Licensed Material — Property of IBM
LY12-5016-7
File No. S370/4300-50

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

Program Product

Program Number 5746-XX1

Eighth Edition (December 1983)

This edition, LY12-5016-7, is a major revision of LY12-5016-6. It applies to Version 1, Release 7 (Version 1.7) of Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1 and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information contained 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, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see page iii.

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

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

or

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

Summary of Changes

Summary of Changes
for DL/I Version 1.7
LY12-5016-7

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

Interactive Utility Generation
This provides an interactive facility to assist with the generation of utility job streams.

IMF Enhancements
The Interactive Macro Facility (IMF) has been enhanced to support the DL/I Documentation Aid facility.

Documentation Aid
This provides an ease-of-use facility to document DL/I definitions that can be accessed directly by ISQL.

IMF Adaptation to ISPF
The Interactive Mactor Facility (IMF) now runs on the Interactive System Productivity Facility (ISPF) program product, 5668-960.

Variable Length Index Source Segment
This allows an Index Source Segment of a DL/I secondary index to be variable in length.

Utilities Operational Improvements
Various modifications have been made to the DL/I utilities to permit tape rewind options, omission of partition dump, suppression of informational messages to SYSLOG, selectable creation of secondary indexes, automatic open of WORKFIL, and change accumulation data base specification.

HLPI Support of Boolean Operations
The Boolean AND and OR operators can now be used with the WHERE clause.

MPS Restart
MPS batch environment jobs can be restarted after a failure. This supports the use of VSE Checkpoint/Restart with the DL/I Checkpoint.

Key Feedback
KEYFEEDBACK and FEEDBACKLEN can be specified with GET commands to retrieve the key feedback area form the PCB.

Online Initialization Messages
Messages have been added to provide status information during online initialization. These messages include information concerning DL/I logging status, DL/I version currently being run, and the program isolation status.

Summary of Changes
DL/I Version 1.6
LY12-5016-6

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 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 Print 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.

Separate Index Reorganization
With this function, the user can now reorganize an index data base separately by using the HISAM unload and reload 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.

Summary of Changes
DL/I Version 1.5
LY12-5016-5

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.

Preface

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* 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.

Related Publications

- *DL/I DOS/VS General Information Manual,* GH20-1246
- *DL/I DOS/VS Application Programming: CALL and RQDLI Interfaces,* 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 Interactive Resource Definition and Utilities,* SH24-5029
- *DL/I DOS/VS Recovery/Restart Guide,* SH24-5030
- *DL/I DOS/VS Application Programming: High Level Programming Interface,* SH24-5009
- *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 VSE and VSE/VSAM messages and return codes:

- *VSE/Advanced Functions Messages and Codes,* SC33-6098
- *VSE/Advanced Functions Application Programming: User's Guide,* SC24-5210
- *VSE/Advanced Functions Application Programming: Reference,* SC24-5211
- *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/DOS/VS Installation and Operations Guide,* SC33-0070
- *CICS/VS Customization Guide,* SC33-0131
- *CICS/VS Performance Guide,* SC33-0134
- *CICS/VS Resource Definition Guide,* SC33-0149
- *CICS/VS Application Programmer's Reference Manual (Macro Level),* SC33-0079
- *CICS/VS System/Application Design Guide,* SC33-0068

Contents

Figures

Section 1. Introduction

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 on page 1-2. 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 on page 1-3):

- 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.

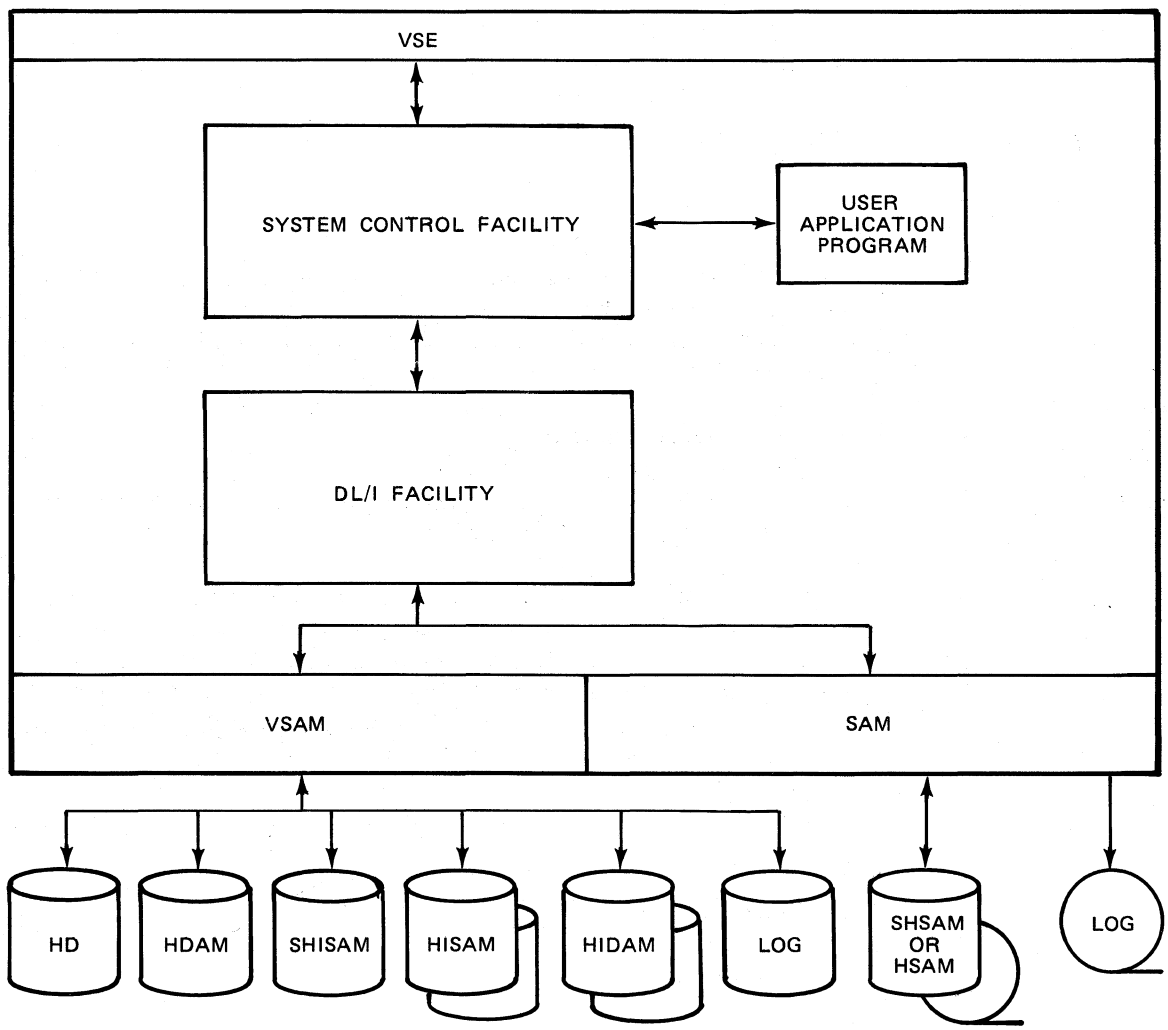


Figure 1-1. Elements of a DL/I DOS/VS Batch Partition

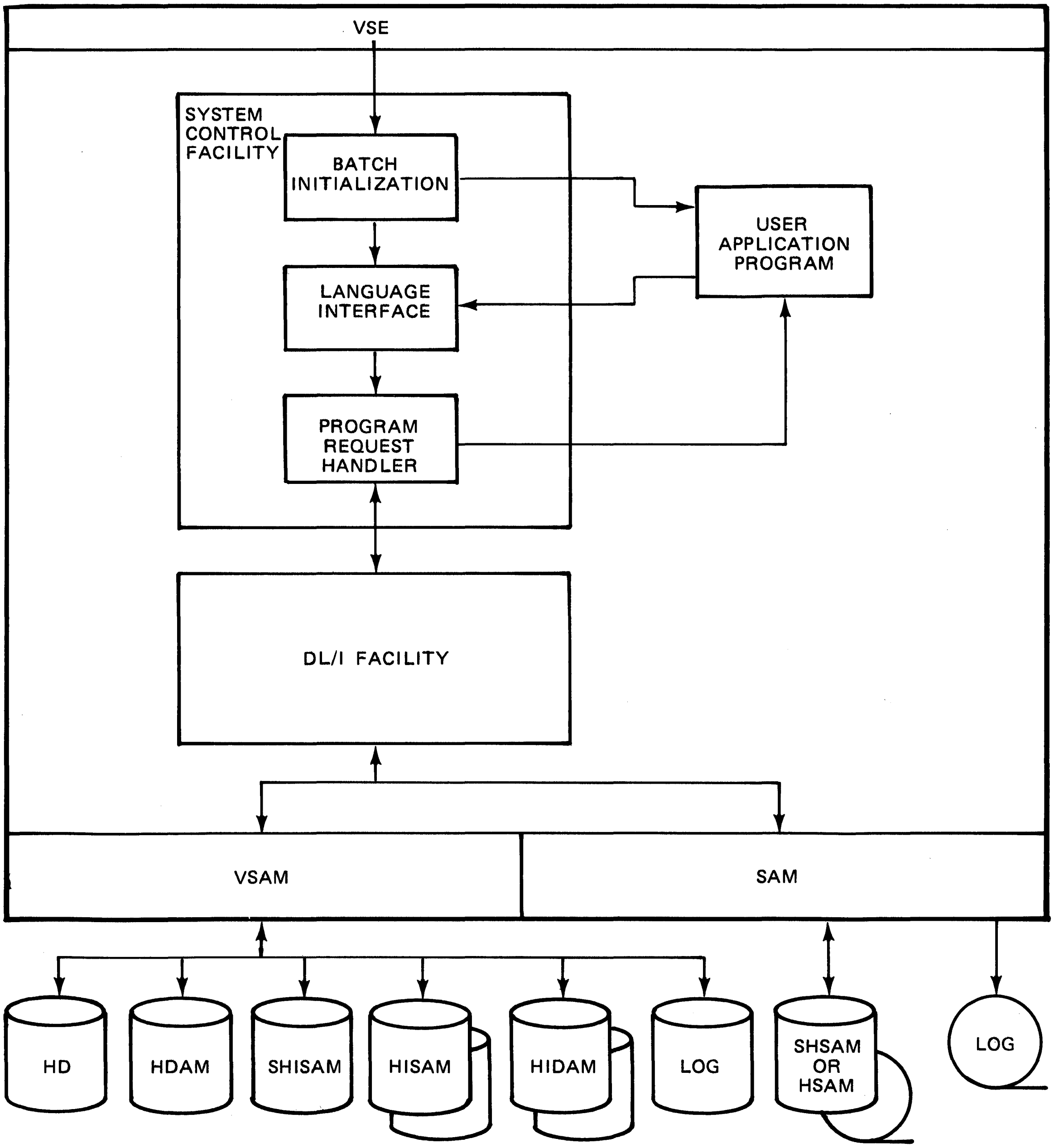


Figure 1-2. System Control Facility Relationships

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 completion and return of DL/I resources after the application program has issued a CICS/VS synchronization point (SYNCPOINT) command or has completed processing.

When the user application program issues a DL/I call, control passes to a language interface module, the EXEC interface program (if HLPI is used), 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.

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 on page 1-7). 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.

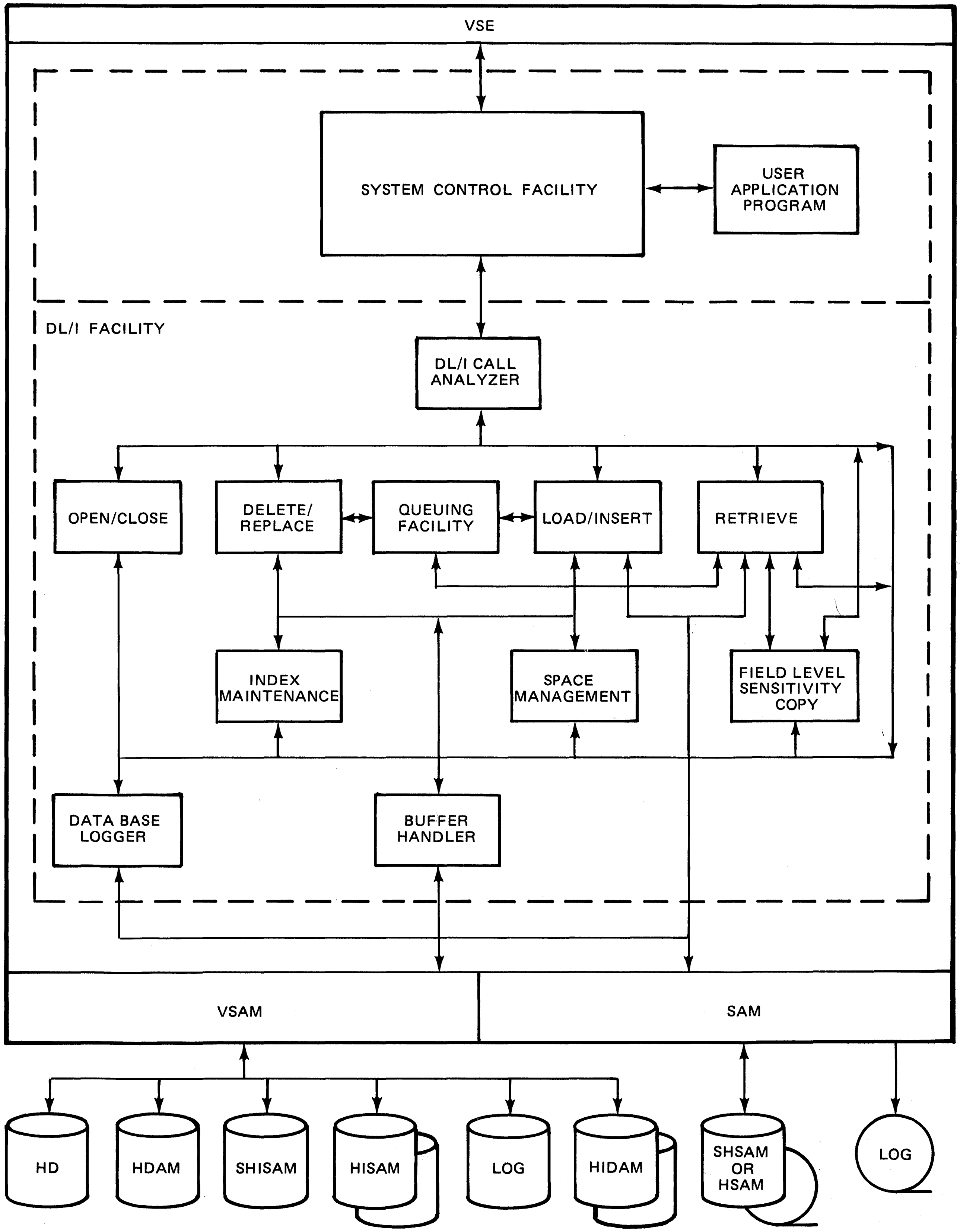


Figure 1-3. DL/I Facility Relationships

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.

- ISQL Extract Defines Utility: this utility creates and stores an ISQL routine composed of ISQL Extract Define commands from data previously gathered and stored in tables with the DL/I Documentation Aid. Once the routine is created, it can be run under ISQL to define the necessary DL/I information to the EXTRACT facility of SQL/DS.

- Problem determination: this includes the log print utility which enables you to print the contents of DL/I log files to help you recover from system failures, and the trace print utility which enables you to print trace entries from tape or disk input files which are created by the DL/I trace facility.

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) Logic Manual, Volume 2,* 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 parameter 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 VSE 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)
- Load and initialize PSBs and DMBs
- Allocate and format buffers

- Branch to application program control module (DLZPCC00)

Entry Interface - DLZRRC10

Receives control through branch from DLZRRC00

Entry Register Contents

R7 Address of ERRORMSG
R10 Entry point address

Exit Interface - Parameter Analysis

Passes control by fall through to DLZRRA00

Exit 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
- 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
- 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 VSE 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

| *DLZBPJRA - DL/I COBOL Preinitialization Module*

This module is linked with batch COBOL programs to call ILBDSETO. An entry card with the name CBLCALLA is required in the link edit job step of the batch COBOL program.

DLZBPJRA does the following:

- Branches to IBLDSETO, the COBOL routine entry point
- Then exits by branching to the application program entry point (DLITCBL).

Interface

This module interfaces with the following:

ILBDSETO - COBOL routine entry point Application program.

Control Blocks

None.

Normal Entry Point

The only entry point to this module is CBLCALLA

Entry Register Contents

R14 Linkage register
R15 Base register

Exit Register Contents

R14 Linkage register
R15 Application program entry point

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
DFHEI1 Common entry point

Entry Register Contents

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, or DFHEI01.

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 or DFHEI01 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
DFHEI01	Common entry point

Entry Register Contents

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:

R2-12	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 VSE 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 VSE 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 VSE

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 VSE 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 VSE.
- 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 DLZSTRB0

Entry Register Contents

R1 PST address
R13 Current register savearea address

DLZSTRO0 - 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:

DLZDLA00 - 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.

Entry Register Contents

R1 PST address
R13 Current register savearea address

Online DL/I Processor Modules

Before attempting to use this section, 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 perform the following functions in a CICS/VS-DL/I environment:

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/VS 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.

The online nucleus, which is link edited into a VSE core image library, consists of the following DL/I nucleus modules and tables:

- module DLZODP
- module DLZEIPO0
- module DLZSTRO0
- module DLZCOM00
- module DLZLOC00
- module DLZODPEX
- ACT (Application Control Table)
- SCD (System Contents Directory)
- DFHDLIAL (CICS/VS-DL/I Interface Address List)
- SCD Extension
- PDIR (PSB Directory)

- RPDIR (Remote PSB Directory) only if a Remote PSB is defined
- PPST (PST Prefix Table)
- PDCA (Problem Determination Control Area)
- EIPL (EXEC Interface Parameter List)
- module DLZMMSGT
- module DLZFTDP0
- module DLZISC00 (only if a Remote PSB is defined)

The application control table (ACT) is used by DL/I online at CICS/VS 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 program is permitted to use DL/I. It is also used by DL/I default scheduling to acquire a PSB to use if none was explicitly specified in the PSB scheduling call or command.

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 on page 3-16.

A. A(SCDSTART)
4

A. Address of the System Contents Directory (SCD)

B.	C.	D.	
4	8	2	bytes

A

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

Generated from:
DLZACT TYPE = PROGRAM

Program entry '1'

E.	F.	G.	H.	•••	H.	
8	1	2	2		2	bytes

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

I.	
4	bytes

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

A

Generated from:
DLZACT TYPE = BUFFER

HDBFR entry (subpool '1')

J.	K.	•••	K.	L.	
2	8		8	2	bytes

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

HDBFR entry (subpool 'n')

		•••		

HSBFR entry (DBD #1)

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

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

HSBFR entry (BDB #n)

R.	
4	bytes

R. 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. 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 - DL/I Task Scheduling

DL/I Scheduling

A DL/I call or HLPI SCHEDULE command initiates DL/I PSB scheduling. The call function code is 'PCB' and the call contains the name of the PSB to be used. 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 (by specifying a PSB name of 'SYSTEMDL'), the password is tested against the one generated in the nucleus via the DLZACT macro and the system 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 schedule another concurrent task. The SCD maximum task indicator is tested. If it is on, the task cannot be scheduled and the SCD suspended task counter is incremented by one. 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.

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 local/remote call router routine, DLZCOM00. If a remote PSB is to be scheduled, control is passed to the remote scheduling subroutine, DLZISC00, which transfers the request to the remote system. If a local PSB is to be scheduled, control is passed to the local PSB scheduling routine, DLZLOC00.

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 has scheduled the same PSB. If the task cannot be scheduled, a CICS/VS SUSPEND is issued. If not in use, but retrieve sensitive only, the in-use indicator is set and control is passed to PSB initialization. If neither of the above is true, the PSB segment intent list entry is scanned. If program isolation is not active and the PSB is not retrieve only sensitive, the PSB segment intent list entry is 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 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 PSB are initialized and a duplicate PSB directory entry is created. The level table(s) are then reset and control passed to the PSB initialization subroutine of DLZLOC00.

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/VS 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 update sensitivity, a call is made to the DL/I data base logger module (DLZRDBL0) or CICS/VS journal interface routine (DLZDRBL1) to write an application program scheduling record (X'08'). Control is then returned to the application program.

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 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 returns the addresses of local copies of the PCBs acquired in the remote system.

DLZPRH00 - 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 program DLZEIPO0. This module converts translator-generated calls into standard DL/I calls for each HLPI command.

Language Interface Module

The language interface module is link-edited with each application program. 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 VSE 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. The CICS/VS stubs pass control to the CICS/VS EXEC interface program and then the DL/I EXEC interface program before control is given to the program request handler.

Program Request Handler

The program request handler validates the DL/I call parameters. For scheduling calls, control is then 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.

System calls 'CMXT', 'STRT', 'STOP', 'TSTR', and 'TSTP' are processed by the system call routine, PROCSYS in DLZODP, after being routed there by the program request handler.

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 is issued.
- Upon return, checks are made to ensure the logger and formatted dump routines are not busy.
- All registers are restored.
- Control is returned to the calling module via register 14.

DLZODP01 - Task Termination

DL/I task termination is entered from 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 issues a DL/I 'TERM' call. This routine is responsible for purging 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 suspended for the maximum task limit.

Task Termination

Task termination writes a termination trace entry in the CICS/VS trace table. Then it determines whether this task is scheduled to use a remote PSB. If it is, control is given to the remote termination call subroutine, DLZISC02. This subroutine issues a CICS/VS sync point call which causes the remote mirror program, DFHMIR, which processes 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 after being suspended during task scheduling. In this case, the suspended count accumlator 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 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/VS 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 suspended for scheduling because of segment intent conflict are resumed.

Next the PSB directory entry is released. 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.

If the system call interface is active, the DDIR entries for the terminating PSB are checked to see if the system task is waiting to close this data base. If it is and the use count of the DMB is now zero, the system task is posted to continue processing.

System Resource Allocation

This routine is responsible for resuming tasks which are suspended due to the maximum task limit. 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. 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).
- If in use, the DL/I log DTF is closed via a VSE CLOSE macro.
- If MPS is active, control is given to the MPS system termination routine in DLZMPC00.
- Control is returned to CICS/VS.
- DL/I system termination is re-entered by CICS/VS system termination.

- 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.
- Control is returned to CICS/VS.

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:

- A switch is set to avoid closing data bases on invocation of DLZODP02.
- Control is returned to CICS/VS which later calls DLZODP02.

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 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. Instead, a CICS/VS deferred work element is created to ensure that DL/I will be given control again after additional CICS/VS sync point processing has been completed.

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 VSE 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 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 text is generated using DLZMMSGT and logged to destination CSMT via CICS/VS Transient Data Management and to the operator's console.
- Control is returned to the calling routine.

If an error occurs while writing to transient data, an ABEND indicator is placed in the TCA and control is returned to the calling routine.

DLZOVSEX - VSAM EXCP EXIT Processor

This routine prevents the CICS/VS partition from being put into a WAIT state due to DL/I initiated VSAM I/O. It does this by issuing a CICS/VS WAIT instead of letting VSAM issue a VSE WAIT. 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 DLZBNUC0 in batch and DLZODP02 in online.

Exit Interface

This module returns control to caller.

Entry Register Contents

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

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 is called by DLZODP06.

Exit Interface

This module returns control to DLZODP06.

Entry Register Contents

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

| DLZCBDP0 - DL/I Formatted Control Block Program

This module is a collection of subroutines that build a list of addresses that are used to print DL/I control blocks.

Entry Interface - DLZCBDP0

This module interfaces with DLZFSDP0 and DLZFTDP0, the formatted dump programs.

Exit Interface

This module returns control to the caller.

Entry Register Contents

R11 Address of desired subroutine
R14 Caller return address
R15 Module entry point address

Control blocks

ACB	PDIR
ACT	PPST
BFFR	PSDB
BFPL	PST
DDIR	RIB
DIB	RPCB
DMB	RPDIR
EIPL	SBIF

FDB	SCD
FERT	SDIB
FLD	SSA
FSB	SSAP
PATH	SSAX
PCB	UIB
PDCA	

DL/I Facility Modules

DLZDLA00 - Call Analyzer

The call analyzer module is used for initiation of all data base calls. It receives control from the DL/I common data base call routine (DLZCOM01) in the CICS/VS-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/VS-DL/I partition.

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 functions 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.

Control Blocks - DLZDLA00

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

Register Contents

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

Interfaces - DLZDLA00

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

The function of module DLZDLOC0 is to open and close the DL/I data bases in either the CICS/VS online control region or the batch partition. VSE 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/VS 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

Bit	Value	Meaning
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 or replace.
`...x ....`	Reserved
`.... 1...`	This variable-length segment has its data separated from the prefix.
`.... .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
DLZRDBL0
DLZDXMT0
DLZQUEF0

Control Blocks - DLZDLD00

- Delete workspace prefix
- Delete work area.

Entry Register Contents

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

Exit Register Contents

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

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
DLZDBHO0	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 phase is responsible for retrieval of all segments, independent of physical data base organization. When an application program requests the retrieval of a segment, this phase (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 phase 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 phase.

It is the responsibility of the retrieve phase to "deblock" segments within the block. Once the desired segment is located, the retrieve phase 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 phase to the buffer handler results only in the address of the desired data being returned to the retrieve phase. 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 phase 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 phase performs "positioning" functions for all segment insertion. In this case, the retrieve phase 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 phase 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 phase 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 phase consists of 60 subroutines, a main entry routine (DLZDLR0), a main exit routine (DLZDLR1), and a general linkage and maintenance support routine (DLZRLNKD), 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 phase is supplied as eight modules. 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 first module, DLZDLRA0, also contains the routines DLZDLR0 and DLZDLR1. The distribution of the subroutines within the CSECTs contained in the modules DLZDLRA0 to DLZDLRG0 is arbitrary and can be changed at will, necessitating only that the affected modules 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 phase interfaces with the following modules:

DLZDDLE0	Load/insert
DLZDBH00	Buffer handler
DLZQUEF0	Queuing facility

Entry Register Contents 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 over the data set.

The routines in module DLZDHDS0 perform the following functions:

DLZDHDS0	contains the entry point for the combined module. It saves registers, initializes the work words in the PST, and branches to the appropriate module.
GETSPACE	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 section 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.
FRESPACE	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.

SRCHBLK	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.
SRCHPOOL	searches the DL/I buffer pool for a block in the range passed to it. If one is found, module SRCHBLK is called to search it. If the block is rejected, the search continues to the end of the pool, and control is returned to GETSPACE. 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 SRCHBLK and may be used without being requested from the buffer handler. In a DL/I online environment, the buffer is passed to SRCHBLK. If the request can be satisfied from it, the buffer is then requested from DLZDBH00 and again passed to SRCHBLK for actual alteration.
SRCHBTMP	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 GETSPACE. If all bits are zero, PSTNOSPC is returned to GETSPACE. 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.
CALCSRLM	calculates search limits for GETSPACE. 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.
BITMPLOC	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.
BITMPON BITMPOFF	turns the appropriate bit ON or OFF according to the entry point involved. The log is also called to reflect the change.
DEVCHARI	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.
FORMAT	formats a new control interval. Builds initial FSEs and root anchor points.

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

R15 0 - No errors occurred
4 - Error has occurred; check PSTRTCDE

PSTRTCDE 4 - RBN is beyond the end of the data set
8 - I/O error
C - No space in data set
1C - Insufficient space in buffer pool

For other return codes, see “PST - Partition Specification Table” in “ Section 5: Data Areas”.

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 contains the control interval is marked as altered by this specific user.

Symbol Function	Hex Function	Description
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

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. The following fields are used by the ENQ/DEQ routine:

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

The ENQ/DEQ routines use the field BFPLNQW1 in the buffer pool prefix as work space.

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.

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

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

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.

b. Post the waiting PSTs and unchain the current PST.
c. 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 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 extension queue	When a new block past the VSAM SEOF is created, the task must wait until processing of previous tasks that created new blocks have been processed.

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. Functions 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 PSTOFFST	Core address of desired segment 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.		

A

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 PSTBUFFA	Address of work area containing the logical record Prefix Address	PSTBLKNM	RBA of the record added to the ESDS as calculated by VSAM

B

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 PSTDATA	RBA of the logical record retrieved Core address of record
PSTSTLBG			PSTBYTNM PSTDATA	RBA of the logical record retrieved Core address of record
PSTGETNX			PSTBYTNM PSTDATA	RBA of the logical record retrieved Core address of record
PSTPUTKY	PSTDATA PSTBUFFA	Address of work area containing the logical record Prefix address		
PSTMSPUT	PSTDATA PSTBUFFA	Address of work area containing the logical record 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
PSTNUMR	Number of reads done during this call
PSTNUMWT	Number of writes done during this call
PSTCLRWT	Bit 0 - This caller waited during request Bits 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.

Return Code Function	Hex Function	Description
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 separate ly 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 place d 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, 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 DLZODP.

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.

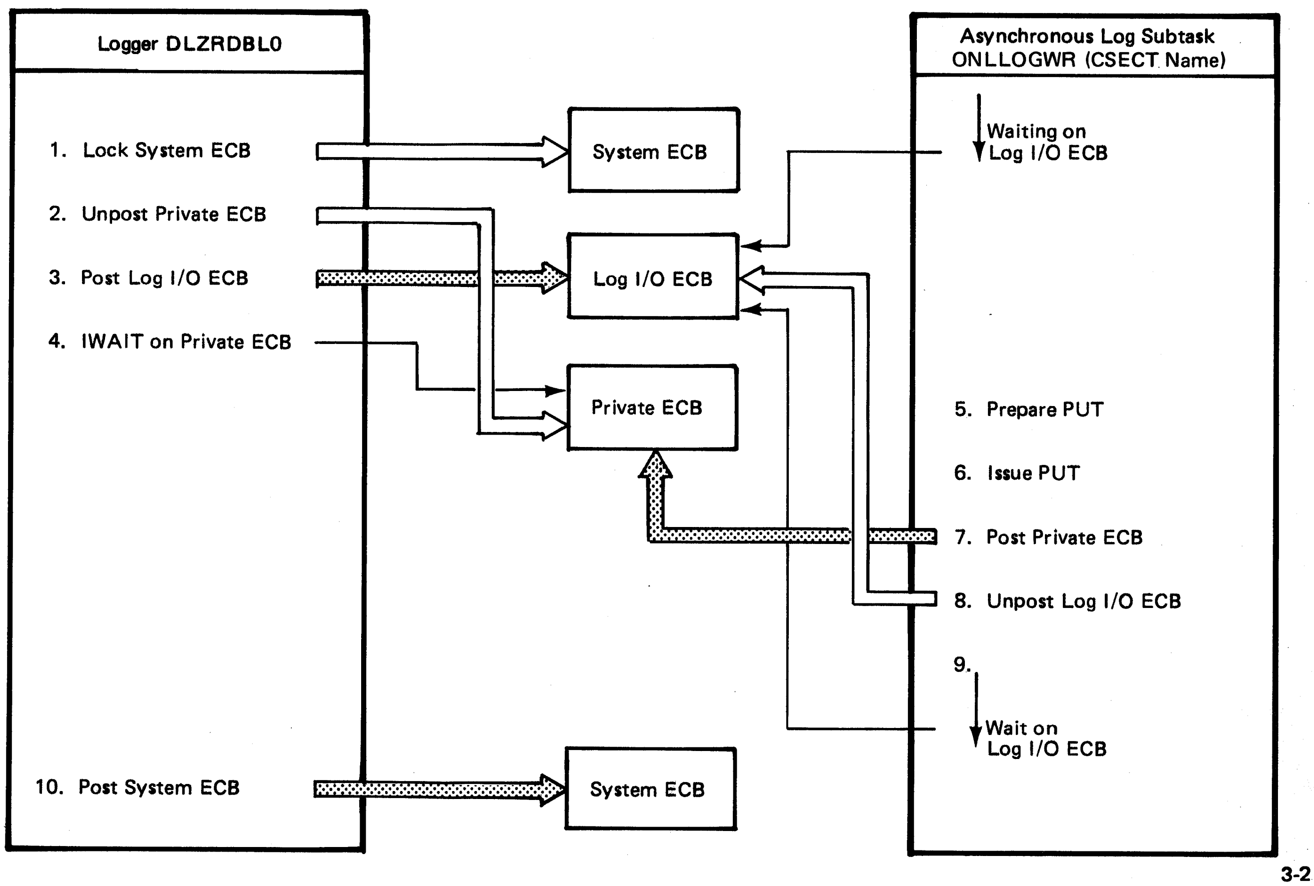


Figure 3-2. Online Log Block Put Operation

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

	DLZODP PRH Scheduler Routine Terminate Routine Message Routine IWAIT Routine EXCPAD Routine	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/VS Journal Logger

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

This is possible because CICS/VS 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/VS journal records and CICS/VS journal blocks are illustrated in Section 5.

If the user requests logging by CICS/VS 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/VS.

- 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/VS, 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/VS wait is issued before control is passed to any action module.

When log information is written by using CICS/VS 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/VS ECN) 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

DLZQUEF0 - 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 DLZQUEF0 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 3-3 on page 3-59, 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 WAIT 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.

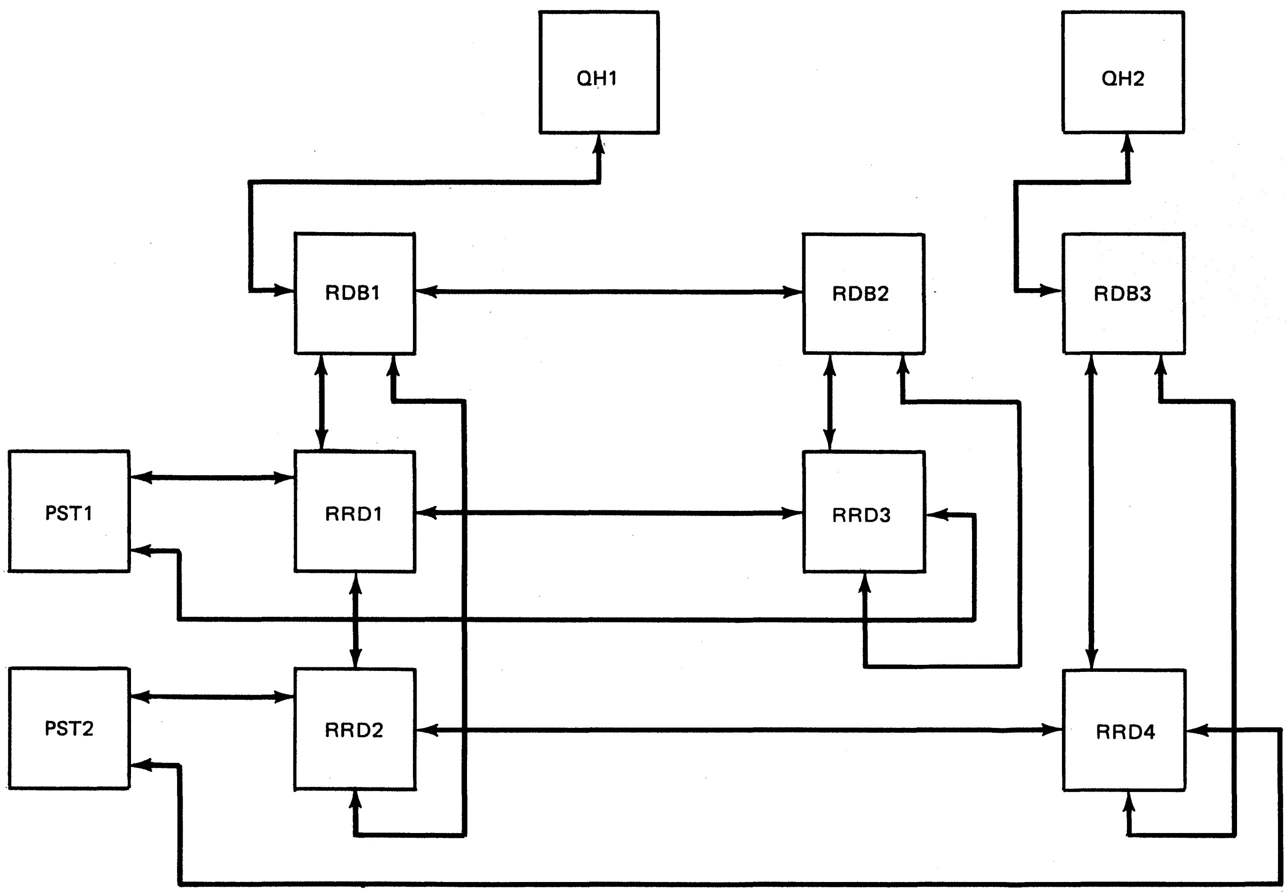


Figure 3-3. Enqueue/Dequeue Control Block Relationships

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 CSECTs: 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

Entry Register Contents

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 functions 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.

- Perform some management of CICS/VS temporary storage queue (TSQ) entries for MPS batch jobs using MPS Restart.

- 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
DCA	Dispatch Control Area
DLZTSQE	Temporary Storage Queue Entry
DLZXCB1	Batch Communication 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
DFHSP	TYPE=USER
DFHTS	TYPE=GETQ
DFHTS	TYPE=PUTQ
DFHTS	TYPE=PURGE
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.
- Update temporary storage queue entry for MPS Restart if the batch partition issues a combined checkpoint.
- 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	Batch Communication Area
DLZTSQE	Temporary Storage Queue Entry

Register Contents

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

Macros Used

DFHKC	TYPE=WAIT
DFHPC	TYPE=RETURN
DFHPC	TYPE=ABEND
DFHPC	TYPE=SETXIT
DFHTS	TYPE=PUTQ
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, DLZXCBnl, 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
XCBl	XECB DLZXCBnl and data following it
DTFs for	SYSLST, SYSLOG, and SYSIPT
STXIT AB	Savearea
STXIT PC	Savearea

XECBs	DLZXCB02, DLZXCBn2, DLZXCBn3
PDIR	PSB Directory
PSB	Program Specification Block
PCB	Program Control Block
DLZEIPL	HLPI Control Block

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	
GETIME	
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
DLZEIPL	HLPI Control Block
PCB	Program Control Block

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:
  Rl4 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 l4

Macros Used

PUT

DLZMSTP0 - 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

DLZMPUR0 - Purge Temporary Storage Transaction

This module is invoked by the user via a specific transaction code (CSDP) to purge the temporary stroage queue (TSQ) used by MPS Restart.

If MPS is active when this module is invoked, then a flag is set behind the stop partition XECB (DLZXCB01) which signals to the master partition controller (MPC) that the TSQ is to be purged, and the stop partition XECB is posted. (It serves a dual purpose in this way.)

If MPS is not active, then the TSQ is purged by this module.

Control Blocks Addressed

TDOA	CICS/VS Transient Data Output Area
CSA	CICS/VS Common System Area
TCA	CICS/VS Task Control Area
DLZXCB01	Stop Partition XECB

Entry Register Contents

R12	TCA address
R13	CSA address
R14	Routine entry point

Macros Used.

DFHPC	TYPE=RETURN
DFHPC	TYPE=ABEND
DFHSC	TYPE=GETMAIN
DFHSC	TYPE=FREEMAIN

DFHTD	TYPE=PUT
DFHTS	TYPE=PURGE
DFHWTO	
EXTRACT	
MAPBDY	
XECBTAB	TYPE=CHECK

Data Base Recovery Utilities

DLZBACK0 - Batch Backout Interface

The batch backout interface module reads and validates any 'LI' 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.

Entry Register Contents

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.

DLZBACM0 - Message writing

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:

1. DLZBACK0 in a batch environment, or
2. DFHDBP in an online environment during dynamic transaction backout, or
3. DFHTBP in an online environment during CICS/VS emergency restart.

with a log record to process. They call 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 when using the DL/I logger to record the changes made to the data base. DLZRDBL1 is called when using the CICS/VS journal 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 the calling module.

Entry Register Contents and Control Blocks

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
DLZRDBL1	Called to log backout modifications to data base (online)

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	PSTOCDMB + PSTOCOPN
SCDCWRK	address of normal log record work area

DLZDBH00

R1 address of PST

PSTBLKNM	RBN if HD ESDS
PSTACBNO	1
PSTDMBNO	1
STBYTNM	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/DLZRDBL1

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

Exit Register Contents

All registers are restored with the exception of register 15 which contains a return code.

Error Codes and Handling - DLZRDBC0

All error codes are passed 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.

Entry Register Contents

R1 pointer to fullword containing address of PCB

Exit Register Contents

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.

Entry Register Contents

R1 pointer to recovery common area.

Exit Register Contents

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 Copy

The data base data set image copy utility module DLZUDMP0 is executed as a standard VSE 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 a n 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.

| ***DLZCUMM0 - 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 depending on parameters passed to it, and prints a message to the SYSLST and SYSLOG devices.

Normal Entry Point

| The only entry to this module is DLZCUMM0.

Entry Conditions

This module is entered to output all error messages.

Entry Register Contents

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.)

Exit Register Contents

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.

Exit Register Contents

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 to sort and/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.

Exit Register Contents

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.

Entry Register Contents

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.

Exit Register Contents

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.

Entry Register Contents

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.

Exit Register Contents

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.

Data Base Reorganization Utilities

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 depend ent 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.

Error Codes and Handling - DLZURUL0

All error codes are in the form of 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 3-4.

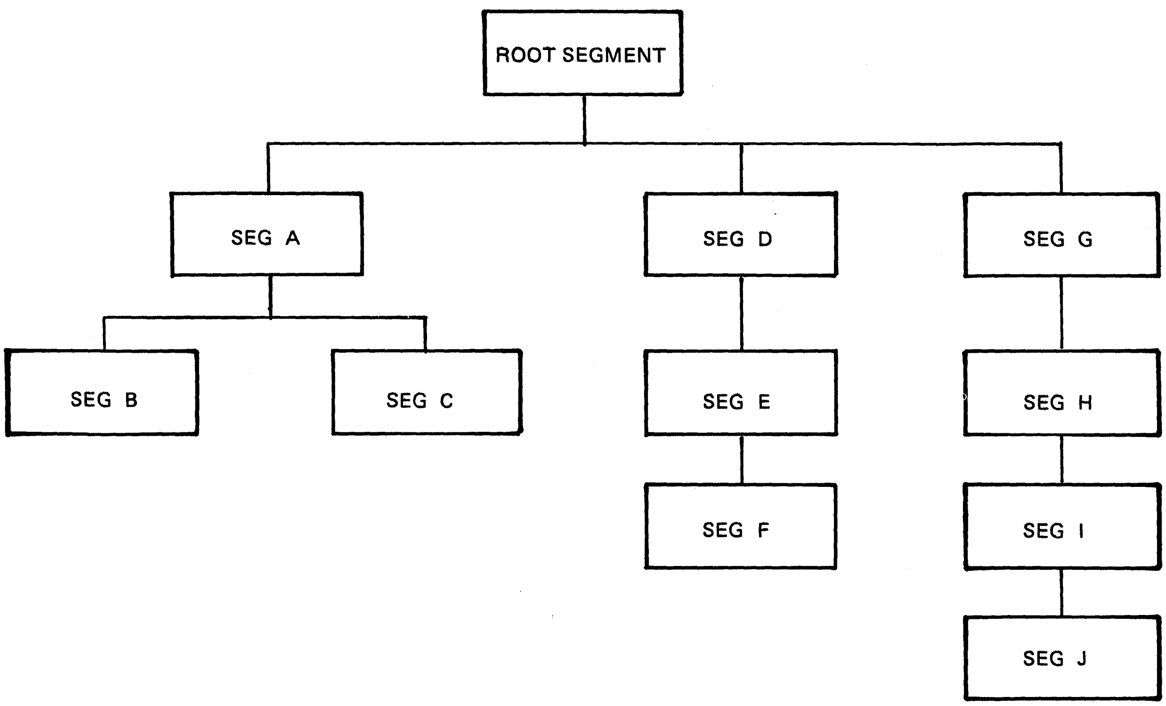


Figure 3-4. HISAM Data Base with One Root Segment

The input for the HISAM Reorganization Unload Utility appears as shown in Figure 3-5 on page 3-82.

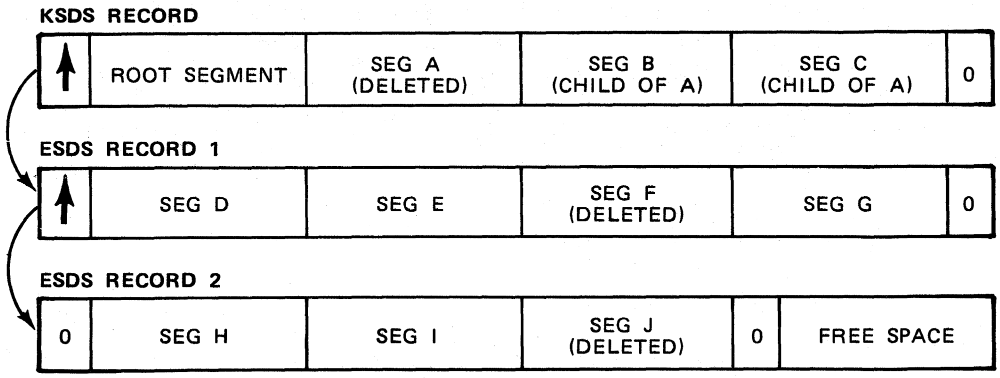


Figure 3-5. Input for HISAM Reorganization Unload Utility

Given this input, the HISAM Reorganization Unload Utility provides the output shown in Figure 3-6.

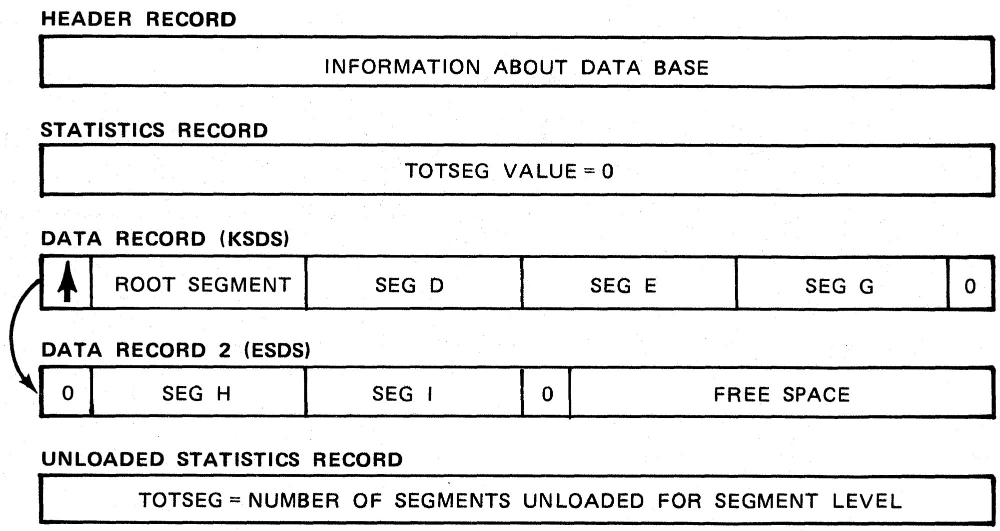


Figure 3-6. HISAM Reorganization Unload Utility Output

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). 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.

5. 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.

Entry Register Contents

Standard register conventions are used for linkage to this module.

Exit Register Contents

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

- COMAREA - common area

Normal Entry Point

The only entry point to this module is DLZPRABC.

Entry Register Contents

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

Exit Register Contents

All registers are the same as 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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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

- 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 DLZPRURC.

Entry Register Contents

R13	Save area address
R14	Return address
R15	Entry point address

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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
DLZFRWFM	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.

Entry Register Contents

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

Exit Register Contents

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.

Entry Register Contents

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

Exit Register Contents

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 Initialization

This module has two logical functions. An initialization routine which processes the HLPI EXEC DLI INIT call and a routine that loads in DLZEIPB1, and passes control to it.

All CICS/VS application programs which issue DL/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 in DLZEIPB0.

The language interface module (DLZLIPLI or DLZLICBL) calls DLZEIPB0, which first checks to see if this is a DL/I HLPI INIT call. If it is, it checks to see if storage has been acquired for the UDIB/SDIB. Following this acquisition of storage, DLZEIPB0 returns to the caller.

If this is not an initialization call, DLZEIPB0 checks the integrity of SDIB, issuing DLZ037I if the SDIB has been inadvertently destroyed. If the SDIB is fine, DLZEIPB0 determines if DLZEIPB1 has been loaded and loads it if not. DLZEIPB0 then branches to entry point DLZEIP0 in DLZEIPB1.

Interface

This module interfaces with the following:

DLZEIPB1	HLPI DL/I Batch/MPS EXEC interface
DLZLICBL	COBOL language interface module
DLZLIPLI	PL/I language interface module

Control Blocks

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

Normal Entry Point

The only entry point to this module is DLZEIPI.

Entry Register Contents

R1	HLPI parameter list address
R2	System DIB pointer (If storage has been acquired for System DIB)
R13	Caller's register save area address
R14	Caller's return address
R15	Entry point of DLZEIPB0

Exit Register Contents

R1	HLPI parameter list address
R2	System DIB pointer

R3 ARG0 parameter list address
R6 EIP parameter list address
R8 User DIB address
R13 Caller's register save area
R14 Caller's return address

DLZEIPB1 - Batch/MPS EXEC Interface Program

This module handles all DL/I BATCH HLPI calls except the translator generated INIT call. It translates HLPI EXEC DLI statements into DL/I Call parameter lists. DLZEIPO in DLZEIPO0 carries out the same function as this module.

There are differences between DLZEIPB1 and DLZEIPO0 because of the different environments. First, DLZEIPB1 uses 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. Thirdly, DLZEIPO0 calls the online program request handler DLZPRHO0.

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

On entry, DLZEIPB1 determines if the call is a data base call. If so, it does the following:

Checks to see if Key feedback is requested
Determines which PCB in the PCB list to use
Checks to see if a data transfer is to take place
Checks to see if segment name has been specified
Checks to see if the call is a replace call with a previous get path call
Acquires storage for the SSA
Checks to see if the call is a valid insert call
Establishes the correct command codes
Builds field qualifications
Sets up the correct SSA for use by the DL/I Program Request Handler

After DLZEIPB1 finishes building the SSA, it calculates the required I/O area size and builds a common I/O area for path calls. Then DLZEIPB1 passes control to the correct Program Request Handler.

If the call is not a data base call, DLZEIPB1 does the following:

Terminates task if it is a SCHEDULE call (Invalid in a batch environment)
Terminates task if it is a TERMINATE call (Invalid in a batch environment)
Builds the checkpoint call if it is a CHECKPOINT call and passes control to the correct Program Request Handler.

On return from the Program Request Handler, DLZEIPB1 does the following:

Initializes the UDIB with information passed by DL/I in the PCB
Checks the status code
Moves Key Feedback information to user area if requested
Transfers data to user segment I/O areas
Returns to Caller

Interface

This module interfaces with the following:

DLZEIPB0	HLPI DL/I Batch/MPS EXEC interface initialization
DLZBNUC0	Batch nucleus (Routine DLZPRHB0 - Batch Program Request Handler)
DLZMPI00	MPS Batch (Routine DLZMPRH - MPS Batch Program Request Handler)
DLZMMSGT	Message Module for error and informational messages

Control Blocks

ARG0	ARG0 parameter list
DBPCB	DL/I Program Control Block
DIB	User DL/I interface block
EIPL	EIP parameter list
HLPIL	HLPI parameter list
PATH	Path header control block
SDIB	System DL/I interface block
SSA	Segment Search Argument control block
SSAP	SSA path call appendage block
SSAX	SSA extension block

Normal Entry Point

The only entry point to this module is DLZEIP0.

Entry Register Contents

R1	HLPI parameter list address
R2	System DIB pointer
R3	ARG0 parameter list address
R6	EIP parameter list address
R13	Caller's register save area address
R14	Caller's return address
R15	Entry point of DLZEIPB1

Exit Register Contents

R14	Caller's return address

DLZEIPO0 - DL/I Online EXEC Interface Program

DLZEIPO0 handles all DL/I ONLINE HLPI calls. It 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 counterparts DLZEIPB0 and DLZEIPB1. DLZEIPO0 builds data base calls to the online program request handler (DLZPRHO0) according to HLPI command syntax.

On entry, DLZEIPO0 determines if the call is the initialization call. If it is, it acquires storage for the SDIB/UDIB.

If it is not the initialization call, DLZEIPO0 goes to the routine DLZEIP0 where it verifies the integrity of the system DIB. If the call is a data base call, it does the following:

Checks to see if Key feedback is requested.
Determines which PCB in the PCB list to use
Checks to see if a data transfer is to take place
Checks to see if segment name has been specified
Checks to see if the call is a replace call with a previous get path call
Acquires storage for the SSA
Checks to see if the call is a valid insert call
Establishes the correct command codes
Builds field qualifications
Sets up the correct SSA for use by the DL/I Program Request Handler

After DLZEIPO0 finishes building the SSA, it calculates the required I/O area size and builds a common I/O area for path calls. Then DLZEIPO0 passes control to DLZPRHO0 (Online Program Request Handler).

If the call is not a database call, DLZEIPO0 does the following:

Builds a SCHEDULE call if requested
Builds a TERMINATE call if requested
Builds the CHECKPOINT call if requested
DLZEIPO0 then passes control to the Program Request Handler.

After returning from the Program Request Handler, DLZEIPO0 does the following for data base calls:

Initializes the UDIB with information passed by DL/I in the PCB
Checks the status code
Moves Key Feedback information to user area if requested
Transfers data to user segment I/O areas if necessary
Returns to DFHEIP

After returning from the Program Request Handler, DLZEIPO0 does the following for the SCHEDULE call:

Counts the number of PCBs
Acquires storage for the path header control blocks
Returns to DFHEIP

Interface

This module interfaces with the following:

DFHEIP	CICS/VS EXEC Interface Program
DLZPRHO0	Online Program Request Handler
DLZMMSGT	Message module for error and informational messages

Control Blocks

ARG0	ARG0 parameter list
DBPCB	DL/I Program Control Block
DIB	User DL/I interface block
EIPL	EIP parameter list
HLPIL	HLPI parameter list
PATH	Path header control block
SDIB	System DL/I Interface block
SSA	Segment Search Argument control block
SSAP	SSA path call appendage block
SSAX	SSA extension block
UIB	User Interface Block

Normal Entry Point

The only entry point to this module is DLZEIPI

Entry Register Contents

R1	HLPI parameter list address
R7	CICS/VS CSA address
R13	Register save area address
R14	Caller's return address
R15	Entry point of DLZEIPO0

Exit Register Contents

R14	Caller's return 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 VSE 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 programs. The program accepts control card input to determine what functions are required. For the DL/I-SQL/DS user, if requested, all DBD and PSB data definitions will be collected and stored into the DL/I Documentation Aid SQL/DS tables.

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 blocks builder module DLZDLBL0 to 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
USERIDCB
PSBSQLIO

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

1. 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.)

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

   ```
   L 15,=V(INSRCH)
   A 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.

3. 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.

4. 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.

Entry Register Contents - 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, DLZDLBPP, DLZDLBL1, DLZDLBDP, DLZDLBL2, DLZDLBL3 - ACB BUILDER

The four modules, (DLZDLBL0, DLZDLBL1, DLZDLBL2, and DLZDLBL3), are 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 VSE core image library.

The two modules, (DLZDLBPP and DLZDLBDP), are responsible for collecting all the data definitions for the DBDs and PSBs and storing them into the DL/I Documentation Aid SQL/DS tables.

The first module, DLZDLBL0, loads the specified PSB and calls module DLZDLBPP, if the USERID parameter was specified on the BUILD statement.

Module DLZDLBPP creates the PSBBASICDATA, PSBPCBDATA, PSBSEGMENTDATA, and PSBFIELDDATA records from information retrieved from the DL/I PSB control blocks and inserts them into the appropriate DL/I Documentation Aid SQL/DS tables. After all processing is completed for the PSB, control is returned to DLZDLBL0.

Module DLZDLBL0 then 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 calls module DLZDLBDP, if the USERID parameter was specified on the BUILD statement.

Module DLZDLBDP creates the DBDBASICDATA, DBDACCESSDATA, DBDSEGMENTDATA, DBDLCHILDDATA, and DBDFIELDDATA records from information retrieved from the DL/I DBD control blocks and inserts them into the appropriate DL/I Documentation Aid SQL/DS tables. After all processing is completed for the DBD, control is returned to DLZDLBL1.

Module DLZDLBL1 then 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

Entry Register Contents

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

Parameter List

PST address
USERIDCB address

Exit Register Contents

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.

Entry Register Contents

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.

Exit Register Contents

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

R15 = 0 Valid return
R15 ≠ 0 Errors encountered

Data Base Logical Relationship Utilities

DLZURPR0 - 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 - DLZURPR0

- 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 - DLZURPR0

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 SCDDSEH0)

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)

PSTWRK1 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.

Entry Register Contents

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

Exit Register Contents

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

Extract Defines Utility

DLZEXDFP - Extract Defines Utility

The Extract Defines utility creates an ISQL Routine containing EXTRACT DEFINE commands. This utility uses the information about the the DL/I databases in the DL/I Documentation Aid SQL/DS tables created by the Application Control Blocks Creation and Maintenance Utility.

The logic flow is as follows: Each control statement is syntax-checked and processed individually based on the PCB. The DEFINE commands are created and inserted into the SQL/DS ROUTINE table in the following order:

```
DEF PCB NAME=xxxxxxxx,PSB=(xxxxxxx,nnn),PROC=xxxxxxxx
DEF SEGMENT NAME=xxxxxxxx,PCB=xxxxxxxx,PARENT=0
DEF FIELD NAME=(xxxxxxxx,{SEQ|NOSEQ}),SEGM=xxxxxxxx,PCB=xxxxxxxx,
    TYPE=x,START=nnnnn,BYTES=nnn
                                  .
                                  .
(and all other DEFINE FIELD commands associated with this
 segment)
                                  .
                                  .
DEF SEGMENT NAME=xxxxxxxx,PCB=xxxxxxxx,PARENT=xxxxxxxx
DEF FIELD NAME=(xxxxxxxx,{SEQ|NOSEQ}),SEGM=xxxxxxxx,PCB=xxxxxxxx,
    TYPE=x,START=nnnnn,BYTES=nnn
                                  .
                                  .
(and all other DEFINE FIELD commands associated with this
 segment)
                                  .
                                  .
```

The process for creating DEFINE SEGMENT and FIELD commands is repeated until all SEGMENTs have been processed for this PCB.

```
DEF PCB NAME=xxxxxxxx,PSB=(xxxxxxx,nnn),PROC=xxxxxxxx
                                  .
                                  .
(and all DEFINE SEGMENT and FIELD commands associated with
 this PCB)
                                  .
                                  .
```

The process for creating SEGMENT and FIELD commands is repeated until all PCBs have been processed for this PSB.

Blocks and Tables - DLZEXDFP

EXINOUT	DEFINE COMMAND build area
DLZEXWCB	DEFINE work control block
DSQLCA	SQL Communication Area
SQLDSECT	SQL/DS Interface Control Block

Interfaces - DLZEXDFP

This module interfaces with the following modules:

DLZEXDFM	Called to format pre-built messages
ARIPRDID	SQL/DS interface module

Register Contents

R5	SQLCA Address
R6	EXINOUT Address
R7	DLZEXWCB Address
R8	Error Information
R9	SQLDSECT Address

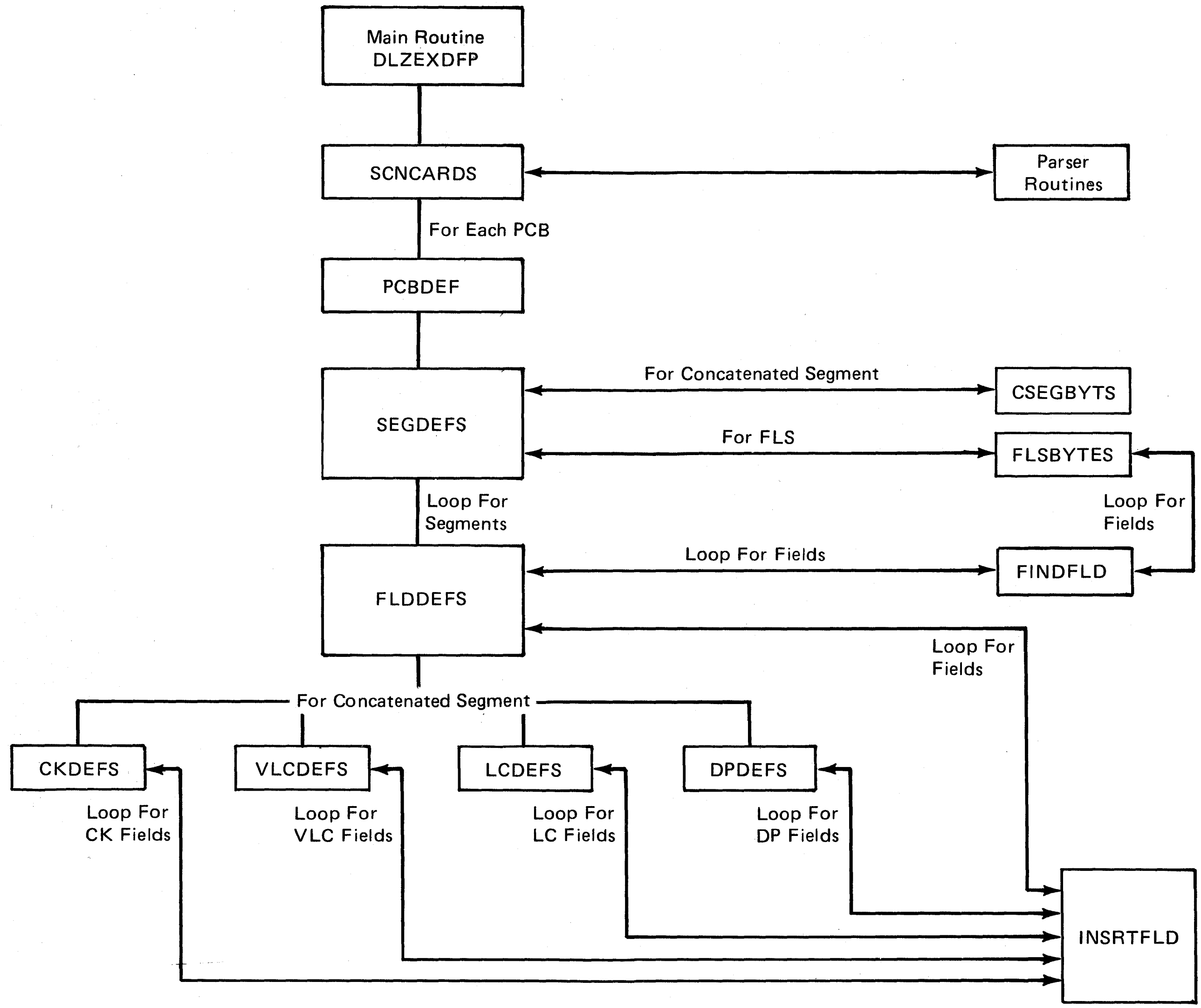


Figure 3-7. Extract Defines Utility Overview Flow

General Flow - DLZEXDFP

DLZEXDFP: Mainline routine that does the utility intitialization.

SCNCARDS: Scans the control statements and retreives all information.

PARSER Routines:

PROPSBNM - Parses the PSBNAME parameter and saves the psbname and pcbnumber. If the pcbnumber is omitted, it defaults to 1.

PROPCBNM - Parses the PSBNAME parameter and saves the pcbname.

PRODLIPR - Parses the DLIPROC parameter and saves the DL/I procedure name.

PROREP - Parses the REPLACE parameter and sets the replace flag accordingly.

PROUSRID - Parses the USERID parameter and saves the user-id and password.

PCBDEF: This routine selects the information required from the DL/I PSB or DBD SQL/DS tables and builds the DEFINE PCB command. If REPLACE was specified, the old SQL/DS routine by the pcbname being processed in the ROUTINE table is deleted (if there) and replaced by these new DEFINE commands. If REPLACE was not specified, a check is made to see if the routine already exists, and if so, an error message is issued. If no routine exists for the specified pcbname, the new commands are inserted.

SEGDEFS: This routine builds the DEFINE SEGMENT commands and inserts them into the ROUTINE table. It determines if the segment is a concatenated segment, a virtual logical child, and if the PSB is field level sensitive. If the PSB is field level sensitive, routine FLSBYTES is called to calculate the length of the segment. If the segment is a concatenated segment, routine CSEGBYTS is called to calculate the length of the segment. If neither of the above, the length of the segment is obtained from the physical DBD.

CSEGBYTS: This routine obtains the length of the logical child segment and calculates the length of the destination parent's concatenated key. If the current segment is a virtual logical child, it calculates the logical parent's concatenated key. It also obtains the length of the destination parent's segment.

For a logical child segment:

Segment length = logical child segment + logical parent segment lengths

For a virtual logical child segment:

Segment length = logical child segment + logical parent concatenated key + physical parent concatenated key + physical parent segment lengths

FLSBYTES: This routine calculates the segment length for a field level sensitive segment. It also calculates the starting position of the sensitive field and updates the PSBFIELDDATA table entry for this field with the length and the datatype of the field.

FLDDEFS: This routine builds the DEFINE FIELD commands and inserts them into the ROUTINE table. If the segment is a concatenated segment, routines CKDEFS, VLCDEFS, LCDEFS, and DPDEFS are called to obtain the field information. If the segment is field level sensitive, field information is obtained from the PSBFIELDDATA table. For segments other than field level sensitive or concatenated segments, the field information is obtained from the physical DBD definition.

FINDFLD: This routine locates the sensitive field and returns the length and datatype to the caller. If the segment is not a concatenated segment, the information is obtained from the physical field definition. If the segment is a concatenated segment, the order of search is:

If segment is a virtual logical child, first select from the virtual logical child. If not found or if segment is not a virtual logical child, select from the logical child. If still not found, select from the destination parent.

CKDEFS: This routine obtains the information to build the DEFINE FIELD commands for the concatenated key.

VLCDEFS: This routine obtains the information to build the DEFINE FIELD commands for the virtual logical child segment fields.

LCDEFS: This routine obtains the information to build the DEFINE FIELD commands for the logical child segment fields.

DPDEFS: This routine obtains the information to build the DEFINE FIELD commands for the destination parent segment fields.

INSRTFLD: This routine inserts the DEFINE FIELD commands into the SQL/DS ROUTINE table.

DLZEXDFM - Extract Defines Utility Error Message Handler

This module is used to contain, select, and format error messages for the Extract Defines utility. 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 specific registers. Control is returned to DLZEXDFP who in turn calls routine 'PRTMSG' to print the message.

Entry Register Contents - DLZEXDFM

R1 Message number
R2 Message Buffer Address
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.

Section 4. Directory

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 level 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 statement when assembling this module, for example, END DLZBEGIN.

Note: The figure number shown after the descriptive 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,* LY24-5215.

System Control Modules

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** Batch Initialization ** (See Figure 2-3.)					
DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRCST
	*ERRORMSG				
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
	DLZRDR				
	DLZCONSL				
	DLZRRC10				
	*DLZ2MSGT			DLZ2MSGT	
	*DLZ3MSGT			DLZ3MSGT	
	*DLZ4MSGT			DLZ4MSGT	
	*DLZRRA00				
	*DLZPCC00				
	*DLZDBLM0				
	*LOADDMBS				
	*PCBROUT				
	*DLZCPI00				
	*DMBLOADR				
** Batch Nucleus ** (See Figure 2-4.)					
DLZBNUC0	SCDCSECT	DLZBNUC0	DLZBNUC0	DLZBNUC0	
	SCDSTART				
	*DLZIWAIT				
	*DLZPRHB0				
	*DLZABEND				
	DLZEIPI	DLZEIPB0	DLZEIPB0	DLZEIPB0	
** Online Initialization ** (See Figure 2-5.)					
DFHDIDL	DLIOLI00	DLZOLI00	DLZOLI00	DLZOLI00	
	*DLZCPI00				
	INITLODR				
	DLIOLI10				

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** Online Nucleus ** (See Figure 2-6.)					
	DLZEIPO0				
DLZNUCxx	DLZODP	DLZODP	DLZODP	DLZNUCXX	
	DLZODP00				
	DLZSCHDL				
	DLZODP03				
	DLZODP02			DLZODP02	
	DLZODP04			DLZODP04	
	DLZODP07			DLZODP07	
	DLZODP06				
	DLZODP01			DLZODP01	
	DLZTKTRM				
	DLZTKBAD				
	DLZODP05			DLZODP05	
	DLZPRHO0			DLZPRHO0	
	DLZABNDO				
	DLZOLT00			DLZOLT00	
	DLZOLT02				
	DLZOLT01				
	DLZOWAIT			DLZOWAIT	
	DLZOVSEX			DLZOVSEX	
	DLZERMSG			DLZERMSG	
	DLZODP10			DLZODP10	
	DLZODP11			DLZODP11	
	DLZEIPI	DLZEIPO0	DLZEIPO0	DLZEIPO0	
	DLZSTRO0	DLZSTRO0	DLZSTRO0	DLZSTRO0	
	DLZCOM00	DLZCOM00	DLZCOM00	DLZCOM00	
	DLZCOM01			DLZCOM01	
	DLZLOC00	DLZLOC00	DLZLOC00	DLZLOC00	
	DLZLOC01			DLZLOC01	
	DLZODPEX			DLZODPEX	
	DLZNUC				
	SCDSTART				
	DLZEIPL				
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
	*DLZ2MSGT			DLZ2MSGT	
	*DLZ3MSGT			DLZ3MSGT	
	*DLZ4MSGT			DLZ4MSGT	
	DLZFTDP0	DLZFTDP0	DLZFTDP0	DLZFTDP0	
	DLZISC00	DLZISC00	DLZISC00	DLZISC00	
	DLZISC01			DLZISC01	
	DLZISC02			DLZISC02	
	DLZISC03			DLZISC03	

Note: xx is the suffix specified during ACT generation.

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** DL/I Online System Termination ** (See Figure 2-7.)					
DLZSTP00	DLZSTP00	DLZSTP00	DLZSTP00	DLZSTP00	

DL/I Facility Modules

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** 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				
	DLZRLNKD	DLZRLNKD	DLZRLNKD	DLZRLNKD	
	DLZPOST0	DLZDLRG0	DLZDLRG0	DLZDLRG0	
	DLZSKPG0				
	DLZSKPS0				
	DLZSKPD0				
	DLZSKPE0				
	DLZHIDA0	DLZDLRE0	DLZDLRE0	DLZDLRE0	
	DLZHDAM0				
	DLZHISA0				
	DLZSTLA0				
	DLZSTLG0				
	DLZUPDT0				
	DLZKDTE0				
	DLZPCHK0				
	DLZSSA0	DLZDLRC0	DLZDLRC0	DLZDLRC0	
	DLZTAG0				
	DLZLTW0				
	DLZNOSS0				
	DLZISRT0	DLZDLRF0	DLZDLRF0	DLZDLRF0	
	DLZVLRT0				
	DLZAREJ0				
	DLZVLCH0				
	DLZXDFT0				
	DLZHSAM0				
	DLZALTS0				
	DLZFLD0				
	DLZLOGR0	DLZDLRD0	DLZDLRD0	DLZDLRD0	

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
	DLZRETK0				
	DLZRETI0				
	DLZKDRK0				
	DLZKDTL0				
	DLZUPDC0				
	DLZUPDL0				
	DLZAPST0				
	DLZYENT0				
	DLZYSTC0				
	DLZYEND0				
	DLZDEQ0				
	DLZLPSL0				
** Load/Insert ** (See Figure 2-10.)					
DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	SVA
	HDROUTIN				
	VLROUTIN				
	HSROUTIN				
** Delete/Replace ** (See Figure 2-11.)					
DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	SVA
	DLZDLDA0				
	DLZDLDD0				
	DLZDLDR0				
** Index Maintenance ** (See Figure 2-12.)					
DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	SVA
** HD Space Management ** (See Figure 2-13.)					
DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	SVA
	*GETSPACE				
	*CALCSRLM				
	*SRCHPOOL				
	*SRCHBTMP				
	*FRESPACE				
	*SRCHBLK				
	*FORMAT				
	*BITMPLOC				
	*BITMPOFF				
	*BITMPON				
	*DEVCHARI				
	DFSRLO30				
	SNAPDCB				
	SNPSW				
	SNPCNT				
** Open/Close ** (See Figure 2-14.)					
DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** DB Buffer Handler ** (See Figure 2-15.)					
DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	SVA
	*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	
	*MRKEMPT				
	*PGUSR				
** DB Logger ** (See Figure 2-16.)					
DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	
	DLZIDBL0				
	IOFILA1				
	LOGOUT				
	LSCDADDR				
	IJFUZZZN	IJFUZZZN			
	IJFUZZZZ				
	IJ2Nnnnn	DLZRDBL0	DLZRDBL0		
	ONLLOGWR				
(DLZRDBL0)	SAVE				
	PRIVECB				
** CICS/VS Journal Logger ** (See Figure 2-17.)					
DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	
	DLZIDBL0				
** Queuing Facility ** (See Figure 2-23.)					
DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	
DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	
** Field Level Sensitivity Copy ** (See Figure 2-41.)					
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
** MPS 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 Initialization ** (See Figure 2-21.)					
DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	
	*DLZMPRH				
	*DLZMINIT				
	*DLZMTERM				
	*DLZ2MSGT			DLZ2MSGT	
	*DLZ3MSGT			DLZ3MSGT	
	*DLZ4MSGT			DLZ4MSGT	
	*DLZMMSG				
	*DLZMABND				
	DLZCONSL				
	DLZDIMOD				
	DLZEIPI	DLZEIPB0	DLZEIPB0	DLZEIPB0	
	DLZMMSGT	DLZMMSGT	DLZMMSGT	DLZMMSGT	
** Stop Transaction ** (See Figure 2-22.1.)					
DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	
** Purge Temporary Storage Transaction ** (See Figure 2-22)					
DLZMPUR0	DLZMPUR0	DLZMPUR0	DLZMPUR0	DLZMPUR0	

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	
DLZPRNT					
DLZSLOG					
PRNTAREA					
	IJ2Mnnnn	DLZUDMP0	DLZUDMP0		
	DLZDMPM0	DLZDMPM0	DLZDMPM0		
	IJJFCBZD	IJJFCBZD			
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
** 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				
	DLZCUMM0	DLZCUMM0	DLZCUMM0	DLZCUMM0	
	IJFSZZWN	IJFSZZWN			
	IJFVZZWZ				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJ2Mnnnn	DLZUCUM0	DLZUCUM0		
	IJFUZZZZ	IJFUZZZZ			
** DB Data Set Recovery ** (See Figure 2-27.)					
DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	
	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			
		IJGQICZZ			

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** DB Change Backout ** (See Figure 2-28.)					
DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0	
	READAREA				
	DLZPRNT				
	DLZSLOG				
	DLZRDBC0	DLZRDBC0	DLZRDBC0	DLZRDBC0	
	DLZBACM0	DLZBACM0	DLZBACM0	DLZBACM0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	IJFUBZZZ	IJFUBZZZ			
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJ2Mnnnn	DLZBACK0	DLZBACK0		
** Log Print Utility ** (See Figure 2-40.)					
DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGPE
	DLZLGPCN				
	DLZLGPMT				
	DLZLPCC0	DLZLPCC0	DLZLPCC0	DLZLPCC0	
	DLZLGPM0	DLZLGPM0	DLZLGPM0		
	IJJFCBID	IJJFCBID			
	IJJFCIID				
	IJFUZZZN	IJFUZZZN			

Data Base Reorganization Utilities

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** 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			
** 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

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** ACB Creation ** (See Figure 2-33.)					
DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	
	PRTMSG				
	DLZDLBL0	DLZDLBL0	DLZDLBL0	DLZDLBL0	
	PSBPASS				
	DLZDLBL4				
	DLZDLBPP	DLZDLBPP	DLZDLBPP	DLZDLBPP	
	DLZDLBL1	DLZDLBL1	DLZDLBL1	DLZDLBL1	
	DLZDLBDP	DLZDLBDP	DLZDLBDP	DLZDLBDP	
	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

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** Prereorganization ** (See Figure 2-35.)					
DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	DLZURGM0	DLZURGM0	DLZURGM0		
	IJJFCBZD	IJJFCBZD			
	IJGFOCZZ	IJGFOCZZ			
** DB Scan ** (See Figure 2-36.)					
DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	
	DLZCONSL				
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJGFICZZ	IJGFICZZ			
** Prefix Resolution ** (See Figure 2-37.)					
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 2-38.)					
DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	CBLTDLI				
	IJJFCBZD	IJJFCBZD			

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
	IJJFCIZD				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
** Work File Generator ** (See Figure 2-39.)					
DLZDSEHO	DLZDSEHO	DLZDSEHO	DLZDSEHO	DLZDSEHO	DLZBEGIN
	DLZBEGIN				
	OPENWORK				
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJGFICZZ	IJGFICZZ			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				

Diagnostic and Test Modules

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
**** System Formatted Dump ****					
DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0	
		DLZCBDP0	DLZCBDP0	DLZCBDP0	
**** 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	
		DLZCBDP0	DLZCBDP0	DLZCBDP0	
**** Run and Buffer Statistics **** (See Figure 2-43.)					
DLZSTTL	DLZSTTL	DLZSTTL	DLZSTTL	DLZSTTL	
**** Trace Print Utility **** (See Figure 2-42.)					
DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRTE
	DLZTPRM0	DLZTPRM0	DLZTPRM0		DLZTPRM0
	IJJFCBIC				
	IJJFCIZD	IJJFCIZD			
	IJFVZZZZ	IJFVZZZZ			
	IJGVIEZZ	IJGVIEZZ			
	IJ2Mnnnn	IJ2Mnnnn			

Core Image Library	CSECT(S)/ Entry Point(s)	Relo Library	Source Library	Storage ID	Suppl Inf
** HD Partial Reorganization Utility ** (See Figure 2-44.)					
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			
	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	

Section 5. Data Areas

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.
- 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 on page 5-3 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 (PSIL), PSB, and one DMB directory (DDIR) entry 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.

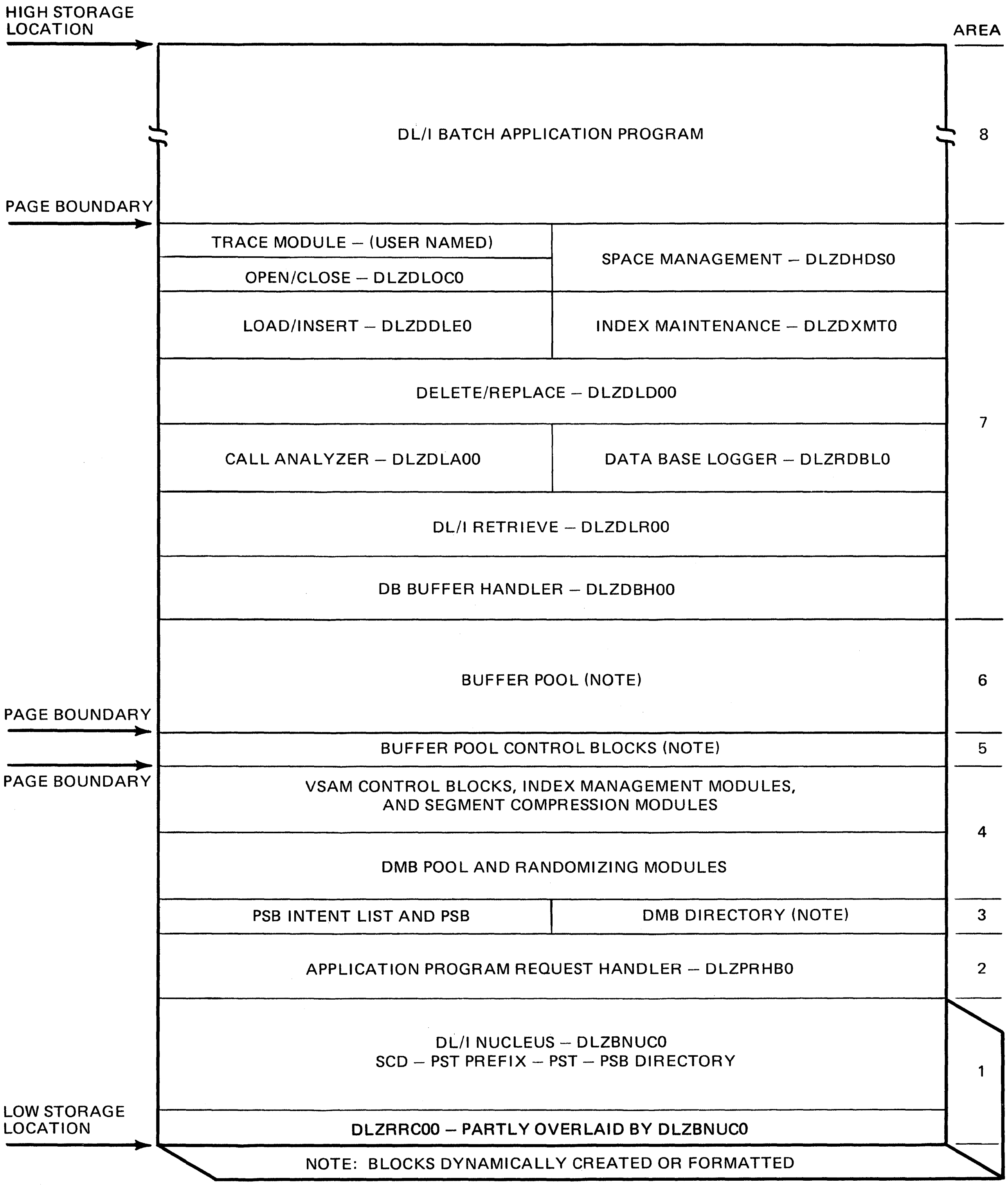


Figure 5-1. Map of Main Storage in the DL/I Batch Partition

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 these control blocks. The following discussion references Figure 5-2 on page 5-6 and Figure 5-3 on page 5-7. (Figure 5-2 on page 5-6 shows the DL/I control block relationships in the batch environment; Figure 5-3 on page 5-7 shows these relationships in the online environment.)

The SCD is the major control block in the DL/I system. It is located in the DL/I nucleus. The SCD contains DL/I copyright information, entry point addresses of DL/I routines, 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 PSB directory from which the PSB and PSB intent list may be obtained. In a batch system, there is only one PSB directory entry. In an online system, there may be many PSB directory entries.
- The DMB directory. There is one DMB directory entry 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 provides task-local storage for batch and CICS/DOS/VS - DL/I online tasks while they are being served by DL/I. 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 issue DL/I calls or commands. In a CICS/DOS/VS - DL/I online environment, the maximum is 255; in batch, there can be only one. The PSB directory entry 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 entry contains the address of the DMB. Each DMB contains a prefix, one ACB extension for each data set in the DMB (HISAM has two data sets), 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 logical 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 pointers to the VSAM ACB and RPL for the data set. Each PSDB contains:

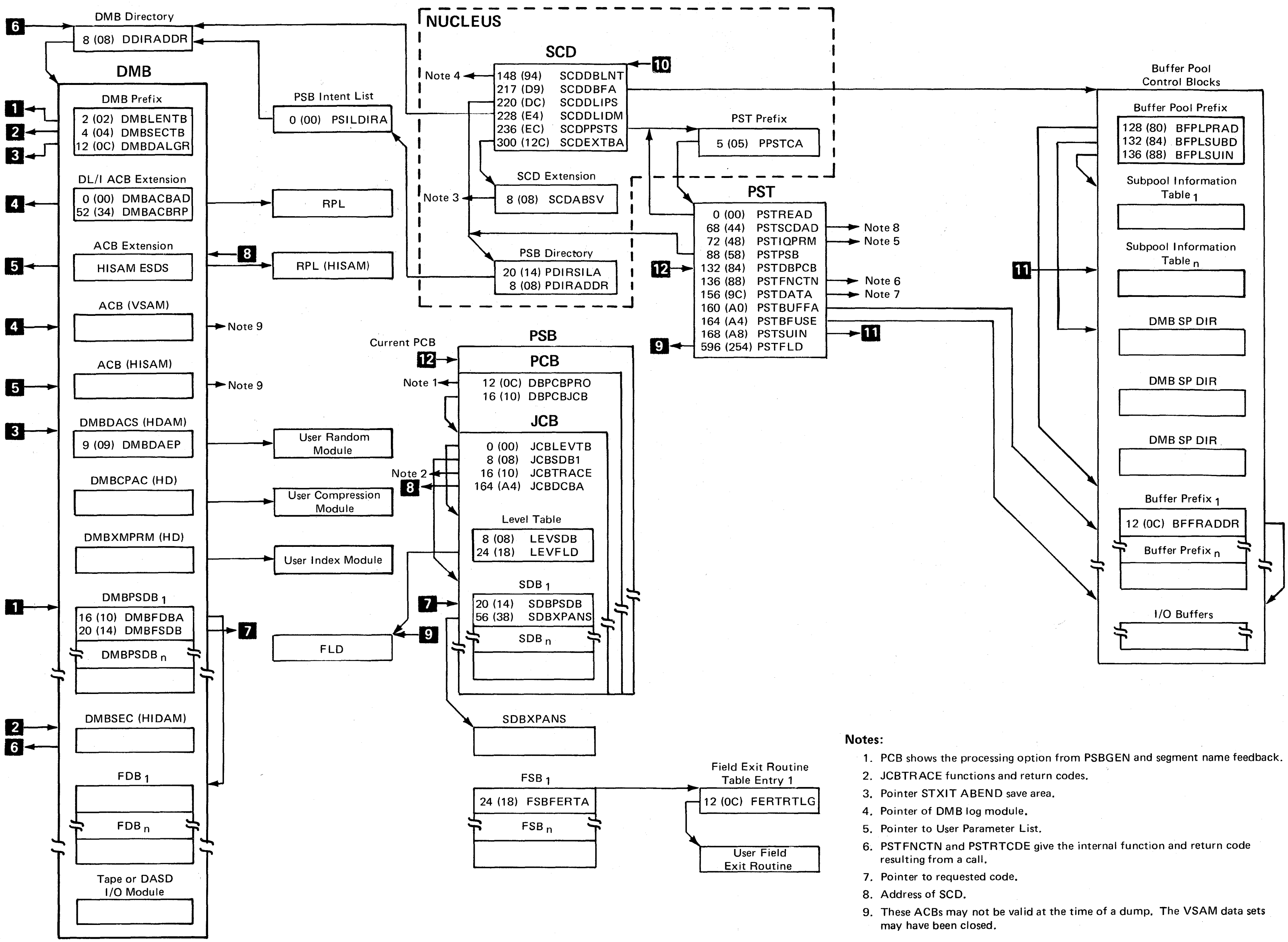
- A pointer to the first FDB for the segment
- A pointer to the SDB for the active PCB which is sensitive to this segment type. If more than one PCB is sensitive to this segment type, the address of the SDB for the next PCB is contained in the active PSDB.

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)
- A pointer to the first buffer prefix
- A pointer to the first DMB subpool directory entry

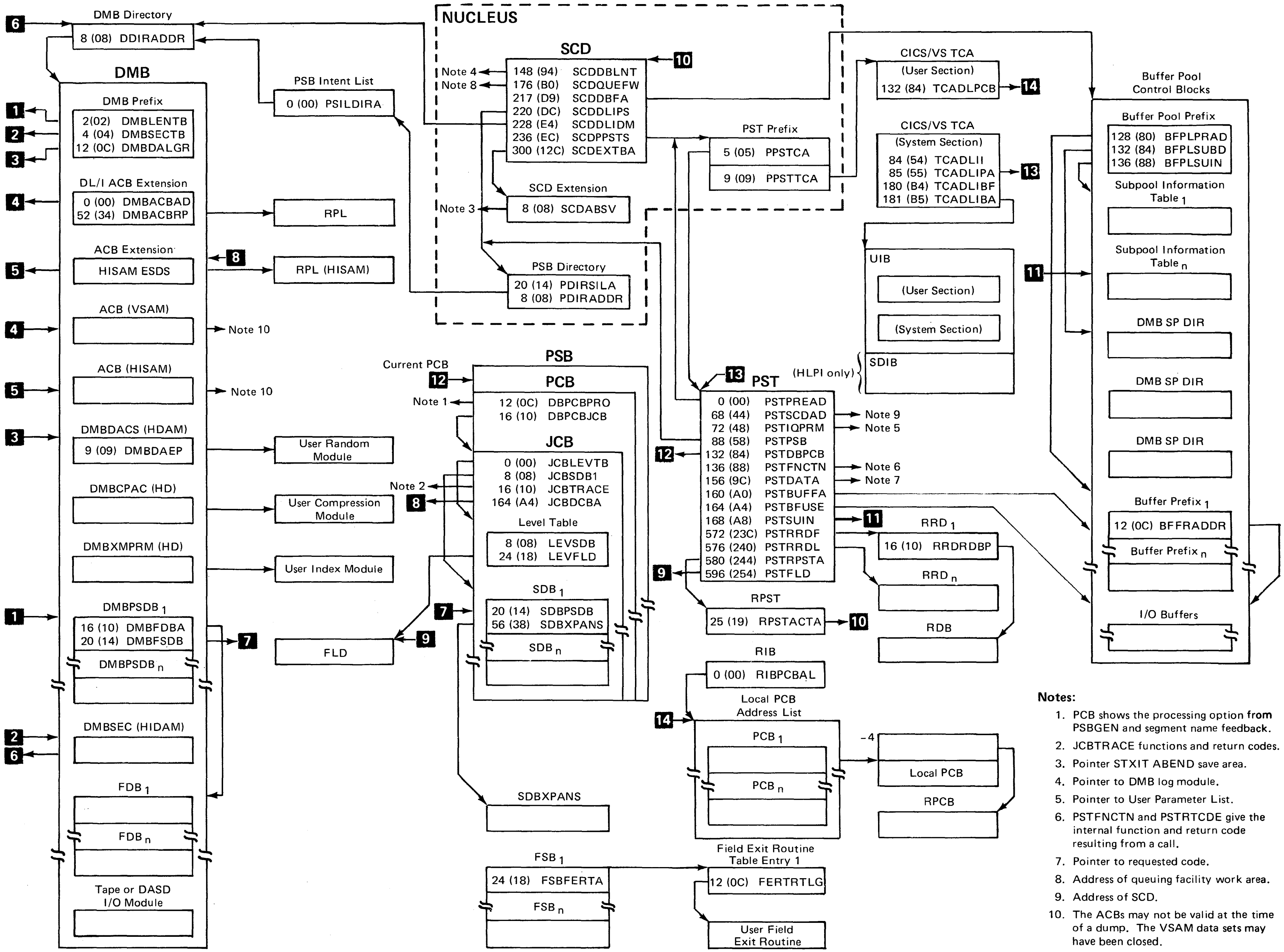
The buffer prefix contains a pointer to the I/O buffer which it references.



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 SCD.
9. These ACBs may not be valid at the time of a dump. The VSAM data sets may have been closed.

Figure 5-2. DL/I Batch 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 to 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.
10. The ACBs may not be valid at the time of a dump. The VSAM data sets may have been closed.

5-3

Figure 5-3. DL/I Online Control Block Relationships

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 routine (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 (SHISAM), only the KSDS ACB extension is created. The ACBs are generated only when the blocks are loaded for execution by the DLZDBLM0 routine 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 5-4 on page 5-9.

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 DTF EXTENSION DSECT Name: DMBDTFXT	ACBXT	— ACB Extension
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 5-4. 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 routine (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 the current position after 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 after it is loaded into storage is shown in Figure 5-5 on page 5-11.

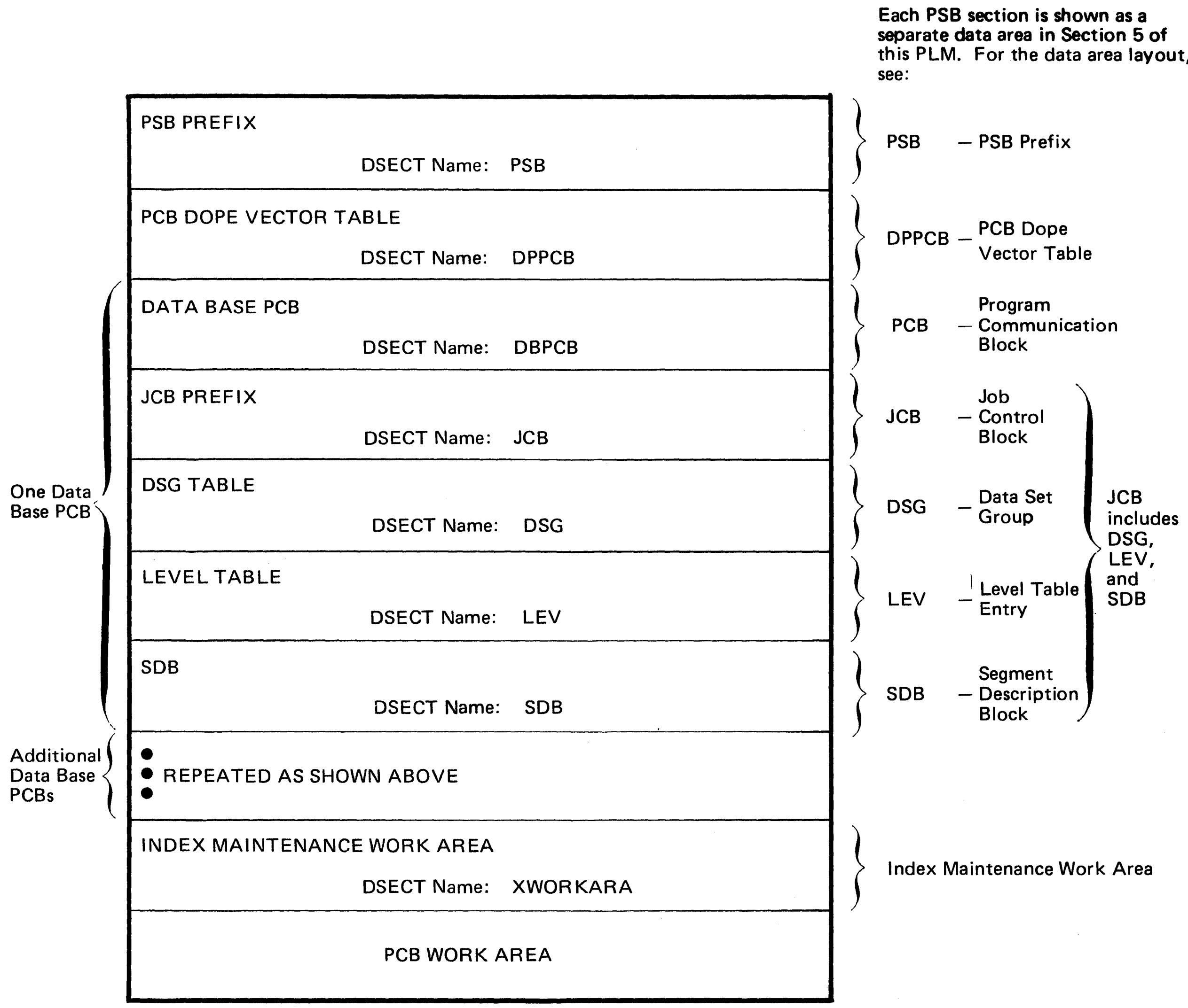


Figure 5-5. General Structure of PSB.

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 into 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 5-6.

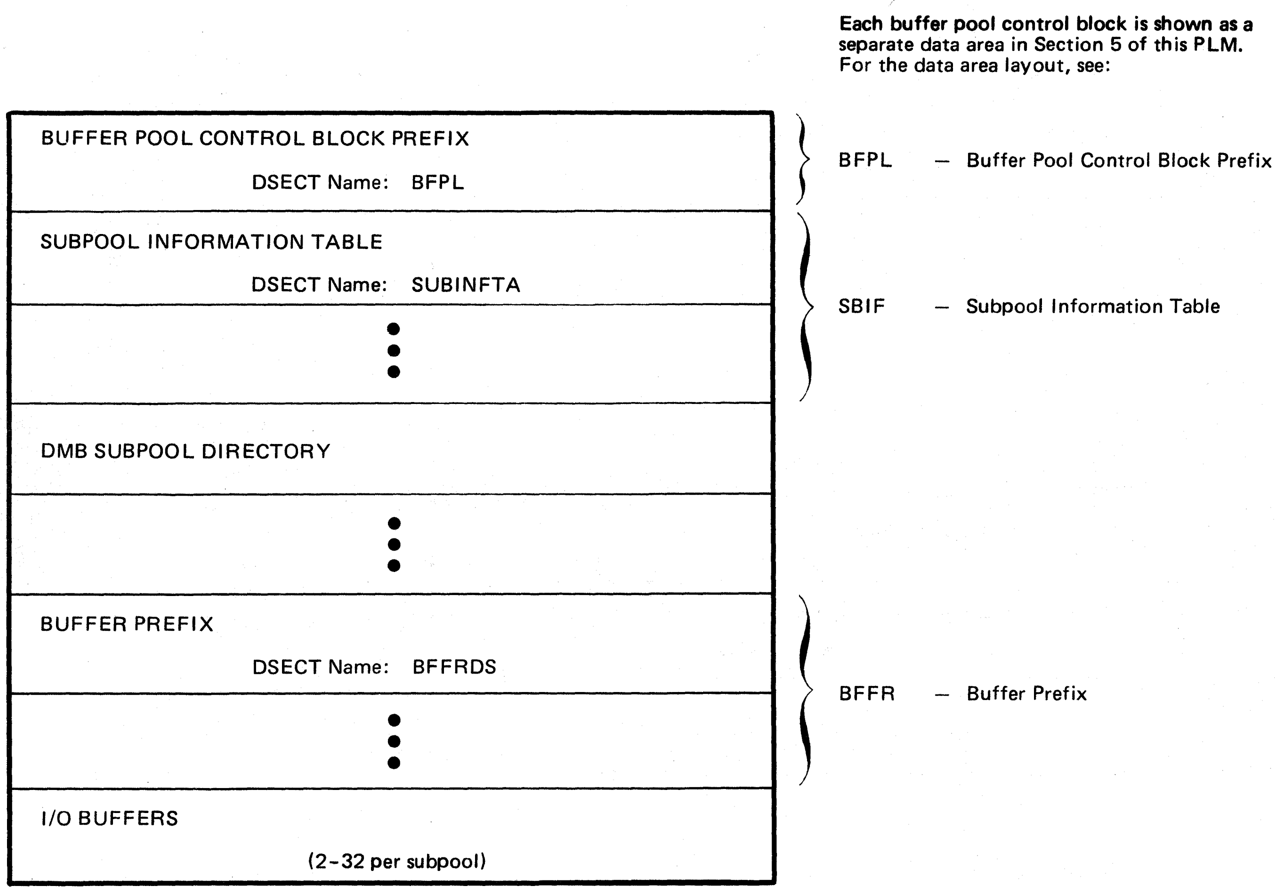


Figure 5-6. General Structure of DL/I Buffer Pool Control Blocks

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), which is part of the DLZIDLI macro. The information in ACBXT is repeated for each data set in the DMB. The ACB extension is immediately behind the DMB Prefix. For HISAM data bases, there is a second ACB extension immediately behind the first.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBACBXT		
0	(0)	4	DMBACBST		Start of ACB extension
0	(0)	4	DMBACBAD		Address of corresponding ACB
4	(4)	2	DMBCINV		Control interval size
6	(6)	1	DMBACBDL		Delta cylinders to scan
7	(7)	1	DMBACBAP		Number of root anchor points per control interval (HDAM)
8	(8)	2	DMBACBMX		Length of the largest segment in data set
10	(A)	2	DMBACBMN		Length of the smallest segment in data set
12	(C)	4	DMBECB		ACB/ECB for buffer handler
16	(10)	4	DMBHIBLK		Highest possible RBN (CI)
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 interval per cylinder
30	(1E)	1	DMBCITRK		Number of control interval per track
31	(1F)	1	DMBKEYLE		Key length of KSDS
32	(20)	2	DMBRKP		Relative key position
34	(22)	1	DMBOFLGS		Open flags
			DMBIGNOR	.1..	"X'40'" IGN specified for workfile on load

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DMBNUSE	..1.	"X'20'" ACB does not have resolved secondary index entries; workfile must be used
			DMBOPEN	...1	"X'10'" The corresponding ACB is open
			DMBPUTKY	.... 1...	"X'08'" Simulate not load mode to VSAM
			DMBUNLD	.... .1..	"X'04'" Unload issued for this DMB
35	(23)	1	DMBVSFLG		Flags
			DMBCISPL	1...	"X'80'" CONTROL INTERVAL split occurred
			DMBPSEQ	...1	"X'10'" SEQUENTIAL processing 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	DMBIND0		Permanent indicators
			DMBWCHK	.... 1...	"X'08'" Write check option
			DMBKEY	1...	"X'80'" Data set contains keys (HISAM/SHISAM)
			DMBBESDS	.1..	"X'40'" Blocked ESDS
			DMBFBA	..1.	"X'20'" FBA device
			DMBINIT	...1	"X'10'" Space management has been entered for this DMB
47	(2F)	1			Reserved
48	(30)	4	DMBSPLCT		Control interval split count
52	(34)	4	DMBACBRP		Address of this ACB's RPL
56	(38)	2	DMBACBLC		Log count (HISAM ONLY)
58	(3A)	2	DMBFRSPC		Distributed free space parameter
59	(3B)		DMBFRSP1		"*-1" SECOND FREE SPACE parameter
60	(3C)	8	DMBACBNM		Data set name as in ACB

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DMBACLNO		"*-4-DMBACBST" Length OF version 1.0 ACB extension
68	(44)	4	DMBACBEX		Address of the exit list for this ACB
72	(48)	2	DMBFBASN		FBA scan value
74	(4A)	2	DMBEXQUE		Queue header for tasks that are extending the data base
76	(4C)	4	DMBVSEOF		Relative block number of where software end-of-file will be when all control intervals on the write chain have been written
80	(50)		DMBACBND		END OF ACB EXTENSION
			DMBACBLN		"DMBACBND-DMBACBST" Length of ACB extension

LABEL EQUATES

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DMBDCBDL	.... .11.	"DMBACBDL"
			DMBDCBAP	.... .111	"DMBACBAP"
			DMBDCBMX	.... 1...	"DMBACBMX"
			DMBDCBMN	.... 1.1.	"DMBACBMN"
			DMBDCBLN	.1.1	"DMBACBND-DMBACBST"
			DCBHIBLK	...1	"DMBHIBLK"
			DCBRBASN	...1 .1..	"DMBRBASN"
			DCBRLBLK	...1 1...	"DMBRLBLK"
			DCBKEYLE	...1 1111	"DMBKEYLE"
			DCBLRECL	..1. 1.1.	"DMBLRECL"
			DCBBFACT	..1. 11..	"DMBBFACT"
			DCBRKP	..1.	"DMBRKP"
			DCBINDO	..1. 111.	"DMBINDO"
			DCBKEY	1...	"X'80'"
			ACBHIBLK	...1	"DMBHIBLK"
			ACBRBASN	...1 .1..	"DMBRBASN"
			ACBRLBLK	...1 1...	"DMBRLBLK"
			ACBKEYLE	...1 1111	"DMBKEYLE"
			ACBLRECL	..1. 1.1.	"DMBLRECL"
			ACBBFACT	..1. 11..	"DMBBFACT"
			ACBINDO	..1. 111.	"DMBINDO"

HSAM DTF EXTENSION

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBDTFXT		
0	(0)	4	DMBDTFIN		Address of HSAM input DTF
4	(4)	4	DMBDTFOT		Address of HSAM output DTF

Cross Reference

Name	Hex Offset	Hex Value
ACBBFACT	50	2C
ACBHIBLK	50	10
ACBINDO	50	2E
ACBKEYLE	50	1F
ACBLRECL	50	2A
ACBRBASN	50	14
ACBRLBLK	50	18
DCBBFACT	50	2C
DCBHIBLK	50	10
DCBINDO	50	2E
DCBKEY	50	80
DCBKEYLE	50	1F
DCBLRECL	50	2A
DCBRBASN	50	14
DCBRKP	50	20
DCBRLBLK	50	18
DMBACBAD	0	
DMBACBAP	7	
DMBACBDL	6	
DMBACBEX	44	
DMBACBLC	38	
DMBACBLN	50	50
DMBACBMN	A	
DMBACBMX	8	
DMBACBND	50	
DMBACBNM	3C	
DMBACBRP	34	
DMBACBST	0	
DMBACBXT	0	
DMBACLNO	40	40
DMBBESDS	2E	40
DMBBFACT	2C	
DMBCICYL	1C	
DMBCINV	4	
DMBCISPL	23	80
DMBCITRK	1E	
DMBDCBAP	50	07
DMBDCBDL	50	06
DMBDCBLN	50	50
DMBDCBMN	50	0A
DMBDCBMX	50	08
DMBDTFIN	0	
DMBDTFOT	4	
DMBDTFXT	0	
DMBECB	C	
DMBEXQUE	4A	
DMBFBA	2E	20
DMBFBASN	48	
DMBFRSPC	3A	
DMBFRSP1	3A	3B
DMBHIBLK	10	
DMBHIRBA	24	
DMBIGNOR	22	40
DMBINDO	2E	
DMBINIT	2E	10
DMBKEY	2E	80
DMBKEYLE	1F	
DMBLRECL	2A	
DMBNUSE	22	20
DMBOFLGS	22	
DMBOPEN	22	10
DMBPSEQ	23	10
DMBPUTKY	22	08
DMBRBASN	14	
DMBRKP	20	
DMBRLBLK	18	
DMBSPLCT	30	
DMBUNLD	22	04
DMBVSBFR	28	
DMBVSEOF	4C	
DMBVSFLG	23	
DMBWCHK	2E	08

ACT - Partial Reorganization Action Table

DSECT Name: DLZPRACT

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).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	ACT		
0	(0)	4	ACTSTART		
0	(0)	1	ACTCRTYP		Action record type
1	(1)	1	ACTCROW		Action row number
2	(2)	1			Reserved
3	(3)	1	ACTGOPTN		Optional action identifier
			ACTQOPT2	1...	"X'80'" Option with two ACT entries
4	(4)	1	ACTGDEST		Destination indicator flags
			ACTQSRT1	1...	"X'80'" Record goes to sort 1
			ACTQSRT2	.1..	"X'40'" Record goes to sort 2
			ACTQSRT3	..1.	"X'20'" Record goes to sort 3
			ACTQSRT4	...1	"X'10'" Record goes to sort 4
5	(5)	1	ACTCSDS		DS of moved segment for sort 1
6	(6)	2	ACTOSGT		Offset in SGT from which this is chained
8	(8)	2	ACTOSUPD		Offset in SGT for segment to be updated
10	(A)	2	ACTOSZID		Offset in SGT for Z segment in physical pair
12	(C)	2	ACTOPRMV		Offset in prefix of pointer to be extracted
14	(E)	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-zero

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
22	(16)	2	ACTOANXT		Offset in ACT of next action
24	(18)	4	(2)		Reserved action table entry

Cross Reference

Name	Hex Offset	Hex Value
ACT	0	
ACTCROW	1	
ACTCRTYP	0	
ACTCSDS	5	
ACTGDEST	4	
ACTGOPTN	3	
ACTLLEN	18	20
ACTOANXT	16	
ACTOCHED	10	
ACTOCNXT	12	
ACTOPRMV	C	
ACTOPUPD	E	
ACTOSGT	6	
ACTOSUPD	8	
ACTOSZID	A	
ACTOTEST	14	
ACTQOPT2	3	80
ACTQSRT1	4	80
ACTQSRT2	4	40
ACTQSRT3	4	20
ACTQSRT4	4	10
ACTSTART	0	

ARG0 - HLPI ARG0 Parameters

DSECT Name: DLZARG0

This DSECT describes the fields contained in the DL/I HLPI ARG0 Interface Parameter list.

ARG0

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZARG0		
0	(0)	4	ARG0		
0	(0)	1	ARG0FNID		ARG0 ID x'00'
1	(1)	1	ARG0FNCD		Function code
			INITCALL	.... ..1.	"X'02'" Initialize call
			SCHDCALL	.... .1..	"X'04'" Schedule call
			TERMCALL	.... .11.	"X'06'" Termination call
			CHKPCALL	.... 1...	"X'08'" Checkpoint call
			GUCALL	.... 1.1.	"X'0A'" Get unique call
			GNCALL	.... 11..	"X'0C'" Get next call
			GNPCALL	...1	"X'10'" Get next in parent call
			ISRTCALL	...1 ..1.	"X'12'" Insert call
			REPLCALL	...1 .1..	"X'14'" Replace call
			DLETCALL	...1 .11.	"X'16'" Delete call
			LOADCALL	...1 1...	"X'18'" Load call
2	(2)	1	ARG0FLG1		Argument Flag 1
3	(3)	1	ARG0FLG2		Argument Flag 2
4	(4)	1	ARG0FLG3		Argument Flag 3
			APPLPLI	.... ..1.	"X'02'" Application program is PL/I
5	(5)	1	ARG0MODI		Index for start of symbol HLPIQS in DLZHLPIL DSECT
6	(6)	1	ARG0RELN		Relative number of this call
7	(7)	1	ARG0TOTN		Total number of calls in this statement
8	(8)	8	ARG0RMGR		Resource manager's ID
16	(10)	8	ARG0STMT		Statement identifier
24	(18)	1	ARG0OPTS		Statement level options
			USINGPCB	.1..	"X'40'" Using PCB

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
25	(19)	1	ARGOCCOD		Command codes
			CCFIRST	1...	"X'80'" First
			CCLAST	.1..	"X'40'" Last
			CCLOCKED	..1.	"X'20'" Locked
			CCINFROM	...1	"X'10'" Into (Get) or from (Insert, Load, Replace)
26	(1A)	1	ARGOOPT1		Call options
			ARGOKFBA	.... ..1.	"X'02'" Key feedback specified
			ARGOKFBL	1	"X'01'" Feedback length specified
27	(1B)	1	ARGOSOPT		Segment options
			OPTSEGL	1...	"X'80'" Seglength present
			OPTWHERE	.1..	"X'40'" Where
				....	"X'20'" Boolean where (IMS only)
			OPTFLDL	...1	"X'10'" Field length present
			OPTVAR	.... 1...	"X'08'" Variable
			OPTSEGM	.... .1..	"X'04'" Segment name present
			OPTOFF	.... ..1.	"X'02'" Offset specified
28	(1C)	1			Reserved

Cross Reference

Name	Hex Offset	Hex Value
APPLPLI	4	02
ARG0	0	
ARG0CCOD	19	
ARG0FLG1	2	
ARG0FLG2	3	
ARG0FLG3	4	
ARG0FNCD	1	
ARG0FNID	0	
ARG0KFBA	1A	02
ARG0KFBL	1A	01
ARG0MODI	5	
ARG0OPTS	18	
ARG0OPT1	1A	
ARG0RELN	6	
ARG0RMGR	8	
ARG0SOPT	1B	
ARG0STMT	10	
ARG0TOTN	7	
CCFIRST	19	80
CCINFROM	19	10
CCLAST	19	40
CCLOCKED	19	20
CHKPCALL	1	08
DLETCALL	1	16
DLZARG0	0	
GNCALL	1	0C
GNPCALL	1	10
GUCALL	1	0A
INITCALL	1	02
ISRTCALL	1	12
LOADCALL	1	18
OPTFLDL	1B	10
OPTOFF	1B	02
OPTSEGL	1B	80
OPTSEGM	1B	04
OPTVAR	1B	08
OPTWHERE	1B	40
REPLCALL	1	14
SCHDCALL	1	04
TERMCALL	1	06
USINGPCB	18	40

BFFR - Buffer Prefix

DSECT Name: DLZBFFR

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	BFFRDS		
0	(0)	7	BFFRCIID		Control interval Identifier
0	(0)	4	BFFRCIRB		Control interval RBN
4	(4)	2	BFFRDMB		DMB number
6	(6)	1	BFFRDCB		ACB number
7	(7)	1	BFFRSW		Flags
			BFFRWCH	1...	"X'80'" Buffer on write chain
			BFFRWRT	.1..	"X'40'" Buffer being written
			BFFRREAD	..1.	"X'20'" Buffer being read
			BFFRMT	...1	"X'10'" Buffer empty
			BFFRPRED	.... 1...	"X'08'" Buffer waiting for predecessor being written
			BFFRWERR	.... .1..	"X'04'" Buffer has permanent write error
			BFFREXNQ	.... ..1.	"X'02'" Existing control interval ID enqueued
			BFFRPNNQ	1	"X'01'" Pending control interval ID enqueued
8	(8)	2	BFFRPST		PST prefix numbers for enqueue/dequeue
8	(8)	1	BFFRPSTF		PST prefix number of the controlling task
9	(9)	1	BFFRPSTL		PST prefix number of the task being last in the chain of waiting tasks
10	(A)	2	BFFRLOCU		Log count
12	(C)	1	BFFRUSCT		Use count
12	(C)	4	BFFRADDR		Address of the 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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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
27	(1B)	1	BFFRSW1		Flags
			BFFRNORU	1...	"X'80'" Buffer is not reusable
			BFFRLOCK	.1..	"X'40'" Buffer locked by logger
			BFFRREL	.... 1...	"X'08'" Buffer is released
			BFFRLAST	1	"X'01'" Last buffer prefix for this subpool
28	(1C)	2	BFFRNPST		PST prefix numbers for ENQ/DEQ
28	(1C)	1	BFFRNPSF		PST prefix number of task which enqueued on new control interval ID and is first in the chain
29	(1D)	1	BFFRNPSL		PST prefix number of task which enqueued on new control interval ID and is last in chain
30	(1E)	2	BFFRHOLE		Length of largest space available in the buffer buffer prefix
			BFFRLEN		Length of buffer prefix

Cross Reference

Name	Hex Offset	Hex Value
BFFRADDR	C	
BFFRCIID	0	
BFFRCIRB	0	
BFFRDCB	6	
BFFRDMB	4	
BFFRDS	0	
BFFREXNQ	7	02
BFFRHOLE	1E	
BFFRLAST	1B	01
BFFRLEN	20	20
BFFRLOCK	1B	40
BFFRLOCU	A	
BFFRMT	7	10
BFFRNACB	1A	
BFFRNCID	14	
BFFRNCII	14	
BFFRNDMB	18	
BFFRNORU	1B	80
BFFRNPSF	1C	
BFFRNPSL	1D	
BFFRNPST	1C	
BFFRPNNQ	7	01
BFFRPRED	7	08
BFFRPST	8	
BFFRPSTF	8	
BFFRPSTL	9	
BFFRREAD	7	20
BFFRREL	1B	08
BFFRSW	7	
BFFRSW1	1B	
BFFRUSCT	C	
BFFRUSID	10	
BFFRWCBW	13	
BFFRWCFW	12	
BFFRWCH	7	80
BFFRWERR	7	04
BFFRWRT	7	40

BFPL - Buffer Pool Control Block Prefix

DSECT Name: DLZBFPL

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	BFPL		
0	(0)	4	BFPLID		
0	(0)	4			Buffer pool control block ID (BFPL)
4	(4)	4	(3)		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
61	(3D)	1	BFPLWERT		Largest number of write error buffers ever in pool
62	(3E)	1			Reserved for future use
63	(3F)	1			Reserved for future use
64	(40)	4	BFPLNQW1		Enqueue/Dequeue workarea 1. Byte 0 indicates the following:
			BFPLEXCI	....	"X'00'" Enqueue/dequeue existing control interval code

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			BFPLPECI	.... .1..	"X'04'" Enqueue/dequeue pending control interval code
			BFPLSUPO	.... 1...	"X'08'" Enqueue/dequeue subpool code
			BFPLCIXT	.... 11..	"X'0C'" Enqueue/dequeue on control interval extension queue bytes 1-3 contains a pointer to the PST prefix numbers of first and last task waiting for the resource
68	(44)	4	(15)		Reserved for future use
128	(80)	4	BFPLPRAD		Beginning address of the buffer prefix area
132	(84)	4	BFPLSUBD		Beginning address of the DMB-subpool-directory
136	(88)	4	BFPLSUIN		Beginning of the subpool information table entries
			BFPLLEN	1... 1...	"*-BFPL" length of the buffer pool control block prefix

Cross Reference

Name	Hex Offset	Hex Value
BFPL	0	
BFPLALTR	1C	
BFPLBKWT	24	
BFPLCHBK	30	
BFPLCHWT	2C	
BFPLCIXT	40	0C
BFPLEXCI	40	00
BFPLID	0	
BFPLIGET	38	
BFPLINPL	14	
BFPLISTL	34	
BFPLLEN	88	88
BFPLNQW1	40	
BFPLNWBK	28	
BFPLOSWT	20	
BFPLPECI	40	04
BFPLPRAD	80	
BFPLRDCT	18	
BFPLRQCT	10	
BFPLSUBD	84	
BFPLSUIN	88	
BFPLSUPO	40	08
BFPLWERR	3C	
BFPLWERT	3D	

COM - Common Area

DSECT Name: DLZPRCOM

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

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	COMAREA		
0	(0)	4	COMSTART		
0	(0)	12	COMCID		Identifier
GENERAL ADDRESS SECTION					
12	(C)	4	COMFCOML		Length of common
16	(10)	4	COMACOM		Address of common
20	(14)	4	COMASIOA		Address of an I/O area for GU, GN calls
24	(18)	4	COMAERRS		Entry point of error message writer
28	(1C)	4	COMAFILE		Entry point of file manager
32	(20)	4	COMADLII		Entry point of DL/I interface module
36	(24)	4	COMACHKP		Entry point of checkpoint processor
40	(28)	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
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

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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 number 1
100	(64)	2	COMHNSGX		Number of SGX entries
102	(66)	2	COMHLACT		Length of an ACT entry
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

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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		Dl/I common services request code
			COMQGPRE	1	"X'01'" Get prefix address of last segment retrieved
			COMQIPRE	.... ..1.	"X'02'" Get prefix address of last segment inserted
			COMQBKLC	11	"X'03'" Locate block
			COMQBYLC	.... .1..	"X'04'" Byte locate
			COMQBYAL	.... .1.1	"X'05'" Locate byte for updating
			COMQGRBA	.... .11.	"X'06'" Get RBA of last seg retrieved/inserted
			COMQFREE	.... .111	"X'07'" Free space occupied by a segment
			COMQRKEY	.... 1...	"X'08'" Find key of HDAM root at block N

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			COMQLOLD	.... 1..1	"X'09'" Log data before change
			COMQLNEW	.... 1.1.	"X'0A'" Log data after change
			COMQBMON	.... 1.11	"X'0B'" Turn bit maps on
			COMQBMOF	.... 11..	"X'0C'" Turn bit maps off
			COMQBFAL	.... 11.1	"X'0D'" Mark buffer altered
			COMQULHB	.... 111.	"X'0E'" Set LO and HI block
			COMQXRMA	 1111	"X'0F'" Swap randomizer entry points
			COMQINT2	...1	"X'10'" Initialize for Part 2
			COMQINTU	...1 ...1	"X'11'" Initialize for UNLOAD
			COMQRSTU	...1 ..1.	"X'12'" Reset after UNLOAD
			COMQINTR	...1 ..11	"X'13'" Initialize for RELOAD
			COMQRSTR	...1 .1..	"X'14'" Reset after RELOAD
			COMQCRAP	...1 .1.1	"X'15'" Clear HDAM root anchor point
			COMQGNDX	...1 .11.	"X'16'" Retrieve an index record
129	(81)	1	COMGOUT		Output control switches
			COMQSALL	1...	"X'80'" Full statistics required
			COMQSUMM	.1..	"X'40'" Summary of statistics required
			COMQSNON	..1.	"X'20'" No statistics to be produced
			COMQNPSB	1	"X'01'" No PSB to be generated
130	(82)	1	COMGUCTL		Update process control switches
			COMQDUNQ	1...	"X'80'" Q record update is complete
			COMQSPUS	.1..	"X'40'" Spill is in use
			COMQSPOF	..1.	"X'20'" Spill overflow has unprocessed records
			COMQNDBR	...1	"X'10'" No database record in HDAM range

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
131	(83)	1	COMGPART		Part in progress indicator
			COMQSTP1	1...	"X'80'" Part 1 is in progress
			COMQSTP2	.1..	"X'40'" Part 2 is in progress
			COMQRIP	.... ..1.	"X'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 number 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

DLI ADDRESS SECTION

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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
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 ASBTDLI
228	(E4)	4			Reserved

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
FILE SECTION					
232	(E8)	4	COMSFL01(9)		FCB for PRWRKF1
268	(10C)	4	COMSFL02(9)		FCB for PRWRKF2
304	(130)	4	COMSFL03(9)		FCB for PRWRKF3
340	(154)	4	COMSFL04(9)		FCB for PRWRKF4
376	(178)	4	COMSFL05(9)		FCB for PRWRKF5
412	(19C)	4	COMSFL06(9)		FCB for PRWRKF6
448	(1C0)	4	COMSFL07(9)		FCB for PRWRKF7
484	(1E4)	4	COMSFL08(9)		FCB for PRWRKF8
520	(208)	4	COMSFL09(9)		FCB for PRWRKF9
556	(22C)	4	COMSFL10(9)		FCB for PRWRKFA
592	(250)	4	COMSFL11(9)		FCB for SYSPRINT
628	(274)	4	COMSFL12(9)		FCB for SYSPUNCH
664	(298)	4	COMSFL13(9)		FCB for SYSIN

CHECKPOINT SECTION

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
700	(2BC)	4	COMFCKID		ID of last DL/I checkpoint taken
704	(2C0)	1	COMCPROC		PART2 phase in process indicator
			COMQPHUR	1	"X'01'" UNLOAD/RELOAD in progress
			COMQPHSC	.... ..1.	"X'02'" SCAN in progress
			COMQPHSO	11	"X'03'" SORT in progress
			COMQPHUD	.... .1..	"X'04'" UPDATE in progress
705	(2C1)	1	COMGPHAS		Phase GO NOGO switches
			COMQSCAN	1...	"X'80'" SCAN required
			COMQSRT1	.1..	"X'40'" SORT 1 required
			COMQSRT2	..1.	"X'20'" SORT 2 required
			COMQSRT3	...1	"X'10'" SORT 3 required
			COMQSRT4	.... 1...	"X'08'" SORT 4 required
			COMQUPDT	.... .1..	"X'04'" UPDATE required
			COMQOPTN	.... ..1.	"X'02'" Option selection required
			COMQUPIX	1	"X'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	1...	"X'80'" Record in buffer for update
711	(2C7)	1			Reserved
712	(2C8)	4	(2)		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 which follow are the list of areas to checkpoint and recover. This list is passed to DL/I.

720	(2D0)	4	COMAPMCT	D-> 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	Fullword parameter count
772	(304)	4	COMLCXPL	Equate for end of parameter list
772	(304)	4	(3)	Reserved

END OF CHECKPOINT RESTART PARM LIST

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
DATA SET GROUP TABLE					
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	.... 1..1	"*-COMCDDNM" length of a DSG table entry
793	(319)	81	(9)		Space for 9 more DSG entries
874	(36A)	2			Reserved
876	(36C)	16	COMCTRAC		Trace of last 16 requests to DL/I services
892	(37C)	4	(4)		Reserved
			COMLLEN		"*-COMSTART" length of common
PRINT HEADER LINE					
908	(38C)	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		

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
COMABUFH	C8		COMFPMCT	300	
COMACHKD	2F0		COMFRETC	74	
COMACHKP	24		COMFSGTL	3C	
COMACHXR	2D4		COMFSGTM	44	
COMACOM	10		COMFWRK1	A8	
COMADBD	2C		COMFWRK2	AC	
COMADBT	34		COMFWRK3	B0	
COMADDIR	C0		COMFWRK4	B4	
COMADLI	E0		COMGOUT	81	
COMADLII	20		COMGPART	83	
COMAERRS	18		COMGPHAS	2C1	
COMAFILE	1C		COMGPHS2	2C6	
COMAFL25	2FC		COMGUCTL	82	
COMAGBUF	2F4		COMHEADC	3DB	
COMAIOWK	2E0		COMHEADD	3F2	
COMAIPCB	2D8		COMHEADP	401	
COMALMXS	2DC		COMHEADR	38C	
COMALOG	C4		COMHEADV	3B7	
COMAMSGN	9E		COMHKYLN	70	
COMAPLST	2E4		COMHLACT	66	
COMAPMCT	2D0		COMHLDBT	5A	
COMAPREF	D0		COMHLRGT	6A	
COMAPST	BC		COMHLSGT	5E	
COMAREA	0		COMHMXPR	62	
COMARGT	54		COMHMXSG	6E	
COMASCD	B8		COMHNACT	68	
COMASGT	40		COMHNDBT	5C	
COMASIOA	14		COMHNRGT	6C	
COMASMGR	CC		COMHNSGT	60	
COMASTWR	28		COMHNSGX	64	
COMAVTXT	A4		COMHPAGE	405	
COMCDDNM	310		COMLCXPL	304	
COMCID	0		COMLDSGT	318	09
COMCIREQ	80		COMLLEN	37C	038C
COMCPROC	2C0		COMLRGT	58	
COMCPSBN	78		COMOCRGT	2C2	
COMCSDIA	9A		COMODBSN	2C4	
COMCSMSG	91		COMPAGEM	407	
COMCSSIZ	88		COMQBFAL	80	0D
COMCSTEC	84		COMQBKLC	80	03
COMCTRAC	36C		COMQBMOF	80	0C
COMFACTL	48		COMQBMON	80	0B
COMFACTM	50		COMQBYAL	80	05
COMFCKID	2BC		COMQBYLC	80	04
COMFCOML	C		COMQCKND	2C8	02D0
COMFCXPL	2E8		COMQCRAP	80	15
COMFDBTL	30		COMQDUNQ	82	80
COMFDBTM	38		COMQFREE	80	07
COMFFCBL	2F8		COMQGNDX	80	16

COMAACT 4C COMFLCKD 2EC

Cross Reference

Name	Hex Offset	Hex Value
COMQGPRE	80	01
COMQGRBA	80	06
COMQINBF	2C6	80
COMQINTR	80	13
COMQINTU	80	11
COMQINT2	80	10
COMQIPRE	80	02
COMQLNEW	80	0A
COMQLOLD	80	09
COMQNDBR	82	10
COMQNPSB	81	01
COMQOPTN	2C1	02
COMQPHSC	2C0	02
COMQPHSO	2C0	03
COMQPHUD	2C0	04
COMQPHUR	2C0	01
COMQRIP	83	02
COMQRKEY	80	08
COMQRSTR	80	14
COMQRSTU	80	12
COMQSALL	81	80
COMQSCAN	2C1	80
COMQSNON	81	20
COMQSPOF	82	20
COMQSPUS	82	40
COMQSRT1	2C1	40
COMQSRT2	2C1	20
COMQSRT3	2C1	10
COMQSRT4	2C1	08
COMQSTP1	83	80
COMQSTP2	83	40
COMQSUMM	81	40
COMQULHB	80	0E
COMQUPDT	2C1	04
COMQUPIX	2C1	01
COMQXRMA	80	0F
COMRHIPT	DC	
COMRLOPT	D8	
COMRLSEG	D4	
COMSFL01	E8	
COMSFL02	10C	
COMSFL03	130	
COMSFL04	154	
COMSFL05	178	
COMSFL06	19C	
COMSFL07	1C0	
COMSFL08	1E4	
COMSFL09	208	
COMSFL10	22C	
COMSFL11	250	
COMSFL12	274	
COMSFL13	298	
COMSTART	0	
COMXBR14	72	
COMXDGID	318	

CPAC - HDAM/HIDAM Variable Length Segment Compression/Expansion

Routine Interface Table

DSCET Name: DMBCPAC

This table is described as part of the general structure and description of the data management block (DMB), which is part of the DLZIDLI macro. There is one entry for each compressible segment in the DMB.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBCPAC		
0	(0)	8	DMBCPCNM		Segment name
8	(8)	8	DMBCPCSG		Compression routine name
16	(10)	4	DMBCPEP		Entry point of compression routine
20	(14)	1	DMBCPFLG		Flag byte
			DMBCPSEQ	.... 1...	"X'08'" Segment has sequence field defined
			DMBCPVLR	.... .1..	"X'04'" Segment is variable length
			DMBCPKEY	.... ..1.	"X'02'" Segment has key compression option
			DMBCPNIT	1	"X'01'" Initialization and termination processing required
21	(15)	1	DMBCPSQF		Length of key field-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 length, constants plus user data
28	(1C)	4	DMBCPRES		Reserved for initialization

Cross Reference

Name	Hex Offset	Hex Value
DMBCPAC	0	
DMBCPCNM	0	
DMBCPCSG	8	
DMBCPEP	10	
DMBCPFLG	14	
DMBCPKEY	14	02
DMBCPLNG	1A	
DMBCPNIT	14	01
DMBCPRES	1C	
DMBCPSEQ	14	08
DMBCPSGL	18	
DMBCPSQF	15	
DMBCPSQL	16	
DMBCPVLR	14	04

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), which is in the DLZIDLI macro.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBDACS		
0	(0)	8	DMBDANME		Name of address conversion algorithm load module
8	(8)	1	DMBDAKL		Root key length-1
8	(8)	4	DMBDAEP		Entry point of conversion module
12	(C)	2	DMBDASZE		Size of this DSECT
14	(E)	2	DMBDARAP		Number of root anchor pointers per block
16	(10)	4	DMBDABLK		Number of highest block in root addressable area
20	(14)	4	DMBDABYM		Maximum number of bytes per root before overflow outside of root addressable area
24	(18)	4	DMBDABYC		Current number of bytes consecutively inserted or loaded under root

Cross Reference

Name	Hex Offset	Hex Value
DMBDABLK	10	
DMBDABYC	18	
DMBDABYM	14	
DMBDACS	0	
DMBDAEP	8	
DMBDAKL	8	
DMBDANME	0	
DMBDARAP	E	
DMBDASZE	C	

DBPCB - 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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DBPCB		
0	(0)	8	DBPCBDBD		DBD name
8	(8)	2	DBPCBLEV		Level feedback

The following fields are used for communication from PSBGEN to ACBGEN only.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
8	(8)	1	DBPCBLE1		Flag byte
9	(9)	1	DBPCBLE2		Flag byte
			DBPCBGO	.... ..1.	"X'02'" GO or GOP PROCOPT for PCB
			DBPCBAE	1	"X'01'" Program isolation suppressed for this PCB
10	(A)	2	DBPCBSTC		Status codes
12	(C)	4	DBPCBPRO		DL/I processing options
16	(10)	4	DBPCBJCB		JCB address
			DBPCBTKW	1...	"X'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 the key feedback area
			DBPCBMUL	1	"X'01'" Positioning is multiple (for ACBGEN only)
32	(20)	4	DBPCBNSS		Number of sensitive segments in the PCB (after ACBGEN only)
32	(20)	2	DBPCBSSN		Number of sensitive segments (for ACBGEN only)
34	(22)	2	DBPCBSOF		Offset to the first segment (for ACBGEN only)
36	(24)	256	DBPCBKFD		Key feedback area

Cross Reference

Name	Hex Offset	Hex Value
DBPCB	0	
DBPCBAE	9	01
DBPCBDBD	0	
DBPCBGO	9	02
DBPCBJCB	10	
DBPCBKFD	24	
DBPCBLEV	8	
DBPCBLE1	8	
DBPCBLE2	9	
DBPCBLKY	1C	
DBPCBMKL	1C	
DBPCBMUL	1C	01
DBPCBNSS	20	
DBPCBPRO	C	
DBPCBSFD	14	
DBPCBSOF	22	
DBPCBSSN	20	
DBPCBSTC	A	
DBPCBTKW	10	80

DBT - Data Base Table

DSECT Name: DLZPRDBT

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DBT		
0	(0)	4	DBTSTART		
0	(0)	8	DBTCName		Data base name
8	(8)	4	DBTADBD		Address of loaded DMB or DBD
12	(C)	4	DBTAPCB		Address of primary PCB
16	(10)	4	DBTAJCB		Address of JCB for primary PCB
20	(14)	4			Reserved
24	(18)	4	DBTASPCB		Address of a second PCB for scan
28	(1C)	4	DBTASJCB		Address of JCB for secondary PCB
32	(20)	4			Reserved
36	(24)	4	DBTADMB		Address of DMB
40	(28)	4	DBTFRASZ		No. of blocks in root addressable area
44	(2C)	2	DBTHRAPB		No. of root anchor points per block
46	(2E)	2	DBTHDMBN		DMB number for this data base
48	(30)	1	DBTCID		Data base internal ID
49	(31)	1	DBTGFlag		DBT flag byte
			DBTQSCAN	1...	"X'80'" Scan required, not completed
			DBTQSOPT	.1..	"X'40'" Optional scan required
			DBTQVSAM	..1.	"X'20'" Access method is VSAM
			DBTQHISM	...1	"X'10'" Entry is for HISAM data base
			DBTQHDAM	.... 1...	"X'08'" Entry is for HDAM data base
			DBTQHIDM	.... .1..	"X'04'" Entry is for HIDAM data part

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DBTQXPRI	.... ..1.	"X'02'" Entry is for HIDAM prime index part
			DBTQXSEC	1	"X'01'" Entry is for secondary index DB
50	(32)	8	DBTCKEY		Name of key field for root segment
58	(3A)	2	DBTHSIZ1		Block size for first data set group
60	(3C)	18			Block sizes for 9 more data set groups
78	(4E)	510	DBTOSGT		Offsets in SGT for segments in this DB
588	(24C)				Force full word alignment
			DBTLLEN		"*-DBTSTART" length of a DBT entry

Cross Reference

Name	Hex Offset	Hex Value
DBT	0	
DBTADBD	8	
DBTADMB	24	
DBTAJCB	10	
DBTAPCB	C	
DBTASJCB	1C	
DBTASPCB	18	
DBTCID	30	
DBTCKEY	32	
DBTCName	0	
DBTFRASZ	28	
DBTGFlag	31	
DBTHDMBN	2E	
DBTHRAPB	2C	
DBTHSIZ1	3A	
DBTLLEN	24C	24C
DBTOSGT	4E	
DBTQHDAM	31	08
DBTQHIDM	31	04
DBTQHISM	31	10
DBTQSCAN	31	80
DBTQSOPT	31	40
DBTQVSAM	31	20
DBTQXPRI	31	02
DBTQXSEC	31	01
DBTSTART	0	

DDIR - DMB Directory

DSECT Name: DLZDDIR

The DMB directory contains an entry for every physical 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), entry length (SCDDLIDL), and the number of entries (SCDDLIDN) are contained in the system contents directory (SCD).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DDIR		
			DDIRBGIN	....	"*" Start of DMB directory entry
0	(0)	4	DDIRQE		QE of DMB directory entry
4	(4)	4	DDIRQE2		
			NOENQ	1	"X'01'" No more secondary list ENQ
8	(8)	8	DDIRSYM		DMB symbolic name
DDIRADDR HI BYTE Flag					
			INDEX	1	"X'01'"
			LOGIC	.... ..1.	"X'02'"
			BILT	.... .1..	"X'04'"
			ACTIV	.... 1...	"X'08'"
			LDSEQ	...1	"X'10'"
			LOAD	..1.	"X'20'"
			DMBINCIL	.1..	"X'40'" This DMB already in CIL or built during this run.
			NOTLOAD	1...	"X'80'"
16	(10)	4	DDIRADDR		DMB storage address
20	(14)	4	DDIRBLDL		TTR of DMB work area
24	(18)	2	DDIRSGS		Size of SEGTEB entry
26	(1A)	2			Unused
28	(1C)	2	DDIRCPMP		Number segment table entries
30	(1E)	2	DDIRNUMB		DMB number
32	(20)	1	DDIRCODE		Codes
			DDIRSECL	1...	"X'80'" Secondary locked
			DDIROPEN	.1..	"X'40'" One DCB open
			DDIRWFC	..1.	"X'20'" DMB res and waiting storage
			DDIRIOP	...1	"X'10'" HSAM/SHSAM input OPEN

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DDIROOP	.... 1...	"X'08'" HSAM/SHSAM output open
			DDIRNOSC	.... .1..	"X'04'" Do not schedule
				....	"X'02'" Reserved
			DDIRNOUP	1	"X'01'" Do not schedule updates
33	(21)	1	DDIRCOD2		Codes
			DDIRLRSW	.... 1...	"X'08'" DMB located but not relocated
			DDIR1GRP	.... .1..	"X'04'" DMB first in a shared index
			DDIRGRP	.... ..1.	"X'02'" DMB belongs to a shared index
			DDIRBAD	1	"X'01'" DMB init failed
			DDIRREFR	1...	"X'80'" Index DBD referenced
34	(22)	2	DDIROPT		DMB resident option
36	(24)	8	DDIRDMBN		DMB name converted from DBD name
44	(2C)	4	DDIRDBPT		Addr of entry in DMBName table flags in first byte of DDIRDBPT
			NEWDBD	1...	"X'80'" Index table entries exist
			DBDREL10	.1..	"X'40'" Release 1.0 DBD
			IMSCF	..1.	"X'20'" IMS compatability req
			DDIREND	..11	"*" Last address of DMB direct
			DDIRLEN	..11	"*-DDIR" length of one DMB directory entry

Cross Reference

Name	Hex Offset	Hex Value
ACTIV	8	08
BILT	8	04
DBDREL10	2C	40
DDIR	0	
DDIRADDR	10	
DDIRBAD	21	01
DDIRBGIN	0	00
DDIRBLDL	14	
DDIRCODE	20	
DDIRCOD2	21	
DDIRCPMP	1C	
DDIRDBPT	2C	
DDIRDMBN	24	
DDIREND	2C	30
DDIRGRP	21	02
DDIRIOP	20	10
DDIRLEN	2C	30
DDIRLRSW	21	08
DDIRNOSC	20	04
DDIRNOUP	20	01
DDIRNUMB	1E	
DDIROOP	20	08
DDIROPEN	20	40
DDIROPT	22	
DDIRQE	0	
DDIRQE2	4	
DDIRREFR	21	80
DDIRSECL	20	80
DDIRSGS	18	
DDIRSYM	8	
DDIRWFC	20	20
DDIR1GRP	21	04
DMBINCIL	8	40
IMSCF	2C	20
INDEX	8	01
LDSEQ	8	10
LOAD	8	20
LOGIC	8	02
NEWDBD	2C	80
NOENQ	4	01
NOTLOAD	8	80

DIB - DL/I Interface Block

DSECT Name: DLZDIB

This DSECT describes the HLPI DL/I interface block fields.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZDIB		
0	(0)	4	DIB		
0	(0)	2	DIBVER		Version of user DIB
2	(2)	2	DIBSTAT		DL/I status code
4	(4)	8	DIBSEGM		Target segment name
12	(C)	1	DIBFLAG		DIB flag byte
			DIBWAIT	1111 1111	"X'FF'" task waiting (resource conflict)
13	(D)	1			Reserved
14	(E)	2	DIBSEGLV		Level feedback
16	(10)	2	DIBKFBL		Key feedback area length
18	(12)	2	(3)		Reserved
24	(18)	4			Length is fullword multiple
			DIBCLRLN	...1 .11.	"*-DIBSTAT" Length for clearing DIB
			DIBLEN	...1 1...	"*-DIB" length of DIB

Cross Reference

Name	Hex Offset	Hex Value
DIB	0	
DIBCLRLN	18	16
DIBFLAG	C	
DIBKFBL	10	
DIBLEN	18	18
DIBSEGLV	E	
DIBSEGM	4	
DIBSTAT	2	
DIBVER	0	
DIBWAIT	C	FF
DLZDIB	0	

DIB - DL/I System Interface Block

DSECT Name: DLZDIB

This DSECT describes the HLPI DL/I system interface block fields.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZSDIB		
0	(0)	4	DIBS		System DIB
0	(0)	8	DIBID		DIB identifier 'DLZSDIB'
8	(8)	4	DIBEIPAD		EXEC interface program address (BATCH/MPS only)
12	(C)	4	DIBREGSV(18		Register save area
			DIBRBKWD	...1	"DIBRESSV+4" savearea backward pointer
			DIBRERC	...1 1...	"DIBREGSV+12" savearea for registers 14 thru 12
84	(54)	4	DIBPCBAD		PCB address list address
88	(58)	4	DIBIO		Address of EIP common IOAREA
92	(5C)	4	DIBIOSIZ		Size of EIP common IOAREA
96	(60)	4	DIBPSIZE		IOAREA size required on call
100	(64)	4	DIBCOUNT		DL/I call parm-count
104	(68)	4	DIBPRCNT		Previous GET path call DL/I parm-count
108	(6C)	2	DIBPATHC		Data transfer segment count
110	(6E)	1	DIBSFlag		Flag byte
			DIBPSPLI	1...	"X'80'" PSB generated for PL/I program (online only)
			DIBGPATH	.1..	"X'40'" Previous call was a GET path call
			DIBKF	..1.	"X'20'" Keyfeedback specified
			DIBKFL	...1	"X'10'" Feedbacklen specified
111	(6F)	1	DIBTOTN		Number of calls on previous GET path call
112	(70)	2	DIBNOPCB		Maximum PCB index
114	(72)	2	DIBRTNCD		Failing return code
116	(74)	2	DIBPCBNO		PCB number for currentcall
118	(76)	2			Reserved
120	(78)	4	DIBPATHP		Address of path call header control blocks

124	(7C)	4	DIBKFBAA		Save KFB area address
128	(80)	2	DIBKFBLL		Save KFB length address
132	(84)	4	DIBMSG		Address of message number
136	(88)	4	DIBMSGSC		Address of DL/I status code
140	(8C)	4	DIBMSGRC		Address of failing return code address of statement identifier
144	(90)	4	DIBLUDIB		Address of last user DIB
148	(94)	4	DIBHLPIA		Address of HLPI parm-list
152	(98)	4	DIBPARM		Start of call parm list
152	(98)	4	DIBCNTAD		Address of parm-count
156	(9C)	4	DIBPARM1		Parm 1 = A(function)
160	(A0)	4	DIBPARM2		Parm 2 = A(PCB)
164	(A4)	4	DIBPARM3		Parm 3 = A(IOAREA)
168	(A8)	4	DIBSSAS		Start of SSAS
168	(A8)	4	DIBPARM4		Parm 4 = A(SSA1)
172	(AC)	4	DIBPARM5		Parm 5 = A(SSA2)
176	(B0)	4	DIBPARM6		Parm 6 = A(SSA3)
180	(B4)	4	DIBPARM7		Parm 7 = S(SSA4)
184	(B8)	4	DIBPARM8		Parm 8 = A(SSA5)
188	(BC)	4	DIBPARM9		Parm 9 = A(SSA6)
192	(C0)	4	DIBPARMA		Parm 10 = A(SSA7)
196	(C4)	4	DIBPARMB		Parm 11 = A(SSA8)
200	(C8)	4	DIBPARMC		Parm 12 = S(SSA9)
204	(CC)	4	DIBPARMD		Parm 13 = A(SSA10)
208	(D0)	4	DIBPARME		Parm 14 = A(SSA11)
212	(D4)	4	DIBPARMF		Parm 15 = A(SSA12)
216	(D8)	4	DIBPARMG		Parm 16 = A(SSA13)
220	(DC)	4	DIBPARMH		Parm 17 = A(SSA14)
224	(E0)	4	DIBPARMI		Parm 18 = A(SSA15)
228	(E4)	4			Length is fullword multiple
			DIBSLEN	111. .1..	"*-DIBS" length of system DIB

Cross Reference

Name	Hex Offset	Hex Value
DIBCNTAD	98	
DIBCOUNT	64	
DIBEIPAD	8	
DIBGPATH	6E	40
DIBHLPIA	94	
DIBID	0	
DIBIO	58	
DIBIOSIZ	5C	
DIBKF	6E	20
DIBKFBAA	7C	
DIBKFBLL	80	
DIBKFL	6E	10
DIBLUDIB	90	
DIBMSG	84	
DIBMSGRC	8C	
DIBMSGSC	88	
DIBNOPCB	70	
DIBPARM	98	
DIBPARMA	C0	
DIBPARMB	C4	
DIBPARMC	C8	
DIBPARMD	CC	
DIBPARME	D0	
DIBPARMF	D4	
DIBPARMG	D8	
DIBPARMH	DC	
DIBPARMI	E0	
DIBPARM1	9C	
DIBPARM2	A0	
DIBPARM3	A4	
DIBPARM4	A8	
DIBPARM5	AC	
DIBPARM6	B0	
DIBPARM7	B4	
DIBPARM8	B8	
DIBPARM9	BC	
DIBPATHC	6C	
DIBPATHP	78	
DIBPCBAD	54	
DIBPCBNO	74	
DIBPRCNT	68	
DIBPSIZE	60	
DIBPSPLI	6E	80
DIBRBKWD	C	10
DIBREGSV	C	
DIBRERC	C	18
DIBRTNCD	72	
DIBS	0	
DIBSFLAG	6E	
DIBSLEN	E4	E4
DIBSSAS	A8	
DIBTOTN	6F	
DLZSDIB	0	

DMB - Data Management Block (DMB) Prefix

DSECT Name: DMB

The DMB prefix is described as part of the general structure and description of the data management block (DMB), which is in the DLZIDLI macro.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMB		
0	(0)	2	DMBSIZE		DMB size
			DMBV11	1...	"X'80'" DL/I version 1.1 or later
2	(2)	2	DMBLENTB		Offset from DMB to first PSDB (DMBPSDB)
4	(4)	2	DMBSECTB		Offset from DMB to end of PSDB's+1
6	(6)	1	DMBORG		DMB organization
			DMBSHIS	1	"X'01'" Simple HISAM
			DMBISAM1	.... ..1.	"X'02'" HISAM
			DMBISAM2	11	"X'03'" (Not used in DL/I DOS/VS)
			DMBSSAM	.... .1..	"X'04'" Simple HSAM
			DMBHSAM	.... .1.1	"X'05'" HSAM
			DMBHD	.... .11.	"X'06'" HDAM
			DMBHI	.... .111	"X'07'" HIDAM
			DMBNDEX	.... 1...	"X'08'" Index data base
7	(7)	1	DMBLDDCB		ACB number-1 of sequential data set used to write index records on data base load
7	(7)	1	DMBRES1		(Not used in DL/I DOS/VS)
8	(8)	2	DMBPDATA		Length of system data in index data base (protected)

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
10	(A)	1	DMBPFLG		
			DMBIMSC	1...	"X'80'" IMS compatibility required
11	(B)	1			Reserved
12	(C)	1	DMBNREF		Number of entries in external reference table
12	(C)	4	DMBDALGR		Address of direct algorithm communication table if HDAM (DMBDACS); LRECL number if HSAM
12	(10)		DMBPPRND		"*" End of DMB prefix

Note:
This is also the address of the first ACB extension (DMBACBXT).

			DMBPPRLN		"DMBPPRND-DMB" length of the DMB prefix

Cross Reference

Name	Hex Offset	Hex Value
DMB	0	
DMBDALGR	C	
DMBHD	6	06
DMBHI	6	07
DMBHSAM	6	05
DMBIMSC	A	80
DMBISAM1	6	02
DMBISAM2	6	03
DMBLDDCB	7	
DMBLENTB	2	
DMBNDEX	6	08
DMBNREF	C	
DMBORG	6	
DMBPDATA	8	
DMBPFLG	A	
DMBPPRLN	C	10
DMBPPRND	C	10
DMBRES1	7	
DMBSECTB	4	
DMBSHIS	6	01
DMBSIZE	0	
DMBSSAM	6	04
DMBV11	0	80

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).

RECORD LAYOUT - DPPCB

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	4	DPPCBDBD		The address of the location that contains DBPCBDBD
4	(4)	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	(6)	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	(8)	4	DPPCBLEV		The address of the location that contains DBPCBLEV
12	(C)	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
14	(E)	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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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
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

ALPHABETIC LIST OF FIELD/FLAG NAMES

Name	Hex Offset	Hex Value
DPPCBDBD	0	
DPPCBJCB	20	
DPPCBKFD	34	
DPPCBLEV	8	
DPPCBLKY	2C	
DPPCBPRO	18	
DPPCBSFD	24	
DPPCBSTC	10	
DPPCBNSS	30	

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), which is part of the DLZIDLI macro.

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DSG		
			DSGDSG	....	"*" Start of each DSG section of the JCB
0	(0)	4	DSGDCBA		Address of ACB extension for this data set (KSDS ACB extension if HISAM)
4	(4)	2	DSGDMBNO		DMB number for this DSG
6	(6)	1	DSGDCBNO		ACB number of ACB in DMB (KSDS ACB number if HISAM)
7	(7)	1	DSGINDA		JCB indicators
			DSGDSOLS	1...	"X'80'" This is last DSG in JCB
			DSGDSORI	.1.. .1..	"X'44'" Data set group is root in index
			DSGDSOHD	..1.	"X'20'" Data set group is HDAM
			DSGDSOHI	...1	"X'10'" Data set group is HIDAM
			DSGDSOH2	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			DSGDSOH1	.... .1..	"X'04'" Data set group is HISAM or simple HISAM
			DSGDSOHS	.... ..1.	"X'02'" Data set group is HSAM or simple HSAM
			DSGDSOUP	1	"x'01'" Data set group is SHSAM or SHISAM
8	(8)	4	DSGIRECA		
8	(8)	4	DSGHSADD		HSAM I/O area after open
8	(8)	4	DSGTTR		
12	(C)	2	DSGBOFF		HSAM block size
14	(E)	1	DSGINDB		JCB indicators
			DSGSETLR	1...	"X'80'" (Not used in DL/I DOS/VS)
			DSGGETR	.1..	"X'40'" (Not used in DL/I DOS/VS)
			DSGBATIS	..1.	"X'20'" (Not used in DL/I DOS/VS)

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DSGNXTIS	...1	"X'10'" (Not used in DL/I DOS/VS)
			DSGSETL2	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			DSGGETGT	.... .1..	"X'04'" (Not used in DL/I DOS/VS)
			DSGKEYSR	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
			DSGSTLIS	1	"X'01'" (Not used in DL/I DOS/VS)
15	(F)	1	DSGINDC		JCB indicators
			DSGBLDEL	1...	"X'80'" This DSG belongs to delete/replace
			DSGHDULD	.1.. ...	"X'40'" HD unload is running
			DSGCONST	..1.	"X'20'" Index data set contains constant
			DSGPADKY	...1	"X'10'" Search argument not equal to key length
			DSGDUPS	.... 1...	"S'08'" Non-unique secondary index keys
			DSGSPOST	.... .1..	"X'04'" (Not used in DL/I DOS/VS)
			DSGSWAP	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
			DSGHSWLR	1	"X'01'" HSAM wrong length record
16	(10)	1	DSGINDG		DSG indicators retrieves variable length flags
				1...	"X'80'" Segment prefix has been moved to work area
				.1..	"X'40'" Segment has been completely expanded
				...1	"X'10'" Force complete segment expansion
				.... 1...	"X'08'" The variable length routine has been entered for segment
				.... .1..	"X'04'" Data return call
				.... ..1.	"X'02'" Path return call
			DSGRTNER	1	"X'01'" Used by retrieve
17	(11)	1	(3)		Reserved

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
20	(14)	4	DSGNOSAM		Retrieves HSAM ID
24	(18)	4	DSGLROOT		RBA of current root
			DSGDSEND	...1 11..	"*"
			DSGDSGLN	...1 11..	"DSGDSEND-DSGDSG" length of each DSG section of the JCB

Cross Reference

Name	Hex Offset	Hex Value
DSG	0	
DSGBATIS	E	20
DSGBLDEL	F	80
DSGBOFF	C	
DSGCONST	F	20
DSGDCBA	0	
DSGDCBNO	6	
DSGDMBNO	4	
DSGDSEND	18	1C
DSGDSG	0	00
DSGDSGLN	18	1C
DSGDSOHD	7	20
DSGDSOHI	7	10
DSGDSOHS	7	02
DSGDSOH1	7	04
DSGDSOH2	7	08
DSGDSOLS	7	80
DSGDSORI	7	44
DSGDSOUP	7	01
DSGDUPS	F	08
DSGGETGT	E	04
DSGGETR	E	40
DSGHSADD	8	
DSGHSWLR	F	01
DSGINDA	7	
DSGINDB	E	
DSGINDC	F	
DSGINDG	10	
DSGIRECA	8	
DSGKEYSR	E	02
DSGLROOT	18	
DSGNOSAM	14	
DSGNXTIS	E	10
DSGPADKY	F	10
DSGRTNER	10	01
DSGSETLR	E	80
DSGSETL2	E	08
DSGSPOST	F	04
DSGSTLIS	E	01
DSGSWAP	F	02
DSGTTR	8	

DWR - Data Work Record

DSECT Name: DLZPRDWR

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DWR		
0	(0)	4	DWRSTART		
0	(0)	4	DWRRMOVE		New RBA of a moved segment
4	(4)	4	DWRRCOMP		Old RBA of a segment for compare
8	(8)	2	DWROACT		Offset in ACT which built this record
10	(A)	1	DWRCTYPE		Record type code
11	(B)	1	DWRCSORT		Minor sort key
11	(B)	1	DWRCDSG		DSG of moved segment in K record
12	(C)	6	DWRCSKEY		Update sort key: DB ID, DSG, RBA
12	(C)	1	DWRCRDB		Data base ID of segment to be updated
13	(D)	1	DWRCRDSG		Data set group ID of segment to be update
14	(E)	4	DWRRUPDT		RBA of segment to be updated
			DWRLLEN		*-DWR Length of DWR

Cross Reference

Name	Hex Offset	Hex Value
DWR	0	
DWRCDSG	B	
DWRCRDB	C	
DWRCRDSG	D	
DWRCSKEY	C	
DWRCSORT	B	
DWRCTYPE	A	
DWRLLEN	12	12
DWROACT	8	
DWRRCOMP	4	
DWRRMOVE	0	
DWRRUPDT	E	
DWRSTART	0	

EIPL - Exec Interface Program Parameter List

DSECT Name: DLZEIPL

This DSECT describes the DL/I HLPI interface program parameter list fields.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZEIPL		
0	(0)	4	EIPPARML		DL/I-EIP parameter list
0	(0)	4	EIPERMSG		Address of DL/I message routine (online/batch/MPS)

The following fields are used only in batch/MPS environment

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
4	(4)		EIPEPB0	V-ADDRESS	"V(DLZEIPI)" address of exec interface program (DLZEIPB0)
8	(8)	4	EIPABEND		Address of DL/I ABEND routine

Note:
The following three 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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
12	(C)	4	EIPPCBL		Address of PCB list
16	(10)	4	EIPPLILN		Pointer to length of initial storage area
20	(14)	4	EIPPLISA		Address of initial storage area
			EIPSPCLN	.... 11..	"*-EIPPCBL" length of PL/I parameter list
24	(18)	4	EIPSDIB		Address of system DIB
28	(1C)	1	EIPFLAG		Flag byte
			EIPPLIPS	1...	"X'80'" PSB generated for PL/I program
			EIPPLIPG	.1..	"X'40'" Application program is PL/I
			EIPMPS	..1.	"X'20'" MPS environment
29	(1D)	3			reserved
			EIPLLEN	..1.	"*-EIPPARML" EIP parameter list length

Cross Reference

Name	Hex Offset	Hex Value
DLZEIPL	0	
EIPABEND	8	
EIPEPB0	4	
EIPERMSG	0	
EIPFLAG	1C	
EIPLLEN	1D	20
EIPMPS	1C	20
EIPPARML	0	
EIPPCBL	C	
EIPPLILN	10	
EIPPLIPG	1C	40
EIPPLIPS	1C	80
EIPPLISA	14	
EIPSDIB	18	
EIPSPCLN	14	0C

EXWCB - EXTRACT DEFINEs Work Control Block

DSECT Name: EXWCB

The EXTRACT DEFINES work control block is the work area used to build DEFINE commands for the automatic ISQL EXTRACT DEFINES utility.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	EXWCB		
0	(0)	8	DECWORK		Work area for CVD
8	(8)	4	WORK		Full word work area
12	(C)	4	LINKSAV1		Linkage save area
16	(10)	4	LINKSAV2		
20	(14)	4	LINKSAV3		
24	(18)	4	LINKSAV4		
28	(1C)	4	LINKSAV5		
32	(20)	4	LINKSAV6		
36	(24)	4	LINKSAV7		
40	(28)	4	SAVEBAS1		Variables to save base
44	(2C)	4	SAVEBAS2		Register
48	(30)	4	SAVEBAS3		
52	(34)	4	ASQLD		Address of SQLDSECT
56	(38)	4	ASQLCA		Address of SQLCA
60	(3C)	1	DELIMIT		Delimiter found
61	(3D)	1	DEL1		Three valid delimiters
62	(3E)	1	DEL2		
63	(3F)	1	DEL3		
64	(40)	12	SYNERMSG		Syntax error word
66	(42)	10	WORD		The word
76	(4C)	4	WORDE		Pointer to end of word
80	(50)	4	NEXTPOS		Next default field starting position
84	(54)	4	DPCKLEN		Destination parent concatenated key length
88	(58)	4	DPCKNUM		Destination parent concatenated key number
92	(5C)	4	LPCKLEN		Logical parent concatenated key length
96	(60)	4	TEMPSTR		Used as pointer to start position of concatenated key field

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
100	(64)	4	LENCKFT		Length of concatenated key field table
104	(68)	4	ACKFTAB		Address of first entry in concatenated key field table
108	(6C)	4	LASTCKFE		Address of the last concatenated key field entry that was added to the concatenated key field table
112	(70)	2	CKFTCNT		Number of entries in concatenated key field table
114	(72)	2	COUNTER		Used as counter for indexing into concatenated key field table
116	(74)	2	SEQSAVE		Save for sequence number for routine table
118	(76)	2	RETCD		Return code for GETVIS
118	(76)	1			
119	(77)	1	RETCD2		
120	(78)	4	SAVSQLCD		Save for SQLCode
120	(78)	3	SAVSQLC1		Labels for making last
123	(7B)	1	SAVSQLC2		Byte printable
124	(7C)	8	SAVEERRP		Save for SQLERRP
124	(7C)	1	SAVERRP1		Labels for clearing area
125	(7D)	7	SAVERRP2		To blanks
132	(84)	4	SV1ERRD		Save for first SQLERRD value
132	(84)	3	SV1ERRD1		Labels for making last
135	(87)	1	SV1ERRD2		Byte printable
136	(88)	4	SV2ERRD		Save for second SQLERRD value
136	(88)	3	SV2ERRD1		Labels for making last
139	(8B)	1	SV2ERRD2		Byte printable
140	(8C)	70	SAVEERRM		Save for SQLEERRM
140	(8C)	1	SAVERRM1		Labels for clearing area
141	(8D)	69	SAVERRM2		To blanks
210	(D2)	1	TFLAGS		Flags for execution
			PNUMFLAG	1...	"X'80'" PCB number specified
			COMMA	.1..	"X'40'" Just passed comma

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			COMMAOFF	1.11 1111	"X'BF'" Used to clear comma flag
			NONALPHA	..1.	"X'20'" Looking for PCBnumber
			PSBFLAG	...1	"X'10'" 'PSBName' processed
			PCBFLAG	.... 1...	"X'08'" 'PCBName' processed
			PROCFLAG	.... .1..	"X'04'" 'DLIPROC' processed
			REPFLAG	.... ..1.	"X'02'" 'REPLACE' processed
			USERFLAG	1	"X'01'" 'USERID' processed
211	(D3)	1	PCBFLAGS		Flags for PCB processing
			LOGICAL	1...	"X'80'" Access is logical
212	(D4)	1	SEGFLAGS		Flags for segment processing
			CONCAT	1...	"X'80'" Concatenated segment
			VLC	.1..	"X'40'" Virtual logical child
			FLS	..1.	"X'20'" Field level sensitivity
			XDFLD	...1	"X'10'" This is XDFLD
213	(D5)	1	EFLAGS		Flags for errors
			INVALDEL	1...	"X'80'" Invalid delimiter found
			CONTIN	.1..	"X'40'" Bad continuation
			WARNING1	..1.	"X'20'" Duplicate field name warning to be printed
			WARNING2	...1	"X'10'" Warning to be printed for a field beginning with a non-alphabetic character
			WORDERR	.... 1...	"X'08'" Error occurred while parsing the word
			CANCEL	.... .1..	"X'04'" Indicates a severe SQL error occurred and a cancel is necessary
214	(D6)	1	INITFLGS		SQL connect flag
			CONNSQL	1	"X'01'" SQL/DS connect established
			EXWCBLEN	11.1 .111	"*-EXWCB" length of EXWCB DSECT

Cross Reference

Name	Hex Offset	Hex Value
ACKFTAB	68	
ASQLCA	38	
ASQLD	34	
CANCEL	D5	04
CKFTCNT	70	
COMMA	D2	40
COMMAOFF	D2	BF
CONCAT	D4	80
CONNSQL	D6	01
CONTIN	D5	40
COUNTER	72	
DECWORK	0	
DELIMIT	3C	
DEL1	3D	
DEL2	3E	
DEL3	3F	
DPCKLEN	54	
DPCKNUM	58	
EFLAGS	D5	
EXWCB	0	
EXWCBLEN	D6	D7
FLS	D4	20
INITFLGS	D6	
INVALDEL	D5	80
LASTCKFE	6C	
LENCKFT	64	
LINKSAV1	C	
LINKSAV2	10	
LINKSAV3	14	
LINKSAV4	18	
LINKSAV5	1C	
LINKSAV6	20	
LINKSAV7	24	
LOGICAL	D3	80
LPCKLEN	5C	
NEXTPOS	50	
NONALPHA	D2	20
PCBFLAG	D2	08
PCBFLAGS	D3	
PNUMFLAG	D2	80
PROCFLAG	D2	04
PSBFLAG	D2	10
REPFLAG	D2	02
RETCD	76	
RETCD2	77	
SAVEBAS1	28	
SAVEBAS2	2C	
SAVEBAS3	30	
SAVEERRM	8C	
SAVEERRP	7C	
SAVERRM1	8C	
SAVERRM2	8D	
SAVERRP1	7C	
SAVERRP2	7D	
SAVSQLCD	78	
SAVSQLC1	78	
SAVSQLC2	7B	
SEGFlagS	D4	
SEQSAVE	74	
SV1ERRD	84	
SV1ERRD1	84	
SV1ERRD2	87	
SV2ERRD	88	
SV2ERRD1	88	
SV2ERRD2	8B	
SYNERMSG	40	
TEMPSTR	60	
TFLAGS	D2	
USERFLAG	D2	01
VLC	D4	40
WARNING1	D5	20
WARNING2	D5	10
WORD	42	
WORDE	4C	
WORDERR	D5	08
WORK	8	
XDFLD	D4	10

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FILECB		
0	(0)	4	FCBSTART		
0	(0)	4	FCBADTF		Address of the DTF for this file
4	(4)	4	FCBABUF		Address of the current record
8	(8)	4	FCBAEOD		Address of the end of data routine
12	(C)	4	FCBFRECT		Number of records read or written
16	(10)	4			Reserved
20	(14)	2	FCBHLLRL		Last logical record length
22	(16)	2			Spare half word
24	(18)	2	FCBHLREC		Logical record length
26	(1A)	2	FCBHBLKS		Physical block size
28	(1C)	2	FCBOCREC		Offset of current record in block
30	(1E)	1	FCBGSTAT		File status flags
			FCBQINPT	1...	"X'80'" File is in input mode
			FCBQOUTP	.1..	"X'40'" File is in output mode
31	(1F)	1	FCBGREQU		Request flags
			FCBQOPNI	1...	"X'80'" Open file for input
			FCBQOPNO	.1..	"X'40'" Open file for output
			FCBQGET	..1.	"X'20'" Get next record
			FCBQPUT	...1	"X'10'" Put a record
			FCBQCLOS	.... 1...	"X'08'" Close the file
			FCBLLEN	..1.	"*-FCBSTART" length of a FCB entry

Cross Reference

Name	Hex Offset	Hex Value
FCBABUF	4	
FCBADTF	0	
FCBAEOD	8	
FCBFRECT	C	
FCBGREQU	1F	
FCBGSTAT	1E	
FCBHBLKS	1A	
FCBHLLRL	14	
FCBHLREC	18	
FCBLLEN	1F	20
FCBOCREC	1C	
FCBQCLOS	1F	08
FCBQGET	1F	20
FCBQINPT	1E	80
FCBQOPNI	1F	80
FCBQOPNO	1F	40
FCBQOUTP	1E	40
FCBQPUT	1F	10
FCBSTART	0	
FILECB	0	

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), which is in the DLZIDLI macro.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FDB		
0	(0)	8	FDBSYMBL		Symbolic name
8	(8)	2	FDBOFFST		Field offset from beginning of segment
10	(A)	1	FDBDCENF		Flags
			FDBLAST	1...	"X'80'" Last FDB for this segment
			FDBKEY	.1..	"X'40'" This is segment's sequence field
			FDBEQOK	..1.	"X'20'" Duplicate sequence fields allowed
			FDBSPEC	...1	"X'10'" Special FDB (XDFLD, CK, or SX)
			FDBTYPE	.... .111	"X'07'" Field format bits
			FDBZD	.... .111	"X'07'" Field is zoned decimal
			FDBFP	.... .1..	"X'04'" Field is floating point
			FDBPACK	.... ..1.	"X'02'" Field is packed decimal
			FDBHex	1	"X'01'" Field is hexadecimal
			FDBCHAR	11	"FDBPACK+FDBHex" field is character
11	(B)	1	FDBFLENG		Executable length of field

The following fields describes the /CK system related field

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	3	FDBSYSNM		C'/CK'
3	(3)	5			Remainder of field name
8	(8)	2	FDBOFFCK		Offset from beginning of concatenated key
10	(A)	2	FDBSYSLN		Bits 0-3 = X'0001' bits 4-15 = length-1

The following fields describe the XDFLD

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	8	FDBXDNM		FDB name
8	(8)	2	FDBXDSEC		Offset to the secondary list for this index
10	(A)	1	FDBXDFLG		Flags
			FDBXDLST	1...	"X'80'" Last FDB
			FDBXDSYM	.1..	"X'40'" Pointer is symbolic
			FDBXDSSS	..1.	"X'20'" Pointer contained in source/SUBSEQ data
			FDBXDSPC	...1	"X'10'" Special FDB
			FDBXDCON	.... 1...	"X'08'" Constant present
			FDBXDSSQ	.... .1..	"X'04'" SUBSEQ present
			FDBXDSOR	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
			FDBXDEQ	1	"X'01'" Indexed segment same as indexed source segment
11	(B)	1	FDBXDLEN		Length of search field
			FDBEND	.... 11..	"*" End of FDB entry
			FDBLEN	.... 11..	"FDBEND-FDBSYMBL" length of FDB entry

Cross Reference

Name	Hex Offset	Hex Value
FDB	0	
FDBCHAR	A	03
FDBDCENF	A	
FDBEND	B	0C
FDBEQOK	A	20
FDBFLENG	B	
FDBFP	A	04
FDBHex	A	01
FDBKEY	A	40
FDBLAST	A	80
FDBLEN	B	0C
FDBOFFCK	8	
FDBOFFST	8	
FDBPACK	A	02
FDBSPEC	A	10
FDBSYMBL	0	
FDBSYSLN	A	
FDBSYSNM	0	
FDBTYPE	A	07
FDBXDCON	A	08
FDBXDEQ	A	01
FDBXDFLG	A	
FDBXDLEN	B	
FDBXDLST	A	80
FDBXDNM	0	
FDBXDSEC	8	
FDBXDSOR	A	02
FDBXDSPC	A	10
FDBXDSSQ	A	04
FDBXDSSS	A	20
FDBXDSYM	A	40
FDBZD	A	07

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FER		
0	(0)	1	FERPEC		Entry code
			FERPGET	11.. .111	"C'G'" GET
			FERPPUT	11.1 .111	"C'P'" PUT
1	(1)	1	FERPFNCT		Function code
			FERPRET	11.. .111	"C'G'" retrieve segment conversion
			FERPINS	11.. 1..1	"C'I'" insert
			FERPREP	11.1 1..1	"C'R'" replace
			FERPSSA	111. ..1.	"C'S'" retrieve SSA conversion
			FERPXDF	111. .111	"C'X'" retrieve SSA conversion for XDFKD
2	(2)	1	FERPCSC		Conversion status code
			FERPCSOK	.1..	"C' '" OK
			FERPCSNT	11.. ...1	"C'A'" numeric truncation error
			FERPCSCT	11.. ..1.	"C'B'" character truncation error
			FERPCSFE	11.. ..11	"C'C'" format error
			FERPCSTC	11.. .1..	"D'D'" type conflict
4	(4)	4	FERPPSA		Physical segment address (if variable length points to two byte length field)
8	(8)	2			Reserved
10	(A)	2	FERPPFL		Physical field length (zero if virtual field)
12	(C)	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

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
24	(18)	4	FERPUFA		User field address
28	(1C)	4	FERPFSBA		FSB address
32	(20)	4	FERPUWA(12)		User work area
			FERPEND	.1.1	"*" End of field exit routine interface list
			FERPLEN	.1.1	"FERPEND-FERPEC" length of the field exit routine interface list

Cross Reference

Name	Hex Offset	Hex Value
FER	0	
FERPCSC	2	
FERPCSCT	2	C2
FERPCSFE	2	C3
FERPCSNT	2	C1
FERPCSOK	2	40
FERPCSTC	2	C4
FERPEC	0	
FERPEND	20	50
FERPFNCT	1	
FERPFSBA	1C	
FERPGET	0	C7
FERPINS	1	C9
FERPLEN	20	50
FERPPFA	C	
FERPPFL	A	
FERPPSA	4	
FERPPUT	0	D7
FERPREP	1	D9
FERPRET	1	C7
FERPSSA	1	E2
FERPUFA	18	
FERPUFL	16	
FERPUSA	10	
FERPUWA	20	
FERPXDF	1	E7

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FERT		
0	(0)	8	FERTNAME		Module name
8	(8)	4	FERTRTEP		Module entry point
12	(C)	4	FERTRTLG		Module length
16	(10)	4	FERTPRES		Pointer to next FRT entry
20	(14)	1	FERTFLAG		Flag byte
			FERTDUMP	1...	"X'80'" Control block dumped
21	(15)	3			Reserved
			FERTEND	...1 1...	"*" end of field exit routine table
			FERTLEN	...1 1...	"FERTEND-FERTName" length of field exit routine table

Cross Reference

Name	Hex Offset	Hex Value
FERT	0	
FERTDUMP	14	80
FERTEND	15	18
FERTFLAG	14	
FERTLEN	15	18
FERTNAME	0	
FERTPRES	10	
FERTRTEP	8	
FERTRTLG	C	

FLD - Field Level Description

DSECT Name: FLD

The FLD (field level description) block is used to hold information about fields, operators, and connectors.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FLD		
0	(0)	1	FLDF1		FLD flags
			FLDDATA1	1...	"X'80'" Field qualified on data
			FLDKEY1	.1..	"X'40'" Field qualified on key
			FLDNOCOV	..1.	"X'20'" No conversion for this field
			FLDNXTSM	...1	"X'10'" Next field is the same
			FLDDPAR	.... 1...	"X'08'" Field in destination parent
			FLDLCH	.... .1..	"X'04'" Field in logical child
1	(1)	1	FLDMBR		Encoded operators/connectors
			FLDMEMEQ	1...	"X'80'" Operator has = sign
			FLDMEMLT	.1..	"X'40'" Operator has < sign
			FLDMEMGT	..1.	"X'20'" Operator has > sign
			FLDMEMNE	.11.	"FLDMEMGT+FLDMEMLT" operator is not equal
			FLDMEMAD	.... .1..	"X'04'" AND connector
			FLDMEMOR	.... ..1.	"X'02'" OR connector
			FLDMEMRP	1	"X'01'" Right parenthesis
2	(2)	2	FLDSSAOF		Offset to value area in SSA for this field
4	(4)	1	FLDFLENG		Executable length of field
5	(5)	3			Reserved
			FLDEND	.... 1...	"*"
			FLDLENG	.... 1...	"FLDEND-FLD" length of each FLD entry

Cross Reference

Name	Hex Offset	Hex Value
FLD	0	
FLDDATA1	0	80
FLDDPAR	0	08
FLDEND	5	08
FLDFLENG	4	
FLDF1	0	
FLDKEY1	0	40
FLDLCH	0	04
FLDLENG	5	08
FLDMBR	1	
FLDMEMAD	1	04
FLDMEMEQ	1	80
FLDMEMGT	1	20
FLDMEMLT	1	40
FLDMEMNE	1	60
FLDMEMOR	1	02
FLDMEMRP	1	01
FLDNOCOV	0	20
FLDNXTSM	0	10
FLDSSAOF	2	

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FSB		
0	(0)	8	FSBFLDNM		Field name
0	(0)	4	FSBFDBP		FDB address (ACBGEN only)
4	(4)	2	FSBPCHA		Physical view chain pointer (ACBGEN only)
6	(6)	2	FSBPHYAD		Field physical adjustment factor (ACBGEN only)
8	(8)	2	FSBPVLOC		Displacement in physical segment
10	(A)	1	FSBPVTYP		Physical field type
			FSBLAST	1...	"X'80'" Last FSB
			FSBKEY	.1..	"X'40'" Sequence field
			FSBEQOK	..1.	"X'20'" Duplicate sequence allowed
			FSBDPF	...1	"X'10'" Field is in destination parent
				.... 1...	"X'08'" Reserved
			FSBTYPE	.... .111	"X'07'" Field format bits
			FSBZD	.... .111	"X'07'" Field format is zoned decimal
			FSBFP	.... .1..	"X'04'" Field format is floating point
			FSBCHAR	11	"X'03'" Field format is character
			FSBPACK	.... ..1.	"X'02'" Field format is packed decimal
			FSBHEX	 1	"X'01'" Field format is binary
11	(B)	1	FSBFLAG		Flags
			FSBSSA	1...	"X'80'" Field may be used in an SSA
			FSBOVF	.1..	"X'40'" Field has subfields
			FSBCR	..1.	"X'20'" Conversion required
12	(C)	2	FSBPVLEN		Physical field length (executable)
14	(E)	2	FSBUVLOC		Field displacement in user's view

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
16	(10)	1	FSBUVTYP		User's field type "X'80'" Reserved
			FSBIV	.1..	"X'40'" Initial value specified
			FSBFER	..1.	"X'20'" Field exit routine specified
			FSBVF	...1	"X'10'" Field is virtual
			FSBNR	.... 1...	"X'08'" Replace prohibited
			FSBUZD	.... .111	"X'07'" User field format is zoned decimal
			FSBUFP	.... .1..	"X'04'" User field format is floating point
			FSBUCHAR	11	"X'03'" User field format is character
			FSBUPACK	.... ..1.	"X'02'" User field format is packed decimal
			FSBUHex	1	"X'01'" User field format Is binary
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
			FSBEND	..1.	"*" end of FSB entry
			FSBLEN	..1.	"FSBEND-FSBFLDNM" length of FSB entry

Cross Reference

Name	Hex Offset	Hex Value
FSB	0	
FSBCHAIN	1C	
FSBCHAR	A	03
FSBCR	B	20
FSBDPF	A	10
FSBEND	1C	20
FSBEQOK	A	20
FSBFDBP	0	
FSBFER	10	20
FSBFERTA	18	
FSBFLAG	B	
FSBFLDNM	0	
FSBFP	A	04
FSBHex	A	01
FSBIV	10	40
FSBIVA	14	
FSBKEY	A	40
FSBLAST	A	80
FSBLEN	1C	20
FSBNR	10	08
FSBOVF	B	40
FSBPACK	A	02
FSBPCHA	4	
FSBPHYAD	6	
FSBPVLEN	C	
FSBPVLOC	8	
FSBPVTYP	A	
FSBSSA	B	80
FSBTYPE	A	07
FSBUCHAR	10	03
FSBUFP	10	04
FSBUHEX	10	01
FSBUPACK	10	02
FSBUVLEN	12	
FSBUVLOC	E	
FSBUVTYP	10	
FSBUZD	10	07
FSBVF	10	10
FSBZD	A	07

HLPIL - High Level Program Interface Parameter List

DSCET Name:. DLZHLPIL

This DSECT describes the fields contained in the HLPIL parameter list.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZHLPIL		
0	(0)	4	HLPIARGO		Address of ARGO parameter list
4	(4)	4	HLPIDIBP		Address of user DIB
8	(8)	4	HLPIPSBN		Pointer to PSBName for scheduling call
8	(8)	4	HLPICKID		Pointer to checkpoint ID for checkpoint call
8	(8)	4	HLPIPCBI		Pointer to PCB index number
12	(C)	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	HLPIKFBA		Pointer to key feedback area
32	(20)	4	HLPIKFBL		Pointer to key feedback area length
			HLPIQS	..1.	"*" the next four fields are repeated for each qualification section of a Boolean SSA
32	(20)	4	HLPIOPER		Pointer to the operators

Note:
HLPIOPER actually points to an area that contains a two-byte relational operator field and a one-byte boolean operator field

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			HLPIOPRL	.... ..1.	"2" Relational operator length, same length as SSARO field in DSECT DLZSSA
			HLPIOPBL	1	"1" Boolean operator length, same length as SSABO field in DSECT DLZSSA
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
			HLPIQSLN	...1	"*-HLPIQS" length of repeating section

Cross Reference

Name	Hex Offset	Hex Value
DLZHLPIL	0	
HLPIARG0	0	
HLPICKID	8	
HLPIDIBP	4	
HLPIFLDN	24	
HLPIFLDV	28	
HLPIKFBA	1C	
HLPIKFBL	20	
HLPILFLD	2C	
HLPILIOA	14	
HLPIOFST	18	
HLPIOPBL	20	01
HLPIOPER	20	
HLPIOPRL	20	02
HLPIPCBI	8	
HLPIPSBN	8	
HLPIQS	20	20
HLPIQSLN	2C	10
HLPISEGN	C	
HLPISIOA	10	

IDBD - DBD Directory

DSECT Name: DLZIDBD

The DLZIDBD macro maps the control blocks which are used to pass information about data base structure from the DBD generation step (DBDGEN) to the block builder step (ACBGEN).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DBDDSECT		
0	(0)	1	AMODLEV		DL/I Release level during DBDGEN X'00'=Release 1.0 X'11'=Release 1.1 and later
0	(0)	4	APREFIX		Address of prefix
4	(4)	4	ASEGTAB		Address of SEGTAB
8	(8)	4	AFLDTAB		Address of FLDTAB
12	(C)	4	ALCHILD		Address of LCHILD
16	(10)	4	AEXTDBD		Address of EXTDBD
20	(14)	4	ASORTAB		Address of source segment table
24	(18)	4	ARMVTAB		Address of direct conversion CSECT
28	(1C)	4	AINDXTAB		Address of index secondary list table
32	(20)	4	ADSGCB		Address of DSG control blocks (ACB or DTF)

PREFIX

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	PREFIX		
			PREBGIN	....	"*"
0	(0)	8	PREDBDNM		DBD name
8	(8)	2	PRENOLEV		Number of levels
10	(A)	2	PRENOSEG		Number of segments
12	(C)	1	PREACCES		Access method see below for values
13	(D)	1	PRENODSG		Number of data sets
14	(E)	2	PRENODBD		Number of external data bases referenced
16	(10)	8	PRERNDM		Randomizing algorithm name
24	(18)	2	PRENOLCH		Number of logical children
26	(1A)	2	PREAP		Number of root anchor points
28	(1C)	4	DBDPFRBN		Maximum relative block number (HD)
32	(20)	4	DBDPFBYT		Maximum number bytes in prime area (HD)

'PREACCES' values -- type of organization

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			PREIMSC	1...	"X'80'" IMS compatibility required
			PREACCCD	 1111	"X'0F'" Organization flags
			PRESHIS	1	"1" simple HISAM
			PREISAM1	.... ..1.	"2" HISAM case I
				11	"3" Reserved
			PRESSAM	.... .1..	"4" Simple HSAM
			PREHSAM	.... .1.1	"5" HSAM
			PREHD	.... .11.	"6" HDAM
			PREHI	.... .111	"7" HIDAM data
			PRENDEX	.... 1...	"8" HIDAM index

DMAN entry (one per DSG; same as DMAN DSECT)

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
36	(24)	8	PREDD1		Input file name
44	(2C)	4	PREDEV1		Device type
48	(30)	1	PREID		DMAN number
49	(31)	1	PRENSGA		No of segments in this dataset
50	(32)	2	PREDELTA		Number of free space scan cylinders
52	(34)	2	PRELSL		Longest
54	(36)	2	PRESSL		Shortest segment
56	(38)	2	PRELKL		Longest
58	(3A)	2	PRESKL		Shortest key this data set
60	(3C)	2	PRELRECL		Logical record length
62	(3E)	2	PREBLKSZ		Block size
64	(40)	2	PREOLREC		ESDS logical record length
66	(42)	2	PREOBLKS		ESDS block size
68	(44)	8	PREDD2		Output file name

DMAN DSECT

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMAN		
			DMANBGIN	....	"*"
0	(0)	8	DMNDD1		Input file name
8	(8)	4	DMNDEV1		Device type
12	(C)	1	DMNID		DMAN number

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
13	(D)	1	DMNNSGA		Number of segments in this data set
14	(E)	2	DMNDELTA		Number of free space scan cycles
16	(10)	2	DMNLSL		Longest
18	(12)	2	DMNSSL		Shortest segment
20	(14)	2	DMNLKL		Longest
22	(16)	2	DMNSKL		Shortest key this data set
24	(18)	2	DMNLRECL		Logical record length
26	(1A)	2	DMNBLKSZ		Block size
28	(1C)	2	DMNOLREC		ESDS logical record length
30	(1E)	2	DMNOBLKS		ESDS block size
32	(20)	8	DMNDD2		Output file name
			DMANEND	..1. 1...	"*"
			DMANSZE	..1. 1...	"DMANEND-DMANBGIN"

SEGTAB

Note that this DSECT can be used for both REL 1.0 and 1.1

0	(0)	0	SEGTAB		
			SEGBGIN	....	"*"
0	(0)	1	SEGDSNO		DMAN number
1	(1)	1	SEGPHYCD		Segment code
2	(2)	1	SEGPARPC		Parent segment code
3	(3)	1	SEGLEVEL		Level
4	(4)	1	SEGNOLCH		Number of logical children
5	(5)	1	SEGNOFLD		Number of fields
6	(6)	2	SEGLENG		Data length-segment length for fixed length segments

Max length for variable length segs
Max length for compressible segs

8	(8)	4	SEGFREQ		Frequency X 100
12	(C)	8	SEGSEGNM		Segment name

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
20	(14)	1	SEGFLG1		Pointer description
			CTR	1...	"X'80'" CTR
			PTF	.1..	"X'40'" PT FWD
			PTB	..1.	"X'20'" PT BKD
			PP	...1	"X'10'" PP
			LTF	.... 1...	"X'08'" LT FWD
			LTB	.... .1..	"X'04'" LT BKD
			LP	.... ..1.	"X'02'" LP
			NOTWIN	1	"X'01'" 7 NOTWIN
21	(15)	1	SEGFLG2		Update rules
			PHYISRT	1...	"X'80'" Physical insert
			VIRISRT	.1..	"X'40'" Virtual insert
			LOGISRT	11..	"X'C0'" Logical insert
			PHYDLET	..1.	"X'20'" Physical delete
			VIRDLET	...1	"X'10'" Virtual delete
			LOGDLET	..11	"X'30'" Logical delete
			PHYRPL	.... 1...	"X'08'" Physical replace
			VIRRPL	.... .1..	"X'04'" Virtual replace
			LOGRPL	.... 11..	"X'0C'" Logical replace
			ISRTF	.... ..1.	"X'02'" Insert first
			ISRTL	1	"X'01'" Insert last
			ISRTH	11	"X'03'" Insert here
22	(16)	1	SEGFLG3		
			SEGPRDT	1...	"X'80'" Paired
			SEGRESB1	.1..	"X'40'" Reserved
			SEGRESB2	..1.	"X'20'" Reserved
			SEGTYP64	...1	"X'10'" Type 64 processing required
			SEGPCBT	.... 1...	"X'08'" Parent has PC backward to this segment
			SEGSPARE	.... .111	"X'07'" Number of spare pointers

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
23	(17)	1	SEGFLG4		Number of physical children
24	(18)	4	SEGLCHLD		Offset of first LCHILD
28	(1C)	2	DBDSSN		Number of source segments for segment
30	(1E)	2	DBDSSOFF		Offset to first source segment entry
32	(20)	4	SEGFLDTB		Offset of first field in FLDTAB
36	(24)	2	DBDSPFSZ		Prefix size
38	(26)	2	SEGLENGV		Minimum length or 0 for variable length segment
			SEGEND10	..1. 1...	"*" End of Release 1.0 SEGTAB entry
			SEGSZE10	..1. 1...	"*-SEGBGIN" length of release 1.0 SEGTAB entry

The following fields exist only in Rel 1.1 DBDs

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
40	(28)	4	SEGPACRV		Reserved
44	(2C)	1	SEGPACOP		Segment compaction options
			SEGCPRT	.... 1...	"X'08'" segment has compress routine
			SEGTYPVL	.... .1..	"X'04'" Segment is variable length
			SEGPACKY	.... ..1.	"X'02'" Key expand entry point is defined
			SEGPACIT	1	"X'01'" Initialization entry point is defined
45	(2D)	3	SEGPACRT		Address of segment compact table for compact route
			SEGEND11	..11	"*" End of release 1.1 SEGTAB entry
			SEGSZE11	..11	"*-SEGBGIN" length of release 1.1 entry

FLDTAB

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	FLDTAB		
			FLDBGIN	....	"*"
0	(0)	8	FLDNAME		Name
8	(8)	2	FLDSTART		Start position offset

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
10	(A)	1	FLDFLAG		Flags
			LSTFLD	1...	"X'80'" Last field for a SEGTAB
			KEYFLD	.1..	"X'40'" This field is SEQ Field
			FLDMSEQ	..1.	"X'20'" SEQ field is NON-UNIQUE
			FLDSPEC	...1	"X'10'" Special field
			FLDX	1	"X'01'" Field is hexadecimal
			FLDP	.... ..1.	"X'02'" Field is packed decimal
			FLDC	11	"X'03'" Field is character
			FLDFP	.... .1..	"X'04'" Field is floating point
			FLDZ	.... .111	"X'07'" Field is zoned decimal
11	(B)	1	FLDLEN		Length
12	(C)	8	FLDSNAME		Source field name
20	(14)	4	FLDSEGTB		SEGTAB entry offset
			FLDEND	...1 1...	"*"
			FLDSZE	...1 1...	"FLDEND-FLDBGIN"

LCHILD

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	LCHILD		
			LCHBGIN	....	"*"
0	(0)	8	LCHSEGNM		Name
8	(8)	1	LCHCODE		
			TSGTAB	1...	"128" LCHEDBD address is a SEGTAB entry
				1	logical insert rule - last (same as ISRTL)
				.... ..1.	logical insert rule - first (same as ISRTF)
				11	logical insert rule - here (same as ISRTH)
8	(8)	4	LCHEDBD		Offset to EXTDBD or SEGTAB entry

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
12	(C)	1	LCHFLAG		
			LSTLCH	1...	"X'80'" last entry for SEGTAB
			LCHPRI	.1..	"X'40'" primary HIDAM index definition (BB only)
			LCHIEN	..1.	"X'20'" indexing segment definition
			LCHTKPH	...1	"X'10'" LPCK is carried physically
			LCHLP	.... 1...	"X'08'" entry is LP definition
			LCHNDX	.... .1..	"X'04'" indexed segment definition
			LCHLCF	.... ..1.	"X'02'" Single logical child forward pointer used
			LCHLCB	1	"X'01'". Double logical child forward/backward pointer

If bit 2 of LCHFLAG is set;
bits 5-7 have the following meaning:

Bit 5 Blank values are not indexed.
Bit 6 Zero values are not indexed.
Bit 7 LCHIBYTE values are not indexed.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
13	(D)	1	LCHIBYTE		Non-index value
14	(E)	2	LCHPRDSG		Offset to paired segment
16	(10)	8	LCHFLDNM		Indexed field name
			LCHEND	...1 1...	"*"
			LCHSZE	...1 1...	"LCHEND-LCHBGIN"

EXTDBD

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	EXTDBD		
0	(0)	8	EXTDBNM		
8	(8)	4	EXTRSVD		
			EXDBDSZ	.... 11..	"*-EXTDBD"

SORTAB

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DBDSORTB		
0	(0)	8	DBDSORNM		Source segment name
8	(8)	1	DBDSSFLG		Source flag
			DBDSSK	1...	"X'80'" Data option=key
			DBDSSP	.1..	"X'40'" =path
			DBDSSD	..1.	"X'20'" =data

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
8	(8)	4	DBDSSDBO		Offset to database entry
			DBDSORSZ	.... 11..	"*-DBDSORTB"

INDEXTAB

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	INDXTAB		
0	(0)	1	INDXFLAG		Index table flag byte
1	(1)	3			Remaining fields are as defined
4	(4)	4	(3)		For DMB secondary lists
			INDXTBSZ	...1	"*-INDXTAB" size of one entry

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
ADSGCB	20		FLDFP	A	04
AEXTDBD	10		FLDLEN	B	
AFLDTAB	8		FLDMSEQ	A	20
AINDXTAB	1C		FLDNAME	0	
ALCHILD	C		FLDP	A	02
AMODLEV	0		FLDSEGTB	14	
APREFIX	0		FLDSNAME	C	
ARMVTAB	18		FLDSPEC	A	10
ASEGTAB	4		FLDSTART	8	
ASORTAB	14		FLDSZE	14	18
DBDDSECT	0		FLDTAB	0	
DBDPFBYT	20		FLDX	A	01
DBDPFRBN	1C		FLDZ	A	07
DBDSORNM	0		INDXFLAG	0	
DBDSORSZ	8	0C	INDXTAB	0	
DBDSORTB	0		INDXTBSZ	4	10
DBDSPFSZ	24		ISRTF	15	02
DBDSSD	8	20	ISRTL	15	01
DBDSSDBO	8		KEYFLD	A	40
DBDSSFLG	8		LCHBGIN	0	00
DBDSSK	8	80	LCHCODE	8	
DBDSSN	1C		LCHEDBD	8	
DBDSSOFF	1E		LCHEND	10	18
DBDSSP	8	40	LCHFLAG	C	
DMAN	0		LCHFLDNM	10	
DMANBGIN	0	00	LCHIBYTE	D	
DMANEND	20	28	LCHIEN	C	20
DMANSZE	20	28	LCHILD	0	
DMNBLKSZ	1A		LCHLCB	C	01
DMNDD1	0		LCHLCF	C	02
DMNDD2	20		LCHLP	C	08
DMNDELTA	E		LCHNDX	C	04
DMNDEV1	8		LCHPRDSG	E	
DMNID	C		LCHPRI	C	40
DMNLKL	14		LCHSEGNM	0	
DMNLRECL	18		LCHSZE	10	18
DMNLSL	10		LCHTKPH	C	10
DMNNSGA	D		LSTFLD	A	80
DMNOBLKS	1E		LSTLCH	C	80
DMNOLREC	1C		NOTWIN	14	01
DMNSKL	16		PREACCCD	20	0F
DMNSSL	12		PREACCES	C	
EXDBDSZ	8	0C	PREAP	1A	
EXTDBD	0		PREBGIN	0	00
EXTDBNM	0		PREBLKSZ	3E	
EXTRSVD	8		PREDBDNM	0	
FLDBGIN	0	00	PREDD1	24	
FLDC	A	03	PREDD2	44	
FLDEND	14	18	PREDELTA	32	
FLDFLAG	A		PREDEV1	2C	

Cross Reference

Name	Hex Offset	Hex Value
PREFIX	0	
PREHD	20	06
PREHI	20	07
PREHSAM	20	05
PREID	30	
PREIMSC	20	80
PREISAM1	20	02
PRELKL	38	
PRELRECL	3C	
PRELSL	34	
PRENDEX	20	08
PRENODBD	E	
PRENODSG	D	
PRENOLCH	18	
PRENOLEV	8	
PRENOSEG	A	
PRENSGA	31	
PREOBLKS	42	
PREOLREC	40	
PRERNDM	10	
PRESHIS	20	01
PRESKL	3A	
PRESSAM	20	04
PRESSL	36	
SEGBGIN	0	00
SEGCPRT	2C	08
SEGDSNO	0	
SEGEND10	26	28
SEGEND11	2D	30
SEGFLDTB	20	
SEGFLG1	14	
SEGFLG2	15	
SEGFLG3	16	
SEGFLG4	17	
SEGFREQ	8	
SEGLCHLD	18	
SEGLENG	6	
SEGLENGV	26	
SEGLEVEL	3	
SEGNOFLD	5	
SEGNOLCH	4	
SEGPACIT	2C	01
SEGPACKY	2C	02
SEGPACOP	2C	
SEGPACRT	2D	
SEGPACRV	28	
SEGPARPC	2	
SEGPCBT	16	08
SEGPHYCD	1	
SEGPRDT	16	80
SEGRESB1	16	40
SEGRESB2	16	20
SEGSEGNM	C	
SEGSPARE	16	07
SEGSZE10	26	28
SEGSZE11	2D	30
SEGTAB	0	
SEGTYPVL	2C	04
SEGTYP64	16	10
TSGTAB	8	80

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	JCB		
0	(0)	4	JCBLEVTB		Address of level table
4	(4)	4	JCBLEVND		Address of end of level table+1
8	(8)	4	JCBSDB1		Address of first SDB entry (root's)
12	(C)	4	JCBSDBND		Address of end of SDB's+1
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 calls 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
FUNCGHN	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
31	(1F)	1	JCBPREVR		Prior return code (right byte)

32	(20)	4	JCBLEV1C		Address of 1st level table entry in call; address of lowest level table entry successfully processed by retrieve; if hi-order byte = X'80' open error
36	(24)	2	JCBSIZE		PCB + JCB size
38	(26)	2	JCBMKYL		Maximum length of key feedback area
40	(28)	4	JCBRES1		Action-modules work area

The following breakdown of JCBRES1 identifies its use by the CALL Analyzer

40	(28)	1	JCBRES11		First flag byte
			JCBNSSA	1...	"X'80'" No SSAs
			JCBQSSA	.1..	"X'40'" Qualified SSAs
			JCBUQSSA	..1.	"X'20'" Unqualified SSAs
			JCBMSSA	...1	"X'10'" Multiple SSAs
			JCBMUSSA	.... 1...	"X'08'" Multiple unqualified SSAs
			JCBQAUQ	.... .1..	"X'04'" Qualified SSA after an unqualified SSA
			JCBLSSAQ	.... ..1.	"X'02'" Last SSA qualified
41	(29)	1	JCBRES12		Second flag byte
			JCBCCALL	.... .1..	"X'04'" Call has C command code

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			JCBTCALL	.... ..1.	"X'02'" Call has T command code
			JCBLV1C	1	"X'01'" JCBLEVLC has been filled on this call
42	(2A)	1	JCBRES13		Third flag byte
			JCBALQD	1...	"X'80'" Any level qualified on data
			JCBALD	.1..	"X'40'" Any level had D command code
			JCBQSAD	..1.	"X'20'" Qualified SSA follows D command code
43	(2B)	1	JCBRES14		Fourth flag byte
			JCBFNSL	1...	"X'80'" Field is not in sub list
			JCBQFLP	.1..	"X'40'" Qualification field is in logical parent
			JCBTSKF	1	"X'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
			JCBTARPR	1...	"X'80'" DLZPOST update twin pointers only
			JCBDEFDL	.1..	"X'40'" Re-insert of a deleted segment
			JCBRETDL	..1.	"X'20'" (Not used in DL/I DOS/VS)
			JCBTAREX	...1	"X'10'" Re-position for GN (no SSA) with multiple positioning
			JCBMLPOS	.... 1...	"X'08'" Retrieve keeping multiple positions
			JCBSGRET	.... .1..	"X'04'" Used in positioning after not found
			JCBRTIST	.... ..1.	"X'02'" Retrieve positioning for insert
			JCBRDREQ	1	"X'01'" DLZSKPG start at next occurence of segment
61	(3D)	1	JCBORGN		Open switch and composite organization of all SDBs in the JCB

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			JCBOPEN	1...	"X'80'" Open done for all data sets in the JCB
			JCBORGRI	.1.. .1..	"X'44'"Organization is root of index
			JCBORGHD	..1.	"X'20'" Organization is HDAM
			JCBORGHI	...1	"X'10'" Organization is HIDAM
			JCBORGH2	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			JCBORGSH	.... .1.1	"X'05'" Organization is SIMPLE HISAM
			JCBORGH1	.... .1..	"X'04'" Organization is HISAM
			JCBORGHS	.... ..1.	"X'02'" Organization is HSAM
			JCBORGSS	1	"X'01'" Organization is SIMPLE HSAM
62	(3E)	1	JCBRWKF		Retrieve's working-function
63	(3F)	1	JCBPRESF		Present coded function (see function codes DSECT for contents)
64	(40)	1	JCBLVT		Switches used in accessing segments via DLZSKPG routine (see routine prolog for details)
			JCBDOPI	.... 1...	"X'08'" program isolation is to be done for associated PCB
			JCBALLEX	.... .1..	"X'04'" All sensitive segments have exclusive intent
			JCBNMFDB	.... ..1.	"X'02'" Field name found in FDB
			JCBFLS	1	"X'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
68	(44)	4	JCBSTOR1		Insert's use across I/O or calls

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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 position 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		
148	(94)	1	JCBWK12A		Program isolation switches (retrieve only)
			JCBNODEQ	1...	"X'80'" No DEQ processing, all level table entries empty after CHKP, TERM, etc.
			JCBRAP	.1..	"X'40'" Root anchor point enqueued (HDAM only)
			JCBPCHK	..1.	"X'20'" DLZPCHK calling DLZPOST enqueue not required
			JCBPPENQ	...1	"X'10'" DLZKDTL enqueued on physical parent searching on data field

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			JCBNTFD	.... 1...	"X'08'" DLZPCHK processing not found condition
			JCBSKPG	.... .1..	"X'04'" DLZDEQ should release all outstanding enqueues
			JCBDLET	.... ..1.	"X'02'" ENQ/DEQ required in DLZPCHK due to delete
			JCBISRT	1	"X'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

End of JCB prefix - beginning of primary DSG

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			JCBDSG	1.1. .1..	"*" 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
			JCBDSOLS	1...	"X'80'" This is last DSG in JCB
			JCBDSORI	.1.. .1..	"X'44'" Data set group is root in index
			JCBDSOHD	..1.	"X'20'" Data set group is HDAM
			JCBDSOHI	...1	"X'10'" Data set group is HIDAM
			JCBDSOH2	.... 1...	"x'08'" (Not used in DL/I DOS/VS)
			JCBDSOH1	.... .1..	"X'04'" Data set group is HISAM or simple HISAM
			JCBDSOHS	.... ..1.	"X'02'" Data set group is HSAM or simple HSAM
			JCBDSOUP	1	"X'01'" Data set group is SHSAM or SHISAM

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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		JCB indicators
			JCBSETLR	1...	"X'80'" (Not used in DL/I DOS/VS)
			JCBGETR	.1..	"X'40'" (Not used in DL/I DOS/VS)
			JCBBATIS	..1.	"X'20'" (Not used in DL/I DOS/VS)
			JCBNXTIS	...1	"X'10'" (Not used in DL/I DOS/VS)
			JCBSETL2	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			JCBGETGT	.... .1..	"X'04'" (Not used in DL/I DOS/VS)
			JCBKEYSR	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
			JCBSTLIS	1	"X'01'" (Not used in DL/I DOS/VS)
179	(B3)	1	JCBINDC		JCB indicators
			JCBBLDEL	1...	"X'80'" This DSG belongs to delete/replace
			JCBHDULD	.1..	"X'40'" HD unload is running
			JCBCONST	..1.	"X'20'" Index data set contains constant
			JCBPADKY	...1	"X'10'" Search argument not equal to key length
			JCBDUPS	.... 1...	"X'08'" Non-unique secondary index keys
			JCBSPOST	.... .1..	"X'04'" (Not used in DL/I DOS/VS)
			JCBSWAP	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
				1	"X'01'" HSAM wrong length record

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
180	(B4)	1	JCBINDG		JCB indicators retrieve variable length flags
			JCBPREM	1...	"X'80'" Segment prefix has been moved to work area
			JCBDATX	.1..	"X'40'" Segment has been completely expanded
			JCBKEYX	..1.	"X'20'" (Not used in DL/I DOS/VS)
			JCBXP	...1	"X'10'" Force complete segment expansion
			JCBVL	.... 1...	"X'08'" The variable length routine has been entered for segment
			JCBRETD	.... .1..	"X'04'" Data return call
			JCBCOMMD	.... ..1.	"X'02'" Path return call
			JCBRTNER	1	"X'01'" (Not used in DL/I DOS/VS)
181	(B5)	1	(3)		Reserved
184	(B8)	4	JCBNOSAM		Retrieve's HSAM ID
188	(BC)	4	JCBLROOT		RBA of current root
			JCBDSEND	11..	"*"
			JCBPRLEN	1.1. .1..	"JCBDSG-JCB" length of JCB prefix
			JCBDSGLN	...1 11..	"JCBDSEND-JCBDSG" length of each DSG section of JCB

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
JCB	0		JCBLV1C	29	01
JCBALD	2A	40	JCBMKYL	26	
JCBALLEX	40	04	JCBMLPOS	3C	08
JCBALQD	2A	80	JCBMSSA	28	10
JCBBATIS	B2	20	JCBMUSSA	28	08
JCBBLDEL	B3	80	JCBNMFDB	40	02
JCBCCALL	29	04	JCBNODEQ	94	80
JCBCode	3C		JCBNOSAM	B8	
JCBCOMMD	B4	02	JCBNSSA	28	80
JCBCONST	B3	20	JCBNTFD	94	08
JCBDATX	B4	40	JCBNXTIS	B2	10
JCBDCBA	A4		JCBOPEN	3D	80
JCBDCBNO	AA		JCBORGHD	3D	20
JCBDEFDL	3C	40	JCBORGHI	3D	10
JCBDLET	94	02	JCBORGHS	3D	02
JCBDMBNO	A8		JCBORGH1	3D	04
JCBDOPI	40	08	JCBORGH2	3D	08
JCBDSEND	BC	C0	JCBORGN	3D	
JCBDSG	A0	A4	JCBORGRI	3D	44
JCBDSGLN	BC	1C	JCBORGSH	3D	05
JCBDSOHD	AB	20	JCBORGSS	3D	01
JCBDSOHI	AB	10	JCBPADKY	B3	10
JCBDSOHS	AB	02	JCBPC	42	
JCBDSOH1	AB	04	JCBPCHK	94	20
JCBDSOH2	AB	08	JCBPOP	43	
JCBDSOLS	AB	80	JCBPPENQ	94	10
JCBDSORI	AB	44	JCBPREM	B4	80
JCBDSOUP	AB	01	JCBPRESF	3F	
JCBDUPS	B3	08	JCBPREVF	1E	
JCBFLS	40	01	JCBPREVR	1F	
JCBFNSL	2B	80	JCBPRLEN	BC	A4
JCBGETGT	B2	04	JCBQAUQ	28	04
JCBGETR	B2	40	JCBQFLP	2B	40
JCBHDULD	B3	40	JCBQSAD	2A	20
JCBHSADD	AC		JCBQSSA	28	40
JCBINDA	AB		JCBRAP	94	40
JCBINDB	B2		JCBRDREQ	3C	01
JCBINDC	B3		JCBRES1	28	
JCBINDG	B4		JCBRES11	28	
JCBIRECA	AC		JCBRES12	29	
JCBISRT	94	01	JCBRES13	2A	
JCBKEYSR	B2	02	JCBRES14	2B	
JCBKEYX	B4	20	JCBRES2	2C	
JCBLEVND	4		JCBRES3	30	
JCBLEVTB	0		JCBRES4	34	
JCBLEV1C	20		JCBRES5	38	
JCBLROOT	BC		JCBRETD	B4	04
JCBLSSAQ	28	02	JCBRETDL	3C	20
JCBLVC	41		JCBRTIST	3C	02
JCBLVT	40		JCBRTNER	B4	01

Cross Reference

Name	Hex Offset	Hex Value
JCBRWKF	3E	
JCBSDBND	C	
JCBSDB1	8	
JCBSETLR	B2	80
JCBSETL2	B2	08
JCBSGRET	3C	04
JCBSIZE	24	
JCBSKPG	94	04
JCBSPOST	B3	04
JCBSTLIS	B2	01
JCBSTOR1	44	
JCBSTOR2	48	
JCBSTOR3	4C	
JCBSTOR4	50	
JCBSTOR5	54	
JCBSTOR6	58	
JCBSTOR7	5C	
JCBSTOR8	60	
JCBSWAP	B3	02
JCBTAREX	3C	10
JCBTARPR	3C	80
JCBTCALL	29	02
JCBTRACE	10	
JCBTSKF	2B	01
JCBTTR	AC	
JCBUQSSA	28	20
JCBVL	B4	08
JCBWKR0	64	
JCBWKR1	68	
JCBWKR10	8C	
JCBWKR11	90	
JCBWKR12	94	
JCBWKR13	98	
JCBWKR14	9C	
JCBWKR15	A0	
JCBWKR2	6C	
JCBWKR3	70	
JCBWKR4	74	
JCBWKR5	78	
JCBWKR6	7C	
JCBWKR7	80	
JCBWKR8	84	
JCBWKR9	88	
JCBWK12A	94	
JCBWK12B	95	
JCBXP	B4	10

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	LEV		
0	(0)	1	LEVLEV		Level number
1	(1)	1	LEVPC		Current physical segment code

Note:
This portion of the level table, once set by RETRIEVE/INSERT is never cleared to zeros, only changed as needed.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
2	(2)	2	LEVSEGOF		Segment's physical code offset from start of record (relative offset to segment from start of buffer)
4	(4)	4	LEVTTR		Relative byte address
8	(8)	4	LEVSDB		SDB entry address for current segment physical code in this entry
12	(C)	1	LEVF1		Flags
			LEVDLET	1...	"X'80'" Segment at this level newly deleted
			LEVEMPTY	.1..	"X'40'" This level table entry is empty
			LEVHELD	..1.	"X'20'" Segment at this level in hold status
			LEVHIER	...1	"X'10'" Segment at this level in hierarchic path (HISAM only)
			LEVDATA	.... 1...	"X'08'" Segment at this level moved to user
			LEVPLAST	.... .1..	"X'04'" Segment is last of type for parent
			LEVPFRST	.... ..1.	"X'02'" Segment is first of type for parent
			LEVLAST	1	"X'01'" This is last level table for PCB
13	(D)	1	LEVF2		Flags
			LEVCDB	1...	"X'80'" Verify enqueues required in data base of current segment

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			LEVNFPOS	.1..	"X'40'" Level has not found position for higher level
			LEVEOD	..1.	"x'20'" EOD flag
				...1	"X'10'" Reserved
			LEVCONT	.... 1...	"X'08'" The SSA at this level allows retrieve to obtain the next sequential segment
			LEVSTOP	.... .1..	"X'04'" Used to determine the setting of LEVCONT by retrieve
			LEVLSW	.... ..1.	"X'02'" Used by retrieve
			LEVNDB	1	"X'01'" Verify enqueues required in destination parents data base
14	(E)	2	LEVUSEOF		Offset of segment in user's I/O area (PSTUSER)

Note:
Fields LEVNUPC through LEVSSA is a description of the SSA set by the Call Analyzer for this entry.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
16	(10)	1	LEVNUPC		Physical code of requested segment
17	(11)	1	LEVF3		Flags
			LEVISRT	1...	"X'80'" Inserting at this level (set by retrieve)
			LEVHOLD	...1	"X'10'" At least one Boolean expression in range at this level
			LEVPSUDO	.... 1...	"X'08'" This is pseudo SSA filling gap
			LEVDATA1	.... .1..	"X'04'" At least one member qualified on data
			LEVKEY1	.... ..1.	"X'02'" Every Boolean set has at least one key field
			LEVNOCOV	1	"X'01'" No conversion to be done for this segment

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
18	(12)	1	LEVF4		Flags
			LEVCOMMT	1...	"X'80'" T command code retrieve by direct address
			LEVCOMMC	.1..	"X'40'" C command code qualifier is concatenated key
			LEVCOMMX	..1.	"X'20'" X command code index maintenance internal call
			LEVCOMMV	.... ..1.	"X'02'" V command code maintain existing position at all levels
			LEVCOMMU	1	"X'01'" U command code maintain existing position this level
19	(13)	1	LEVF5		Flags
				1...	"X'80'" Reserved
			LEVCOMMP	.1..	"X'40'" (Not used in DL/I DOS/VS)
			LEVCOMMF	..1.	"X'20'" F command code get first of segment type
			LEVCOMML	...1	"X'10'" L command code get last of segment type
			LEVCOMMA	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			LEVCOMMD	.... .1..	"X'04'" D command code transfer data this level
			LEVCOMMN	.... ..1.	"X'02'" N command code do not replace this level
			LEVCOMMQ	1	"X'01'" Q command code enqueue on segment
20	(14)	1	LEVMEMBR		Switch for each member
				1...	"X'80'" Reserved
				.1..	"X'40'" Reserved
				..1.	"X'20'" Reserved
				...1	"X'10'" Reserved
			LEVMEMAC	.... 1...	"X'08'" This member in use
				.... .1..	"X'04'" Reserved
				.... ..1.	"X'02'" Reserved
				1	"X'01'" Reserved

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
21	(15)	3			Reserved
24	(18)	4	LEVFLD		Pointer to first FLD entry (SSA unqualified if zeroes)
28	(1C)	4	LEVNUSDB		SSA's SDB address
32	(20)	4	LEVSSA		SSA's left parenthesis (position address
			LEVEND	..1. .1..	"*" end of level table entry
			LEVLEN	..1. .1..	"LEVEND-LEVLEV" length of level table entry

Cross Reference

Name	Hex Offset	Hex Value
LEV	0	
LEVCDB	D	80
LEVCOMMA	13	08
LEVCOMMC	12	40
LEVCOMMD	13	04
LEVCOMMF	13	20
LEVCOMML	13	10
LEVCOMMN	13	02
LEVCOMMP	13	40
LEVCOMMQ	13	01
LEVCOMMT	12	80
LEVCOMMU	12	01
LEVCOMMV	12	02
LEVCOMMX	12	20
LEVCONT	D	08
LEVDATA	C	08
LEVDATA1	11	04
LEVDLET	C	80
LEVEMPTY	C	40
LEVEND	20	24
LEVEOD	D	20
LEVFLD	18	
LEVF1	C	
LEVF2	D	
LEVF3	11	
LEVF4	12	
LEVF5	13	
LEVHELD	C	20
LEVHIER	C	10
LEVHOLD	11	10
LEVISRT	11	80
LEVKEY1	11	02
LEVLAST	C	01
LEVLEN	20	24
LEVLEV	0	
LEVLSW	D	02
LEVMEMAC	14	08
LEVMEMBR	14	
LEVNDB	D	01
LEVNFPOS	D	40
LEVNOCOV	11	01
LEVNUPC	10	
LEVNUSDB	1C	
LEVPC	1	
LEVPFRST	C	02
LEVPLAST	C	04
LEVPSUDO	11	08
LEVSDB	8	
LEVSEGOF	2	
LEVSSA	20	
LEVSTOP	D	04
LEVTTR	4	
LEVUSEOF	E	

MPC Partition Table Entry

DSECT Name: MPCPT

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	MPCPT		
0	(0)	4	MPCDELIM		MPCPT delimiter field
0	(0)	1	MPCFLAG		MPC activity flags
			MPCPACT	1...	"X'80'" Partition active indicator
			MPCERR	.1..	"X'40'" Error condition encountered on DL/I scheduling call, or BPC attach failure
			MPCTSTP	..1.	"X'20'" Stop transaction indicator
			MPCPSTP	...1	"X'10'" Stop partition indicator
			MPCABWT	.... 1...	"X'08'" Indicates that MPC should wait on the ABEND XECB in the event of a BPC ABEND
					"X'04'" Not used
					"X'02'" Not used
					"X'01'" Not used
1	(1)	1	MPCRC1		Error return code from TCAFCTR
2	(2)	1	MPCRC2		Error return code from TCADLTR
3	(3)	1	MPCPID		XECB identifier generated by DLZMPC00
4	(4)	4	MPCTCA		Address of TCA
8	(8)	4	MPCSXECB		Address of stop partition XECB (DLZXCB01)
12	(C)	4	MPCAXECB		Address of partition ABEND XECB (DLZXCBN3)
16	(10)	2			Reserved
18	(12)	2	MPCTSQE		TSQ entry number
20	(14)	1	MPCFLAG1		MPC activity flags
			MPCCNBPC	1...	"X'80'" Cancel BPC at stop transaction when MPS batch partition is not active

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			MPCNOVER	.1..	"X'40'" Checkpoint ID verification not possible during restart
			MPCWNGCP	..1.	"X'20'" Wrong checkpoint ID used on restart
			MPCCKP	...1	"X'10'" A combined checkpoint was issued
			MPCABND	.... 1...	"X'08'" Job abnormally ended if combined checkpoint was not issued, then reinitialize TSQ entry
			MPCRST	.... .1..	"X'04'" MPS restart processing on for this job
21	(15)	2	MPCPIDHX		Partition identifier
23	(17)	1	MPCRSTRC		MPS restart return code for batch partition
			MPCRST0	....	"0" MPS restart normal function
			MPCRST1	1	"1" CICS/VS journaling not active
			MPCRST2	.... ..1.	"2" Temporary storage processing not available or error
24	(18)	4	MPCCPID		Correct checkpoint ID to be used for restart
			MPCPTLN	...1 11..	"*-MPCPT" length of a partition table entry
			MPCNPTE	.... 11..	"12" number of partition table entries defined by DLZMPC00

Cross Reference

Name	Hex Offset	Hex Value
MPCABND	14	08
MPCABWT	0	08
MPCAXECB	C	
MPCCKP	14	10
MPCCNBPC	14	80
MPCCPID	18	
MPCDELIM	0	
MPCERR	0	40
MPCFLAG	0	
MPCFLAG1	14	
MPCNOVER	14	40
MPCNPTE	18	0C
MPCPACT	0	80
MPCPID	3	
MPCPIDHX	15	
MPCPSTP	0	10
MPCPT	0	
MPCPTLN	18	1C
MPCRC1	1	
MPCRC2	2	
MPCRST	14	04
MPCRSTRC	17	
MPCRST0	17	00
MPCRST1	17	01
MPCRST2	17	02
MPCSXECB	8	
MPCTCA	4	
MPCTSQE	12	
MPCTSTP	0	20
MPCWNGCP	14	20

MPC - Start Partition DLZXCB02

DSECT Name: MPCSPART

The MPCSPART maps the start partition XECB parameter list.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	MPCSPART		
0	(0)	4	MPCXECBS		DLZXCB02 XECB
4	(4)	4	MPCSPRO		Address of start partition processing routine in DLZMPC00
8	(8)	4	MPCPTABE		Address of next partition table entry to be used for batch partition start

Cross Reference

Name	Hex Offset	Hex Value
MPCPTABE	8	
MPCSPART	0	
MPCSPRO	4	
MPCXECBS	0	

PATH - Path Header Control Block

DSECT Name: DLZPATH

This DSECT describes the fields for DL/I HLPI PATH header control block.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZPATH		
0	(0)	4			Full word alignment
0	(0)	4	PATHSSAP		Address of path SSA appendage
4	(4)	4	PATHPRCT		Previous GET path call DL/I parameter count
8	(8)	1	PATHFlag		Flag byte
			PATHCALL	1...	"X'80'" Previous call was a GET path call
9	(9)	1	PATHTOTN		Number of calls on previous GET path call
10	(A)	2			Reserved
			PATHLEN	.... 11..	"*-PATHSSAP" length of path header control block

Cross Reference

Name	Hex Offset	Hex Value
DLZPATH	0	
PATHCALL	8	80
PATHFlag	8	
PATHLEN	A	0C
PATHPRCT	4	
PATHSSAP	0	
PATHTOTN	9	

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).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DBPCB		
0	(0)	8	DBPCBDBD		DBD name
8	(8)	2	DBPCBLEV		Level feedback

THE following fields are used for communication from PSBGEN to ACBGEN only.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
8	(8)	1	DBPCBLE1		Flag byte
9	(9)	1	DBPCBLE2		Flag byte
			DBPCBGO	.... ..1.	"X'02'" GO or GOP PROCOPT for PCB
			DBPCBAE	1	"X'01'" Program isolation suppressed for this PCB
10	(A)	2	DBPCBSTC		Status codes
12	(C)	4	DBPCBPRO		DL/I processing options
16	(10)	4	DBPCBJCB		JCB address
			DBPCBTKW	1...	"X'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 the key feedback area
			DBPCBMUL	1	"X'01'" Positioning is multiple (for ACBGEN only)
32	(20)	4	DBPCBNSS		Number of sensitive segments in the PCB (after ACBGEN only)
32	(20)	2	DBPCBSSN		Number of sensitive segments (for ACBGEN only)
34	(22)	2	DBPCBSOF		Offset to the first segment (for ACBGEN only)
36	(24)	256	DBPCBKFD		Key feedback area

Cross Reference

Name	Hex Offset	Hex Value
DBPCB	0	
DBPCBAE	9	01
DBPCBDBD	0	
DBPCBGO	9	02
DBPCBJCB	10	
DBPCBKFD	24	
DBPCBLEV	8	
DBPCBLE1	8	
DBPCBLE2	9	
DBPCBLKY	1C	
DBPCBMKL	1C	
DBPCBMUL	1C	01
DBPCBNSS	20	
DBPCBPRO	C	
DBPCBSFD	14	
DBPCBSOF	22	
DBPCBSSN	20	
DBPCBSTC	A	
DBPCBTKW	10	80

PDCA - Problem Determination Control Area

DSECT Name: PDCA

The PRCA (Problem Determination Control Area) is used to hold miscellaneous data used in problem determination.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	PDCA		
0	(0)	4	PDCACPAC		Variable length segment compression routine list pointer
4	(4)	4	PDCAXPRM		Secondary index suppression routine list pointer
8	(8)	4	PDCAFERT		Field exit routine list pointer
12	(C)	1	PDCAFlag		PDCA flag byte
			PDCASTOP	1...	"X'80'" Stop saving messages
13	(D)	3	PDCAMSG		Abend code
16	(10)	4	PDCADecB		Online formatted dump ECB

MPS TERMINATION CLEANUP ROUTINE ADDRESSES

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
20	(14)	4	PDCBPCNT		Address of DLZBPC00 normal termination 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
			PDCAEND	..1.	"*" end of problem determination control area

Cross Reference

Name	Hex Offset	Hex Value
PDCA	0	
PDCACPAC	0	
PDCADecB	10	
PDCAEND	1C	20
PDCAFERT	8	
PDCAFlag	C	
PDCAMSG	D	
PDCASTOP	C	80
PDCAXPRM	4	
PDCBPCAT	18	
PDCBPCNT	14	
PDCSYSTT	1C	

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), the entry length (SCDDLIPL), and the number of entries (SCDDLIPN) are contained in the SCD (system contents directory).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZPDIR		PSB directory entry dummy section
			PDIR	....	"*" label used to establish address
0	(0)	8	PDIRSYM		PSB execution name converted from name supplied during PSBGEN
8	(8)	4	PDIRADDR		PSB address
12	(C)	4	PDIRPSBL		Storage required for PSB
16	(10)	2	PDIRZWA		Storage required for index workarea
18	(12)	1	PDIRCode		PSB code byte
			PDIRUPD	1...	"X'80'" This PSB is update sensitive
			PDIREXC	.1..	"X'40'" This PSB requires DMB exclusive control
			PDIRPLI	..1.	"X'20'" This PSB for PL/I
			PDIRDUPL	...1	"X'10'" This PSB is duplicate
			PDIRMPLI	.... 1...	"X'08'" PSB host language is PL/I
			PDIRDELT	.... ..1.	"X'02'" This PSB is delete sensitive
			PDIRTFAL	1	"X'01'" PSDB-SDB chaining error detected during on-line task termination
19	(13)	1	PDIROPTC		PSB scheduling codes
			PDIRNOSC	1...	"X'80'" Do not schedule this PSB
			PDIRSCHD	.1..	"X'40'" This PSB is scheduled
			PDIRREM	..1.	"X'20'" PSB is on remote system
			PDIRNTNT	...1	"X'10'" This PSB waiting for intent

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			PDIRBPLI	.... 1...	"X'08'" Reserved
			PDIRXPSB	.... .1..	"X'04'" Remote PST + local component
			PDIRBAD	1	"X'01'" PSB initialization failed
20	(14)	4	PDIRSILA		Address of PSB segment intent list
24	(18)	4	PDIREMOT		Remote PDIR (RPDIR) entry address
			PDIRLEN	...1 11..	"*-PDIR" PSB directory entry Length

Cross Reference

Name	Hex Offset	Hex Value
DLZPDIR	0	
PDIR	0	00
PDIRADDR	8	
PDIRBAD	13	01
PDIRBPLI	13	08
PDIRCode	12	
PDIRDELT	12	02
PDIRDUPL	12	10
PDIREMOT	18	
PDIREXC	12	40
PDIRLEN	18	1C
PDIRMPLI	12	08
PDIRNOSC	13	80
PDIRNTNT	13	10
PDIROPTC	13	
PDIRPLI	12	20
PDIRPSBL	C	
PDIRREM	13	20
PDIRSCHD	13	40
PDIRSILA	14	
PDIRSYM	0	
PDIRTFAL	12	01
PDIRUPD	12	80
PDIRXPSB	13	04
PDIRZWA	10	

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 a 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. Field SCDPPSTS points to the PST prefix and field SCDPPSTW indicates the number of PPSTs.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZPPST		PST prefix dummy section
			PPST	....	"*"
0	(0)	1	PPSTCF		Prefix chain forward pointer
1	(1)	1	PPSTCB		Prefix chain backward pointer
2	(2)	1	PPSTECB		Post/wait byte of PST ECB
3	(3)	1	PPSTCW		PST prefix program isolation wait chain
4	(4)	1	PPSTIND		Task schedule and dispatch indicators
			PPSTIO	1...	"X'80'" Waiting for I/O
			PPSTSI	.1..	"x'40'" Can not schedule due to segment intent conflict
			PPSTTC	..1.	"X'20'" Can not schedule task count limit exceeded
			PPSTBF	...1	"X'10'" Task enqueued by buffer handler
			PPSTMPS	.... 1...	"X'08'" Indicates MPS task
			PPSTACT	.... .1..	"X'04'" This is current task
			PPSTMSDL	.... ..1.	"X'02'" Scheduled by BPC
			PPSTA	1	"X'01'" Task scheduled
5	(5)	3	PPSTCA		Address of PST
8	(8)	1	PPSTID		DL/I task ID (PPST number)
9	(9)	3	PPSTTCA		Task's TCA address

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description

Section used by buffer handler for ENQ/DEQ. The following four fields must be contiguous and their order must not be changed.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
12	(C)	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 = Byte 0-1 = subpool number Byte 2-3 = PPST number of task next in chain
24	(18)	4	PPSTEXTQ		Enqueue/dequeue pointers for data set extension queue Byte 0-1 = not used Byte 2-3 = PPST number of task next in chain
28	(1C)	1	PPSTIND1		Flag byte
			PPSTEXQ	1...	"X'80'" This task was on the control interval extension queue
29	(1D)	3			Reserved
			PPSTEND	..1.	"*" end of prefix DSECT

Section used for online segment intent scheduling

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
12	(C)	4	PPSTPDIR		Task's PDIR entry address
16	(10)	1	PPSTTSKP		Task dispatching priority
			PPSTLEN	..1.	"*-PPST" Length of the PST prefix

Cross Reference

Name	Hex Offset	Hex Value
DLZPPST	0	
PPST	0	00
PPSTA	4	01
PPSTACT	4	04
PPSTBF	4	10
PPSTCA	5	
PPSTCB	1	
PPSTCF	0	
PPSTCW	3	
PPSTECB	2	
PPSTEND	1D	20
PPSTEXCI	C	
PPSTEXQ	1C	80
PPSTEXTQ	18	
PPSTID	8	
PPSTIND	4	
PPSTIND1	1C	
PPSTIO	4	80
PPSTLEN	10	20
PPSTMPS	4	08
PPSTMSDL	4	02
PPSTPDIR	C	
PPSTPECI	10	
PPSTSI	4	40
PPSTSUPO	14	
PPSTTC	4	20
PPSTTCA	9	
PPSTTSKP	10	

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	PSB		
0	(0)	1	PSBVMID		DOS DL/I version ID
			PSBV11	1	"X'01'" Version 1.1 or later
1	(1)	3	PSBIOASZ		Length of I/O workarea (PSBIOAWK)
0	(0)	4	PSBFRTA		Field exit routine address if no entries in table, low order 3-bytes = 0 (used only during initialization)
4	(4)	4	PSBXIOWK		Address of index I/O work area or user's version of a segment built by retrieve
8	(8)	4	PSBSEGWK		Address of variable length segment work area
12	(C)	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	..1.	"X'20'" PSB is for non-PL/I language
			PSBPLI	...1	"X'10'" PL/I is source language
			PSBFLS	1	"X'01'" PSB contains field sensitive segment
			PSBLOGDB	.... ..1.	"X'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)	256	PSBLIST		Beginning of the PCB list

Cross Reference

Name	Hex Offset	Hex Value
PSB	0	
PSBCode	1D	
PSBDBOFF	22	
PSBFLS	1D	01
PSBFRTA	0	
PSBINDEX	1C	
PSBIOASZ	1	
PSBIOAWK	18	
PSBLIST	24	
PSBLOGDB	1D	02
PSBNDXWK	14	
PSBNPLI	1D	20
PSBPLI	1D	10
PSBPST	C	
PSBSEGWK	8	
PSBSIZE	1E	
PSBTPOFF	20	
PSBVMID	0	
PSBV11	0	01
PSBXIOWK	4	
PSBXPCB	10	

PSBSQLIO - PSB SQL/DS I/O Area

DSECT Name: DLZPSBIO

This DSECT is used to create the DL/I SQL/DS tables: PSBBASICDATA, PSBSEGMENTDATA, and PSBFIELDDATA.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	PSBSQLIO		
0	(0)	8	CREADATE		Date 'yyyymmdd' of creation
0	(0)	2	CREACENT		Century
2	(2)	2	CREAYEAR		Year
4	(4)	2	CREAMON		Month
6	(6)	2	CREADAY		Day
8	(8)	8	CREAUSER		Userid of creator
16	(10)	8	CHANDATE		Date 'yyyymmdd' of change
24	(18)	8	CHANTIME		Time 'hhmmssff' of change
24	(18)	2	CHANHOUR		Hours
24	(18)	1	CHANHOU1		Labels for making hour
25	(19)	1	CHANHOU2		Printable
26	(1A)	2	CHANMIN		Minutes
26	(1A)	1	CHANMIN1		Labels for making minutes
27	(1B)	1	CHANMIN2		Printable
28	(1C)	2	CHANSEC		Seconds
28	(1C)	1	CHANSEC1		Labels for making seconds
29	(1D)	1	CHANSEC2		Printable
30	(1E)	2	CHANFRAC		Fraction of seconds
30	(1E)	1	CHANFRA1		Labels for making fraction
31	(1F)	1	CHANFRA2		Printable
32	(20)	8	CHANUSER		Userid of modifier

This section contains information unique to the PSBBASICDATA table.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
40	(28)	7	PSBName		Name of PSB
47	(2F)	5	LANGUAGE		Cobol or PL/I

This section contains information unique to the PSBPCBDATA table.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
48	(30)	4	PCBNUMBR		Number of PCB in hierarchy

52	(34)	2	DBTYPE		'DB' for upward compatibility with IMS
54	(36)	7	DBDName		Name of DBD
61	(3D)	4	PCBPROCO		Processing option
68	(44)	4	KEYLEN		Longest concatenated key
72	(48)	8	POSITING		Single or multiple positioning
80	(50)	7	PROSEQ		Name of the secondary index
88	(58)	2	PROSEIND		SQL indicator variable
			PSBPCBLN	..1. 1.11	"*-language" length of the PSBPCB table to clear

This section contains information unique to the PSBSEGMENTDATA table.

52	(34)	4	SEGNUMBR		Segment number
56	(38)	8	SEGMName		Name of sensitive segment
64	(40)	8	PARENTNM		Name of parent of segment
72	(48)	2	PARENIND		SQL indicator variable
74	(4A)	4	SEGPROCO		Processing option
			PSBSEGLN	...1 111.	"*-PCBNUMBR" length of the PSBSEGMENT table to clear

This section contains information unique to the PSBFIELDDATA table.

64	(40)	4	FLDNUMBR	Field number
68	(44)	8	FLDName	Name of sensitive field
76	(4C)	4	BYTES	Length of the field
80	(50)	2	BYTESIND	SQL indicator variable
82	(52)	8	POSName	Name of the previously defined field
90	(5A)	2	POSNAIND	SQL indicator variable
92	(5C)	4	POSITION	Starting position of field
96	(60)	2	POSITIND	SQL indicator variable
98	(62)	1	DATATYPE	Type of data in field
100	(64)	2	DATATIND	SQL indicator variable
102	(66)	8	RTNName	Name of field exit routine
110	(6E)	2	RTNIND	SQL indicator variable
112	(70)	1	FLDTYPE	Sensitive or virtual
113	(71)	3	REPLACE	Can field be modified?
116	(74)	2	REPLAIND	SQL indicator variable

118	(76)	2	CValue		Initial character value
			CVALLEN	.111 .11.	"CValue" length of character value
			CValueS	.111 1...	"CValue+2" the character value
374	(176)	2	CVALIND		SQL indicator variable
376	(178)	4	IValue		Initial integer value
380	(17C)	2	IVALIND		SQL indicator variable
382	(17E)	8	DValue		Initial decimal value
390	(186)	2	DVALIND		SQL indicator variable
392	(188)	8	FValue		Initial float value
400	(190)	2	FVALIND		SQL indicator variable
			PSBFLDLN		" *-PCBNUMBR" length of the PSBFIELD table to clear
			PSBIOLEN		"*-PSBSQLIO" length of PSBFieldDATA

Cross Reference

Name	Hex Offset	Hex Value
BYTES	4C	
BYTESIND	50	
CHANDATE	10	
CHANFRAC	1E	
CHANFRA1	1E	
CHANFRA2	1F	
CHANHOUR	18	
CHANHOU1	18	
CHANHOU2	19	
CHANMIN	1A	
CHANMIN1	1A	
CHANMIN2	1B	
CHANSEC	1C	
CHANSEC1	1C	
CHANSEC2	1D	
CHANTIME	18	
CHANUSER	20	
CREACENT	0	
CREADATE	0	
CREADAY	6	
CREAMON	4	
CREAUSER	8	
CREAYEAR	2	
CVALIND	176	
CVALLEN	76	76
CVALUE	76	
CVALUES	76	78
DATATIND	64	
DATATYPE	62	
DBDNAME	36	
DBTYPE	34	
DVALIND	186	
DVALUE	17E	
FLDNAME	44	
FLDNUMBR	40	
FLDTYPE	70	
FVALIND	190	
FVALUE	188	
IVALIND	17C	
IVALUE	178	
KEYLEN	44	
LANGUAGE	2F	
PARENIND	48	
PARENTNM	40	
PCBNUMBR	30	
PCBPROCO	3D	
POSITIND	60	
POSITING	48	
POSITION	5C	
POSNAIND	5A	
POSNAME	52	
PROSEIND	58	
PROSEQ	50	
PSBFLDLN	190	0162
PSBIOLEN	190	0192
PSBNAME	28	
PSBPCBLN	58	2B
PSBSEGLN	4A	1E
PSBSQLIO	0	
REPLACE	71	
REPLAIND	74	
RTNIND	6E	
RTNNANE	66	
SEGMNAME	38	
SEGNUMBR	34	
SEGPROCO	4A	

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBPSDB		
0	(0)	1	DMBSC		Segment code "X'01'" Root segment code
1	(1)	1	DMBPSC		Parent's segment code
2	(2)	1	DMBLEV		Segment level
3	(3)	1	DMBXNULL		(Not used in DL/I DOS/VS)
4	(4)	1	DMBPPFD		Pointer number in parent to first occurrence of segment for parent
5	(5)	1	DMBPPBK		Pointer number in parent to last occurrence of segment for parent
6	(6)	1	DMBDCB		ACB number
7	(7)	1	DMBPTR		Prefix flags
			DMBCTR	1...	"X'80'" Counter present
			DMBPTFD	.1..	"X'40'" Segment has physical twin forward pointer
			DMBPTBK	..1.	"X'20'" Segment has physical twin backward pointer
			DMBPP	...1	"X'10'" Segment has physical parent pointer
			DMBLTFD	.... 1...	"X'08'" Segment has logical twin forward pointer
			DMBLTBK	.... .1..	"X'04'" Segment has logical twin backward pointer
			DMBLP	.... ..1.	"X'02'" Segment has logical parent pointer
			DMBHIER	1	"X'01'" (Not used in DL/I DOS/VS)
8	(8)	2	DMBPRSZ		Prefix length of segment
10	(A)	2	DMBDL		Data length of segment
12	(C)	1	DMBISRT		Insert rules
			DMBXPROT	1...	"X'80'" System data in index is protected

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			DMBIHERE	..11	"X'30'" For no key field, insert at current position
			DMBILST	..1.	"X'20'" For no key field, insert after existing segment
			DMBIFST	...1	"X'10'" For no key field, insert before existing segment
			DMBNOTW	.... .1..	"X'04'" No twin allowed
			DMBIRL	11	"X'03'" Insert rule is logical
			DMBIRP	.... ..1.	"X'02'" Insert rule is physical
			DMBIRV	1	"X'01'" Insert rule is virtual
13	(D)	1	DMBDLT		Delete/replace rules
			DMBDRLC	..11	"X'30'" (Not used in DL/I DOS/VS)
			DMBDRPC	..1.	"X'20'" (Not used in DL/I DOS/VS)
			DMBDRVC	...1	"X'10'" (Not used in DL/I DOS/VS)
			DMBRRL	.... 11..	"X'0C'" Replace rule is logical
			DMBRRP	.... 1...	"X'08'" Replace rule is physical
			DMBRRV	.... .1..	"X'04'" Replace rule is virtual
			DMBDRL	11	"X'03'" Delete rule is logical
			DMBDRP	.... ..1.	"X'02'" Delete rule is physical
			DMBDRV	1	"X'01'" Delete rule is virtual
14	(E)	2	DMBCKL		Concatenated key length of parent of this segment
16	(10)	1	DMBUSE		
			DMBEX	1...	"X'80'" This PSDB in use exclusively
			DMBUP	.1..	"X'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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
20	(14)	4	DMBFSDB		Address of first SDB for this segment
24	(18)	1	DMBVLDFG		Variable length data flag
			DMBPI	1...	"X'80'" Program isolation should be done for this segment
			DMBCPT	.... 1...	"X'08'" Segment has compression routine
			DMBVLS	.... .1..	"X'04'" Segment is variable length
			DMBCPTKY	.... ..1.	"X'02'" Compression routine has key expand routine
			DMBCPTIT	1	"X'01'" Compression routine has initialization processing
25	(19)	3	DMBSCTAB		Address of segment compaction table
28	(1C)	2	DMBSGMN		For variable length segment minimum length of segment
30	(1E)	2	DMBSGMX		For variable length segment maximum length of segment
32	(20)	1	DMBFlag		Secondary list flags
			DMBPAIR	.1.. 1...	"X'48'" (Not used in DL/I DOS/VS)
			DMBLPEX	.1..	"X'40'" A logical parent exists (segment is a logical child)
			DMBLCEX	..1.	"X'20'" One or more logical children exist (segment is a logical parent parent)
			DMBNXEX	...1	"X'10'" One or more indexes exist
			DMBXDEX	.... .1..	"X'04'" An indexed segment exists
32	(20)	4	DMBLST		Address of secondary list for this segment
			DMBPSDBN	..1. .1..	"*" end of one PSDB entry
			DMBPLEN	..1. .1..	"DMBPSDBN-DMBSC" length of each PSDB entry

Cross Reference

Name	Hex Offset	Hex Value
DMBCKL	E	
DMBCPT	18	08
DMBCPTIT	18	01
DMBCPTKY	18	02
DMBCTR	7	80
DMBDCB	6	
DMBDL	A	
DMBDLT	D	
DMBDRL	D	03
DMBDRLC	D	30
DMBDRP	D	02
DMBDRPC	D	20
DMBDRV	D	01
DMBDRVC	D	10
DMBEX	10	80
DMBFDBA	10	
DMBFLAG	20	
DMBFSDB	14	
DMBHIER	7	01
DMBIFST	C	10
DMBIHERE	C	30
DMBILST	C	20
DMBIRL	C	03
DMBIRP	C	02
DMBIRV	C	01
DMBISRT	C	
DMBLCEX	20	20
DMBLEV	2	
DMBLP	7	02
DMBLPEX	20	40
DMBLST	20	
DMBLTBK	7	04
DMBLTFD	7	08
DMBNOTW	C	04
DMBNXEX	20	10
DMBPAIR	20	48
DMBPI	18	80
DMBPLEN	20	24
DMBPP	7	10
DMBPPBK	5	
DMBPPFD	4	
DMBPRSZ	8	
DMBPSC	1	
DMBPSDB	0	
DMBPSDBN	20	24
DMBPTBK	7	20
DMBPTFD	7	40
DMBPTR	7	
DMBRRL	D	0C
DMBRRP	D	08
DMBRRV	D	04
DMBSC	0	
DMBSCTAB	19	
DMBSGMN	1C	
DMBSGMX	1E	
DMBUP	10	40
DMBUSE	10	
DMBVLDFG	18	
DMBVLS	18	04
DMBXDEX	20	04
DMBXNULL	3	
DMBXPROT	C	80

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).

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZPSIL		PSB segment intent list entry
0	(0)	8	PSILDMBN		DMB name for this list entry as produced by the block builder utility
0	(0)	4	PSILDIRA		"PSILDMBN" DDIR address of corresponding DMB resolved during initialization
4	(4)	2	PSILDIRN		"PSILDIRA+4" DMB number of this DMB
8	(8)	1	PSILNTNT		Segment intent descriptor byte
			PSILDBEX	1...	"X'80'" PSB requires data base exclusive control
			PSILDBUP	.1..	"X'40'" PSB is update sensitive to this data base
			PSILBFRI	..1.	"X'20'" Buffer pool space request for this KSDS
			PSILGOPO	...1	"X'10'" PSB references are all 'GO'
			PSILNREF	.... 1...	"X'08'" Data base is not referenced
			PSILNOPI	.... .1..	"X'04'" No translate for PI
9	(9)	1	PSILLNGH		Length of segment intent list
10	(A)	Var	PSILSEGD		Start of segment intent bits

Each segment in the DMB pointed to by an intent list entry is described by two-bit fields beginning at PSILSEGD. There is a list entry for each DMB referenced in the associated PSB. The two bits represent the PSBs sensitivity to each PSDB and have the following meanings:

00 - PSB not sensitive to the segment
01 - PSB read only sensitive
10 - PSB is update sensitive
11 - PSB req exclusive ctl(HISAM root insert)

Field/Flag Name	Flag Code (Bit)	Description
PSILNS	1111 1111	"X'FF'" Mask used to test 4 segments for no sensitivity
PSILRO	1.1. 1.1.	"X'AA'" Mask used to test 4 segments for update

The bits are allocated to segments in the following manner:

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			PSILEND		End of PSB intent list indicator

Cross Reference

Name	Hex Offset	Hex Value
DLZPSIL	0	
PSILBFRI	8	20
PSILDBEX	8	80
PSILDBUP	8	40
PSILDIRA	0	00
PSILDIRN	0	04
PSILDMBN	0	
PSILEND		
PSILGOPO	8	10
PSILLNGH	9	
PSILNOPI	8	04
PSILNREF	8	08
PSILNS	A	FF
PSILNTNT	8	
PSILRO	A	AA
PSILSEGD	A	

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.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZPST		PST dummy section
			PST	....	"*" start of PST
0	(0)	4	PSTPREAD		Address of this PST's prefix
4	(4)	4	PSTDLIW0		Action-modules work area HD unload (DLZURGU0) return address for retrieve
8	(8)	4	PSTDLIW1		Action-modules work area
12	(C)	4	PSTDLIW2		Action-modules work area
16	(10)	4	PSTDLIW3		Action-modules work area
20	(14)	4	PSTDLIW4		Action-modules work area
24	(18)	4	PSTDLIW5		Action-modules work area
28	(1C)	4	PSTDLIW6		Action-modules work area
32	(20)	4	PSTDLIW7		Action-modules work area
36	(24)	4	PSTDLIW8		Action-modules work area
40	(28)	4	PSTDLIW9		Action-modules work area
44	(2C)	4	PSTDLIWA		Action-modules work area
48	(30)	4	PSTDLIWB		Action-modules work area
52	(34)	4	PSTDLIWC		Action-modules work area
56	(38)	4	PSTDLIWD		Action-modules work area
60	(3C)	4	PSTDLIWE		Action-modules work area
64	(40)	4	PSTDLIWF		Action-modules work area
USER CALL PROCESSING SECTION					
68	(44)	1	PSTCode1		
			PSTSCALL	1...	"X'80'" PST for system call
			PSTINTNT	.1..	"X'40'" Cannot schedule intent not satisfied
			PSTSCHED	...1	"X'10'" OK to complete scheduling

Note:
If PSTINTNT and PSTSCHED are both set, DL/I Backout is in control.

			PSTPRVWT	.... 1...	"X'08'" Logger private wait indicator
			PSTOLTW	.... .1..	"X'04'" Another task waiting for resource owned by this task
			PSTLOGIP	.... ..1.	"X'02'" Log I/O in progress
68	(44)	4	PSTSCDAD		Address of addressable portion of SCD
72	(48)	1	PSTABIND		Task/system ABEND indicator
			PSTERMSP	1...	"X'80'" PUT error message indicator
			PSTSABND	..1.	"X'20'" System ABEND indicator bit
			PSTTABND	...1	"X'10'" Task ABEND indicator bit
72	(48)	4	PSTIQPRM		Address of callers parameter list
76	(4C)	1	PSTMI		Return segment indicator
76	(4C)	4	PSTUSER		Address of user's I/O area
80	(50)	4	PSTSEGL		Retrieved segment length
84	(54)	4	PSTSEG		Retrieved segment address
88	(58)	4	PSTPSB		PDIR entry address

USER TASK STATISTICS

92	(5C)	4	PSTACCT		
92	(5C)	4	PSTDGU		Number of GU calls issued
96	(60)	4	PSTDGN		Number of GN calls issued
100	(64)	4	PSTDGNP		Number of GNP calls issued
104	(68)	4	PSTDGHU		Number of GHU calls issued
108	(6C)	4	PSTDGHN		Number of GHN calls issued
112	(70)	4	PSTDGHNP		Number of GHNP calls issued
116	(74)	4	PSTDISRT		Number of ISRT calls issued
120	(78)	4	PSTDDLET		Number of DLET calls issued
124	(7C)	4	PSTDREPL		Number of REPL calls issued
128	(80)	4	PSTDCHKP		Number of CHKP calls issued

			PSTSTLN	..1. 1...	"*-PSTACCT" length of user call statistics

Action module section

132	(84)	4	PSTDBPCB		Address of current PCB
136	(88)	1	PSTFNCTN		Function codes

Equates for buffer handler function codes

PSTBKLCT	1	"X'01'" Locate relative block number
PSTBYLCT	.... ..1.	"X'02'" If HD, locate relative byte number. If HISAM or HIDAM index, read a record by RBA from a KSDS. If HISAM, read a record by RBA from an ESDS
PSTGBSPC	11	"X'03'" Get buffer space
PSTFBSPC	.... .1..	"X'04'" Free buffer space
PSTBFMPT	.... .1..	"X'04'" Mark buffers empty
PSTBFALT	.... .1.1	"X'05'" If HD; mark a buffer containing data altered if HISAM or HIDAM index; write a record by RBA to a KSDS if HISAM write a record by RBA to an ESDS
PSTBYALT	.... .11.	"X'06'" Locate a relative byte number and mark buffer altered
PSTPGUSR	.... .111	"X'07'" Purge all buffers altered by a task
PSTWRITE	.... 1...	"X'08'" Write new record to HISAM ESDS
PSTSTLEQ	.... 1..1	"X'09'" Read a record by key from a KSDS
PSTERASE	.... 1.1.	"X'0A'" Erase a record in a KSDS
PSTGETNX	.... 1.11	"X'0B'" Read the next record in a KSDS
PSTSTLBG	.... 11..	"X'0C'" Read the record containing the first root in a KSDS
PSTPUTKY	.... 11.1	"X'0D'" Insert a record by key into a KSDS
PSTMSPUT	.... 111.	"X'0E'" insert record(s) sequentially into a KSDS

Equates for OPEN/CLOSE function codes

PSTOCDMB	1	"X'01'" Close DMB address of DMB in R2
PSTOCPCB	.... ..1.	"X'02'" Close PCB address of PCB in R2
PSTOCALL	.... .1..	"X'04'" Close all DMB's
PSTOCCLS	....	"X'00'" Close call bit 4=0
PSTOCOPN	.... 1...	"X'08'" Open call bit 4=1
PSTOCDCB	...1	"X'10'" Open/close the dmb in pstdcbnm dsg address in PSTDSGA
PSTOCLD	..1.	"X'20'" Open for load
PSTOCDSG	.1..	"X'40'" Open the DSG in PSTDSGA
PSTOCBAD	1...	"X'80'" Open unsuccessful

Equates for Space Management function codes

	1...	"X'80' Backout in control
PSTGTSPC	1	"X'01'" et space for segment (R5 contains pointer to PSDB)
PSTFRSPC	.... ..1.	"X'02'" Free space for segment (R5 contains pointer to PSDB)
PSTBTMPF	11	"X'03'" Do bit map update
PSTGTRAP	.... .1..	"X'04'" Get space close to rap in PSTBYTNM

Equates for Index Maintenance function codes

PSTXMDLT	1.1.	"X'A0'" Perform index maintenance for segment to be deleted
PSTXMRPL	1.1. ...1	"X'A1'" Perform index maintenance for segment to be replaced
PSTXMISR	1.1. ..1.	"X'A2'" Perform index maintenance for segment to be inserted
PSTXMUNL	1.1. ..11	"X'A3'" Perform index maintenance for segment to be unloaded

Equates for Program Isolation function codes

			PSTQENQ	....	"X'00'" Enqueue (queueing facility)
			PSTQVER	.... .1..	"X'04'" Verify (queueing facility)
			PSTQDEQ	.... 1...	"X'08'" Dequeue (queueing facility)
			PSTQPUR	.... 11..	"X'0C'" Purge (queueing facility)
137	(89)	1	PSTRTCDE		Return codes

Equates for buffer handler return codes

PSTCLOK	....	"X'00'" No error occurred
PSTGTDS	.... .1..	"X'04'" RBN is beyond the end of the data set
PSTIOERR	.... 1...	"X'08'" I/O error
PSTRDERR	.... 1...	"X'08'" Permanent read error
PSTNOSPC	.... 11..	"X'0C'" No space for adds
PSTBDCAL	...1	"X'10'" Illegal call
PSTNOTFD	...1 .1..	"X'14'" No record found (retrieve by key)
PSTNWBLK	...1 1...	"X'18'" New block was created in the buffer pool
PSTNPLSP	...1 11..	"X'1C'" Insufficient space in the buffer pool
PSTWROSI	..1.	"X'20'" Size of requested buffer exceeds the size of buffers in any subpool
PSTENDDA	..1. .1..	"X'24'" End of data set no record returned
PSTBYEND	..1. 1...	"X'28'" Key or RBA higher than the highest key or rba in the data set
PSTEOD	..1. 11..	"X'2C'" End of data set reached on a request issued by open
PSTINLD	..11 .1..	"X'34'" Invalid request during data set loading

Space Management return codes

			PSTFRBLK	..11	"X'30'" Block not used due to distributed free space parm
			PSTNOBFA	..11 ...1	"X'31'" Unsuccessful mark buffer alter request
			PSTBTMPF	11	"X'03'" Bit map update required

Equates for Program Isolation return codes

			PSTQRWR	1	"X'01'" Wait was required
			PSTQROOP	.... ..1.	"X'02'" Other owners present
			PSTQRDDL	.... .1..	"X'04'" Terminated due to deadlock
			PSTQRBDC	.... 1...	"X'08'" Terminated due to bad call
			PSTQRNSE	...1	"X'10'" Terminated insufficient storage
			PSTPISIU	.1..	"X'40'" Secondary index updated
			PSTPIPIU	1...	"X'80'" Primary index updated
138	(8A)	2	PSTOFFST		Offset to PSTDATA from start of buffer
140	(8C)	4	PSTDSGA		Address of DSG portion of the JCB

PSTBLKNM, PSTDMBNM, and PSTACBNM must be in this order and contiguous for the buffer handler

144	(90)	4	PSTBLKNM		Relative block number
			PSTWABUF	1...	"X'80'" Indicator that buffer is being used as a workarea and is not associated with an RBA
148	(94)	2	PSTDMBNM		DMB number
150	(96)	1	PSTACBNM		ACB number
151	(97)	1			Reserved
152	(98)	4	PSTBYTNM		RBA or relative record number hi-order byte contains x'80' if request is for HISAM ESDS
156	(9C)	4	PSTDATA		Address of requested data
160	(A0)	4	PSTBUFFA		Address of buffer prefix

Buffer Handler and Space Management section

164	(A4)	4	PSTBFUSE		Address of the buffer prefix to be used
168	(A8)	4	PSTSUIN		Address of the subpool information table to be used
172	(AC)	4	PSTPREAR		Beginning address of the buffer prefix area for the subpool information table used
176	(B0)	2	PSTSUBNM		Subpool number used during this call
178	(B2)	2	PSTSWI		Work space
			PSTHISES	1...	"X'80'" HISAM ESDS is being processed
			PSTHDWIP	.... 1...	"X'08'" HD write in progress
			PSTCIOAF	.... .1..	"X'04'" Control interval in overflow area full
			PSTSMRQ	.... ..1.	"X'02'" Request made to the buffer handler by space management
			PSTPBRQC	1111 1111	"X'FF'" Purge buffer request completed
180	(B4)	1	PSTPOSEL		Count for position of use chain element
			PSTNOERR	.1..	"X'40'" No error message
181	(B5)	1			Reserved
182	(B6)	2			Reserved
184	(B8)	4	PSTSAVRE		Work area used by buffer

					handler when processing a request

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.

			PSTCTGPL	1.11 1...	"PSTSAVRE" area used as the VSAM catalog parameter list (CTGPL) by DLZGGSP0 and DLZDLOC0 to do locate
			PSTCPLN	..1. 1...	"40" length of CTGPL block
			PSTCTGNM	1	"1" number of CTGFL entries

This area is used by DLZDCI00 for SHOWCAT and GETVCE for FBA support.

			PSTSWKAR	1.11 1...	"PSTSAVRE" SHOWCAT work area used by space management DLZDCI00
			PSTSDATA	11.. 111.	"PSTSWKAR+22" location of needed data returned by SHOWCAT
			PSTRBAL	.... .1..	"4" RBA data length
			PSTVSL	.... .11.	"6" volume serial number length
			PSTSWKL	.1..	"64" length of SHOWCAT work area
			PSTVLSR	1111 1.1.	"PSTSWKAR+66" volume serial number save area
			PSTSPL	11.1 .1..	"PSTSWKAR+28" SHOWCAT parameter list
			PSTGVPL	111. 11..	"PSTSWKAR+52" GETVCE parameter list
			PSTGVWKL	..11 .1..	"52" length of GETVCE workarea
224	(E0)	4	PSTRETRE(8)		Buffer handler subroutine linkage register (R14) save area when processing a request

The following fields are used by OPEN/CLOSE (DLZDLOC0) and Space Management for VSAM field parameter list when processing an out-of-space condition for HIDAM data base.

			PSTCTGFL	111.	"PSTRETRE" area used as the VSAM field parameter list (ctgfl) by DLZGGSP0 and DLZDLOC0 to do locate
			PSTCTGWK	1111 1...	"PSTRETRE+24" VSAM catalog workarea
			PSTCTGL1	1111 1...	"PSTRETRE+24" catalog workarea length 1
			PSTCTGL2	1111 1.11	"PSTRETRE+27" catalog workarea length 2
			PSTCTGRT	1111 11..	"PSTRETRE+28" VSAM catalog return area for high-RBA
			PSTCFLN	...1 1...	"24" length of CTGFL block
			PSTCWKLN	.... 1...	"8" length of catalog workarea
			PSTCDATL	.... .1..	"4" data length for high-RBA

Buffer Handler statistics

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	1...	"X'80'" IWAIT issued during this call
			PSTEXPAD	.1..	"X'40'" Expad done for this I/O operation
259	(103)	1			Reserved
260	(104)	4	PSTTSKID		Hashed task ID high-order byte binary date low-order 3 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 cancelled while waiting for I/O completion.

264	(108)	4	PSTXPSV1	User VSAM savearea address
268	(10C)	4	PSTXPSV2	EXCPAD return address
272	(110)	4	PSTXPSV3	EXCPAD parm list address
276	(114)	4	PSTUIB	Address of UIB

PST WORK AREAS

280	(118)	4	PSTWRK1	Work words for
284	(11C)	4	PSTWRK2	buffer handler (DLZDBH00)

288	(120)	4	PSTWRK3	and
292	(124)	4	PSTWRK4	data base logger
296	(128)	4	PSTWRKT1	work space preserved
300	(12C)	4	PSTWRKT2	across
304	(130)	4	PSTWRKT3	calls
308	(134)	4	PSTWRKT4	to the
312	(138)	4	PSTWRKT5	buffer handler

The hi-order byte of PSTWRKT4 is used to pass the following function codes to index maintenance:

.... .1..	"X'04'" Re-insert index
.... ..11	"X'03'" Secondary indices only
.... ..2.	"X'02'" Primary indices only
....1	"X'01'" Both primary and secondary indices

316	(13C)	4	PSTWRKD1	Work space for use
320	(140)	4	PSTWRKD2	by
324	(144)	4	PSTWRKD3	delete/replace
328	(148)	4	PSTWRKD4	field storage manager
332	(14C)	4	PSTWRKD5	retrieve
336	(150)	4	PSTWRKD6	
340	(154)	4	PSTWRKD7	Load/insert and by buffer purge
344	(158)	4	PSTCURWA	Current delete work area
348	(15C)	4	PSTDLTWA	First delete work area address
352	(160)	4	PSTDLROM(21	Retrieve's save and maintenance work area (online only) CMF hook work area at task termination

Data base log section

436	(1B4)	4	PSTLOGWA	Address of work area for log O/P
440	(1B8)	4	PSTLOGQ	Address of reuse Q QCB in pool

Data base log use of PSTWRK1

PSTWRK1	Physical SDB address if new block - low-order 2 bytes is call count high-order byte used for function code
PSTDBLFC	"PSTWRK1"

Data base log function codes

			DBLNDXC	1...	"X'80'" Index maintenance call
			DBLCMC	....	"X'00'" Bits 1-3=0 chain maintenance call
			DBLNTCR	.111	"X'70'" Counter maintenance
			DBLLGDLT	.11.	"X'60'" Delete byte maintenance
			DBLPHYI	.1..	"X'40'" Insert
			DBLPHYD	..1.	"X'20'" Physical delete
			DBLPHYR	...1	"X'10'" Replace
			DBLOOPS	.... 1.1.	"X'0A'" No data end of user call
			DBLLASTC	.... 1...	"X'08'" Last change for this user call
			DBLFSE1	....	"X'00'" Bit 5=0 one FSE (if bits 1 or 2 on)
			DBLFSE2	.... .1..	"X'04'" Two FS's (if bits 1 or 2 on)
			DBLPHYRO	.... ..1.	"X'02'" Old copy of a replace
			DBLNEWBL	1	"X'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 FSE'S (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 DBDNAME
460	(1CC)	8	PSTPCPSB		PSB name
468	(1D4)	1	PSTPCT1		Partition/task option
			PSTBATCH	1...	"X'80'" PST is in batch partition
			PSTLODU	.1..	"X'40'" Load utility
			PSTLODUH	..1.	"X'20'" Load HDAM DB
			PSTHISMR	...1	"X'10'" HISAM data base recovery in progress
			PSTUST	.... 1...	"X'08'" Statistics utility
			PSTUDR	.... .1..	"X'04'" Data base recovery utility

			PSTULU	.... ..1.	"X'02'" Data base load/unload utility
			PSTUSM	1	"X'01'" Security maintenance utility
469	(1D5)	1	PSTPCT2		PGM options/info overlaid on every call to batch program request handler
			PSTXCONM	1...	"X'80'" Exclude console message
			PSTXPRTM	.1..	"X'40'" Exclude printer message
			PSTCHKP	.... .1..	"X'04'" User CKPT call successful
			PSTHLPI	.... ..1.	"X'02'" Application program using HLPI
			PSTPLI	1	"X'01'" User pgm is PL/I
470	(1D6)	2	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		Variable error message data
479	(1DF)	8	PSTCNVB		Work area for HD randomizing module
479	(1DF)	8	PSTERDT2		Variable error message data
487	(1E7)	1	PSTERIND		Error routine indicator
			PSTDUMPI	1...	"X'80'" Issue dump after error message put
			PSTCANLI	.1..	"X'40'" Issue cancel after error message put
488	(1E8)	72	PSTLIPRM		Area to build user parameter list and also register save area for MPS start and stop calls

The following labels can be used to address fields in the CALL parameter list

			PSTCCALL		"PSTLIPRM" current call
			PSTCPCB		"PSTLIPRM+4" current PCB address
			PSTCIO		"PSTLIPRM+8" current I/O area address
			PSTCSSA		"PSTLIPRM+12" current SSA address start if call is qualified
560	(230)	8	PSTPLIPR		PL/I region STXIT processor
568	(238)	4	PSTNORO		Number owned resources

572	(23C)	1	PSTQLEV		Queue request level
			PSTQLRO	....	"X'00'" Read only level
			PSTQLUPD	.... .1..	"X'04'" Update level
			PSTQLEXC	.... 1...	"X'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 address (RPST)
584	(248)	12	PSTSAVTR		Trace save area used only if output = CICS/VS

Field level descriptor block control fields

596	(254)	4	PSTFLD		Start of field level descriptor block entries
600	(258)	4	PSTFLDN		Next available fld entry
604	(25C)	4	PSTFLDE		FLD area end address+1
			PSTFLDAL	1...	"128" initial FLD area length
608	(260)	4	PSTFLDC		Current FLD 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	.1..	"X'40'" Open for partial reorganization
621	(26D)	3			Reserved
624	(270)	4	PSTCRRBA		RBA of current root insert
628	(274)	4			Reserved

Register save area

Registers are saved Register 14 - Register 12

640	(280)	72	PSTSV1	Seven register save
712	(2C8)	72	PSTSV2	areas
784	(310)	72	PSTSV3	required
856	(358)	72	PSTSV4	for
928	(3A0)	72	PSTSV5	processing
1000	(3E8)	72	PSTSV6	DL/I user
1072	(430)	72	PSTSV7	calls
			PSTLNGTH	"*-PST" length of PST (See the DL/I DOS/VS Diagnostic Guide for information on the save areas)

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
DBLCMC	1B8	00	PSTCTGPL	B8	B8
DBLFSE1	1B8	00	PSTCTGRT	E0	FC
DBLFSE2	1B8	04	PSTCTGWK	E0	F8
DBLLASTC	1B8	08	PSTCURWA	158	
DBLLGDLT	1B8	60	PSTCWKLN	E0	08
DBLNDXC	1B8	80	PSTDATA	9C	
DBLNEWBL	1B8	01	PSTDBLFC	1B8	0118
DBLNTCR	1B8	70	PSTDBPCB	84	
DBLOOPS	1B8	0A	PSTDCHKP	80	
DBLPHYD	1B8	20	PSTDDLET	78	
DBLPHYI	1B8	40	PSTDELTA	268	
DBLPHYR	1B8	10	PSTDGHN	6C	
DBLPHYRO	1B8	02	PSTDGHNP	70	
DLZPST	0		PSTDGHU	68	
PST	0	00	PSTDGN	60	
PSTABIND	48		PSTDGNP	64	
PSTACBNM	96		PSTDGU	5C	
PSTACCT	5C		PSTDISRT	74	
PSTBATCH	1D4	80	PSTDLIWA	2C	
PSTBDCAL	89	10	PSTDLIWB	30	
PSTBFALT	88	05	PSTDLIWC	34	
PSTBFMPT	88	04	PSTDLIWD	38	
PSTBFUSE	A4		PSTDLIWE	3C	
PSTBKLCT	88	01	PSTDLIWF	40	
PSTBLKNM	90		PSTDLIW0	4	
PSTBTMPF	88	03	PSTDLIW1	8	
PSTBUFFA	A0		PSTDLIW2	C	
PSTBYALT	88	06	PSTDLIW3	10	
PSTBYEND	89	28	PSTDLIW4	14	
PSTBYLCT	88	02	PSTDLIW5	18	
PSTBYTNM	98		PSTDLIW6	1C	
PSTCANLI	1E7	40	PSTDLIW7	20	
PSTCCALL	1E8	01E8	PSTDLIW8	24	
PSTCDATL	E0	04	PSTDLIW9	28	
PSTCFLN	E0	18	PSTDLROM	160	
PSTCHKP	1D5	04	PSTDLTWA	15C	
PSTCIO	1E8	01F0	PSTDMBNM	94	
PSTCIOAF	B2	04	PSTDREPL	7C	
PSTCLOK	89	00	PSTDSGA	8C	
PSTCLRWT	102		PSTDUMPI	1E7	80
PSTCNVB	1DF		PSTENDDA	89	24
PSTCode1	44		PSTEOD	89	2C
PSTCPCB	1E8	01EC	PSTERASE	88	0A
PSTCPLN	B8	28	PSTERCD1	1D6	
PSTCRRBA	270		PSTERCD2	1D7	
PSTCSSA	1E8	01F4	PSTERCOD	1D6	
PSTCTGFL	E0	E0	PSTERDT1	1D8	
PSTCTGL1	E0	F8	PSTERDT2	1DF	
PSTCTGL2	E0	FB	PSTERIND	1E7	
PSTCTGNM	B8	01	PSTERMSP	48	80

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
PSTEXPAD	102	40	PSTOCLD	88	20
PSTFBSPC	88	04	PSTOCOPN	88	08
PSTFLD	254		PSTOCPCB	88	02
PSTFLDAL	25C	80	PSTOFFST	8A	
PSTFLDC	260		PSTOLTW	44	04
PSTFLDE	25C		PSTOPEN	26C	
PSTFLDN	258		PSTOPEN1	26C	40
PSTFNCTN	88		PSTPBRQC	B2	FF
PSTFRBLK	89	30	PSTPCPGM	1C4	
PSTFRSPC	88	02	PSTPCPSB	1CC	
PSTGBSPC	88	03	PSTPCT1	1D4	
PSTGETNX	88	0B	PSTPCT2	1D5	
PSTGTDS	89	04	PSTPGUSR	88	07
PSTGTRAP	88	04	PSTPIPIU	89	80
PSTGTSPC	88	01	PSTPISIU	89	40
PSTGVPL	B8	EC	PSTPLI	1D5	01
PSTGVWKL	B8	34	PSTPLIPR	230	
PSTHDWIP	B2	08	PSTPOSEL	B4	
PSTHISES	B2	80	PSTPREAD	0	
PSTHISMR	1D4	10	PSTPREAR	AC	
PSTHLPI	1D5	02	PSTPRTGT	264	
PSTINLD	89	34	PSTPRVWT	44	08
PSTINTNT	44	40	PSTPSB	58	
PSTIOERR	89	08	PSTPUTKY	88	0D
PSTIQPRM	48		PSTQDEQ	88	08
PSTIWAIT	102	80	PSTQENQ	88	00
PSTLIPRM	1E8		PSTQLEV	23C	
PSTLNGTH	430	0478	PSTQLEXC	23C	08
PSTLODU	1D4	40	PSTQLRO	23C	00
PSTLODUH	1D4	20	PSTQLUPD	23C	04
PSTLOGIP	44	02	PSTQPUR	88	0C
PSTLOGQ	1B8		PSTQRBDC	89	08
PSTLOGWA	1B4		PSTQRDDL	89	04
PSTMI	4C		PSTQRNSE	89	10
PSTMSPUT	88	0E	PSTQROOP	89	02
PSTNOBFA	89	31	PSTQRWR	89	01
PSTNOERR	B4	40	PSTQVER	88	04
PSTNORO	238		PSTRAEND	1BC	
PSTNOSPC	89	0C	PSTRBAL	B8	04
PSTNOTFD	89	14	PSTRDERR	89	08
PSTNPLSP	89	1C	PSTRETRE	E0	
PSTNUMRO	100		PSTRPSTA	244	
PSTNUMWT	101		PSTRRDF	23C	
PSTNWBLK	89	18	PSTRRDL	240	
PSTOCALL	88	04	PSTRTCDE	89	
PSTOCBAD	88	80	PSTSABND	48	20
PSTOCCLS	88	00	PSTSAVRE	B8	
PSTOCDCB	88	10	PSTSAVTR	248	
PSTOCDMB	88	01	PSTSCALL	44	80
PSTOCDSG	88	40	PSTSCDAD	44	

Cross Reference

Name	Hex Offset	Hex Value
PSTSCHED	44	10
PSTSDATA	B8	CE
PSTSEG	54	
PSTSEGL	50	
PSTSMRQ	B2	02
PSTSPL	B8	D4
PSTSTLBG	88	0C
PSTSTLEQ	88	09
PSTSTLN	80	28
PSTSUBNM	B0	
PSTSUIN	A8	
PSTSV1	280	
PSTSV2	2C8	
PSTSV3	310	
PSTSV4	358	
PSTSV5	3A0	
PSTSV6	3E8	
PSTSV7	430	
PSTSWI	B2	
PSTSWKAR	B8	B8
PSTSWKL	B8	40
PSTTABND	48	10
PSTTSKID	104	
PSTUDR	1D4	04
PSTUIB	114	
PSTULU	1D4	02
PSTUSER	4C	
PSTUSM	1D4	01
PSTUST	1D4	08
PSTVLSR	B8	FA
PSTVSL	B8	06
PSTWABUF	90	80
PSTWRITE	88	08
PSTWRKD1	13C	
PSTWRKD2	140	
PSTWRKD3	144	
PSTWRKD4	148	
PSTWRKD5	14C	
PSTWRKD6	150	
PSTWRKD7	154	
PSTWRKT1	128	
PSTWRKT2	12C	
PSTWRKT3	130	
PSTWRKT4	134	
PSTWRKT5	138	
PSTWRK1	118	
PSTWRK2	11C	
PSTWRK3	120	
PSTWRK4	124	
PSTWROSI	89	20
PSTXCONM	1D5	80
PSTXMDLT	88	A0
PSTXMISR	88	A2
PSTXMRPL	88	A1
PSTXMUNL	88	A3
PSTXPRTM	1D5	40
PSTXPSV1	108	
PSTXPSV2	10C	
PSTXPSV3	110	

QWA - Queueing 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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZQWA		
0	(0)	36	QWA		
0	(0)	0			Module identifier

Page pointers for free block management

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	4	QWAFPP		First page pointer
4	(4)	4	QWACPP		Current page pointer flag bytes
8	(8)	1	QWAFLG1		First flag byte
			QWADDDF	1	"X'01'" Do deadlock detection
			QWANPOF	.... ..1.	"X'02'" New prime owner exists
9	(9)	1	QWAFLG2		Second flag byte
10	(A)	1	QWAFLG3		Third flag byte
11	(B)	1	QWAFLG4		Fourth flag byte work fields
12	(C)	4	QWAWFLD		Work field RDB queue area
16	(10)	4	QWANOQH		Number of queue heads
20	(14)	4	QWAHMLT		Hashing multiplier
24	(18)	Var	QWARDBQH		RDB chain queue headers of four bytes each

Cross Reference

Name	Hex Offset	Hex Value
DLZQWA	0	
QWA	0	
QWACPP	4	
QWADDDF	8	01
QWAFLG1	8	
QWAFLG2	9	
QWAFLG3	A	
QWAFLG4	B	
QWAFPP	0	
QWAHMLT	14	
QWANOQH	10	
QWANPOF	8	02
QWARDBQH	18	
QWAWFLD	C	

RDB - Resource Descriptor Block

DSECT Name: DLZRDB

The RDB (Resource Descriptor Block) is used to describe a resource 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. For more information, refer to "DLZQUEF0 -Queueing Facility" in Section 3 of this book.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRDB		
0	(0)	24	RDB		
0	(0)	1	RDBPOID		Primary owner PST prefix number
0	(0)	4	RDBRDBF		RDB forward chain pointer
4	(4)	1	RDBUOID		Update owner PSTpst prefix number
4	(4)	4	RDBRDBB		RDB backward chain pointer
8	(8)	1	RDBMAXL		Top enqueue level of current owners
8	(8)	4	RDBRRDF		Pointer to first RRD
12	(C)	1	RDBNOWN		Current number of owners
12	(C)	4	RDBRRDL		Pointer to last RRD
16	(10)	7	RDBRID		Resource ID
23	(17)	1			Reserved
			RDBLEN	...1 1...	"*-RDB" Length of RDB

Cross Reference

Name	Hex Offset	Hex Value
DLZRDB	0	
RDB	0	
RDBLEN	17	18
RDBMAXL	8	
RDBNOWN	C	
RDBPOID	0	
RDBRDBB	4	
RDBRDBF	0	
RDBRID	10	
RDBRRDF	8	
RDBRRDL	C	
RDBUOID	4	

RGT - Range Table

DSECT Name: DLZPRRGT

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 control cards. This control block is used by the partial reorganization utility.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	RGT		
0	(0)	4	RGTSTART		
0	(0)	4	RGTFBKLO		First block in range to be reorganized
4	(4)	4	RGTFBKHI		Last block in range to be reorganized
8	(8)	4	RGTFTLO1		First block in DSG 1 for reload to use
12	(C)	4	RGTFTHI1		Last block in DSG 1 for reload to use
16	(10)	4	RGTFTLO2		Same for DSG 2
20	(14)	4	RGTFTHI2		Same for DSG 2
24	(18)	4	RGTFTLO3		
28	(1C)	4	RGTFTHI3		
32	(20)	4	RGTFTLO4		
36	(24)	4	RGTFTHI4		
40	(28)	4	RGTFTLO5		
44	(2C)	4	RGTFTHI5		
48	(30)	4	RGTFTLO6		
52	(34)	4	RGTFTHI6		
56	(38)	4	RGTFTLO7		
60	(3C)	4	RGTFTHI7		
64	(40)	4	RGTFTLO8		
68	(44)	4	RGTFTHI8		
72	(48)	4	RGTFTLO9		
76	(4C)	4	RGTFTHI9		
80	(50)	4	RGTFTL10		First block in DSG 10 for reload to use
84	(54)	4	RGTFTH10		Last block in DSG 10 for reload to use
88	(58)	1	RGTKIND1		Key range format indicator 1 (C or X)

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
89	(59)	1	RGTKIND2		Key range format indicator 2 (C or X)
90	(5A)	2			Reserved
			RGTFLEN	.1.1 11..	"*-RGTSTART" length of a RGT entry
			RGTKEYAR	.1.1 11..	"*"

Cross Reference

Name	Hex Offset	Hex Value
RGT	0	
RGTFBKHI	4	
RGTFBKLO	0	
RGTFLEN	5A	5C
RGTFTHI1	C	
RGTFTHI2	14	
RGTFTHI3	1C	
RGTFTHI4	24	
RGTFTHI5	2C	
RGTFTHI6	34	
RGTFTHI7	3C	
RGTFTHI8	44	
RGTFTHI9	4C	
RGTFTH10	54	
RGTFTLO1	8	
RGTFTLO2	10	
RGTFTLO3	18	
RGTFTLO4	20	
RGTFTLO5	28	
RGTFTLO6	30	
RGTFTLO7	38	
RGTFTLO8	40	
RGTFTLO9	48	
RGTFTL10	50	
RGTKEYAR	5A	5C
RGTKIND1	58	
RGTKIND2	59	
RGTSTART	0	

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 communications (ISC) support. It defines fields passed between CICS/VS and DL/I.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRIB		
0	(0)	4	RIB		ISC remote interface block this control block follows immediately after the RPST
0	(0)	4	RIBPCBAL		Local PCB address list
4	(4)	4	RIBCHAIN		Remote PSB storage chain
8	(8)	4	RIBIOAWK		Local PSB I/O work area
12	(C)	4	RIBUPPER		Highest address of caller partition
16	(10)	2	RIBINDEX		PCB index number
18	(12)	1	RIBISC		ISC scheduling duration flags
			RIBPCBM	1...	"X'80'" PCBM scheduling call issued
			RIBBUFAL	.1..	"X'40'" RIBIOAWK buffer allocated
			RIBCHKP	..1.	"X'20'" Dl/I checkpoint call in progress
19	(13)	1	RIBISCO		ISC outbound flags
			RIBSYNC	1...	"X'80'" Sync point issued
			RIBHLPI	.1..	"X'40'" DL/I HLPI command with SSA and I/O lengths provided
20	(14)	1	RIBISCI		ISC inbound flags
			RIBFUNC	1...	"X'80'" Function string invalid
			RIBCALL	.1..	"X'40'" User call parameter list invalid
			RIBLNKNA	..1.	"X'20'" Link does not exist
			RIBLNKSH	...1	"X'10'" Link is out of service
			RIBNOSTT	.... 1...	"X'08'" CICS/VS not counting DL/I calls
21	(15)	1	RIBFCTR		ISC response code
22	(16)	1	RIBDLTR		Additional response information

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
22	(16)	1	RIBDLTR		Additional response information
			RIBRSET	.... .1..	"*-RIBISCO" length of function dependent flags
23	(17)	1			Reserved
24	(18)	4	RIBIOLEN		I/O area length for HLPI data base command
28	(1C)	4			Length always fullword multiple
			RIBLEN	...1 11..	"*-RIB" length of RIB

Cross Reference

Name	Hex Offset	Hex Value
DLZRIB	0	
RIB	0	
RIBBUFAL	12	40
RIBCALL	14	40
RIBCHAIN	4	
RIBCHKP	12	20
RIBDLTR	16	
RIBFCTR	15	
RIBFUNC	14	80
RIBHLPI	13	40
RIBINDEX	10	
RIBIOAWK	8	
RIBIOLEN	18	
RIBISC	12	
RIBISCI	14	
RIBISCO	13	
RIBLEN	1C	1C
RIBLNKNA	14	20
RIBLNKSH	14	10
RIBNOSTT	14	08
RIBPCBAL	0	
RIBPCBM	12	80
RIBRSET	16	04
RIBSYNC	13	80
RIBUPPER	C	

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRPCB		
0	(0)	4	RPCB		Start of ISC remote PCB
0	(0)	4	RPCBMIOS		Max PCB I/O area size
4	(4)	4	RPCBSEGL		Length of last retrieve
8	(8)	1	RPCBFlag		Flag byte
			RPCBPATH	1...	"X'80'" Previous get hold path call
9	(9)	3			Reserved
12	(C)	4			Length always fullword multiple
			RPCBLEN	.... 11..	"*-RPCB" length of RPCB

Cross Reference

Name	Hex Offset	Hex Value
DLZRPCB	0	
RPCB	0	
RPCBFlag	8	
RPCBLEN	C	0C
RPCBMIOS	0	
RPCBPATH	8	80
RPCBSEGL	4	

RPDIR - Remote PSB Directory

DSECT Name: DLZRPDIR

This DSECT describes remote PSB directory fields. The RPDIR is an extension of the PDIR. It contains information used by DL/I for CICS/VS intersystem communication (ISC) support.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRPDIR		
0	(0)	4	RPDIR		Remote PSB directory start
0	(0)	4	RPDIRSYS		System name on which remote PSB is defined
4	(4)	8	RPDIRPSB		Name of PSB to use on remote system
12	(C)	2	RPDIRLOC		Optional local PSB PDIR pointer
14	(E)	1	RPDIRFLG		Flag byte
			RPDIRORD	1	"X'01'" Local PCBS follow remote
15	(F)	1			Reserved
16	(10)	4			Length is fullword multiple
			RPDIRLEN	...1	"*-RPDIR" length of RPDIR

Cross Reference

Name	Hex Offset	Hex Value
DLZRPDIR	0	
RPDIR	0	
RPDIRFLG	E	
RPDIRLEN	10	10
RPDIRLOC	C	
RPDIRORD	E	01
RPDIRPSB	4	
RPDIRSYS	0	

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRPST		
0	(0)	4	RPST		Start of DSECT
0	(0)	4	RPSTISC1		ISC parameter 1
4	(4)	4	RPSTISC2		ISC parameter 2
8	(8)	4	RPSTISC3		ISC parameter 3
12	(C)	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 A(PCB address list)
36	(24)	4	RPSTXPSB		Local PSB PDIR entry address
40	(28)	4	RPSTXPCB		Local PSB A(PCB address list)
44	(2C)	4	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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
76	(4C)	4	RPSTREPL		Number of REPL calls issued
80	(50)	4	RPSTCHKP		Number of CHKP calls issued
			RPSTSTLN	..1. 1...	"*-RPSTACCT" Length of remote statistics section
84	(54)	4			Length always fullword multiple
			RPSTLEN	.1.1 .1..	"*-RPST" Length of RPST

Cross Reference

Name	Hex Offset	Hex Value
DLZRPST	0	
RPST	0	
RPSTACCT	2C	
RPSTACTA	19	
RPSTATUS	18	
RPSTCHKP	50	
RPSTDLET	48	
RPSTGHN	3C	
RPSTGHNP	40	
RPSTGHU	38	
RPSTGN	30	
RPSTGNP	34	
RPSTGU	2C	
RPSTISC1	0	
RPSTISC2	4	
RPSTISC3	8	
RPSTISC4	C	
RPSTISC5	10	
RPSTISC6	14	
RPSTISRT	44	
RPSTLEN	54	54
RPSTREPL	4C	
RPSTRPCB	20	
RPSTRPSB	1C	
RPSTSTLN	50	28
RPSTXPCB	28	
RPSTXPSB	24	

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. For more information, refer to "DLZQUEF0 - Queueing Facility" in section 3 of this book.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZRRD		
0	(0)	24	RRD		
0	(0)	1	RRDNQRO		Number of read-only ownerships for task
0	(0)	4	RRDPSTQF		Next RRD for task
4	(4)	1	RRDNQUP		Number of update ownerships for task
4	(4)	4	RRDPSTQB		Prior RRD for task
8	(8)	1	RRDNQEX		Number of exclusive ownerships for task
8	(8)	4	RRDRDBQF		Next RRD for resource
12	(C)	1	RRDMAXL		Current top enqueue level for resource by task
12	(C)	4	RRDRDBQB		Prior RRD for resource
16	(10)	1	RRDFLAG		Flag byte
			RRDOWNF	1	"X'01'" Task owns resource
			RRDWAITF	.... ..1.	"X'02'" Task is waiting for resource
			RRDPOWNF	.... .1..	"X'04'" Task is prime owner
16	(10)	4	RRDRDBP		RDB for resource
20	(14)	4	RRDPSTP		PST for task
			RRDLEN	...1 1...	"*-RRD" length of RRD

Cross Reference

Name	Hex Offset	Hex Value
DLZRRD	0	
RRD	0	
RRDFLAG	10	
RRDLEN	14	18
RRDMAXL	C	
RRDNQEX	8	
RRDNQRO	0	
RRDNQUP	4	
RRDOWNF	10	01
RRDPOWNF	10	04
RRDPSTP	14	
RRDPSTQB	4	
RRDPSTQF	0	
RRDRDBP	10	
RRDRDBQB	C	
RRDRDBQF	8	
RRDWAITF	10	02

SBIF - Subpool Information Table

DSECT Name: DLZSBIF

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SUBINFTA		
0	(0)	1	SUBNQFI		PST prefix number of first task in chain for enqueue subpool
1	(1)	1	SUBNQLA		PST prefix number of last task in chain for enqueue subpool
2	(2)	1	SUBBFNO		Number of buffers in this subpool
3	(3)	1	SUBBFHD		HDBFR indicator
			SUBFRSV	1...	"X'80'" DMB assigned to this subpool by HDBFR parameter
			SUBDUMP	.1..	"X'40'" Buffers associated with subpool dumped
4	(4)	40	SUBUSCHA		Beginning of the use chain
4	(4)	4	SUBUCPRE		Accumulated number of buffers in preceding subpools
8	(8)	4	SUBUCHAI(8)		Buffer use chain
40	(28)	4	SUBUCSUF		(Not used in DL/I DOS/VS)
44	(2C)	1	SUBBFSIZ		Size of 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
			SUBLEN	..1. 111.	"*-SUBINFTA" length of the subpool information table

Cross Reference

Name	Hex Offset	Hex Value
SUBBFHD	3	
SUBBFNO	2	
SUBBFSIZ	2C	
SUBDMBCT	2D	
SUBDUMP	3	40
SUBFRSV	3	80
SUBINFTA	0	
SUBLEN	2D	2E
SUBNQFI	0	
SUBNQLA	1	
SUBUCHAI	8	
SUBUCPRE	4	
SUBUCSUF	28	
SUBUSCHA	4	

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZSCD		DOS/DLI SCD dummy section
0	(0)	96	CPYRITE		Reserved for copyright information

System contents directory

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
96	(60)	8	SCD		Start of addressable SCD

System configuration section

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
96	(60)	1	SCDDLIV		DL/I version number
97	(61)	1	SCDDLIM		DL/I modification level
98	(62)	4	SCDDATE		System date-Julian
102	(66)	2	SCDMXTSK		DL/I max 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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
132	(84)	4	SCDPRHED		Entry point of program request handler (Batch = DLZPRHB0, Online = DLZPRHO0)
136	(88)	4	SCDDDBH0		Entry point of buffer handler (DLZDBH00)
140	(8C)	4	SCDDLIRE		Entry point of retrieve (DLZDLR00)

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
144	(90)	4	SCDDLICT		Entry point of call analyzer (DLZDLA00)
			SCDDLARE	..1.	"32" offset to entry point on return to call analyzer
148	(94)	4	SCDDBLNT		Entry point of data base log module (DLZRDBL0) = E.P. of log initialization until after initialization
152	(98)	4	SCDDLIDR		Entry point of delete/replace (DLZDLD00)
156	(9C)	4	SCDDLIIN		Entry point of load/insert for retrieve (DLZDDLE0)
160	(A0)	4	SCDDHDS0		Entry point to 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)
			SCDQFSCD	.... .1..	"4" displacement to SCD address field in DLZQUEF0
			SCDQFJRN	.... 1...	"8" displacement to JRNAD exit address 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		Batch = ENTRY POINT of DLZIWAIT Online = address of branch table 1st entry = entry point DLZOWAIT 2nd entry = entry point DLZCMFHK
192	(C0)	4	SCDERRMS		Entry point of error message routine (Batch = ERRORMSG, Online =DLZERMSG)

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
196	(C4)	4	SCDASE		Entry point of online task Schedule routine (DLZSCHDL)
200	(C8)	4	SCDABEND		Entry point of DL/I ABEND routine (Batch = DLZABEND online = DLZABND0)
204	(CC)	4	SCDTKTRM		Entry point of on-line 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

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			SCDDBFPL	11.1 1...	"*" label for buffer handler
216	(D8)	1	SCDBFPL		Number of buffer sub-pools
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
			SCDUSAVE	1111 .1..	"SCDPPAF" address of save area passed to user by DL/I batch initialization for PL/I high level language debugging
			SCDFLPC	1111 .1..	"SCDUSAVE" flag byte for HLL debugging
			SCDLIPLI	1...	"X'80'" 0 = in DL/I code or non PL/I language 1 = in PL/I code

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			SCDFLSAV	.1..	"X'40'" Batch STXIT PC savearea 0 = in DL/I storage 1 = in user storage
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		Reserved
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		A(user TCA) of first online task suspended for max task
276	(114)	4	SCDDLIUP		Address of batch DL/I upper boundary
276	(114)		SCDCSABA		"SCDDLIUP" address of online CICS/VS CSA
280	(118)	2	SCDTKCNT		Reserved
282	(11A)	2	SCDSPCNT		Count of suspended tasks due to maximum task
284	(11C)	1	SCDSIND		System indicator
			SCDMTI	1...	"X'80'" DL/I maximum task indicator
			SCDCMTI	.1..	"X'40'" DL/I current maximum task indicator
			SCDDELT	..1.	"X'20'" On-line indicator for PSB delete sensitivity
			SCDUPD	...1	"X'10'" On-line indicator some PSB has update sensitivity
			SCDTWFI	.... 1...	"X'08'" Task waiting for segment intent
			SCDHLRE	.... 1...	"X'08'" HLL re- entry indicator STXIT
			SCDNDMP	.... .1..	"X'04'" No dump at ABEND
			SCDNLOGI	.... ..1.	"X'02'" No data base logging to be done

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			SCDNABND	1	"X'01'" Batch no STXIT abend to be issued
			SCDNJNL	1	"X'01'" Online no CICS/VS journal in use
285	(11D)	1	SCDSIND2		System flags
			SCDSYSAB	1...	"X'80'" System ABEND on-line
			SCDSYACT	.1..	"X'40'" System task active
			SCDSYWAT	..1.	"X'20'" System task waiting
			SCDRLRST	...1	"X'10'" HD reload restart
			SCDRELOD	.... 1...	"X'08'" HD reload utility
			SCDRLABN	.... .1..	"X'04'" HD reload or reload/restart abend is in process
			SCDSYINT	.... ..1.	"X'02'" Initialization bit
			SCDSOPLG	1	"X'01'" Open records written to CICS/VS journal
286	(11E)	2	SCDNTWC		Count of suspended tasks due to scheduling intent conflict
288	(120)	4	SCDABSAV		Pointer to pseudo-ABEND save area (DLZABSAV)
292	(124)	4	SCDLSTAD		Address of CICS/VS interface address list (DFHDLIAL)
296	(128)	4	SCDMPCPT		Address of MPC partition table
300	(12C)	4	SCDEXTBA		Pointer to SCD extension
304	(130)	1	SCDDBMPS		Flag byte
			SCDXECB	1...	"X'80'" XECBs defined by MPC
			SCDPI	.1..	"X'40'" Program isolation active
			SCDRPSB	..1.	"X'20'" Remote PSB defined
305	(131)	1			Reserved
306	(132)	2			Reserved
308	(134)	4	SCDPDCA		Address of problem determination control area

Data base change log section

312	(138)	4	SCDREENT		Entry point of log write only

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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
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	1...	"X'80'" Data base log is open
			SCDDBLOR	.1..	"X'40'" Data base log open required
			SCDDBLTD	..1.	"X'20'" Disk logging used
			SCDDBLD2	...1	"X'10'" Two disk extents used
			SCDDBLSP	.... 1...	"X'08'" Pause before extent switch
			SCDDBLCJ	.... .1..	"X'04'" CICS/VS journal in use
			SCDDBASL	.... ..1.	"X'02'" Data base asynchronous log required
347	(15B)	3	SCDLOCO3		Current log count
347	(15B)	1			First byte of log count
348	(15C)	2	SCDLOCOU		Last two bytes of log count
350	(15E)	2			Reserved
352	(160)	4	SCDBKWRK		Backout log workarea pointer D
Trace section					
356	(164)	4	SCDTRACE		Entry point of trace module if present
360	(168)	8	SCDTRCNM		Name of trace module

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
368	(170)	1	SCDTRFL1		Trace option byte 1
			SCDTUSER	1...	"X'80'" User call interface
			SCDTAMOD	.1..	"X'40'" Action module trace
			SCDTRETR	..1.	"X'20'" Retrieve (for get calls)
			SCDTCPOS	...1	"X'10'" Current position information
			SCDTSEGM	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			SCDTVSAM	.... .1..	"X'04'" VSAM interface
			SCDTBHCL	.... ..1.	"X'02'" Buffer handler interface
			SCDTINDX	1	"X'01'" Requests to index maintenance
369	(171)	1	SCDTRFL2		Trace option byte 2
			SCDTOLBH	1...	"X'80'" Online trace
			SCDTPITR	.1..	"X'40'" Program isolation trace
370	(172)	2			Reserved

Statistics section

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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 max task
388	(184)	4	SCDPDUP		Number of duplicate PSBs created
392	(188)	128	SCDPATCH		DL/I patch area
			SCDLNGTH		"*-DLZSCD" length of SCD

Cross Reference

Name	Hex Offset	Hex Value	Name	Hex Offset	Hex Value
CPYRITE	0		SCDDLIPN	E2	
DLZSCD	0		SCDDLIPS	DC	
SCD	60		SCDDLIRE	8C	
SCDABEND	C8		SCDDLIS	110	
SCDABSAV	120		SCDDLIUP	114	
SCDACTBA	108		SCDDLIV	60	
SCDASE	C4		SCDDLOCT	17C	
SCDATSKC	6A		SCDDSEH0	AC	
SCDBFPL	D8		SCDDXMT0	A4	
SCDBKWRK	160		SCDERRMS	C0	
SCDCDTA	10C		SCDEXTBA	12C	
SCDCMTCT	180		SCDFLPC	F4	F4
SCDCMTI	11C	40	SCDFLSAV	F4	40
SCDCMXT	68		SCDHLRE	11C	08
SCDCOMRG	7C		SCDIWAIT	BC	
SCDCPY10	B4		SCDLIPLI	F4	80
SCDCSABA	114	0114	SCDLNGTH	188	0208
SCDCWRK	150		SCDLOCOU	15C	
SCDCWRKL	154		SCDLOCO3	15B	
SCDDATE	62		SCDLOWER	6C	
SCDDBASL	15A	02	SCDLOWID	78	
SCDDBFA	D9		SCDLSTAD	124	
SCDDBFPL	D4	D8	SCDMPCPT	128	
SCDDBLAS	144		SCDMTI	11C	80
SCDDBLCJ	15A	04	SCDMXTSK	66	
SCDDBLCL	140		SCDNABND	11C	01
SCDDBLD2	15A	10	SCDNAVID	74	
SCDDBLFW	13C		SCDNDMP	11C	04
SCDDBLNT	94		SCDNJNL	11C	01
SCDDBLO	15A	80	SCDNLOGI	11C	02
SCDDBLOP	15A		SCDNTWC	11E	
SCDDBLOR	15A	40	SCDPATCH	188	
SCDDBLSP	15A	08	SCDPDCA	134	
SCDDBLSV	148		SCDPDUP	184	
SCDDBLTD	15A	20	SCDPI	130	40
SCDDBLWO	14C		SCDPPAB	F8	
SCDDBMPS	130		SCDPPAF	F4	
SCDDDBH0	88		SCDPPFB	100	
SCDDELT	11C	20	SCDPPFF	FC	
SCDDHDS0	A0		SCDPPSTL	F0	
SCDDLARE	90	20	SCDPPSTN	F2	
SCDDLICL	A8		SCDPPSTS	EC	
SCDDLICT	90		SCDPRHED	84	
SCDDLIDL	E8		SCDPSTLN	104	
SCDDLIDM	E4		SCDQFJRN	AC	08
SCDDLIDN	EA		SCDQFSCD	AC	04
SCDDLIDR	98		SCDQUEFW	B0	
SCDDLIIN	9C		SCDQUEF0	AC	
SCDDLIM	61		SCDREENT	138	
SCDDLIPL	E0		SCDRELOD	11D	08

Cross Reference

Name	Hex Offset	Hex Value
SCDREPLN	158	
SCDRLABN	11D	04
SCDRLRST	11D	10
SCDRPSB	130	20
SCDSEQ	156	
SCDSIND	11C	
SCDSIND2	11D	
SCDSOPLG	11D	01
SCDSPCNT	11A	
SCDSTR00	D0	
SCDSYACT	11D	40
SCDSYINT	11D	02
SCDSYSAB	11D	80
SCDSYWAT	11D	20
SCDTAMOD	170	40
SCDTBHCL	170	02
SCDTCPOS	170	10
SCDTINDX	170	01
SCDTKCNT	118	
SCDTKTRM	CC	
SCDTOLBH	171	80
SCDTPITR	171	40
SCDTRACE	164	
SCDTRCNM	168	
SCDTRETR	170	20
SCDTRFL1	170	
SCDTRFL2	171	
SCDTSEGM	170	08
SCDTSKCT	174	
SCDTUSER	170	80
SCDTVSAM	170	04
SCDTWFI	11C	08
SCDUPD	11C	10
SCDUPPER	70	
SCDUSAVE	F4	F4
SCDWAIT	106	
SCDXECB	130	80

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SCDEXTDS		
0	(0)	4	SCDELECB		Logger I/O ECB
4	(4)	4	SCDESECB		System enqueue ECB
8	(8)	4	SCDEFECB		System function call ECB
12	(C)	4	SCDEVSEX		Address of VSAM exception 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
			SCDEXLEN	..11 .1..	"*-SCDEXTDS" length of SCD extension

Batch usage of the SCD Extension

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	4	SCDEREEN		Address of utility block call entry point
4	(4)	4	SCDEABEX		Address of STXIT ABEND routine (DLZAABND)
8	(8)	4	SCDEABSV		Address of STXIT ABEND save area
12	(C)	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
24	(18)	4	SCDAPSTR		Application program start address
28	(1C)	4	(2)		Not used in batch

The following constants have the same labels as online

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
36	(24)	4			Current entry in incore table
40	(28)	4			End address+1 of trace table
44	(2C)	4			Start address of trace table
48	(30)	4			Reserved

Cross Reference

Name	Hex Offset	Hex Value
SCDAPSTR	18	
SCDEABEX	4	
SCDEABSV	8	
SCDEFECB	8	
SCDEIDNX	18	
SCDEIDST	14	
SCDEIDWK	1C	
SCDELECB	0	
SCDEMSGT	20	
SCDEPASS	10	
SCDEPCEX	C	
SCDEREEN	0	
SCDESECB	4	
SCDETRAN	10	
SCDETRSV	14	
SCDETRTB	24	
SCDETRTE	28	
SCDETRTS	2C	
SCDEVSEX	C	
SCDEXLEN	30	34
SCDEXTDS	0	

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).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SDB		
0	(0)	8	SDBSYM		Segment symbolic name
0	(0)	4	SDBLTP		Prior segment on logical twin chain
4	(4)	4	SDBLTN		Next segment on logical twin chain
8	(8)	1	SDBLEVEL		Level of this segment (logical)
9	(9)	1	SDBORGN		Organization of data base containing segment
			SDBORGRI	.1.. .1..	"X'44'" This segment is root of index
			SDBORGHD	..1.	"X'20'" This segment is in a HDAM organization
			SDBORGHI	...1	"X'10'" This segment is in a HIDAM organization
			SDBORGH2	.... 1...	"X'08'" (Not used in DL/I DOS/VS)
			SDBORGSH	.... .1.1	"X'05'" This segment is in a simple HISAM organization
			SDBORGH1	.... .1..	"X'04'" This segment is in a HISAM organization
			SDBORGHS	.... ..1.	"X'02'" This segment is in a HSAM organization
			SDBORGSS	1	"X'01'" This segment is in a simple HSAM organization
10	(A)	1	SDBF3		Call sensitivity
			SDBSENG	1...	"X'80'" Sensitivity is read only
			SDBSENI	.1..	"X'40'" Sensitivity is insert
			SDBSENR	..1.	"X'20'" Sensitivity is replace
			SDBSEND	...1	"X'10'" Sensitivity is delete
			SDBSENK	.... 1...	"X'08'" Sensitivity is key only
			SDBSENP	.... .1..	"X'04'" Sensitivity is path only

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
			SDBSENX	.... ..1.	"X'02'" Sensitivity is exclusive
			SDBSENL	1	"X'01'" Sensitivity is load
11	(B)	1	SDBF4		Flags
			SDBPOSL	.... ..1.	"X'02'" Position lost
			SDBCISP	.... .1..	"X'04'" Control interval split in HISAM KSDS
			SDBALTSQ	.1..	"X'40'" Secondary index is main processing sequence
			SDBALTSC	..1.	"X'20'" Secondary index search fields require conversion
					"X'10'" Reserved
			SDBDCHG	1	"X'01'" Temporary switch for replace data changed
12	(C)	1	SDBPHYCD		Segment code
12	(C)	4	SDBDDIR		DMB directory address
16	(10)	4	SDBNSDB		Next SDB for this PSDB
20	(14)	4	SDBPSDB		Address of PSDB
24	(18)	1	SDBKEYLN		Executable key length of key field
24	(18)	4	SDBPARA		Parent SDB (address of PCB for root SDB) address of prior SDB on 'SDBTARG' chain for generated SDBs (SDBGEN on in SDBTFLG)
28	(1C)	4	SDBDSGA		Address of the DSG section of JCB for data set containing segment

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
32	(20)	1	SDBTFLG		Logical relationship code
			SDBPPTSP	11..	"X'C0'" Segment is physical parent of target of SDBPARA
			SDBPPSP	1...	"X'80'" Segment is physical parent of SDBPARA
			SDBPCTSP	.1..	"X'40'" Segment is physical child of target of SDBPARA
			SDBLCPK	..1.	"X'20'" (Not used in DL/I DOS/VS)
			SDBGEN	...1	"X'10'" This SDB is a generated SDB
			SDBSPP	.... 1...	"X'08'" Segment is a virtual logical child
			SDBSNX	.... .1..	"X'04'" Segment is retrieved via index
			SDBSLC	.... ..1.	"X'02'" See PLM for detailed explanation
			SDBSLP	1	"X'01'" Segment is a logical child
32	(20)	4	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	1...	"X'80'" This logical parent segment has a counter
			SDBPTF	.1..	"X'40'" This segment has a physical twin forward pointer
			SDBPTB	..1.	"X'20'" This segment has a physical twin backward pointer
			SDBPP	...1	"X'10'" This segment has a physical parent pointer
			SDBLTFD	.... 1...	"X'08'" This segment has a logical twin forward pointer
			SDBLTBK	.... .1..	"X'04'" This segment has a logical twin backward pointer
			SDBLP	.... ..1.	"X'02'" This segment has a logical parent pointer
			SDBHIER	1	"X'01'" (Not used in DL/I DOS/VS)

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
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
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 In generated SDB for 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	1	"X'01'" SDB expansion for secondary index processing sequence is present (secondary index is main processing sequence)
			SDBFLS	.... ..1.	"X'02'" Segment has field level sensitivity
56	(38)	4	SDBXPANS		SDB expansion address
			SDBEND	..11 11..	"*" end of SDB entry
			SDBLEN	..11 11..	"SDBEND-SDBSYM" length of each SDB

Cross Reference

Name	Hex Offset	Hex Value
SDB	0	
SDBALTSC	B	20
SDBALTSQ	B	40
SDBCISP	B	04
SDBCTR	25	80
SDBDCHG	B	01
SDBDDIR	C	
SDBDSGA	1C	
SDBEND	38	3C
SDBFLS	38	02
SDBF3	A	
SDBF4	B	
SDBGEN	20	10
SDBHIER	25	01
SDBKEYFD	28	
SDBKEYLN	18	
SDBLCPK	20	20
SDBLEN	38	3C
SDBLEVEL	8	
SDBLP	25	02
SDBLTBK	25	04
SDBLTFD	25	08
SDBLTN	4	
SDBLTP	0	
SDBNSDB	10	
SDBORGHD	9	20
SDBORGHI	9	10
SDBORGHS	9	02
SDBORGH1	9	04
SDBORGH2	9	08
SDBORGN	9	
SDBORGRI	9	44
SDBORGSH	9	05
SDBORGSS	9	01
SDBPARA	18	
SDBPCB	27	
SDBPCF	26	
SDBPCTSP	20	40
SDBPHYCD	C	
SDBPOSC	30	
SDBPOSL	B	02
SDBPOSN	34	
SDBPOSP	2C	
SDBPP	25	10
SDBPPSP	20	80
SDBPPTSP	20	C0
SDBPSDB	14	
SDBPTB	25	20
SDBPTDS	25	
SDBPTF	25	40

Name	Hex Offset	Hex Value
SDBPTNO	24	
SDBSEND	A	10
SDBSENG	A	80
SDBSENI	A	40
SDBSENK	A	08
SDBSENL	A	01
SDBSENP	A	04
SDBSENR	A	20
SDBSENX	A	02
SDBSLC	20	02
SDBSLP	20	01
SDBSNX	20	04
SDBSPP	20	08
SDBSYM	0	
SDBTARG	20	
SDBTFLG	20	
SDBXFL	38	
SDBXPANS	38	
SDBXPRES	38	01

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBSEC		
0	(0)	1	DMBSCDE		Code byte
			DMBSLP	1	"X'01'" Secondary list describes a logical parent
			DMBSLC	.... ..1.	"X'02'" Secondary list describes a logical child
			DMBSRCH	.... .1..	"X'04'" Secondary list describes index search field(s)
			DMBSLCF	.... 1...	"X'08'" Secondary list describes logical twin sequence field
			DMBSLCPR	...1 ...1	"X'11'" (Not used in DL/I DOS/VS)
			DMBSOURC	..1.	"X'20'" Secondary list describes index data field(s)
			DMBSUBSQ	..1. .1..	"X'24'" Secondary list describes index SUBSEQ field(s)
			DMBEXTRN	.1..	"X'40'" Secondary list describes user index exit routine
			DMBINDXD	.1.. .1..	"X'44'" Secondary list describes index target segment as seen from index pointer segment
			DMBNXISS	.11.	"X'60'" Secondary list describes index relationship as seen from index source segment
			DMBNXXDS	.11. .1..	"X'64'" Secondary list describes index relationship as seen from index target segment. this list is not present if ISS = target
			DMBSND	1...	"X'80'" Last entry in secondary list

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
The following fields are listed by code type					
Code 01 - describes logical parent					
			DMBSFLDS	1	"*"
1	(1)	1	DMBSFLG		
			DMBVKY	111. .1.1	"C'V'" Key of logical parent is virtual
			DMBPKY	11.1 .111	"C'P'" (Not used in DL/I DOS/VS)
2	(2)	2	DMBSFD		Logical parent key length
4	(4)	1	DMBSECSC		Segment code of referenced segment
4	(4)	4	DMBSECDB		DDIR address of referenced data base
8	(8)	8	DMBSECNM		Segment name of referenced segment
Code 02 - describes logical child					
1	(1)	1	DMBSLCIR		Logical twin sequence insert rule
2	(2)	2	DMBSLCFL		Number of first and last logical child pointers in logical parent prefix remainder of fields same as code 01
Code 04 - describes index SRCH fields					
1	(1)	1	DMBFDFLG(5)		Five one-byte flags associated with the following FDB offsets
			DMBSYM1	.... 1...	"X'08'" First part of symbolic pointer
			DMBSYMN1	.... .1..	"X'04'" Not first part of symbolic pointer (middle or last)
			DMBSYSFD	.... ..1.	"X'02'" This slot for system related field
			DMBFDUSE	1	"X'01'" This slot in use
			DMBFDONE	...1	"X'10'" This entry processed by block builder
6	(6)	2	DMBFDOFF(5)		Offset to FDB from first FDB of ISS if this slot is in use, otherwise zero

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
Code 08 - describes logical twin sequence field					
1	(1)	1	DMBSFPSC		Virtual logical child physical segment code
2	(2)	8	DMBSFNAM		FDB field name
10	(A)	2	DMBSFOFF		Offset to field in segment
12	(C)	1	DMBSFCEN		Code byte (same as FDBDCENF in FDB)
13	(D)	1	DMBSFLEN		Executable field length
14	(E)	2	DMBXSOFF		Offset to field in indexed segment
Code 20 - describes DDATE field					
					Same fields as code 04
Code 24 - describes SUBSEQ field					
					Same fields as code 04
Code 40 - describes index exit routine					
1	(1)	1	DMBSFLG1		Flag byte
			DMBSNULL	1	"X'01'" Null field present
			DMBEXIT	.... ..1.	"X'02'" Exit routine present
			DMBNLXIT	11	"X'03'"
			DMBEXLOD	.... .1..	"X'04'" Exit routine has been loaded
2	(2)	2			Reserved
4	(4)	1	DMBNBYTE		If index field equals this byte bypass indexing
4	(4)	4	DMBXITAD		Address of index maintenance parameter CSECT
8	(8)	8	DMBSUPRT		Suppression routine name

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
Code 44 - describes index target segment					
1	(1)	1	DMBSKYLN		Executable length of key
2	(2)	2	DMBSOFF		Offset to PSDB address pointer of index target segment
4	(4)	1	DMBXDSSC		Segment code of index target segment
4	(4)	4	DMBXDSDB		DDIR address of index target segment
8	(8)	1	DMBXDSC		Segment code of index target segment
8	(8)	4	DMBXPSDB		PSDB address of index target segment
12	(C)	1	DMBXDFLG		Code byte from associated FDB
			DMBXDLST	1...	"X'80'" Last FDB in list
			DMBXDSYM	.1..	"X'40'" Index pointer is symbolic
			DMBXDSSS	..1.	"X'20'" Pointer contained in source/subseq data
			DMBXDSPC	...1	"X'10'" Special FDB for secondary index
			DMBXDCON	.... 1...	"X'08'" Constant present
			DMBXDSSQ	.... .1..	"X'04'" SUBSEQ present
			DMBXDSOR	.... ..1.	"X'02'" (Not used in DL/I DOS/VS)
			DMBXDEQ	1	"X'01'" XDS = ISS
13	(D)	1	DMBXDPAD		Padding constant
14	(E)	2	DMBSYMOF		Offset to symbolic pointer indexing segment
Code 60 - describes index from ISS					
1	(1)	3			Same as code 44
4	(4)	1	DMBXNSSC		Segment code of index pointer segment
4	(4)	4	DMBXNSDB		DDIR address of index remaining fields same as code 44

Code 64 - describes index from index target

1	(1)	1			Same as code 44
2	(2)	2	DMBISSOF		Offset to code 60 from from start of ISS secondary list
4	(4)	4			Same as code 60
8	(8)	1	DMBISSSC		Segment code of index source segment
8	(8)	4	DMBIPSDB		PSDB address of index source segment
12	(C)	4			Same as code 44
			DMBSECND	...1	"*" End of each secondary list entry
			DMBSECLN	...1	"DMBSECND-DMBSEC" length of each secondary list entry

Cross Reference

Name	Hex Offset	Hex Value
DMBEXIT	1	02
DMBEXLOD	1	04
DMBEXTRN	0	40
DMBFDFLG	1	
DMBFDOFF	6	
DMBFDONE	1	10
DMBFDUSE	1	01
DMBINDXD	0	44
DMBIPSDB	8	
DMBISSOF	2	
DMBISSSC	8	
DMBNBYTE	4	
DMBNLXIT	1	03
DMBNXISS	0	60
DMBNXXDS	0	64
DMBPKY	1	D7
DMBSCDE	0	
DMBSEC	0	
DMBSECDB	4	
DMBSECLN	C	10
DMBSECND	C	10
DMBSECNM	8	
DMBSECSC	4	
DMBSFCEN	C	
DMBSFD	2	
DMBSFLDS	0	01
DMBSFLEN	D	
DMBSFLG	1	
DMBSFLG1	1	
DMBSFNAM	2	
DMBSFOFF	A	
DMBSFPSC	1	
DMBSKYLN	1	
DMBSLC	0	02
DMBSLCF	0	08
DMBSLCFL	2	
DMBSLCIR	1	
DMBSLCPR	0	11
DMBSLP	0	01
DMBSND	0	80
DMBSNULL	1	01
DMBSOFF	2	
DMBSOURC	0	20
DMBSRCH	0	04
DMBSUBSQ	0	24
DMBSUPRT	8	
DMBSYMN1	1	04
DMBSYMOF	E	
DMBSYM1	1	08
DMBSYSFD	1	02
DMBVKY	1	E5
DMBXDCON	C	08
DMBXDEQ	C	01
DMBXDFLG	C	
DMBXDLST	C	80
DMBXDPAD	D	
DMBXDSC	8	
DMBXDSDB	4	
DMBXDSOR	C	02
DMBXDSPC	C	10
DMBXDSSC	4	
DMBXDSSQ	C	04
DMBXDSSS	C	20
DMBXDSYM	C	40
DMBXITAD	4	
DMBXNSDB	4	
DMBXNSSC	4	
DMBXPSDB	8	
DMBXSOFF	E	

SGT - Segment Table

DSECT Name: DLZPRSGT

This DSECT describes the segments used by the partial reorganization process. It is built during the DBD analysis phase and is 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).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SGT		
0	(0)	4	SGTSTART		
0	(0)	8	SGTCName		Segment name
8	(8)	4	SGTROLD		Old RBA of last segment un/reloaded
12	(C)	4	SGTRNEW		New RBA of last segment reloaded
16	(10)	4	SGTFCNT1		Statistical counter
20	(14)	4	SGTFCNT2		Statistical counter
24	(18)	4	SGTFCNT3		Statistical counter
28	(1C)	4	SGTFCNT4		Statistical counter
32	(20)	4	SGTFCNT5		Statistical counter
36	(24)	4	SGTFCNT6		Statistical counter
40	(28)	2	SGTODBT		Offset to DBT entry for this segments DB
42	(2A)	2	SGTOKEY		Segment key start POS roots only
44	(2C)	2	SGTHKLEN		Segment key length roots only
46	(2E)	2	SGTHPLEN		Segment prefix length
48	(30)	2	SGTHDLEN		Segment data length maximum if variable
50	(32)	2	SGTORACT		Offset in ACT to first reload action
52	(34)	2	SGTOSACT		Offset in ACT to first scan action
54	(36)	2			Spare offset field
56	(38)	2	SGTOPCF		Offset in SGT to first physical child of this
58	(3A)	2	SGTOSIBL		Offset in SGT to next SESIBLING segment
60	(3C)	1	SGTCSC		DL/I segment code
61	(3D)	1	SGTCDS		DL/I data set code

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
62	(3E)	1	SGTCLEV		DL/I level code
63	(3F)	1			Reserved
64	(40)	1	SGTGATR1		Segment physical attributes
			SGTQMOVE	1...	"X'80'" Segment to be moved for reorganization
			SGTQHIDR	.1..	"X'40'" Segment is HIDAM root
			SGTQPP	..1.	"X'20'" Segment has PP pointer
			SGTQPTB	...1	"X'10'" Segment has PTB pointer
			SGTQPCL	.... 1...	"X'08'" Segments parent has PCL pointer to this
			SGTQHIER	.... .1..	"X'04'" Segment has hierarchic pointers
			SGTQHB	.... ..1.	"X'02'" Segment has hierarchic backward pointer
			SGTQVRLN	1	"X'01'" Segment is variable length
65	(41)	1	SGTGATR2		Segment logical attributes
			SGTQLC	1...	"X'80'" Segment is a logical child
			SGTQUNID	.1..	"X'40'" Segment is logical child in unidirectional relation
			SGTQVPR	..1.	"X'20'" Segment has virtual pair
			SGTQPPR	...1	"X'10'" Segment has physical pair
			SGTQSYM	.... 1...	"X'08'" Segment has only symbolic pointer to logical parent
			SGTQDRCT	.... .1..	"X'04'" Segment has direct pointer to logical parent
			SGTQLTB	.... ..1.	"X'02'" Segment has LTB pointer
			SGTQLCL	1	"X'01'" Segments logical parent has LCL pointer to this

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
66	(42)	1	SGTGATR3		Segment logical and index attributes
			SGTQNOLT	1...	"X'80'" Virtually paired with no LTWIN pointers
			SGTQLP	.1..	"X'40'" Segment is a logical parent
			SGTQXDRT	.... 1...	"X'08'" Segment is index segment with direct pointer
			SGTUNIQ	.... .1..	"X'04'" Segment is in a unique index
			SGTQXSX	.... ..1.	"X'02'" Segment is index segment with SX field
			SGTQPTF	1	"X'01'" Segment has PTF pointer
67	(43)	1	SGTGATR4		Segment PSB attributes
			SGTQKSEN	1...	"X'80'" Key only sensitivity required
			SGTQDSEN	.1..	"X'40'" Data sensitivity required
			SGTQNPRO	..1.	"X'20'" Segment not processed used to reach PC
			SGTQSOPT	.... ..1.	"X'02'" Scan is option for this segment
			SGTQSCAN	1	"X'01'" This segment will be scanned

Segment Extension Table

This part of the DSECT is for additional information about the segments used by the partial re-organization process. It contains offsets needed to create the action table (ACT). It is created during the DBD analysis phase.

68	(44)	2	SGXOCTR		Offset in prefix of log rel counter
70	(46)	2	SGXOPTF		Offset in prefix of PTF pointer
72	(48)	2	SGXOPTB		Offset in prefix of PTB pointer
74	(4A)	2	SGXOPP		Offset in prefix of PP pointer
76	(4C)	2	SGXOLTF		Offset in prefix of LTF pointer
78	(4E)	2	SGXOLTB		Offset in prefix of LTB pointer
80	(50)	2	SGXOLP		Offset in prefix of LP pointer
82	(52)	2	SGXOHIER		Offset in prefix of hier pointer
84	(54)	2	SGXOHB		Offset in prefix of hier back pointer
86	(56)	2	SGXOPCF		Offset in segments PP of PCF to this segment
88	(58)	2	SGXOLCF		Offset in segments logical parent of LCF to this segment
90	(5A)	2	SGXOSPP		Offset in SGT of physical parent
92	(5C)	2	SGXOSLP		Offset in SGT of logical parent
94	(5E)	2	SGXOPAIR		Offset in SGT of physical pair
96	(60)	2	SGXOTARG		Offset in SGT of target of this segment
98	(62)	2	SGXOSRCE		Offset in SGT of source of this segment
100	(64)	2	SGXOPCWK		Work area to hold offset to first physical child pointer
102	(66)	2	SGXOLCWK		Work area to hold offset to first logical child pointer
104	(68)	4	SGXFBLK		Last block un/reloaded used in PART2
			SGTLLEN	.11. 11..	"*-SGTSTART" length of a SGT ENTRY

Cross Reference

Name	Hex Offset	Hex Value
SGT	0	
SGTCDS	3D	
SGTCLEV	3E	
SGTCName	0	
SGTCSC	3C	
SGTFCNT1	10	
SGTFCNT2	14	
SGTFCNT3	18	
SGTFCNT4	1C	
SGTFCNT5	20	
SGTFCNT6	24	
SGTGATR1	40	
SGTGATR2	41	
SGTGATR3	42	
SGTGATR4	43	
SGTHDLEN	30	
SGTHKLEN	2C	
SGTHPLEN	2E	
SGTLLEN	68	6C
SGTODBT	28	
SGTOKEY	2A	
SGTOPCF	38	
SGTORACT	32	
SGTOSACT	34	
SGTOSIBL	3A	
SGTQDRCT	41	04
SGTQDSEN	43	40
SGTQHB	40	02
SGTQHIDR	40	40
SGTQHIER	40	04
SGTQKSEN	43	80
SGTQLC	41	80
SGTQLCL	41	01
SGTQLP	42	40
SGTQLTB	41	02
SGTQMOVE	40	80
SGTQNOLT	42	80
SGTQNPRO	43	20
SGTQPCL	40	08
SGTQPP	40	20
SGTQPPR	41	10
SGTQPTB	40	10
SGTQPTF	42	01
SGTQSCAN	43	01
SGTQSOPT	43	02
SGTQSYM	41	08
SGTQUNID	41	40
SGTQVPR	41	20
SGTQVRLN	40	01
SGTQXDRT	42	08
SGTQXSX	42	02
SGTRNEW	C	
SGTROLD	8	
SGTSTART	0	
SGTUNIQ	42	04
SGXFBLK	68	
SGXOCTR	44	
SGXOHB	54	
SGXOHIER	52	
SGXOLCF	58	
SGXOLCWK	66	
SGXOLP	50	
SGXOLTB	4E	
SGXOLTF	4C	
SGXOPAIR	5E	
SGXOPCF	56	
SGXOPCWK	64	
SGXOPP	4A	
SGXOPTB	48	
SGXOPTF	46	
SGXOSLP	5C	
SGXOSPP	5A	
SGXOSRCE	62	
SGXOTARG	60	

SQLID - Userid Control Block

DSECT Name: DLZSQLID

This USERID control block is used to pass data from DLZUACBO to DLZDCBL0 in the Application Control Block Maintenance Utility (ACBGEN). This control block contains information when the 'USERID' parameter is used on the BUILD statement.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	USERIDCB		USERID control block
0	(0)	1	USRIDFLG		Flags byte for USERID parameter
			PROUSRID	1	"X'01'" USERID specified on build
1	(1)	8	PASSWORD		Password for SQL/DS connect
12	(C)	4	IOAREAAD		Address of SQL IOAREA
16	(10)	4	DLBDPADD		Address of DLZDLBDP module
20	(14)	4	DLBPPADD		Address of DLZDLBPP module
			USERIDLN	...1 1...	"*-USERIDCB" length of USERID control block

Cross Reference

Name	Hex Offset	Hex Value
DLBDPADD	10	
DLBPPADD	14	
IOAREAAD	C	
PASSWORD	1	
PROUSRID	0	01
USERIDCB	0	
USERIDLN	14	18
USRIDFLG	0	

SSA - Segment Search Argument

DSECT Name: DLZSSA

This DSECT describes the DL/I HLPI Segment Search Argument fields. It includes the command codes and qualification arguments. It points to the SSAX DSECT for the length of the SSA.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SSAXPTR0		The following field is not mapped by DLZSSA DSECT
0	(0)	4	SSAXPTR		Pointer that points to SSAX, always placed before SSA
0	(0)	0	SSA		
0	(0)	8	SSASEGNM		Segment name
8	(8)	1	SSAASTRK		Asterisk (*) or blank if not qualified
9	(9)	4	SSACMND		Command codes
13	(D)	1	SSALPARN		Left parenthesis
			SSAQS	.... 111.	"*" SSA qualification sections

Note:
The next four fields are repeated for each qualification section of a boolean SSA. The SSABO field will have a right parenthesis in the last (or the only) qualification section.

14	(E)	8	SSAFLDNM		Field name
22	(16)	2	SSARO		Relational operators

Note:
Length of SSARO field is maintained in the equate symbol HLPIOPRL in DSECT DLZHLPIL.

24	(18)	255	SSAKEYFL		Maximum key field value
279	(117)	1	SSABO		Boolean operator or right parenthesis

The actual position of SSABO depends on the length of the SSAKEYFL field.

Note:
Length of SSABO field is maintained in the equate symbol HLPIOPBL in DSECT DLZHLPIL

			SSAQSLN		"*-SSAQS" length of Q.S. section
			SSALGTH		"*-SSA" length of SSA

Cross Reference

Name	Hex Offset	Hex Value
SSA	0	
SSAASTRK	8	
SSABO	117	
SSACMND	9	
SSAFLDNM	E	
SSAKEYFL	18	
SSALGTH	117	0118
SSALPARN	D	
SSAQS	D	0E
SSAQSLN	117	010A
SSARO	16	
SSASEGNM	0	
SSAXPTR	0	
SSAXPTRO	0	

SSAP - Segment Search Appendage

DSECT Name: DLZSSAP

This DSECT describes the fields contained in the DL/I HLPI Segment Search Argument get path call appendage.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SSAP		
0	(0)	4			
0	(0)	8	SSAPSEGM		Segment name
8	(8)	1	SSAPFLAG		SSA flag
			SSAPVARL	1...	"X'80'" Variable length segment
			SSAPDATT	.1..	"X'40'" Data to be transferred
			SSAPPROC	..1.	"X'20'" Segment already processed
9	(9)	3	SSAPIOA		Address of IOAREA for this segment
12	(C)	2	SSAPLIOA		Length of the IOAREA for this segment
14	(E)	2	SSAPSGOF		Offset to length of the destination parent
			SSAPLEN	...1	"*-SSAPSEGM" length of SSA appendage
			SSAPSTOR	1111	"SSAPLEN*15" length for required number of SSA appendages
			SSAAPLN	.... 1...	"*-SSAPFlag" length of appendage information

Cross Reference

Name	Hex Offset	Hex Value
SSAAPLN	E	08
SSAP	0	
SSAPDATT	8	40
SSAPFlag	8	
SSAPIOA	9	
SSAPLEN	E	10
SSAPLIOA	C	
SSAPPROC	8	20
SSAPSEGM	0	
SSAPSGOF	E	
SSAPSTOR	E	F0
SSAPVARL	8	80

SSAX - Segment Search Argument Extension

DSECT Name: DLZSSAX

This DSECT describes the DL/I HLPI Segment Search Argument extension fields. It includes the SSA length, the I/O area address and length, and several flags for processing. This is an extension to the SSA DSECT to allow multiple Boolean qualification conditions and any further expansion.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SSAX		
0	(0)	1	SSAXFlag		SSA flag
			SSAXVARL	1...	"X'80'" Variable length segment
			SSAXDATT	.1..	"X'40'" Data to be transferred
			SSAXPROC	..1.	"X'20'" Segment already processed
1	(1)	3	SSAXIOA		Address of IOAREA for this segment
4	(4)	2	SSAXLIOA		Length of the IOAREA for this segment
6	(6)	2	SSAXSGOF		Offset to length of the destination parent
8	(8)	2	SSACLEN		SSA length, less than or equal to SSAXSTOR
			SSAMLEN		"304" maximum length for non-HLPI SSA
10	(A)	2	SSAXSTOR		Length of storage acquired for SSA
			SSAXLEN	.... 11..	"*-SSAX" length of SSAX

Cross Reference

Name	Hex Offset	Hex Value.
SSACLEN	8	
SSAMLEN	8	0130
SSAX	0	
SSAXDATT	0	40
SSAXFlag	0	
SSAXIOA	1	
SSAXLEN	A	0C
SSAXLIOA	4	
SSAXPROC	0	20
SSAXSGOF	6	
SSAXSTOR	A	
SSAXVARL	0	80

STA - Statistics Table

DSECT Name: DLZPRSTA

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	SGT		
0	(0)	4			
0	(0)	2	STBLCT		Block count
2	(2)	80	STBLBK40		Counters for blocks 1 to 40
82	(52)	2	STBLBK41		Counter for blocks over 40
84	(54)	2	STMXBL		Maximum number blocks this range
88	(58)	4	STHASHS		Number blocks over 40
92	(5C)	4	STLOHICT(20)		10 pairs of low-high block numbers
172	(AC)	4	STROV(10)		For reload
212	(D4)	4	STHDOV		HDAM roots in overflow
216	(D8)	2	STNDCNT		
0	(0)	216	STATCNTR		Length statistics counters
216	(D8)	2	STRG		Range counter for statistics

Cross Reference

Name	Hex Offset	Hex Value
SGT	0	
STATCNTR	0	
STBLBK40	2	
STBLBK41	52	
STBLCT	0	
STHASHS	58	
STHDOV	D4	
STLOHICT	5C	
STMXBL	54	
STNDCNT	D8	
STRG	D8	
STROV	AC	

SUIB - 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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZUIB		
0	(0)	4	UIB		Extended call user interface block
0	(0)	4	UIBPCBAL		PCB address list
4	(4)	2	UIBRCODE		DL/I return codes
4	(4)	1	UIBFCTR		Return code
5	(5)	1	UIBDLTR		Additional information
6	(6)	1	(2)		Reserved
8	(8)	4			Length is fullword multiple
			UIBLEN	.... 1...	"*-UIB" length of UIB

Cross Reference

Name	Hex Offset	Hex Value
DLZUIB	0	
UIB	0	
UIBDLTR	5	
UIBFCTR	4	
UIBLEN	8	08
UIBPCBAL	0	
UIBRCODE	4	

TSQE - Temporary Storage Queue Entry

DSECT Name: DLZTSQE

The DLZTSQE macro maps the contents of the entries in the CICS/VS temporary storage queue used by the MPS Restart facility. The name of this temporary storage queue is "DLZTSQ00". It is used to maintain the checkpoint IDs of the combined DL/I-VSE checkpoints taken by MPS batch jobs. There is one non-blank entry for each MPS batch job currently using the MPS Restart facility and also for each MPS batch job which abnormally ended after having taken a combined DL/I-VSE checkpoint.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZTSQE		
0	(0)	1	TSQFlag		Flag byte
			TSQACTV	11.. ...1	"C'A'" TSQ entry active
			TSQBLANK	.1..	"C' '" TSQ entry initial character
1	(1)	5	TSQRSRVD		Reserved for future use
6	(6)	4	TSQCPID		Checkpoint ID
10	(A)	26	TSQJBID		Job ID next 4 fields
10	(A)	8	TSQJBNM		Job name
18	(12)	2	TSQPTID		Partition ID
20	(14)	8	TSQDATE		Job start date
28	(1C)	8	TSQTIME		Job start time
			TSQLEN	..1. .1..	"*-DLZTSQE" length of TSQ entry

Cross Reference

Name	Hex Offset	Hex Value
DLZTSQE	0	
TSQACTV	0	C1
TSQBLANK	0	40
TSQCPID	6	
TSQDATE	14	
TSQFlag	0	
TSQJBID	A	
TSQJBNM	A	
TSQLEN	1C	24
TSQPTID	12	
TSQRSRVD	1	
TSQTIME	1C	

DLZTWAB - Transaction Work Area

MACRO 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
- Scheduling MPS batch jobs
- Online message module

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZTWAB		
0	(0)	4	TWABPC		
0	(0)	1	TWAMPSFG		BPC flag byte
			TWABPCOK	1...	"X'80'" BPC abnormal termination processing completed
			TWAEOJSW	.1..	"X'40'" EOJ processing reached for MPS batch partition
1	(1)	3	TWAMPCPT		Address of MPC partition table
4	(4)	1	TWABPCID		Batch partition XECB identifier
5	(5)	3	TWAMPCE		Address of specific MPC partition table entry

The following is the BPC's CICS/VS WAITM ECB list, delimiter and XECB.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
8	(8)	4	TWAWLIST		
8	(8)	4	TWAXCB2		Pointer to BPC's XECB (DLZXCBN2)
12	(C)	4	TWAXCB3		Pointer to ABEND XECB (DLZXCBN3)
16	(10)	4	TWAXCBDL		ECB list delimiter
20	(14)	4	TWAXCBN2		XECB for BPC

The following fields are used for communication with the batch partition.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
24	(18)	8	TWAXCBN1		XECBNSME for batch initialization
32	(20)	4	TWAN1PTR		XECBTAB table entry address for batch initializations XECB (DLZXCBN1)

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description

The following fields are used for the BPC's DL/I scheduling CALL parameter list and the PSBNAME to be scheduled.

36	(24)	4	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
48	(30)	7	TWAPSBNM		PSB-name
55	(37)	1	TWAPSBDL		PSB-name delimiter

The following field contains the SCD address.

56	(38)	4	TWABPSCD		SCD address

The following fields are used as an I/O buffer during temporary storage queue (TSQ) processing for MPS Restart.

60	(3C)	4	TWATSBUF		TSQ entry I/O buffer
60	(3C)	4	TWATSLEN		4 byte length field
64	(40)	36	TWATSQE		TSQ entry
			TWATSQBL	..1. 1...	"*-TWATSBUF" TSQ buffer length

Batch partition controller register save area

100	(64)	72	TWABPCSV		BPC register save area

The following are the parameter list pointers, parameters and message fillers passed to the DL/I on-line message module DLZERMSG for all BPC messages.

172	(AC)	4	TWAMSG		
172	(AC)	4	TWAMSGNO		Message number pointer for all BPC messages
176	(B0)	4	TWAMSGID		For messages DLZ082I, DLZ084I, DLZ103I, DLZ123I, pointer to the partition ID; for message DLZ104I, pointer to the BPC module ID; for message DLZ127I, pointer to the job name
180	(B4)	4	TWAMSG01		For message DLZ082I and DLZ084I module name pointer; for message DLZ103I the termination condition pointer and delimiter; for message DLZ104I the CICS/VS abend code pointer and delimiter for message DLZ123I, pointer to the routine name; for message DLZ127I, pointer to the partion ID

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
184	(B8)	4	TWAMSG02		For message DLZ082I and pointer; for message DLZ104I the PSW pointer and delimiter; for message DLZ123I, pointer to dfhts message DLZ127I, pointer to the message text offset
188	(BC)	4	TWAMSG03		For message DLZ082I and XECBName pointer; for message DLZ123I, CICS/VS response code pointer and delimiter; for message DLZ127I, checkpoint ID pointer and delimiter
192	(C0)	4	TWAMSG04		For message DLZ082I and DLZ084I the return code pointer and delimiter
196	(C4)	4	TWAPSW(2)		Program interrupt PSW
204	(CC)	2	TWAMPSID		Batch partition ID of the form BG,F1,F2,...
214	(D6)	2	TWARCode		Return code from XECBTAB macro or response code from DFHTS macro
216	(D8)	10	TWACOND		BPC termination condition (abnormally or normally)
226	(E2)	4	TWABEND		CICS/VS abend code (ASRA)
230	(E6)	2	TWARSTRT		Restart message text offset
230	(E6)	1			First byte contains zeros
231	(E7)	1	TWAOFFST		Restart message text offset
232	(E8)	24			Reserved
			TWABPCLN		"*-TWABPC" length of BPC's TWA

Cross Reference

Name	Hex Offset	Hex Value
DLZTWAB	0	
TWABEND	E2	
TWABPC	0	
TWABPCID	4	
TWABPCLN	E8	0100
TWABPCOK	0	80
TWABPCSV	64	
TWABPSCD	38	
TWACALL	28	
TWACOND	D8	
TWAEOJSW	0	40
TWAMPCE	5	
TWAMPCPT	1	
TWAMPSFG	0	
TWAMPSID	CC	
TWAMSG	AC	
TWAMSGID	B0	
TWAMSGNO	AC	
TWAMSG01	B4	
TWAMSG02	B8	
TWAMSG03	BC	
TWAMSG04	C0	
TWAN1PTR	20	
TWAOFFST	E7	
TWAPARMC	24	
TWAPSBDL	37	
TWAPSBN	2C	
TWAPSBNM	30	
TWAPSW	C4	
TWARCODE	D6	
TWARSTRT	E6	
TWASCHDC	24	
TWATSBUF	3C	
TWATSLEN	3C	
TWATSQBL	40	28
TWATSQE	40	
TWAWLIST	8	
TWAXCBDL	10	
TWAXCBN1	18	
TWAXCBN2	14	
TWAXCB2	8	
TWAXCB3	C	
TWAXNAME	CE	

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.)

UIB

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLIUIB		
0	(0)	4	UIB		Extended call user interface block
0	(0)	4	UIBPCBAL		PCB address list
4	(4)	2	UIBRCode		DL/I return codes
4	(4)	1	UIBFCTR		Return code
5	(5)	1	UIBDLTR		Additional information
6	(6)	1	(2)		Reserved
8	(8)	4			Length is fullword multiple
			UIBLEN	.... 1...	"*-UIB" length of UIB

Cross Reference

Name	Hex Offset	Hex Value
DLIUIB	0	
UIB	0	
UIBDLTR	5	
UIBFCTR	4	
UIBLEN	8	08
UIBPCBAL	0	
UIBRCODE	4	

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.

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZUIB		
0	(0)	4	UIB		Extended call user interface block
0	(0)	4	UIBPCBAL		PCB address list
4	(4)	2	UIBRCode		DL/I return codes
4	(4)	1	UIBFCTR		Return code
5	(5)	1	UIBDLTR		Additional information
6	(6)	1	(2)		Reserved
8	(8)	4			Length is fullword multiple
			UIBLEN	.... 1...	"*-UIB" length of UIB

System section of the UIB
DL/I online task control blocks

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
8	(8)	4	UIBPST		Task PST address
12	(C)	4	UIBUTCA		A(CICS/VS user TCA)
16	(10)	4	UIBSDIB		System DIB address
20	(14)	4	UIBACTA		A(ACT entry) of program which issued last PCB call

DL/I online task status information

Offsets (Dec)	Offsets (Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
24	(18)	8	UIBPROG		Name of current program
32	(20)	4	UIBSUSP		Max task suspend chain a(user TCA of next task)
36	(24)	4	UIBTSTAT		Task PSB schedule status CMAX = maximum task suspended CONF = intent conflict SUSP SCHD = currently SCHEDULED TERM = normal TERMINATION ABND = abnormal termination
40	(28)	8	UIBPSB		PSBNAME from last PCB call
48	(30)	8	UIBSPSB		Name of PSB currently scheduled

56	(38)	1	UIBTYPSB		Type of PSB ' ' = local PSB '+' = remote PSB '*' = local and remote PSBS
57	(39)	3	UIBLOPSB		Location of PSB 'loc' = on local system 'rem' = on remote system 'XRM' = on local and remote
60	(3C)	1	UIBPRIOR		MAXTASK suspend priority
61	(3D)	3	UIBTSKID		Current CICS/VS task id

DL/I online call status information

64	(40)	8	UIBDPROG		Reserved
72	(48)	4	UIBFUNC		Call function type
76	(4C)	4	UIBIPCBA		Internal A(PCB address list
80	(50)	2	UIBICODE		Internal DL/I return code
82	(52)	2			Reserved
84	(54)	1	UIBFLAG1		UIB flag byte
			UIBREMOT	1...	"X'80'" PSB on remote system indicator
			UIBHLPI	.1..	"X'40'" HLPI command level program
			UIBXRPSB	..1.	"X'20'" Remote with local PSB schedule
			UIBX1	...1	"X'10'" Reserved
			UIBMPS	.... 1...	"X'08'" UIB acquired for MPS task
			UIBTERM	.... .1..	"X'04'" Term call indicator
			UIBDB	.... ..1.	"X'02'" Data base call indicator
			UIBSCHD	1	"X'01'" Schedule call indicator
85	(55)	1	UIBFLAG2		UIB flag byte
			UIBDUMP	1...	"X'80'" Task dump taken
			UIBTERM1	.1..	"X'40'" Abend during buffer purge
			UIBTERM2	..1.	"X'20'" DL/I phase 1 term complete
			UIBTERM3	...1	"X'10'" Reserved
			UIBTERM4	.... 1...	"X'08'" Reserved
			UIBTERM5	.... .1..	"X'04'" Reserved
			UIBTERM6	.... ..1.	"X'02'" Reserved

			UIBTERM7	1	"X'01'" Reserved
86	(56)	1	UIBFLAG3		CMF clock status
			UIBCLK1	1...	"X'80'" CMF clock 1 started
			UIBCLK2	.1..	"X'40'" CMF clock 2 started
			UIBCLK3	..1.	"X'20'" CMF clock 3 started
			UIBCLK4	...1	"X'10'" CMF clock 4 started
			UIBCLK5	.... 1...	"X'08'" CMF clock 5 started
			UIBCLK6	.... .1..	"X'04'" CMF clock 6 started
			UIBCLK7	.... ..1.	"X'02'" CMF clock 7 started
			UIBCLK8	1	"X'01'" Reserved
87	(57)	1	UIBRSTAT		Local and remote status
			UIBXBGUN	1...	"X'80'" XPSB schedule call in progress
			UIBXLOC	.1..	"X'40'" Local PSB scheduled
			UIBXREM	..1.	"X'20'" Remote PSB scheduled
			UIBXSTOR	...1	"X'10'" PCB list storage acquired
			UIBXUNSC	.... 1...	"X'08'" Local PSB unscheduled
			UIBX2	.... .1..	"X'04'" Reserved
			UIBX3	.... ..1.	"X'02'" Reserved
			UIBX4	1	"X'01'" Reserved

DL/I online task work areas

88	(58)	4	UIBMSGPM		Message parameter list
92	(5C)	4	UIBMSGP2		Second message parameter
96	(60)	4	UIBMSGP3		Third message parameter
100	(64)	4	UIBWORK		Work area
104	(68)	4	UIBTRCSV(18		DLZOLT00 register save area
176	(B0)	4	UIBREGSV(18		Register save area
248	(F8)	4			Length is fullword multiple
			UIBSLEN	1111 1...	"*-UIB" length of system UIB

Cross Reference

Name	Hex Offset	Hex Value
DLZUIB	0	
UIB	0	
UIBACTA	14	
UIBCLK1	56	80
UIBCLK2	56	40
UIBCLK3	56	20
UIBCLK4	56	10
UIBCLK5	56	08
UIBCLK6	56	04
UIBCLK7	56	02
UIBCLK8	56	01
UIBDB	54	02
UIBDLTR	5	
UIBDPROG	40	
UIBDUMP	55	80
UIBFCTR	4	
UIBFLAG1	54	
UIBFLAG2	55	
UIBFLAG3	56	
UIBFUNC	48	
UIBHLPI	54	40
UIBICODE	50	
UIBIPCBA	4C	
UIBLEN	8	08
UIBLOPSB	39	
UIBMPS	54	08
UIBMSGPM	58	
UIBMSGP2	5C	
UIBMSGP3	60	
UIBPCBAL	0	
UIBPRIOR	3C	
UIBPROG	18	
UIBPSB	28	
UIBPST	8	
UIBRCODE	4	
UIBREGSV	B0	
UIBREMOT	54	80
UIBRSTAT	57	
UIBSCHD	54	01
UIBSDIB	10	
UIBSLEN	F8	F8
UIBSPSB	30	
UIBSUSP	20	
UIBTERM	54	04
UIBTERM1	55	40
UIBTERM2	55	20
UIBTERM3	55	10
UIBTERM4	55	08
UIBTERM5	55	04
UIBTERM6	55	02
UIBTERM7	55	01
UIBTRCSV	68	
UIBTSKID	3D	
UIBTSTAT	24	
UIBTYPSB	38	
UIBUTCA	C	
UIBWORK	64	
UIBXBGUN	57	80
UIBXLOC	57	40
UIBXREM	57	20
UIBXRPSB	54	20
UIBXSTOR	57	10
UIBXUNSC	57	08
UIBX1	54	10
UIBX2	57	04
UIBX3	57	02
UIBX4	57	01

XCB1 - MPS Batch Partition Communication Area

DSECT Name: DLZXCB1

The DLZXCB1 macro maps the area behind the MPS batch partition XECB which is used to pass data and control information back and forth between the MPS batch partition and the online partition MPS transactions. These online transactions are the Master Partition Controller (MPC) and the Batch Partition Controller (BPC).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DLZXCB1		
0	(0)	4	XCB1ECB		Batch partition ECB
4	(4)	4	XCB1PSB		Pointer to PSB name
8	(8)	4	XCB1PROG		Pointer to program name
12	(C)	4	XCB1CNT		Address of number of parms
			XCB1HIGH	.... 11..	"XCB1CNT" on initial XPOST, address of highest partition addr + 1
16	(10)	4	XCB1PARM(18)		Address of call parameters
88	(58)	4	XCB1SDIB		System DIB address (for HLPI)
92	(5C)	1	XCB1FLAG		Flag byte
			XCB1EOJ	1	"X'01'" EOJ indicator
			XCB1PLI	.... ..1.	"X'02'" Put on if PL/I
			XCB1RST	.... .1..	"X'04'" MPS restart processing on
			XCB1CPRS	.... 1...	"X'08'" Checkpoint/restart call
			XCB1CRP	...1	"X'10'" Continue restart processing even though checkpoint ID verification is not possible
93	(5D)	1	XCB1HLPI		R0 on call to PRH
94	(5E)	1	XCB1PPIK		Batch partition PIK
95	(5F)	1	XCB1RES		Reserved for future use
96	(60)	2	XCB1TSQE		TSQ entry number
98	(62)	4	XCB1CPID		Current VSE checkpoint ID
102	(66)	26	XCB1JBID		Job ID next 4 fields
102	(66)	8	XCB1JBNM		Job name
110	(6E)	2	XCB1PTID		Partition ID
112	(70)	8	XCB1DATE		Job start date
120	(78)	8	XCB1TIME		Job start time

Cross Reference

Name	Hex Offset	Hex Value
DLZXCB1	0	
XCB1CNT	C	
XCB1CPID	62	
XCB1CPRS	5C	08
XCB1CRP	5C	10
XCB1DATE	70	
XCB1ECB	0	
XCB1EOJ	5C	01
XCB1FLAG	5C	
XCB1HIGH	C	0C
XCB1HLPI	5D	
XCB1JBID	66	
XCB1JBNM	66	
XCB1PARM	10	
XCB1PLI	5C	02
XCB1PPIK	5E	
XCB1PROG	8	
XCB1PSB	4	
XCB1PTID	6E	
XCB1RES	5F	
XCB1RST	5C	04
XCB1SDIB	58	
XCB1TIME	78	
XCB1TSQE	60	

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 (DNB).

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	DMBXMPRM		
0	(0)	8	DMBXMSGN		Name of indexed segment
8	(8)	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		Address of the next secondary index suppression routine interface table X'FFFFFFFF' if this is the last table

Cross Reference

Name	Hex Offset	Hex Value
DMBACB	0	00
DMBVSALN	0	00
DMBXMPLN	1C	
DMBXMPRM	0	
DMBXMRES	20	
DMBXMSGN	0	
DMBXMXDN	8	
DMBXMXEP	18	
DMBXMXNM	10	

XWR - Index Work Record

DSECT Name: DLZPRXWR

This DSECT describes an index work record that is created by the partial reorganization utility while performing pointer maitnenance.

Offsets (Dec)	(Hex)	Length	Field/Flag Name	Flag Code (Bit)	Description
0	(0)	0	XWR		
0	(0)	4	XWRSTART		
0	(0)	2	XWRHLL		VLR length control field
2	(2)	2	XWRH00		VLR control binary zeros
4	(4)	4	XWRRMOVE		New RBA of a moved segment
8	(8)	4	XWRRCOMP		Old RBA of a segment for compare
12	(C)	4	XWRFSEQ		Record sequence number for nonunique index
16	(10)	2	XWROACT		Offset in ACT which built this record
18	(12)	1	XWRCTYPE		Record type code
19	(13)	1	XWRGFLAG		Processing option flags
20	(14)	1	XWRCRDB		Data base id of segment to be updated
21	(15)	1	XWRCRDSG		Data set group id of segment to be update
			XWRLFIX	...1 .11.	"*-XWRSTART" length of fixed part of record
22	(16)	Var	XWRCKEY		Key of segment to be updated

Cross Reference

Name	Hex Offset	Hex Value
XWR	0	
XWRCKEY	16	
XWRCRDB	14	
XWRCRDSG	15	
XWRCTYPE	12	
XWRFSEQ	C	
XWRGFLAG	13	
XWRHLL	0	
XWRH00	2	
XWRLFIX	15	16
XWROACT	10	
XWRRCOMP	8	
XWRRMOVE	4	
XWRSTART	0	

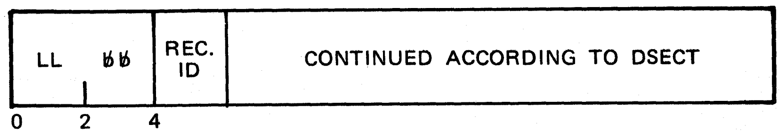
Record Layouts

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

Record/Block Structures

The general structure of DL/I log records, CICS/VS journal records, and CICS/VS journal blocks is shown in Figure 5-7, Figure 5-8 , and Figure 5-9, respectively.



Note: DL/I Log Records are described under "Data Base Log Records."

Figure 5-7. DL/I Log Record

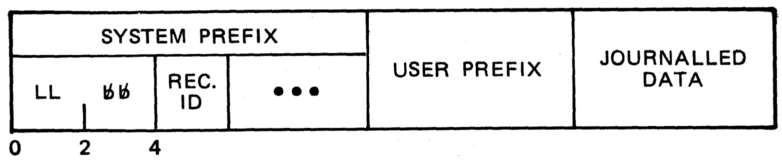


Figure 5-8. CICS/VS Journal Record

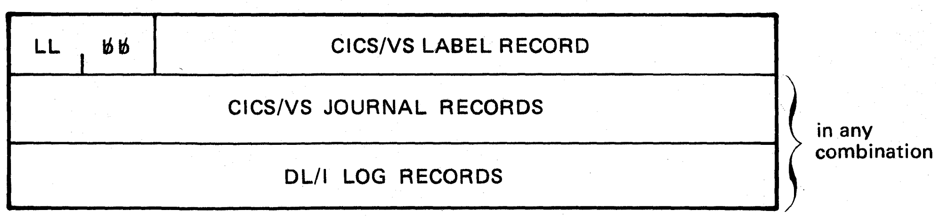


Figure 5-9. Layout of a Journal Block

Accumulation Header Record

This record is used by modules DLZUC350 and DLZURDB0.

Hex	Dec	Name	Length	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	Length	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 - HHMMSS0F
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	Length	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
6	6	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	Length	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	Length	Description
0	0	CHKPLEN	2	Length of log record
2	2	CHKPSPC	2	Blanks/zeros
4	4	CHKPCODE	1	Log record ID
		Flag Name	**Hex Code**	**Meaning**
		CHKPLRID	41	Checkpoint Log record ID
5	5	CHKPPSB	8	Checkpoint PSB name
D	13	CHKPID	8	User checkpoint ID

Hex	Dec	Name	Length	Description
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	Length	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:

1. Dummy checkpoint record does not contain statistics table.
2. 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	Length	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 '.

Hex	Dec	Name	Length	Description
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	Length	Description
0	0	LEFPTR	4	List entry forward pointer (to next data base list entry)
4	4	LENAME	8	DBD name.
C	12	LESLPTR	4	List entry sublist pointer (to first segment list entry)
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	Length	Description
0	0	LEFPTR	4	List entry forward pointer (to next segment list entry)
4	4	LENAME	8	Logical parent segment name.
C	12	LESLPTR	4	List entry sublist pointer (to first secondary list entry)
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
14	20	LEPSDB	4	PSDB for segment entry
18	24	LELSDB	4	LSDB for segment entry

Secondary List Entry

Hex	**Dec**	**Name**	**Length**	**Description**
0	0	LEFPTR	4	List entry forward pointer (to next secondary list entry)
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 of logical parent
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
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

This record is used by modules DLZRDBL0, DLZRDBL1, DLZBACK0, DLZLOGP0, DLZURDB0, DLZUC150, and DLZUC350.

Hex	Dec	Name	Length	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	Byte 1 - PPST ID
				Byte 2 - DL/I Status Code (Right byte)
10	16	DPGMNAME	8	PSB name
18	24	DDBDNAME	8	Data base name from the DMB

Hex	Dec	Name	Length	Description
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')

Hex	Dec	Name	Length	Description
		DDATA	4	New pointer value
			4	Old pointer value

LOGICAL DELETE record (DDATALN is set to H'2')

Hex	Dec	Name	Length	Description
		DDATA	2	Segment code and new delete byte
			2	Segment code and old delete byte

PHYSICAL INSERT record (DDATALN is set to segment length)

Hex	Dec	Name	Length	Description
		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)

Hex	Dec	Name	Length	Description
		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)

Hex	Dec	Name	Length	Description
		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	Length	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	Length	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	Length	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:
		Flag Name	**Hex Code**	**Meaning**
		TABF3N	80	Record to LOGOUT
		TABF3DT	40	Purge date specified
3	3	TABFLAG4	1	Reserved for future use

Hex	Dec	Name	Length	Description
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	Length	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
	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

Hex	Dec	Name	Length	Description
		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	Length	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	Length	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

Hex	Dec	Name	Length	Description
2D	45		Var	Reserved to end of control interval

Dump Header Record

This record is used by modules DLZUDMP0 and DLZURDB0.

Hex	Dec	Name	Length	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 dase 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	Length	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	Length	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	Length	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	Length	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)

Hex	Dec	Name	Length	Description
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	Length	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

Hex	Dec	Name	Length	Description
		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)

Hex	Dec	Name	Length	Description
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 segment
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	Length	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

Hex	Dec	Name	Length	Description
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, LASTMD, ENTLOC	12	Work areas used by INSRCH and LOCSRCH

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	Length	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 record
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	Length	Description
0	0	RGUSEGLV	1	Always X‘00’
1	1	RGUHSDF	1	X‘80’ for first table record 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	Length	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

Hex	Dec	Name	Length	Description
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	Length	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	Length	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	Length	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	Length	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	Length	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	Length	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	Length	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	Length	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	Length	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.

Hex	Dec	Name	Length	Description
		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. Use this field to tell which value ALFLAG2 and ALFLAG3 hold.
		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 (length of field minus 1) of sequence field, if present, or executable length of indexed field if present.
7	7	ALFLAG3	1	Executable length (length of field minus 1) 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	Var	Logical parent's concatenated key (Length of field in ALFLAG3)
		ALPOADDR	4	Logical parent's old address
		ALCDBNAM	8	Data base of logical child
		ALCSEG	1	Segment code of logical child

FOR TYPE 00 AND 01 RECORDS

Hex	Dec	Name	Length	Description
		ALCFL	4	Old value of logical child first or logical child last pointer
		ALT0001	1	X'00' or X'01'
		ALPLSGOF	2	Value of logical parent's LEVSEGOF after BYLCT
		ALCCTR	4	Old value of counter field
		ALPDCB	1	DCB NUMBER FOR LP

(TYPE 01 RECORD ENDS HERE)

Hex	Dec	Name	Length	Description
		ALPSEQA	2	Pointer to secondary list entry within control data set

FOR TYPE 02 RECORDS

Hex	Dec	Name	Length	Description
		ALCOAD	4	Logical child old address
		ALT02	1	X'02'

FOR TYPE 10, 20, AND 30 RECORDS

Hex	Dec	Name	Length	Description
		ALFIL	1	X'FF'
		ALCSEQ		Logical child sequence field (Length of field in ALFLAG2)
		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
		ALT123	1	X'10', or X'20', or X'30'
		ALCDCB	1	DCB number for LC
		ALCSEQA	2	Pointer to secondary list entry within 'control data set'

FOR TYPE 40 RECORDS

Hex	Dec	Name	Length	Description
8	8	AILCOA	4	Logical child old address
C	12	AIDBNAM	8	Index data base name
14	20	AIFLDVAL	Var	Indexed field value (variable length)
		AISC	1	Index segment's segment code
		AISEQ	1	Index segment's sequence code (if second level and present)
		AISEGN	8	Index segment's name (For level 2 index segments)
		AIFLDN	8	Indexed field name (For level 1 index segments)
		AISDBN	8	Indexed segment's data base name
		AISSC	1	Indexed segment's segment code
		AILCNA	4	Logical child new address
		AIDATA	Var	Indexed segment data (for source fields)

FOR TYPE 40 RECORD USED AS SSA AND I/O AREA

Hex	Dec	Name	Length	Description
9	9	AISSFN	8	Index segment name or field name
11	17	AISSAID	3	SSA ID and command code
14	20	AISFLDV	Var	Indexed segment's indexed field value (variable length)
		AISSEQ	Var	Index segment's sequence field value (variable length)

Hex	Dec	Name	Length	Description
		AXSC	1	Segment code of indexed segment
		AXDDIR	3	DDIR address of indexed data base
		AXLCNA	4	Logical child new address
		AXDATA	Var	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	Length	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**	**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**	**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

Hex	Dec	Name	Length	Description
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

Hex	Dec	Name	Length	Description
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

COMMON FIELDS FOR BOTH TYPE 0 AND TYPE 1 RECORDS

Hex	Dec	Name	Length	Description
24	36	CDCB	1	DCB number
25	37			
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	Var	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.

DL/I Message Cross Reference

This table cross-references message numbers (in numeric order) with the module(s) or phase(s) that can cause that message to be issued. In addition, if the message is described in the HIPO diagram in Section 2, the HIPO figure number is also shown. The modules and phases are described in Section 3 of this publication. The messages are described in Chapter 1 of *DL/I DOS/VS Messages and Codes*.

Note: DLZ000I is issued when a module requests a message that does not exist.

Message Number	Module/ Phase	Figure Number
DLZ000I	DLZMMSGT (see note above)	
DLZ001I	DLZBNUC0	2-4.2
DLZ002I	DLZBNUC0	2-4.2
	DLZDXMT0	
	DLZPRABC	
DLZ003I	DLZDDLE0	
DLZ004I	DLZDBH02	
	DLZRDBL0	2-16.7
DLZ005I	DLZDBH02	
DLZ006I	DLZOLI00	2-5.4
DLZ007I	DLZDSEH0	2-39
	DLZDXMT0	
DLZ009I	DLZRRC00	2-3.8
	DLZPRSTC	
DLZ010A	DLZRRC00	
	DLZLOGP0	
	DLZMPI00	2-21.1
DLZ011I	DLZRRC00	2-3.2
	DLZEIPB0	2-45.6
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
	DLZOLI00	2-5.3
DLZ016I	DLZDLOC0	
DLZ017I	DLZRRC00	2-3.7
DLZ018I	DLZRRC00	2-3.7
DLZ019I	DLZRRC00	2-3.3, 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	
	DLZDXMT0	

Message Number	Module/ Phase	Figure Number
DLZ025I	DLZDLOC0	2-14.1
DLZ026I	DLZRRC00	2-3.8
DLZ027I	DLZDLOC0	2-14.1
DLZ028I	DLZDLOC0	2-14.1
DLZ030I	DLZOLI00	2-5.8
	DLZLOGP0	
DLZ031I	DLZOLI00	2-5.1
DLZ032A	DLZOLI00	2-5.4
	DLZRDBL1	
DLZ033I	DLZISC00	2-6.18, 2-6.19
DLZ037I	DLZEIPB0	2-46.4, 2-47.21
	DLZEIPO0	2-48.25, 2-48.29
DLZ038I	DLZEIPB0	2-46.5, 2-46.6
	DLZEIPB1	2-47.22
	DLZMPI00	
	DLZODP	2-6.12, 2-6.13
	DLZRRC00	2-3.4
	DLZSTRB0	2-51.2
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
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	
DLZ059A	DLZMPI00	
DLZ060I	DLZOLI00	2-5.9
DLZ061A	DLZOLI00	2-5.9
DLZ062I	DLZODP	
DLZ063I	DLZODP	
DLZ064I	DLZOLI00	
DLZ065I	DLZODP	
DLZ066I	DLZODP	
DLZ067I	DLZODP	2-6.2
DLZ068I	DLZODP	
DLZ069I	DLZODP	
DLZ070I	DLZODP	2-6.2
DLZ071I	DLZOLI00	2-5.2
DLZ072I	DLZOLI00	2-5.3

Message Number	Module/ Phase	Figure Number
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	2-22.1
DLZ081I	DLZMPI00	2-21.1
DLZ082I	DLZBPC00	2-20.1, 2-20.5
	DLZMPC00	2-19.2, 2-19.4 through 2-19.8
	DLZMPI00	2-21.1, 2-21.3
DLZ083I	DLZMSTR0	2-18
DLZ084I	DLZBPC00	2-20.2, 2-20.4
	DLZMPC00	2-19.4, 2-19.10
	DLZMPI00	2-21.1, 2-21.3
DLZ085I	DLZMPI00	2-21.1
DLZ086I	DLZMPC00	2-19.7
DLZ087A	DLZMPI00	2-21.1
DLZ088I	DLZMPI00	2-21.1
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
DLZ094I	DLZMPC00	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	2-6.21
DLZ106I	DLZQUEF0	
DLZ108I	DLZQUEF0	
DLZ1121	DLZRRC00	2-3.3
DLZ113I	DLZOLI00	2-5.5
DLZ114I	DLZOLI00	
	DLZRRC00	2-3.3
DLZ115I	DLZOLI00	2-5.9
	DLZRRC00	2-3.9, 2-3.3
DLZ116I	DLZRRC00	2-3.3
DLZ117I	DLZRRC00	2-3.9, 2-3.3

Message Number	Module/ Phase	Figure Number
DLZ118I	DLZOLI00	2-5.1
DLZ119I	DLZOLI00	2-5.5
DLZ120I	DLZTRACE	
DLZ121I	DLZMPC00	
DLZ122I	DLZMPC00	2-19.1
DLZ123I	DLZBPC00	
	DLZMPC00	
	DLZMPUR0	
DLZ124I	DLZMPI00	
DLZ125I	DLZMPI00	
DLZ126I	DLZMPI00	
DLZ127I	DLZMPC00	2-19.11
DLZ128I	DLZMPC00	
	DLZMPUR0	2-22.2
DLZ129I	DLZMPI00	
DLZ130I	DLZMPC00	
	DLZMPUR0	2-22.2
DLZ131I	DLZMPI00	
DLZ132I	DLZMPI00	
DLZ133I	DLZMPI00	
DLZ260I	DLZBNUC0	2-4.1
	DLZODP	2-6.6
DLZ261I	DLZBNUC0	2-4.1
	DLZODP	2-6.6
DLZ262I	DLZRRC00	2-3.8
	DLZOLI00	2-5.9
DLZ263I	DLZRRC00	2-3.7
DLZ264I	DLZRDBL1	
DLZ265I	DLZRDBL1	
DLZ266I	DLZRRC00	2-3.7
	DLZOLI00	2-5.3
DLZ267I	DLZQUEF0	2-23
DLZ268I	DLZDDLE0	
DLZ280I	DLZSTTL	2-43
DLZ281I	DLZSTTL	
DLZ282I	DLZSTTL	
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

Message Number	Module/ Phase	Figure Number
	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
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
	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	
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	
	DLZLOGP0	
DLZ320I	DLZURUL0	
	DLZURGU0	
	DLZUDMP0	
DLZ321I	DLZURUL0	

Message Number	Module/ Phase	Figure Number
	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	
	DLZURUL0	2-29
DLZ335I	DLZURDB0	
DLZ336I	DLZURDB0	
DLZ337I	DLZURDB0	
DLZ338I	DLZURDB0	
DLZ339I	DLZURDB0	
	DLZTPRT0	
	DLZURGL0	2-32
	DLZURGU0	2-31
	DLZBACK0	
	DLZLOGP0	
	DLZUCUM0	
	DLZUDMP0	
DLZ340I	DLZURDB0	
	DLZUDMP0	
DLZ341I	DLZURDB0	
DLZ342I	DLZBACK0	
	DLZLPCC0	
	DLZURCC0	2-27.1
	DLZUCCT0	
DLZ343I	DLZURDB0	
DLZ344I	DLZURRL0	2-30
	DLZURUL0	
DLZ345I	DLZURGU0	2-31
	DLZUDMP0	
	DLZURUL0	
DLZ346I	DLZURGU0	
DLZ348I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ349I	DLZURGU0	2-31
DLZ350I	DLZUDMP0	
DLZ351I	DLZURGL0	2-32
DLZ353I	DLZURRL0	
DLZ356I	DLZURRL0	
DLZ357I	DLZURUL0	
	DLZUDMP0	

Message Number	Module/ Phase	Figure Number
DLZ358I	DLZURUL0	
DLZ359I	DLZURGU0	2-31
DLZ360I	DLZUCCT0	
DLZ361I	DLZUCCT0	
DLZ363I	DLZUCCT0	
DLZ364I	DLZUCCT0	
DLZ365I	DLZUCCT0	
DLZ366I	DLZUCCT0	
	DLZURDB0	
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	
DLZ380I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ381I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ382I	DLZURUL0	
DLZ383I	DLZURUL0	
DLZ384I	DLZUCUM0	
	DLZURDB0	
	DLZUDMP0	
	DLZLOGP0	
	DLZBACK0	
	DLZURGL0	2-32
	DLZURGU0	2-31
DLZ385I	DLZUCUM0	
	DLZURGL0	2-32
	DLZURGU0	2-31
	DLZBACK0	
	DLZLOGP0	
	DLZUDMP0	
	DLZURDB0	
DLZ386I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ387I	DLZURGL0	
DLZ388I	DLZURGL0	
DLZ389I	DLZURGL0	2-32

Message Number	Module/ Phase	Figure Number
	DLZURRL0	
DLZ390I	DLZUC150	
	DLZLOGP0	
DLZ391I	DLZUDMP0	
	DLZURDB0	
	DLZURUL0	
	DLZURRL0	
	DLZBACK0	
	DLZLOGP0	
	DLZUC150	
	DLZUC350	
	DLZURPR0	2-35
	DLZURGS0	2-36
	DLZURGU0	
	DLZURG10	2-37
	DLZURGP0	
	DLZUCCT0	
	DLZTPRT0	
DLZ392I	DLZURUL0	
	DLZURGU0	2-31
	DLZURRL0	
DLZ393I	DLZURRL0	
DLZ394I	DLZURRL0	
	DLZURDB0	
DLZ395I	DLZBACK0	
DLZ396I	DLZRDBC0	
	DLZBACK0	
DLZ397I	DLZRDBC0	
	DLZBACK0	
DLZ398I	DLZRDBC0	
	DLZBACK0	
DLZ399I	DLZRDBC0	
	DLZBACK0	
DLZ400I	DLZURGU0	2-31
DLZ401I	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	
	DLZURCC0	
DLZ402I	DLZBACK0	
	DLZURDB0	
	DLZUC150	
DLZ404I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
	DLZUDMP0	
DLZ405I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ406I	DLZBACK0	

Message Number	Module/ Phase	Figure Number
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ407I	DLZLPCC0	
	DLZTPRT0	
	DLZURCC0	
	DLZBACK0	
DLZ408I	DLZBACK0	
DLZ409I	DLZLPCC0	
DLZ410I	DLZLPCC0	
DLZ411I	DLZLPCC0	
DLZ412I	DLZLPCC0	
DLZ413I	DLZLPCC0	
DLZ414I	DLZLPCC0	
	DLZURCC0	
	DLZTPRT0	
DLZ415I	DLZLPCC0	
	DLZURCC0	
DLZ416I	DLZLOGP0	2-40.1
	DLZLPCC0	
DLZ417I	DLZLOGP0	
DLZ418I	DLZLOGP0	
DLZ419I	DLZLOGP0	
	DLZUDMP0	
	DLZURRL0	
	DLZURUL0	
DLZ420I	DLZLOGP0	
DLZ421I	DLZLOGP0	
DLZ422I	DLZLOGP0	
	DLZUDMP0	
	DLZURRL0	
	DLZURUL0	
DLZ423I	DLZLOGP0	
DLZ424I	DLZLOGP0	
DLZ425I	DLZLOGP0	
DLZ426I	DLZLPCC0	
DLZ427I	DLZLOGP0	
DLZ428I	DLZLOGP0	
DLZ429I	DLZLOGP0	
DLZ430I	DLZLPCC0	
DLZ432I	DLZLPCC0	
DLZ433I	DLZLPCC0	
DLZ434I	DLZLPCC0	
DLZ435I	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	
	DLZUDMP0	
	DLZURCC0	
DLZ436I	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	

Message Number	Module/ Phase	Figure Number
	DLZURCC0	
DLZ437I	DLZLPCC0	
	DLZUCCT0	
	DLZURCC0	
DLZ440I	DLZTPRT0	
DLZ442I	DLZTPRT0	
DLZ443I	DLZTPRT0	
DLZ445I	DLZTPRT0	
DLZ446I	DLZTPRT0	
DLZ447I	DLZTPRT0	
DLZ448I	DLZTPRT0	
DLZ449I	DLZTPRT0	
DLZ450I	DLZTPRT0	
DLZ451I	DLZTPRT0	
DLZ452I	DLZTPRT0	
DLZ453I	DLZTPRT0	
DLZ454I	DLZTPRT0	
DLZ476I	DLZDLA00	
	DLZISC00	2-6.19
DLZ500I	DLZEXDF	
	DLZEXDFP	2-34.1
DLZ501I	DLZEXDF	
	DLZEXDFP	2-34.2
DLZ502I	DLZEXDF	
	DLZEXDFP	2-34.1
DLZ503I	DLZEXDF	
	DLZEXDFP	2-34.2
DLZ504I	DLZEXDF	
	DLZEXDFP	2-34.12
DLZ505I	DLZEXDF	
	DLZEXDFP	2-34.1
DLZ506I	DLZEXDF	
	DLZEXDFP	2-34.12
DLZ507I	DLZEXDF	
	DLZEXDFP	2-34.3
DLZ570I	DLZDLBL3	
	DLZUACB0	2-33.18
DLZ571I	DLZUACB0	2-33
DLZ572I	DLZDLBL0	
	DLZDLBL1	
DLZ573I	DLZDLBL0	
	DLZDLBL1	
DLZ574I	DLZUACB0	2-33
DLZ575I	DLZDLBD	
	DLZDLBP	
	DLZDLBPP	2-33.4, 2-33.8
	DLZDLBDP	2-33.28, 2-33.21
	DLZEXDF	
	DLZEXDFP	2-34.7
DLZ576I	DLZDLBDP	2-33.21, 2-33.28
	DLZDLBPP	2-33.4, 2-33.8

Message Number	Module/ Phase	Figure Number
	DLZEXDFP	2-34.1, 2-34.7
DLZ577I	DLZDLBPP	2-33.4
	DLZEXDFP	2-34.1
DLZ578I	DLZDLBD	
	DLZDLBP	
	DLZDLBDP	2-33.21, 2-33.29
	DLZDLBL3	2-33.14
	DLZDLPP	
	DLZEXDF	
	DLZEXDFP	2-34, 2-34.6, 2-34.8
	DLZUACB0	2-33
DLZ583I	DLZUACB0	
DLZ584I	DLZUACB0	
DLZ585I	DLZUACB0	
DLZ587I	DLZUACB0	2-33
DLZ588I	DLZUACB0	2-33
DLZ589I	DLZUACB0	2-33
DLZ600I	DLZPRCT2	
DLZ602I	DLZPRPAR	2-44.8
DLZ603I	DLZPRPAR	2-44.8
DLZ604I	DLZPRPAR	2-44.8
	DLZPRDBD	2-44.4
DLZ605I	DLZPRPAR	2-44.8
DLZ606I	DLZPRPAR	2-44.8
DLZ608I	DLZPRPAR	2-44.8
DLZ609I	DLZPRPAR	2-44.8
DLZ610I	DLZPRPAR	2-44.8
DLZ611I	DLZPRPAR	2-44.8
DLZ612I	DLZPRPAR	2-44.8
	DLZPRDBD	2-44.4
DLZ613I	DLZPRDBD	2-44.4
	DLZPRPAR	2-44.8
DLZ614I	DLZPRDBD	2-44.4
	DLZPRPAR	2-44.8
DLZ615I	DLZPRDBD	2-44.4
	DLZPRABC	2-44.2
	DLZPRDLI	2-44.14
	DLZPRWFM	2-44.13
DLZ616I	DLZPRDBD	2-44.4
DLZ617I	DLZPRDBD	2-44.4
DLZ618I	DLZPRDBD	2-44.4
DLZ623I	DLZPRABC	2-44-2
DLZ626I	DLZPRCT2	
DLZ627I	DLZPRPSB	2-44.5
DLZ633I	DLZPRPAR	
DLZ634I	DLZPRCT2	2-44-7
DLZ635I	DLZPRCT1	2-44.1
	DLZPRERR	
DLZ636I	DLZPRCT2	2-44.7
	DLZPRDLI	2-44.14
	DLZPRERR	

Message Number	Module/ Phase	Figure Number
DLZ639I	DLZPRCT2	2-44.7
	DLZPRCT1	
	DLZPRUPD	
DLZ641I	DLZPRURC	
	DLZPRDBD	
	DLZPRERR	
DLZ642I	DLZPRSTW	2-44.15
DLZ643I	DLZPRURC	2-44.12
DLZ644I	DLZPRURC	2-44.12
DLZ645I	DLZPRDBD	2-44.4
	DLZPRURC	
DLZ646I	DLZPRURC	2-44.12
DLZ647I	DLZPRSTC	2-44-11
DLZ648I	DLZPRSTC	2-44.11
DLZ649I	DLZPRSTC	2-44.11
DLZ650I	DLZPRUPD	2-44.10
DLZ651I	DLZPRDLI	2-44.14
DLZ652I	DLZPRDLI	
DLZ653I	DLZPRURC	2-44.12
	DLZPRSCC	2-44.9
	DLZPRUPD	2-44.10
	DLZPRDLI	
DLZ655I	DLZPRDLI	2-44.14
DLZ659I	DLZPRUPD	2-44.10
DLZ772I	DLZDXMT0	
DLZ796I	DLZDLD00	
DLZ797I	DLZDDLE0	
DLZ798I	DLZDLRG0	
	DLZDLRD0	
	DLZDLRC0	
DLZ799I	DLZDLD00	
	DLZCPY10	
DLZ800I	DLZDLRF0	
DLZ801I	DLZDLRB0	
	DLZDLRF0	
DLZ802I	DLZDLD00	
	DLZDHDS0	
DLZ803I	DLZDLD00	
DLZ804I	DLZDLD00	
DLZ805I	DLZDLD00	
DLZ806I	DLZDLD00	
	DLZCPY10	
	DLZDHDS0	
DLZ807I	DLZDLD00	
DLZ808I	DLZDLD00	
DLZ809I	DLZDLD00	
DLZ830I	DLZDHDS0	
DLZ831I	DLZDHDS0	2-13.5
DLZ832I	DLZDHDS0	
DLZ841I	DLZDBH00	
DLZ844I	DLZDBH02	

Message Number	Module/ Phase	Figure Number
DLZ845I	DLZDBH00	
DLZ847I	DLZDBH00	
DLZ848I	DLZDBH00	
DLZ850I	DLZDDLE0	
DLZ855I	DLZDDLE0	
DLZ860I	DLZDDLE0	
	DLZDXMT0	
DLZ861I	DLZDDLE0	
DLZ862I	DLZDDLE0	
DLZ863I	DLZDDLE0	
DLZ864I	DLZDDLE0	
DLZ868I	DLZDXMT0	
DLZ869I	DLZDXMT0	
DLZ870I	DLZDXMT0	
DLZ888I	DLZBACK0	
DLZ890I	DLZBACK0	
DLZ894I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ900I	DLZDLBL1	
DLZ901I	DLZDLBL2	
DLZ902I	DLZDLBL2	
DLZ903I	DLZDLBL2	
DLZ904I	DLZDLBL0	
DLZ905I	DLZDLBL0	2-33.15
	DLZDLBL1	
	DLZDLBL2	
	DLZDLBL3	
	DLZDLBLC	
	DLZUACB0	2-33
	DLZUAMB0	2-33.18, 2-33.19
	DLZDPSB0	2-33.20
DLZ906I	DLZDLBL0	
DLZ907I	DLZDLBL3	
DLZ908I	DLZDLBL3	
DLZ909I	DLZDLBL2	2-33.13
DLZ910I	DLZDLBL0	
	DLZDLBL1	
DLZ911I	DLZDLBL2	
DLZ912I	DLZDLBL1	2-33.11
DLZ913I	DLZDLBL1	
DLZ914I	DLZDLBL2	
DLZ915I	DLZDLBL1	
DLZ916I	DLZDLBL1	
DLZ917I	DLZDLBL1	
DLZ918I	DLZDLBL2	
DLZ919I	DLZDLBL2	
DLZ920I	DLZDLBL1	
DLZ921I	DLZDLBL0	
DLZ922I	DLZDLBL1	

Message Number	Module/ Phase	Figure Number
DLZ923I	DLZDLBL1	
DLZ924I	DLZDLBL1	
DLZ925I	DLZDLBL1	
DLZ926I	DLZDLBD	
	DLZDLBDP	2-33.22, 2-33.23, 2-33.25, 2-33.30
	DLZDLBP	
	DLZDLBPP	2-33.9
	DLZDLBL0	
	DLZDLBL1	
	DLZDLBL2	
	DLZDLBL3	
	DLZUAMB0	2-33.18, 2-33.19
DLZ927I	DLZDLBL1	
DLZ928I	DLZDLBL1	
DLZ929I	DLZDLBL0	
	DLZDLBL1	
DLZ930I	DLZDLBL2	
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	2-36
DLZ953I	DLZURGP0	
DLZ954I	DLZURPR0	2-35
	DLZURGS0	2-36
	DLZURG10	2-37
	DLZURGP0	
DLZ955I	DLZURG10	2-37.2, 2-37.4
	DLZURGP0	
DLZ956I	DLZURPR0	2-35
	DLZURGS0	2-36
	DLZURGP0	
DLZ957I	DLZURGS0	2-36
	DLZURG10	2-37
DLZ958I	DLZURGS0	2-36

Message Number	Module/ Phase	Figure Number
	DLZURGP0	
DLZ959I	DLZURGS0	
	DLZURGP0	
DLZ960I	DLZURGP0	
DLZ961I	DLZURPR0	
	DLZURGS0	
	DLZURG10	
DLZ962I	DLZURPR0	2-35
DLZ963I	DLZURPR0	2-35
DLZ964I	DLZURPR0	2-35
DLZ965I	DLZURPR0	2-35
DLZ966I	DLZURPR0	2-35
	DLZURGS0	2-36
	DLZURG10	2-37
	DLZURGP0	
DLZ967I	DLZURGS0	2-36
DLZ968I	DLZURGS0	
	DLZURPR0	
	DLZURG10	2-37
	DLZURGP0	
DLZ969I	DLZURGS0	2-36
DLZ970I	DLZURGS0	2-36
DLZ971I	DLZURGS0	2-36
DLZ972I	DLZURGS0	
DLZ973I	DLZURGS0	
DLZ974I	DLZURGS0	
DLZ975I	DLZURGS0	2-36
DLZ976I	DLZURPR0	2-35
DLZ977I	DLZURG10	2-37.2
DLZ978I	DLZURG10	2-37.2
DLZ979I	DLZURG10	2-37.2
DLZ980I	DLZURG10	2-37.2, 2-37.4
DLZ981I	DLZURG10	2-37.4
DLZ982I	DLZURG10	2-37
	DLZURGP0	
DLZ983I	DLZURGP0	
DLZ984I	DLZURPR0	2-35
	DLZURGP0	
	DLZURGS0	2-36
	DLZURG10	2-37
DLZ985I	DLZURPR0	2-35
DLZ989I	DLZURG10	2-37.2
DLZ990I	DLZURGS0	
	DLZURGP0	
	DLZURG10	
	DLZURPR0	
DLZ991I	DLZURPR0	

DL/I Status Codes Cross Reference

This table cross-references DL/I status codes (in alphabetic order) with the module(s) or phase(s) that can cause that status code to be set. The modules and phases 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	DLZDLD00
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

Status Code	Module
TO	DLZEIPB1, DLZEIPO0
TP	DLZEIPB1, DLZEIPO0
V1	DLZDLA00
V2	DLZEIPB1, DLZEIPO0
V3	DLZEIPB1, DLZEIPO0
V4	DLZEIPB1, DLZEIPO0
V5	DLZEIPB1, DLZEIPO0
V8	DLZEIPB1, DLZEIPO0
XD	DLZDLA01
XH	DLZDLA00
XR	DLZMPI00

Section 7. Appendixes

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 on page 7-3). 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.

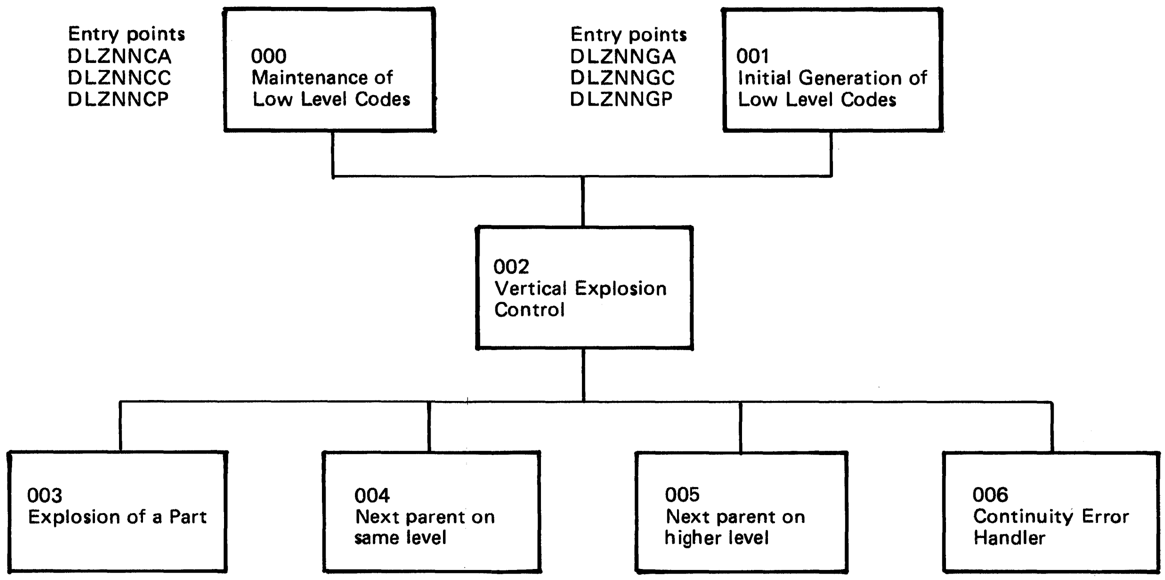


Figure 7-1. Structure of LLC/CC in DL/I

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.

CHART: DLZNN
PAGE 1 OF 2

Input	Processing	Output
	This module is entered from an application program via CALL.	
INPUT: PARTS PCB, CTL.-PCB, PAR. PART, COMP. PART	[01] Obtain and adjust input data.	
PARTS DB	[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.	LLC: INITIAL, ACTUAL
	[04] Insert root segment LLCTL with key = X to start the control data base.	CONTROL DB
	[05] Insert dependent segments into the control data base for parent part and for component part.	CONTROL DB

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

Notes	Routine	Label	Ref	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					

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

CHART: DLZNN
PAGE 2 OF 2

Input

Processing

[06] Replace the old LLC in the component part by the actual LLC.

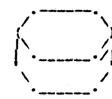
[07] Execute LLC/CC in DL/I.

002
VERTICAL EXPLOSION CONTROL
3.1.3

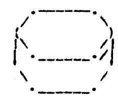
[08] Remove all control information from the control data base by deleting the root segment LLCTL with key = X.

[09] Set up return information. Return to calling application program.

Output



PARTS DB



CONTROL DB

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

Input

INPUT
- PARTS PCB
- CTL-PCB
- PART

PARTS DB

PARTS DB

Processing

This module is entered from an application program via CALL.

[01] Obtain and adjust input data.

[02] Read part. If not found, go to step 9.

[03] Test the LLC of the part. If it is not 0, go to step 8. Else set initial and actual LLC to 0.

[04] The part is tested whether it has component parts. If no components are found, control is passed to step 9.

[05] Insert root segment LLCTL with key = X to start the control data base.

[06] Insert a segment PARTBEXP with key composed of packed zeros, i.e., actual LLC plus part key.

Output

LLC
- INITIAL
- ACTUAL

CONTROL DB

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

CHART: DLZNN
PAGE 2 OF 2

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

HIPOMAT 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.		DLZNNEC					

001 - INITIAL GENERATION OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.2-02

CHART: DLZNN
PAGE 1 OF 3

Input	Processing	Output
LLC: ACTUAL, INITIAL; CONTROL DB	[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.	

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-01

Notes	Routine	Label	Ref	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.							

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-01

CHART: DLZNN
PAGE 2 OF 3

Input	Processing	Output
PARTS DB	[04] If a segment is found, read the root segment of the part.	PART: ACTUAL
	[05] Increment the actual LLC by 1 to post into the components of the new part.	LLC: ACTUAL
	[06] Perform continuity check and go to step 9. If the check fails, restore old LLC status in 006.	CONTROL DB
	006 ERROR RECOVERY HANDLER 3.1.3.4	OUTPUT DATA: LOOP KEY, ERROR INFO
	[07] Upon return from 006, control is passed to step 12.	

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-02

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	

Notes	Routine	Label	Ref

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-02

CHART: ELZNN
PAGE 3 OF 3

Input

Processing

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

Output

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

CHART: DLZNN
PAGE 1 OF 2

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

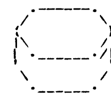
Notes	Routine	Label	Ref
01 If the no-component-found LECBSNOC condition was raised when retrieving the first segment, a switch indicates to 002 that the actual part does not have any component parts at all and another part has to be selected for explosion.		LFCBSNOC	

Notes	Routine	Label	Ref

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-01

CHART: ELZNN
PAGE 2 OF 2

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: LL2nn
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	Notes	Routine	Label	Ref
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.							
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.							

004 - NEXT PARENT ON SAME LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.2-01

CHART: DLZNN
PAGE 1 OF 1

Input

LLC
ACTUAL

CONTROL DB

LLC
ACTUAL

CONTROL DB

PART KEY

005 - NEXT PARENT ON HIGHER LEVEL

Processing

01 Decrement the actual LLC by 1.

02 Delete the first segment PARTBEXP identified by a key composed of the actual LLC and of hex zeros.

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.

Output

LLC
ACTUAL

PART KEY

HIPOMAT 1.1 Diagram - 3.1.3.3-01

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.			

Notes	Routine	Label	Ref

005 - NEXT PARENT ON HIGHER LEVEL

HIPOMAT 1.1 Diagram - 3.1.3.3-01

CHART: DLZNN
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	DZLALPHA	
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
B = binary
C = character
R = reference
S = set
U = reference/set

7-2 1/5

Figure 7-2 (Part 1 of 5). DBDGEN MACRO-GLOBAL Symbol Cross Reference

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 (Part 2 of 5). DBDGEN MACRO-GLOBAL Symbol Cross Reference

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 (Part 3 of 5). DBDGEN MACRO-GLOBAL Symbol Cross Reference

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 (Part 4 of 5). DBDGEN MACRO-GLOBAL Symbol Cross Reference

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 (Part 5 of 5). DBDGEN MACRO-GLOBAL Symbol Cross Reference

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 so me entries within the DTFSD (HSAM) and ACB-extension. The capacities are returned using global arithmetic variables (GBLA). Input values are:

DEVICE	2314, 3330, 3340, 3350, 3375, FBA
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
Bit 2	TWIN
Bit 3	TWINBWD
Bit 4	PARNT
Bit 5	LTWIN
Bit 6	LTWINBWD
Bit 7	LPARNT
Bit 8	NOTWIN

If TWINBWD and/or LTWINBWD is specified, Bit 2 and/or Bit 5 is set on, in addition to Bit 3 and/or Bit 6, respectively.

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	Length	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	Length	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	Length	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	Length	Description
0	0	PREDD1	8	Input or prime filename
8	8	PREDEV1	4	Device type
C	12	PREID	1	Data set group ID

Hex	Dec	Name	Length	Description
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	Length	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

Hex	Dec	Name	Length	Description
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

Hex	Dec	Name	Length	Description
16	22	SEGFLG3	1	

EQU	Meaning
X'08'	Parent has backward pointers to this segment

Hex	Dec	Name	Length	Description
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

Hex	Dec	Name	Length	Description
2D	45	SEGPACRT	3	Address of compression table

6. FLDTAB Layout

Hex	Dec	Name	Length	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
X‘01’	Hexadecimal field
X‘02’	Packed field
X‘03’	Character field
X‘04’	Floating point field

Hex	Dec	Name	Length	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	Length	Description
0	0	EXTDBNM	8	Externally referenced data base name
8	8	EXTRSVD	4	Reserved

8. LCHDTAB Layout

Hex	Dec	Name	Length	Description
0	0	LCHSEGNM	8	Segment name
8	8	LCHCODE	1	

Bit	Meaning
0=0	LCHEDBD address is an EXTDBD entry
0=1	LCHEDRD address is a SEGTAB entry
1-7	Reserved

Hex	Dec	Name	Length	Description
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

Hex	Dec	Name	Length	Description
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	Length	Description
0	0	DBDSORNM	8	Source segment name
8	8	DBDSSFLG	1	Source segment flag - reserved
9	9	DBDSSDB0	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	Length	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
   PL/I dope vectors precede PCB if LANG=PL/I

Hex	Dec	Length	Description
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	Length	Description
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	Length	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	Length	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	Length	Description
0	0	1	Number of FSBs for segment
1	1	3	Address of first FSB for segment

Appendix D: DL/I Macros

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.

Registers 0 and 1 are destroyed unless specified for the length register. All other registers are unchanged.

DLZBLKLD

This macro is used by some DOS/VS DL/I utility programs to request the initialization module to load all control blocks needed to process a specified utility

PSB. A utility PSB is built by the application control block creation and maintenance utility for every user DBD except a primary HIDAM index, logical, or HSAM.

The utilities which use this special function have 'ULU' in the first three bytes of the parameter card. When batch initialization determines (by utility name - either DLZURPR0, DLZURGS0, or DLZURGP0) that the DLZBLKLD macro will be used, it does not load any control blocks. The action modules and PST and SCD are loaded, however. When the utility first receives control, register 1 contains the address of the PST.

Operand

When the utility reaches the point where blocks are needed, the DLZBLKLD macro is executed:

```
                [(reg)]
DLZBLKLD DMB=   [label]
```

The DMB operand indicates the address of the 8-byte DMB name for which blocks are required. Either the register number (reg) or the label of the field may be specified to indicate the address. If this operand is omitted, register 1 is assumed to contain the address of the DMB name.

The expansion replaces the ending 'D' of the DMB name with a 'U'. A 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

After execution of this DLZBLKLD macro, register 15 contains a return code:

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.

Operands

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 concatenated 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).

DLZTSQE

This macro maps the entries in the CICS/VS temporary storage queue used by MPS Restart which contain combined checkpoint IDs for MPS batch jobs.

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
DLZREC0
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
DLZUMSG	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.

DL/I Documentation Aid Macros

DLZDLBP

Creates (PREP) the DL/I PSB Documentation Aid access module.

DLZDLBD

Creates (PREP) the DL/I DBD Documentation Aid access module.

DLZDATAB

Acquires public DBSPACE name DLIDBDDB and creates the following DL/I DA SQL/DS tables for storing the Data Base Description (DBD) data:

DBDBASICDATA
DBDACCESSDATA
DBDSEGMENTDATA
DBDLCHILDDATA
DBDFIELDDATA

It also acquires public DBSPACE name DLIPSBDB and creates the following DL/I DA SQL/DS tables for storing the Program Specification Blocks (PSB) data:

PSBBASICDATA
PSBPCBDATA
PSBSEGMENTDATA
PSBFIELDDATA

DLZDANDX

Creates the DL/I Documentation Aid SQL/DS table indexes.

DLZDARTN

Dataloads the DL/I Documentation Aid ISQL Routines in the Routine table.

DLZEXDF

Creates (PREP) the Extract Defines access module.

Index

D

E

F

G

H

I

DL/I DOS/VS Logic Manual,
Volume 1
LY12-5016-7

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? ____________________

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
Dept 812BP
1133 Westchester Avenue
White Plains, NY 10604, USA

Fold | Fold

If you would like a reply, *please print:*

Your Name ______________________

Company Name ______________ *Department* __________

Street Address ______________________

City ______________________

State ______________ *Zip Code* __________

IBM Branch Office serving you ______________________

IBM®

LY12-5016-7

DOS/VS Logic Manual, Volume 1 (File No. S370/4300-50) Printed in U.S.A. LY12-5016-7

LY12-5016-07

IBM