627I	DLZPRPSB	2-43.5
DLZ633I	DLZPRPAR	2-43.8
DLZ634I	DLZPRCT2	2-43-7
DLZ635I	DLZPRCT1	2-43.1
DLZ636I	DLZPRCT2	2-43.7
	DLZPRDLI	2-43.14
DLZ639I	DLZPRCT2	2-43.7
DLZ641I	DLZPRURC	2-43-12
DLZ642I	DLZPRSTW	2-43.15
DLZ643I	DLZPRURC	2-43.12
DLZ644I	DLZPRURC	2-43.12
DLZ645I	DLZPRDBD	2-43.4
	DLZPRURC	2-43.12
DLZ646I	DLZPRURC	2-43.12
DLZ647I	DLZPRSTC	2-43-11
DLZ648I	DLZPRSTC	2-43.11
DLZ649I	DLZPRSTC	2-43.11
DLZ650I	DLZPRUPD	2-43.10
DLZ651I	DLZPRDLI	2-43.14
DLZ652I	DLZPRDLI	2-43.14
DLZ653I	DLZPRURC	2-43.12
	DLZPRSCC	2-43.9
	DLZPRUPD	2-43.10
	DLZPRDLI	2-43.14
DLZ655I	DLZPRDLI	2-43.14
DLZ659I	DLZPRUPD	2-43.10
DLZ772I	DLZDXMT0	
DLZ796I	DLZDLD00	
DLZ797I	DLZDDLE0	
DLZ798I	DLZDLRG0	
	DLZDLRD0	
DLZ799I	DLZDLD00	
	DLZCPY10	
DLZ800I	DLZDLR00	
DLZ801I	DLZDLR00	
DLZ802I	DLZDLD00	
DLZ803I	DLZDLD00	
DLZ804I	DLZDLD00	
DLZ806I	DLZDLD00	
	DLZCPY10	
DLZ807I	DLZDLD00	

Message Number	Module	Figure Number
DLZ808I	DLZDLD00	
DLZ830I	DLZDHD00	
	DLZGGSP0	
DLZ831I	DLZDHDS0	2-13.5
	DLZDCI00	
DLZ841I	DLZDBH00	
DLZ844I	DLZDBH02	
DLZ845I	DLZDBH00	
DLZ847I	DLZDBH00	
DLZ848I	DLZDBH00	
DLZ850I	DLZDDLE0	
DLZ855I	DLZDDLE0	
DLZ860I	DLZDDLE0	
	DLZDXMT0	
DLZ861I	DLZDDLE0	
DLZ862I	DLZDDLE0	
DLZ863I	DLZDDLE0	
DLZ864I	DLZDDLE0	
DLZ868I	DLZDXMT0	
DLZ888I	DLZBACK0	
DLZ890I		
DLZ894I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ900I		
DLZ901I	DLZDLBL2	
DLZ902I	DLZDLBL2	
DLZ903I	DLZDLBL2	
DLZ904I	DLZDLBL0	
DLZ905I	DLZDLBL0	2-33.9
	DLZDLBL1	
	DLZDLBL2	
	DLZDLBL3	
	DLZUACB0	2-33
	DLZUAMB0	2-33.12, 2-33.13
	DLZDPSB0	2-33.14
DLZ906I	DLZDLBL0	
DLZ907I	DLZDLBL3	
DLZ908I	DLZDLBL3	
DLZ909I	DLZDLBL2	
DLZ910I	DLZDLBL0	
	DLZDLBL1	
DLZ911I	DLZDLBL2	
DLZ912I	DLZDLBL1	
DLZ913I	DLZDLBL1	
DLZ914I	DLZDLBL2	
DLZ915I	DLZDLBL1	
DLZ916I	DLZDLBL1	
DLZ917I	DLZDLBL1	
DLZ918I	DLZDLBL2	
DLZ919I	DLZDLBL2	
DLZ920I	DLZDLBL1	
DLZ921I	DLZDLBL0	
DLZ922I	DLZDLBL1	
DLZ923I	DLZDLBL1	
DLZ924I	DLZDLBL1	
DLZ925I	DLZDLBL1	
DLZ926I	DLZDLBL0	
	DLZDLBL1	

Message Number	Module	Figure Number
	DLZDLBL2	
	DLZDLBL3	
	DLZUAMB0	2-33.12, 2-33.13
DLZ927I	DLZDLBL1	
DLZ928I	DLZDLBL1	
DLZ929I	DLZDLBL0	
	DLZDLBL1	
DLZ930I		
DLZ931I	DLZDLBL1	
DLZ932I	DLZDLBL1	
DLZ933I	DLZDLBL3	
DLZ934I	DLZDLBL2	
DLZ935I	DLZDLBL2	
DLZ936I	DLZDLBL1	
DLZ937I	DLZDLBL1	
DLZ938I	DLZDLBL2	
DLZ939I	DLZDLBL1	
DLZ940I	DLZDLBL2	
DLZ941I	DLZDLBL2	
DLZ942I	DLZDLBL2	
DLZ943I	DLZDLBL2	
DLZ944I	DLZDLBL2	
DLZ945I	DLZDLBL0	
DLZ946I	DLZDLBL2	
DLZ947I	DLZDLBL2	
DLZ948I	DLZDLBL2	
DLZ949I	DLZDLBL2	
DLZ952I	DLZURPR0	
	DLZURGS0	2-35
DLZ953I	DLZURGP0	
DLZ954I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
	DLZURGP0	
DLZ955I	DLZURG10	2-36.2, 2-36.4
	DLZURGP0	
DLZ956I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURGP0	
DLZ957I	DLZURGS0	2-35
	DLZURG10	2-36
DLZ958I	DLZURGS0	2-35
	DLZURGP0	
DLZ959I	DLZURGS0	
	DLZURGP0	
DLZ960I	DLZURGP0	
DLZ961I	DLZURPR0	
	DLZURGS0	
	DLZURG10	
DLZ962I	DLZURPR0	2-34
DLZ963I	DLZURPR0	2-34
DLZ964I	DLZURPR0	2-34
DLZ965I	DLZURPR0	2-34
DLZ966I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
	DLZURGP0	
DLZ967I	DLZURGS0	2-35
DLZ968I	DLZURGS0	
	DLZURPR0	

Message Number	Module	Figure Number
	DLZURG10	2-36
	DLZURGP0	
DLZ969I	DLZURGS0	2-35
DLZ970I	DLZURGS0	2-35
DLZ971I	DLZURGS0	2-35
DLZ972I	DLZURGS0	
DLZ973I	DLZURGS0	
DLZ974I	DLZURGS0	
DLZ975I	DLZURGS0	2-35
DLZ976I	DLZURPR0	2-34
DLZ977I	DLZURG10	2-36.2
DLZ978I	DLZURG10	2-36.2
DLZ979I	DLZURG10	2-36.2
DLZ980I	DLZURG10	2-36.2, 2-36.4
DLZ981I	DLZURG10	2-36.4
DLZ982I	DLZURG10	2-36
	DLZURGP0	
DLZ983I	DLZURGP0	
DLZ984I	DLZURPR0	2-34
	DLZURGP0	
	DLZURGS0	2-35
	DLZURG10	2-36
DLZ985I	DLZURPR0	2-34
DLZ989I	DLZURG10	2-36.2
DLZ990I	DLZURGS0	
	DLZURGP0	
	DLZURG10	
DLZ991I	DLZURPR0	

DL/I STATUS CODES/MODULE CROSS REFERENCE

This table cross-references DL/I status codes (in alphabetic order) with the module(s) that can cause that status code to be set. The modules are described in Section 3 of this publication. The status codes are described in DL/I DOS/VS Messages and Codes.

Status Code	Module
AB	DLZDLA00
AC	DLZDLA00
AD	DLZDLA00, DLZISC00
AH	DLZDLA00
AI	DLZDLA00, DLZDLD00
AJ	DLZDLA00
AK	DLZDLA00, DLZDLRD0, DLZDLRE0
AM	DLZDLA00, DLZDLD00
AO	DLZDLD00, DLZDLR00, DLZDDLE0, DLZCPY10
DA	DLZDLD00
DJ	DLZDLA00
DX	DLZDLDD0
GA	DLZDLRC0
GB	DLZDLRA0, DLZDLRF0
GE	DLZDLRA0, DLZDLRC0, DLZDLRD0, DLZDLRE0
GK	DLZDLRC0
GP	DLZDLRA0
II	DLZDLRD0, DLZDLRF0, DLZDDLE0
IX	DLZDDLE0
KA	DLZCPY10
KB	DLZCPY10
KC	DLZCPY10
KD	DLZCPY10
KE	DLZCPY10
LB	DLZDLA00, DLZDDLE0
LC	DLZDLA00
LD	DLZDLA00
LE	DLZDLA00
NA	DLZDXMT0
NE	DLZDXMT0
NI	DLZDXMT0
NO	DLZDXMT0
RX	DLZDLD00
TA	DLZEIPO0
TB	DLZEIPO0
TC	DLZEIPO0
TE	DLZEIPO0
TF	DLZEIPO0
TG	DLZEIPO0
TH	DLZEIPO0
TI	DLZEIPB0, DLZEIPO0
TJ	DLZEIPO0
TK	DLZEIPO0
TL	DLZEIPO0
TN	DLZEIPB1, DLZEIPO0
TO	DLZEIPB1, DLZEIPO0
TP	DLZEIPB1, DLZEIPO0
V1	DLZDLA00
V2	DLZEIPB1, DLZEIPO0

Status Code	Module
V3	DLZEIPB1, DLZEIPO0
V4	DLZEIPB1, DLZEIPO0
V5	DLZEIPB1, DLZEIPO0
XD	DLZDLA01
XH	DLZDLA00

This section consists of the following appendixes:

Appendix A: Low-Level Code/Continuity Checking in DL/I.

Appendix B: DBD Generation.

Appendix C: PSB Generation.

Appendix D: DL/I Macros

APPENDIX A: LOW-LEVEL CODE/CONTINUITY CHECK IN DL/I

FLOW OF CONTROL

Low Level Code/Continuity Check (LLC/CC) in DL/I is used as a subroutine of a user-written application program that runs under DOS/VS. Control passes to and from the subroutine using standard calls.

LLC/CC in DL/I is a single control section (CSECT) which is structured into seven modules (see Figure 7-1). The entry modules 000 for update and 001 for initial generation of low-level codes have multiple entry points for call statements issued by the user-written application program, that is, a separate entry point for each source language that is supported. All modules have only a single exit point, all lower level modules 002 through 006 are only entered at one point.

All modules assemble and issue DL/I calls. The entry point for DL/I depends on the source language that is identified by the entry point into LLC/CC in DL/I. The language bits in the LLC/CC execution control block (LECB) identify the source language of the application program. If an unexpected status code of DL/I is reported in the appropriate PCB, the error bits in the LECB are turned on, and control is routed back directly to the entry modules 000 or 001.

LLC/CC in DL/I consists of the following modules:

- Module 000 is the entry module for maintenance of low level codes. It passes control to module 002 for execution.
- Module 001 is the entry module for initial generation of low level codes. It passes control to module 002 for execution.
- Module 002 is the common mainline control module. It follows down a hierarchical path of a product structure. For actual explosion, control is passed to module 003. If a particular hierarchical path is exhausted, module 004 is executed to process a parallel path on the same hierarchical level. If all parts on the same level are processed, module 005 steps up one level to identify a parallel path on the higher level. If the original starting level is reached, the complete structure is processed, and control is returned to module 000 or 001. Module 002 also detects loops and executes continuity check recovery in module 006.
- Module 003 explodes a particular part into all its components. Control is passed from and to module 002.
- Module 004 removes the part which has previously been processed from the hierarchical path thus opening a new hierarchical path via the next parent part on the same level. Control is passed from and to module 002.
- Module 005 steps up one level and removes the higher level part from the hierarchical path to open another path. Control is passed from and to module 002. If module 002 is not able to follow a new path on this level, module 005 may be executed repetitively.
- Module 006 handles restoring of old low-level codes if a continuity check is detected. Control is passed to and from module 002.

For a more detailed description, see the relevent HIPO charts at the end of Appendix A.

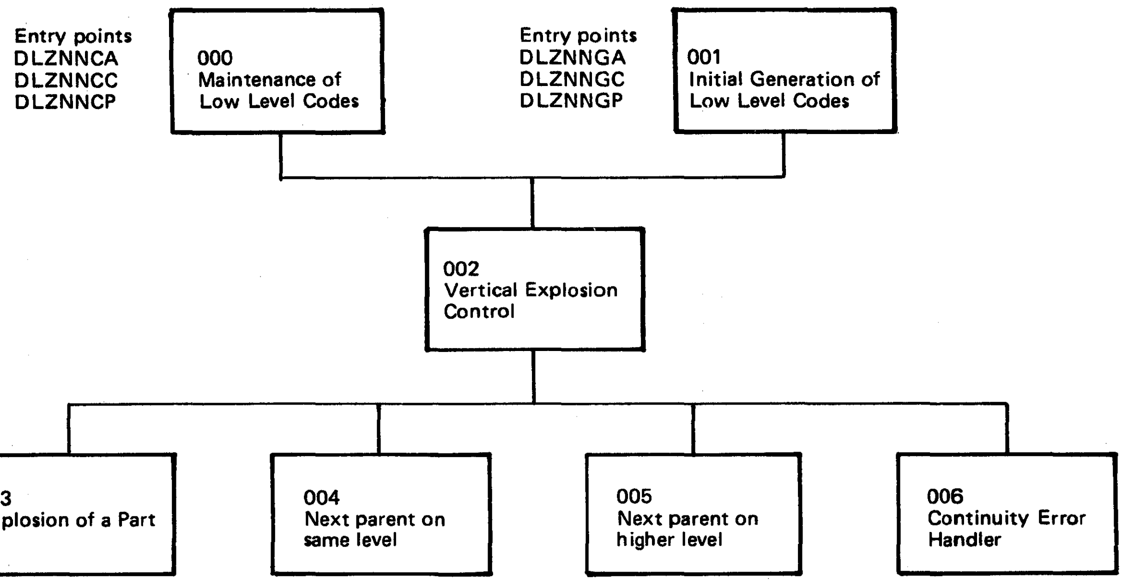


Figure 7-1 Structure of LLC/CC in DL/I

MODIFICATION AIDS

EXTERNAL NAMES

LLC/CC in DL/I uses external names in the directories and libraries of DOS/VS. The following table presents a list of all external names which are used. The user should obtain a DSERV listing to avoid duplicate names.

Type of program	SSL		RL		CIL
	A.books	E.books	Directory entries	Entry points	
Execution program	DLZNN	DLZNN	DLZNN*	DLZNNCA* DLZNNCC* DLZNNCP* DLZNNEC* DLZNNGA* DLZNNGC* DLZNNGP*	
Initialization program for the control data base	DLZNNICT	DLZNNICT			DLZNNICT

* May be modified by the user during customization.

LLC/CC EXECUTION CONTROL BLOCK (LECB)

The LECB of LLC/CC in DL/I is the focal point for all information related to actual operation of the execution program. It consists of 16 bytes which are subdivided into 4 fullwords. An entry point DLZNNEC is provided so that an application program may access the contents of the LECB.

The LECB contains the following information:

1. Identification portion (fullword 0):
   Bytes 0 through 3: C'LECB'=X'D3C5C3C2'
   This identifier facilitates location of the LECB in a main storage dump.

2. Execution control portion (fullword 1):
   Byte 4:

   - Bits 0 through 3: Run type bits
     Bit 0 and bit 1: Reserved
     Bit 2: 1 if IG run
     Bit 3: 1 if U run

   - Bits 4 through 7: Not used

   Byte 5:

   - Bits 0 through 3: Language bits
     Bit 0: Reserved
     Bit 1: 1 if Assembler
     Bit 2: 1 if COBOL
     Bit 3: 1 if PL/I

   - Bits 4 through 7: Not used

   Byte 6: Status byte

   - Bits 0 through 3: Completion bits (mutually exclusive)
     Bit 0: 1 if not completed, abnormal condition encountered
     Bit 1: 1 if component requires no change (U run only)
     Bit 2: 1 if part is already processed (IG run only)
     Bit 3: 1 if part has no components (IG run only, and only if bit 2 is off)

     Besides its function as an indicator, bit 3 also serves to transfer information whether a particular part in an explosion sequence has component parts. Bit 3 is turned off in module 002 before entering module 003. If no component parts are found during the execution of module 003, the bit is turned on. Upon return to module 002, the bit is tested to decide whether module 004 must be called.

   - Bits 4 through 7: Error bits, extending completion bit 0. A single error bit does not reflect a particular error condition, therefore, the hexadecimal representation of the total bit pattern in the status byte has to be analyzed.

     X'80' Parent part not found
     X'81' Component part not found (U run only)
     X'84' Continuity check for parent part
     X'85' Continuity check for any component part
     X'87' Input parameter in error

X'88'	Unexpected DL/I status code for parts data base
X'8A'	Unexpected DL/I status code for control data base
X'8C'	Both error conditions X'84' and X'88'
X'8D'	Both error conditions X'85' and X'88'
X'8E'	Both error conditions X'84' and X'8A'
X'8F'	Both error conditions X'85' and X'8A'

Byte 7: Not used

3. Parameter list portion (fullword 2):

   Bytes 8 through 11: Address constant pointing to the parameter list which has been previously submitted to DL/I by LLC/CC in DL/I. Contents is defined hexadecimal zeros prior to the first run through LLC/CC in DL/I. The address constant is not affected by insertion of locators if the application program is written in PL/I.

4. PCB save area portion (fullword 3):

   Bytes 12 through 15: Address constant pointing to a 64-byte save area for a PCB. This save area is initialized to blanks (X'40'), however, in case of an unexpected DL/I status code, the related PCB is saved into this save area. The PCB is stored left justified. If the length of the PCB exceeds 64 bytes, the exceeding data is truncated.

The contents of the status bytes is externally represented by the return codes of LLC/CC in DL/I.

IG stands for "initial generation of low level codes", U stands for "update of low level codes".

The LECB is located at the very end of the code of LLC/CC in DL/I. Therefore, the last byte of LLC/CC in DL/I may be addressed DLZNNEC+15.

LANGUAGE CONSIDERATIONS

During PSB generation, the source language of application programs using DL/I facilities is defined in the PSBGEN statements. While COBOL is handled like Assembler, the PCB has a different layout if PL/I is specified. Therefore, LLC/CC in DL/I has to use different entry points into DL/I depending on the source language of the invoking user-written application program.

The entry routines of the execution program of LLC/CC in DL/I offer different entry points. The x identifies initial generation mode (G) or update mode (C). Six different entry points are available for transfer of control:

- DLZNNxA and DLZNNxC are the entry points for application programs written in Assembler or COBOL, respectively. No special processing is required.

- DLZNNxP are the entry points for application programs written in the PL/I Optimizer language. Upon entry, the address constants in the parameter list pointing to the locators of the parameters transmitted are replaced by the addresses which are stored in the respective locators.

For each source language, the appropriate language bit in the LLC/CC execution control block (LECB) is set upon entry.

When a DL/I call is issued, the language bits are tested to specify the right entry point in DL/I: ASMTDLI, CBLTDLI, or PLITDLI. If the source language is PL/I, the parameter list is encoded to transfer address constants pointing to locators rather than pointing directly to the parameters.

SAVE AREAS

LLC/CC in DL/I contains a set of save areas which facilitate tracing main storage dumps. The most important save areas are:

- Standard save area, addressed by register 13. Symbolic name is SAVE.
- Return addresses for subroutines, that is, contents of register 14. Symbolic names are CALLSV, PARMJUSV, INSRSAVE, SETUPSV, M002SV through M006SV. Save areas M002SV through M006SV are reset to hexadecimal zeros when the respective modules M002 through M006 are left again.
- Save area for the contents of register 1 when entering LLC/CC in DL/I, that is, address of the parameter list submitted from the application program. Symbolic name is R1SAVE.
- Save area for the leftmost 240 bytes of a PCB if an unexpected DL/I status code is encountered. Symbolic name is PCBSAVE. The address of PCBSAVE is also available in fullword 3 of the LECB.

REGISTER USAGE

R0:	Work register
R1:	Work register, address of parameter lists during parameter transfer
R2:	Address of parameter list when preparing parameter transfer
R5:	Work register
R6:	Address of PCB for parts data base
R7:	Address of PCB for control data base
R8:	Base register
R9:	Second base register
R12:	Reserved
R13:	Address of register save area
R14:	Standard return address
R15:	Standard linkage register

HIPO DIAGRAMS FOR LLC/CC

The following HIPO diagrams describe the seven modules (000-006) of LLC.

Input

INPUT
- PARTS PCB
- CTL.-PCB
- PAR. PART
- COMP. PART

PARTS DB

Processing

This module is entered from an application program via CALL.

[01] Obtain and adjust input data.

[02] Read parent and component part. if not found, go to step 9.

[03] Increment the LLC of the parent part by 1 to obtain the initial LLC. Set actual LLC to initial LLC. if the actual LLC is not higher than the LLC of the component part, no processing is required and control is passed to step 9.

[04] Insert root segment LLCTL with key = X to start the control data base.

[05] Insert dependent segments into the control data base for parent part and for component part.

Output

LLC
- INITIAL
- ACTUAL

CONTROL DB

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

Notes	Routine	Label	Ref
[01] The calling application program uses three different entry points for Assembler, COBOL or PL/I. A parameter list consisting of 6 pointers identifies 6 fields, 4 of them containing input data, 2 of them expecting output data.		DLZNNCA DLZNNCC DLZNNCP	
[05] The original LLC of the component is saved in an UPDMASTR segment. A PARTBEXP segment for continuity check control with a key composed of hexa zeros plus the key of the parent part is inserted. The continuity check itself is explained in note 6 of 002 - VERTICAL EXPLOSION CONTROL. A PARTBEXP segment for explosion control with a key composed of the actual LLC plus key of the component part is inserted.		PARTBEXP	

Notes	Routine	Label	Ref

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

Input

Processing

Output

[06] Replace the old LLC in the component part by the actual LLC. ⟹ PARTS DB

[07] Execute LLC/CC in DL/I.

002
VERTICAL EXPLOSION CONTROL
3.1.3

[08] Remove all control information from the control data base by deleting the root segment LLCTL with key = X. ⟹ CONTROL DB

[09] Set up return information. Return to calling application program. ⟹ OUTPUT: RETURN CD. / LOOP KEY

000 - MAINTENANCE OF LOW - LEVEL CODES

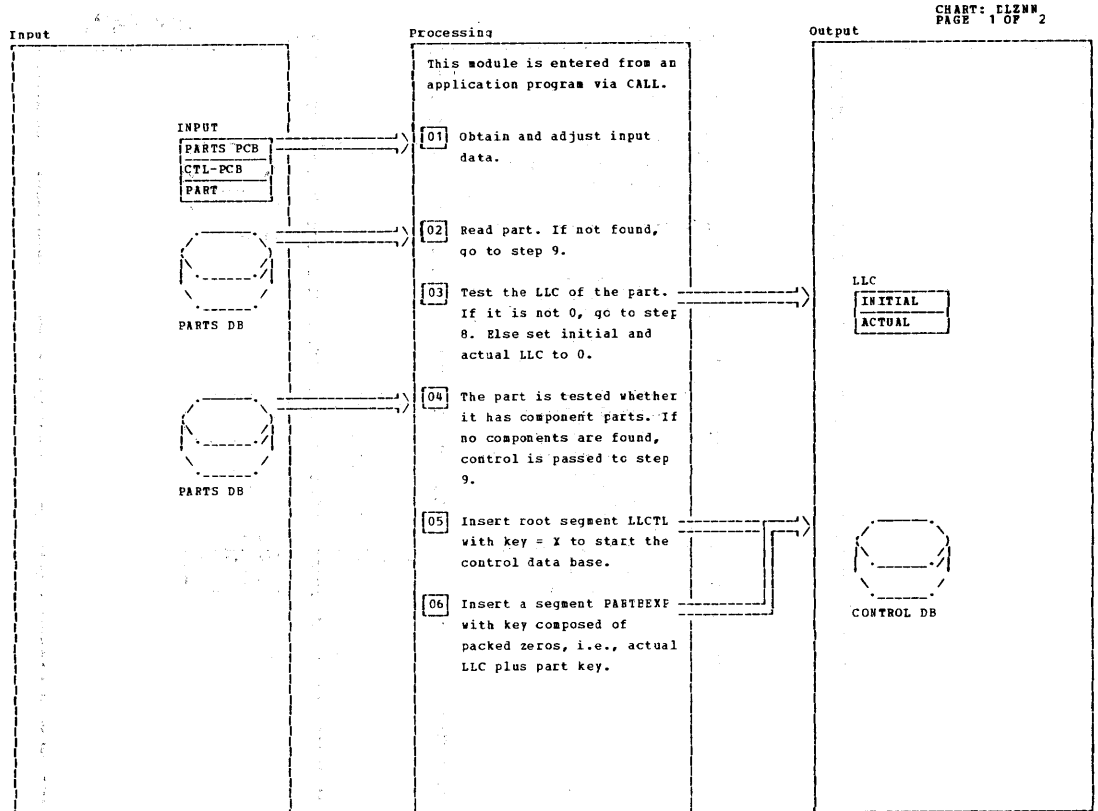
HIPOMAT 1.1 Diagram - 3.1.1-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
[09] Return information is obtained from the status bits of the LECB and from the internal loop key field.		DLZNNEC					

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-02

CHART: DLZNN
PAGE 1 OF 2



001 - INITIAL GENERATION OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.2-01

Notes	Routine	Label	Ref
01 The calling application program has three entry points for Assembler, COBOL or PL/I. A parameter list consisting of 5 pointers identifies 5 fields, 3 of them containing input data, 2 of them expecting output data.		DLZNNGA DLZNNGC DLZNNGP	
04 A bit is set in the LECB to indicate that no component part exists.		LECBSNOC	

Notes	Routine	Label	Ref

001 - INITIAL GENERATION OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.2-01

Input	Processing	Output
	07 Execute LLC/CC in DL/I.	
	002 VERTICAL EXPLOSION CONTROL 3.1.3	
	08 Remove all control information from the control data base by deleting the root segment LLCTL with key = X	CONTROL DB
	09 Set up return information. Return to calling application program.	OUTPUT: RET.-CODE, LOOP KEY

001 - INITIAL GENERATION OF LOW - LEVEL CODES

HIPOBAT 1.1 Diagram - 3.1.2-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
09 Return information is obtained from the status bits of the LECB and from the internal loop key field.		DLZNNFC					

001 - INITIAL GENERATION OF LOW - LEVEL CODES

HIPOBAT 1.1 Diagram - 3.1.2-02

Input

LLC
ACTUAL
INITIAL

CONTROL DB

Processing

[01] The actual low-level code is used to identify the next part to be processed. A segment PARTBEXP is read with key equal or greater to LLC plus hex zeros.

[02] If no segment PARTBEXP is found , the actual LLC is exhausted. Control is passed to module 005.

005
NEXT PARENT ON HIGHER LEVEL. 3.1.3.3

[03] Upon return from 005, the actual LLC is tested. If it is higher than the initial LLC, control is passed to step 1, else processing is completed and control is passed to step 8.

Output

002 - VERTICAL EXPLOSION CONTROL — HIPOMAT 1.1 Diagram - 3.1.3-01

Notes	Routine	Label	Ref
[01] Vertical explosion control is performed by means of PARTBEXP segments. Each time a new component part is encountered with a low-level code which needs replacement, a PARTBEXP segment - key = LLC + part key - is created. When going down a product-structure tree, this step of LLC/CC in DL/I identifies a new component part to become a parent part within the recursive process of explosion. Explosion proceeds on a FIFO basis.		PARTBEXP	
[02] During previous explosions, no component part was found requiring the replacement of its current low-level code, or no component part was found at all. Therefore, no segment PARTBEXP was inserted.			
[03] The initial low-level code was established either in module 000 or in module 001, resp.			

Notes	Routine	Label	Ref

002 - VERTICAL EXPLOSION CONTROL — HIPOMAT 1.1 Diagram - 3.1.3-01

Input

PARTS DB

Processing

04 If a segment is found, read the root segment of the part.

05 Increment the actual LLC by 1 to post into the components of the new part.

06 Perform continuity check and go to step 9. If the check fails, restore old LLC status in 006.

006
ERROR RECOVERY HANDLER
3.1.3.4

07 Upon return from 006, control is passed to step 12.

Output

PART
ACTUAL

LLC
ACTUAL

CONTROL DB

OUTPUT DATA
LOOP KEY
ERROR INFO

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
06 The continuity check is performed using the segment type PARTBEXP. Each time a new part is becoming exploded, a segment is inserted which only consists of the part key preceded by 2 bytes hexa zeros. If a part occurs twice in a particular hierarchical path, DL/I will reject the request for insertion because a segment with same key is already existing. LLC/CC in DL/I tests this condition and signals continuity check. Insertion is processed here. However if in updating mode, LLC/CC in DL/I inserts a PARTBEXP segment of this type for the part identified by PARM3 already in 000, step 5.		PARTBEXP					

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-02

Input	Processing	Output
	[08] If the continuity check did not indicate a loop, the actual part will be exploded into its component parts. 003 EXPLOSION OF A PART 3.1.3.1 [09] Upon return from 003, a test is made to detect whether the actual part has had component parts at all. If components were found, control is passed back to step 1. [10] Else, 004 is employed. 004 NEXT PARENT ON SAME LEVEL 3.1.3.2 [11] Upon return, go to step 1. [12] Go back to higher level module 000 or 001	

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-03

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
[09] A switch in the LFCB is used to transfer information whether a part has component parts. The switch is turned off before entering 003, i.e., it is assumed that the part has components. Upon return from 003, the status of this switch is tested. If the switch is on, 003 has indicated that the part does not have components.		LFCBSNOC					

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-03

Input	Processing	Output
PARTS DB	[01] Read the first or next component segment of the actual part. If not found, processing is completed and control is passed to step 6.	
LLC ACTUAL	[02] Compare actual LLC and LLC in the component. If the actual LLC is not higher, no further processing is required and control is passed back to step 1.	
	[03] Save the old LLC of the component in an UFDMASTR segment.	CONTROL DB
	[04] Replace the old LLC of the component by the actual LLC.	PARTS DB

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-01

Notes	Routine	Label	Ref
[01] If the no-component-found LECBSNOC condition was raised when retrieving the first segment, a switch indicates to 002 that the actual part does not have any component parts at all and another part has to be selected for explosion.		LECBSNOC	

Notes	Routine	Label	Ref

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-01

Input	Processing	Output
	[05] Insert a segment PARTBEXP with key composed of actual LLC plus PARTKEY of the component. Go back to step 1.	CONTROL DB
	[06] Go back to module CC2.	

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-02

CHART: DLZNN
PAGE 1 OF 1

Input	Processing	Output
PART ACTUAL	01 Remove the actual part form the hierarchical path.	
CONTROL DB	02 Delete segment PARTBEXP with key composed of hex zeros and the key of the actual part.	
LLC ACTUAL	03 Decrement actual LLC by 1.	LLC ACTUAL
CONTROL DB	04 Remove the first segment PARTBEXP with key = actual LLC + hex zeros since it has been completely exploded.	
	05 Return to module 002.	

004 - NEXT PARENT ON SAME LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.2-01

Notes	Routine	Label	Ref
02 A part may occur multiple times within a product-structure tree. However, it must not pccur twice within a hierarchical path. Therefore, if a hierarchical path is left or is modified, all PARTBEXP segments for continuity check related to branches which have become obsolete will be removed.			
04 When returning to step 1 in module 002, the next part on the same level will be read. Step 3 in 004 neutralizes step 4 in 002.			

Notes	Routine	Label	Ref

004 - NEXT PARENT ON SAME LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.2-01

Input	Processing	Output
LLC ACTUAL	01 Decrement the actual LLC by 1.	LLC ACTUAL
CONTROL DB; LLC ACTUAL	02 Delete the first segment PARTBEXP identified by a key composed of the actual LLC and of hex zeros.	PART KEY
CONTROL DB; PART KEY	03 Delete the segment PARTBEXP identified by a key composed of hex zeros and the part key retrieved in step 2.	
	04 Return to module 002.	

005 - NEXT PARENT ON HIGHER LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.3-01

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
01 This allows to continue in module 002 at step 1 on the next higher, i.e., numerically lower level.							
02 A part may occur multiple times within a product-structure tree. However, it must not occur twice within a hierarchical path. Therefore, if a hierarchical path is left or is modified, all PARTBEXP segments for continuity check related to branches which have become obsolete will be removed.							
03 Since this hierarchical path is exhausted, the control segment for explosion is deleted.							

005 - NEXT PARENT ON HIGHER LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.3-01

CHART: ELZNN
PAGE 1 OF 1

Input	Processing	Output
CONTROL DB	[01] Read sequentially all UPDMASTR segments.	
PARTS DB	[02] Restore all LLCs of parts referenced by an UPDMASTR segment to its original value.	PARTS DB
	[03] Return to module 002.	

006 - CONTINUITY ERROR HANDLER

HIPOMAT 1.1 Diagram - 3.1.3.4-01

Notes	Routine	Label	Ref

Notes	Routine	Label	Ref

006 - CONTINUITY ERROR HANDLER

HIPOMAT 1.1 Diagram - 3.1.3.4-01

APPENDIX B: DBD GENERATION

DESCRIPTION OF DBD GENERATION

DBD generation is composed of a set of DL/I macro instructions, the execution of which creates the user-specified data base description (DBD) and places it in the DOS/VS source statement library. The following macro instructions represent DBD generation:

Macro Instruction Name	Purpose
DBD	Allows the DL/I user to define the name of the DBD and the data base organization
DATASET	Allows the DL/I user to define names for data sets representing a data base, the device type used for storage of the data base, the logical record length, and the blocking factor for the physical records in the data sets representing the data base
ACCESS	Used in conjunction with ACCESS=HD to define external access points, primary and secondary, to the data base.
SEGM	Allows the user to specify a DL/I segment, its parent segment, the segment length, the segment name, and segment prefix information
LCHILD	Allows the user to define an index relationship or a logical relationship in which a segment will participate.
XDFLD	Allows the user to define secondary indexing relationships.
FIELD	Allows the DL/I user to specify a data field or key field for a segment. The field definition includes the related segment field name, field start position in segment, field length, and field type.
DBDGEN	Causes the segments, fields, and data sets defined in the SEGM, FIELD, and DATASET macro instructions to be generated into an object module.
FINISH	Checks whether a DBDGEN statement was present.

The DBD generation macros utilize a universal set of globals. The COPY book for these globals is in the DOS/VS Source Statement Library and is named DLZDBGLB.

DBDGEN MACRO CALLING SEQUENCE

External Macro	Inner 1	Inner 2
DBD	DLZALPHA	
DATASET	DLZALPHA	
	DLZCKDDN	
	DLZDEVSI	
ACCESS	DLZALPHA	
	DLZXTDBD	
SEGM	DLZALPHA	
	DLZSOURS	DLZXPARM
		DLZALPHA
		DLZXTDBD
	DLZXPARM	
	DLZXTDBD	
	DLZSETFL	DLZSEGPT
XDFLD	DLZALPHA	
LCHILD	DLZALPHA	
	DLZXTDBD	
FIELD		
DBDGEN	DLZSEGPT	
	DLZLRECL	
	DLZSOURS	FIELD
	DLZXTDBD	
	DLZCAP (See Note)	
	DLZHIERS	DLZSDURS
		DLZHIERS
FINISH		

Note: Not called if device is FBA.

GLOBAL SYMBOLS			MACROS																			
NAME	TYPE	SIZE	DBD	DATASET	ACCESS	SEGM	FIELD	LCHILD	XDFLD	DBDGEN	FINISH	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD
A#	A		S		U	R		U	R	R												
ACC	C		U	U	R	R		R	R	R									R			
ACCAC	B	255			S					R						R				R		
ACCCH	A	255								U						R						
ACCDKV	B	255			S				S	R												
ACCDL	A	255								R										U		
ACCEDS#	A	255			S			S		R												
ACCGDBD	B	255			S					R												
ACCIAD	B	255			S			S		R						R						
ACCKL	A	255								R										U		
ACCNDXF	C	255			S			S	S	R						R				R		
ACCPRI	B	255						S	S	U										R		
ACCRAD	B	255			S					R												
ACCREF	C	255			S			S		R										R		
ACCSEC	B	255			S				S	R												
ACCSS#	A	255								U										R		
ACCSSN	C	255			S				S	U												
ACCSSS	B	255			S				S											R		
ACCTES	B	255			S			S	S	R										R		
ACCTS#	A	255			S			S	S	U										R		
ACCTSN	C	255			S					U												
ACCXD#	A	255			S				S	R												
ALIAS	B					U		R														
CAPCYL	A									R			S									
CAPTRK	A									R			S									
CIIL	A									R							U	R				
CSB	B					S	R															
DBD	B		U	R	R	R	R	R	R	R												
DBDERR	B		S	S	U	S	S	S	S	U	R			S		S		S	S	U	S	S
DBDTERM	B			R	R	U	R	R	R													
DBN	C		S			R		R		R										R		
DBNAME	C	255								R												U
DD#	A													U								
DDNAME	C	20												U								
DEVADR1	C			S						R												
DEVADR2	C			S						R												
DS#	A			U		U				R					R		R					

A = algebraic　　C = character　　S = set
B = binary　　R = reference　　U = reference/set

Figure 7-2. DBDGEN MACRO-GLOBAL Symbol Cross Reference (Part 1 of 5)

GLOBAL SYMBOLS			MACROS																			
NAME	TYPE	SIZE	DBD	DATASET	ACCESS	SEGM	FIELD	LCHILD	XDFLD	DBDGEN	FINISH	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD
DS#SEG	A	10				U				R												
DSBLK	A	10		U						R							U					
DSBLKS	A	10															S					
DSDEV	C	10		U						R								R				
DSFBFF	A	10		S						R												
DSFBLK	A	10		S						R												
DSFSPF	A	10		S						R												
DSLKL	A	10								R						U						
DSLSL	A	10								R						U	R					
DSNAME	C	10		S						R												
DSOBLK	A	10		S						R							U					
DSOBLKS	A	10								U												
DSONAME	C	10		S						R												
DSOREC	A	10		U						U							U	R				
DSREC	A	10		U						U							U	R				
DSSCN	A	10		S						R												
DSSKL	A	10								R						U						
DSSSL	A	10								R						U						
DSTRK	A	10		R											S		R	R				
DSTRK2	A	10													S		R					
DSTRK3	A	10													S		R					
DSTRK4	A	10													S		R					
EC	C											U										
ERROR	B					R														R	S	
EXTDB#	A				R					R												U
EXTDBN	A.				R	R		R		R										R		U
F#	A						U			R										R		
FLDCH	A	1020					U			U						U				R		
FLDLG	A	1020					U			R						R				R		
FLDMV	B	1020					S			R												
FLDNM	C	1020					U			R						R						
FLDS#	A	1020					S			R												
FLDSC	B	1020					S									R						
FLDSO	B	1020					S			U						R						
FLDST	A	1020					U			R						R						
FLDTY	A	1020					S			R												
GENCHK	B									S	R											

A = algebraic C = character S = set
B = binary R = reference U = reference/set

Figure 7-2. DBDGEN MACRO-GLOBAL Symbol Cross Reference (Part 2 of 5)

GLOBAL SYMBOLS			MACROS																			
NAME	TYPE	SIZE	DBD	DATASET	ACCESS	SEGM	FIELD	LCHILD	XDFLD	DBDGEN	FINISH	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD
HDAM	B		U	S	S	R		R	U	R						R						
HDB	A		S	S						R												
HDORG	B		U	U		R		R	R	R						R	R	R	R			
HDRBN	A		S	S						R												
HIDAM	B		U	S		S		S	R	U							R					
HIORG	B		U	S						R												
HISAM	B		U	U						R								R				
HSAM	B			S		S																
HSORG	B		U	U						R					R		R	R				
IMSC	B		S							R						R	R					
INDX	B		U	U		R		R		R						R		R				
LC#	A					U		U		R												
LCCHN	A	255								U												
LCDBLP	B	255						S		R												
LCDS#	A	255				S		S		R												
LCLC	B	255				S				R												
LCLP	B	255						S														
LCNM	C	255				S		S		R												
LCO	A															U						
LCPS	C	255						S		R												
LCRULE	A	255						S		R												
LCS#	A	255				S		S		R												
LCSNAME	C	255																				
LCSNGP	B	255						S		R												
LCXD	A					U		U	U	R												
LEV	A									R						U						
LOGICAL	B		U	U		R	R	R		R						R				R		
LRECIL	A																U	R				
MAXACC	A		S		R			R														
MAXAPS	A		S							R												
MAXDB#	A		S																			R
MAXDS#	A		S	R																		
MAXFLDS	A		S				R															
MAXFPS	A		S				R															
MAXLCH	A		S			R		R														
MAXSEGS	A		S			R																
MAXSS	A		S																	R		

A = algebraic   C = character   S = set
B = binary   R = reference   U = reference/set

Figure 7-2. DBDGEN MACRO-GLOBAL Symbol Cross Reference (Part 3 of 5)

GLOBAL SYMBOLS			MACROS																			
NAME	TYPE	SIZE	DBD	DATASET	ACCESS	SEGM	FIELD	LCHILD	XDFLD	DBDGEN	FINISH	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD
MAXXDF	A		S		R				R													
NSTRT	A					S	U															
ORG	A		S		S	S		S		U												
PLIST	C	100				U													R	R	S	
PLISTK	A	100				R														R	S	
PNBR	A					U				U						U			R	R	U	
POS	A											U										
QUITB	B		R	R	R	R		R	R			S								R		
RAPS	A		S	S						R							U					
RMN	C		S	S	S					R												
RMSEGM	C				U					R												
RMSFLD	C				S					R												
ROOT	B					U																
S#	A					U	R	R	R	R							R	R	R	R		R
S#ACC	A	255	S				S			U						R						
S#FLD	A	255					U			R						R						
S#LC	A	255				U		U								R						
S#PC	A	255								U						R						
SCK	A	255														U						
SCRN	C	255				S				R												
SDL	A	255				S	U			R						R	R	R				
SDPPP	B	255				S										R						
SDS#	A	255				S										R	R	R				
SFACC	A	255						S		U						R						
SFFLD	A	255					U			R						U				R		
SFLC	A	255								U												
SFPC	A	255								U						R						
SHISAM	B		U	U		R				R						R	R	R				
SHSAM	B		U	S		U										R	R					
SICOMP	B	255				S				R						R						
SIITS	B	255								S						R						
SILC	B	255				U				R						R			R			
SIVAR	B	255				S				R						R						
SIVLC	B	255				S	R			R						R						
SLEV	A	255								R						U						
SLFLD	A					S	U															
SMINDL	A	255				S										R						

A = algebraic   C = character   S = set
B = binary   R = reference   U = reference/set

Figure 7-2. DBDGEN MACRO-GLOBAL Symbol Cross Reference (Part 4 of 5)

GLOBAL SYMBOLS			MACROS																			
NAME	TYPE	SIZE	DBD	DATASET	ACCESS	SEGM	FIELD	LCHILD	XDFLD	DBDGEN	FINISH	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD
SLLC	A					S		U														
SNAME	C	255				U	R	R	R	R						R		R		R		R
SP #	A	255								U						R						
SPC	A	255								R						U				R		
SPCCHN	A	255								S						R						
SPCTR	B	255						U								R						
SPL	A	255								U						U	R	R				
SPLTW	B	255				S				R						R			U			
SPLTWB	B	255														R			U			
SPNT	B	255														R			U			
SPPNAME	C	255				S				R												
SPPP	B	255				S		S		S						U						
SPRD	B	255								U												
SPTW	B	255				S										R			U			
SPTWB	B	255														R	R		U			
SRULES	A	255														R			U			
SS #	A																			U		
SSDB #	A	510																		U		
SSNAME	C	510																		U		
SSS #	A	510																		U		
SSX	A	255					U													R		
SVLINIT	B	255				S				R						R						
XD #	A				U				U											R		
XDACC #	A	4095			S				S											R		
XDF #	A	4095																		U		
XDIDDF	B	4095							S											R		
XDISCF	B	4095			S				S											R		
XDISHF	B	4095			S				S											R		
XDISRP	B	4095			S				S											R		
XDISSF	B	4095			S				S											R		
XDNAME	C	4095			S				U	R										U		
XDSUPC	C	4095			S				S											R		

A = algebraic C = character S = set
B = binary R = reference U = reference/set

Figure 7-2. DBDGEN MACRO-GLOBAL Symbol Cross Reference (Part 5 of 5)

DBDGEN MACRO DESCRIPTIONS

DATASET MACRO

This is an external macro through which data set/data set group information is specified by the user.

DBD MACRO

This is an external macro through which DBD control information is specified by the user.

DBDGEN MACRO

This macro terminates the DBD specification process. If the error switch, DBDERR, is not set, the control block generation phase is entered to create the required block entries.

DLZALPHA MACRO

```
|        | DLZALPHA |  A1
|        |          |  AN
|        |          |  AN1 ,FIELD=,CHAR=,MAC=,OPER=
|        |          |  ALL
|        |          |  DBD
|        |          |  HEX
|        |          |  BINARY
|        |          |  BYTE
```

This macro is used to check the syntax of macro operands. The first (positional) parameter identifies the valid format as follows:

A1 or omitted	=	First character must be A-Z, @, #, or $.
AN1	=	First character must be 0-9, A-Z, @, #, or $.
ALL	=	First character must be A-Z, @, #, or $. Remaining characters must be A-Z, @, #, $, or 0-9.
AN	=	All characters must be 0-9, A-Z, @, #, or $.
HEX	=	All characters must be 0-9, A-F.
BINARY	=	All characters must be 0 or 1.
DBD	=	First character must be A-Z, #, or $. Remainder must be A-Z, #, $, or 0-9.

BYTE	=	Operand must be a valid one byte self defining term.

The other parameters are:

FIELD	=	Field to be checked.
CHAR	=	Starting position for check if other than first position.
MAC	=	MNOTE prefix for error MNOTEs.
OPER	=	Name of operand being processed for MNOTEs.

DLZCAP MACRO

	DLZCAP	DEVICE, BLOCKSIZ

This macro is called by DBDGEN to calculate the block capacity per track and cylinder provided the blocks do not have keys. These numbers are required to generate some entries within the DTFSD (HSAM) and ACB-extension. The capacities are returned using global arithmetic variables (GBLA). Input values are:

DEVICE:	2314, 3330, 3333, 3340, 3375, 3380
BLOCKSIZ:	in bytes (key length = 0)

Output (GBLA) and MNOTE:

CAPTRK:	number of blocks per track (GBLA)
CAPCYL:	number of blocks per cylinder (GBLA)
MNOTE:	DMAN150 if invalid device
MNOTE:	Comment containing $CAPTRK and $CAPCYL if calculation was successful

DLZCKDDN MACRO

	DLZCKDDN	FILENAME

This macro checks the validity of filenames specified by the user and verifies that the specified filenames are not duplicated.

The operand is:

FILENAME

is the one- to seven-character filename to be checked.

DLZDEVSI MACRO

	DLZDEVSI	DEVICE

This macro is called by the DATASET macro to set device capacity values for the specified device type. The device value specified in the DEVICE operand of the DATASET statement is passed to this macro.

DLZHIERS MACRO

	DLZHIERS	S,LV,LCP

This macro is called twice by the DBDGEN macro. The first time is to validate segment hierarchies, field names, and locations. The second time, LV is set to 'GENERATE', to generate the segment table entries for the DBD.

The macro calls itself to process dependent segment definitions.

The first time operands are:

S = Segment table entry number of the segment to be processed.

LV = Level for the segment to be processed.

LCP = If one, it indicates that the segment to be processed is below a logical child in the physical hierarchy.

The second time operands are:

S = Segment table entry number of entry to be generated.

LV = 'GENERATE'

LCP = Ignored.

DLZLRECL MACRO

	DLZLRECL	

This macro is called by DBDGEN to calculate LRECL and BLKSIZE.

DLZSEGPT MACRO

	DLZSEGPT	

This macro is called by DBDGEN to maintain the globals DSLSL and DSSSL, which contain the sizes of the largest and smallest segments in a data set, respectively. This macro produces error messages DGEN250, 251, 252, 253, 254, 255, 256, and 257 if the segment referenced by the operand value violates those rules.

DLZSETFL MACRO

	DLZSETFL	RULES=

This macro processes the POINTER or PTR operand of the SEGM macro and sets the globals to reflect the entered values. The globals set by this macro comprise bytes 0 and 1 of the 4-byte flags field of the SEGTAB entry for this segment.

This macro is not entered if the DLZXPARM macro encountered an error while generating the &PLIST matrix, or if the SEGM macro detected an error in the POINTER or PTR parameter list.

Messages:

An error message is produced and processing is terminated if:

- An invalid keyword is encountered in the parameter list, or
- The RULES operand is omitted or invalid

Flag Byte 1 is set as follows:

Bit 1 - CTR	If TWINBWD and/or LTWINBWD is specified,
2 - TWIN	Bit 2 and/or Bit 5 is set on, in
3 - TWINBWD	addition to Bit 3 and/or Bit 6,
4 - PARNT	respectively.
5 - LTWIN	
6 - LTWINBWD	
7 - LPARNT	
8 - NOTWIN	

Flag Byte 2 &SRULES is set as follows:

Bits 1 & 2 — Indicate segment insert rule, where:

10 - Physical
01 - Virtual
11 - Logical (Default)

Bits 3 & 4 — Indicate delete rule and set same as insert. (Default value is LOGICAL).

Bits 5 & 6 — Indicate replace rule and set same as insert. (Default value is VIRTUAL).

Bits 7 & 8 — Indicate physical location of inserts for nonsequenced segments, where:

10 - First
01 - Last (Default value)
11 - Here

The operands are:

RULES=

specifies the RULES= operand as specified on the SEGM statement

DLZSOURS MACRO

	DLZSOURS	PARM=,OPTION

This macro is called by the SEGM macro to process the SOURCE parameter, by DBDGEN to validate index table entries and generate the source and index tables, and by DLZHIERS to generate the segment table entries for number of source segments and offset to first entry.

The parameters are:

OPTION = ADD - process source operand. PARM = operand.

= CHECK - validate and connect index table entries. PARM ignored.

= LIST - generate SOURCE and index tables. PARM ignored.

= FIND - generate segment table entries, PARM=segment table number.

DLZXPARM MACRO

	DLZXPARM	PARM=,MODEL=,MSG=

When used this macro extracts parameters from a sublist and stores them in a global matrix (PLIST). Null values in the parameter list are stored as null values in the PLIST matrix.

The operands are:

PARM=

specifies the input parameter list values

MODEL=

identifies the model for a fully defined sublist, indicating the locations in the PLIST matrix for the parameters. (for example, MODEL=(1,2), (3,4,5)).

MSG=

identifies the parameter being processed in the first operand and the MNOTE prefix in the second operand.

DLZXTDBD MACRO

	DLZXTDBD	DB,CODE

This macro builds an external data base reference table. It is called by SEGM, LCHILD, and DBDGEN.

The operands are:

DB

specifies a data base name or segment name

CODE

specifies the value SEGM or is omitted.

If the value SEGM is specified in the CODE operand, the segment name (SN) is searched to locate the value specified in the DB operand; when found, the symbol EXTDBN is set to contain an 01 in byte 0, and bytes 1, 2, and 3 contain an offset into SEGTAB. If the segment is not found, an MNOTE error message is produced.

If the CODE operand is omitted, the external data base reference table (DBNAME) is searched for the DB entry, and, if found, the symbol EXTDBN is set to contain the position of the found entry. If the DB value is not found, the value is added to the table and EXTDBN is set to that entry.

FIELD MACRO

This is an external macro used to define fields within a segment.

FINISH MACRO

This is an external macro used to check whether a DBDGEN statement is supplied.

LCHILD MACRO

This is an external macro used to define index or logical relationships for HIDAM and HDAM or logical relations for HD.

SEGM MACRO

This is an external macro used to define data base segments.

XDFLD MACRO

This is an external macro used to define in connection with the LCHILD statement secondary index relationships for HIDAM and HDAM.

ACCESS MACRO

This is an external macro used to define external access points to the data base for ACCESS=HD.

DBD GENERATION CONTROL BLOCK OUTPUT - DBDGEN

The data base description block (DBD) is the result of each data base generation.

- DIAGRAM OF DBDGEN CONTROL BLOCK OUTPUT

GENERAL STRUCTURE:

DIRECTORY
PREFIX
DMANTAB
ACB EXTENSION (SAME AS DMB) (If HSAM or SSAM, DTFs)
SEGTAB
FLDTAB
EXTDBD
LCHILD
SORTAB
INDXTAB
DACT (Same as DMB)
COMPRESSION EXIT CSECTS (same as DMB)
INDEX EXIT CSECTS (same as DMB)

1. DIRECTORY LAYOUT

Hex	Dec	Name	Ln	Description
0	0	AMODLEV	1	Release level (X'00'=1.0, X'11'=1.1)
1	1	APREFIX	3	Address of PREFIX
4	4	ASEGTAB	4	Address of SEGTAB
8	8	AFLDTAB	4	Address of FLDTAB
C	12	ALCHILD	4	Address of LCHILD
10	16	AEXTDBD	4	Address of EXTDBD
14	20	ASORTAB	4	Address of SORTAB
18	24	ARMVTAB	4	Address of DMBDACS
1C	28	AINDXTAB	4	Address of INDXTAB
20	32	ADSGCB	4	Address of ACB extension

2. PREFIX LAYOUT

Hex	Dec	Name	Ln	Description
0	0	PREDBDNM	8	DBD name
8	8	PRENOLEV	2	Number of levels in data base
A	10	PRENOSEG	2	Number of segments
C	12	PREACCES	1	Organization

Name	EQU	Meaning
PRESHIS	X'01'	Simple HISAM
PREISAM1	X'02'	HISAM
PRESSAM	X'04'	Simple HSAM
PREHSAM	X'05'	HSAM
PREHD	X'06'	HDAM
PREHI	X'07'	HIDAM
PRENDEX	X'08'	INDEX
PREIMSC	X'80'	IMS compatibility required

Hex	Dec	Name	Ln	Description
D	13	PRENODSG	1	Number of data sets
E	14	PRENODBD	2	Number of externally referenced data bases
10	16	PRERNDM	8	Randomizing algorithm name
18	24	PRENOLCH	2	Number of logical children
1A	26	PREAP	2	Number of root anchor points
1C	28	DBDPFRBN	4	Maximum relative block number (HD)
20	32	DBDPFBYT	4	Maximum bytes in prime area (HD)

3. DMANTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	PREDD1	8	Input or prime filename
8	8	PREDEV1	4	Device type
C	12	PREID	1	Data set group ID
D	13	PRENSGA	1	Number of segments in data set
E	14	PREDELTA	2	Delta scan cylinders (HD)
10	16	PRELSL	2	Length of longest segment plus prefix
12	18	PRESSL	2	Length of shortest segment plus prefix
14	20	PRELKL	2	Length of longest key
16	22	PRESKL	2	Length of shortest key
18	24	PRELRECL	2	Prime/input record length
1A	26	PREBLKSZ	2	Prime/input block size (control interval)
1C	28	PREOLREC	2	ESDS/output record length
1E	30	PREOBLKS	2	ESDS/output block size (control interval)
20	32	PREDD2	8	ESDS/output filename

4. ACB EXTENSION

See "ACB Extension - ACBXT".

5. SEGTAB LAYOUT

One of these tables exists for each segment.

Hex	Dec	Name	Ln	Description
0	0	SEGDSNO	1	Segment data set number
1	1	SEGPHYCD	1	Segment code
2	2	SEGPARPC	1	Parent segment code
3	3	SEGLEVEL	1	Segment level
4	4	SEGNOLCH	1	Number of logical children
5	5	SEGNOFLD	1	Number of fields
6	6	SEGLENG	2	Segment data length (maximum length if variable length segment)
8	8	SEGFREQ	4	Reserved
C	12	SEGSEGNM	8	Segment name

14	20	SEGFLG1	1	Prefix pointer flag
			EQU	**Meaning**
			X'80'	Counter
			X'40'	Physical twin forward
			X'20'	Physical twin backward
			X'10'	Physical parent
			X'08'	Logical twin forward
			X'04'	Logical twin backward
			X'02'	Logical parent
			X'01'	Hierarchical
15	21	SEGFLG2	1	Segment update rules
			EQU	**Meaning**
				Insert rule
			X'C0'	Logical
			X'80'	Physical
			X'40'	Virtual
				Delete rule
			X'30'	Logical
			X'20'	Physical
			X'10'	Virtual
				Replace rule
			X'0C'	Logical
			X'08'	Physical
			X'04'	Virtual
				Physical location of inserts, when no key field
			X'03'	Here (current position)
			X'02'	First
			X'01'	Last
16	22	SEGFLG3	1	
			X'08'	Parent has backward pointers to this segment
17	23	SEGFLG4	1	Number of physical children pointed to directly by this segment
18	24	SEGLCHLD	4	Offset to first LCHILD entry
1C	28	DBDSSN	2	Number of source segments
1E	30	DBDSSOFF	2	Offset to first source segment
20	32	SEGFLDTB	4	Offset to first FLDTAB
24	36	DBDSPFSZ	2	Segment prefix size
26	38	SEGLENGV	2	Minimum segment length (0 if fixed length)
28	40	Reserved	4	Reserved

2C	44	SEGPACOP	1	VL-Compression options

Name	EQU	Meaning
SEGCPRT	X'08'	Segment has compression routine
SEGTYPVL	X'04'	Segment is variable length
SEGPACIT	X'01'	Initialization exit requested for compression routine

2D	45	SEGPACRT	3	Address of compression table

6. FLDTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	FLDNAME	8	Field name
8	8	FLDSTART	2	Start position offset
A	10	FLDFLAG	1	

EQU	Meaning
X'80'	Last field for a SEGTAB
X'40'	Sequence field
X'20'	Multiple sequence fields
X'10'	Special FDB
	Field type
X'01'	Hexadecimal
X'02'	Packed
X'03'	Character
X'04'	Floating point

Hex	Dec	Name	Ln	Description
B	11	FLDLEN	1	Field length
C	12	FLDSNAME	8	Source field name
14	20	FLDSEGTB	4	Pointer to SEGTAB entry

7. EXTDBD LAYOUT

Hex	Dec	Name	Ln	Description
0	0	EXTDBNM	8	Externally referenced data base name
8	8	EXTRSVD	4	Reserved

8. LCHDTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	LCHSEGNM	8	Segment name
8	8	LCHCODE	1	

Bit	Meaning
0=0	LCHEDBD address is a EXTDBD entry
0=1	LCHEDBD address is a SEGTAB entry

			1-7	Reserved
9	9	LCHEDBD	3	Offset to EXTDBD or SEGTAB entry
C	12	LCHFLAG	1	

EQU	Meaning
X'80'	Last entry for a SEGTAB
X'40'	Reserved
X'20'	INDEX entry
X'10'	Reserved
X'08'	LP definition
X'04'	INDEX pointer
X'02'	SNGL pointer
X'01'	DBLE pointer

D	13	LCHIBYTE	1	Reserved
E	14	LCHPRDSG	2	Offset to paired segment
10	16	LCHFLDNM	8	Indexed field name

9. SORTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	DBDSORNM	8	Source segment name
8	8	DBDSSFLG	1	Source segment flag - reserved
9	9	DBDSSDBO	3	Offset to data base entry

10. INDXTAB

See "Secondary List - SEC (Codes 64, 44, 40, 24, 20, 04)".

11. DACT

See "Direct Algorithm Communication Table - DACT".

12. COMPRESSION EXIT CSECTS

See "Compression CSECT - CPAC".

APPENDIX C: PSB GENERATION

DESCRIPTION OF PSB GENERATION

PSB generation is composed of a set of DL/I macro instructions, the execution of which creates the user-specified program specification block (PSB).
The following macro instructions represent PSB generation:

Macro Instruction Name	Purpose
PCB	Allows the DL/I user to define a program communication block (PCB), one or more of which exist within a single PSB. A PCB must exist for each data base with which the associated application program PSB intends to interact. The PCB macro saves the type of PCB, associated data base name, the intended processing options on that data base, and the maximum key length within the data base. One or more PCB macros can be used in a single PSB generation. The limit is 20 PCB macros per PSB generation.
SENSEG	The SENSEG macro instruction allows the DL/I user to specify a segment within a data base to which the application program associated with this PSB is sensitive. Up to 255 SENSEG macros may follow a PCB macro.
PSBGEN	The PSBGEN macro allows the user to specify the associated application program language and the name of the PSB control block to be generated. The PSBGEN macro is the generating macro for the entire PSB control block and its internal PCB control blocks.
SENFLD	The SENFLD macro gives the DL/I user the ability to specify segment sensitivity on a field level. Up to 255 fields within a segment, and 4095 fields within a PSB may be specified.
VIRFLD	The VIRFLD macro gives the DL/I user the capability of defining fields in the user's view of a segment that do not exist in the physical view. In conjunction with the SENFLD macro, up to 255 fields per segment, and 4095 fields per PSB may be specified.

PSBGEN MACRO CALLING SEQUENCE

External Macro	Inner 1	Inner 2
PCB	DLZCKOPT DLZALPHA	
SENSEG	DLZCKOPT	
PSBGEN	DLZPCBPD	

GLOBAL SYMBOLS			MACROS							
NAME	TYPE	SIZE	DLZALPHA	DLZCKOPT	DLZPCBPD	PCB	PSBGEN	SENFLD	SENSEG	VIRFLD
DBNAME	C	255				U	R			
E	B			S		S	U	S	S	S
EXTDB	A					U	R			
FERTNA	A	4095					R	U		U
FERTNM	C	4095					R	U		U
FSLNGT	A	4095					R	U		U
FSNAME	C	4095					R	U		U
FSRTNA	A	4095					R	S		S
FSSTRT	A	4095					R	U		U
FSTYPE	A	4095					R	U		U
FSVALU	A	4095					R			S
NFER	A						R	U		U
NFLD	A						R	U	R	U
P	A			R		U	R	U	U	U
PGE	A	255		U			U			
PIO	B	255		U						
PK	A	255				S	R			
PN	C	255				U	R			
PO	C	255		S		S	R		R	
PPI	B	255		S		S	R			
PS	B	255				S	R			
PSEQ	C	255				S	R			
PSS	A	255				S	R		U	
QUITB	B		S			R		R		R
S	A			R		R	R	U	U	U
S#FLD	A						R	U		U
SEG	B					S			U	
SFC	A	500					R		S	
SFF	A							R	S	R
SLC	A	500					U		U	
SN	C	500					R		U	
SP	A	500					R		S	
SPC	A	500					R		S	
SPO	C	500		S			R		S	
SPTC	A	500					R		S	
SS	A	255				R	R	U	U	U

A = algebraic
B = binary
C = character

R = reference
S = set
U = reference/set

Figure 7-3. PSBGEN MACRO-GLOBAL Symbol Cross Reference

PSBGEN MACRO DESCRIPTIONS

DLZALPHA MACRO

A description of the DLZALPHA macro appears in Appendix B.

DLZCKOPT MACRO

	DLZCKOPT	OPT,M

This macro is called by the PCB macro or SENSEG macro to validate the PROCOPT operand. The macro generates either the PCB or the SENSEG 'PROCOPT OPERAND IS INVALID' error message. Global symbol PO or SPO is set to contain the processing option.

The operands are:

OPT specifies the PROCOPT operand as entered on the PCB or SENSEG statement

M is PCB or SENSEG message number

DLZPCBPD MACRO

This is an inner macro called by the PSBGEN macro. It generates the PL/I dope vector table if LANG=PL/I is specified in the PSBGEN statement.

PCB MACRO

This is an external macro used to define a DB PCB.

PSBGEN MACRO

This is an external macro used to terminate PSB specifications, and, if no errors have been encountered, to cause the generation of the PSB control blocks.

SENFLD MACRO

This is an external macro used to specify sensitive fields within a sensitive segment.

SENSEG MACRO

This is an external macro used to specify sensitive segments in a data base PCB.

VIRFLD MACRO

This is an external macro used to specify fields that exist in the user's view of a sensitive segment, but not in the physical view.

PSB GENERATION CONTROL BLOCK OUTPUT - PSBGEN

1. PSB - PREFIX

Hex	Dec	Ln	Description
0	0	4	Address of SEGTAB
4	4	4	Address of SORTAB
8	8	4	Address of DBREFTAB
C	12	4	Reserved
10	16	4	PST address (prefix size)
14	20	12	Reserved
20	32	1	Reserved
21	33	1	PSB code
22	34	2	PSB prefix size
24	36	2	Reserved
26	38	2	Offset to first DB PCB address
28	40	Var	Address of PCB(s) (one 4-byte address for each PCB)

2. DB PCB

Hex	Dec	Ln	Description
PL/I dope vectors precede PCB if LANG=PL/I			
0	0	8	Data base name
8	8	1	Reserved
9	9	1	Flags 04 - I,O,R,E, or A PROCOPT specified 02 - Go PROCOPT for PCB 01 - All segment processing options are either E or GO for PCB
A	10	2	Status code
C	12	4	Processing options
10	16	4	JCB address
14	20	8	Segment name feedback
1C	28	1	Position
1D	29	3	Key feedback length
20	32	2	Number of sensitive segments

22	34	2	Offset to first SENSEG
24	36	Var	Key feedback area

3. SEGTAB ENTRY

Hex	Dec	Ln	Decription
0	0	8	Segment name
8	8	4	Processing options
C	12	1	Flag 80 Last table entry 40 Field Level sensitivity for segment
D	13	3	Offset to PCB for secondary processing sequence entry
D	13	3	PCB address
10	16	2	Offset to parent segment
12	18	2	Offset to FSB list

4. DBREFTAB ENTRY

Hex	Dec	Ln	Description
0	0	12	Data base name
C	12	4	Flag byte 40 - Secondary processing sequence
D	13	3	Offset to PCB for secondary processing sequence entry

5. FLS TABLE

Hex	Dec	Ln	Description
0	0	4	FSB list address
4	4	4	FSB table address
8	8	4	Field exit routine table address
C	12	4	Field exit routine table length
10	16	4	Initial value table address
14	20	4	Initial value table length

6. FSB LIST ENTRY

Hex	Dec	Ln	Description
0	0	1	Number of FSBs for segment
1	1	3	Address of first FSB for segment

This section describes the executable processing macros that standardize some processing routines and DSECTS and lists the macros that provide the DSECTs.

DLZBLDL

This macro is used to search the core image libraries to determine if a specified load module is present.
Optionally, if the phase is present, the length of it is calculated for the caller.
The DOS/VS LOAD macro (TXT=NO) is used to obtain the directory entry information.

OPERANDS

The descriptions and valid parameters for the two keyword operands are as follows:

- PHASE — The name of the phase in the core image library.

  =(reg) — The register specified in parenthesis must point to the 8-byte name (padded with blanks if necessary).

  ='name' — The actual phase name may be specified enclosed in single quotes.

  = label — This is the label of an 8-byte field containing the phase name with any necessary blanks.

  Register 1 is the default which must be loaded with the address of the name.

- LENGTH — Specified if the caller desires the actual length of the load module to be calculated by this macro.

  =(reg) — The register specified in parenthesis will contain the length in binary of the load module as indicated in the directory entry.
Register 15 is invalid.

  = label — This is the label of a fullword in the calling program which will contain the length of the found phase on exit.

  If LENGTH is omitted, no length will be calculated.

EXIT CONDITIONS

R15 = 0 The phase was found and the length, if requested, has been returned.

R15 = 4 The phase was not found.

egisters 0 and 1 are destroyed unless specified for the length register.
ll other registers are unchanged.

LZBLKLD

his macro is used by some DOS/VS DL/I utility programs to request the initialization module o load all control blocks needed to process a specified utility PSB.
, utility PSB is built by the application control block creation and maintenance tility for every user DBD except a primary HIDAM index, logical, or HSAM.

he utilities which use this special function have 'ULU' in the first three bytes f the parameter card.
hen batch initialization determines (by utility name - either DLZURPR0, DLZURGS0, or DLZURGP0) hat the DLZBLKLD macro will be used, it does not load any control blocks.
he action modules and PST and SCD are loaded, however.
hen the utility first receives control, register 1 contains the address of the PST.

PERAND

hen the utility reaches the point where blocks are needed, he DLZBLKLD macro is executed:

```
                 [(reg)]
LZBLKLD     DMB=[label]
```

he DMB operand indicates the address of the 8-byte DMB name or which blocks are required.
ither the register number (reg) or the label of the field may be specified o indicate the address.
f this operand is omitted, register 1 is assumed to contain the address of the DMB name.

he expansion replaces the ending "D" of the DMB name with a "U".
CALL is made to ASMTDLI with the parameter list as follows:

```
       DC  A(FUNC)     Address of function
       DS  CL8         The name of the utility PSB

FUNC   DC  C'BLDB'     Function
```

EXIT CONDITIONS

fter execution of this DLZBLKID macro, register 15 contains a return ode:

R15 = 0 The blocks were loaded successfully. Register 1 contains the address of the list of PCB addresses.

R15 ≠ 0 The blocks were not loaded successfully. Register 1 contains the address of the name of the block which could not be loaded.

Any previously loaded blocks have been overloaded and new buffer pools have been allocated.

When the utility program returns to the language interface at end-of-job, a return code is expected in register 15. If register 15 is 0, normal unload processing will occur. If register 15 is non-zero, no UNLD call will be made. This return is used when no blocks have been successfully loaded.

DLZCAT

This macro is used to provide the module CATALR statement. It is updated for each release with the current version/release number. By having all modules use DLZCAT, it ensures that the CATALR statement will always contain the latest version/release number.

DLZDVCE

The DLZDVCE macro is available for the utilities to:

- Determine whether a logical unit is assigned or not.
- Determine if it is assigned to disk or tape.
- Modify the corresponding DTF.

The format of the macro is as follows:

```
DLZDVCE [MF={E|R|L|C}][,{listname|(r)}]
            [,DISKDTF={dtfname1|(r)}]
            [,MODIFY={NO|YES}]
            [,TAPEDTF={dtfname2|(r)}]
            [,FNAME={filename|(r)}]
            [,RECFM={FIXUNB|VARUNB|UNDEF|FIXBLK|VARBLK}]
            [,DEVADDR={SYSnnn|(r)}]
            [,DTFADDR={fieldname|(r)}]
            [,LNAME=listname]
            [,EOXTNT=routinename]
            [,REWIND={optionaddr|(r)}]
```

The operands have the following meaning:

MF specifies the type of code to be generated by this expansion. This allows for multiple invocations of the function without generating multiple copies of the code itself.

E generates the mainline code and, unless 'listname' is specified, a parameter list.

Note: Only one execute form of the macro is allowed for one single assembly. One, however, is required. If encountered more than once, it will be reset to R for all macros but the first one.

The entry point of the mainline routine is always DLZDTENT. This will be used by all calls generated by R type macros.

R A series of instructions to invoke the main routine, and, unless 'listname' is also specified, a parameter list will be generated. DLZDTENT is used as branch address to the main routine.

listname specifies a parameter list to be used with this execution or invocation. The list must be defined in the program with an MF=L macro or using the LNAME operand in an MF=E or MF=R macro. Listname is only valid with E or R. If listname is specified, any other operands specified will permanently override the corresponding parameters in the list. Not specifying an operand, however, will not clear the corresponding field in the list.

Register notation may be used, in which case the register must contain the address of the list.

L Only a parameter list but no code will be generated. Either the label field or the LNAME parameter (or both) can be used to assign a name to the list which can be referred to by any E of R form.

Register notation in the operands of an L form macro is not allowed, except for the DTFADDR operand.

C causes a check to be performed on all parameter lists generated during this assembly. All references to a single list are totaled and the presence of all required operands is checked. An error summary is printed. This form of the macro should be used as the last occurrence of DLZDVCE in any single assembly.

Note that passing this check error-free does not necessarily guarantee error-free execution, since the check cannot foresee the sequence in which the various DLZDVCE invocations are executed.

If the MF operand is omitted or invalid, it will default to E in the first macro encountered, and R in all other occurrences.

DISKDTF specifies the name of the disk DTF to be modified if the logical unit is assigned to a disk device. If register notation is used, the register must contain the address of the DTF.

Specifying DISKDTF=0 or a register containing zero will nullify the parameter.

If this operand is not present at execution time (after any overriding), the routine will consider assignment to a disk device as invalid.

TAPEDTF specifies the name of the tape DTF to be modified if the logical unit has been assigned to a tape device. If register notation is used, the register must contain the address of the DTF.

Specifying TAPEDTF=0 or a register containing zero will nullify the parameter.

If this operand is not present at execution time (after any overriding), the routine will consider an assignment to tape as invalid.

If MF=E or R without listname was specified, either DISKDTF or TAPEDTF or both must be specified.

MODIFY specifies whether or not the selected DTF is to be modified accordingly or not. MODIFY=YES is the default. If MODIFY=NO was specified, and a valid device type was found, register 15 will have a negative return code, indicating that no modification has been done.

FNAME specifies the filename to be moved into the appropriate DTF. If not present at execution time, the DTF field is not changed. For register notation, the register must point to a seven-byte field containing the file name.

Specifying a register pointing to a hex zero string will nullify the parameter.

RECFM specifies the record format of the file. One of the values shown must be specified. Omission or invalid specification defaults to VARBLK.

DEVADDR specifies the logical unit number to be tested. It must be in the form SYSnnn, where nnn is 000 to 243, or in register notation, in which case the register must contain the unit number as a binary number in the same range.

This parameter is required if MF=E or R without listname was specified.

DLZER

This macro is used in module DLZLBLM0 to specify a message. Code is also generated to support selection by message id.

OPERANDS

```
DLZER  ID=nnn,TEXT=text[,LAST=NO ]
                       [      YES]
```

ID = one to three digit message number ('NNN' in 'DLZNNNI').

TEXT = message text. Text is a string of parameters enclosed in left and right parentheses. Each parameter is either a character string enclosed in quotes; or a set of two values, the first indicating a length to be reserved for a field to be dynamically inserted, and the second the register that will contain the address of the field to be inserted (not register R1 or R15).

(The message number is generated by the macro and need not be included in the text.)

TEXT=('THIS IS ',3,R5,' AN EXAMPLE ',8,R4)

LAST = 'YES' indicates that no further messages exist. This is a special message. The contents of the specified register will be converted to BCD and stored in the field for each insert field.

This macro also generates the code to select and format a message. Preceding the first call of DLZER, code must be supplied to establish addressability and equates must be supplied for 'R1' and 'R14'.

INPUT:

'R1' should contain the message code in binary format.
'R14' must contain the address of the routine to process a message once it has been located and formatted.

OUTPUT:

'R1' will contain a pointer to a two byte field containing the length of the message. The message directly follows this two byte field. The message is formatted as:

ODLZNNNI TEXTTEXTTEXTTEXTTEXTTEXTTEXTTEXT

DLZDLIST

This macro is used to build the parameter list for the IPCS Dump Hooks. This parameter list is required by the DLZIDUMP macro.

DLZID

label	DLZID	MOD=mod-name ,VR=version-number ,PTF=ptf-number

This macro is used to provide module identification for all DL/I modules. It sets the global, &DLZMOD, which contains the module name, and the global, &DLZVR, which contains the version, release, and PTF number. These globals can then be used by other macros or referenced by the module itself.

In addition to the constants generated to include the version/release level of when the module was last changed as entered by the caller, another set of constants is automatically included for the current version/release number of DL/I. This macro contains a Base Code Indicator which identifies who last assembled or updated the module.

The operands are:

label	1-8 character label (mod-name)
mod-name	Name of module, If omitted, the present CSECT name is used. This name appears in an 8-byte character constant.
version-number	1-3 digit version/release number. If omitted, this field is set to zeros. Zeros are concatinated to the number specified to insure three digits.

	This field is divided into three 1-byte character constants.
ptf-number	1-digit number of the latest PTF applied. If omitted, this field appears as a 1-byte character constant.

DLZIDUMP

This macro is used to call the IDUMP facility to provide a dump in the format acceptable for analysis by IPCS Service Routines. If the conditions for the dump are satisfied, the IDUMP macro is executed. If IDUMP has not been activated, the alternate dump path is taken.

DLZIPOST

This macro is used by DL/I to post ECBs in an online environment.

There are no operands. Register 2 must contain the address of the ECB to be posted. Bit 0 of byte 2 is set on.

DLZIWAIT

This macro is used by DL/I to communicate with an IWAIT routine (DLZIWAIT) to wait until an ECB is unposted.

There are no operands. The PST must be addressable and register 2 must contain the address of the ECB that is to be waited for. The caller must have provided a USING SCD,15. Registers 14 and 15 are used to branch to the DLZIWAIT routine.

DLZTRCAL

This macro is used by action modules to invoke the tracing facility. Refer to DL/I DOS/VS Diagnostic Guide for a description of this macro.

DLZREL

This macro defines a macro variable, &DLZVER, and sets it to indicate the current version of DL/I.

DLZTRPRM

This macro is called by the DLZTRACE macro to parse parameter lists. It is similar to the DLZXPARM macro of DBDGEN (see "DLZXPARM Macro" in Chapter 6). In addition to the interface described for DLZXPARM, the length of each parameter list member is passed to the caller in the GBLA fields $PLEN(25).

DLZMPCPT

The master partition controller (MCP) partition table is used to pass control information when processing batch partition application programs under MPS (Multiple Partition Support). The MPC partition table resides in the transaction work area.

DLZTWAB

This macro provides the mapping for the BPC batch partition control information for the DL/I task termination routine under MPS (Multiple Partition Support). This information resides in the BPC's task transaction work area.

DLZXTAB

This macro provides the mapping for the XECBTAB macro DEFINE, DELETE, and CHECK options under MPS (Multiple Partition Support).

DLZXCB1

This macro maps the DLZXCBn1 and the data that follows it. It is used to check data under MPS (Multiple Partition Support).

MACROS USED TO CREATE DSECTS FOR DL/I SYSTEM CONTROL BLOCKS

The following macros are used to generate DSECTS for the DL/I control blocks:

DLZBFFR
DLZBFPL
DLZDDIR
DLZIDLI
DLZPDIR
DLZPPST
DLZPSIL
DLZPST
DLZSCD.

Macros used only by utilities to generate DSECTs:

DLZCKPT
DLZDTF
DLZIDBD
DLZRECO
DLZUCHDR
DLZUCOLD
DLZUCREC
DLZUCUMC
DLZUDHDR
DLZURGUF
DLZURHDR

DLZUSTAT
DLZTRENT.

Miscellaneous macros:

DLZDLIST	Creates parameter list for DLZIDUMP macro
DLZDLP	Log record DSECTs and declarations
DLZHDS0	Work area for DLZDHDS0
DLZIDUMP	IPCS dump hook macro
DLZQUATE	Register equates
DLZSBIF	Work area for DLZDBH00
DIZUMSG	Messages for utilities
DLZWA	Work area used by DLZDLD00
DLZXMTWA	Work area used by DLZDXMT0.

DL/I QUEUING FACILITY MACROS

Four macros are available to request processing of a specific function by the queuing facility module (DLZQUEF0). The functions that can be requested and the macros that can be used are:

Function Requested	Macro Used
Enqueue	DLZENQ
Verify	DLZVER
Dequeue	DLZDEQ
Purge	DLZPUR

The functions are described in Section 3 of this manual. The format of each macro and the description of the operands is as follows:

Formats

DLZENQ [PST=r1][,LEV={RO|UPD|EXC}][,ID=r2][,FLAG=x'hh']

DLZVER [PST=r1][,LEV={RO|UPD|EXC}][,ID=r2][,FLAG=x'hh']

DLZDEQ [PST=r1][,LEV={RO|UPD|EXC}][,ID=r2][,FLAG=x'hh']

DLZPUR [PST=r1][,FLAG=x'hh']

Operands

PST=r1

specifies the symbolic (or absolute) name of a register containing the address of the PST. It this operand is omitted, register one is assumed.

LEV={RO|UPD|EXC}

specifies the level involved; RO = read only, UPD = update, and EXC = exclusive. If omitted, it is assumed the PSTQLEV field in the PST is set with the proper code.

ID=r2

specifies the symbolic (or absolute) name of a register containing the address of the seven byte field containing the resource ID. If omitted, it is assumed the address is stored in the PSTWRK2 field in the PST.

FLAG=x'hh'

specifies the byte value that is 'OR'ed into the return code for those tasks currently waiting for the resource.

Index

DL/I DOS/VS Logic Manual
LY12-5016-6

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. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

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

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	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.)

Please use pressure sensitive or other gummed tape to seal this form.

LY12-5016-6

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 10 ENDICOTT, N.Y. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

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 ______________________

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

DL/I DOS/VS Logic Manual, Volume 1 (File No. S370/4300-50) Printed in U.S.A. LY12-5016-6

LY12-5016-6

DL/I DOS/VS Logic Manual, Volume 1 (File No. S370/4300-50) Printed in U.S.A. LY12-5016-6

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601