LY12-5016-5

Program Product

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

OBSOLETE

Program Number 5746-XX1

Sixth Edition (June 1979)

This edition applies to Version 1, Release 5 (Version 1.5) of IBM System/370 Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1. It supersedes LY12-5016-4 with Technical Newsletter LN24-5614.

This edition, LY12-5016-5, is a major revision of LY12-5016-4.

Summary of Amendments

For a list of changes, see page 3. Changes and additions are indicated by a vertical line to the left of the change.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Programming Publications, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatsoever. You may, of course, continue to use the information you supply.

DL/I Version 1.5

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

Field Level Sensitivity

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

Extended Logical Relationships

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

Unique Segment Support

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

Selective Log Print

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

DL/I FBA Device Support ICR

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

FBA Device Support

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

DL/I Version 1.4

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

RPG II Support

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

Prefix Resolution Improvement

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

Extended DL/I Call Interface

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

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

Intersystem Communication

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

High Level Language Debugging for PL/I

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

Performance Improvements

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

DL/I Version 1.3

This version of DL/I provides system changes and functional enhancements such as: checkpoint capability with the new DL/I call function CHKP; program isolation capability for online and MPS users as an optional replacement for intent scheduling; the distributed free space feature to improve performance of data bases with high insert activity; a log print utility to enable the printing of log files; a disk logging facility for the disk-only user; support for the IBM 3350 direct access storage; and other servicability, performance, and functional capabilities.

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.

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 of this manual.

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 L-LC/CC in DL/I, DBD generation, PSB generation and DL/I macros.

An index is also included.

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

Related Publications

DL/I DOS/VS General Information Manual, GH20-1246
DL/I DOS/VS Application Program Reference Manual, SH12-5411
DL/I DOS/VS Utilities and Guide for the System Programmer, SH12-5412
DL/I DOS/VS System Application/Design Guide, SH12-5413

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

For DOS/VS messages and return codes:

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

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

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

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

Section I summarizes and describes the following:

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

DL/I BATCH SYSTEM

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

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

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

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

- Region control
- Application program control
- Language interface
- Program request handler.

Region control is responsible for a general group of housekeeping functions common to various optional processing modes of the DL/I DOS/VS partition (also called a region). These functions are:

- Initial interface with DOS/VS job management
- Analysis and validity checking of DL/I parameter information
- Loading the batch nucleus.

Application program control is entered from region control and performs the following functions:

- 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.

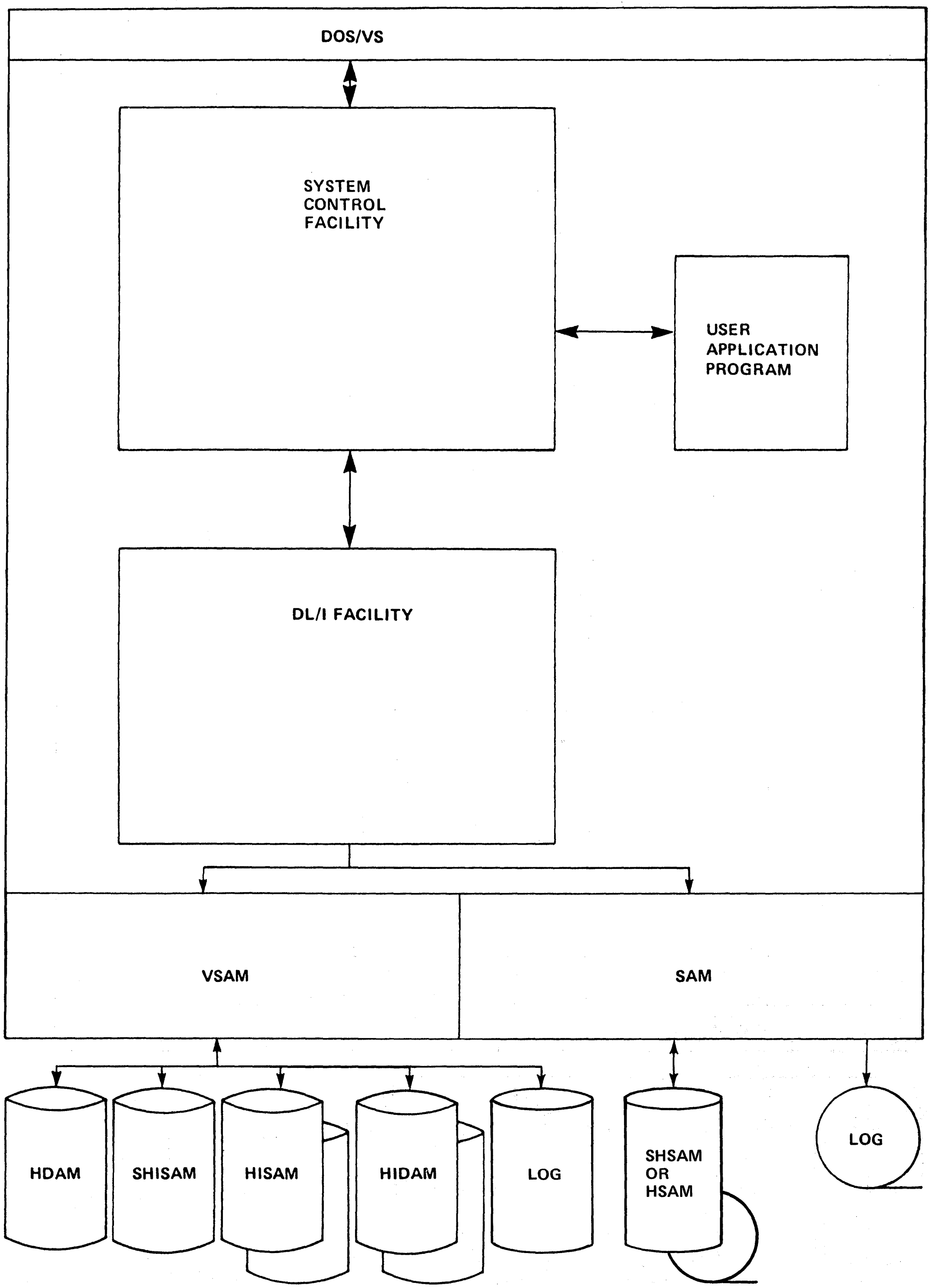


Figure 1-1. Elements of a DL/I DOS/VS Batch Partition

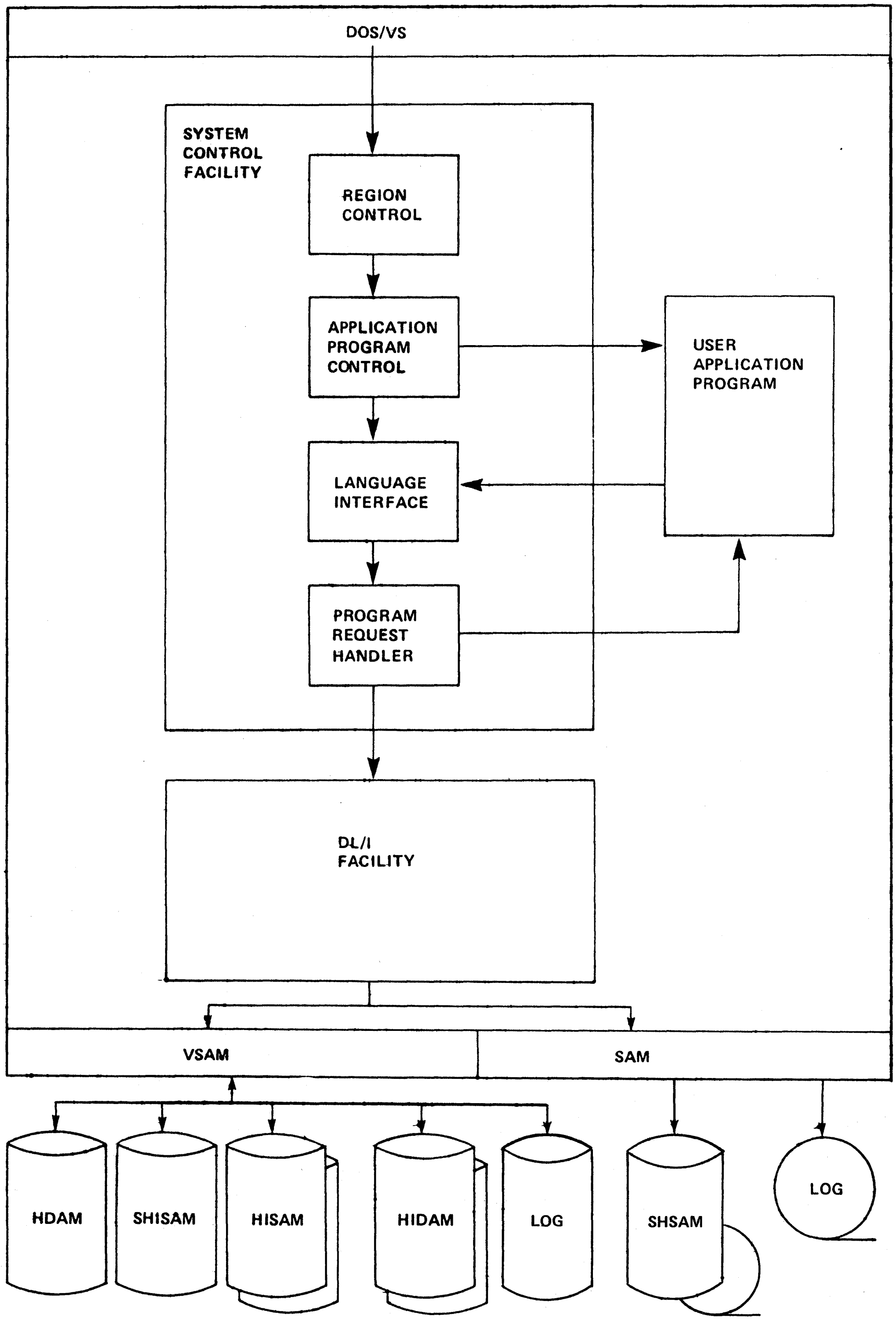


Figure 1-2. System Control Facility Relationships

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

The DL/I system operating in a teleprocessing environment under CICS/VS contains all the functional parts listed for the batch system, plus a set of service routines called the DL/I online processor. These routines establish a connection between DL/I and the CICS/VS-DL/I interface.

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

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

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

DL/I FACILITY MODULES

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

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

- Open/Close Module
  - Open DL/I data bases
  - Close DL/I data bases
  - Interface with data base logger to write data set open record to log file
- Delete/Replace Module
  - Delete segment of DL/I data base in conjunction with buffer handler
  - Replace segment of a DL/I data base in conjunction with 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 Simple HSAM and 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 Simple HSAM and 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 Simple HISAM and HISAM 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.
- FLS Copy Module
  - Provide user view/physical view conversion for field level sensitivity.

See Section 3 for a detailed description of the modules.

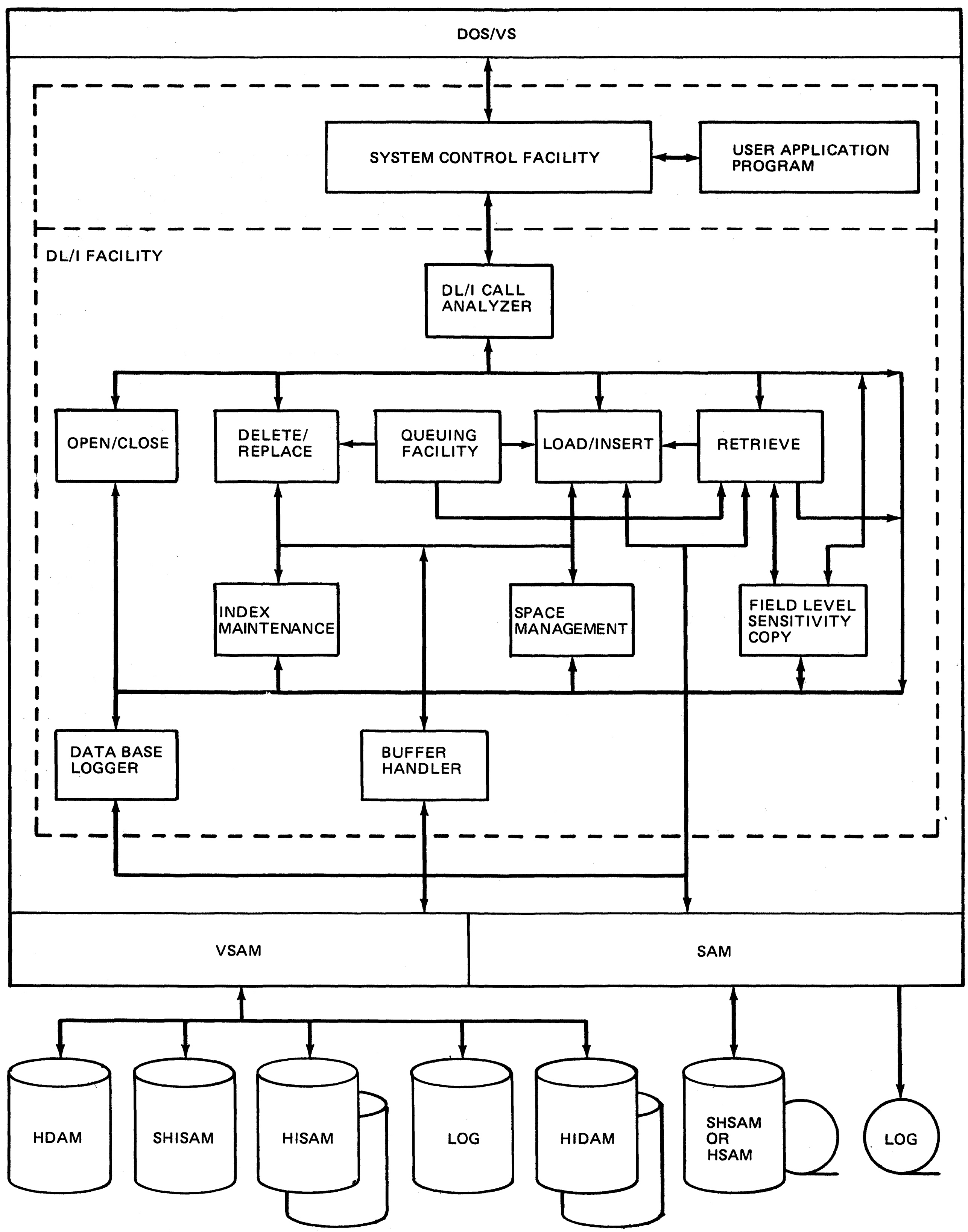


Figure 1-3. DL/I Facility Relationships

MULTIPLE PARTITION SUPPORT (MPS)

DL/I has the capability to enable application programs executing in different partitions to access the same data base concurrently. This capability, multiple partition support (MPS), permits, for example, online applications to issue inquiries to a data base while a batch program updates it.

In addition, MPS enables multiple batch and online application programs to access a data base concurrently instead of serially. 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 requirments 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.

SECTION 2: METHOD OF OPERATION

This section contains HIPO (Hierarchy, plus Input, Process, Output) diagrams.

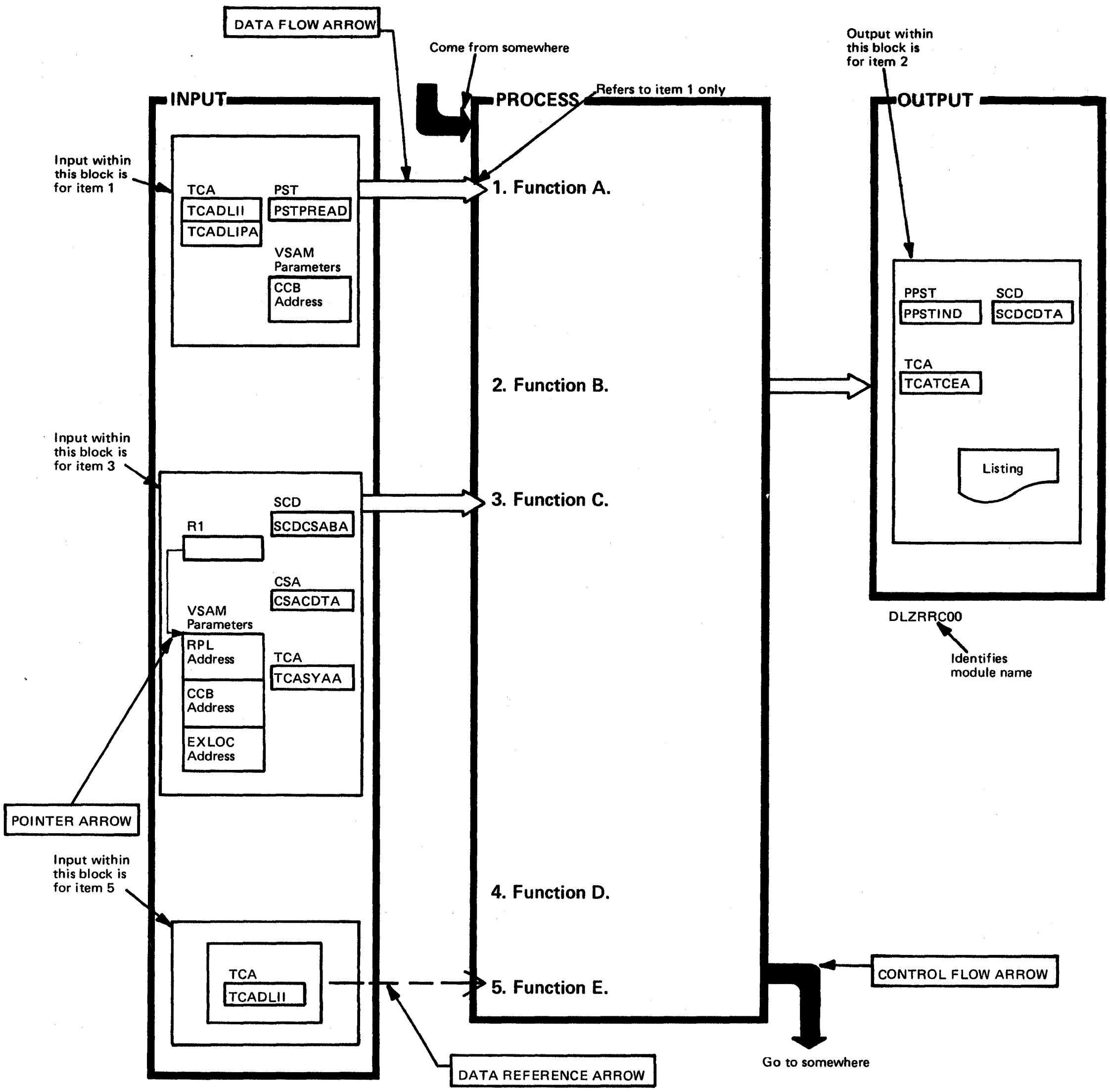
The three areas of each HIPO diagram are, from left to right, the input area, process area, and output area. Read the diagrams beginning with the process area. This describes a function that is performed. Arrows leading from the input area show what, if any, input is used to perform that function. Arrows leading to the output area show what output, if any, is produced.

At the bottom of each HIPO diagram is an area called "extended descriptions". This area contains comments not included in the process area of the diagram. For most items in the process area, extended description items with the same numbers give details that cannot be easily shown in diagram form or in the space allowed.

Various forms of arrows represent different usage conventions. Also, items are often boxed in to show that they are related to the same function. Figure 2-1 shows the conventions used in the HIPO diagrams.

Figure 2-2 is a visual table of contents with figure numbers. The figure numbers refer to the HIPO diagrams.

Figure 2-1. Guide to Reading Method of Operation Diagrams



Extended Description	Module	Label
1. More about function A.		
2. More about function B.		

Extended Description	Module	Label

Figure 2-2. Visual Table of Contents for DL/I DOS/VS HIPO Charts

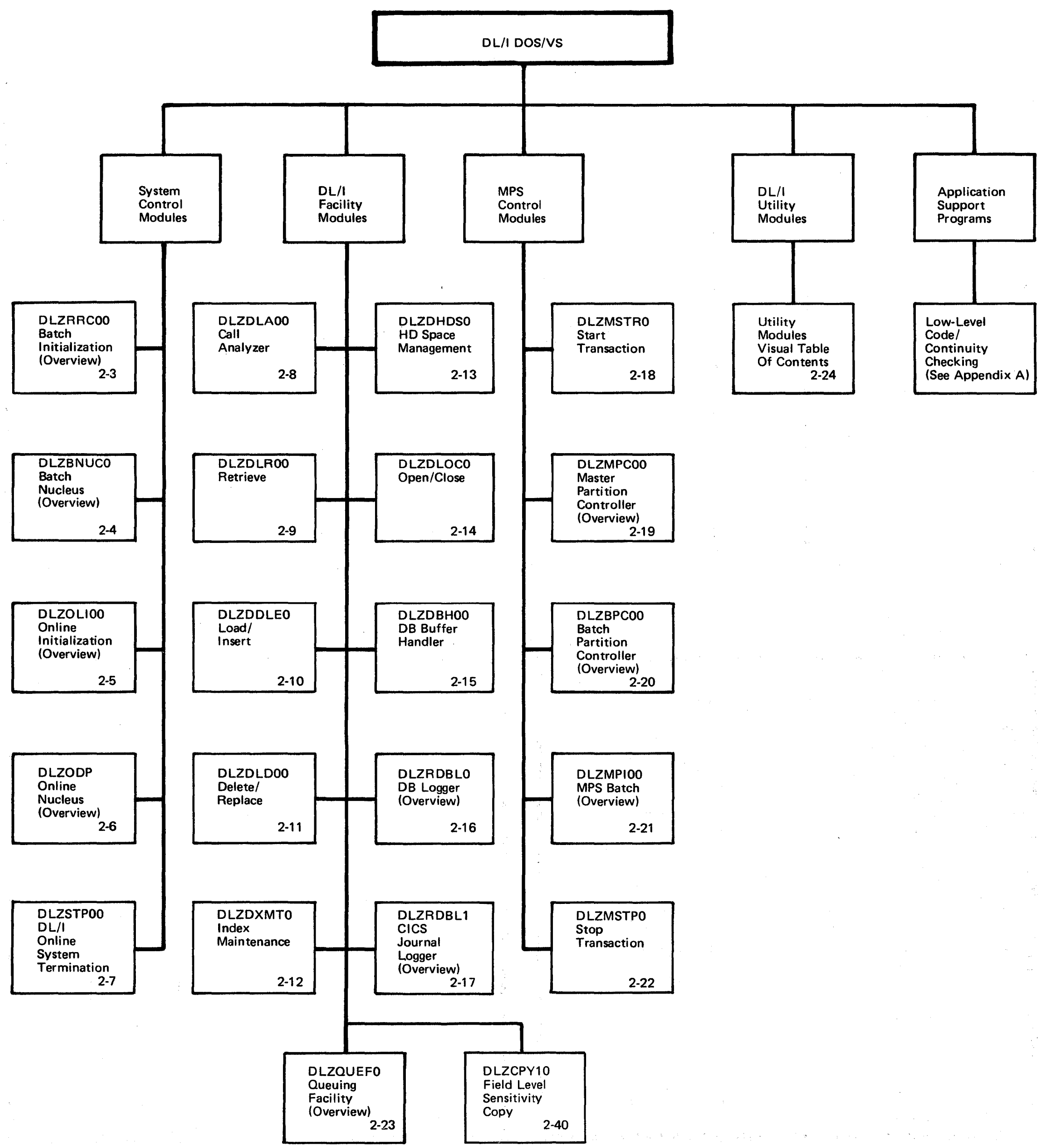


Figure 2-3. Batch Initialization (Overview)

INPUT

From DOS/VS

PROCESS

1. Enter batch initialization. (See Figure 2-3.1)
2. Begin batch partition control. (See Figure 2-3.2)
3. Parameter scan and validation. (See Figure 2-3.3)
4. Begin application program control. (See Figure 2-3.4)
5. Utility block build request handler. (See Figure 2-3.5)
6. Complete application program control. (See Figure 2-3.6)

To DOS/VS

7. Block loader and relocator. (See Figure 2-3.7)
8. Complete control program initialization. (See Figure 2-3.8)
9. DL/I control card analyze routine. (See Figure 2-3.9)

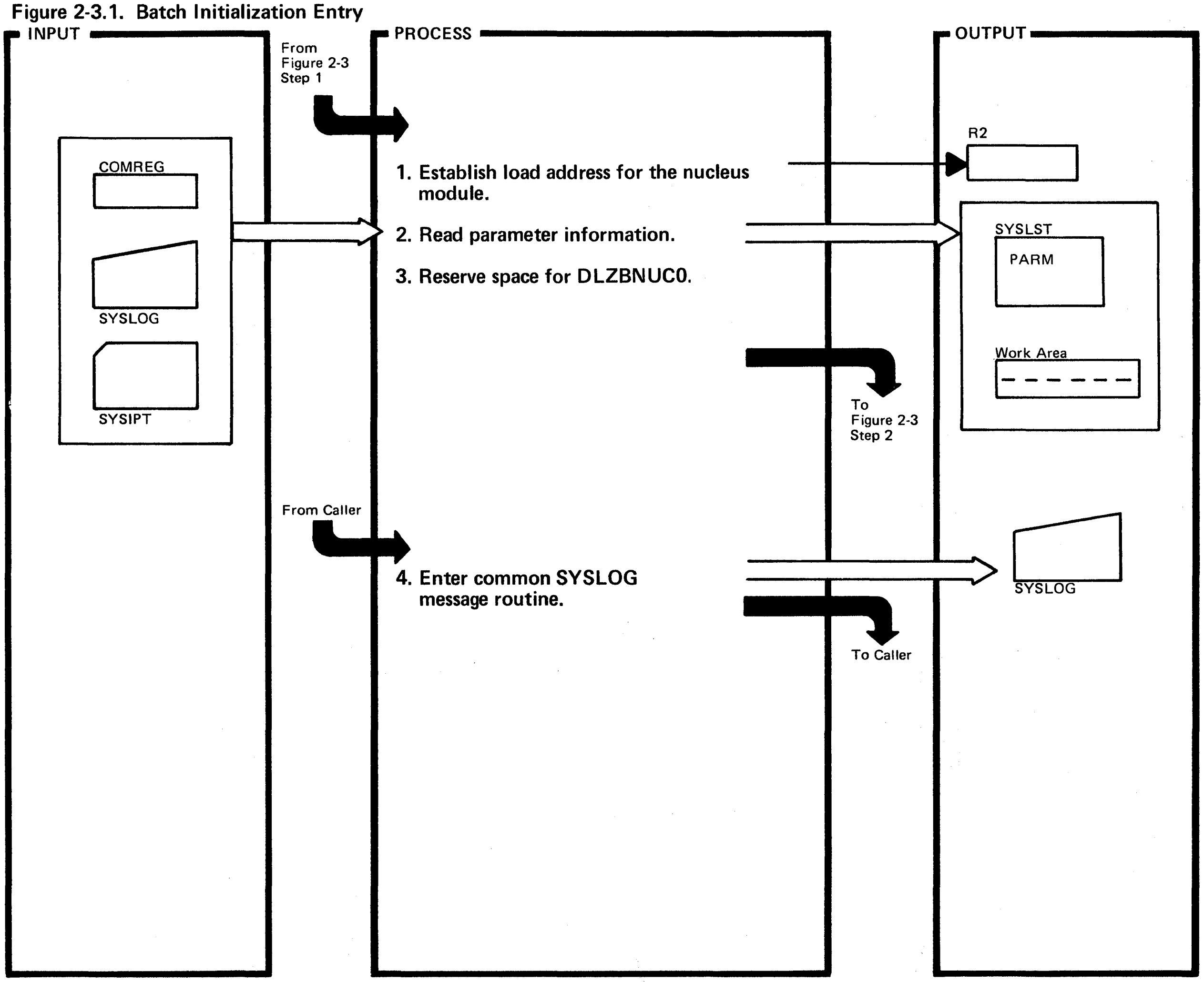
OUTPUT

DLZRRC00

Extended Description	Routine	Label
1.	DLZRRC00	
2.	DLZRRC10	
3.	DLZRRA00	
4.	DLZPCC00 DLZPINIT	
5.	ULUPRHEP	
6.	DLZPCC00	
7.	DLZPINIT	
8.	DLZCPI00	
9.	NXTPORT	

Extended Description	Routine	Label

Figure 2-3.1. Batch Initialization Entry



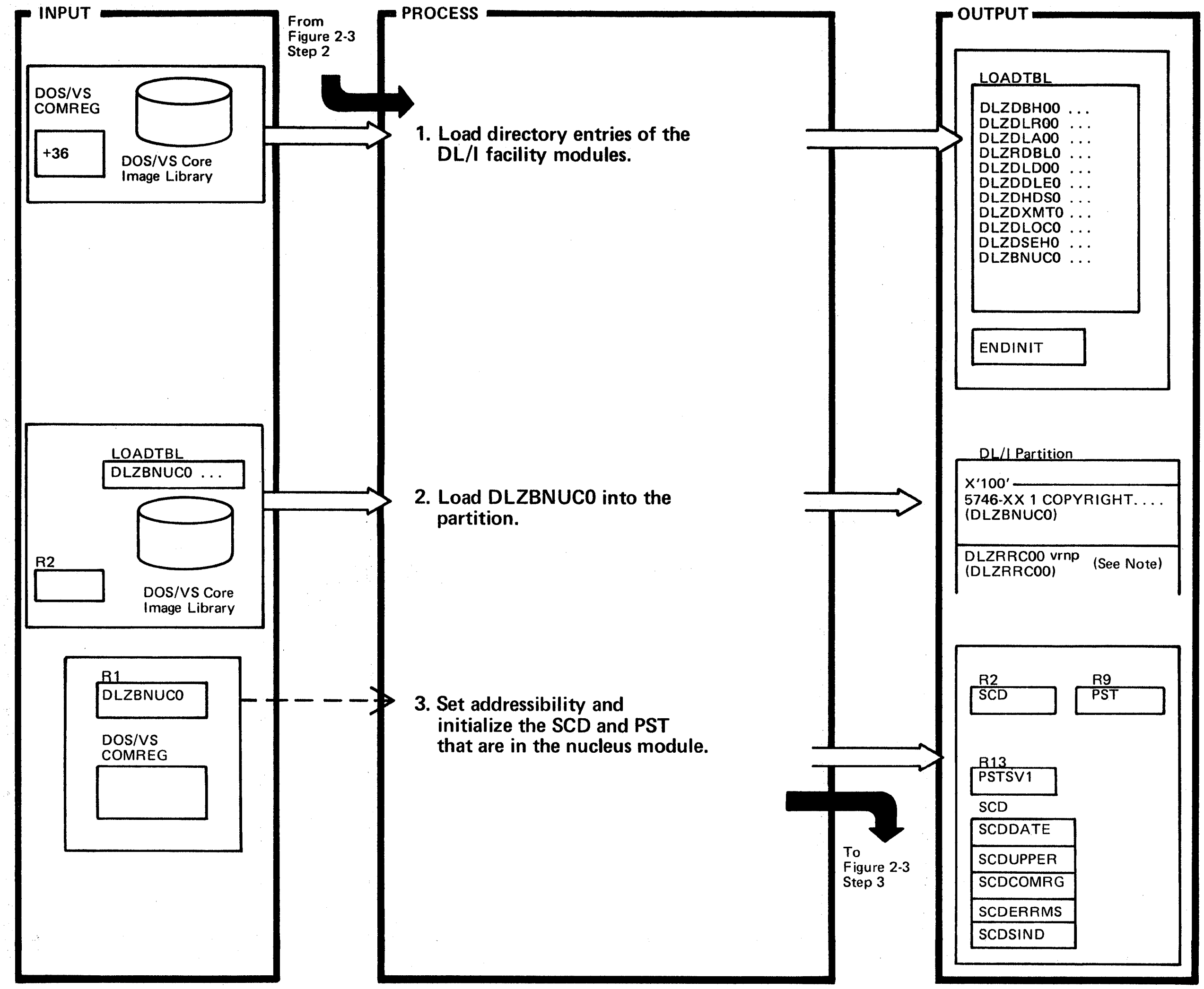
DLZRRC00 — Batch Initialization E.P. CSECT

DLZRRC00

Extended Description	Routine	Label
1. DLZBNUC0 load address is set at DLZRRC00 start + X'100'.	DLZRRC00	DLZRRC00
2.		PARMGET
3. The reserved space allows loading of DLZBNUC0 without overlaying critical code in this module.		
4.		ERRORMSG

Extended Description	Routine	Label

Figure 2-3.2. Batch Partition Control



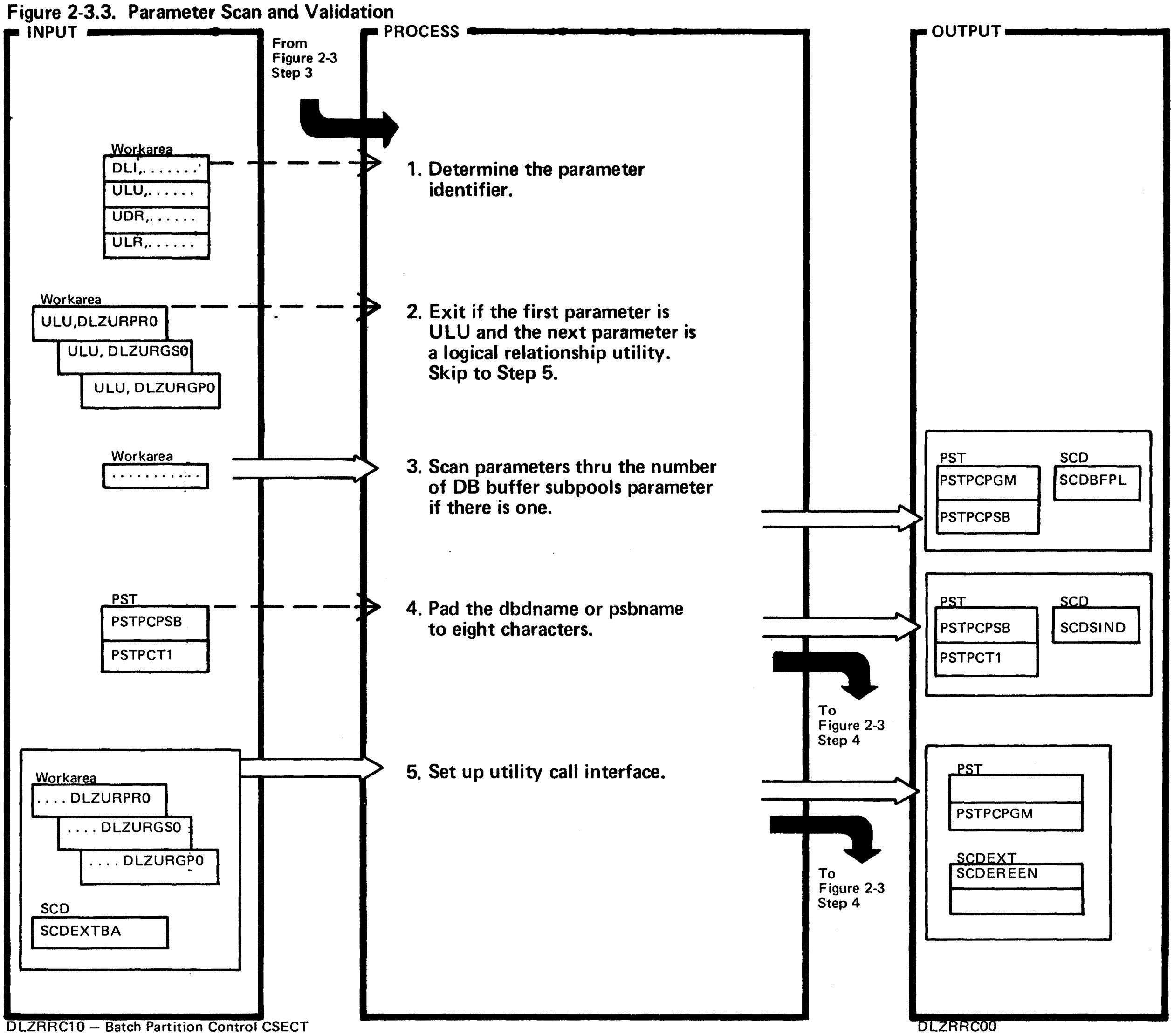
DLZRRC10 — Batch Partition Control CSECT

DLZRRC00

Extended Description	Routine	Label
1. The end address of phase DLZRRC00 is obtained from the DOS/VS COMREG and saved at ENDINIT. Write message DLZ011I if a required module is not found in the DOS/VS core image library.	DLZRRC10	DLZRRC10 LOADNUC
2. Note: DLZRRC00vrnp is the module identifier. Each DL/I module is identified with its full eight-byte module identifier in character format followed with a four-byte field identifying the module level. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification digit, and 'p' is the latest PTF number that has been applied.		LOAD2

Extended Description	Routine	Label

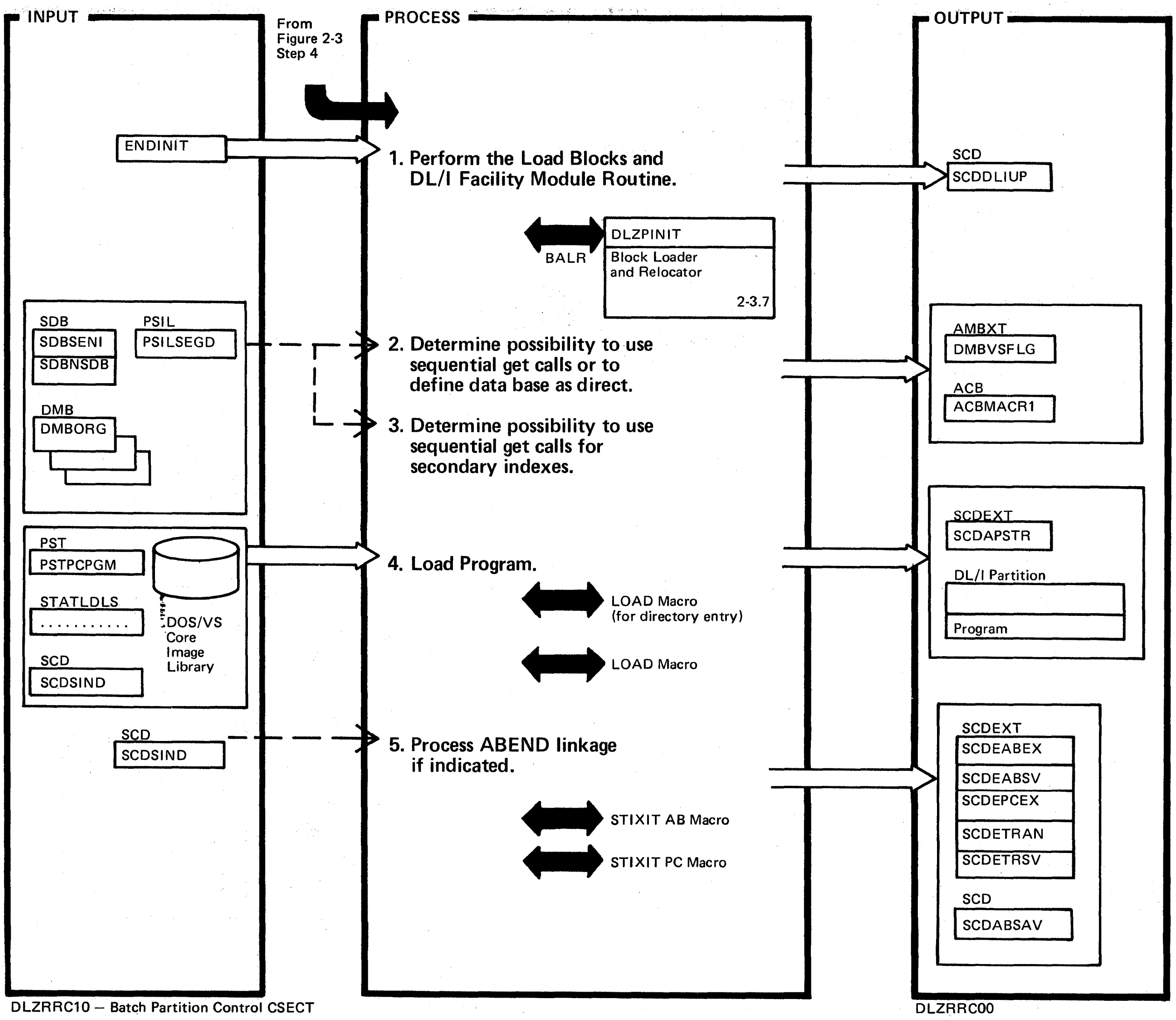
Figure 2-3.3. Parameter Scan and Validation



Extended Description	Routine	Label
1. Write message DLZ015I if the first parameter is not DLI, ULU, UDR, or ULR.	DLZRRAOO	DLZRRAOO
2. Although the DB prefix resolution utility is a logical relationship utility, it is not processed with the others because it executes directly, not as an application to DL/I.		CHK1ST
3. Write message DLZ015I if syntax error occurs.		SCANPARM
4. PSTPCPSB now contains the dbdname from the ULU, UDR, or ULR parameter card or the psbname from the DLI parameter card. Insert a utility DBD suffix (U) or insert a PSB suffix (P).		PARMPAD
5. No control blocks are loaded for DLZURPRO, DLZURGS0, or DLZURGP0 during batch initialization. These three		ULUSTART

Extended Description	Routine	Label
utilities issue the DLZBLKLD macro specifying the utility PSB and the BLDB call for each data set used. The ACB utility builds the utility PSBs they use.		

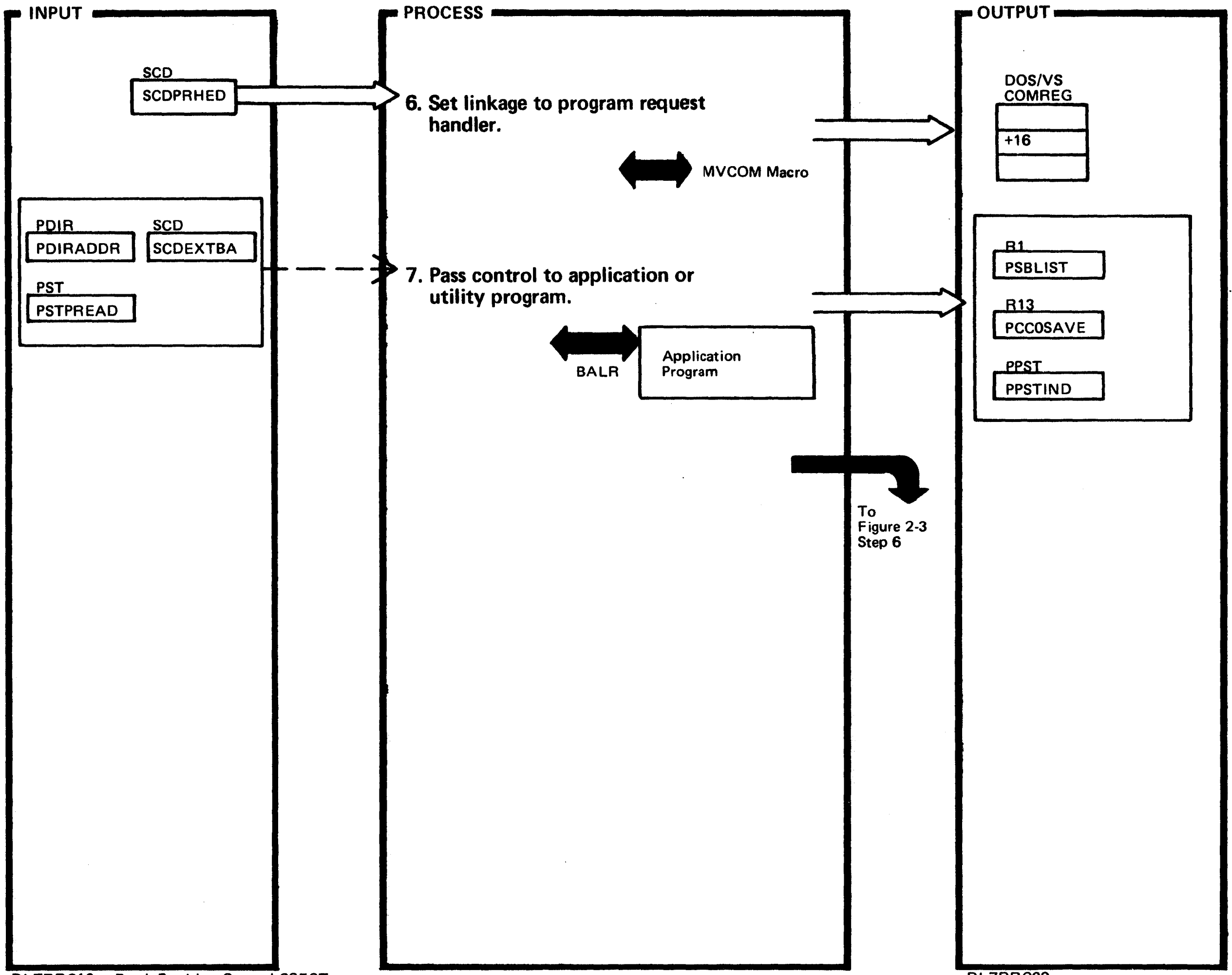
Figure 2-3.4. Application Program Control (Part 1 of 2)



Extended Description	Routine	Label
1. This module's end address is used to initialize the beginning of storage available for control block building.	DLZPCC00	DLZPCC00
2.		LOOKDMBS
3.		FINDISS
4. Write message DLZ012I if program is not found.		CONTPCC LOAD5
5. UPSI card information has been moved to the SCD.		STXITAB

Extended Description	Routine	Label

Figure 2-3.4. Application Program Control (Part 2 of 2)



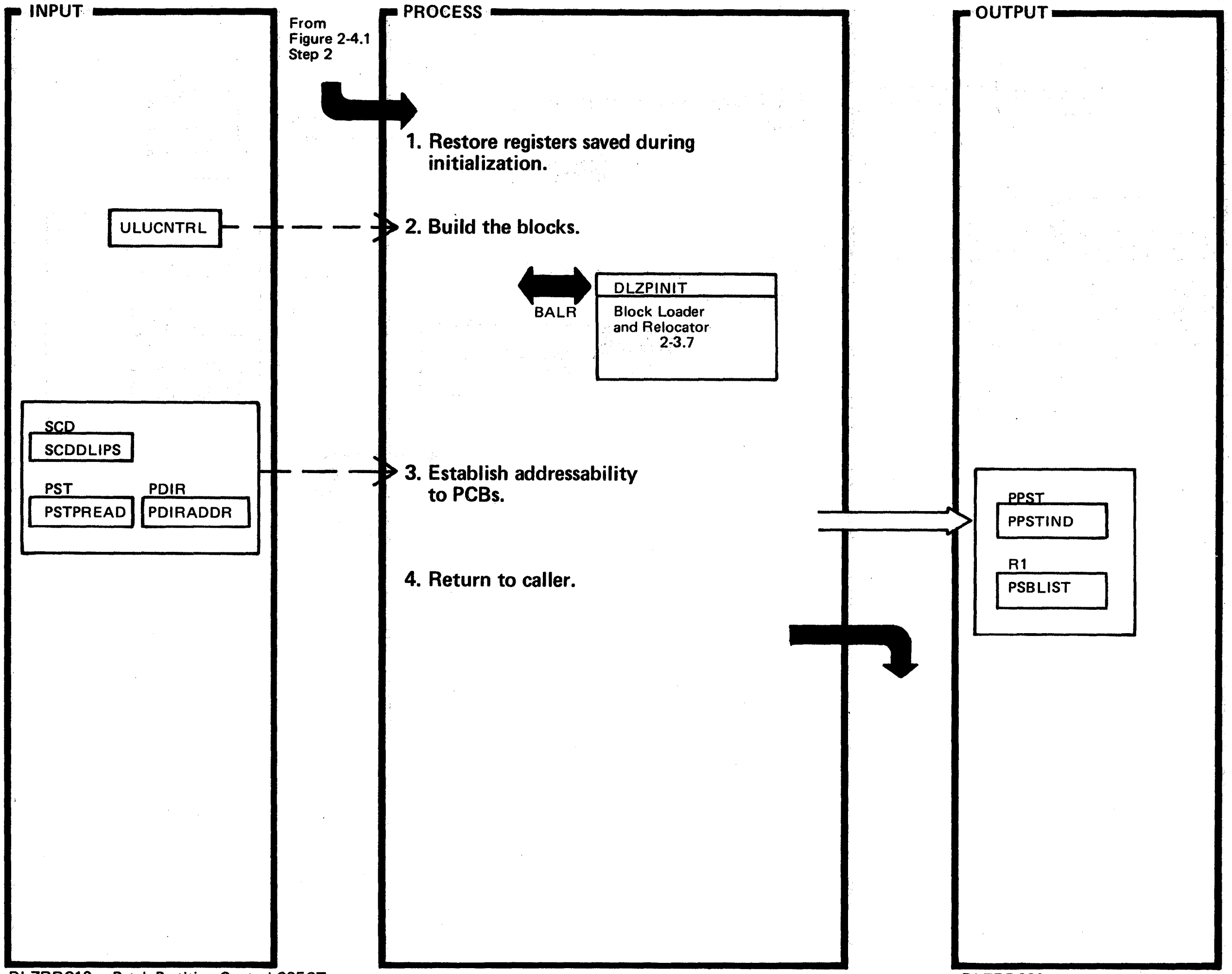
DLZRRC10 — Batch Partition Control CSECT

DLZRRC00

Extended Description	Routine	Label
6. Linkage to DLZPRHB0 is done via MVCOM macro.		MVCOM
7. If utility program is a logical relationship utility, set R1 to point to the PST before passing control to the utility. Set R1 to point to the user PCB list for all other programs.		BALRUSER

Extended Description	Routine	Label

Figure 2-3.5. Utility Block Build Request Entry



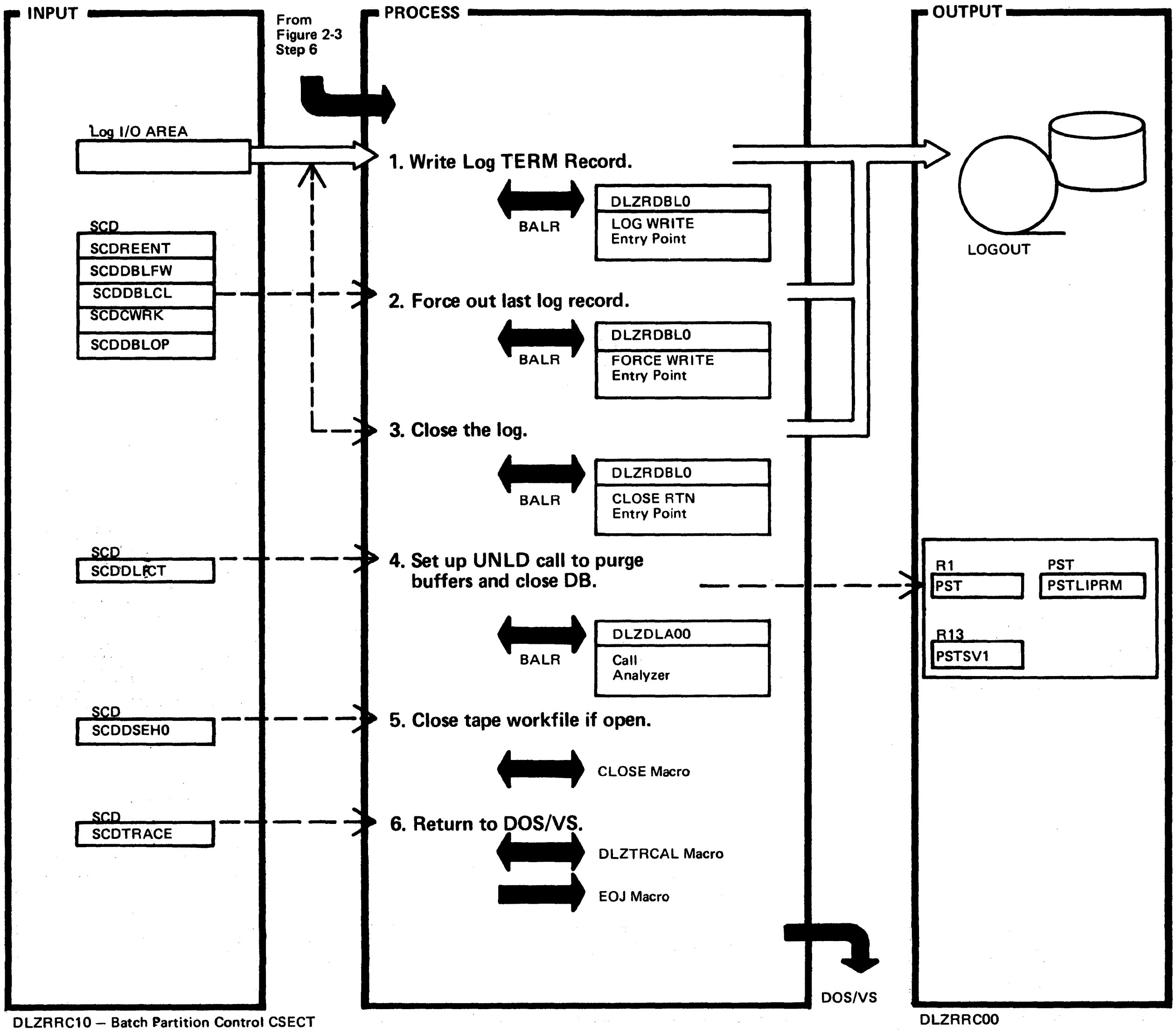
DLZRRC10 — Batch Partition Control CSECT

DLZRRC00

Extended Description	Routine	Label
1. Control comes from the batch program request handler (DLZBNUC0) when a utility block build request (BLDB) is detected.	ULUPRHEP	ULUPRHEP
2. If a block build error is indicated in ULUCNTRL, X'0C' is set in register 15 and control returns to the utility program.		ULUGOOD
4.		ULUEXTZ

Extended Description	Routine	Label

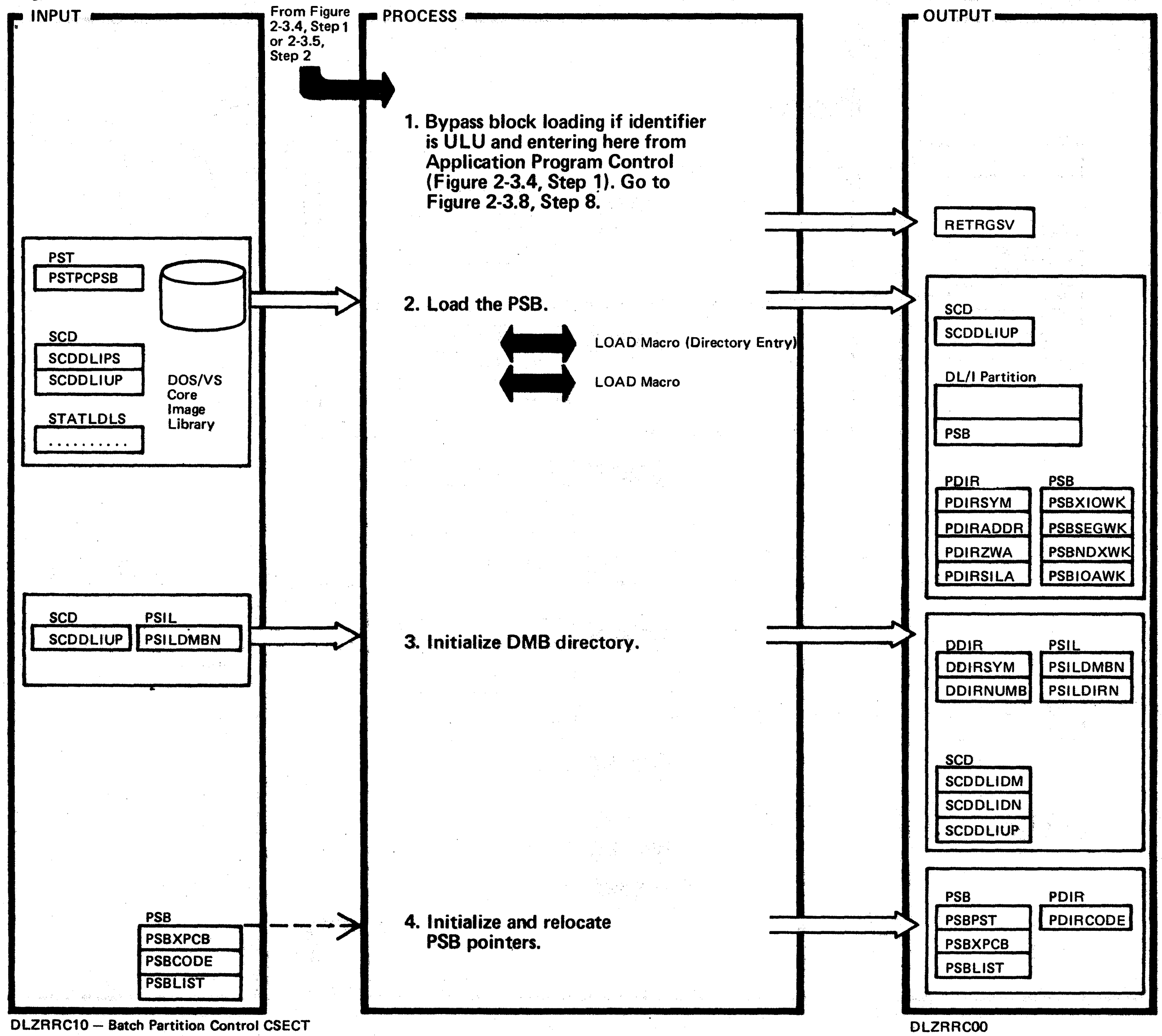
Figure 2-3.6. Application Program Control Completion



Extended Description	Routine	Label
1. TERM record ID=X'07'.	DLZPCC00	BALRUSER
4. The UNLD call is bypassed if the ULU return code in register 15 is not zero.		BYULUEND
6. Issue macro DLZTRCAL TYPE=STOP. Trace ID=X'FC'.		DLZEOJ

Extended Description	Routine	Label

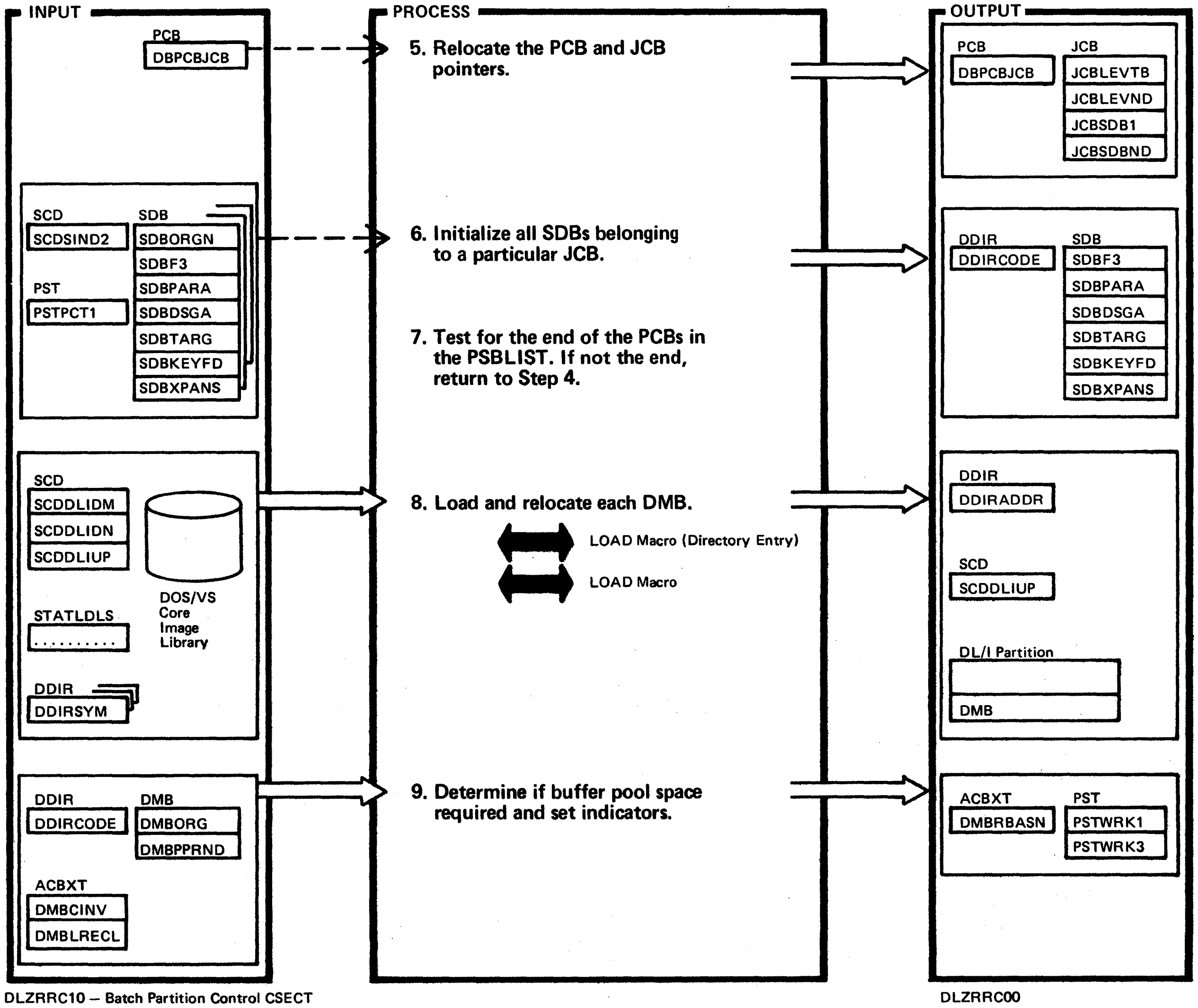
Figure 2-3.7. Block Loader and Relocator (Part 1 of 5)



Extended Description	Routine	Label
1. There are no blocks to load for the logical relationship utilities if this is the first call. The return address is saved in RETRGSV.	DLZPINIT	DLZPINIT
2. If the PCB is not found and the parameter identifier is ULU for a logical relationship utility, set a block load error indicator and return to Figure 2-3.5, Step 3. Write message DLZ012I if the PSB is not found. Write message DLZ017I if the PSB version/modification level is incorrect.		DLZDBLM0

Extended Description	Routine	Label
3. The PSILs are scanned for DMB names and a DDIR is created for each unique DMB encountered. The address of the DDIR replaces the respective DMBNAME in each PSIL.		DDIRBILD
4.		PCBRLUIP

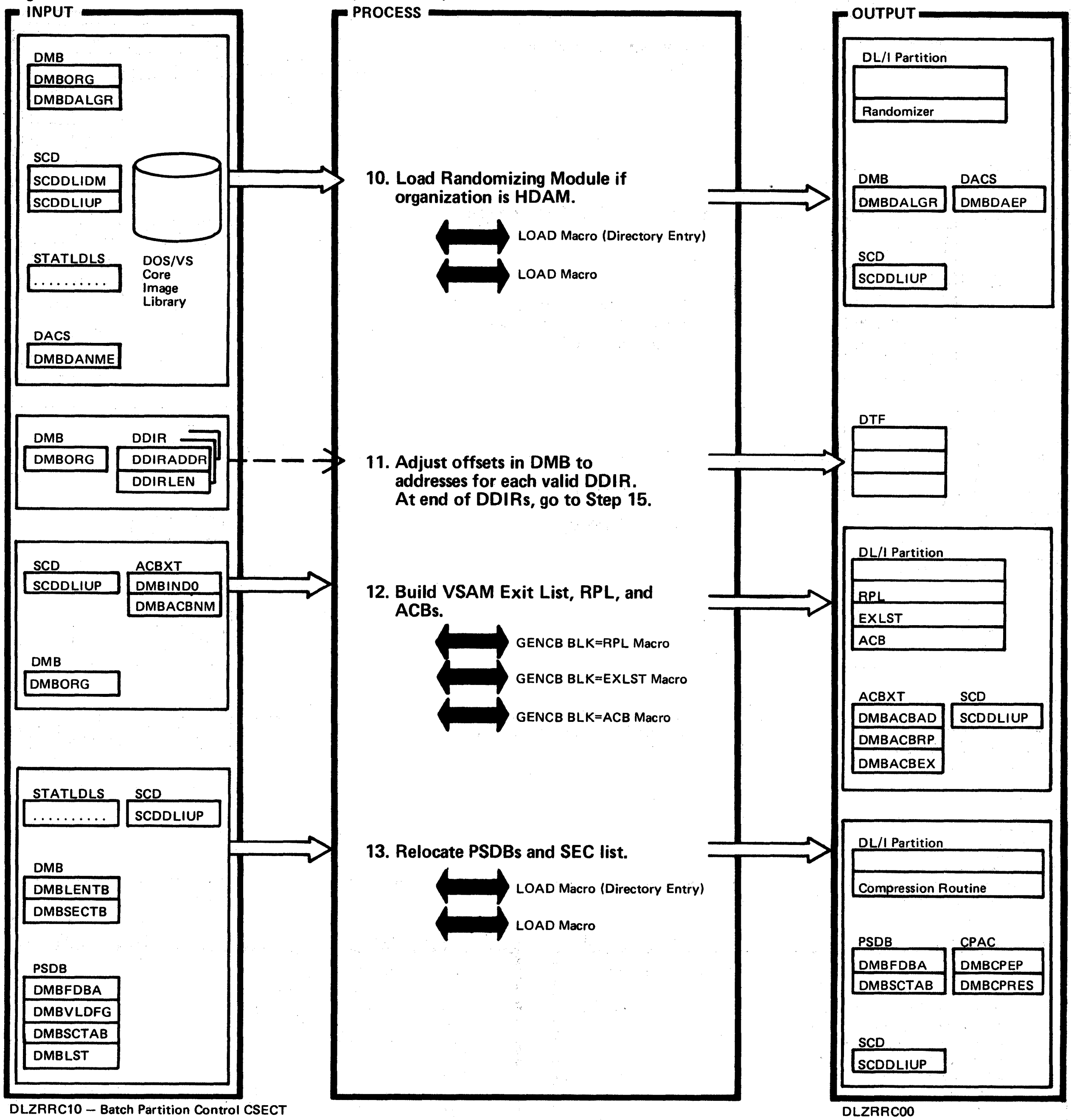
Figure 2-3.7. Block Loader and Relocator (Part 2 of 5)



Extended Description	Routine	Label
5.	DLZPINIT	PCBPLIB
6. The call sensitivity and data base organization of the SDB is checked to see if buffer pool space is required. An indicator in the DDIR is turned on if space is required.		SDBRELO
7. The pointer to the PSBLIST is bumped to the next PCB pointer entry and processing returns to Step 4 if we are not at the last PCB.		NPCBCK
If the index PCB exists, return to Step 5 and relocate it.		
8. Write message DLZ012I if the DMB is not found.		LOADDMBS DMBLOADR
Write message DLZ018I if the DMB version/modification level is incorrect.		

Extended Description	Routine	Label
If the DMB is not found and the parameter identifier is ULU for a logical relationship utility, set a block load error indicator and return to Figure 2-3.5, Step 3.		
9. If buffer pool space is required, the size of each control interval (rounded to the next multiple of 512) is indicated in PSTWRK1 and PSTWRK2 for later allocation of the buffer pool.		GETBUFRS

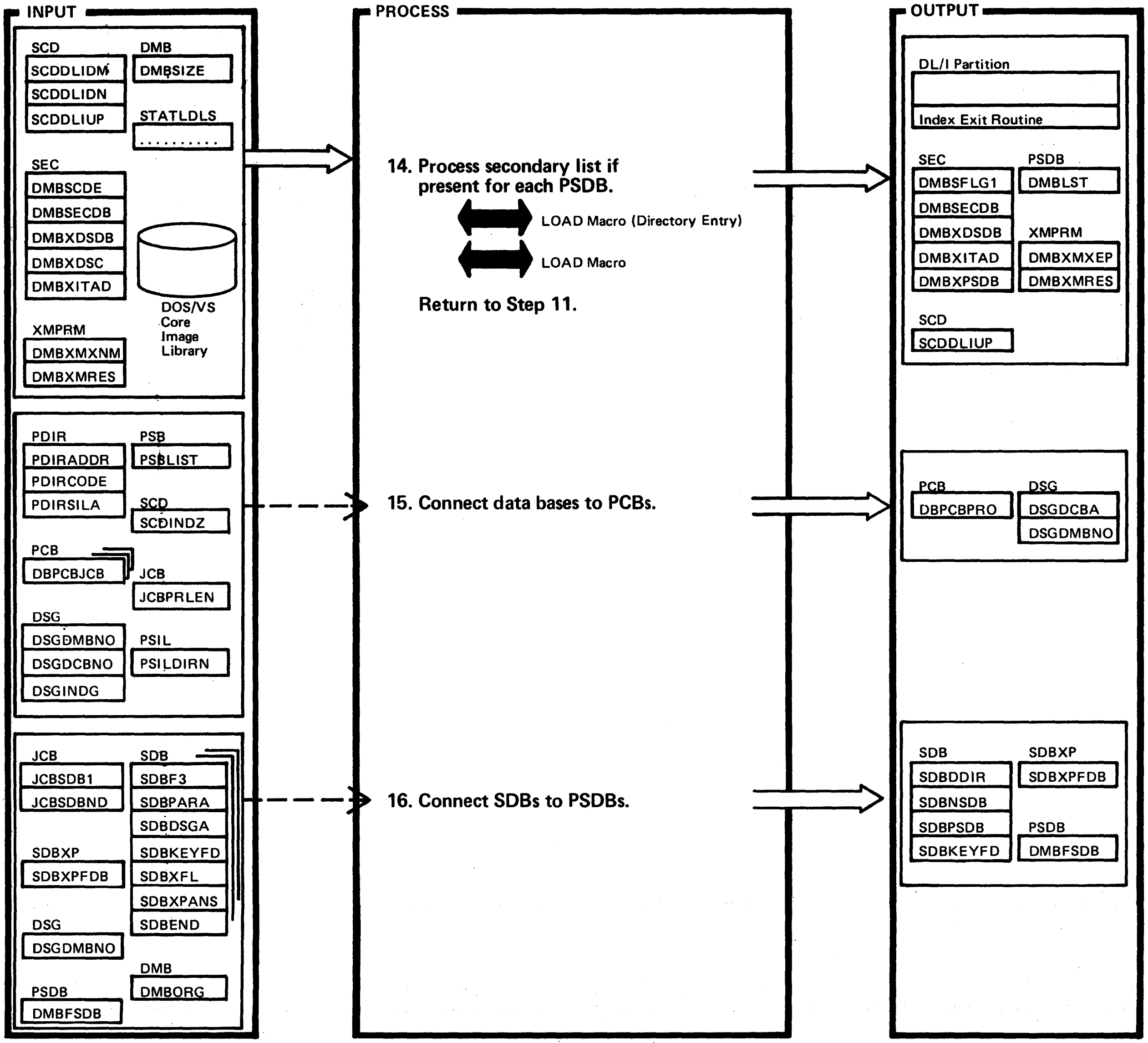
Figure 2-3.7. Block Loader and Relocator (Part 3 of 5)



Extended Description	Routine	Label
10. Before loading the randomizer a check is made with all currently loaded randomizers. If one with the same name as the one we are loading is found, the entry point is resolved and the actual load is bypassed. Return to Step 8 until there are no more DDIRs. Write message DLZ012I if the randomizing module is not found.	DLZPINIT	RANCKLUP

Extended Description	Routine	Label
11. The DTF address constants are adjusted if access is HSAM or simple HSAM.		DMBOFFAJ
12. If HISAM, two sets of control blocks will be built.		DLZBVBLD
13. The segment compression routine for each PSDB is loaded (if it hasn't already been loaded for a previous PSDB). Write message DLZ012I if the compression routine is not found.		PSDBROUT

Figure 2-3.7. Block Loader and Relocator (Part 4 of 5)



DLZRRC10 — Batch Partition Control CSECT

DLZRRC00

Extended Description	Routine	Label
14. If a secondary list is present, its code is tested and referenced DMBs are resolved to DDIR pointers and placed in the list. If an index user exit routine is present it is loaded if it hasn't already been loaded for a previous SEC. Write message DLZ266I if there is an invalid secondary list code. Write message DLZ012I if the user exit routine is not found. Write message DLZ263I if the SEC makes an invalid DMB reference.	DLZPINIT	PROCSEC

Extended Description	Routine	Label
15. If this is Reload Restart, the parameter ID is ULR. Bypass checking whether the processing option should be changed to load or not.		PCBROUT
16.		CONSDBS

Figure 2-3.7. Block Loader and Relocator (Part 5 of 5)

INPUT

SDB
- SDBXFFSB
- SDBXFL
- SDBXFNB
- SDBXPANS
- SDBXPSZ

SCD
- SCDDLIUP
- SCDPDCA

PSB Prefix
- PSBCODE
- PSBFRTA

FSB
- FSBUVTYP
- FSBIVA
- FSBFERTA
- FSBLEN

FERT
- FERTRTEP
- FERTNAME
- FERTPRES

DLZRRC10 — Batch Partition Control CSECT

PROCESS

17. Initialize field level sensitivity tables.

To Figure 2-3 Step 8

OUTPUT

FSB
- FSBIVA
- FSBFERTA

PDCA
- PDCAFERT

SCD
- SCDCPY10
- SCDDLIUP

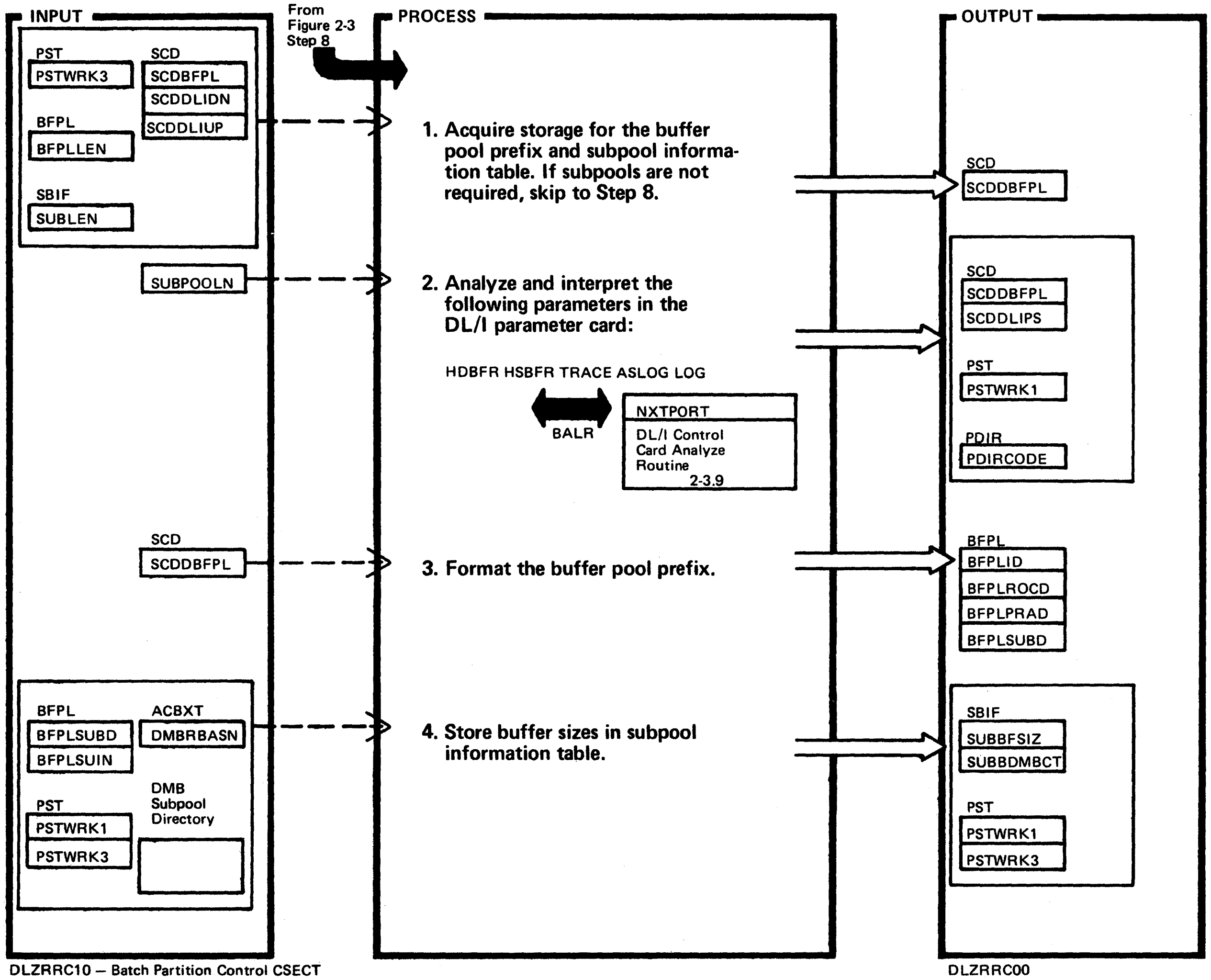
FERT
- FERTRTEP
- FERTRTLG
- FERTPRES

DLZRRC00

Extended Description	Routine	Label
17. Includes FERT, FSB, and loading user field exit routines.		

Extended Description	Routine	Label

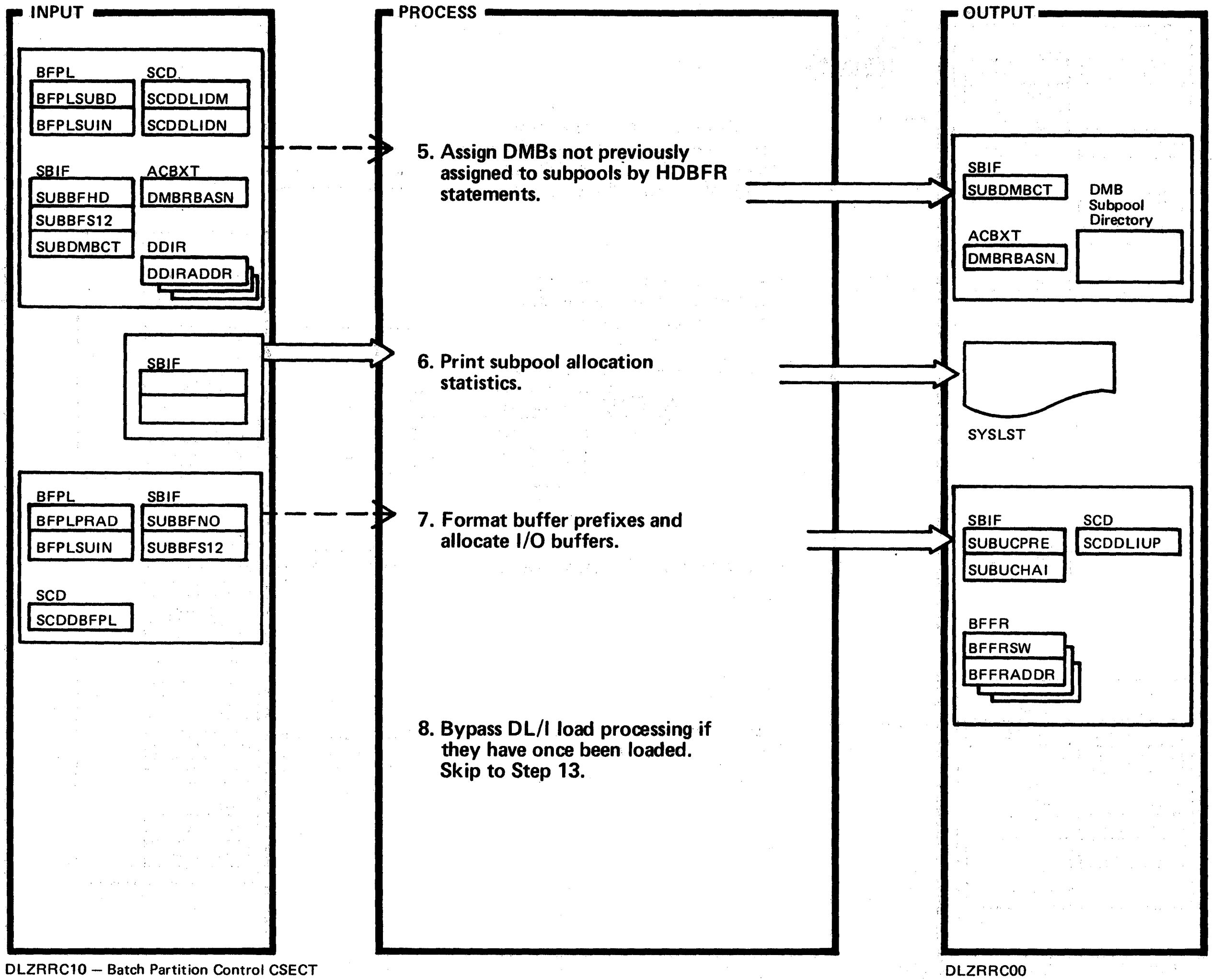
Figure 2-3.8. Control Program Initialization Completion (Part 1 of 3)



Extended Description	Routine	Label
1.	DLZCPI00	DLZCPI00 BFRPRNT
2. Write message DLZ009I if number of subpools specified in the parameter statement are not equal to the number of HDBFR statements.		BFPNDCLR PRMSRET
3.		BFPREADY
4. This step determines the size of the subpools. They are allocated, largest first, until the specified number is exhausted. Remaining DMBs requiring subpools are assigned evenly across all existing subpools. If you specified more subpools than necessary, an additional pool of 512 buffer size is allocated for delete workspace. The subpool sizes are sorted so that the largest subpool appears first in the information table.		NODMMOV

Extended Description	Routine	Label

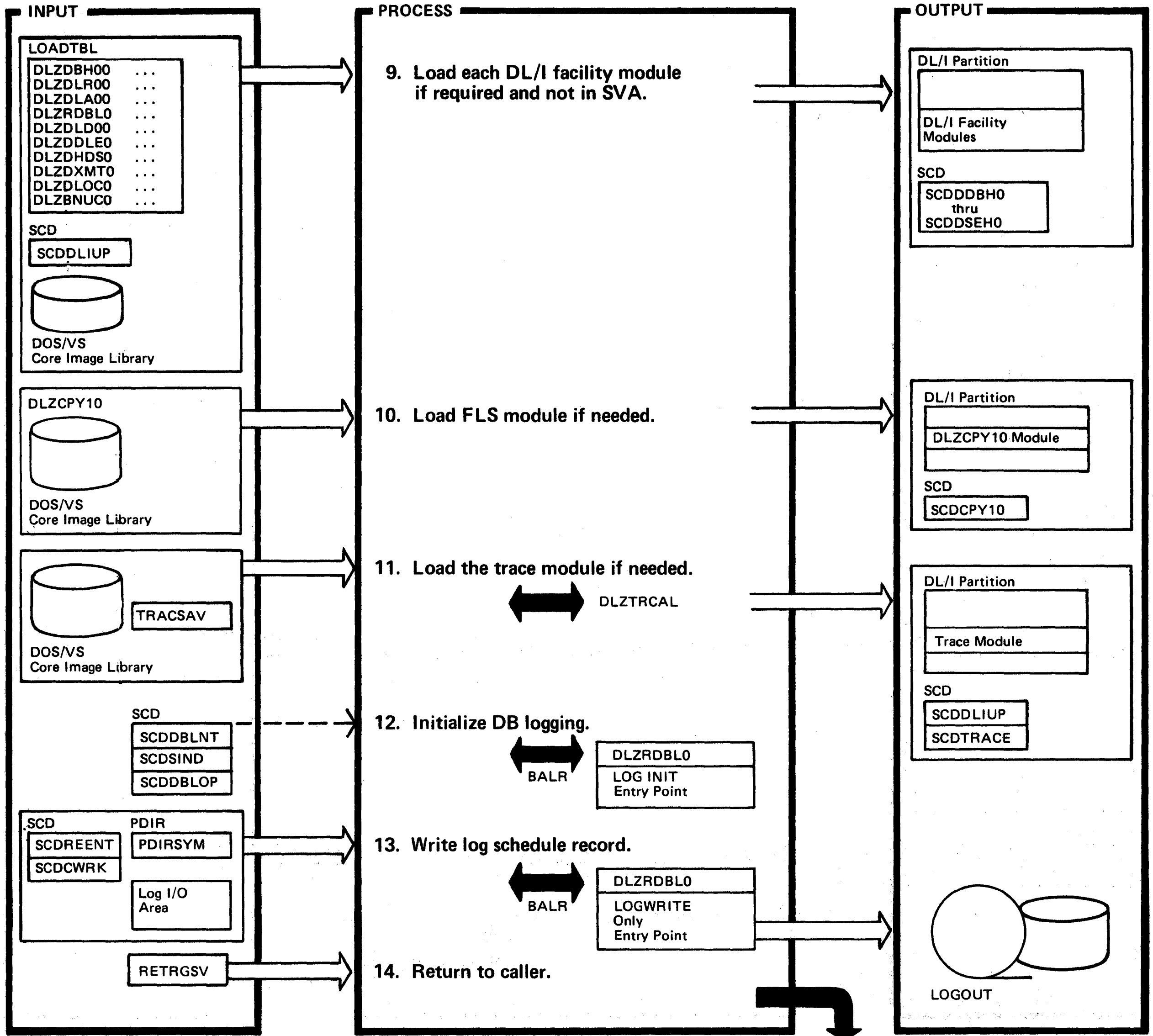
Figure 2-3.8. Control Program Initialization Completion (Part 2 of 3)



Extended Description	Routine	Label
5. Write message DLZ262I if buffer allocation error occurs.		GREATPRO
6.		SPSTAT
7.		BFRINIT
8.	DLZCPI00	DLILOAD

Extended Description	Routine	Label

Figure 2-3.8. Control Program Initialization Completion (Part 3 of 3)



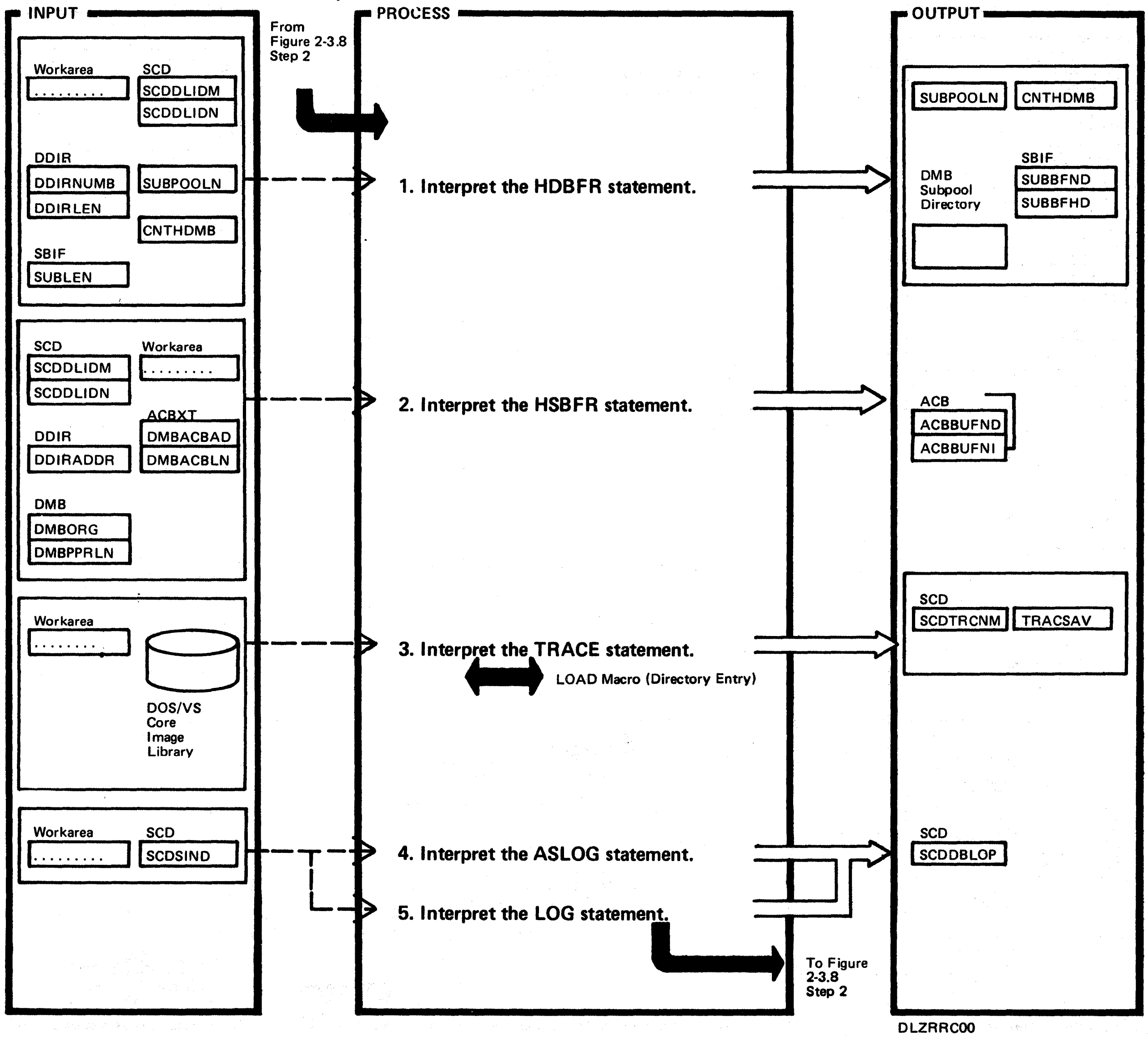
DLZRRC10 — Batch Partition Control CSECT

DLZRRC00

Extended Description	Routine	Label
9.		NUCLODUC
11. Issue macro DLZTRCAL TYPE=START following the load. Trace ID=X'FE'. Write message DLZ026I if initialization fails.		LOAD9
12. Cancel if open error returned. Upon return, the entry points to DLZRDBL0 in the 'Data Base Change Log Section' of the SCD (beginning with the SCDREENT) are initialized.		NOLOMOD
13. The scheduled record ID='08'.		
14. Return is made to the instruction following the BALR to DLZPINIT.		PCCORET

Extended Description	Routine	Label

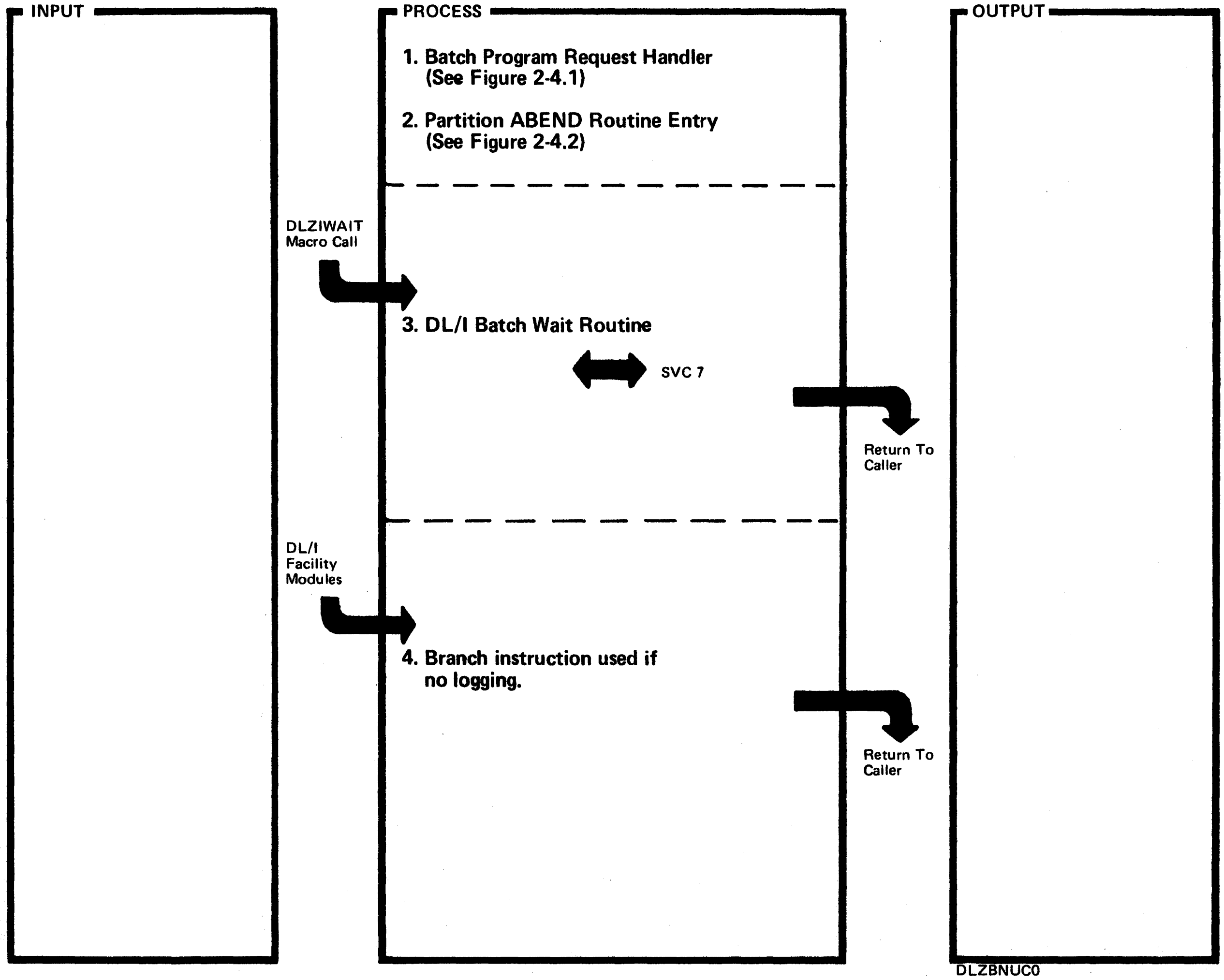
Figure 2-3.9. DL/I Control Card Analyze Routine



Extended Description	Routine	Label
1. The number of buffers/subpools specified in the HDBFR statement is set in the SBIF. Write message DLZ019I if the number is greater than 32 or less than 2. Default is 2. The SUBPOOLN is incremented 1 for every HDBFR statement. Each DMB is assigned by placing the relative subpool number (SUBPOOLN) it is being assigned to into a byte of the DMB SUBP DIR which corresponds to that DMB. The length in bytes of the DMB SUBP DIR equals the total number of DMBs. Write message DLZ008I if this DMB has already been assigned a subpool. CNTHDMB is a count of all the data bases assigned by the user in the HDBFR statements.	NXTPORT	NXTPORT HDBFR

Extended Description	Routine	Label
Write message DLZ008I if a DMB name is invalid.		
2. The user specified VSAM buffer allocations are set in the ACB for HISAM and INDEX DBDs.		HSBFR
Write message DLZ008I for an invalid DMB reference. Write message DLZ019I if valid values were not specified.		
3. Write message DLZ012I if module is not found.		TRACE
4. Write message DLZ015I if there is a syntax error.		ASLOG
5. Write message DLZ078I if UPSI card said no log.		LOG
Write message DLZ075I if invalid parameters.		

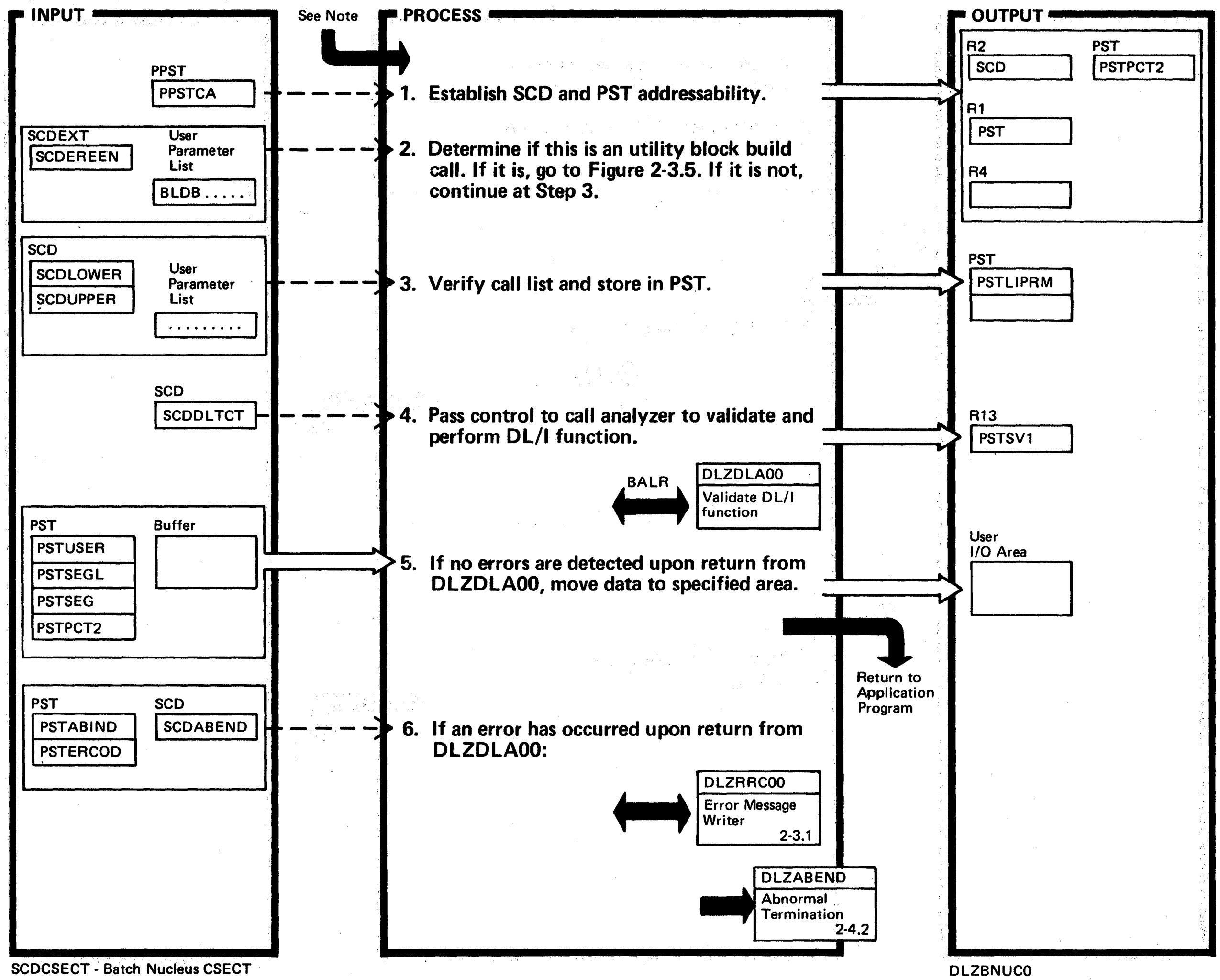
Figure 2-4. Batch Nucleus (Overview)



Extended Description	Routine	Label
3. The DLZIWAIT macro is used by DLZRDBH00, DLZDBH02 and DLZRDBL0.	DLZIWAIT	DLZIWAIT
4. After the DLZBNUC0 module is loaded, SCDDBLNT contains the entry point of this routine. If, however, batch initialization (DLZRRC00) determines that the DB logger is required, the entry point of the log initialization routine in DLZRDBL0 is stored in SCDDBLNT. The log initialization routine changes SCDDBLNT once more to point to the log writer entry point. With this routine, the DL/I facility modules need not know if logging is required or not.	DLZBR14	DLZBR14

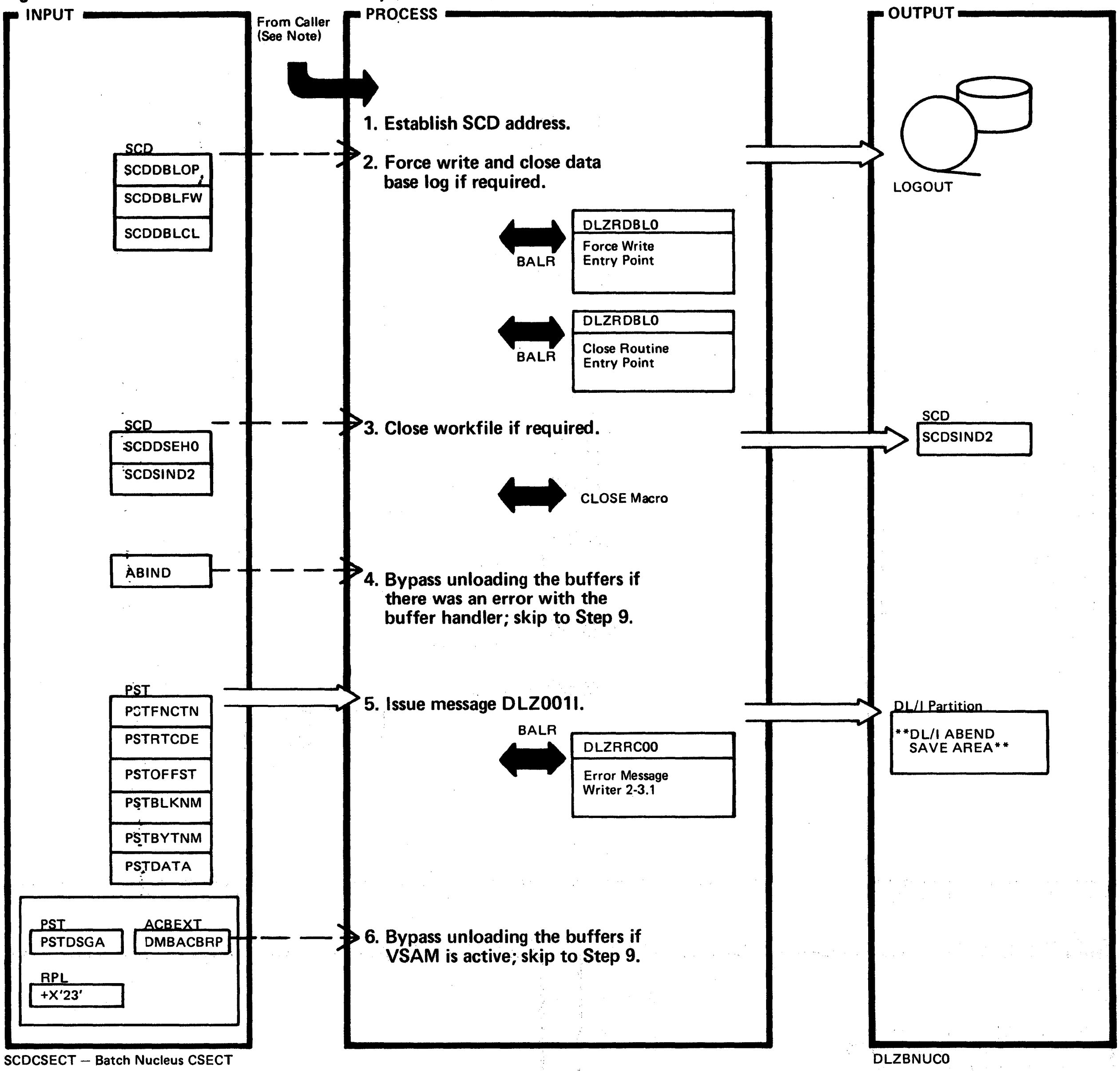
Extended Description	Routine	Label

Figure 2-4.1. Batch Program Request Handler



Extended Description	Routine	Label
Note: This routine receives control from the language interface module (DLZLI000) linked with the application program.		
1. When control is passed to the program request handler, register 1 must point to the user parameter list and register 13 to the user save area.	DLZPRHB0	DLZPRHB0
During the first entry to DLZPRHB0, the PL/I STXIT routine and savearea addresses from the PC option table are saved if the application program is written in PL/I. DLZPRHB0 also sets/resets a switch (SCDLIPLI flag in SCD) on exit/entry to indicate whether current execution is in DL/I code or PL/I code. This is done to enable high level language debugging for PL/I to give diagnostic information if a program check occurs in PL/I code.		
Reset PC exits if this is a PL/I application.		
2.		BYPLSTXT
3. Write message DLZ260I if invalid list count. Write message DLZ261I if invalid parameter address. Then exit to DLZABEND.		CNTLUP
4.		MOVLUPBP
5. Write message DLZ105I if a checkpoint was taken.		
6. If a DL/I routine determined that DL/I should be terminated, go to the common error message routine to write an error message using the message number stored in PSTERCOD by the DL/I routine.		PRHABEND

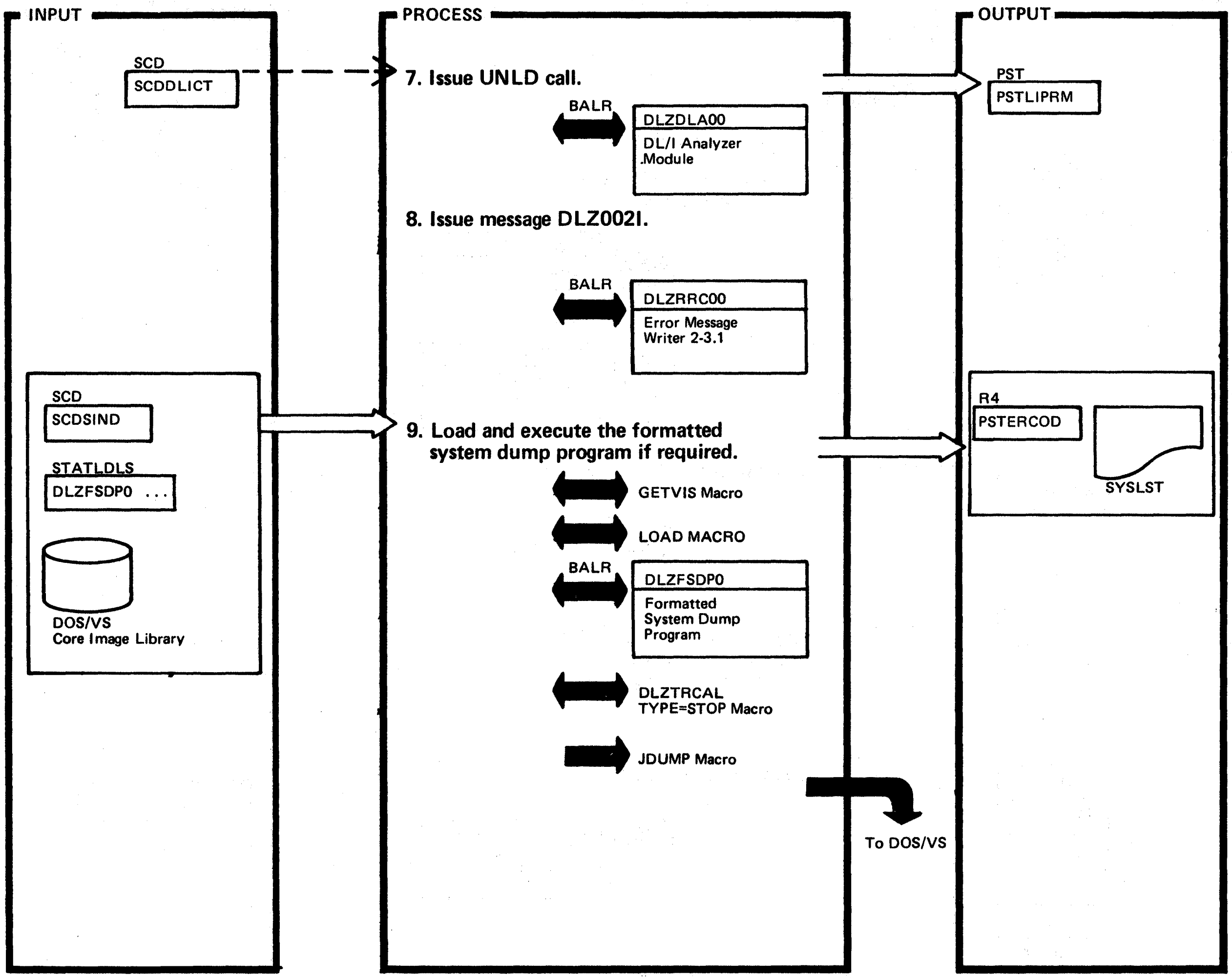
Figure 2-4.2. Partition ABEND Routine Entry (Part 1 of 2)



Extended Description	Routine	Label
Note: The ABEND routine is invoked by the DOS/VS supervisor if (1) there is a program check or other ABEND situation found by DOS/VS, (2) if the job is being abnormally ended by a DL/I routine that determines DL/I should be abnormally ended, or (3) specifically by the buffer handler when there is an error concerning buffers.		
1. If there is a program check, DLZABEND checks the switch (SCDLIPLI flag in SCD) set by DLZPRHB0 to determine if program check occurred in PL/I code. If error occurred while in PL/I code (SCDLIPLI=1), a return address is modified and a branch is made to PL/I STXIT PC routine. (The address of the PL/I STXIT PC routine was saved during	DLZABEND	DLZABEND

Extended Description	Routine	Label
the first entry to DLZPRHB0 - see Figure 2-4.4, Step 1.) After PL/I completes diagnostic information, processing returns to the modified address in DLZABEND.		
3. If the HD reorganization reload module (DLZURGL0) is running for either a standard reload or a reload restart, close the workfile generator file if it is open.		ABLOGCBP
4.		RELODCBP

Figure 2-4.2. Partition ABEND Routine Entry (Part 2 of 2)



SCDCSECT — Batch Nucleus CSECT

DLZBNUCO

Extended Description	Routine	Label
7.		ABUNLD
9. The GETVIS macro is used to acquire storage for DLZFSDP0. If there is not enough storage available to DLZFSDP0, only JDUMP output is put to SYSLST.		ABBYMSG

Extended Description	Routine	Label

Figure 2-5. Online Initialization (Overview)

INPUT

From CICS/VS Overlay Supervisor

PROCESS

1. Online initialization start. (See Figure 2-5.1)
2. PSB processing. (See Figure 2-5.2)
3. DMB processing. (See Figure 2-5.3)
4. Control program initialization. (See Figure 2-5.4)
5. DMB open processing and online initialization completion. (See Figure 2-5.5)

To CICS/VS Overlay Supervisor

OUTPUT

CICS/VS Partition

DLZOLI00vrnp

DLZOLI00vrnp is the module identifier where 'v' is the version, 'r' is the release, 'n' is an additional identification digit, and 'p' is the latest PTF number that has been applied.

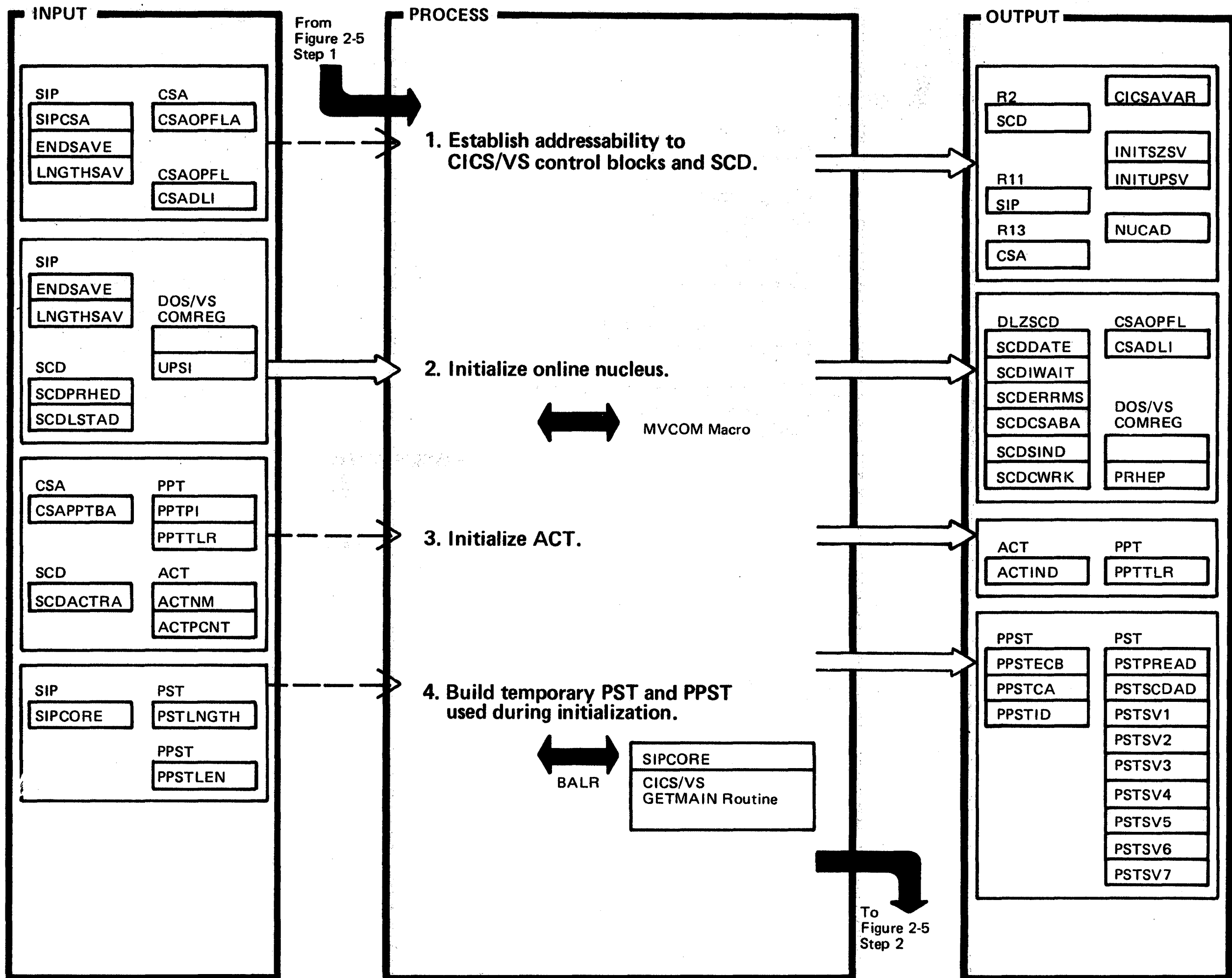
DLZOLI00

Extended Description

	Routine	Label
1.	DLZOLI00	
2.	PSBLOADL	
3.	DDIRINIT	
4.	DLZCPI00	
5.	DMBOPENA	

	Routine	Label

Figure 2-5.1. Online Initialization Start



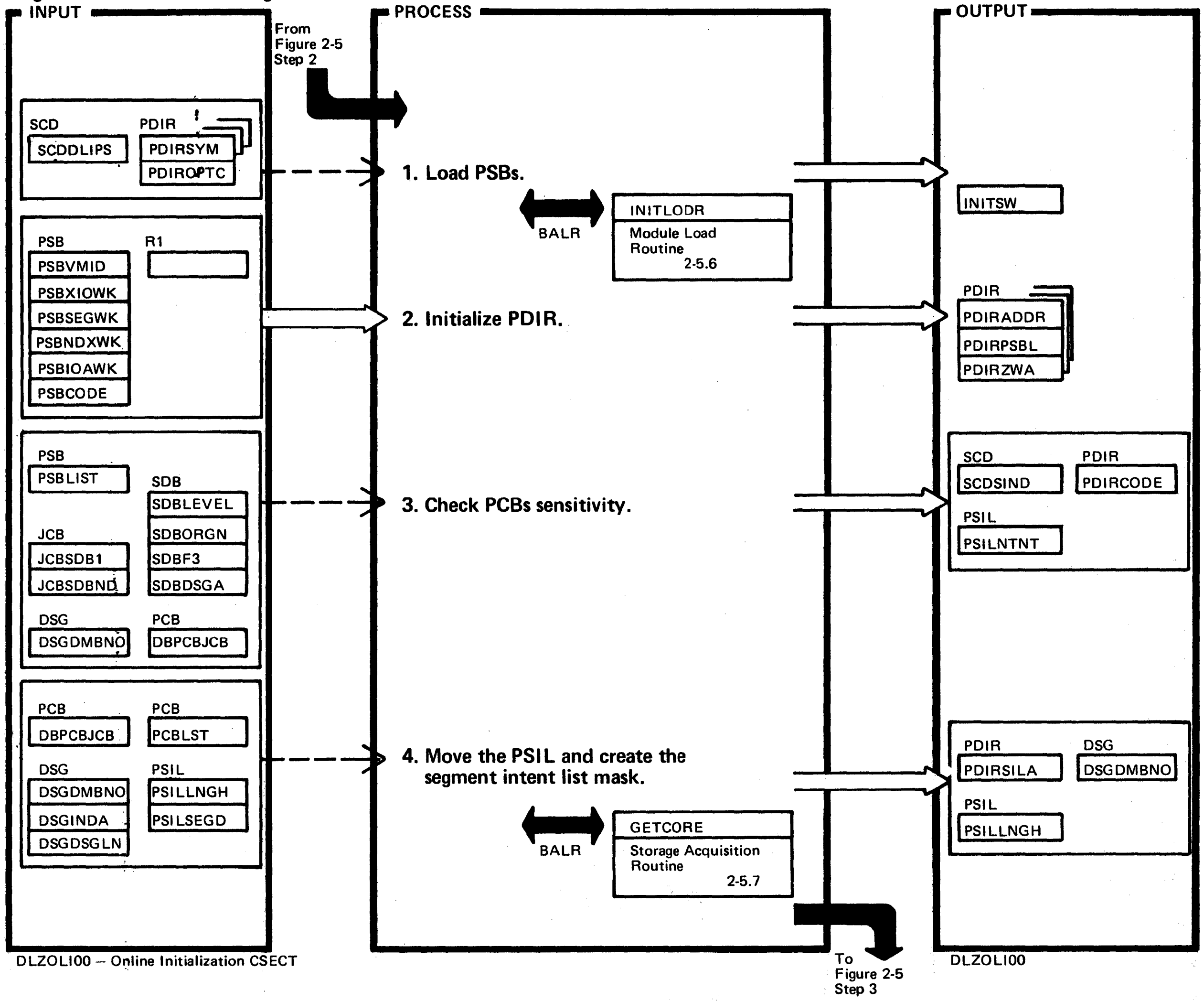
DLZOLI00 – Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
1. Module identifier (DLZOLI00vrnp) is defined here.	DLZOLI00	DLZOLI00
Upon entry from the CICS/VS Overlay Supervisor, SIPBAR2 contains the overlay entry point, and SIPBAR1 contains the SIP common communications area. The current storage allocation information is saved in order to release storage if DL/I initialization fails.		
The DL/I systems contents directory is located from the CSADLI field in the CSA optional features list (CSAOPFL).		
Write message DLZ031I if program isolation is being used and either CICS/VS journalling is not being used or program name DFHDBP (dynamic transaction backout) is not in the PPT.		
Write message DLZ050I if the nucleus is not found.		
Write message DLZ064I if the nucleus is an invalid version.		
2. CSADLI is modified to point to the DL/I interface module address list, (DFHDLIAL DSECT), which is a table of entry points for the task initiation module and the task and system termination routines.		NUCFOUND

Extended Description	Routine	Label
SCDSIND is initialized with bits 6 and 7 of the UPSI switch from the COMREG. The program request handler entry point is moved to byte 16 of the COMREG and temporary entry points are established for the error message routine and the DL/I wait routine. Also, SCDCWRK is initialized at this time to point to the beginning of the low end of free storage.		
3. Indicators are set in the CICS/VS PPT marking the program eligible for DL/I services. They are set in the DL/I ACT entry indicating the program was located in the PPT.		ACTLOOP
Write message DLZ034I for each ACT program not in the PPT. Write message DLZ051I if any ACT program is not in the PPT.		
4. The PST and PPST are built directly after the initialization overlay high storage address. The save areas are chained and SCDCWRK is updated to indicate the new upward core allocation starting address. A dummy task ID of '01' is set in the PPST.		PSTPPST
Since a DL/I task with its associated PST is not yet involved, a temporary PST is needed to provide work space and save areas during the execution of this module.		

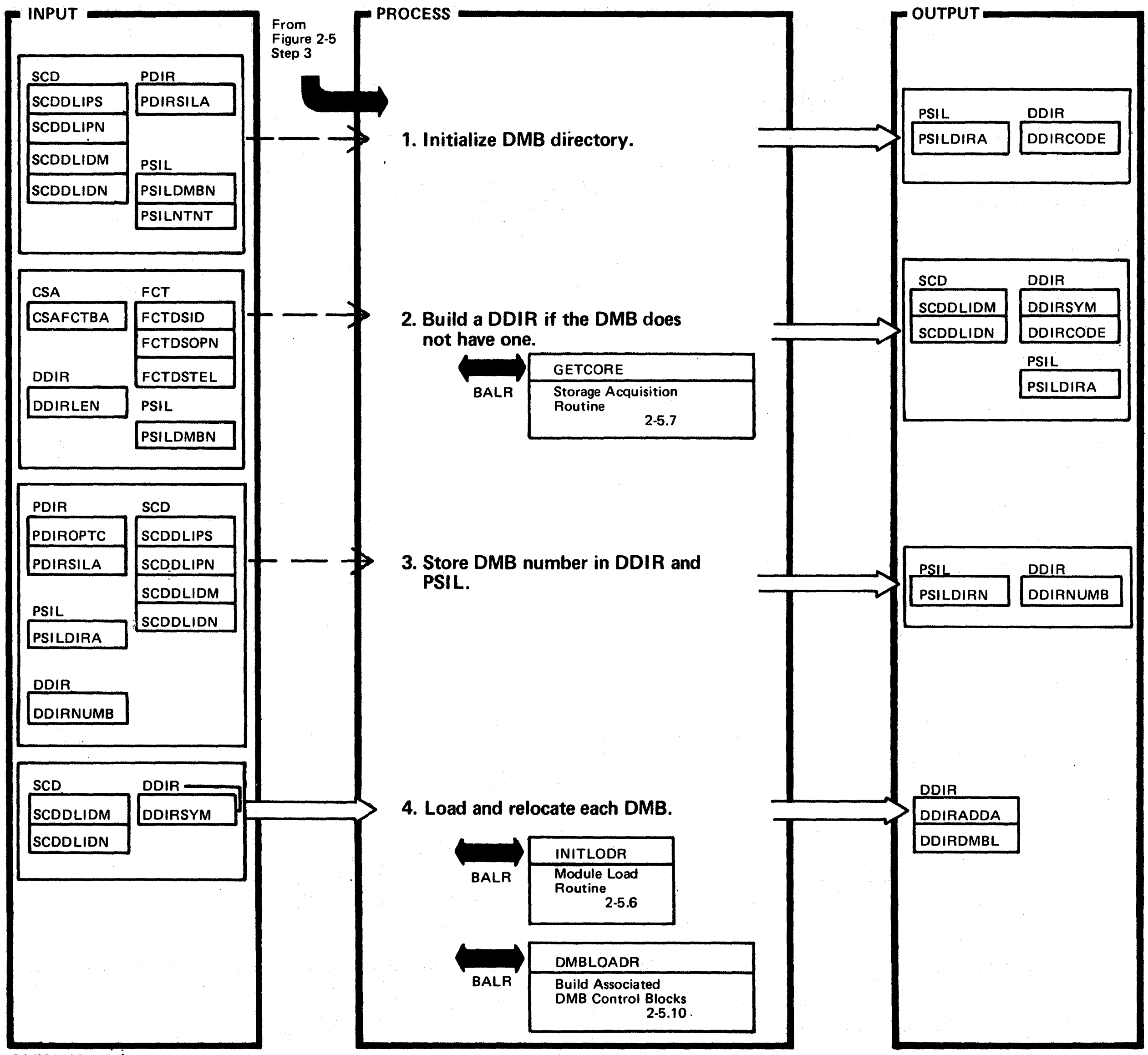
Figure 2-5.2. PSB Processing



Extended Description	Routine	Label
1. The PDIR address is located in the SCD and each local PSB is loaded temporarily, directly behind the dummy PST. The PSIL entries that indicate the DMBs which may be used by this PSB are loaded along with the PSB. They are appended in front of the PSB. If PSB initialization is successful, it will be moved up prior to completion of initialization.	PSBLOADL	PSBLOADL
Write message DLZ044I if the PSB is not found.		
2. Write message DLZ071I if the PSB is not version/modification 1.1 or later.		
3. All the SDBs for each PCB in a PSBLIST are tested for correct processings options. Indicators are set in the corresponding PDIR entry, PSIL entry, and the SCD to indicate intent. The corresponding SIL entry is found by using the offset value found in DSGDMBNO.		PDIRNPCB
Write message DLZ042I if a PSB accesses a HSAM DBD online.		

Extended Description	Routine	Label
Write message DLZ043I if load sensitivity is detected. A PSB contains a PCB with PROCOPT=L which is invalid online.		
4. The PSILs are moved from the temporary position and as they are moved, the size of each entry is increased by the size of PSILSEGD to allow for the segment intent bits mask copy. For program isolation, non-exclusive intent bits are translated to read-only to allow simultaneous scheduling of update-sensitive segments. All the DSGDMBNOs are adjusted to show the new offset. Also, each PSILLNGH is adjusted to show each new PSIL entry length as each entry is moved.		PDIRSMUV
Return is made to Step 1 to repeat this routine for PSB until there are no more.		PSBNXT

Figure 2-5.3. DMB Processing (Part 1 of 3)



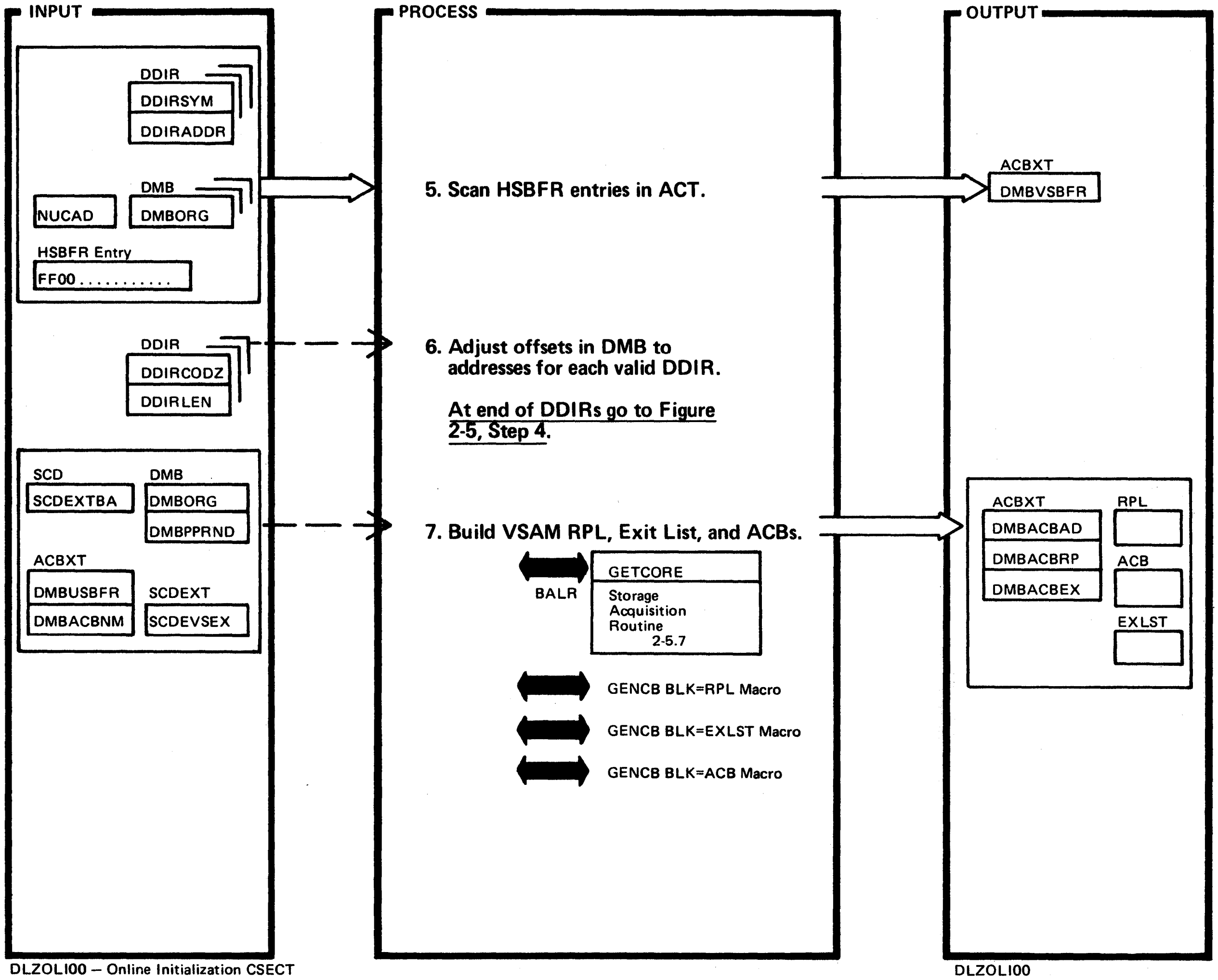
DLZOLI00 — Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
1. The PSILs are scanned for DMB names and a DDIR is created for each unique DMB encountered. The address of the DDIR replaces the respective dmbname in each PSIL.	DDIRINIT	DDIRINIT DDIRFOND
2. Write message DLZ045I if no CICS/VS FCT. Write message DLZ046I if no FCT entry existed matching the dbdname.		DDIRBLD
3. Write message DLZ049I if no valid DMBs are found.		DDIRNUML
4. Write message DLZ047I if DMB not in library. Write message DLZ072I if the DMB is not version 1.1 or later.		DMBLLUP

Extended Description	Routine	Label
Write message DLZ048I if the randomizing module is not found. Write message DLZ049I if no valid DMBs are found.		

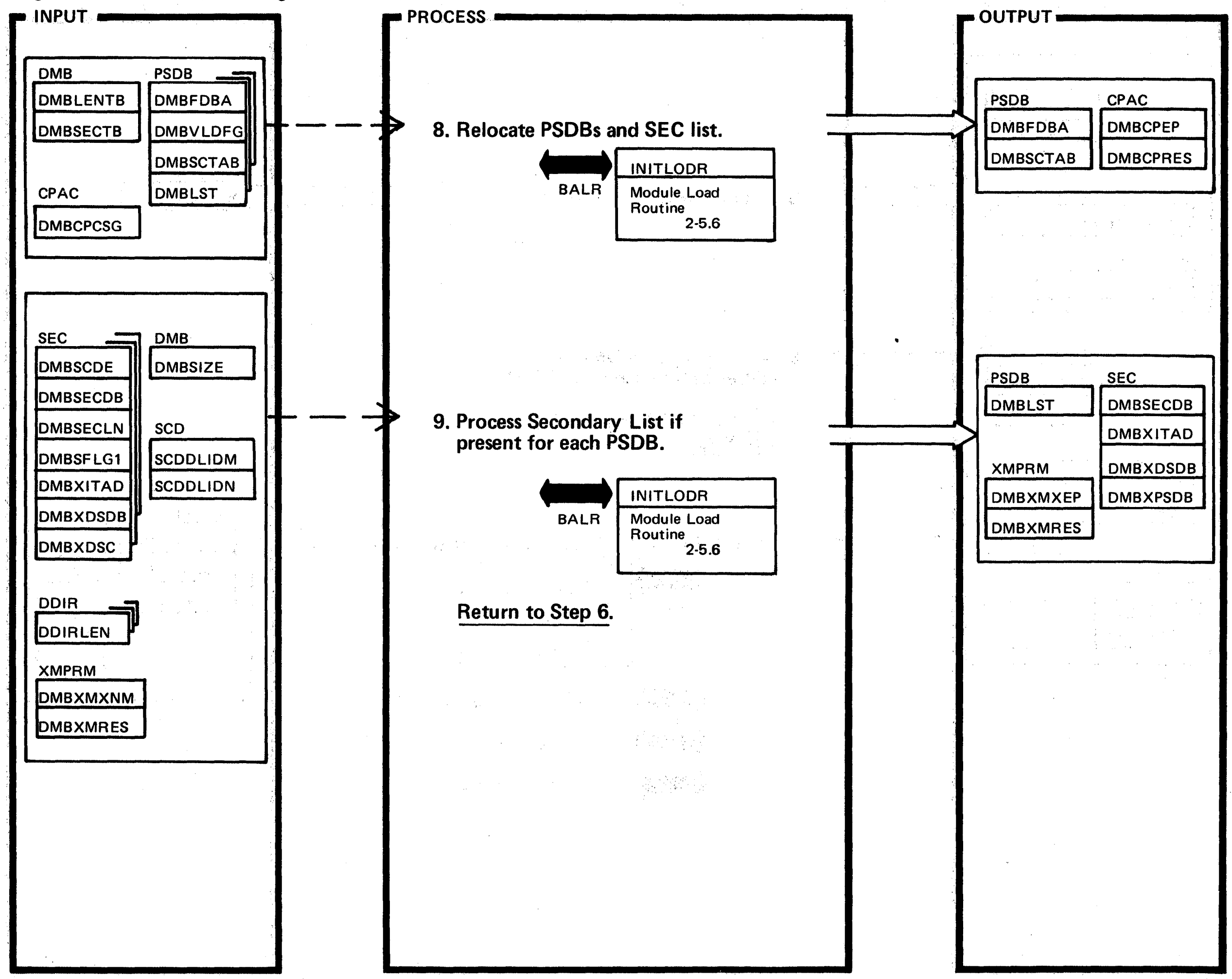
Figure 2-5.3. DMB Processing (Part 2 of 3)



Extended Description	Routine	Label
5. Write message DLZ029I if invalid DBDNAME in HSBFR statement. The number of index buffers and KSDS buffers in the HSBFR entry is moved to the ACB extension. If the organization is HISAM the number of ESDS buffers is moved to the second ACBXT. These values are used in building the VSAM ACBs (in Step 7).	DDIRINIT	CHKHSB
6.		DMBRLUP
7. If HISAM, two sets of control blocks will be built. Information obtained from HSBFR statements is used for the GENCB BLK=ACB BUFND=parameter and BUFNI parameter. If none was specified, the default of 3 index buffers and 2 data buffers is used.		DMBOFFAJ ACBADLUP

Extended Description	Routine	Label

Figure 2-5.3. DMB Processing (Part 3 of 3)



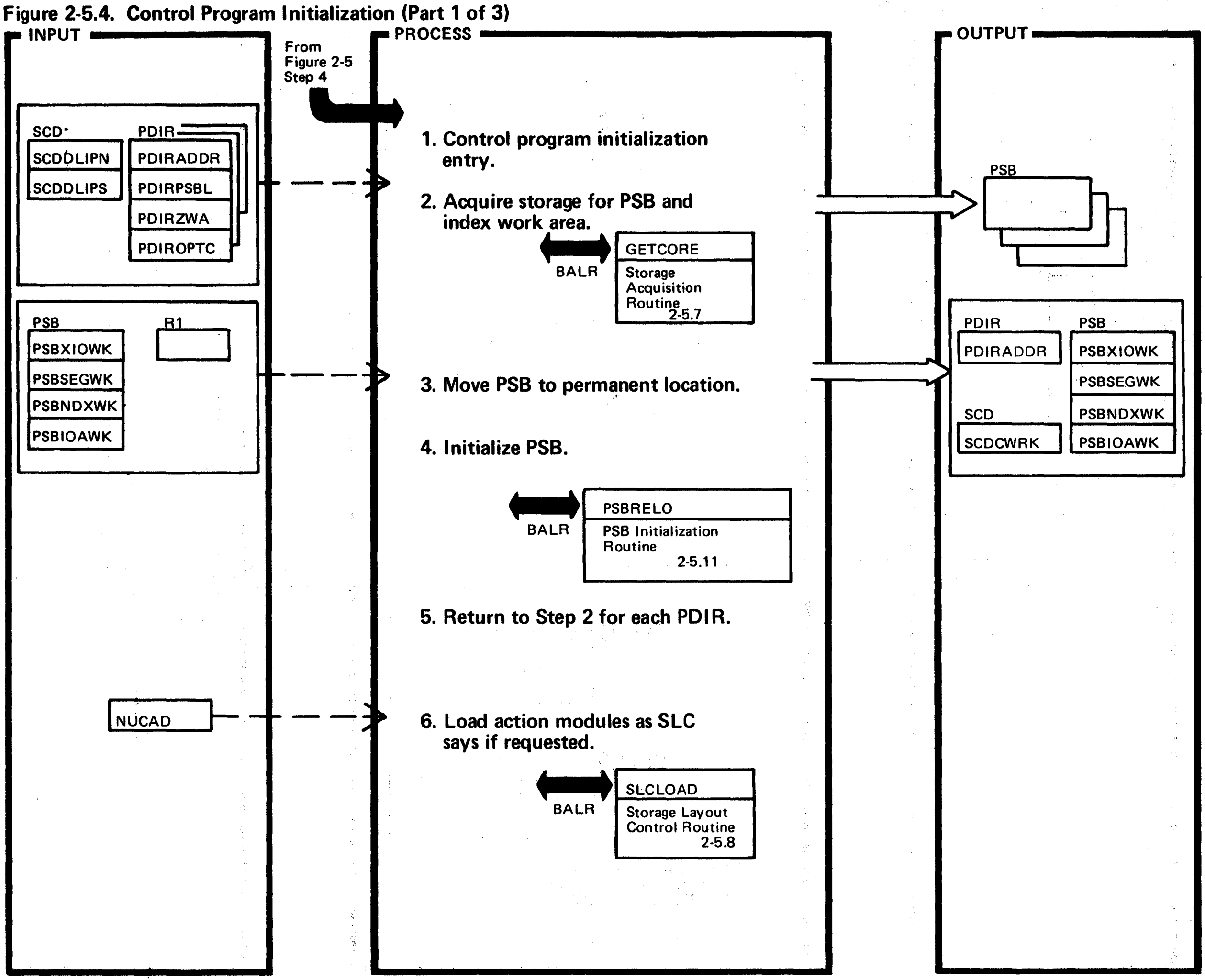
DLZOLI00 – Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
8. The segment compression modules for each PSDB is loaded if present. Write message DLZ073I if the compression module is not found.	DDIRINIT	PSDBROUT
9. If a secondary list is present, its code is tested, and referenced DMBs are resolved to DDIR pointers and placed in the list. If an index user exit routine is present it is loaded. Write message DLZ013I if the SEC makes an invalid DMB reference. Write message DLZ266I if there is an invalid secondary code. Write message DLZ074I if the indexing module is not found.		PROSEC

Extended Description	Routine	Label

Figure 2-5.4. Control Program Initialization (Part 1 of 3)



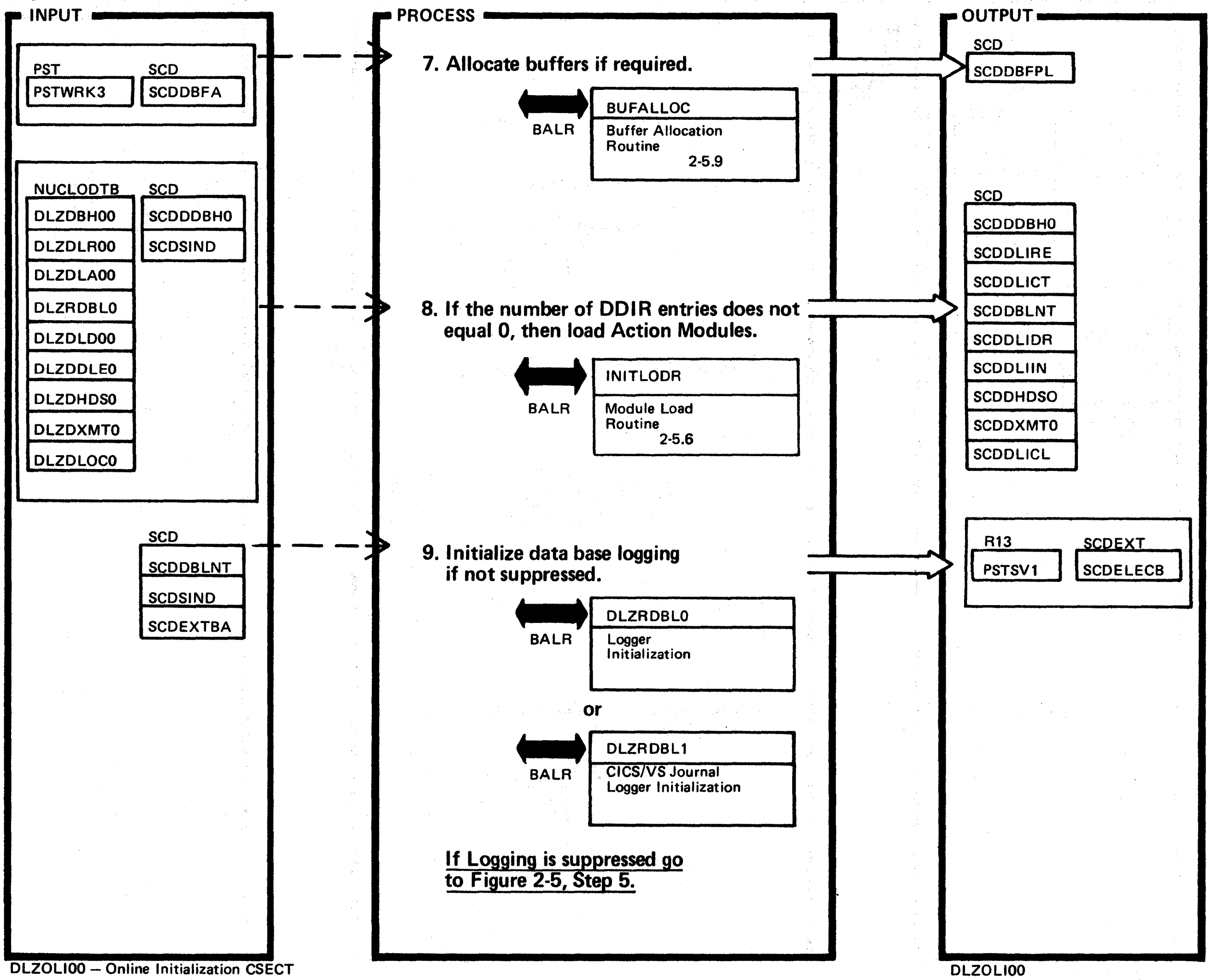
DLZOLI00 – Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
1.	DLZCPI00	DLZCPI00
2. The address of the storage acquired is used as the 'move-to' address in Step 3.		PCCORET PSBMLUP
3. Once the PSB is moved, its new address is stored in the PDIR and the old PSB address becomes the new code upper boundary address (upward core allocation starting address), SCDCWRK.		
4. Write message DLZ056I if PSB fails to initialize.		PSBNWA4
6. If there is an SLC name at the nucleus address +8, the SLCLOAD routine loads the action modules. Write message DLZ032A and terminate DL/I initialization if CICS/VS journalling is to be used but there is no system log entry (X'01') in the JCT.		DLILOAD NUCLDNJL

Extended Description	Routine	Label

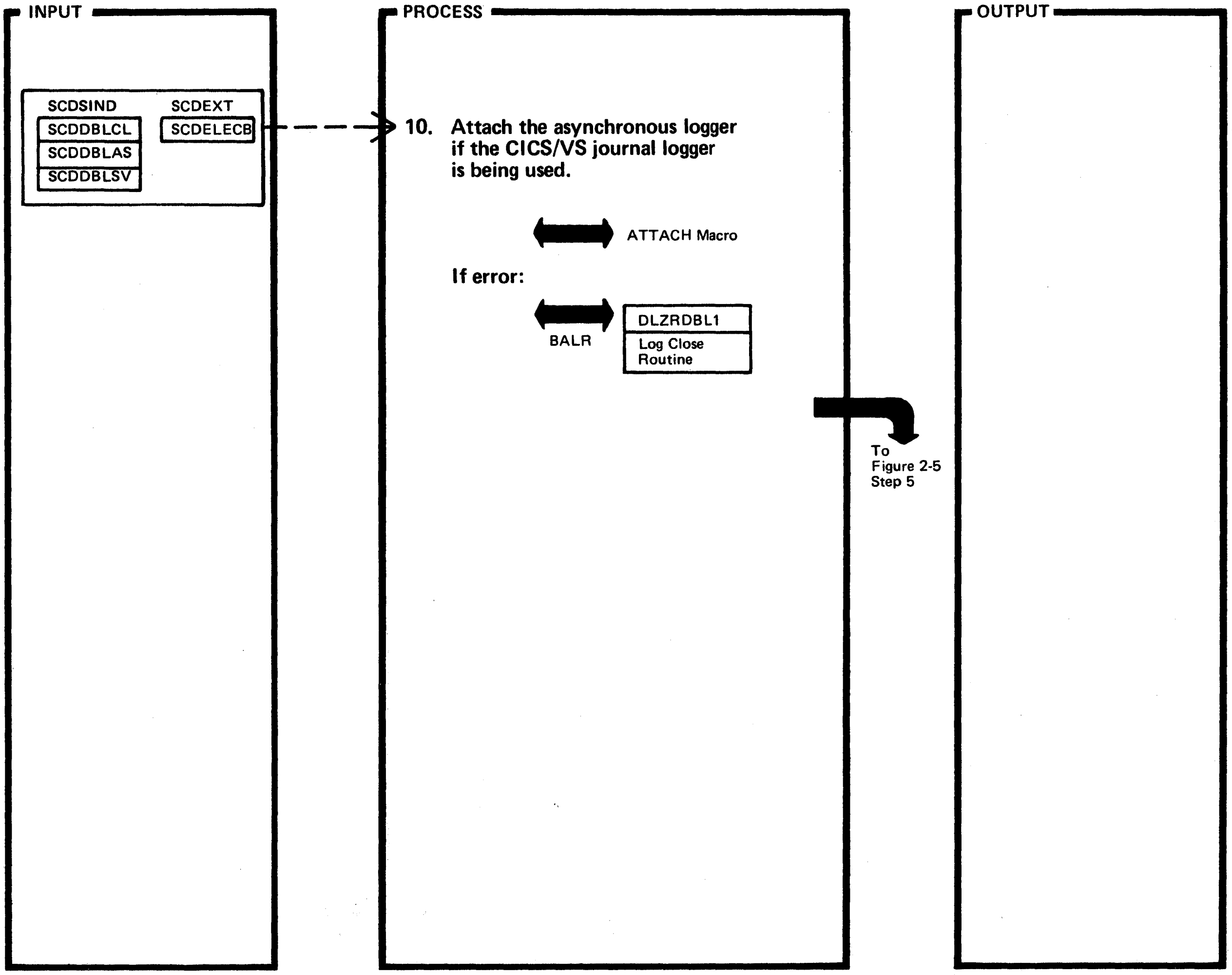
Figure 2-5.4. Control Program Initialization (Part 2 of 3)



Extended Description	Routine	Label
7. If there is not a buffer pool prefix address in the SCD indicating the buffers have already been allocated and the number of subpools required is not zero, go allocate the buffers.	DLZCPI00	BFRALOC
8. If CICS/VS journaling is supported, DLZRDBL0 in the NUCLODTB is replaced with DLZRDBL1.		DLILOAD
Any module already loaded by SLCLOAD will not be reloaded.		
Write message DLZ055I if a DL/I facility module is not found.		
9. SCDDBLNT is loaded with the entry point at the proper logger initialization routine (DLZRDBL0 or DLZRDBL1). If logging was suppressed SCDDBLNT will continue to point to a branch 14 for immediate return to caller.		NUCLODNX

Extended Description	Routine	Label

Figure 2-5.4. Control Program Initialization (Part 3 of 3)



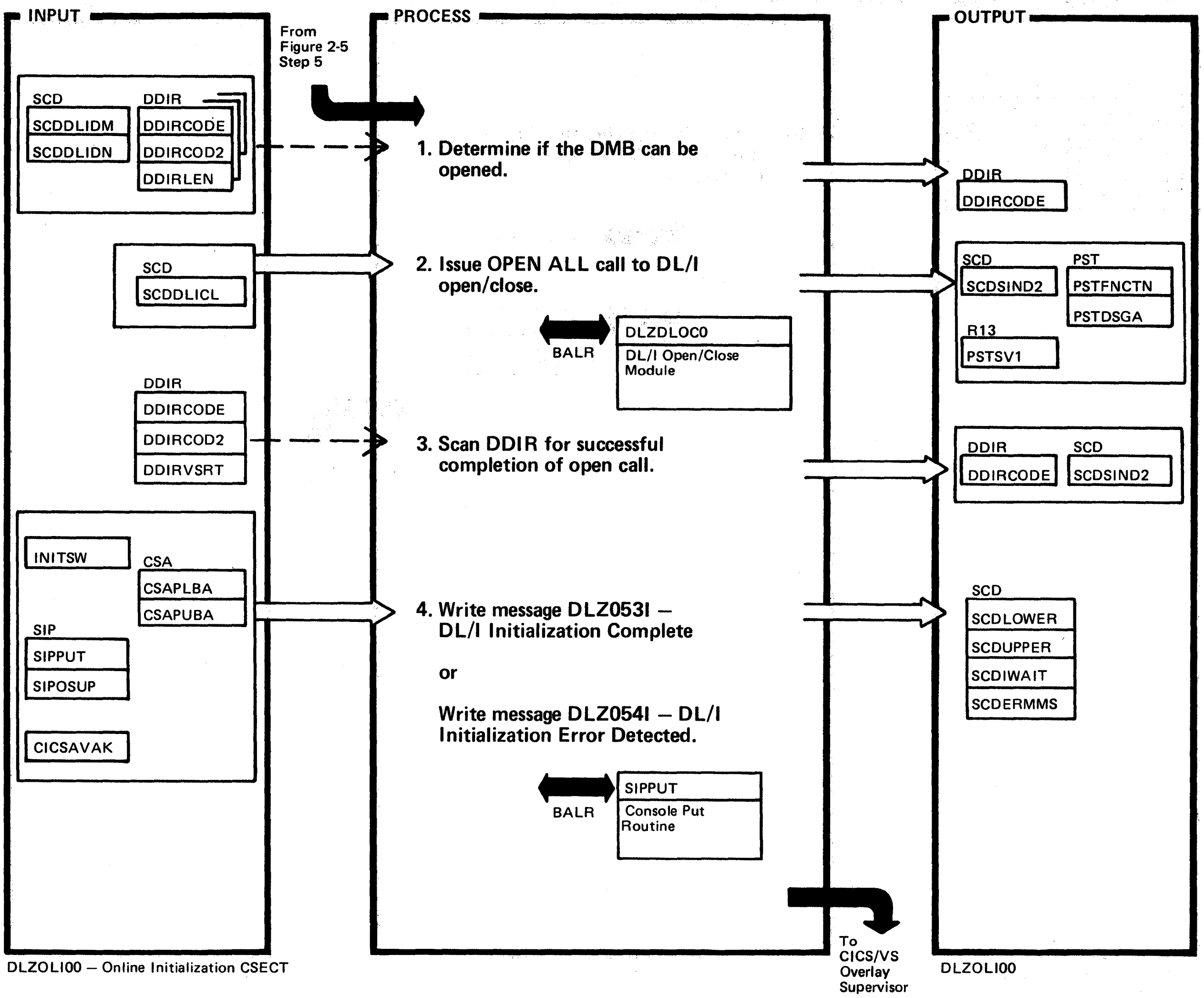
DLZOLI00 — Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
10. Write message DLZ006I if the asynchronous logger did not successfully attach and go close the log. The address list for the asynchronous portion of the database logger and its save area address are located in the database log load module just prior to the entry point. If the attach fails, the database log is closed and the system continues without log support	DLZCPI00	NUCLODNX

Extended Description	Routine	Label

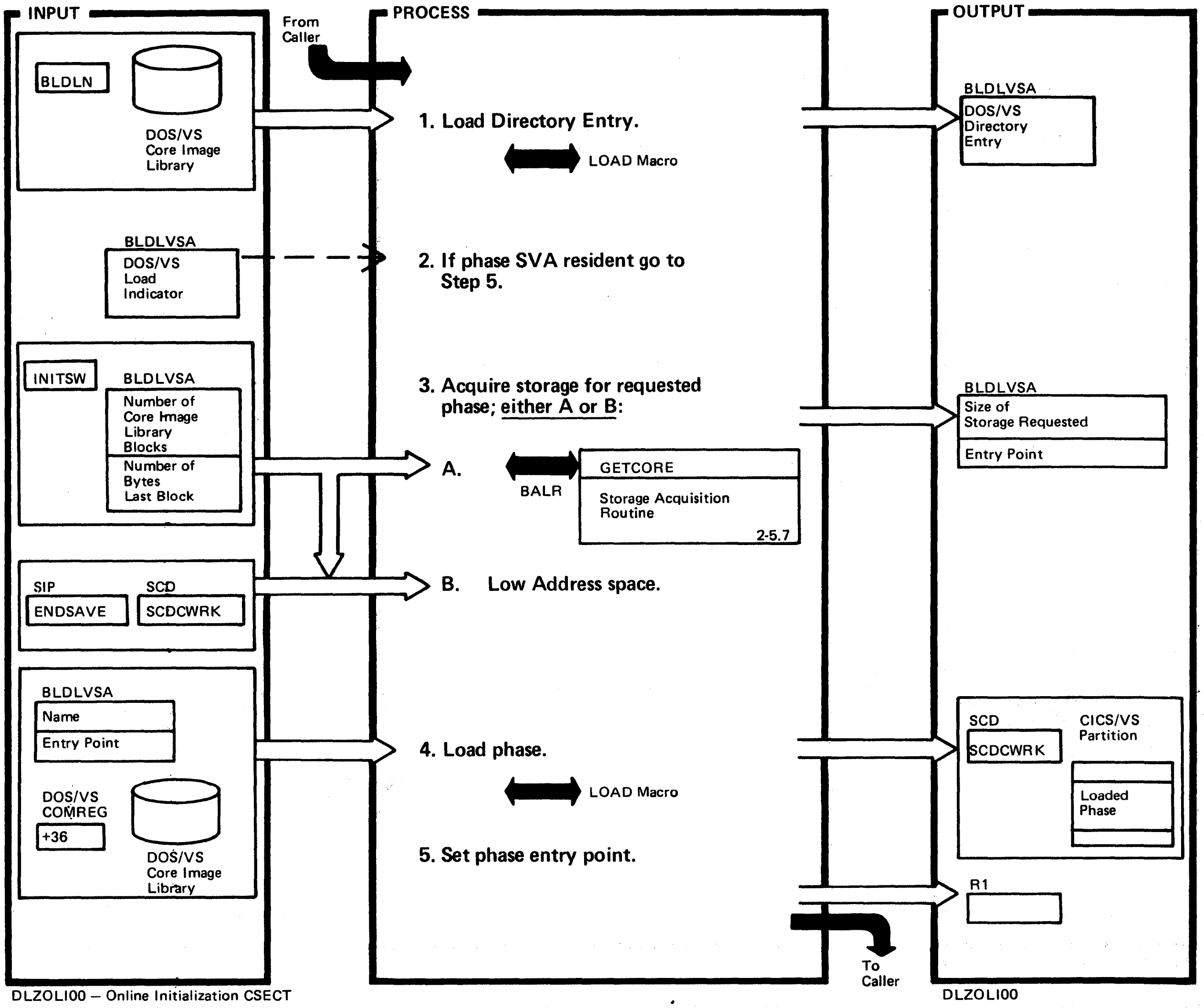
Figure 2-5.5. DMB Open Processing and Online Initialization Completion



Extended Description	Routine	Label
1. If all PSBs are remote, do not attempt OPEN.	DMBOPENA	DMBOPENA
2.		DMBSCNX
3. Write message DLZ057I if an open error occurred attempting to open a DMB.		DMBSCLP2
4. During the course of initialization an error can also cause a direct return to CICS/VS with message DLZ052I - Initialization Failed.		EXITOVL

Extended Description	Routine	Label

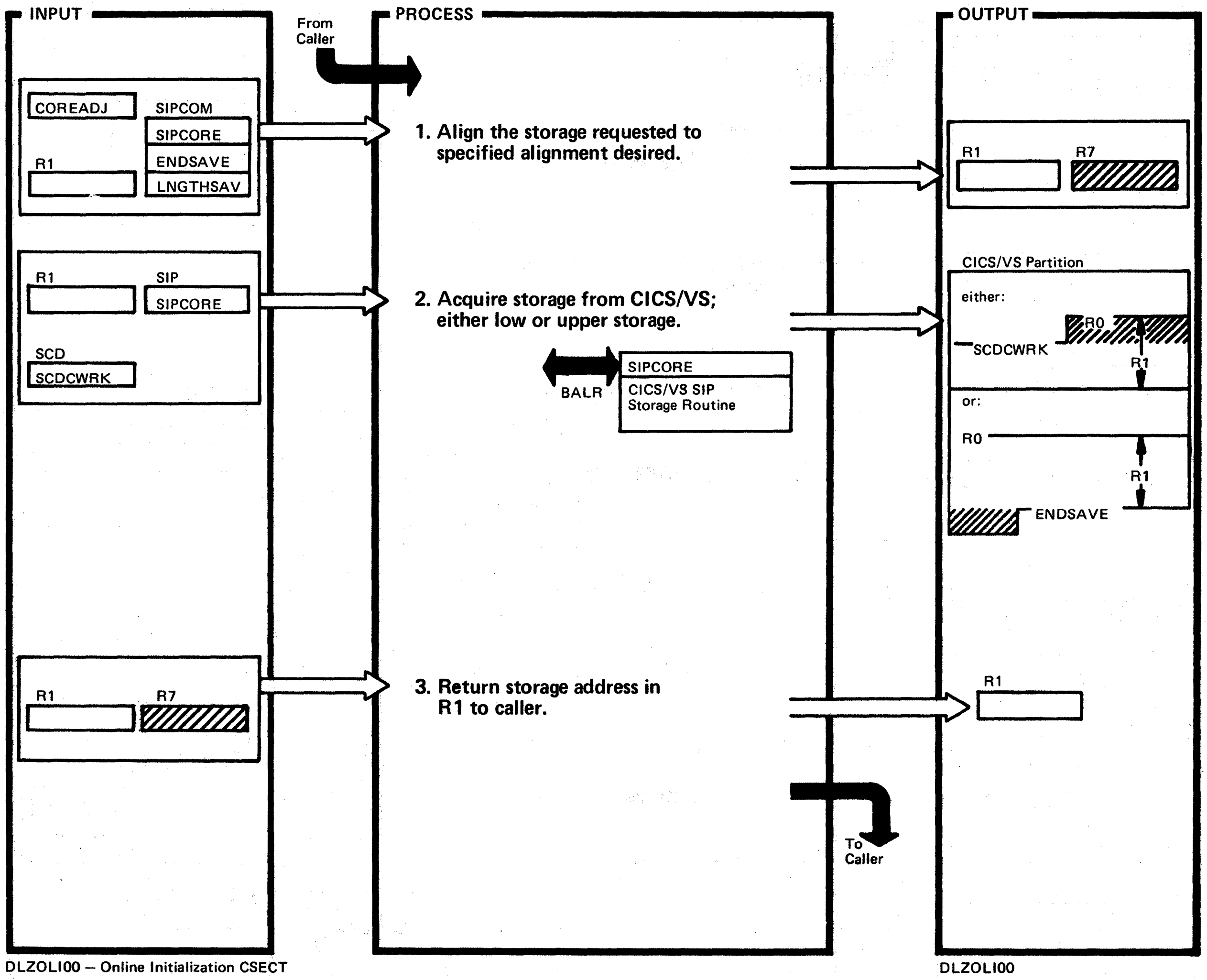
Figure 2-5.6. Module Load Routine.



Extended Description	Routine	Label
1. Caller passes requested phase name in a work field BLDLN. The output of the load call is a DOS/VS directory entry at BLDLVSA.	INITLODR	INITLODR
2.		BLDLFND
3. Amount of storage is determined by: Number of Library Block x 1024 + Number of Bytes in Last Block. INITSW indicates whether caller wants low address space or CICS/VS to acquire the space. B. Write message DLZ058I if insufficient core to initialize DL/I.		

Extended Description	Routine	Label
4. If low address space is requested by the caller SCDCWRK is used as the load address. After the load macro the end address of the module will be in the DOS/VS COMREG. SCDCWRK is updated with this end address to show the new low end of free storage.		LODRLOD
5. The phase entry point is passed back to the caller in register 1.		LODROK

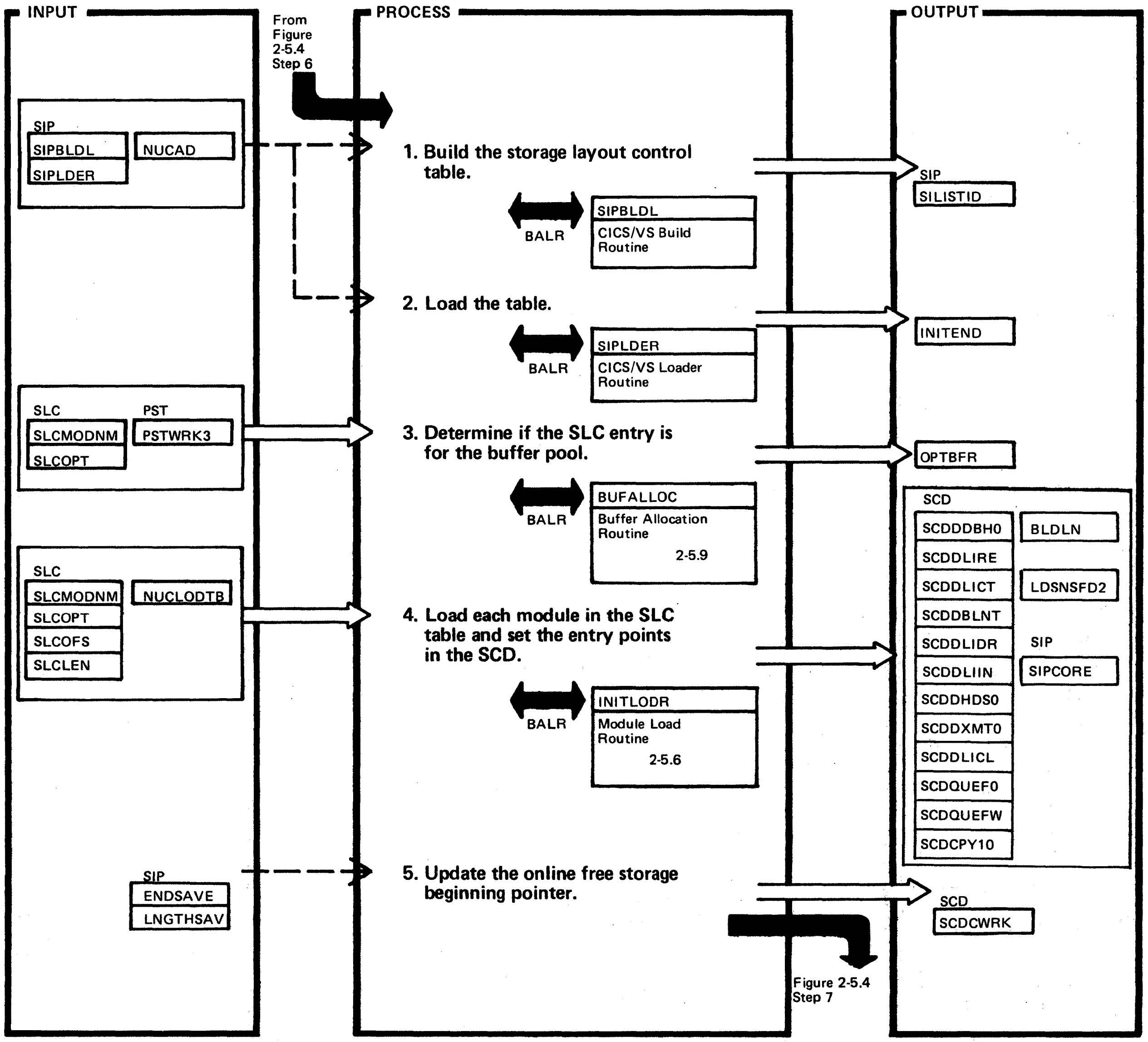
Figure 2-5.7. Storage Acquisition Routine



Extended Description	Routine	Label
1. If alignment is desired the number of additional bytes needed to align is calculated and put in R7. Register 1 is updated to show the new total number of bytes required.	GETCORE	GETCOREA GETCORE
2. Write message DLZ058I if insufficient storage to initialize DL/I. The SLCOPT flag byte (moved to SIPCORE from a storage layout control table entry) is used by CICS/VS GETMAIN routine. If we are doing our own alignment SLCOPT is examined to determine whether low or high storage was desired by the user.		BYCRALGN
3. The load point returned in R0 by Step 2 is adjusted by the additional bytes in R7 to get the requested alignment. This needs to be done only if low storage was acquired.		

Extended Description	Routine	Label

Figure 2-5.8. Storage Layout Control Routine



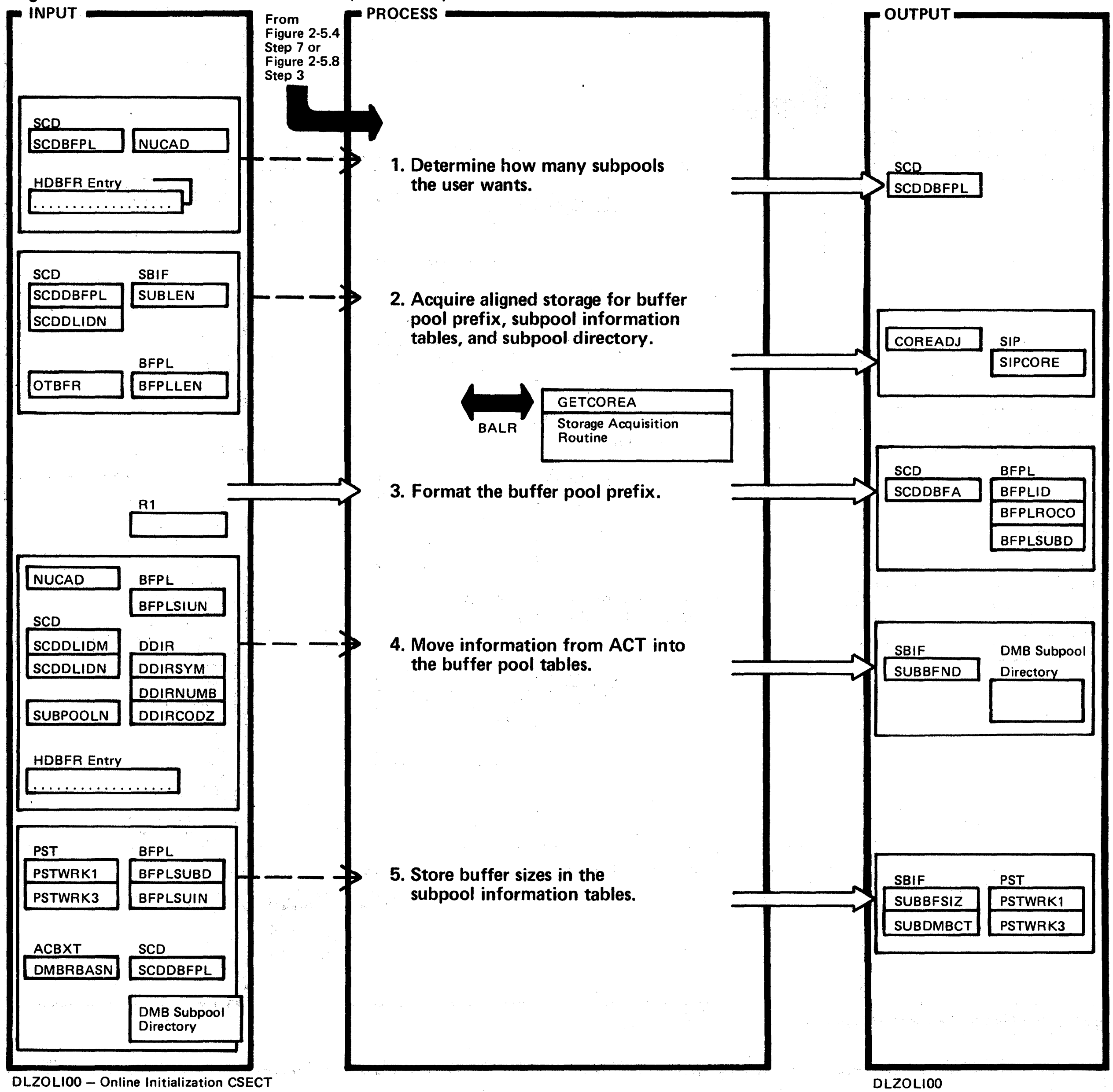
DLZOLI00 — Online Initialization CSECT

DLZOLI00

Extended Description	Routine	Label
1.	SLCLOAD	SLCLOAD
2. Write message DLZ030I if the loaded SLC table does not begin with *DLZSLC*. The table is loaded directly after module DLZOLI00.		
3. The user would have specified the DLZSLC statement with MODULE=BUFFER.		SLCLUP SLCBUF
4. As each module in the SLC table is loaded, the need to load flag is turned off so that upon return to Figure 2-5.4 these modules will not be reloaded. The SLCOPT for each module, as it is loaded, is moved to SIPCORE for use by the GETCORE routine.		SLCLUP

Extended Description	Routine	Label
5.		SLCXIT

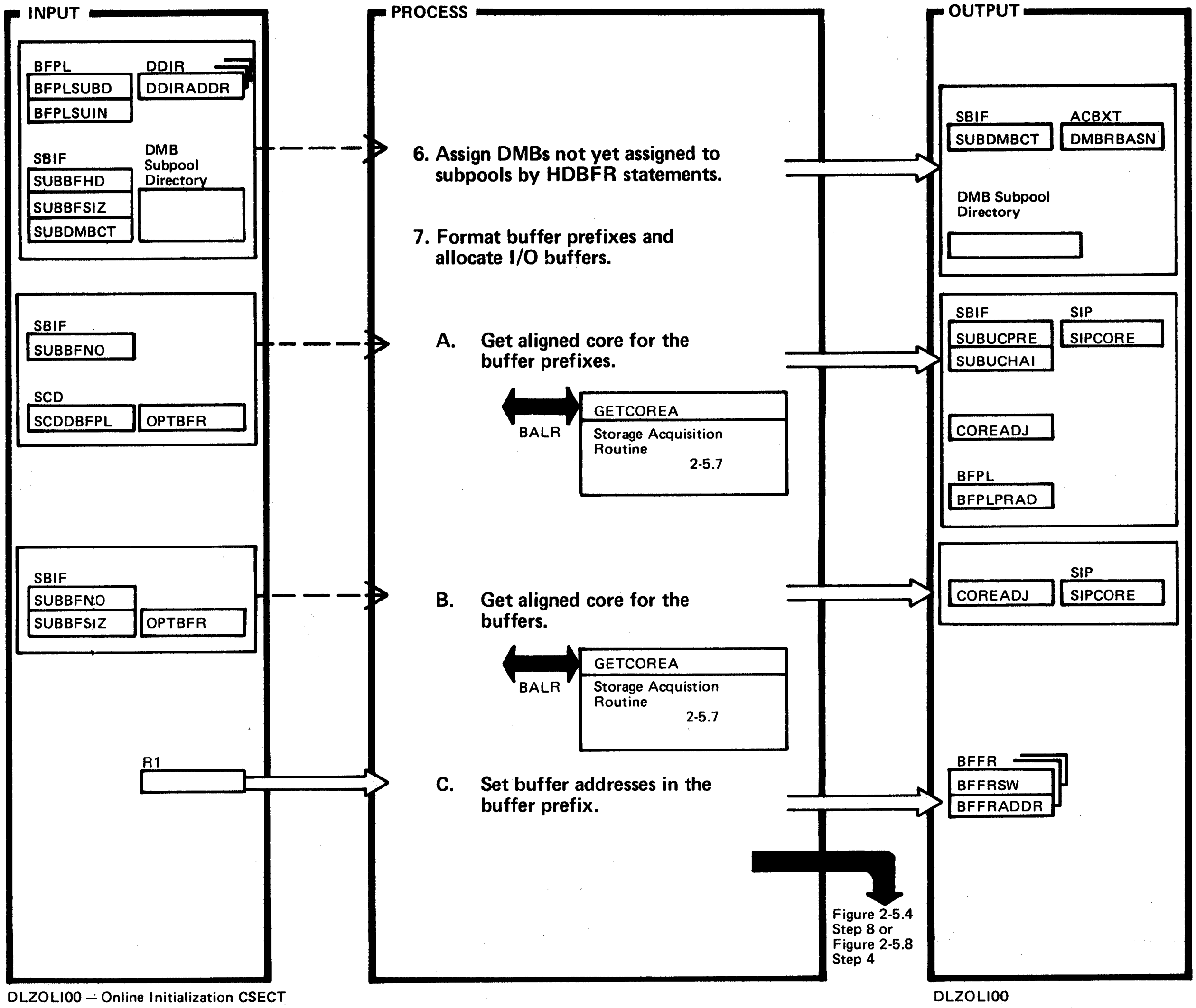
Figure 2-5.9. Buffer Allocation Routine (Part 1 of 2)



Extended Description	Routine	Label
1. Buffer allocation is done by this subroutine. The required number is set to the user specified total in the DLI parameter if the user number is smaller than required. Write message DLZ060I followed by DLZ061A if buffer pool allocation is missing or invalid.	BUFALLOC	BUFALLOC
2.		BFPREADY
4. Write message DLZ029I if there is an invalid DBDNAME in HDBFR entry.		SCANHD

Extended Description	Routine	Label
5. At this point the size of the subpools are determined. They are allocated, largest first, until the specified number is exhausted. Remaining DMBs requiring subpools are assigned evenly across all existing subpools. If the user specified more subpools than necessary, an additional pool of 512 buffer size is allocated for delete workspace.		PRPEND
The subpool sizes are sorted so that the largest subpool appears first in the subpool information table.		SUBTSHFL

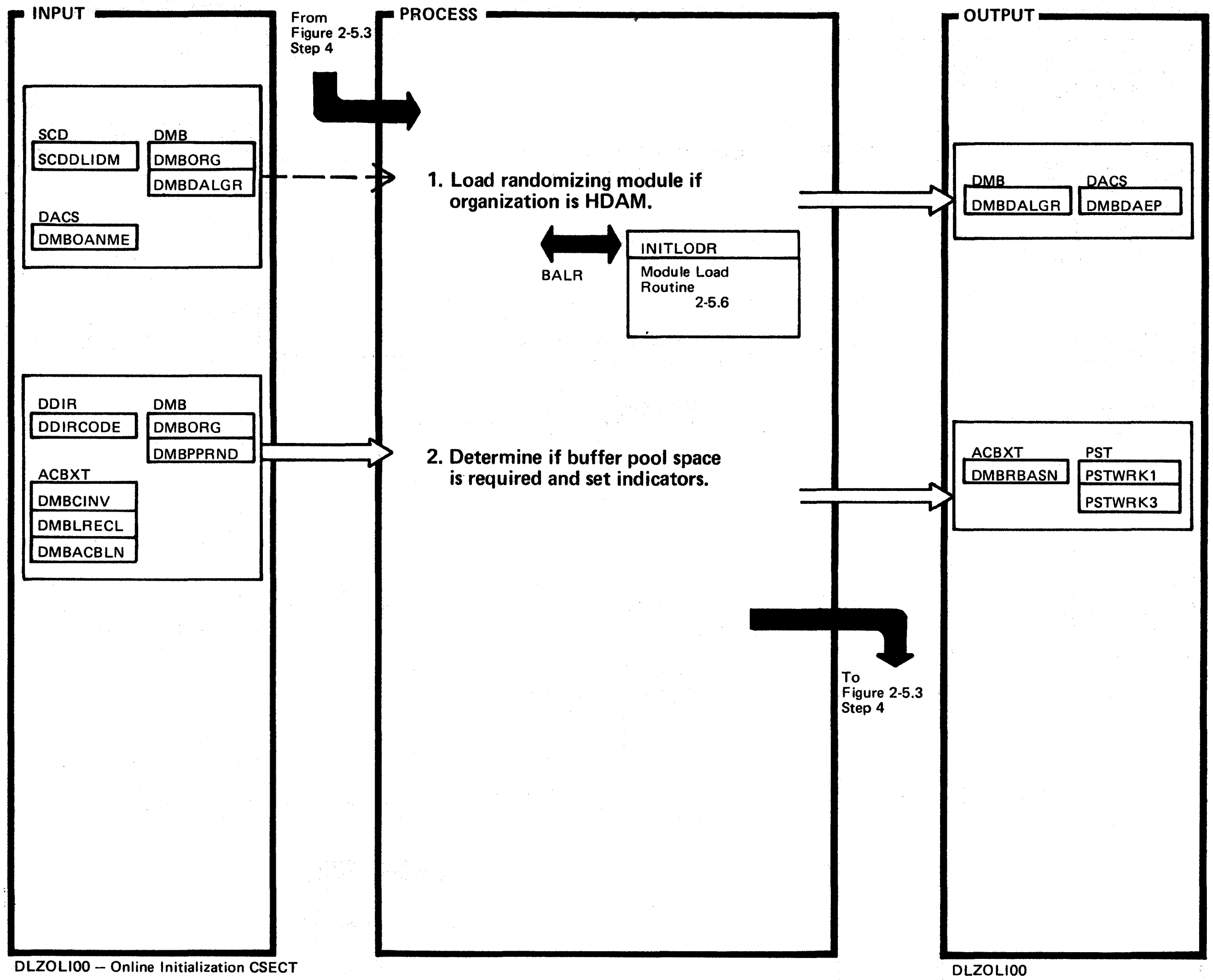
Figure 2-5.9. Buffer Allocation Routine (Part 2 of 2)



Extended Description	Routine	Label
6. Assign DMBs by corresponding control interval sizes. Each DMB is assigned by placing its DDIR position pointer into the subpool directory.	BUFALLOC	GREATPRO
It is possible to get message DLZ262I if there is a buffer allocation logic error.		
7. The user specified number of buffers is allocated per pool; default is 32.		BFRINIT
A.		ACCLOOP
B.		BFFRFRMT
C.		BFRSPLUP

Extended Description	Routine	Label

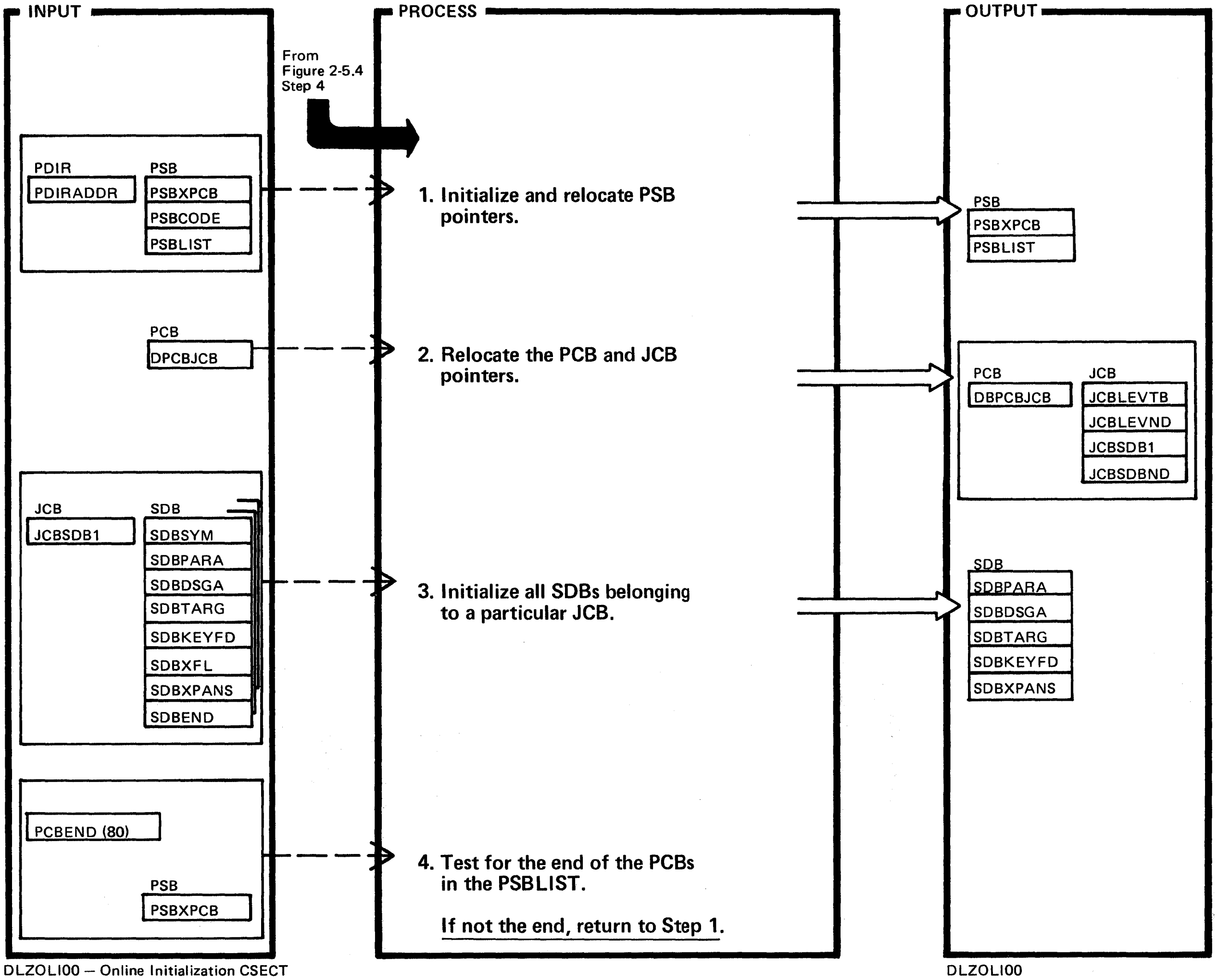
2-5.10. Build Associated DMB Control Blocks



Extended Description	Routine	Label
1. Before loading the randomizer a check is made with all currently loaded randomizers. If one with the same name as the one we are loading is found, the entry point is resolved and the actual load is bypassed.	DMBLOADR	DMBLOADR
If the randomizing module is not found, the DDIR is updated to show DMB initialization failed.		
2. If buffer pool space is required, the size of each control interval rounded to the next multiple of 512 is indicated in PSTWRK1 for later allocation of the buffer pool.		GETBUFRS

Extended Description	Routine	Label

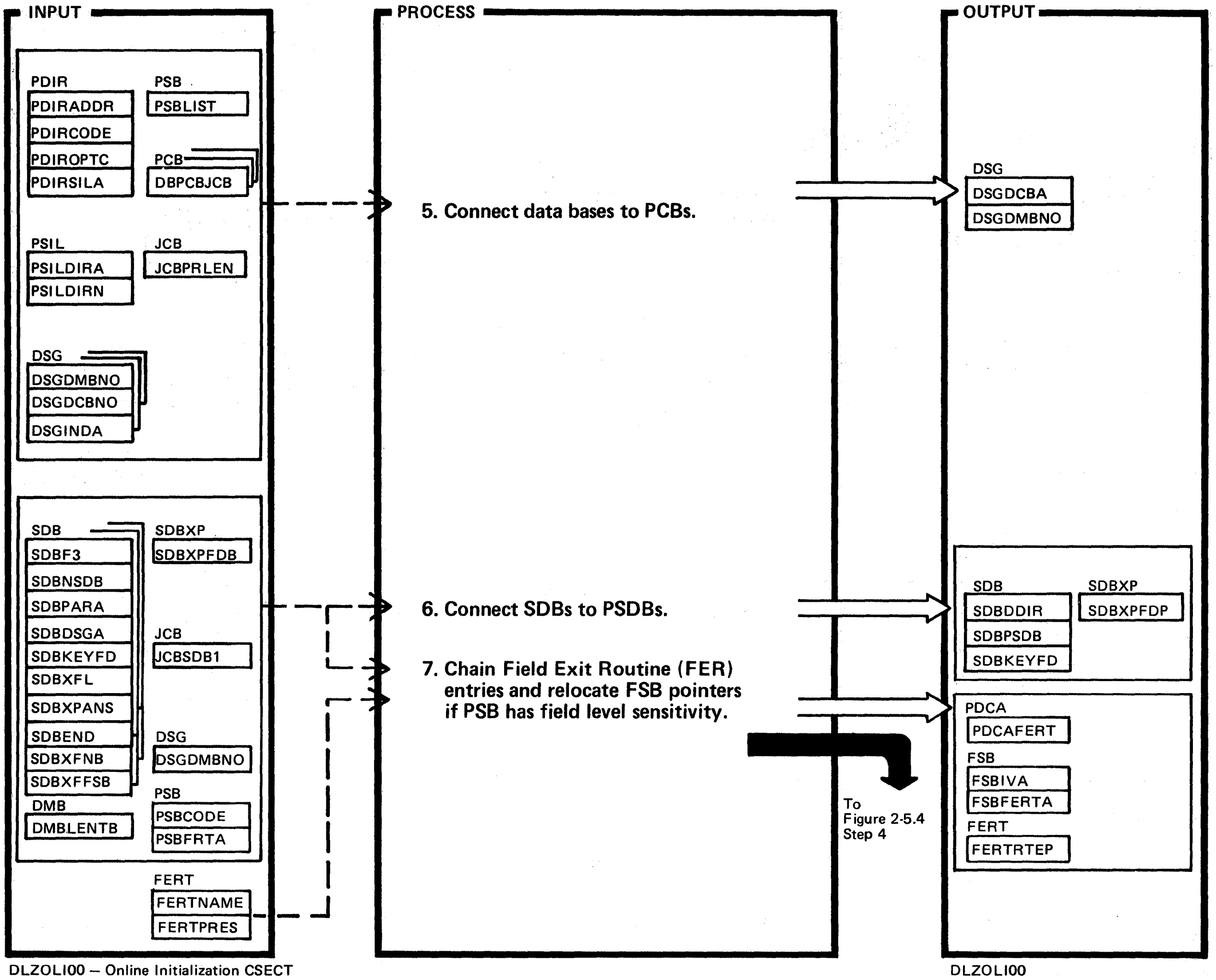
Figure 2-5.11. PSB Initialization Routine (Part 1 of 2)



Extended Description	Routine	Label
1.	PSBRELO	PSBRELO PCBRLLUP
2.		PCBPLIP
3.		SDBRELO
4. The pointer to the PSBLIST is bumped to the next PCB pointer entry and processing returns to Step 1 if we are not at the last PCB. If the index PCB exists and has not been relocated return to Step 2.		NPCBCK

Extended Description	Routine	Label

Figure 2-5.11. PSB Initialization Routine (Part 2 of 2)



Extended Description	Routine	Label
5. For each DSG belonging to a PCB, the offset to the corresponding PSIL is found by the offset in DSGDMBNO. The DMB number at the corresponding DMB to this PSIL is then moved to DSGDMBNO. Thus, data bases specifically used for this PCB are connected.	PSBRELO	PCBROUT
7. Return to Step 4 to process the next PCB in the PSBLIST when there are no more SDBs.		CONSDBS

Extended Description	Routine	Label

Figure 2-6. Online Nucleus (Overview)

INPUT

PROCESS

1. DL/I pre-scheduling and PSB scheduling routines. (See Figure 2-6.1)
2. System abnormal and normal termination routine. (See Figure 2-6.2)
3. Task termination routine. (See Figure 2-6.3)
4. Start-of-task record writer. (See Figure 2-6.4)
5. Sync-point record writer. (See Figure 2-6.5)
6. Online program request handler. (See Figure 2-6.6)
7. Remote scheduling call interface. (See Figure 2-6.15)
8. Remote data base call interface. (See Figure 2-6.16)
9. Remote term call interface. (See Figure 2-6.17)
10. Online error message routine. (See Figure 2-6.7)
11. Online wait routine. (See Figure 2-6.8)
12. VSAM asynchronous error processor. (See Figure 2-6.9)

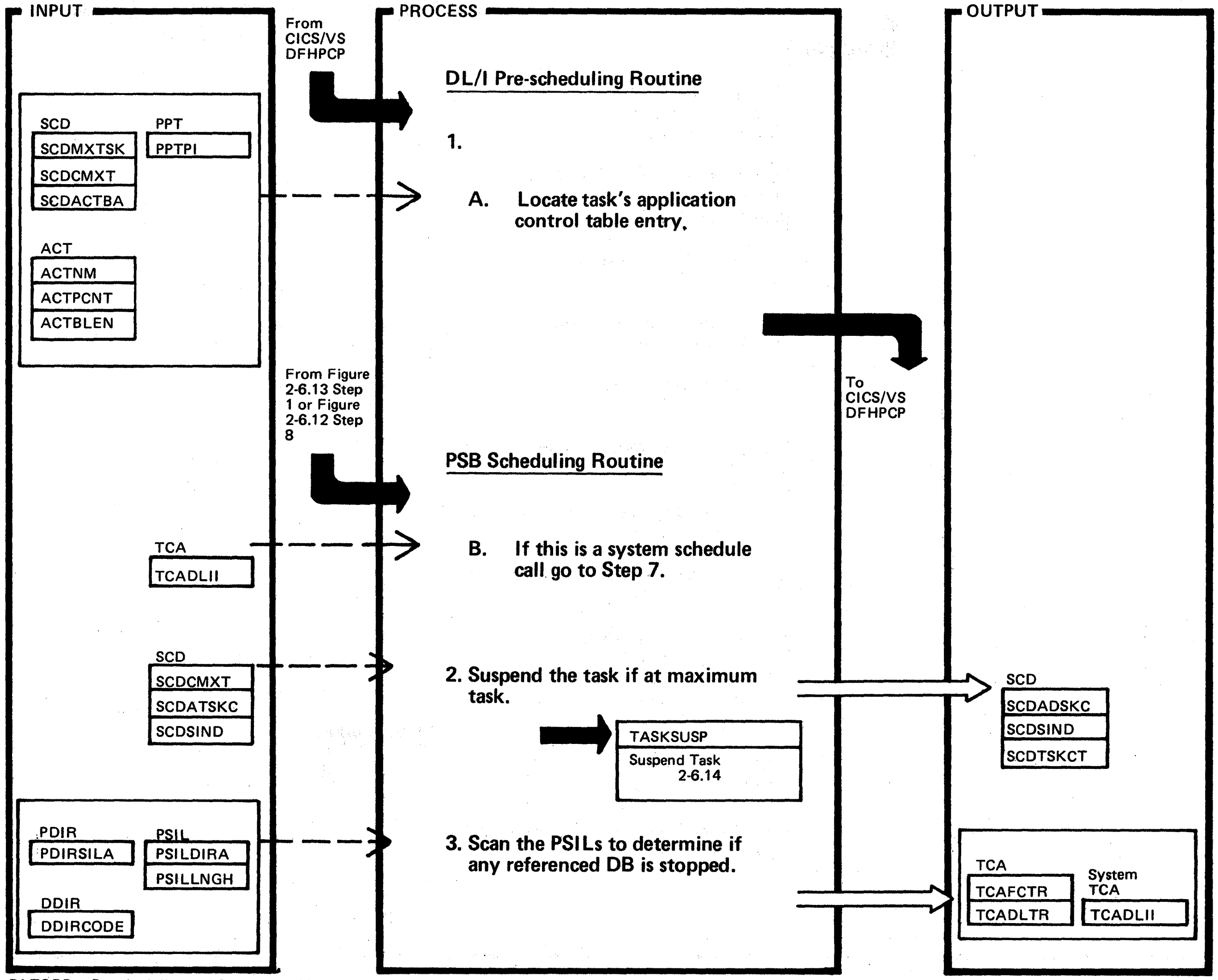
OUTPUT

DLZODP

Extended Description	Routine	Label
1.	DLZODP00	
2.	DLZODP03 DLZODP02	
3.	DLZODP01	
4.	DLZODP04	
5.	DLZODP05	
6.	DLZPRHO0	
7.	DLZISC00	
8.	DLZISC01	
9.	DLZISC02	
10.	DLZERMSG	
11.	DLZOWAIT	
12.	DLZOVSEX	

Extended Description	Routine	Label

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 1 of 7)



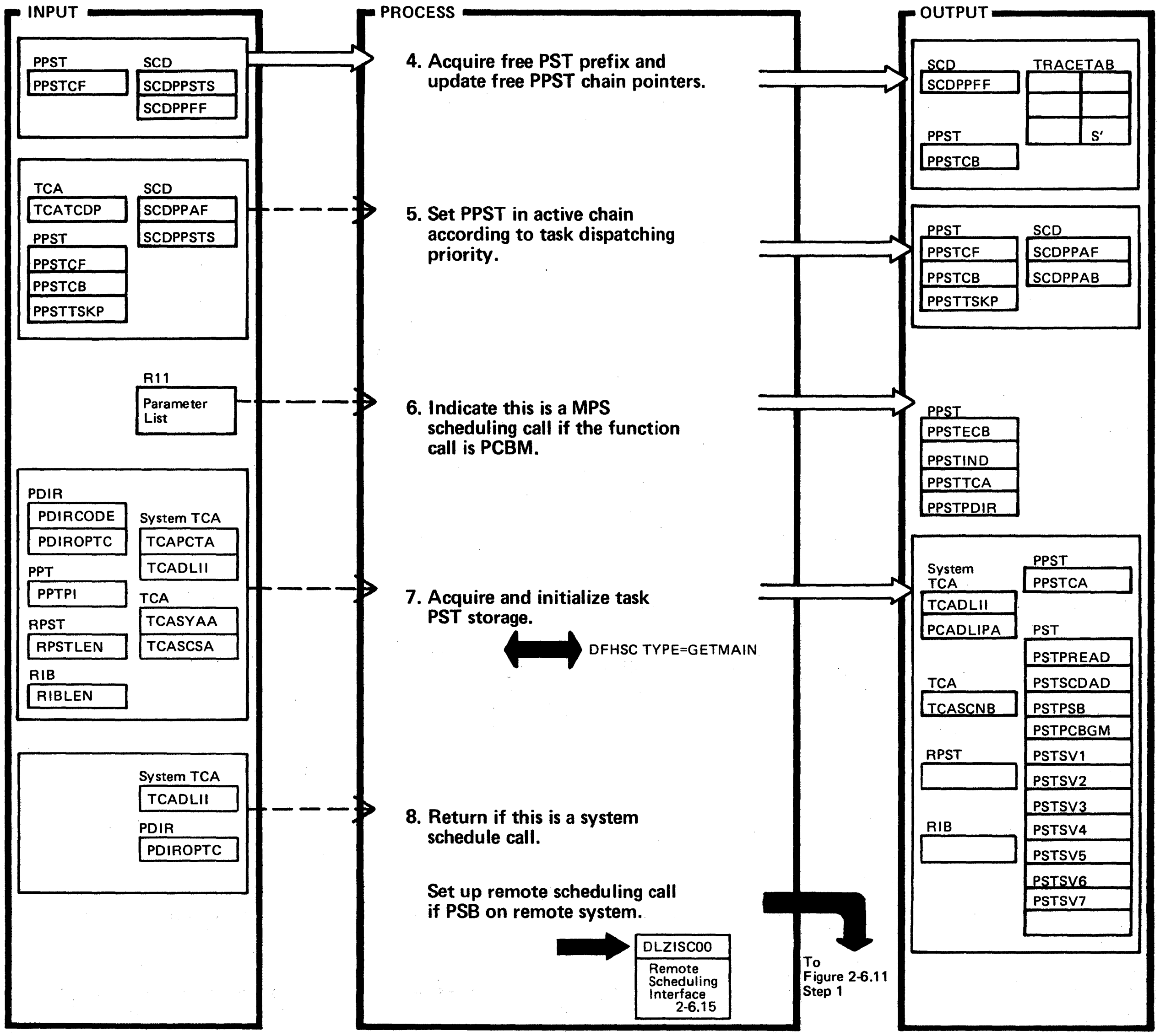
DLZODP — Prescheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
1. A. Nucleus identifier (*DLZNUCXX*0140) and module identifier (DLZODPvrnp) are defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied. This step checks the authorization of the CICS application program to use DL/I. If the program name is not located in the ACT an error indicator is turned on in the TCA (TCADLISE). If the ACT search is successful the ACT entry address is placed in the system TCA and DL/I scheduling initiated indicator (TCADLISI) is turned on. If trace is enabled an initial scheduling trace entry with the ID=X'F8', type of request=X'D0', is made showing the current and absolute maximum task counters and the PPT address.	DLZODP	DLZODP DLZODP00

Extended Description	Routine	Label
1. B. TCADLPAS indicates a system schedule call (PCB,SYSTEMDL).		TASKSCHD DLZSCHDL
3. Indicate a scheduling error and return to caller if the DB is stopped.		CHKLOP1
4. There is a trace table area within this module. Each entry is 2 bytes in length and the last entry is the newest. The PPST number of the acquired PPST is put in the trace table at this time along with an 'S' to indicate a scheduling entry.		

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 2 of 7)



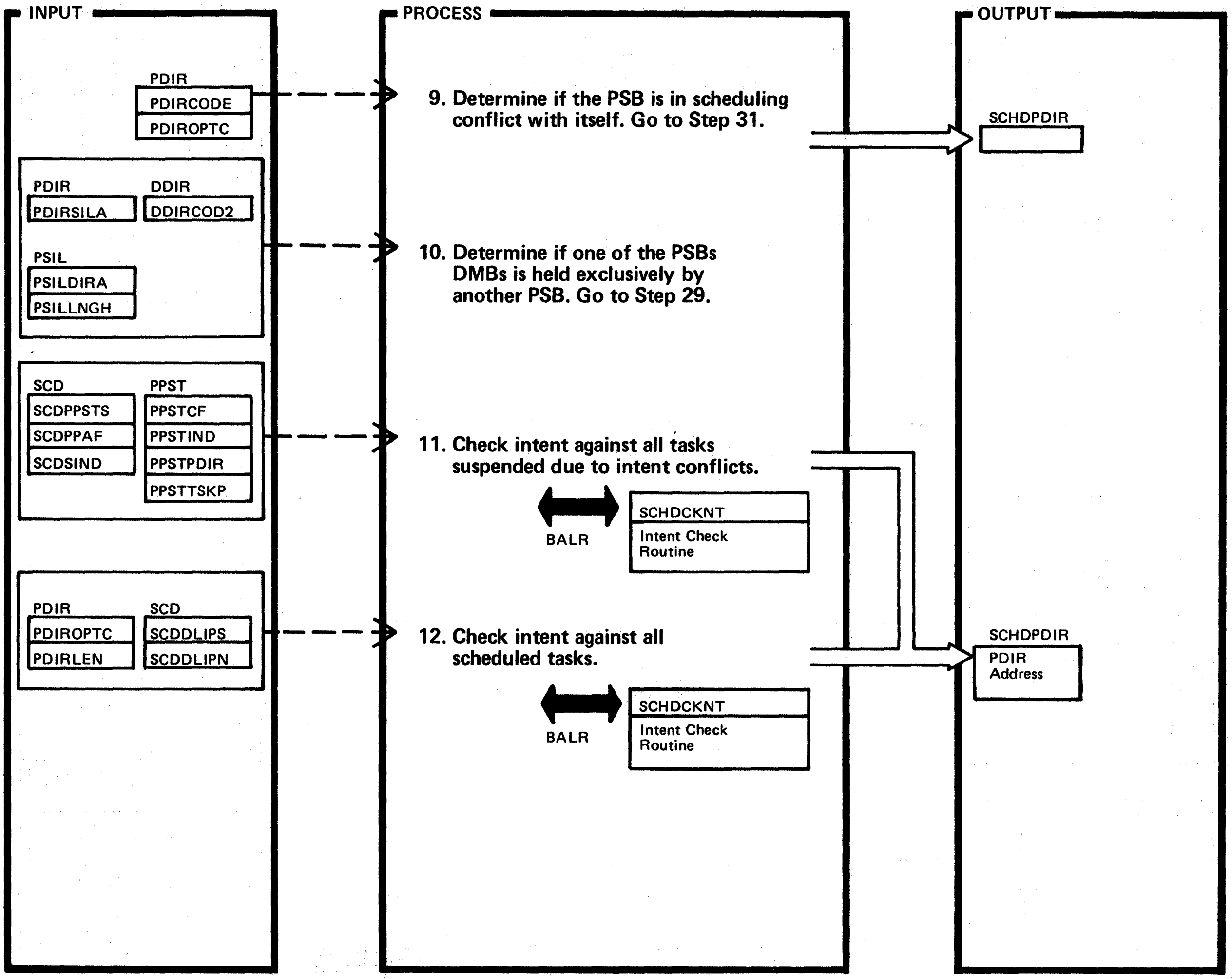
DLZODP — Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
5.	DLZODP	TASKPPAC
6. Set bits in PPST (PPSTMPS and PPSTMSDL) to indicate task is an MPS job. The parameter list was initialized by GETPSBN before calling this resource scheduling routine.		TASKPABV
7. If PSB to be scheduled is on a remote system, acquire storage for RPST and RIB also.		TASKPSTG
8. TCADLPAS indicates a system schedule call (PCB,SYSTEMDL). PDIRREM indicates PSB is remote.		TASKPSTL

Extended Description	Routine	Label

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 3 of 7)



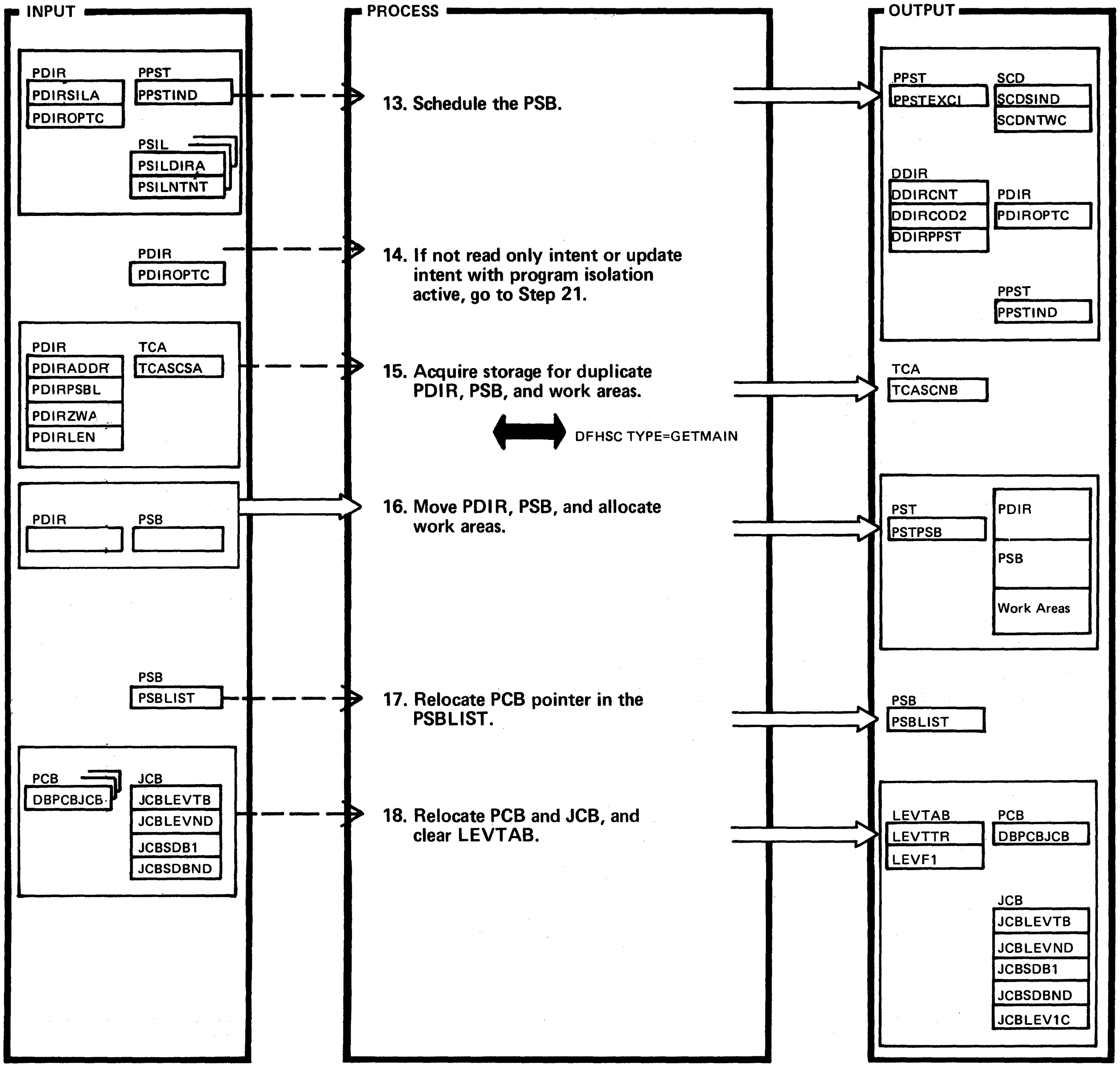
DLZODP — Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
9. If PSB is in use and is update sensitive or exclusive intent, go to Step 32 and wait for this task. If program isolation is active skip checks for update sensitivity conflicts.	DLZODP	SCHDNTCK
10. If any referenced DB is stopped, return a scheduling error to caller.		SCHDCKSI
11. Determine if one of the PSBs segments is being updated by another PSB that is waiting for intent and is at the same or higher task dispatching priority. The PDIR address of the task being checked is passed to the intent check routine at SCHDPDIR.		SCHDFCLP
12.		SCHDPACT

Extended Description	Routine	Label

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 4 of 7)



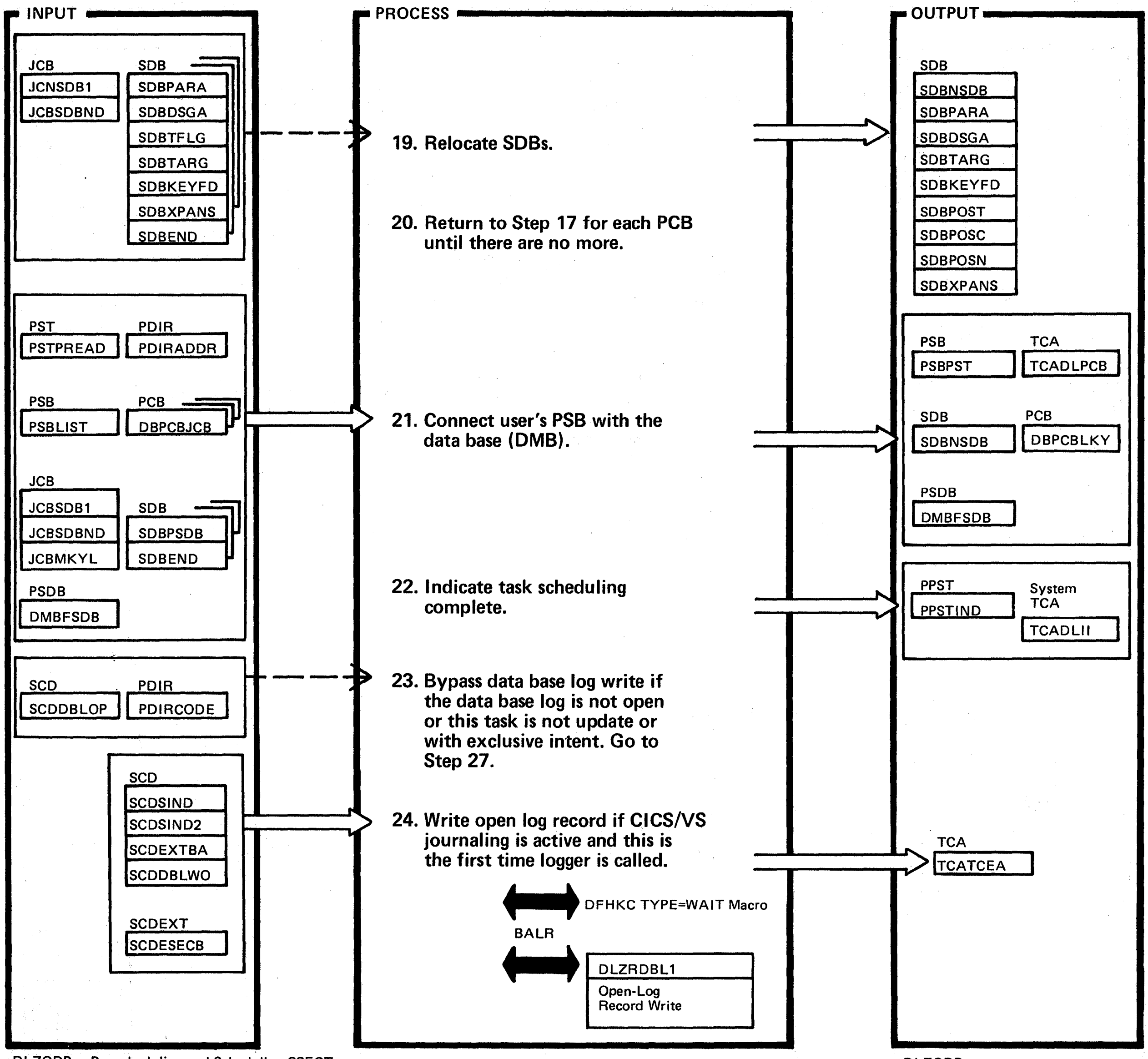
DLZODP – Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
13. The waiting-for-intent bit (PPSTSI) is turned off it it is on and the wait count is decremented (SCDNTWC). Turn on DDIREXCL if the PSB requires DB exclusive control. Turn on the PSB scheduled flag (PDIRSCHD).	DLZODP	SCHDESCD
14. If the PSB scheduled flag (PDIRSCHD) is already on, it must be read only.		
15. Task suspend is possible because of unconditional CICS/VS GETMAIN.		TASKDUPP
17.		SCHDDPL
18.		SCHDDPGO

Extended Description	Routine	Label

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 5 of 7)



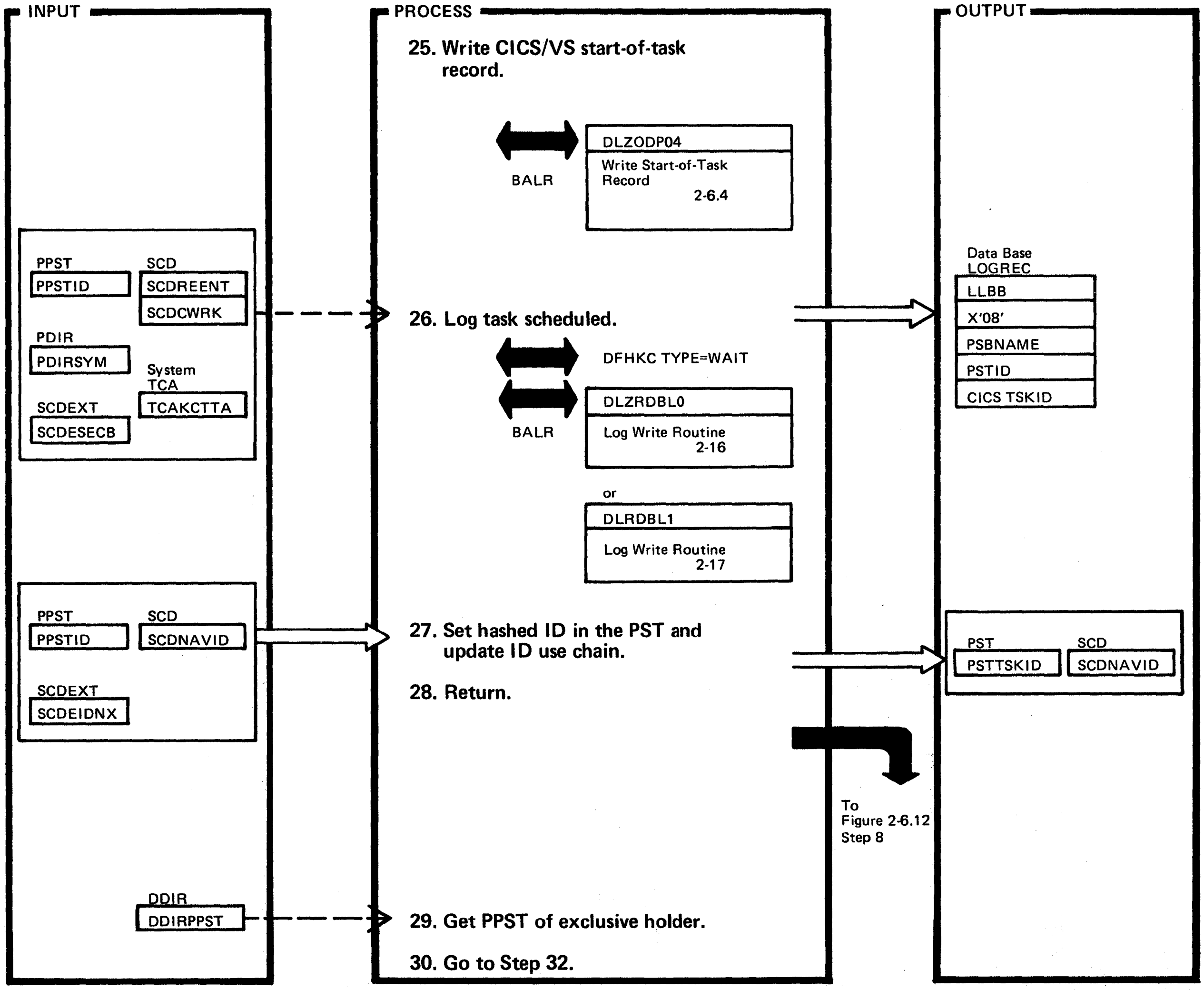
DLZODP → Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
19.	DLZODP	SCHDSDBL
20.		SCHDSDBN
21. Each PCB is scanned and the SDBs (sensitive segment types for each PCB) are placed in the SDB chain of the appropriate PSDB. DMBFSDB points to the SDB of the first PCB sensitive to this PSDB and SDBNSDB points to a SDB within the next PCB sensitive to the same PSDB. Move maximum key length for use by intersystem communication (ISC) transformer in calculating size of local copy of PCB.		TASKPSBI
22. Turn on scheduled indicator (PPSTA) and turn off scheduling initiated (TCADLITC).		TASKSCOM

Extended Description	Routine	Label
23. The scheduling record is only created for tasks with update intent.		
24.		TSKLGN1

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 6 of 7)



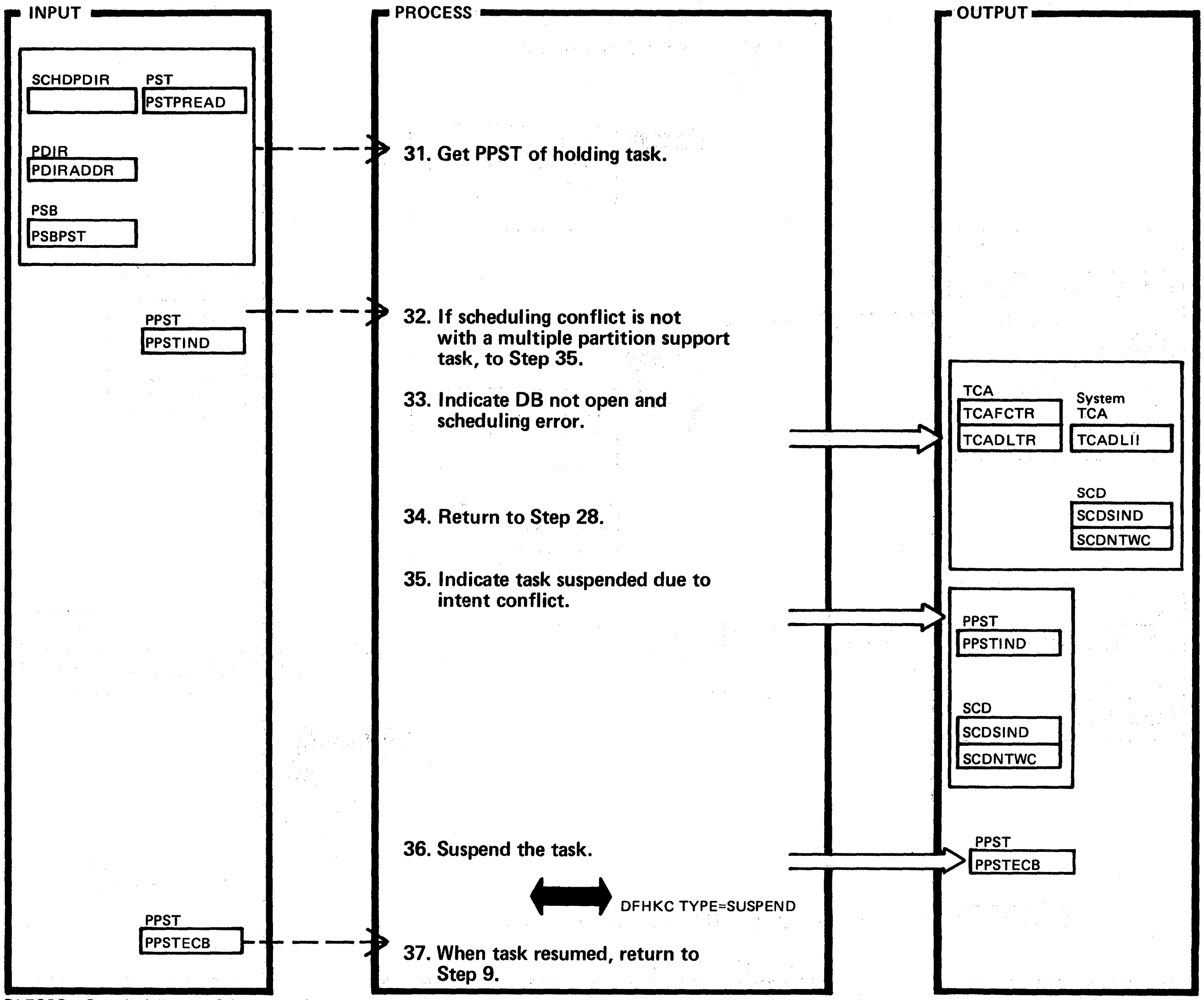
DLZODP — Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
25.	DLZODP	TASKLGNX
27. The hashed ID is used by space management to prevent freed space from being reused before the task terminates.		TASKEXIT
28.		TASKEXTF
29. PSB is being used exclusively. PPST address of holding task is in DDIR.		SCHDWTCK

Extended Description	Routine	Label

Figure 2-6.1. DL/I Pre-scheduling and PSB Scheduling Routines (Part 7 of 7)



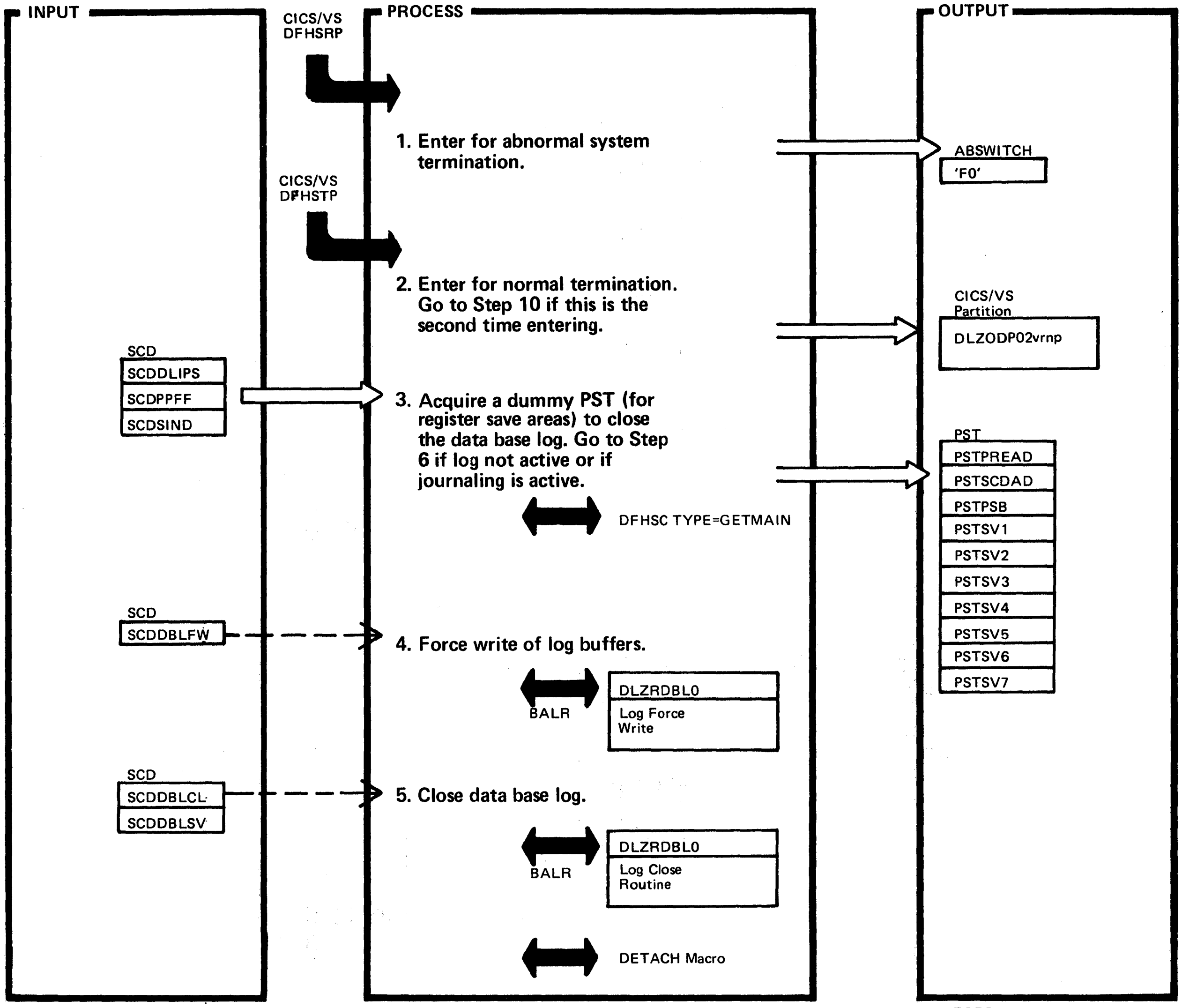
DLZODP — Pre-scheduling and Scheduling CSECT

DLZODP

Extended Description	Routine	Label
31. PSB is being used with update intent.	DLZODP	SCHDWAIT
32. Skip suspension of task if scheduling conflict is with a MPS task (because this situation could result in a long delay).		SCHDMPS
33. If the holder is an MPS scheduled task, a data base not open error X'0C' is set in TCAFCTR and MPS conflict reason code is set in TCADLTR. Scheduling error indicator TCADLISE is turned on.		
35. Flag PPSTSI is turned on in PPST and the wait counter is incremented. The waiting indicator SCDTWFI is turned on.		SCHDOWAT
36.		SCHDTKWT

Extended Description	Routine	Label

Figure 2-6.2. System Abnormal and Normal Termination (Part 1 of 3)



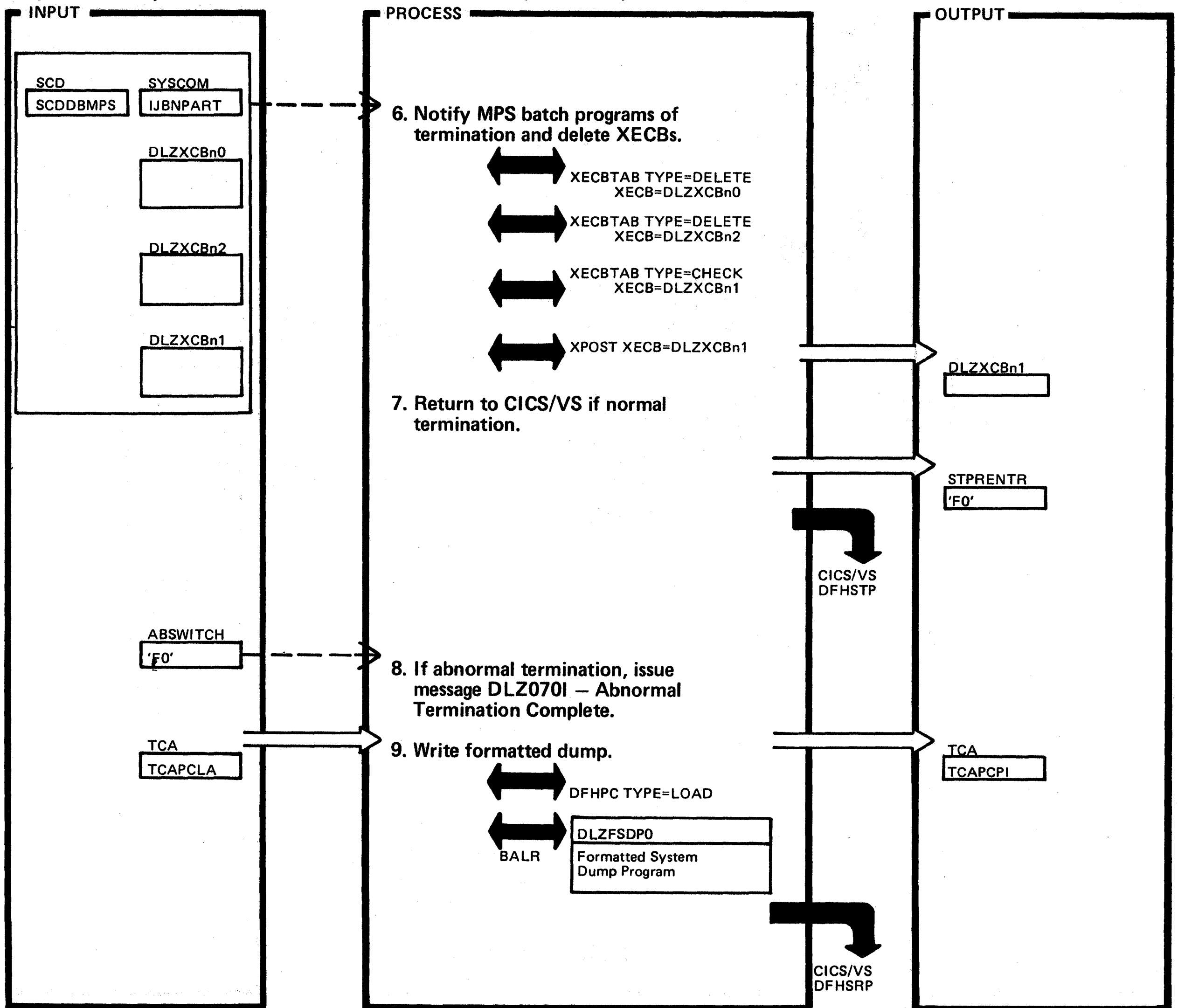
DLZODP02 — System Abnormal and Normal Termination CSECT

DLZODP

Extended Description	Routine	Label
1. Entry is made from CICS/VS System Recovery Program on abnormal termination. Return is back to DFHSRP.	DLZODP03	DLZODP03
2. Routine identifier (DLZODP02vrnp) is defined here. For normal termination DFHSTP enters this routine twice.	DLZODP02	DLZODP02
3. Issue message DLZ067I if there is insufficient storage to terminate DL/I.		STPRENTR
4.		STPSVCH1

Extended Description	Routine	Label

Figure 2-6.2. System Abnormal and Normal Termination (Part 2 of 3)



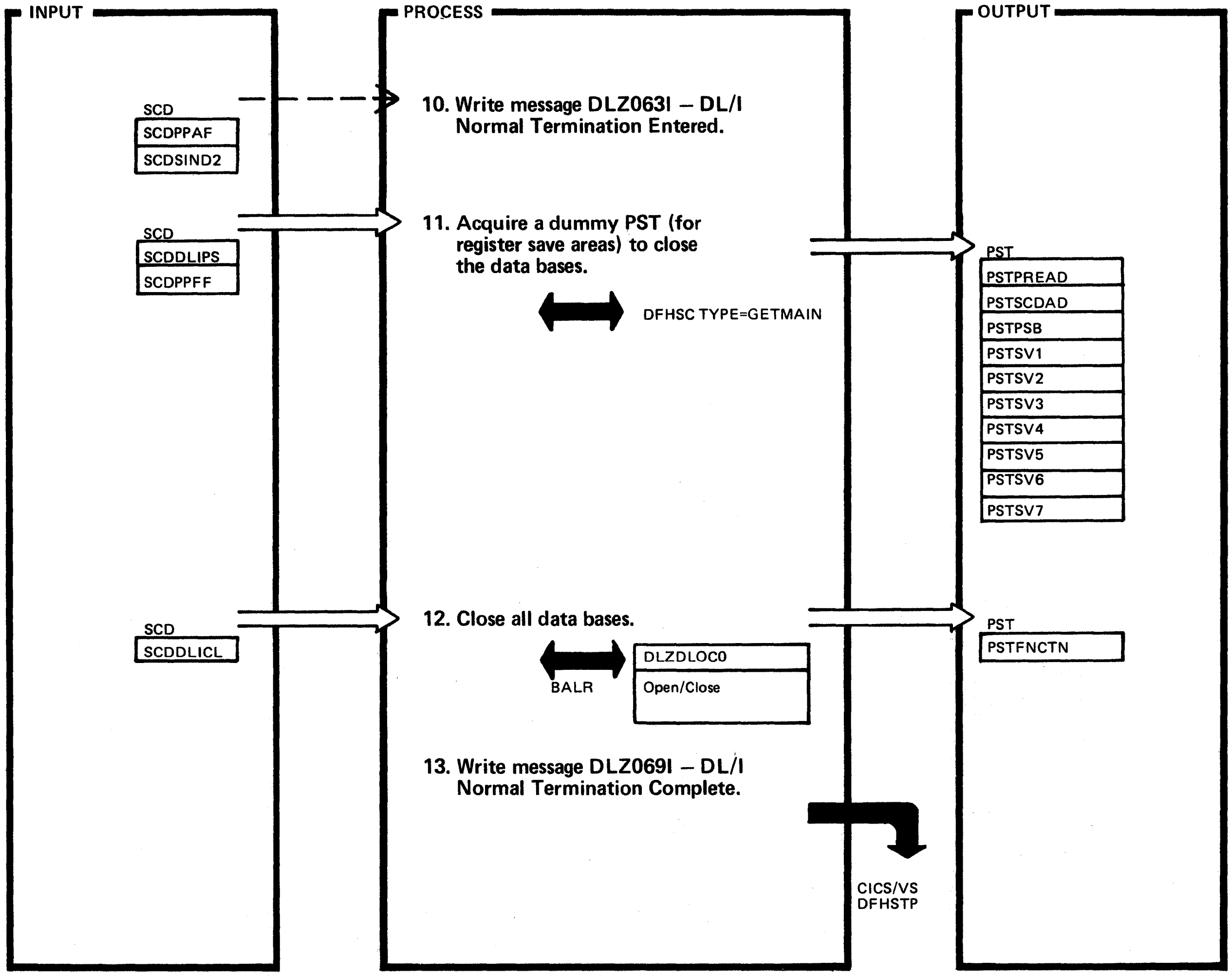
DLZODP02 – System Abnormal and Normal Termination CSECT

DLZODP

Extended Description	Routine	Label
6. MPS batch programs may be active or waiting for online MPS processing and are therefore notified if CICS/VS terminates. Online XECBs defined for MPS are also deleted. The indicator SCDXECB at SCDDBMPS is on if any XECBS are defined by module DLZMPC00 The Start Partition XECBs (DLZXCBn0 – where n is equal to the partition ID) for each partition are deleted first. Then the BPC XECBs (DLZXCBn2) for each partition are deleted. Finally, locate each MPS Batch XECB (DLZXCBn1) and XPOST them.	DLZODP02	STPEXLOG
7.		STPNOXCB
8.		ABTERM
9.		STPEXIT3

Extended Description	Routine	Label

Figure 2-6.2. System Abnormal and Normal Termination (Part 3 of 3)



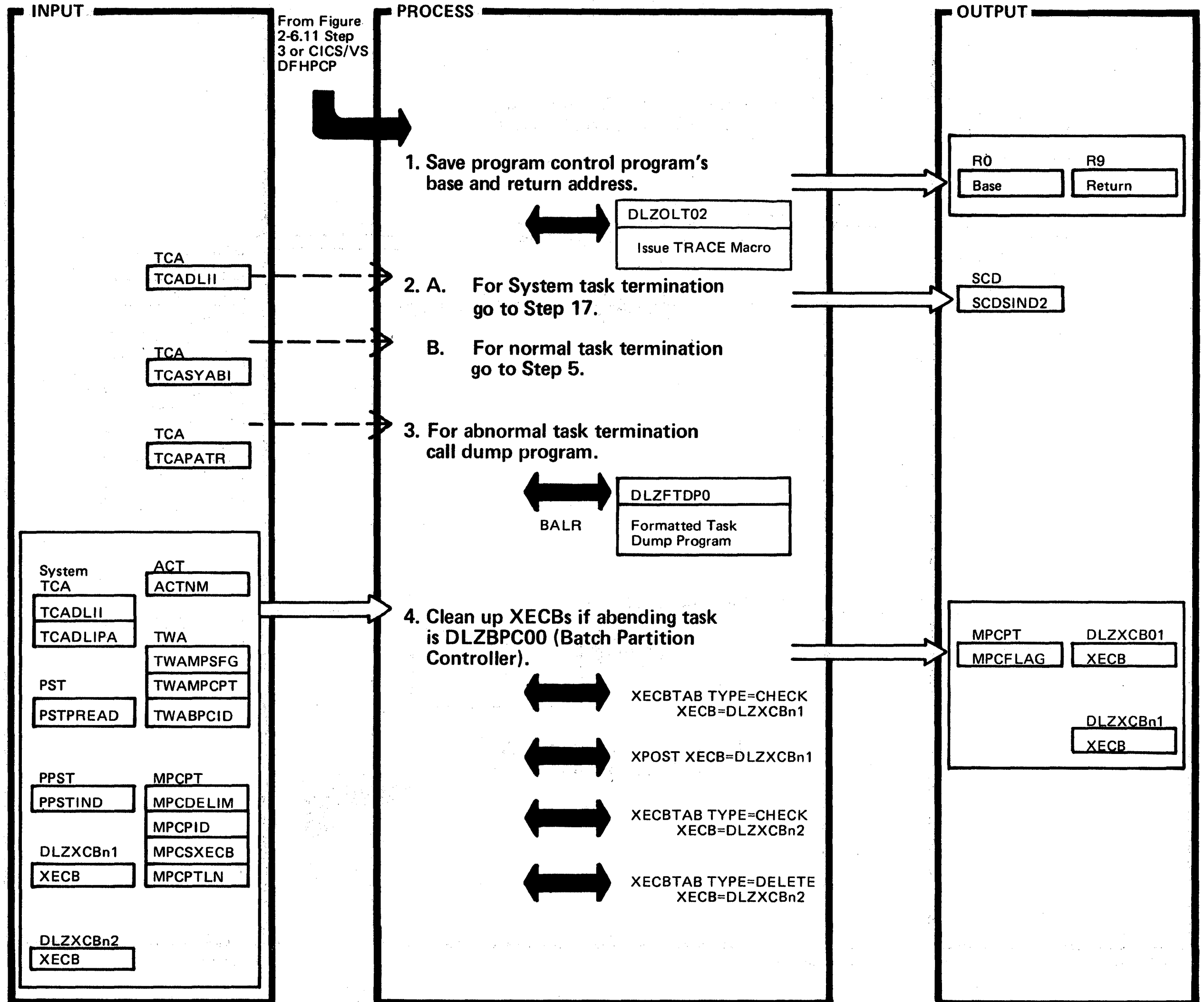
DLZODP02 – System Abnormal and Normal Termination CSECT

DLZODP

Extended Description	Routine	Label
10. Write message DLZ068I – System Previously Abended, DL/I Abnormally Terminated – if the system abend indicator (SCDSYSAB) is on at this time.	DLZODP02	STPFLUSH
Write message DLZ065I – Active DL/I Tasks – if a PPST is still active.		
11. Write message DLZ067I if there is insufficient core to terminate DL/I.		STPBFFL
12. Write message DLZ066I if an error occurred during close processing. Then load and execute dump module DLZFSDP0 via DFHPC TYPE=LOAD and return to DFHSTP.		STPCLOSE

Extended Description	Routine	Label

Figure 2-6.3. Task Termination Routine (Part 1 of 5)



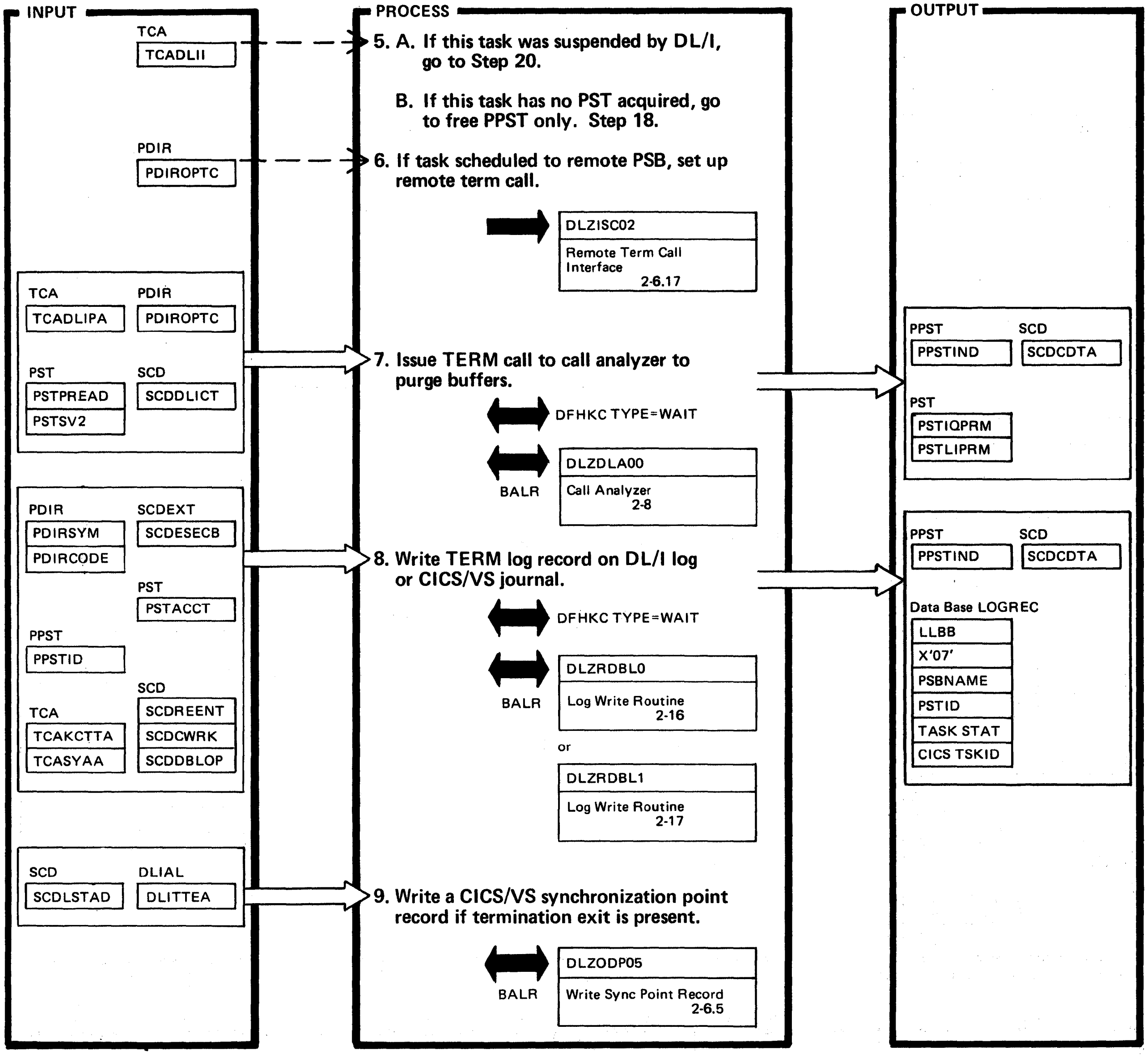
DLZODP01 — Task Termination Routine CSECT

DLZODP

Extended Description	Routine	Label
1. Routine identifier (DLZODP01vrnp) is defined here.	DLZODP01	DLZODP01
This is the entry point for CICS/VS PCP termination exit indicator (TCADLITE) is on and the task is about to be detached. Also, if a TERM or T call is detected by the program request handler, an entry is made here to unschedule the task.		DLZTKTRM
If trace is enabled, a task termination entry with a ID=X'F8'; type of request =X'E3', is made showing why termination was requested, and the DL/I status.		DLZOLT02
2. A. The system interface active indicator (SCDSYACT) at SCDSIND2 is turned off if this is a system task (TCADLPAS is on).		
B.		NOSYSTSK
3. No formatted dump will be produced in case of missing PST or insufficient storage available. If SYSDUMP=YES was specified for the DOS/VS partition, an IDUMP is taken instead of the formatted dump.		DLZODP06

Extended Description	Routine	Label
DL/I system ABEND will be reduced to task ABENDs. In case of DL/I system ABEND all DL/I tasks will be abended by DL/I. For each task, DLZFTDP0 will be called.		
DLZFTDP0 uses the CICS dump macro DFHDC, that dumps DL/I blocks on the CICS dump data set. To get the dump on printer, use offline CICS program DFHDUP.		
4. If BPC (DLZBPC00) is the terminating task, the POST bit in the Stop Partition XECB (DLZXCB01) is set on to signal MPC (DLZMPC00) that BPC abended. Note that the XPOST macro is not needed because the XECB (DLZXCB01) was defined in this same partition.		NODUMP
The partition ID (TWABPCID) this terminating BPC was attached for is used to locate MPS Batch XECB (DLZXCBn1) via XECBTAB/CHECK which is then posted (XPOST). The BPC XECB (DLZXCBn2) is then located and deleted.		

Figure 2-6.3. Task Termination Routine (Part 2 of 5)



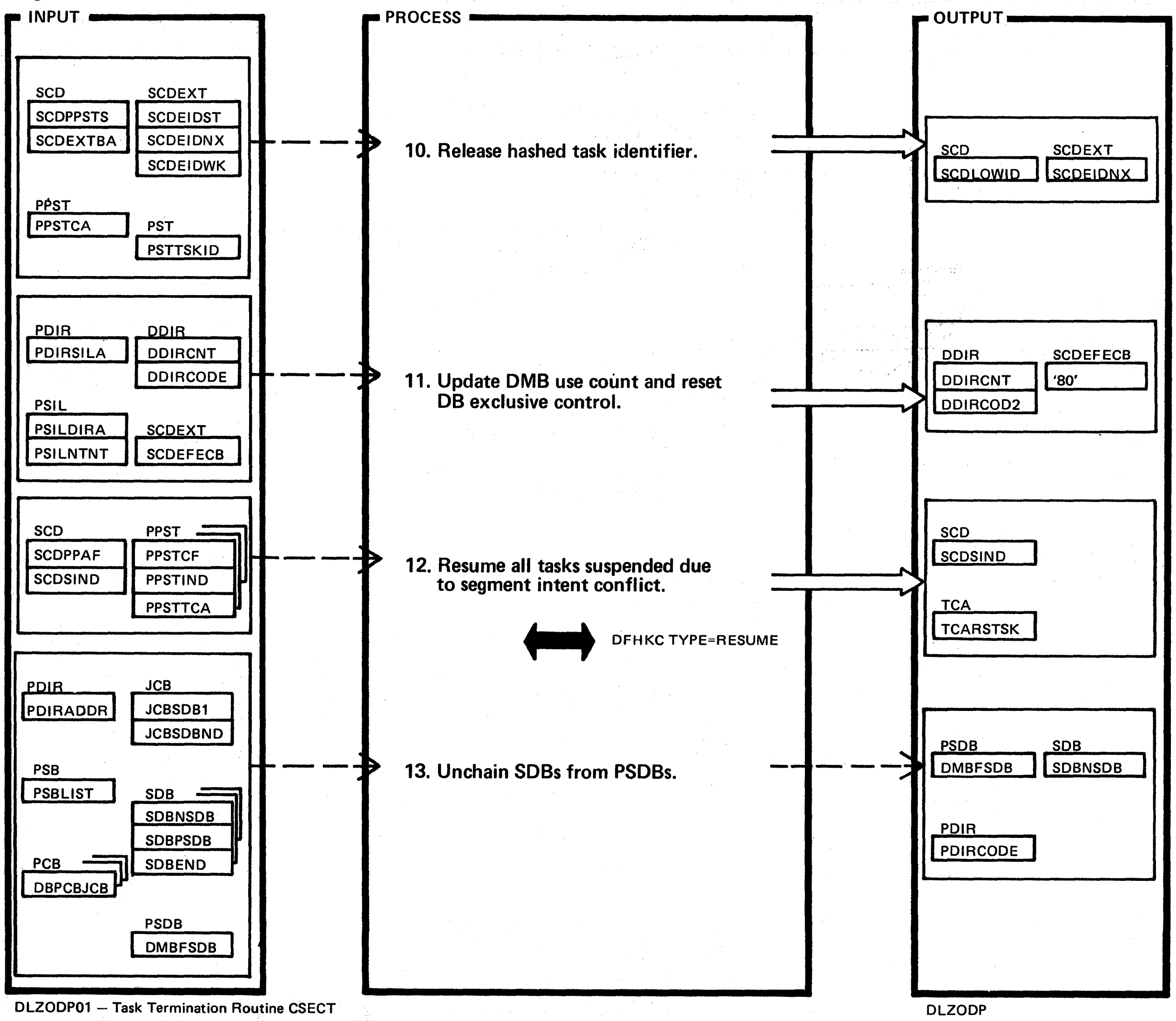
DLZODP01 – Task Termination Routine CSECT

DLZODP

Extended Description	Routine	Label
5.	DLZODP01	NORMTTRM
7. A wait may be done at this time for the logger if it is not available.		
8. The termination record is logged for normal termination of update users only.		CHKLOGGR
9.		TRMLGBY

Extended Description	Routine	Label

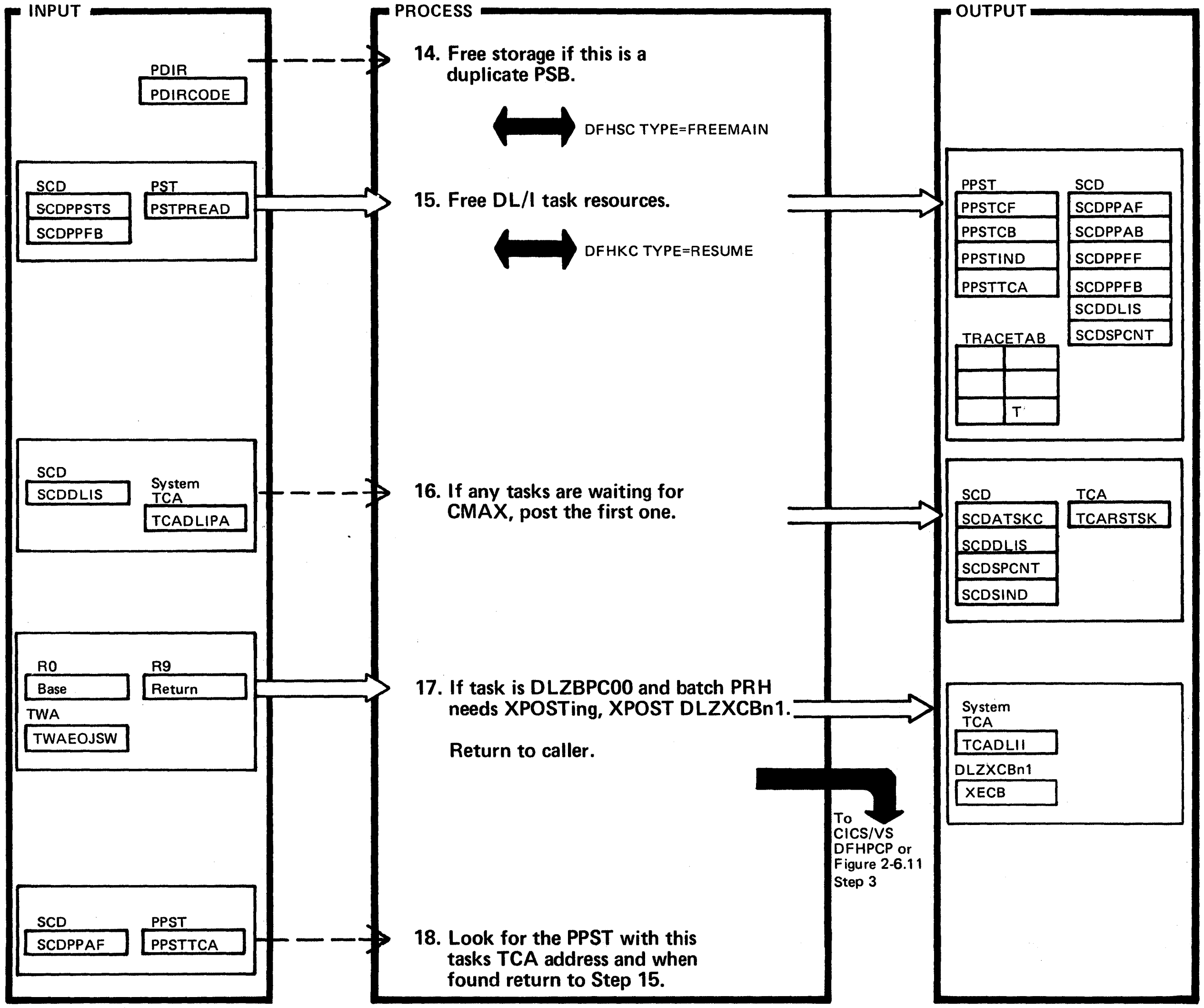
Figure 2-6.3. Task Termination Routine (Part 3 of 5)



Extended Description	Routine	Label
10. The lowest active identifier is maintained in the SCD. DL/I Space Management uses the low and high identifiers to exclude free space belonging to active tasks from reuse.	DLZODP01	TRMXTBY
11.		TRMDMBLP
12. All tasks suspended for intent are resumed. At the next dispatch cycle they will attempt to schedule.		TRMPOLUP
13.		TRMFRESD

Extended Description	Routine	Label

Figure 2-6.3. Task Termination Routine (Part 4 of 5)



DLZODP01 — Task Termination Routine CSECT

DLZODP

Extended Description	Routine	Label
14. For read only or update with program isolation. Duplicate PSBs are identified by the PDIR indicator PDIRDUPL.	DLZODP01	TRMPSBFR
15. This cleans up the terminating PPST, removes it from the active chain, and places it on the free chain. It checks the suspend chain and resumes a task suspended due to MAXTASK.		TRMFREPP TRMFREPE
The PPST number of the terminating PPST is put in the internal trace table along with a 'T' to indicate terminating entry. Upon resuming, the total suspended count is decremented 1 and the current maximum task indicator is turned off if it was on (SCDCMTI).		
17. Turn off DL/I scheduled (TCADLISD) and DL/I termination required (TCADLITE) indicators. Issue message DLZ084I if XPOST fails.		TRMEXIT
18.		TRMFREPO

Extended Description	Routine	Label

Figure 2-6.3. Task Termination Routine (Part 5 of 5)

INPUT

SCD: SCDDLIS
TCA: TCADLIPA

PROCESS

19. Issue a CICS/VS dump if the PPST is not found and go to Step 17.

DFHDC TYPE=PARTIAL

20. Remove task from the suspend chain and go to Step 17.

OUTPUT

SCD: SCDATSKC, SCDSPCNT
TCA: TCADLIPA

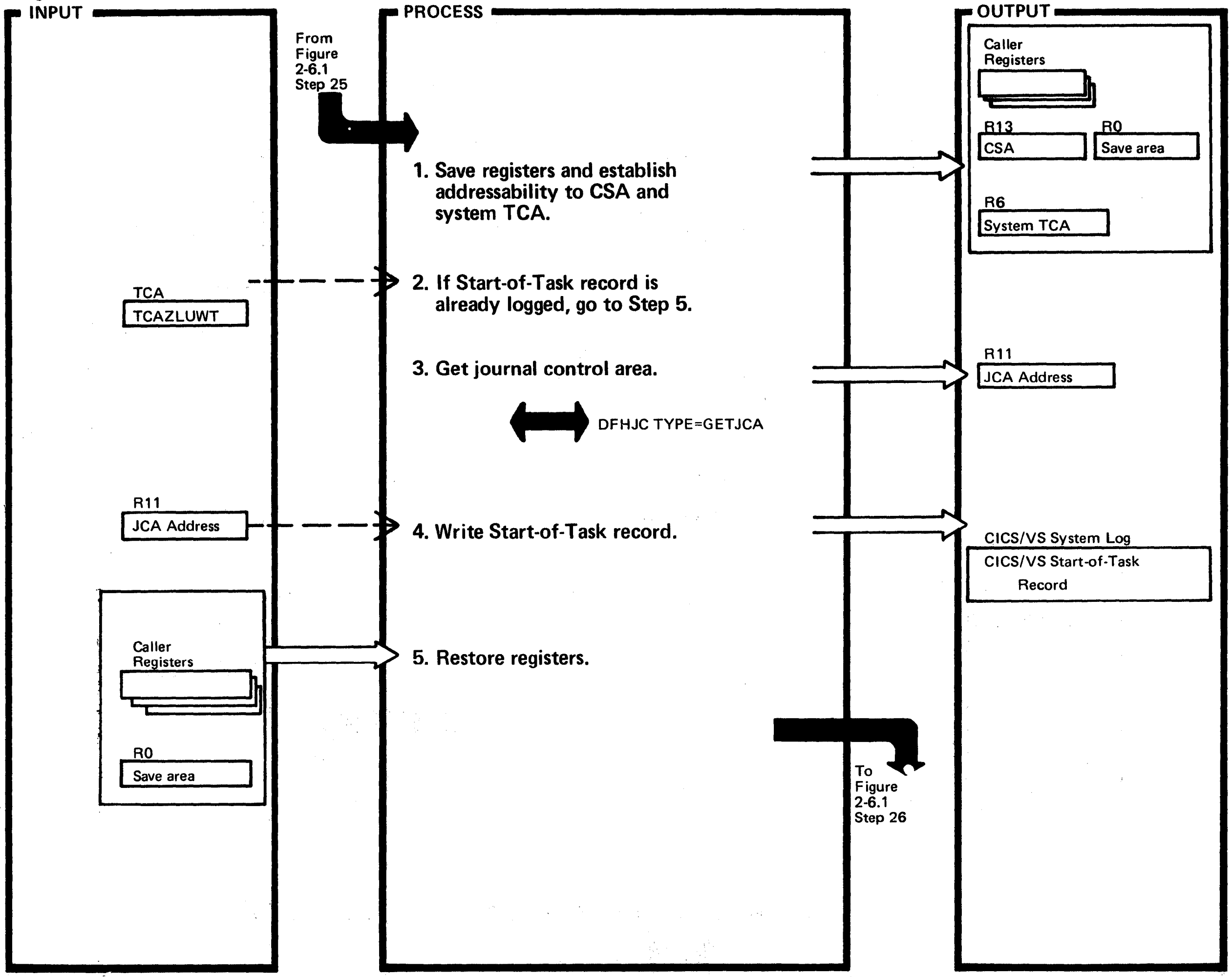
DLZODP01 – Task Termination Routine CSECT

DLZODP

Extended Description	Routine	Label
19.	DLZODP01	TRMPOABA
20.		TRMSUSPA

Extended Description	Routine	Label

Figure 2-6.4. Start-of-Task Record Writer



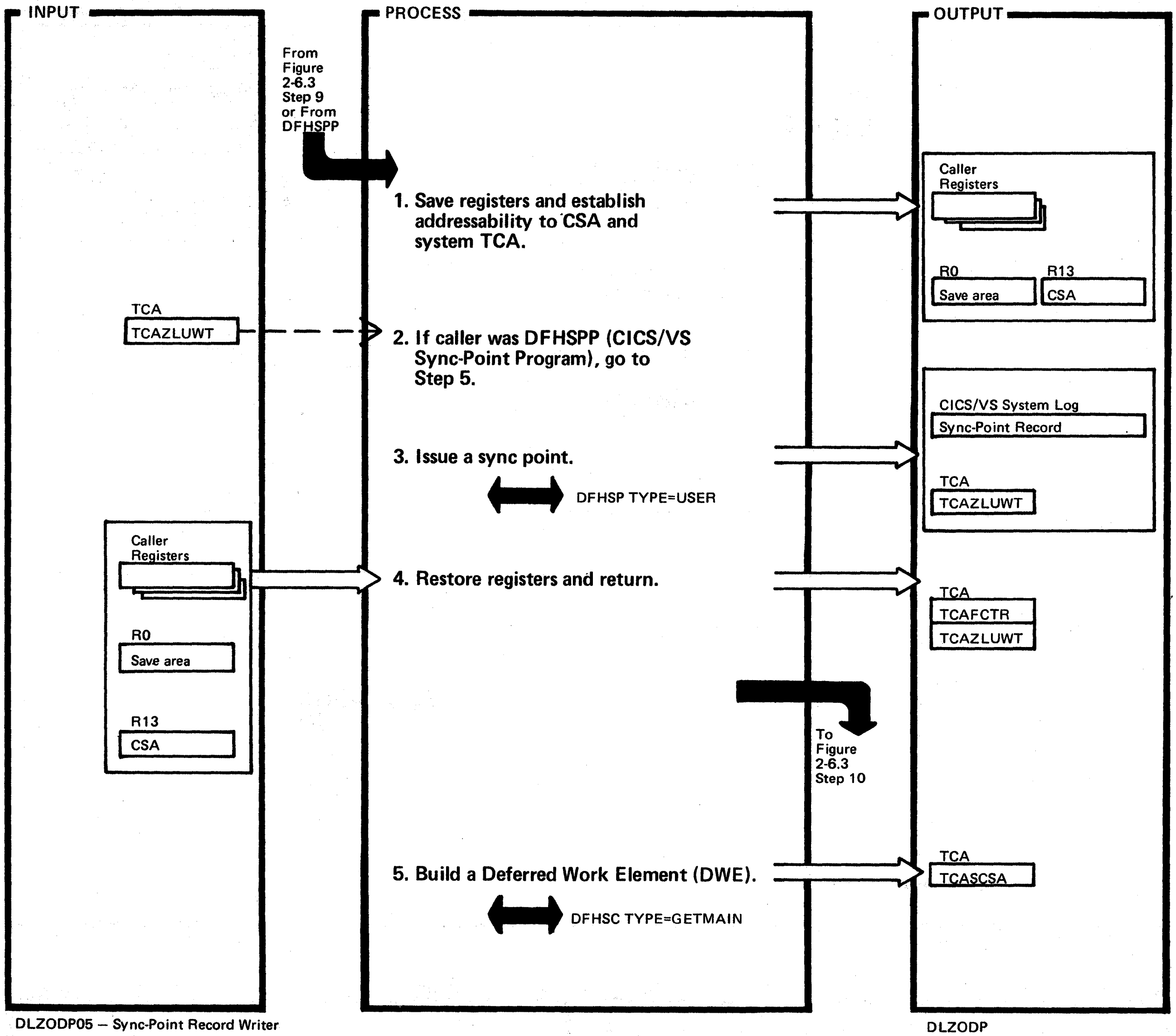
DLZODP04 – Start-of-Task Record Writer

DLZODP

Extended Description	Routine	Label
1. Routine identifier DLZODP04vrnp is defined here.	DLZODP04	DLZODP04
3. Acquire task's JCA and establish JCA addressability.		
4. Dummy record is written to system log and CICS Start-of-Task indicator is set in it to mark start-of-task on system log.		
5. Restore save area address in register 13 and reload registers 14 through 12 from the save area.		DLXSCRET

Extended Description	Routine	Label

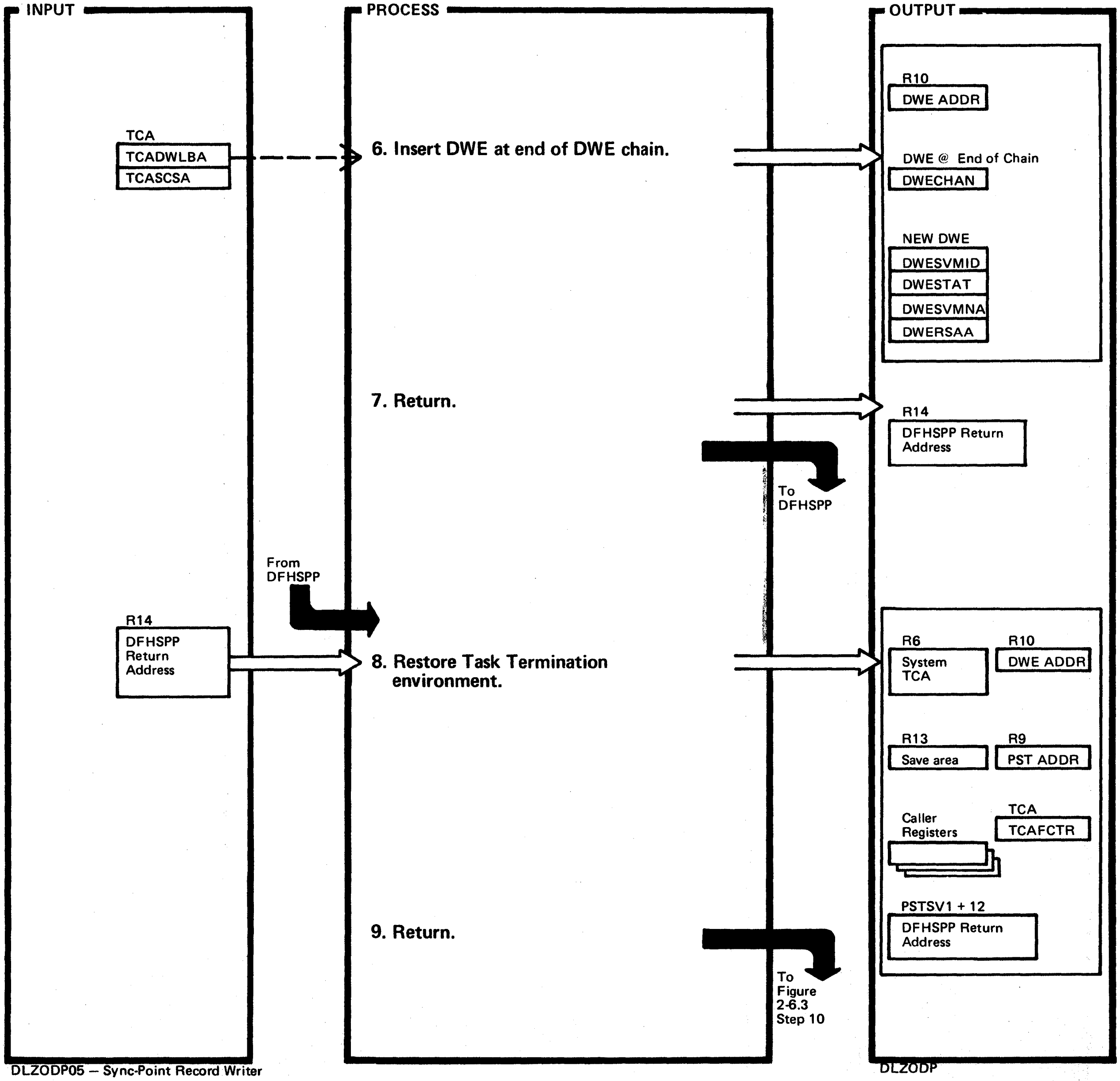
Figure 2-6.5. Sync-Point Record Writer (Part 1 of 2)



Extended Description	Routine	Label
1. Routine Identifier DLZODP05vrnp is defined here.	DLZODP05	DLZODP05

Extended Description	Routine	Label

Figure 2-6.5. Sync-Point Record Writer (Part 2 of 2)

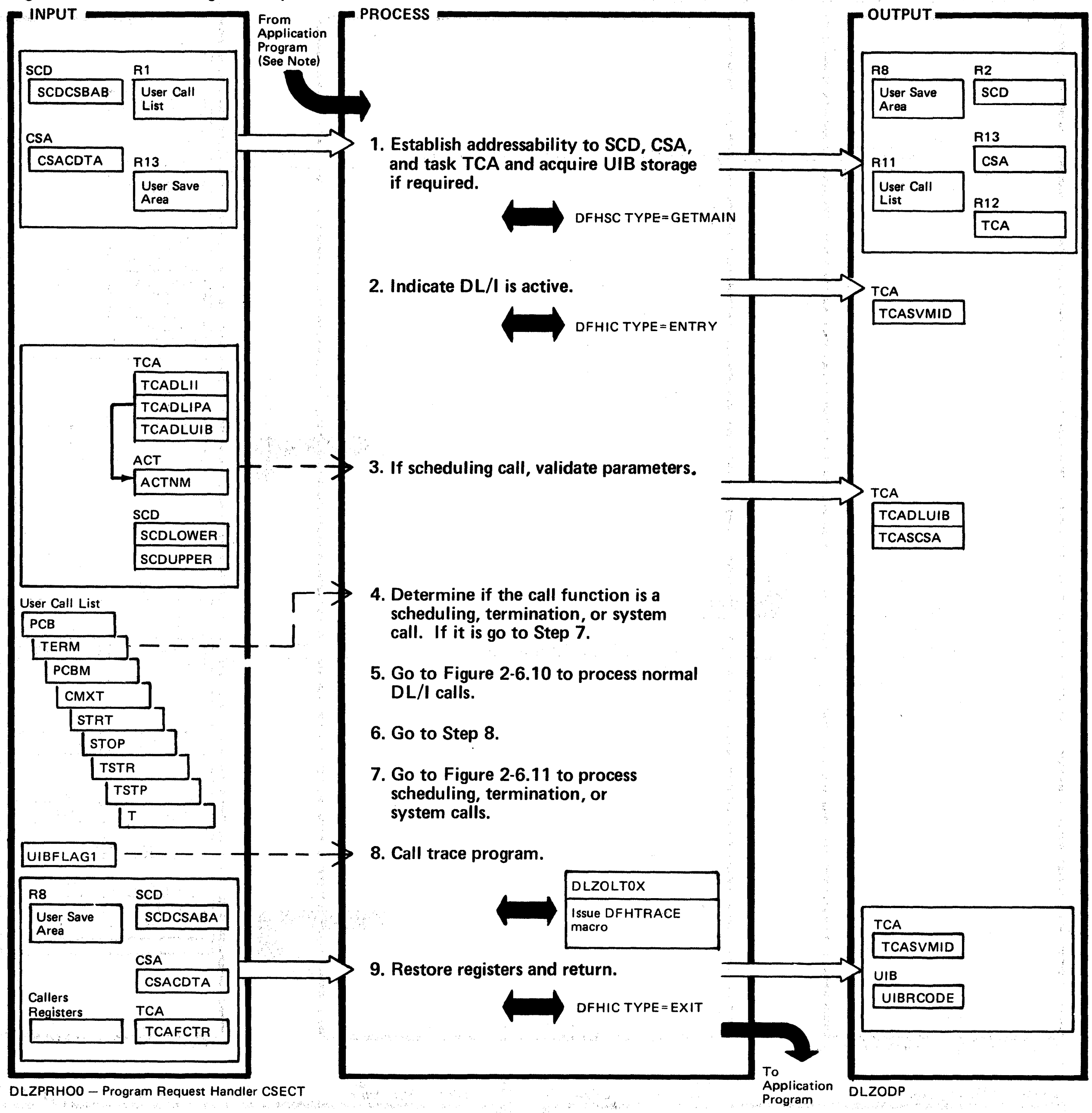


DLZODP05 — Sync-Point Record Writer

Extended Description	Routine	Label
6. Store address of DWE entry point of DLZODP05 in DWE for deferred processing by CICS/VS.	DLZODP05	DLXDWECN
8. Set up registers and PST save area to complete task termination and then return to DFHSPP.		DLXTDWEN

Extended Description	Routine	Label

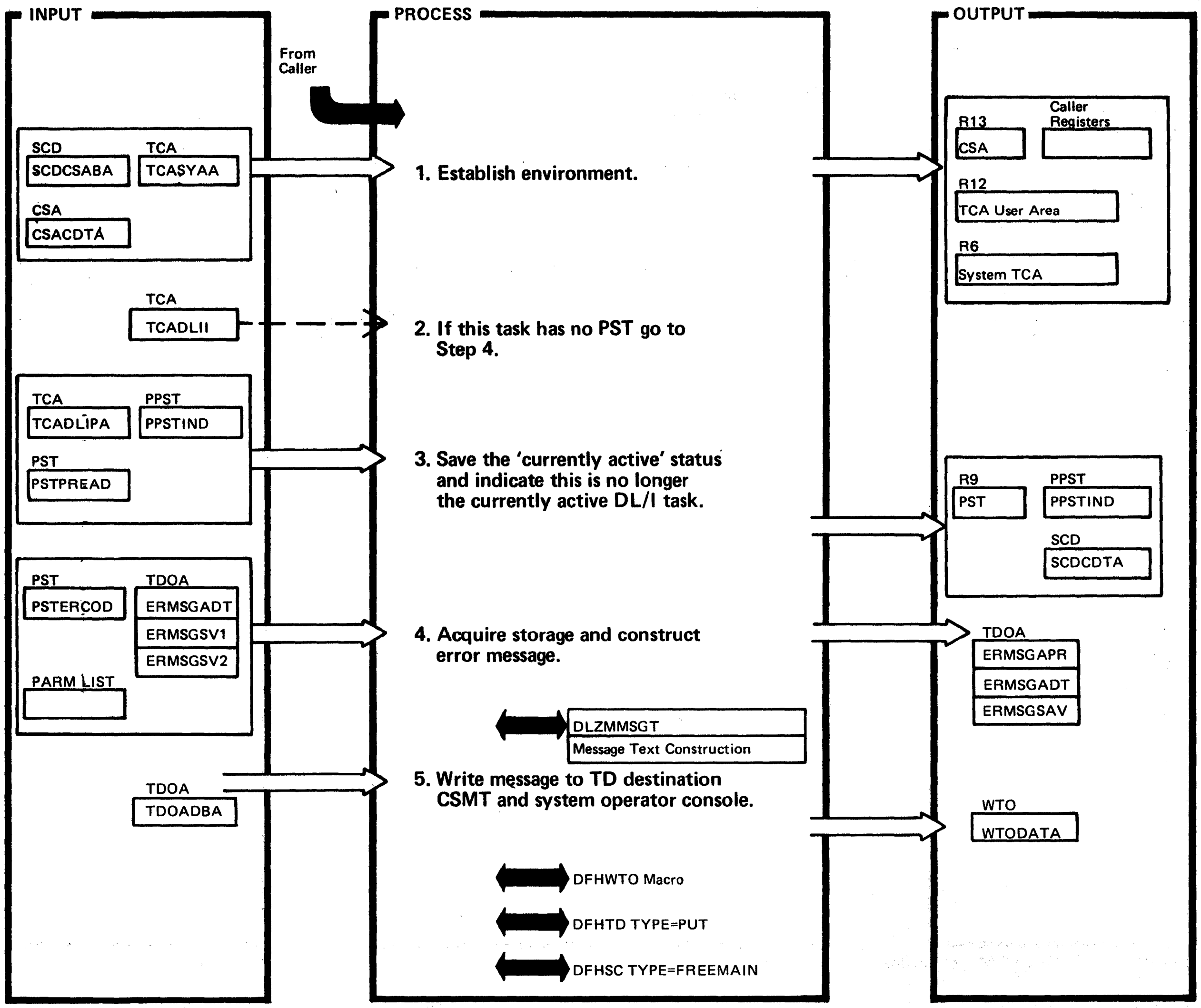
Figure 2-6.6. Online Program Request Handler



Extended Description	Routine	Label
Note: This routine receives control from the Language Interface Module (DLZLI000) linked with the application program.		
1. Module identifier (DLZPRHO0 vrnp) is defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied.	DLZPRHO0	DLZPRHO0
2.		OVERID3

Extended Description	Routine	Label
3.		ECIVAL
4.		TESTFUNC
8. UIBFLAG1 bit setting determines whether DLZOLT00, DLZOLT01, or DLZOLT02 is called for scheduling, data base or termination call, respectively.		
9. If UIB is used, update UIBRCODE from TCAFCTR.		RETURN

Figure 2-6.7. Online Error Message Routine (Part 1 of 2)



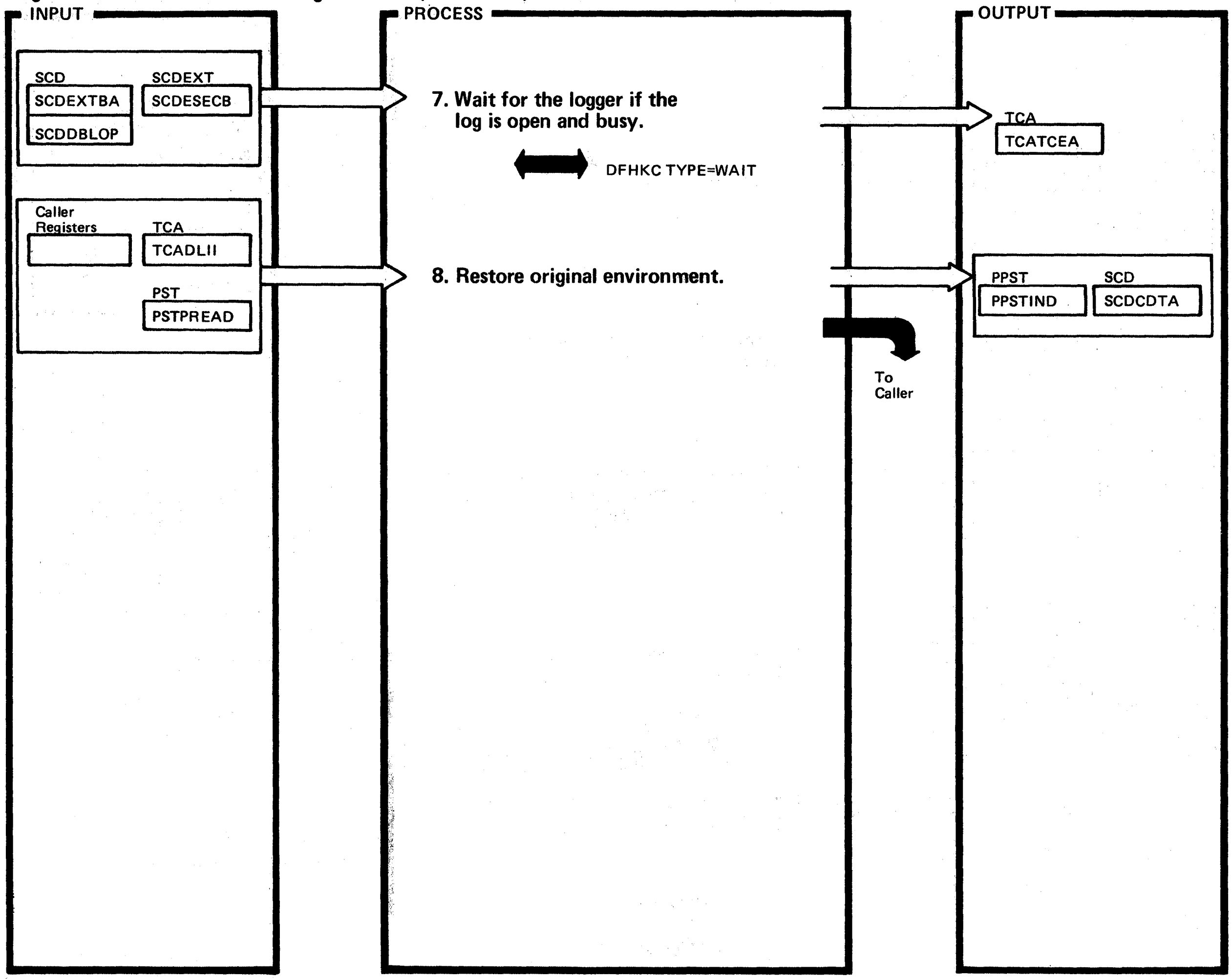
DLZERMSG – Online Error Message Routine

DLZODP

Extended Description	Routine	Label
1. Routine identifier (DLZERMSGvrnp) is defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied.	DLZERMSG	DLZERMSG
2. If there is no PST the message number will be in the parameter list which is pointed to by R1.		
3. 'Currently active' has a special meaning. There may be many DL/I tasks active at this time. Therefore, DL/I uses a bit (PPSTACT) in the PPST to make it easy to spot (in a dump) the single DL/I task that is currently processing non-scheduling DL/I calls (Non-scheduling calls being calls handled by the call analyzer and other DL/I action modules).		
Tasks like the MPS Batch Partition Controller can have a PST and can call DLZERMSG while not being the currently active task.		

Extended Description	Routine	Label
Since CICS/VS can schedule another DL/I task during a CICS/VS operation, (for example, I/O) PPSTACT is reset by DLZERMSG for the duration of the write because there can only be one task marked as 'currently active' by definition.		
4. The GETMAIN output buffer is needed by CICS/VS transient data services.		ERMSGETM
DLZMMSGT is used to construct text for messages with message numbers from 1 – 255.		
5.		ERMSGPUT

Figure 2-6.7. Online Error Message Routine (Part 2 of 2)



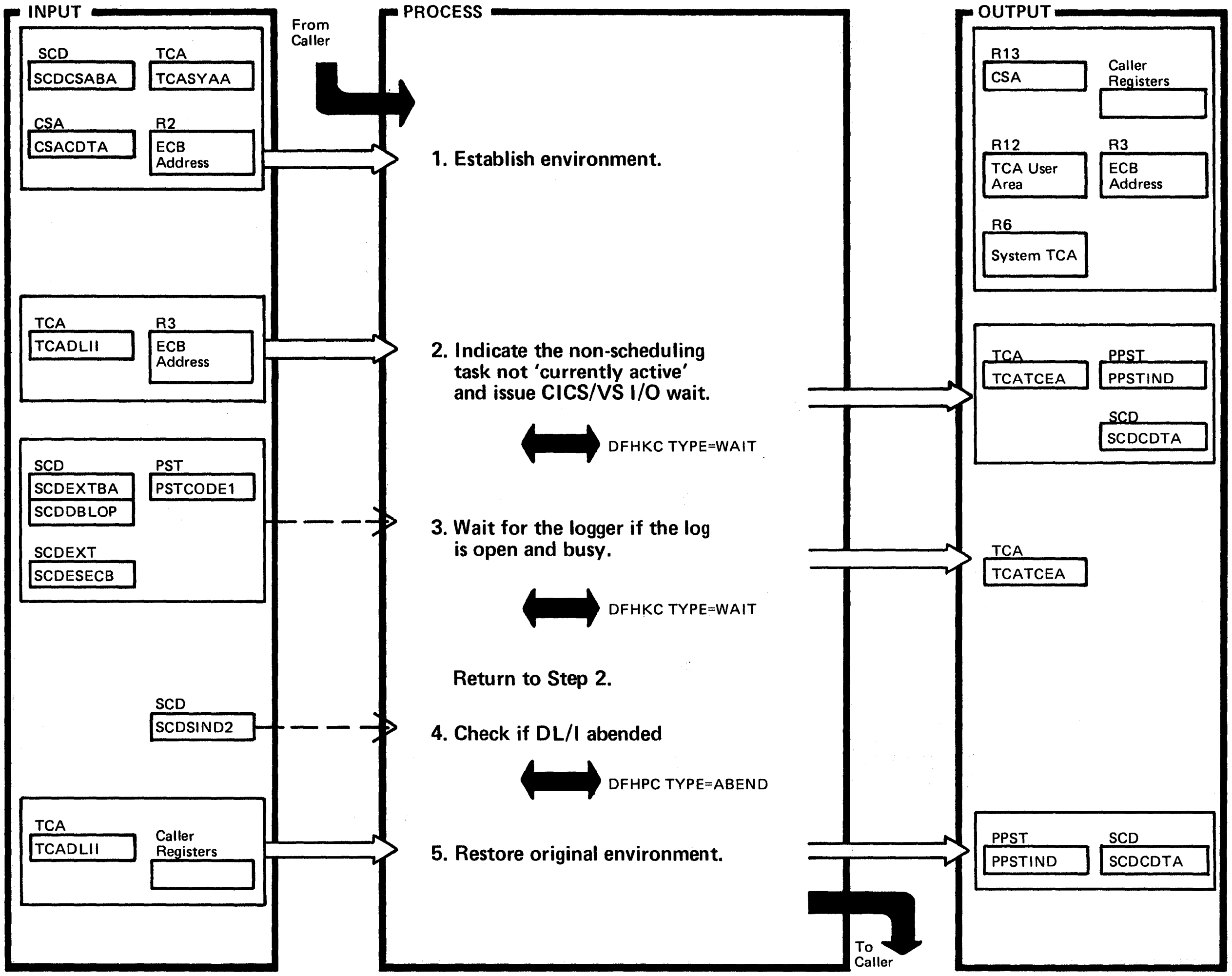
DLZERMSG – Online Error Message Routine

DLZODP

Extended Description	Routine	Label
6. Since control was lost during the I/O another job may have activated the logger. To assure serialization, this routine must wait until the logger is done.	DLZERMSG	ERMSGERT
7. Reset 'currently active' status to the value it was on entry.		

Extended Description	Routine	Label

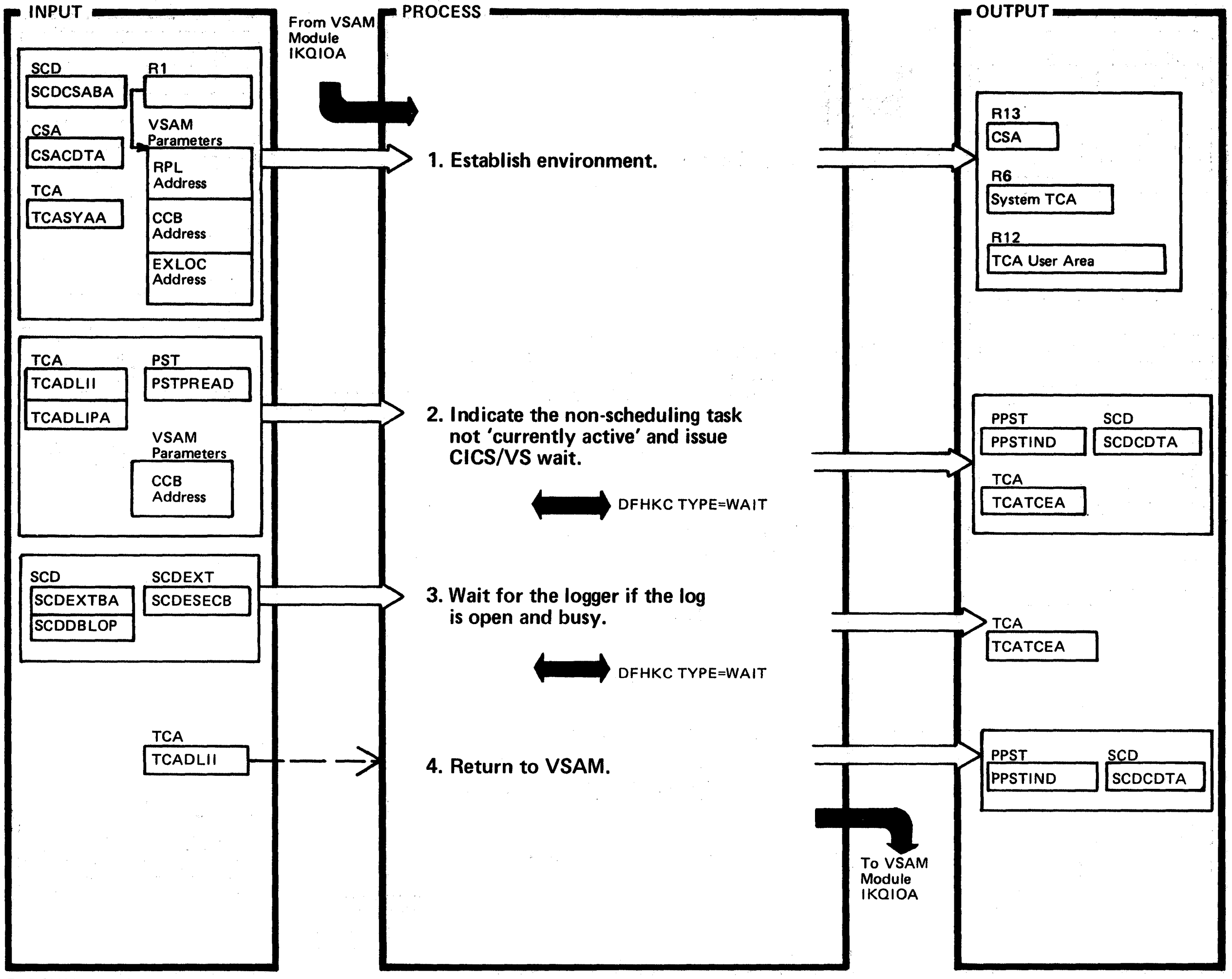
Figure 2-6.8. Online Wait Routine



Extended Description	Routine	Label
1. Routine identifier (DLZOWAITvrnp) is defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied.	DLZOWAIT	DLZOWAIT
2. A non-scheduling task is a task that does not issue the special scheduling call (PCB,SYSTEMDL,password) to schedule itself so that it may issue system calls: CMXT, STRT, STOP, TSTR, and TSTP. 'Currently active' has a special meaning. There may be many DL/I tasks active at this time. Therefore, DL/I uses a bit (PPSTACT) in the PPST to make it easy to spot (in a dump) the single DL/I task that is currently processing non-scheduling DL/I calls (non-scheduling calls being calls processed by the call analyzer and other DL/I action modules). Since CICS/VS can schedule another DL/I task during a CICS/VS operation (for example, I/O) PPSTACT is turned off until return is made to the caller of		OWATRECK

Extended Description	Routine	Label
DLZOWAIT because there can only be one task marked as 'currently active' by definition.		
3. Serialization of the logger resource is another job of the online nucleus. Since control was lost during the wait, another job may have activated the logger. To assure serialization, this routine must wait until the logger is done. Return is then made to Step 2 to recheck that this task's ECB is posted and to reissue the wait again if it isn't.		
4.		OWAITRET
5. Reset non-scheduling task to currently active status as it was on entry.		OWAITCON

Figure 2-6.9. VSAM Asyncronous Exit Processor



DLZOVSEX — VSAM Asyncronous Exit Processor

DLZODP

Extended Description	Routine	Label
1. Routine identifier (DLZOVSEXvrnp) is defined here.	DLZOVSEX	DLZOVSEX
2. A non-scheduling task is a task that does not issue the special scheduling call (PCB,SYSTEMDL,password) to schedule itself so it may issue system calls: CMXT, STRT, STOP, TSTR, and TSTP. 'Currently active' has a special meaning. There may be many DL/I tasks active at this time. Therefore, DL/I uses a bit (PPSTACT) in the PPST to make it easy to spot (in a dump) the single DL/I task that is currently using the call analyzer and other DL/I action modules.		OWATSYS
Since CICS/VS can schedule another DL/I task during a CICS/VS operation (for example, I/O) PPSTACT is turned off until return is made to the caller of DLZOVSEX because there can only be one task marked as 'currently active' by definition.		

Extended Description	Routine	Label
3. Since control was lost during the wait another job may have activated the logger. To assure serialization, this routine must wait until the logger is done.		
4. Reset non-scheduling task to 'currently active' status as it was on entry.		OWATNLOG

Figure 2-6.10. Online PRH Processing of Normal DL/I Calls (Part 1 of 2)

INPUT	PROCESS	OUTPUT
TCA: TCADLII, TCADLIPA	From Figure 2-6.6 Step 6 1. If caller not scheduled, set error and go to Step 16.	
SCD: SCDLOWER, SCDUPPER PST: PSTPREAD, PSTLIPRM PPST: PPSTIND	2. Validate and move call parameter list to PST.	PST: PSTLIPRM
PST: PSTRPSTA	3. If task scheduled to remote PSB, set up remote data base call. DLZISC01 — Remote Data Base Call Interface 2-6.16	PST: PSTSV1
SCD: SCDEXTBA, SCDDBLOP SCDEXT: SCDESECB	4. Wait for the logger if the log is open and busy. DFHKC TYPE=WAIT	
PST: PSTPREAD, PSTCODE1, PSTLIPRM, PSTSV1 SCD: SCDDLICT	5. Exit to DL/I Call Analyzer. BALR DLZDLA00 — Call Analyzer 2-8	SCD: SCDCDTA PPST: PPSTIND PST: PSTIQPRM
PST: PSTREAD, PSTCODE1	6. Clear the active task indicators.	SCD: SCDCDTA PPST: PPSTIND
SCD: SCDLSTAD DLIAL: DLITTEA PST: PSTPCT2	7. Write CICS/VS synchronization point record if call is CHKP. If not, go to Step 11. BALR DLZODP05 — Task Term Sync Point Return 2-6.5	

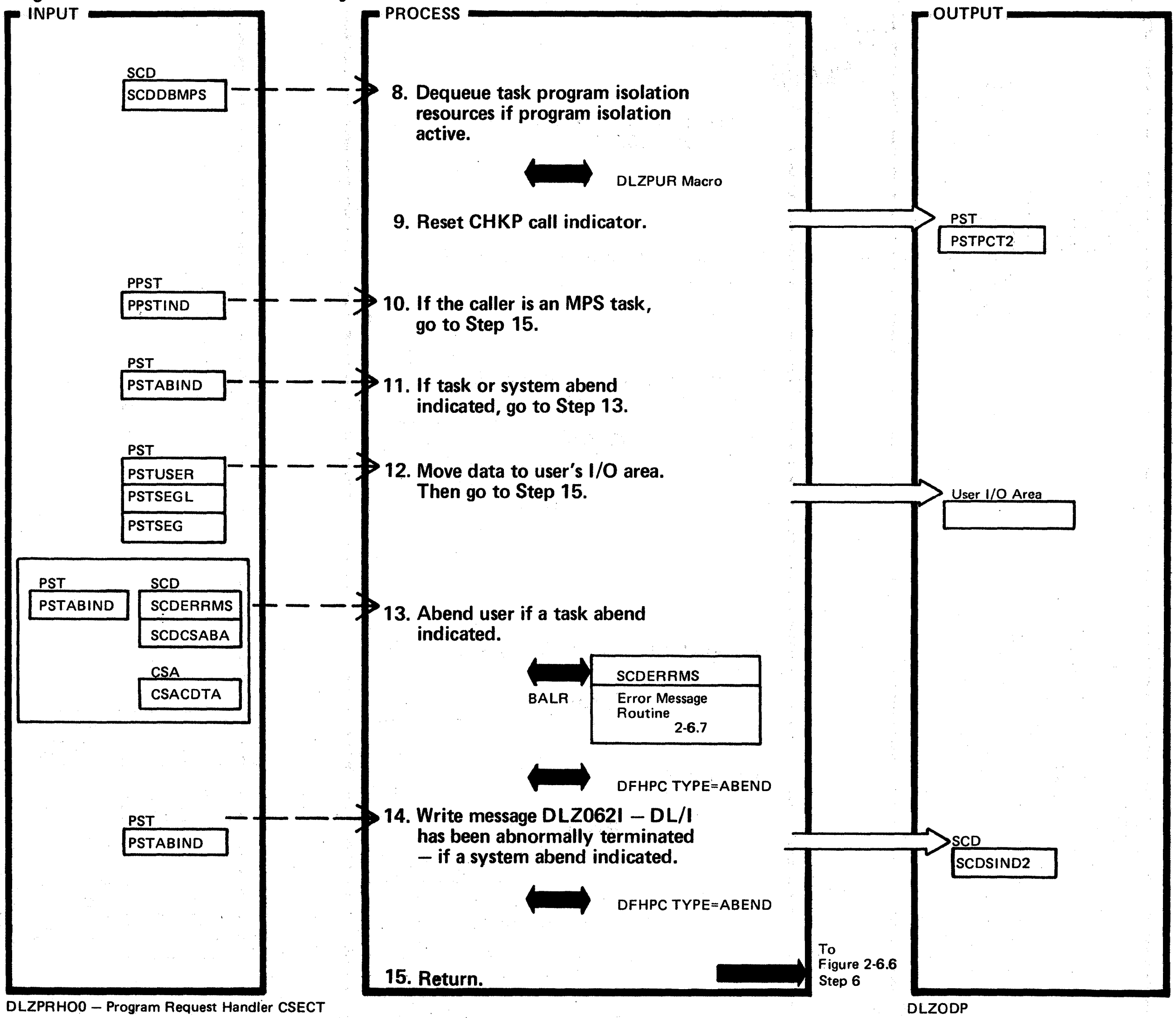
DLZPRHOO – Program Request Handler CSECT

DLZODP

Extended Description	Routine	Label
1. Set 08 in TCADLTR if TCADLISD, task scheduled indicator not on (there was not a previous PCB or PCBM call).		PRODLIC
2. Write message DLZ260I if the parameter count is invalid. Write message DLZ261I if the parameter address is invalid.		EXPLST
3. Allow ISC mirror (DFHMIR) to use IOAREA address=0.		

Extended Description	Routine	Label
5. At this point the system is switched from the CICS/VS state to the DL/I state (standard register assignments). The task TCA address is put in SCDCDTA as the current dispatched task TCA. Also, the current task indicator (PPSTACT) is turned on.		EXITANAL

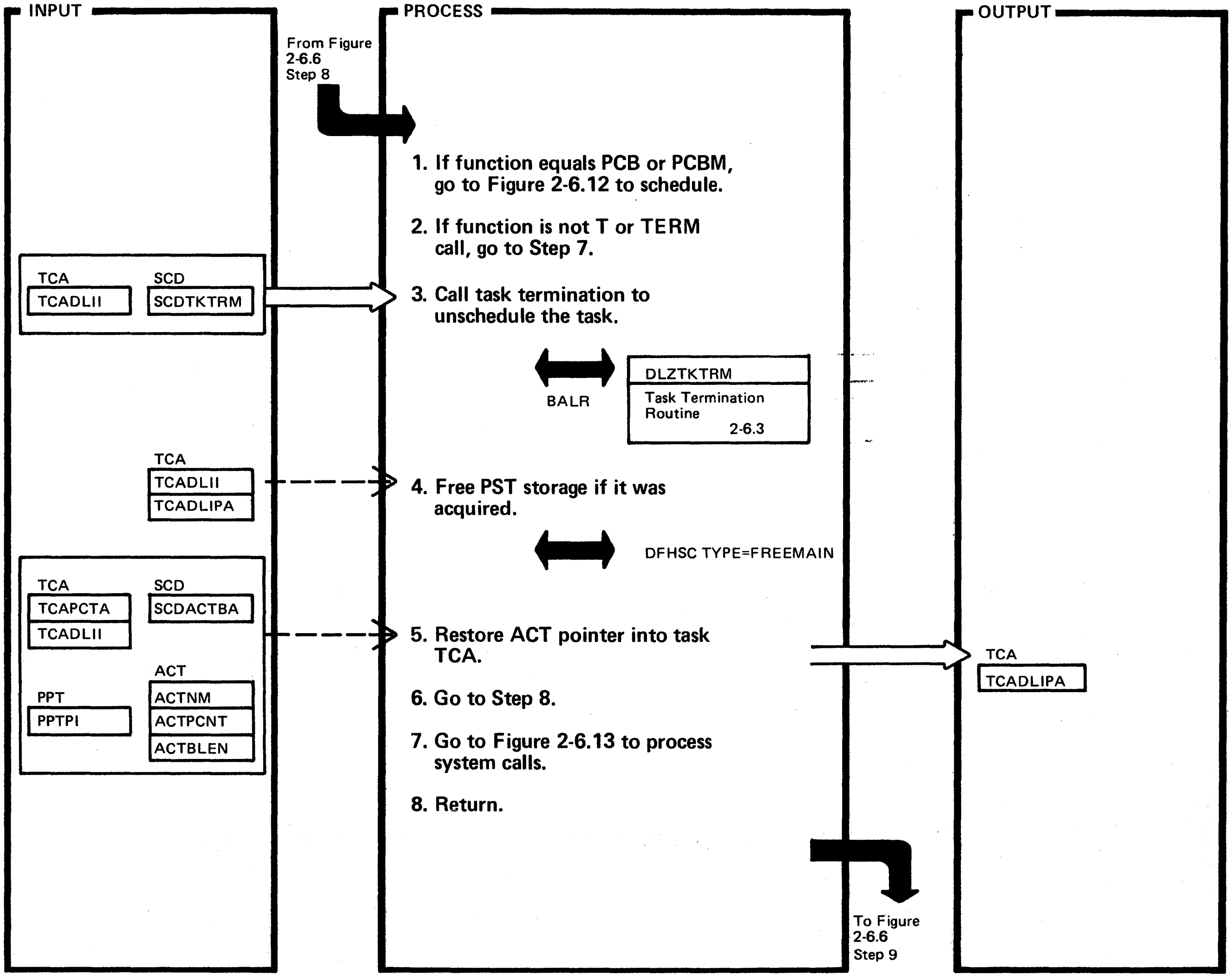
Figure 2-6.10. Online PRH Processing of Normal DL/I Calls (Part 2 of 2)



Extended Description	Routine	Label
9.		ENDCHKP
10. If the caller is not in the CICS/VS DL/I partition (MPS caller – PPSTMPS), skip abend check (DLZMPI00 will handle abend) and skip move of data to user area because you cannot move across partition (DLZMPI00 will handle data move).		NOCHKP
11. If user's IOAREA address=0, skip moving of data.		PSTPCB1
13. If this is a task abend, a CICS/VS abend is issued.		PRHABEND
14. If this is a system abend (PSTSABND on) the system abend indicator SCDSYSAB is set in the SCD and all active DL/I tasks will abend at the earliest opportunity.		PRHSYSAB

Extended Description	Routine	Label

Figure 2-6.11. Online PRH Processing of Schedule Termination, and System Calls



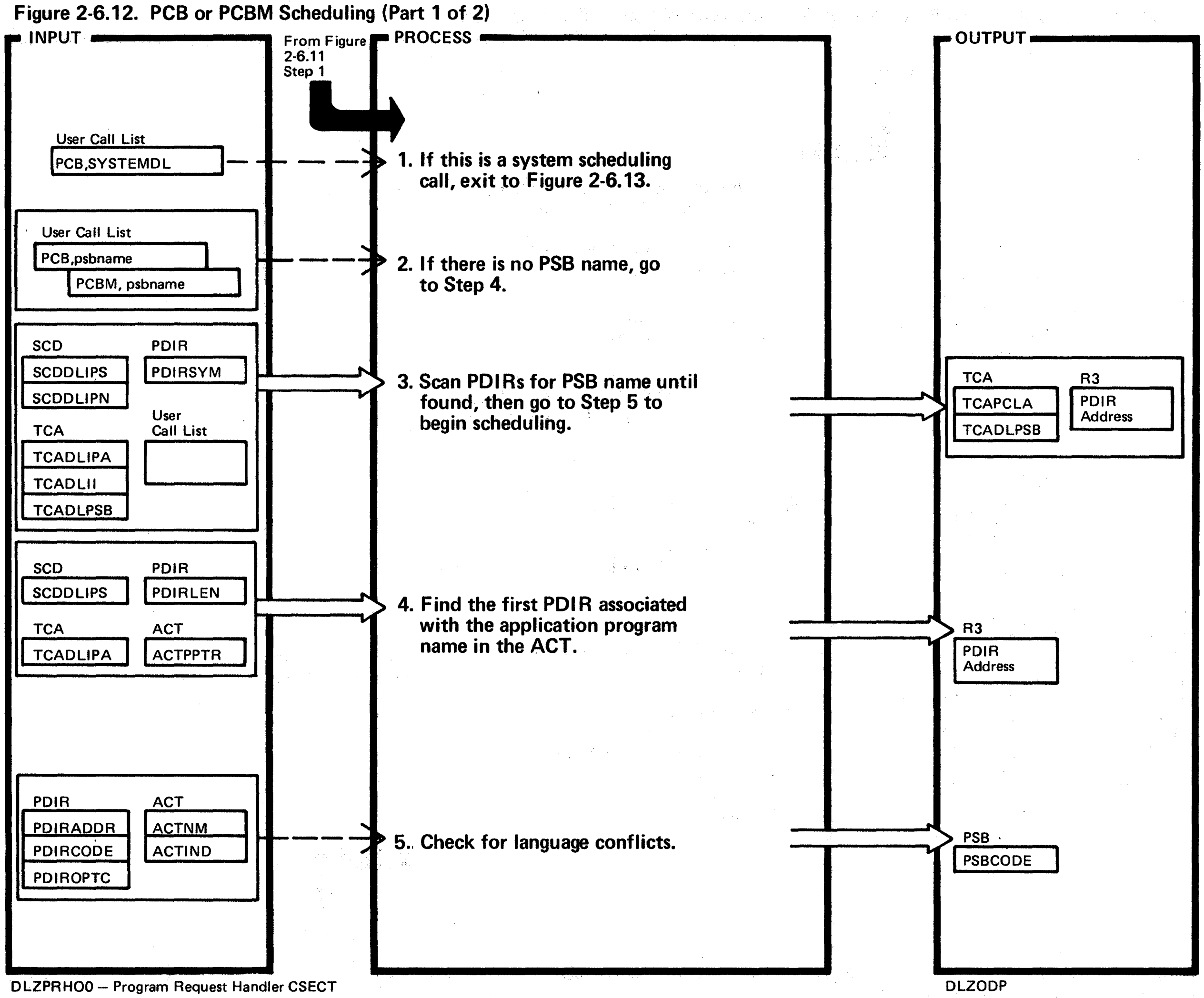
DLZPRHOO — Program Request Handler CSECT

DLZODP

Extended Description	Routine	Label
1.		TESTFUNC
3. Set 07 in TCADLTR if TERM call and a DL/I task termination is not required (TCADLITE is not on).		ISUTERM
4.		FREEPST
5. The ACT table entries are searched to find the entry name with the program name that is in the PPT entry. The address of the ACT entry found is stored into TCADLIPA. If an ACT entry is not found and this is not a system task (TCADLPAS) DFHPC TYPE=ABEND macro is called with abend code = DACT.		TRMACTLP

Extended Description	Routine	Label

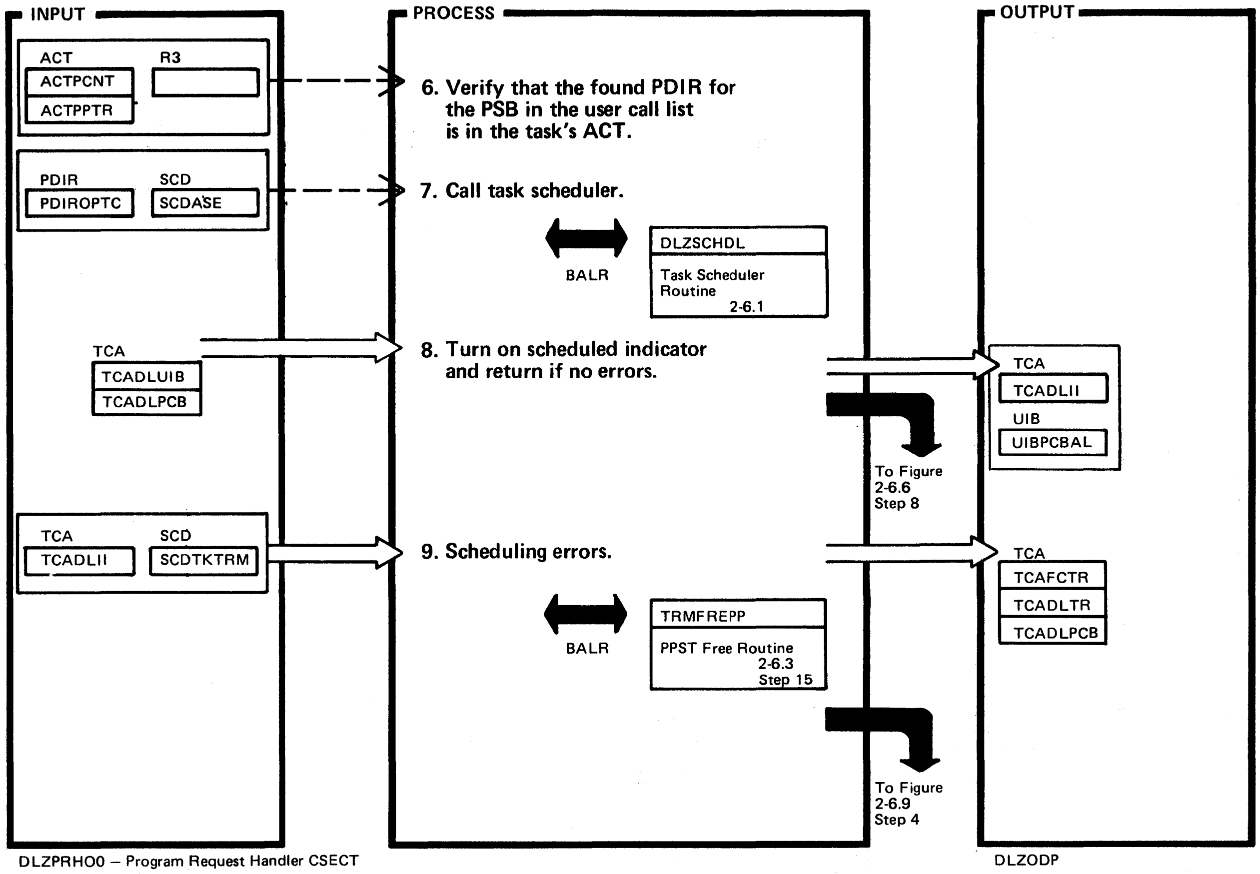
Figure 2-6.12. PCB or PCBM Scheduling (Part 1 of 2)



Extended Description	Routine	Label
1. Set 04 in TCADLTR if PL/I with no count field.		GETPSBN
2. Set 03 in TCADLTR if the task is currently scheduled (TCADLISD is on). Set 02 in TCADLTR if the task is not a DL/I task (TCADLISI). Set 06 in the TCADLTR if the PSB name is too long (PSBNAME has a maximum of 7 characters with a blank in 8th position) or if there is no name specified.		PSBNODEF
3. Set 01 in TCADLTR if the PSB name is not in a PDIR. If PSB name='*b̸', use default PSB name.		PSBEFOND
4. The first PDIR pointer is determined from the task's ACT entry and is used to generate the PSB name.		DEFPSBSC

Extended Description	Routine	Label
5. If call function code is PCBM (MPS scheduling) or program name is DFHMIR (ISC mirror program), skip language conflict checks because an assembler language routine is issuing scheduling calls for programs written in all supported languages. Set 04 in TCADLTR for language conflict.		DFRSKCNT

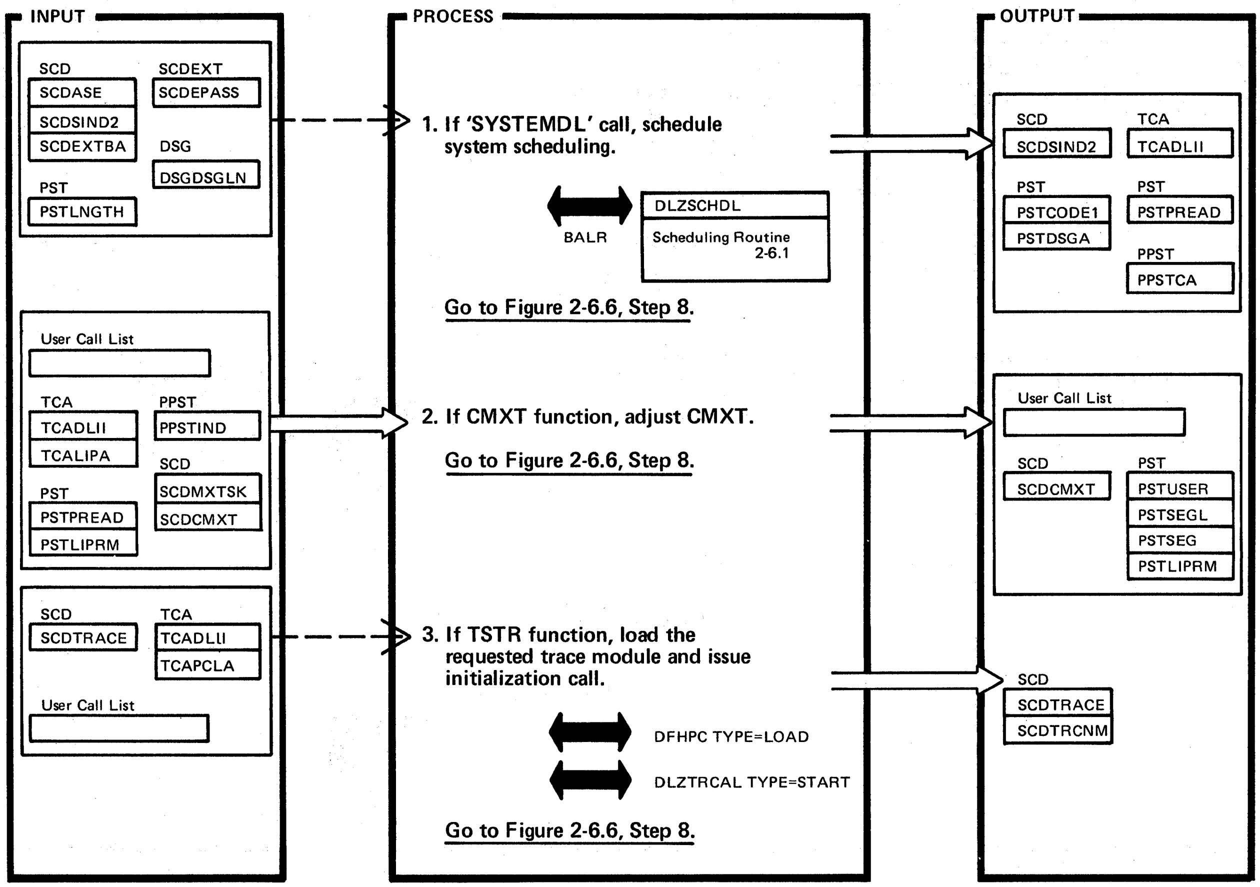
Figure 2-6.12. PCB or PCBM Scheduling (Part 2 of 2)



Extended Description	Routine	Label
6. Set 06 in TCADLTR if the PSB is not in the task's ACT entry.		DFRSPSBC
7. Set 05 in TCADLTR if PCB failed to initialize.		DFRSPOK
8. Turn on TCADLISD to indicate the task is scheduled.		PRHRETN
If UIB is used, update UIBPCBAL with PCB address list address.		
9. For errors where the caller was already scheduled and no PST storage was acquired, return directly to Figure 2-6.6, Step 8.		DFRSERRS

Extended Description	Routine	Label

Figure 2-6.13. System Scheduling Call (Part 1 of 4)



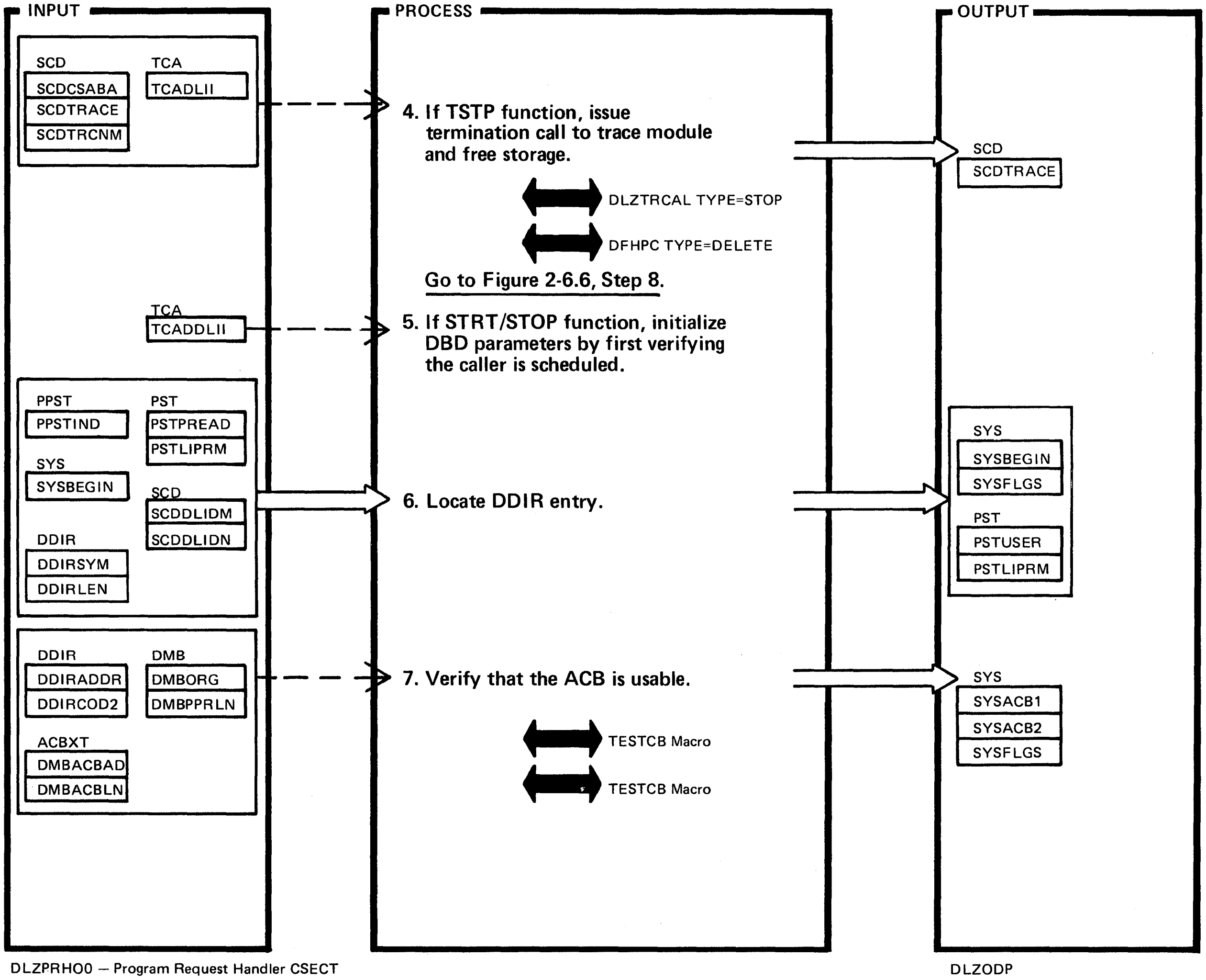
DLZPRHOO – Program Request Handler CSECT

DLZODP

Extended Description	Routine	Label
1. A task requesting services through the system calls must have been previously scheduled by password with this special schedule PCB SYSTEMDL call. If the password does not match, the caller abends via DFHPC with code DLPV. Important indicators set are: SCDSYACT – system interface active TCADLITE – termination required TCADLPAS – system task scheduled TCADLISD – task scheduled Exit is taken to scheduling routine to get a PST and initialized upon return. PSTSCALL (system call in progress) is set and return is made to caller.		PROCSYS
2. The value passed by the user validated and moved to the SCD.		PROCMXT

Extended Description	Routine	Label
Without MPS, data is moved to the user call list area for work space. With MPS this would cause a storage protection exception. To avoid this, an area in the PST (PSTLIPRM) (in the CICS/VS DL/I partition) is used as a work area. The MPS batch program request handler (DLZMPI00) then moves the data from the PST into the user call list. If the new request is zero, negative, or exceeds MAXTASK indicate an invalid request error (set 08 in TCAFCTR).		
3. If task not scheduled for system calls (TCADLPAS not on) abend via DFHPC. If tracing is already active set X'01' in TCAFCTR. If the load fails, set X'02' in TCAFCTR. If GETMAIN fails during initialization, set X'04' in TCAFCTR.		PROCTSTR

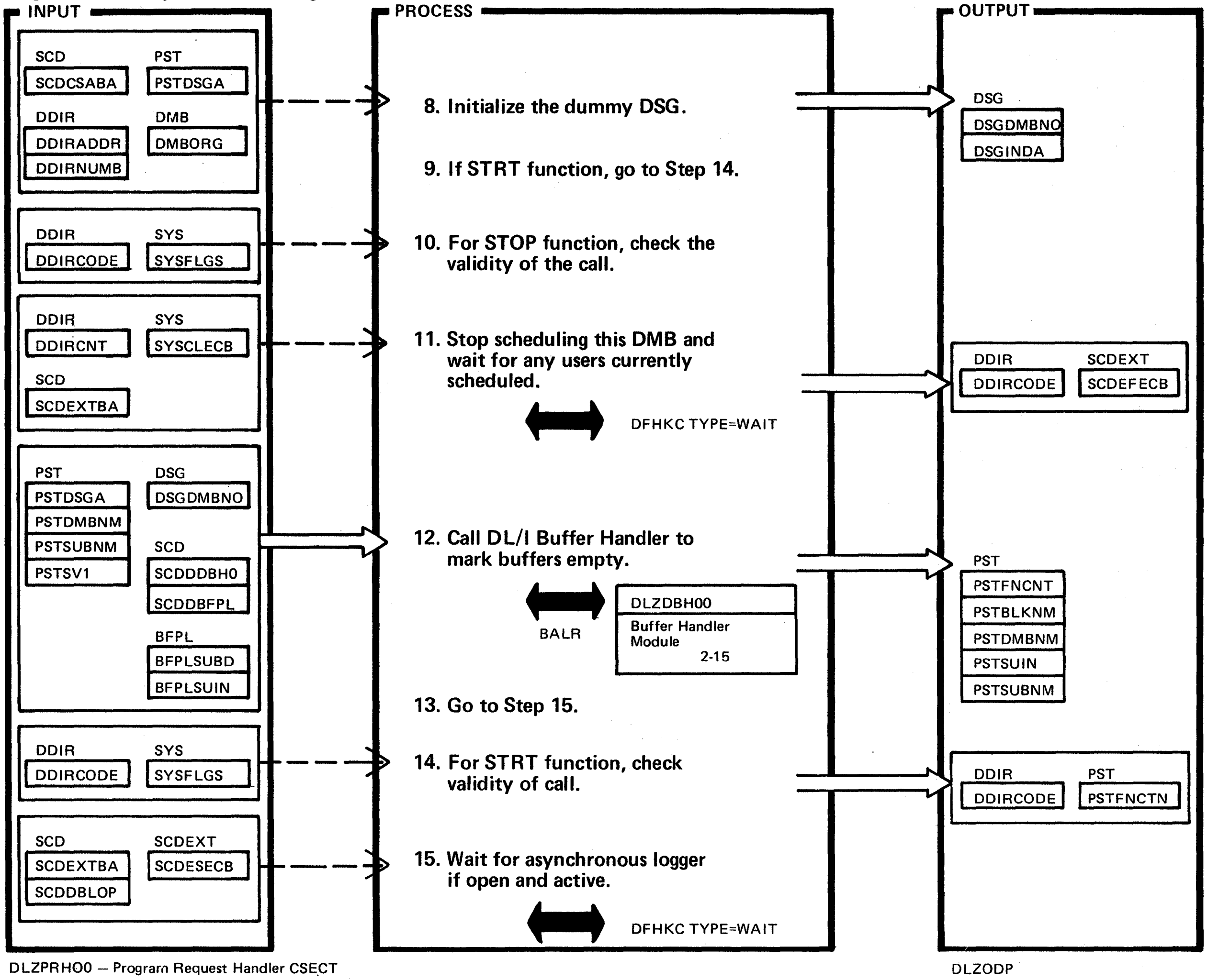
Figure 2-6.13. System Scheduling Call (Part 2 of 4)



Extended Description	Routine	Label
4. If task not scheduled for system calls (TCADLPAS not on) abend via DFHPC. If tracing not active, set X'01' in TCAFCTR for invalid request.		PROCTSTP
5. If task not scheduled for system calls (TCADLPAS not on) abend via DFHPC.		PROCINIT
6. The DMB name passed by the caller is used to scan the DDIR. If this is not an MPS task, data (VSAM return code and ACB address) is moved into the user call list for the caller. With MPS, a work area is used in the PST (PSTLIPRM) to build the data from the PST to the user call list. If the DDIR is not found, set X'08' in TCAFCTR to indicate an invalid request.		PROICON

Extended Description	Routine	Label
7. The ACBs are checked for open/close status. The ACB address (2 if HISAM organization) and whether the ACB is open or not is put into the user call list (or if this is an MPS task, into the PST). Reference to fields within either of these two areas is by the system call parameter list DSECT (DLZSYSDS). If the DDIR failed to initialize, set X'02' in TCAFCTR. If TESTCB request fails, set X'03' in TCAFCTR.		PROCACB

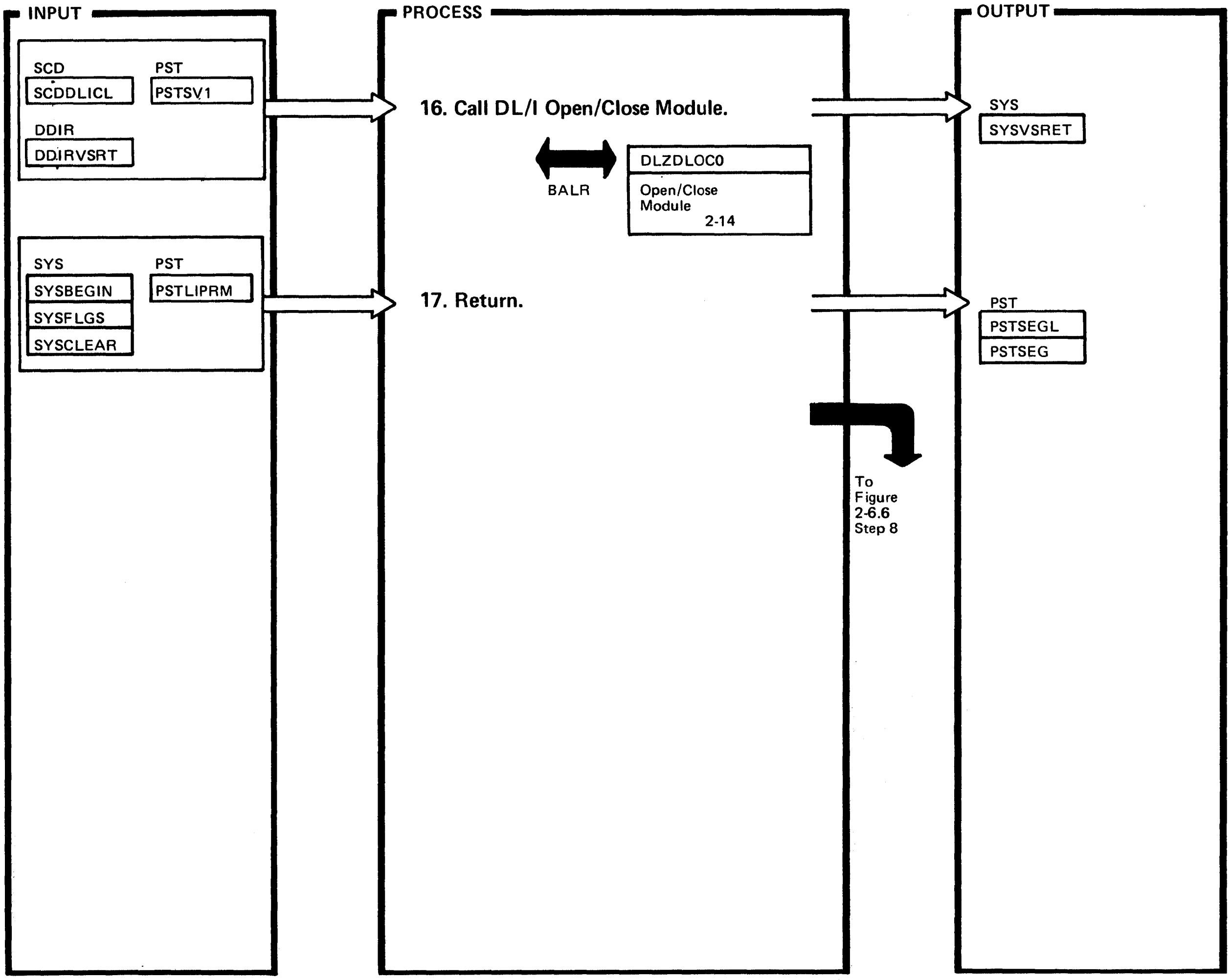
Figure 2-6.13. System Scheduling Call (Part 3 of 4)



Extended Description	Routine	Label
8.		PROCDSG
10. If the ACB is not open, set X'01' to indicate the STOP call is invalid.		PROCSTOP
12. Set X'04' in PSTFNCNT to indicate 'mark buffers empty request to buffer handler'. The DMB number is used to index into the subpool directory to get the subpool number for this DMB. If there is no subpool number, bypass calling the buffer handler. Set X'01' in PSTFNCNT to indicate 'close DMB request' to open/close module upon return from buffer handler.		GOTOBUFF
14. If this data base is not stopped or if the ACB is open, X'01' in TCAFCTR to indicate the STRT call is invalid. Set X'09' in PSTFNCTN to indicate 'open DMB request' to the open/close module.		PROCSTRT
15.		PROCOCR

Extended Description	Routine	Label

Figure 2-6.13. System Scheduling Call (Part 4 of 4)



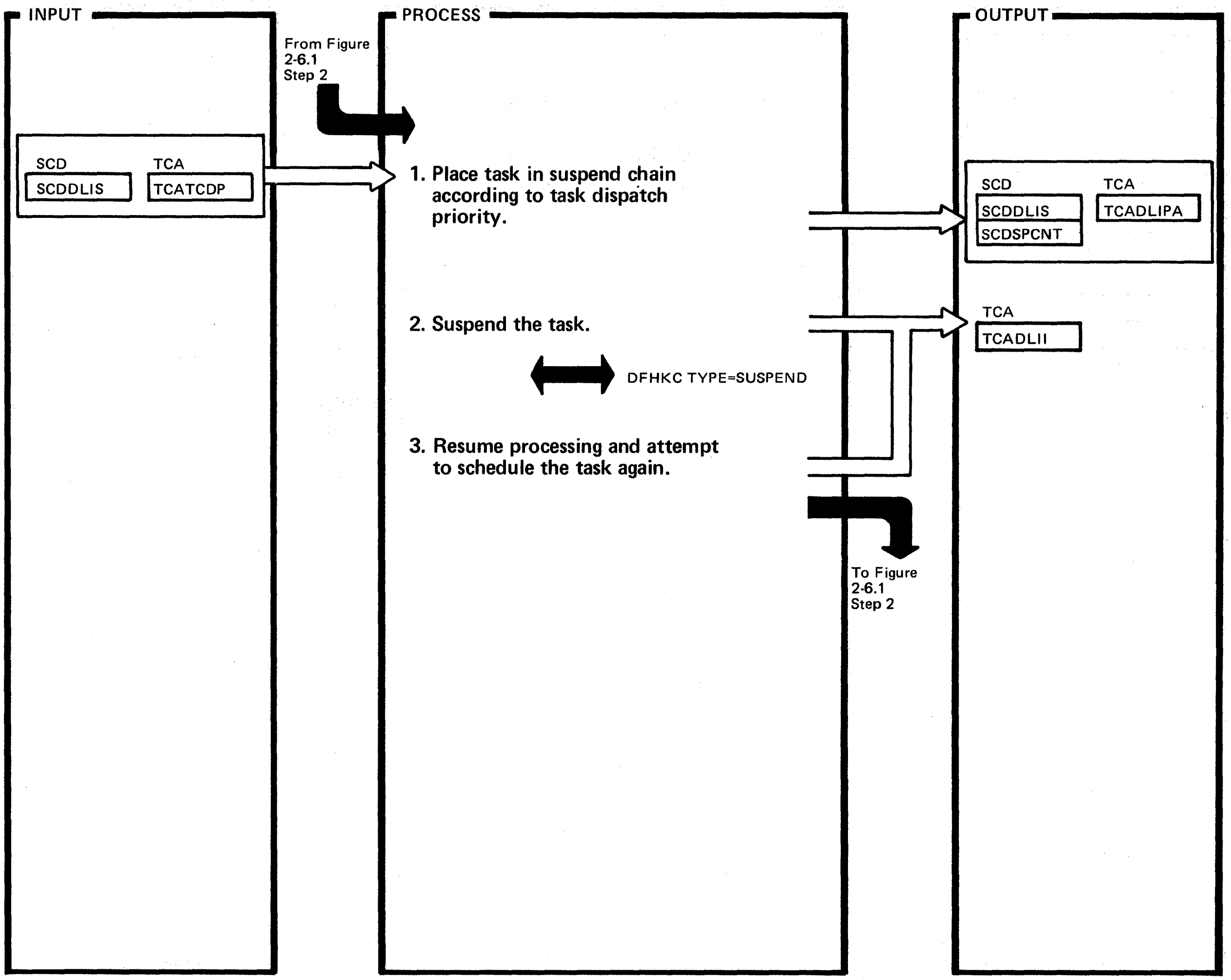
DLZPRHOO – Program Request Handler CSECT

DLZODP

Extended Description	Routine	Label
16. The open/close module issues and SVC2 and CICS/VS loses control for the duration of the call. The VSAM return code in the DDIR upon return from the buffer handler, is moved to the user call list area (or if this is an MPS task, into the PST) which is mapped by DLZSYSDS – the system call parameter list DSECT.		BYSYSWAT
17. If this is an MPS task, PSTSEGL and PSTSEG must be set up for MPS batch PRH (DLZMPI00) to move data from PSTLIPRM to the user call list area. PSTUSER already contains the address of user call list area.		RETURNB

Extended Description	Routine	Label

Figure 2-6.14. Suspend Task Processing



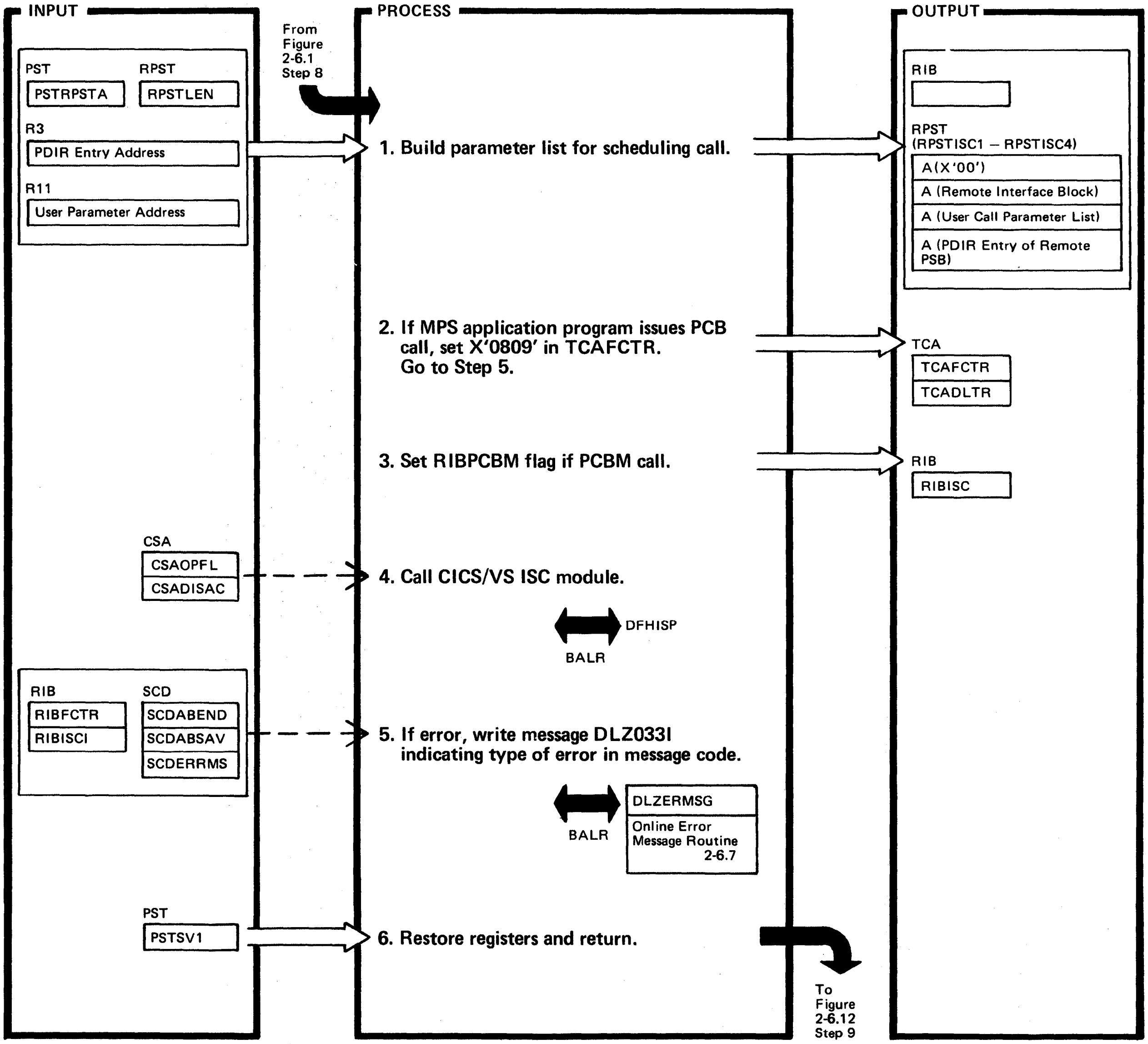
DLZODP – Task and Resource Scheduling CSECT

DLZODP

Extended Description	Routine	Label
1		TASKSUSP
2. Turn on task suspended by DL/I indicator (TCADLIST).		SUSPONE
3. Task has been resumed by DL/I task termination (DLZODP01). Turn off the task suspend indicator (TCADLIST) and attempt to schedule again.		

Extended Description	Routine	Label

Figure 2-6.15. Remote Scheduling Call Interface Routine



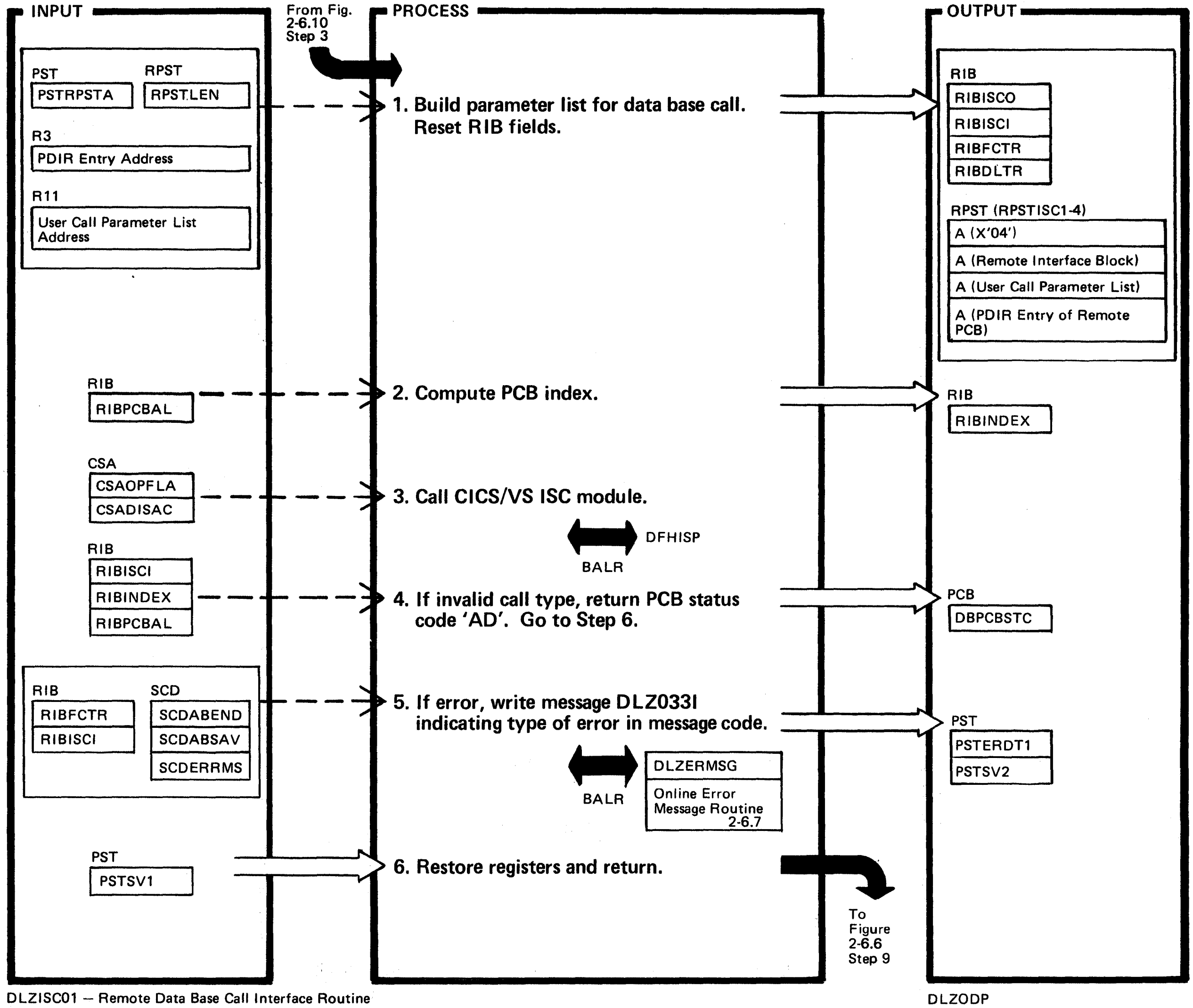
DLZISC00 – Remote Scheduling Call Interface Routine

DLZODP

Extended Description	Routine	Label
1.		DLZISC00
2. If CSADISAC=0 (no DFHISP module available), write message DLZ033I indicating no ISC module found.		ISCNOMOD
4.		ISCBALR
5. Abend task after writing message if user call parameter list is invalid, function string is invalid, or internal error detected.		ISCRIBER
6.		ISCRET00

Extended Description	Routine	Label

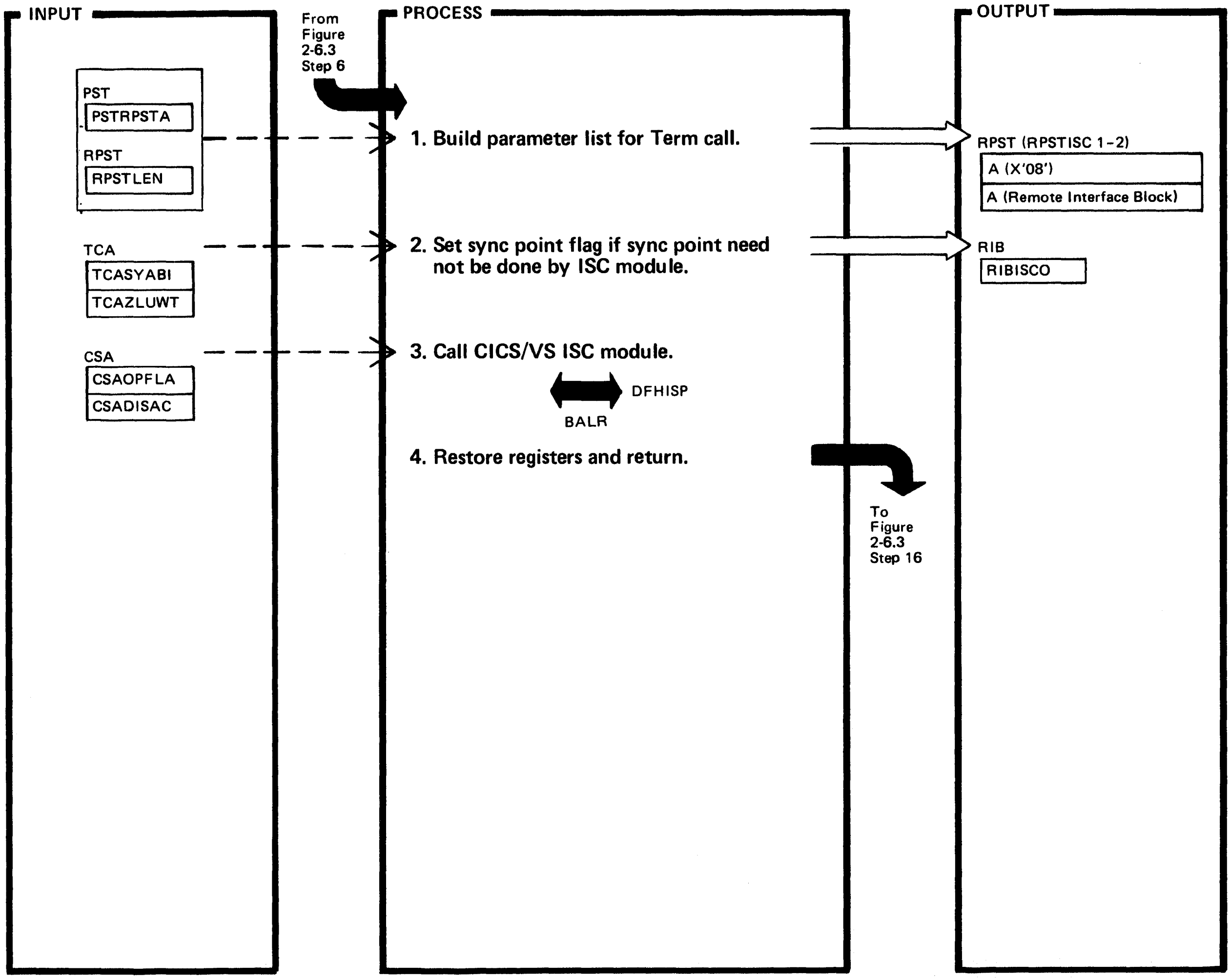
Figure 2-6.16. Remote Data Base Call Interface Routine



Extended Description	Routine	Label
1.		DLZISC01
2. The PCB address specified by the user is the address of a local copy of the real PCB in the remote system. It must be converted to a PCB index to identify the corresponding PCB in the remote system.		ISCINDEX
If PCB index cannot successfully be computed, write message DLZ476I and abend task.		ISCNONDX
3.		ISCBALR1
4.		ISCFUNC
5. Abend task after writing message if user call parameter list is invalid, link with remote system is out of service, or an internal error was detected.		ISCRIBC1
6.		ISCRET01

Extended Description	Routine	Label

Figure 2-6.17. Remote Termination Call Interface Routine



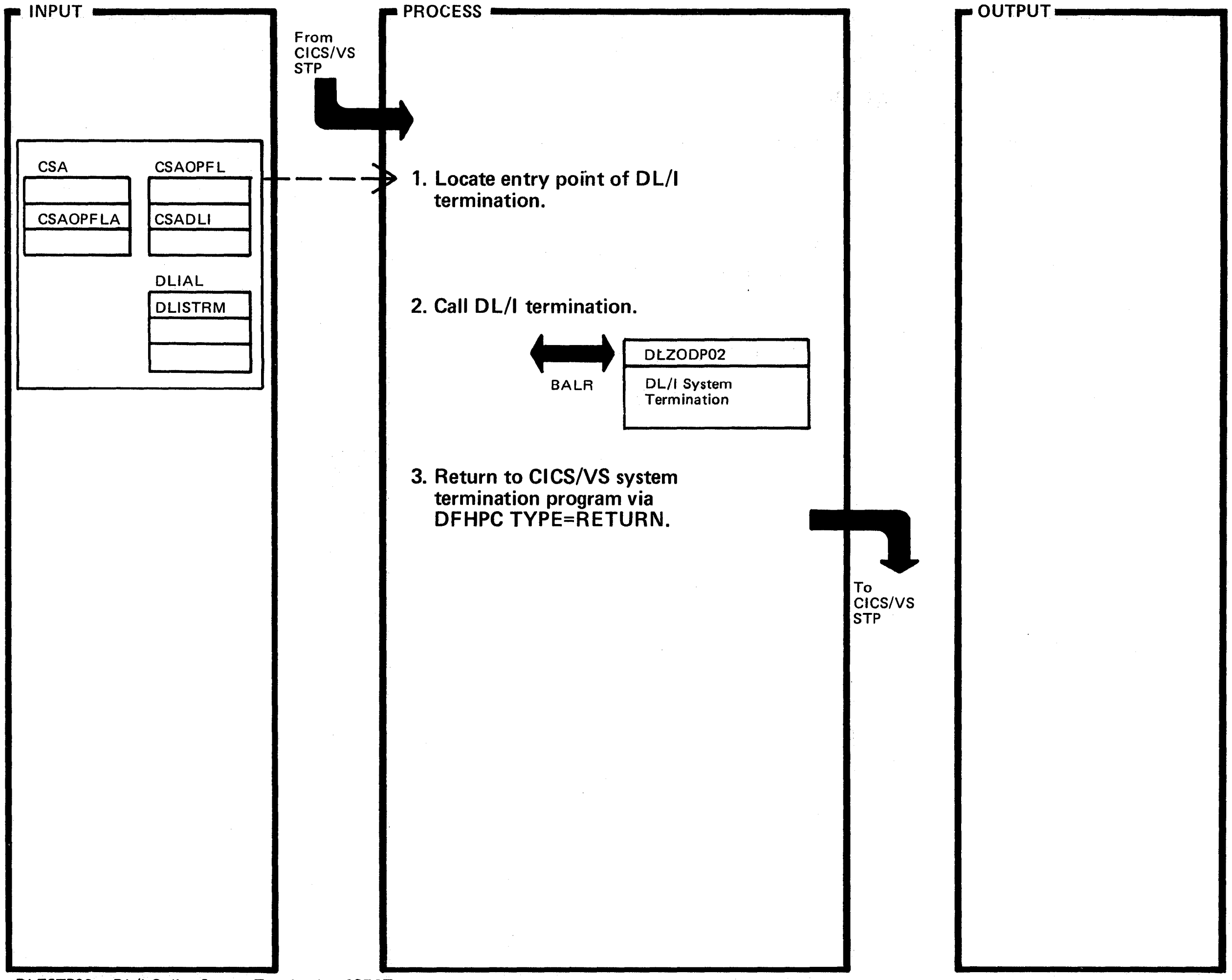
DLZISC02 — Remote Termination Call Interface Routine

DLZODP

Extended Description	Routine	Label
1.		DLZISC02
3.		ISCBALR2

Extended Description	Routine	Label

Figure 2-7. DL/I Online System Termination



DLZSTP00 — DL/I Online System Termination CSECT

DLZSTP00

Extended Description	Routine	Label
1. Control is gained from CICS/VS System Termination Program (STP) because of DLZSTP00s presence in the program list table (DFHPLT).	DLZSTP00	DLZSTP00

Extended Description	Routine	Label

Figure 2-8. Call Analyzer

INPUT

R1 PST-ADDR

R13

PSTIQPRM

Parameter List

- Function
- PCB-ADDR
- I/O AREA
- SSAs

From Caller (See Note)

PROCESS

1. Save registers and initialize.

2. Encode function:

A. Normal Function — Figure 2-8.1.

B. Pseudo function — Figure 2-8.3.

3. Update JCB trace and PCB segment name, level and key-length. Restore registers.

Return to Caller

OUTPUT

R13

PSTSEG PSTSEGL 00000000

JCB

R13

PCB

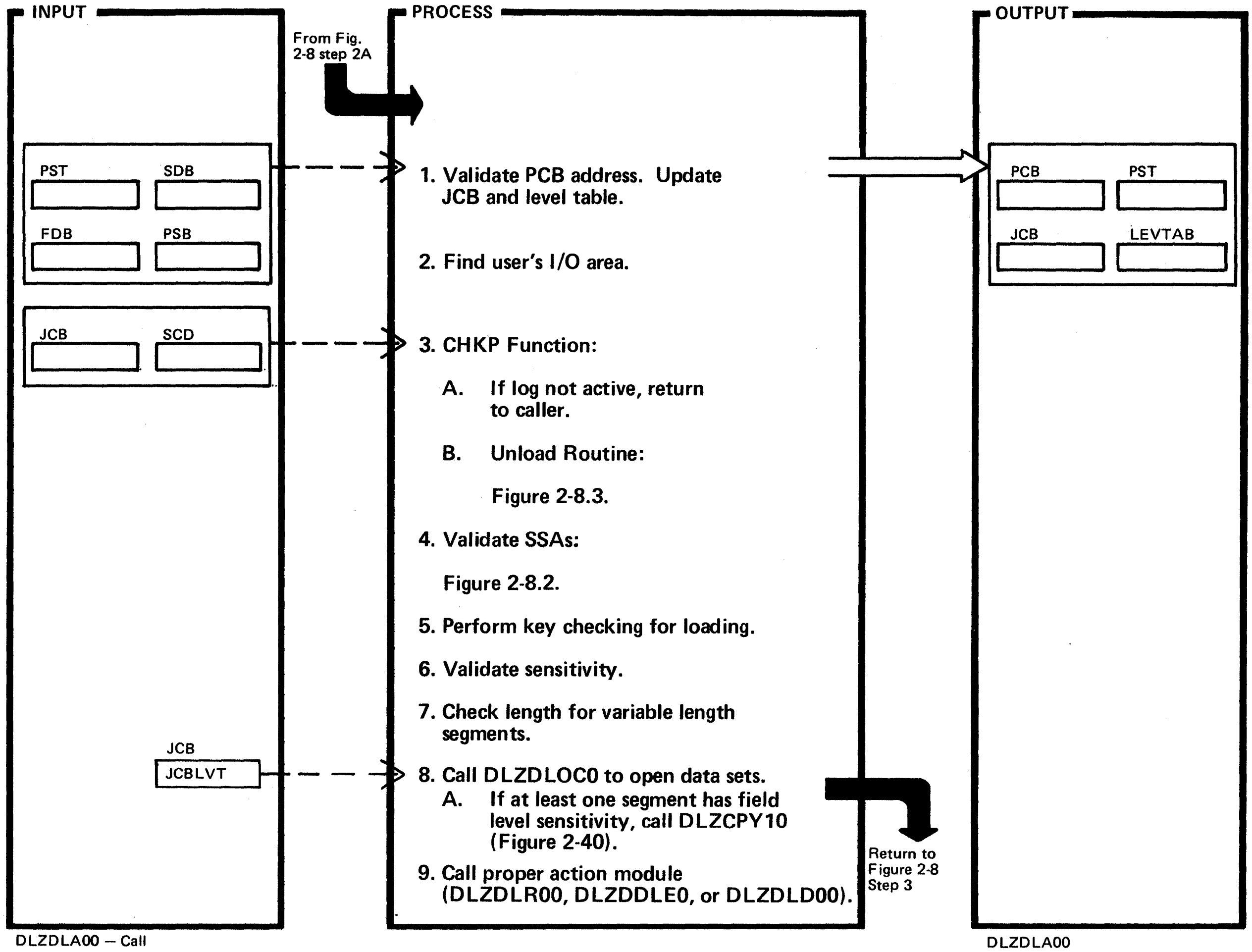
DLZDLA00 — Call

DLZDLA00

Extended Description	Routine	Label
Note: DLZDLA00 is called from the program request handler (DLZBNUCO-DLZPRHB0) in a batch system, from (DLZODP-DLZPRHO0) in an online system, or if at termination, it is called from either the application program control (DLZRRC00-DLZPCC00) or from online task termination (DLZODP-DLZODP01). It is also called from DLZDXMT0.		
2. The function (first parameter in list) is encoded. If no valid function is found, 'AD' status is returned.		
Normal functions are GU, GN, GHN, GHU, GNP, GHNP, DLET, REPL, ISRT, ASRT, and CHKP.		
Pseudo functions are GSCD, UNLD, and TERM.		

Extended Description	Routine	Label

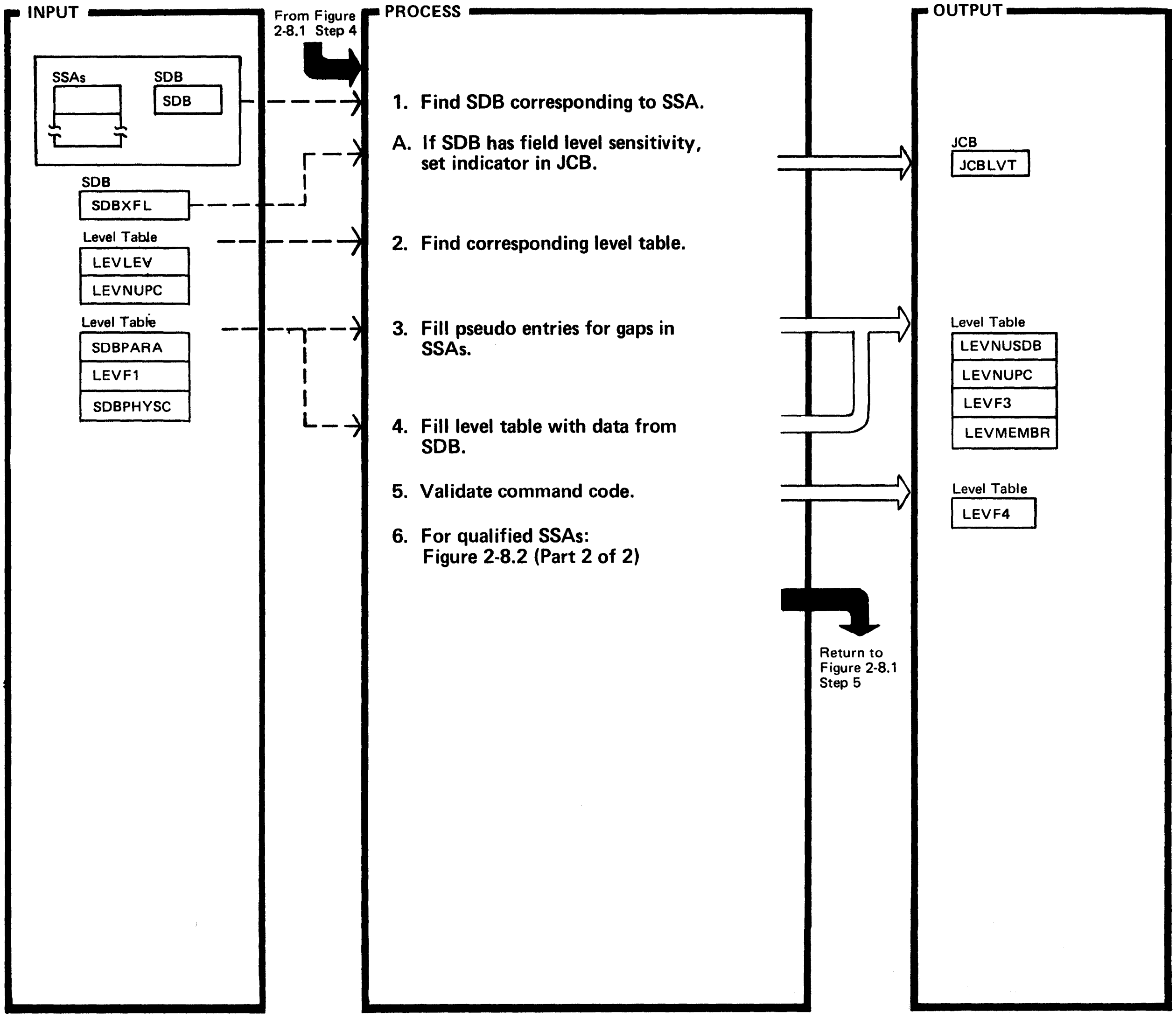
Figure 2-8.1. Call Analyzer — Normal Function



Extended Description		Routine	Label
1.	If no valid PCB address is provided, abend code '476' is returned. The JCB and PCB are updated and the second part of the level tables cleared.		TESTPCB VALIDCK2 DBPCBFND GETJCB
2.	If no I/O area is provided, 'AB' status code is returned.		
3A.	If log is not active, return to caller with 'XH' status code in the PCB. The function call is ignored.		
3B.	Purge all buffers.	DLZDLA01	DLBUNLD
4.	All SSAs in the call are checked.		SDBLOOP SDBLOOP1
5.	Key checking is done for load mode and the last SSA of an ISRT call. For PROCOPT=LS and for HISAM, the root key is compared to the previously loaded root. Status code 'LB' indicates invalid sequence.		LDCHCK
6.	Sensitivity checking is done for ISRT, DLET, and REPL calls. Violations return 'AM'. Extra checking is done for DLET and REPL calls, if successful GH call was executed before 'DJ' status.		NOTLOAD7 FSTDATAL ISREPL TSTISRTS

Extended Description		Routine	Label
7.	For variable length segments, 2-byte field in the user I/O area is compared to the maximum length and to the key+ keyoffset. If it is greater or smaller, 'V1' status is returned.		DOVLTST
8.	When the data base that the PCB references is not open, DLZDLOC0 is called to open all data bases related to this PCB.		ANYSEN
A.	If field level sensitivity indicator is set, exit is made to DLZCPY10 to map the user view to the physical view. Only done if ISRT, REPL, or Retrieve (called on behalf of ISRT) action modules will be executed.		
9.	For GET calls, DLZDLR00 is called. For DLET/REPL calls, DLZDLD00 is called. For ISRT/ASRT calls in load mode, DLZDDLE0 is called for all segments except for HIDAM root, where DLZDLR00 is called. For ISRT not load mode, DLZDLR00 is called for all segments except HISAM root, where DLZDDLE0 is called.	-	ACTION

Figure 2-8.2. Call Analyzer — Validate SSAs (Part 1 of 2)



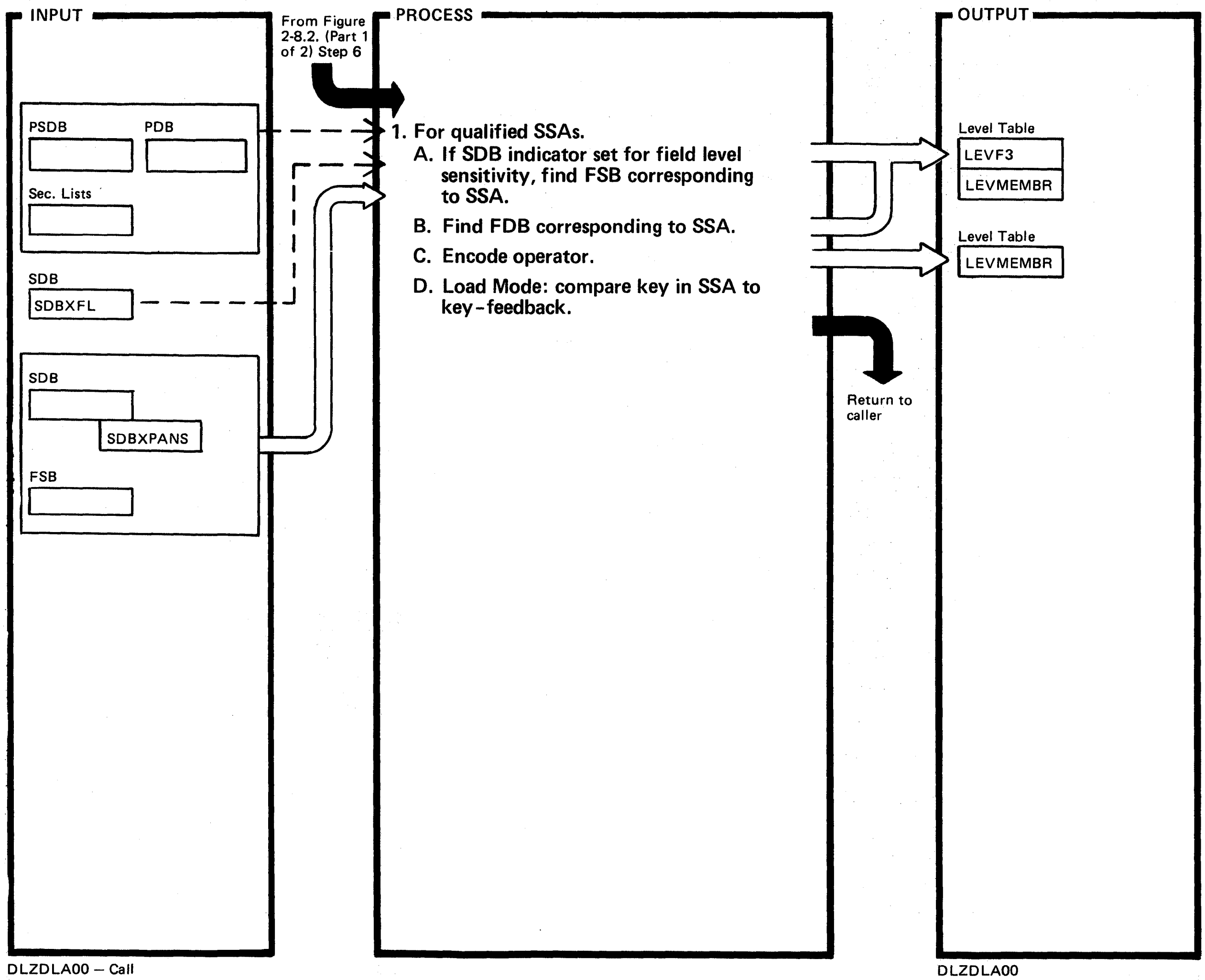
DLZDLA00 — Call

DLZDLA00

Extended Description	Routine	Label
1. When the segment named specified in the SSA cannot be found in the SDB, 'AC' status is returned.		SDBLOOP
A. Flag SDBFLS is on in field SDBXFL if SDB has field level sensitivity. Flag JCBFLS is set on in field JCBLVT to indicate at least one segment in call has field level sensitivity.		SSASDBEQ
2. When a hierarchy error is detected, an 'AC' status is returned.		GETLEV RIGHTLEV
3. The levels corresponding to gaps in the SSAs are filled with data from the previous call. For loading, no parent level may be empty. 'LD' status is returned.		
4. Extra checks are made for DLET and REPL calls. When no GH call was previously made for this SDB, a 'DJ' status is returned.		

Extended Description	Routine	Label
5. Valid command codes are C, D, F, L, N, Q, T, and X. The status code for invalid command code is 'AJ'. For D call and no path sensitivity, the status code is 'AM'.		NOTDORR

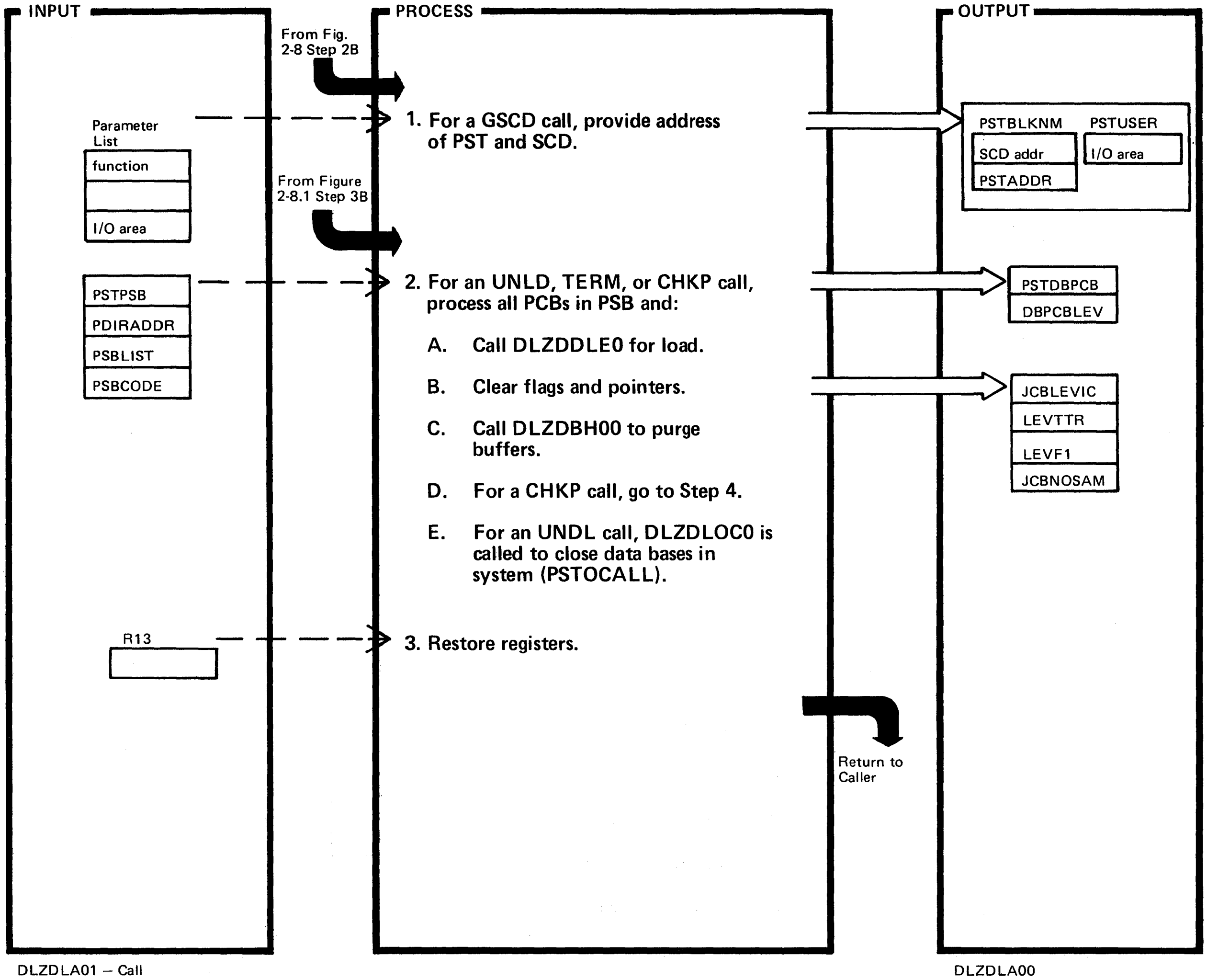
Figure 2-8.2. Call Analyzer — Validate SSAs (Part 2 of 2)



Extended Description	Routine	Label
1. For errors in qualification, format 'AJ' status is returned.		
A. Flag SDBFLS is on in field SDBXFL if SDB has field level sensitivity. If an FSB is not found, or if the FSB is not marked as an allowable field, status code 'AK' is returned in PCB.		
B. Valid field names are any normal field of the segment, the XDFLD name (if the secondary processing sequence is used). For a concatenated segment field, names of the logical child and the destination parent are valid. 'AK' status for invalid field name. 'AC' status if /CK or /SX is used.		NXTBOOL PDBEQUAL
C. Invalid operator returns status code 'AJ'		CODES ROHIT
D. If qualified SSAs are specified for loading, the key has to correspond to the key-feedback area, otherwise 'LD' status code is returned.		

Extended Description	Routine	Label

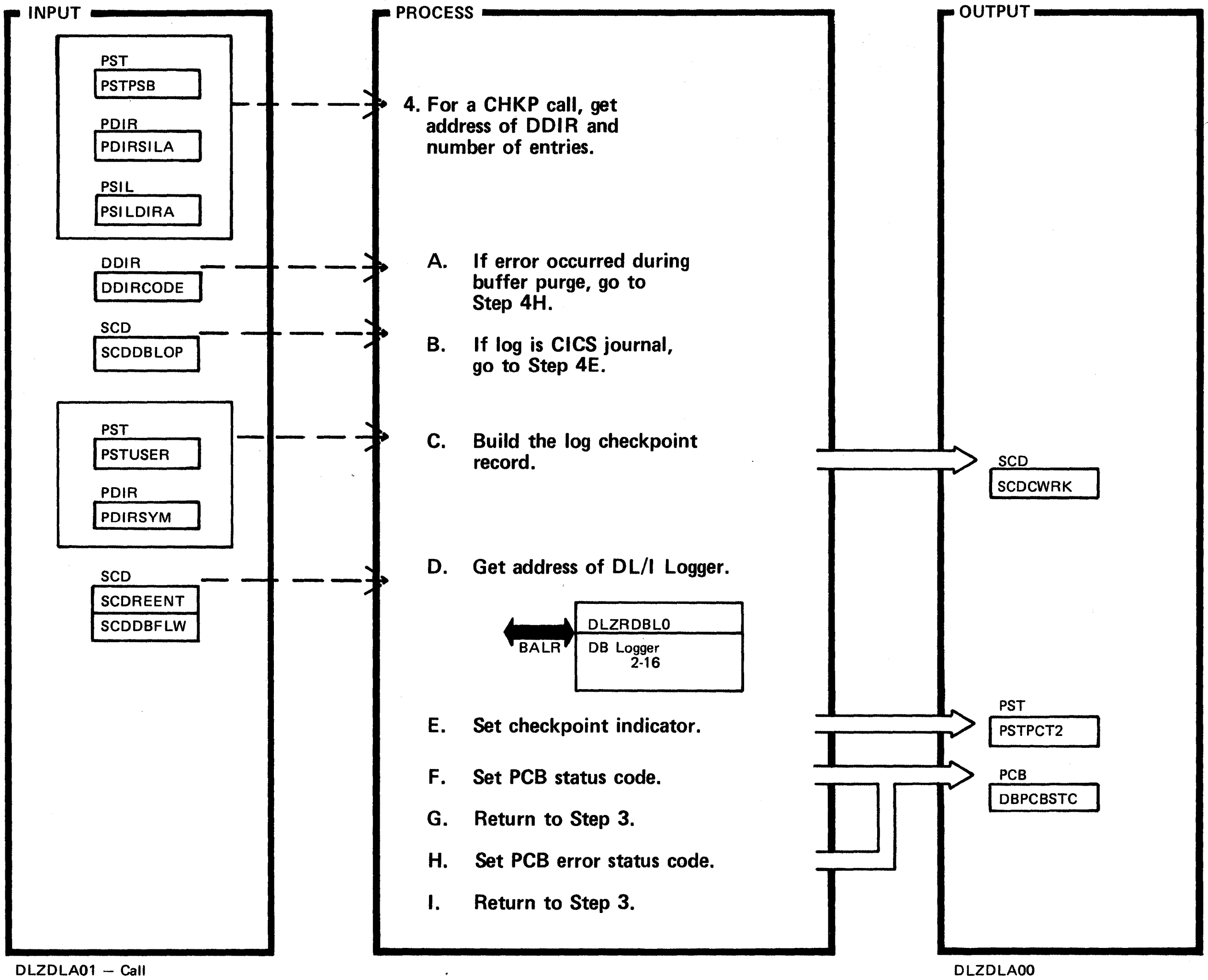
Figure 2-8.3. Call Analyzer — Pseudo Function (Part 1 of 2)



Extended Description	Routine	Label
1. Input to the GSCD call is function and I/O area address. DLZDLA01 puts the SCD and PST addresses in PSTBLKNM.Program request handler moves it to the I/O area.		PSEUDOCA
2. The TERM call is issued in online to end a task. The UNLD call is issued in batch to end the batch program.		DLBUNLD UNLDLOOP
A. If the UNLD call is made for load mode, DLZDDLE0 is called to write the last records for HSAM and HISAM. For HISAM and index data bases, a record is written with FF keys.		
B. Flags and pointers are cleared so that the PSB can be used by another task. If program isolation is active, clear all enqueue indicators in all level table entries.		
C. All user buffers are written to the data base now. RSTBLKNM, DMBNM, and ACBNM are cleared. PSTPGUSR flag of PSTFNCTN is set.		

Extended Description	Routine	Label
3. If an error occurs during the purge of the buffers, an 'XD' status code is returned in the PCB.		

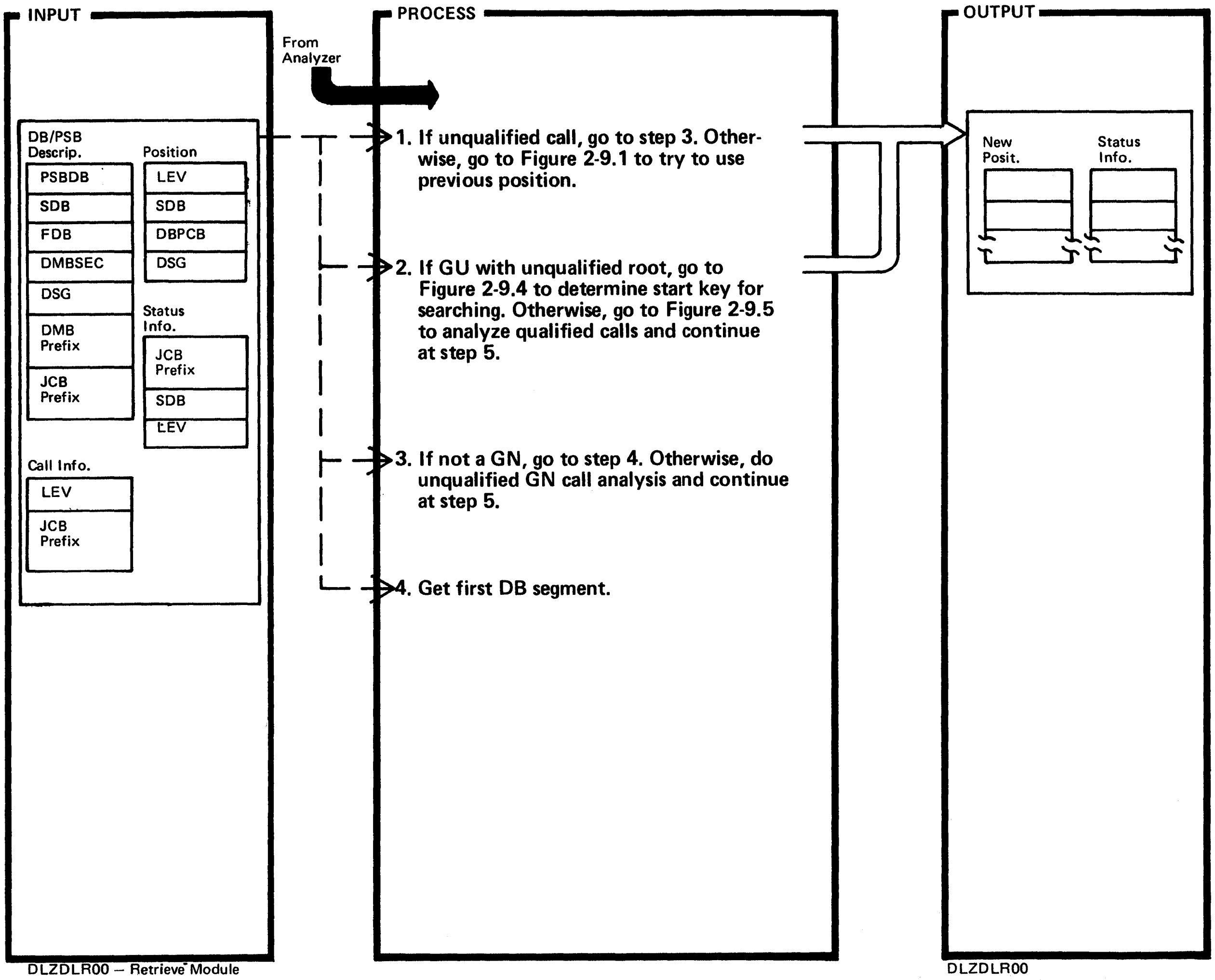
Figure 2-8.3. Call Analyzer – Pseudo Function (Part 2 of 2)



Extended Description	Routine	Label
4. A. All DDIR entries are scanned to see if an error occurred (bit DDIRNOSE X'04' in byte DDIRCODE set on) during purge of the buffers.		DDIRCHK
B. If the log is the CICS journal, checkpoint record is not written, but a CICS synch point is.		DDIRCHK1
D. The DL/I Logger is entered twice; 1st to move the checkpoint record to its buffer, and 2nd to force-write the checkpoint record.		
E. On return from the logger, the checkpoint indicator (bit PSTCHKP X'04' in byte PSTPCT2 is set on) to notify the program request handler to issue the checkpoint message.		BYPASSCK

Extended Description	Routine	Label
F. Status code of 'blanks' is set in the PCB indicating successful completion of the CHKP call.		
H. Status code of 'XD' is returned in the PCB indicating an error occurred during checkpoint processing.		DDIRER

Figure 2-9. Retrieve (part 1 of 2)



Extended Description	Routine	Label
1. I/O information: • The Position block includes RBA of segment (HD) or lrec (HS), RBA of previous and next positions (HD), offset to segment from begin lrec (HS), concatenated key, level, block number (HSAM), and block number and RAP number of current RAP (HDAM). RAP = root anchor point. • The DB/PSB Description block includes segment and data set descriptions, data base specifications, sensitivity, and HDAM randomizing facility. • The Status Information block includes prior status codes, segment status, and (for output) pseudo abends (801 and 800). • The Call Information block includes SSA and call type. • Processing starts with initialization. Level of previous call stored in LASTLEV.		LTWSSA

Extended Description	Routine	Label
2.		XLTFINDR
3.		MTNOSSA NOSSA

Figure 2-9. Retrieve (Part 2 of 2)

INPUT

DB/PSB Descrip.: PSBDB, SDB, FDB, DMBSEC, DSG, DMB Prefix, JCB Prefix

Position: LEV, SDB, DBPCB, DSG

Status Info.: JCB Prefix, SDB, LEV

Call Info.: LEV, JCB Prefix

PROCESS

5. For ISRT calls:

   A. Determine positioning and move segment to user.

   B. Call DLZGETS (Fig 2-9.7) to make retrieved data available to user.

6. Exit.

To Analyzer or Load Insert

OUTPUT

SDB: position for insert, SUBPOSC, SUBPOSN

PST: PSTSEG, PSTSEGL

DLZDLR00 – Retrieve Module

DLZDLR00

Extended Description	Routine	Label
5. DLZSSA is called, if necessary, to find insert position for key. Control is then passed to DLZISRT to prepare position information in the SDB for INSERT. Return is to DLZGETS (Figure 2-9.6).		RETURNIE NOTEOD
PSTSEG is address of data, PSTSEGL gives its length.		
For IRST calls DLZGETS does only housekeeping (no data moving). DLZGETS will pass control to DLZRETN and DLZDLR1 to exit.		ARETURNA
For a segment with logical relationship, DLZGETS will call DLZLOGR for data move/insert positioning.		
6. If call type is GET, go to Analyzer. If it is ISRT, go to Load/Insert.		

Extended Description	Routine	Label

Figure 2-9.1. Retrieve – DLZLTW Routine

INPUT

LEV

position, call info.

From Fig. 2-9 step 1

PROCESS

1. Set KEEPIT=1.
2. Check previous call's hierarchical path against SSA's. Loop through levels. Check segment type, and for qualified SSA, check key feedback area (Figure 2-9.2).
3. If path is accepted, locate previous segment (Figure 2-9.3), set KEEPIT=0, and go to step 6.
4. If discrepancy at root level, set KEEPIT=0 and go to step 6.
5. If discrepancy at lower level, set KEEPIT=0 and go to step 6.

   If GN call, position to previous segment.
6. Exit.

To Fig. 2-9 Step 2

OUTPUT

LEV

JCB: JCBLEVIC

R6 — SDB: SDBPOSN

R6 — SDB: SDBPOSN

DLZDLR00 – Retrieve Module

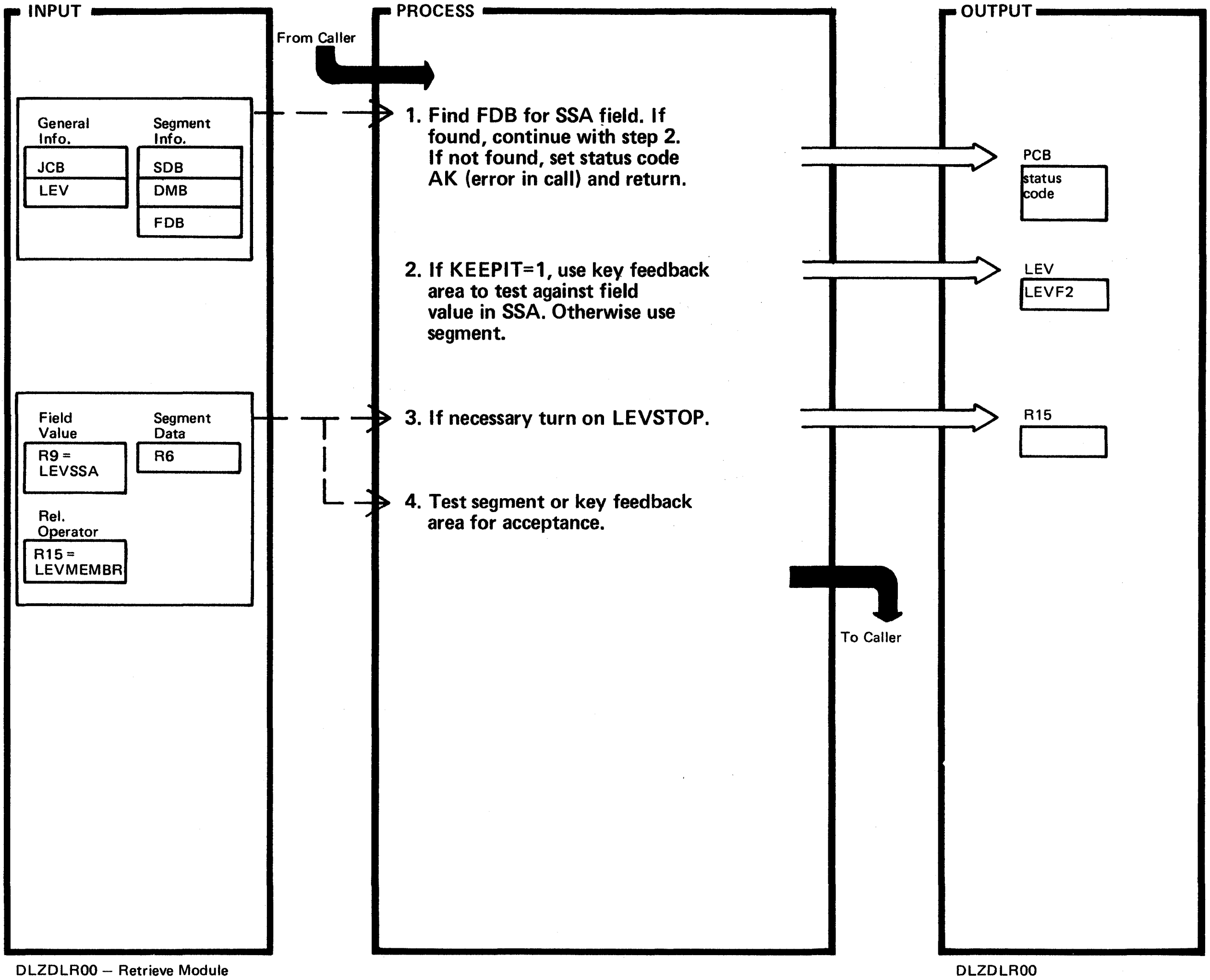
DLZDLR00

Extended Description

Extended Description	Routine	Label
1. KEEPIT=1 indicates: try to use previous position. KEEPIT=0 indicates: DLZLTW has been left. Other values have special meanings. (Entry point when R15 = 0.)		LTWSSA
2. DLZKDTE is invoked via DLZSSA which is called by return to DLZDLR0 and back to DLZLTW. Logically, this is part of DLZLTW as indicated by KEEPIT=1.		
Qualified SSA test: After entering several routines, return to DLZLTW entry LTWSSACA, LTWSSAF, or LTWSSAG.		LTWSSAQ
Lowest level found valid is stored in JCBLEVIC.		LTWSSACA
3. Set code for exit: Entry UNQLA in DLZSSA for GU or ISRT, entry SSAEVALH for GN.		
DLZPCHK loads buffer location of previous segment into register 6 (except for HD or GN calls) and, for HD, loads available SUBPOSN positions.		

Extended Description	Routine	Label
5. Set exit code for entry SSAEVALL in DLZSSA.		

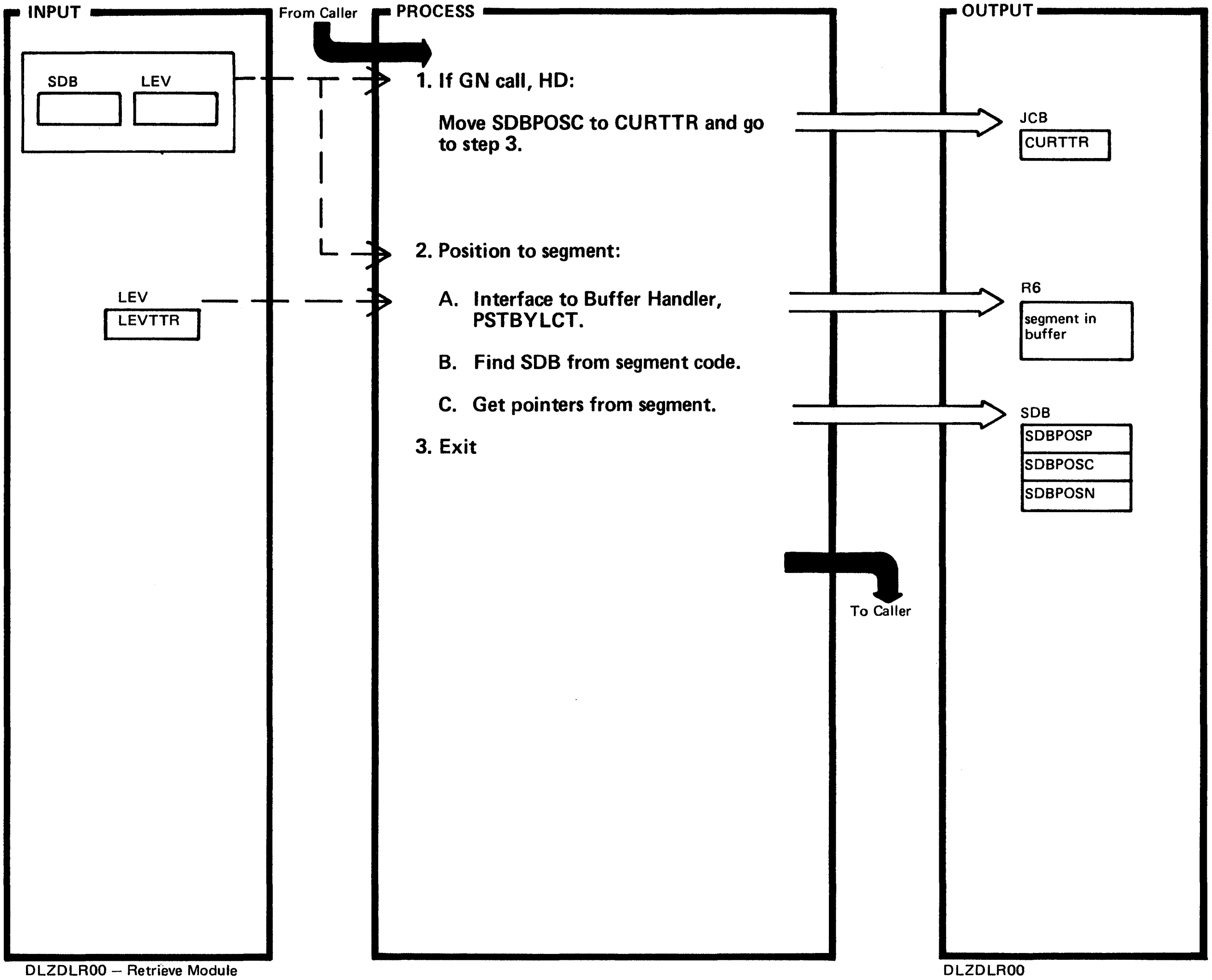
Figure 2-9.2. Retrieve — DLZKDTE Routine



Extended Description	Routine	Label
1.		KDTESTI KDTESTK
2. If logical relationship, build concatenated segment. If variable length, build data.	DLZKDTL DLZVLRT	KDTESTER
3. If qualification is on key, relational operator is greater than or equal to, and key is less than or equal to, SSA.		KDTESTHA
4. If accepted, R15=0, otherwise, R15=4.		KDTESTE

Extended Description	Routine	Label

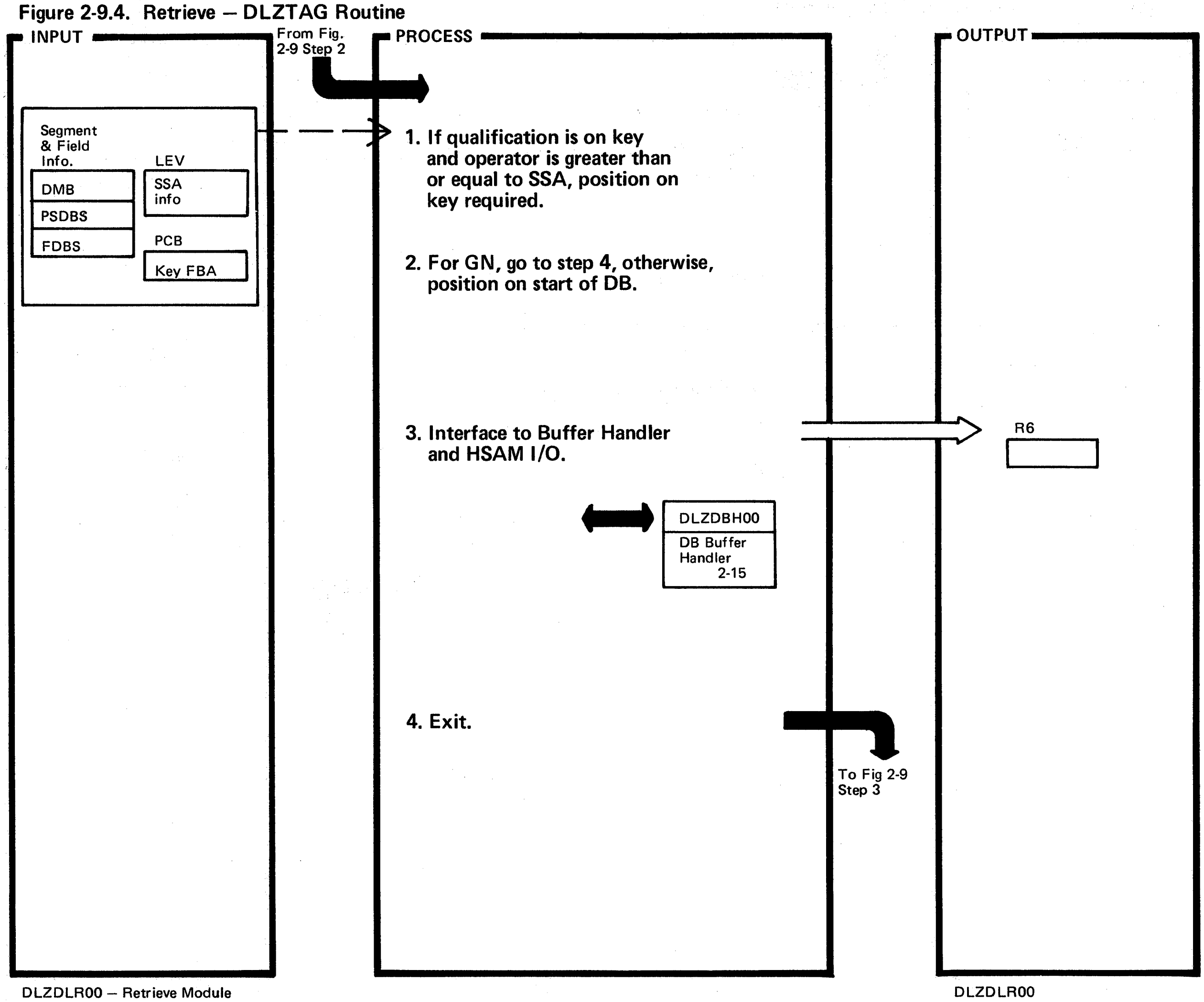
Figure 2-9.3. Retrieve — DLZPCHK Routine



Extended Description	Routine	Label
2. For HSAM, more than 1 PCB: restore position.		POSCHKA
For HISAM: take care of control interval splits.		POSCHKA2
B. If not found (segment not sensitive), turn on LEVDLET and go to step 3.		
C. For HS, relational record number and offset to SDBPOSC. SDBPOSN already posted by DLZSETL.		
For HD, post twin pointers.	DLZPSTN	
Clear dependent positions (SUBPOSP, SDBPOSC, and SDBPOSN). For HD, post child pointers.	DLZPSTA	
For HD logical relation with inverted structure, post child pointers. Subroutine called by DLZPSTA.	DLZAPST	
Clear SDBPOSP, SDBPOSC, and SDBPOSN in preceding sibling SDBs unless multi-processing.	DLZPOSA	

Extended Description	Routine	Label

Figure 2-9.4. Retrieve — DLZTAG Routine



Extended Description	Routine	Label
1. Set code PSTSTLEQ for DLZSETL. Set exit code for entry SSAEVAL in DLZSSA.		MTWISSA
2. Set code PSTSTLBG for DLZSETL. Set exit code for entry SSAEVAL or SSAEVALM in DLZSSA.		NOLL
For GN, set exit code for entry SSAEVALM in DLZSSA.		KPURTC
3. DLZSETL branches to subroutines according to DB organization. R6 points to segment in buffer pool.		

Extended Description	Routine	Label

Figure 2-9.5. Retrieve — DLZSSA Routine (Part 1 of 2)

INPUT

First Part of LEV

Position & status conditions set by RETRIEVE in previous or current call

Start Level

R4 — level where search will start

Second Part of LEV

SSA description set by ANALYZER

DLZDLR00 — Retrieve Module

From Caller

PROCESS

1. If SSA is unqualified, go to step 4.
2. Go to DLZKDTE routine (Fig 2-9.2) to check acceptability of a segment.
3. If segment is not accepted, go to step 9.
4. Update level table.
5. Go to next lower level.
6. If level is not qualified, go to step 10.
7. Go to DLZSKPG routine (Fig 2-9.6) to skip segments.
8. Go to step 1.

OUTPUT

First Part of LEV

Description of last acceptable segment including its hierarchical path

DLZDLR00

Extended Description	Routine	Label
6. Prepare input (segment type, etc.) before entering the central DLZSKPG routine.	DLZSKPG	SKIPGENS

Extended Description	Routine	Label

Figure 2-9.5. Retrieve – DLZSSA Routine (Part 2 of 2)

INPUT

First part of LEV

Position and status conditions set by RETRIEVE in previous or current call

PROCESS

9. Segment is not acceptable.

   A. If LEVCONT is on and not at root level, go to Step 9E.

   B. If LEVCONT is on and at root level, go to Step 9D.

   C. If current level is root, exit to subroutine DLZGER.

      Otherwise, back up one level and return to Step 9A.

   D. Get next root segment.

      DLZSETL — Interface to buffer handler and HSAM I/O

   E. Go to routine DLZSKPG (Fig 2-9.6) to skip to next segment of same type under present root.

   F. Return to Step 1.

10. Exit.

Return to Caller

OUTPUT

DLZDLR00 – DLZSSA Routine

DLZDLR00

Extended Description	Routine	Label

Extended Description	Routine	Label

Figure 2-9.6 Retrieve – DLZSKPG Routine

INPUT

R15

Option code

SDB

SDBORGN

SDBPOSC

SDBPOSN

R6

Buffer location of old segment

JCB

JCBCODE, bit JCBRDREQ

DLZDLR00 – Retrieve Module

From Caller

PROCESS

1. If 'Skip to Next' option, go to step 3.
2. Prepare control input.
3. Skip to next segment.
4. If skip failed, go to step 8.
5. If 'Skip to Next' option, test if segment is sensitive.

   If not sensitive, go to step 3.
6. If 'Skip to Next Segment' option, test segment level and segment code.

   If accepted, go to step 8.
7. If position still before segment searched, go to step 3. Otherwise, indicate failure.
8. Exit.

Return to Caller

OUTPUT

JCB

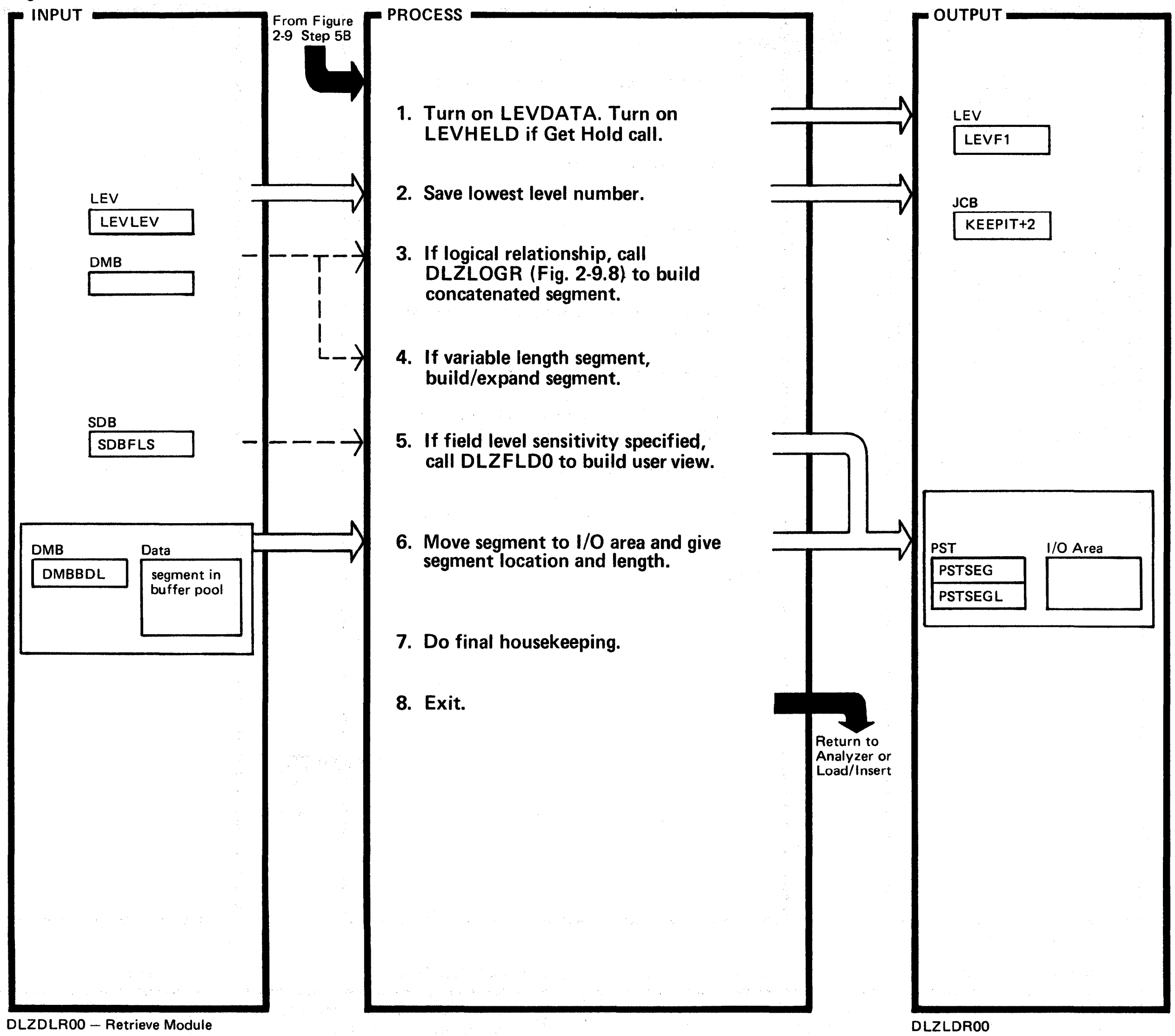
JCBLVT

DLZDLR00

Extended Description	Routine	Label
1. Option is 'Skip to Next Segment' if R15 is positive. If R15 is negative, the option is 'Skip to Specified Segment'.		
2. If JCBLVT = X'02', require segment code, segment level in physical DB, and parentage level.		SKIPGENS
3. For JCBRDREQ off, current segment is examined first.		SKIPGEN
DLZSKPS calls general skip routine DLZSKPE, which calls specific skip routines:		
For HS, DLZSKPD.		
For HD using SUBPOSN, DLZSTLA.		
In some cases (HS, skip to first child of current segment), DLZSKPD is called directly from DLZSKPS.		
4. End of ESDS chain reached for HISAM.		

Extended Description	Routine	Label
7. If segment code of segment found is not larger than that required.		

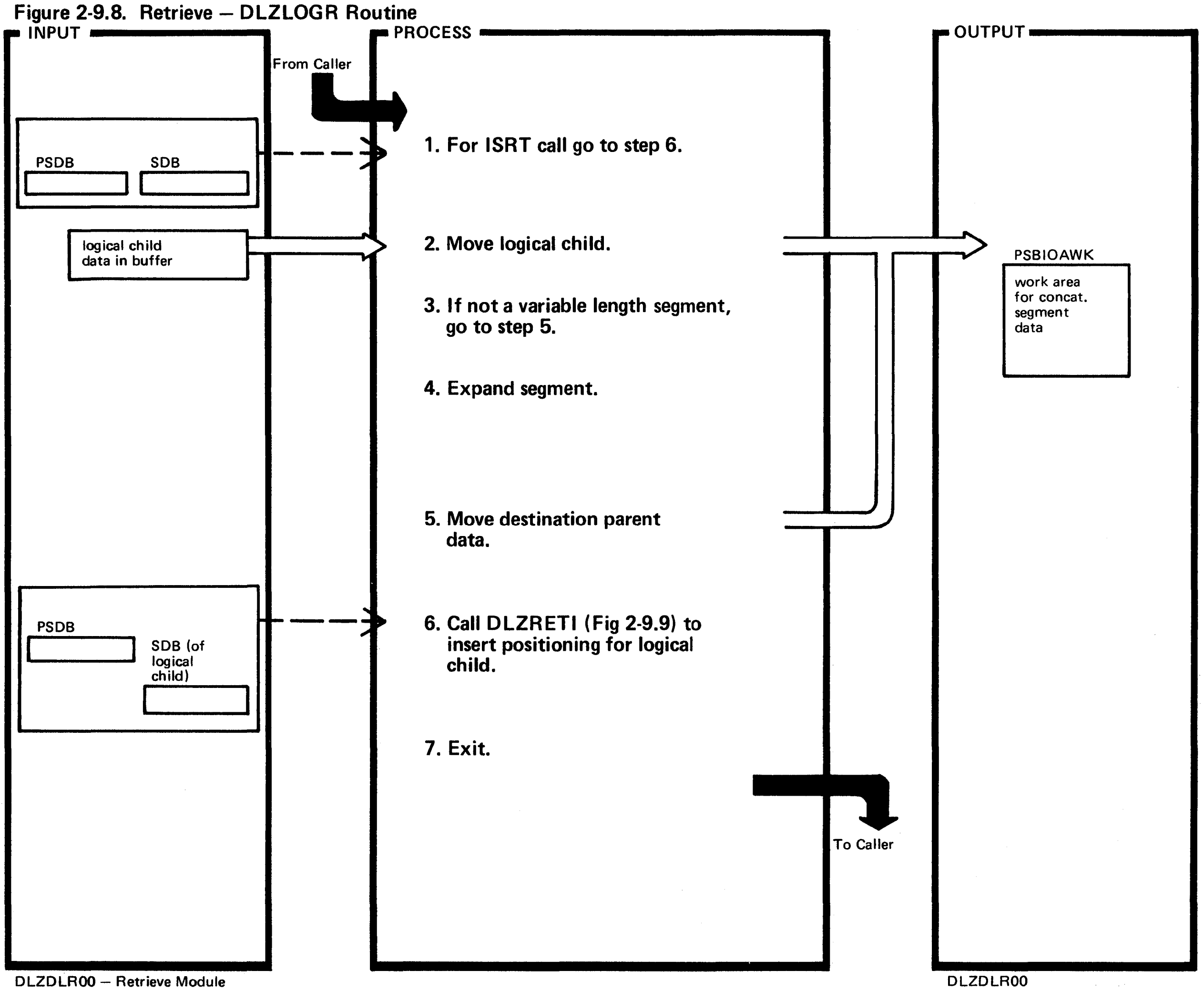
Figure 2-9.7. Retrieve – DLZGETS Routine



Extended Description	Routine	Label
6. If batch, only one task active, or no field level sensitivity specified, and if segment is fixed length and not involved in logical relationship, segment data is not moved (left in buffer pool). The same is true for Insert calls. For a path call (*D command), data has already been moved in DLZUPDT and is not moved here. Address of I/O area is PSBIOAWK.		
7. For Insert calls, return is to Load/Insert.		

Extended Description	Routine	Label

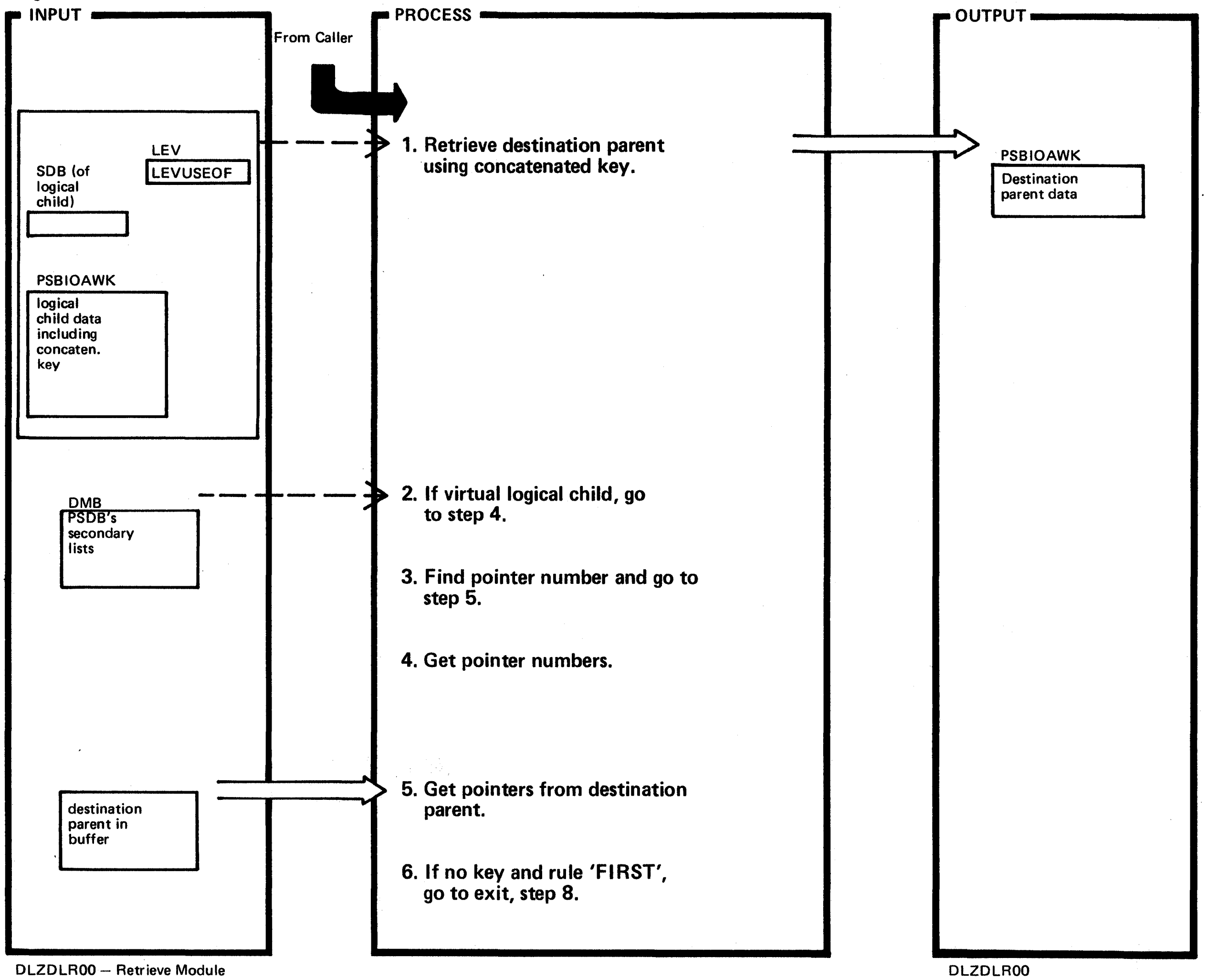
Figure 2-9.8. Retrieve — DLZLOGR Routine



Extended Description	Routine	Label
1. Destination parent concatenated key and logical child data.	DLZYENT	
6. Destination parent exists. Position segment on alternate twin chain.		

Extended Description	Routine	Label

Figure 2-9.9. Retrieve – DLZRETI Routine (Part 1 of 2)



Extended Description	Routine	Label
1. LEVUSEOF indicates offset of key for this level in concatenated key.		
Destination parent data is stored behind concatenated key and logical child.	DLZRETK	
2. For virtual logical child (insert through logical path), positioning on physical twin chain is required.		
3. Find logical twin pointer number. Find logical child first and last pointers in logical parent. Find FDB for key of logical child, if present.		RETISRTF
4. Find physical twin pointer number. Find physical child first and last pointers in parent. Find FDB for key of virtual logical child, if present.		RETISRTR
Logical twin key is moved to key feedback area.	DLZUPDL	

Extended Description	Routine	Label

Figure 2-9.9. Retrieve – DLZRETI Routine (Part 2 of 2)

INPUT

PROCESS

7. Follow alternate twin chain until key (if present) larger than key of inserted segment, or to end of chain. Go to exit, step 8.

   Special case: matching key found in chain:

   A. Key unique: If segment deleted logically, go to exit, step 8. Otherwise, set status code 'II', and go to exit, step 8.

   B. Key not unique: If rule is 'FIRST', go to exit, step 8. Otherwise, follow twin chain until larger key is found or to the end of chain.

8. Exit.

Return to Caller

OUTPUT

DBPCB

DBPCBSTC

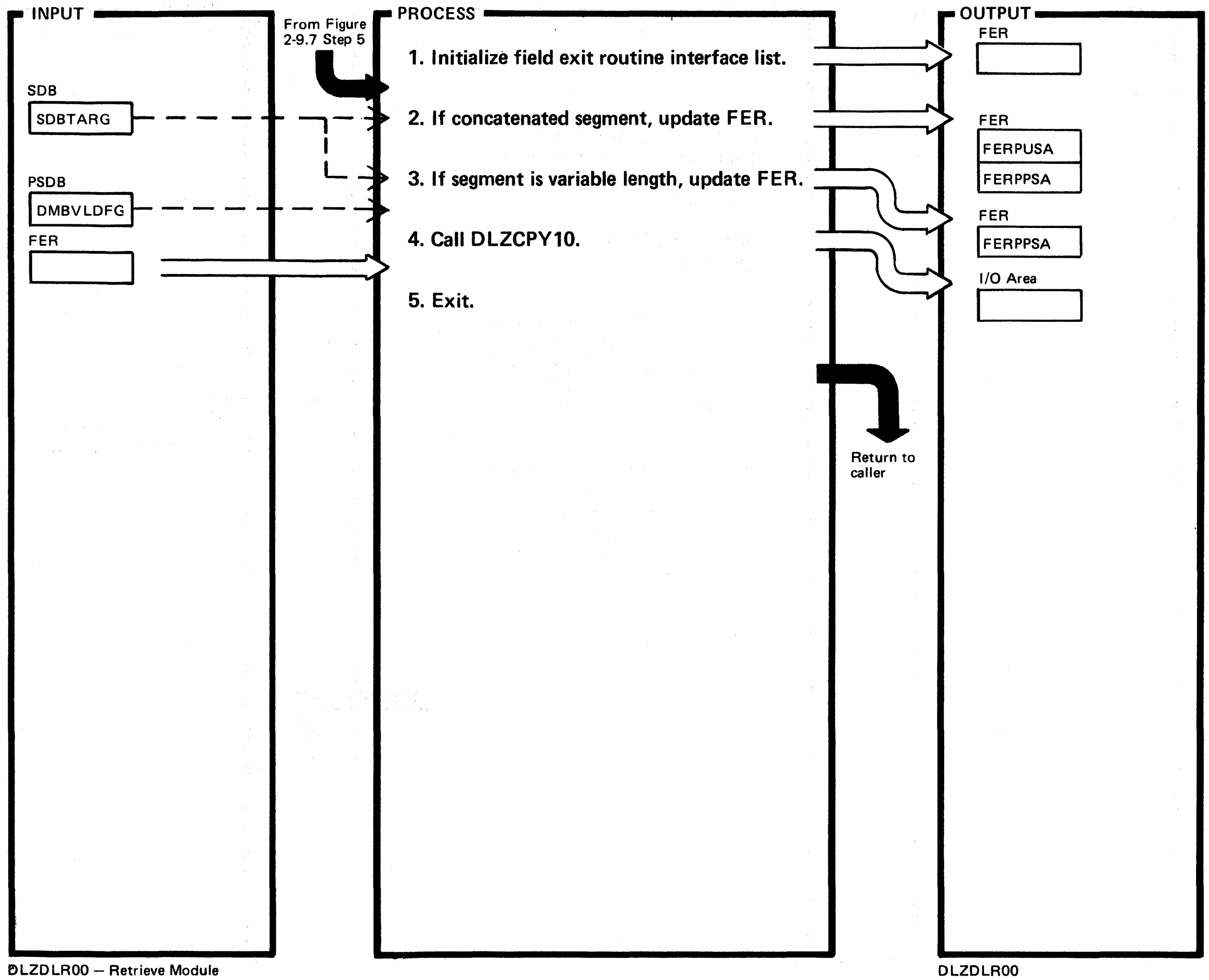
DLZDLR00 – Retrieve Module

DLZDLR00

Extended Description	Routine	Label
7. Alternate means: Logical twin chain if entering from physical path, physical if entering from logical path.		RETISRTL
If sequence field is in destination parent concatenated key (possible only for virtual logical child), the virtual area (physical parent, concatenated key, and logical child data) is built in PSBIOAWK, calling routines DLZYSTC and DLZMOVA. As an indication, the first byte of PSTWRKT5 is set to X'FF'		RETIVK
A. For logically deleted segment, turn on bit JCBDEFDL in field JCBCODE.		

Extended Description	Routine	Label

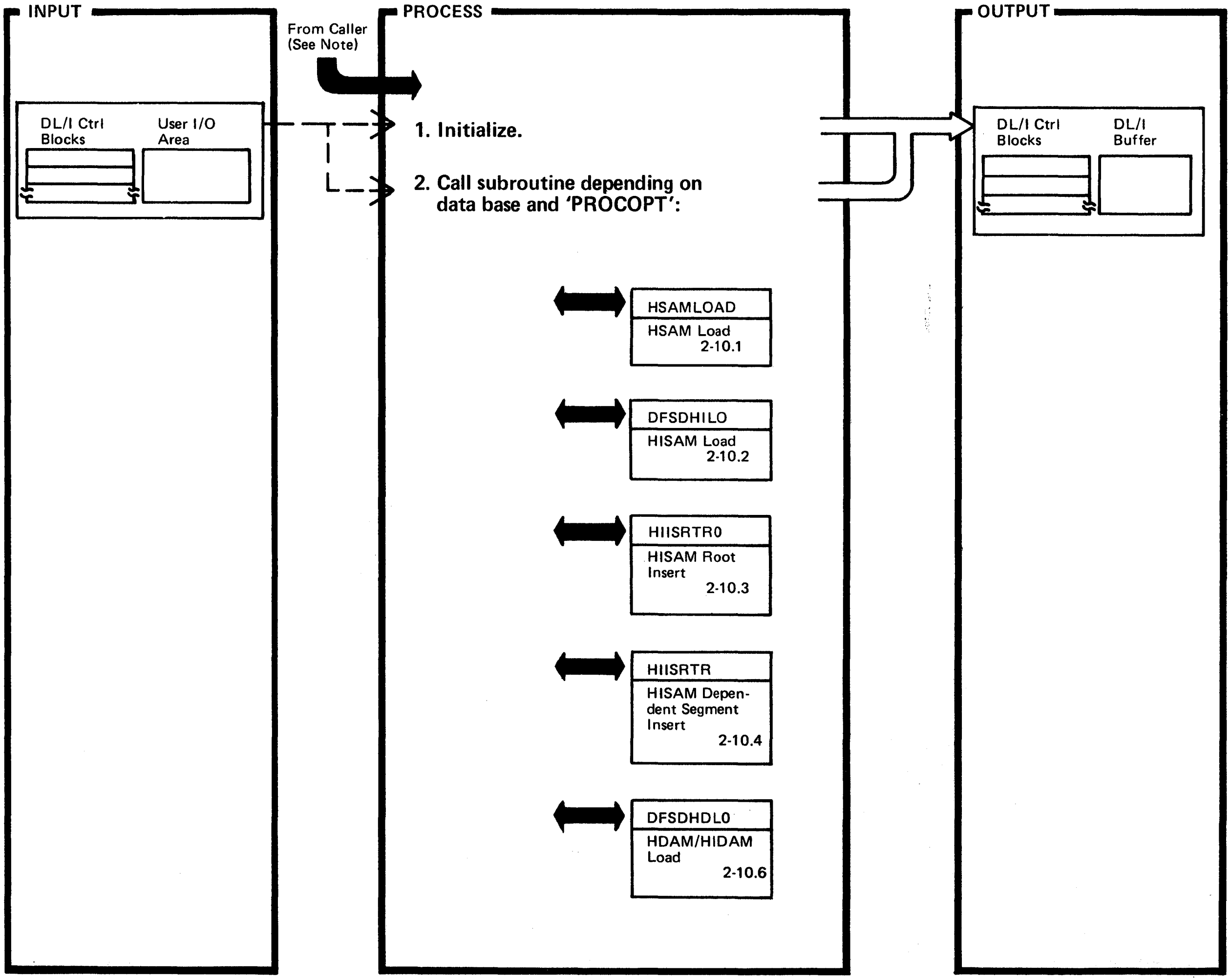
Figure 2-9.10. Retrieve — DLZFLD0 Subroutine



Extended Description	Routine	Label
1. FER is located at address in PSBNDXWK.		
2. The concatenated segment has been built in PSBIOAWK and the user's view must be constructed in another area (PSBXIOWK). For path calls (*D command), the user's view will be moved back to PSBIOAWK after conversion to the user's view.		FLDCSEG
3. Fields may be defined that are outside the physical segment, so they must be defaulted so conversion errors do not occur. If such fields do exist, the segment is moved to PSBXIOWK and the defaults provided.		FLDVAR
4. If a conversion error is detected, an immediate exit to the Call Analyzer is taken.		FLDERR

Extended Description	Routine	Label

Figure 2-10. Load/Insert (Part 1 of 2)



DLZDDLE0 — Load Insert Module

DLZDDLE0

Extended Description	Routine	Label
Note: DLZDDLE0 is called from DLZDLA00 (Call Analyzer) or from DLZDLR00 (Retrieve Module).		

Extended Description	Routine	Label

Figure 2-10. Load/Insert (part 2 of 2)

INPUT

PROCESS

2. (con't)

HDAM/HIDAM
Not Load 2-10.7

3.

DFSDNXTO
Not Load Ending Routine 2-10.8

HIISNXLV
Load Ending Routine 2-10.9

4. Return

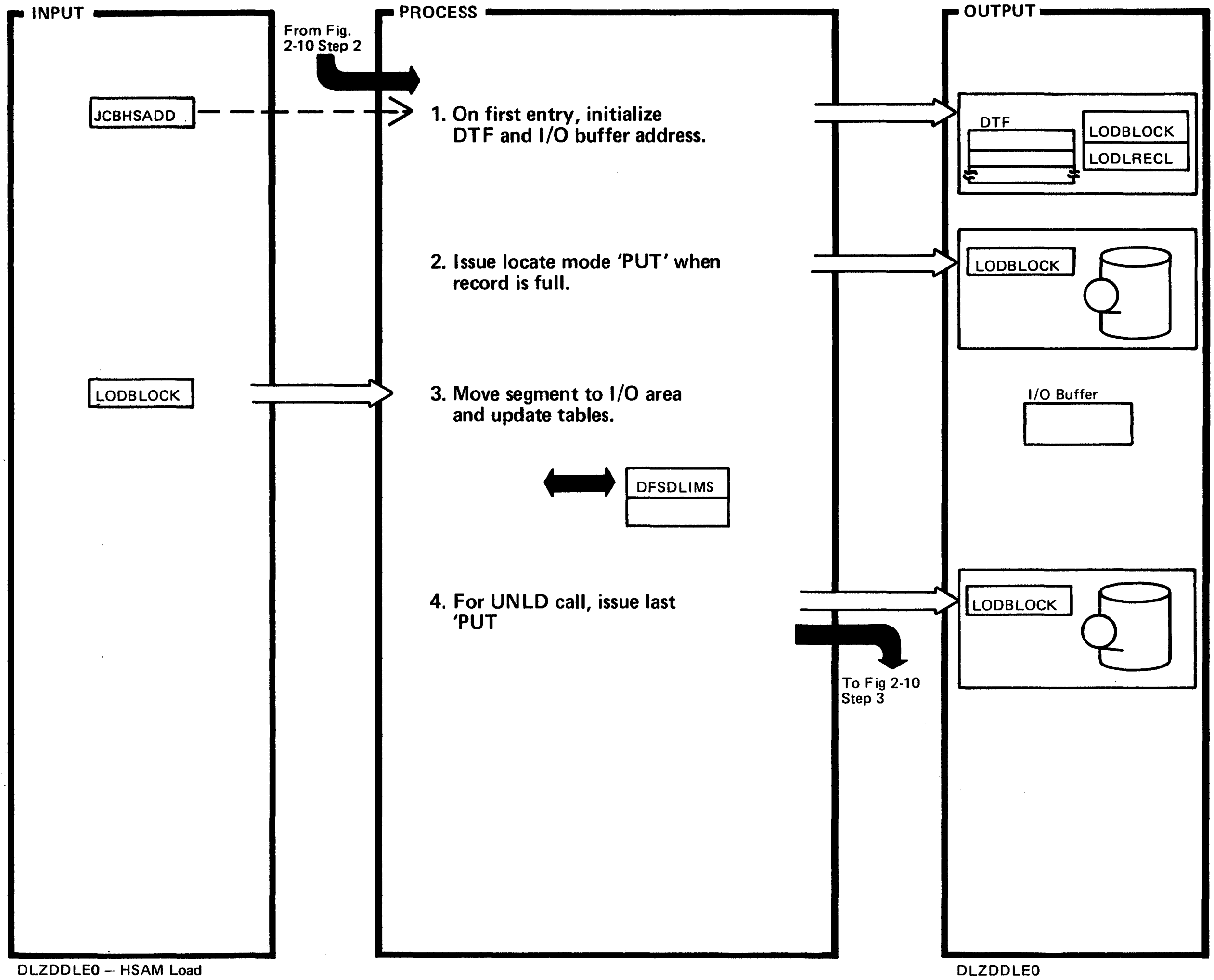
Return to Caller (See Note)

OUTPUT

Extended Description	Routine	Label
Note: DLZDDLE0 is called from DLZDLA00 (Call Analyzer) or from DLZDLR00 (Retrieve Module).		

Extended Description	Routine	Label

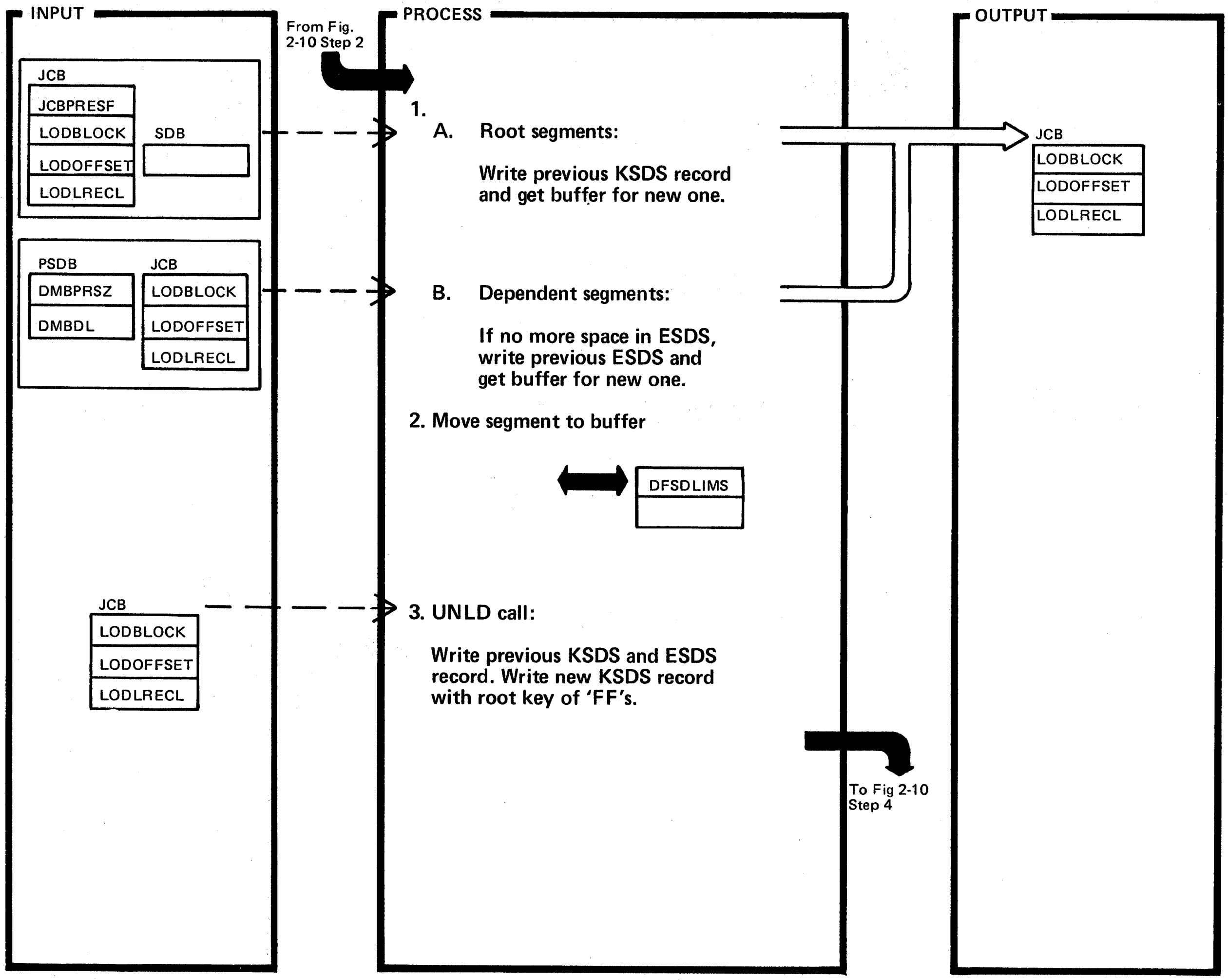
Figure 2-10.1. HSAM Load



Extended Description	Routine	Label
1. DLZDLOC0 stores the I/O area address in the JCB. It is updated with every 'PUT' The record size is taken from the DTF and the error exit address in the DTF is updated.		HSAMFRST

Extended Description	Routine	Label

Figure 2-10.2. HISAM Load



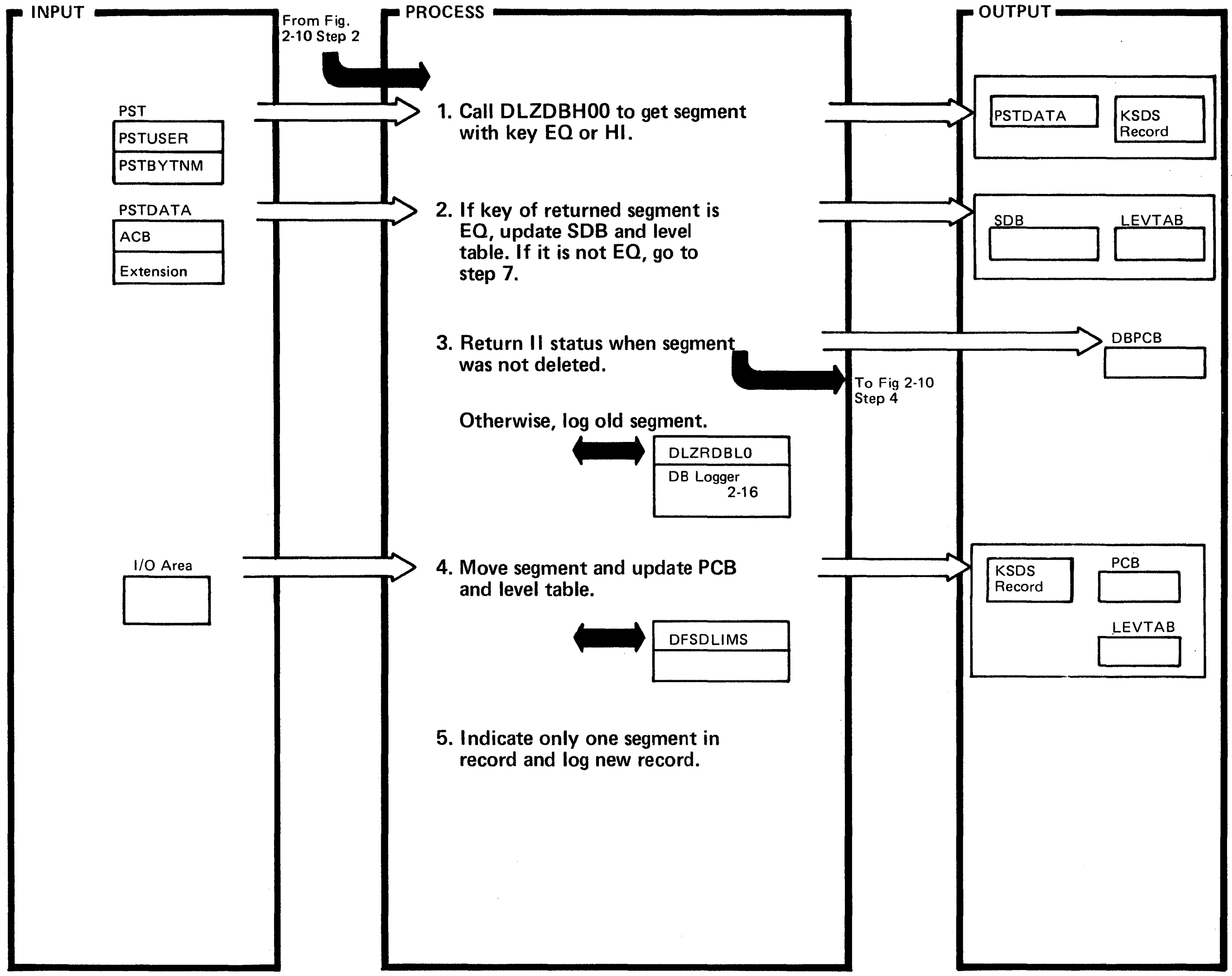
DLZDDLE0 – HISAM Load

DLZDDLE0

Extended Description	Routine	Label
1. A. Record length, buffer address, and offset into buffer is stored in the JCB and passed from call to call.	WRITEOLD	HISIMPLS
When a call for a new root segment is made, the buffer handler (DLZDBH00) is called to write the previous KSDS record and to get buffer space for the new one.		
B. If there is space left in the ESDS records, continue with step 2. Otherwise, the RBA of the next ESDS record is calculated, the pointer of the current ESDS record updated, and the buffer handler called to write the ESDS. Another call to DLZDBH00 is made to get buffer space for a new ESDS record.	NEWRBA	NEEDOSAM
ABEND 855 is given if VSAM returns an RBA different from the calculated one.		CATERROR

Extended Description	Routine	Label
2. The segment is moved, the PCB key fed back, and the level table updated.		

Figure 2-10.3. HISAM Root Insert (Part 1 of 2)



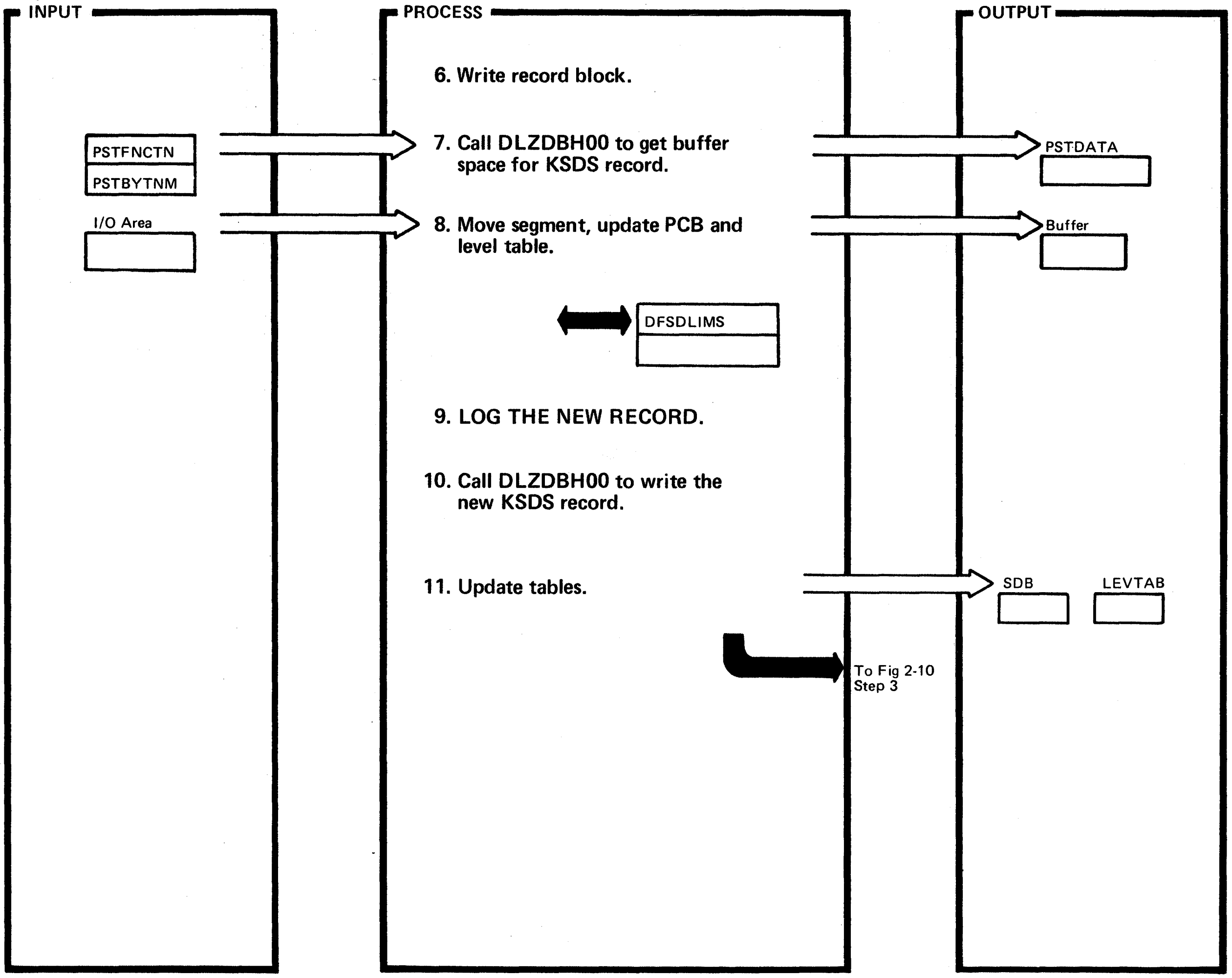
DLZDDLEO – HISAM Root Insert

DLZDDLEO

Extended Description	Routine	Label
1. The buffer handler is called with 'PSTSTLEQ' to get a segment with key equal or higher than the one to be inserted.	GOTOFUNC	HIISRTRO
2. If the key returned is higher, processing continues with step 7.		ISIS015
3. When the delete flag is not on in the segment returned, status code 'II' is returned to the caller.		
The data base log module is called to log the old KSDS record.		ISSDELET
4. The new root segment is moved to the KSDS record. The PCB key feedback area and level table are updated.		
5. The pointer to the ESDS record is cleared and '00' moved to the KSDS record behind the root segment. The data base log module is called to log the new KSDS record.		

Extended Description	Routine	Label

Figure 2-10.3. HISAM Root Insert (Part 2 of 2)



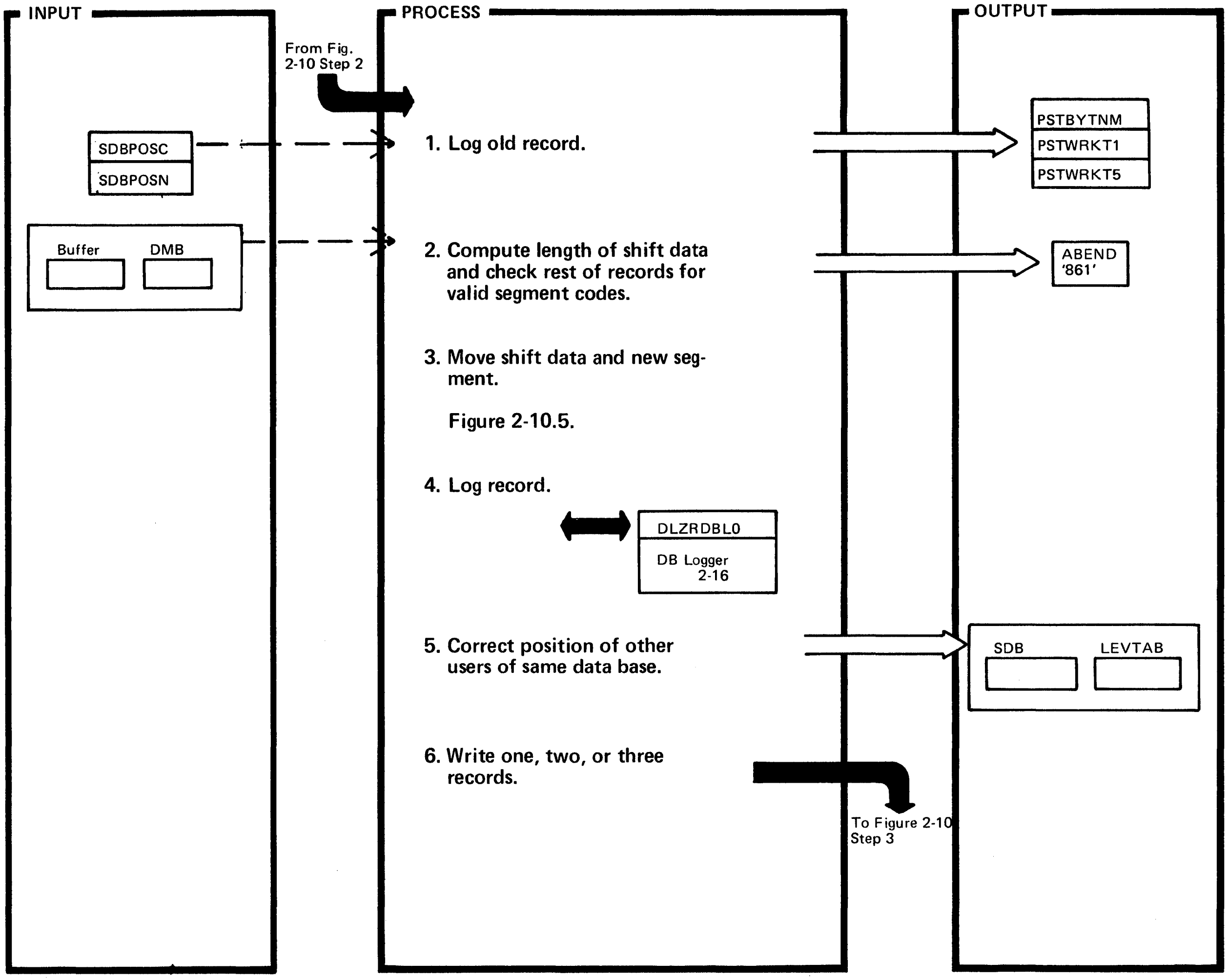
DLZDDLE0 – HISAM Root Insert

DLZDDLE0

Extended Description	Routine	Label
6. The buffer handler is called to write the KSDS record back (PSTBFALT).	GOTOFUNC	
7. The buffer handler is called (PSTGBSPC) to get buffer space for one KSDS record.	GOTOFUNC	ISNOTEQ
10. PSTPUTKY is used to write the new KSDS record.	GOTOFUNC	ISSIMPL1

Extended Description	Routine	Label

Figure 2-10.4. HISAM Dependent Segment Insert



DLZDDLEO — HISAM Dependent Segment Insert

DLZDDLEO

Extended Description	Routine	Label
1. DLZDLR00 has located within a KSDS or ESDS record, where the new segment has to be inserted. The old record is logged from insert point on to the right.		HIISRTR
2. The record is inspected from the insert point to the right. The segment code is checked and the length of the remaining segments is added to give the 'shift data'.		HAVELREC COMPSHFT ABEND861
4. Log the old record from insert point to the right.	DLZDLBLO	LOGLEVCO
5. SDBs and level tables of other PCBs that are positioned in the same record are updated to show the shifted position of the segments.		INSADJUS
6. DLZDBH00 is called to write back the old record and to write one or two new ESDS records.	GOTOFUNC	KNNDONEX

Extended Description	Routine	Label

Figure 2-10.5. NOTSC Routine (Part 1 of 2)

INPUT

DMB

PSTUSER

I/O Area

From Fig. 2-10.4 Step 3

PROCESS

1. Segment and shift data fit in old record:

   A. Move 'shift data' right.

   B. Move segment to buffer and update tables.

2. Segment fits in old record but not shift data

   A. Calculate RBA of new ESDS record.

   B. Get buffer space for one ESDS record.

   C. Chain old and new record and log chain.

   D. Move 'shift data' to new ESDS record.

   E. Log new record.

   F. Move segment to old record.

OUTPUT

old record

old record

new record

DLZDDLE0 – HISAM Dependent Segment Insert

DLZDDLE0

Extended Description	Routine	Label
1. When both the new segment and the shift data fit in the old record, the shift data is moved right by segment length. The segment is moved to the record and the PCB and level table are updated.	DFSDLIMS	OVERLAPL
2. A new ESDS record has to be built.	GETNESDS LOGCHAIN COMMOVE LOGNEWOS DFSDLIMS	SEGTOOLD

Extended Description	Routine	Label

Figure 2-10.5. NOTSC Routine (Part 2 of 2)

INPUT

I/O Area

DMB

PROCESS

3. Move segment and 'shift data' to new ESDS.

   A. Calculate RBA of new ESDS record.

   B. Get buffer space for new ESDS record.

   C. Chain the two records and log the chain.

   D. Move segment to new record and update records.

   E. Move 'shift data' to new record. If it does not fit, repeat step 3 A, B, and C.

   F. Log one or two new ESDS records.

DLZRDBLO

DB Logger

2-16

To Fig 2-10.4 Step 4

OUTPUT

old record

new record(s)

DLZDDLEO — HISAM Dependent Segment Insert

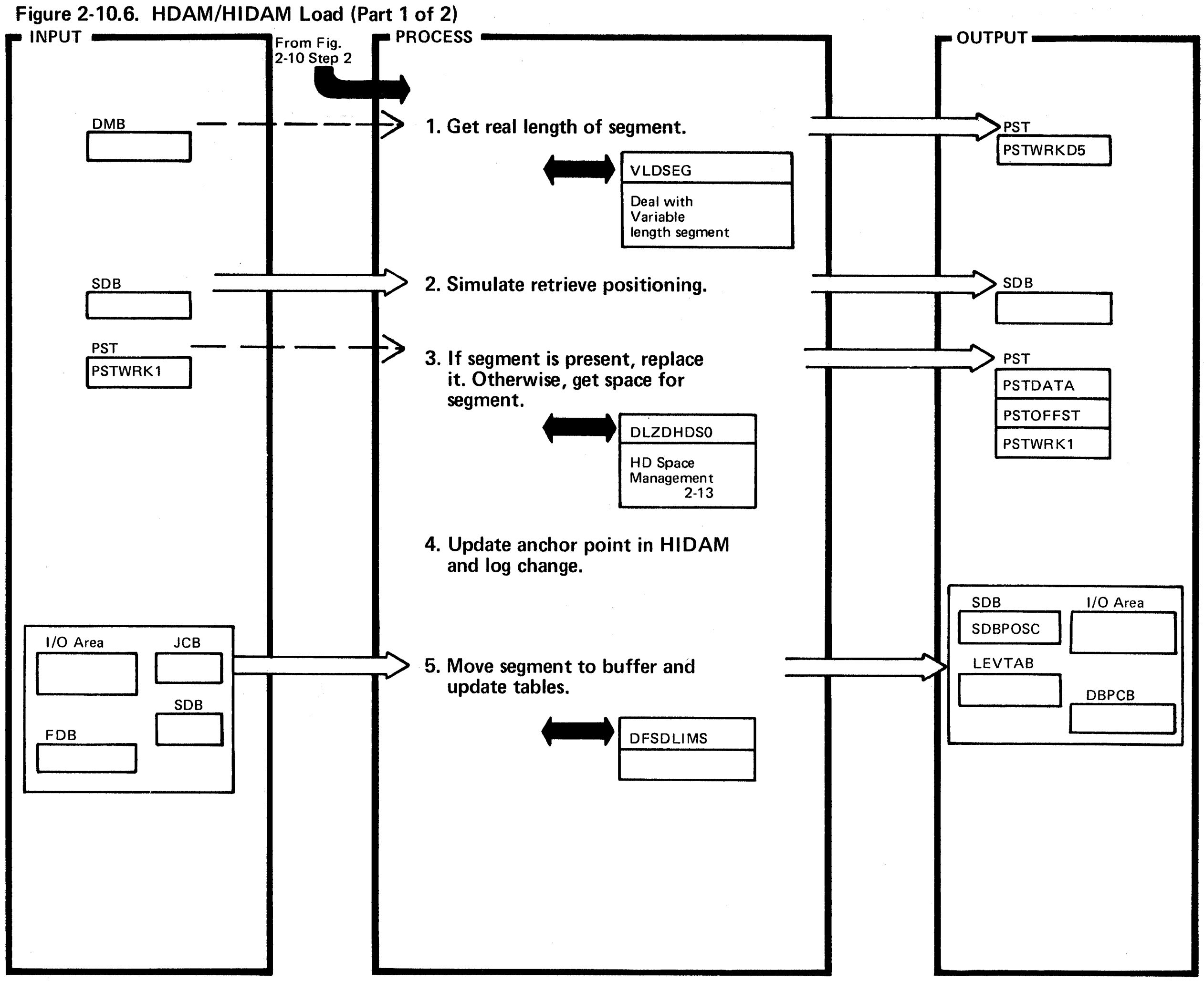
DLZDDLEO

Extended Description

Extended Description	Routine	Label
3. Neither segment or 'shift data' fit in the old record. A new record has to be built. If it does not have room for the segment and 'shift data', another new ESDS record has to be built. The records are chained and logged.	GETNESDS LOGCHAIN DFSDLIMS COMMOVE LOGNEWOS NEWRBA GOTOFUNC	SEGTONEW SHIFT00 SHIFTOS2 LOGLEVCO

Extended Description	Routine	Label

Figure 2-10.6. HDAM/HIDAM Load (Part 1 of 2)



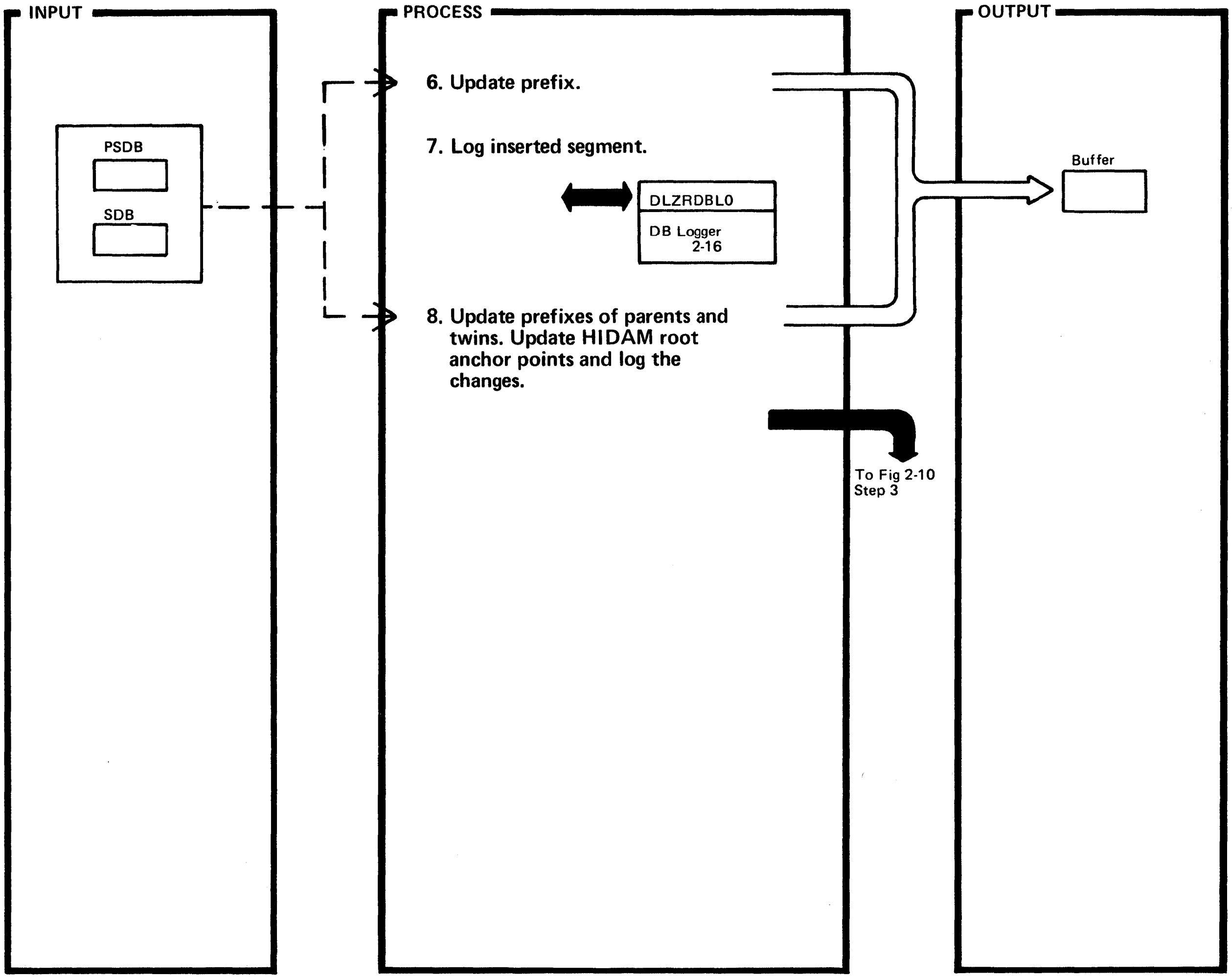
DLZDDLE0 – HDAM/HIDAM Load

DLZDDLE0

Extended Description	Routine	Label
1. The subroutine VLDSEG takes the length from the PSDB for fixed length segments and from the user's I/O area for variable length segments. The compaction exit routine is called, if it exists.	VLDSEG	DFSDHDLO
ABEND '863' is given when the compaction routine changes the sequence field.		ABEND863
2. For HIDAM root segments, DLZDLR00 did the positioning. For other segments, it is done here.		
3. Space management is called to get space for the segment. If the segment was deleted in one path only, i.e. it was not removed by DLZDLD00, the segment is replaced with the new data.	TOSPACE	GETSPACE SPACEOUT
4. HIDAM root segments without PTB pointers are chained off the anchor point in chronological sequence.		SPACEOK
5. Move segment to buffer, update PCB key feedback, and update level table.	DFSDLIMS	ANCHOROK

Extended Description	Routine	Label

Figure 2-10.6. HDAM/HIDAM Load (Part 2 of 2)



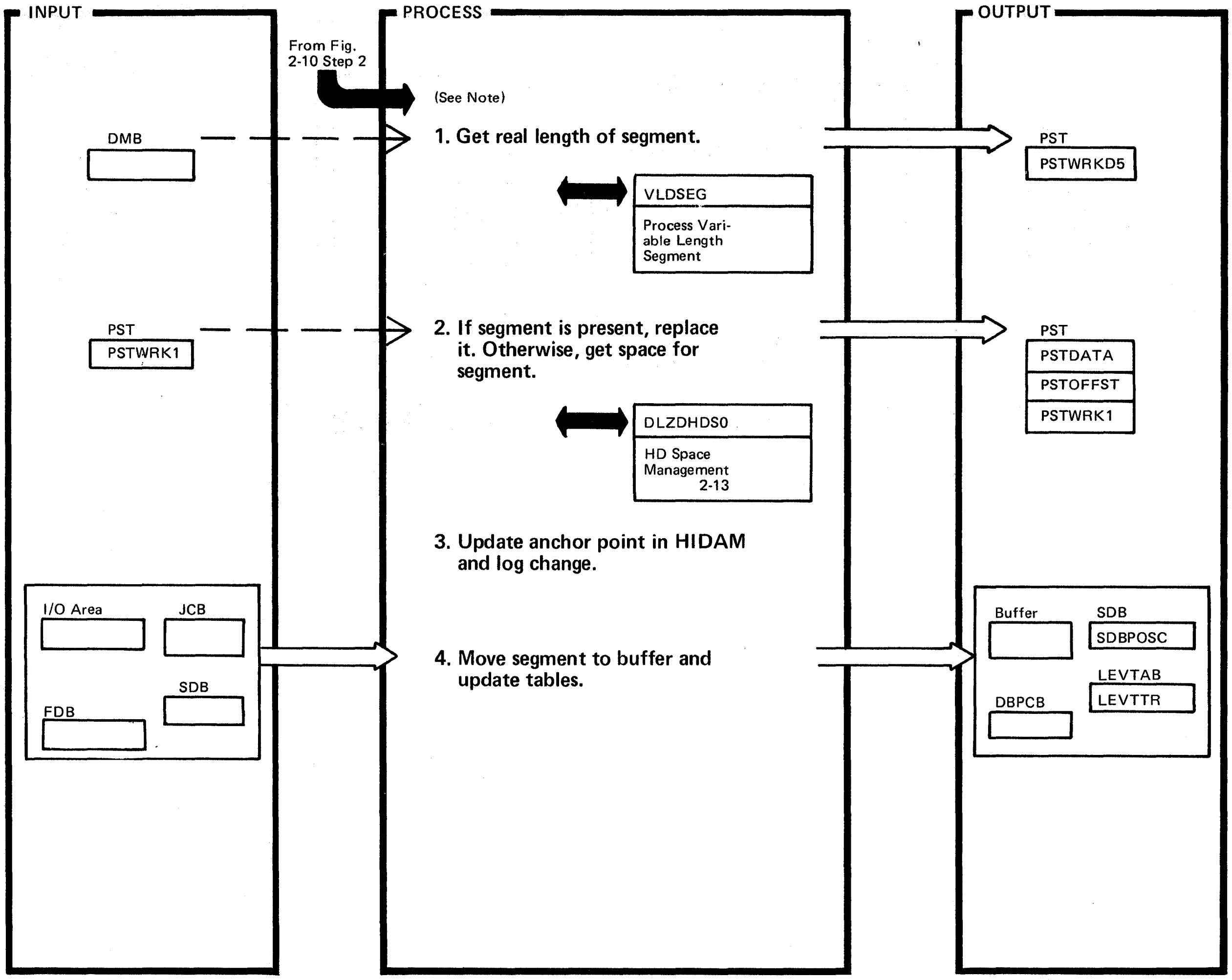
DLZDDLEO – HDAM/HIDAM Load

DLZDDLEO

Extended Description	Routine	Label
6. The prefix of the segment is updated: physical twin pointers, physical parent pointer, logical parent pointer, and logical twin pointers.		
7. The data base log module is called to log the inserted segment.		MYPREOK
8. Call space management (DLZDHDS0) to update the prefix of physical twins, logical twins, physical parents, and logical parents. Update anchor point for HDAM root segments and call the data base log module to log all changes.	TOSPACE UPPARENT UPPREFIX	UPBITMAP BITMAPOK HDDANCOR

Extended Description	Routine	Label

Figure 2-10.7. HDAM/HIDAM Not Load (Part 1 of 2)



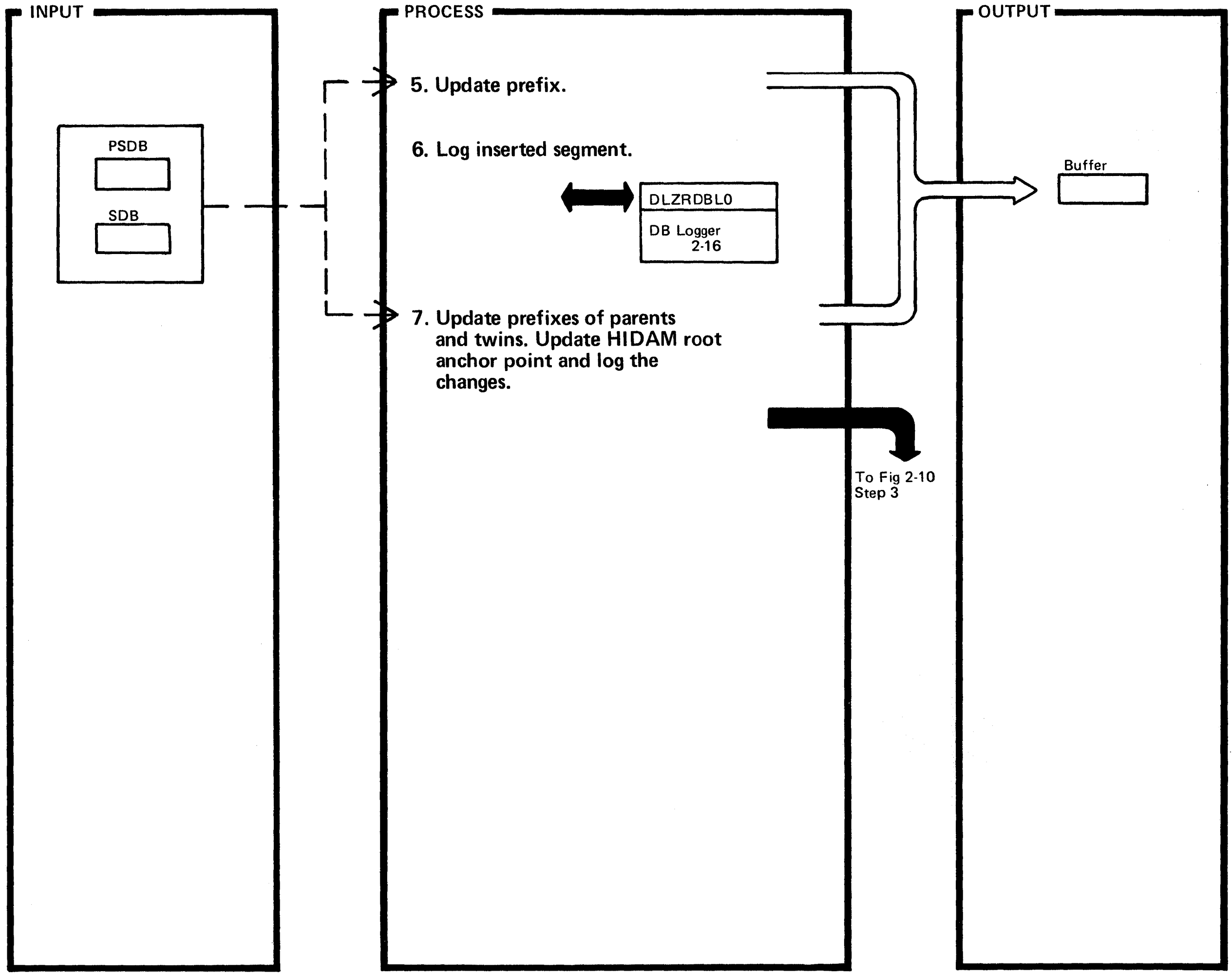
DLZDDLEO – HDAM/HIDAM Not Load

DLZDDLEO

Extended Description	Routine	Label
Note: When this entry is used, DLZDLR00 had done the positioning.		DFSDHDIO
2. Space management (DLZDHDS0) is called to get space for the segment. If the segment was deleted in one path only, i.e. it was not removed by DLZDLD00, the segment is replaced with the new data.	TBSPACE	GETSPACE SPACEOUT POSTPST SPACEOK
3. HIDAM root segments without PTB pointers are chained off the anchor point in chronological sequence.		SPACEOK
4. Move segment to buffer, update PCB feedback and the level table.	DFSDLIMS	ANCHOROK

Extended Description	Routine	Label

Figure 2-10.7. HDAM/HIDAM Not Load (Part 2 of 2)



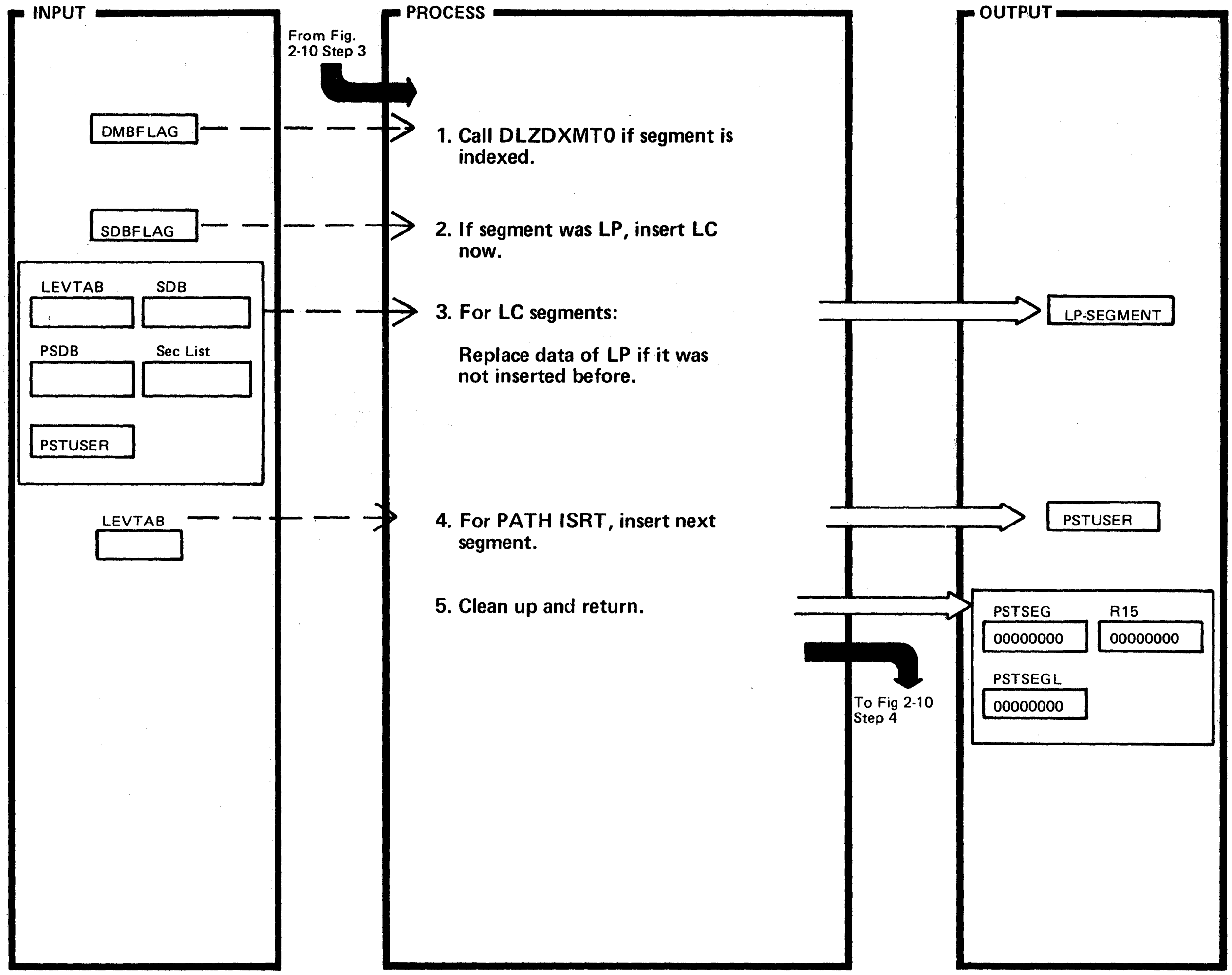
DLZDDLE0 – HDAM/HIDAM Not Load

DLZDDLE0

Extended Description	Routine	Label
5. The prefix of the segment is updated: physical twin pointers, physical parent pointer, logical parent pointer, and logical twin pointers.		
6. The data base log module is called to log the inserted segment.		MYPREOK
7. Call space management (DLZDHDS0) to update the bitmap if required: update prefix of physical twins, logical twins, physical parents, and logical parents. Update anchor point for HDAM root segments, call the data base log module to log all changes.	TOSPACE UPPARENT UPPREFIX	UPBITMAP BITMAPOK HDDANCOR

Extended Description	Routine	Label

Figure 2-10.8. Not Load Ending Routine



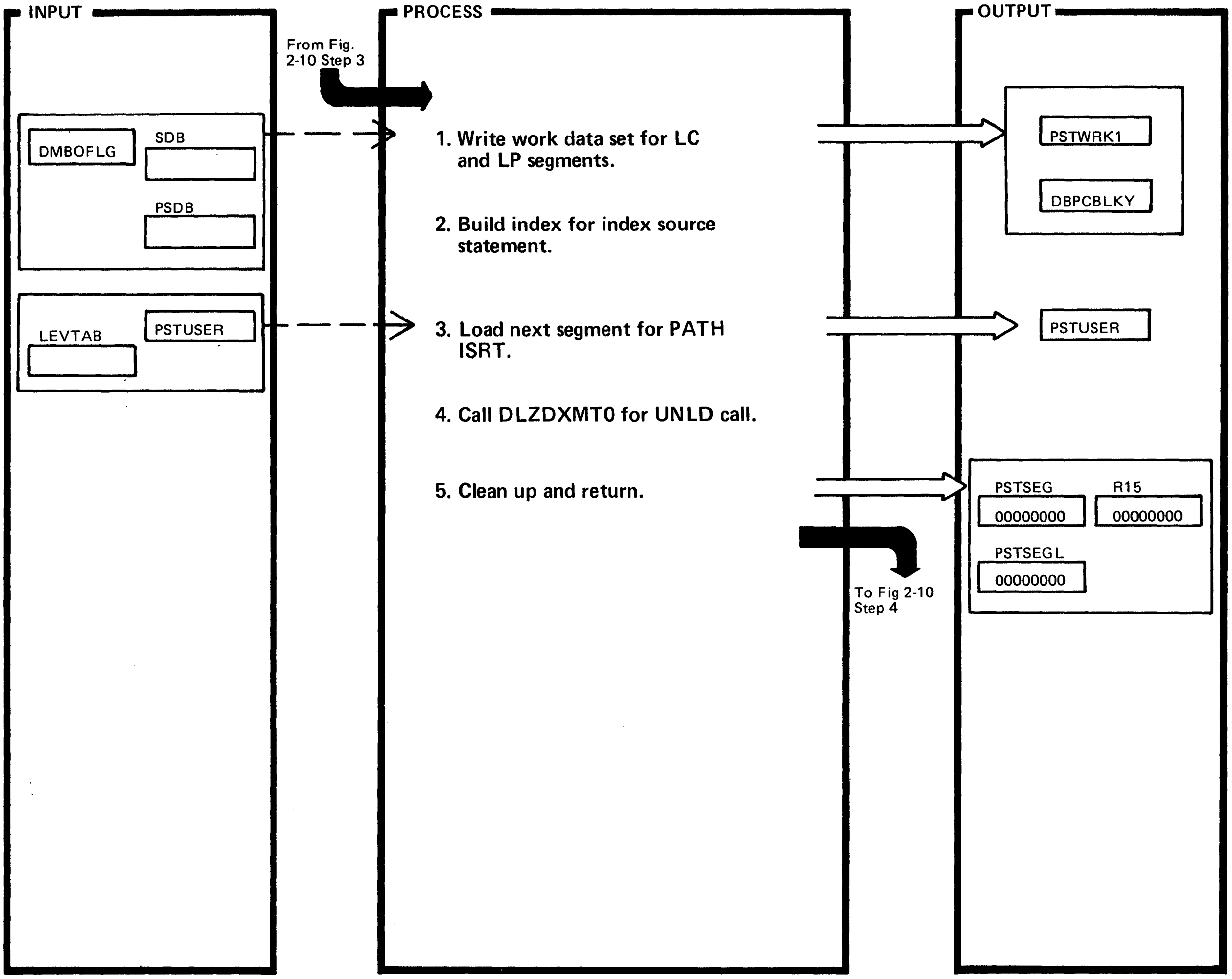
DLZDDLE0 – DFSDXNT0 Ending Routine for Not Load

DLZDDLE0

Extended Description	Routine	Label
1. Index Maintenance is called to build the primary or secondary index for an index source segment.		
2. If the ISRT call was for a concatenated segment, the destination parent was inserted first (if it did not exist before the ISRT call). The next step is to insert the logical child segment. The insert process is repeated from Figure 2-10 step 2.		NXTLEVIS
3. If the ISRT rule of the destination parent is virtual and this segment existed already, then the data of the destination parent is replaced. DLZDXMT0 is called to replace the index if the destination parent is an index source segment.		
4. If there are more segments to be inserted in a PATH, then point to the next segment in the I/O area and continue with Figure 2-10 step 2.		NOLPAREN

Extended Description	Routine	Label

Figure 2-10.9. Load Ending Routine



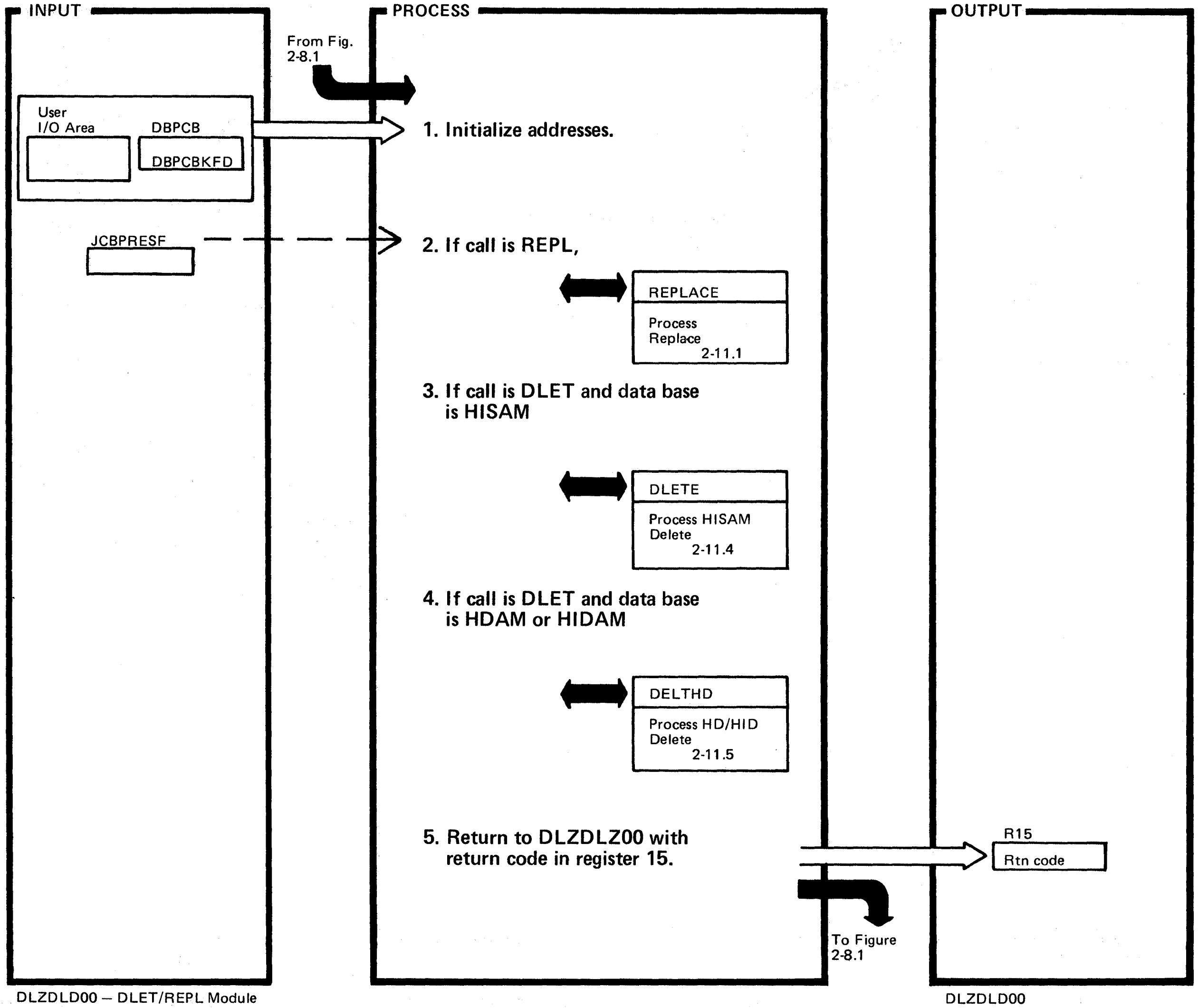
DLZDDLE0 – HIISNXLV Ending Routine Load Mode

DLZDDLE0

Extended Description	Routine	Label
1. If the segment just loaded was a logical child or a logical parent segment, DLZDSEH0 is called to write the work data set. If opening of the work data set fails due to 'ASSGN SYS013,IGN' and the segment was an LP, processing continues. On any other open failure, 'ABEND 864 is given.		CALLERN CALLWORK
2. If the segment is an index source segment, DLZDXMT0 is called. It writes the work data set or writes the index pointer segment directly.		NOLOAD NCALLNDX
3. For PATH ISRT, the pointer to the I/O area is updated and processing continues with Figure 2-10 step 2.		NOINDEX2
4. DLZDXMT0 is called to inspect all PSDBs of the DMB for index source segments and builds an FF key index pointer record for it.		

Extended Description	Routine	Label

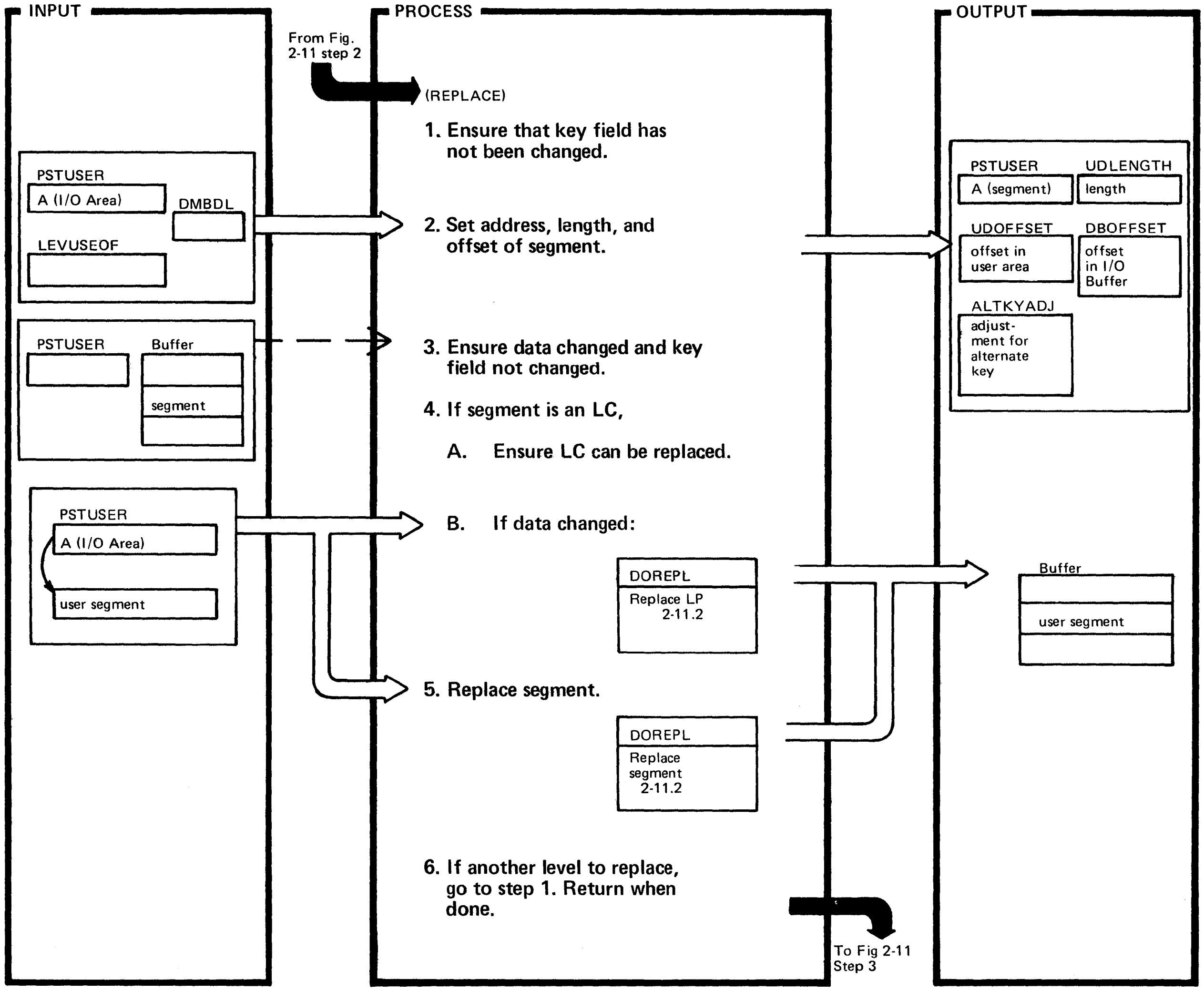
Figure 2-11. Delete/Replace



Extended Description	Routine	Label
1. The segment to be deleted or replaced is identified by the contents of JCBLEVIC. Position is established by DLZDLR00 in the previous call.	DLZDLD00	DELREPEP
2.		REPLACE
3.		DELETE
5. If a user error occurred, DBPCBSTC has return code. If abend, PSTERCD1 has abend code and registers are saved at SCDABSAV + 8.		RETURN

Extended Description	Routine	Label

Figure 2-11.1. Replace



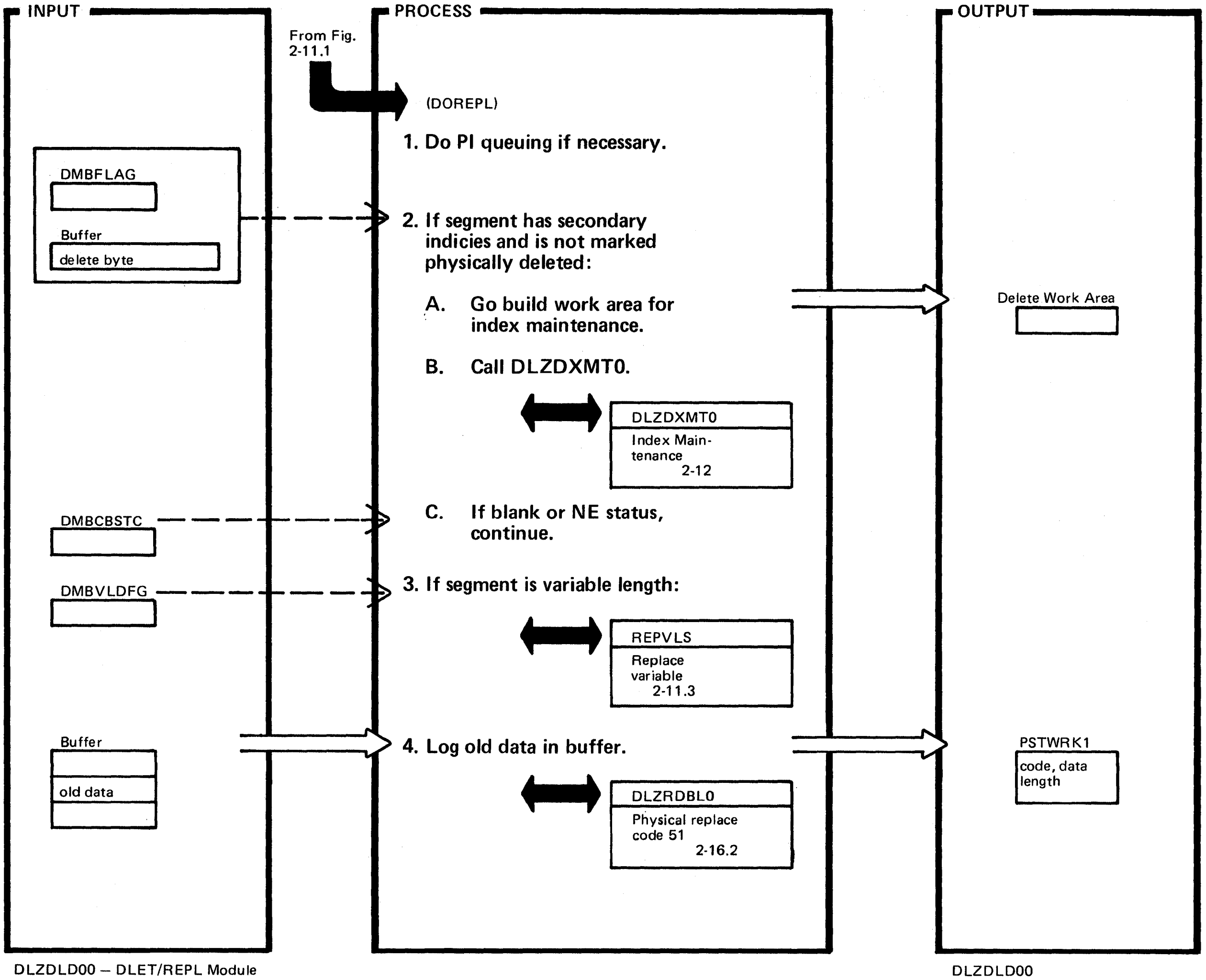
DLZDLD00 – DLET/REPL Module

DLZDLD00

Extended Description	Routine	Label
1.		REPCK01
2. PSTUSER will have new value if path call had been made. The length is taken from the first two bytes of the I/O area if segment is variable length.		
3. Additional logic is needed if segment is variable length or if PROCSEQ is specified.		CHKRLP CHKREPL1
4. A. The following check is made for the LC: Neither the physical nor logical key fields can be changed (DA status).		CHKREPFF
The following checks are made for the destination parent:		CHKRLP01
a) If data didn't change, no replace.		

Extended Description	Routine	Label
4. (con't)		
b) If replace rule is physical, RX status. If logical, no change and blank status. If virtual, the key of the LP cannot be changed (DA status). The segment can be replaced.		
B.		REPPAR01
5. This replaces normal segment or LC.		REPFINAL
6. If path call, see if another segment in hierarchy can be replaced.		LEVDONE

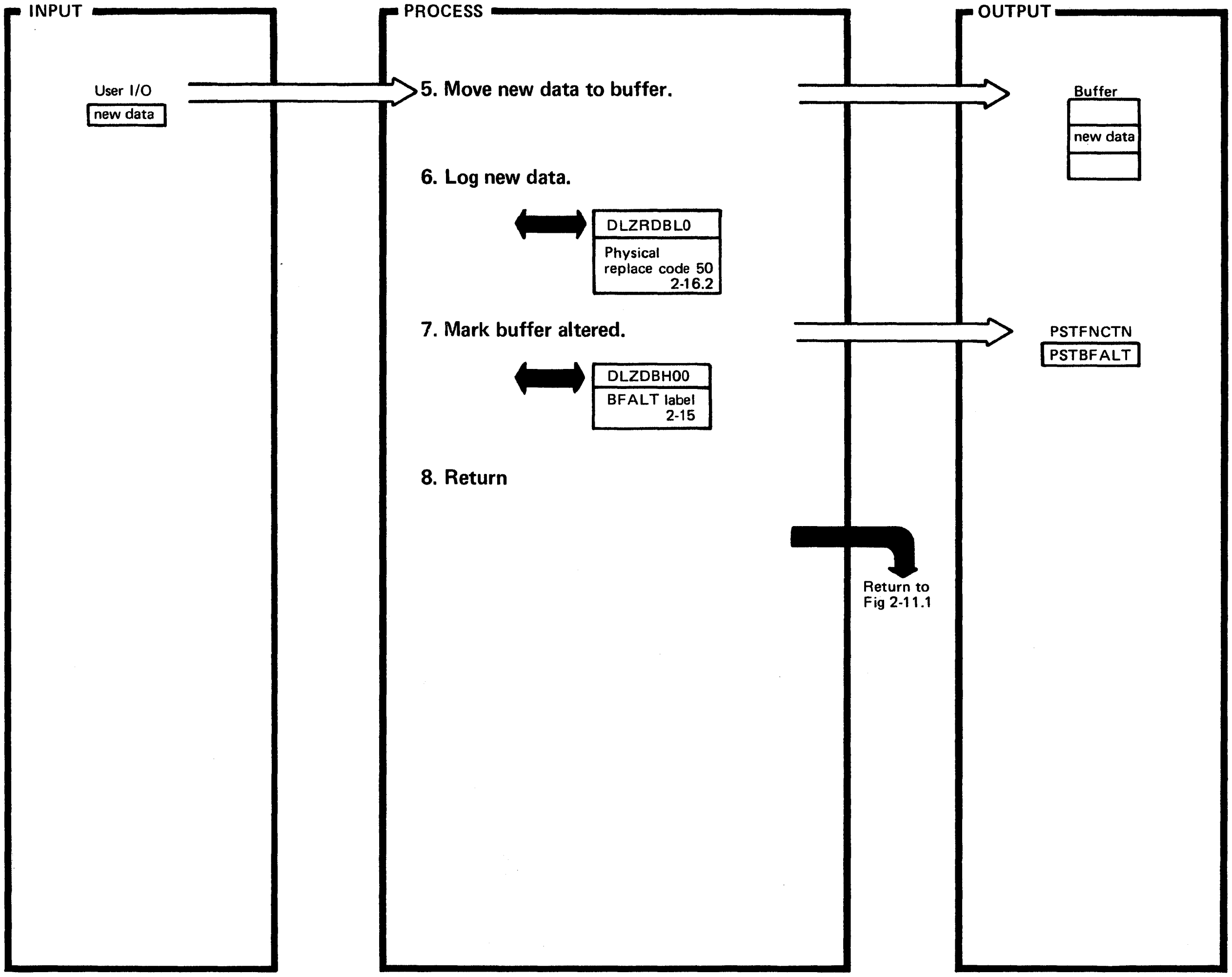
Figure 2-11.2. Replace Data (Part 1 of 2)



Extended Description	Routine	Label
2. Index Maintenance needs the actual concatenated key of this segment. If return code is NE, we still continue processing because index is now set as per new data. Work area is freed.		DOREPL09
3.		DOREPL10
4. DBLPHYR+DBLPHYR0 is set in first byte of PSTWRK1.		

Extended Description	Routine	Label

Figure 2-11.2. Replace Data (part 2 of 2)



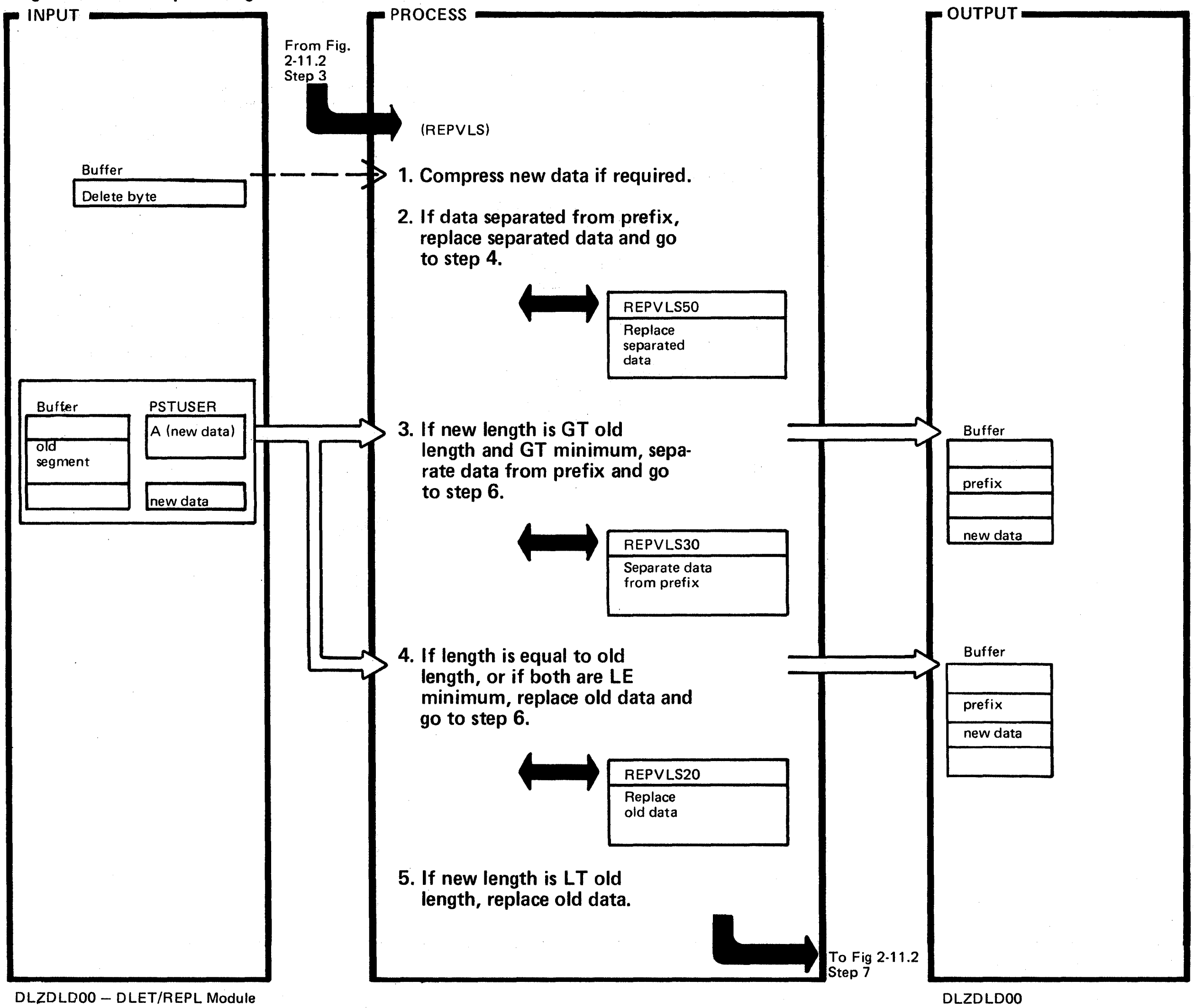
DLZDLD00 – DLET/REPL Module

DLZDLD00

Extended Description	Routine	Label
5. The address of the user's I/O area is in PSTUSER.		
6. DBLPHYR is set in PSTWRK1 with the length of the segment.		DOREPL92 REPL18

Extended Description	Routine	Label

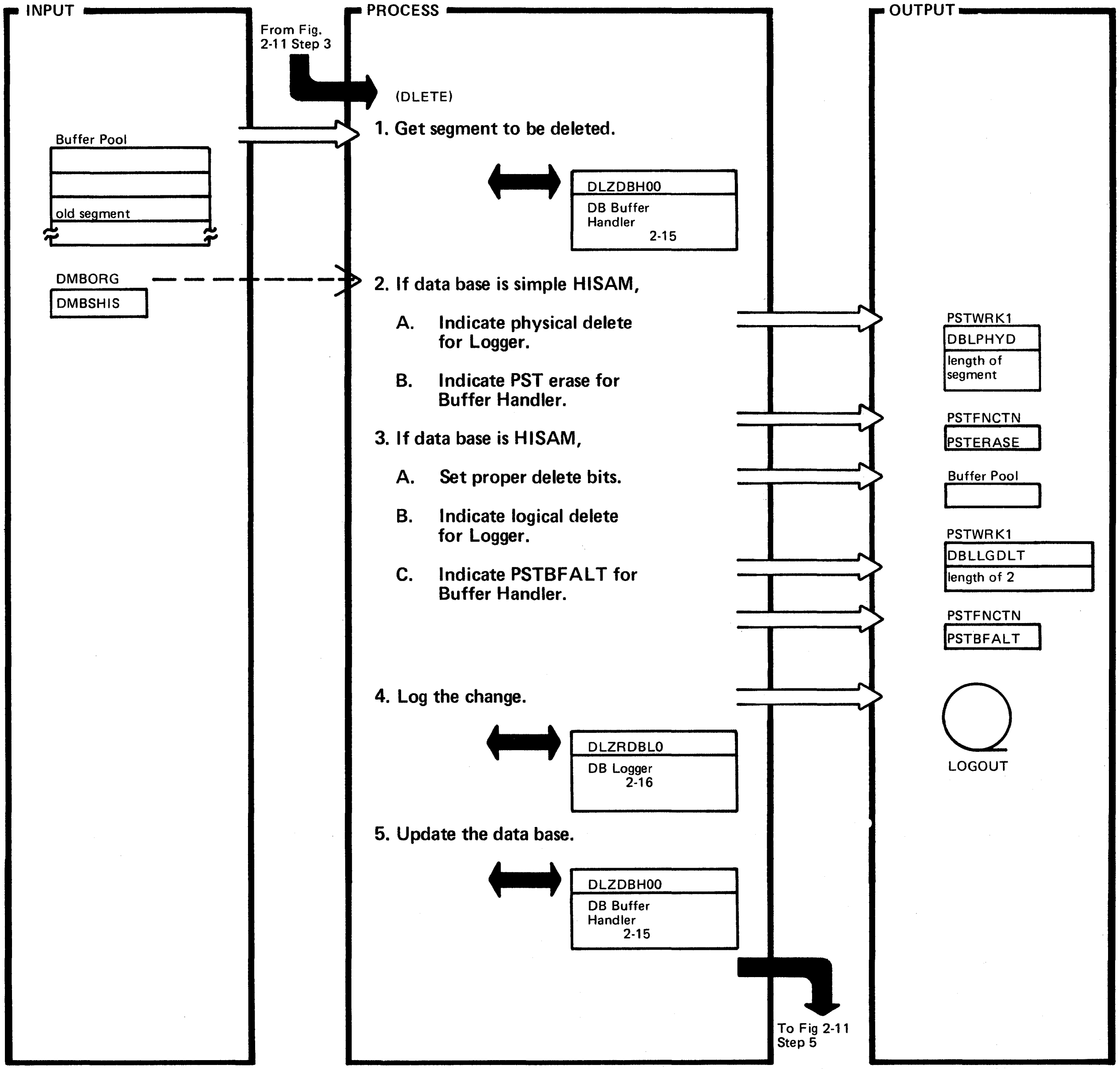
Figure 2-11.3. Replace Segment



Extended Description	Routine	Label
2. When the data is previously separated and the new data length is less than the old length, an attempt is made to relocate the new data adjacent to the prefix.	DLZDLDR0	REPVLS01
3 When the old segment size is not large enough for the new segment, the data is separated from the prefix. A pointer overlays the first four bytes of the old data and will be used to find the new.		REPVLS03
5 When the new data will fit in the old location, it is moved over the old data with any excess bytes being freed.		REPVLS10
All changes to the data base have been logged.		REPVLS38

Extended Description	Routine	Label

Figure 2-11.4. HISAM Delete



DLZDLD00 — DLET/REPL Module

DLZDLD00

Extended Description	Routine	Label
1.		DELT01
2. The entire segment to be erased is logged.		SHISAM
3. Only the segment code and delete byte are logged.		DELT41 LOGDLT

Extended Description	Routine	Label

Figure 2-11.5. HDAM/HIDAM Delete

INPUT

PSDBs

Buffer

segment

DLZDLD00 – DLET/REPL Module

From Fig. 2-11 Step 4

PROCESS

(DELTHD)

1. Scan PSCBs looking for LC or LP.

   A. If starting segment is an LC retrieved from logical path, mark him LD, is possible.

   B. Ensure no violations of the physical delete rule.

2. Build work area for path.

3. Read and process all segments from top to bottom. Determine how to delete LC or LP.

4. At bottom,

   REQBOTM
   Delete segment(s) 2-11.6

5. Return

To Fig 2-11 Step 5

OUTPUT

Delete Work Area

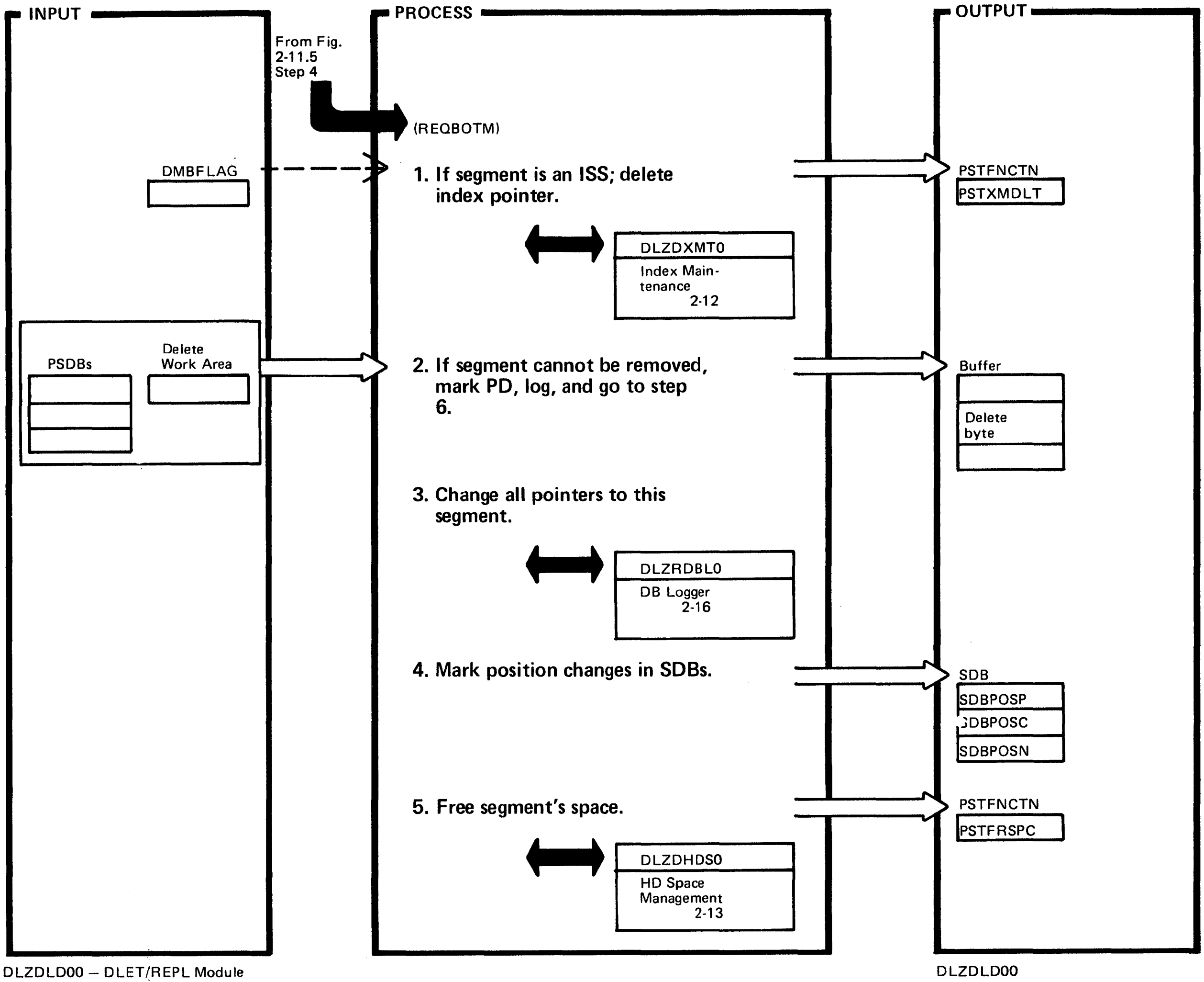
Buffer

DLZDLD00

Extended Description	Routine	Label
1. A. LC will be marked logically deleted (LD) if delete rule = physical or logical and segment not PD (physically deleted).	DLZDLD00	DELTHD ILCDLT
B. A logical parent can have no active logical children. An LC must not be accessable by his logical path.		DELT09 PHYSCAN
2. This is needed to remember where we are, during scan of data base and to build concatenated keys.		DELTHA NEWOMB
3. LCF and LCL pointers in logical parents, and LTF and LTB pointers in logical children, will be updated now.		REQSCAN2 SCANDMB REQDOWN
4. Segments may be marked deleted or physically removed.		REQBOTM
5. All work sets are freed.		ENDLTSCN

Extended Description	Routine	Label

Figure 2-11.6. Delete Segment (Part 1 of 2)



Extended Description	Routine	Label
1. If the index source segment (ISS) has been marked physically deleted (PD), no index maintenance is performed. Delete processing continues with blank or 'NE' status from DLZDXMTO.	DLZDLDD0	REQB01
2. A segment will not be physically removed if still required because of a logical relationship. Note that the delete work area and DL/I blocks (primary PSDBs) are used as input to every step.		REQB02
3. If segment is an LC or LP, the logical relationship pointers (LC, LP, and LT) have already been changed.	DLZDLDD0 DLZDLDA0	FREESPCE FRSPC00
4. The current position (SDBPOSC) is marked 'lost' in this caller's PCB. If any other PCB has position on this segment, the position should be changed to bypass this segment.	DLZDLDA0	PRSPC05 MARKSDB
5. DLZDHDS0 makes the log calls for the physical delete.	DLZDLDA0	FRSPC05G

Extended Description	Routine	Label

Figure 2-11.6. Delete Segment (part 2 of 2)

INPUT	PROCESS	OUTPUT
	6. If starting segment is not yet deleted, get next segment and process it. → To Fig 2-11.5 Step 3	
	7. Return → To Fig 2-11 Step 5	

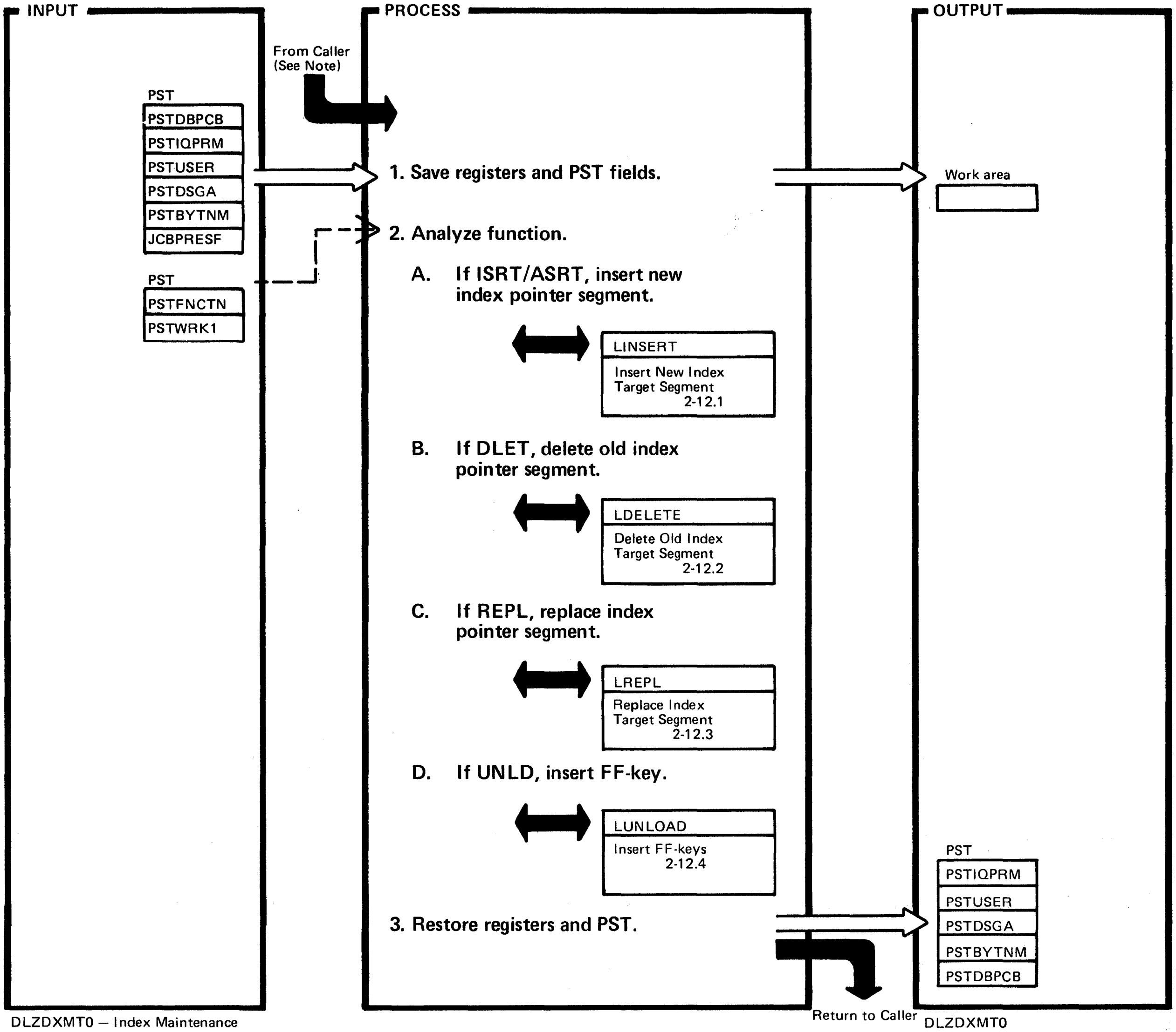
DLZDLD00 – DLET/REPL Module

DLZDLD00

Extended Description	Routine	Label
6. Next segment is physical twins, sibling, or parent.	DLZDLD00	BOTM1B
7. At end, a final log call is made to DLZRDBL0 which signifies delete is finally accomplished.	DLZDLD00	ENDLTSCN

Extended Description	Routine	Label

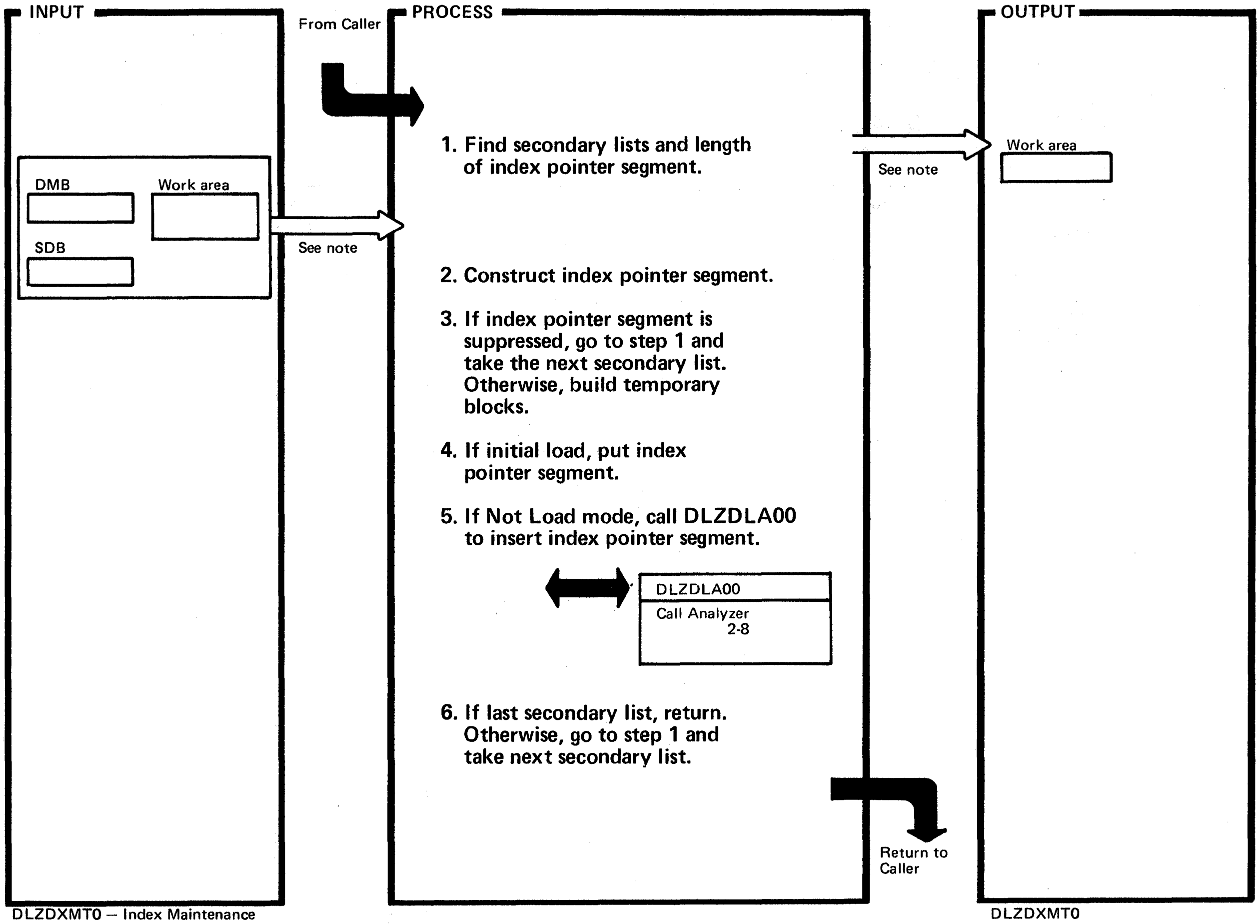
Figure 2-12. Index Maintenance



Extended Description	Routine	Label
Note: DLZDXMT0 is called from DLZDDLE0 or DLZDLD00.		
2. When called from DLZDDLE0, the function is ISRT, ASRT, UNLD, or REPL. When called from DLZDLD00, the function is REPL or DLET. PSTWRK1 contains the PSDB address of the index source segment for DLET or the LSDB address of the index source segment.		
A. Construct and insert all index pointer segments for this index source segment that should not be suppressed.	LINSERT	
B. Construct and delete all old index pointer segments existing for this index source segment.	LDELETE	

Extended Description	Routine	Label
2. (con't)		
C. Construct all old and new index pointer segments that can be constructed from that index source segment. Depending on the data changed and the status of suppression, delete old index pointer segment, or insert new index pointer segment, or delete old and insert new index pointer segment, or replace data of index pointer segment.	LREPL	
D. If DLBL card is provided, write index pointer segment with all FF-keys for all index data bases to belonging to this PCB.	LUNLOAD	

Figure 2-12.1. Insert New Index Target Segment



Extended Description	Routine	Label
Note: The input control blocks are used in all process steps. The output work area is modified in all process steps.		
1. Find SECLISTs and PSDBs of index source segment, index target segment, and index pointer segment and save their address in work area. Decide if primary or secondary has to be built. Find length of index pointer segment, sequence field, segment length, and protected data length.	LBLDWKA	
2. For primary indexes, move HIDAM root sequence field from user I/O area to work area. For secondary indexes, construct SRCH, SUBSEQ, and DDATA fields.	LBLDXNS LGRBACK LNULSUP LCALLBH	
3. When the index entry has to be suppressed due to SRCH equal to NULLVALUE or due to exit routine return code, the index pointer segment is not inserted.		

Extended Description	Routine	Label
3. (con't)		
Build temporary blocks: • SDB • Segment name=sequence field name of index pointer segment • Update Index Maintenance JCB and DSG.	LBLDCTLB	
4. If DLBL cards are provided, write index pointer segment to index data base and call DLZDLOC0 to open index data base if not open yet. Otherwise, write index pointer segment to workfile and call DLZDSEH0 to open the workfile.	LLOAD LWORKDS LCALLBH DLZDLOC0	
5. Prepare DL/I call list to call DLZDLA00 with an *X call.	LINXNS DLZDLA00	
6. When the last secondary list is reached, exit is to LRETURN. On error in secondary lists, exit is to LABND772 (abend code 772).		

Figure 2-12.2. Delete Old Index Target Segment (Part 1 of 2)

INPUT

DMB

Work Area

SDB

See note

From Figure 2-12 Step 2B

PROCESS

1. **Find secondary lists and length of index pointer segment.**
2. **Construct index pointer segment.**
3. **If old index pointer segment is suppressed, go to step 1 and take the next secondary list. Otherwise, build temporary blocks.**
4. **Call DLZDBH00 to read old index pointer segment.**

   DLZDBH00 — DB Buffer Handler 2-15

5. **Change delete flag and zero pointer in index pointer segment.**
6. **Call Logger to log index pointer segment changes.**

   DLZRDBLO — DB Logger 2-16

See note

OUTPUT

Work Area

DLZDXMTO — Index Maintenance

DLZDXMTO

Extended Description	Routine	Label
Note: The input control blocks are used in all process steps. The output work area is modified in all process steps.		
1. Find SECLISTs and PSDBs of index source segment, index target segment, and index pointer segment and save their address in work area. Decide if primary or secondary has to be built. Find length of index pointer segment, sequence field, segment length, and protected data length.	LBLDWKA	
2. For primary indexes, move HIDAM root sequence field from user I/O area to work area. For secondary indexes, construct SRCH, SUBSEQ, and DDATA fields.	LBLDXNS LGRBACK LNULSUP LCALLBH	
3. When the old index entry has to be suppressed due to SRCH equal to NULLVALUE or due to exit routine return code, the index pointer segment is not inserted.		

Extended Description	Routine	Label
3. (con't)		
Build temporary blocks: ● SDB ● Segment name=sequence field name of index pointer segment ● Update Index Maintenance JCB and DSG.	LBLDCTLB	
4. The Buffer Handler is called (PSTSTLEQ) to find the old index pointer segment. If it is not found, or it is already deleted, or the pointer or key are not correct, an NE status code is returned to the caller.	LGOXNS LDOXNS	
5. Delete flag is set to C0.		
6. Chain maintenance and logical delete calls are made to data base module.	DLZRDBLO	

Figure 2-12.2. Delete Old Index Target Segment (Part 2 of 2)

INPUT

PROCESS

7. Call DLZDBH00 to write back the changed index pointer segment.

DLZDBH00
DB Buffer Handler
2-15

8. If last secondary list, return. Otherwise, go to step 1 and take next secondary list.

To Fig 2-12 Step 2B

OUTPUT

DLZDXMT0 – Index Maintenance

DLZDXMT0

Extended Description	Routine	Label
7. The Buffer Handler is called (PSTBFALT) to write the changed index pointer segment back.	DLZRDBL0	
8. When the last secondary list is reached, exit is to LRETURN. On error in secondary lists, exit is to LABND772 (abend code 772).		

Extended Description	Routine	Label

Figure 2-12.3. Replace Index Target Segment (Part 1 of 2)

INPUT

DMB — Work area

SDB

See note

From Figure 2-12 Step 2C

PROCESS

1. Find secondary lists and length of index pointer segment.
2. For primary index, continue with step 1. Otherwise, construct old index pointer segment.
3. Construct new index pointer segment.
4. Replace index pointer segment:
   - A. If old and new index pointer segment are suppressed, go to step 1 and take next secondary list.
   - B. If old index pointer segment was suppressed, insert new index pointer segment.
   - C. If old and new index pointer segment were not suppressed and SRCH and SUBSEQ fields not changed, replace index pointer segment.
   - D. If the old and new index pointer segment were not suppressed and SRCH and SUBSEQ fields were changed, delete old index pointer segment and insert new index pointer segment.

See Note

OUTPUT

Work area

DLZDXMT0 — Index Maintenance

DLZDXMT0

Extended Description	Routine	Label
Note: The input control blocks are used in all process steps. The output work area is modified in all process steps.		
1. Find SECLISTs and PSDBs of index source segment, index target segment, and index pointer segment and save their address in work area. Decide if primary or secondary has to be built. Find length of index pointer segment, sequence field, segment length, and protected data length.	LBLDWKA	
2. Construct old index pointer segment from SRCH, SUBSEQ, and DDATA fields.	LBLDXNS LGRBACK LNULSUP LCALLBH	
3. Construct new index pointer segment from SRCH, SUBSEQ, and DDATA fields.	LBLDXNS LGRBACK LNULSUP LCALLBH	

Extended Description	Routine	Label
4. Replacing of the index pointer is done in different ways, depending on suppression of old and new index pointer segment.		
A. When both old and new index pointer segments are suppressed, no action takes place.		
B. Continue with insert subroutine	LINXNS	
C. DLZDBH00 is called to read the old index pointer segment. On errors, NE is returned. The data base log module is called to log the old index pointer segment, and after the change of the DDATA fields, the new index pointer segment. DLZDBH00 is called again to write the index pointer segment back (PSTBFALT).	LGOXNS LROXNS	
D.	LGOXNS LDOXNS LINXNS	

Figure 2-12.3. Replace Index Target Segment (part 2 of 2)

INPUT

PROCESS

4. (con't)

   E. If the old index pointer segment was not suppressed, but the new one was, delete the old index pointer segment.

5. If last secondary list, return. Otherwise, go to step 1 and take next secondary list.

To Fig 2-12 Step 2C

OUTPUT

DLZDXMTO — Index Maintenance

DLZDXMTO

Extended Description	Routine	Label
5. When the last secondary list is reached, exit is to LRETURN. On error in secondary lists, exit is to LABND772 (abend code 772).		

Extended Description	Routine	Label

Figure 2-12.4. Insert FF-Keys

INPUT

DMB

Work area

SDB

From Fig. 2-12 Step 2D

See Note

PROCESS

1. Loop through all PSDBs to find index source segment.
2. Find length and keylength of index pointer segment.
3. Move FF-key to index pointer segment.
4. Build temporary blocks.
5. Write the index pointer segment to the data base or to the work-file.
6. If last PSDB, return. Otherwise, go to step 1 and get next PSDB.

See Note

To Fig 2-12 Step 3

OUTPUT

Work area

DLZDXMT0 Index Maintenance

DLZDXMT0

Extended Description	Routine	Label
Note The input control blocks are used in all process steps. The output work area is modified in all process steps.		
1. DLZDDLE0 passes the LSDB address of the root segment with an UNLD call. The DDIR address is used and all PSDBs in that DMB are inspected if an index exists.		
2. Find length of index pointer segment and its key length. Decide if primary or secondary index has to be built.	LBLDWKA	
3. Move FFs in the length of the index pointer segment sequence field to the index pointer segment.	LBLDXNS	

Extended Description	Routine	Label
4. Build temporary blocks: ● SDB ● segment name = sequence field name of index pointer segment ● update index maintenance JCB and DSG.	LBLDCTLB	
5. Write index pointer segment to index data base if DLBL cards are provided. Call DLZDLOC0 to open index data base if not yet open.	LLOAD LWORKD5 LCALLBH DLZLOC0	

Figure 2-13. HD Space Management

INPUT

DSG: DSGDMBNO
PST: PSTWRK1

DMB: DMBFBA

PST: PSTFNCTN

From Caller

PROCESS

1. Initialize work fields in the PST.

2. Test for FBA.

A. If yes, go to Figure 2-13.5.

3. Determine function requested.

A. If Get Space, go to Figure 2-13.1.

B. If Get Space Close to Root Anchor Point, go to Figure 2-13.1.

C. If Free Space, go to Figure 2-13.2.

D. If Modify the Bit Map, go to Figure 2-13.3.

E. If Backout Get Space, go to Figure 2-13.4.

F. If Backout Free Space, go to Figure 2-13.2.

G. If Backout Modify Bit Map, go to Figure 2-13.3.

4. Exit.

Return to Caller

OUTPUT

PST: PSTDMBNM, PSTACBNM, USPCE, UMAX

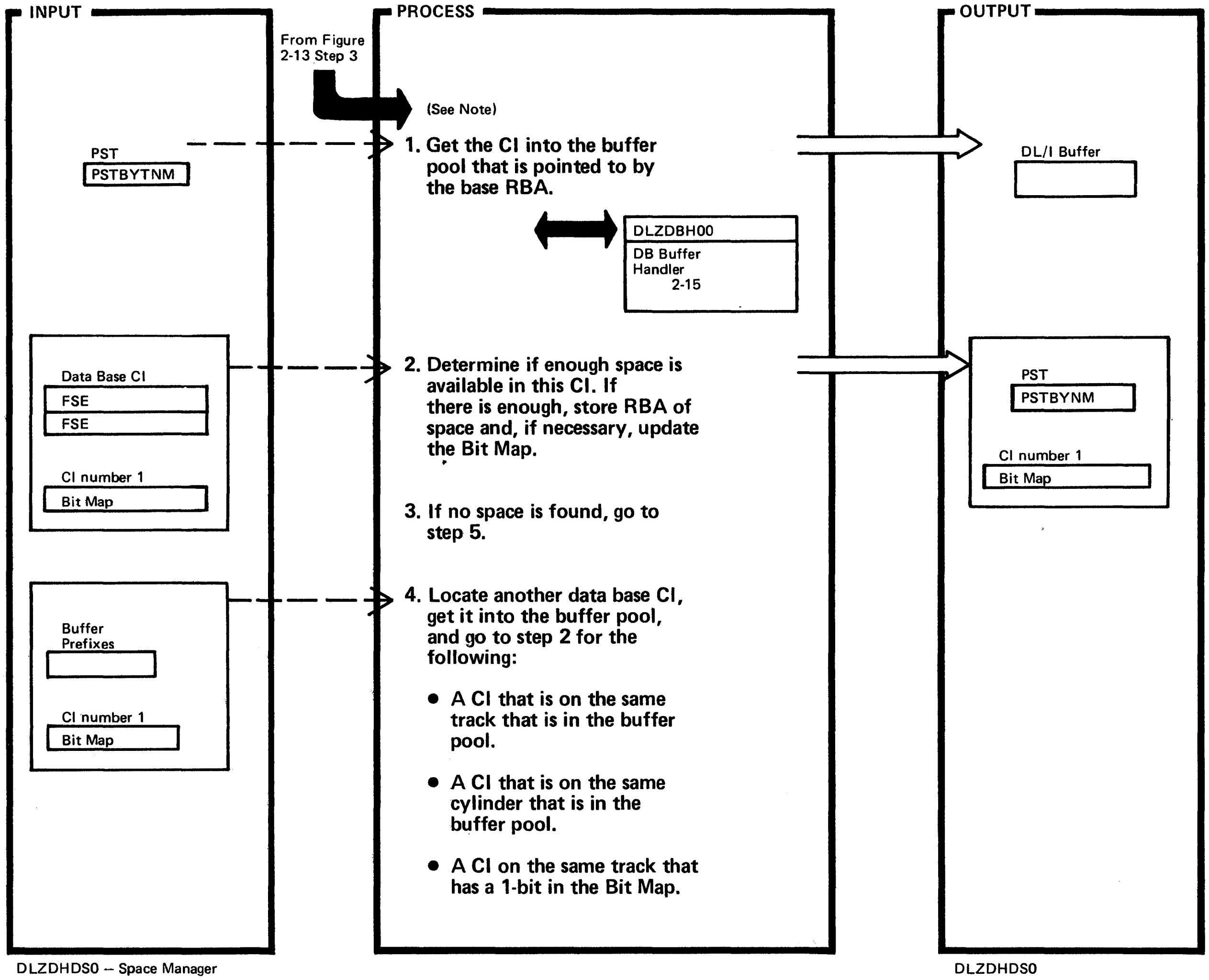
DLZDHDS0 – Space Manager

DLZDHDS0

Extended Description	Routine	Label
1. PSTWRK1 contains the length of the space to be obtained or freed.		
2. A. If the device is FBA, the device characteristics must be obtained and the number of CIs per track and CIs per cylinder calculated.	DLZDCI00	
3. A. Get space in a data base CI for the specified segment as close as possible to a specified base RBA. The caller passes the address of the involved segment's PSDB in R5 and the base RBA in PSTBYTNM.	DLZGGSPC	
B. Get space in a data base CI for the specified segment as close as possible to a root anchor point. The caller passes the address of the involved segment's PSDB in R5 and the CI number/RAP number (in the format BBBR) of the involved root anchor point in PSTBYTNM.	DLZGGSPC	

Extended Description	Routine	Label
3. (con't)		
C. Free space that has been allocated for the specified segment in a data base CI. The caller passes the address of the involved segment's PSDB in R5.	DLZFRSPC	
D. Turn on or off the bit in the Bit Map representing the specified CI of a data base. The caller specifies the CI number in PSTBLKNM.	DLZDHDS0	FIXBTMP
E. Backs out a previously processed 'Get Space' call.	DLZDHDS0	
F. Backs out a previously processed 'Free Space' call.	DLZFRSPC	
G. Backs out a previously processed 'Modify Bit Map' call.	DLZDHDS0	FIXBTMP

Figure 2-13.1. Get Space (Part 1 of 2)



Extended Description	Routine	Label
Note: For the functions 'Get Space' and 'Get Space Close to RAP', the following csects are used: Main routine: DLZDHDS0 DLZDHDS0 calls DLZGGSPC. DLZGGSPC calls DLZRCHBK, DLZLLCLC, DLZRRHPL, DLZRRHMP, DLZMNLCT, and DLZMMUDT. DLZRRHPL calls DLZRCHBK. DLZRRHMP calls DLZRCHBK. 2. If distributed free space has been specified, a check is made if this block is to be left free. If not, a check is made to see if a percentage of this block is to be left free. If so, this percentage is added to the space requested. To determine if enough space is available in a CI, the FSE's in this CI are checked. If there is more than one FSE in a CI, the free space with the largest of the following values that will not cause a Bit Map change is taken:		

Extended Description	Routine	Label
2. (con't) • the size itself • the size+minimum segment length • the size+2. A Bit Map Change is necessary if the data base CI cannot accomodate the maximum size segment because the available space has been used. The Bit Map update is performed by DLZMMUDT. 4. The calculation of the CI numbers for a given range is done by routine DLZLLCLC. Searching through the buffer prefixes is done by routine DLZRRHPL. Searching through the Bit Map is done by routine DLZRRHMP.		

Figure 2-13.1. Get Space (Part 2 of 2)

INPUT

PROCESS

- A CI in the same cylinder that has a 1-bit in the Bit Map.
- A CI being within the delta cylinders that is in the buffer pool.
- A CI being within the delta cylinders that has a 1-bit in the Bit Map.
- The next available CI at the end of the data base.

5. If no space is found, store the error code.

6. Exit.

To Figure 2-13 Step 4

OUTPUT

PST

PSTRTCDE

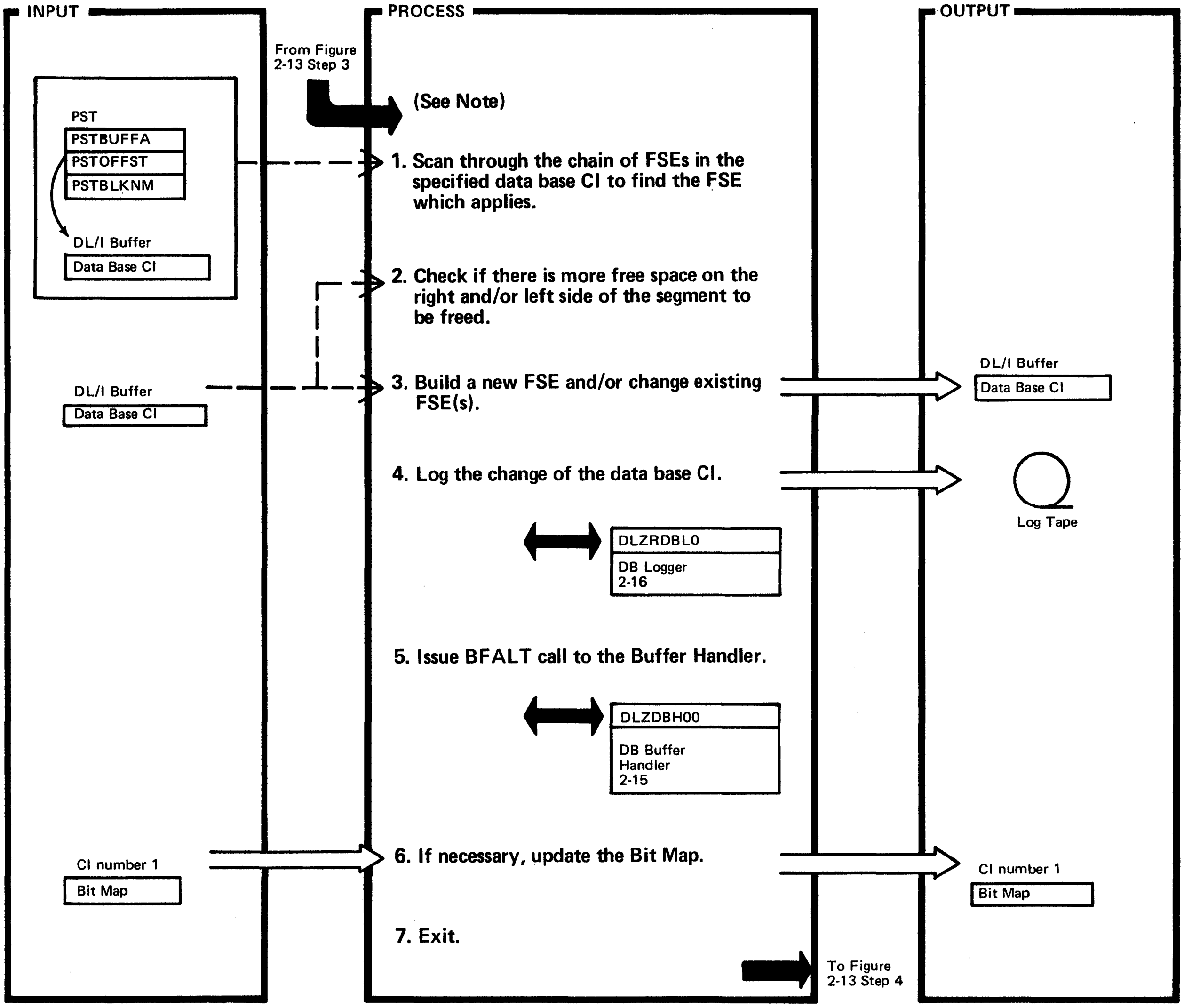
DLZDHDSO — Space Manager

DLZDHDSO

Extended Description	Routine	Label
5. A return code of X'0C' will be returned to the caller.		

Extended Description	Routine	Label

Figure 2-13.2. Free Space



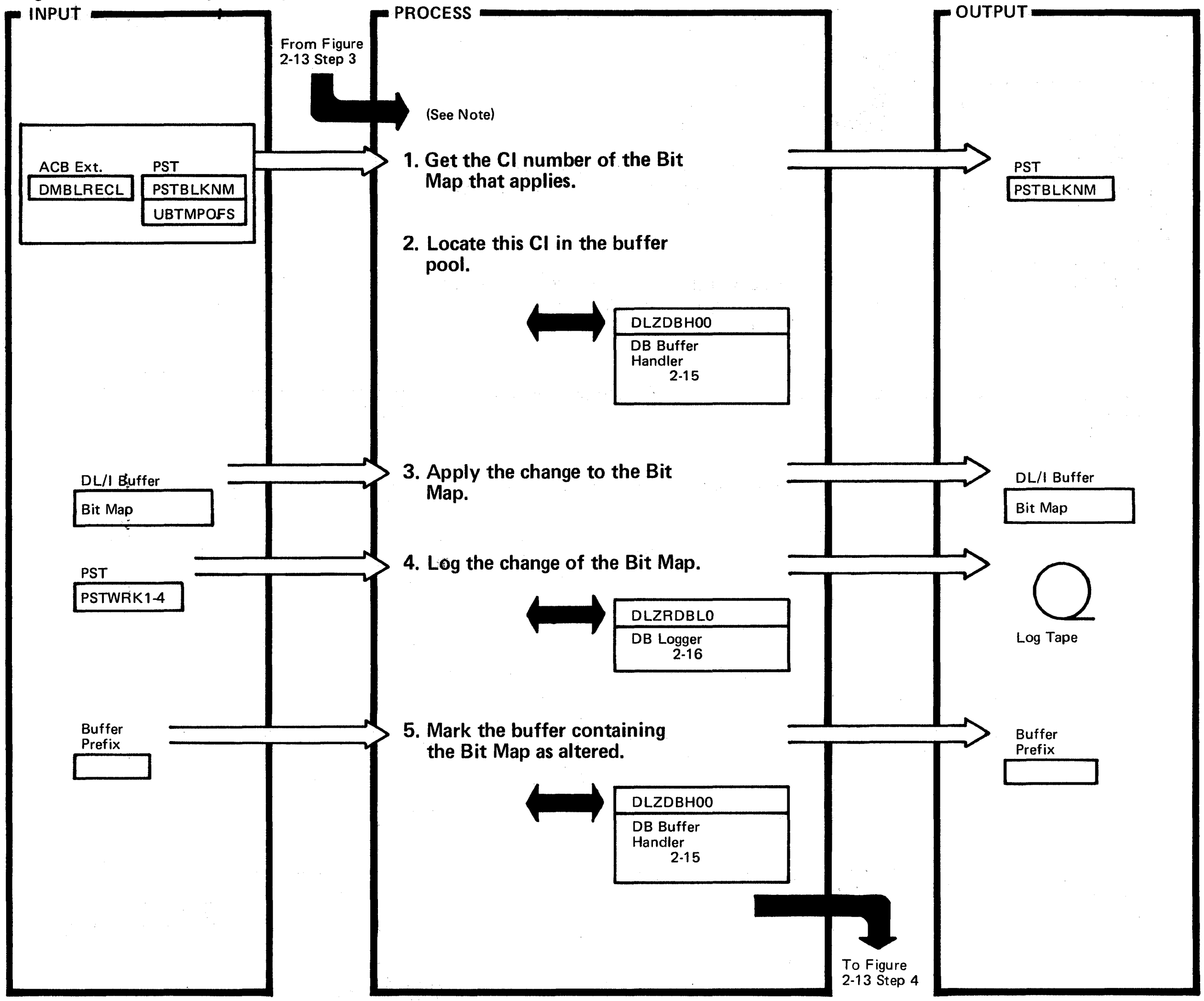
DLZDHDS0 — Space Manager

DLZDHDS0

Extended Description	Routine	Label
Note: For the functions 'Free Space' and 'Backout Free Space', the following csects are used: Main routine: DLZDHDS0 DLZDHDS0 calls DLZFRSP0 DLZFRSP0 calls DLZMMLCT and DLZMMUDT.		
1. The scan will be finished when a FSE with a higher offset than the one in PSTOFFST is reached, or, when the end of the FSE chain is reached.		
2. The purpose of this check is to find out whether there will be a contiguous piece of free space after processing the current Free Space call.		
6. A Bit Map change is necessary if the data base CI can accomodate the maximum size segment after process-		

Extended Description	Routine	Label
6. (con't) ing the Free Space call. In this case, the appropriate bit in the Bit Map has to be turned on. The Bit Map update is performed by routine DLZMMUDT.		

Figure 2-13.3. Modify Bit Map



DLZDHDS0 — Space Manager

DLZDHDS0

Extended Description	Routine	Label
Note: For functions 'Fix Bit Map' and 'Backout Fix Bit Map', the following csects are used:		
Main Routine: DLZDHDS0		
DLZDHDS0 calls DLZMMLCT and DLZMMUDT.		
1. This step is performed by routine DLZMMLCT.		
2. A 'Byte Locate' call is issued.	DLZDBH00	
3. The update of the Bit Map is performed by routine DLZMMUDT.		
5. A 'Buffer Alter' call is issued.	DLZDBH00	

Extended Description	Routine	Label

Figure 2-13.4. Backout Get Space

INPUT

PST: PSTBUFFA, PSTOFFST

DL/I Buffer

PST: PSTOFFST, AOFF

DL/I Buffer

Buffer Prefix

CI number 1: Bit Map

DLZDHDS0 — Space Manager

PROCESS

From Figure 2-13 Step 3

(See Note)

1. Scan through the FSEs of the given CI and find the FSE that applies.
2. If the previously freed space cannot be found, store error code and go to step 9.
3. If the previously freed space is not in the middle of a freed area, go to step 5.
4. Create a dummy FSE.
5. Change the FSE(s) to reflect the aquisition of the space.
6. Prepare the information for logging the data base change.
7. Mark the buffer as altered.

   DLZDBH00 — DB Buffer Handler 2-15
8. Update the Bit Map if necessary.
9. Exit.

To Figure 2-13 Step 4

OUTPUT

AOFF

PST: PSTRTCDE

DL/I Buffer

PST: PSTWRK1-4

Buffer Prefix

CI number 1: Bit Map

DLZDHDS0

Extended Description	Routine	Label
Note: This function backs out a Free Space call processed previously. The following csects are used: Main routine: DLZDHDS0 DLZDHDS0 calls DLZRCHBK.		
2. PSTOFFST contains the offset to the part of the data base CI which was freed during the Free Space call to be backed out.		
3. A return code of X'0C' is stored in PSTRTCDE.		
7. A 'Buffer Alter' call is issued to the Buffer Handler.	DLZDBH00	

Extended Description	Routine	Label
8. A Bit Map update is necessary if the data base CI cannot accomodate the maximum length segment after backing out the previously processed Free Space call. The Bit Map update is performed by routine DLZMMUDT.		

Figure 2-13.5. FBA Support Device Characteristics Routine

INPUT

AMD
AMDDSN

PST
PSTVLSR

DLZDHDS0 — Space Manager

PROCESS

From Figure 2-13 Step 2

1. Issue SHOWCAT to obtain volume serial number of data base.
2. Issue GETVCE to obtain device characteristics.
3. Is device FBA?
4. If no, issue message DLZ831I and end task.
5. If yes, calculate CIs per track and CIs per cylinder.
6. Calculate scan value.
7. Exit.

To Figure 2-13 Step 3

OUTPUT

PST
PSTSWKAR

PST
PSTSWKAR

DMB
DMBCICYL
DMBCITRK

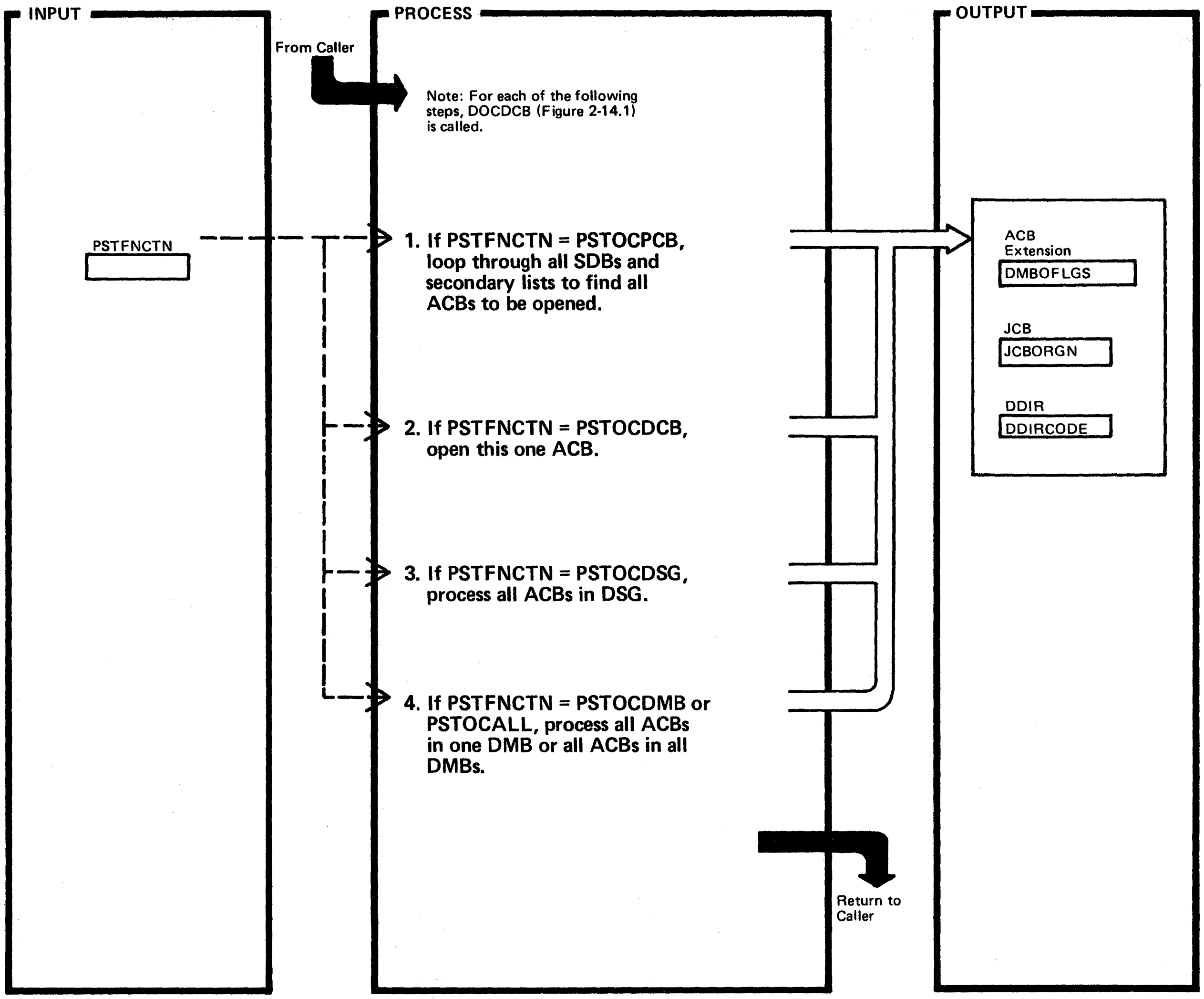
DMB
DMBACBDL

DLZDHDS0

Extended Description	Routine	Label

Extended Description	Routine	Label

Figure 2-14. Open/Close



DLZDLOCO – Open/Close

DLZDLOCO

Extended Description	Routine	Label
1. This function is used by the utilities DLZRDBC0 and DLZURGP0. It is also used by DLZDLA00 when the first data base call to a not open data base is issued (batch only). For PROCOPT=L, only one data base is opened. For all other processing options, all related data bases are opened as well (index data bases and logically related data bases).	PCENTRY PSROUT DOCDCB	AROUND
2. DLZDLR00 uses this function for positioning a HSAM data base at the start point. It is also used by DLZURDB0. It opens only one ACB, i.e. for HISAM only KSDS or ESDS.	ACBENTRY DOCDCB	
3. DLZDLD00 uses this function when it finds a logically related data base that is not opened (this can happen because of delete sensitivity propagation).	DGENTRY DOCDCB	

Extended Description	Routine	Label
4. PSTOCALL + PSTOCOPN : DLZOLI00 uses this call to open all data bases in the system eligible for initial opening (online only). PSTOCALL + PSTOCCLS : This call is used to close all ACBs in the DL/I system (e.g. DLZDLA00). PSTOCDMB : This call is used by DLZOLI00 for deferred opening (online). It is also used by DLZDXMT0 and by data base utilities. It opens/closes one ACB (two ACBs for HISAM).	DENTRY DROUTINE DOCDCB	

Figure 2-14.1. Open/Close DOCDCB Routine

INPUT	PROCESS	OUTPUT
R11: ACBEXT	From Fig. 2-14 1. Issue DOS OPEN. 2. Issue 'MODCB' to update the exit list. 3. Issue 'SHOWCB' and compare DMB entries to VSAM define entries. 4. For an empty ESDS file, write control record. 5. Log the open record. 6. For HISAM KSDS, go back to Step 1 to open ESDS. 7. Call compression routine, if necessary. Return to Fig 2-14	ACBEXT: DMBFLGS

DLZDLOCO – Open/Close

DLZDLOCO

Extended Description	Routine	Label
1. This part is called from all steps of Figure 2-14 for opening. If the data base is open, return immediately. Immediate return is also done when the call is PSTOCALL and initial opening is not planned. Unsuccessful opens have return error code in PST and flag in JCB. 'DLZ020I' is issued.	DOCDCB	DOCOPEN
2. The exit list is updated with the address of the error handling routines of DLZDBH00.	DOCMOD	
3. Control interval size, relative key position, and key length of DMB is compared to VSAM catalog entries. MISMATCH: DLZ025I, DLZ027I, and DLZ028I. For HISAM, the number of logical records in VSAM catalog has to be zero for PROCOPT=L. For HD, the high used RBA is inspected. Message 'DLZ023I' is issued for conflicts.	DOCSHOW	

Extended Description	Routine	Label
4. The first control interval is written (for HISAM, as many records as fill one CI). It contains DL/I control information. For HD, the ACB is closed and opened again to simulate 'NOT LOAD' to VSAM.	DOCFIRST	NOTHIDAM
7. All PSDBs are inspected to determine if a compaction routine with 'INIT' specified exists.	DOCVARI	

Figure 2-15. DB Buffer Handler (Part 1 of 2)

INPUT

PST

PSTFNCTN

Flag Name	Hex Code
PSTBYLCT	02
PSTBKLCT	01
PSTBYALT	06
PSTBFALT	05
PSTGBSPC	03
PSTFBSPC	04
PSTBFMPT	04
PSTPGUSR	07

DLZDBH00 – Buffer Handler CSECT

From Caller

PROCESS

DLZDBH00:

What function is requested?

1. For HDAM or HIDAM:

   A. Byte locate.
   (See Figure 2-15.1, Step 1)

   B. Block locate.
   (See Figure 2-15.1, Step 4)

   C. Byte alter.
   (See Figure 2-15.2, Step 1)

   D. Mark buffer as altered.
   (See Figure 2-15.2, Step 3)

   E. Get buffer space.
   (See Figure 2-15.3)

   F. Free buffer space.
   (See Figure 2-15.9)

   G. Mark buffer as empty.
   (See Figure 2-15.9)

   H. Purge buffers.
   (See Figure 2-15.10)

OUTPUT

DLZDBH00

Extended Description	Routine	Label
1.		
A. Locate relative byte number.	DLZDBH00	BYLCT
B. Locate relative block number.	DLZDBH00	BKLCT
C. Locate a relative byte number and mark buffer altered.	DLZDBH00	BYALT
D. Mark a buffer containing data as altered.	DLZDBH00	BFALT
E.	DLZDBH00	GBSPC
F.	DLZDBH03	MRKEMPT
G.	DLZDBH03	MRKEMPT
H. Purge all buffers altered by a task.	DLZDBH03	PGUSR1

Extended Description	Routine	Label

Figure 2-15. DB Buffer Handler (Part 2 of 2)

INPUT

PST

PSTFNCTN

Flag Name	Hex Code
PSTBYLCT	02
PSTSTLEQ	09
PSTSTLBG	0C
PSTWRITE	08
PSTPUTKY	0D
PSTMSPUT	0E
PSTBFALT	05
PSTGETNX	0B
PSTERASE	0A

DLZDBH00 — Buffer Handler CSECT

PROCESS

2. For HIDAM index, Simple HISAM, or HISAM (ESDS or KSDS).

   A. Read a record.

   B. SETL equal.

   C. SETL begin.

   D. Write a new record to HISAM ESDS.

   E. Insert a record by key into a HISAM KSDS.

   F. Insert record(s) sequentially into a HISAM KSDS.

   G. Write a record by RBA to a KSDS or ESDS.

   H. Read the next record in a KSDS.

   I. Erase a record from a KSDS.

3. Exit

Return to Caller

OUTPUT

DLZDBH00

Extended Description	Routine	Label
2.		
A. Read a record by RBA from a KSDS or ESDS.	DLZDBH02	HSREAD
B. Get a record by root key from a KSDS.	DLZDBH02	STLEQ
C. Read the record containing the first root segment in a KSDS.	DLZDBH02	STLBG
D.	DLZDBH02	LOWRITE
E.	DLZDBH02	PUTKY
F.	DLZDBH02	MSPUT
G.	DLZDBH02	HSWRITE
H.	DLZDBH02	GETNX
I.	DLZDBH02	HSWRITE

Extended Description	Routine	Label

Figure 2-15.1. Byte Locate/Block Locate

INPUT

PST
PSTBYTNM

PST
PSTBLKNM

DLZDBH00 — Buffer Handler CSECT

PROCESS

From Figure 2-15 Step 1A

BYLCT:

1. Convert the given RBA to a VSAM control interval (CI) number and an offset within the control interval.

2. Locate the buffer address.

LOCATE Routine 2-15.4

3. Go to Step 6.

From Figure 2-15 Step 1B

BKLCT:

4. Locate the buffer address.

LOCATE Routine 2-15.4

5. Convert the CI number to a RBA.

6. Exit.

To Figure 2-15 Step 3

OUTPUT

PST
PSTBLKNM
PSTOFFST

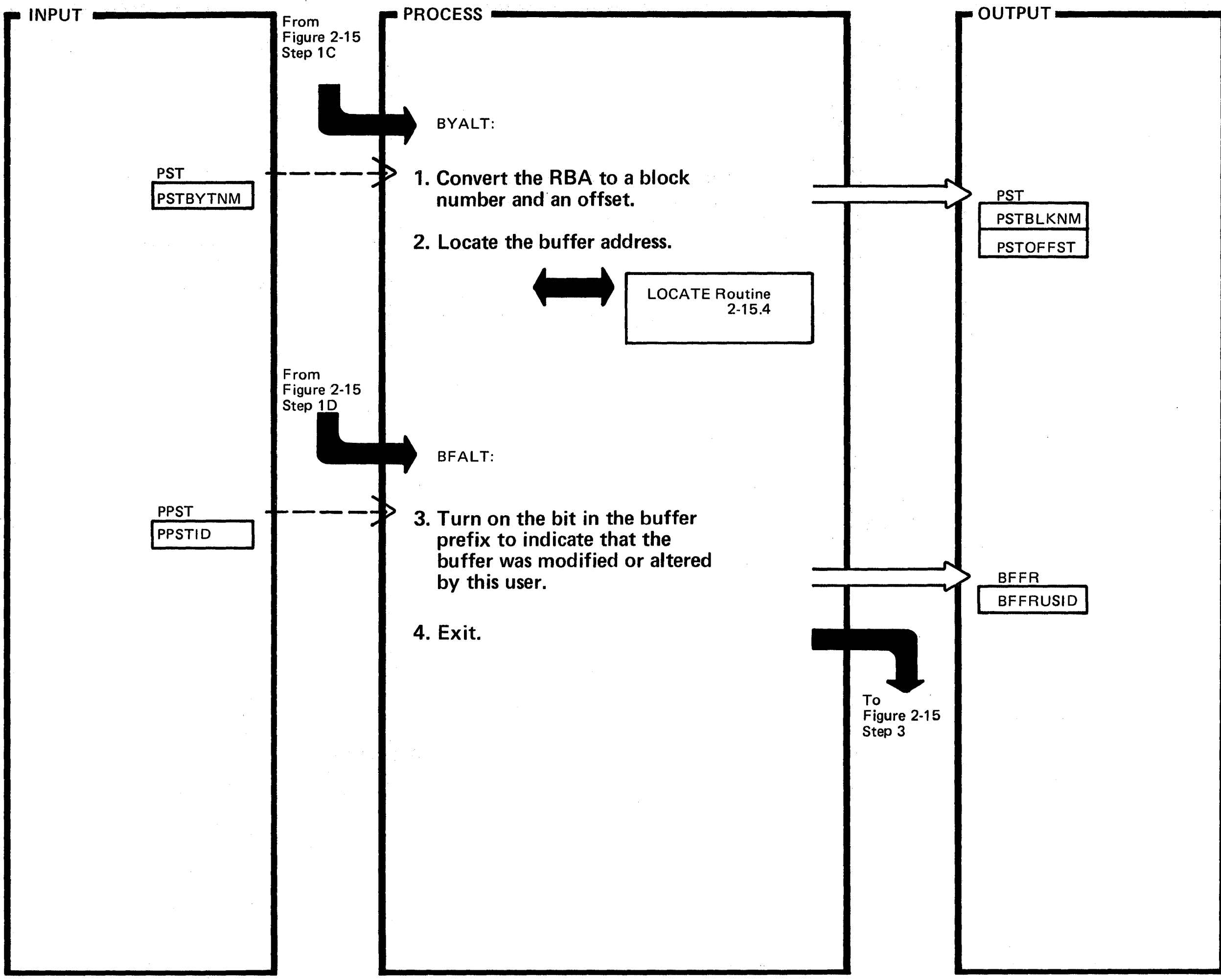
PST
PSTBYTNM

DLZDBH00

Extended Description	Routine	Label
1. The relative byte number for the control interval is retrieved from PSTBYTNM.	DLZDBH00	CONVER
5. The same as in Step 1 except that a control interval number is passed in PSTBLKNM.		

Extended Description	Routine	Label

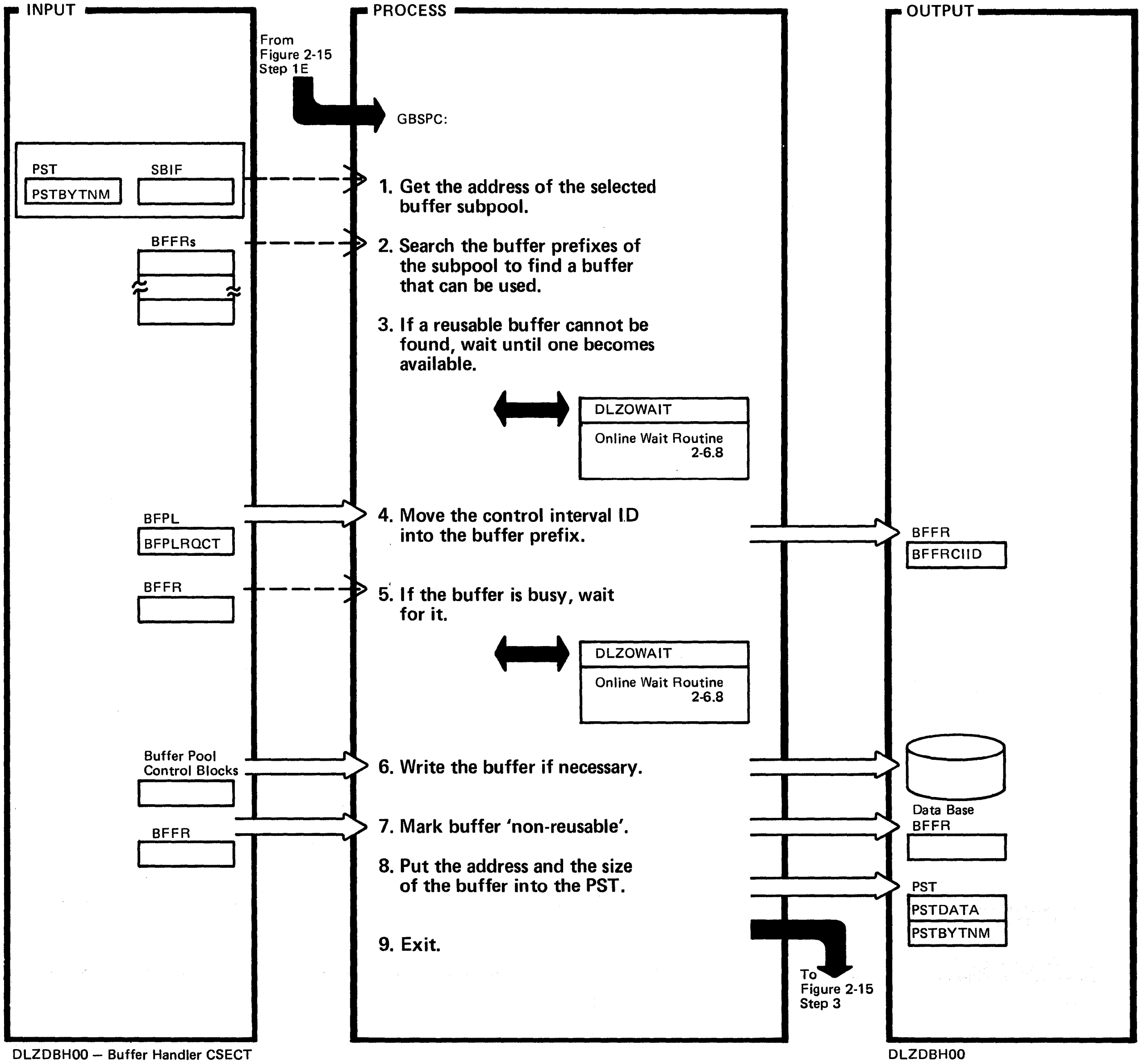
Figure 2-15.2. Byte Alter/Buffer Alter



Extended Description	Routine	Label
1. Byte alter is a combination of byte locate (Figure 2-15.1, Steps 1-3) and buffer alter (Figure 2-15.2, Steps 3 and 4).		
3. The bit that is turned on is in the 2-byte field BFFRUSID. The 16 bits correspond from right to left to the user ID indicated in the PPST. If a user ID higher than 16 is assigned, two or more users share the same bit.	DLZDBH00	MARKALT

Extended Description	Routine	Label

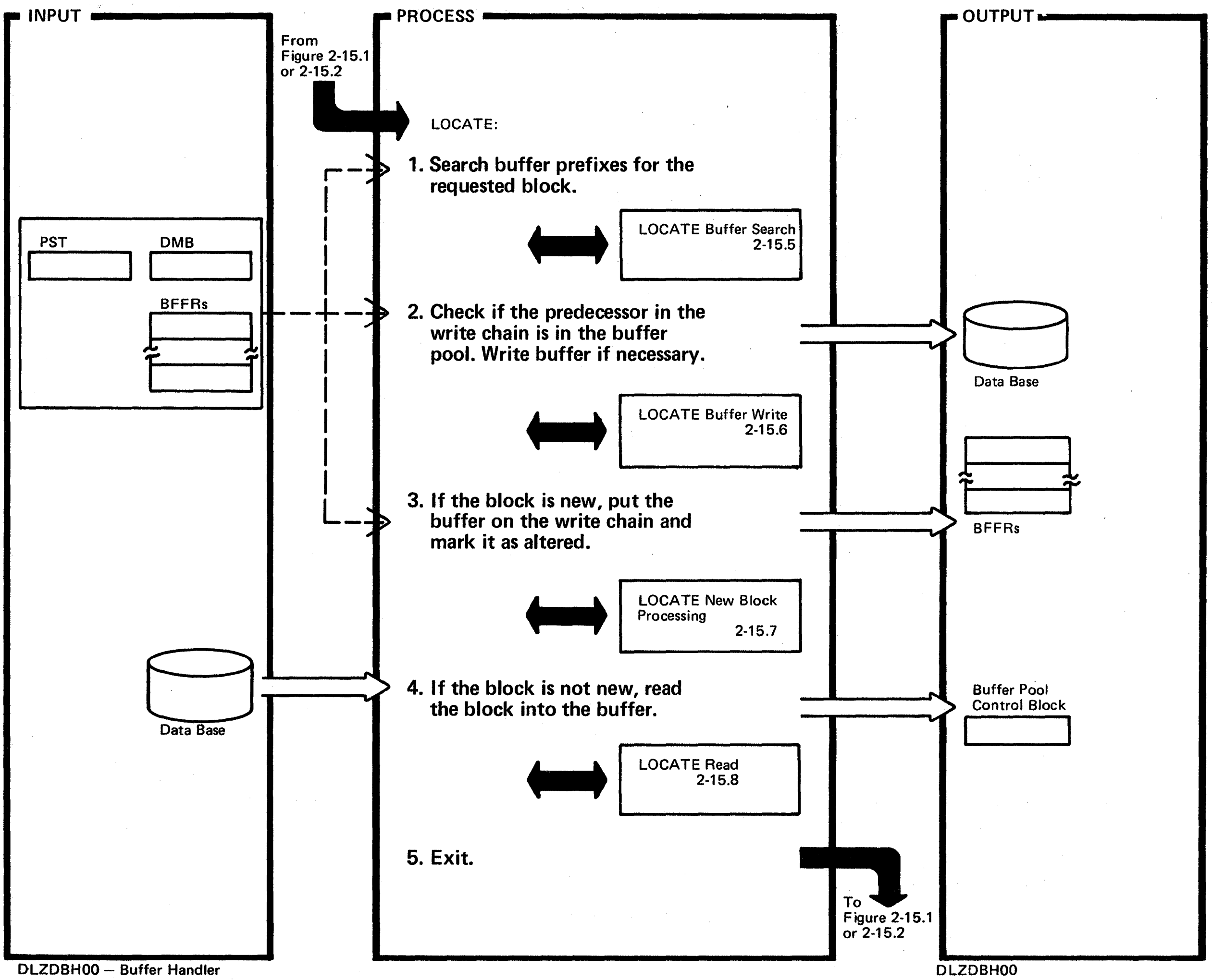
Figure 2-15.3. Get Buffer Space



Extended Description	Routine	Label
1. The subpool information table (SBIF) is used to find a buffer subpool with buffers that contain the least number of bytes needed for this space request.		
2. Buffers that are marked non-reusable or are permanent write error buffers cannot be used.		

Extended Description	Routine	Label

Figure 2-15.4. LOCATE Routine



Extended Description	Routine	Label

Extended Description	Routine	Label

Figure 2-15.5. LOCATE Buffer Search

INPUT

BFFRs | PST
PSTBLKNM
PSTACBNM
PSTDMBNM

BFFRs | SBIF
Use Chain

DLZDBH00 – Buffer Handler CSECT

PROCESS

From Figure 2-15.4 Step 1

LOCATE:

1. Search the buffer prefixes for the requested data base CI (Control Interval).

   A. If the CI is not found, go to Step 4.

   B. If the CI is found as pending in the buffer pool, ENQ on this CI. After control is returned, go back to Step 1.

   C. If the CI is found in the buffer pool and is not busy, go to Step 2.

   D. If the CI is found in the buffer pool and is busy, ENQ on this CI. After control is returned, go back to Step 1.

2. Pass the buffer to the requestor.

3. Exit.

Return to Caller

4. Search the use chain from top bottom for a buffer that can be used.

5. If no buffer is available, ENQ on the pending CI of the buffer on the bottom of the use chain. After control is returned, go back to Step 4.

6. Exit.

To Figure 2-15.4 Step 2

OUTPUT

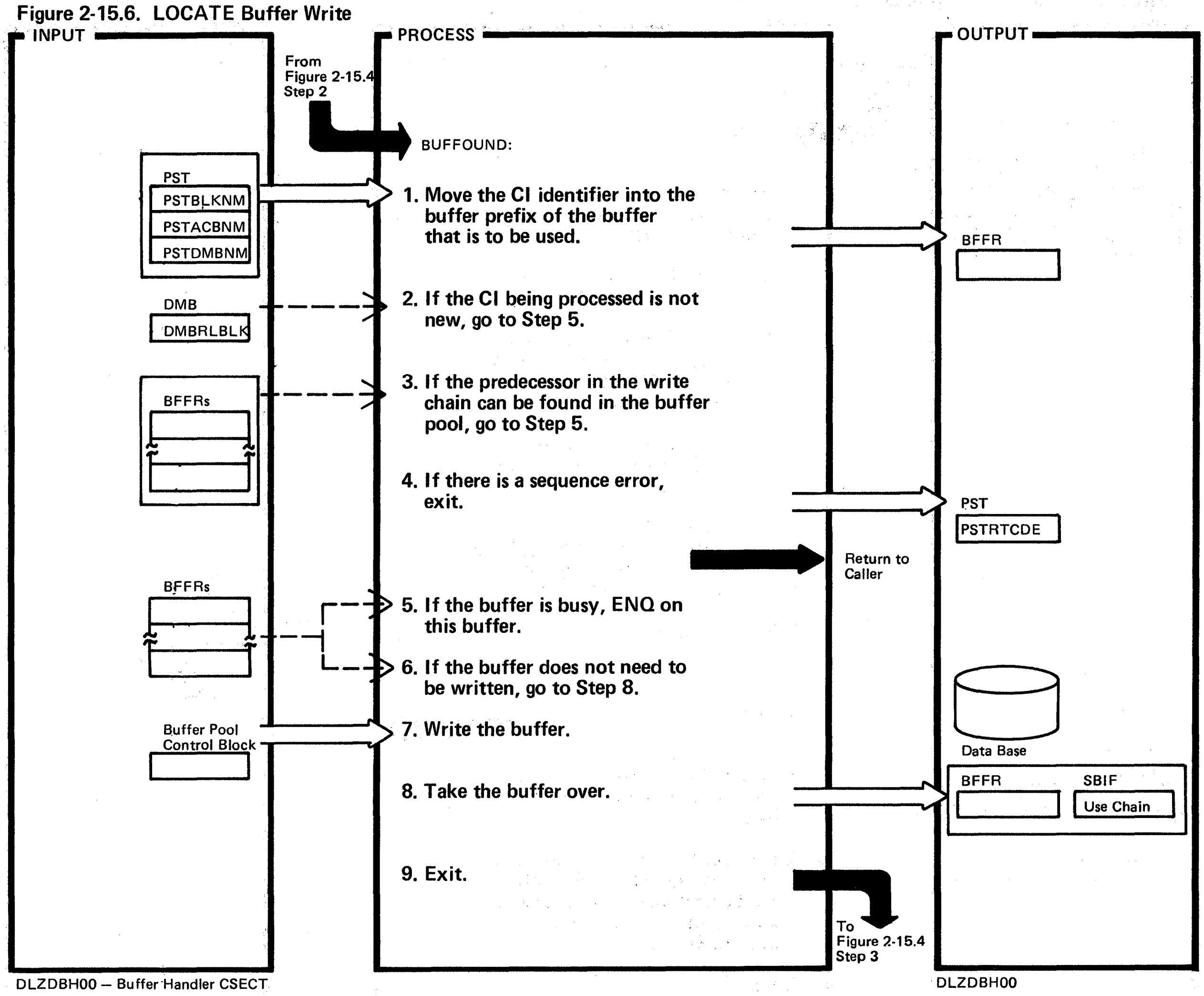
PST
PSTBUFFA
PSTDATA

DLZDBH00

Extended Description	Routine	Label
2. Put the buffer prefix address into PSTBUFFA and the buffer address into PSTDATA. 4. A buffer can be used if: ● It is not marked non-reusable. ● It is not a permanent write error buffer. ● It is not currently enqueued for a pending CI.		

Extended Description	Routine	Label

Figure 2-15.6. LOCATE Buffer Write



Extended Description	Routine	Label
1. Moving the CI identifier means enqueuing on the pending CI.		
2. This check is made to ensure that the CIs of the data base get initialized in sequence.		
4. X'04' is stored in PSTRTCDE.		
5. A buffer is busy if: ● It is being read into. ● It is being written. ● It is waiting for its predecessor in the write chain to be written.		
8. 'Taking over' a buffer consists of: ● Moving the CI identifier from BFFRNPST to BFFRPST. ● Turning off BFFRPNNQ and turning on BFFREXNQ in BFFRSW. ● Putting the buffer at the top of the use chain. ● Clearing the buffer (with zeros).		

Extended Description	Routine	Label

Figure 2-15.7. LOCATE New Block Processing

INPUT

BFFRs

PPST

PPSTID

PROCESS

From Figure 2-15.4 Step 3

TESTNEW1:

1. Search buffer pool for the predecessor in the write chain. If it is found, go to Step 3.
2. If the predecessor cannot be found, go to system ABEND.

DLZABEND
Abend 845
2-4.2

3. If the predecessor is not being written, go to Step 5.
4. Enqueue the predecessor that is being written.
5. If the predecessor is not a permanent write error buffer, go to Step 7.
6. Mark the current buffer as a permanent write error buffer. Then go to Step 9.
7. Put the buffer on the bottom of the write chain.
8. Mark the buffer as altered.
9. Put the buffer prefix address and the buffer address into the PST.
10. Exit.

To Figure 2-15.4 Step 5

OUTPUT

BFFR

PST

PSTBUFFA

PSTDATA

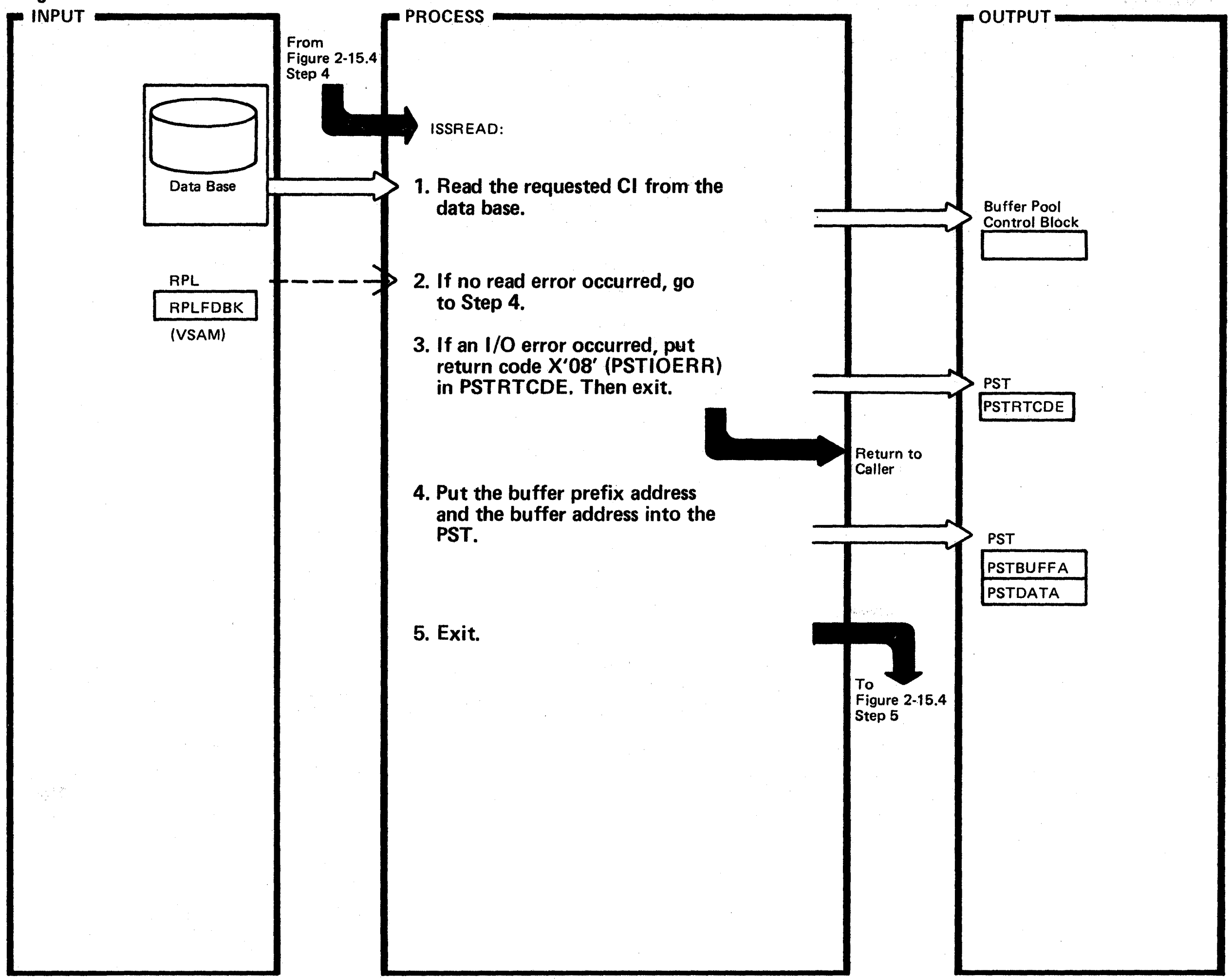
DLZDBH00 – Buffer Handler CSECT

DLZDBH00

Extended Description	Routine	Label
4. The purpose for enqueuing the predecessor is to wait for completion of the writing. This is necessary to find out if the buffer is a permanent error buffer.		

Extended Description	Routine	Label

Figure 2-15.8. LOCATE Read



DLZDBH00 – Buffer Handler CSECT

DLZDBH00

Extended Description	Routine	Label

Extended Description	Routine	Label

Figure 2-15.9. Free Buffer Space

INPUT

PST
PSTBLKNM
PSTDMBNM
PSTACBNM

BFPL
BFPLSUBD

DMB
Subpool
Directory

BFFRs

BFFR

DLZDBH03 – Buffer Handler CSECT

PROCESS

From
Figure 2-15
Steps 1F
and 1G

MRKEMPT:

1. Get the first buffer prefix in the subpool that applies to the call.

2. If PST fields PSTDMBNM, PSTACBNM, and PSTBLKNM do not match BFFR fields BFFRDMB, BFFRDCB, and BFFRCIID, go to Step 4.

3. If the PST fields match the BFFR fields:

   A. Mark the buffer as empty.

   B. Issue RELPAG macro.

   C. Put the buffer on the bottom of the use chain.

4. If this is not the last buffer prefix, get the next one and go back to Step 2.

To
Figure 2-15
Step 3

OUTPUT

BFFR

SBIF

DLZDBH00

Extended Description	Routine	Label
1. The DMB subpool directory and the DMB number in PSTDMBNM are used to find the buffer subpool that applies to the call. 2. The caller can have the buffer handler free: • Only one buffer. (PSTDMBNM, PSTACBNM, and PSTBLKNM ≠0). • All buffers of a data set. (PSTDMBNM and PSTACBNM ≠0; PSTBLKNM = 0). • All buffers of a data base. (PSTDMBNM = DMB number of the data base; PSTACBNM and PSTBLKNM = 0).		

Extended Description	Routine	Label

Figure 2-15.10. Purge Buffers (CHKP Function) (Part 1 of 2)

INPUT	PROCESS	OUTPUT
	From Figure 2-15 Step 1H → PGUSR:	
PST: PSTDBPCB; PCB: DBPCBJCB	1. Get address of the JCB.	
JCB: JCBPRESF	2. If function is not CHKP, go to Step 19.	
PST: PSTPSB; PDIR: PDIRSILA; PSIL: PSILDIRA	3. Get address of DDIR entry.	
DDIR: DDIRADDR	4. Get address of DMB prefix.	
DMB: DMBORG	5. If data base organization is not SHISAM, HISAM, or index data base, go to Step 17.	
ACBXT: DMBACBRP	6. Get the RPL address.	
RPL: RPLACB (VSAM)	7. If no request were made on this RPL, go to Step 11.	
	8. Go test if ACB busy. ↔ TESTACB — Test ACB Routine 2-15.12	
DMB: DMBORG	9. Issue VSAM ENDREG for this RPL.	

DLZDBH03 – Buffer Handler CSECT

DLZDBH00

Extended Description	Routine	Label
2. Field JCBPRESF in the JCB is checked for encoded checkpoint function (FUNCCHKP X'85'). If function is not CHKP, go purge the DL/I buffers.		
5. If data base organization is not SHISAM (field DMBORG; bitDMBSHIS X'01' not set on), or HISAM (bit DMBISAM1 X'02' not set on), or an index data base (bit DMBNDEX X'08' not set on) go update to next PSIL entry.		
9. VSAM ENDREQs are issued for every SHISAM, HISAM, and index data base to ensure that the VSAM buffers are written to the data base.		

Extended Description	Routine	Label

Figure 2-15.10. Purge Buffers (CHKP Function) (Part 2 of 2)

INPUT

ACBXT
DMBACBRP

RPL
RPLACB
(VSAM)

PSIL
PSILLNGH

PSIL
PSILDIRA

DLZDBH03 – Buffer Handler CSECT

PROCESS

10. Post DMB ECB complete.
11. If HISAM data base, update to next ACB extension.
12. Get the RPL address.
13. If no request were made on this RPL, go to Step 17.
14. Go test if ACB busy.

TEST ACB
Test ACB Routine 2-15.12

15. Issue VSAM ENDREG for this RPL.
16. Post DMB ECB complete.
17. Update to next PSB intent list entry.
18. If not end of PSB intent list, return to Step 3.
19. Go to normal purge.

To Figure 2-15.11

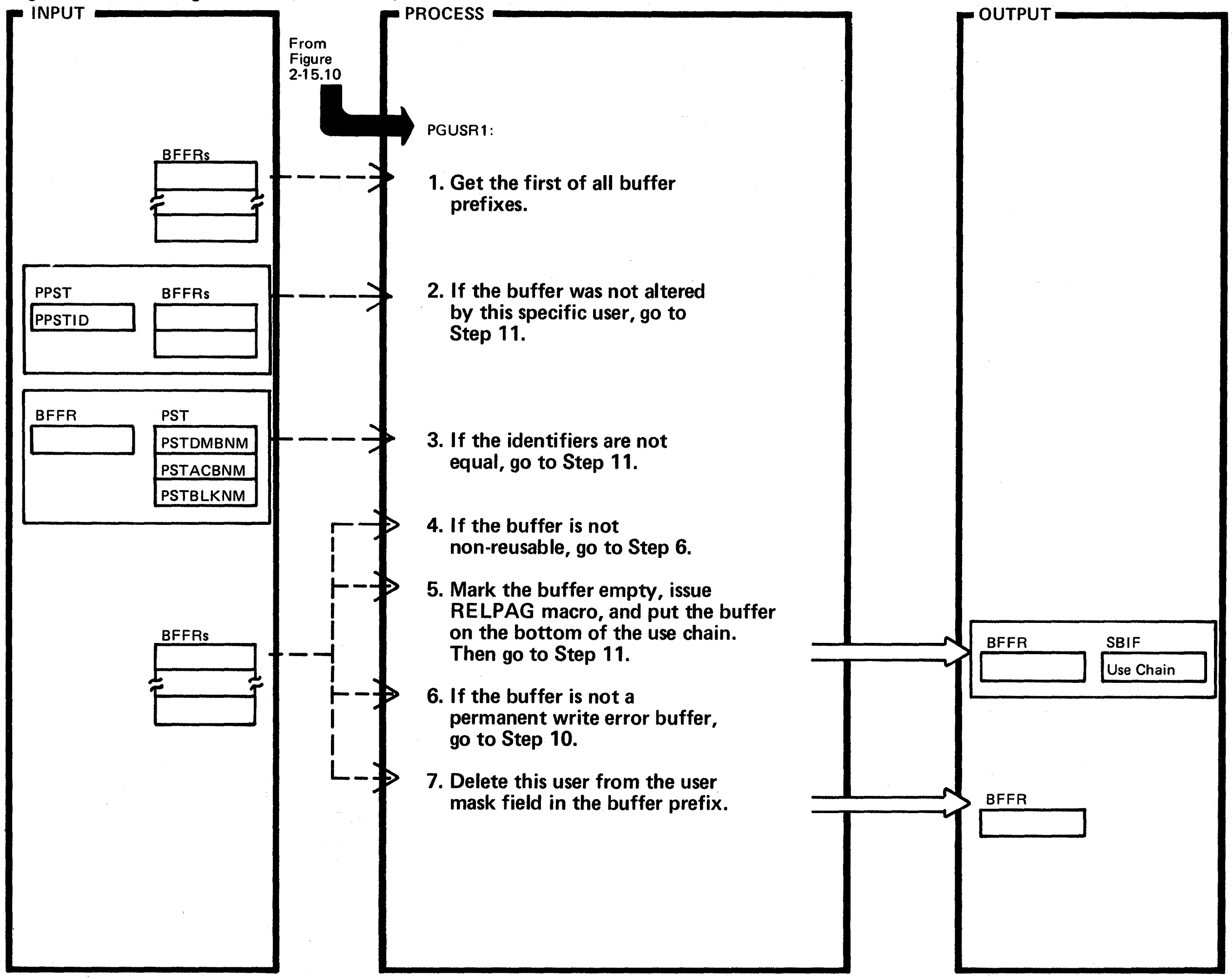
OUTPUT

DLZDBH00

Extended Description	Routine	Label
11. If the entry is for HISAM, update to the next ACB extension. For HISAM, ENDREQs must be issued for both the KSDS and ESDS.		
18. Continue scan of the PSB intent list until all have been processed. When processing is completed, go purge the DL/I buffers.		

Extended Description	Routine	Label

Figure 2-15.11. Purge Buffers (Part 1 of 2)



DLZDBH03 — Buffer Handler CSECT

DLZDBH00

Extended Description	Routine	Label
1. This routine scans all buffer prefixes.		
3. The caller may select a certain data base, a certain data set, or certain buffers to be purged. The choice is indicated by putting the number of the desired item into PSTDMBNM, PSTACBNM, or PSTBLKNM. Zeroes in these fields indicate that purging of all components of the item on the next higher level is desired. This module checks the contents of the above mentioned PST fields against the contents of fields BFFRDMB, BFFRACB, and BFFRCIID in the buffer prefix.		
4. Buffers that are non-reusable are freed during a purge call.		
6. Permanent write error buffers are not freed until all tasks, which either altered the buffer, or might be interested in it because they use the data base, have terminated.		

Extended Description	Routine	Label
7. Before the bit in BFFRUSID, which corresponds to the user identifier (in the PPST) of the current task, is turned off, a check is made whether any tasks are active that would share the bit with the current task. (Refer to the notes in the Figure for routine BFALT.)		

Figure 2-15.11. Purge Buffers (Part 2 of 2)

INPUT

BFFR

PSB
PCB
JCB
DSG

Buffer Pool Control Block
Data Base Control Interval (CI)

DLZDBH03 – Buffer Handler CSECT

PROCESS

8. If BFFRUSID is now zero, or if there are no more potential users for this buffer:

   A. Mark the buffer empty.

   B. Issue RELPAG macro.

   C. Put the buffer on the bottom of the use chain.

9. Go to Step 11.

10. Write buffer to Disk.

11. If this is not the last buffer prefix, get the next one and return to Step 2.

12. Exit.

To Figure 2-15 Step 3

OUTPUT

BFFR

SBIF
Use Chain

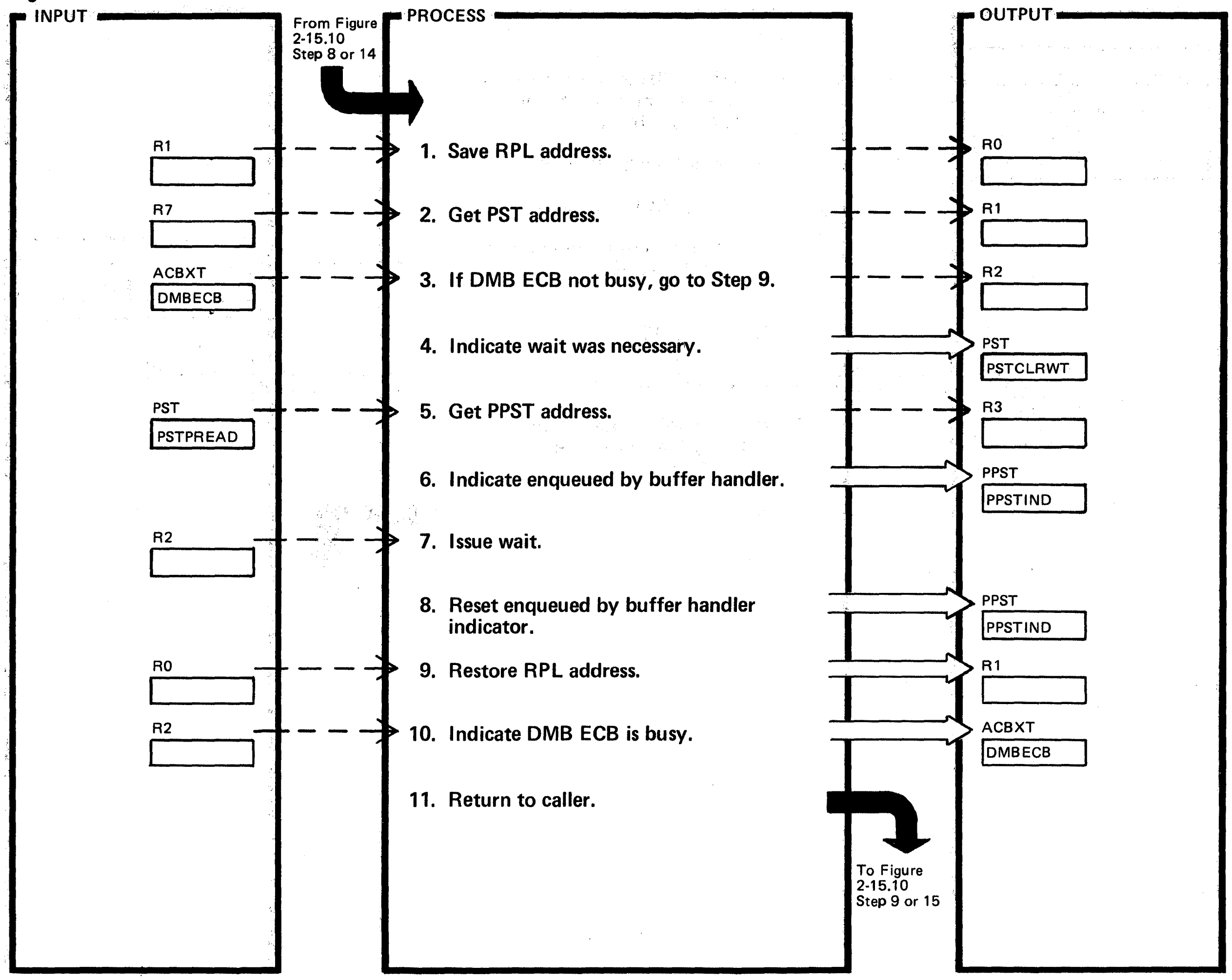
Data Base

DLZDBH00

Extended Description	Routine	Label
8. A task is a potential user of a buffer if at least one of the DSGs in the PSB has the same DMB and ACB as in fields BFFRDMB and BFFRACB of the buffer prefix.		

Extended Description	Routine	Label

Figure 2-15.12. Test ACB Routine



DLZDBH03 – Buffer Handler CSECT

DLZDBH00

Extended Description	Routine	Label
3. Byte 2 of DMB ECB set to X'80'	TESTACB	
4. Bit PSTIWAIT in field PSTCLRWT in PST set on.		
6. Bit PPSTBF in field PPSTIND in PPST set on.		
7. DLZIWAIT macro issued.		
8. Bit PPSTBF in field PPSTIND in PPST set off.		
9.		NOWAIT
10. X'80' in byte 2 of DMBECB turned off.		

Extended Description	Routine	Label

Figure 2-16. DB Logger (Overview)

INPUT	PROCESS	OUTPUT
	From Caller →	

PROCESS

Note: The three different functions of the logger are associated with the three different entry points into it:

- DLZIDBL0 (Step 2)
- LOGWR (Step 4)
- WRIAHEAD (Step 5)

1. Initialize logger. (See Figure 2-16.1)
2. Build a log record and move it to the log I/O area. (See Figure 2-16.2)
3. Give control to asynchronous log subtask (online only). (See Figure 2-16.3)
4. Move a log record (which has been built by another module) to the log I/O area. (See Figure 2-16.4)
5. Write log information physically to tape. (See Figure 2-16.5)
6. Close the log file. (See Figure 2-16.6)

Return to Caller

7. Check for disk errors. (See Figure 2-16.7)

To DLZBNUC0 ABEND Routine.

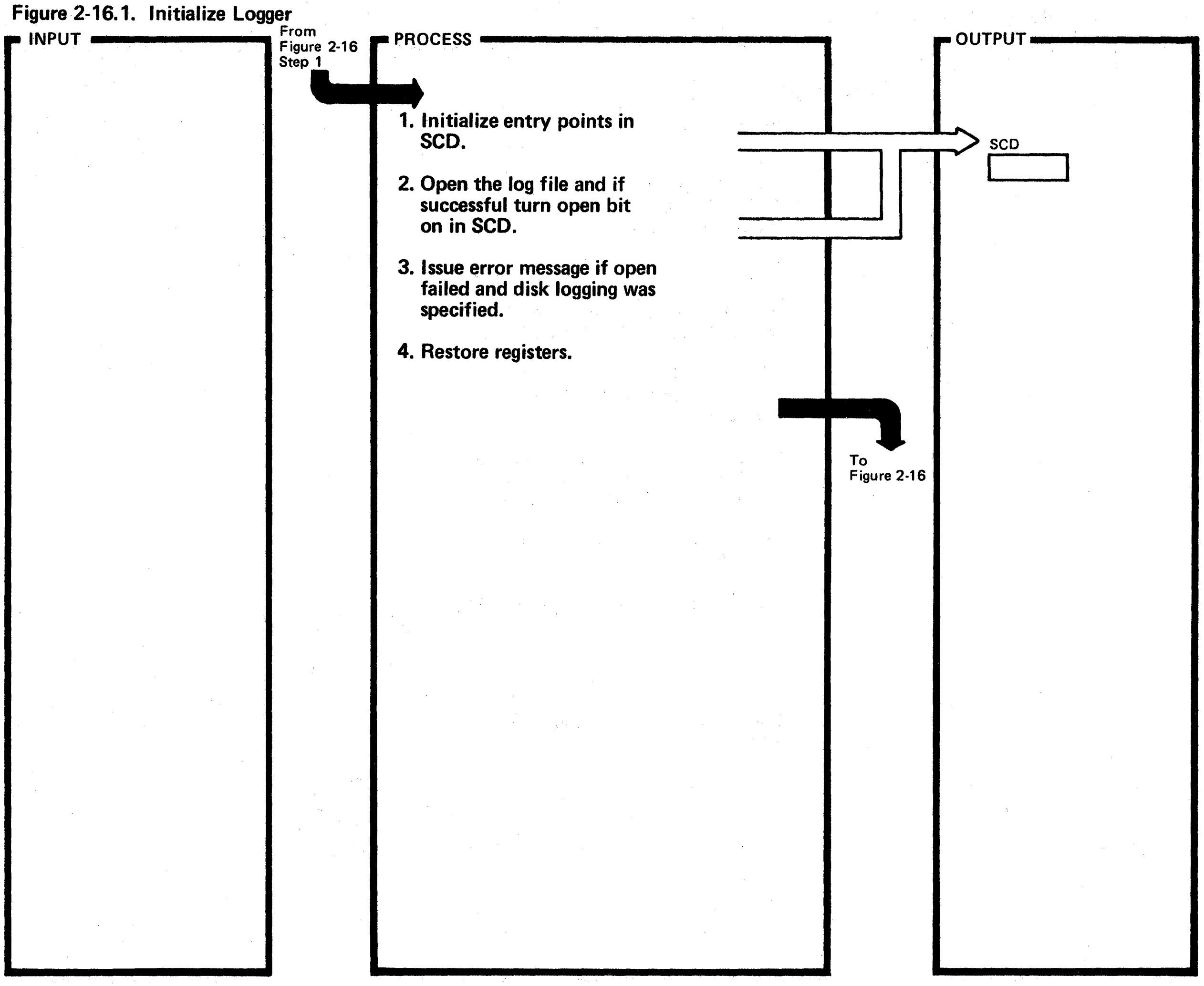
DLZRDBL0 – DB Logger CSECT

DLZRDBL0

Extended Description	Routine	Label
1.	DLZRDBL0	DLZRDBL0
2.		DLZIDBL0
3.		ONLINT ONLLOGWR
4.		LOGWR
5.		WRIAHEAD
6.		LOGCLOSE
7.		PUTERROR

Extended Description	Routine	Label

Figure 2-16.1. Initialize Logger



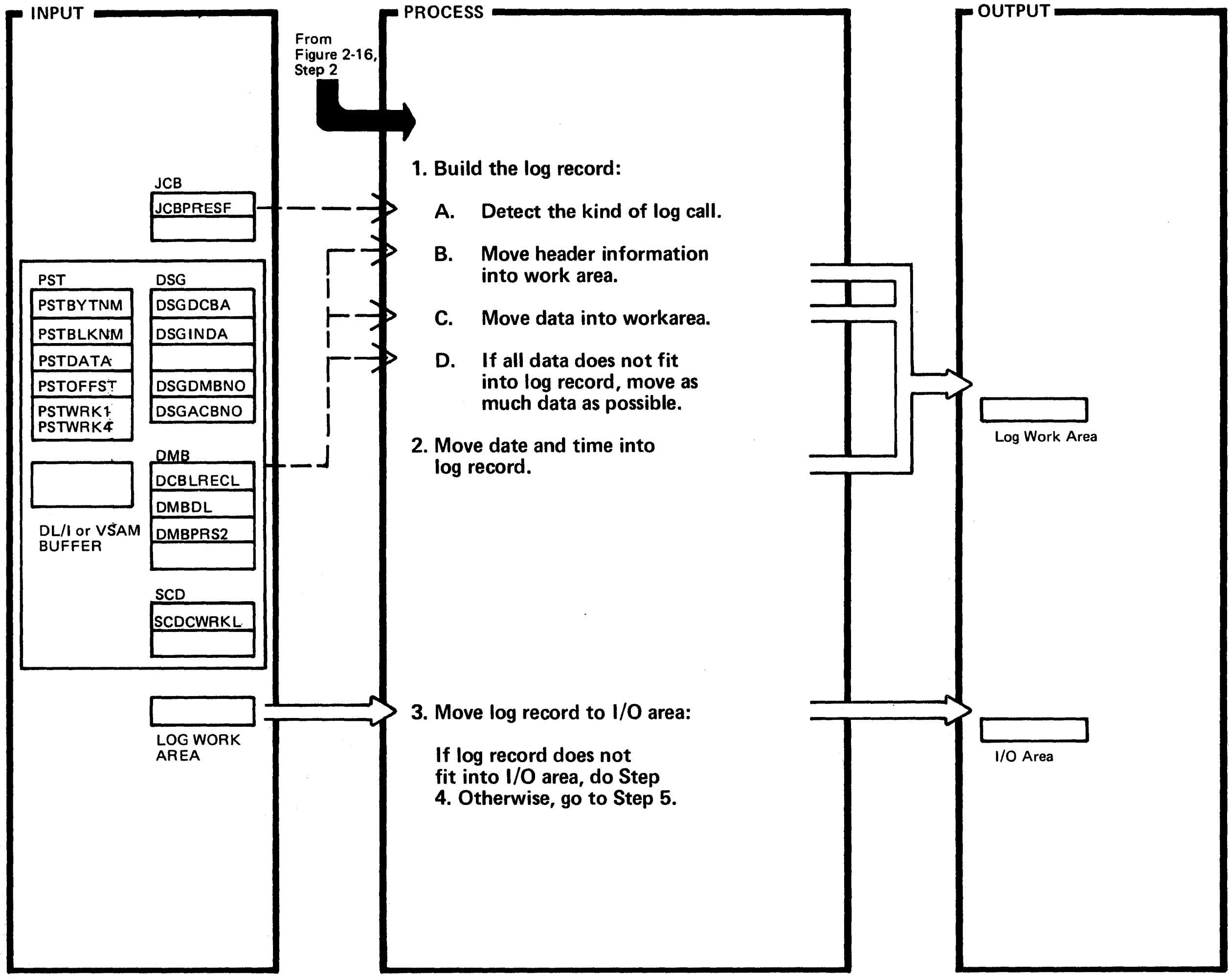
DLZRDBLO – DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
1. The entry point to the logger module initially points to the initialization routine. After initialization it contains the entry point of DLZIDBLO All of the entry points to the various logger routines are in the SCD after initialization.	DLZRDBLO	
2. If tape logging is specified, the DTF is opened. If disk logging is specified, the ACB is opened and tested if successful.		
3. Message DLZ020I is issued if an open error occurred with disk logging. Message DLZ077I is issued if the log was opened successfully with disk logging.		

Extended Description	Routine	Label

Figure 2-16.2. Build Log Record (Part 1 of 2)



DLZRDBLO — DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
1. Depending on the kind of log record being processed, the logger will build one of the following type of log record: ● Physical insert record ● Physical replace record ● Physical delete record ● Logical delete record ● Pointer maintenance record. The maximum logical record size for a log record is 512 bytes. The blocks are undefined with a maximum of 1024 bytes.	DLZRDBLO	DLZIDBLO

Extended Description	Routine	Label

Figure 2-16.2. Build Log Record (Part 2 of 2)

INPUT

I/O Area

SCD

SCDLOCOU

PROCESS

4. Write the contents of the I/O area to tape or disk.

   A. If logging is being done in a batch environment, issue the PUT macro immediately.

   B. If logging is being done in an online environment, pass control to:

   Asynchronous Log Subtask 2-16.3

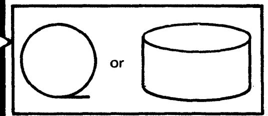
5. Check for disk errors if disk logging. See Figure 2-16.7.

6. If the log request is associated with a buffer, move the number of the last written log block from SCDLOCOU into the buffer prefix.

7. If more data is to be logged, return to Step 1C. Otherwise;

To Figure 2-16

OUTPUT



BFFR

BFFRLOCO

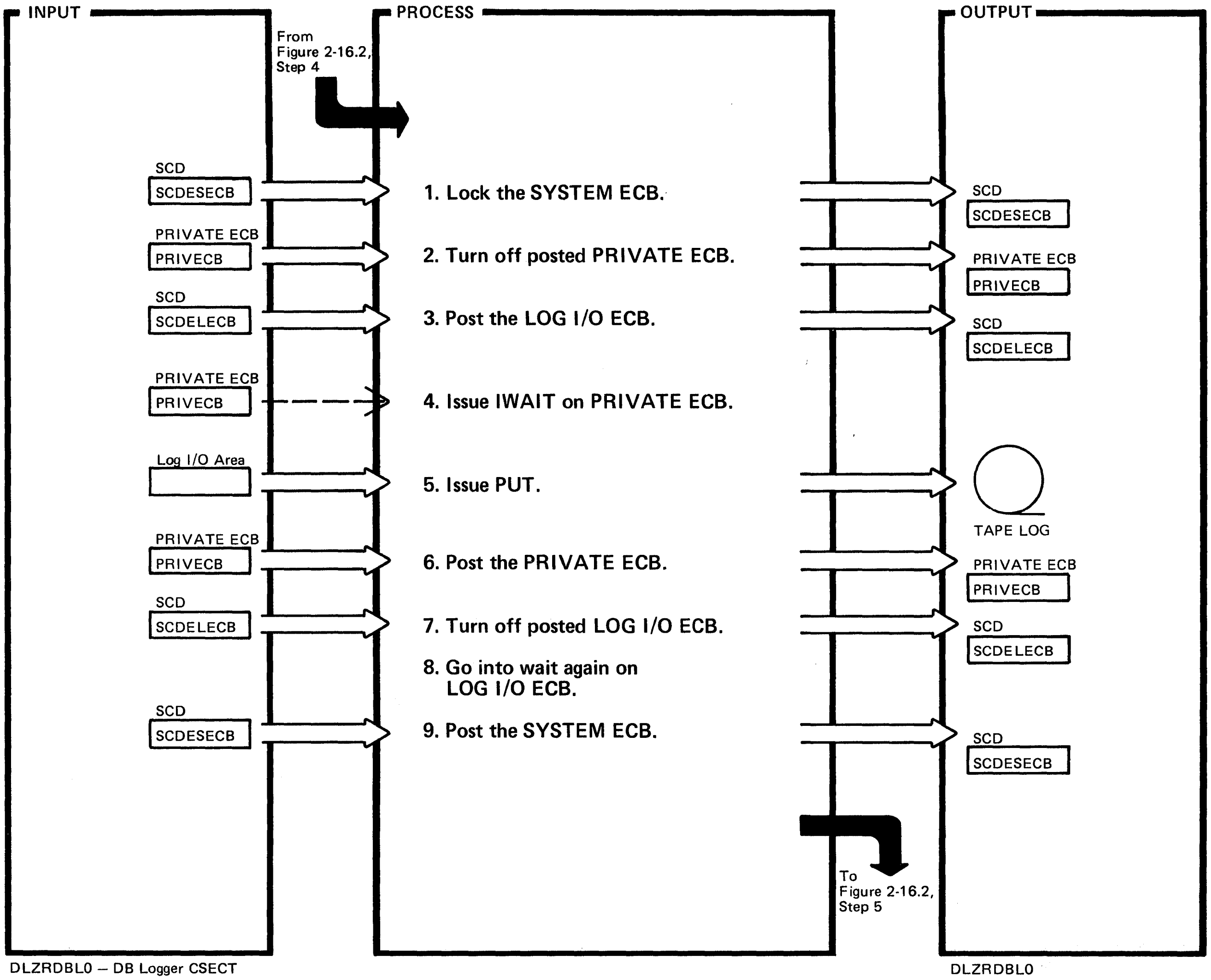
DLZRDBLO — DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
4 B. In an online environment, the PUT macro is issued from the Asynchronous Log Subtask in order to avoid losing tasks when EOV is encountered on the log tape.		
6. The purpose for keeping the number of the last written log block in the SCD and in the BFFR is to enable DLZDBH00 to determine if a log record has to be written out before an update is applied to a data base.		
7. There will be more data to be logged if all data did not fit into the log record. See Step 1D.		

Extended Description	Routine	Label

Figure 2-16.3. Asynchronous Log Subtask



Extended Description	Routine	Label
Steps 1, 2, 3, 4, and 9 are performed within CSECT DLZRDBLO.	DLZRDBLO	ONLINT
Steps 5, 6, 7, and 8 are performed within the Asynchronous Log Writer Subtask.	DLZRDBLO	ONLLOGWR
1. The SYSTEM ECB is used for communication between DLZRDBLO and DLZODP. It is locked in order to prevent any other task from entering the logger while the I/O is going on.		
2. The PRIVATE ECB is used for communication about the completion of I/O between the Asynchronous Log Subtask and DLZRDBLO.		
3. The LOG I/O ECB is used for communication about the need to issue a PUT macro between DLZRDBLO and the Asynchronous Log Subtask.		

Extended Description	Routine	Label
The Asynchronous Log Subtask is waiting on this ECB and when it gets posted, DOS/VS will mark this subtask as dispatchable.		
4. IWAIT will have the effect that the DL/I 'maintask' will be put into wait. The Asynchronous Log Subtask can then be started by DOS/VS.		

Figure 2-16.4. Move Log Record

INPUT

LOG RECORD
DLENGTH

LOG I/O AREA
CURSIZE

LOG I/O AREA

LOG WORK AREA
LOG RECORD

DLZRDBLO — DB Logger CSECT

PROCESS

From Figure 2-16, Step 4

(See Note)

1. If the log record fits into the space currently available in the LOG I/O AREA, go to Step 4.

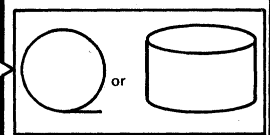
2. Write the current contents of the LOG I/O AREA to the log tape or disk.

3. Check for disk errors if disk logging. (See Figure 2-16.3.)

4. Move the log record to the LOG I/O AREA.

To Figure 2-16

OUTPUT



LOG I/O AREA

DLZRDBLO

Extended Description	Routine	Label
Note: This function is used • Open log records (ID X'2F') • Scheduling records (ID X'08') • Termination records (ID X'07') • Checkpoint records (ID X'41')	DLZRDBLO	LOGWR

Extended Description	Routine	Label

Figure 2-16.5. Write Log Information

INPUT

Log I/O Area

From Figure 2-16, Step 5

PROCESS

1. Issue PUT macro to write current contents of the log I/O area to the log tape or disk.

2. Check for disk errors if disk logging. (See Figure 2-16.7)

To Figure 2-16

OUTPUT

or

DLZRDBLO — DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
1. This function is used by DLZDBH00, when log information associated with a data base update has not been written to tape at the time the data base update was being done.	DLZRDBLO	WRIAHEAD

Extended Description	Routine	Label

Figure 2-16.6. Close Log File

INPUT

SCD

RPL

From Figure 2-16 Step 6

PROCESS

1. Reset open bit in SCD.
2. If tape logging, close tape DTF.
3. If disk logging, close ACB and test for close errors.
4. If close errors, issue error message.
5. Restore registers.

To Figure 2-16

OUTPUT

SCD

DTFMT

(LOGOUT)

ACB

(LOGACB)

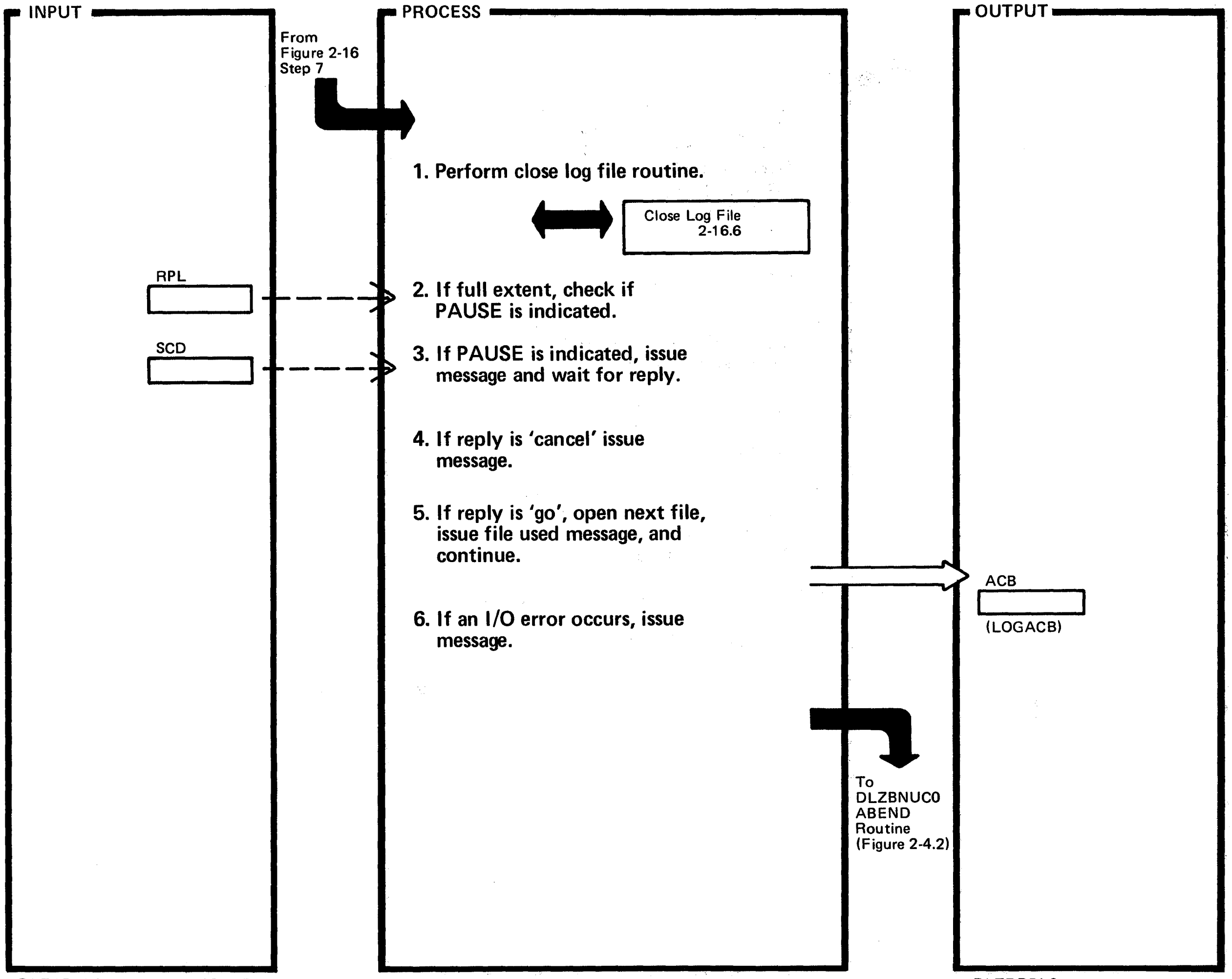
DLZRDBLO – DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
1. Either the tape or disk log file is closed in this subroutine. This subroutine is used by DLZODP, DLZRDBLO, DLZRRC00, and DLZBNUC0.	DLZIDBLO	LOGCLOSE
4. Message DLZ021I is issued.		

Extended Description	Routine	Label

Figure 2-16.7. Disk Errors



DLZRDBLO – DB Logger CSECT

DLZRDBLO

Extended Description	Routine	Label
1. The log file is closed so that the operator could dump the file (optional) before continuing.	DLZIDBLO	LOGCLOSE
2. Checks to see if the user specified PAUSE on the DL/I control parameter.	DLZRDBLO	PUTERROR
3. Message DLZ076I is issued.		
4. Message DLZ077I is issued.		
5. Message DLZ079I is issued.		
6. If the reply is 'GO', a check is made to determine if 1 or 2 disk files are being used for logging. If there are 2 files, the second file is opened and control is returned to the PUT routine.		
Message DLZ004I is issued.		

Extended Description	Routine	Label

Figure 2-17. CICS Journal Logger (Overview)

INPUT

PROCESS

From Caller

Note: The four different functions of the logger are associated with four different entry points into it:

- DLZIDBLO (Step 2)
- OPLOG (Step 3)
- WRITEEXT (Step 4)
- WRIAHEAD (Step 5)

1. Initialize the logger.
2. Build a log record and move it to the CICS journaling buffer. (See Figure 2-17.1)
3. Build open log record(s) and move to CICS journaling buffer. (See Figure 2-17.2)
4. Move a log record which has been built by another module to the CICS journaling buffer. (See Figure 2-17.3)
5. Write journal information physically to tape. (See Figure 2-17.4)

Return to Caller

OUTPUT

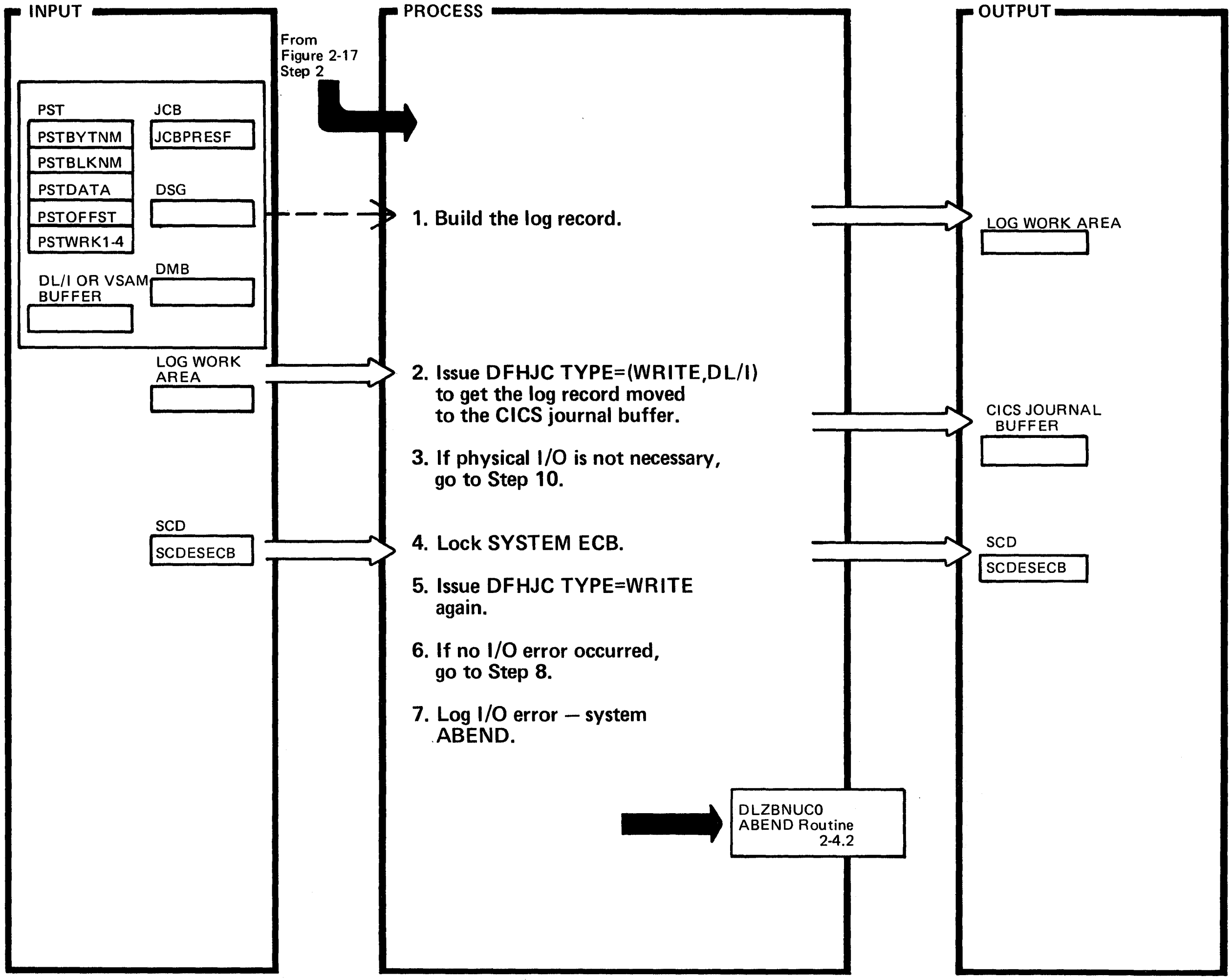
DLZRDBL1 — DB Logger with CICS Journaling CSECT.

DLZRDBL1

Extended Description	Routine	Label
1. Move all of the entry points to the logger into the SCD.	DLZRDBL1	DLZRDBLO
2.		DLZRDBLO
3.		OPLOG
4.		WRITEEXT
5.		WRIAHEAD

Extended Description	Routine	Label

Figure 2-17.1. CICS Build Log Record (Part 1 of 2)



DLZRDBLI — DB Logger with CICS Journaling CSECT

DLZRDBLI

Extended Description	Routine	Label
1.	DLZRDBLI	DLZRDBLO
4. The SYSTEM ECB is locked in order to prevent any other task from entering the logger while the I/O is going on.		IONEC1

Extended Description	Routine	Label

Figure 2-17.1. CICS Build Log Record (Part 2 of 2)

INPUT	PROCESS	OUTPUT
JCT: JCTBLKNM	8. Get the new number of the last written journal block from the JCT.	SCD: SCDLOCOU
	9. Post the SYSTEM ECB.	
	10. If the log request was not associated with a buffer, go to Step 12.	
SCD: SCDLOCOU	11. Store the block number of the last journal buffer written to tape into the buffer prefix.	BFFR: BFFRLOCO
	12. If the data to be logged did not fit into one log record, return to Step 1.	
	To Figure 2-17	

DLZRDBL1 — DB Logger with CICS Journaling CSECT

DLZRDBL1

Extended Description	Routine	Label
8. The purpose for keeping the CICS event control number is to enable DLZRBH00 to determine if a log buffer has to be written before an update is applied to a data base.	DLZRDBL1	GETECN

Extended Description	Routine	Label

Figure 2-17.2. CICS Move Log Record

INPUT

DDIR

DMB

DMB

DMBACBXT

From
Figure 2-17.3
Step 3

PROCESS

(See Note)

1. Locate the first DDIR entry.
2. If the data base was not opened, go to Step 5.
3. Build the open log record for the data base referenced by this DDIR entry.
4. Move open log record to CICS journaling buffer.
5. If this is not the last DDIR entry, get the next DDIR entry and return to Step 2.

To
Figure 2-17

OUTPUT

LOG WORK AREA

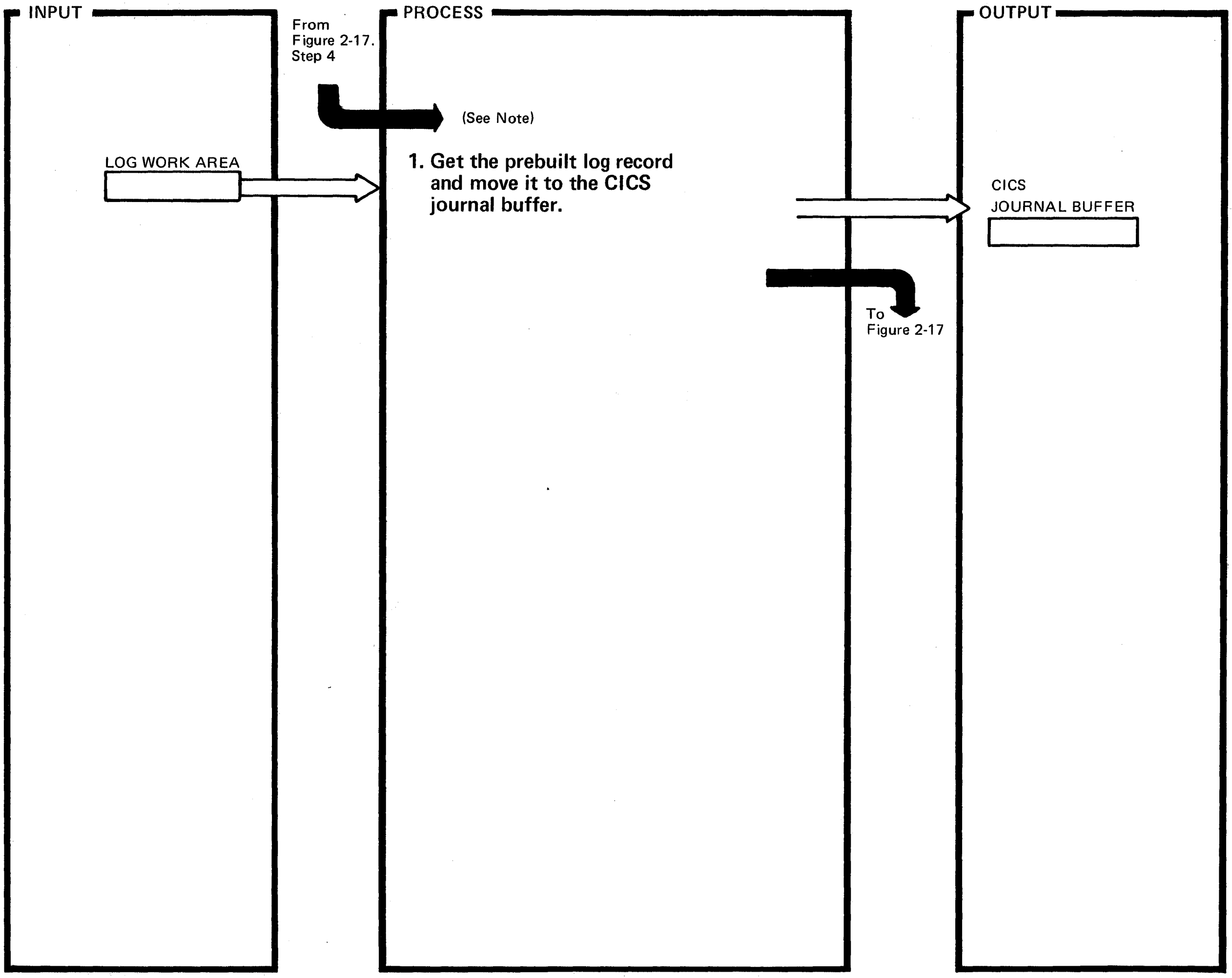
DLZRDBL1 – DB Logger with CICS Journaling CSECT

DLZRDBL1

Extended Description	Routine	Label
Note: Since the CICS journal tape is not yet open at DL/I initialization, the open log record(s) are built and moved before the first scheduling call is logged.	DLZRDBL1	OPLOG
2. A data base might not have been opened because of the OPEN= DEFERRED option or because of an open error.		
4. See Figure 2-17.1, Steps 4-9.		

Extended Description	Routine	Label

Figure 2-17.3. CICS Move Prebuilt Log Record



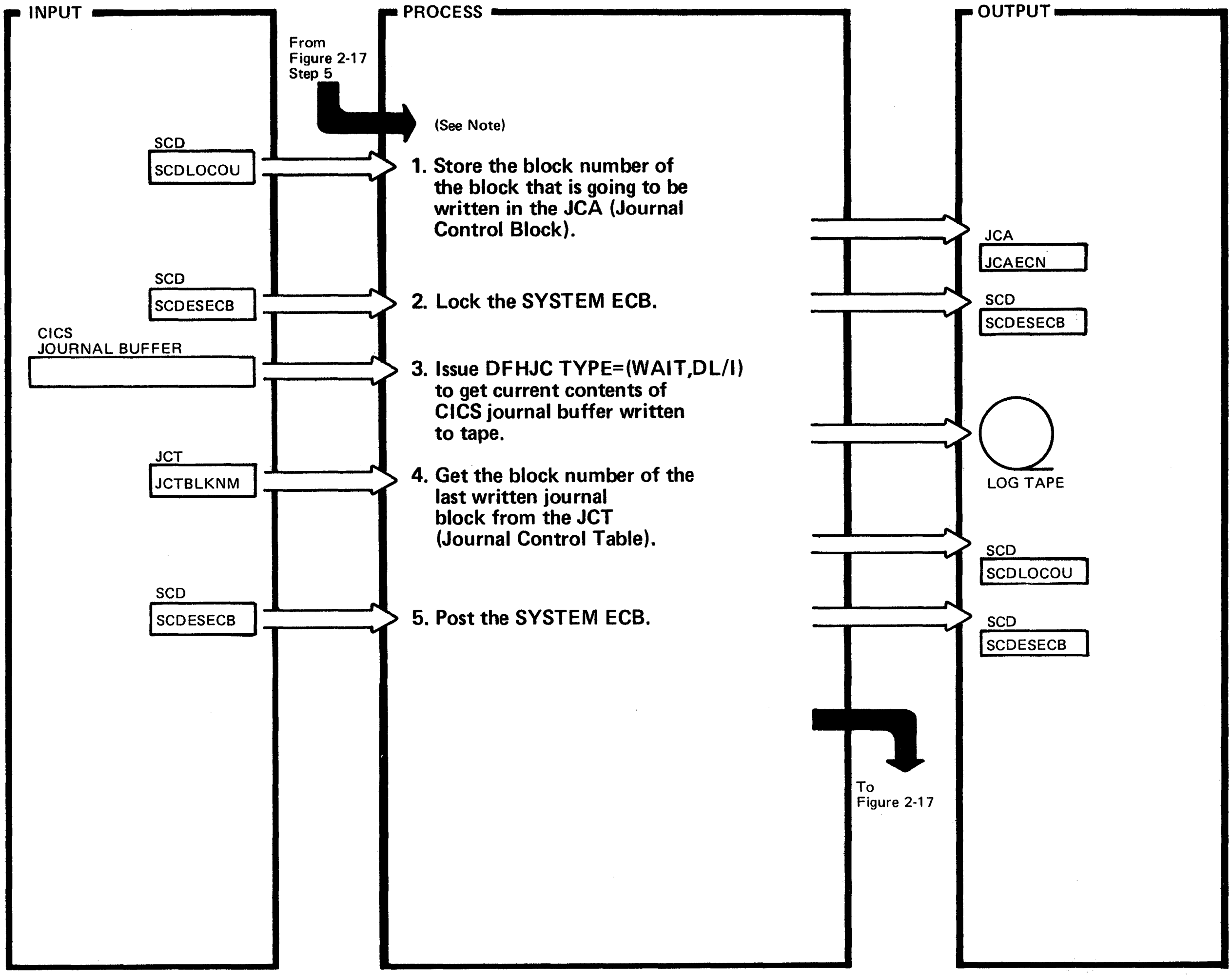
DLZRDBL1 — DB Logger with CICS Journaling CSECT

DLZRDBL1

Extended Description	Routine	Label
Note: This function applies to scheduling and termination records built by the scheduling termination routine. 1. See Figure 2-17.1, Steps 4-9.	DLZRDBL1	WRITEEXT

Extended Description	Routine	Label

Figure 2-17.4. CICS Log Writing



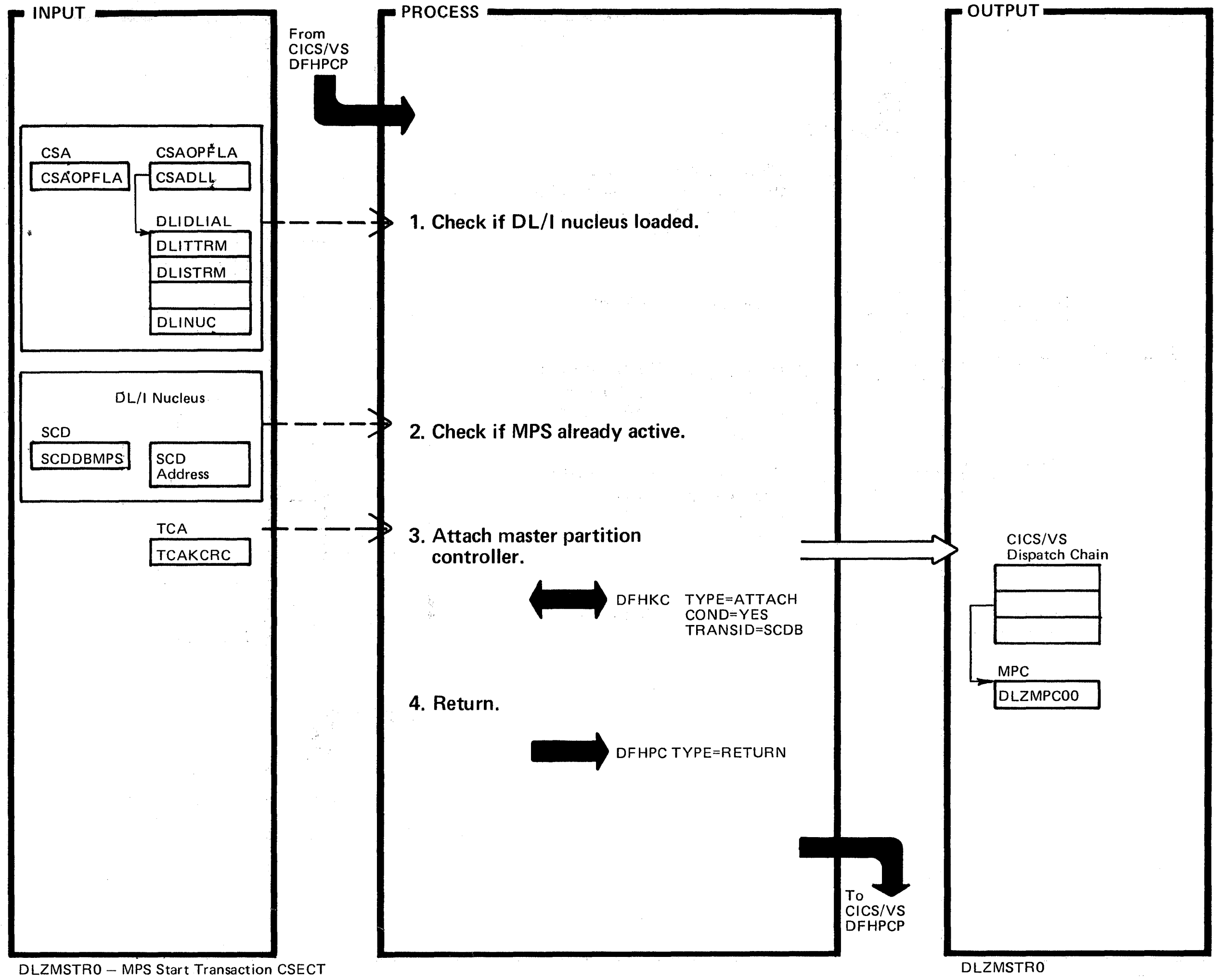
DLZRDBL1 — DB Logger with CICS Journaling CSECT

DLZRDBL1

Extended Description	Routine	Label
Note: This function is used by DLZDBH00 when log information associated with a data base update was not written to tape when the data base update was being done. 1. Refer to note for Step 8 of Figure 2-17.1. 2. Refer to note for Step 4 of Figure 2-17.1.	DLZRDBL1	WRIAHEAD

Extended Description	Routine	Label

Figure 2-18. Start Transaction



Extended Description	Routine	Label
1. Module identifier (DLZMSTRO) is defined here. Write message DLZ097I if nucleus not loaded or not active and go to Step 4.	DLZMSTRO	DLZMSTRO
2. Write message DLZ101I if flag SCDXECB indicates MPS XECBs already defined and go to Step 4.		
3. Write message DLZ083I if attach fails with a return code in TCAKCRC and go to Step 4.		
4.		RETURN

Extended Description	Routine	Label

Figure 2-19. Master Partition Controller (Overview)

INPUT

From CICS/VS DFHPCP

PROCESS

1. Initialize MPC task.
   (See Figure 2-19.1).
2. Defined required XECBs for cross partition control.
   (See Figure 2-19.2)
3. WAITM on Start BPC, Stop Partition, ABEND, and Stop Transaction XECBs.
   (See Figure 2-19.3)
4. If Start Partition XECB posted, attach BPC task.
   (See Figure 2-19.4)
5. If Stop Partition XECB posted, process stop partition requests.
   (See Figure 2-19.5)
6. If ABEND XECB posted, the BPC attach failed, process the ABEND condition.
   (See Figure 2-19.6)
7. If Stop Transaction posted, terminate MPS.
   (See Figure 2-19.7)
8. Terminate MPC task.
   (See Figure 2-19.8)

To CICS/VS DFHPCP

From CICS/VS

9. MPC ABEND exit routine.
   (See Figure 2-19.9)

To CICS/VS DFHPCP

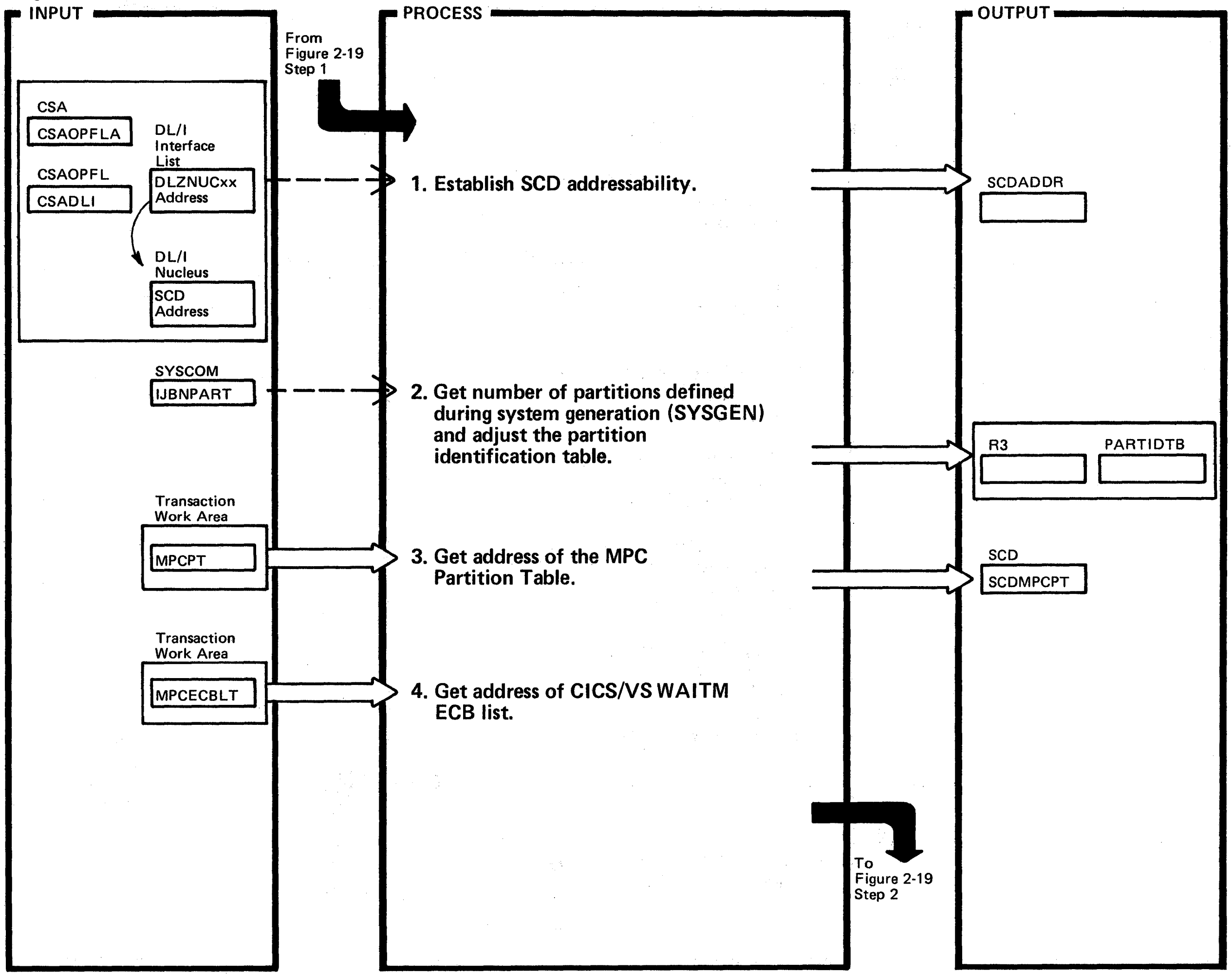
OUTPUT

DLZMPC00

Extended Description	Routine	Label
1.		DLZMPC00 MPCSTART
2.		MPCDEFIN
3.		MPCWAIT
4.		MPCSTRP
5.		MPCSTOP
6.		MPCABNP
7.		MPCSTRN
8.		MPCEXIT
9.		MPCABEXT

Extended Description	Routine	Label

Figure 2-19.1. MPC Task Initialization



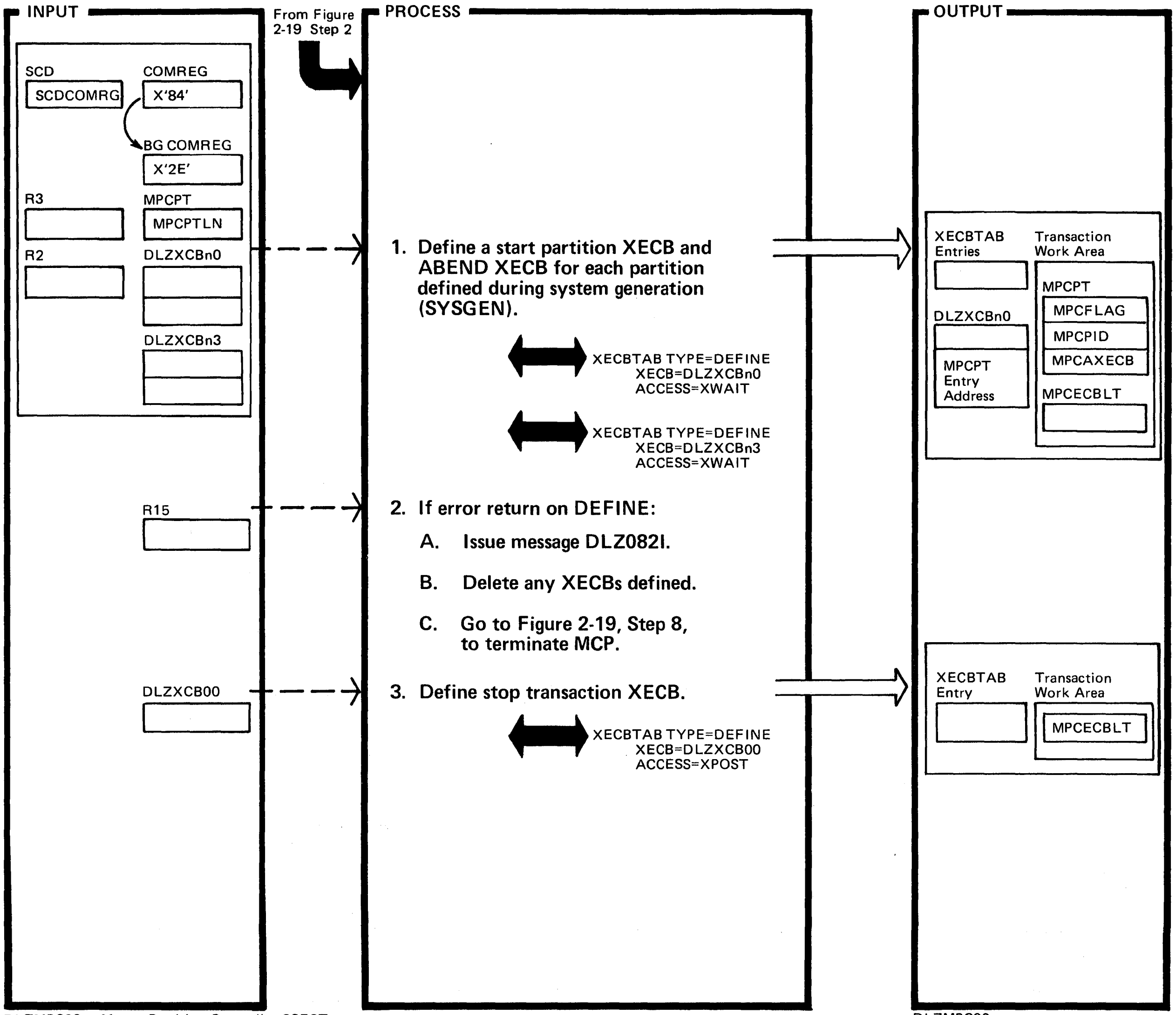
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. Module identifier (DLZMPC00 vrnp) is defined here. MPC is attached by the MPS start transaction (DLZMSTR0) via CICS/VS. Ignore request if DL/I is not defined to CICS/VS or the nucleus is not loaded.	DLZMPC00	DLZMPC00 MPCSTART
2. The number of partitions is saved in R3 to be used later when defining XECBTAB entries for partitions. The PARTIDTB table is used for MPC messages. Write message DLZ088I followed by DLZ094I if only one partition was defined during SYSGEN. In all steps where a message is issued: • R1 is set up with the applicable message parameter list. • Control is passed to the MPC message writer at MPCMSGRT. • BALR to the DL/I online message module, DLZERMSG, to write the message.		
3. The transaction work area is a logical extension of the TCA.		

Extended Description	Routine	Label

Figure 2-19.2. MPC Define XECBs (Part 1 of 2)



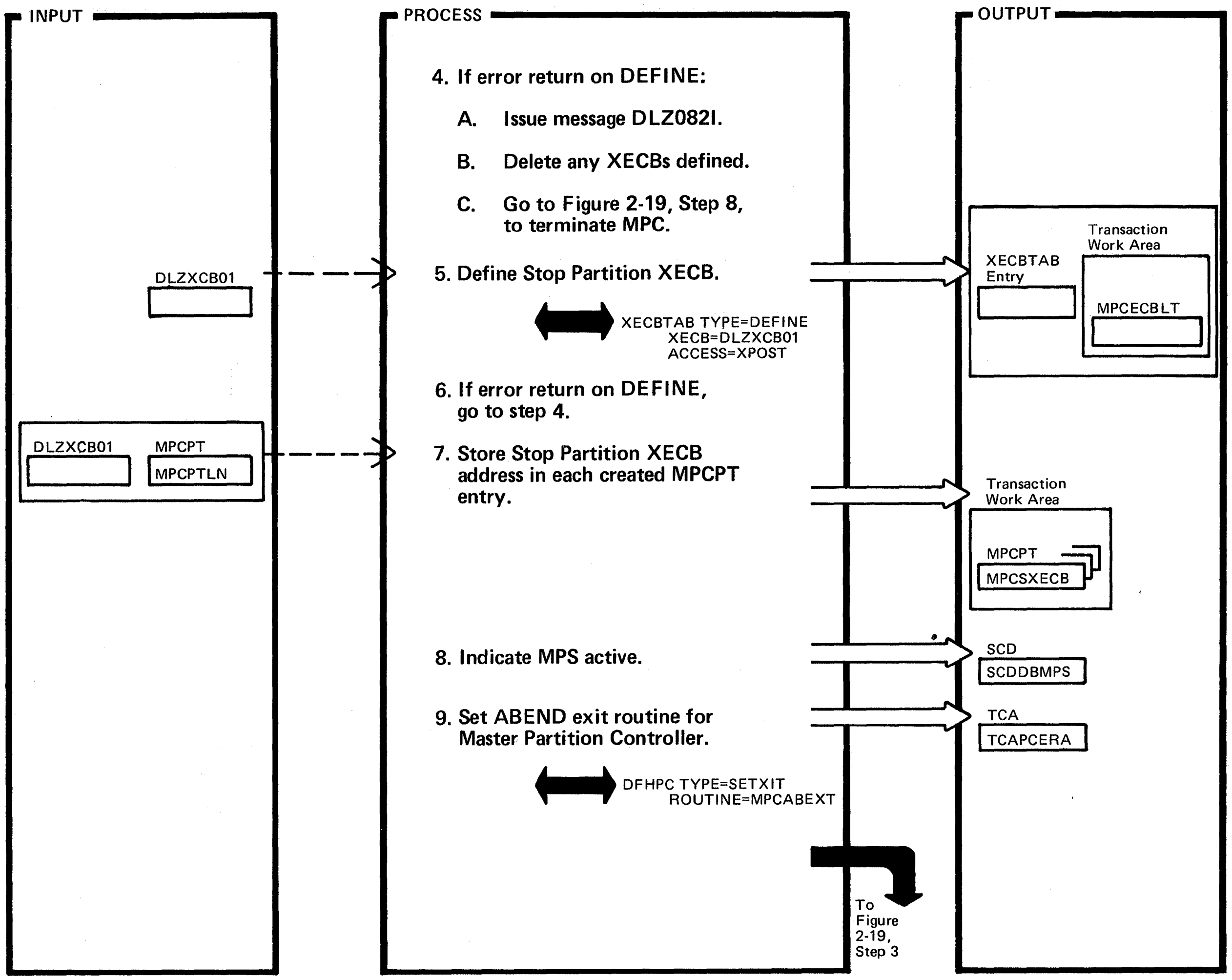
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. The XECBTAB/DEFINE macro is issued to initialize the XECBTAB table with entries representing a partition with a unique name. DLZXCBn0 is the XECB name to start a batch partition controller for a specific partition. DLZXCBn3 is the XECB name for handling an ABEND situation for a specific partition. The n varies and represents the 1st character of the partition identification key (PIK) in character form.	DLZMPC00	MPCDEFIN XECBSTR XECBABN
R3 being the number of partitions defined during SYSGEN lends itself to this application of naming an XECB for a partition.		
For example: If two partitions are defined: R3=2 DLZXCB20 is identical to F2F0=F1 PIK R3=1 DLZXCB10 is identical to F1F0=BG PIK		

Extended Description	Routine	Label
If four partitions are defined: R3=4 DLZXCB40 is identical to F4F0=F1 PIK R3=3 DLZXCB30 is identical to F3F0=F2 PIK R3=2 DLZXCB20 is identical to F2F0=F3 PIK R3=1 DLZXCB10 is identical to F1F0=BG PIK		
XECBTAB is a table in the supervisor containing the name and address of all XECBs defined. XTABD is the DSECT representing an XECBTAB entry.		
The CICS/VS WAITM ECB list (MPCECBLT) is initialized with fullword pointers to the partition start XECBs.		
3. DLZXCB00 is the XECB name to stop the MPS transaction. The two stop XECBs are defined as XPOST so that they can be posted during abnormal system termination. The CICS WAITM ECB list is initialized with the STOP XECB pointers.		XECBFN0

Figure 2-19.2. MPC Define XECBs (Part 2 of 2)



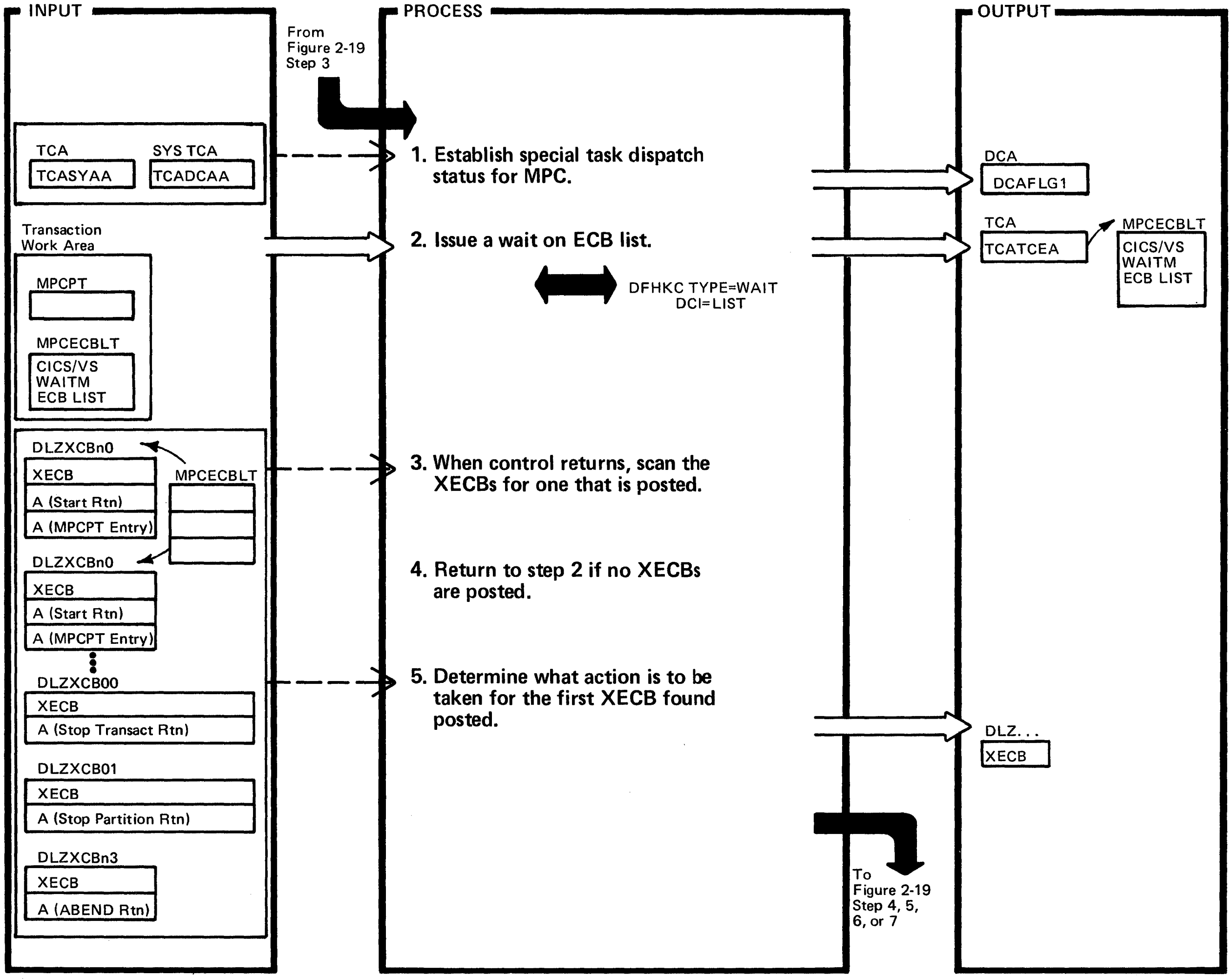
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
5. DLZXCB01 is the XECB name to stop a partition.		XECBDFN1
7.		MPCUPLST
8. Turn on SCDXECB at SCDDBMPS and issue message DLZ093I to indicate MPS started.		
9. MPCABEXT routine is within this module (see Figure 2-19.9).		

Extended Description	Routine	Label

Figure 2-19.3. MPC Wait



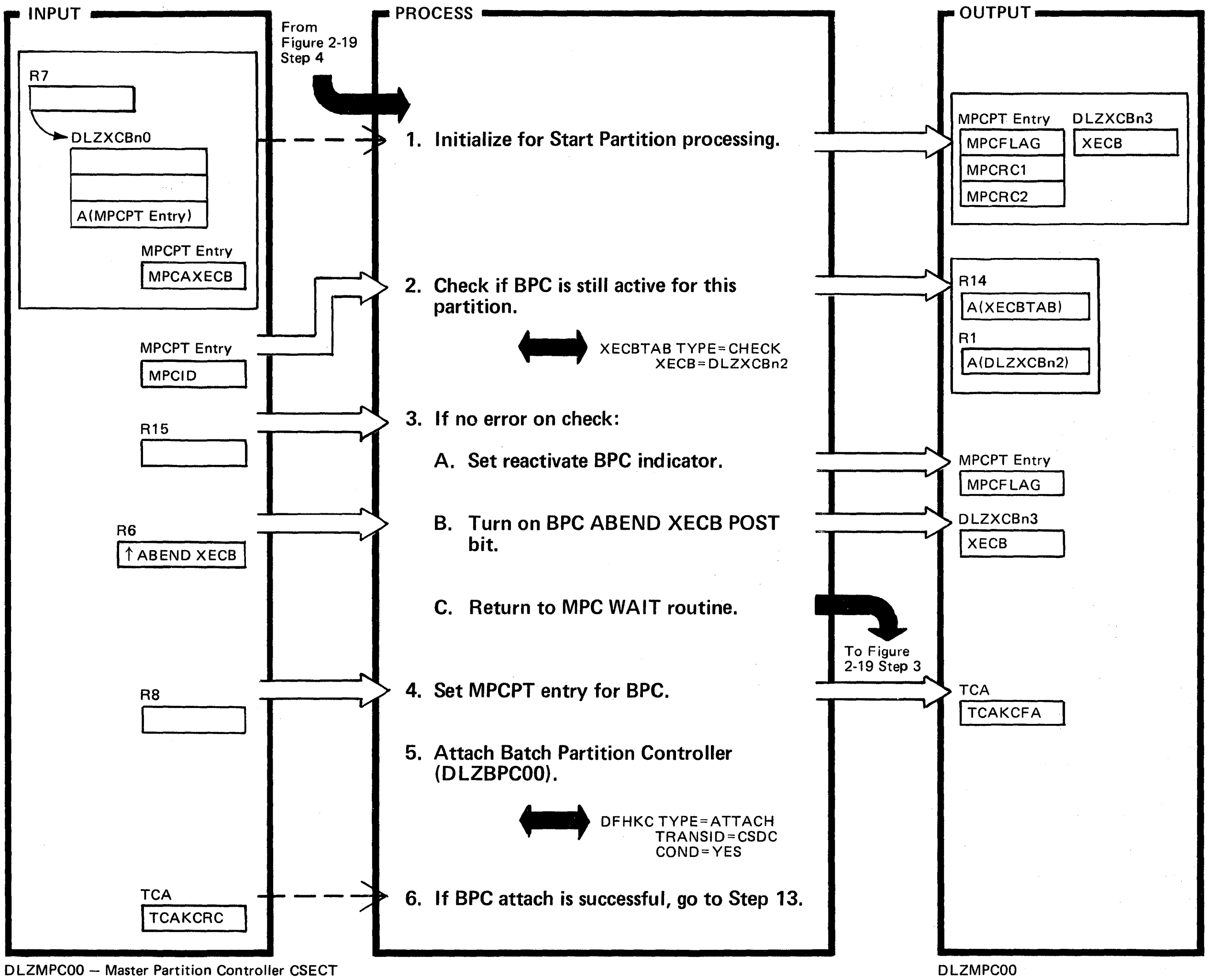
DLZMPC00 – Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. Turn on the DCAAPURG flag in the DCAFLG1 byte of the DCA for this task. When on CICS/VS task control will not count this task as part of AMXT nor will it take the short wait interval if this is the only waiting task in the CICS/VS system.	DLZMPC00	MPCWAIT
2. Note that the ABEND XECB (DLZXCBn3) pointer is placed in the ECB list only when the BPC attach is unsuccessful.		
3. The XECBs are posted on the following conditions: DLZXCBn0 • DLZMPI00 – activate BPC for a specific partition. DLZXCB00 • DLZMSTP0 – terminate MPS. DLZXCB01 • DLZBPC00 – normal batch EOJ; error conditions in BPC or batch partitions. • DLZODP01 – ABEND.		MPCECBCK

Extended Description	Routine	Label
DLZXCBn3 • DLZMPI00 – BPC attach failure.		
5. Before going to the appropriate routine, the post bit in the XECB is turned off.		MPCECBOK

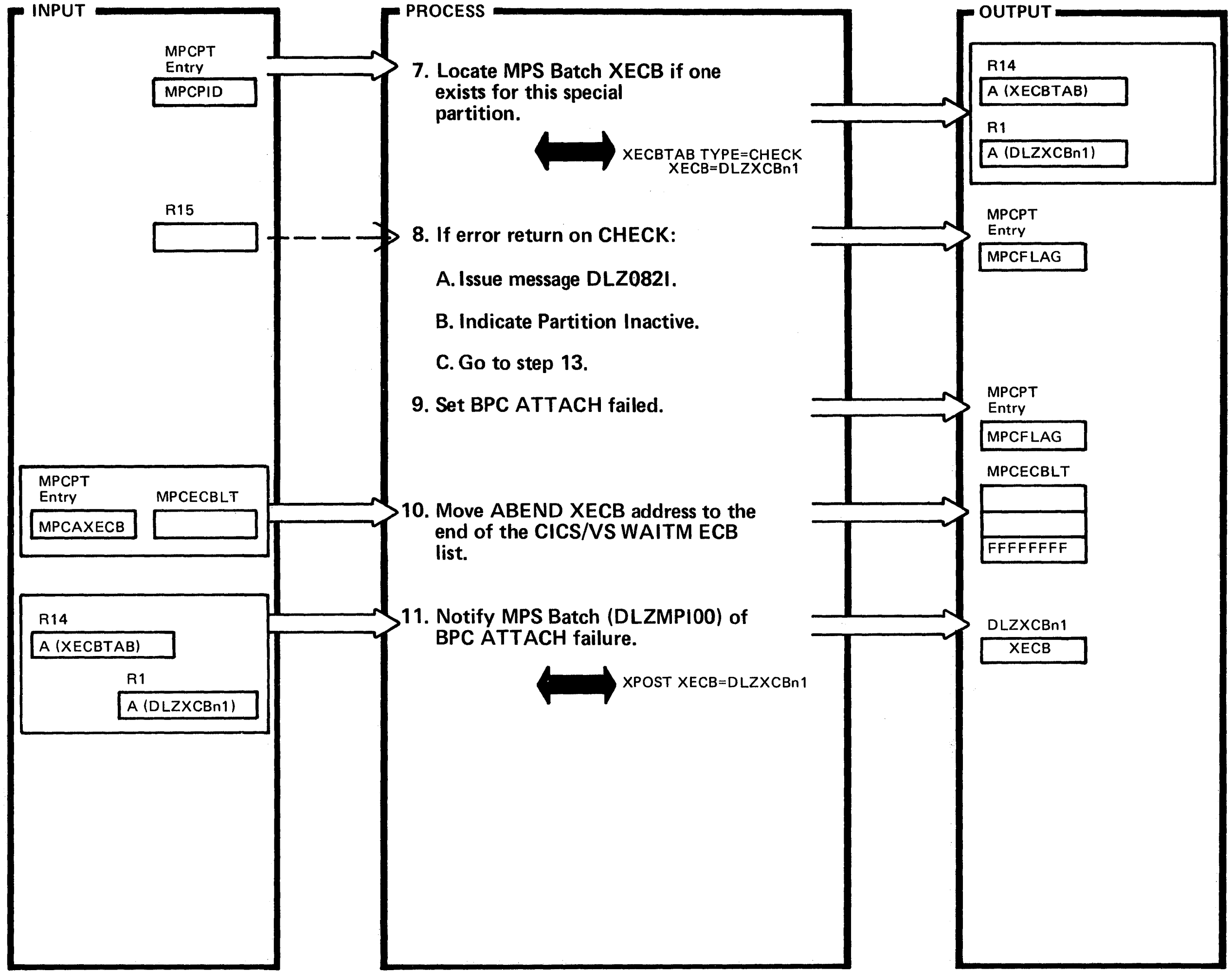
Figure 2-19.4 MPC Start Processing (Part 1 of 3)



Extended Description	Routine	Label
1. This routine is entered from the MPC Wait Routine when a Start Partition XECB (DLZXCBn0) is posted (XPOST) by DL/I MPS Batch Module (DLZMPI00). Register 7 contains the address of the XECB posted.		MPCSTRP
2. The XECBTAB/CHECK macro is used to determine if a BPC is still active for this partition.		XECBBPC
3. A zero return code from XECBTAB/CHECK macro in R15 indicates an active BPC. A. Bit MPCREBPC in field MPCFLAG is turned on. B. R6 contains a pointer to BPC ABEND XECB (DLZXCBn3). Note that the XPOST macro is not needed to turn on the POST bit because the BPC ABEND XECB (DLZXCBn3) is defined by MPC.		
6. A 'X31' in TCAKCRC indicates an ATTACH failure.		

Extended Description	Routine	Label

Figure 2-19.4. MPC Start Processing (Part 2 of 3)



DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
7. The XECBTAB/CHECK macro is issued to obtain the address (in register 14) of Batch Initialization's XECBTAB entry for the specific partition.		XECBCHK
8. B. The partition active flag, MPCPACT, at MPCFLAG is turned off.		
9. The BPC ATTACH failed flag, MPCERR, is turned on.		MPCCHKOK
10. The CICS/VS WAITM ECB list is updated to include a pointer to the ABEND XECB pointer to provide recovery.		MPCXPOST

Extended Description	Routine	Label

Figure 2-19.4. MPC Start Processing (Part 3 of 3)

INPUT

R15

PROCESS

12. If error return on XPOST:
    - A. Issue message DLZ084I.
    - B. Set partition inactive and clear error indicator.
    - C. Remove ABEND XECB pointer from CICS/VS WAITM ECB list.
13. Return to MPC Wait routine.

To Figure 2-19 Step 3

OUTPUT

MPCPT Entry

MPCFLAG

MPCECBLT

FFFFFFFF

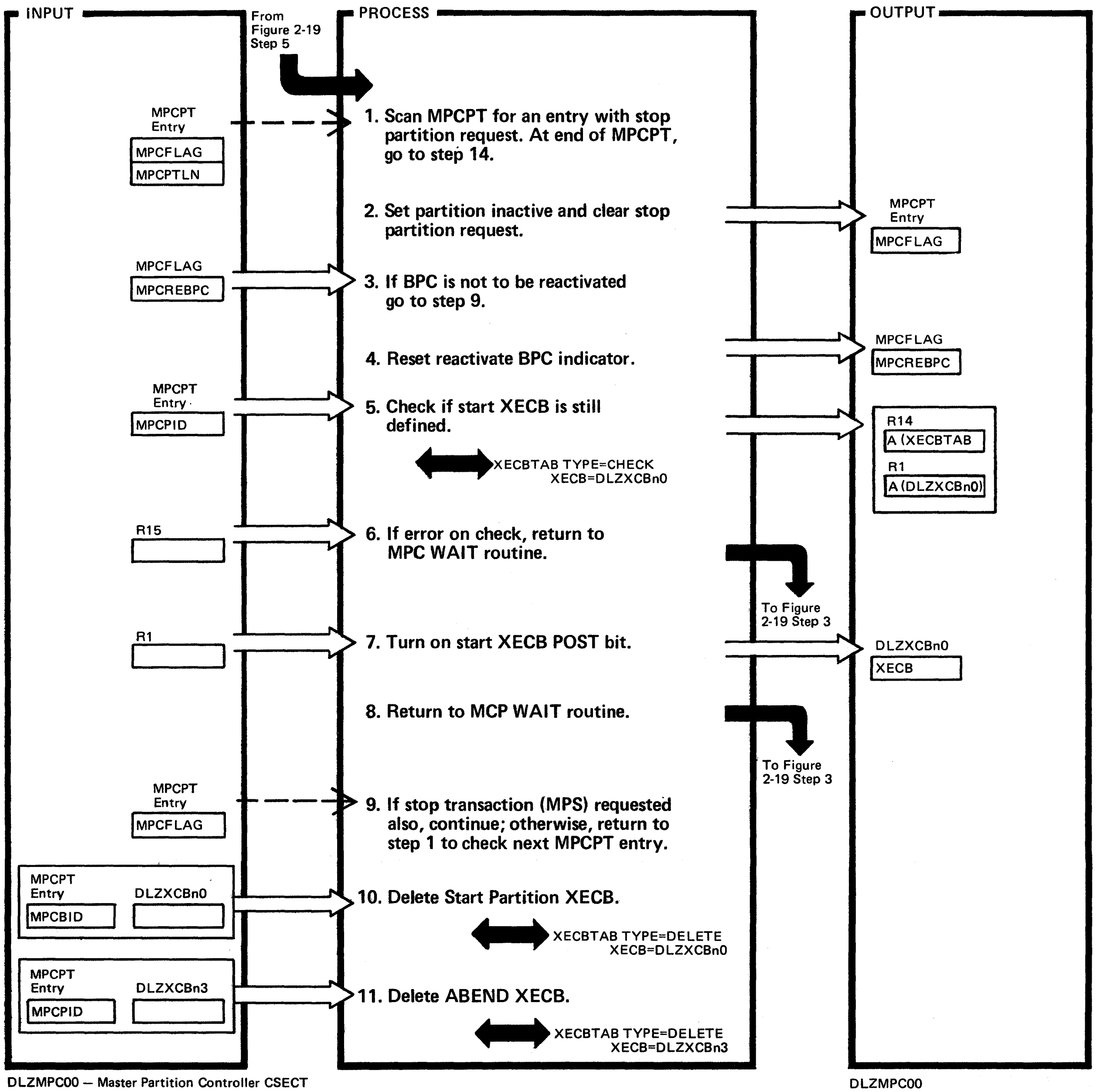
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
12. B. The flags MPCPACT and MPCERR are turned off.		

Extended Description	Routine	Label

Figure 2-19.5. MPC Stop Partition Processing (Part 1 of 2)



Extended Description	Routine	Label
1. This routine is entered from the MPC Wait routine when a Stop Partition XECB (DLZXCB01) is posted (XPOST) by DL/I MPS Batch Partition Controller (DLZBPC00) or Task Termination (DLZODP01). A scan is done on every entry in the partition table to avoid losing a stop partition request on a double post.		MPCSTOP
3. Bit MPCREBPC in field MPCFLAG not on.		
4. Bit MPCREBPC in field MPCFLAG turned off.		

Extended Description	Routine	Label
5. The XECBTAB/CHECK macro is issued to determine if the start XECB (DLZXCBn0) is still defined.		XECBSTRC
7. R1 contains pointer to the start XECB (DLZXCBn0). The XPOST macro is not needed to turn on the POST bit because the start XECB (DLZXCBn0) is defined by MCP.		
10. If error return on DELETE, issue message DLZ082I.		XECBDELS
11. If error return on DELETE, issue message DLZ082I.		XECBDELA

Figure 2-19.5. MPC Stop Partition Processing (Part 2 of 2)

INPUT	PROCESS	OUTPUT
	12. Set XECBs deleted for this partition.	MPCPT Entry MPCFLAG
	13. Return to step 1 to check next MPCPT entry.	
MPCPT Entry MPCFLAG	14. If stop transaction (MPS) has not been requested, go to step 16.	
MPCPT Entry MPCFLAG	15. If all the partitions are inactive, go to Figure 2-19, Step 8 to terminate MPC.	
	16. Return to MPC Wait routine.	
	To Figure 2-19 Step 3	

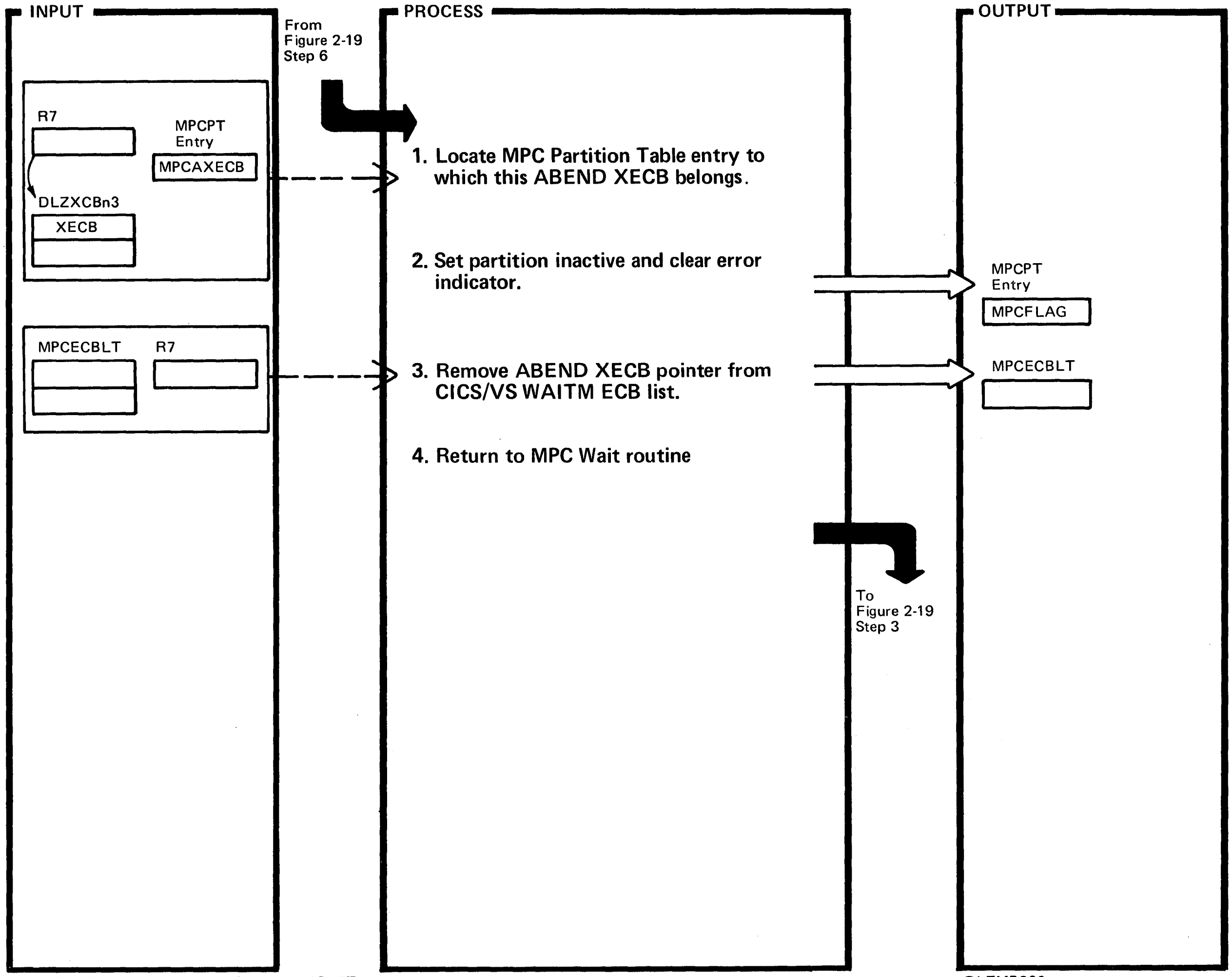
DLZMPC00 – Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
15. Flag MPCPACT in MPCFLAT indicates whether the partition is active or inactive.		

Extended Description	Routine	Label

Figure 2-19.6. MPC ABEND Processing



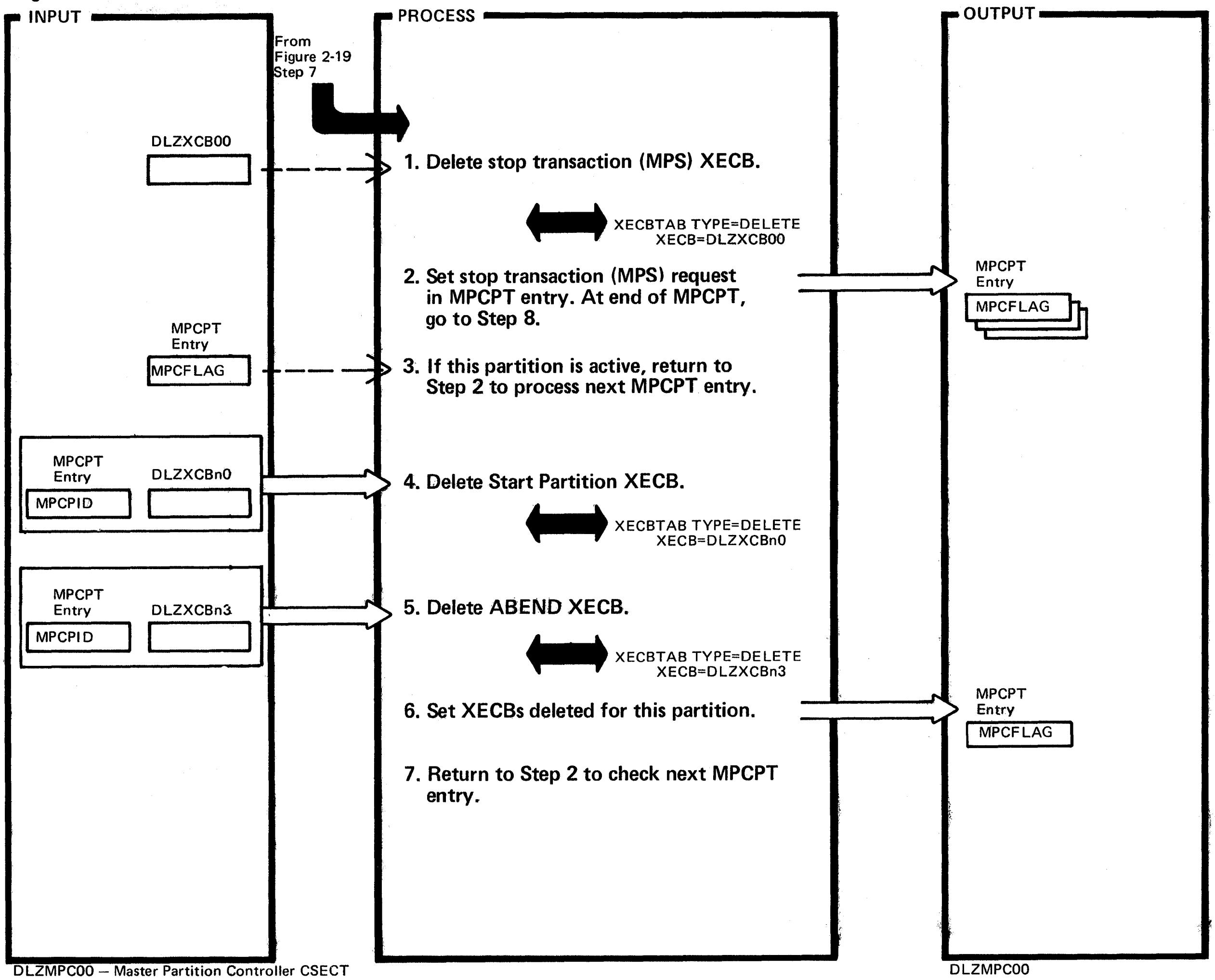
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. This routine is entered from the MPC Wait routine when an ABEND XECB (DLZXCBn3) is posted (XPOST) by DL/I MPS Batch Initialization Module (DLZMPI00) on a BPC ATTACH Failure. Register 7 contains the address of the XECB posted. 2. Flags MPCPACT and MPCERR are turned off.		MPCABNP

Extended Description	Routine	Label

Figure 2-19.7. MPS Termination (Part 1 of 2)



Extended Description	Routine	Label
1. This routine is entered from the MPC Wait routine when the Stop Transaction (MPS) XECB (DLZXCB00) is posted by DL/I Stop Transaction Task (DLZMSTP0). If error return on DELETE, issue message DLZ082I.		MPCSTRN
2. Flag MPCTSTP is turned on.		MPCSXCK
3. The Start/ABEND XECBs are deleted for inactive partitions (flag MPCPACT off) unless they have already been deleted (flag MPCXECB on).		
4. If error return on DELETE, issue message DLZ082I.		XECBSDEL
5. If error return on DELETE, issue message DLZ082I.		XECBADEL
6. Flag MPCXECB is turned on.		MPCSTRN3
7.		MPCDELUP

Extended Description	Routine	Label

Figure 2-19.7. MPS Termination (Part 2 of 2)

INPUT

MPCPT Entry
MPCFLAG

MPCPT Entry
MPCPID

R15

R6

PROCESS

8. Scan MPCPT entries for an active partition. Go to Step 10 if no active partition is found.

9. If an active partition is found:

   A. Check if XECB for batch partition still defined.

      XECBTAB TYPE=CHECK
      XECB=DLZXCBn1

   B. If batch partition still active, go to Step 9F.

   C. Set cancel BPC indicator in MPCPT.

   D. Turn on BPC ABEND XECB POST bit.

   E. Go to Step 9G.

   F. Issue message DLZ086I.

   G. Return to MPC Wait routine

      To Figure 2-19 Step 3

10. Go to MPC Termination.

      To Figure 2-19 Step 8

OUTPUT

R14
A(XECBTAB)

R1
A(DLZXCBn1)

MPCFLAG1
MPCCNBPC

DLZXCBn3
XECB

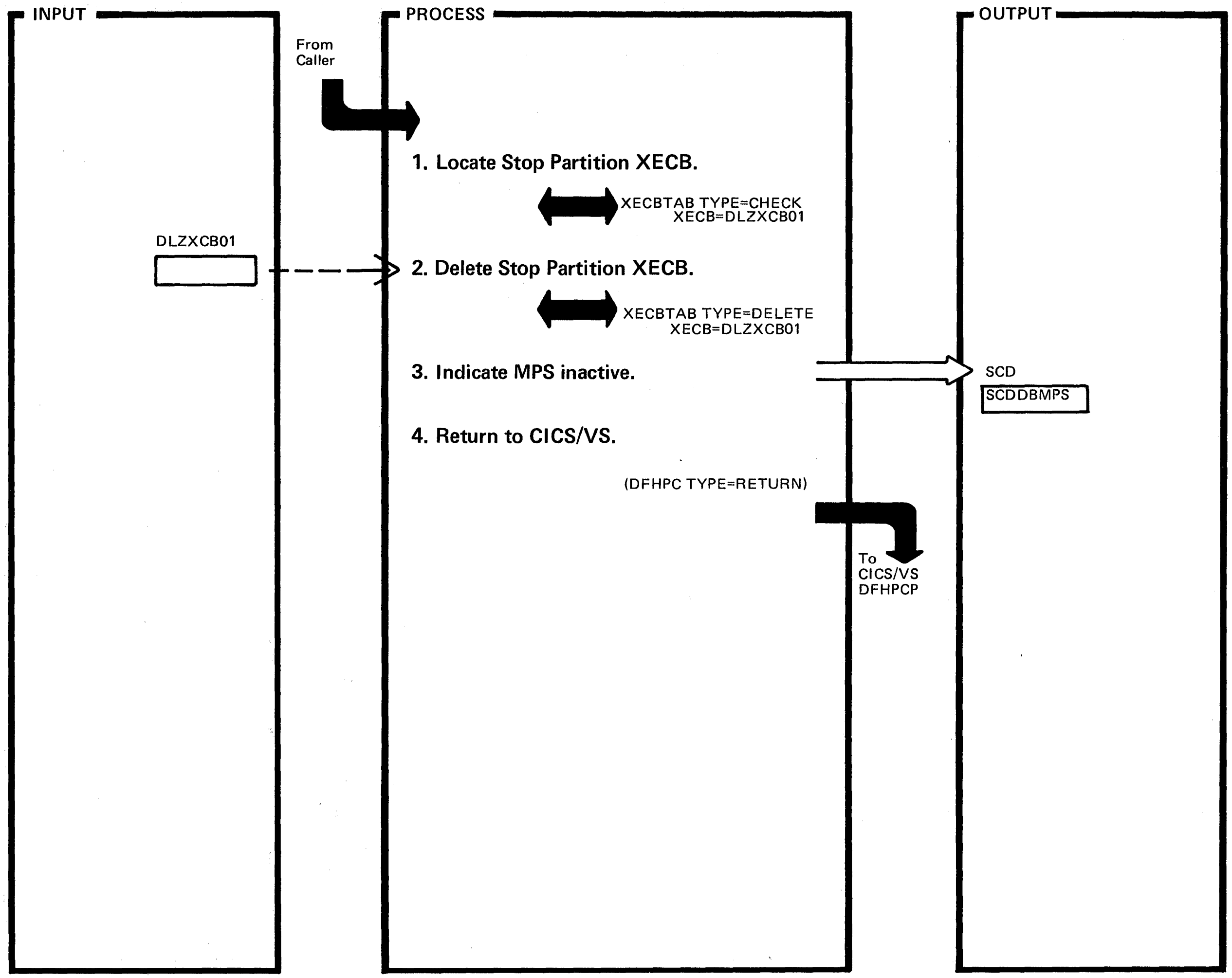
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
8. A partition is active if MPCPACT is on.		MPCDELCP MPCSXEND
9. A. The XECBTAB/CHECK macro is is issued to determine if the batch partition is still defined.		XECBATCH
C. Bit MPCCNBPC in field MPCFLAG1 in the MPCPT is set on.		
D. R6 contains pointer to the BPC ABEND XECB (DLZXCBn3). The XPOST macro is not needed to turn on the POST bit because the ABEND XECB is defined by MPC.		

Extended Description	Routine	Label

Figure 2-19.8. MPC Stop Transaction Processing



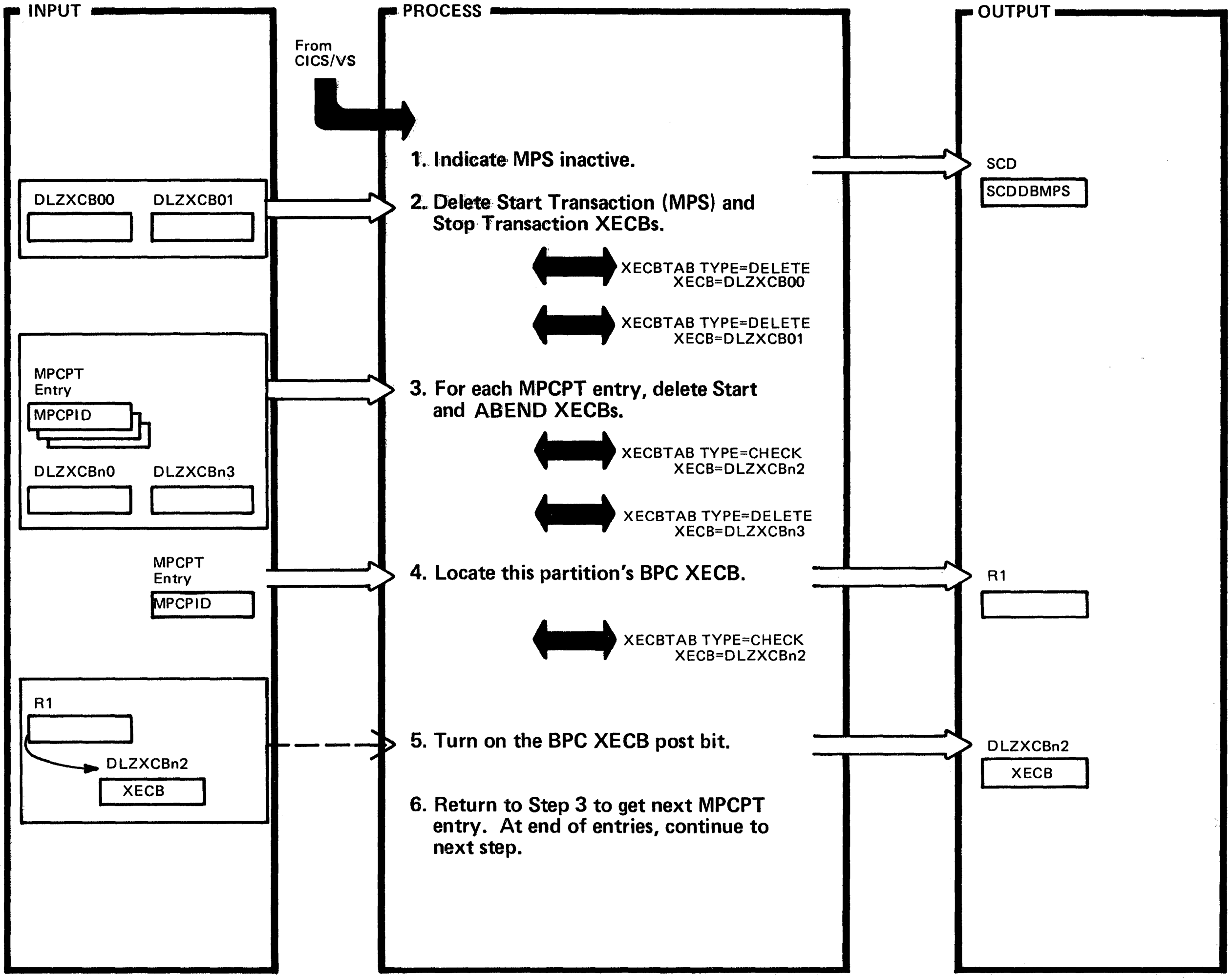
DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. This routine is entered when MPS is to be terminated normally or abnormally.		MPCEXIT
2. If error return on DELETE, issue message DLZ082I.		
3. Flag SCDXECB at SCDDBMPS is turned off and message DLZ094I is issued to indicate MPS stopped.		
4.		CICSRTN

Extended Description	Routine	Label

Figure 2-19.9. MPC ABEND Exit Routine (Part 1 of 2)



DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
1. This routine is entered from CICS/VS if an ABEND occurs in MPC. Linkage was established through DFHPC TYPE=SETXIT in Figure 2-19.2. Flag SCDXECB at SCDDBMPS is turned off to show MPS as inactive.		MPCABEXT
3.		DLETLOOP
4.		CHKXECB
5. Note that the XPOST macro is not needed to turn on the POST bit because BPC XECB (DLZXCBn2) is defined in the same partition as this module. DLZXCBn2 is defined by DLZBPC00.		

Extended Description	Routine	Label

Figure 2-19.9. MPC ABEND Exit Routine (Part 2 of 2)

INPUT

TCA

- TCAPCAC
- TCAPCPSW

PROCESS

7. Set information for message DLZ104I and issue it.

8. Return to CICS/VS.

(DFHPC TYPE=ABEND ABCODE=DMPC)

To CICS/VS

OUTPUT

DLZMPC00 — Master Partition Controller CSECT

DLZMPC00

Extended Description	Routine	Label
7.		MSG104
8. DMPC ABEND code defines MPC failure for CICS/VS dump ID.		

Extended Description	Routine	Label

Figure 2-20. Batch Partition Controller (Overview)

INPUT

From CICS/VS DFHPCP

PROCESS

1. Initialize BPC task. (See Figure 2-20.1)

2. Issue online DL/I scheduling call on behalf of the batch partition. (See Figure 2-20.2)

3. Wait on BPC and ABEND XECBs. (See Figure 2-20.3)

4. If BPC XECB posted, process batch request. (See Figure 2-20.4)

5. If ABEND XECB posted, terminate BPC. (See Figure 2-20.5)

To CICS/VS DFHPCP

From CICS/VS

6. BPC abend exit routine. (See Figure 2-20.6)

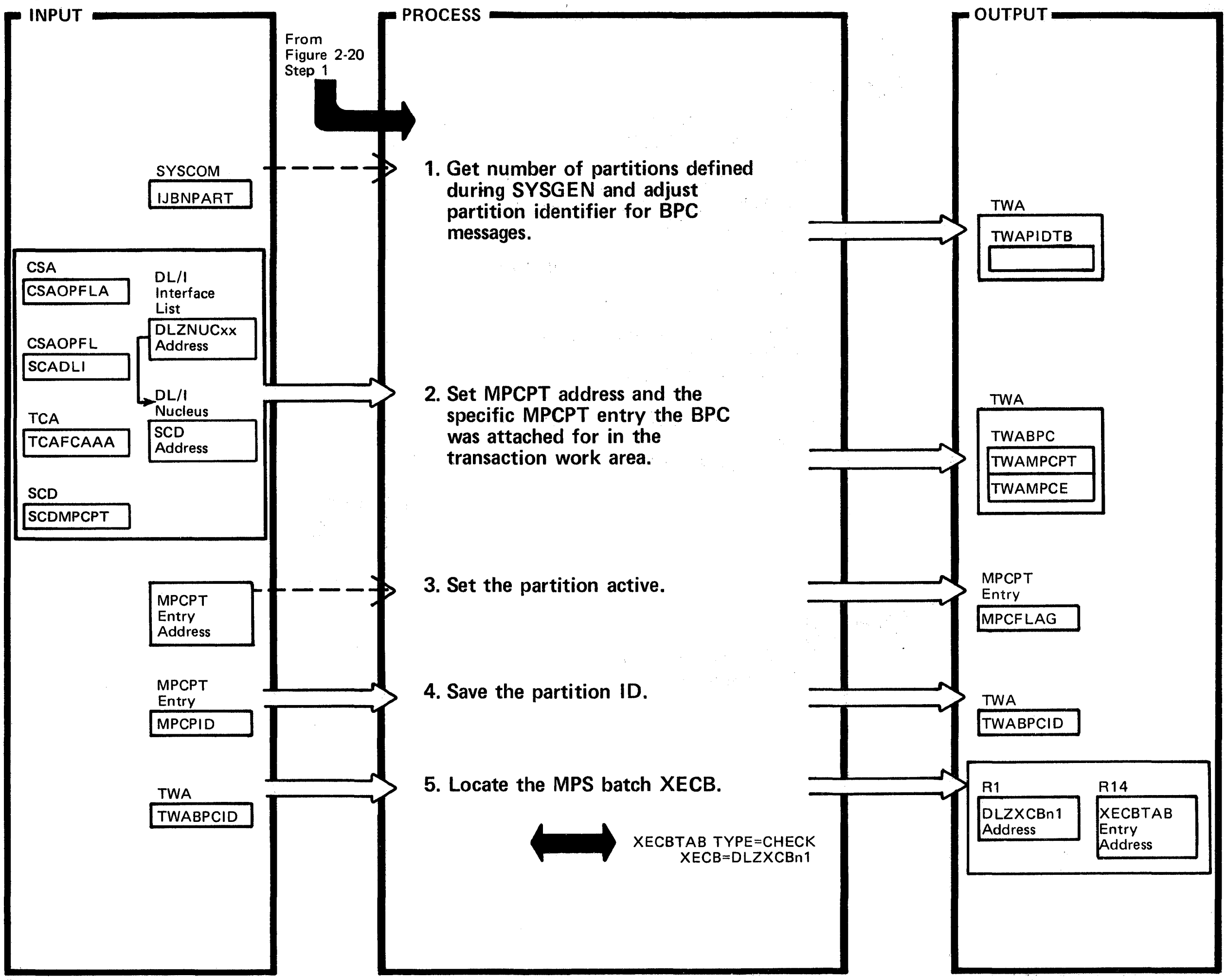
OUTPUT

DLZBPC00

Extended Description	Routine	Label
1.	DLZBPC00	DLZBPC00 BPCSTART
2.		BPCSCHCK
3.		BPCWAIT
4.		BPCCALL
5.		BPCEXIT
6.		BPCABND

Extended Description	Routine	Label

Figure 2-20.1. BPC Task Initialization (Part 1 of 3)



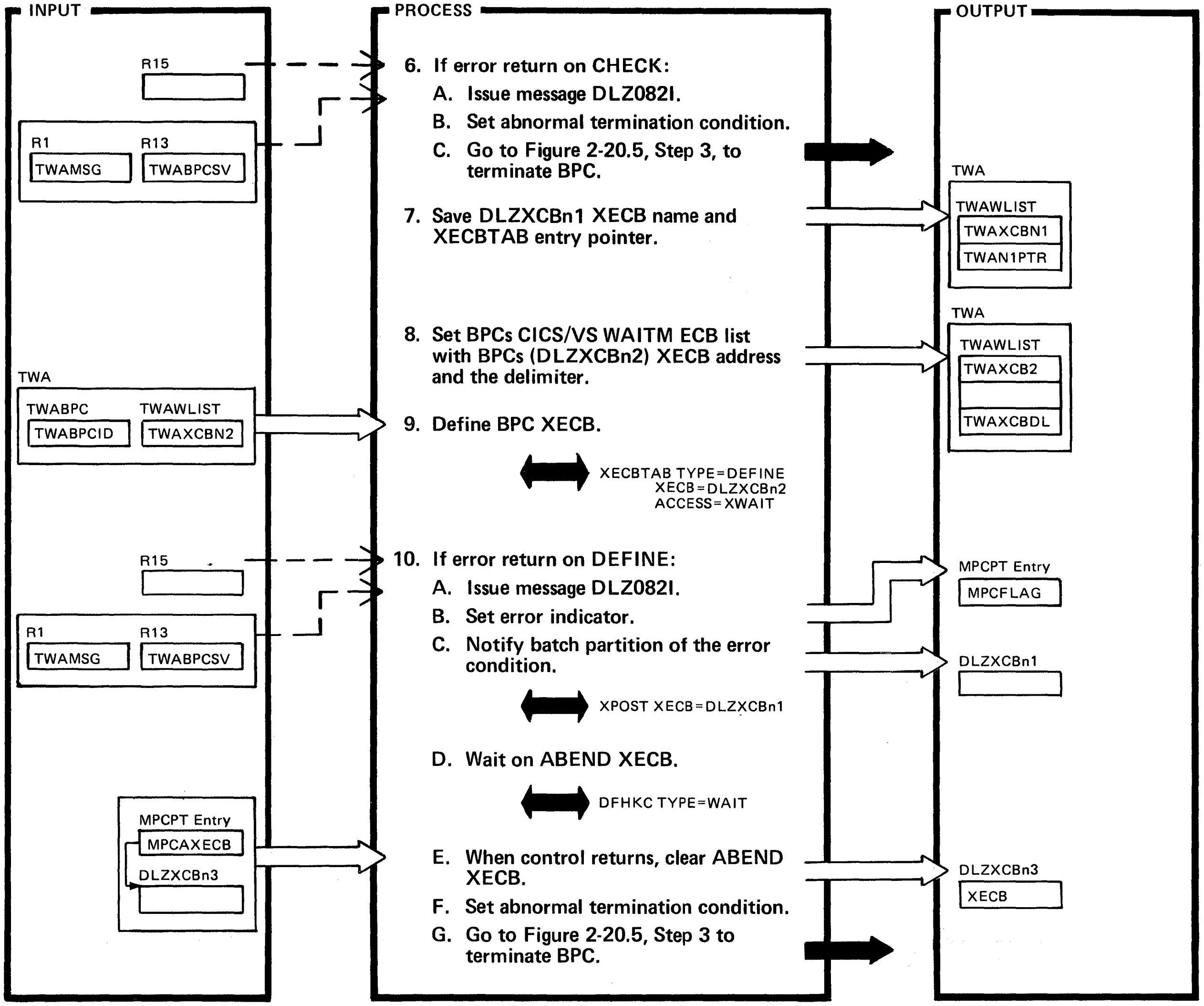
DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. Module identifier (DLZBPC00 vrnp) is defined here. The Batch Partition Controller (BPC) is attached by DLZMPC00 when a start request has been made by a partition. On entry, R12 contains address of TCA and R13 contains address of CSA. DLZXCBn1 = XECB name for MPS batch partition, where n is the partition ID based on key of partition. DLZXCBn2 = XECB name for a BPC for a specific partition (n). DLZXCBn3 = XECB name for handling an abend condition for a specific partition (n).	DLZBPC00	DLZBPC00 BPCSTART
2.		BPCCONT
3. Flag MPCPACT in MPCFLAG indicates this partition is active.		
5. XECBTAB TYPE=CHECK macro is used to obtain the address of MPS batch XECB.		XECBCHK

Extended Description	Routine	Label

Figure 2-20.1. BPC Task Initialization (Part 2 of 3)



DLZBPC00 – Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
9. BPC XECB (DLZXCBn2) is defined for cross partition communication with MPS Batch Initialization (DLZMINIT), MPS Batch Program Request Handler (DLZMPRH), and MPS Batch Termination (DLZMTERM).		XECBDFN
10.		
A. Flag MPCERR indicates an error condition.		BPCDFERR
C. The POST bit X'80' in the XECB is turned off.		

Extended Description	Routine	Label

Figure 2-20.1. BPC Task Initialization CSECT (Part 3 of 3)

INPUT

MPCPT Entry
MPCAXECB

PROCESS

11. Set BPCs CICS/VS WAITM ECB list with ABEND XECB address.

12. Set abend exit routine for Batch Partition Controller.

DFHPC TYPE=SETXIT
ROUTINE=BPCABND

13. Continue processing.

To Figure 2-20 Step 2

OUTPUT

TWA
TWAWLIST
TWAXCB2
TWAXCB3
TWAXCBDL

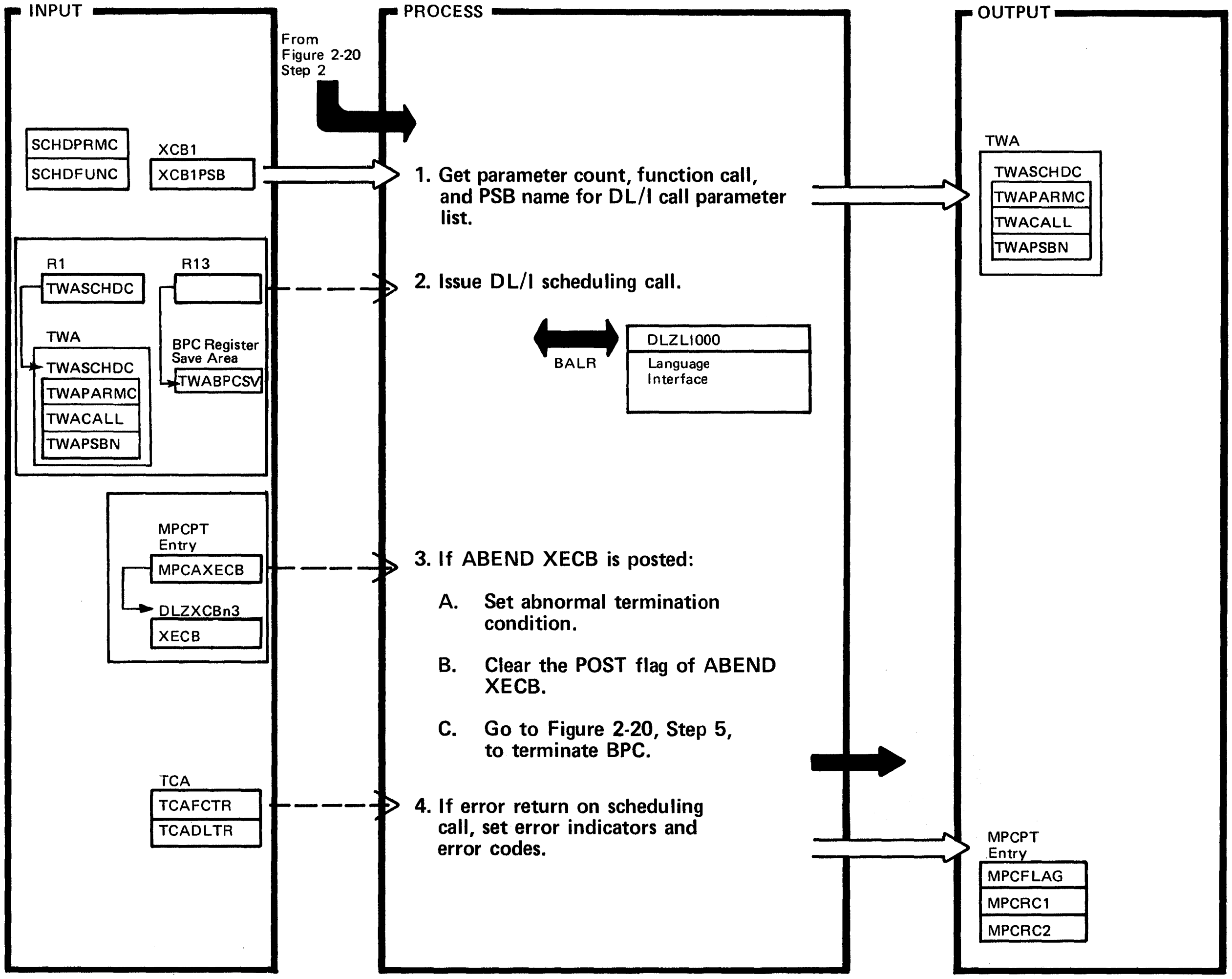
DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
12. The BPCABND routine (see Figure 2-20.6) is within this module.		

Extended Description	Routine	Label

Figure 2-20.2. Issue Online DL/I Scheduling Call (Part 1 of 2)



DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. Macro DLZXCB1 defines a DSECT that represents the format of the fields after the MPS Batch XECB (DLZXCBn1) used here as a parameter list by BPC. Addressability to DLZXCBn1 was obtained by the XECBTAB TYPE=CHECK macro in Figure 2-20.1, Step 5.		
4. Flag MPCERR at MPCFLAG is turned on.		BPCSCHCK

Extended Description	Routine	Label

Figure 2-20.2. Issue Online DL/I Scheduling Call (Part 2 of 2)

INPUT

R12

R1 ↑TWAXCBN1 — R14 TWAN1PTR

R15

R1 ↑TWAMSG — R13 ↑TWABPCSV

DLZBPC00 — Batch Partition Controller CSECT

PROCESS

5. Save the TCA address for MPS Batch (DLZMPI00).

6. Notify Batch Partition that DL/I scheduling call has been completed.

   XPOST XECB=DLZXCBn1

7. If error return on XPOST:

   A. Issue message DLZ084I.

   B. Go to Figure 2-20, Step 5, to terminate BPC.

8. Continue processing.

To Figure 2-20 Step 3

OUTPUT

MPCPT Entry: MPCTCA

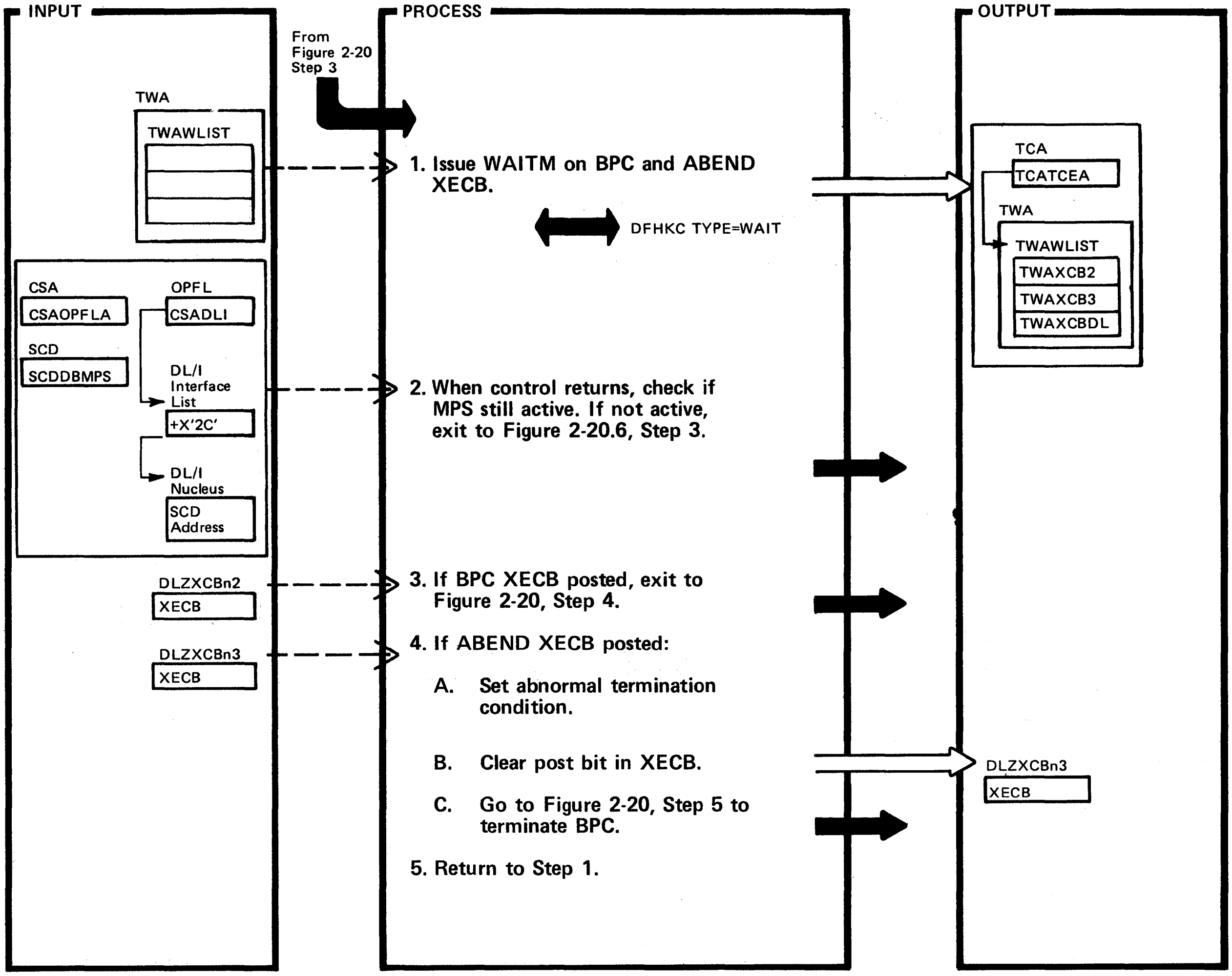
DLZXCBn1: XECB

DLZBPC00

Extended Description	Routine	Label
6.		BPCSCHDK

Extended Description	Routine	Label

Figure 2-20.3. Wait on BPC and ABEND XECBs



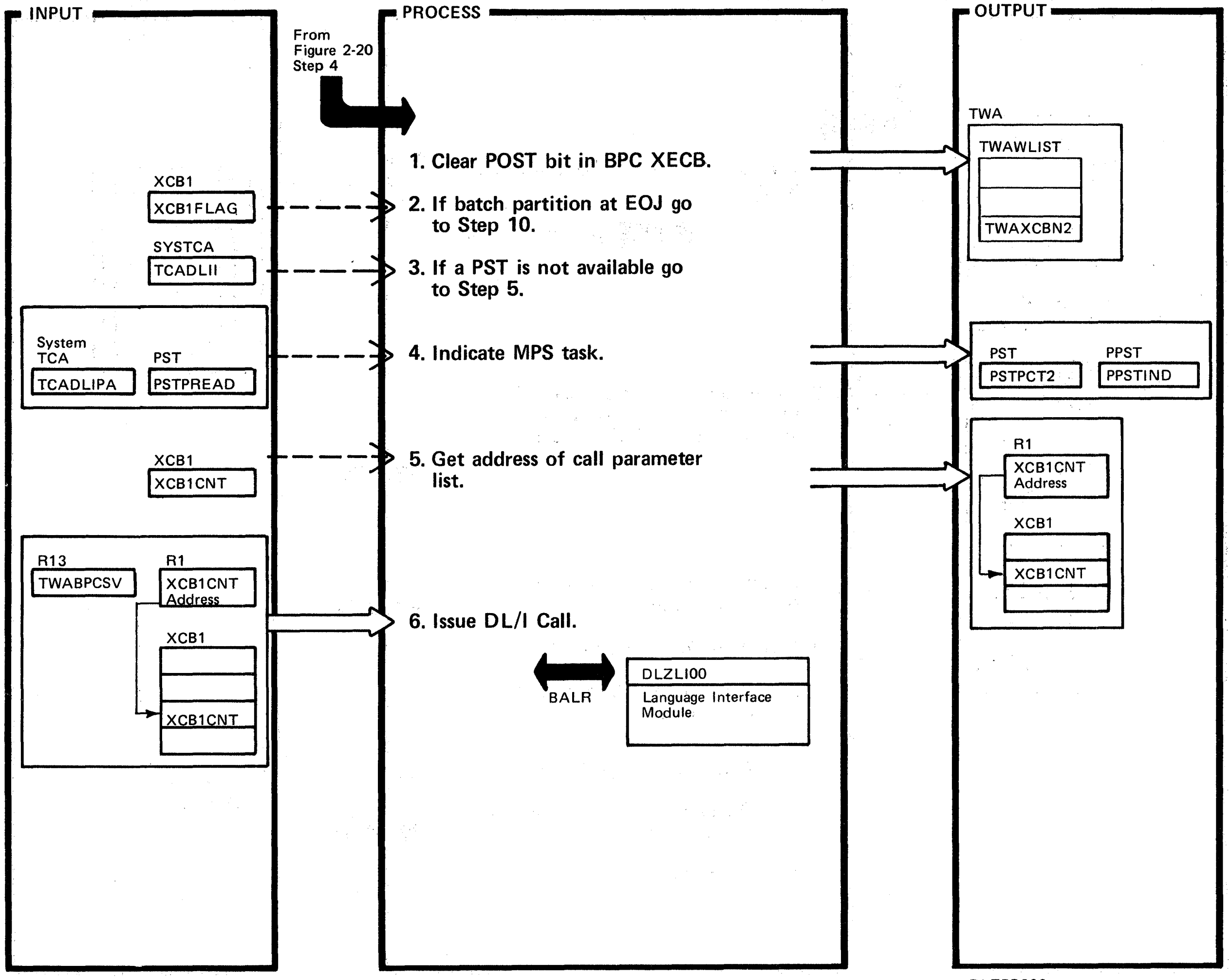
DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. The XECBs are posted for the following conditions: DLZXCBn2 ● Process call on behalf of batch partition. ● EOJ has been encountered in batch partition. DLZXCBn3 ● An ABEND condition has been encountered in the batch partition. 2. Flag SCDXECB is tested in SCDDBMPS.		BPCWAIT

Extended Description	Routine	Label

Figure 2-20.4. Batch Request Processing (Part 1 of 2)



DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. This routine is entered from BPC WAIT routine when the BPC XECB (DLZXCBn2) is posted by the batch program request handler. R10 points to the call parameter list in the MPS Batch partition.		
2. Macro DLZXCB1 defines a DSECT representing the format of the MPS batch XECB and following fields used for communication between the batch and online partitions. The End-of-Job flag is set by DLZMPI00 termination routine in the field following the XECB.		
3. Flag TCADLIPS indicates a PST is available.		
4. Flag PPSTMPS in PPSTIND indicates this is an MPS task.		
5.		BPCNOPST

Extended Description	Routine	Label
6. Entry point in the language interface module will be ASMTDLI or PLITDLI, depending on whether the user program is assembler or PL/I.		BPCDLICD

Figure 2-20.4. Batch Request Processing (Part 2 of 2)

INPUT	PROCESS	OUTPUT
R1 TWAXCBN1, R14 TWAN1PTR	7. Notify batch partition that the DL/I call has been completed. XPOST XECB=DLZXCBn1	DLZXCBn1 XECB
R15	8. If error return on XPOST:	
R1 TWAMSG, R13 TWABPCSV	A. Issue message DLZ084I.	
	B. Go to Figure 2-20, Step 5 to terminate BPC.	
	9. Return to Figure 2-20, Step 3 for BPC WAIT.	
	10. Notify MPS Batch Partition that EOJ processing has completed.	DLZXCBn1 XECB
	To Figure 2-20 Step 5	

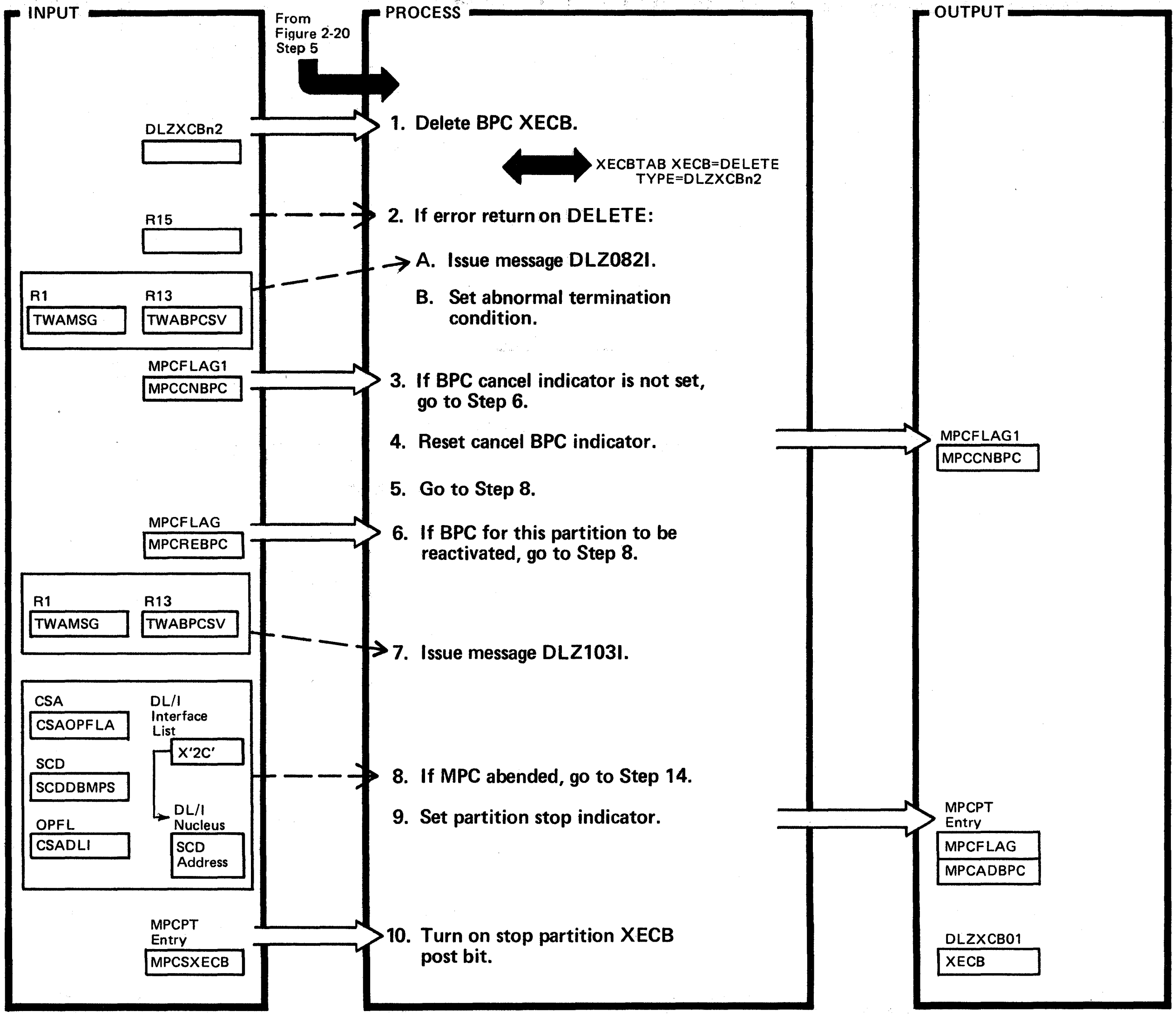
DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
10.		BPCEOJ

Extended Description	Routine	Label

Figure 2-20.5. BPC Termination (Part 1 of 2)



DLZBPC00 – Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. This routine is entered on normal or abnormal termination of BPC.		BPCEXIT
3. Bit MPCCNBPC is set by MPC when it determines that the batch partition no longer is active (i.e. 'cancel' replied to an access method message in batch partition). See 'Note'.		
6. Bit MPCREBPC is set by MPC when it determines that a BPC is still active for a partition and an MPS job is initiated for that same partition. See 'Note'. Note: The test in Step 3 and 6 are done to suppress output of message DLZ103I.		BPCEXIT2
7.		BPCEXIT1
8. Flag SCDXECB at SCDDBMPS indicates if MPS is active or not.		
9. Flag MPCPSTP in MPCFLAG is the stop partition indicator for DLZMPC00.		

Extended Description	Routine	Label
10. Note that the XPOST macro is not used to turn on the POST bit since the XECP was defined (DEFINED) in this partition (by DLZMPC00).		

Figure 2-20.5. BPC Termination (Part 2 of 2)

INPUT

PROCESS

11. If abnormal termination go to Step 13.

12. Return to CICS/VS.

DFHPC TYPE=RETURN

To CICS/VS DFHPCP

13. Set BPC abnormal termination completed indicator.

14. Return to CICS/VS – ABEND.

DFHPC TYPE=ABEND
ABCODE=DBPC
CANCEL=YES

To CICS/VS DFHPCP

OUTPUT

TWA
TWAMPSFG

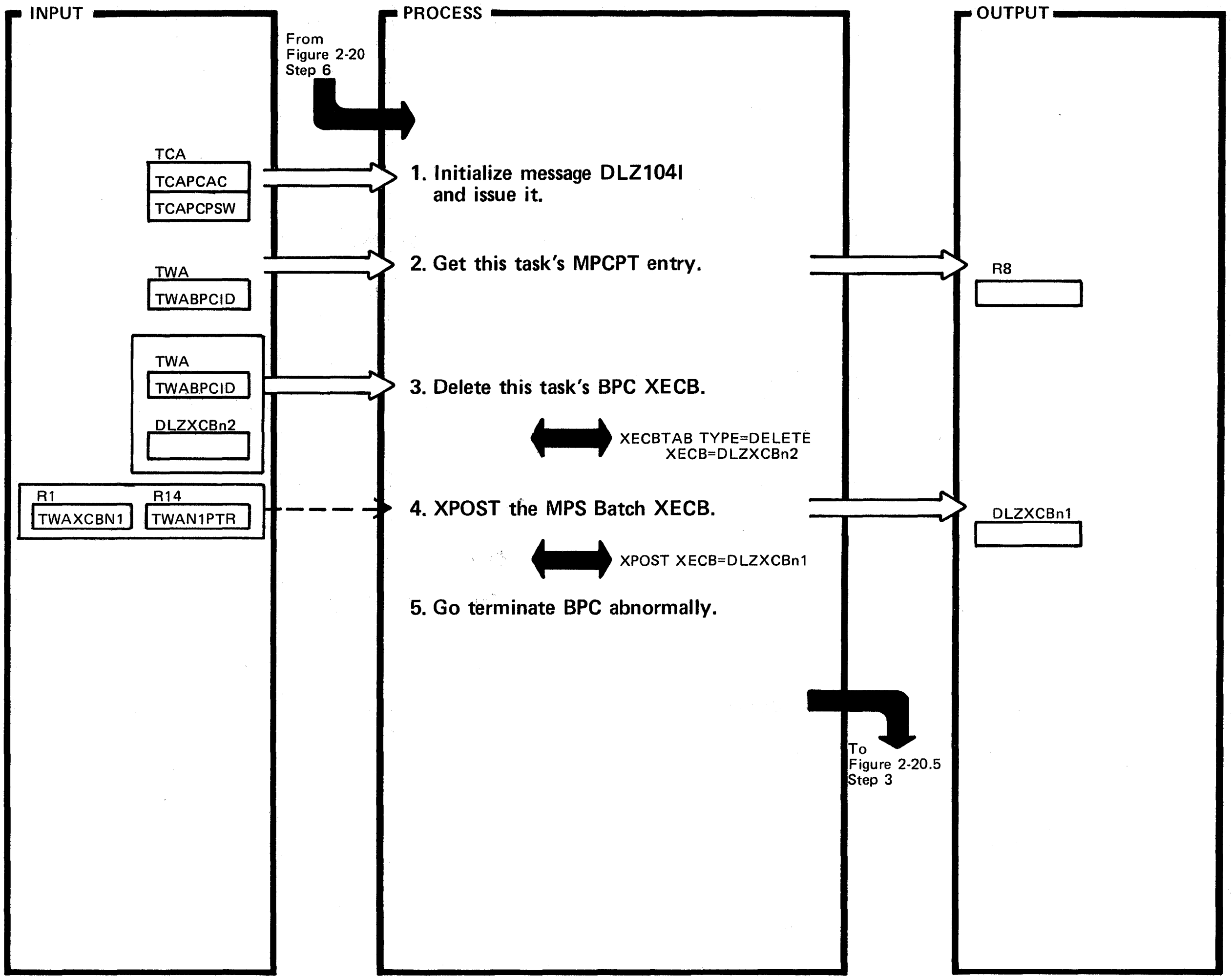
DLZBPC00 – Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
11.		CICSRTN
13. Flag TWABPCOK indicates BPC ABEND processing was successful.		BPCABEND
14. DBPC ABEND code defines BPC failure for CICS/VS dump ID.		

Extended Description	Routine	Label

Figure 2-20.6. BPC ABEND Exit Routine



DLZBPC00 — Batch Partition Controller CSECT

DLZBPC00

Extended Description	Routine	Label
1. This routine is entered from CICS/VS if an abend occurs in the Batch Partition Controller Module (DLZBPC00).		BPCABND
3.		MPCABEND

Figure 2-21. MPS Batch (Overview)

INPUT

From DOS/VS

PROCESS

1. MPS Batch Initialization. (See Figure 2-21.1)
2. MPS Batch Termination. (See Figure 2-21.2)

To DOS/VS

From Application Program

3. MPS Batch Program Request Handler. (See Figure 2-21.3)

To Application Program

4. MPS Batch Message Writer. (See Figure 2-21.4)
5. MPS Batch ABEND Handler. (See Figure 2-21.5)

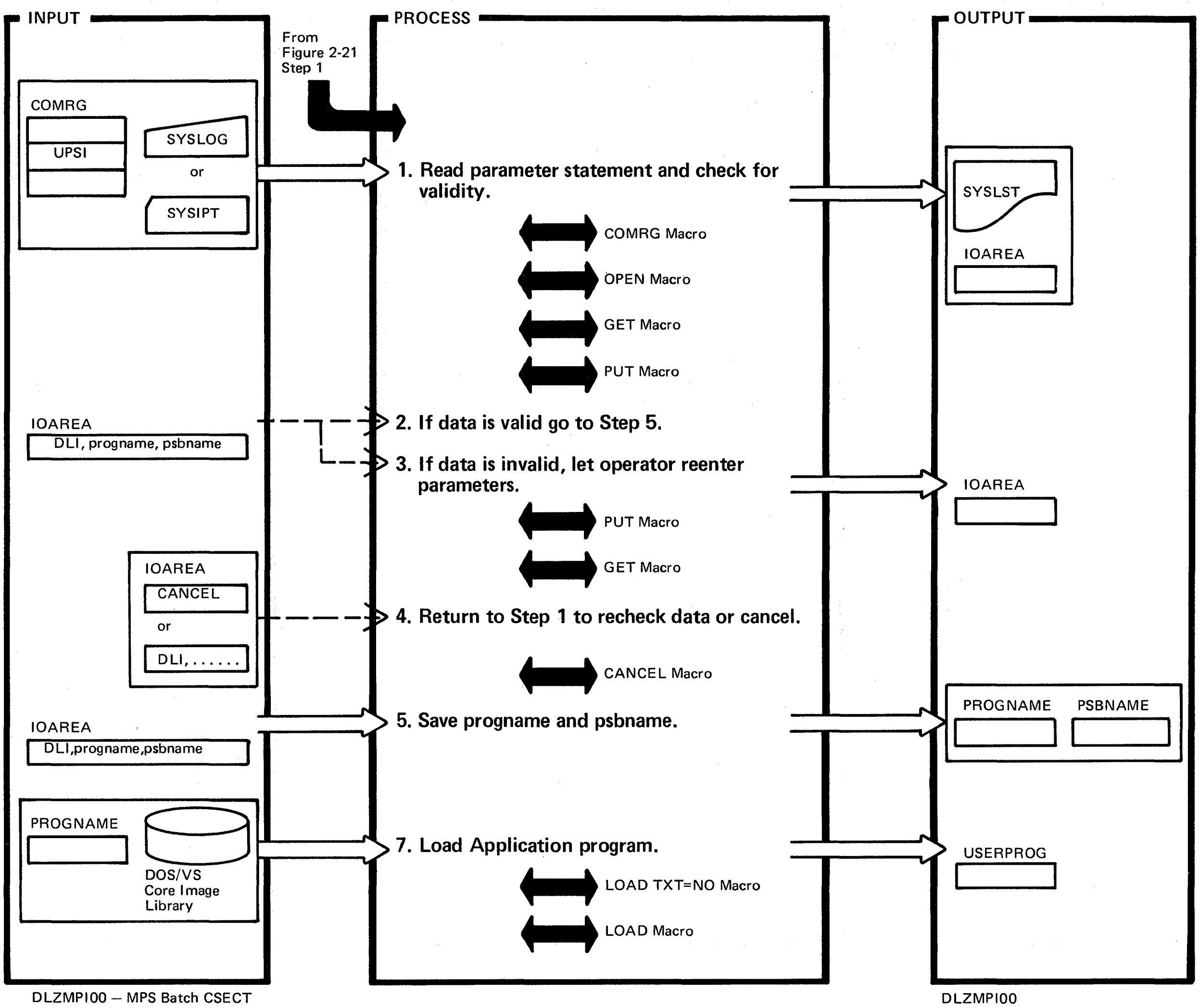
OUTPUT

DLZMPI00

Extended Description	Routine	Label
1.	DLZMPI00	DLZMINIT
2.		DLZMTERM
3.		DLZMPRH
4.		DLZMMSG
5.		DLZMABND

Extended Description	Routine	Label

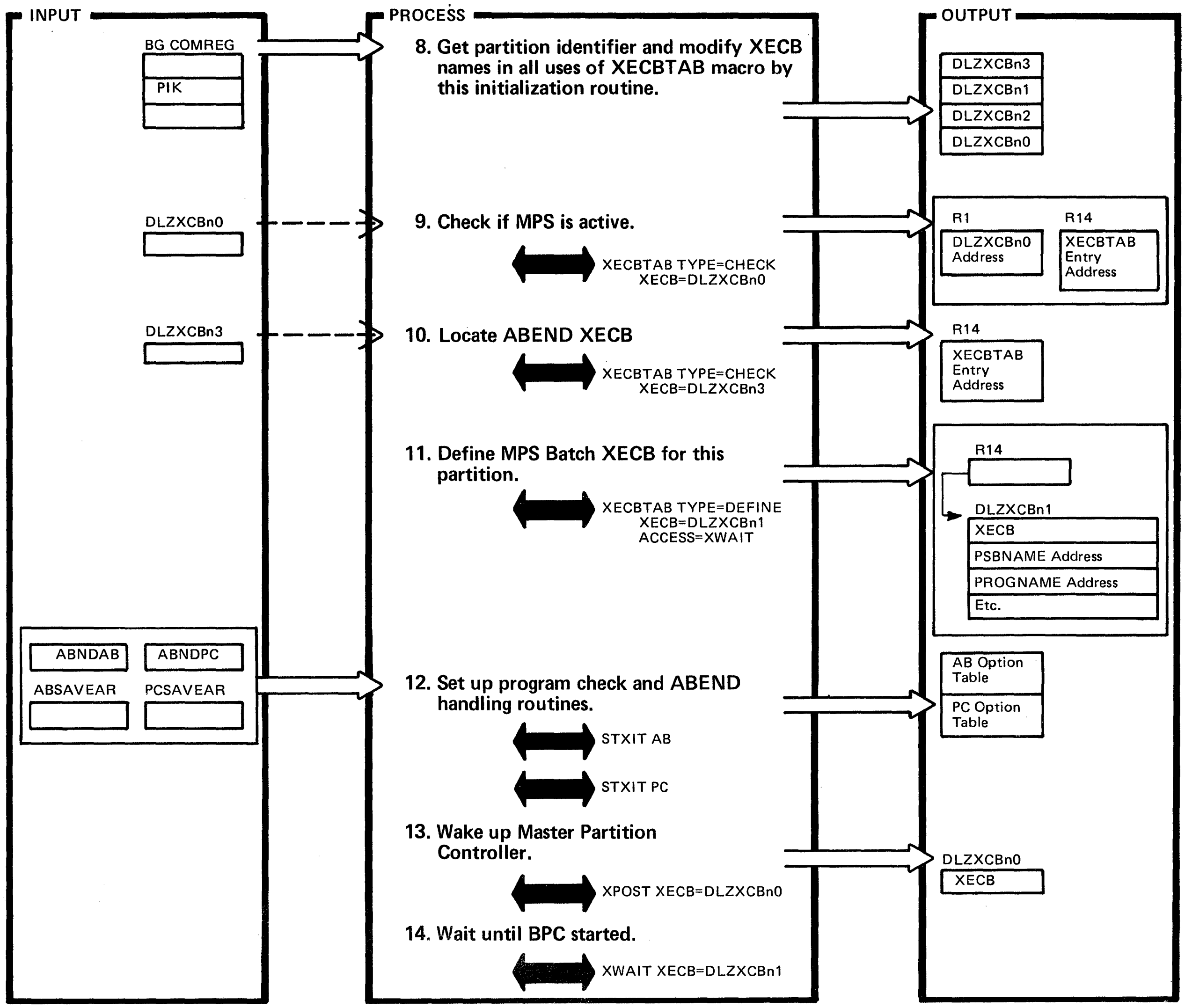
Figure 2-21.1. MPS Batch Initialization (Part 1 of 3)



Extended Description	Routine	Label
1. If UPSI byte bit 0 is on, input is from SYSLOG. Send message DLZ010A to have operator enter information. If UPSI byte bit 0 is off, input is from SYSIPT. Write message DLZ014A if end of file on SYSIPT.	DLZMPI00	DLZMINIT CHECKIN
3. Write message DLZ087A if data invalid.		PARMERR
4. Read operator reply to decide if more parameters provided or if job cancelled.		GETCONS3
5.		CHECKOK
7. If program not found, write message DLZ012I and cancel job.		LOADAP

Extended Description	Routine	Label

Figure 2-21.1. MPS Batch Initialization (Part 2 of 3)



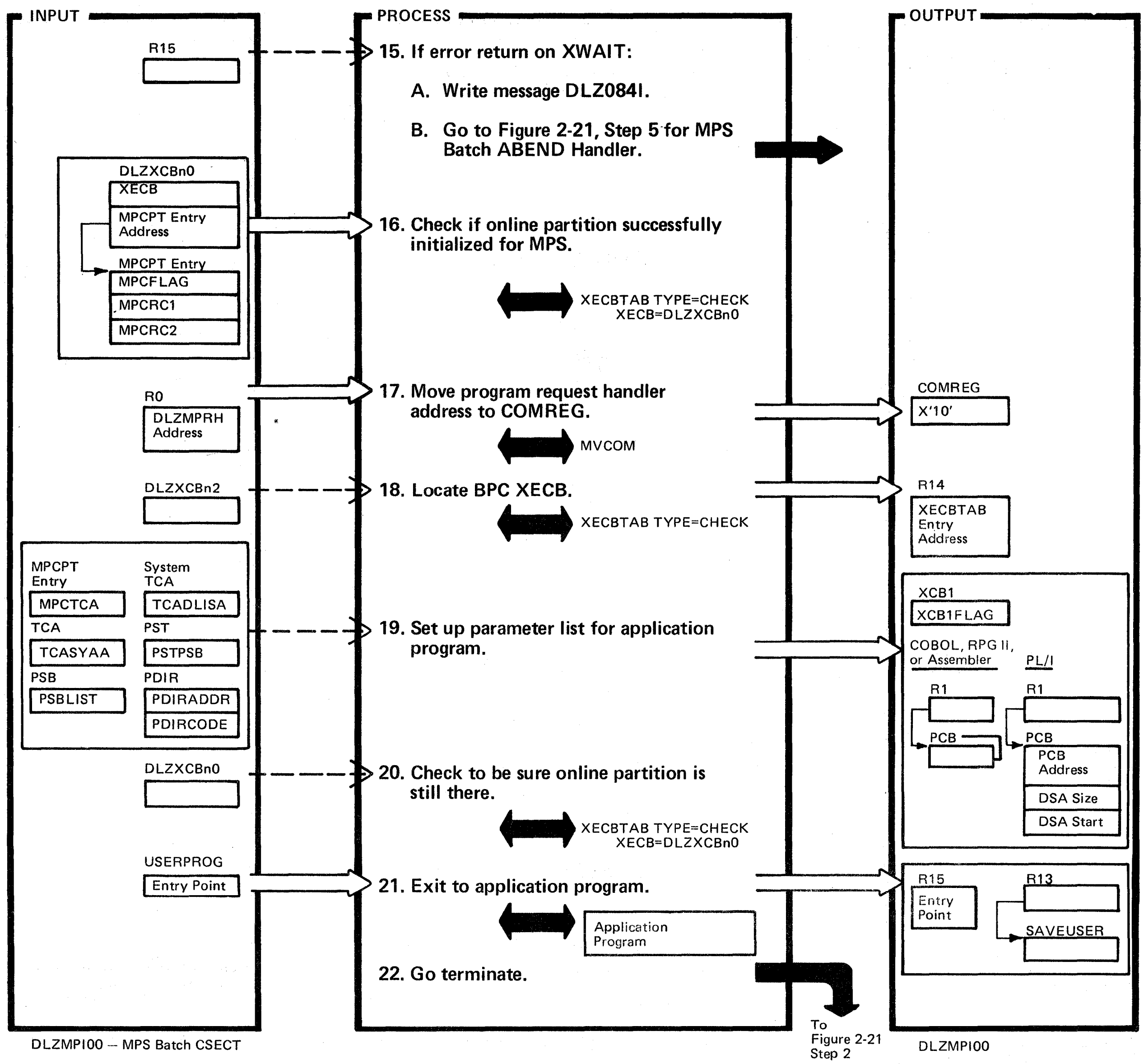
DLZMPI00 – MPS Batch CSECT

DLZMPI00

Extended Description	Routine	Label
8. The XECB names referenced in the XECBTAB macros used by MPS batch initialization routine are modified based on the partition that this MPS batch job is in. The value of the PIK in BG COMREG always reflects the currently active partition's PIK. Because this job is currently active as it is checking the BG PIK, it will be the PIK of the partition where this job is.		
9. If a start batch XECB (DLZXCBn0) is not found for this MPS batch job Partition (n), write message DLZ089I and cancel. When found, the XECBTAB/CHECK macro returns in R1 the XECB address and in R14 the XECB table entry.		
10. Write message DLZ089I if the ABEND XECB for this partition is not found.		XECBCHK3
11. Write message DLZ082I if DEFINE is not successful.		XECBDEF1

Extended Description	Routine	Label
11. (Cont'd) Following MPS batch XECB (DLZXCBn1) are parameter fields used in communicating with the online partition. Macro DLZXCB1 contains the DSECT which describes this XECB and following fields.		
12. Save address of PC option table for DLZMPRH (Figure 2-21.3, Step 1).		
13. Notify the online partition (DLZMPC00 specifically) that an MPS batch job is ready to execute and write batch started message DLZ081I if XPOST successful. If XPOST unsuccessful, delete MPS batch XECB (DLZXCBn1), write message DLZ084I, and cancel.		XPOST1
14. Wait is made for DLZBPC00 to post the MPS batch XECB (DLZXCBn1) to notify us it is initialized and has completed a DL/I scheduling call for us.		XWAIT1

Figure 2-21.1. MPS Batch Initialization (Part 3 of 3)



Extended Description	Routine	Label
16. As the start partition XECBs are defined in DLZMPC00, the corresponding MPCPT entry is saved for each following the XECB. If the XECB is not at the same address as earlier, indicating there was a deletion and new define, or it no longer exists, write message DLZ082I and DLZ099I, then ABEND.		
Write message DLZ095I and the return code if a scheduling error indicated.		
Write message DLZ085I if BPC could not be attached.		MVCOM
18.		XECBCHK2
19. If PL/I – a three-word list is set up with pointers to PCB list, amount of dynamic storage, and start of dynamic storage area for PL/I.		

Extended Description	Routine	Label
19. (con't).		
If Cobol, RPG II, or Assembler – R1 points to first PCBADDR.		
The language indicator is set in the parameter field following the MPS batch XECB (See DLZXCB1 Macro).		
20. If the start partition XECB is not found, write message DLZ082I, then ABEND.		
21. An application program runs as a subroutine of DLZMPI00.		

Figure 2-21.2. MPS Batch Termination

INPUT

R15

DLZXCBn1

R15

DLZXCBn1

DLZMPI00 — MPS Batch CSECT

PROCESS

From Figure 2-21 Step 2

1. Notify online batch partition controller for this partition to terminate.

   XPOST XECB=DLZXCBn2

2. If error return on XPOST:

   A. Write message DLZ090I.

   B. Go to Figure 2-21, Step 5, to ABEND.

3. Wait for online BPC to terminate.

   XWAIT XECB=DLZXCBn1

4. If error return on XWAIT:

   A. Write message DLZ090I.

   B. Go to Figure 2-21, Step 5, to ABEND.

5. Delete MPS batch XECB.

   XECBTAB TYPE=DELETE XECB=DLZXCBn1

6. The end.

   EOJ Macro

To DOS/VS

OUTPUT

XCB1 DLZXCBn2

XCB1ECB

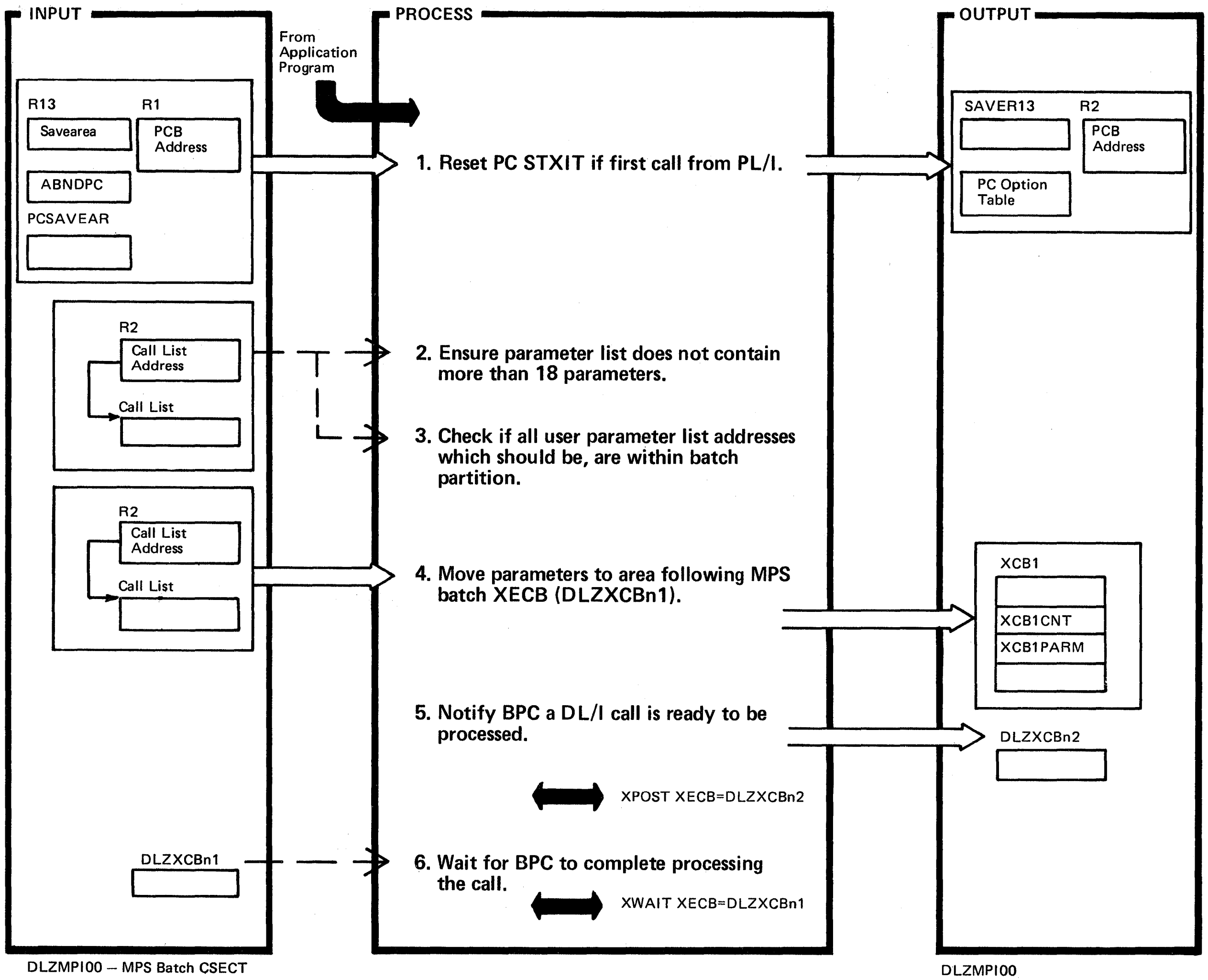
XCB1FLAG

DLZMPI00

Extended Description	Routine	Label
1. This routine is entered when the application program completes. If the application program is written in PL/I code, the SCDLIPLI flag in the SCD is reset to 0. Macro DLZXCB1 defines a DSECT representing the format of the MPS batch XECB and following fields used for communicating with the online partition. The end-of-job flag is set to tell DLZBPC00 the batch partition is at EOJ and the BPC XECB is posted to tell DLZBPC00 to stop.		DLZMTERM
3.		XWAIT2
6.		EOJ

Extended Description	Routine	Label

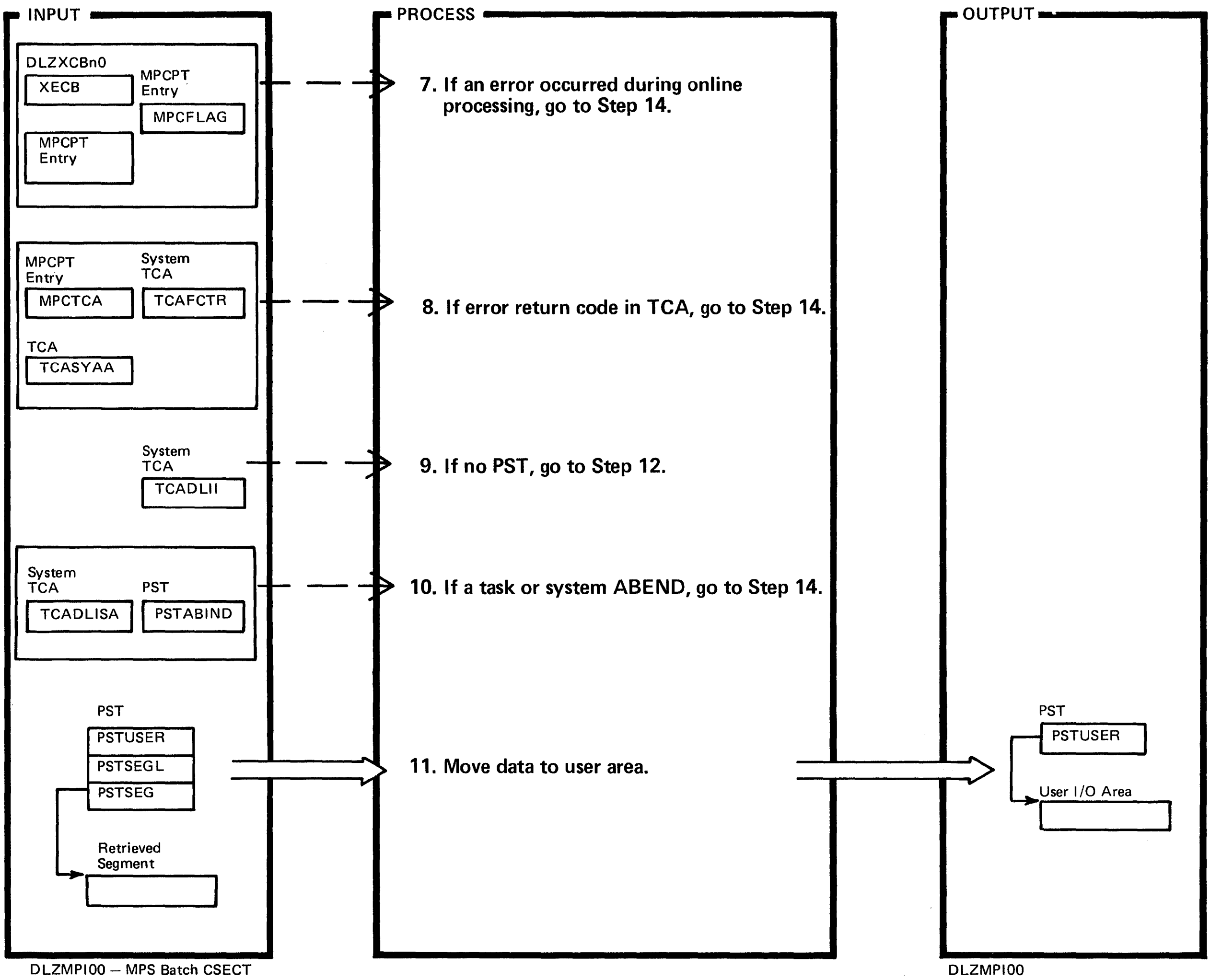
Figure 2-21.3. MPS Batch Program Request Handler (Part 1 of 3)



Extended Description	Routine	Label
1. This routine is entered on each call to DL/I made by the application program. During the first entry to DLZMPRH, the PL/I STXIT routine and savearea addresses from the PC option table are saved if the application program is written in PL/I. DLZMPRH also sets/resets a switch (SCDLIPLI flag in SCD) on exit/entry to indicate whether current execution is in DL/I code or PL/I code. This is done to enable high level language debugging for PL/I to give diagnostic information if a program check occurs in PL/I code. PL/I reissues STXIT PC when application program starts. Therefore, DL/I must reissue STXIT to get control after PL/I issues its STXIT PC.		DLZMPRH
2. Write message DLZ091I if more than 18 parameters.		COUNTLP
3. Ensure call list and addresses it points to are within batch partition (except for PCB). If PL/I, ensure that pointers pointed to by pointers, are within the batch partition.		CHKMOVE1

Extended Description	Routine	Label
3. (cont'd) Write message DLZ092I if there is a bad address and ABEND.		
4. Macro DLZXCB1 defines the DSECT describing the DLZXCBn1 XECB used for communicating with the online batch partition controller (DLZBPC00).		
5. If error return on XPOST, write message DLZ084I, then ABEND.		XPOST0
6. If error return on XWAIT, write message DLZ084I, then ABEND.		XWAIT0

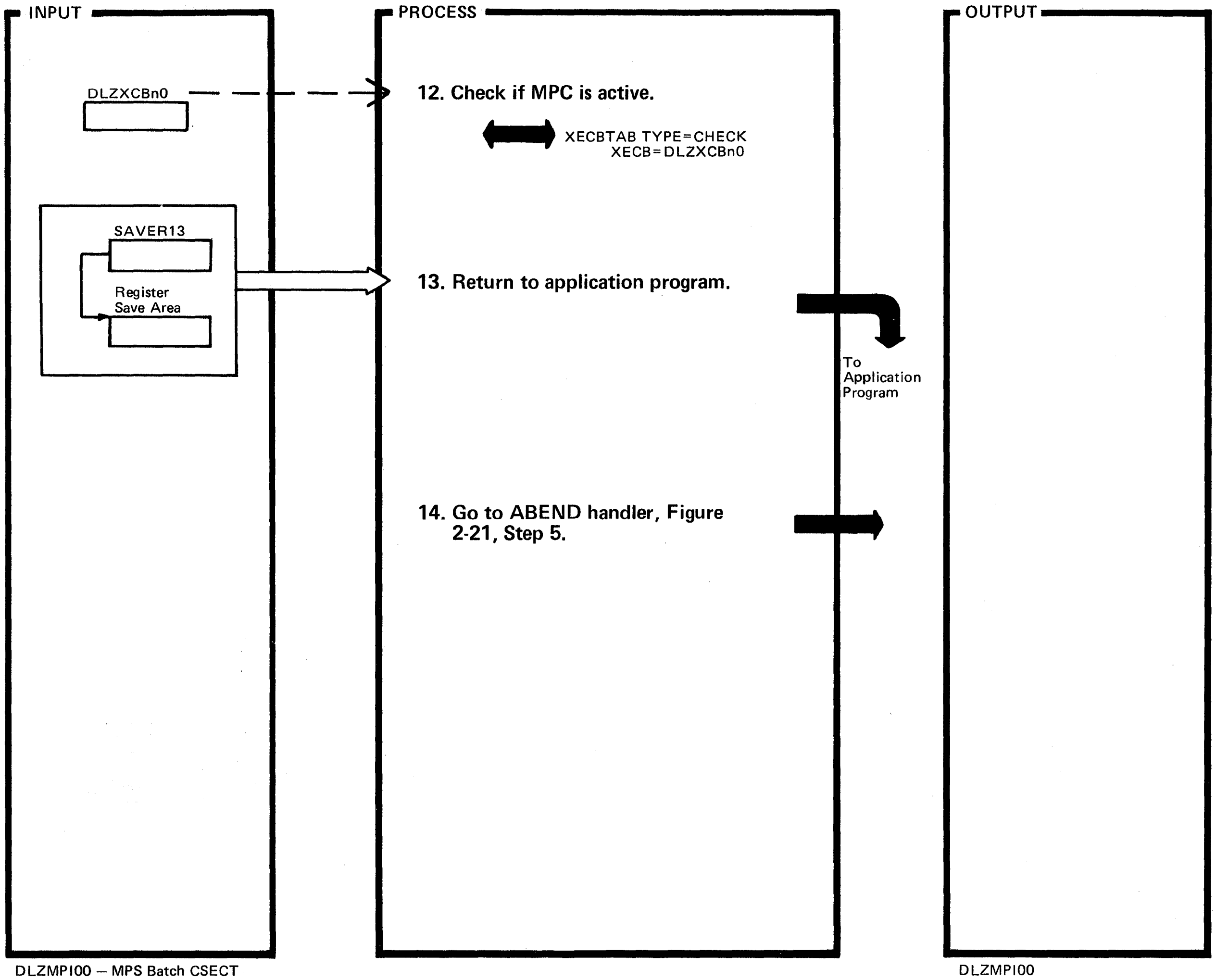
Figure 2-21.3. MPS Batch Program Request Handler (Part 2 of 3)



Extended Description	Routine	Label
7. MPCERR flag indicates an error condition occurred during BPC processing. Write message DLZ100I if on.		
8. Write message DLZ102I including the return code if present.		
9. The storage acquired indicator TCADLIPS is turned on by online nucleus DLZODP when the PST is acquired. If no PST, it just did a TERM call.		
10. Write message DLZ098I if PST contains an error.		
11. Write message DLZ100I if the data addresses are invalid.		

Extended Description	Routine	Label

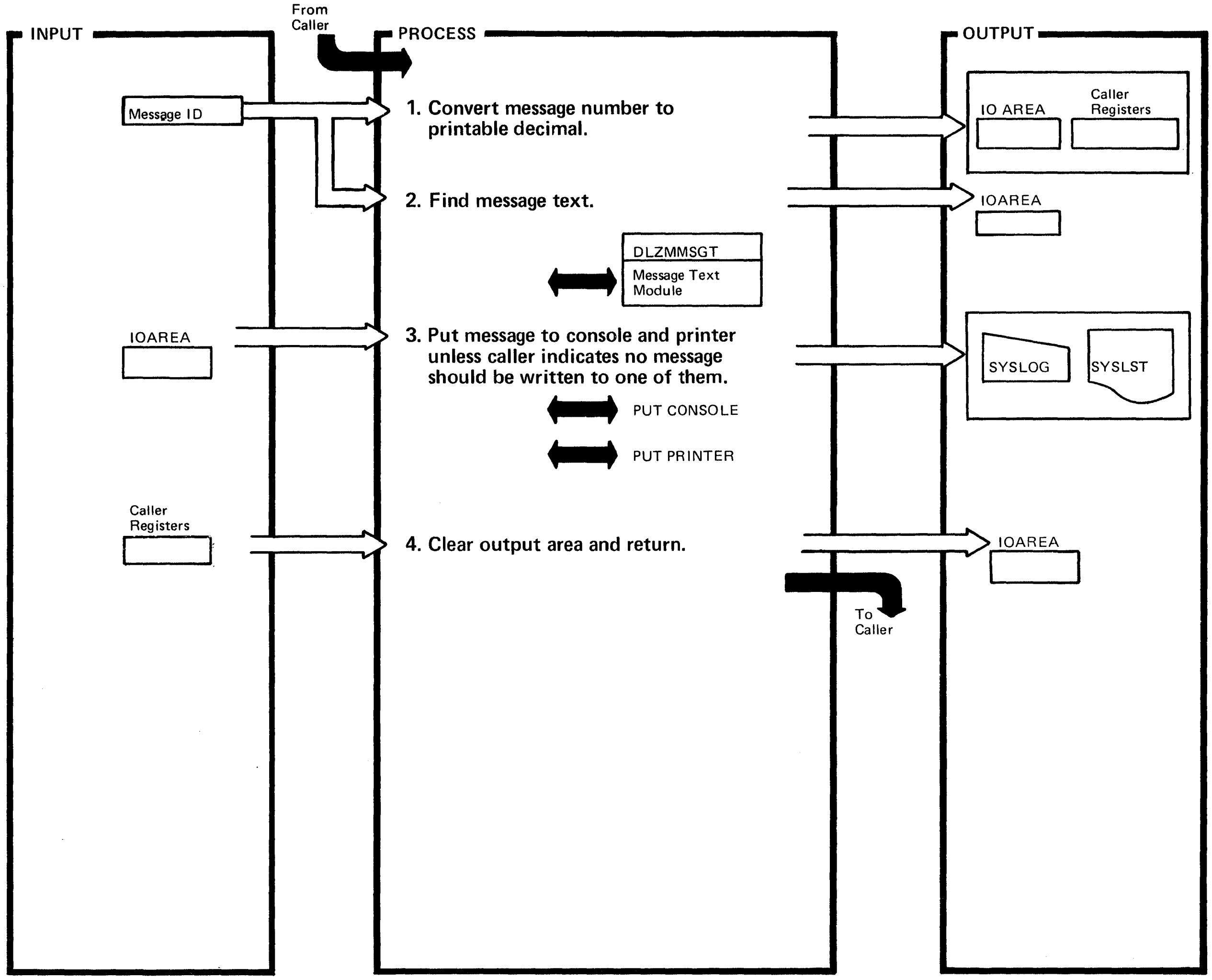
Figure 2-21.3. MPS Batch Program Request Handler (Part 3 of 3)



Extended Description	Routine	Label
12. If the START PARTITION XECB is not at the same address as when the batch job started indicating there was a deletion and new define, or if it no longer exists, write message DLZ082I and go to Step 14.		NODATA

Extended Description	Routine	Label

Figure 2-21.4. MPS Batch Message Writer



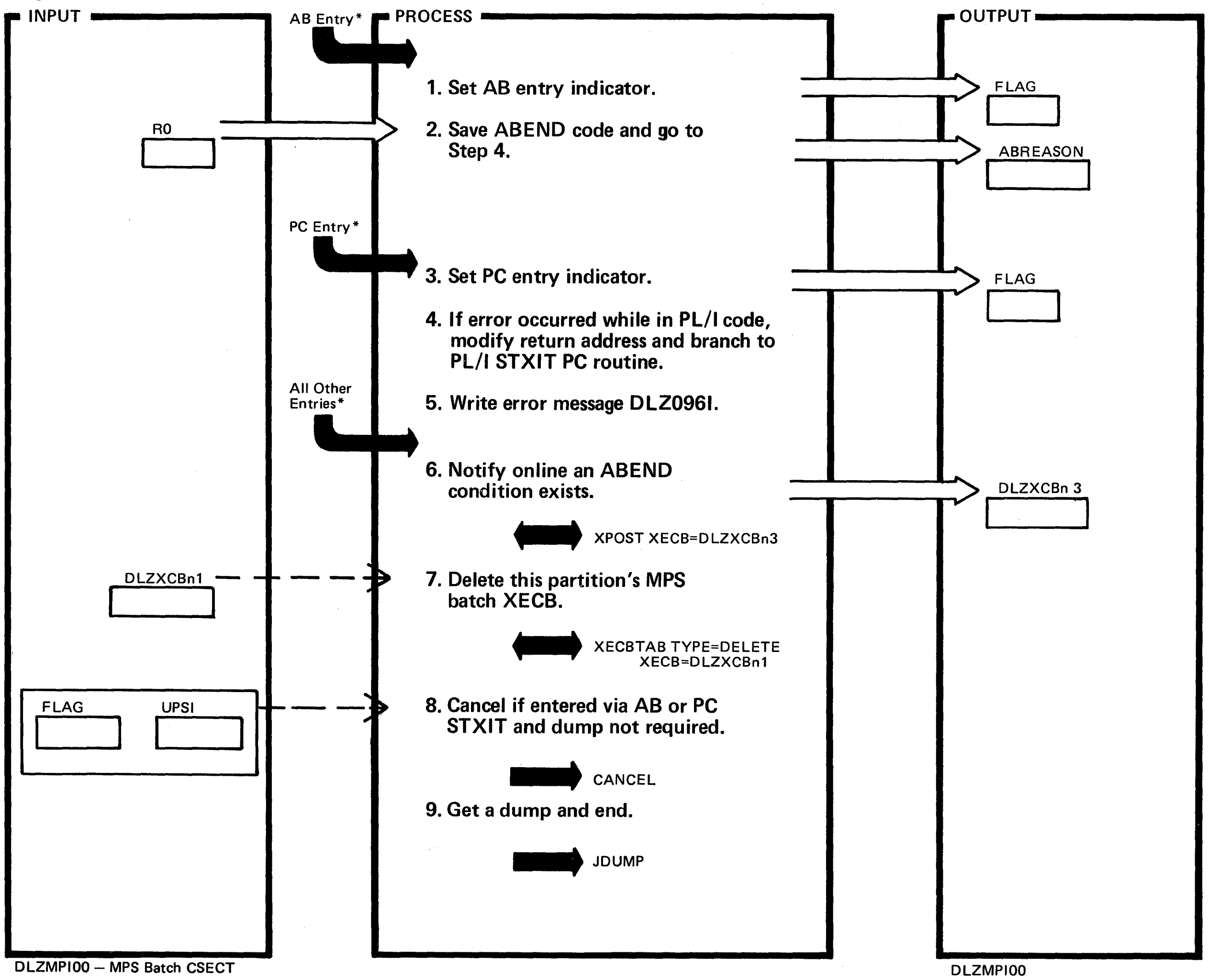
DLZMPI00 – MPS Batch CSECT

DLZMPI00

Extended Description	Routine	Label
1. This routine is entered when a message is to be written to SYSLOG and/or SYSLST.		DLZMMSG
2. The message module DLZMMSGT includes all messages that can be issued by MPS and is used in both the batch and online partitions.		
3.		PUTCONS2

Extended Description	Routine	Label

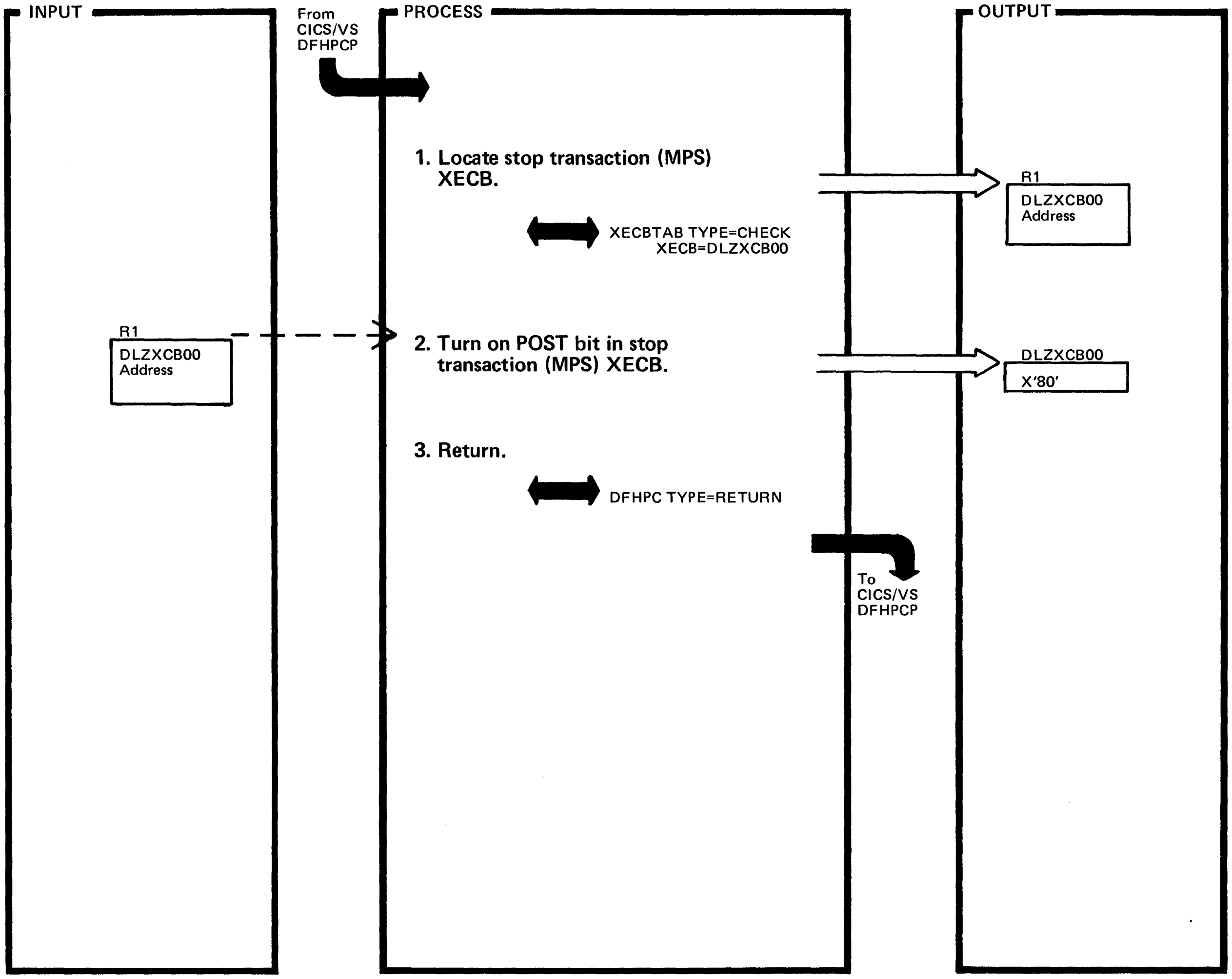
Figure 2-21.5. MPS Batch ABEND Handler



Extended Description	Routine	Label
* There are three entries to this routine: 1. AB STXIT 2. PC STXIT 3. The MPS Batch Initialization, MPS Batch Termination, and MPS Program Request Handler routines (whenever XPOST is needed to tell online that batch completed unsuccessfully).		
1. The AB output area is located in a dump following the DC C 'AB SAVE' characters. The AB reason code is located in a dump following the DC C 'AB ABEND CODE'.		ABNDAB
2. The PC output area is located in a dump following the DC C 'PC SAVE' characters.		ABNDPC
4. The address of the PL/I STXIT PC routine was saved during the first entry to DLZMPRH (see Figure 2-21.3, Step 1). After PL/I completes diagnostic information, processing returns to modified address in DLZMABND.		

Extended Description	Routine	Label
5.		ABPC
6.		DLZMABND
7.		XPOST3
9.		JDUMP

Figure 2-22. Stop Transaction



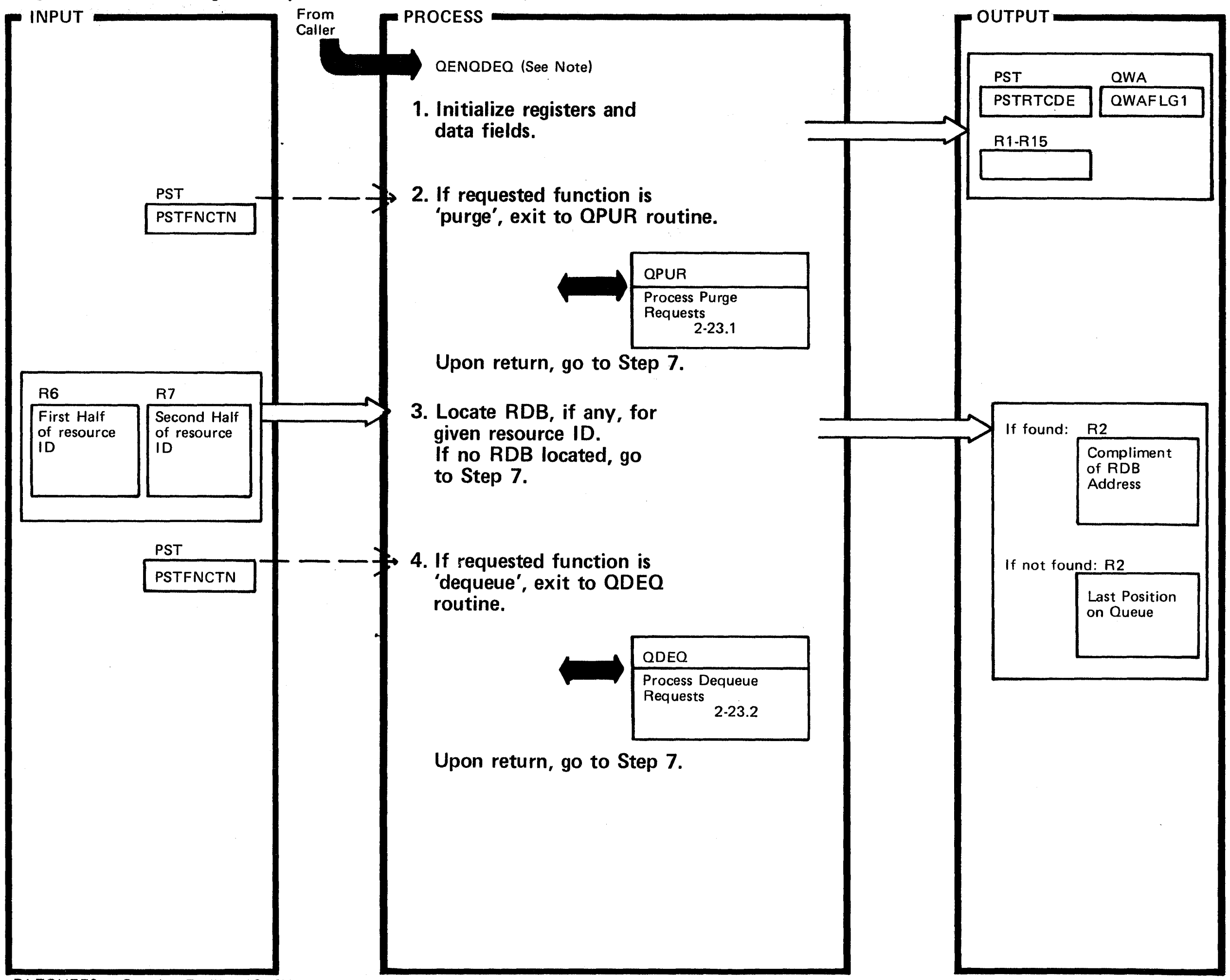
DLZMSTP0 — MPS Stop Transaction CSECT

DLZMSTP0

Extended Description	Routine	Label
1. Module identifier (DLZMSTP0) is defined here. Write message DLZ080I if DLZXCB00 does not exist — MPS not active — and go to Step 3.	DLZMSTP0	DLZMSTP0
2. Note that the XPOST macro is not needed to turn on the POST bit because stop transaction XECB (DLZXCB00) is defined in the same partition as this module. DLZXCB00 is defined by DLZMPC00.		
3.		RETURN

Extended Description	Routine	Label

Figure 2-23. Queuing Facility (Overview) (Part 1 of 2)



DLZQUEF0 – Queuing Facility CSECT

DLZQUEF0

Extended Description	Routine	Label
Note: QENQDEQ is the general purpose entry point for requests to 'enqueue; 'dequeue', or 'verify' a resource, or to 'purge' all enqueues for a task.		
1. Module identifier (DLZQUEF0vrnp) is defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied.	QENQDEQ	
3. The proper queue head is first located by hashing the resource ID. That queue is then searched for a RDB with the same resource ID. If found, its address is passed back complimented. Otherwise, the address of the last position on the queue is returned.	QLOCRDB	

Extended Description	Routine	Label

Figure 2-23. Queuing Facility (Overview) (Part 2 of 2)

INPUT	PROCESS	OUTPUT
PST: PSTFNCTN	5. If requested function is 'enqueue' or 'verify' exit to QENQVER routine. (QENQVER — Process Enqueue/Verify Requests 2-23.3) Upon return, go to Step 7.	
PST: PSTFNCTN; SCD: SCDABEND, SCDERRMS	6. If the PSTFNCTN Field contains an unrecognizable function code, terminate the task. (DL/I ABEND Routine.)	
	7. Return. (To Caller)	R15: 0 – if normal exit; 4 – if error resulting in termination occurred

DLZQUEFO – Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
6. Message DLZ267I is issued.		
7. This routine is used for common processing during exit for enqueue, dequeue, verify, and purge request. If processing is successful, a 0 return code is set in R15. Otherwise, the PST return code field (PSTRTCDE) is checked for error return codes. If present, the R15 return code is set to 4 and the registers are saved in the ABEND save area.	QRETURN	

Extended Description	Routine	Label

Figure 2-23.1. Process Purge Requests

INPUT

R1

PST

PSTRRDF

PSTRRDL

RRD

RRDFLAG

From Figure 2-23 Step 2

PROCESS

QPUR (See Note)

1. Scan each RRD attached to the PST RRD chain. For each RRD:

   A. Unchain the RRD from the PST forward chain by storing the address of the next RRD in the 'first' PST pointer.

   B. If the task is an owner of the resource, the ownership is released and new owners, if any, are located and promoted.

   C. If the task is not an owner, the RRD is unchained from the RDB-RRD chain and its space is returned to the free block queue. Flag parameters are then passed to any other waiting task.

2. When all RRDs on the chain have been processed, the 'last RRD' pointer is reset. Also, the number of resources owned and the wait chain pointer are both set to zero.

3. Exit to QRETURN.

To Figure 2-23, Step 7

OUTPUT

PST

PSTRRDF

PST

PSTRTCDE

PST

PSTRRDL

PSTNORO

PPST

PPSTCW

DLZQUEFO – Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
Note: This routine is used to purge all complete or outstanding enqueue requests for a given task.		
1. A. The address of the next RRD is saved because it would be destroyed when the current RRD is returned to the free space.	QPUR	
B. This routine is called to relinquish ownership of a resource by dequeuing the RRD for the task. It is entered by purge at entry point QDEQPUR to avoid the unnecessary overhead of unchaining the RRD from the PST.	QRELRSC	
C.	QPUR1 QRETBLK QPFLAGP	
2.	QPUR2	
3.	QPURX	

Extended Description	Routine	Label

Figure 2-23.2. Process Dequeue Requests (Part 1 of 2)

INPUT	PROCESS	OUTPUT
	From Figure 2-23 Step 4 → QDEQ (See Note)	
	1. If an RDB previously located (Figure 2-23, Step 3), locate the RRD. If no RRD exists, exit to QRETRUN (Figure 2-23, Step 7).	R3: RRD Address
PST: PSTQLEV	2. Decrement by 1, the ownership count for the specified level.	RRD: RRDNQRO or RRDNQUP or RRDNQEX
RRD: RRDMAXL	3. If the remaining ownership count is not zero (or if it is zero, but the level is not the maximum for the RRD), pass any FLAG parameters to the waiting tasks. Then go to Step 6.	PST: PSTRTCDE

DLZQUEF0 — Queuing Facility CSECT

DLZQUEF0

Extended Description	Routine	Label
Note: This routine is used to process a request to dequeue a resource.		
1.		QLOCRRD
2. This routine is also used by 'verify' processing to dequeue a resource that was enqueued due to a required wait.	QDEQVER	
3.	QDEQVER2 QPFLAGP	

Extended Description	Routine	Label

Figure 2-23.2. Process Dequeue Requests (Part 2 of 2)

INPUT

RRD
- RRDNQUP
- RRDNQRO

PROCESS

4. If any counts are left at lower levels:
   - A. Lower current level of ownership by task.
   - B. Promote any new owners made eligible by the decrease in level.
   - C. Go to step 6.

5. If no counts are left at any level:
   - A. Release the resource.
   - B. Promote any new owners made eligible by the release.

6. Exit

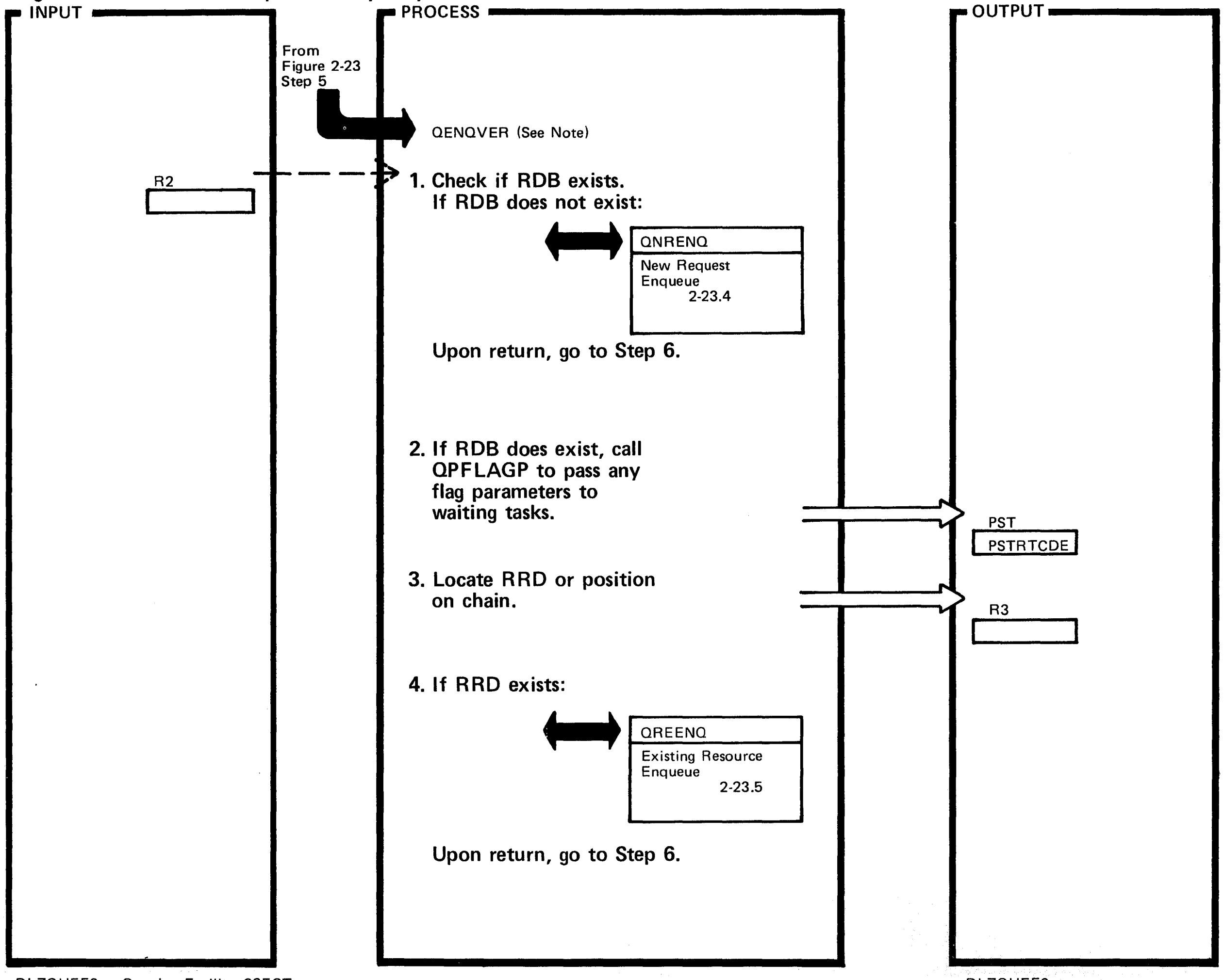
To Figure 2-23 Step 4

OUTPUT

Extended Description	Routine	Label
4.	QDEQVER3 QDEQVER4 QDEQVER5 QPNDWCM	
5. Entry point QDEQPUR in this routine is used by 'purge' processing to release a resource already unchained from the PST/RRD chain.	QRELRSC	
6.	QDEQVERX QRELRSCX QDEQPURX	

Extended Description	Routine	Label

Figure 2-23.3. Process Enqueue/Verify Requests (Part 1 of 2)



DLZQUEFO – Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
Note: This routine is used to process enqueue and verify requests. It determines the type of enqueue (or verify) and enters the proper routine for the type. The possible types are: • New resource enqueue – The enqueue of a resource not currently enqueued. • Existing resource enqueue – The enqueue of a resource currently enqueued, but not by this task. • Re-enqueue – The enqueue of a resource currently enqueued by this task. Processing of enqueue and verify requests is essentially the same. The difference is that on the return from verify processing, the user is not the owner of the resource.		
1.	QENQVER	
2.	QPFLAGP	
3.	QLOCRRD	

Extended Description	Routine	Label

Figure 2-23.3. Process Enqueue/Verify Requests (Part 2 of 2)

INPUT

PROCESS

5. If RRD does not exist:

QERENQ
Re-enqueue 2-23.6

6. Exit

To Figure 2-23 Step 5

OUTPUT

DLZQUEFO — Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
6.	QENQVERX	

Extended Description	Routine	Label

Figure 2-23.4. New Request Enqueue

INPUT

PST

PSTFNCTN

From Figure 2-23.3 Step 1

PROCESS

QNRENQ (See Note)

1. If the function is 'verify', processing is complete at this point because the resource is available. Go to Step 3 to return to caller.

2. If the function is 'enqueue':

   A. Build and initialize an RDB and chain it on the end of the proper RDB queue.

   B. Build and initialize a RRD and attach it to both the RDB and PST RRD chains.

   C. Define the task as a resource owner.

3. Exit.

To Figure 2-23.3 Step 1.

OUTPUT

R2 R3

R3

'New' RRD Address

DLZQUEFO — Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
Note: This routine is used to process an enqueue (or verify) request for a resource that has no current enqueues outstanding.		
2. A. R2 points to RDB chain location. R3 points to RRD chain head in RDB.	QBLDRDB	
B.	QBLDRRD	
C.	QDASOWN	
3.	QNRENQX	

Extended Description	Routine	Label

Figure 2-23.5. Existing Resource Enqueue

INPUT

RRD: RRDMAXL
PST: PSTQLEV

PST: PSTFNCTN

RDB: RDBMAXL
PST: PSTQLEV, PSTFNCTN

PROCESS

From Figure 2-23.3 Step 4

QREENQ (See Note)

1. If the requested level is not higher than the current maximum for the task, the resource is available. Therefore:

   A. If the requested function is 'verify' go to Step 5.

   B. Otherwise, increment ownership count for the level by one. Go to Step 5.

2. If the requested level is compatible with the current maximum for the resource and if the requested function is

   A. 'Verify' – Go to Step 5.

   B. 'Enqueue' – Define the task as an owner at the new level and go to Step 5.

3. Otherwise, do deadlock detection and wait for resource availability.

4. If function is 'verify', dequeue the enqueued resource.

5. Exit.

To Figure 2-23.3 Step 4

OUTPUT

RRD: RRDNQRO, RRDNQUP, RRDNQEX

DLZQUEFO – Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
Note: This routine is used to process an 'enqueue' or 'verify' for a resource that is currently enqueued by the requesting task.		
2.	QREENQ1 QREENQ4	
B.	QREENQ1 QREENQ5	
3. The task will be defined as an owner of the resource during 'dequeue' processing for other task. The QDLKDTN routine detects a deadlock condition and resolves the deadlock by picking and terminating one of the tasks involved. The task terminated is selected as follows:	QREENQ3 QREENQ4 QWAIT QDLKDTN	

Extended Description	Routine	Label
1. Online tasks are picked before MPS tasks. 2. Within a class, the task with the fewest resources currently enqueued is chosen. 3. In the event of a tie, the choice is arbitrary.		
4.	QREENQ3 QREENQ4 QDEQVER	
5.	QREENQX	

Figure 2-23.6. Re-enqueue

INPUT

RDB: RDBMAXL

PST: PSTQLEV, PSTFNCTN

From Figure 2-23.3 Step 5

PROCESS

QERENQ (See Note)

1. If resource is available at requested level:
   - A. If function is 'verify', go to Step 3.
   - B. Build RRD and chain to RDB and PST.
   - C. Define the task as a resource owner.
   - D. Go to Step 3.
2. If resource is not available at requested level.
   - A. Build RRD and chain to RDB and PST.
   - B. Do deadlock detection and wait for resource availability.
   - C. If function is 'verify', dequeue the enqueued resource.
3. Exit.

To Figure 2-23 Step 5

OUTPUT

DLZQUEFO – Queuing Facility CSECT

DLZQUEFO

Extended Description	Routine	Label
Note: This routine is used to process an 'enqueue' or 'verify' request for a resource that is currently enqueued, but not by the requesting task.		
1.	QERENQ	
B. To build and chain RRD, call QBLDRRD.	QBLDRRD	
C. To make task a resource owner, call QDASOWN.	QDASOWN	
2.	QERENQ2	
A. To build and chain RRD, call QBLDRRD.	QBLDRRD	
B. The task will be defined as an owner of the resource, during 'dequeue' processing for other tasks.	QWAIT QDLKDTN	

Extended Description	Routine	Label
The QDLKDTN routine detects a deadlock condition and resolves the deadlock by picking and terminating one of the tasks involved. The task terminated is selected as follows: 1. Online tasks are picked before MPS tasks. 2. Within a class, the task with the fewest resources currently enqueued is chosen. 3. In the event of a tie, the choice is arbitrary.		
C.	QERENQ3 QDEQVER	
3.	QERENQX	

Figure 2-24. Visual Table of Contents for DL/I Utility Modules HIPO Charts

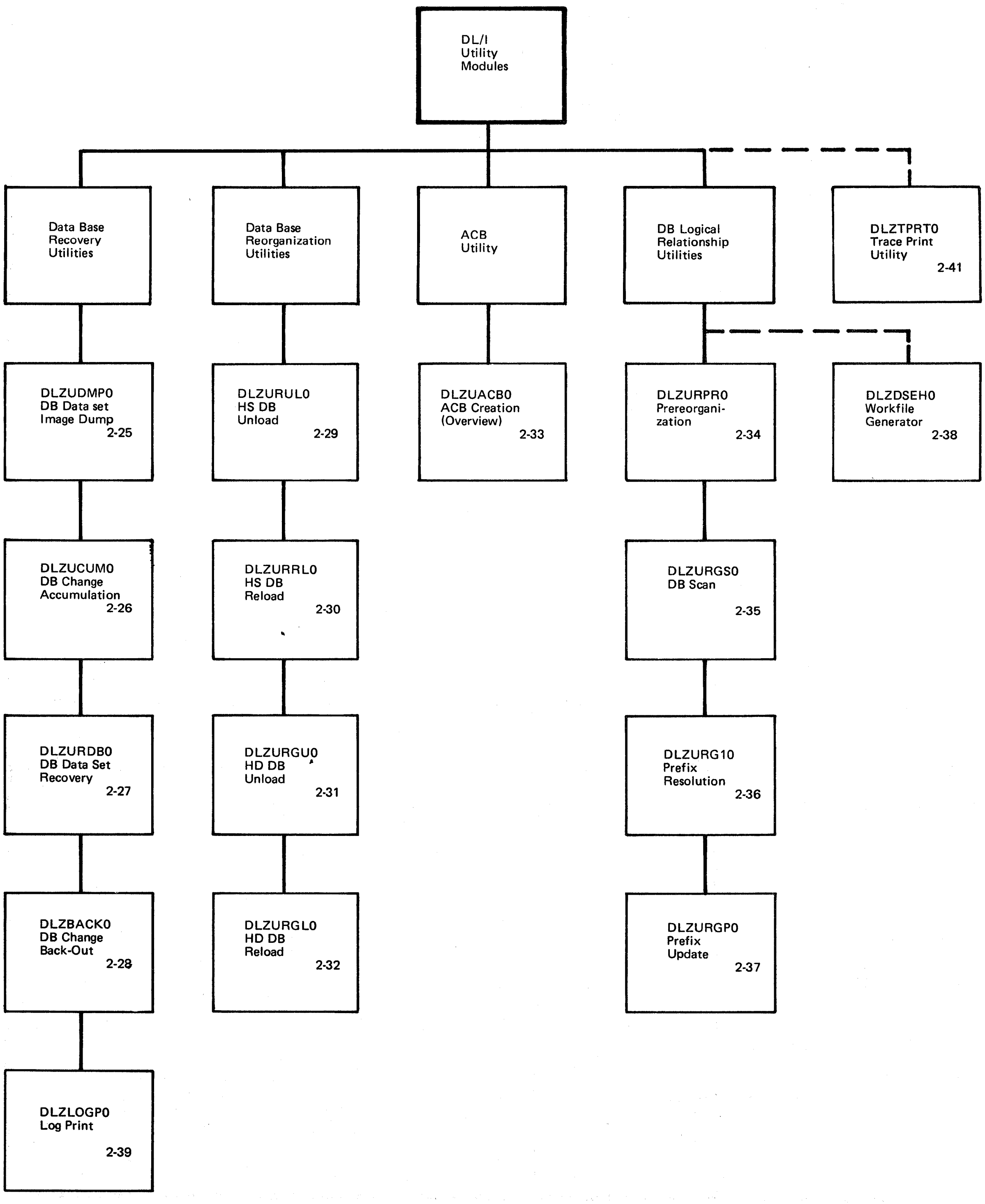
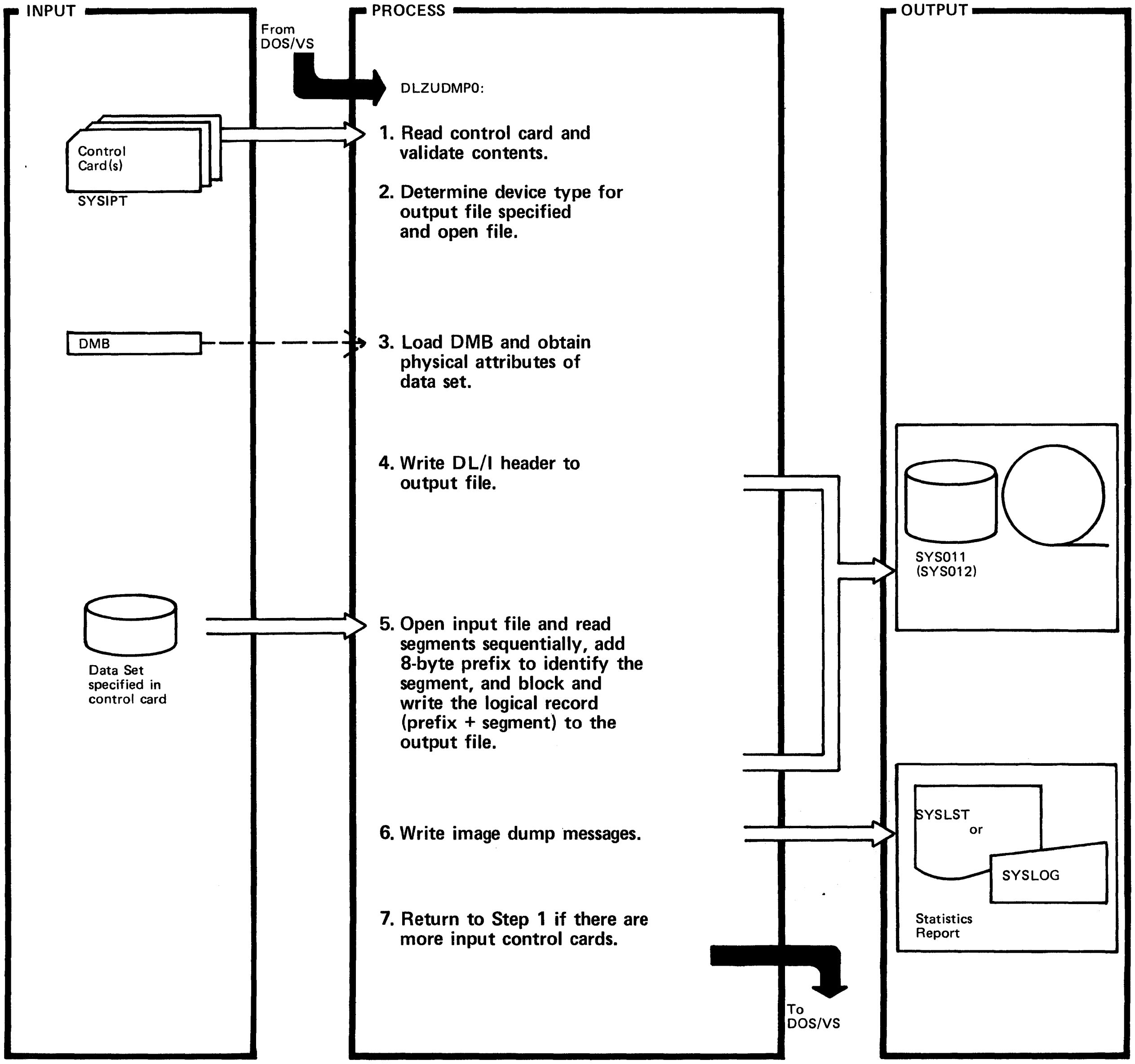


Figure 2-25. DB Data Set Image Dump



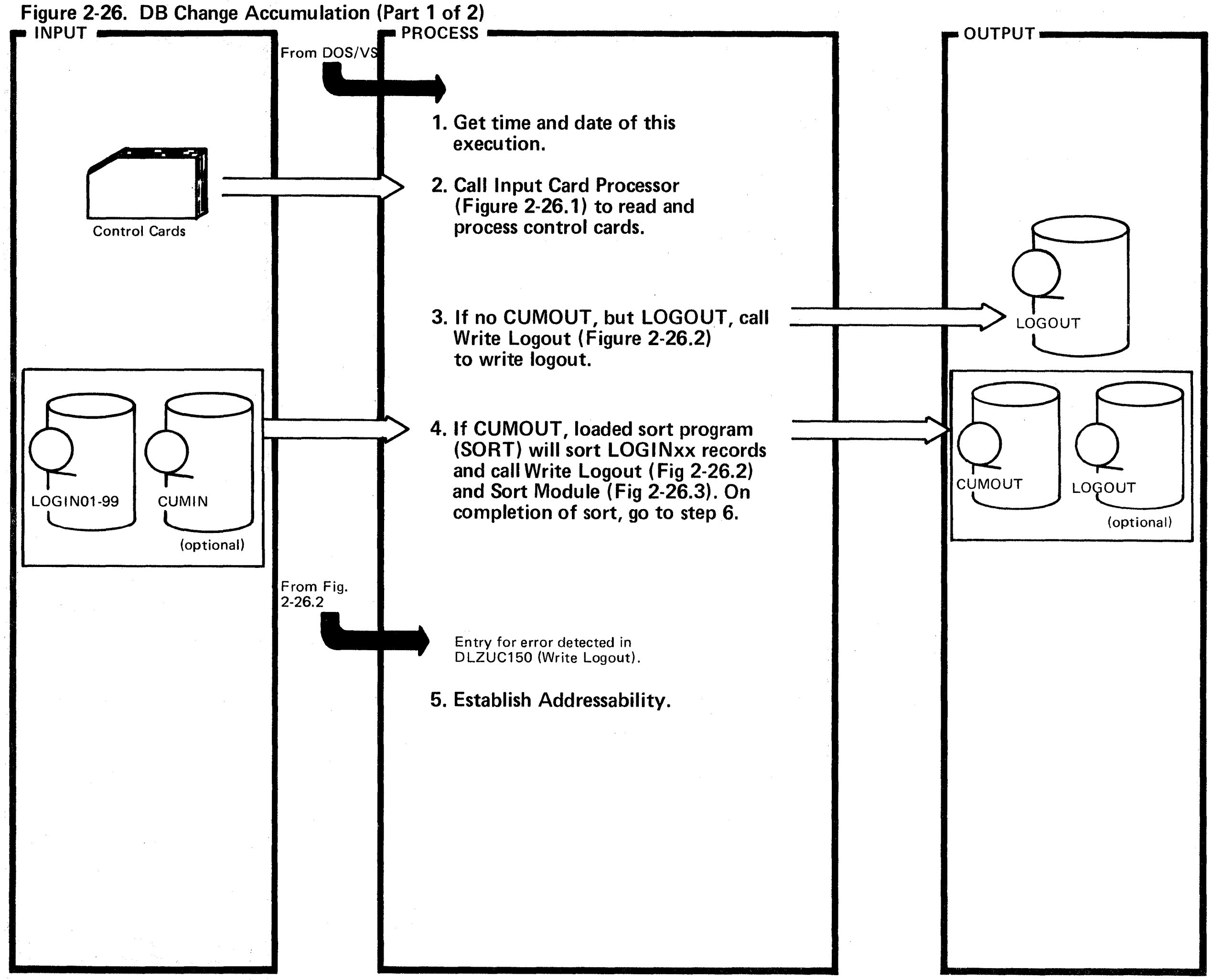
DLZUDMPO — DB Data Set Image Dump Utility

DLZUDMPO

Extended Description	Routine	Label
1. Read and validate control statement. Write the following messages as needed: DLZ302I - Column 1 not D DLZ303I - Column 2 not 1 or 2 DLZ304I - DBD name field not specified DLZ307I - Input filename not specified DLZ308I - Output filename not specified DLZ309I - Error(s) found in control statement DLZ310I - Image of erroneous control statement		
2. DLZDVCE macro obtains data from PUB. Device type may be tape or DASD.		
4. The header record contains information that allows the use of the image dump file by the DB data set recovery utility.		

Extended Description	Routine	Label

Figure 2-26. DB Change Accumulation (Part 1 of 2)



DLZUCUM0 – Change Accumulation Utility

DLZUCUM0

Extended Description	Routine	Label
1. Header line is printed on SYSLST.		TIMEDEC
2. Three returns as follows:		READCD
• Error – issue error message.		BADEND
• No accumulation output, call Write Logout (Fig 2-26.2). Then issue successful run message.		GOODEND
• Accumulation output, call SORT.		SORT
4. SORT is invoked by LOAD and BALR. At exit 35, Sort Module (Fig 2-26.3) is called.		SORT
5. This entry point is necessary because Write Logout, not knowing who called (DB Change Accumulation or SORT), must return to this module if an error was detected.		DLZERRTN

Extended Description	Routine	Label

Figure 2-26. DB Change Accumulation (Part 2 of 2)

INPUT

PROCFLAG

PROCESS

6. Call Write Messages (Figure 2-26.4) to write completion message.

7. Exit.

Return to DOS/VS

OUTPUT

SYSLST SYSLOG

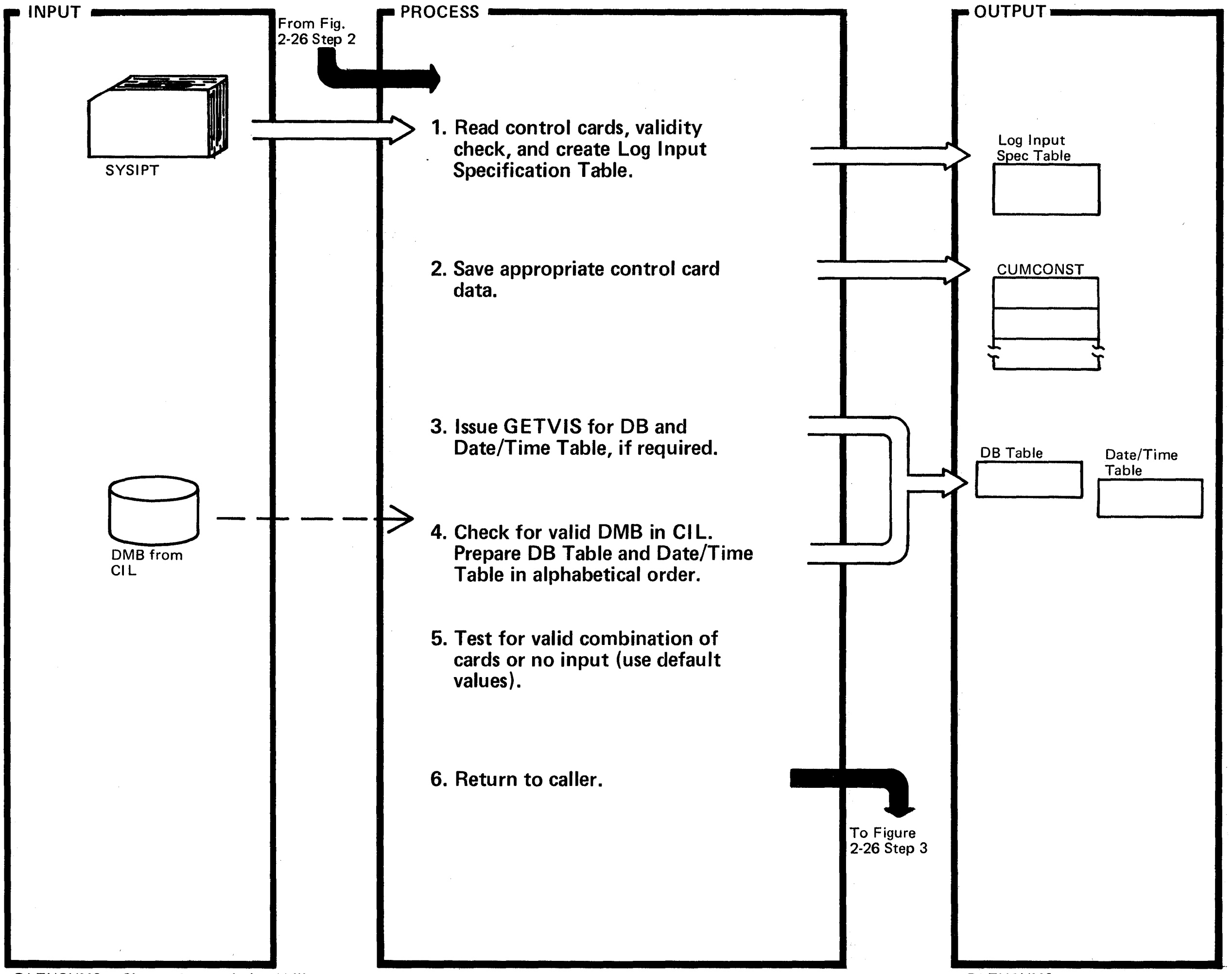
DLZUCUMO — Change Accumulation Utility

DLZUCUMO

Extended Description	Routine	Label
6. May be OK message or error message from SORT, Write Logout, or Sort Module. If PROCTERM X'01' bit is on in PROCFLAG, an error occured.		CLOSE

Extended Description	Routine	Label

Figure 2-26.1. Input Card Processor (DLZUCCTO)



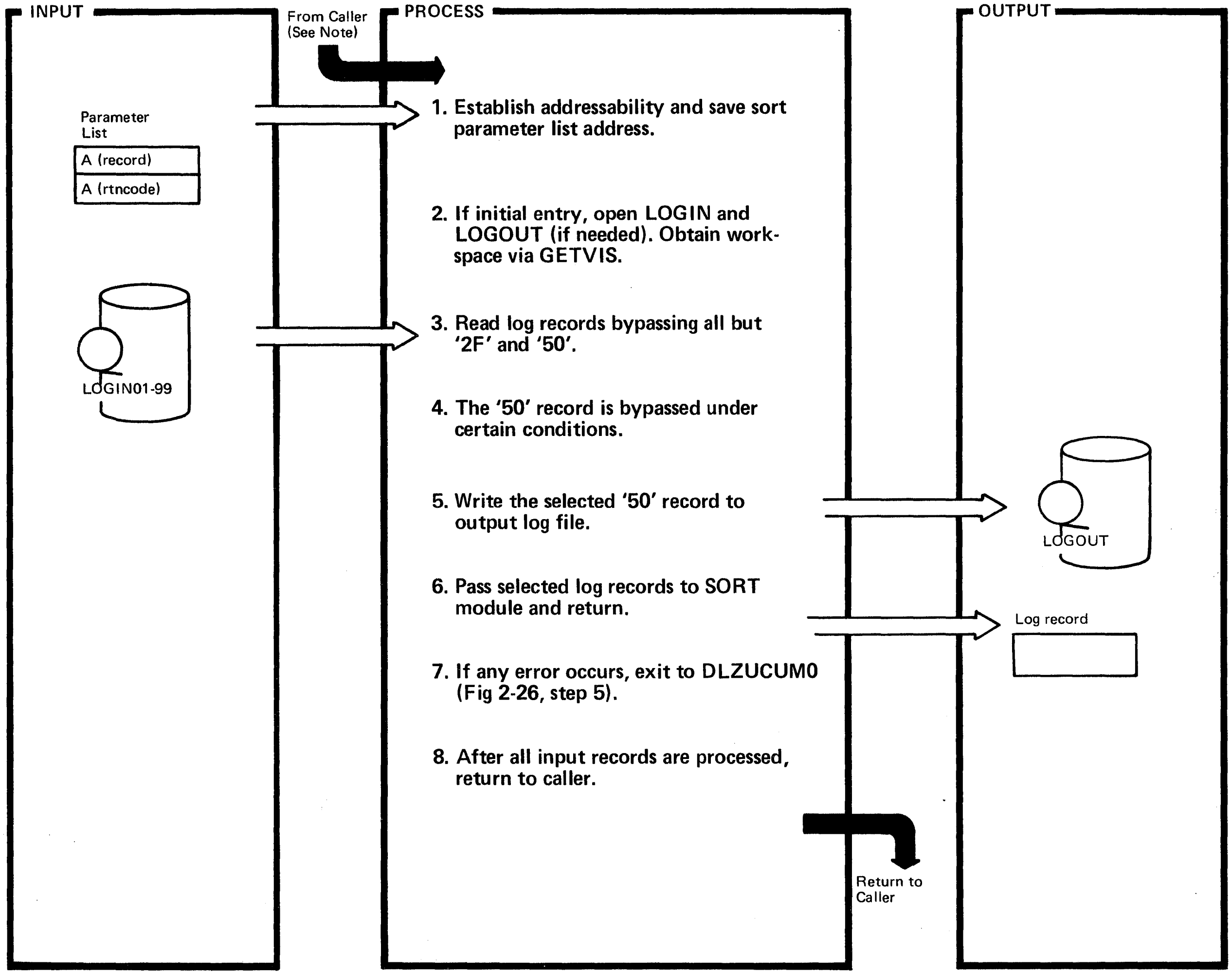
DLZUCUMO – Change Accumulation Utility

DLZUCUMO

Extended Description	Routine	Label
1. Possible card types are:		GETCARD
• 'ID' specifies db number, max key length, number of sort, work, and log files.		
• 'DBO' describes records to be accumulated from input and written to CUMOUT.		
• 'DB1' describes records to be written to new log file.		
• 'LI' describes a log input file.		
• Error card – call Write Message (Fig 2-26.4) to write appropriate error message.		ERROR
2. Data from control card(s) is saved in a dsect residing in DLZUCUMO, addressable by all modules in this utility. The dsect name is DLZUCUMC.		

Extended Description	Routine	Label
3. Tables are not required if *ALL was specified. The number of entries in each table is equal to the number of data bases as specified on the ID control card or default of 16.		GETMAIN
4. This information is filled from the DBO and/or DB1 card(s) if present.		DDNUMCHK
5. If any errors occur during steps 3, 4, of 5, call Write Messages (Fig 2-26.4) to write error messages and exit.		

Figure 2-26.2. Write Logout (DLZUC150)



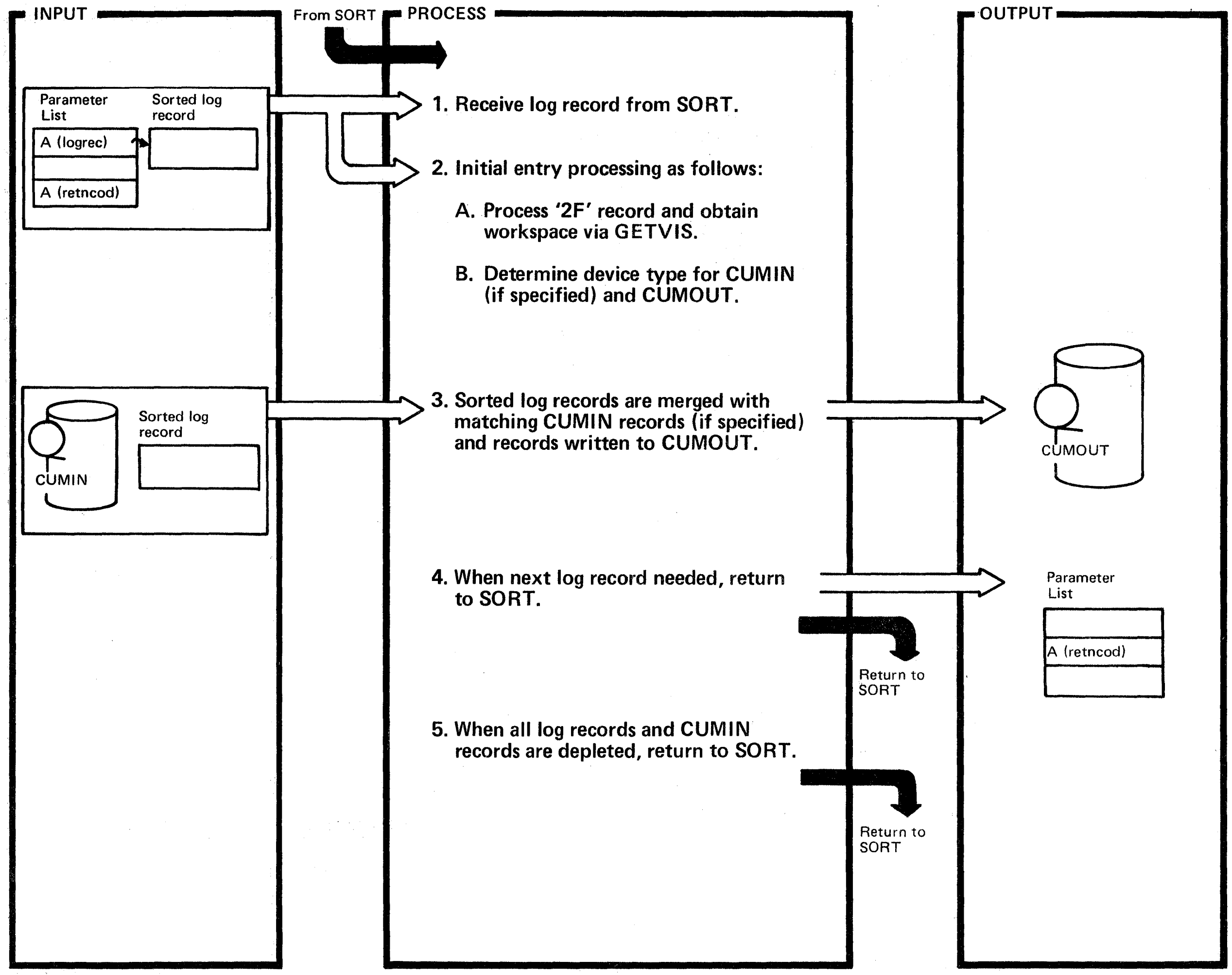
DLZUCUM0 – Change Accumulation Utility

DLZUCUM0

Extended Description	Routine	Label
Note: This program has two entry points: • DLZUC150 – from SORT. Entered when SORT wants another input record. • DLZUEX15 – from Figure 2-26, step 3 (DLZUCUM0).		
3. On EOF, the file is closed. If more input specified, xx (LOGINxx) in the DTF or ACB is incremented by 1 and the next log file is opened.		
4. Bypass '50' record for the following: • *ALL and log date/time less than purge date/time. • dbname match and log date/time less than purge date/time. • No dbname match and *OTHER not specified.		

Extended Description	Routine	Label
4. (con't) • No dbname match and log date/time less than purge date/time.		
5. Write log record for the following '50': • *ALL on DB1 card. • Dbname match and dbname on DB1 card. • No dbname match and *OTHER on DB1 card.		

Figure 2-26.3. Sort Module (DLZUC350)



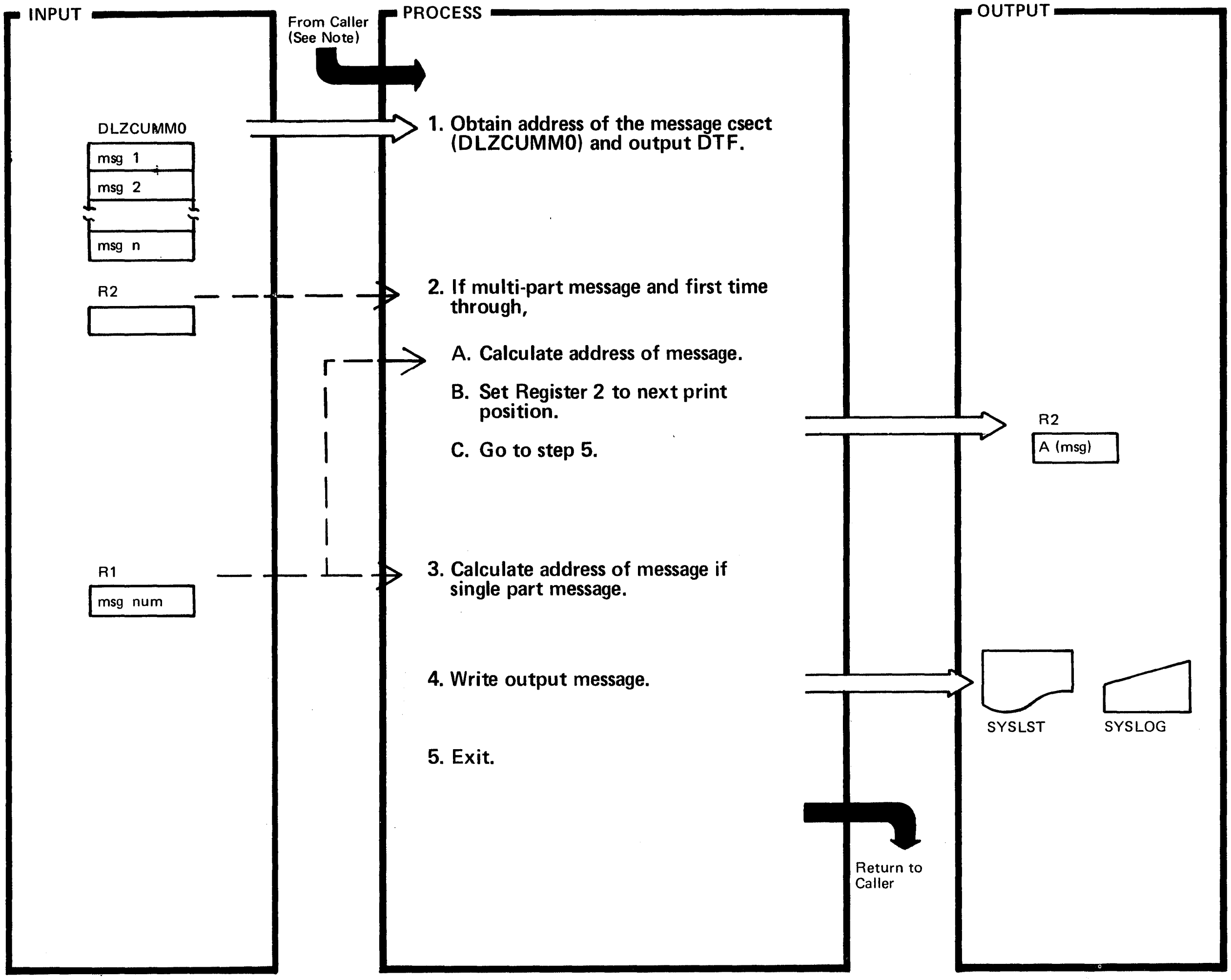
DLZUCUMO – Change Accumulation Utility

DLZUCUMO

Extended Description	Routine	Label
1. SORT returns at EOF with an indication that no more records exist.		DLZUEX35
2. DLZDVCE macro obtains data from PUB. Device type may be TAPE or DASD.		TSTEODDB
3. The following merging logic is used for comparison of LOGIN and CUMIN to create CUMOUT. ● For every new DMB name (data set ID), an accumulation header record is written either from the CUMIN record or created from the '2F' record. ● Every CUMIN record is purge checked by date/time as specified by the user. The DB table as modified by DLZUC150 is used for a specific DMB or the *ALL/ *OTHER purge date is used as applicable. ● If a matching log record is found, all log records with the same data ID will be merged with the CUMIN		

Extended Description	Routine	Label
3. (con't) record and written to CUMOUT. ● If there is no matching log record, the CUMIN record is written to CUMOUT unchanged. ● If log records exist but no CUMIN, the log records are accumulated by data ID and written to CUMOUT.		
4. A 'delete' return code is given to SORT so that SORT does not further process the current record. SORT will prepare the next input record and enter this program at step 1.		
5. Free all work areas and close CUMIN and CUMOUT.		ENDJOB
Indicate 'no return' to SORT.		ENDSORT

Figure 2-26.4. Write Messages (DLZUCER0)



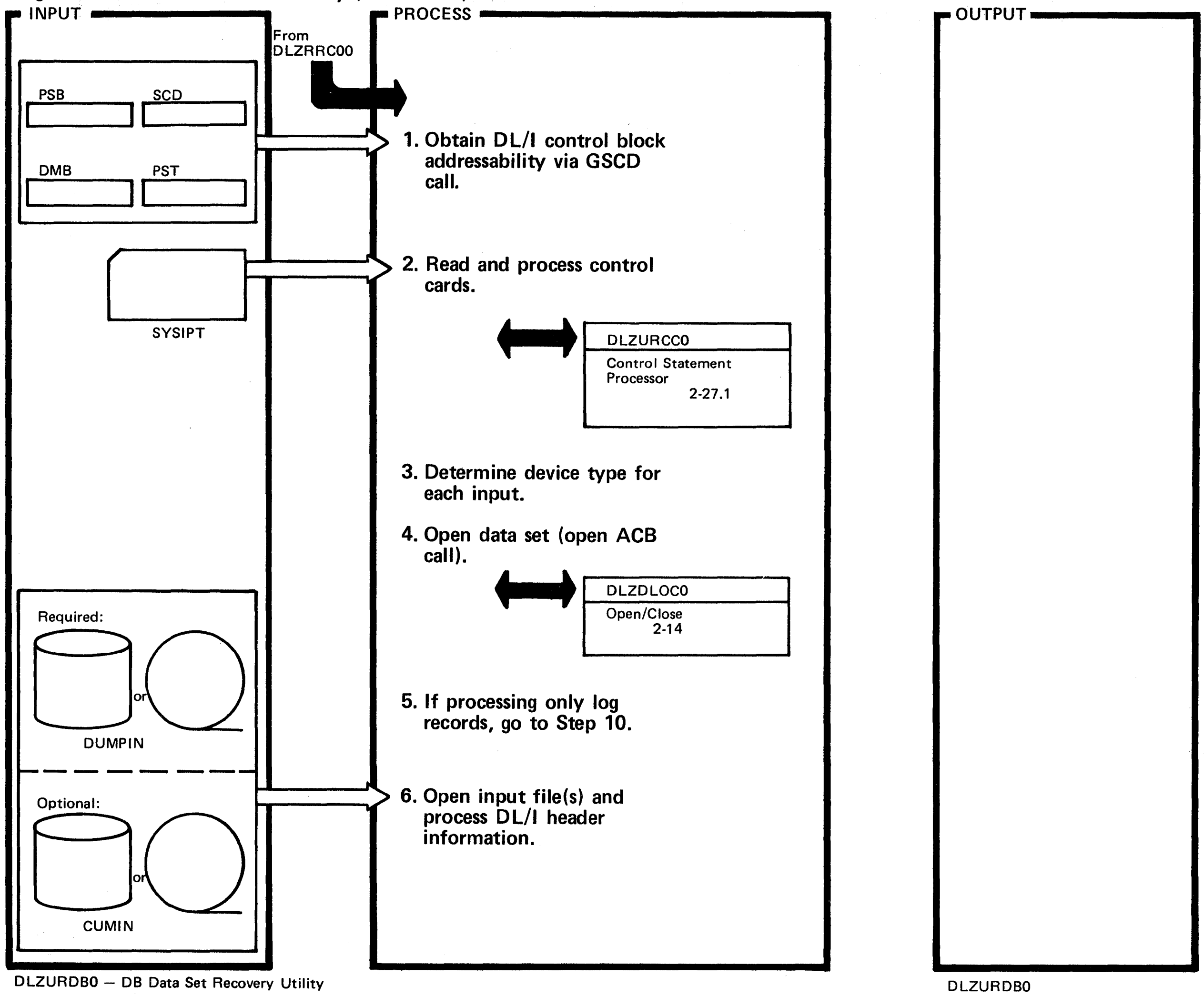
DLZUCUMO — Change Accumulation Utility

DLZUCUMO

Extended Description	Routine	Label
Note: This module can be called by DLZUCUMO, DLZUCCTO, DLZUC150, or DLZUC350.		
1. The address of the output DTF which has already been opened, is found in the CUMCONST table.		INITSV
2. Multi-part message is indicated by negative R2.		TESTR2
3. R1 contains message number.		MSGCOMM
4. Output can be to SYSLST or SYSLOG.		MSGWRT
5.		RETURN

Extended Description	Routine	Label

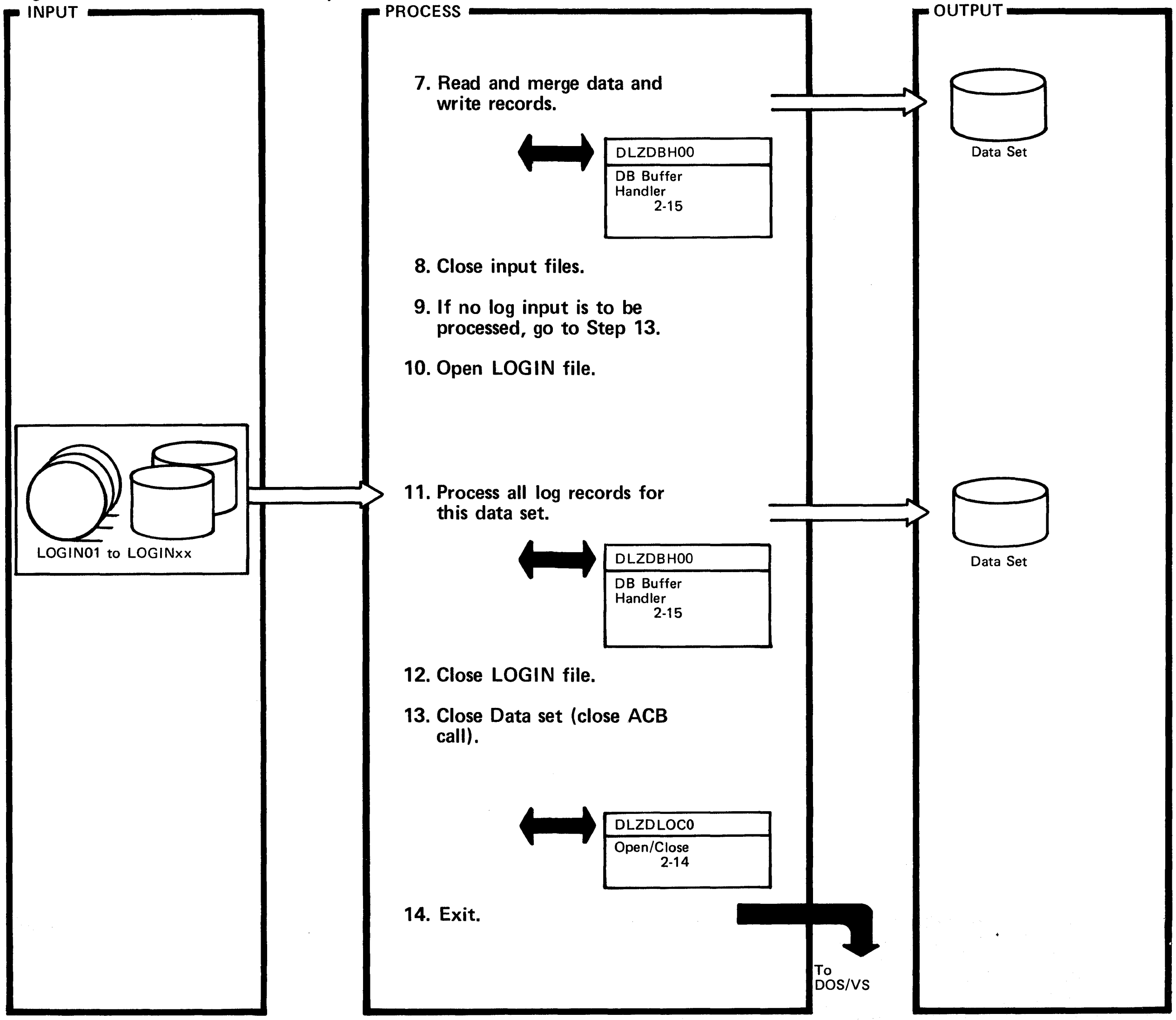
Figure 2-27. DB Data Set Recovery (Part 1 of 2)



Extended Description	Routine	Label
2. There are three returns:		
• No errors – continue processing.		GETDMB
• No input statements – terminate processing.		CLEANUP
• Error – issue error message and terminate processing		BADRUN
3. DLZDVCE macro obtains data from PUB. Device type may be tape or DASD.		
6. DUMPIN file is mandatory and may be output from DLZUDMP0 or DLZURUL0.		
CUMIN file is optional and is output from DLZUCUM0.		

Extended Description	Routine	Label

Figure 2-27. DB Data Set Recovery (Part 2 of 2)



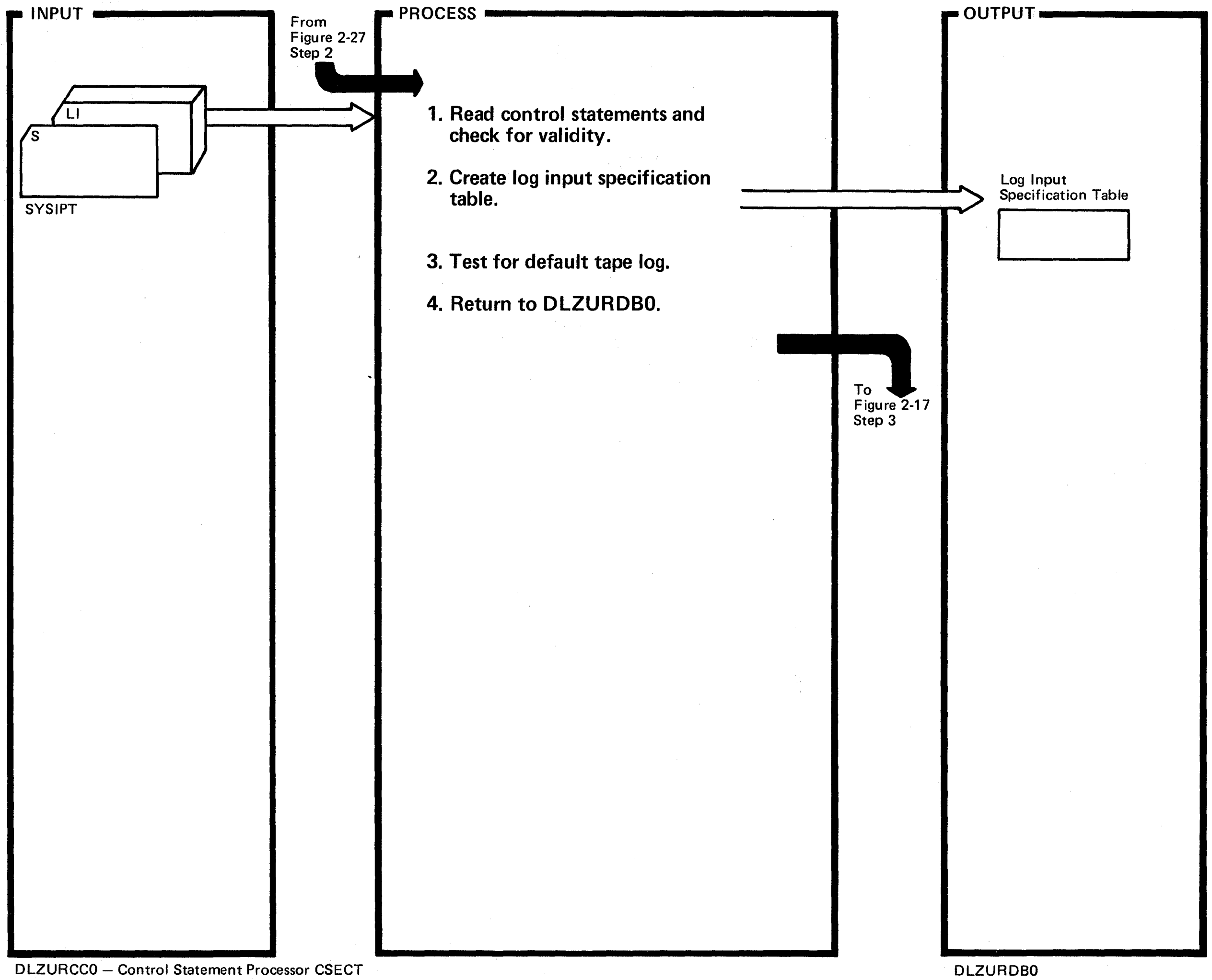
DLZURDB0 — DB Data Set Recovery Utility

DLZURDB0

Extended Description	Routine	Label
7. Records are read from DUMPIN and CUMIN via GET calls and are written in ascending order (compare by key if KSDS, and by RBA if ESDS). The proper PSTFNCTN is supplied for call of buffer handler.		SETFLOW
9. LOGIN file is optional.		
11. LOGIN01 to LOGINxx files are processed sequentially.		PROCLOGS

Extended Description	Routine	Label

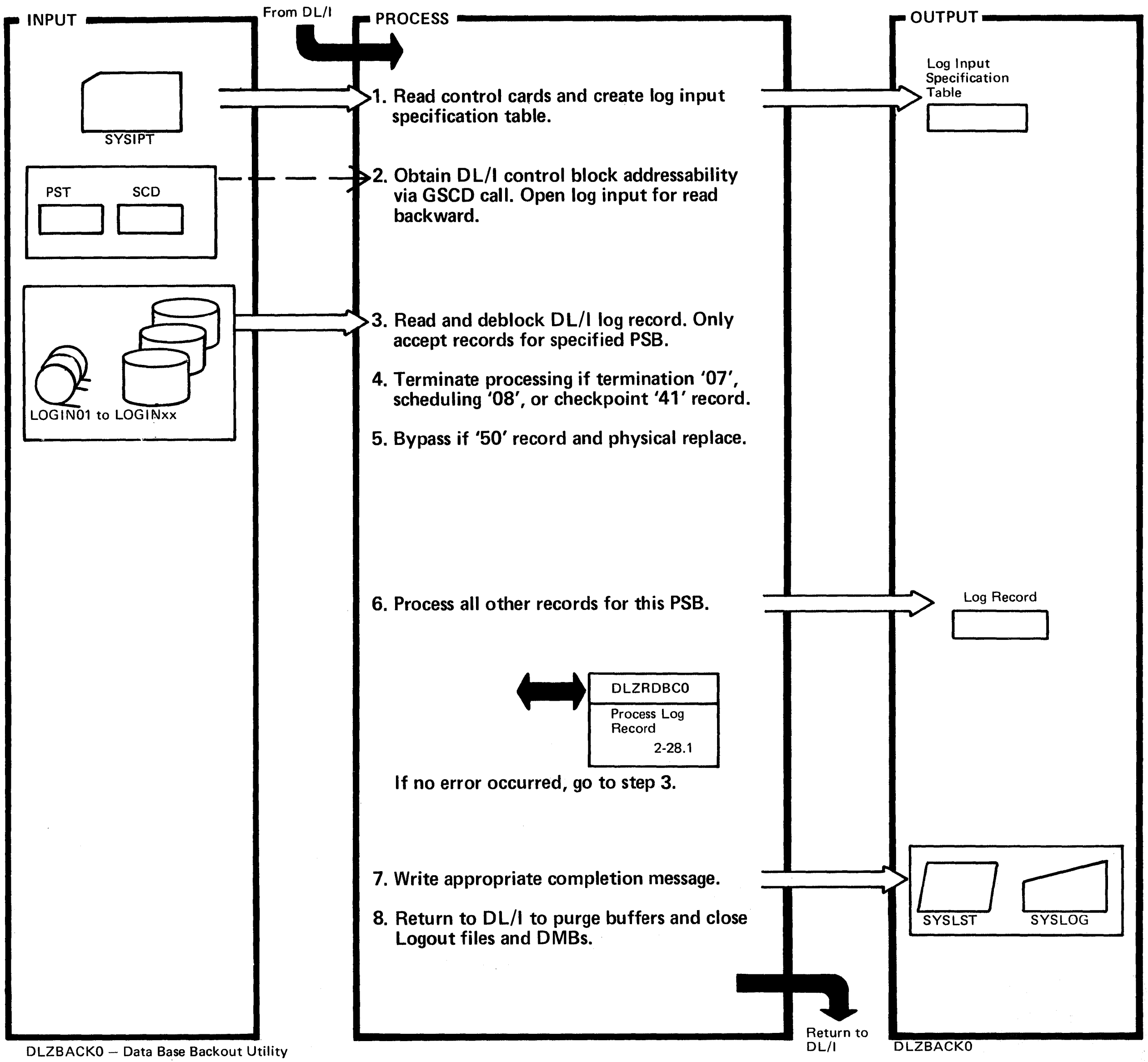
Figure 2-27.1. Control Statement Processor



Extended Description	Routine	Label
1. Possible card types are: 'S' – identifies data set to be recovered. 'LI' – describes log input file(s). Write the following messages as needed: DLZ302I - Column 1 not S DLZ304I - DBD name not specified DLZ307I - Input filename not specified DLZ310I - Image of erroneous control statement DLZ342I - Invalid number of log files DLZ372I - Invalid log buffer size	DLZURCC0	GETCARD
2. One entry in table describing file type, logical unit, and buffer size for each log file.		
3. If no log file is specified, issue macro DLZDVCE to see if SYS013 assigned to tape.		CLEANUP

Extended Description	Routine	Label

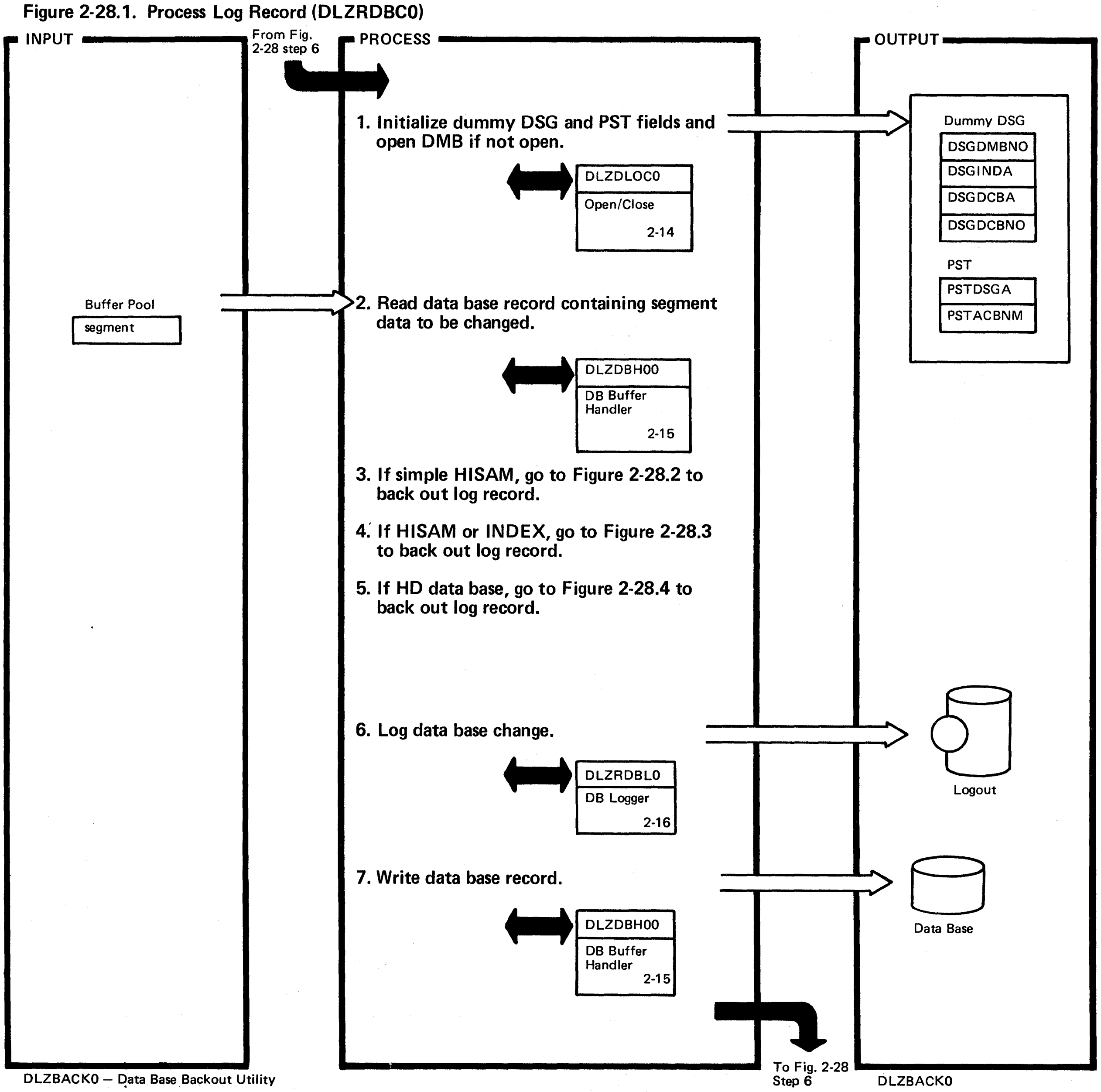
Figure 2-28. DB Change Backout



Extended Description	Routine	Label
1. 'LI' control cards describe one input log file each.		
2. Initialize PSTDBPCB, PSTDGU, and PSTDGN.		INIT
3. At end of file, go to step 7.		READ NXTLREC
4.		CHKLOGT
5.		CHKDPHYR
6. The log record is placed in a work area (READAREA) whose address DLZRDBCO obtains via a V-con.		OK CALLBO
7. The input log file is closed. If another log file exists, it is opened and processing continues with step 2. The message texts are found in DLZBACM0 csect.		EOF MSGGEN

Extended Description	Routine	Label

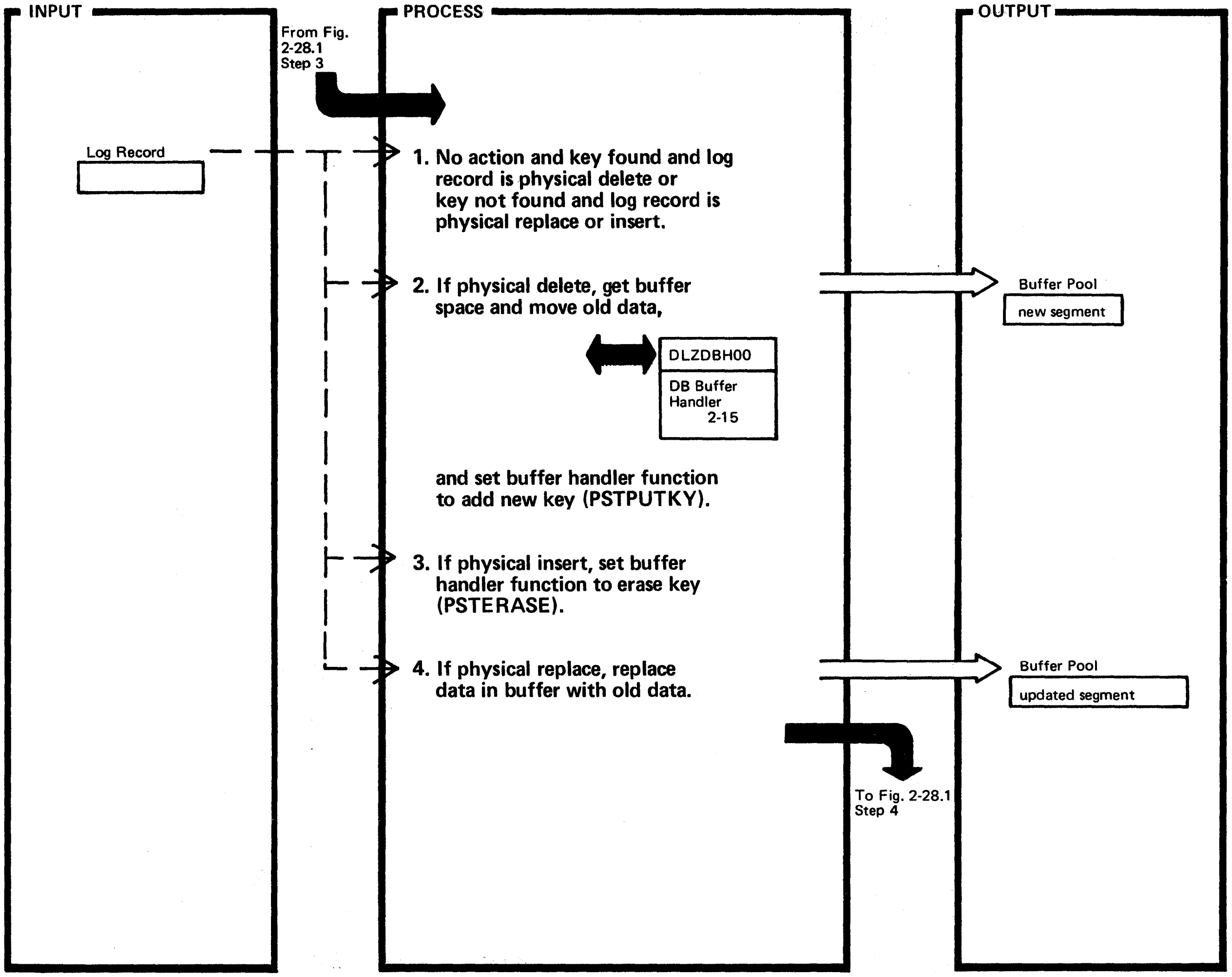
Figure 2-28.1. Process Log Record (DLZRDBC0)



Extended Description	Routine	Label
1.		INIT
2. The following calls were made to the buffer handler:		LOCDCB CALLBFRH
A. If HISAM KSDS, issue PSTSTLEQ call.		SETISAMC
B. If HISAM ESDS, issue PSTBYLCT call.		LOCBLK
C. If HD ESDS, issue PSTBKLCT call.		SETBLKLT
6. Output log records contain the 'opposite' function to which was on the input log.		CALLLOG LOG
7. The return code is checked and appropriate action is taken depending on the call and return code.		WRITEBFR

Extended Description	Routine	Label

Figure 2-28.2. Simple HISAM Backout (DLZRDBC0)



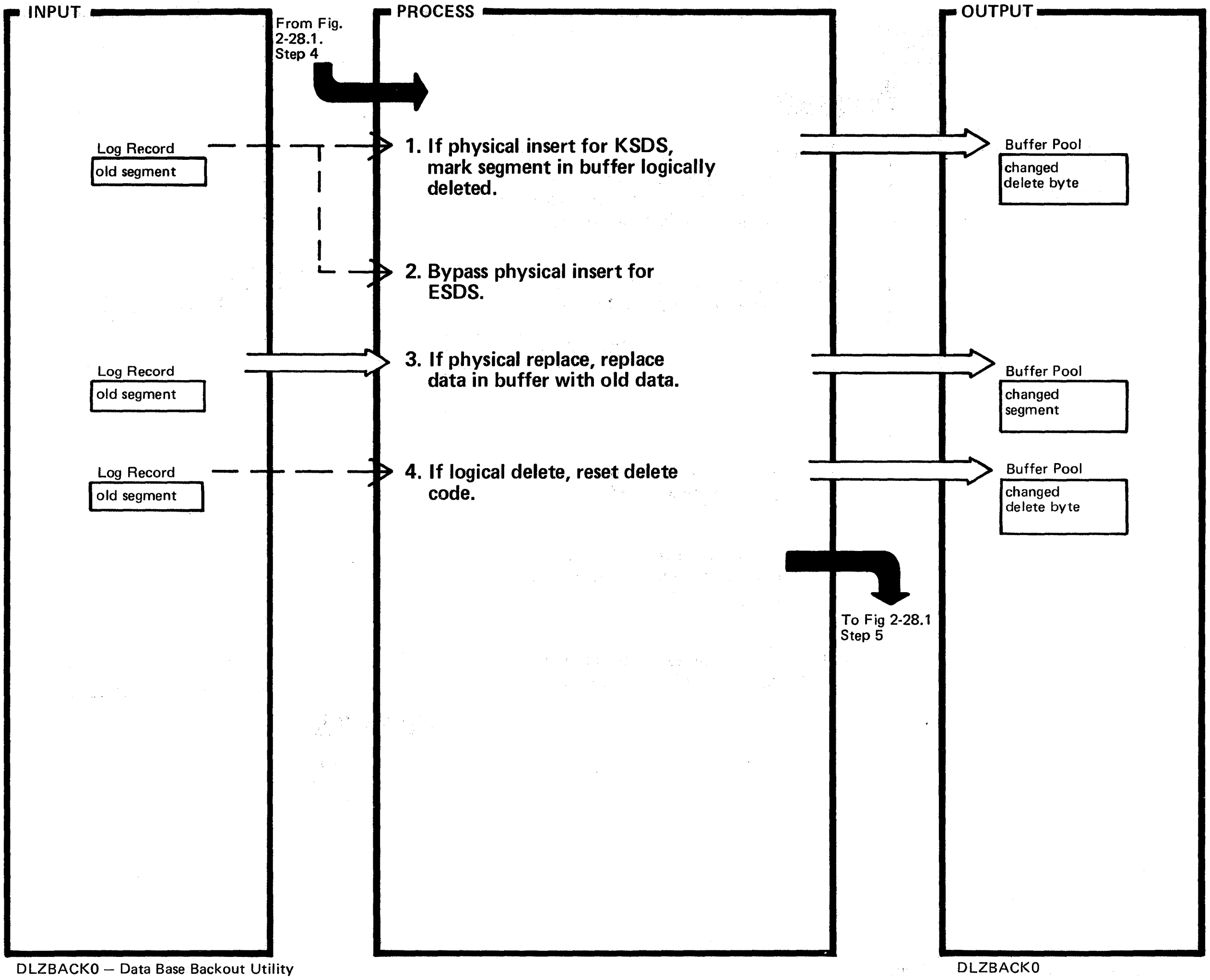
DLZBACK0 — Simple HISAM Backout Utility

DLZBACK0

Extended Description	Routine	Label
1. The address of the log record is input to this routine.		KEYNOTFD CKSHISAM
2.		KEYNOTFD
3.		CKSHISAM
4.		CALLREP

Extended Description	Routine	Label

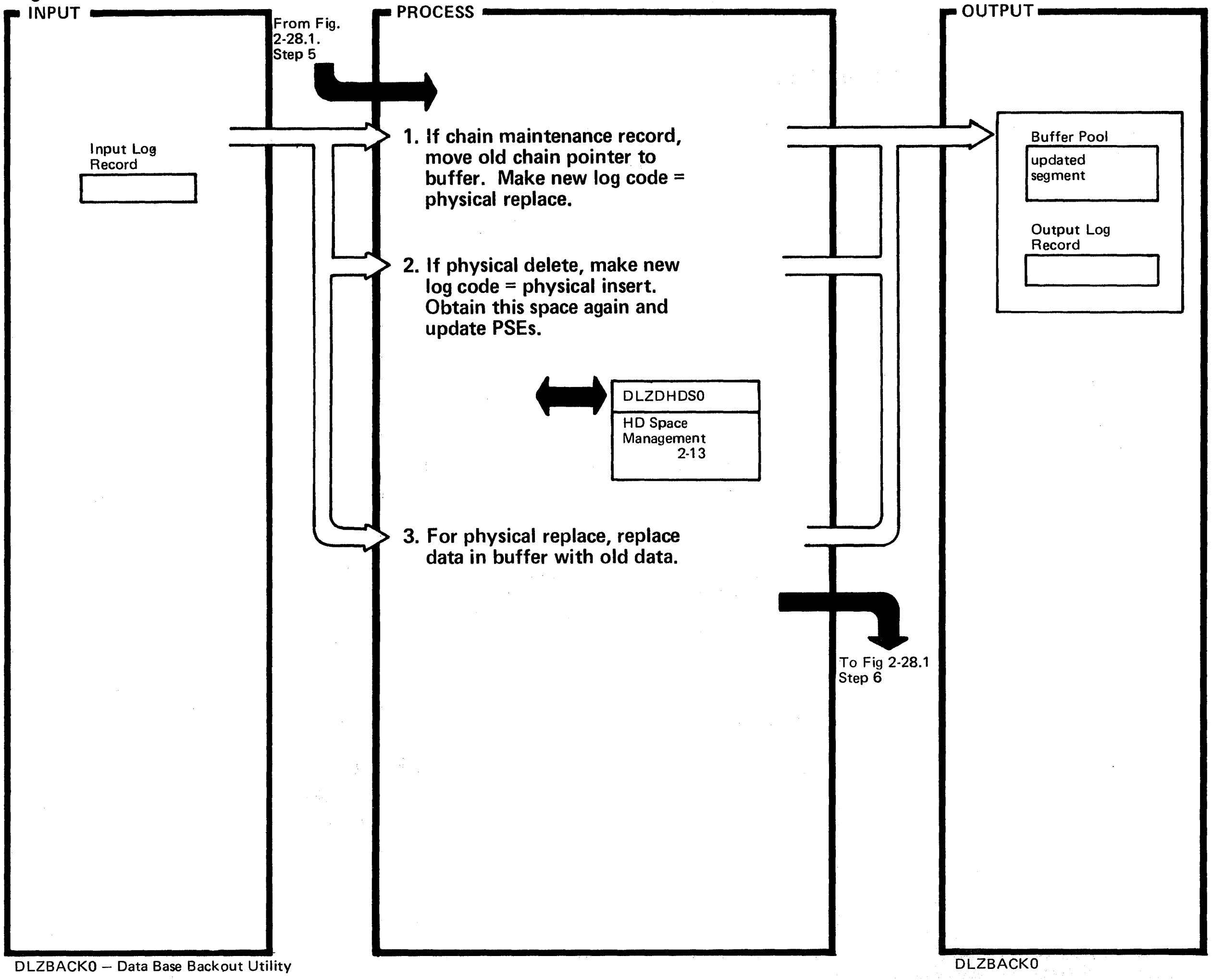
Figure 2-28.3. HISAM or INDEX Backout (DLZRDBC0)



Extended Description	Routine	Label
1. If segment is an INDEX data base (primary or secondary), the pointer to the index target segment is also zeroed.		CHKUSERI LOGDLET SETPHYRP
2. Chain maintenance log records for KSDS effectively back out physical insert to ESDS.		CHKUSERI
3.		CKDICALL
4.		CHKLGDLT

Extended Description	Routine	Label

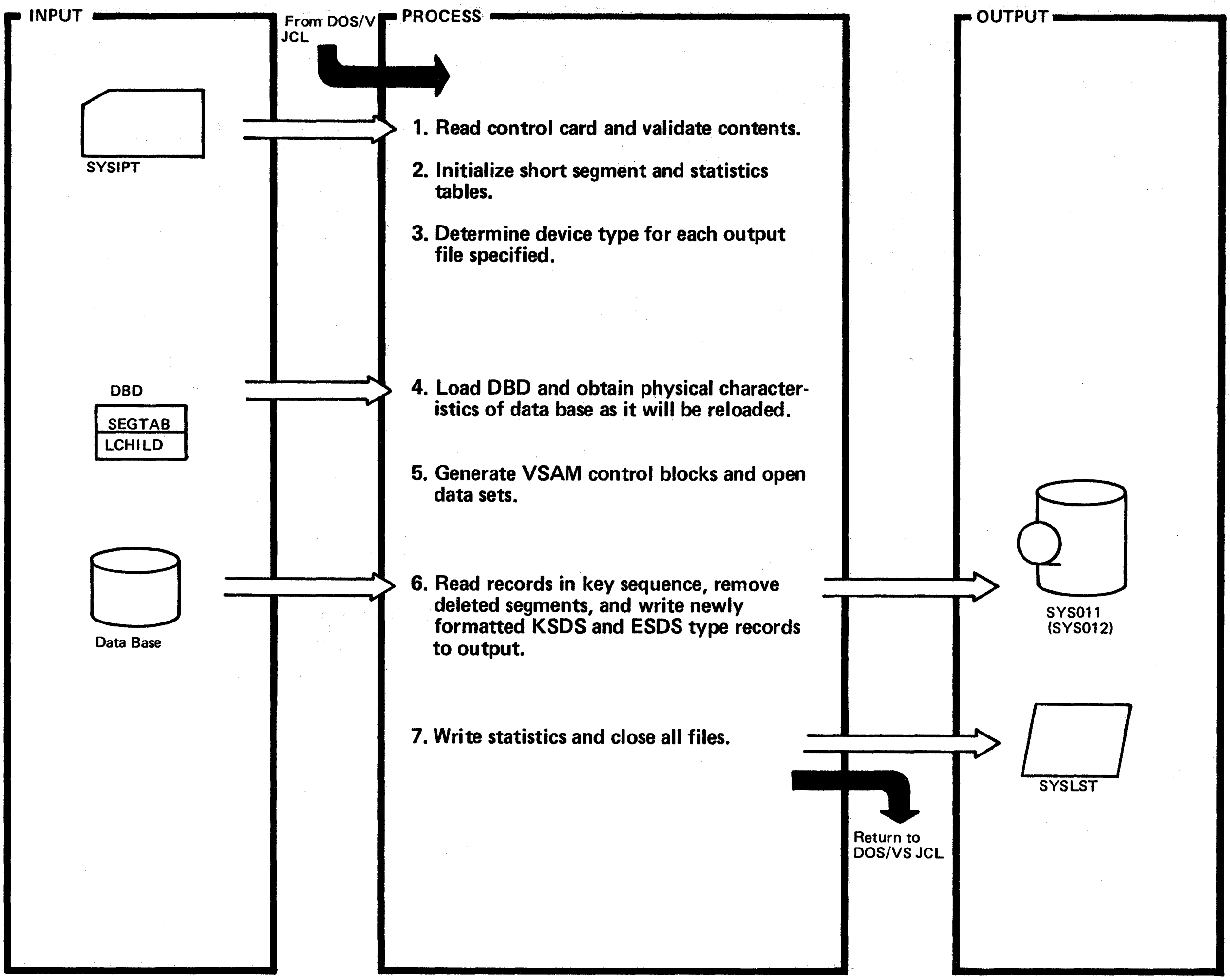
Figure 2-28.4. HD Backout (DLZRDBC0)



Extended Description	Routine	Label
1.		CHKCHAIN
2.		NEXTFSE NOTINFS LASTCOMP
3.		NOTINFS CHKREPPC

Extended Description	Routine	Label

Figure 2-29. HS DB Unload



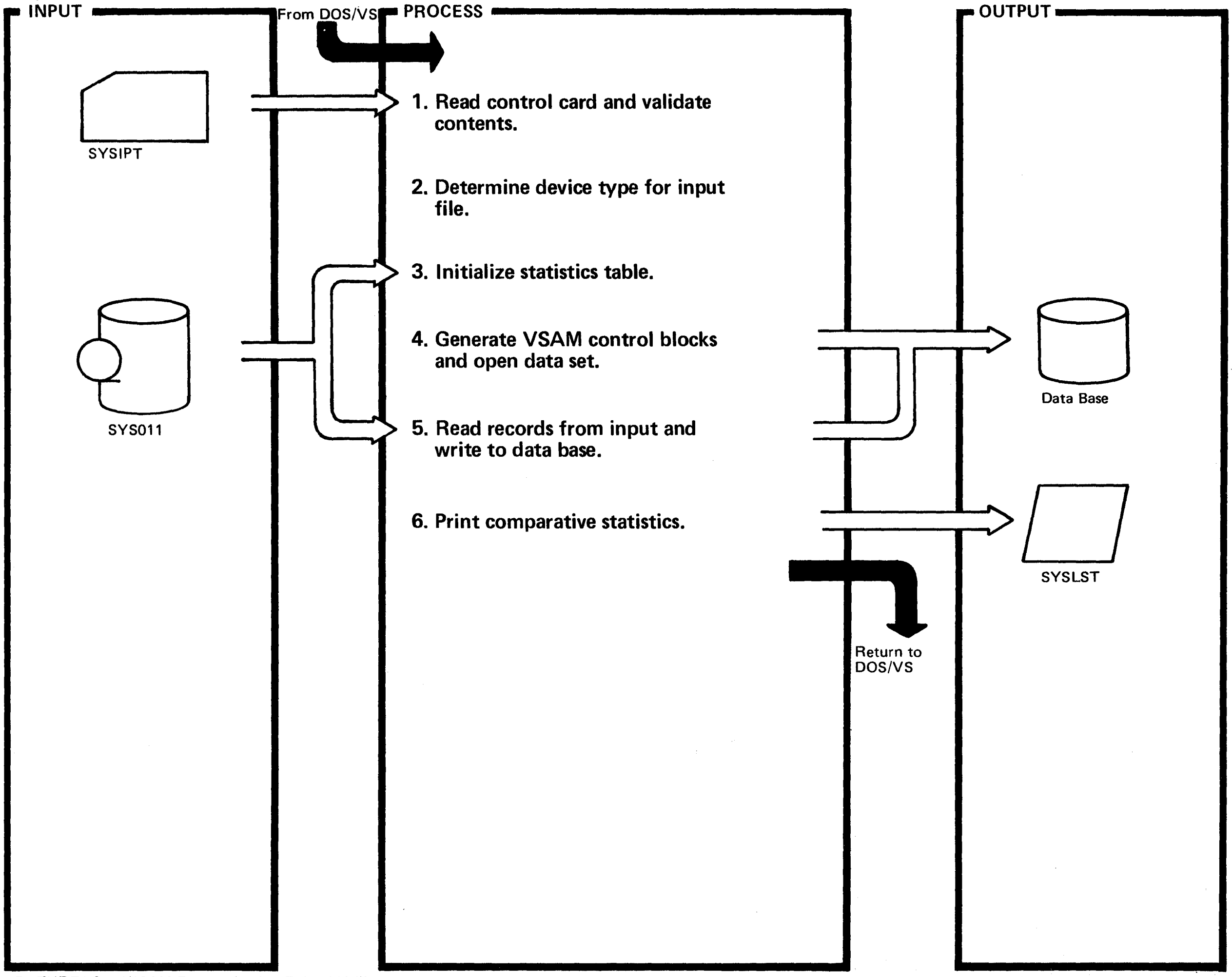
DLZURUL0 – HISAM Reorganization Unload Utility

DLZURUL0

Extended Description	Routine	Label
1. Read and validate control statement. Write the following messages as needed: DLZ302I - Column 1 not R DLZ303I - Column 2 not 1 or 2 DLZ304I - DBD name not specified DLZ307I - Input filename not specified DLZ308I - Output filename not specified DLZ309I - Error(s) found in control statement DLZ310I - Image of erroneous control statement		
3. DLZDVCE macro obtains data from PUB. Device type may be TAPE or DASD.		
5. Issue GENCB for ACB, RPL, and EXLST. Open KSDS and ESDS unless ACCESS=SHISAM (KSDS only).		
6. Processing as follows: A. Read KSDS records in key sequence, bypass if deleted. ESDS records containing overflow dependent segments are read by RBA. B. Format work area like KSDS record with new attributes.		

Extended Description	Routine	Label
6. (con't) C. Move as many segments as will fit into KSDS work area, bypassing deleted segments. Calculate overflow RBA. Write image of KSDS to output. D. Format work area like ESDS record with new attributes. E. Move any dependent segments as will fit into ESDS work area, bypassing deleted segments. Calculate RBA for next record, if required. Write image of ESDS to output.		
7. Statistics also written to SYS011 to be used for comparative purposes during reload. Processing will continue if additional input cards.		

Figure 2-30. HS DB Reload



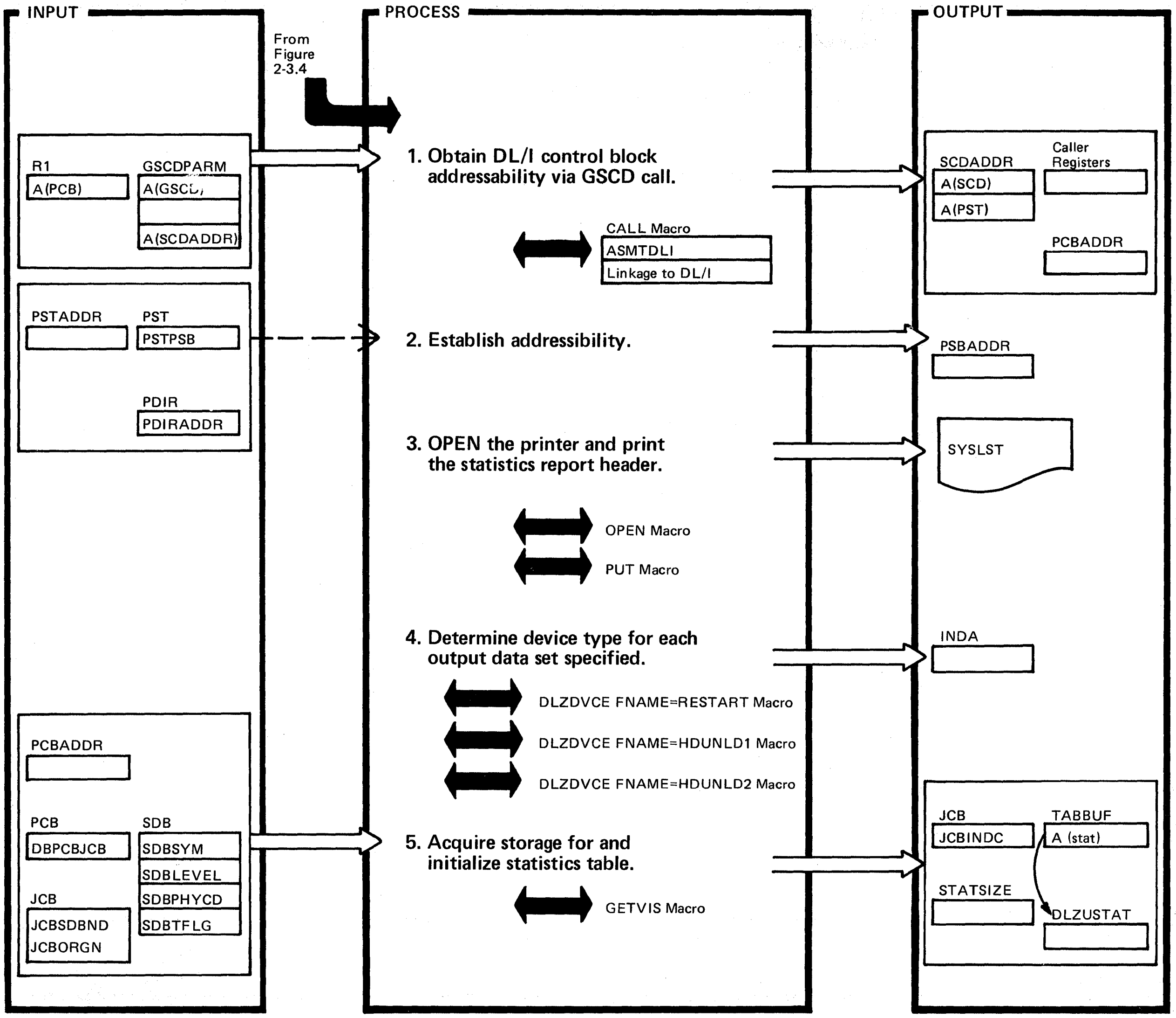
DLZURRLO – HISAM Reorganization Reload Utility

DLZURRLO

Extended Description	Routine	Label
1. Read and validate control statement. Write the following messages as needed: DLZ302I - Column 1 not L DLZ307I - Input filename not specified DLZ309I - Error(s) found in control statement DLZ310I - Image of erroneous control statement		
2. DLZDVCE macro obtains data from PUB. Device may be TAPE or DASD.		
3. The first record on the input file contains a statistics table initialized to zero. Included is the segment code and length for all segment types in the data base.		
4. Issue GENCB for ACB, RPL, and EXLST. Open KSDS and ESDS unless ACCESS=SHISAM (KSDS only).		
5. KSDS image records written to KSDS as key sequence records. ESDS image records written to ESDS as address sequence records.		

Extended Description	Routine	Label

Figure 2-31. HD DB Unload (Part 1 of 5)



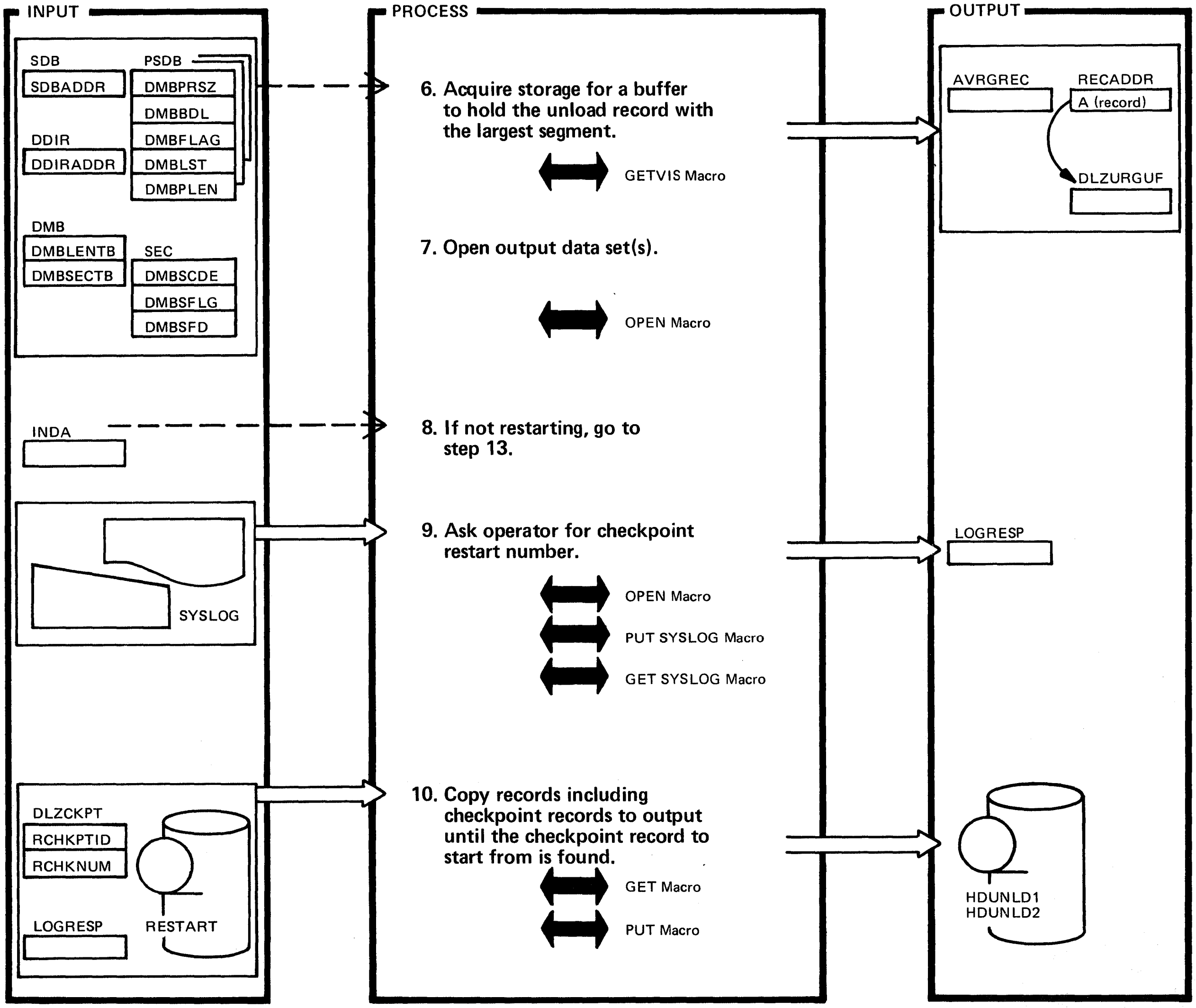
DLZURGU0 – HD Reorganization Unload CSECT

DLZURGU0

Extended Description	Routine	Label
1. Module identifier (DLZURGU0vrnp) is defined here. The GSCD call returns the SCD address +X'60' and the PST address in the call parameter I/O area (SCDADDR). The PCB address is passed to this module in R1 and stored at PCBADDR for later use.	DLZURGU0	DLZURGU0 BEGIN
3.		PUTHEAU
4. If restarting, set the restart in process indicator on at INDA. If HDUNLD1 is IGN or UA, write DLZ311I followed by DLZ347I, and then terminate. If HDUNLD2 is IGN or UA, write DLZ345I and continue processing.		PUBCHK1

Extended Description	Routine	Label
5. Indicate to DL/I Retrieve that HD UNLOAD is running by setting the indicator (JCBHDULD) at JCBINDC. Macro DLZUSTAT contains the DSECT defining the format of a statistics table entry. The table contains the segment code and length for all segment types in the data base.		MAIN1

Figure 2-31. HD DB Unload (Part 2 of 5)



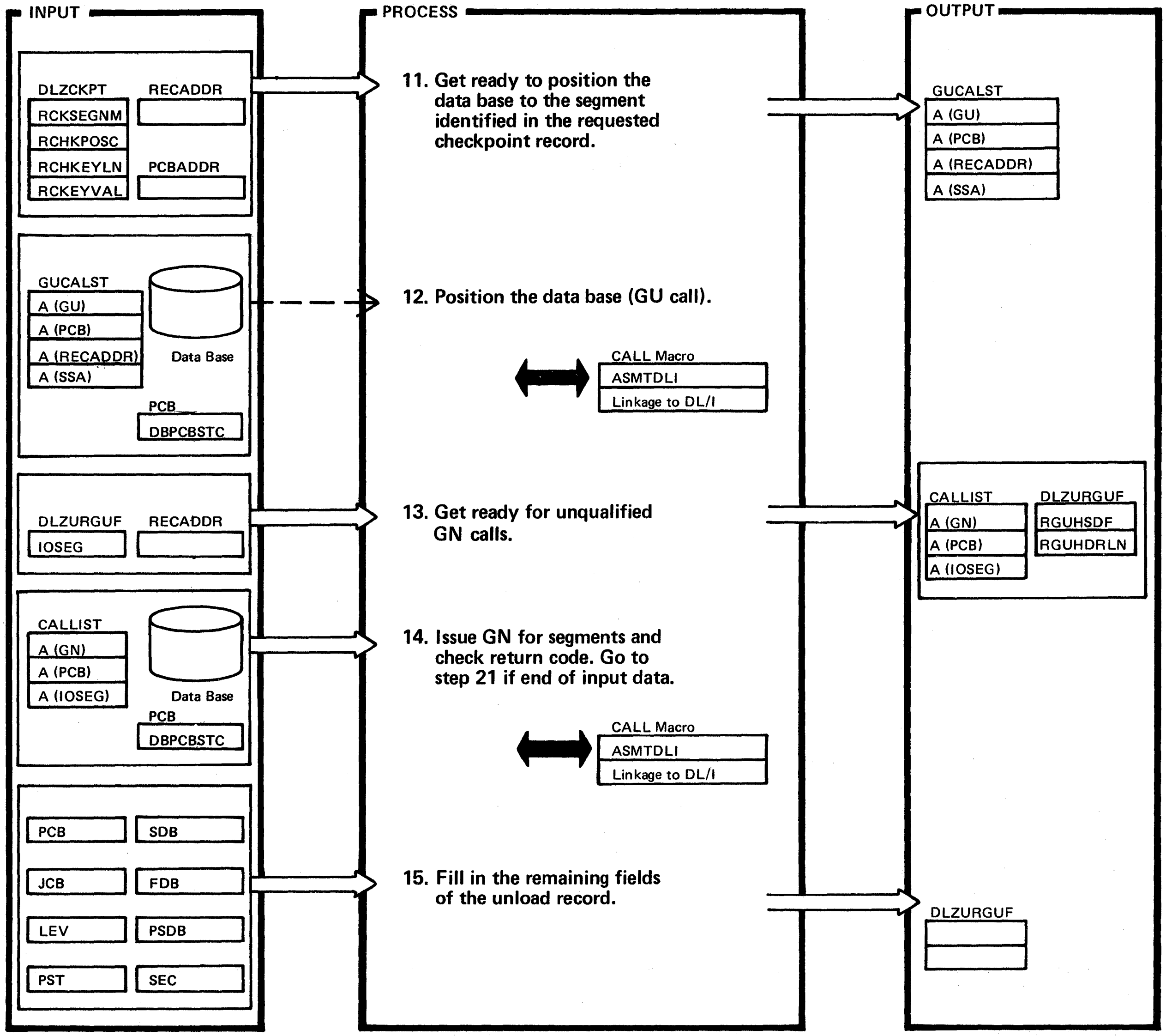
DLZURGU0 – HD Reorganization Unload CSECT

DLZURGU0

Extended Description	Routine	Label
6. Macro DLZURGUF contains the DSECT defining the format of an unload record.		STATEND
7.		TESTSECD
8.		SETOUT
9. Open the RESTART file and write message DLZ318A to SYSLOG requesting restart number and read response.		RESTRT0
10. Macro DLZCKPT contains the DSECT defining the format of a checkpoint record. Write message DLZ315I if end-of-file is reached on the restart data set without finding the requested checkpoint record.		RESTRT2A

Extended Description	Routine	Label

Figure 2-31. HD DB Unload (Part 3 of 5)



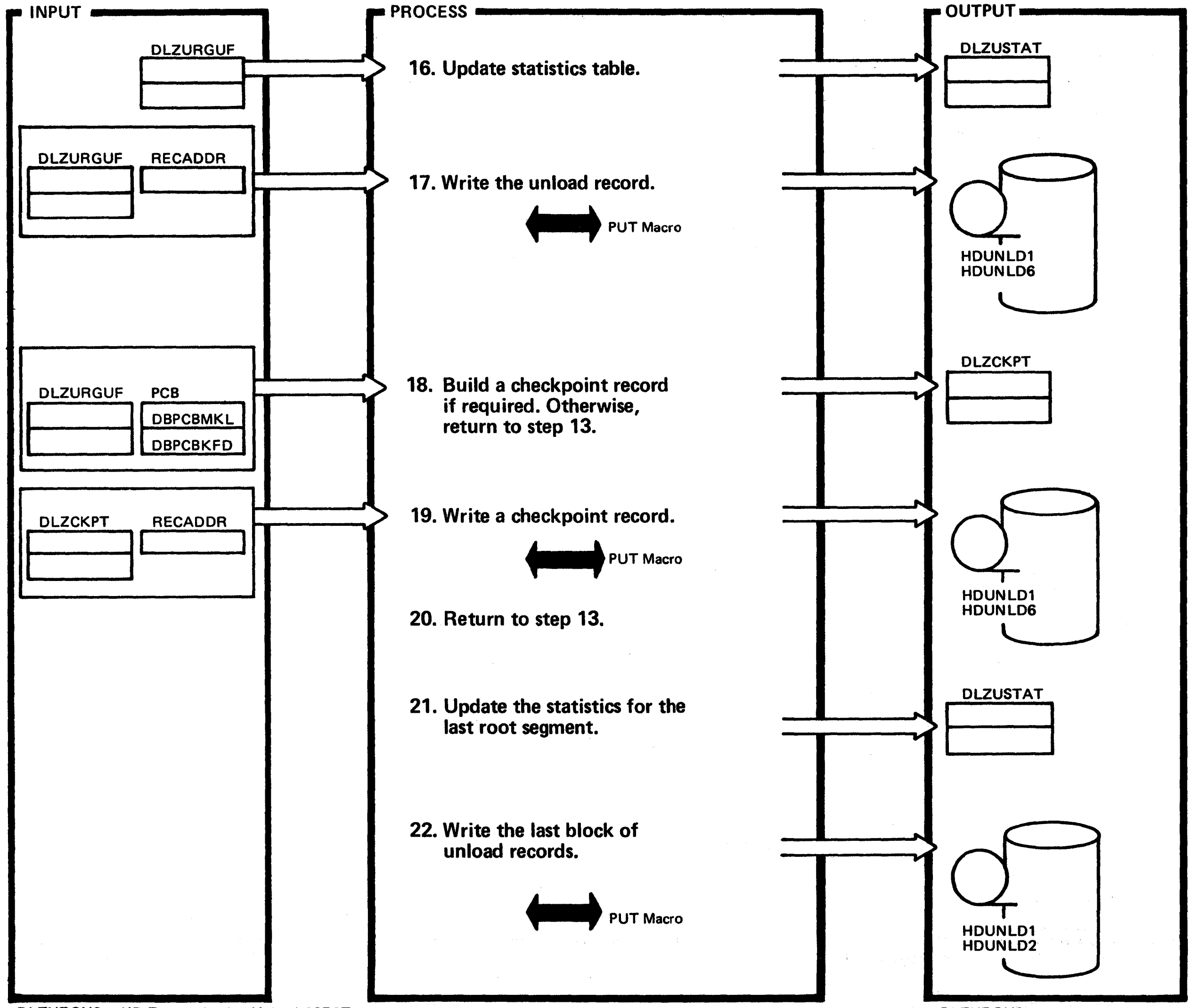
DLZURGUO – HD Reorganization Unload CSECT

DLZURGUO

Extended Description	Routine	Label
11. If the RBN is available in the check-point record, the SSA will be "segname*T (rbn)" (HDAM or HIDAM). If the RBN is not available, a qualified key call is required. The GETVIS macro is used to get a work area to build the SSA for the call. The SSA will be "segname*C (key)". Following the call, the work area is freed.		RESTRT4
12. Write the following messages as needed: DLZ301I – Open failure DLZ348I – Unexpected status from return code DLZ349I – Input I/O error DLZ378I – Restart successful DLZ379I – Restart failed DLZ380I – Segment not found		

Extended Description	Routine	Label
13. IOSEG is the beginning of the variable length data field following the DL/I prefix information of the unload record.		RECREATE
14. Write the following messages as needed: DLZ301I – Open error DLZ348I – Unexpected return code DLZ349I – I/O error		
15. Write message DLZ400I for a sequence error.		HDRFILL

Figure 2-31. HD DB Unload (Part 4 of 5)



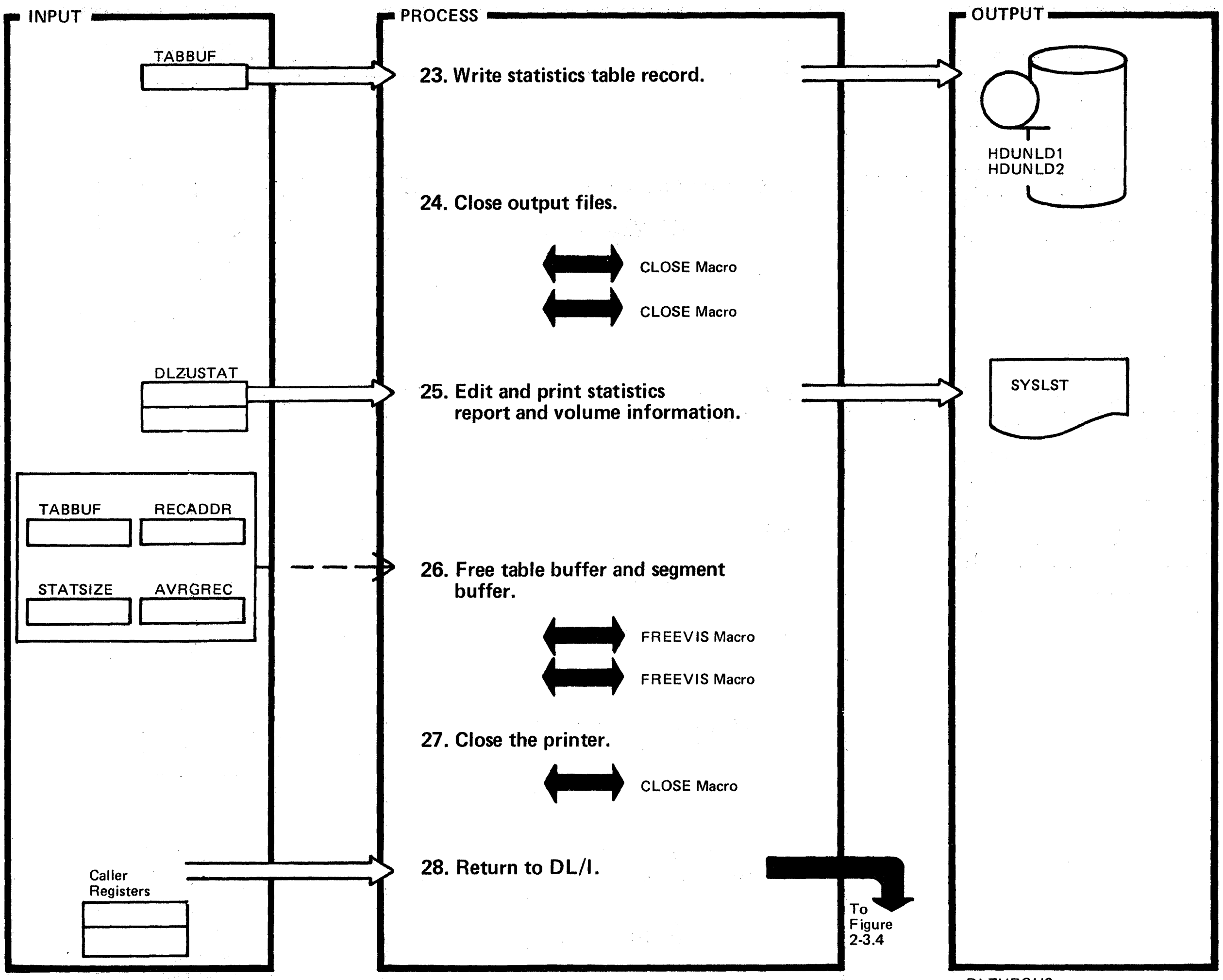
DLZURGU0 — HD Reorganization Unload CSECT

DLZURGU0

Extended Description	Routine	Label
16.		SETDLEN UPSTATS
17. The records are moved to the output block. When the block of records is full, the block is written.		WRITE3
18. Checkpoint records are written at the first root segment after every 5000 segments.		TSTCHK CHKPNT
19. Write message DLZ381I every time a checkpoint is taken.		CHKPNT2
21.		EODINPUT LASTROOT

Extended Description	Routine	Label

Figure 2-31. HD DB Unload (Part 5 of 5)



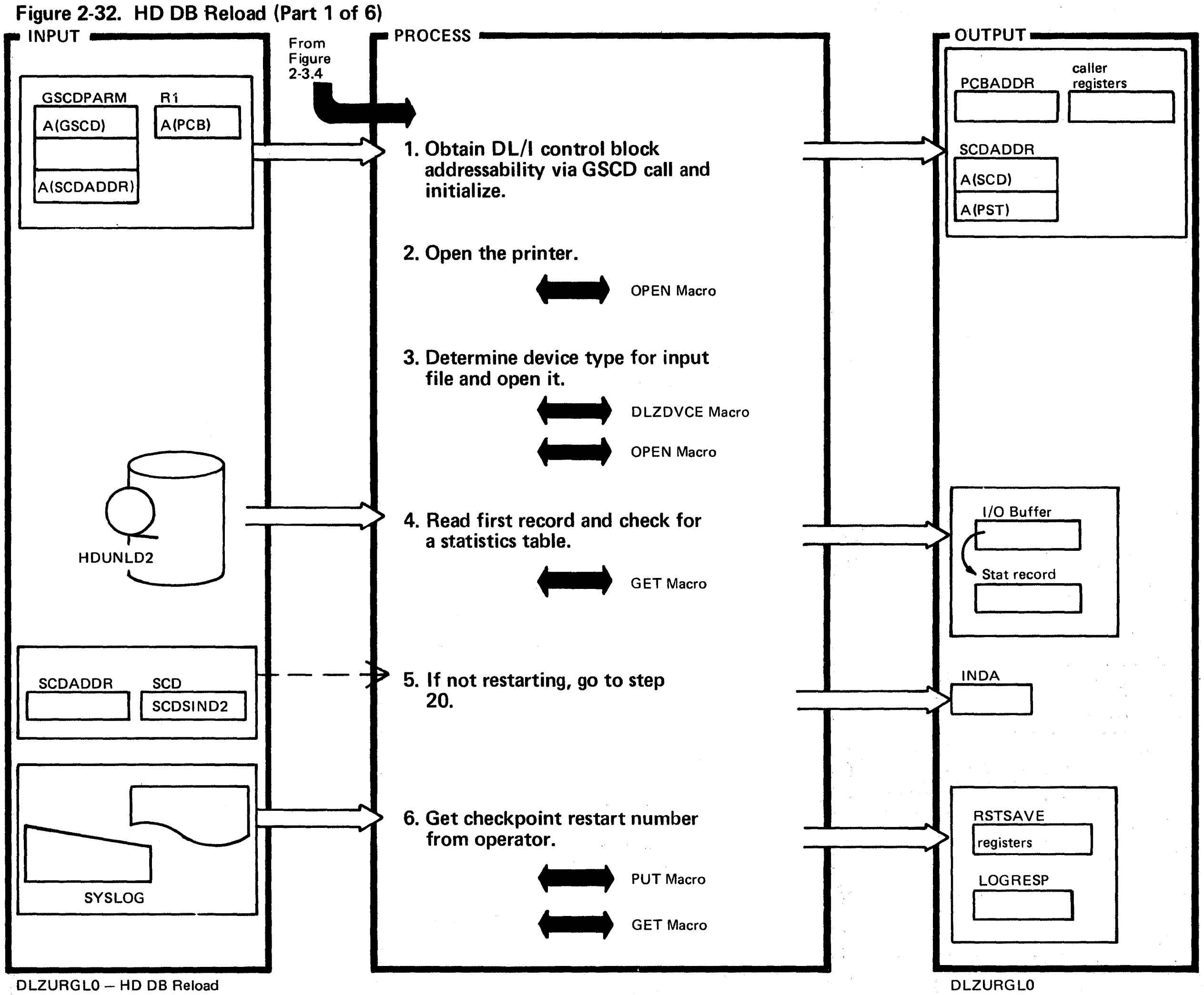
DLZURGU0 – HD Reorganization Unload CSECT

DLZURGU0

Extended Description	Routine	Label
23.		WRTTLST
24.		CLOSE2
25. Write message DLZ352I (no errors detected).		EDITSTAT
26. Write message DLZ392I for FREEVIS error.		STOPRUN
27.		STOPRUN2
28.		NODUMP

Extended Description	Routine	Label

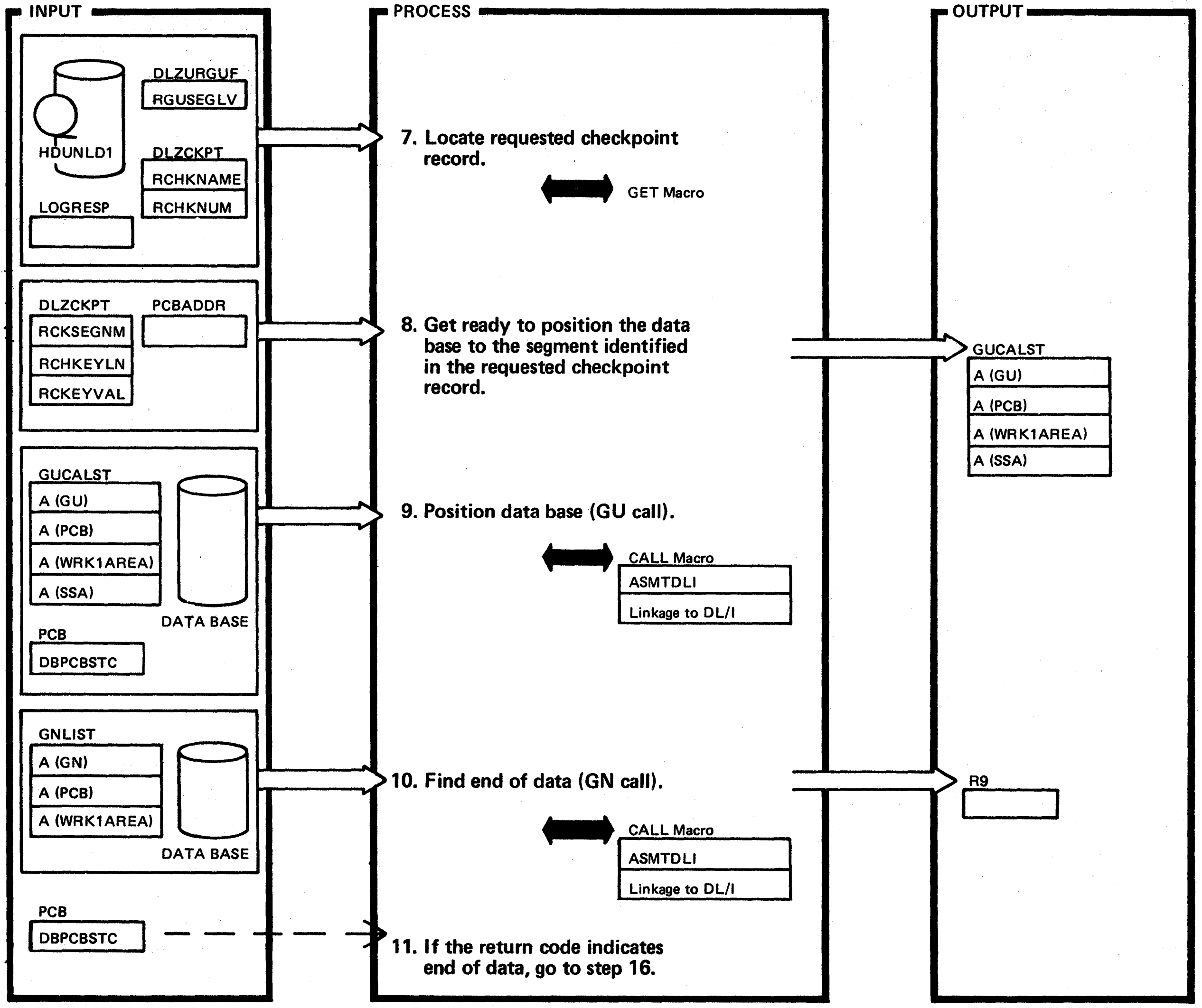
Figure 2-32. HD DB Reload (Part 1 of 6)



Extended Description	Routine	Label
1. Module identifier (DLZURGLOvrnp) is defined here.	DLZURGLO	DLZURGLO BEGIN
The PCB address is passed to this module in R1 by DLZRRC00 and stored at PCBADDR for later use.		
The GSCD call returns to the SCD address + X'60' and the PST address in the call parameter I/O area (SCDADDR).		
3. Write DLZ311I if HDUNLD1 is not assigned.		PUBCHKA
4. Issue DLZ389I if abnormal statistic table record.		STATINIT
5. If the HD DB Reload Utility Program fails, the reload restart capability allows you to restart from a checkpoint record. Before submitting the job for a reload restart, change the parameter card from ULU to ULR.		

Extended Description	Routine	Label
6. Write DLZ318A message to SYSLOG requesting restart number and read response.		RSTMESSG
The number of the last valid checkpoint record on the unloaded file is found in console message DLZ381I. Valid checkpoint numbers are decimal values between 1 and 9999.		

Figure 2-32. HD DB Reload (Part 2 of 6)



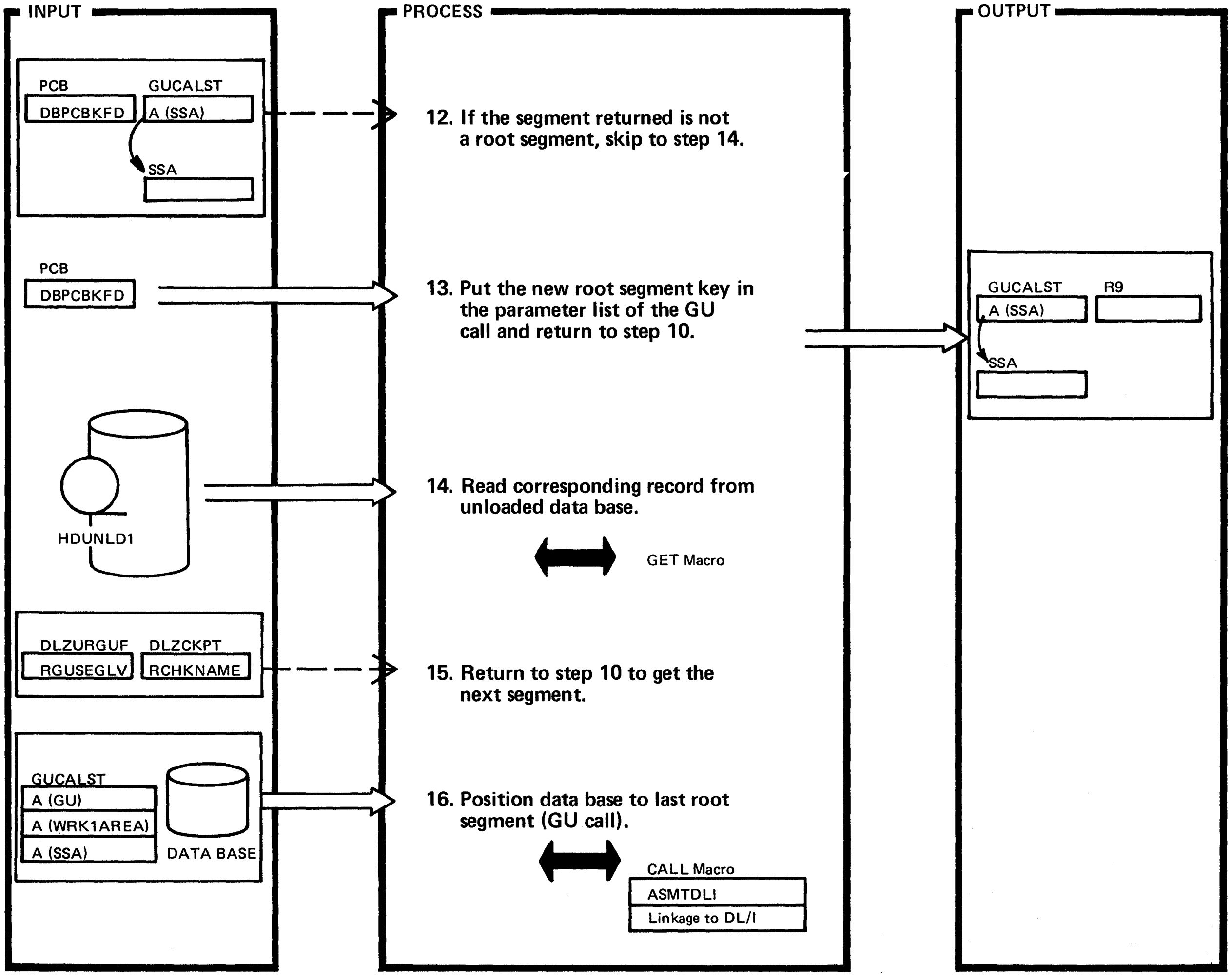
DLZURGLO — HD DB Reload

DLZURGLO

Extended Description	Routine	Label
7. Write DLZ370I if the checkpoint requested is less than the first checkpoint record encountered. Write DLZ315I if checkpoint record not found. Write DLZ381I checkpoint information message		RSTGETLP
8. The SSA for the GU call is 'segname*C (key)', a qualified key call.		RSTPOSIT
9. The return code is checked. Write DLZ380I unable to position DB, checkpoint record not found.		CALLIT

Extended Description	Routine	Label
10. Now that the data base is positioned at the segment identified by the checkpoint record, issue GN calls to the end of the data base at the same time reading corresponding records from the unloaded data base in order to keep the two synchronized. A counter for GN calls (R9) is incremented by one for every GN call following a root segment. Write DLZ380I unable to position DB if invalid return code.		GNCALL

Figure 2-32. HD DB Reload (Part 3 of 6)



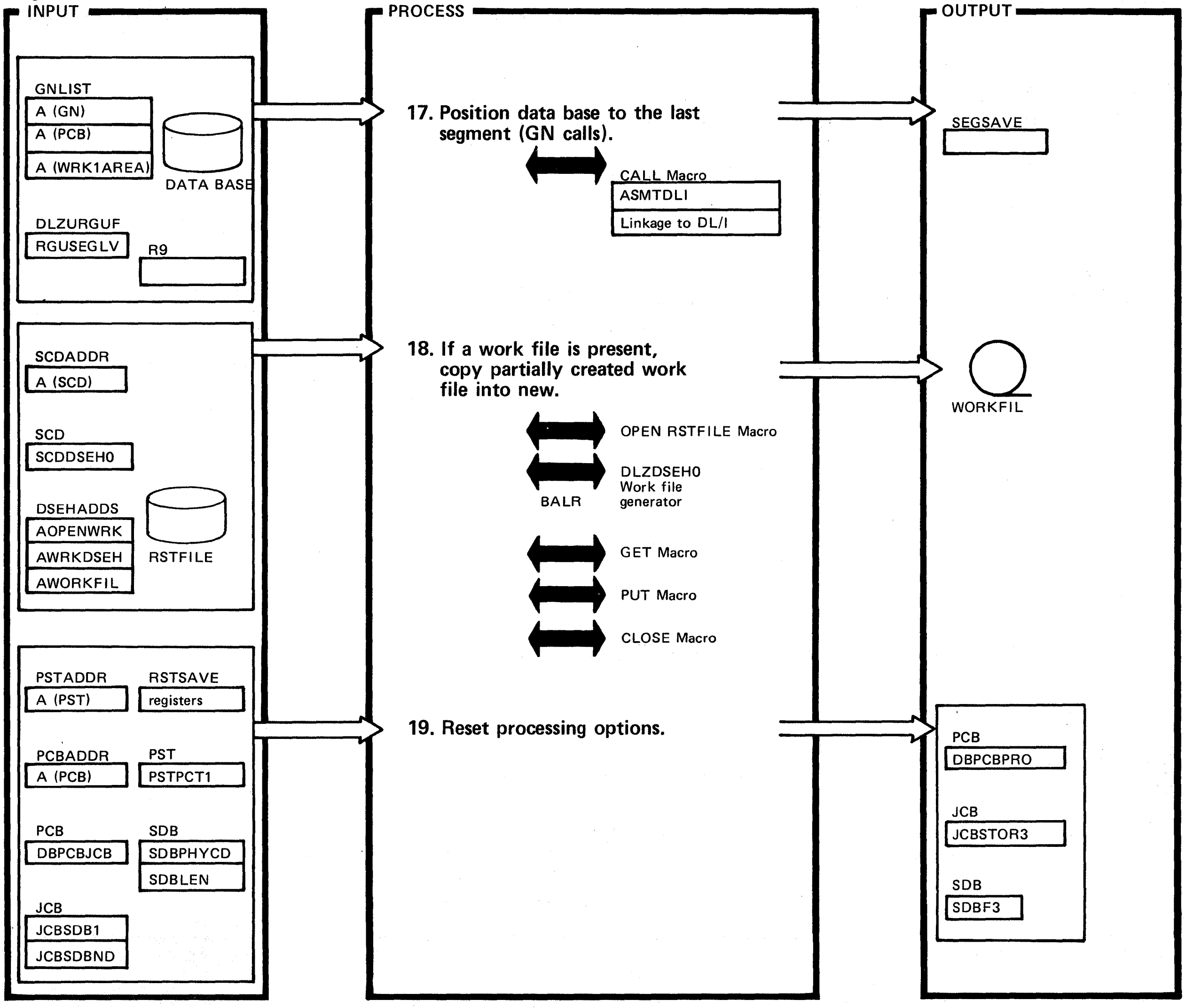
DLZURGLO – HD DB Reload

DLZURGLO

Extended Description	Routine	Label
12.		CHKKEY
13. Reset the counter for GN calls (R9) to zero.		
14.		UNLDGET
15. If the record returned from the unloaded data base is a checkpoint record, return to step 14 to get the next record in order to keep the partially reloaded data base and the input unloaded data base synchronized. Issue DLZ381I checkpoint information message.		
16.		REPOSN

Extended Description	Routine	Label

Figure 2-32. HD DB Reload (Part 4 of 6)

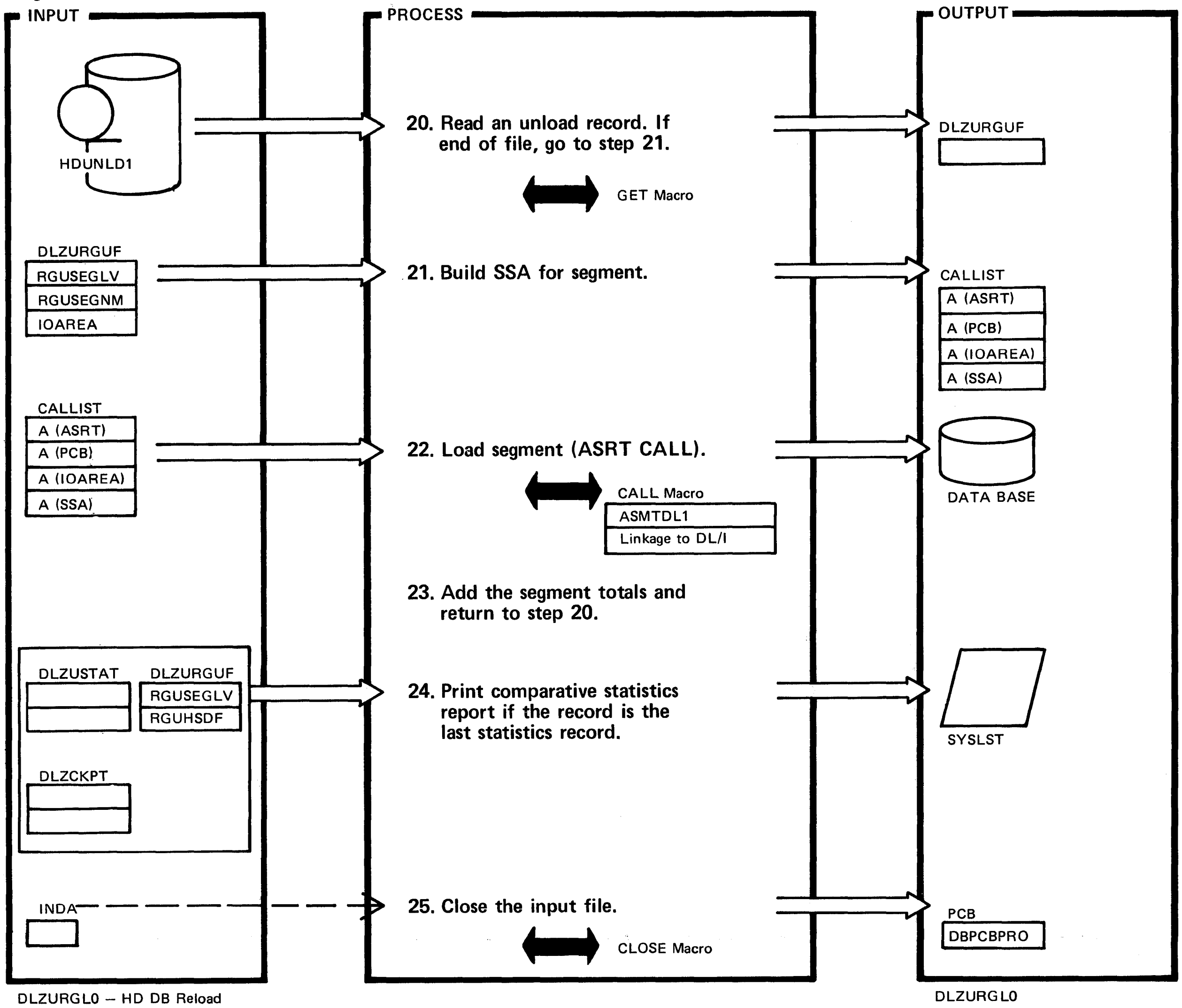


DLZURGLO – HD DB Reload

DLZURGLO

Extended Description	Routine	Label
17. Save the segment code of the last segment. The number of GN calls to make to get to the last segment is in R9.		GNLOOP
18. If a work file (for logical relationships or secondary indices) was being created during the reload, the partially created work file should be submitted as input to the restarted job assigned as SYS010 with a file name of RSTFILE.		SETWKFIL RSTOPEN
Write DLZ376I invalid device assignment.		
19. Now that Reload Restart processing is complete, set the processing options to indicate that a load is in process. Then resume processing as usual.		SETPROPT
If HISAM, set LS in the PCB. Otherwise, set L.		
Save the address of the SDB for insert processing if it is for the last segment found in JCBSTOR3.		

Figure 2-32. HD DB Reload (Part 5 of 6)



Extended Description	Routine	Label
20.		GETLOOP
21. IOAREA is the address of the data portion of the unload record DLZURGUF.		NOSTAT
22. Write DLZ301I OPEN error. Write DLZ319I IO error. Write DLZ348I invalid return code.		
23.		STATCOMP
24.		LASTCOMP
25. Write DLZ355I if reload is not okay, or DLZ379I if checkpoint restart is not okay. Write DLZ354I if reload is okay, or DLZ378I if checkpoint restart is okay. If this was RESTART (ULR), reset the processing option in the PCB back to A.		EOD GOODRUN

Extended Description	Routine	Label

Figure 2-32. HD DB Reload (Part 6 of 6)

INPUT

Caller registers

PROCESS

26. Close the printer.

CLOSE Macro

27. Return to DL/I.

Figure 2-3.4

OUTPUT

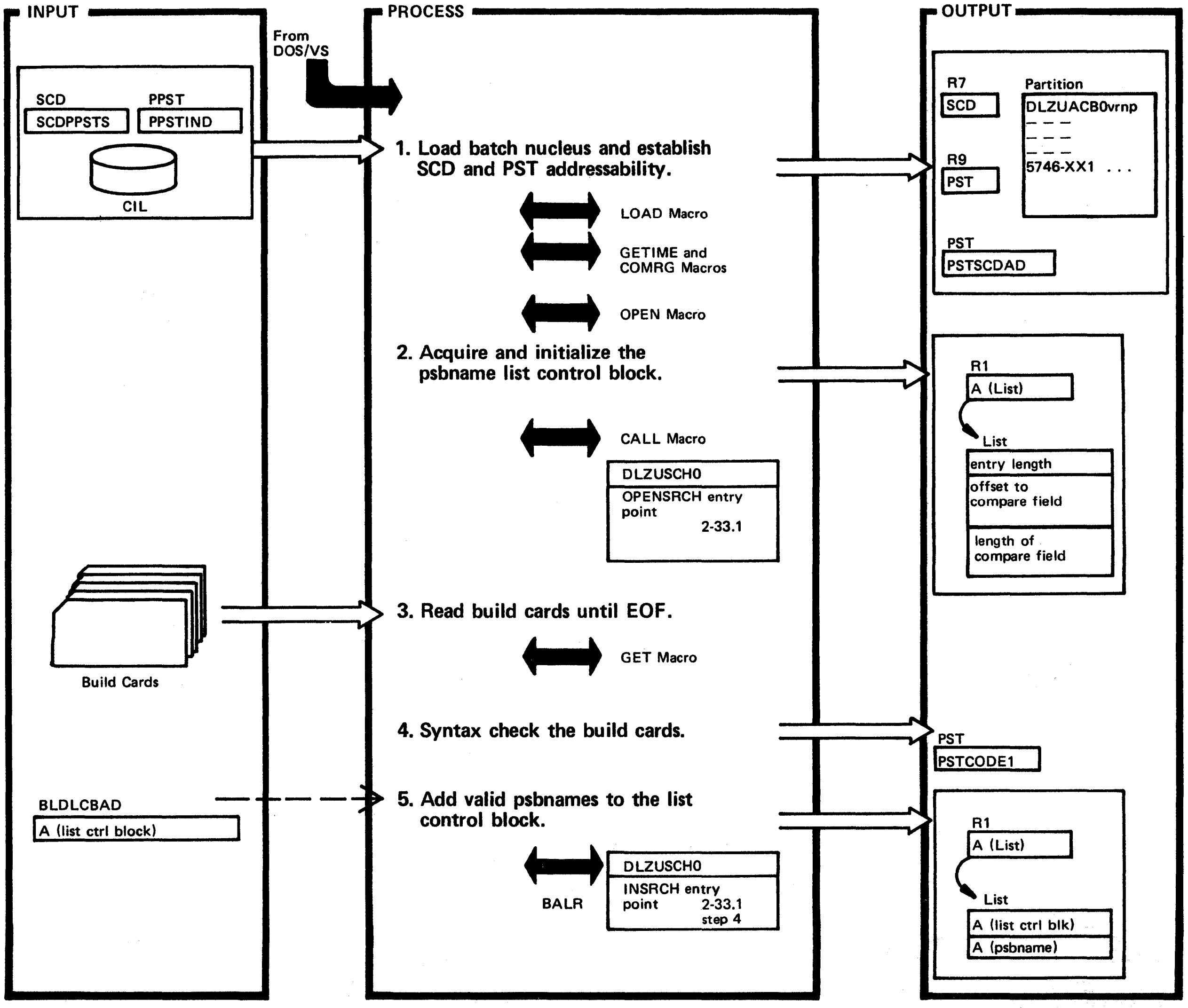
DLZURGL0 – HD DB Reload

DLZURGL0

Extended Description	Routine	Label
26.		STOPRUN
27.		NODUMP

Extended Description	Routine	Label

Figure 2-33. ACB Creation Utility Overview (Part 1 of 3)



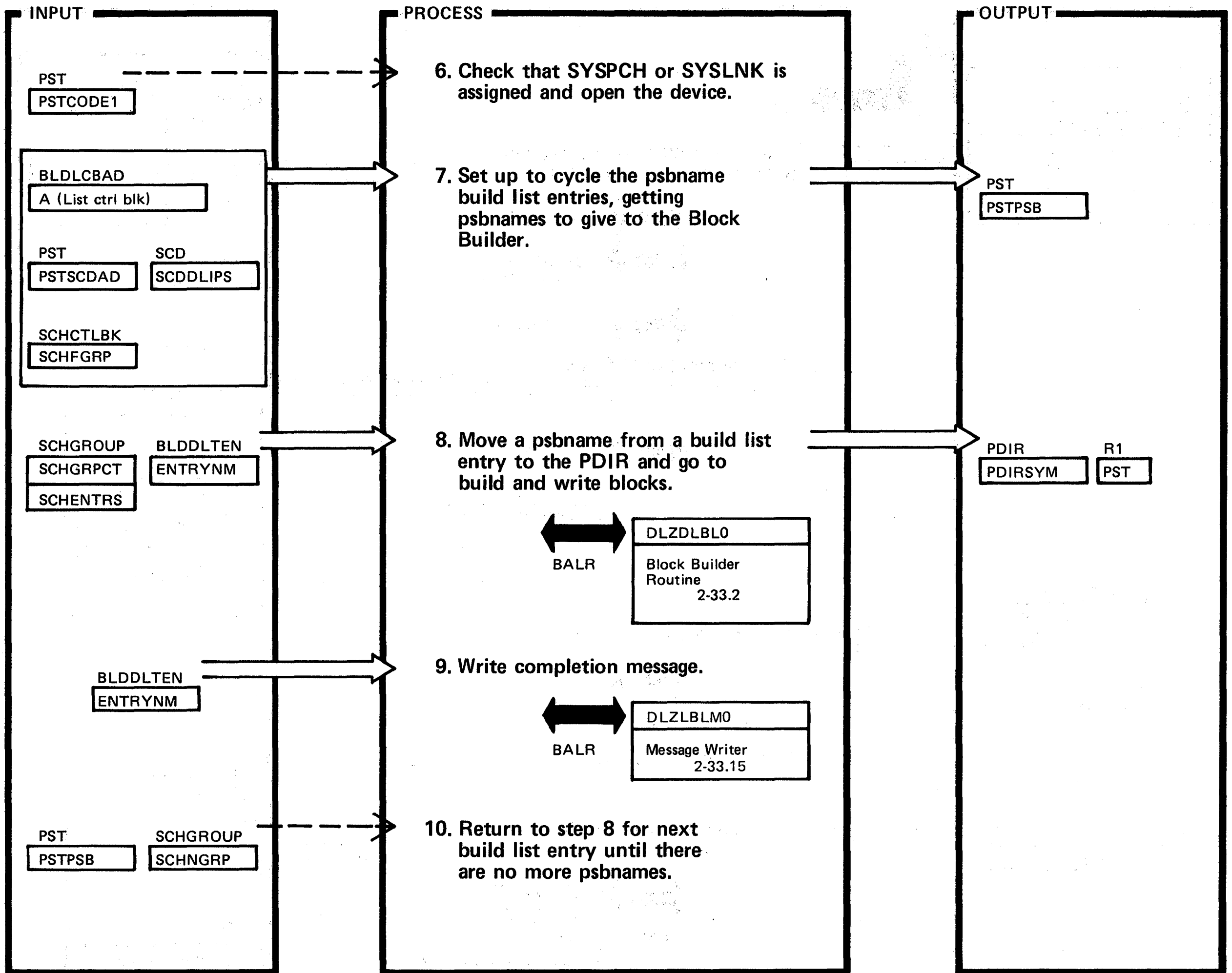
DLZUACB0 – Create ACB Utility Csect

DLZUACB0

Extended Description	Routine	Label
1. Module identifier (DLZUACB0vrnp) is defined here. The level format is vrnp; where 'v' is the version, 'r' is the release, 'n' is an additional identification number, and 'p' is the latest PTF number that has been applied. The time and date are required for the report heading to DLZUACB0 messages and control statements that are printed as this utility executes.	DLZUACB0	DLZUACB0
2. The parameter list for OPENSRCH is passed in R1. OPENSRCH returns the address of the list control block in R1, which is then put in the parameter list for INSRCH. Write DLZ905I if GETVIS error returned from OPENSRCH.		INITSORT
3.		READCARD GETT

Extended Description	Routine	Label
4. Write DLZ588I for an invalid delimeter, label, opcode, block type, operand, continuation, or invalid format. The output device is set at this time (SYSLINK or SYSPCH). If OUT=LINK was specified in the build card, SYSLNK is indicated in the PST.		
5. The psbnames from the PSB= operand are passed one at a time to INSRCH. The parameter list for INSRCH is passed in R1. Write DLZ905I if GETVIS error returned from INSRCH or DLZ571I warning message if a duplicate psbname is found.		MOVEIT

Figure 2-33. ACB Creation Utility Overview (Part 2 of 3)



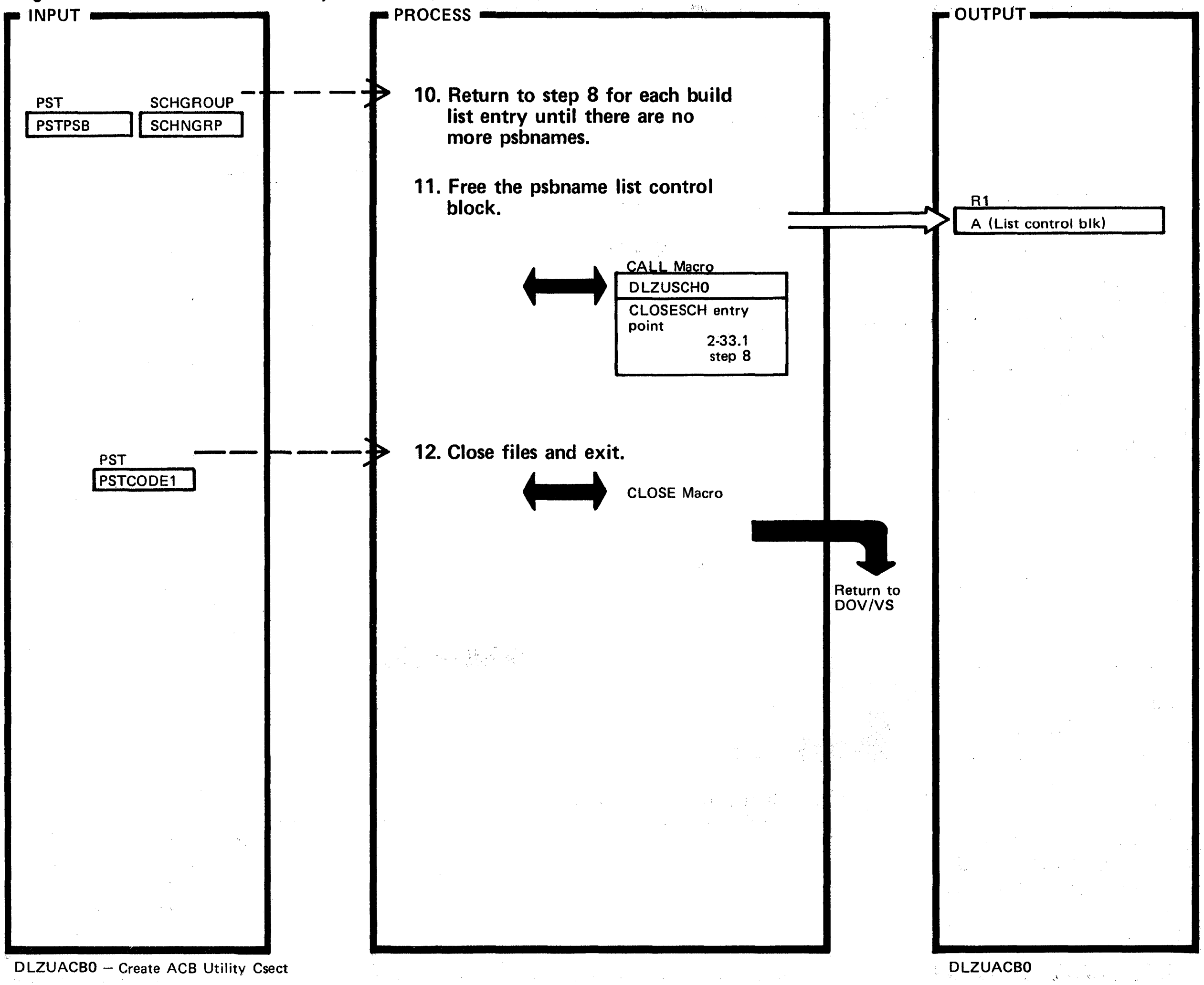
DLZUACB0 — Create ACB Utility Csect

DLZUACB0

Extended Description	Routine	Label
6.		CARDEOF OPEN
8. PSTERCOD is examined to see if any errors were posted by the Block Builder. If not zero, issue message DLZ587I to indicate the PSB will not be built and go to step 10 to attempt to build remaining blocks.		BLDGROUP CALLBB
9. The completion message is normal unless a non-zero return code is found from DLZDLBL0. Write DLZ589I to indicate processing has been completed for the PSB specified.		
10.		NXTPSB

Extended Description	Routine	Label

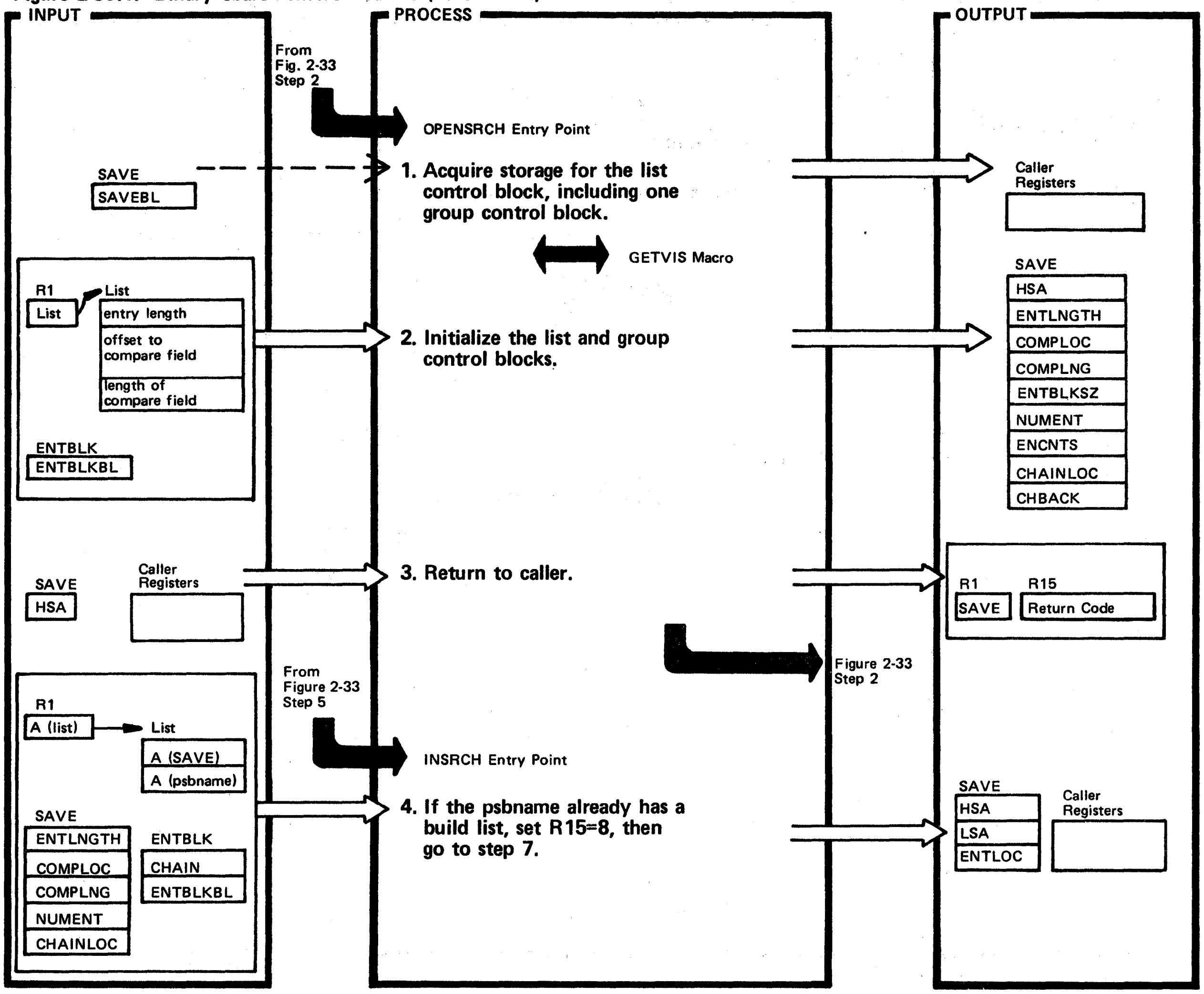
Figure 2-33. ACB Creation Utility Overview (Part 3 of 3)



Extended Description	Routine	Label
10.		NXTPSB
11.		BLDDONE
12.		CLOSE

Extended Description	Routine	Label

Figure 2-33.1. Binary Search Insert Routine (Part 1 of 2)



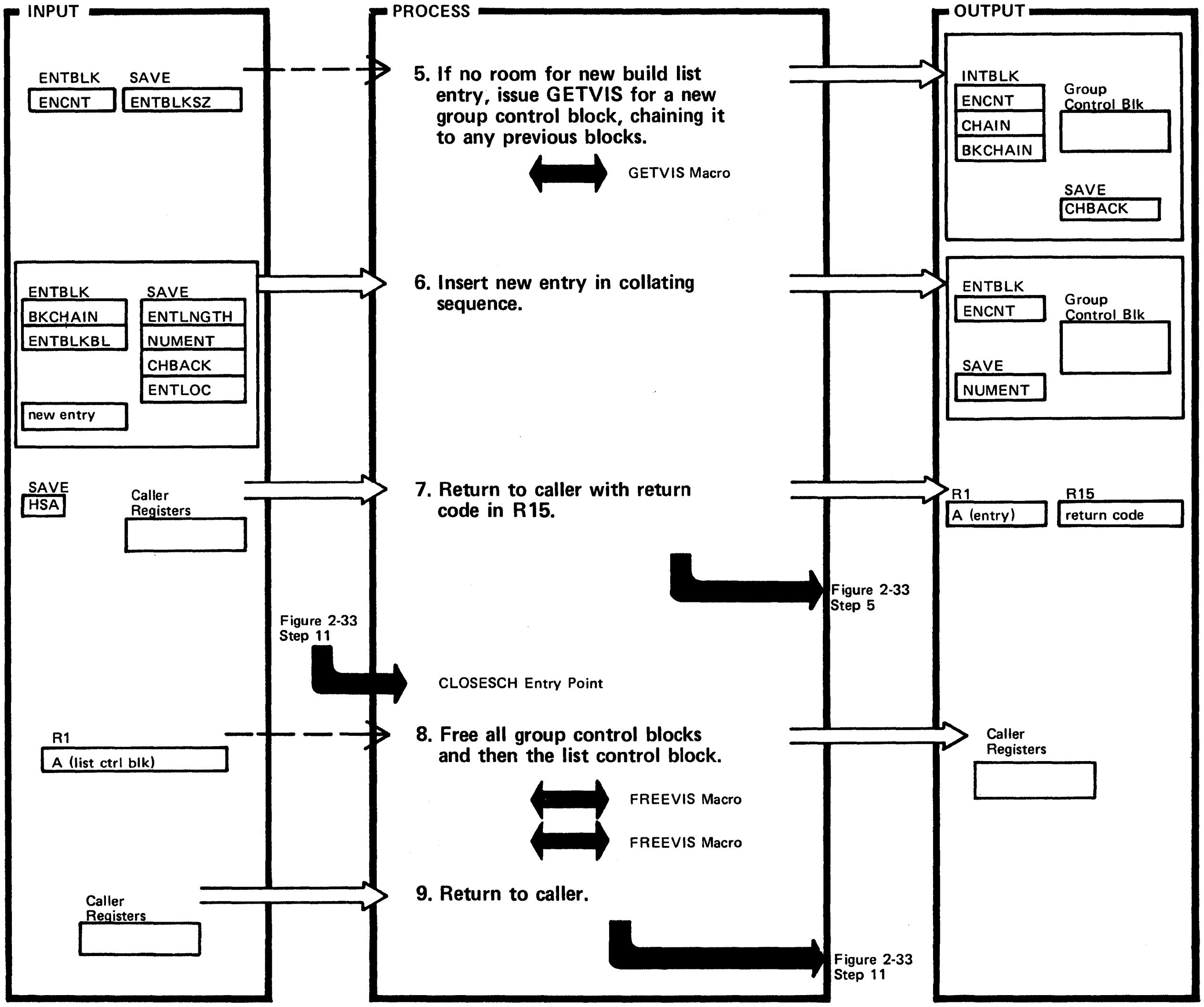
DLZUSCHO — ACB Maintenance Binary Search Insert Routine

DLZUSCHO

Extended Description	Routine	Label
1. Module identifier is defined here. Length acquired is X'80' bytes. One group control block will hold 16 build list entries.	DLZUSCH0	DLZUSCH0 OPENSRCH
2. Block now contains information needed to build a group control block. The first (or only) block is obtained before the first actual insert.		
3. The address of the created block is returned to the caller. Note: This routine is very generalized and could be used for other purposes, but it is only used by DLZUACB0 to build the psbname build list control block.		
4. Routine identifier (INSRCH. .) is defined here.	INSRCH	INSRCH INSRT1

Extended Description	Routine	Label

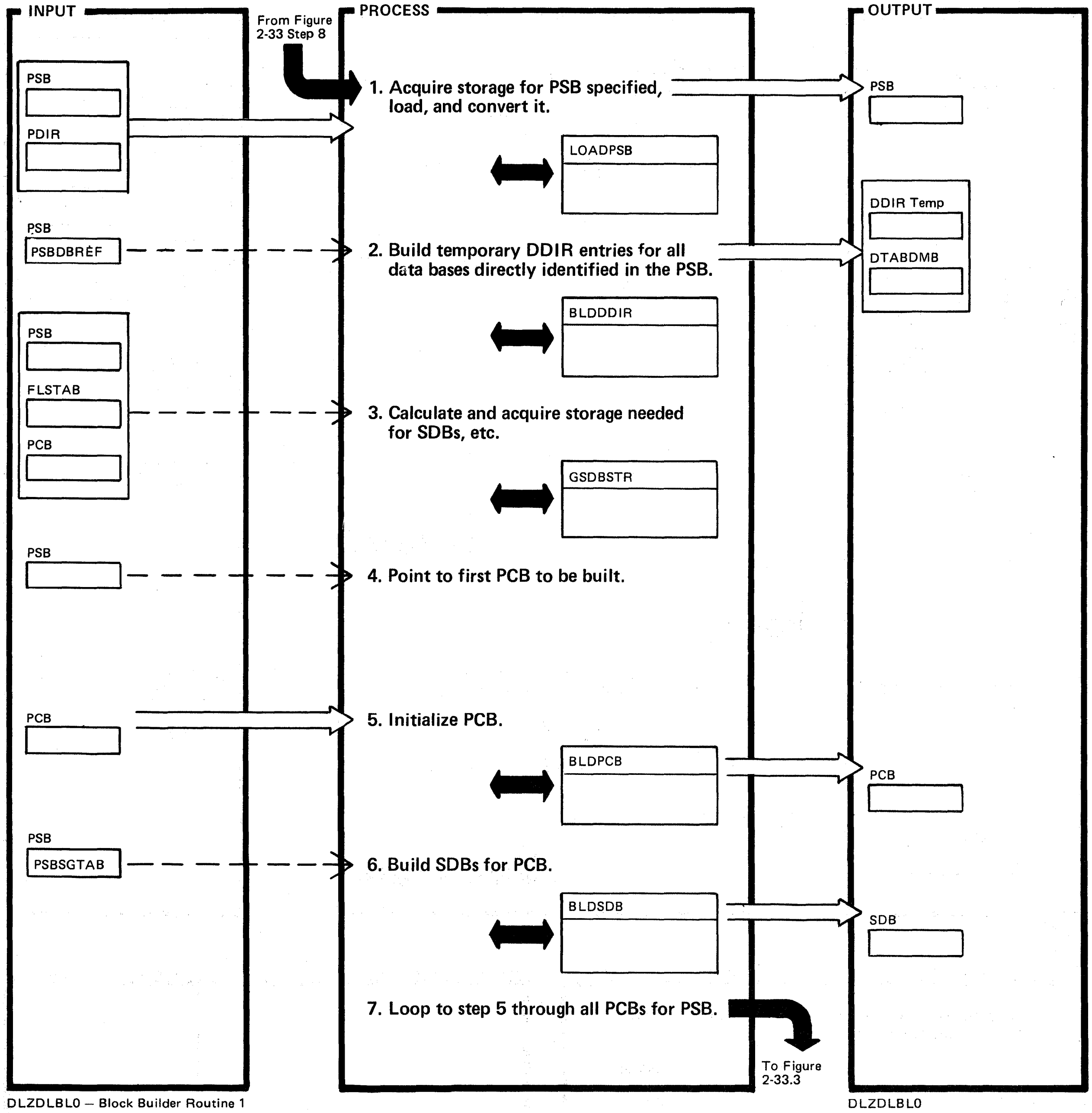
Figure 2-33.1. Binary Search Insert Routine (Part 2 of 2)



Extended Description	Routine	Label
5. There is enough room for 16 entries in a group control block.	INSRCH	NOTFND
6.		MVDK
7.		NOGO
8. Routine identifier (CLOSESCH) is defined here. All group control blocks can be found beginning from the list control block.	CLOSESCH	CLOSESCH

Extended Description	Routine	Label

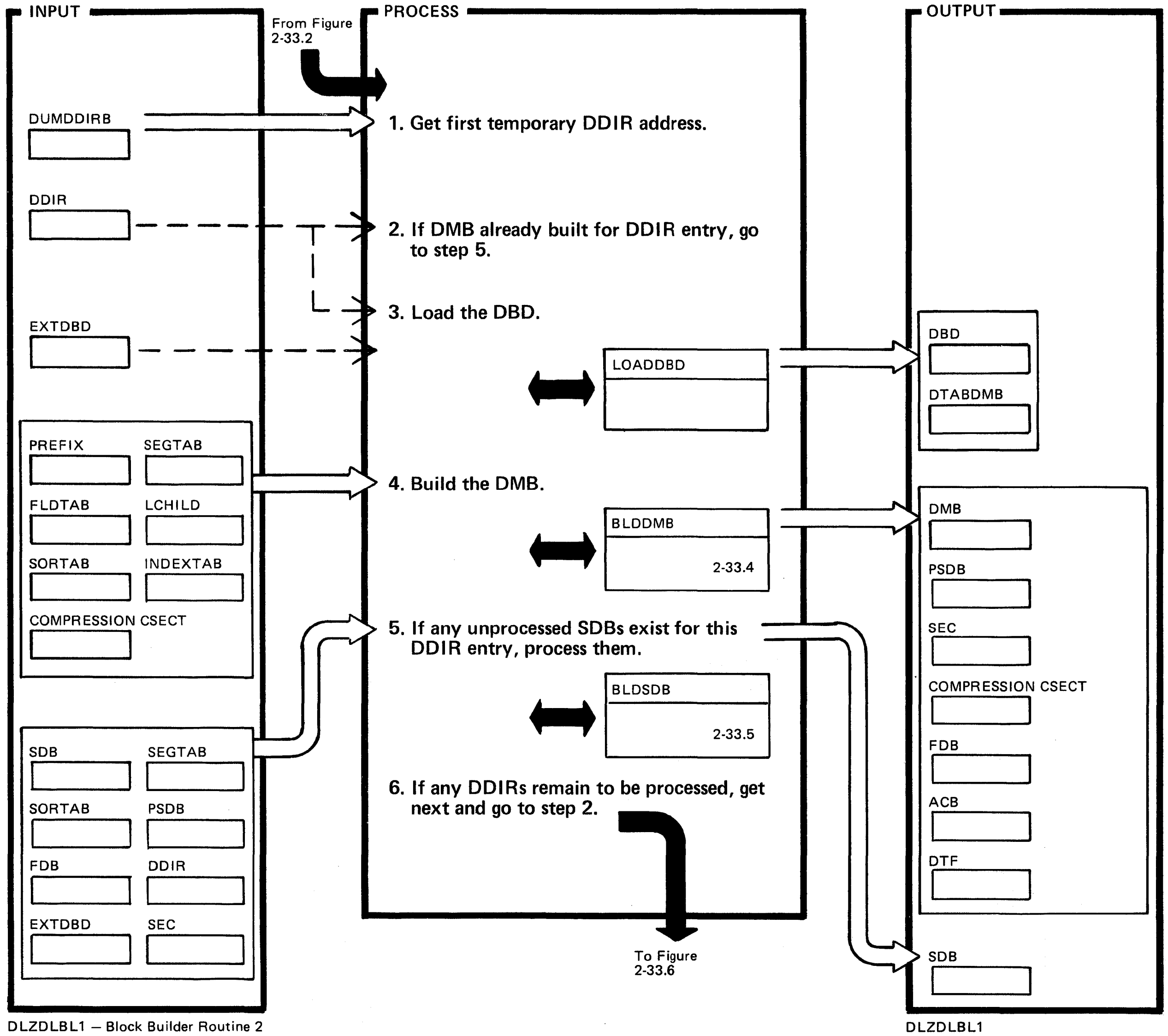
Figure 2-33.2. Block Builder Routine 1



Extended Description	Routine	Label
1.		PSBPASS
5.		PSBPASS1 BLDPCB
6. Only those SDBs for segments directly referenced by SENFLDs are built at this time. Generated SDBs will be built in DLZDLBL1.		PSBPASS3

Extended Description	Routine	Label

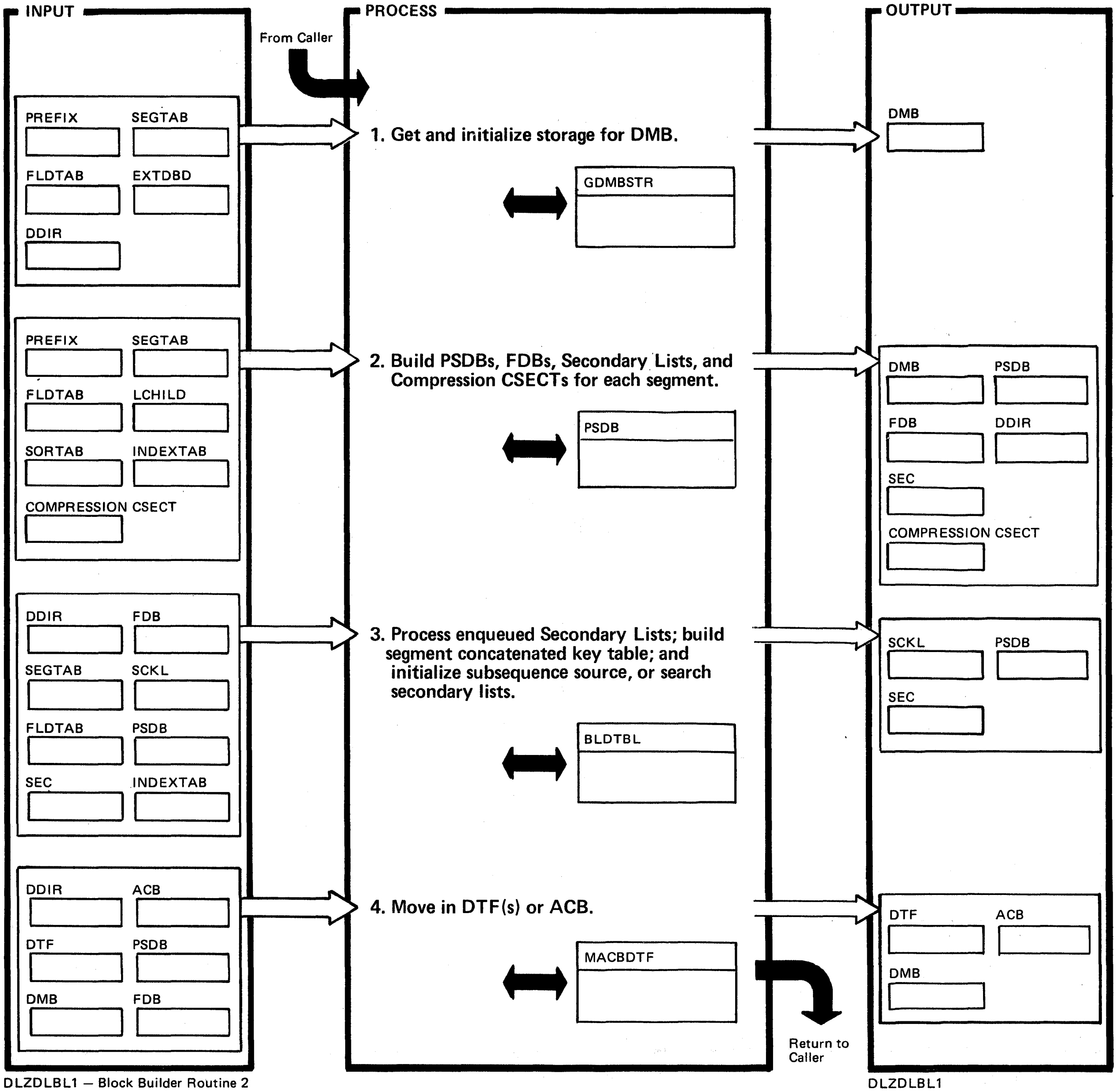
Figure 2-33.3. Block Builder Routine 2



Extended Description	Routine	Label
2. The DBD address is reset from the DMB Name Table.		DMBPASS1
3. Any DBDs referenced by this DBD are added to the dummy DDIR list.		DMBPASS2
4. A. If the DBD is for an index data base, the DDIR entry for the target data base is located and processed first. B. No DMB is built for logical DBDs.		DMBPASS3
5. A. SDBs pointing to VLC or logical segments are reset to point to the physical segment PSDB. B. All generated SDBs are built here.		DMBPASS8
6. If additional SDBs were chained to a prior DDIR during step 5, return to that DDIR, and go to step 2.		DMBPASS9

Extended Description	Routine	Label

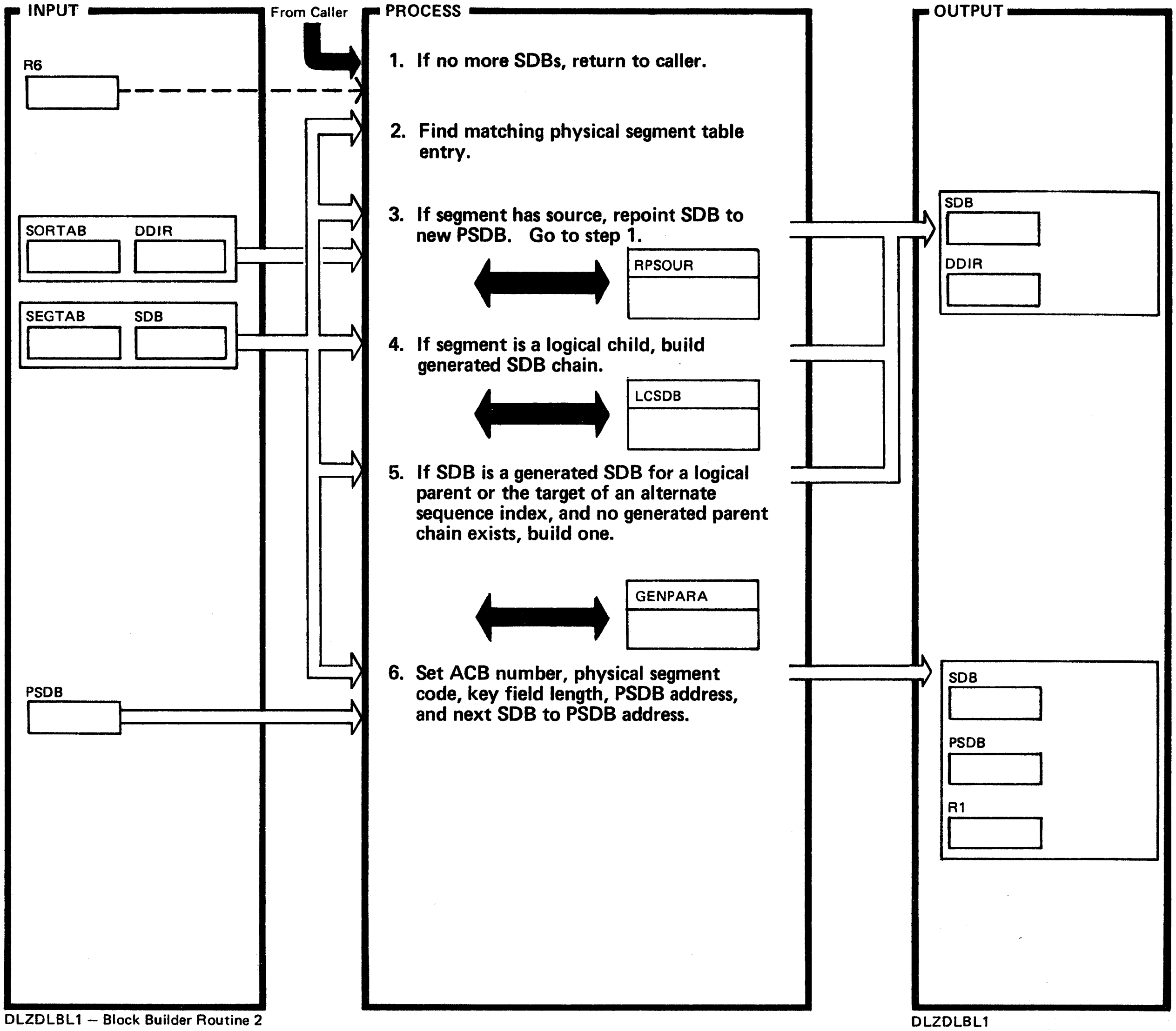
Figure 2-33.4. Block Builder BLDDMB Routine



Extended Description	Routine	Label
1.		BLDDMB
2. Until the concatenated key table is built, all secondary lists are enqueued on the DDIR and will be built during step 3.		BLDDMB1
3.		BLDDMB4

Extended Description	Routine	Label

Figure 2-33.5. Block Builder BLDSDB Routine (Part 1 of 2)



Extended Description	Routine	Label
2. Output message DLZ912I if not found.		BLDSDB1
3. If source is in another DBD, the SDB to be processed is the next one for this DBD. Otherwise, process the original with the new segment name.		BLDSDB3
4. If segment is a normal logical child, the logical parent SDB is generated and flagged to cause generation of the parent chain when the LP is processed. All generated SDBs are chained to the DDIRs for the related data bases.		BLDSDB7
5.		BLDSDB8
6.		BLDSDB9

Extended Description	Routine	Label

Figure 2-33.5. Block Builder BLDSDB Routine (Part 2 of 2)

INPUT

SDB

R1

DLZDLBL1 — Block Builder Routine 2

PROCESS

7. If SDB is not the root for an alternate sequence, go to step 11.

8. Generate SDB for index.

GENSDB

9. Generate SDB for target of index and flag to indicate parent chain is to be generated.

GENSDB

10. Go to step 1 and process SDB generated in step 9.

11. If SDB is not for HIDAM root, go to step 13.

12. Generate SDB for primary index.

13. Point to next SDB for DDIR, and go to step 1.

OUTPUT

SDB

SDB

R6

SDB

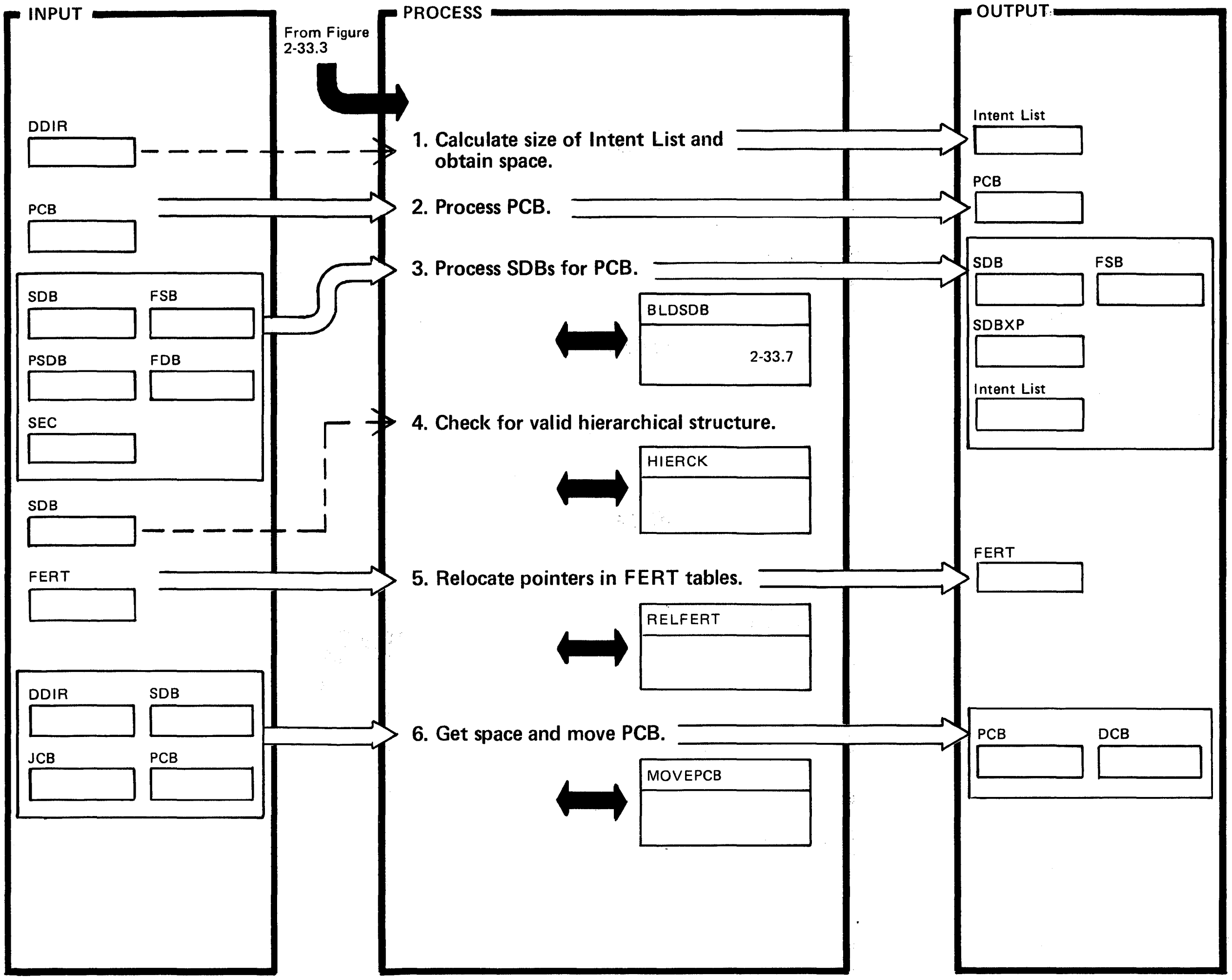
R6

DLZDLBL1

Extended Description	Routine	Label
7.		BLDSDBF
11.		BLDSDBG
13.		BLDSDBI

Extended Description	Routine	Label

Figure 2-33.6. Block Builder Routine 3 (Part 1 of 2)



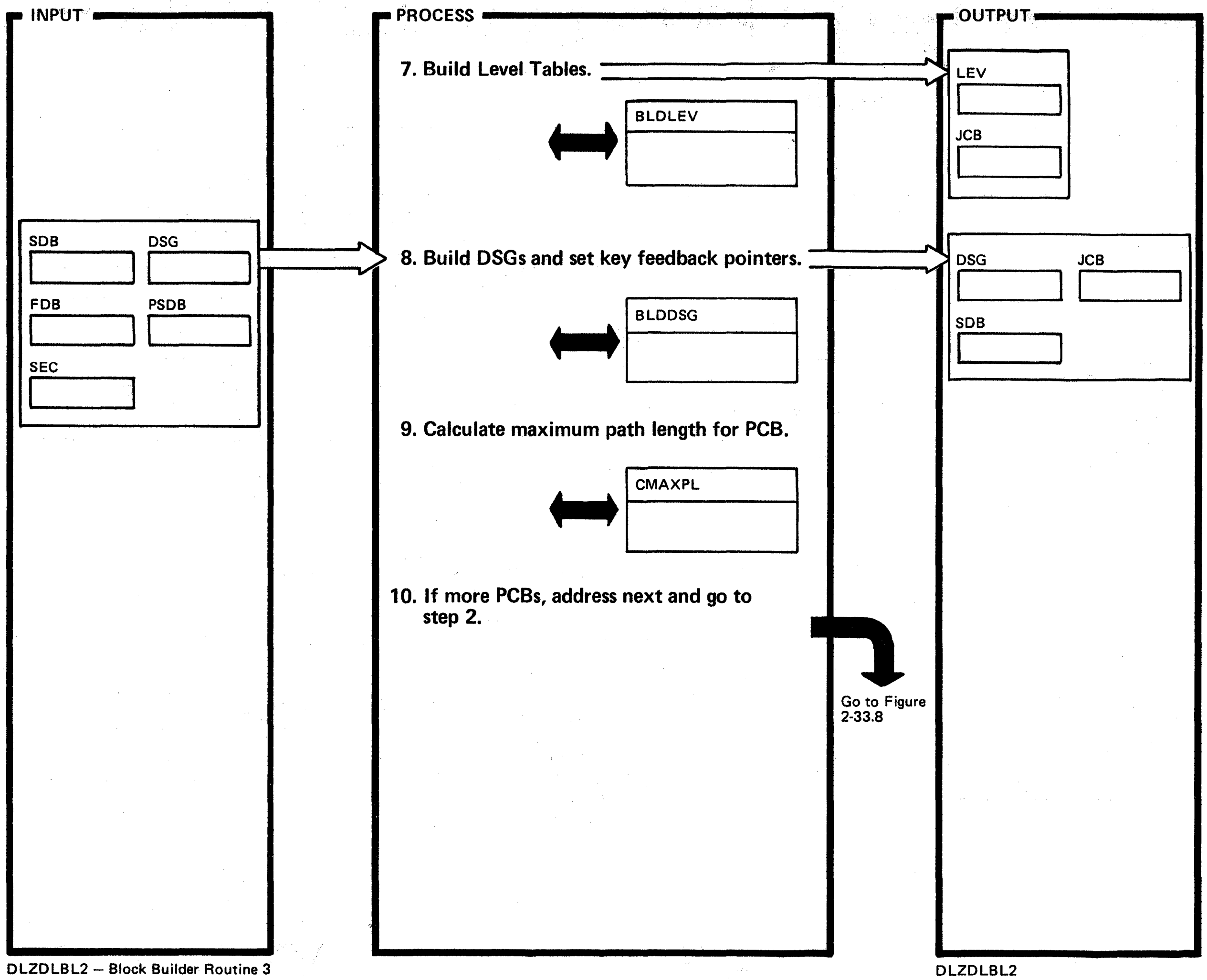
DLZDLBL2 — Block Builder Routine 3

DLZDLBL2

Extended Description	Routine	Label
1.		GTILIST
2.		BLDPCB
3. The SDB expansion and FSBs for any field level sensitivity are built here. The parentage flags in the SDB are set, the Intent List entry is built, and any propagation is done. The index SDB, if any, is initialized.		BLDPCB4
6. Space is allocated for the PCB, JCB, key feedback area, DSG, and Level Table.		BLDPCB5

Extended Description	Routine	Label

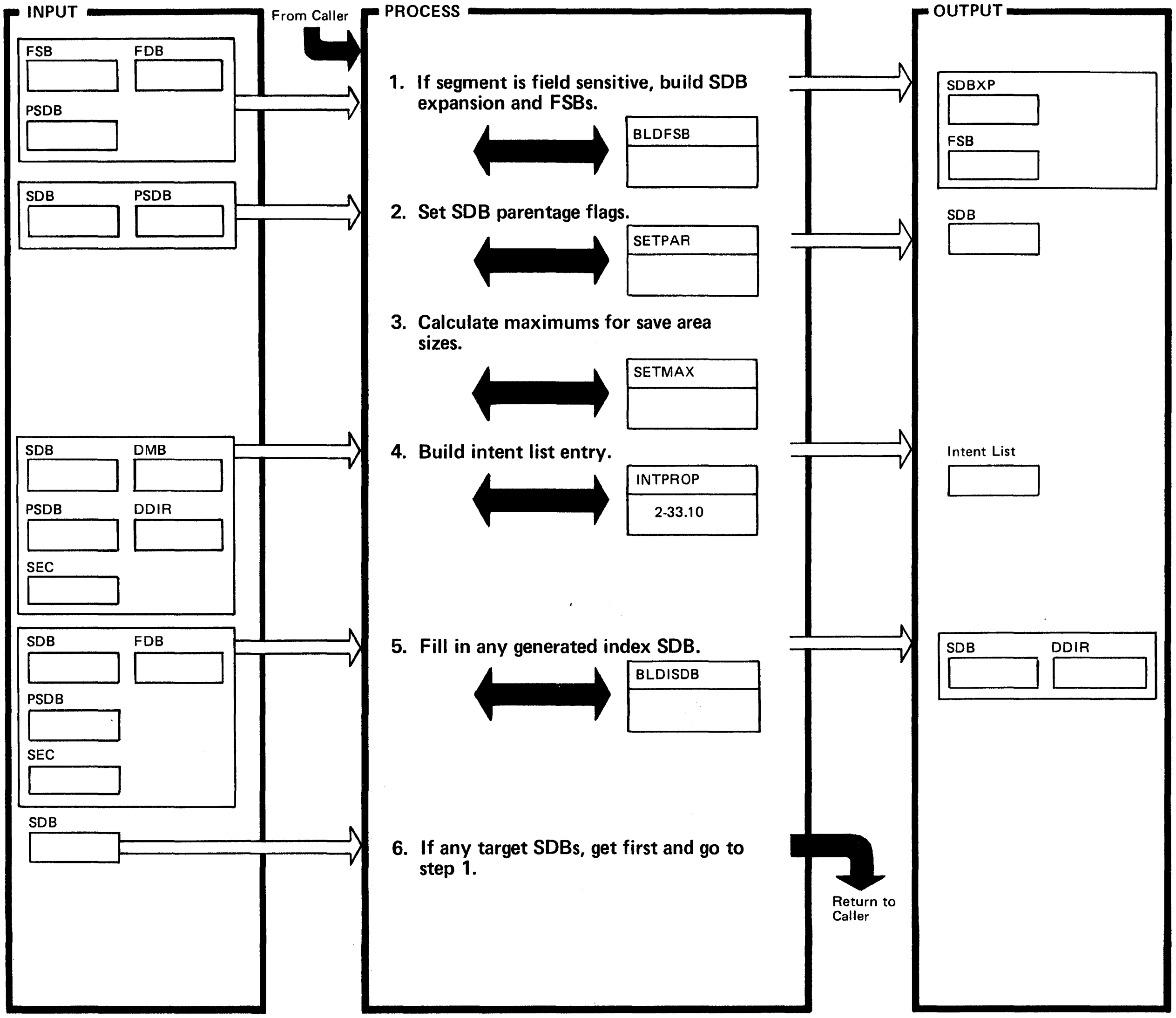
Figure 2-33.6. Block Builder Routine 3 (Part 2 of 2)



Extended Description	Routine	Label

Extended Description	Routine	Label

Figure 2-33.7. Block Builder BLDSDB Routine



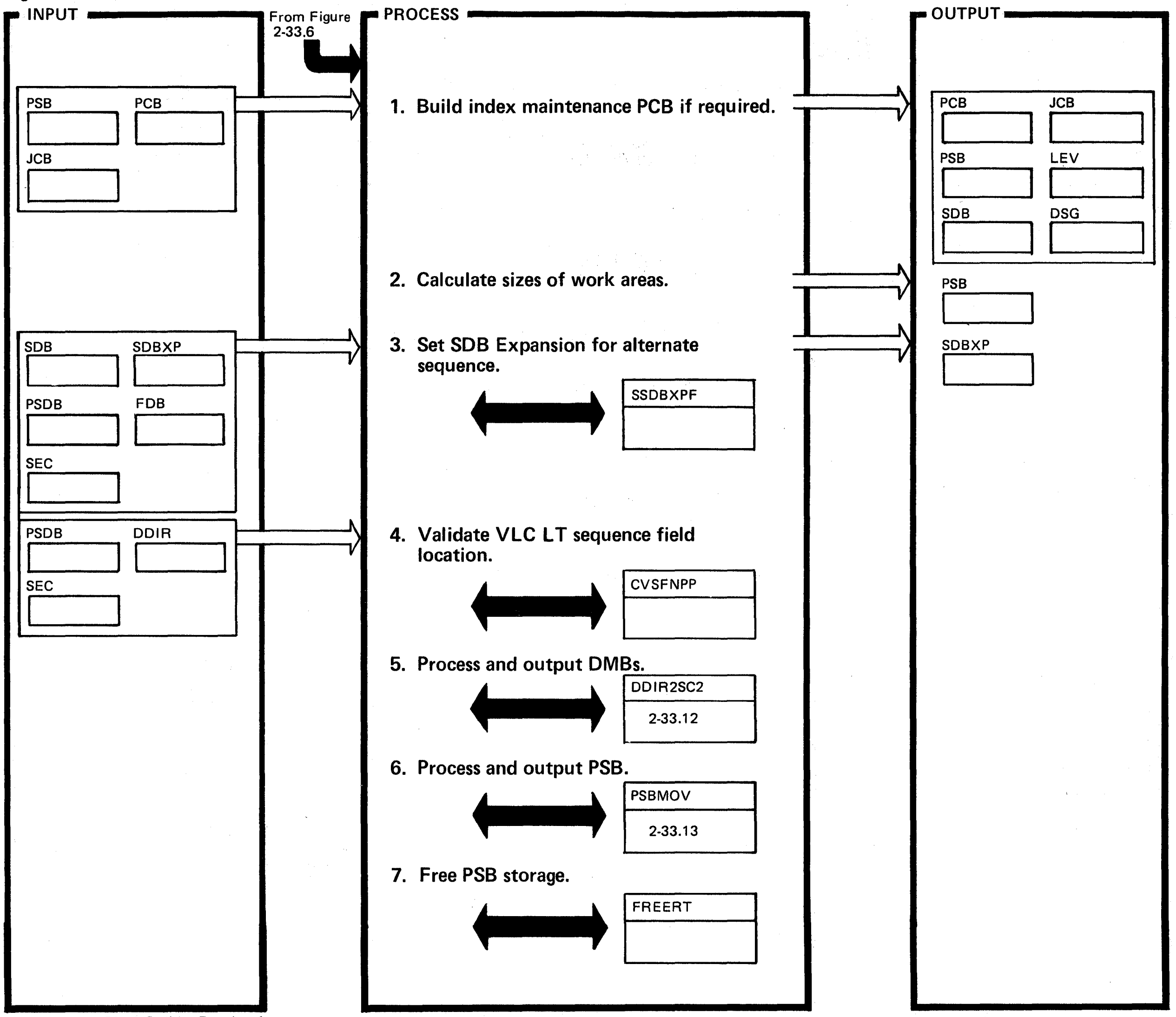
DLZDLBL2 — Block Builder Routine 3

DLZDLBL2

Extended Description	Routine	Label
1.		BLDSDB2
2.		BLDSDB3
3. Maximums set are: a. Maximum segment length in either physical or user's view. b. Maximum concatenated key length. c. Maximum concatenated segment length. d. Longest segment at this level and path sensitive.		
4. a. Output message DLZ909I if PROCOPT changed. b. All intent propagation is done here.		INTENTL
5. Fill in the generated index SDB for HIDAM primary indexes or alternate sequence.		BLDSDB4

Extended Description	Routine	Label

Figure 2-33.8. Block Builder Routine 4 (Part 1 of 2)



DLZDLBL3 – Block Builder Routine 4

DLZDLBL3

Extended Description	Routine	Label
1.		BLDEND
2. Index work area, Index I/O area, segment compression work area, and I/O work area		CALWAS
5.		DDIR2SC2
6.		PSBMOV
7.		FREEPSB

Extended Description	Routine	Label

Figure 2-33.8. Block Builder Routine 4 (Part 2 of 2)

INPUT

DTADMB

PROCESS

8. Search DMBNAME table for DMB that needs a utility PSB. Go to Step 11 if none.

9. Build utility PSB.

DLZDPSB0
2-33.14

10. Go to PSBPASS in module DLZDLBL0 (Step 1., Figure 2-33.2) to process utility PSB just built.

11. Free DBD storage.

12. Return.

To Figure 2-33 Step 9

OUTPUT

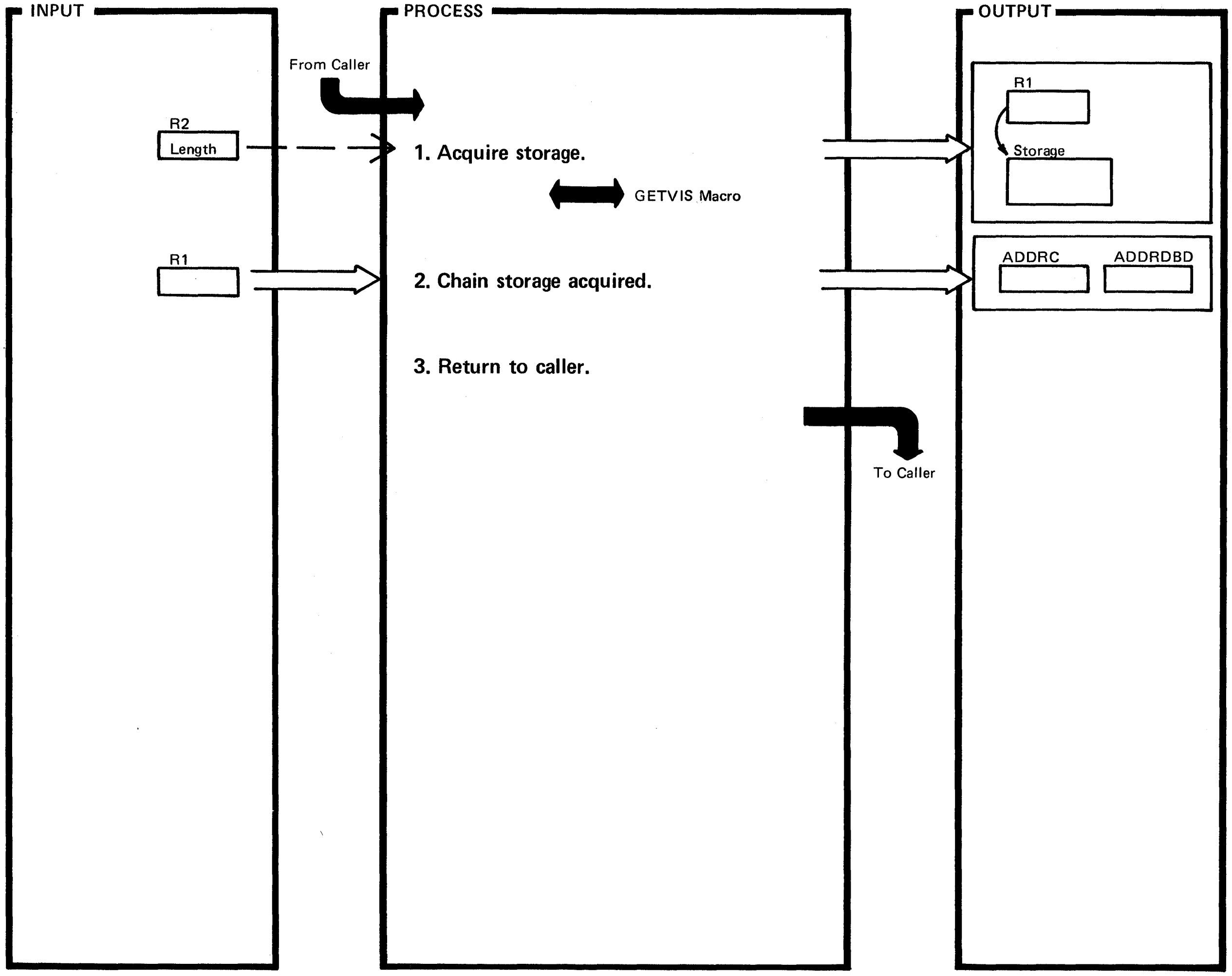
DLZDLBL3 – Block Builder Routine 4

DLZDLBL3

Extended Description	Routine	Label
8. Utility PSB is built for every HISAM, HIDAM, HDAM, and secondary index DMB just outputted.		UTILPSB
9. The dmbname is moved to the psbname location and a suffix 'u' added. The no utility PSB required indicator is turned on at DTABFLAG so we don't try to build another utility PSB for this DBD the next time around. Return to Step 9 to build the PSB. The output of DLZDPSB0 is like PSBGEN output.		
11.		FREEDBD
12. PSTERLOD = 0 if OK and non-zero if not.		RETURN

Extended Description	Routine	Label

Figure 2-33.9. Acquire Storage Routine



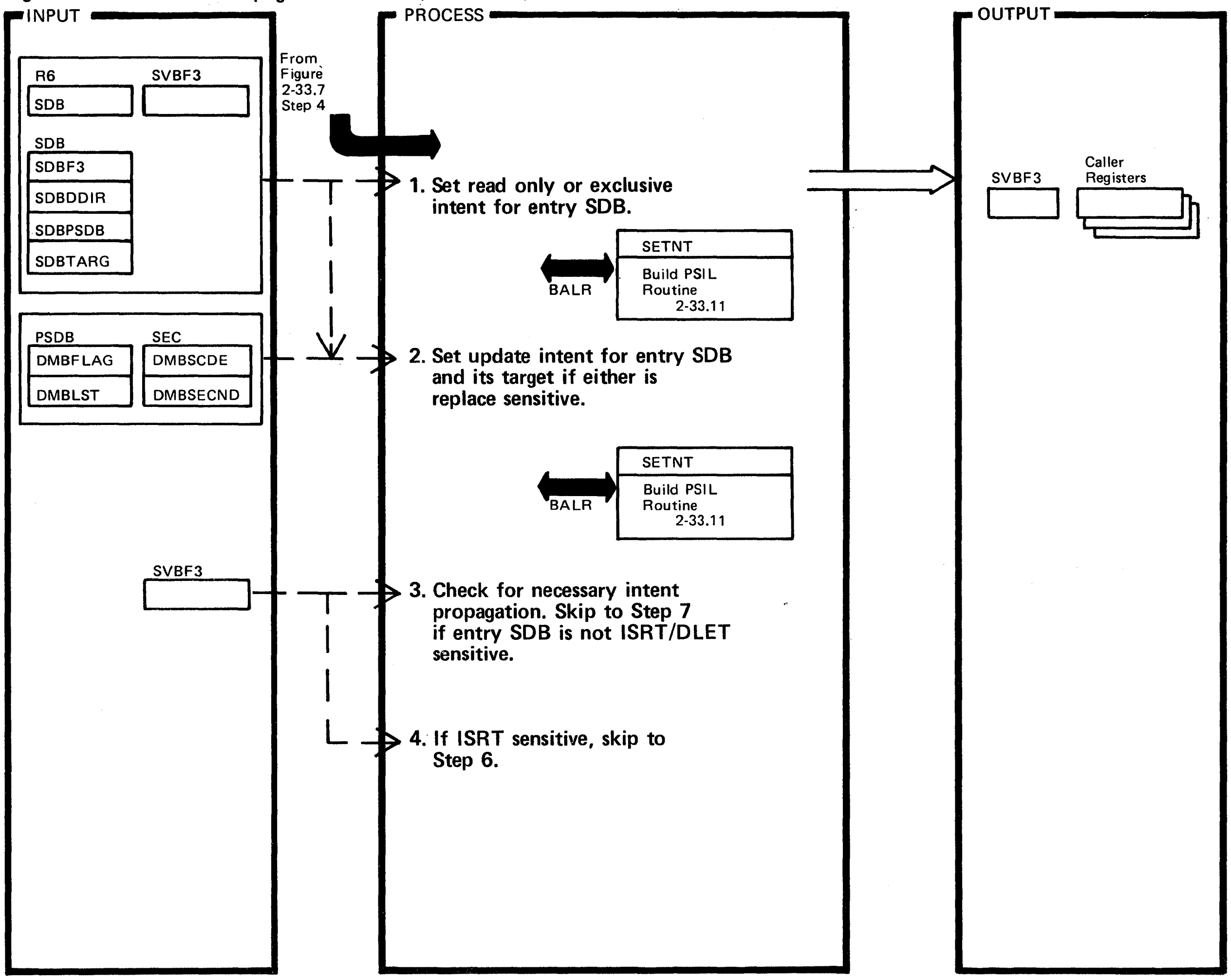
DLZDLBL0 — Batch Control Block Builder CSECT

DLZUACB0

Extended Description	Routine	Label
1. Write message DLZ905I if GETVIS space is not available.		GETSTOR
2. If this is a storage request for a DBD, the storage is chained off at ADDRDBD rather than ADDRC.		GETSTOR1

Extended Description	Routine	Label

Figure 2-33.10. Intent Propagation Routine (Part 1 of 2)



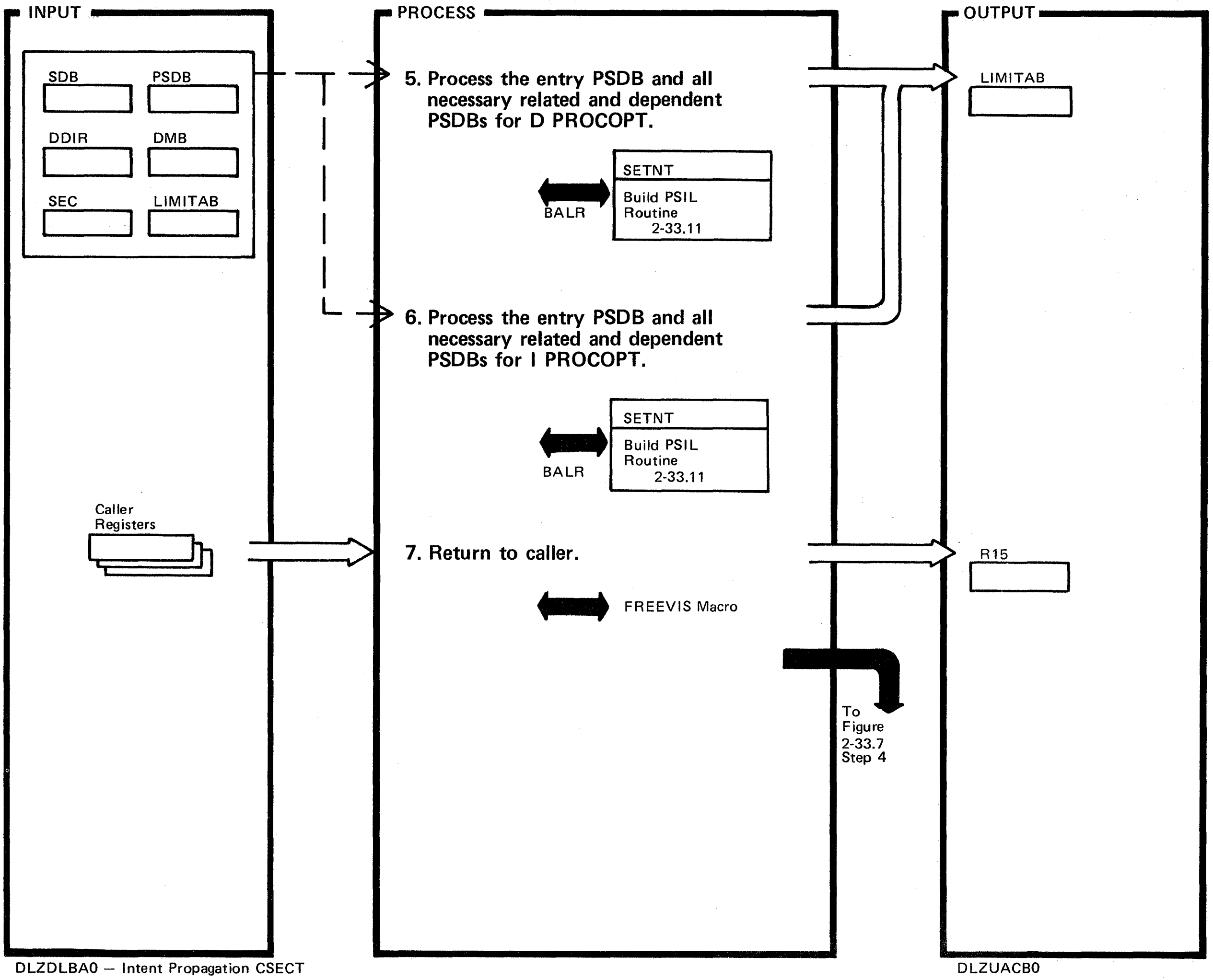
DLZDLBA0 – Intent Propagation CSECT

DLZUACB0

Extended Description	Routine	Label
1. This routine propagates intent to all PSDBs related or dependent on the PSDB for the entry SDB. Only SDBs that are built directly from SENSEG statements in the associated PSB are passed to this routine. For a process option (PROCOPT) of G, the entry SDB and all targets are set to exclusive. For E PROCOPT, the entry SDB and its immediate target are checked for key sensitivity. If the SDB is data sensitive, the intent is set to exclusive. Otherwise, no intent is set.		INTPROP
2.		SETREPL
3.		PROPTYPE
4.		UPTYPE

Extended Description	Routine	Label

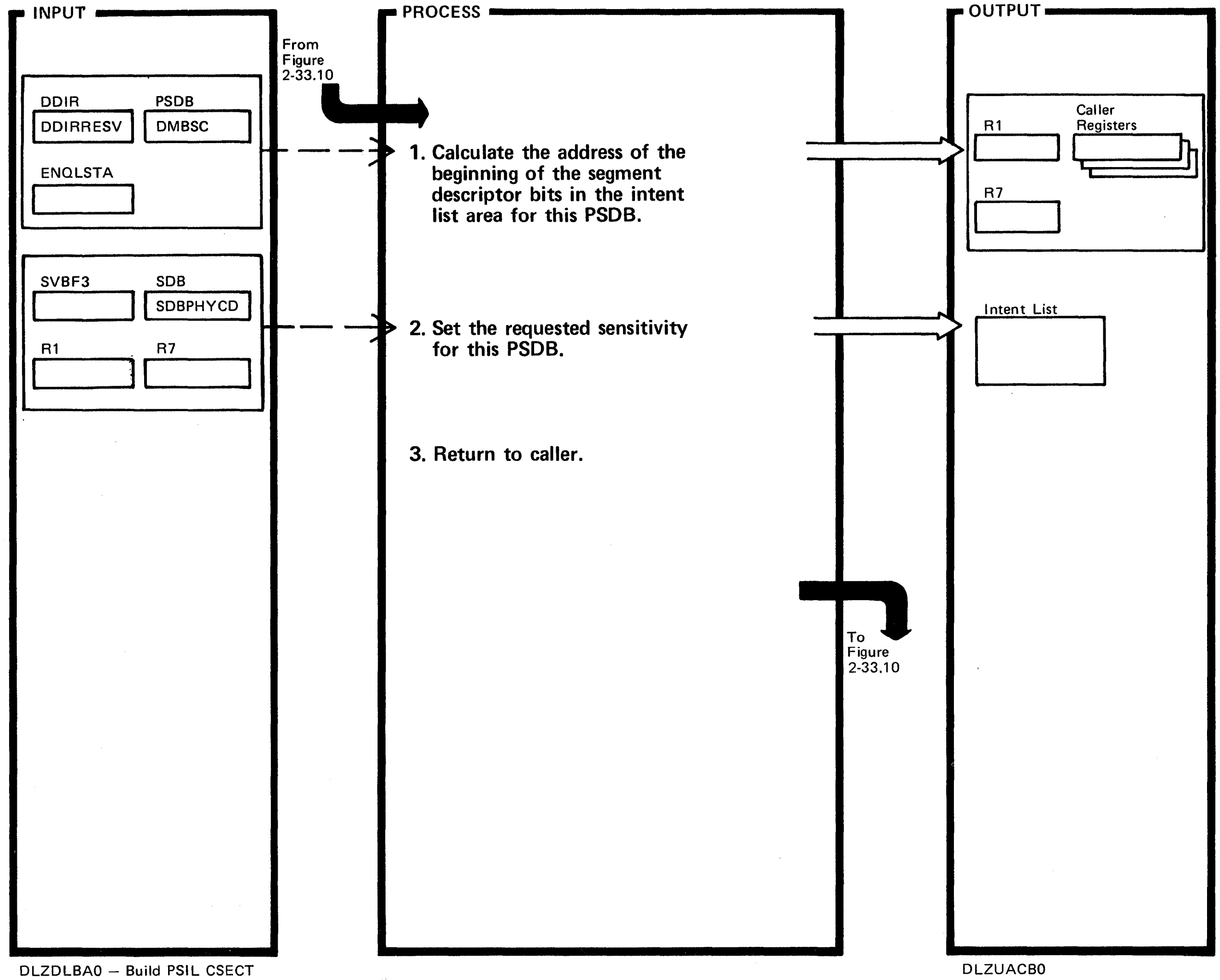
Figure 2-33.10. Intent Propagation Routine (Part 2 of 2)



Extended Description	Routine	Label
5. Any intent set will be update type. Storage is acquired for a limit table and is constructed with PSDB addresses. These addresses show children of the entry PSDB, any necessary higher related PSDBs, any index relationships, any logical children, the logical parent, physical parent, and physical pair of the entry PSDB. After the table is constructed, each entry is passed to SETNT to set update intent and the table area is freed. Exclusive intent is propagated along with delete.		DLETPA
6. Any intent set will be update type. A limit table is constructed with addresses of those PSDBs that will be passed to SETNT.		UPISRT UPCHKPP
It is also determined if the entry SDB can insert its logical parent or physical parent as a result of a concatenated segment definition. If it can, the logical parent or physical parent is processed in the same manner as the entry PSDB. The physical pair is also processed if one exists.		

Extended Description	Routine	Label
After processing the LIMITAB entries built, the area is freed.		
7. Set the return code and make sure all the limit tables are freed.		INTRETO

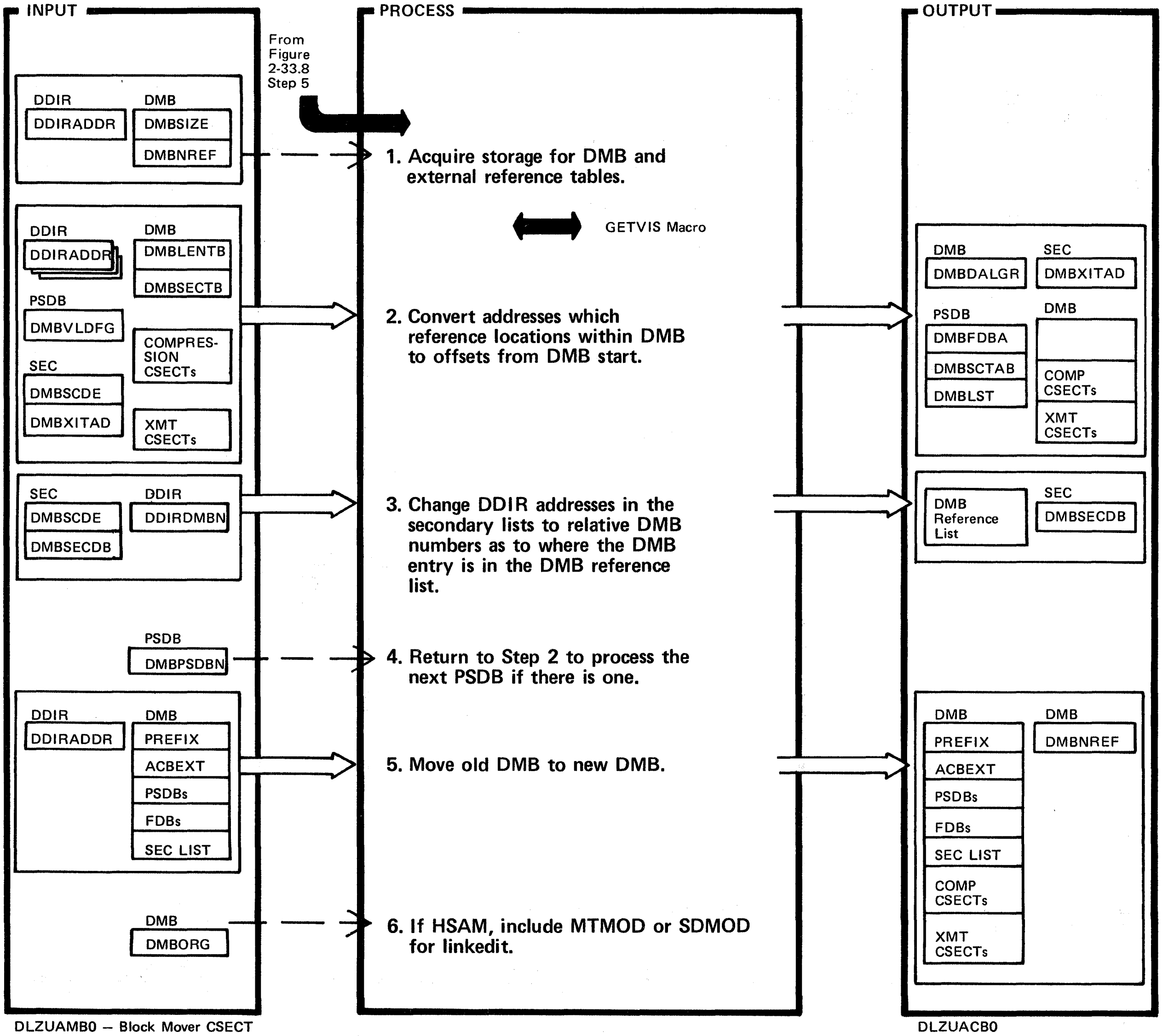
Figure 2-33.11. Build PSIL Routine



Extended Description	Routine	Label
1.	DLZDLBA0	SETNT
2.		ENQ1

Extended Description	Routine	Label

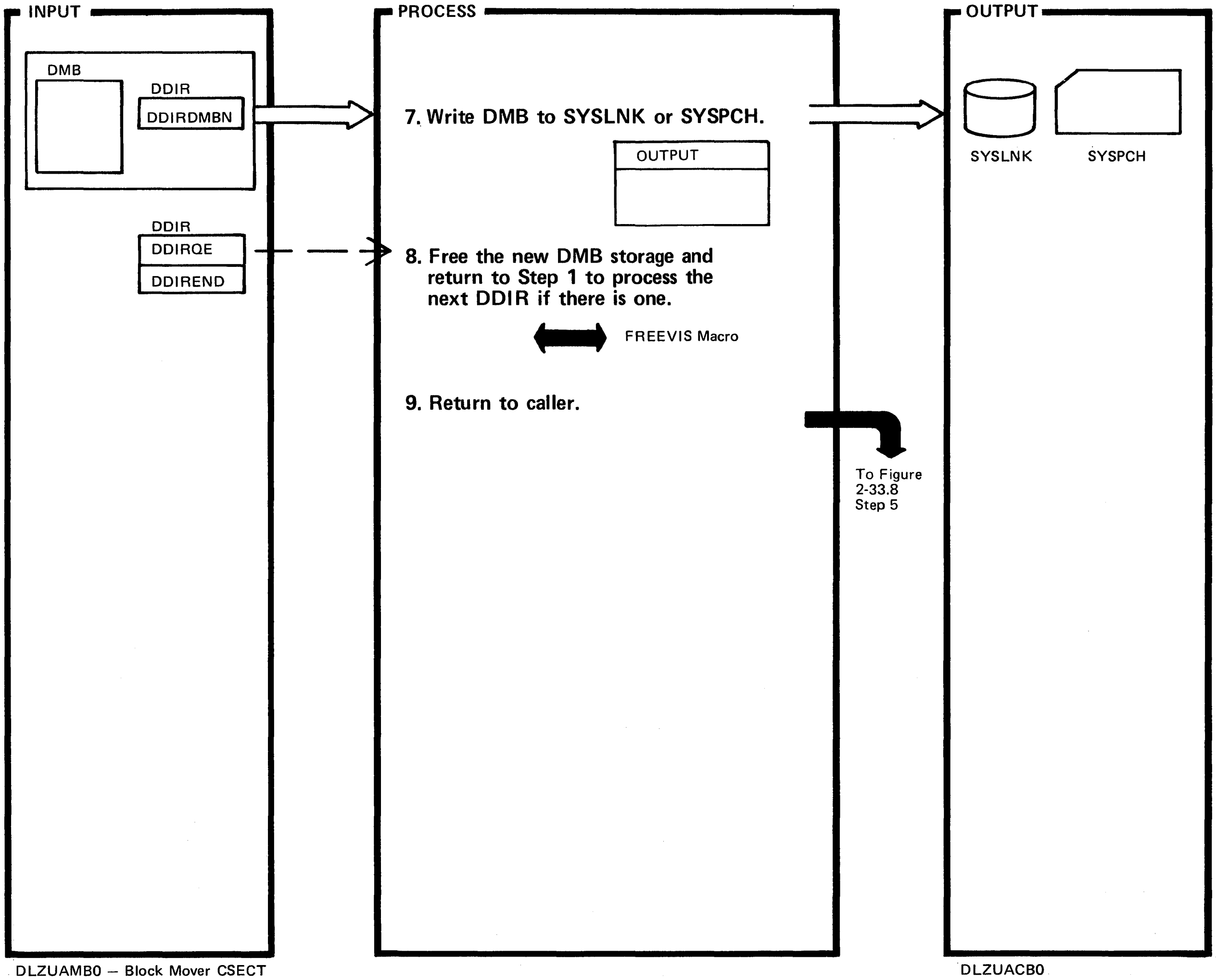
Figure 2-33.12. Write DMBs (Part 1 of 2)



Extended Description	Routine	Label
1. Process all DMBs referenced in DDIR entries unless already built or LOGICAL. Write message DLZ905I for GETVIS error.		DDIR2SC2
2. The compression CSECTs and index maintenance CSECTs are moved to the new DMB. Write message DLZ570I for an invalid SEC list code found in a DMB.		DMBREL
3. The reference list is the last part of the DMB. If the DMB is not in the list, it is added. Repeat this step for each SEC.		SECREL
4.		PSDBREL2
5. Any addresses which do not fall within the DMB are set to zero.		DMBOUT

Extended Description	Routine	Label
6. The names in the relocatable library of the required modules are DLZTAPE or DLZDISKI and DLZDISKO.		DMBOUT1

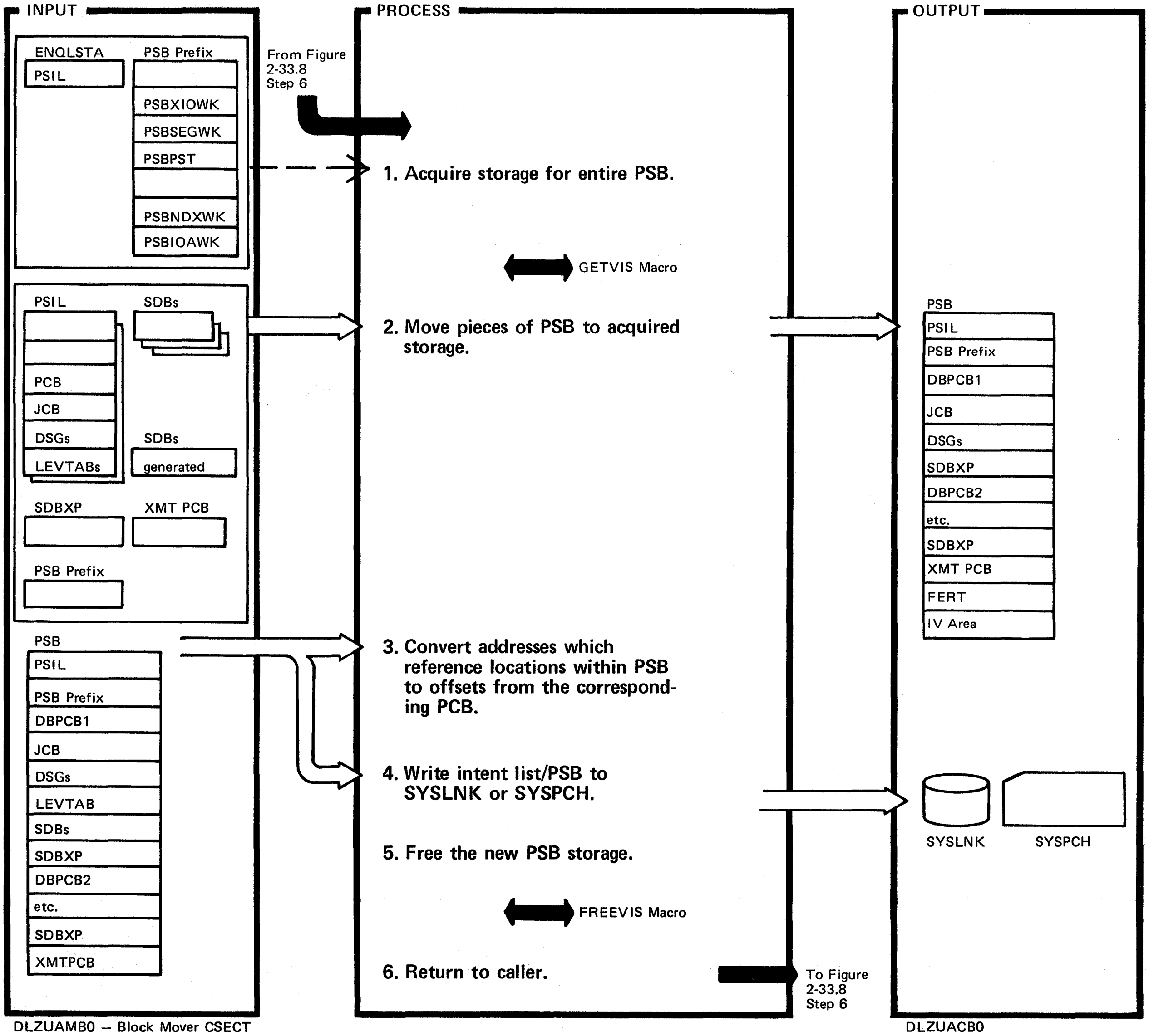
Figure 2-33.12. Write DMBs (Part 2 of 2).



Extended Description	Routine	Label
7. The same subroutine is used for the PSB also.		DMBOUT3
8. Write message DLZ926I for a FREEVIS error.		

Extended Description	Routine	Label

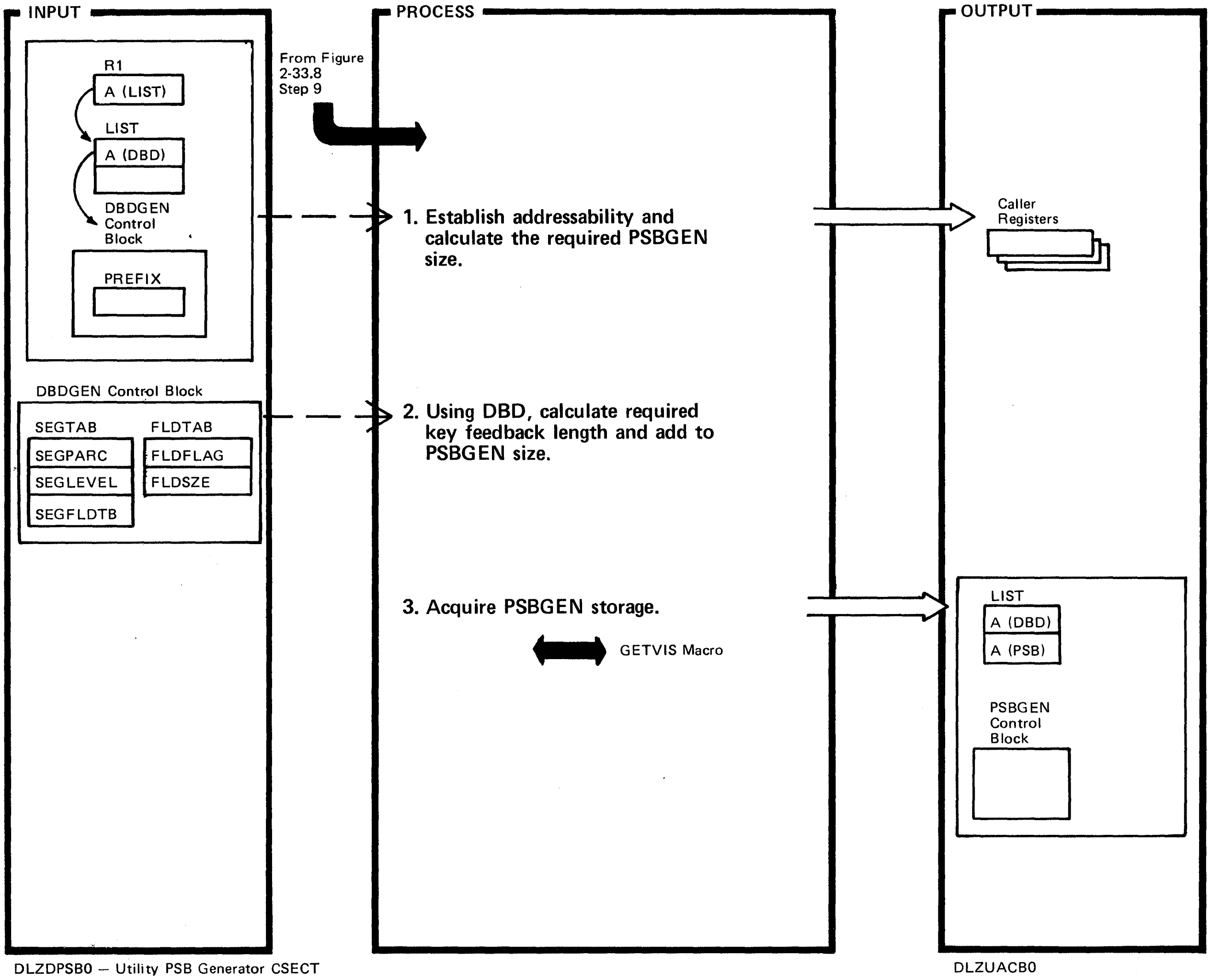
Figure 2-33.13. Write PSB



Extended Description	Routine	Label
1. The size calculation formula is PSBPST – PSBXIOWK – PSBSEGWK – PSBNDXWK – PSBIOAWK + length of PSIL. Write message DLZ905I for GETVIS error.		PSBMOV
2.		PCBMOV SDBMOV FSBMOV FRIVMOV
3.		FERTREL PCBREL DSGREL SDBREL FSBREL
4.		PSBOUT
5. Write message DLZ926I for FREEVIS error.		

Extended Description	Routine	Label

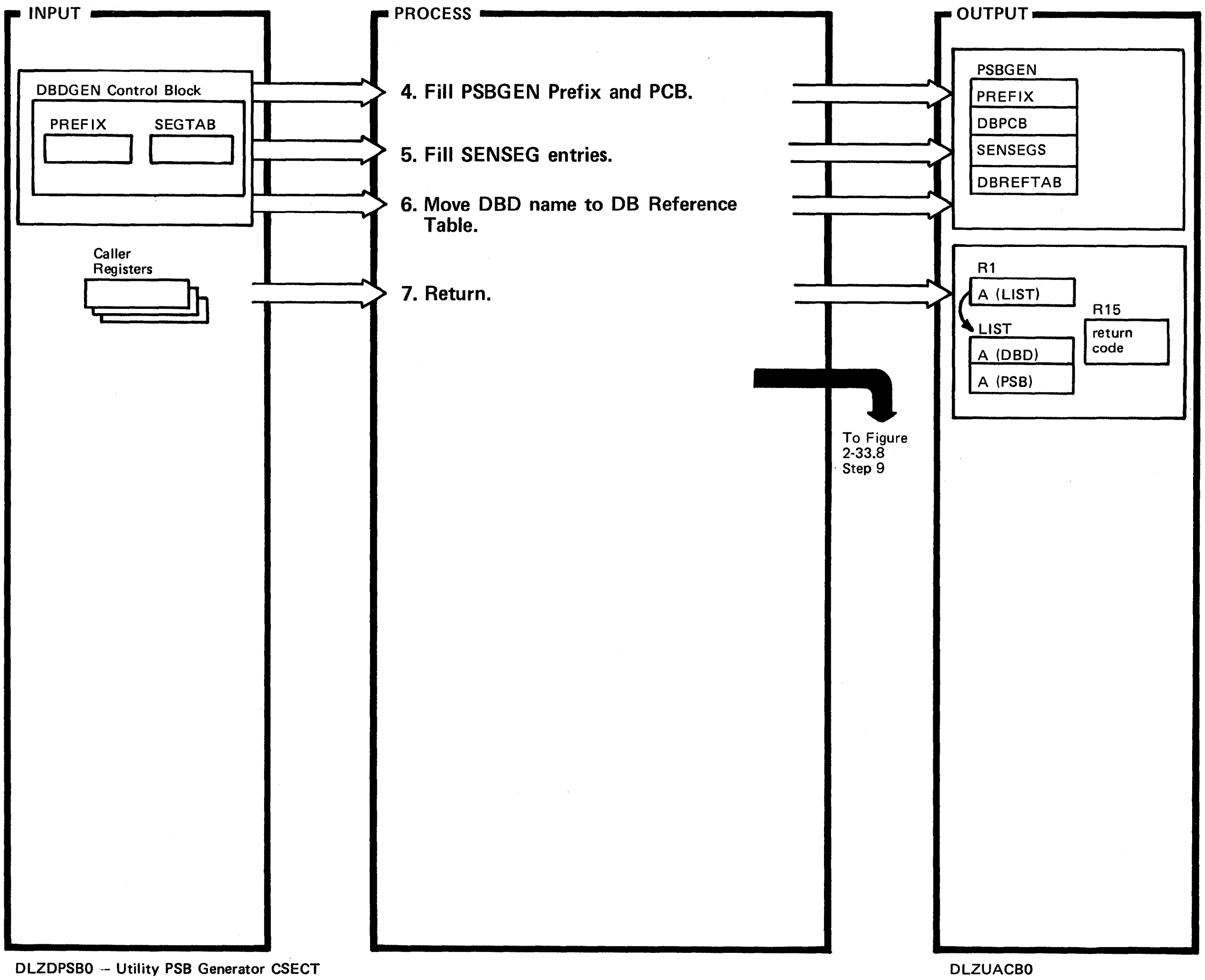
Figure 2-33.14. Build PSB (Part 1 of 2)



Extended Description	Routine	Label
1. Routine identifier DLZDPSB0 vrnp is defined here. A parameter list containing DBD address is passed in Register 1. The contents of this DBD are used to create the utility PSB.	DLZDPSB0	DLZDPSB0 INIT
The PSBGEN size will be the fixed size plus the number of segments times the length of SENSEG entry.		
It is possible to have an invalid access method error to pass back to DLZDLBL0.		
2. The result will be stored in PSB Prefix.		SEGLOOP GETKEYSZ
3. The area is also cleared to zeros. Write message DLZ905I for GETVIS error.		USECURR1

Extended Description	Routine	Label

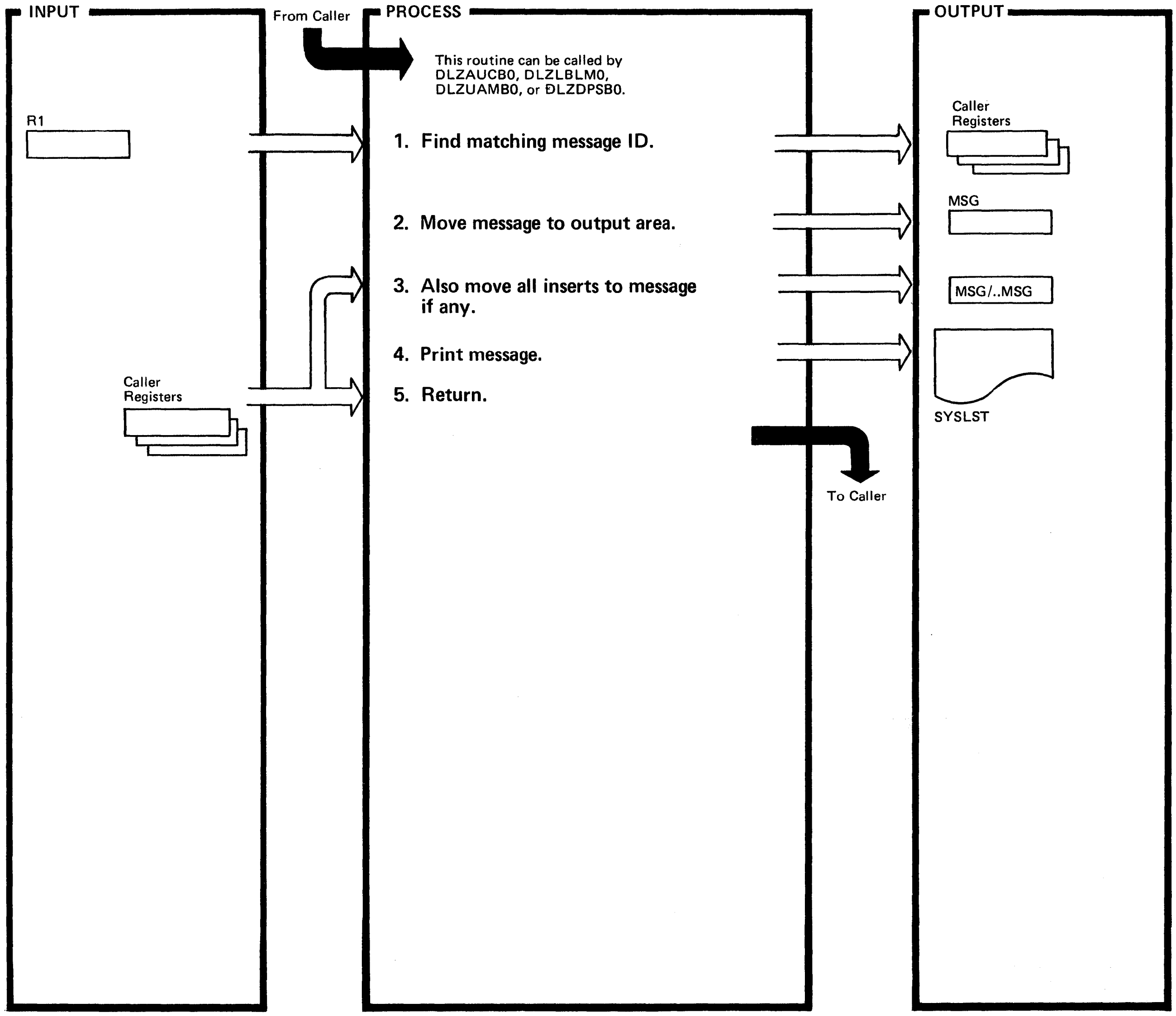
Figure 2-33.14. Build PSB (Part 2 of 2)



Extended Description	Routine	Label
4. PROCOPT of 'A' is set in PCB for all DBDs except secondary index where 'LS' is set. The 'A' is changed to the proper 'load' by batch initialization if necessary.	DLZDPSB0	CLRDONE
5. Same PROCOPT as in Note 4.		PSEGLOOP
6. In addition, no SORTAB is indicated.		SETDBREF
7. The address of the built utility PSB is returned to the caller in the parameter list.		RETURN

Extended Description	Routine	Label

Figure 2-33.15. Message Writer



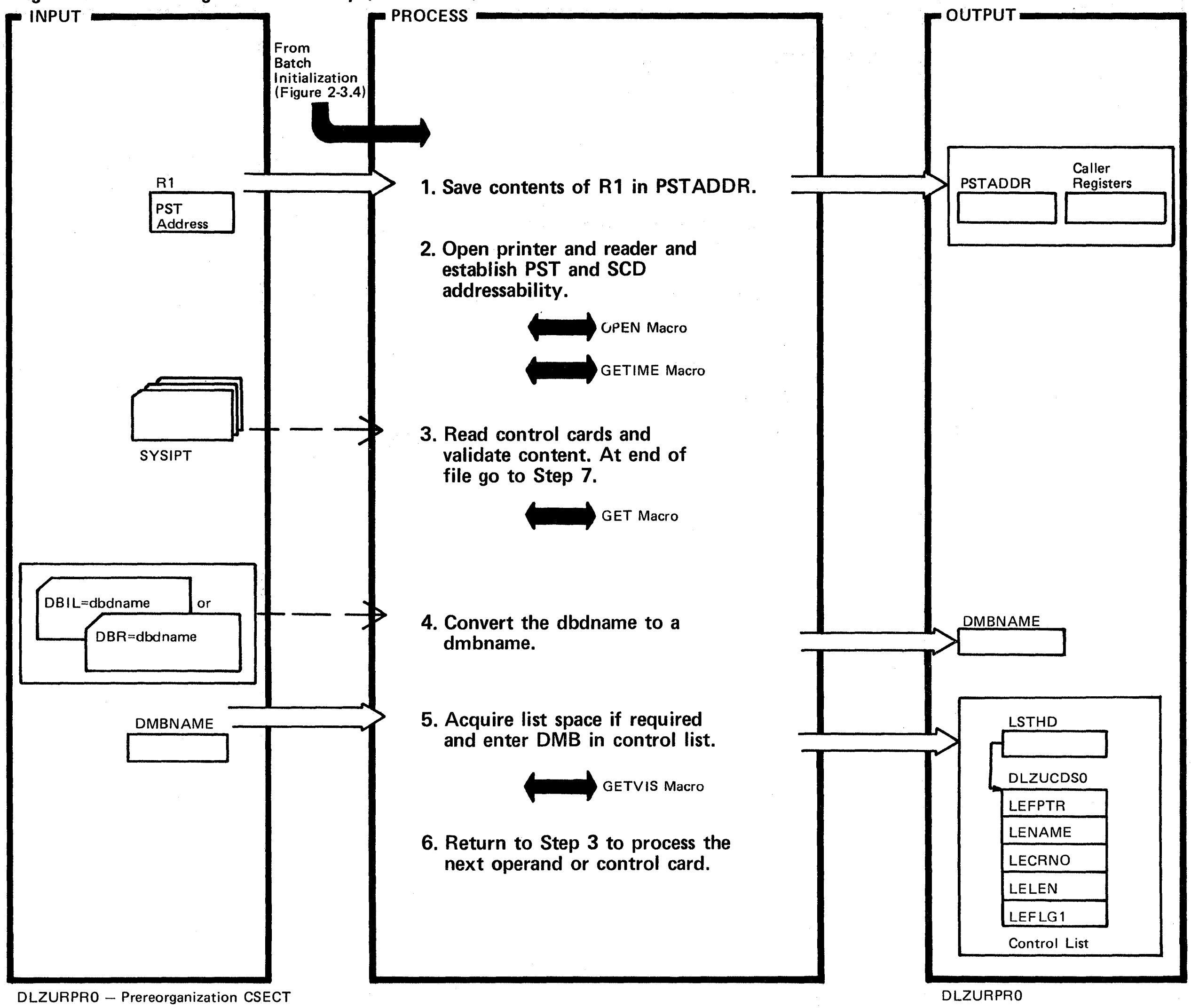
DLZLBLM0 – Message Writer CSECT

DLZUACB0

Extended Description	Routine	Label
4. A subroutine in DLZUACB0 CSECT is called to do the actual printing.		

Extended Description	Routine	Label

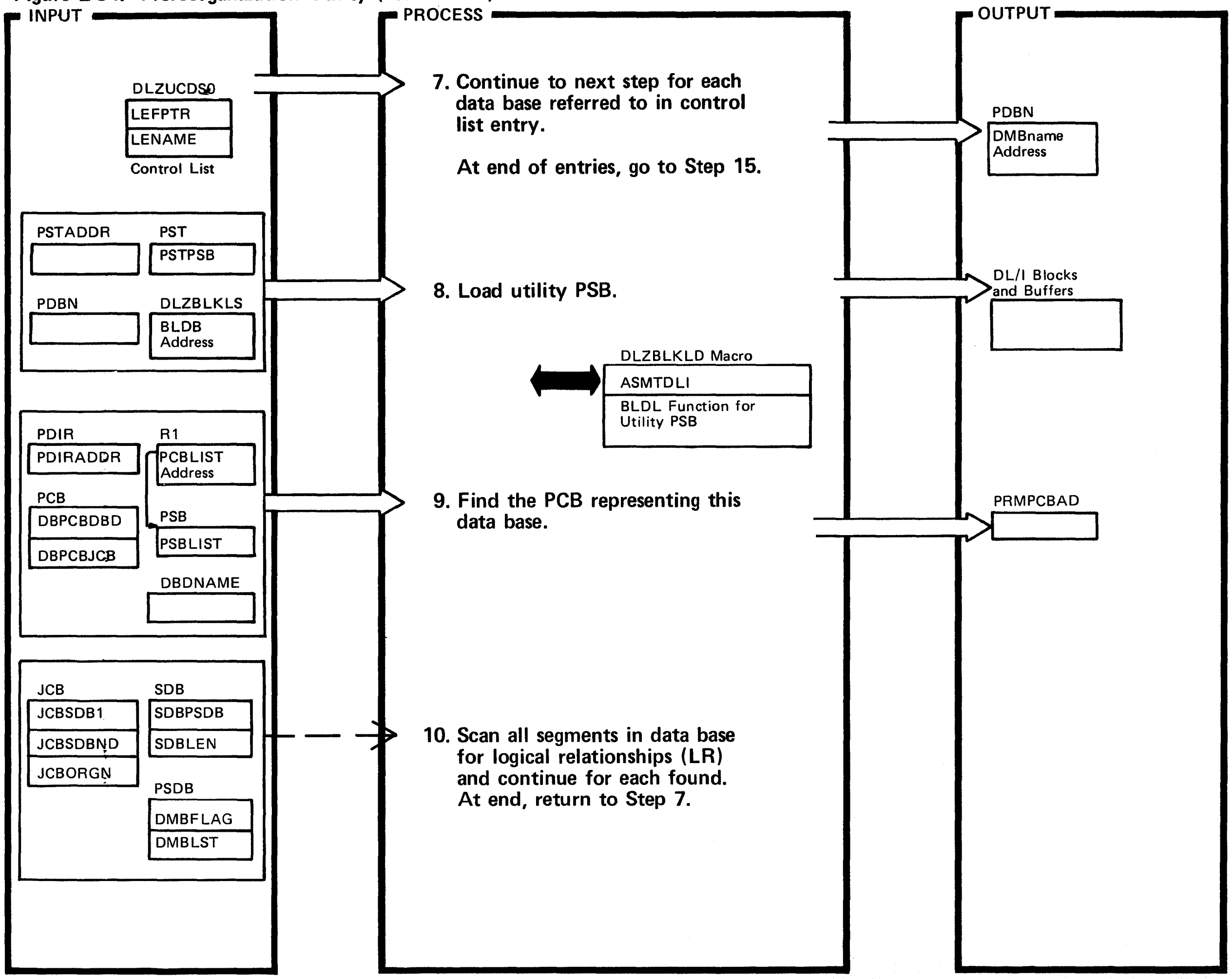
Figure 2-34. Prereorganization Utility (Part 1 of 4)



Extended Description	Routine	Label
1. Module identifier (DLZURPR0vrnp) is defined here. This utility executes as 'ULU' under DL/I control. No blocks or buffers have been loaded yet. Only the nucleus exists. Batch initialization passes the PST address to the logical relationship utilities rather than a PCB address.	DLZURPR0	DLZURPR0 HERE
2.		OPEN1
3. Control card contains identifier as DBIL= (initial load), DBR= (reorganize), OPTIONS=, and dbdnames. In case of an input control card format error, message DLZ954I is printed and job terminates.		NXTCR
4.		DBIREC DBRREC

Extended Description	Routine	Label
5. Control list entries contain DMB names of data base and user options specified in control cards. Macro DLZUCDS0 contains the DSECT defining the format of a control list entry. Write message DLZ963I if an entry already exists for a dmbname. Write message DLZ965I if the number of control list entries exceeds the maximum of 20. Write message DLZ391I if GETVIS fails.		LSTINS

Figure 2-34. Prereorganization Utility (Part 2 of 4)



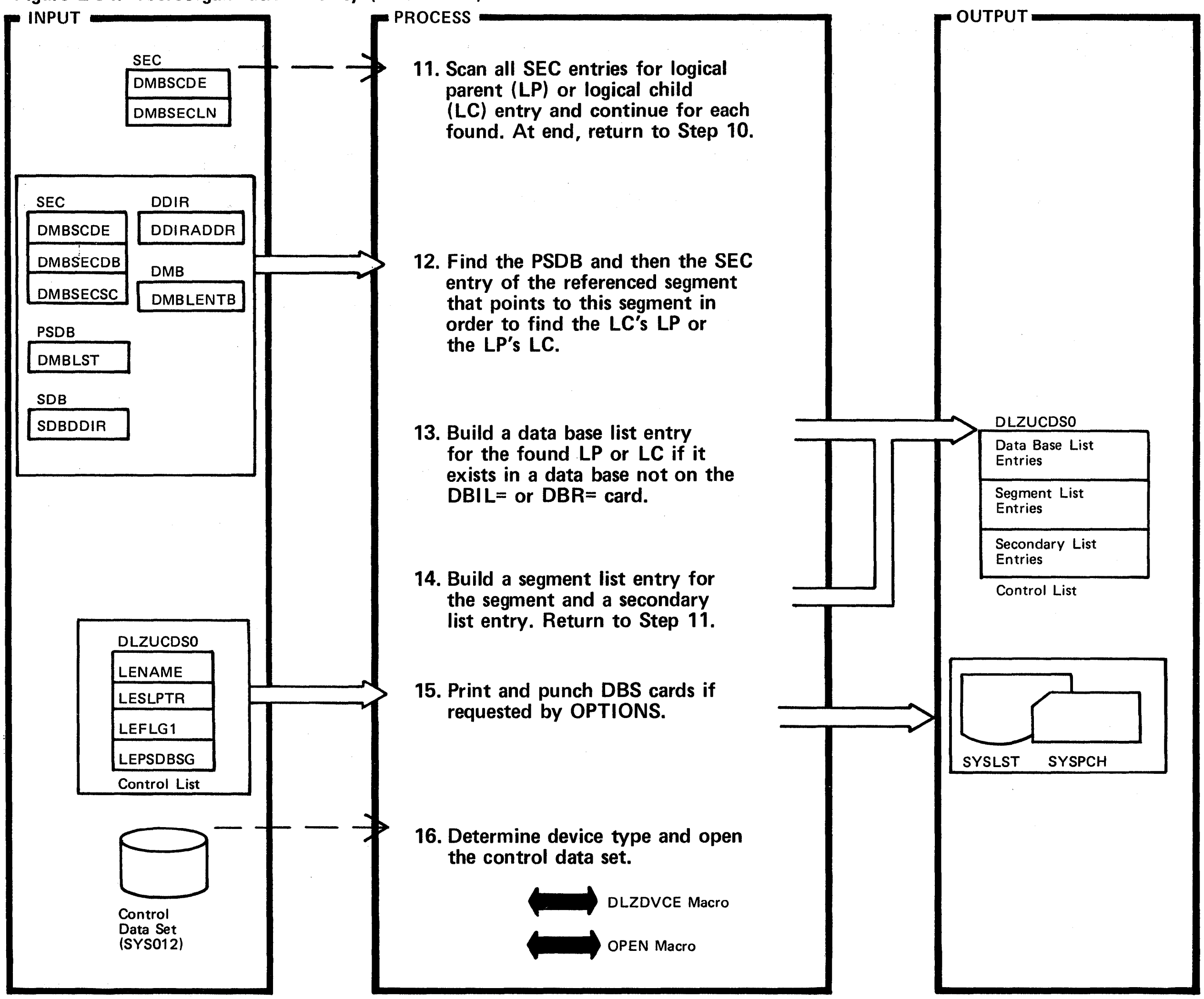
DLZURPR0 — Prereorganization Utility

DLZURPR0

Extended Description	Routine	Label
7. Write message DLZ964I for no DBIL or DBR control cards. Write message DLZ976I if the dbdname specified is an index DBD.	DLZURPR0	SCAN SCAN10
8. The DLZBLKLD macro moves the dmbname to PST at PSTPCPSB and sets utility suffix 'U' and calls DL/I with the BLDB call function. Write message DLZ956I for a data base control block build failure or if there is no PCB. R1 is returned with the address of the PSBLIST.		BLDBLKS
9.		NEWDBLP
10. If a LP/LC exists for a segment, continue to the next step to look at the secondary list entries for the segment. After each SDB with a LR has been processed return to Step 7 to process the next data base control list entry.		SCAN20

Extended Description	Routine	Label

Figure 2-34. Prereorganization Utility (Part 3 of 4)



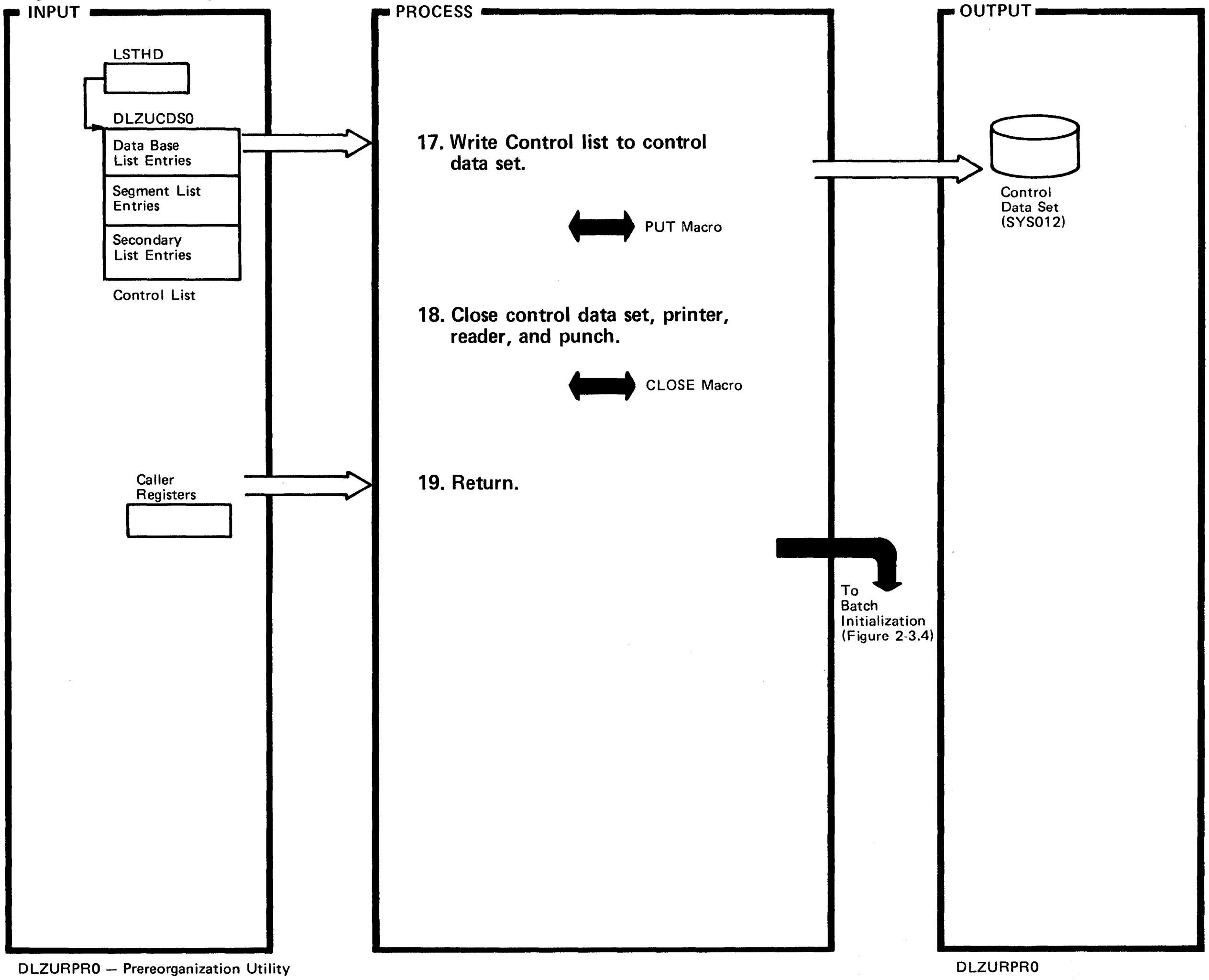
DLZURPR0 — Prereorganization Utility

DLZURPR0

Extended Description	Routine	Label
11.	DLZURPR0	SCAN50
12. The correct SEC is found by comparing the dbdname in the SEC to the dbdname in the SDB. Each SEC for the segment is examined to see if it is a LP or LC entry. Write message DLZ965I if a SEC is not found with a matching dbdname.		SCAN60
13. Write message DLZ985I for a limit check failure.		SCAN110 SCAN140
14. Segment entries contain segment names involved in logical relationships (LR). Secondary list entries contain DMB names which refer to logically related data bases. 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 (Step 17).		

Extended Description	Routine	Label
15. DBS indicates data base must be scanned using SCAN utility (DLZURGS0). Write message DLZ962I to list segments scanned.		SCNLST NXTDB
16. Write message DLZ984I if there is an invalid device assignment for SYS012.		SCAN500 OPENCTL

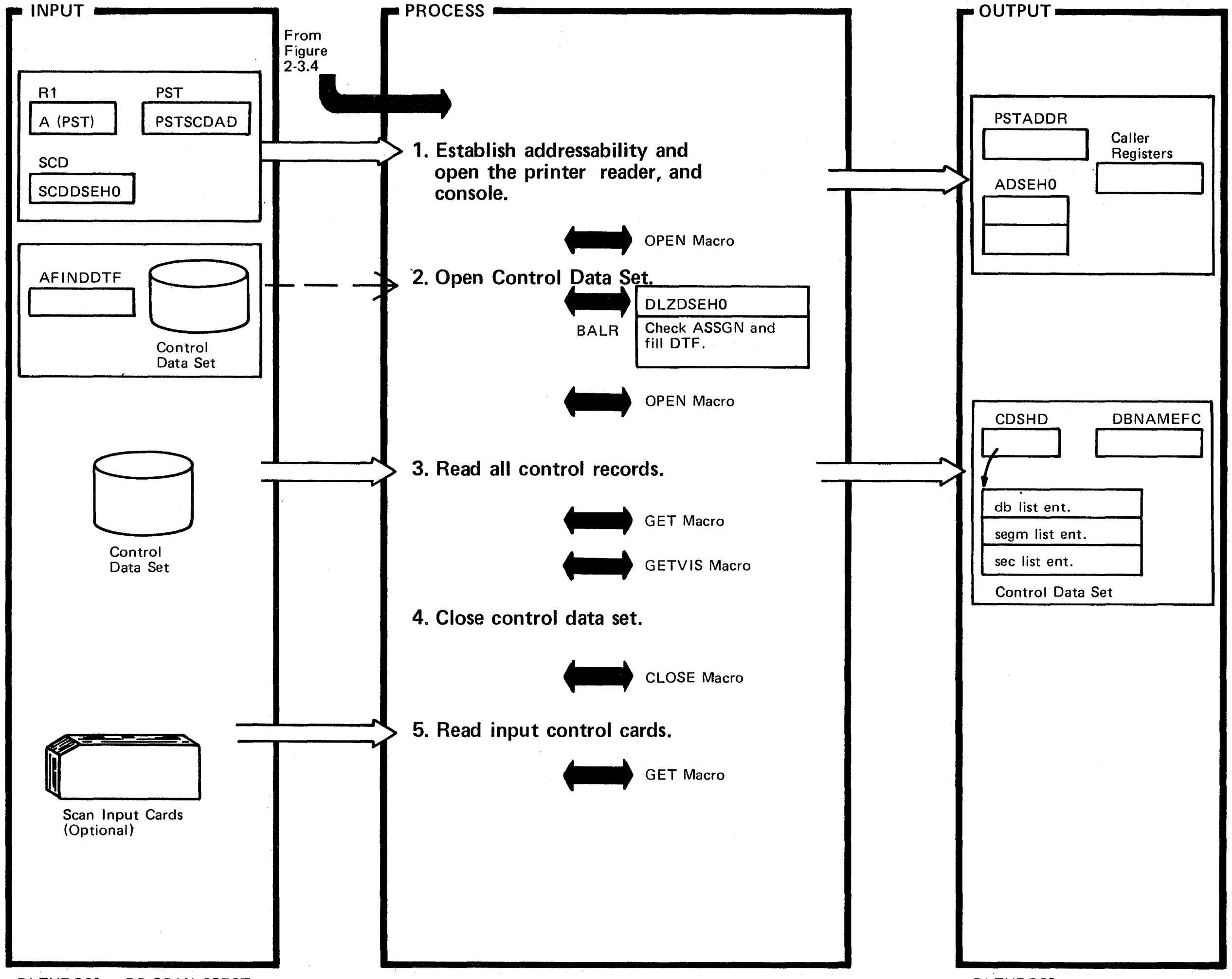
Figure 2-34. Prereorganization Utility (Part 4 of 4)



Extended Description	Routine	Label
17.	DLZURPR0	SCLPS
18. Write message DLZ966I for a normal termination.		SCAN700 TERM
19.		GOODRET

Extended Description	Routine	Label

Figure 2-35. DB SCAN (Part 1 of 8)



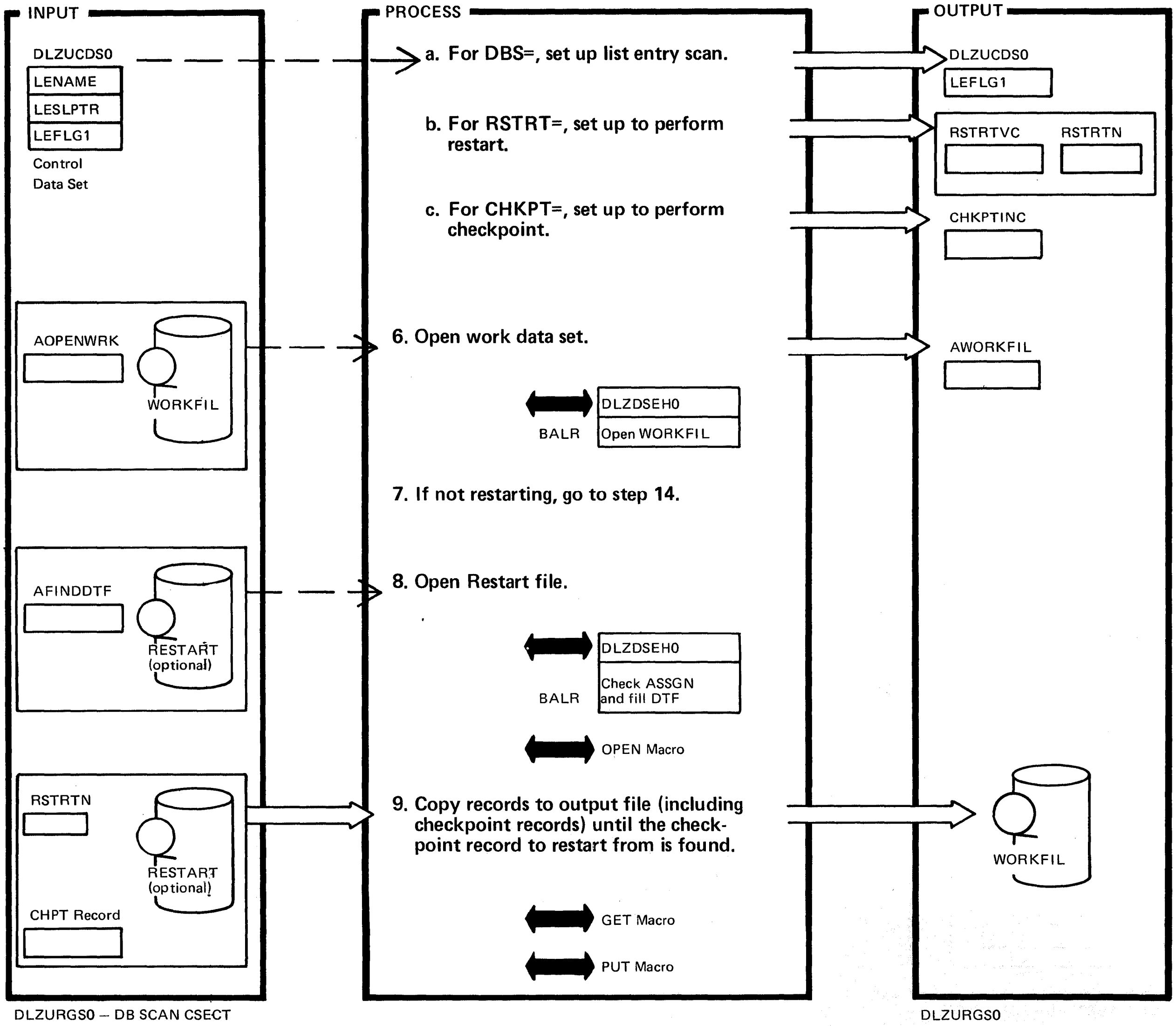
DLZURGSO — DB SCAN CSECT

DLZURGSO

Extended Description	Routine	Label
1. Module identifier is defined here. This utility executes as 'ULU' under DL/I control. No blocks or buffers have been loaded yet. Only the nucleus exists. Batch Initialization passes the PST address to the logical relationship utilities rather than a PCB address. The DLZDSEHO prefix contains addresses to routines within DLZDSEHO. The routines are used by this utility. The addresses are moved to a constant area beginning at ADSEHO.	DLZURGSO	DLZURGSO OPEN1
2. The 'FINDDTF' subroutine of DLZDSEHO is used to check that SYS012 is properly assigned to a disk and to fill the correct device type in DTF. Write DLZ984I if the control file is not assigned to a disk.		PROCCTL OPEN
3. The Control Data Set is moved from the I/O area to storage acquired by GETVIS for further processing.		

Extended Description	Routine	Label
3. (con't) Write DLZ391I for a GETVIS failure. Write DLZ957I for no control data set or, if the ID is not 'CONTROL DATA SET'.		
4.		CDSEOFB
5. Input on 'DBS=' card is used to modify Control Data Set in core. 'RSTRT=' and/or 'CHKPT=' specify checkpoint/restart capabilities. 'ABEND' card used for testing. Write DLZ954I for a control card format error.		NXTCR

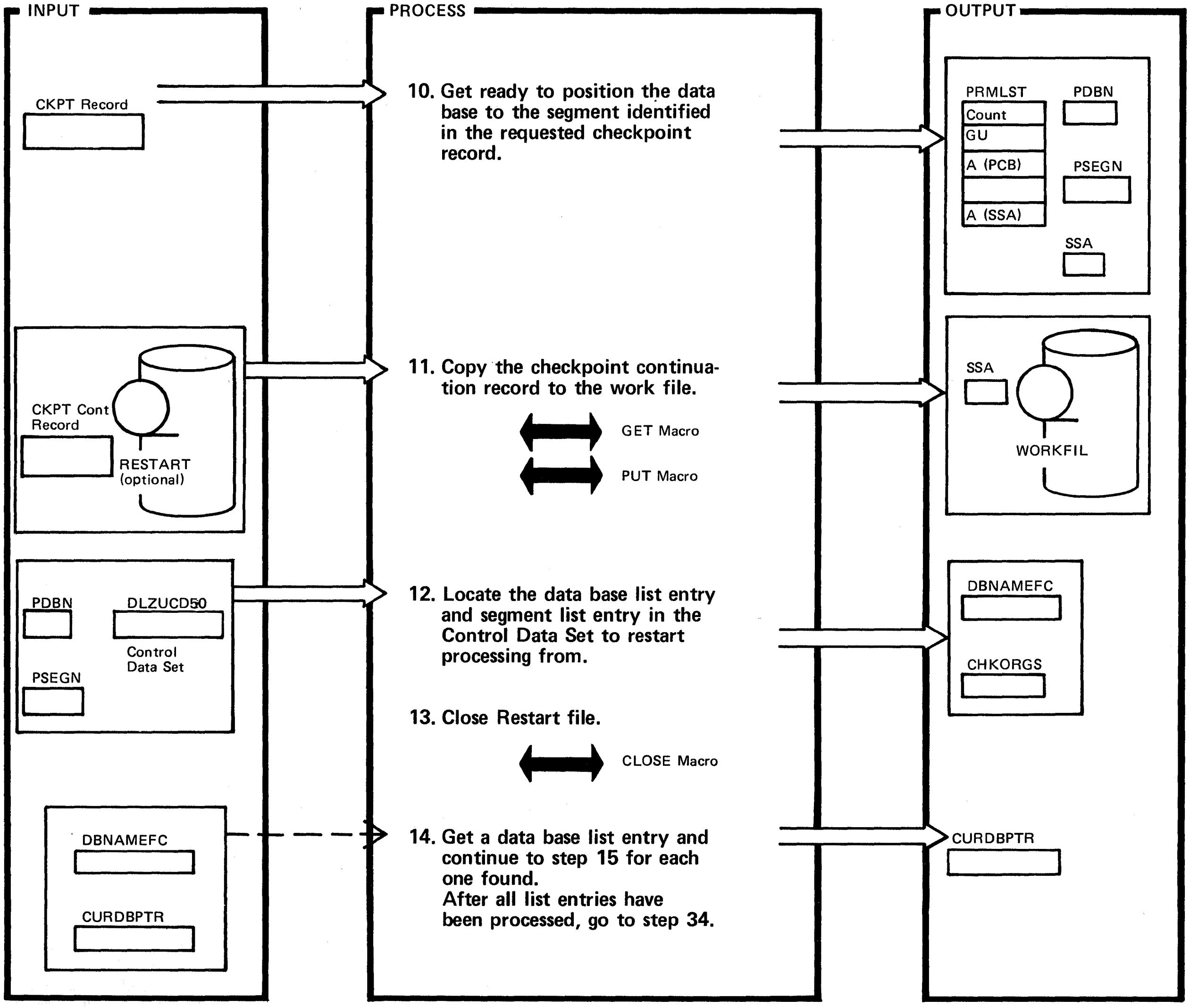
Figure 2-35. DB SCAN (Part 2 of 8)



Extended Description	Routine	Label
a.		DBSREC
b.		RSTRTREC
c.		CHKPTRC
6. Write DLZ984I for unknown or ignored device type.		SCAN
7.		SCAN 10
9. Restart records are copied from the previous WORKFIL until the specified checkpoint record is found. An SSA is set to do qualified GU on last segment to reestablish position.		RSTRT40

Extended Description	Routine	Label

Figure 2-35. DB SCAN (Part 3 of 8)



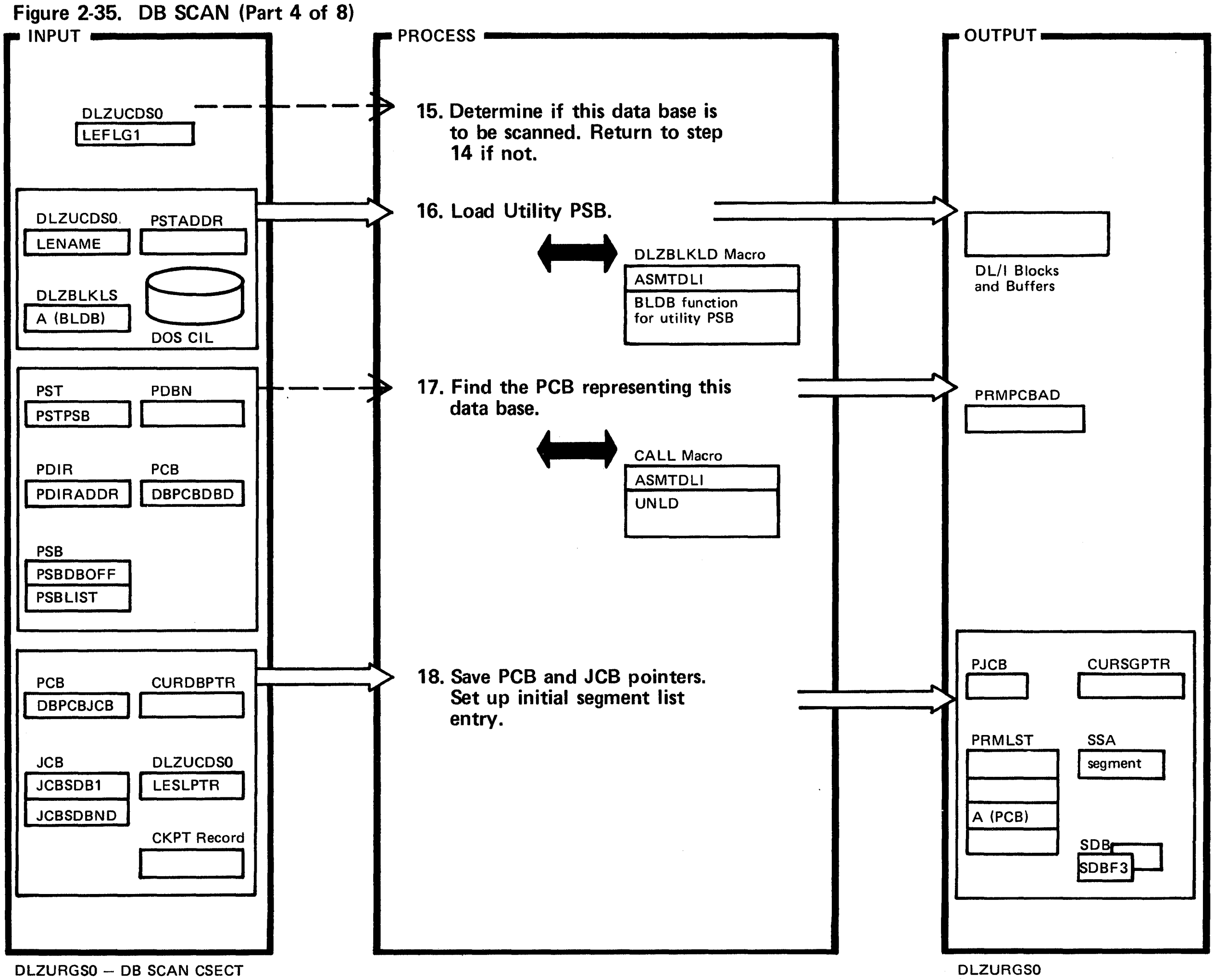
DLZURGS0 – DB SCAN CSECT

DLZURGS0

Extended Description	Routine	Label
10. The SSA built is a qualified key call – 'segname*C (key)' to be used in the call at step 21.		RSTRT50
11. A continuation checkpoint record contains the key value which is moved to the SSA for a qualified key call.		
12.		RSTRT62
13. Write DLZ975I Restart complete.		RSTRT70
14. Write DLZ966I for normal program termination if there are no more DB list entries to process.		NXTDBP NXTDB

Extended Description	Routine	Label

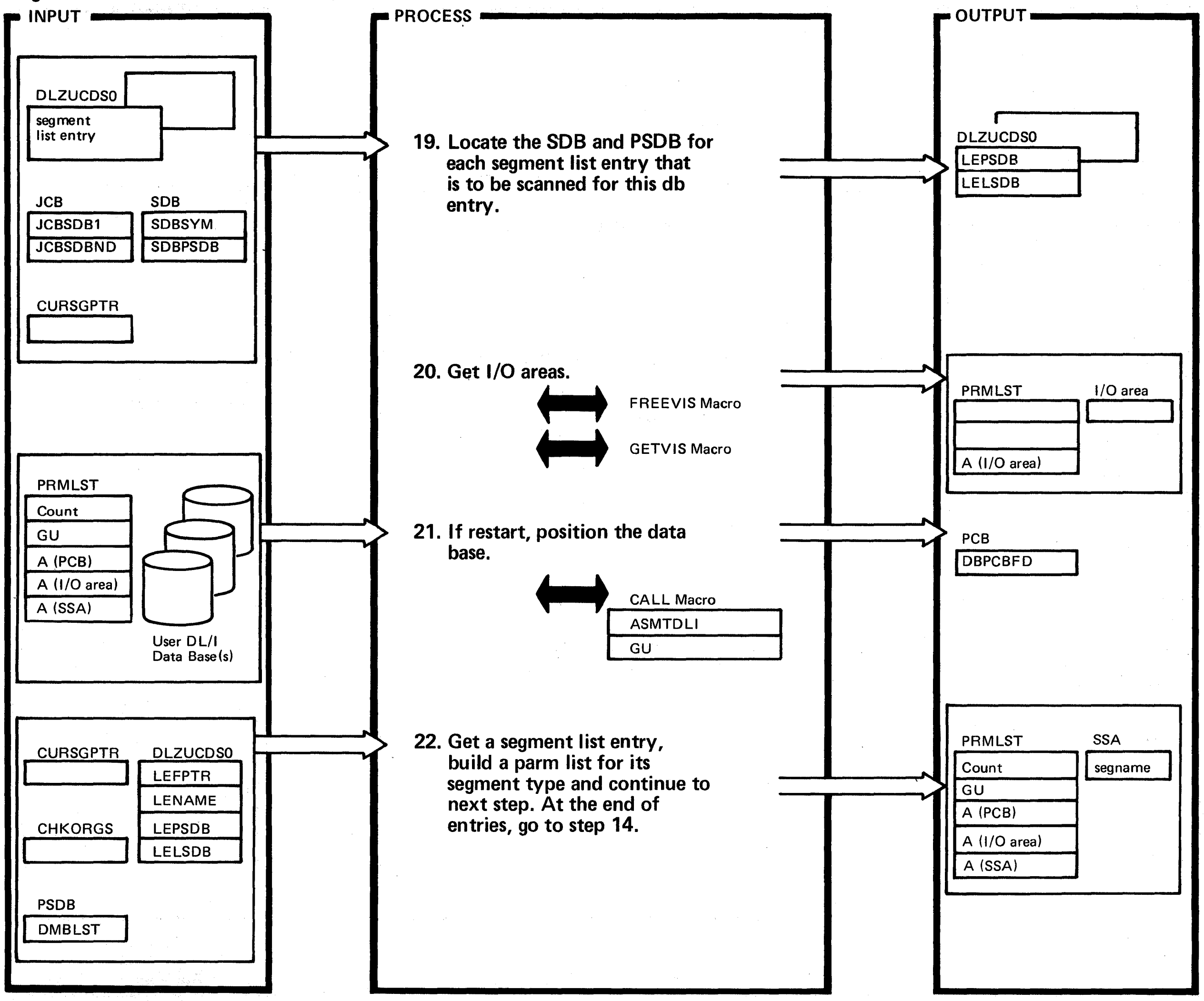
Figure 2-35. DB SCAN (Part 4 of 8)



Extended Description	Routine	Label
15. If DBS control cards were present, only those data bases in the control list that were on DBS cards are scanned.		NXTDBA
Write DLZ970I scan processing started.		
16. The DLZBLKLD macro moves the dmbname to PST at PSTPCPSB and sets the utility suffix '4'. The 'BLDB' call loads all blocks for the PSB specified and allocates buffers.		BLDBLKS
Write DLZ956I for a data base control block build failure.		
17. If a PCB is not found, an UNLD is done to release the buffers before the next BLDB call (return to step 16).		NEWDB

Extended Description	Routine	Label

Figure 2-35. DB SCAN (Part 5 of 8)



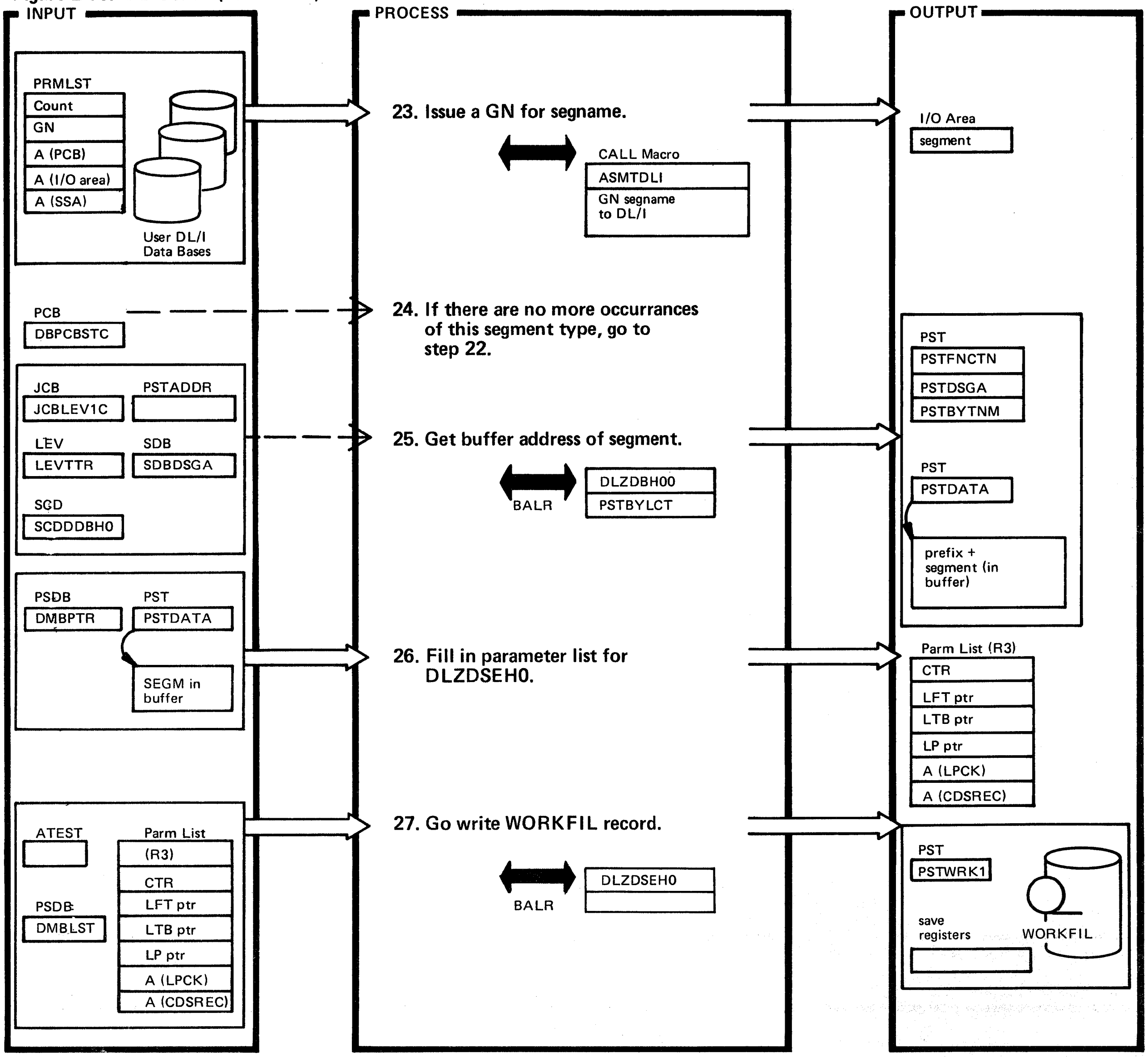
DLZURGS0 — DB SCAN CSECT

DLZURGS0

Extended Description	Routine	Label
19. The Control Data Set entries are modified to save SDB and PSDB addresses.		NXTSEG
Write DLZ969I if an SDB is not found for the segment in the segment list entry.		
20. The size is the longest needed for this data base. Any previous I/O area is freed.		SETUP LENSEG
21. Position is by a qualified key call.		CRST
22. The first segment list entry will either be the initial entry for the db or the segment list entry to restart from.		PROC
Write DLZ971I Scan processing completed for this DB if no more segment entries.		

Extended Description	Routine	Label

Figure 2-35. DB SCAN (Part 6 of 8)



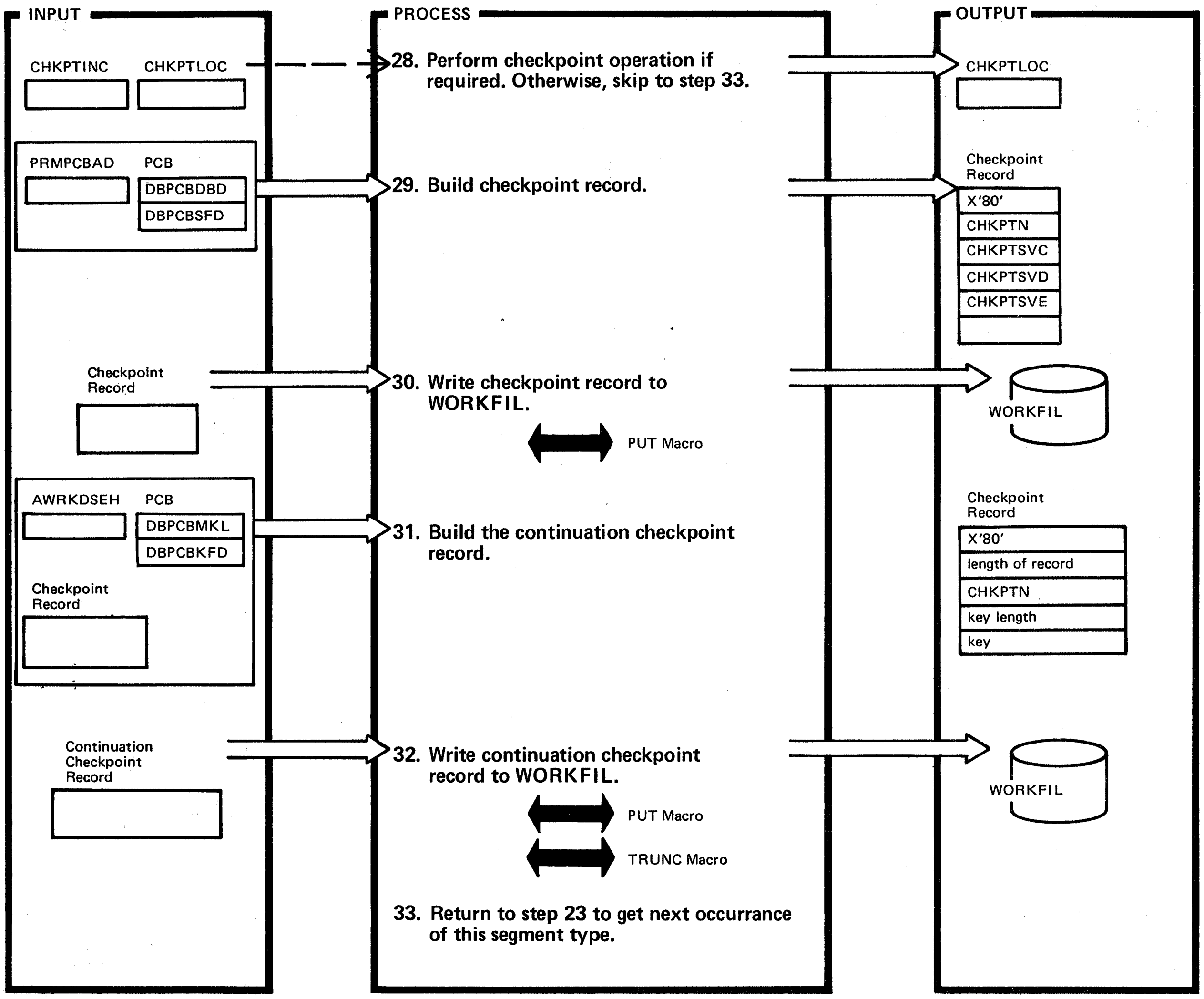
DLZURGS0 – DB SCAN CSECT

DLZURGS0

Extended Description	Routine	Label
23. Every occurrance in the data base of the LC or LP segment type to be scanned is read and a workfile record created for it.		PROC20
24. If the return code is 'GB', indicating EOF for this segment type, return to process next segment list entry.		
25. Scan must have the prefix information to give to DLZDSEH0. Write DLZ958I for a buffer handler error return.		TEST TESTRSTA
26. In addition, R11 has the address of WORKFIL DTF, R1 has the PST address, and PSTWRK1 has the 'FUNCIHPS' and SDB address.		LPOFFR

Extended Description	Routine	Label
26. (con't) Test routine in DLZDSEH0 will determine the output records required by scanning the SEC list.		
27. Registers are saved and then restored upon return. Write DLZ952I for an error return code from DLZDSEH0.		TESTRT

Figure 2-35. DB SCAN (Part 7 of 8)



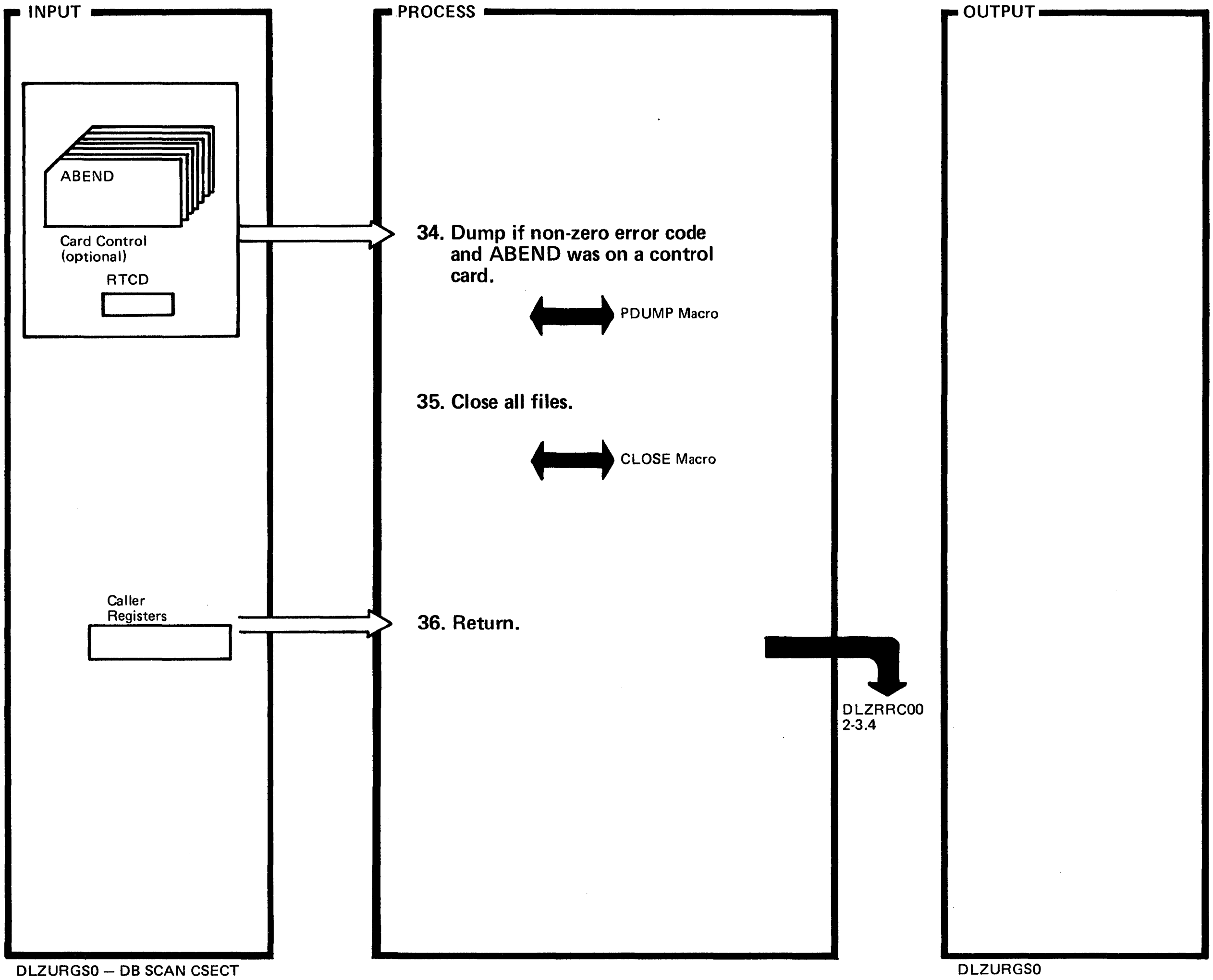
DLZURGS0 – DB SCAN CSECT

DLZURGS0

Extended Description	Routine	Label
28. A checkpoint record is written after every 'n' work file records. 'n' is specified on the CHKPT input card.		CHKPT
29. Note that the checkpoint record is built with zeros where the RBN number would be (CHKPTSUB). This forces an *C call (qualification by concatenated key) instead of *T (retrieve by direct address) during restart processing. Retrieve by direct address would be used if working with HISAM data sets.		CHKPT18
30.		CHKPT19
31.		CHKPT22
32. Message DLZ967I is written to the console giving the current checkpoint record number for later reference.		CHKPT24
33.		TLISTD

Extended Description	Routine	Label

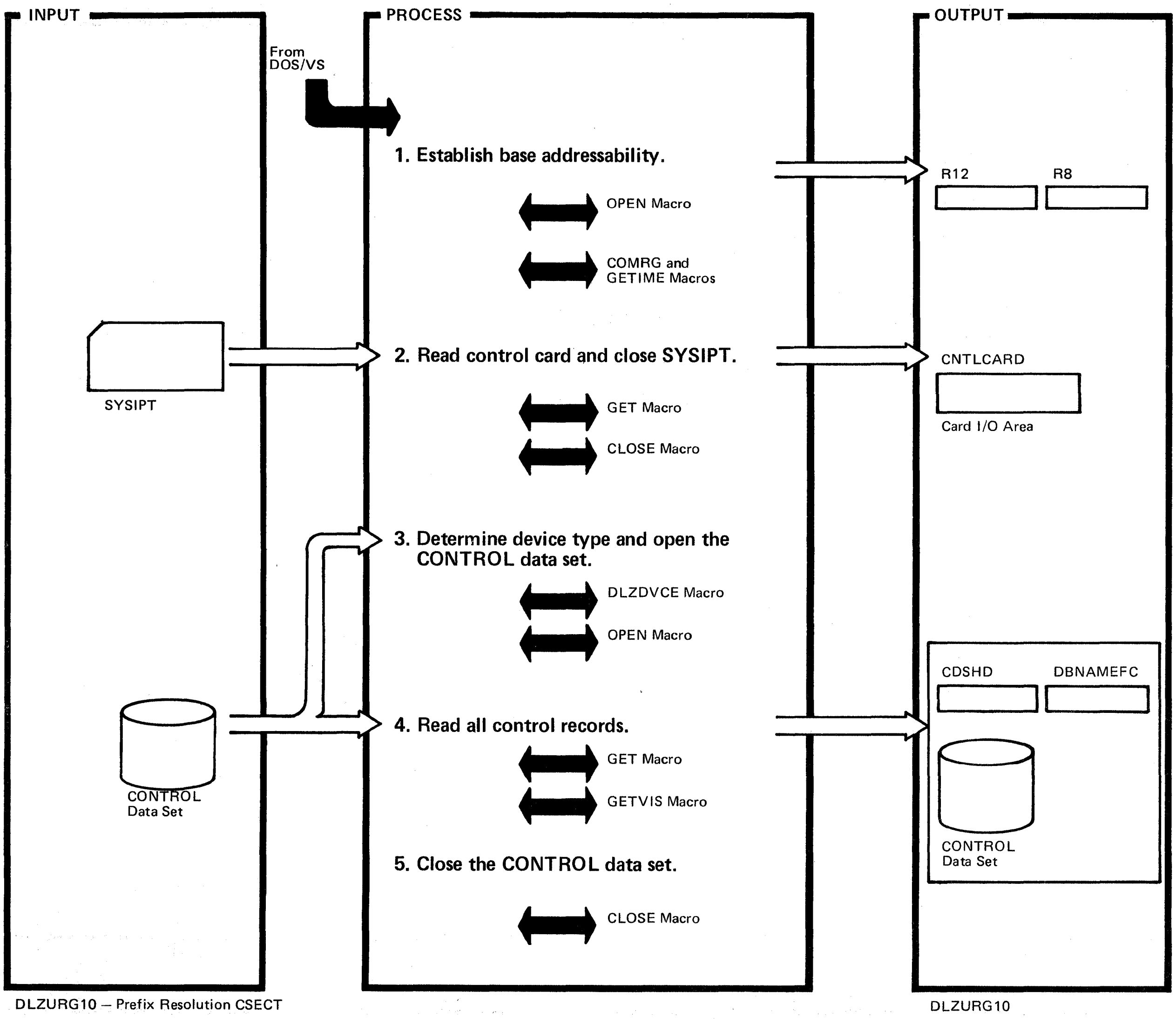
Figure 2-35. DB SCAN (Part 8 of 8)



Extended Description	Routine	Label
34.		TERM
36. Indicate to batch initialization that UNLD is not necessary (R15 = 4) if this utility is terminating before any blocks were loaded or if there was a control block build failure.		Return

Extended Description	Routine	Label

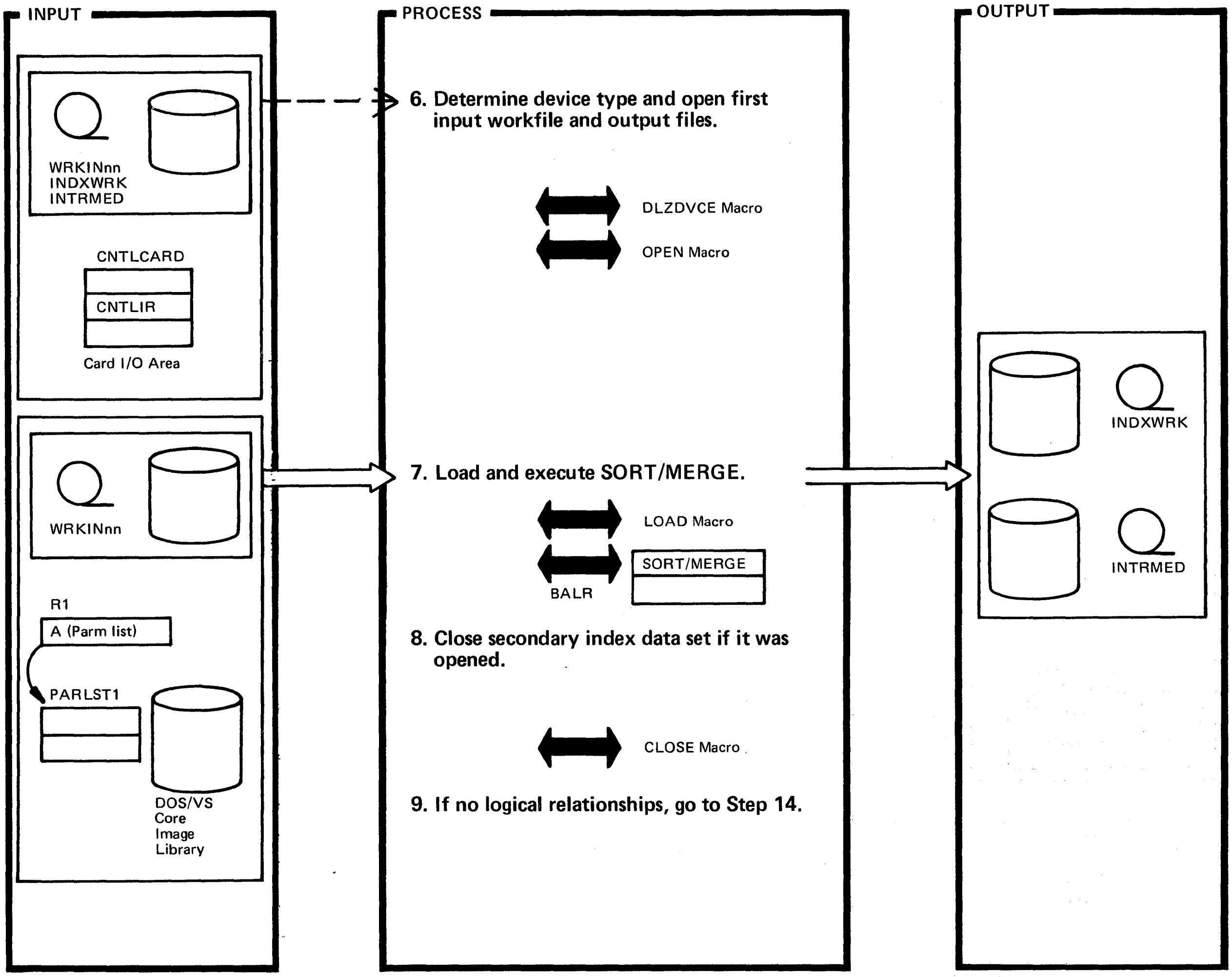
Figure 2-36. Prefix Resolution (Part 1 of 4)



Extended Description	Routine	Label
1. Module identifier (DLZURG10vrnp) is defined here. The time and date are acquired and message DLZ968I is printed at this time to indicate the beginning of execution for DLZURG10.	DLZURG10	DLZURG10
2. Write message DLZ954I for an input control card format error.		
3. Write message DLZ984I for an invalid device assignment for the file.		CDSIN OPENCTL
4. Write message DLZ957I if there is no control data set or if the ID is not "CONTROL DATA SET". Write message DLZ391I for a GETVIS failure. The maximum record length calculated by the prereorganization utility is obtained from field LESRTSZE and passed to SORT.		

Extended Description	Routine	Label
5.		CDSEOF CDSEOFA

Figure 2-36. Prefix Resolution (Part 2 of 4)



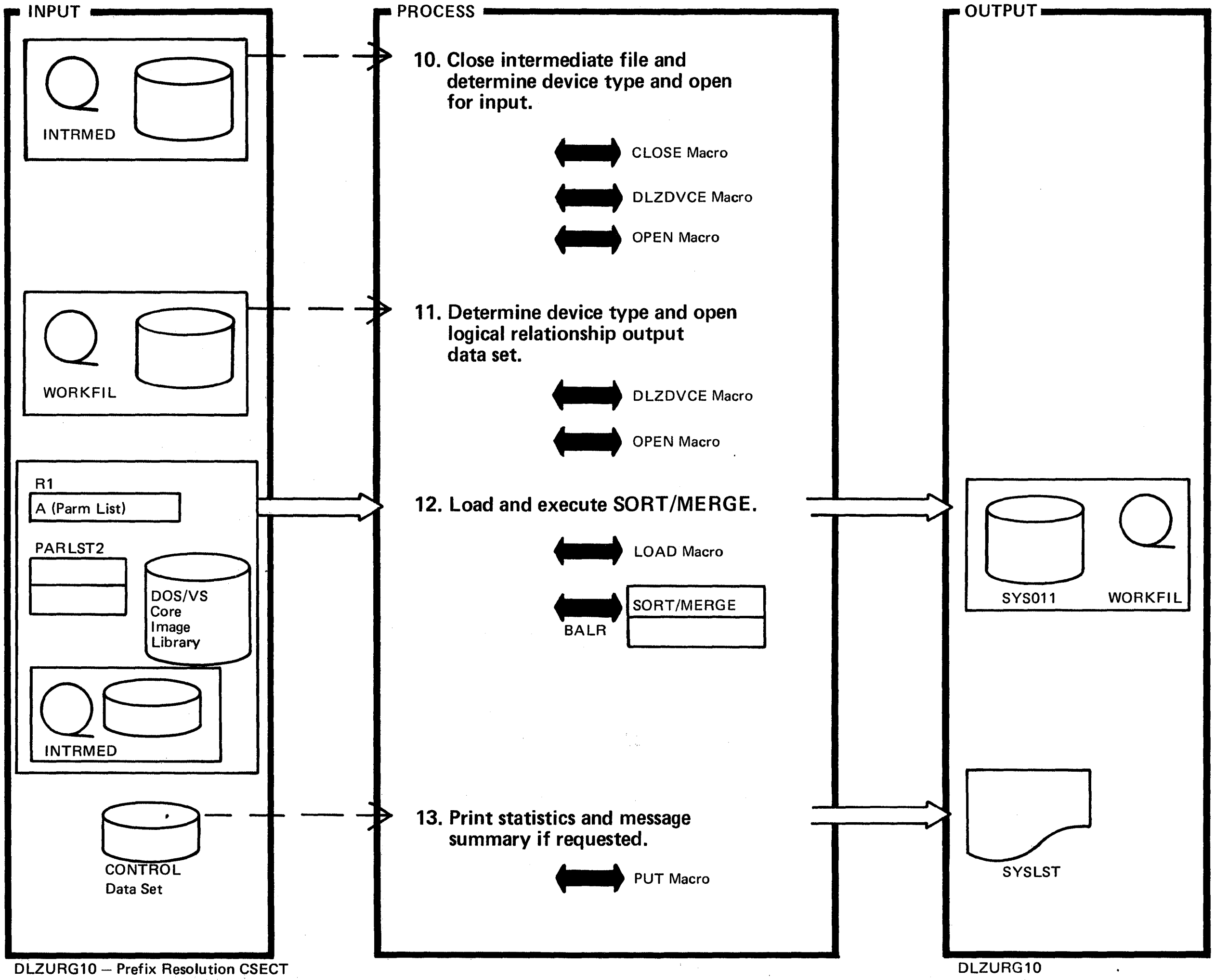
DLZURG10 – Prefix Resolution CSECT

DLZURG10

Extended Description	Routine	Label
6. The secondary or logical data sets may or may not be opened depending on the user option on the input control card.		OPENRT1 OPIND OPENLR
7. Write message DLZ982I if the return code from SORT is not zero and go to Step 15. Sort is by (13, 255, A, 5, 1, A). Exits E15 and E35 are described in Figures 2-36.1 and 2-36.2.		SORT1
8. If there was no data put to the secondary index data set, put a dummy record before closing.		SORT11B
9.		SORT11F

Extended Description	Routine	Label

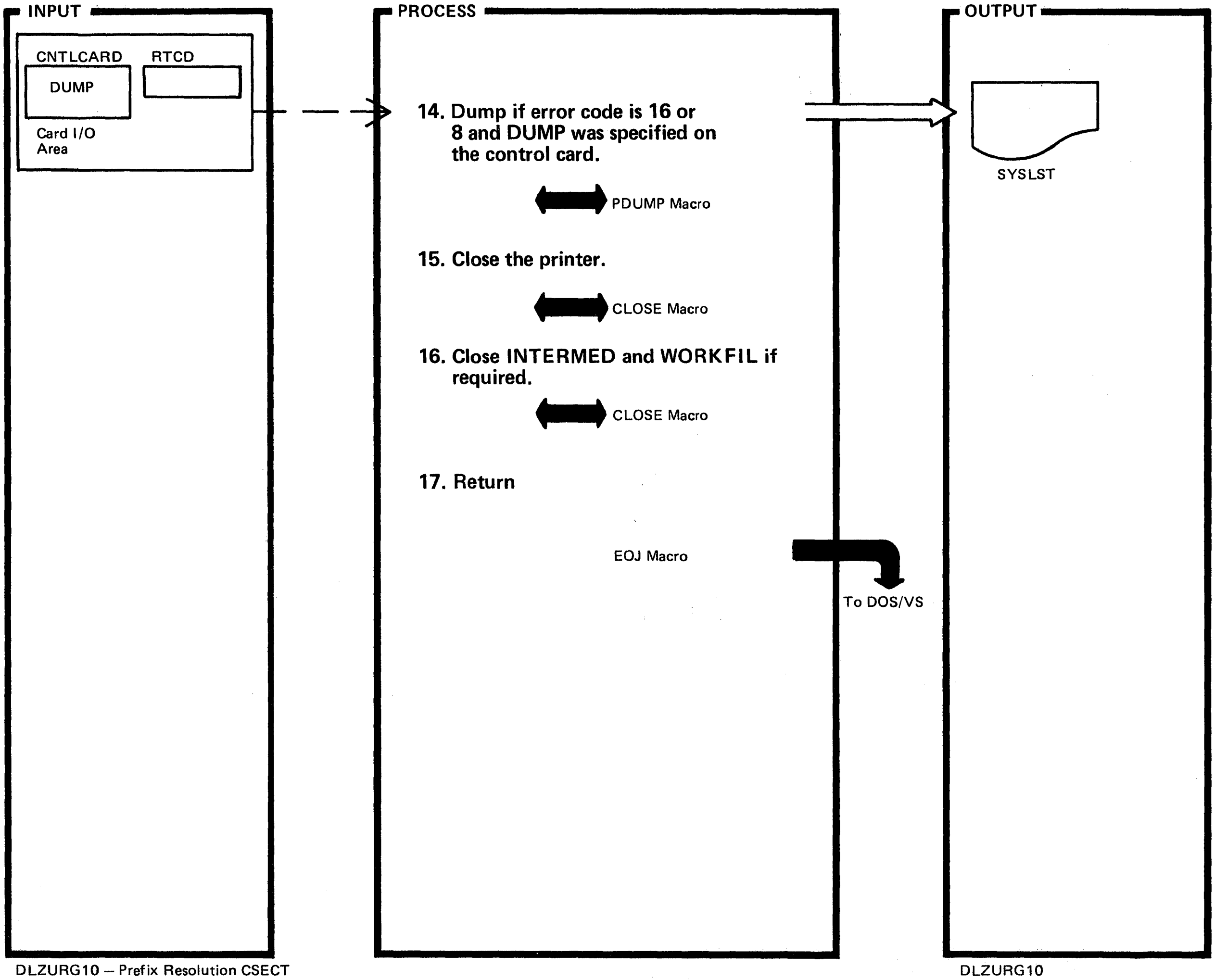
Figure 2-36. Prefix Resolution (Part 3 of 4)



Extended Description	Routine	Label
10.		SORT11D OPENRT2
11.		OPENWLR
12. Sort is by (29, 16, A 5, 1, A). Exits E15 and E35 are described in Figures 2-36.3 and 2-36.4.		
Write message DLZ982I if the return code from SORT is not zero and go to Step 15.		SORT2
13. Control data set contains options as specified in DLZURPR0.		SUMM STATFLG

Extended Description	Routine	Label

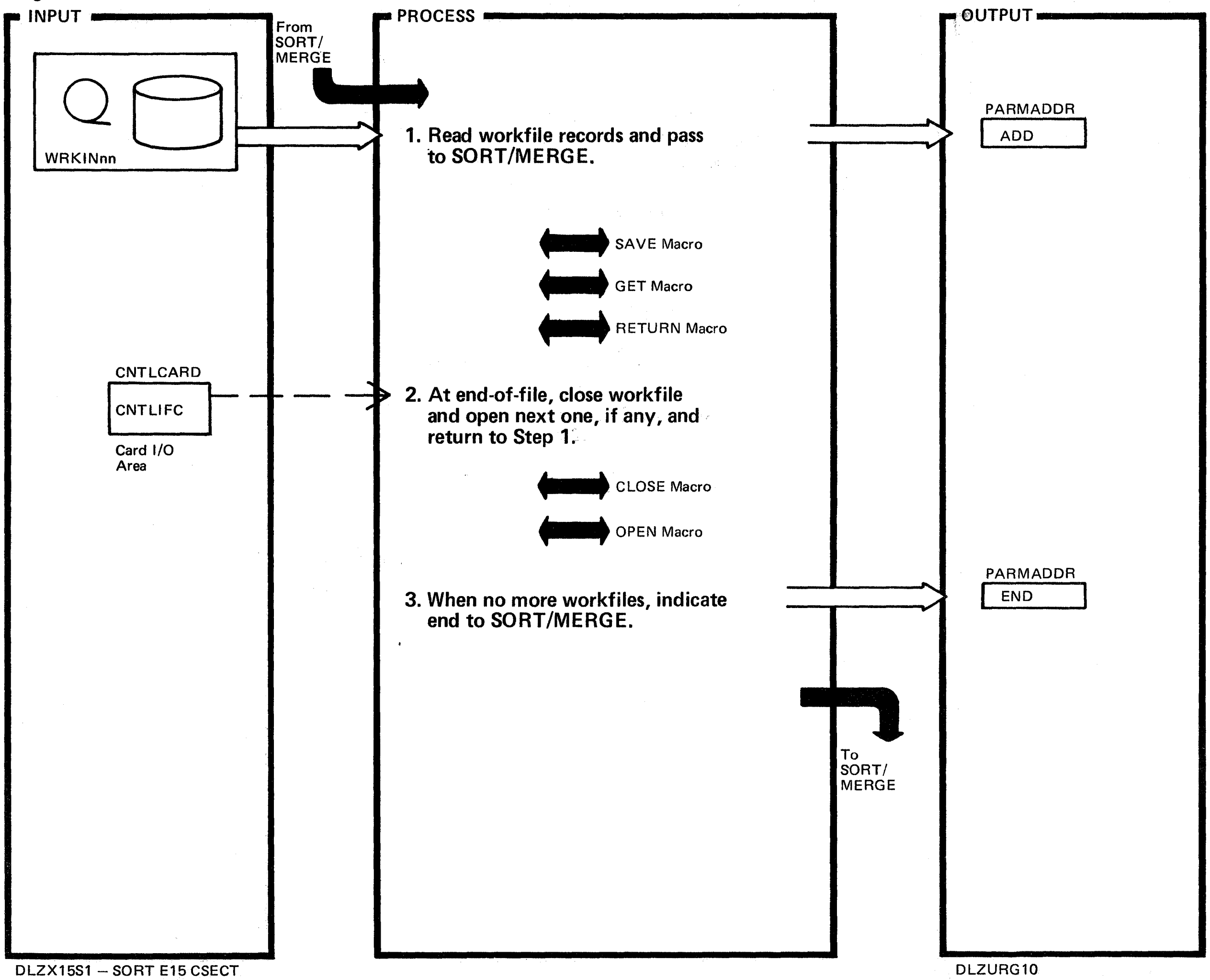
Figure 2-36. Prefix Resolution (Part 4 of 4)



Extended Description	Routine	Label
14. Write message DLZ966I for normal program termination.		STATENO CLOSRT2A
15.		CLOSRT2B
17.		CLOSRT2D

Extended Description	Routine	Label

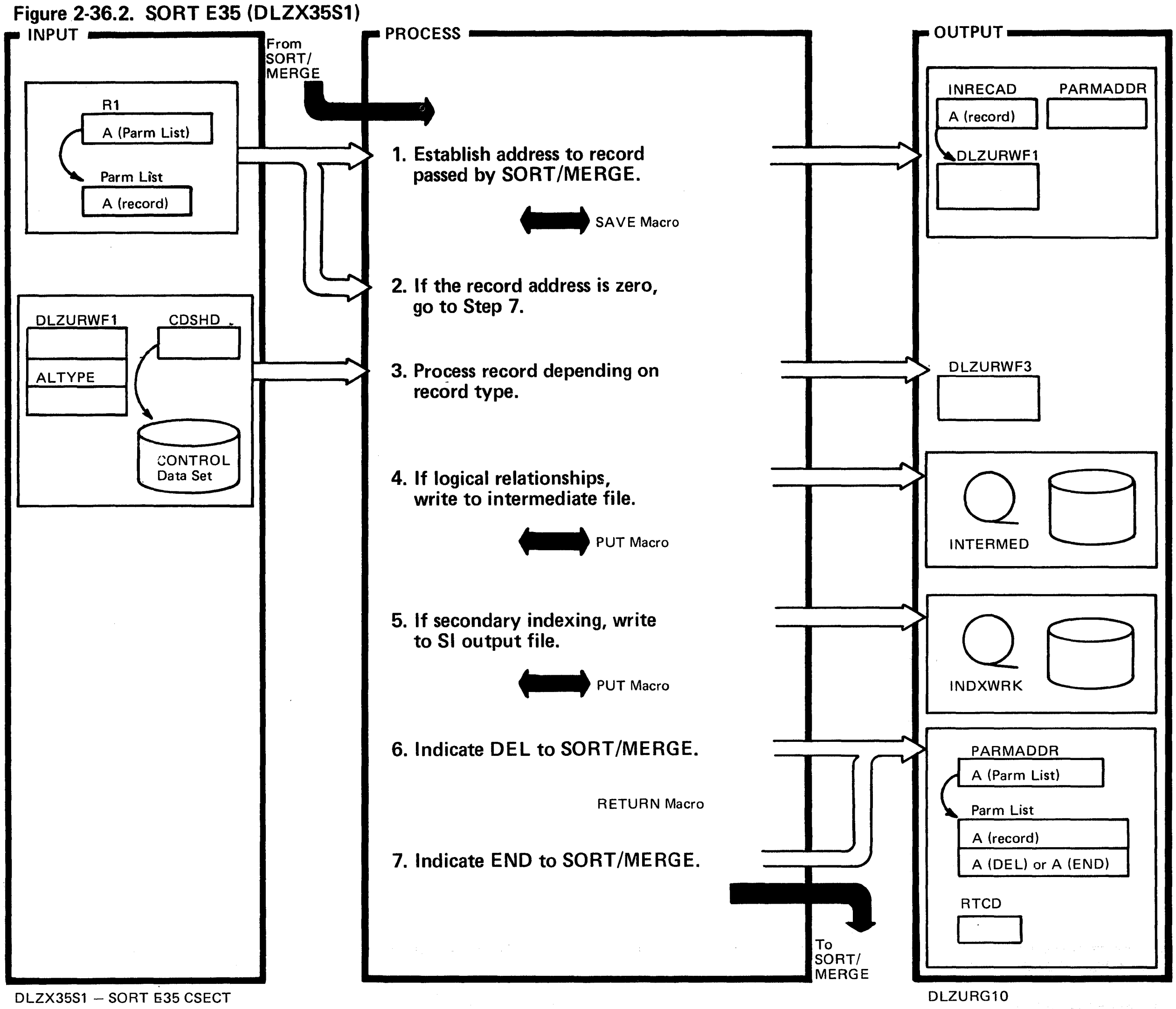
Figure 2-36.1. SORT E15 (DLZX15S1)



Extended Description	Routine	Label
1. Record length is changed to the maximum record length calculated by the prereorganization utility (DLZURPR0) and passed to SORT by the prefix resolution utility (DLZURG10). Original record length is saved in last 2 bytes of LRECL field.	DLZX15S1	DLZX15S1 EXIT15S1
Indicate ADD to SORT/MERGE after each GET.		
2. CNTLIFC is the number of input workfiles specified on the utility control card.		WRKEOFI NXTFILE

Extended Description	Routine	Label

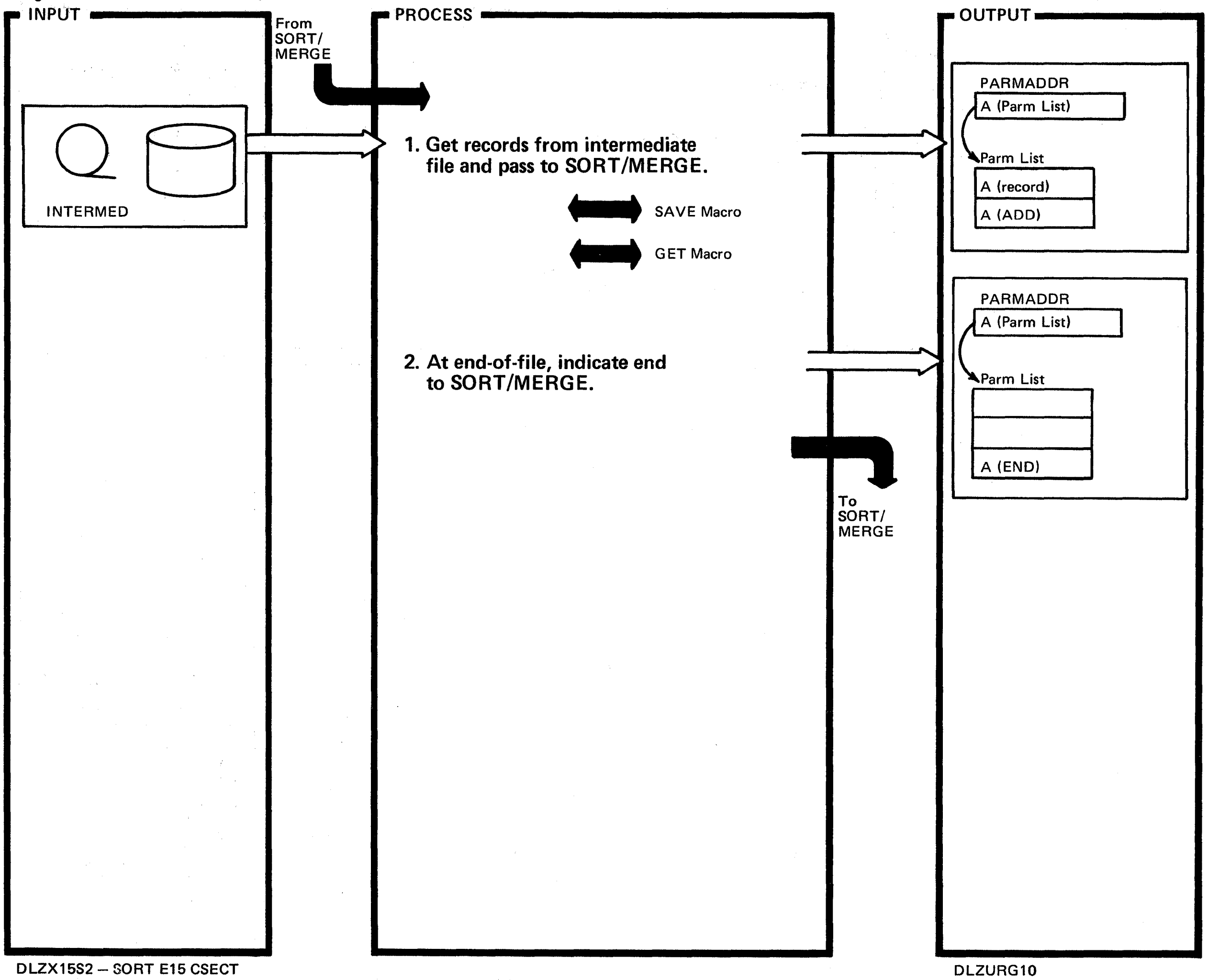
Figure 2-36.2. SORT E35 (DLZX35S1)



Extended Description	Routine	Label
1. SORT/MERGE passes one record at a time to this exit. The record is represented by the macro DLZURWF1 which contains the DSECT defining the format. The original record length is restored before processing.	DLZX35S1	DLZX35S1 EXIT35S1
3. Macro DLZURWF3 contains the DSECT defining the format of the output logical record and later used as input for DLZURGP0. Possible errors are: DLZ955I – Invalid input record. DLZ977I – Duplicate record for LP. DLZ978I – Caution – no LC for LP. DLZ979I – No LP found for LC. DLZ980I – No LC found for LT. DLZ989I – Multiple LC/LP with no LT pointer specified.		ESTTYPE
4. This file used as input for second SORT/MERGE.		STATRIZ OUTPRV1A
5. This is final output for secondary index relationships.		TYPE04RT

Extended Description	Routine	Label
6. SORT/MERGE gets another record and reenters this exit at Step 1.		RETSORT1
7.		ENDSORT1

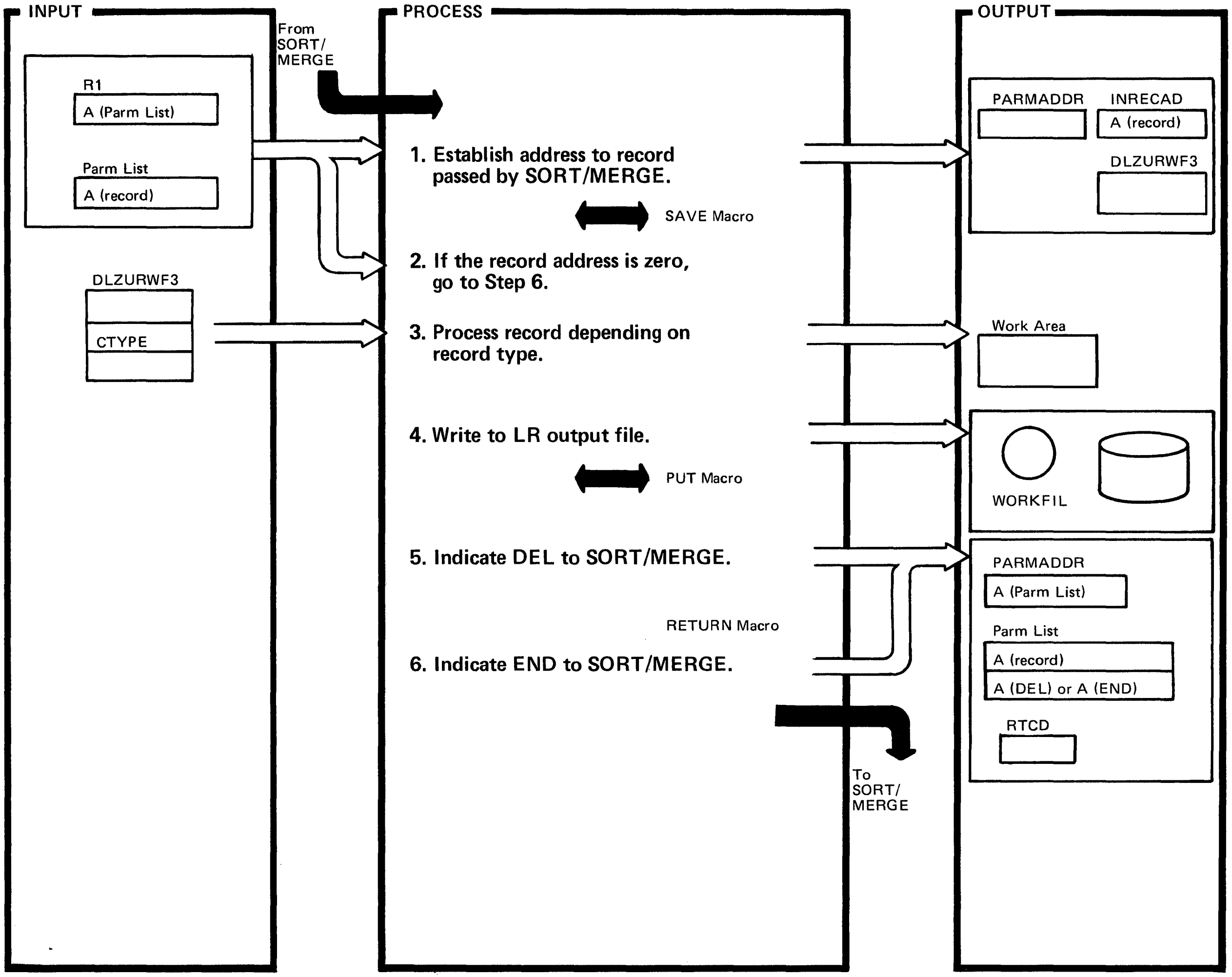
Figure 2-36.3. SORT E15 (DLZX15S2)



Extended Description	Routine	Label
1. This file was written during first sort.	DLZX15S2	DLZX15S2 EXIT15S2
2.		MEDEOFI

Extended Description	Routine	Label

Figure 2-36.4. SORT E35 (DLZX35S2)



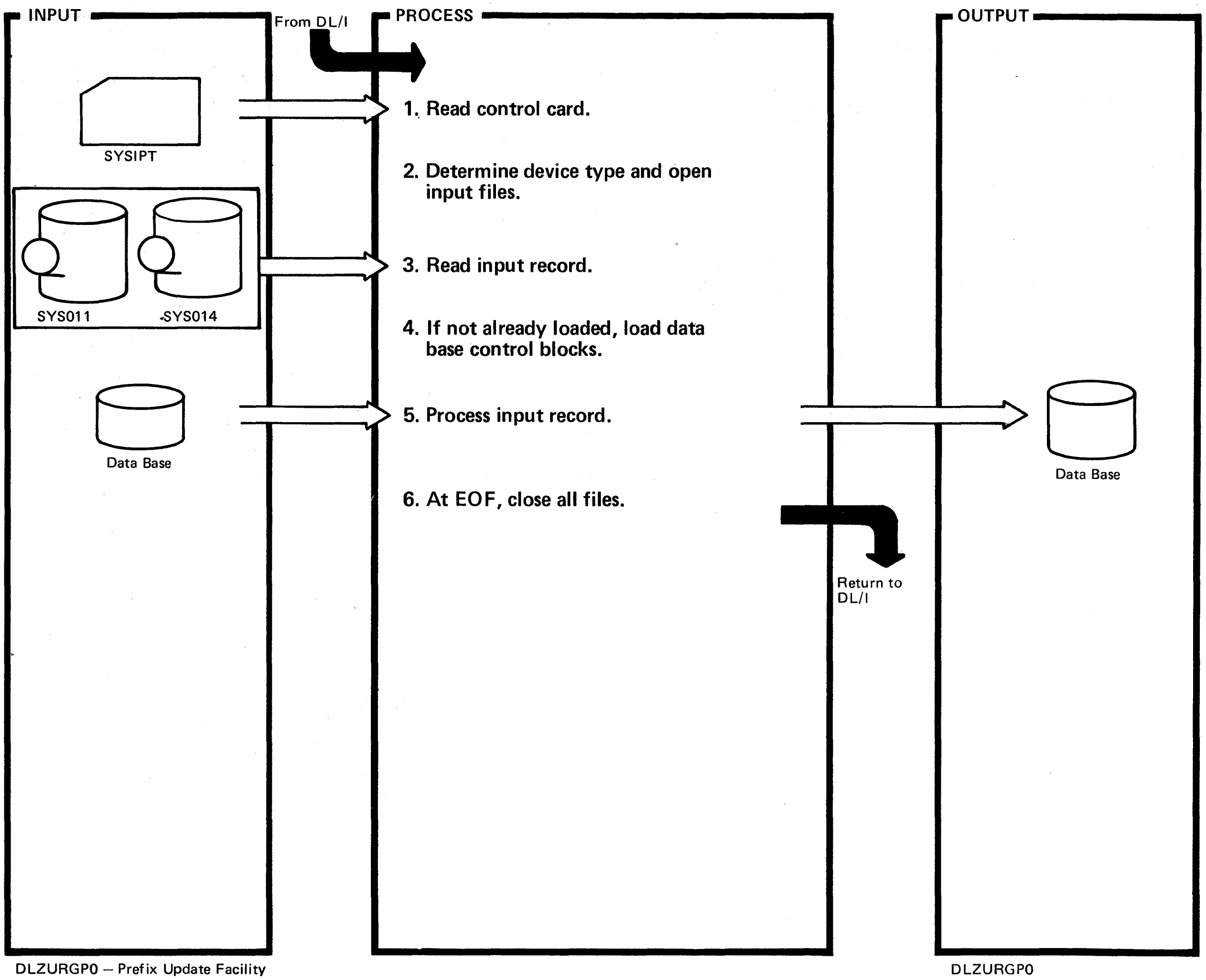
DLZX35S2 – SORT E35 CSECT

DLZURG10

Extended Description	Routine	Label
1. SORT/MERGE passes one record at a time to this exit.	DLZX35S2	DLZX35S2 EXIT35S2
3. Possible errors are: DLZ955I – Invalid input record. DLZ980I – No LC found for LT. DLZ981I – Duplicate record for LT.		TP30RT
4. This file used as input for the prefix update utility (DLZURGP0).		
5. SORT/MERGE gets another record and reenters this exit at Step 1.		RETSORT2
6.		ENDSORT2

Extended Description	Routine	Label

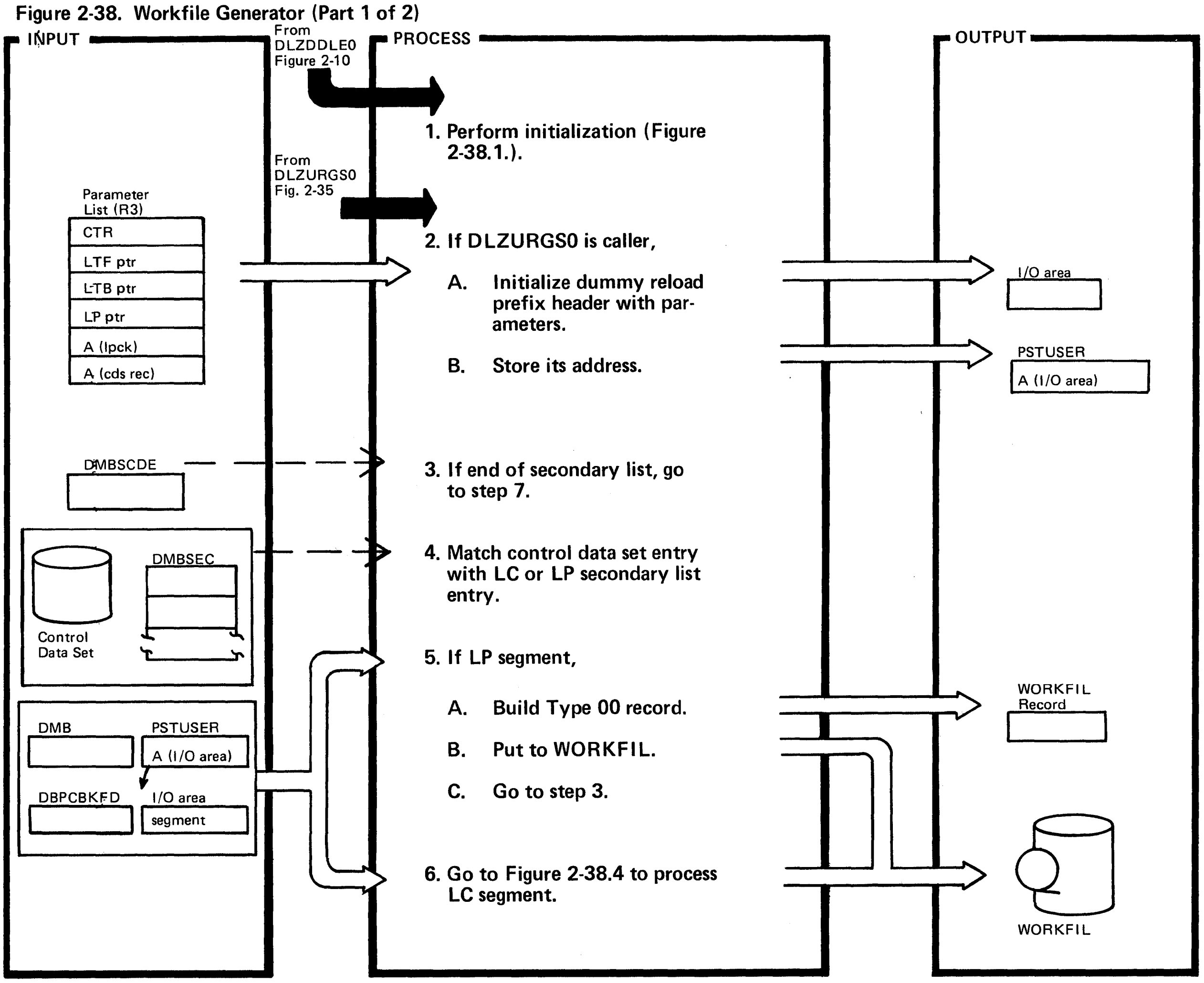
Figure 2-37. Prefix Update Utility



Extended Description	Routine	Label
1.		OPEN1
2. DLZDVCE macro obtains data from PUB. Device type may be TAPE or DASD.		OPENINP
4. DLZBLKLD macro is used to load DB blocks dynamically.		BLDBLKS
5. TYPE 0 and TYPE 1 records (LC/LP) are processed by buffer handler calls. TYPE 4 records (SI) are processed by DL/I INSERT/UPDATE calls.		TYPE0 TYPE1

Extended Description	Routine	Label

Figure 2-38. Workfile Generator (Part 1 of 2)



DLZDSEH0 — Workfile Generator

DLZDSEH0

Extended Description	Routine	Label
1. This primary entry point is used by Load/Insert when a data base is being initially loaded or reloaded. There are 7 fullwords of addresses immediately preceding this entry point used by modules that interface with DLZDSEH0. A logical parent or logical child record is input to this module.	DLZDSEH0	INIT
2. This is the primary entry point for the scan utility.		TEST
3. This routine must be re-entered when the input segment is an LP because it could have more than one LC type.		TLISTEND
4.		TESTC
5. Description of WORKFIL record can be found in DLZURWF1 dsect.	DLZDSEH0	LP1
6.		CHILD

Extended Description	Routine	Label

Figure 2-38. Workfile Generator (Part 2 of 2)

INPUT

PROCESS

7. Exit with return code.

OUTPUT

R15

Return to Caller

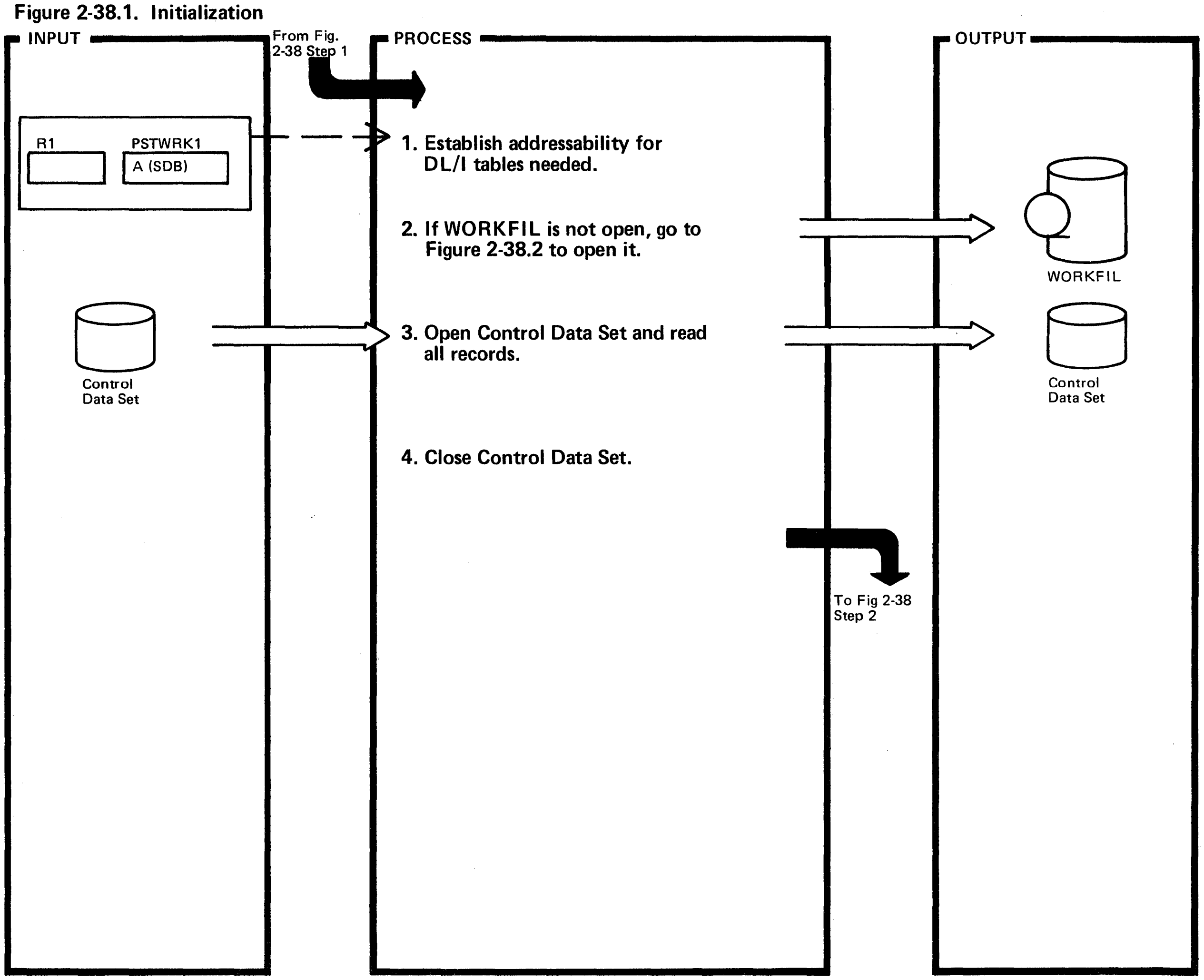
DLZDSEH0 – Workfile Generator

DLZDSEH0

Extended Description	Routine	Label
7. If any error occured, call DL/I error message module to write DLZ007I message on console with return code.		RETURN

Extended Description	Routine	Label

Figure 2-38.1. Initialization



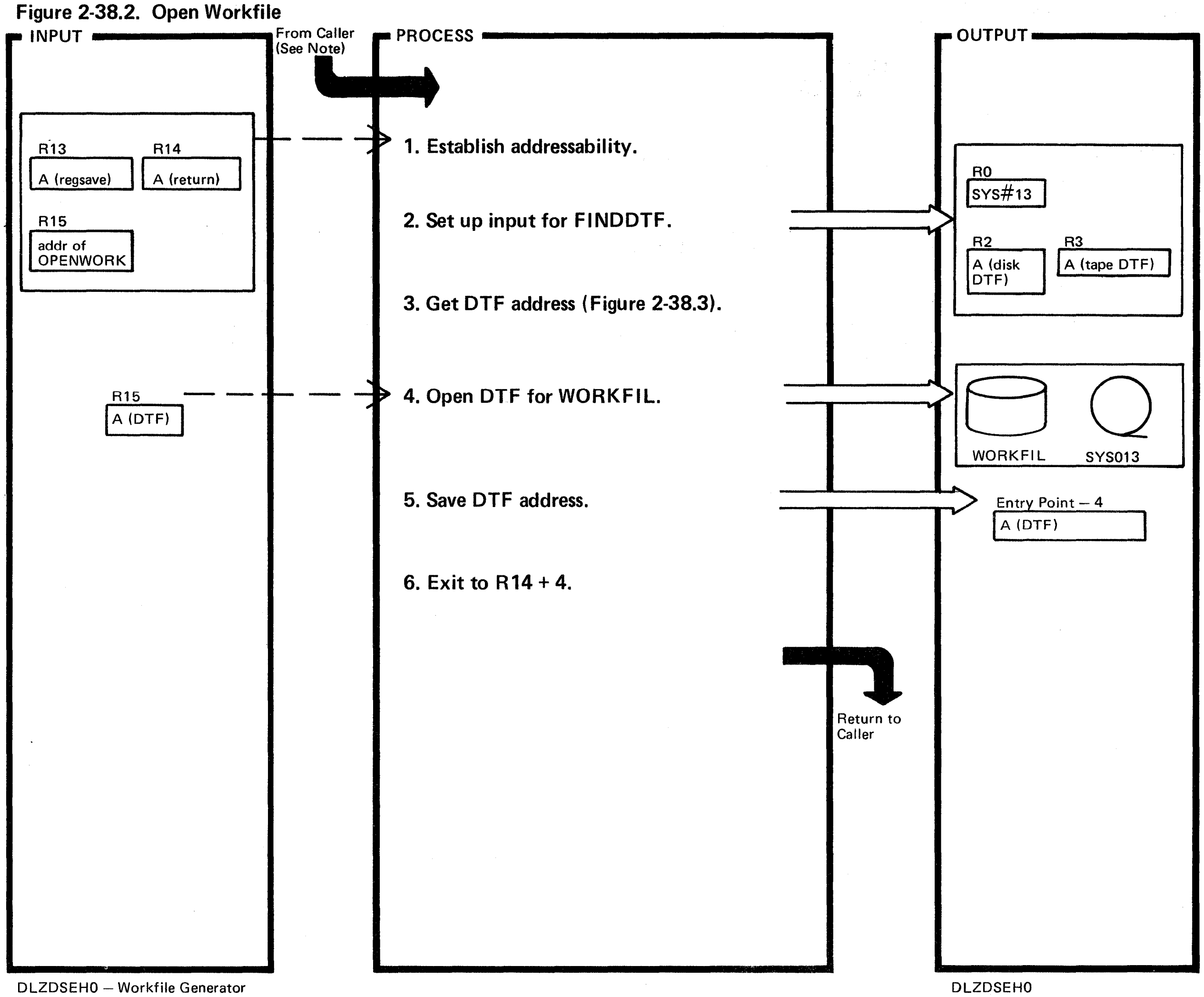
DLZDSEHO — Workfile Generator

DLZDSEHO

Extended Description	Routine	Label
1. The secondary list entries for the input segment are the primary source of information from the DL/I blocks.	DLZDSEHO	INIT
2. The address of the DTF is found in the address list at the beginning of DLZDSEHO. If it is 0, this workfile must be opened.		
3. This open is done only once. The 'FINDDTF' routine is used to determine the correct DTF. If more than one record exists on the CDS, a GETVIS is done to hold the entire file in storage at one time.		LPLCA

Extended Description	Routine	Label

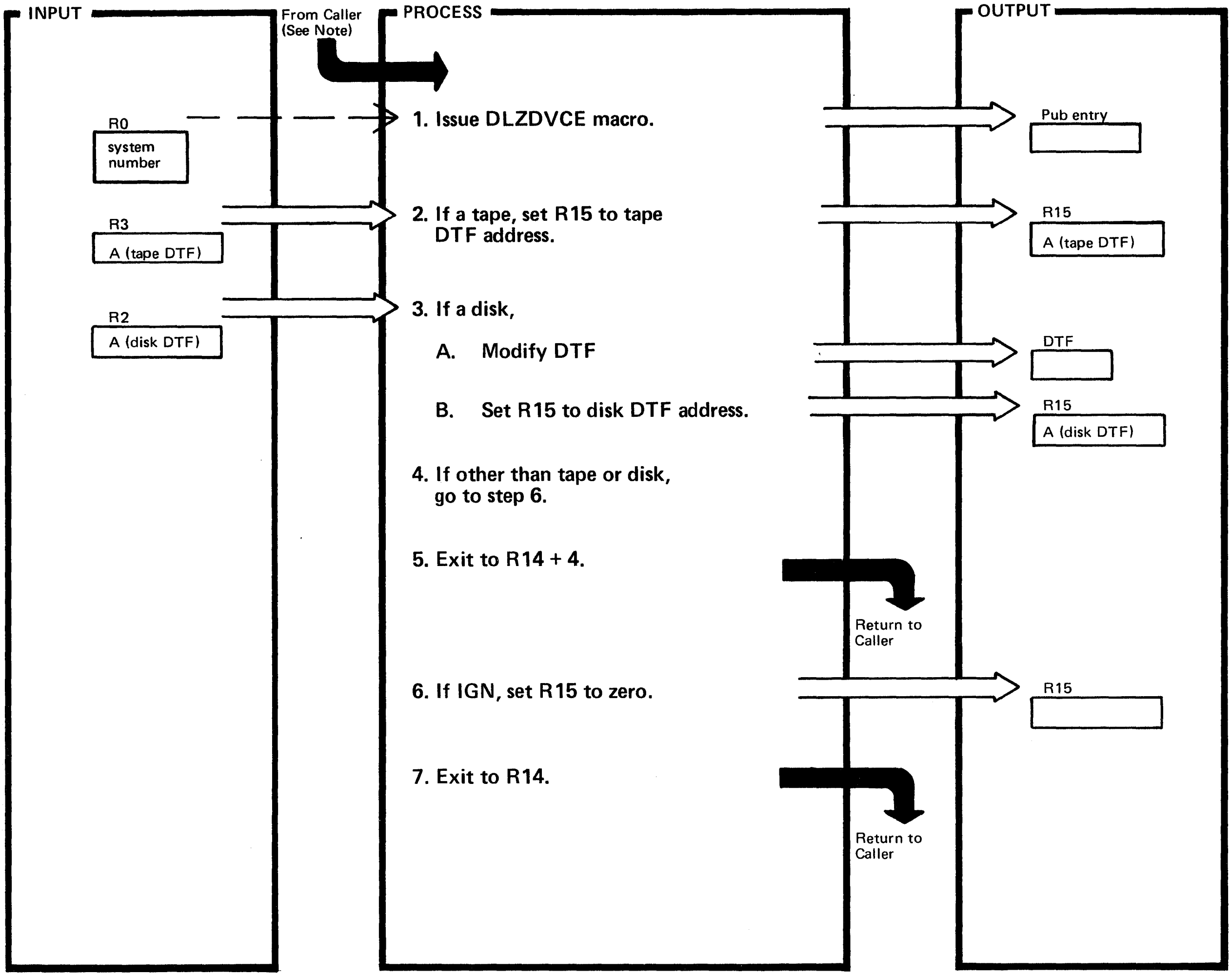
Figure 2-38.2. Open Workfile



Extended Description	Routine	Label
Note: This routine is called by DLZDSEH0, DLZDXMT0, and DLZURGS0.		
1.	OPENWORK	OPENWORK
3. If control is returned to address in R14, an error occurred. R14 + 4 is the normal return.		
4. R15 has address of correct DTF as returned by FINDDTF.		
5. When the WORKFIL is open, the address is saved in the address list in the beginning of DLZDSEH0 csect.		
6. If an error was detected, control is returned to the address in R14. Normal return is R14 + 4.		OPENEXIT

Extended Description	Routine	Label

Figure 2-38.3. Find DTF



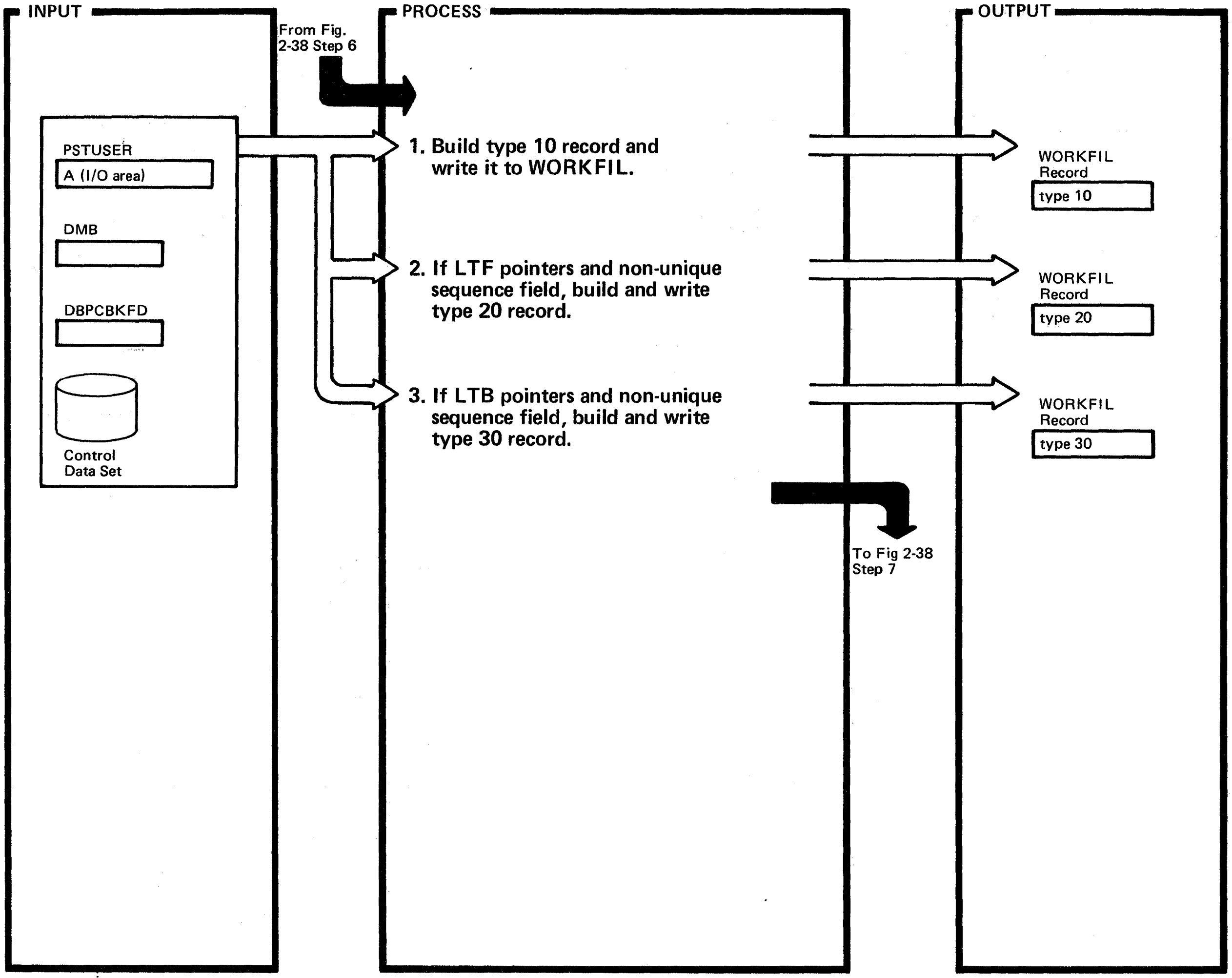
DLZDSEHO – Workfile Generator

DLZDSEHO

Extended Description	Routine	Label
Note: This subroutine is called by OPENWORK, DLZDSEH0, and DLZURGS0. DLZDVCE macro finds PUB entry for given programmer logical unit and the device type byte is used to determine further processing.		
1.	OPENWORK	FINDDTF
2. 2400, 3410, and 3420 are supported.		FINDTF0
3. 2314, 3330, 3333, 3340A & B are supported.		FINDTF1 FINDTF2
5. Normal return.		FINDEXIT
6. This allows DLZDXMT0 to build secondary entries.		FINDERRX
7. This is the error exit.		FINDERRU

Extended Description	Routine	Label

Figure 2-38.4. Build LC Output



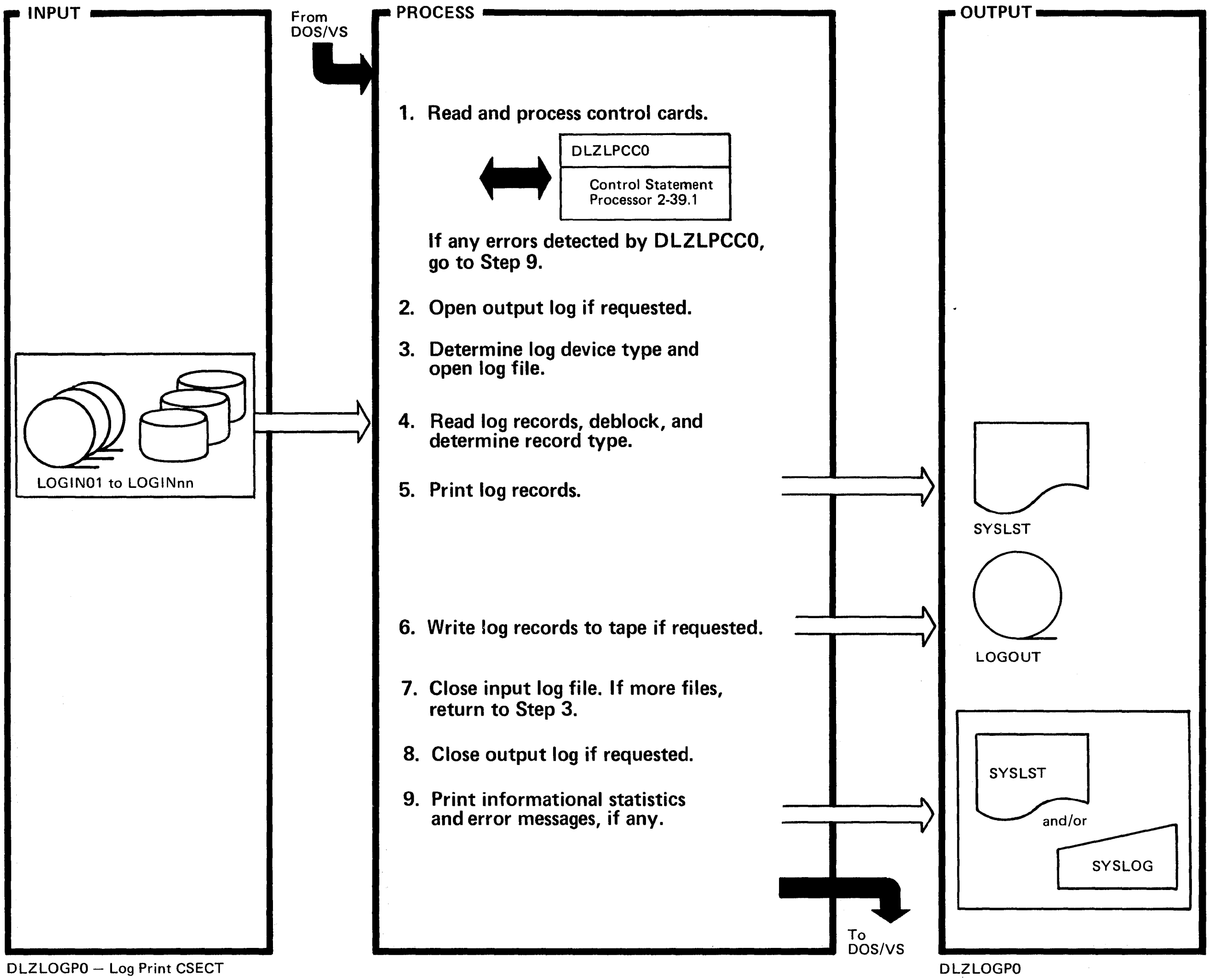
DLZDSEH0 — Workfile Generator

DLZDSEH0

Extended Description	Routine	Label
1.		CHILD LC1
2.		LC120A
3.		LC130B

Extended Description	Routine	Label

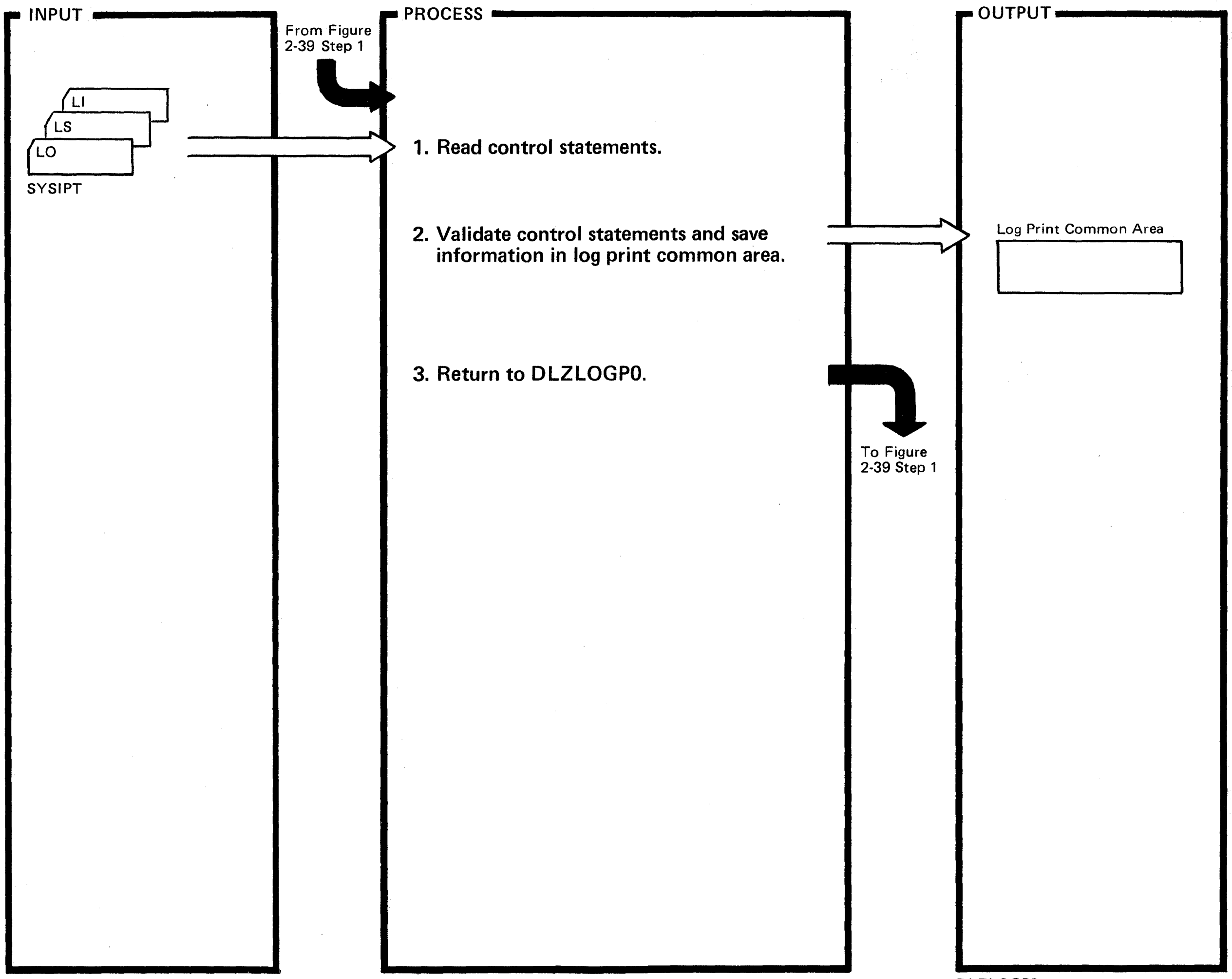
Figure 2-39. Log Print Utility



Extended Description	Routine	Label
1. Register 1 points to log print common area.	DLZLOGP0	GETCARD
2. Output log requested by 'COPY' on 'LO' statement.		CARDEOF
3. DLZDVCE macro obtains data from PUB (physical unit block) and modifies DTF. If VSAM log, ACB is modified manually.		LOGOPEN
4. Valid DL/I record types are: o Data base record (X'50' and X'51') o Open record (X'2F') o Scheduling record (X'08') o Termination record (X'07') o Checkpoint record (X'41')		GETLOG
5. Records are printed in either keyword or dump format.		PRINT
6. Log records are written to tape as read.		GETREC
7. nn of LOGINnn is incremented by 1 if more files.		LOGEOF
8. Output log is closed when log record in error is encountered.		

Extended Description	Routine	Label

Figure 2-39.1. Control Statement Processor (DLZLPCC0)



DLZLOGP0 — Log Print CSECT

DLZLOGP0

Extended Description	Routine	Label
1. Possible card types are: 'LO' – describes print options, 'LS' – describes additional selective print options, 'LI' – describes input log files.	DLZLPCC0	GETCARD
2. Flag ERROROCC in LOGPFLG1 is set if any errors are detected.		LO000 LI000 LS000
3. If no input statements received, print default message DLZ416I.		CARDEOF

Extended Description	Routine	Label

Figure 2-40. Field Level Sensitivity Copy

INPUT

PSTUSER

A (User Segment)

PSBIOAWK

A (Output I/O Area)

PSDB

From Figure 2-8.1 Step 8A

PROCESS

1. Get addresses of user segment(s) and output I/O area.
2. Calculate segment offset in output I/O area for this level.
3. If segment at this level is not FLS, move as is from user I/O area to output I/O area.
4. If segment at this level is FLS and call is ISRT, perform insert processing.

   ISRT — Perform insert processing 2-40.1

5. If segment at this level is FLS and call is REPL, perform replace processing.

   REPL — Perform replace processing 2-40.2

6. If another level to process, go to step 2. Otherwise, point PSTUSER to output I/O area and return.

Return to Caller

OUTPUT

LEVUSEOF

PSTUSER

A (Output I/O Area

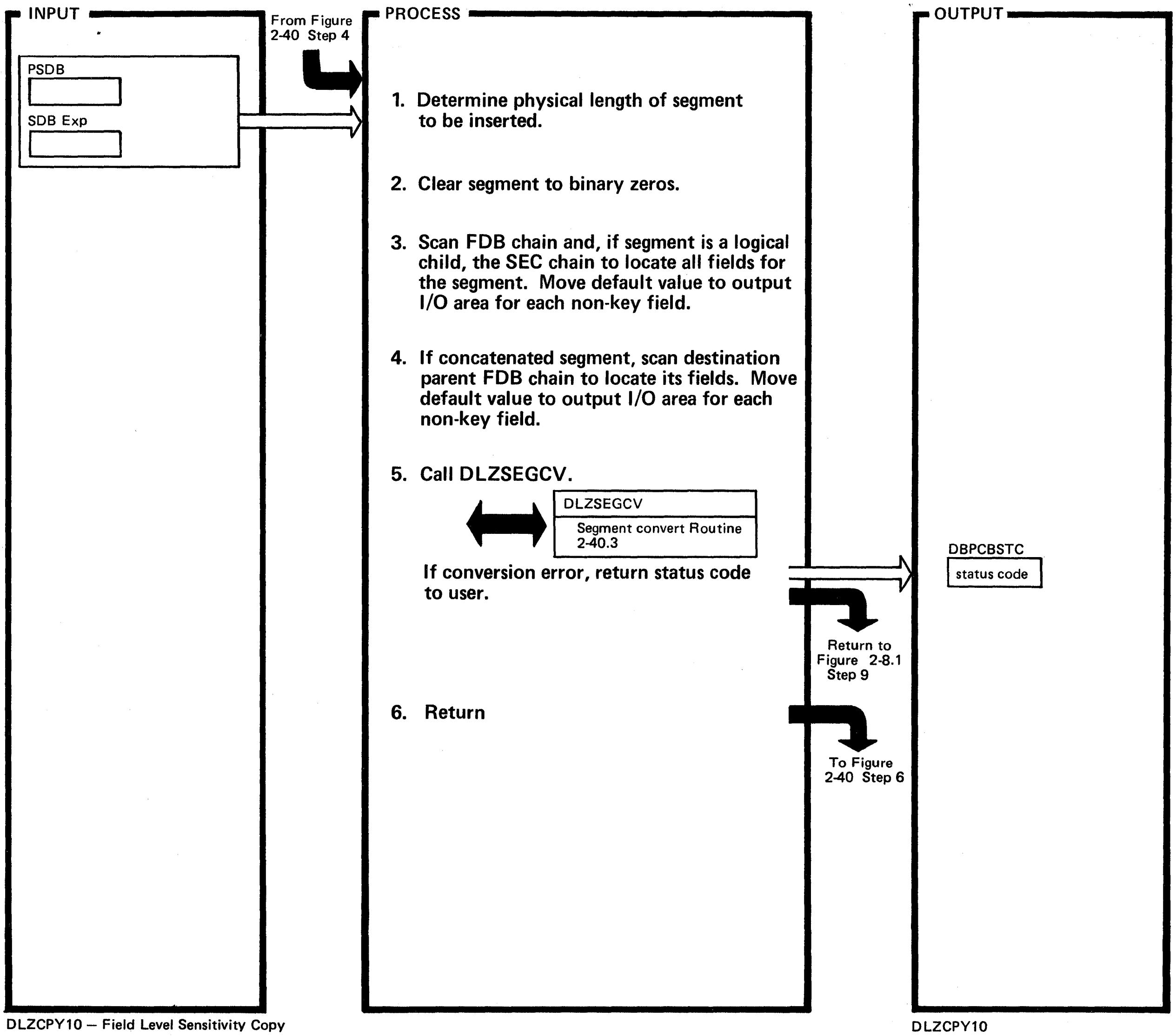
DLZCPY10 – Field Level Sensitivity Copy

DLZCPY10

Extended Description	Routine	Label
1. PSTUSER points to user's view of segment(s). PSBIOAWK points to I/O area to contain physical view of segment(s).	DLZCPY10	DLZCPY10
2. LEVUSEOF will be changed from offset in user I/O area to offset in output I/O area.		LEVLOOP
3. Length of segment to be moved must be determined. If variable length segment, length is in first two bytes of segment, otherwise in PSDB. If concatenated segment, logical child and destination parent lengths must be added.		LEV000
6.		NEXTLEV

Extended Description	Routine	Label

Figure 2-40.1. Field Level Sensitivity Insert



Extended Description	Routine	Label
1. If variable length segment, use insert length in SDB expansion block. Otherwise use length in PSDB. If concatenated segment, logical child and destination parent lengths must be added.	DLZCPY10	ISRT
2.		ISRT100
3.		ISRT105
4.		ISRT180
5. Possible status codes are KA, KB, KC, and KD.		SEGCV

Extended Description	Routine	Label

Figure 2-40.2. Field Level Sensitivity Replace

INPUT

Buffer

old segment

From Figure 2-40 Step 5

PROCESS

1. Read segment to be replaced

   DLZDBH00 — Buffer Handler 2-15

   and move to output I/O area.

2. If concatenated segment, read destination parent segment.

   DLZDBH00 — Buffer Handler 2-15

3. Call DLZSEGCV.

   DLZSEGCV — Segment convert Routine 2-40.3

   If conversion error, return status code to user.

4. If REPLACE=NO option specified, scan FSB chain and compare new field to old field. If field modified, return status code to user.

   Return to Figure 2-8.1 Step 9

5. Return.

   To Figure 2-40 Step 6

OUTPUT

DBPCBSTC

status code

DLZCPY10 – Field Level Sensitivity Copy

DLZCPY10

Extended Description	Routine	Label
1.	DLZCPY10	REPL
3. Possible status codes are KA, KB, KC, and KD.		SEGCV
4. Status code KE is returned.		REPL135

Extended Description	Routine	Label

Figure 2-40.3. Field Level Sensitivity Segment Convert

INPUT

FER

SDB

FSB

FER

From Caller
See note.

PROCESS

1. The SDB is used to locate the SDB expansion block and the first FSB.
2. If SSA call, compare SSA field name to FSB field name. If no match, go to step 7.
3. Initialize FER fields and set to and from parameters.
4. If to and from types different, convert field to desired format.
5. Move field to destination.
6. Call user field exit routine, if any.

   user field exit routine

   If error indicated in FER, return error code to caller.
7. If another FSB to process, go to step 2. Otherwise, return to caller.

Return to Caller

OUTPUT

FERPCSC

error code

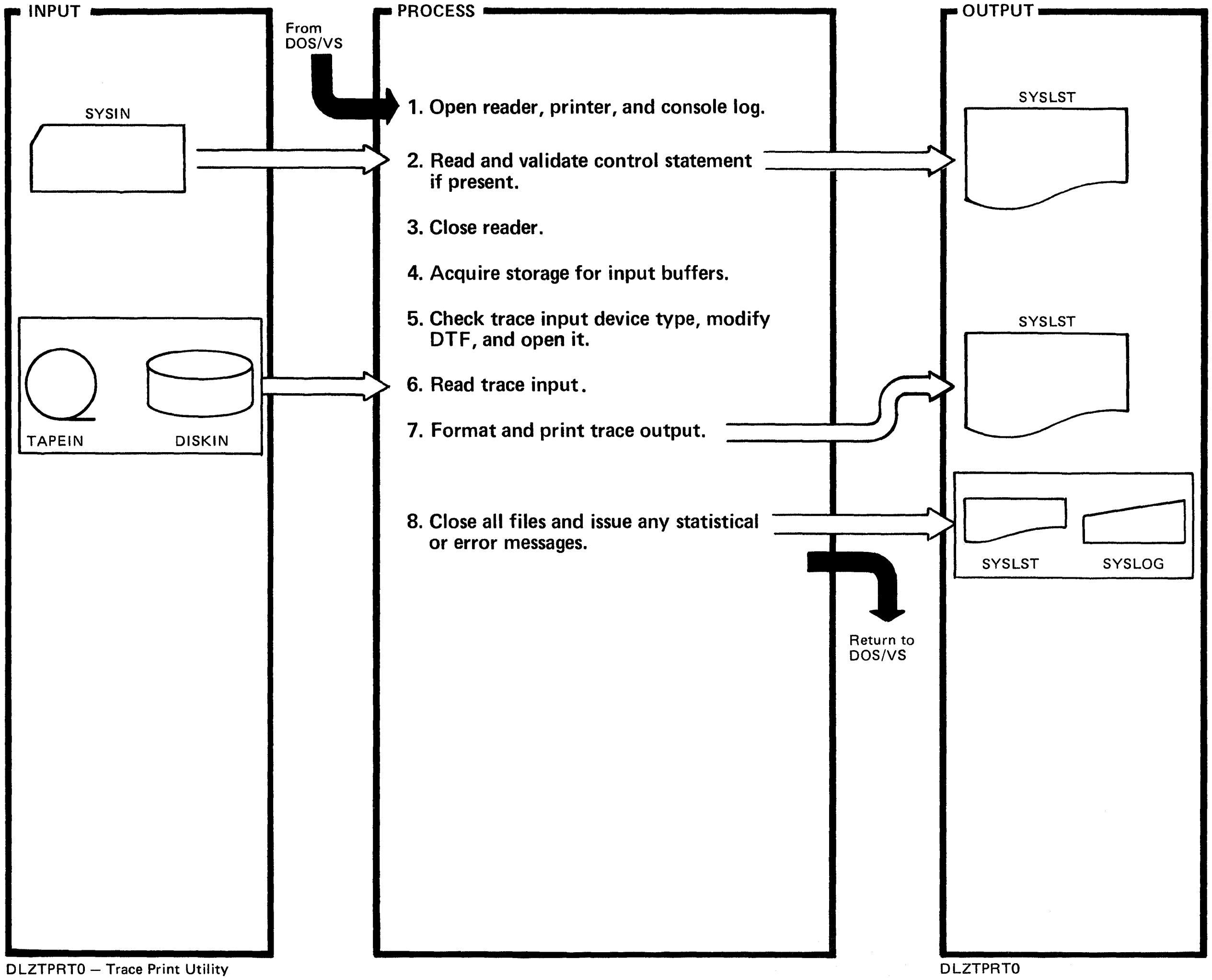
DLZSEGCV — Field Level Sensitivity Segment Convert

DLZSEGCV

Extended Description	Routine	Label
Note: DLZSEGCV is called by DLZCPY10 and DLZDLR00.		
1.	DLZSEGCV	DLZSEGCV
2. DLZDLR00 makes SSA call to convert SSA user field to physical view. Only this field and its subfields will be converted.		FSBLOOP
3.		FSB010
4.		CONVERT
5.		MOVE
6. Possible error codes are A, B, C, and D.		USEREXIT
7.		NEXTFSB

Extended Description	Routine	Label

Figure 2-41. Trace Print Utility



Extended Description	Routine	Label
2. Reads TI statement if present and prints card image on SYSLST. If any other type statement is present, it is also printed, but no further processing of it takes place.	DLZTPRT0	READCARD
4. Storage is acquired for two input buffers.		GETSTOR
5. The DLZDVCE macro is used to validate the trace input device, and modify the tape or disk DTF.		CKDEV
6. The unformatted trace records are read from the trace input file until EOF is returned.		GETENTRY
7. The trace entries are formatted and printed one at a time until all entries in the record are processed. Control is then passed to step 6 for the next record.		PRTRACE

Extended Description	Routine	Label

SECTION 3: PROGRAM ORGANIZATION

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

SYSTEM CONTROL MODULES

DLZRRC00 - BATCH INITIALIZATION

The responsibilities of this module are to:

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

Entry Interface - DLZRRC00

DLZRRC00 receives control from DOS/VS job control

Exit Interface

DLRRC00 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 PRIMARY INTERFACE

This routine receives control from the DL/I initialization routine and serves as the primary interface for all DL/I program executions. Its responsibilities are:

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

- Invoke parameter analysis (DLZRRA00)
- Branch to application program control module (DLZPCC00)

Entry Interface - DLZRRC10

Receives control through branch from DLZRRC00

Register Contents

R7	Address of ERRORMSG
R10	Entry point address

Exit Interface - Parameter Analysis

Passes control through fall through to DLZRRA00

Register Contents:

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

DLZRRA00 - USER PARAMETER ANALYSIS

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

Layout and Description of PARM Field

xxx,aaaaaaaa,bbbbbbb,ccc,keyword operands

xxx	PARM identifier in columns 1-3. DLI = Data base program to be executed. UDR = Data base recovery utility to be executed. ULU = Data base reorganization or logical relationship resolution program to be executed. ULR = HD reorganization reload utility to be restarted from checkpoint record.
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 regions, It performs some functions analogous to those performed by the CICS scheduler in the online control program. It is responsible for the following functions:

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

Data Areas Used

PST
SCD
DDIR
DMB
SDB
PSIL

Entry Interface

Receives control by fall through from DLZRRA00

Entry Register Contents

R2	SCD address
R9	PST address
R13	Save area address

Exit Interface

- Passes control through BAL to DLZPINIT (entry point in DLZDBLM0)
- Passes control through BAL to application program
- Passes control through BAL to call analyzer (DLZDLA00)
- Passes control through BAL to data base logger DLZRDBL0)
- Passes control to DOS/VS supervisor by issuing an SVC 14 normal EOJ supervisor call.

Exit Register Contents

- From exit to DLZPINIT:

R2	SCD address
R9	PST address
R14	Return address

- From exit to application program:

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

- From exit to DLZDLA00:

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

- From exit to DLZRDBL0:

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

DLZDBLM0 - APPLICATION CONTROL BLOCKS LOAD AND RELOCATE

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

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

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

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

These utilities perform dynamic block loading using the DLZBLKLD macro.

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

Entry Register Contents:

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

Exit Register Contents

Same as entry register contents

DLZCPI00 - BATCH CONTROL PROGRAM INITIALIZATION

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

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

Entry Interface - DLZCPI00

Receives control by fall through from routine DLZDBLM0.

Entry Register Contents:

R2	SCD address
R9	PST address
R13	Save area address

Exit Interface

Returns to DLZPCC00

Exit Register Contents

R9	PST address
R2	SCD address
R14	Return address

DLZLI000 - LANGUAGE INTERFACE

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

The language interface has responsibility for:

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

Entry Interface - DLZLI000

Receives control through branch from application program

Entry Register Contents:

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

Exit Interface

Passes control to program request handler through branch from DLZLI000

Exit Register Contents:

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

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), validates them, 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

<u>Data Areas Used</u>

PPST
PST
SCD

<u>Entry Interface</u>

Receives control through branch from language interface (DLZLI000)

<u>Entry Register Contents</u>

R0	Language indicator (zero if COBOL or Assembler; nonzero if PL/I)
R1	Parameter list address (in application program format)
R13	Save area address
R14	Return (to application program)
R15	Entry point address

<u>Exit Interfaces</u>

- 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

<u>Exit Register Contents</u>

- From exit to DLZDLA00:

R1	PST address
R13	Save area address
R14	Return address

R15 Entry point of call analyzer (obtained from SCD) at label SCDDLICT)

- From exit to ERRORMSG:

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

- From exit to DLZABEND:

  R15 entry point to DLZABEND

- From exit to application program:

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

DLZABEND - STXIT ABEND

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

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

Entry Interfaces

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

Exit Interfaces

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

Exit Register Contents

- From exit to DLZRDBL0:

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

- From exit to DLZDLA00:

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

DLZIWAIT - DL/I IWAIT

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

Entry Interface

Receives control through BALR from a DL/I action module

Entry Register Contents:

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

Exit Interface

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

ONLINE DL/I PROCESSOR MODULES

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

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

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

DLZOLI00 - ONLINE INITIALIZATION

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

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

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

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

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

Each ACT program entry is generated from the DLZACT TYPE=PROGRAM statement. These statements define to DL/I which application programs can use DL/I online. They also define which PSB names can be used by each of the application programs. There is one ACT program 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.

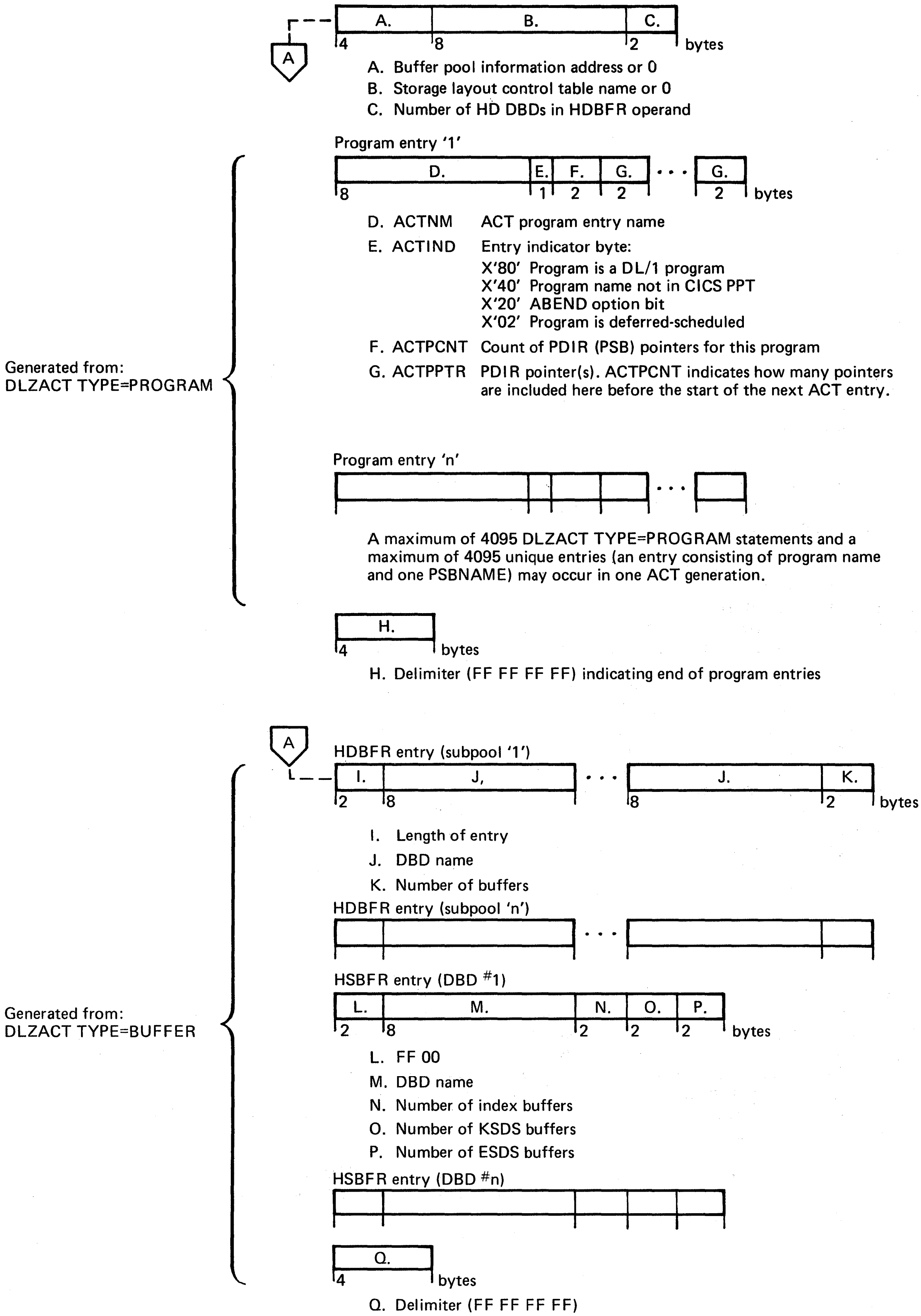


Figure 3-1. Application Control Table (ACT) Format

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 DOS/VS COMREG using the MVCOM macro. Next, the application control table (ACT) is located and an indicator is set in each corresponding PPT entry for all application programs which will use DL/I. Each PSB name in the ACT is eight characters in length.

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
DLZDLE0	-	Load/Insert
DLZDHDS0	-	Space management
DLZDXMT0	-	Index maintenance
DLZDLOC0	-	Open/Close
DLZQUEF0	-	Program Isolation ENQ/DEQ module
DLZQUEFW	-	Program Isolation ENQ/DEQ work area

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

DL/I task scheduling is also comprised of the following subroutines:

- PST initialization
- PSB intent scheduling
- PSB initialization
- Scheduling

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 encountered in this program's ACT entry is provided as default. This subroutine determines whether DL/I can support another task and creates an entry in the PST prefix area for this task.

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

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

PST Initialization

PST storage is acquired from CICS/VS Storage Management and the storage address is saved in the assigned PST prefix. PST initialization consists of formatting the save area chains and storing the address of the assigned PST prefix. Control is passed to the PSB intent scheduling subroutine.

PSB Intent Scheduling

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

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

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

PSB Initialization

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

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

Scheduling

A DL/I call initiates scheduling. The function code is 'PCB' and the call contains the name of the PSB to be executed. The call is passed to the online program request handler via the 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 instead of the CICS/VS TCA for returning return code and PCB address list information to the application program. Next, the task scheduling subroutine is called to schedule the PSB. 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.

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

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.

Language Interface Module

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

Program Request Handler

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

Upon entry, if it is a scheduling call, the scheduling subroutine is entered. If not, addressability to the PST is established and the language indicator is set in the LIPARMS section of the PST. Next the user's call list is inspected to determine whether it is in the proper format. If not, the list is converted to the implicit direct format in an area provided in the PST. The address of the list is stored in the PST. Then the call list is checked to ensure that all addresses are valid. If valid, and if the PSB is on the same system, the call is passed to the call analyzer. Otherwise, if the PSB is on a remote system, control is passed to the remote data base call subroutine which transfers the call to the remote system. If the call is invalid, a return code is inserted in the UIB or TCA and control is returned to the user.

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

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

IWAIT Routine

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

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

DLZODP01 - TASK TERMINATION

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

Task Termination

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

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

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

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

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

System Resource Allocation

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

DLZODP02 - DL/I NORMAL SYSTEM TERMINATION

The following processing occurs prior to CICS/VS termination.

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

DLZODP03 - DL/I ABNORMAL SYSTEM TERMINATION

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

- The DL/I log DTF is located and a DOS/VS CLOSE is issued for the DL/I log.
- The DL/I control blocks are dumped.
- Return is made to the calling CICS/VS program.

DLZERMSG - DL/I ONLINE MESSAGE WRITER

The following processing occurs:

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

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

DLZOVSEX - VSAM EXCP EXIT PROCESSOR

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

DLZDLA00 - CALL ANALYZER

The call analyzer module is used for initiation of all data base calls. Under normal circumstances, it receives control from the DL/I online program request handler (DLZPRH00) in the CICS-DL/I region or from the batch application program request handler (DLZPRHB0). It receives control from application program control (DLZPCC00) at termination of a DL/I batch partition or online task termination (DLZODP01) in a CICS-DL/I region.

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

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

The primary responsibilities of the call analyzer are:

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

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

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

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

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

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

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

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

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 MODULE

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

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

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

  For PROCOPT ≠ L or LS:

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

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

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

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

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

- Control interval size
- Key length (KSDS)
- Relative key position (KSDS)

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

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

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

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

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

Register Contents

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

Control Blocks - DLZDLOC0

* DL/I control record - DLZREC0

* PSTFNCTN field of the PST:

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

DLZDLD00 - DELETE/REPLACE

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

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

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

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

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

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

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

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

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

Delete/Replace Work Space Acquisition and the Work Space Prefix

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

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

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

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

Segment Delete Codes

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

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

Interfaces - DLZDLD00

This module interfaces with the following modules:

DLZDBH00
DLZDHDS0
DLZRDBL0
DLZDXMT0
DLZQUEF0

Control Blocks - DLZDLD00

- Delete workspace prefix
- Delete work area.

Register Contents at Entry

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

Register Contents at Exit

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

Register Contents on ABEND - in the SCD ABEND Save Area

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

DLZDDLE0 - LOAD/INSERT MODULE

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

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

The module performs the following functions:

A. HDAM/HIDAM load/insert -

1 Normal segment:

- Positioning: retrieve positions for inserting and loading of HDAM roots. For all other loading, DLZDDLE0 simulates retrieve positioning.
- Space for new segment is acquired using the space management module, DLZDHDS0.
- The segment is moved from the user's I/O area to the buffer.
- Prefix pointers are updated.
- Actual write is performed by the buffer handler using VSAM.
- Prefix pointers of twins and parents are updated.

- The data base logger (DLZRDBL0) is called to write the new segment and the updated prefixes.

- If the segment is an index source segment, index maintenance (DLZDXMT0) is called.

- Exit is to the call analyzer.

2. Concatenated segment:

- If the destination parent already exists, and the insert rule is physical or logical: same as normal segment.

- If the destination parent exists and the insert rule is virtual: the logical child segment is inserted as for a normal segment, data of destination parent are replaced afterwards.

- If the destination parent does not exist and the rule is not physical, the destination parent is inserted as for a normal segment; afterwards the logical child is inserted as a normal segment.

B. HISAM and simple HISAM load.

- Main storage for a logical record for key sequenced data set (KSDS) and for entry sequenced data set (ESDS) is acquired from the buffer handler.

- The root and all dependent segments that fit into one logical record are written to the KSDS, using the buffer handler. The remaining dependent segments are moved to one or more records of the ESDS.

- Pointers to those records are inserted.

C. HISAM and simple HISAM root insert

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

- Old (if deleted) and new records are logged.

D. HISAM dependent segment insert

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

- Pointers to the new records are created.

- Old and new records are logged.

E. HSAM and simple HSAM load

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

Blocks and Tables - DLZDDLE0

PST
DDIR
DMB
PCB
JCB
Level table
SDB
FDB
SCD

Registers on Entry and to All Called Modules

R1 = PST

Interfaces - DLZDDLE0

This module calls the following modules:

DLZRDBL0	- Data base logger
DLZDBHC0	- Buffer handler
DLZDHDS0	- Space management
DLZDXMT0	- Index maintenance
DLZQUEF0	- Queuing Facility

Status Codes - DLZDDLE0

II
AC
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 DLZDDLEO (load/insert) and DLZDLD00 (delete/replace)

Register Contents

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

PSTWRK1 LSDB of ISS for ISRT, ASTR, REPL calls
LSDB of ROOT for UNLD CALL

```
                PSDB of ISS for DLFT call
PSTFNCTN        'A0'    Delete
                'A1'    Replace
                'A2'    Insert
                'A3'    Unload
PSTBYTNM        RBA of index source segment
```

Interface to called modules:

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

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

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

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

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

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

5. DLZDSEH0 (work data set module)

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

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

Exits:

Back to calling module.

Control Blocks - DLZDXMT0

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

DLZDLR00 - RETRIEVE

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

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

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

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

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

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

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

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

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

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

Command Codes Affecting Retrieval

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

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

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

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

Module Layout - DLZDLR00

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

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

DLZRTLR - Last macro of a subroutine.

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

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

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

Maintenance Support - DLZDLR00

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

Interfaces - DLZDLR00

This module interfaces with the following modules:

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

Register Contents on Entry and Return

R0 = SCD
R1 = PST
R2 = PCB

Register Contents During Execution

R0 = Work
R1 = Work
R2 = 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
R11= Base register for linkage routine DLZRLNKD
R12= Base register
R13= Save area
R14= Work
R15= Work

DLZDHDS0 - HD SPACE MANAGEMENT

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

Module DLZDHDS0 consists of CSECTs which perform the following functions:

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

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

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

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

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

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

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

DLZMPLC0 determines the block number for the bit map block appropriate to the block number passed to it. It also determines the relative bit position in the bit map block of the block number passed to it.

DLZMMUD0 turns the appropriate bit ON or OFF according to the entry point involved. The log is also called to reflect the change.

DLZDCI00 tests to see if the device containing the data base is actually an FBA device if it was specified as such, and, if it is, calculates the CIs per track and per cylinder and the scan value in cylinders equivalent to the number of FBA blocks specified during DBD generation. These values are stored in the DMB for later use.

Interfaces - DLZDHDS0

The following modules are called by DLZDHDS0:

DLZDBH00 - Buffer handler
DLZRDBL0 - Data base logger

Calling Sequence

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

On Return

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

DLZDBH00 - DB BUFFER HANDLER

The primary functions of module DLZDBH00 are:

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

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

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

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

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

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

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

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

(a) Accessed by VSAM RBA

Symbol Function	Hex Function	Description
PSTBYLCT	02	Retrieve the VSAM KSDS record by the RBA which is in PSTBYTNM.
PSTBFALT	05	Write the VSAM KSDS record by the RBA which is in PSTBYTNM.
PSTERASE	0A	Delete the VSAM KSDS record identified by the RBA which is in PSTBYTNM.

(b) Accessed by key

Symbol Hex

Function	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 PSTMSPUT
- PSTGETNX PSTERASE
- PSTSTLBG PSTWRITE
- PSTPUTKY

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

DLZDBH03 Contains code for the functions

- PSTFBSPC
- PSTBFMPT
- PSTPGUSR

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

Write Chain

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

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

Use Chain

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

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

ENQ/DEQ Subroutines

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

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

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

BFPLINMA	interlock detection matrix
BFPLINW1	work areas
BFPLINW2	

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

BFPLNQW1
BFPLNQW2

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

Enqueuing of a resource consists of the following steps.

If the resource is available:

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

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

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

If the resource is not available:

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

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

3. Check whether this waiting would cause an interlock.

4. If no interlock possible:

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

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

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

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

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

Dequeuing of a resource consists of the following steps.

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

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

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

4. If another PST was waiting on this resource:

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

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

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

    d. Return to caller.

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

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

The following types of ENQ requests may occur:

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

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

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

Control Blocks - DLZDBH00

PST
PPST
DDIR
DMB
DSG
SCD
BFPL
BFFR
SBIF

Interfaces - DLZDBH00

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

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

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

Buffer Handler Functions and Required Fields

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

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

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

2. Functions used to access a HISAM ESDS

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

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

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

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

Function	Input		Output	
	Field	Contents	Field	Contents
PSTBYLCT	PSTBYTNM	RBA of the logical record to be retrieved	PSTDATA	Address of the record within the buffer
PSTBFALT	PSTBYTNM	RBA of the logical record to be written		
	PSTDATA	Address of the 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
PSTNUMRO	Number of reads done during this call
PSTNUMWT	Number of writes done during this call

PSTCLRWT

Bit 0	This caller waited during request
1-8	Reserved

PSTRTCDE

Return Code Function	Hex Function	Description
PSTCLOK	00	No error occurred during this request.
PSTGTDS	04	Record, CI, or segment requested is more than one CI beyond the end of the data set - returned on PSTBKLCT, PSTBYLCT, PSTBYALT
PSTIOERR	08	Requested CI, record, or segment could not be read successfully on a PSTBKLCT, PSTBYLCT, or PSTBYALT call or could not be written successfully on a PSTPUTKY, PSTMSPUT, PSTWRITE, or PSTBFALT call.
PSTNOSPC	0C	An out of space condition occurred on the data set DASD while processing this request.
PSTBDCAL	10	The byte at PSTFNCTN is not a valid function or the DMB/ACB/BLKID in the PST do not match corresponding fields pointed to in PSTBUFFA for a PSTBFALT call.
PSTNOTFD	14	A PSTSTLEQ call has been issued for a record whose key is higher than the highest key in the data set.
PSTNWBLK	18	The requested CI, record, or segment will go in the CI, one greater than the current end of the data set. Space has been allocated in the pool to hold the new CI. The address is at PSTDATA.
PSTNPLSP	1C	The pool does not contain enough space to satisfy the request.
PSTWROSI	20	A request (GBSPC) was issued for a buffer size which exceeds the highest buffer size handled by any subpool.

PSTENDDA	24	The end of data set has been reached on a PSTGETNX call.
PSTBYEND	28	A request has been issued with a key or RBA higher than the highest key or RBA in the data set.
PSTEOD	2C	End of data set has been reached on a request by DLZDLOC0.
PSTINLD	34	Invalid request during data set loading.

DLZRDBL0 - DB LOGGER

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

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

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

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

There are five basic types of data base log records.

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

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

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

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

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

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

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

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

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

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

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

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

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

Control Blocks - DLZRDBL0

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

Register Contents

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

High-order byte of PSTWRK1 field in PST:

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

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

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

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

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

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

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

Logging in the Online System

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

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

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

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

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

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

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

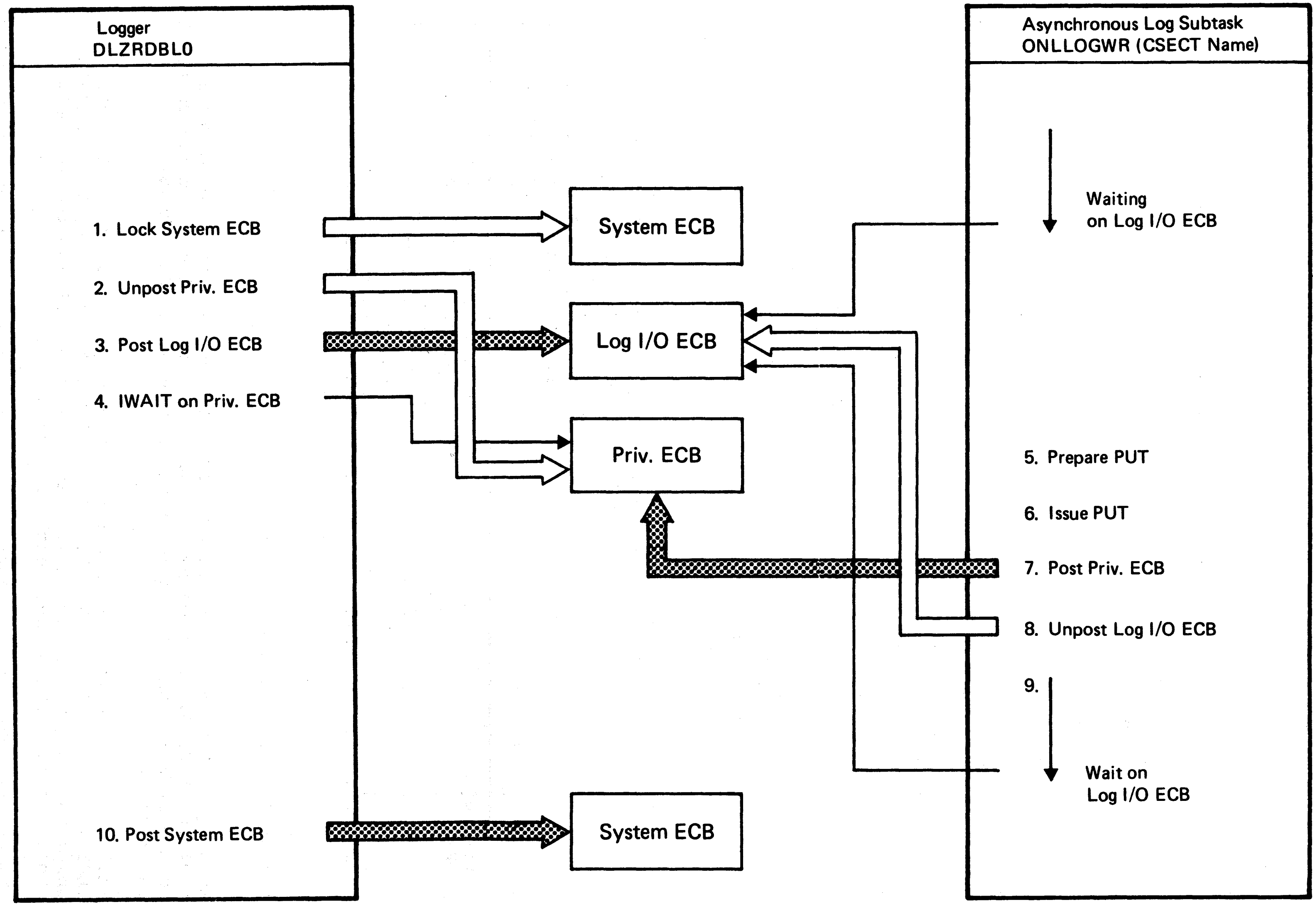


Figure 3-2. Online Log Block Put Operation

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

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

DLZRDBL1 - CICS JOURNAL LOGGER

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

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

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

LL	ƀƀ	REC. ID	CONTINUED ACCORDING TO DSECT
0	2	4	

Figure 3-3. DL/I Log Record

SYSTEM PREFIX				USER PREFIX	JOURNALLED DATA
LL	ƀƀ	REC. ID	...		
0	2	4			

Figure 3-4. CICS Journal Record

LL	ƀƀ	CICS/VS LABEL RECORD	ANY COMBINATION
OF CICS/VS JOURNAL RECORDS AND			
DL/I LOG RECORDS			

Figure 3-5. Layout of a Journal Block

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

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

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

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

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

When log information is written by using CICS journaling, the writing of log information is always ahead of updating the associated data base blocks. The scheme used is the same as with standard logging, the only difference being that the value for the number of written journal blocks (CICS 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 3-6, the RDBs are chained together, both forward and backward, to one of several queue heads located in the QWA (queuing facility work area). Note that the queue heads have only a forward pointer. The proper queue head is determined by hashing the resource ID and using the results as an index to the table of queue headers.

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

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

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

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

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

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

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

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

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

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

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

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

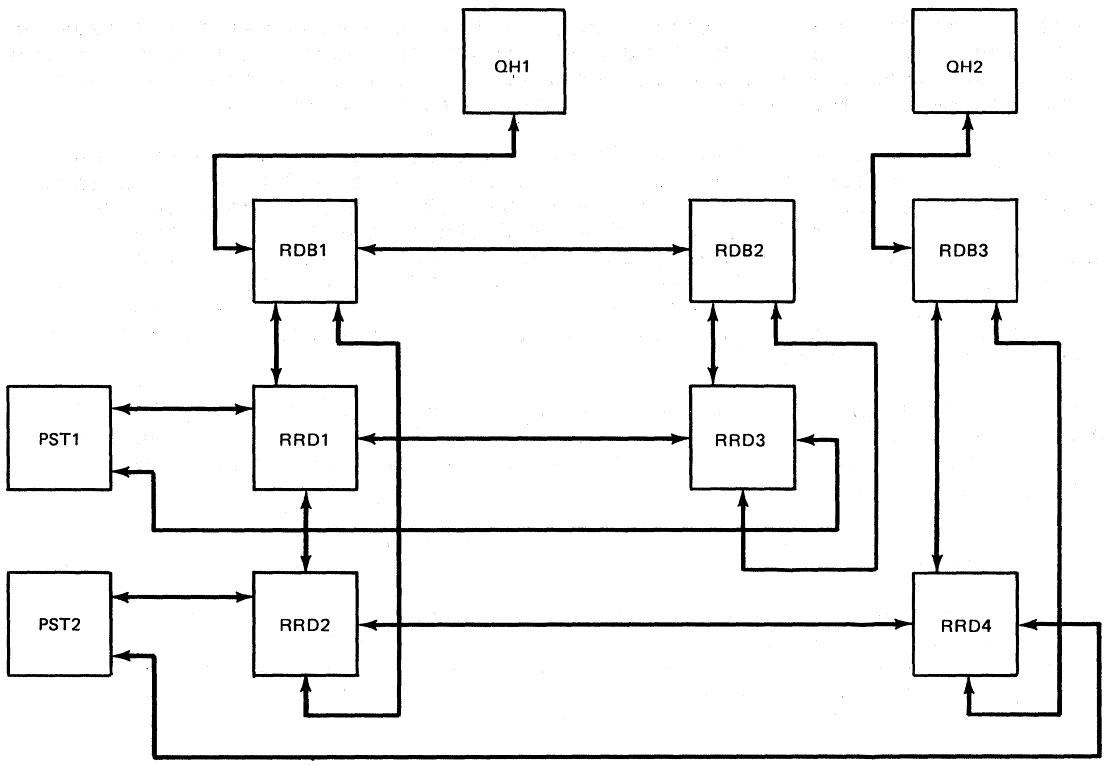


Figure 3-6. Enqueue/Dequeue Control Block Relationships

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

Mainline Routines

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

Functional Subroutines

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

Data Areas Used

SCD
PPST
PST
RDB
RRD
QWA

Entry Points

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

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

DLZCPY10 - FIELD LEVEL SENSITIVITY COPY

DLZCPY10 has two entry points: DLZCPY10 and DLZSEGCV.

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

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

Interfaces - DLZCPY10

This module interfaces with the following module:

DLZDBH00

Register Contents at Entry

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

Control Blocks - DLZCPY10

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

MPS CONTROL MODULES

DLZMSTR0 - START MPS TRANSACTION

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

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

Control Blocks Addressed

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

Register Contents

R13 Contains CSA address

DLZMPC00 - MASTER PARTITION CONTROLLER (MPC)

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

The functions performed by the master partition controller are:

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

Control Blocks Addressed

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

Register Contents

R12 Contains TCA address (at entry)

R13 Contains CSA address (at entry)

Macros Used

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

DLZBPC00 - BATCH PARTITION CONTROLLER (BPC)

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

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

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

Control Blocks Addressed

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

Register Contents

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

Macros Used

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

DLZMPI00 - MPS BATCH

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

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

MPS Batch Initialization - DLZMINIT

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

DLZMINIT reads the input parameter statement and checks it for validity. It then loads the user's program. Then it determines what to use as a partition identifier by checking the PIK in the BG COMREG. This value, modified and made printable, is put into each XECBTAB macro issued.

After saving the program name and PSB name for use by online, an XECB, DLZXCBn1, is defined in the batch partition for communicating with the online partition. The online partition XECB (DLZXCBn0, with n being the identifier) is XPOSTed. This lets the online partition know that there is an MPS batch job ready to run in this batch partition.

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

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

Control Blocks Addressed

MPCPT	MPC Partition Table
TCA	Transaction Control Area
PST	Partition Specification Table
COMREG	Communication Region
XCB1	XECB DLZXCBn1 and data following it
DTFs for	SYSLST, SYSLOG, and SYSIPT
STXIT AB	Savearea
STXIT PC	Savearea
XECBs	DLZXCBn0, DLZXCBn2, DLZXCBn3

Register Contents (at Entry to Other Routines

- User Program
  - R1 PCB list if not PL/I; or a pointer to a list containing the following if PL/I:
    - address of PCB list
    - address of location containing size of dynamic storage
    - address of start of dynamic storage
  - R13 Save area
  - R14 Return address
  - R15 Entry address
- Message Writer (DLZMMSG)
  - R14 Return Address
- ABEND Routine (DLZMABND)
  - No special register values

Macros Used

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

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
XECBTAB TYPE=DELETE

MPS Batch Program Request Handler -DLZMPRH

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

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

Control Blocks Addressed

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

Register Contents

- At entry:
  - R0 If=1, PL/I; if=0, not PL/I and value is ignored
  - 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

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 three entries:

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

The first two each identify which way the abend routine was entered. They then send an error massage. Then the third 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

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

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.

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

Control Blocks Addressed

DTFs for SYSLOG and SYSLST

Register Contents

- At entry:
  R14 Return address
  Base registers already initialized

- At entry to message table (DLZMMSGT):

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

Exits

To calling routine via branch register 14

Macros Used

PUT

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 notifies (XPOST) the particular XECB that causes the MPC to end the MPS environment.

After the XPOST, 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.

<u>Macros Used</u>

XECBTAB TYPE=CHECK

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.

Register Contents on Entry

R1 = PSB list address
R13 = Save area
R14 = Return

R15 = Entry point

Control Blocks - DLZBACK0

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

External Modules Called

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

Record and Message Formats - DLZBACK0

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

DLZRDBC0 - DB CHANGE BACKOUT

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

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

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

Register Contents and Control Blocks on Entry

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

Control Blocks - DLZRDBC0

Data base log record
DDIR
DMB
DSG
PCB
PDIR

PSB
PST
SCD

External Modules Called

DLZDBH00	-	Called to read a data base record and to mark the buffer altered
DLZDHDS0	-	Called to free or reserve space in an HDAM or HIDAM record
DLZDLCC0	-	Called to open data base
DLZRDBL0	-	Called to log backout modifications to data base

Interface with External Modules

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

DLZDLOC0

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

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

DLZDBH00

R1 = address of PST

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

DLZDHDS0

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

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

DLZRDBL0

R0 = SCD address
R1 = PST address

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

Register Contents on Exit

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

Error Codes and Handling - DLZRDBC0

All error codes are passed to DLZBACK0 in register 15.

DLZURDB0 - DB DATA SET RECOVERY

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

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

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

Blocks and Tables - DLZURDB0

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

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

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

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

4. Data base change log records.

Normal Entry Points

The only entry point to this module is DLZURDB0.

Register On Entry

R1 = pointer to fullword containing address of PCB

Registers On Exit

All registers are restored to entry conditions.

Modules Called by DLZURDB0

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

R1 = pointer to recovery common area

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

R1 = pointer to PST

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

R1 = pointer to PST

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

R1 = pointer to PST

Error Codes and Handling - DLZURDB0

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

DLZURCC0 - Recovery Control Statement Processor

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

Normal Entry Point

The only entry point to this module is DLZURCC0.

Registers on Entry

R1 = pointer to recovery common area.

Registers on Exit

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

Error Codes and Handling

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

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

DLZUDMP0 - DB DATA SET IMAGE DUMP

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

1. A control card is read from SYSIPT and preliminary validity checking is performed on various fields. The input card defines the data base/file to be dumped, the dump output symbolic filenames, and the number of output copies to be created.
2. The device type is determined for each output file specified and the file(s) are opened.
3. The DMB is loaded from a core image library to obtain the physical characteristics of the data base file to be dumped.
4. A header record is written to the output file. This record contains information necessary to allow the use of the image dump file by the data base data set recovery utility.
5. The input file is opened.
6. Input segments are read sequentially, an 8-byte prefix is added to identify the segment, and the logical record (prefix + segment) is blocked and written to the output file.
7. After all segments have been copied (EOF), the input and output files are closed.
8. Output statistics for the file are written to SYSLST.
9. Processing continues from step 1 until there are no more input cards, at which time the program terminates.

Control Blocks - DLZUDMP0

- Dump record prefix
- Dump header record.

Error Codes and Handling - DLZUDMP0

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

DLZUCUM0 - DB CHANGE ACCUMULATION UTILITY

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

Control Blocks - DLZUCUM0

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

Normal Entry Point

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

Entry Conditions

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

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

DLZUCER0 - Common Error Routine

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

Normal Entry Point

The only entry to this module is DLZUCER0.

Entry Conditions

This module is entered to output all error messages.

Register Contents on Entry

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

Register Contents on Exit

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

DLZUCCT0 - Control Card Processor

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

Normal Entry Point

The only entry to this module is DLZUCCT0.

Entry Conditions

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

Register Contents on Exit

All registers are restored to entry conditions.

DLZUC150 - Sort Exit 15

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

Normal Entry Point

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

Entry Conditions

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

Register Contents on Exit

All registers are restored.

DLZUC350 - Sort Exit 35

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

Normal Entry Point

This module is entered at DLZUEX35 by sort.

Register Contents on Entry

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

Entry Conditions

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

Register Contents on Exit

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

DLZLOGP0 - LOG PRINT UTILITY

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

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

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

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

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

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

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

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

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

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

Error Codes and Handling

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

DLZLPCC0 - Log Print Control Statement Processor

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

Normal Entry Point

This module is entered at DLZLPCC0 by DLZLOGP0.

Register Contents on Entry

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

Entry Conditions

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

Register Contents on Exit

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

Error Codes and Handling

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

DLZURUL0 - HS DB UNLOAD

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

Next, the input and output data sets are opened. A header record containing information about the data base data sets is constructed, and a statistics record is written. The first KSDS record is then read and the root segment is checked to determine whether the deleted flag is on (no prefix if Simple HISAM). If it is on, the total segment chain for that root is ignored, and the next root is processed. If the root is not deleted, it is moved to the output area, and the first dependent segment, if present, is processed. If the dependent segment is not deleted, it is moved to the output area, and the next segment is processed. This continues until the complete dependent segment chain for this root, including any overflow dependent segments on the ESDS, have been processed. If the segment is deleted, each succeeding segment that is a child of the deleted segment is also deleted. The first segment that is not a child of the deleted segment causes the normal segment processing to be resumed. The last record written is a statistics record which includes information needed for audit trail. The output data set now contains the reorganized KSDS and ESDS logical records in physical sequential format (only KSDS if Simple HISAM). 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 3-7.

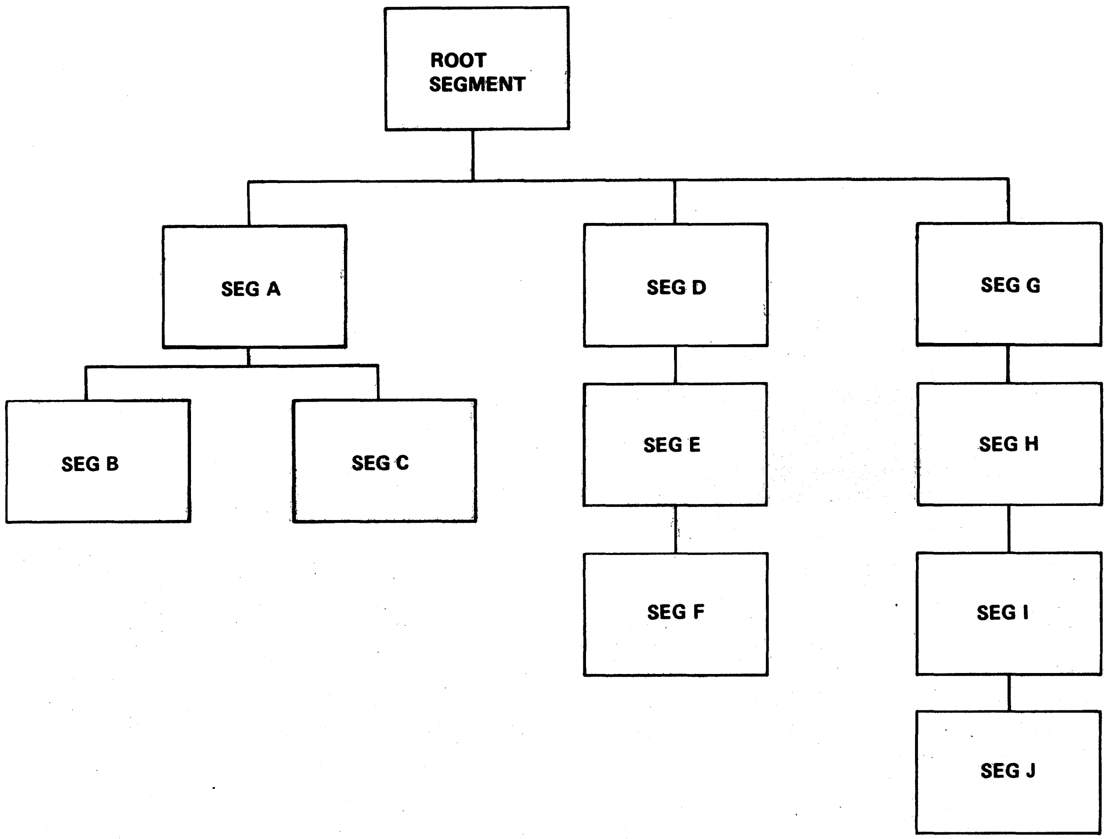


Figure 3-7. HISAM Data Base with One Root Segment

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

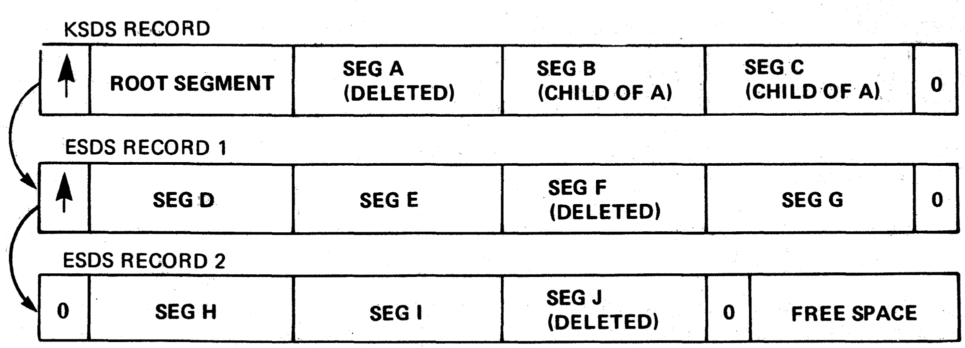


Figure 3-8. Input for HISAM Reorganization Unload Utility

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

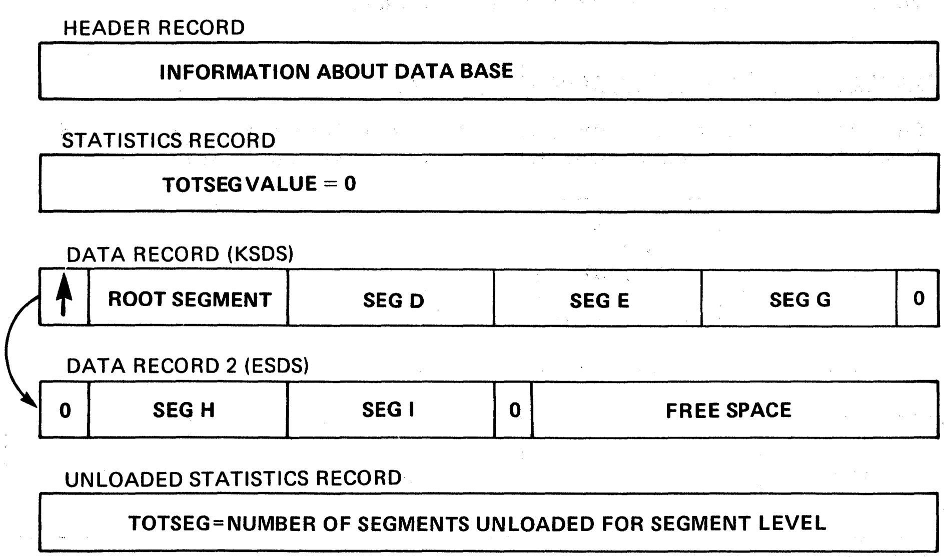


Figure 3-9. HISAM Reorganization Unload Utility Output

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

DLZURRL0 - HS DB RELOAD

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

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

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

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

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

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

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

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

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

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

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

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

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

Control Blocks - DLZURGU0

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

Interfaces - DLZURGU0

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

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). 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.

APPLICATION CONTROL BLOCKS CREATION AND MAINTENANCE

DLZUACB0 - ACB CREATION AND MAINTENANCE

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

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

Each name on the constructed build list is then passed to the application control 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

Interfaces - DLZUACB0

This module interfaces with the following modules:

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

Register Contents

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

DLZUSCH0 - ACB MAINTENANCE BINARY SEARCH/INSERT

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

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

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

Operation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Control Blocks - DLZUSCH0

- List control block
- Sorted list block.

Programming Note

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

DLZLBLM0 - ACB Generation Error Message Handler

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

Register Contents on Entry - DLZLBLM0

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

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

External Routines Called - DLZLBLM0

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

DLZDLBL0, DLZDLBL1, DLZDLBL2, DLZDLBL3 - ACB BUILDER

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

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

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

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

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

Interfaces - DLZDLBL0 - DLZDLBL3

These modules interface with the following modules:

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

Register Contents on Entry

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

Register Contents on Exit

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

PSTERCOD = 0	Valid return
PSTERCOD ≠ 0	Errors encountered

DLZDPSB0 - UTILITY PSB BUILDER

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

Register Content on Entry

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

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

Registers on Exit

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

R15 = 0 Valid return
R15 ≠ 0 Errors encountered

DLZURPRO - PREREORGANIZATION

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

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

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

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

Control Blocks - DLZURPRO

- Control file consisting of one or more records, each with a pointer to the next block of control file and an area containing one or more control list entries.
- List entry.
- Secondary list entry.

Interfaces - DLZURPRO

The interface with the reorganization message module (DLZURGMO) 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 - DLZURPRO

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.

DLZDSEHO - 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)
	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 32767 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. 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.

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 which 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).

- CORE ID

  The core ID for the applicable modules. This is located near the beginning address of each module and is usually followed by the version, release, level, and latest PTF number applied.

- SUPPLEMENTARY INFORMATION

  The entry SVA means that the module concerned is eligible to be loaded into the shared virtual area (SVA). Any other entry in this column is the entry point name that must be present on the END card when assembling this module, for example, END DLZBEGIN.

- FIGURE REFERENCE

  The figure number that is shown after the module name refers to the figure number of the module's HIPO diagram in Section 2 of this manual.

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
SYSTEM CONTROL MODULES					
** Batch Initialization ** (See Figure 2-3)					
DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRC00	DLZRRCST
	*ERRORMSG				
	DLZMMSGT	DLZMMSGT	DLZMMSGT		
	DLZRDR				
	DLZCONSL				
	DLZRRC10				
	*DLZRRA00				
	*DLZPCC00				
	*DLZDBLM0				
	*LOADDMBS				
	*PCBROUT				
	*DLZCPI00				
	*DMBLOADR				
** Batch Nucleus ** (See Figure 2-4)					
DLZBNUC0	SCDCSECT	DLZBNUC0	DLZBNUC0	DLZBNUC0	
	SCDSTART				
	*DLZIWAIT				
	*DLZPRHB0				
	*DLZABEND				
** Online Initialization** (See Figure 2-5)					
DFHSIDL	DLZOLI00	DLZOLI00	DLZOLI00	DLZOLI00	
	*DLZCPI00				
Online Nucleus (See Figure 2-6)					
DLZNUCxx	DLZODP00	DLZODP	DLZODP	DLZNUCxx	DLZODP
	DLZODP01			DLZODP01	
	DLZODP02			DLZODP02	
	DLZODP03				
	DLZODP04			DLZODP04	
	DLZODP05			DLZODP05	
	DLZOPD06				
	DLZOPD07				
	DLZPRH00			DLZPRH00	
	DLZOLT00			DLZOLT00	
	DLZOLT01				
	DLZOLT02				
	DLZISC00			DLZISC00	
	DLZISC01				
	DLZISC02				
	DLZOWAIT			DLZOWAIT	
	DLZERMSG			DLZERMSG	
	DLZOVSEX			DLZOVSEX	

Note: xx is the result of ACT generation.

** DL/I Online System Termination ** (See Figure 2-7)					
DLZSTP00	DLZSTP00	DLZSTP00	DLZSTP00		

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
DL/I FACILITY MODULES					
** Call Analyzer **	(See Figure 2-8)				
DLZDLA00	DLZDLA00	DLZDLA00	DLZDLA00	DLZDLA00	SVA DLZEPDLA
	DLZDLA01	DLZDLA01	DLZDLA01	DLZDLA01	SVA
** Retrieve **	(See Figure 2-9)				
DLZDLR00	DLZDLR00	DLZDLRA0	DLZDLRA0	DLZDLRA0	SVA
	DLZDLR10				
	DLZRETN0				
	DLZEODC0				
	DLZGERC0				
	DLZGER0				
	DLZGETS0				
	DLZCLRP0	DLZDLRB0	DLZDLRB0	DLZDLRB0	
	DLZWIPE0				
	DLZMOVA0				
	DLZMOVB0				
	DLZDELT0				
	DLZPSDB0				
	DLZHUNT0				
	DLZSETL0				
	DLZBH0				
	DLZSSDB0				
	DLZNOOP0				
	DLZCONC0				
	DLZSSA0	DLZDLRC0	DLZDLRC0	DLZDLRC0	
	DLZTAG0				
	DLZLTW0				
	DLZNOSS0				
	DLZHIDA0	DLZDLRE0	DLZDLRE0	DLZDLRE0	
	DLZHDAM0				
	DLZHISA0				
	DLZSTLA0				
	DLZSTLG0				
	DLZUPDT0				
	DLZKDTE0				
	DLZPCHK0				
	DLZISRT0	DLZDLRF0	DLZDLRF0	DLZDLRF0	
	DLZVLRT0				
	DLZAREJ0				
	DLZVLCH0				
	DLZXDFT0				
	DLZHSAM0				
	DLZALTS0				
	DLZLOGR0	DLZDLRD0	DLZDLRD0	DLZDLRD0	
	DLZRETK0				
	DLZRETI0				
	DLZKDRK0				
	DLZKDTL0				
	DLZUPDC0				
	DLZUPDL0				
(DLZDLR00)	DLZAPST0				
	DLZYENT0				
	DLZYSTC0				
	DLZYEND0				
	DLZDEQ0				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	DLZPOST0	DLZDLRG0	DLZDLRG0	DLZDLRG0	
	DLZSKPG0				
	DLZSKPS0				
	DLZSKPD0				
	DLZSKPE0				
	DLZRLNKD	DLZRLNKD	DLZRLNKD	DLZRLNKD	
** Load/Insert ** (See Figure 2-10)					
DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	DLZDDLE0	SVA
	HDROUTIN				
	HSROUTIN				
** Delete/Replace ** (See Figure 2-11)					
DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	DLZDLD00	SVA DELREPEP
	DLZDLDS0				
	DLZDLDD0				
	DLZDLDA0				
	DLZDLDR0				
** Index Maintenance ** (See Figure 2-12)					
DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	DLZDXMT0	SVA
** HD Space Management ** (See Figure 2-13)					
DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	DLZDHDS0	SVA
	DLZGGSPC	DLZGGSP0	DLZGGSP0		
	DLZRRTRN				
	DLZFRSPC	DLZFRSP0	DLZFRSP0		
	DLZLLCLC	DLZLLCL0	DLZLLCL0		
	DLZMMLCT	DLZMMLC0	DLZMMLC0		
	DLZRRHPL	DLZRCHP0	DLZRCHP0		
	DLZRCHBK	DLZRCHB0	DLZRCHB0		
	DLZRCBK2				
	DLZMMUDT	DLZMMUD0	DLZMMUD0		
	DLZMMOFF				
	DLZMMON				
	DLZRRHMP	DLZRRHM0	DLZRRHM0		
	DFSRLO30	DLZDHDS0	DLZDHD00		
	*SNAPDCB				
	*SNPSW				
	*SNPCNT				
	DLZDCI00	DLZDCI00	DLZDCI00		
** Open/Close ** (See Figure 2-14)					
DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	DLZDLOC0	
** DB Buffer Handler ** (See Figure 2-15)					
DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	DLZDBH00	SVA DLZEBH00
	DLZEBH00				
	*MAINROUT				
	ROULINK				
	*PREPENQ				
	*PREPDEQ				
	*ABEXIT				
	*BOTTOUSE				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	*ALLDEQ				
	*BFFERREL				
	*RETURN				
	DLZDBH02	DLZDBH02	DLZDBH02	DLZDBH02	
	*WRITE				
	*READ				
	*HSREAD				
	*HSWRITE				
	*LOWRITE				
	*PUTKY				
	*MSPUT				
	*STLEQ				
	*STLBG				
	*GETNX				
	DETIOERR				
	*TSTPST1				
	DLZDBH03	DLZDBH03	DLZDBH03	DLZBFH03	
	*ENQ				
	*DEQ				
	*CONVADNR				
	*MRKEMPT				
	*PGUSR				
	*CONVNARD				

** DB Logger ** (See Figure 2-16)

DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0	DLZRDBL0
	DLZIDBL0				
	IOFILA1				
	LOGOUT				
	LSCDADDR				
	IJFUZZZN	IJFUZZZN			
	IJFUZZZZ				
	IJ2N0017				
	ONLLOGWR	DLZRDBL0	DLZRDBL0		
(DLZRDBL0)	SAVE				
	PRIVECB				

** CICS Journal Logger ** (See Figure 2-17)

DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1	DLZRDBL1
	DLZRDBL0				

** Queuing Facility ** (See Figure 2-23)

DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0	DLZQUEF0
DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	DLZQUEFW	

** Field Level Sensitivity Copy ** (See Figure 2-40)

DLZCPY10	DLZCPY10	DLZCPY10	DLZCPY10	DLZCPY10	SVA
	DLZSEGCV			DLZSEGCV	

MPS CONTROL MODULES

** Start Transaction ** (See Figure 2-18)

DLZMSTR0	DLZMSTR0	DLZMSTR0	DLZMSTR0	DLZMSTR0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
** Master Partition Controller ** (See Figure 2-19)					
DLZMPC00	DLZMPC00	DLZMPC00	DLZMPC00	DLZMPC00	
** Batch Partition Controller ** (See Figure 2-20)					
DLZBPC00	DLZBPC00	DLZBPC00	DLZBPC00	DLZBPC00	
** MPS Batch ** (See Figure 2-21)					
DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	DLZMPI00	DLZMINIT
	*DLZMPRH				
	DLZMINIT				
	*DLZMTERM				
	*DLZMMSG				
	*DLZMABND				
	DLZCONSL				
	DLZDIMOD				
	DLZMMSGT	DLZMMSGT	DLZMMSGT		
** Stop Transaction ** (See Figure 2-22)					
DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	DLZMSTP0	

DATA BASE RECOVERY UTILITIES

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
** DB Data Set Image Dump ** (See Figure 2-25)					
DLZUDMP0	DLZUDMP0	DLZUDMP0	DLZUDMP0	DLZUDMP0	
	IJ2M0101	DLZUDMP0	DLZUDMP0		
	DLZDMPM0	DLZDMPM0	DLZDMPM0		
	IJJFCBZD	IJJFCBZD			
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				
** DB Change Accumulation ** (See Figure 2-26)					
DLZUCUM0	DLZUCUM0	DLZUCUM0	DLZUCUM0	DLZUCUM0	
	DLZERRTN				
	DLZUSPKL				
	DLZWORK#				
	DLZPRNT				
	DLZSLOG				
	DLZUCONS				
	DLZUCCT0	DLZUCCT0	DLZUCCT0	DLZUCCT0	
	DLZUC150	DLZUC150	DLZUC150	DLZUC150	
	DLZUEX15				
	DLZUC350	DLZUC350	DLZUC350	DLZUC350	
	DLZUEX35				
	DLZUCER0	DLZUCER0	DLZUCER0	DLZUCER0	
	DLZCUMM0	DLZCUMM0	DLZCUMM0		
	IJFSZZWN	IJFSZZWN			
	IJFVZZWZ				
	IJFSZZWZ				
	IJGQICZZ	IJGQICZZ			
	IJGQIZZZ				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	IJGQOCZZ	IJGQOCZZ			
	IJGQOZZZ				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJ2M0014	DLZUCUM0	DLZUCUM0		
	IJFUZZZZ	IJFUZZZZ			
	IJGUIZZZ	IJGUIZZZ			

** DB Data Set Recovery ** (See Figure 2-27)

DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0	DLZURDB0
	DLZURCC0	DLZURCC0	DLZURCC0	DLZURCC0	DLZURCC0
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	CBLTDLI				
	DLZRDBM0	DLZRDBM0	DLZRDBM0		
	IJJFCBID	IJJFCBID			
	IJJFCBZD				
	IJJFCIID				
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJ2M0038	DLZURDB0	DLZURBD0		
	IJFUZZZN	IJFUZZZN			
	IJGUICZZ	IJGUICZZ			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				

** DB Change Backout** (See Figure 2-28)

DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0	DLZBACK0
	READAREA			
	IJ2M0033			
	DLZRDBC0	DLZRDBC0	DLZRDBC0	DLZRDBC0
	DLZBACM0	DLZBACM0	DLZBACM0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000
	ASMTDLI			
	IJFUBZZZ	IJFUBZZZ		
	IJJFCBZD	IJJFCBZD		
	IJJFCIZD			

** Log Print Utility ** (See Figure 2-39)

DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0	DLZLOGP0
	DLZLGPCN			
	DLZLGPMT			
	DLZLPCC0	DLZLPCC0	DLZLPCC0	
	DLZLGPM0	DLZLGPM0	DLZLGPM0	
	IJJFCBID	IJJFCBID		
	IJJFCIID			
	IJFUZZZN	IJFUZZZN		
	IJGUICZZ	IJGUICZZ		

DATA BASE REORGANIZATION UTILITIES

** HS DB Unload ** (See Figure 2-29)

DLZURUL0	DLZURUL0	DLZURUL0	DLZURUL0	DLZURUL0
	DLZRULM0	DLZRULM0	DLZRULM0	
	IJJFCBZD	IJJFCBZD		

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
-------	---------	--------	-------	-----	-----
	IJFVZZWN	IJFVZZWN			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				

** HS DB Reload ** (See Figure 2-30)

DLZURRL0	DLZURRL0	DLZURRL0	DLZURRL0	DLZURRL0	
	DLZRRLM0	DLZRRLM0	DLZRRLM0		
	IJJFCBZD	IJJFCBZD			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJFVZZWN	IJFVZZWN			
	IJFVZZWZ				

** 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

** ACB Creation ** (See Figure 2-33)

DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	DLZUACB0	
	PRTMSG				
	DLZDLBL0	DLZDLBL0	DLZDLBL0	DLZDLBL0	
	PSBPASS				
	DLZDLBL4				
	DLZDLBL1	DLZDLBL1	DLZDLBL1	DLZDLBL1	
	DLZDLBL2	DLZDLBL2	DLZDLBL2	DLZDLBL2	
	DLZDLBL3	DLZDLBL3	DLZDLBL3	DLZDLBL3	
	FREESTOR				
	IJSYSLN				
	PCHDTF				
	DLZLBLM0	DLZLBLM0	DLZLBLM0	DLZLBLM0	
	DLZUSCH0	DLZUSCH0	DLZUSCH0	DLZUSCH0	
	INSRCH				
	CLOSESCH				
	DLZDPSB0	DLZDPSB0	DLZDPSB0	DLZDPSB0	
	IJJCPD1N	IJJCPD1N			

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				

DB LOGICAL RELATIONSHIP UTILITIES

** Prereorganization ** (See Figure 2-34)

DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	DLZURPR0	
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	DLZURGM0	DLZURGM0	DLZURGM0		
	IJJFCBZD	IJJFCBZD			
	IJGFOCZZ	IJGFOCZZ			

** DB Scan ** (See Figure 2-35)

DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	DLZURGS0	
	DLZCONSL				
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJGFICZZ	IJGFICZZ			

** Prefix Resolution ** (See Figure 2-36)

DLZURG10	DLZURG10	DLZURG10	DLZURG10	DLZURG10	
	DLZURGM0	DLZURGM0	DLZURGM0		
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				
	IJGFICZZ	IJGFICZZ			
	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
	IJFVZZWN				
	IJFFZZZN	IJFFZZZN			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				
	DLZX15S1	DLZURG10	DLZURG10		
	DLZX15S2				
	DLZX35S1				
	DLZX35S2				

** Prefix Update ** (See Figure 2-37)

DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	DLZURGP0	
	DLZURGM0	DLZURGM0	DLZURGM0		
	DLZLI000	DLZLI000	DLZLI000	DLZLI000	
	ASMTDLI				
	CBLTDLI				
	IJJFCBZD	IJJFCBZD			
	IJJFCIZD				

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	IJFSZZWN	IJFSZZWN			
	IJFVZZZN				
(DLZURGP0)	IJGQICZZ	IJGQICZZ			
	IJGVICZZ				
** Work File Generator ** (See Figure 2-38)					
DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZDSEH0	DLZBEGIN
	DLZBEGIN				
	OPENWORK				
	IJFSZZWN	IJFSZZWN			
	IJFVZZWN				
	IJGFICZZ	IJGFICZZ			
	IJGQOCZZ	IJGQOCZZ			
	IJGVOCZZ				

DIAGNOSTIC AND TEST MODULES

** System Formatted Dump **				
DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0	DLZFSDP0
		DLZTRPR0		
** 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
** Trace Print Utility ** (see figure 2-41)				
DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0	DLZTPRT0
	DLZTPRM0	DLZTPRM0	DLZTPRM0	

CORE IMAGE LIBRARY	CSECT(S)/ ENTRY POINT(S)	RELO LIBRARY	SOURCE LIBRARY	CORE ID	SUPPL INF
	IJJFCBIC				
	IJJFCIZD	IJJFCIZD			
	IJFVZZZZ	IJFVZZZZ			
	IJGVIEZZ	IJGVIEZZ			
	IJ2M0021	IJ2M0021			

This section describes the major data areas used by DL/I DOS/VS. The description of each data area generally includes:

- Its DSECT name.
- The symbolic names of the fields and flags.
- The displacement of each field, in both decimal and hexadecimal.
- The length of each field.
- An alphabetic listing of all field and flag names (flags are indicated by asterisks).
- The hexadecimal code of each flag.

The data areas are documented in alphabetical order as listed in the Contents of this publication.

This section also describes the DL/I partition in a batch environment and illustrates the relationship of the DL/I control blocks. In addition, the description and general structure is given for the data management block (DMB), the program specification block (PSB), and the DL/I buffer pool control blocks.

THE DL/I PARTITION AND CONTROL BLOCK RELATIONSHIP

The following text describes the DL/I partition in a batch environment and illustrates the relationship of the DL/I control blocks described in this section.

THE DL/I BATCH PARTITION

Figure 5-1 is a map of main storage in the DL/I DOS/VS batch partition. Storage is allocated from the bottom or lowest storage address to the top or highest storage address of the partition. The eight areas in the DL/I batch partition are as follows:

- Area 1 contains the DL/I nucleus. The SCD is the first control block in the nucleus and contains the DL/I copyright information. This block also contains the entry point address for every module in the DL/I system. The PST prefix, PST, and PSB directory (PDIR) are in this area. There is one entry in the PSB directory (PDIR).
- Area 2 contains the DL/I program request handler, DLZPRHB0, which is loaded during DL/I initialization. It is part of the batch nucleus module (DLZBNUC0).
- Area 3 contains the PSB intent list, PSB, and one DMB directory (DDIR) for each DMB referenced by the PSB. The DMB directory is created dynamically during DL/I initialization.
- Area 4 contains DMBs loaded from the DOS/VS Core Image Library by the DL/I Batch Initialization module. Randomizing modules are loaded after the DMBs for HDAM. They are followed by VSAM control blocks, index management modules if secondary indexes are used, and by segment compression modules if variable length segments are used.
- Area 5 contains the DL/I buffer pool control blocks. These blocks are created dynamically. There are one buffer pool prefix, one subpool information table for each subpool specified, one DMB subpool directory entry for each DMB, and 2-32 buffer prefixes for each subpool specified.
- Area 6 contains the DL/I I/O buffers which comprise the buffer pool. There are 2-32 buffers for each subpool specified. Each subpool is aligned on a 2K page boundary.
- Area 7 contains the DL/I action modules and the user trace module if requested.
- Area 8 contains the user batch application program.

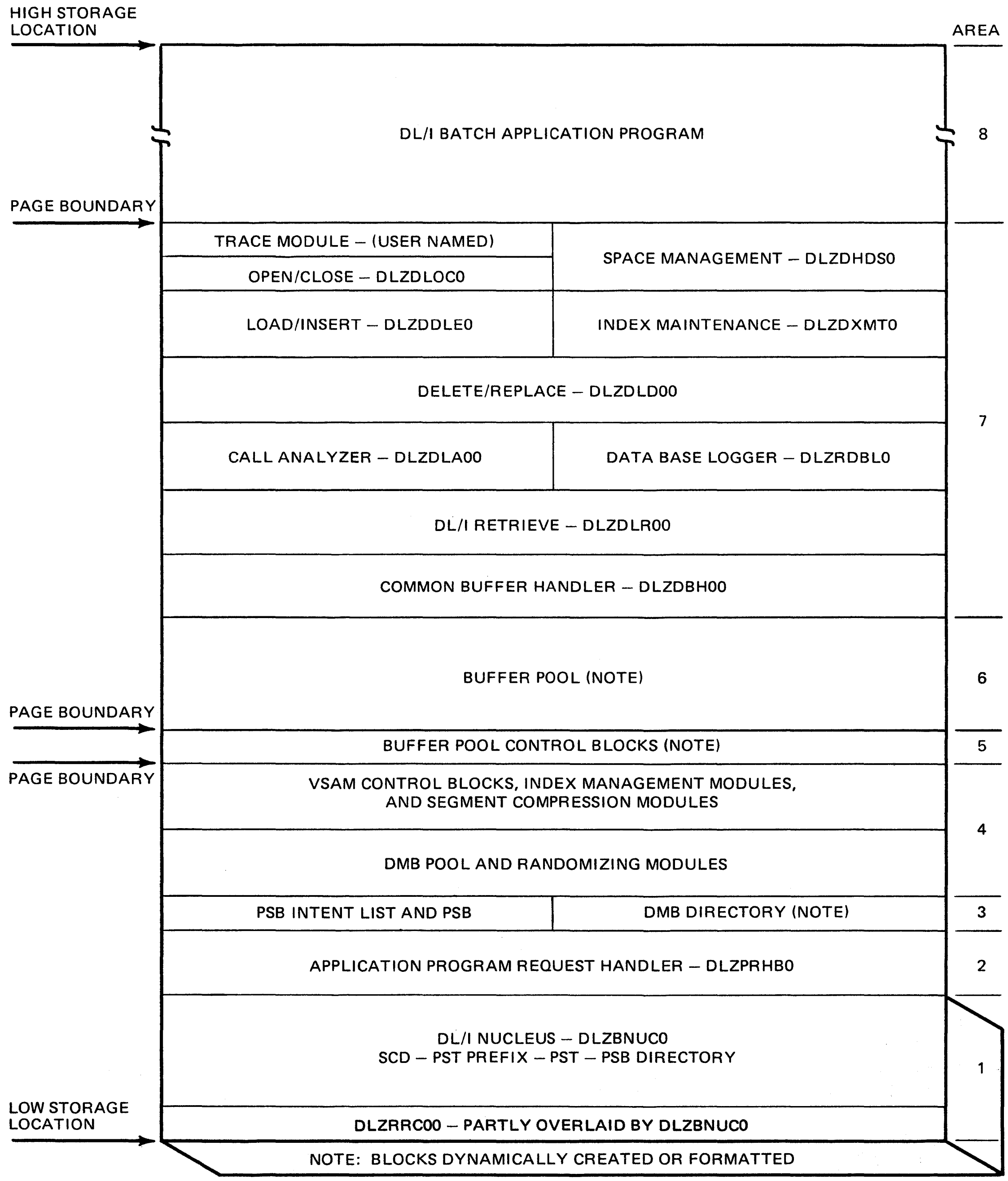


Figure 5-1. Map of Main Storage in the DL/I Batch Partition

DL/I CONTROL BLOCK RELATIONSHIP

The purpose of this section is to show the relationships of the various DL/I control blocks and provide a means by which the user can quickly find his way to these control blocks. The following discussion references Figure 5-2.

The SCD is the major control block in the DL/I system. It is located at the beginning of the DL/I nucleus. The SCD contains DL/I copyright information, entry point addresses of the DL/I logic module, and pointers to the following DL/I control blocks:

- The buffer pool prefix, which is the first block of the buffer pool control blocks.
- The first PSB directory from which the first PSB and PSB intent list may be obtained. In a batch system, there is only one PSB directory.
- The first DMB directory. There is one DMB directory for each DMB referenced by the PCBs.
- The first PST prefix from which the first PST may be obtained. There is only one PST prefix in a batch system.

The PST, including the PST prefix, functionally relates the control blocks for DL/I and represents the batch or CICS/DOS/VS - DL/I online task being served by DL/I. The PST is the dispatching block and is the only parameter passed when calling another module. The address of the PST is contained in the PST prefix. The following pointers are available in the PST:

- Caller's (user program) parameter list
- SCD
- PSB directory for the task
- PCB currently being accessed
- I/O buffer to be used for the data base call (used by the buffer handler)
- Subpool information table assigned to the data base (used by the buffer handler)
- Buffer prefix which points to the I/O buffer containing the segment for the call (used by the buffer handler)

There is one PSB directory entry and one PSB for each program that may be accessed by DL/I. In a CICS/DOS/VS - DL/I online environment, the maximum is 255; in batch, there can be only one. The PSB directory contains address pointers to the PSB and the PSB intent list.

The PSB intent list is a variable-length control block and contains an entry for each DMB referenced by the PSB. Each entry contains the address of the DMB.

The PSB contains prefix information and one or more PCBs. For each PCB there is a JCB, which is made up of the following: JCB prefix, level table, and one or more SDBs. The PCB points to the JCB. The JCB contains working storage for the program's use of that data base and points to the level table. The JCB also points to the SDB for the root segment and the VSAM ACB for the data base (KSDS ACB if HISAM). The level table contains working storage for DL/I to store its positioning data for each level of the data base. The level table points to the current level SDB.

The SDB describes the user's logical use of the sensitive segment. There is one SDB for each segment to which the user is sensitive. Each SDB points to the corresponding PSDB in the DMB.

The DMB directory contains the address of the DMB. Each DMB contains a prefix, one ACB extension for each data set in the DMB (two if HISAM), one PSDB for each physical segment type, and one FDB for each field defined for a segment. In addition, there is one direct algorithm communication table (DMBDACS) if HDAM is used, and secondary list entries if HIDAM or HDAM with index or original relationships is used.

The DMB prefix contains:

- A two-byte relative offset to the first PSDB
- A two-byte relative offset to the end of the last PSDB+1, which is either the first secondary list entry (HIDAM) or the first FDB
- A four-byte pointer to DMBDACS if HDAM

The ACB extension contains information about the data set as well as an address pointer to the VSAM ACB and RPL for the data set.

Each PSDB contains:

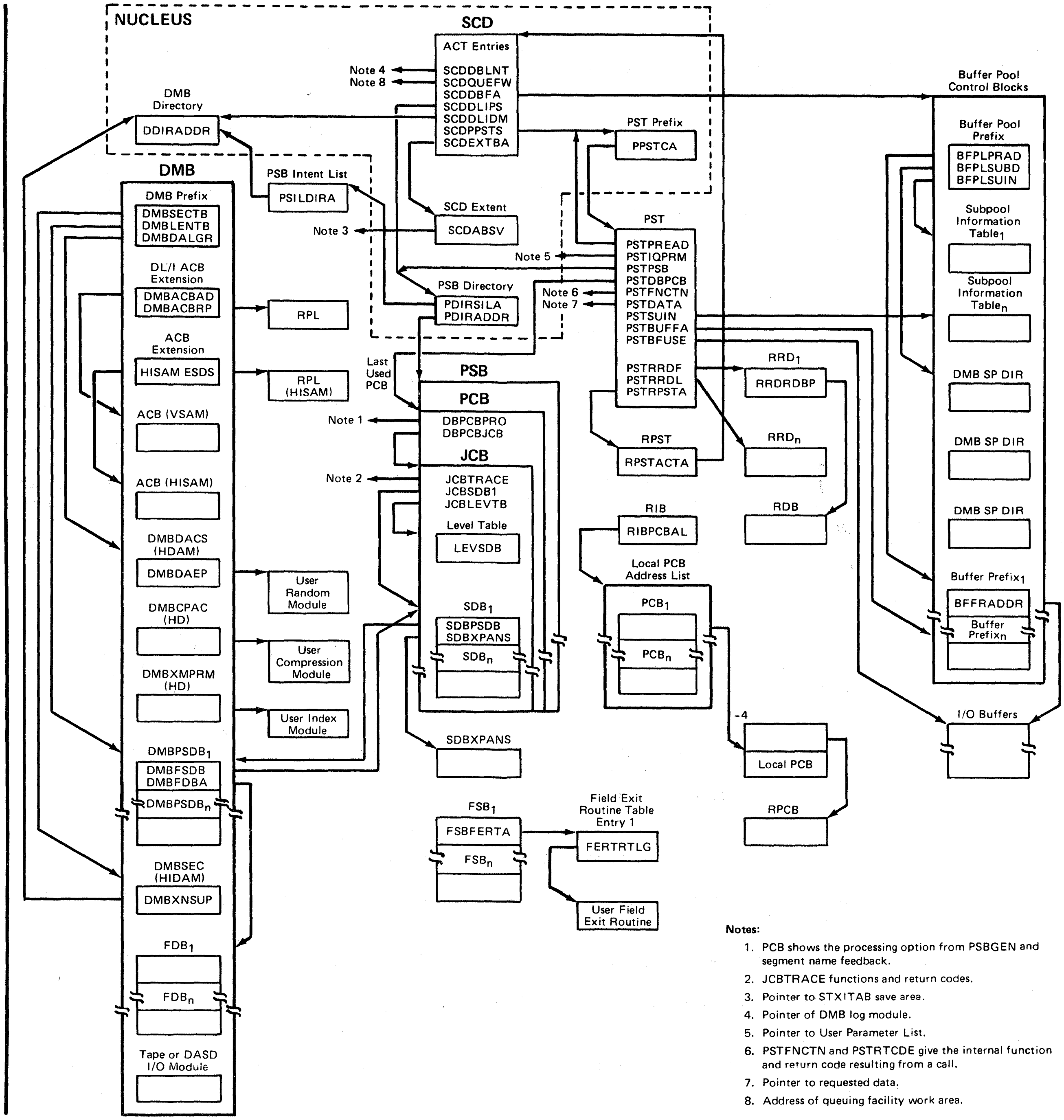
- A pointer to the first FDB for the segment
- A pointer to the SDB for the active PCB which is sensitive to this segment type. If more than one PCB is sensitive to this segment type, the address of the SDB for the next PCB is contained in the active PSDB.

The DMBDACS contains the address of the user's randomizing routine; most of the secondary list entries point to the DMB directory for the described index or logically related data base.

The following items may be obtained from the buffer pool prefix:

- The first subpool information table (immediately following the buffer pool prefix)
- An address pointer to the first buffer prefix
- An address pointer to the first DMB subpool directory entry

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



Notes:

1. PCB shows the processing option from PSBGEN and segment name feedback.
2. JCBTRACE functions and return codes.
3. Pointer to STXITAB 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 data.
8. Address of queuing facility work area.

Figure 5-2. DL/I 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 module (DLZDBLM0).

The DMB consists of the following sections:

- A prefix section containing primarily offsets to subsections of the DMB:
- An ACB extension. For an HISAM organizaton, there is a pair of ACB extensions for each data base; a KSDS ACB and an ESDS ACB. If the data base contains only root segments, only the KSDS ACB extension is created. The ACBs are generated only when the blocks are loaded for execution by DLZDBLM0 from the information in the ACB extensions.
- A DTF extension if SHSAM or HSAM for input and output file.
- A direct algorithm communication table if HDAM.
- A compression section for each compressable segment.
- An index maintenance parameter section for each secondary exit routine.
- A physical segment description block.
- A secondary list to describe indexed fields or logical relationships.
- Field description blocks describing each field in each segment.
- A tape or DASD I/O module if SHSAM or HSAM. This module is included by the ACB utility.

GENERAL STRUCTURE

The general structure of the DMB is shown in Figure 5-3.

Each DMB section is shown as a separate data area in Section 5 of this PLM, For the data area layout, see:

DMB PREFIX	DSECT Name: DMB	DMB	— DMB Prefix
ACB EXTENSION	DSECT Name: DMBACBXT	ACB	— ACB Extension
DTF EXTENSION	DSECT Name: DMBDTFXT		
DIRECT ALGORITHM COMMUNICATION TABLE	DSECT Name: DMBDACS	DACS	— HDAM Randomizing Routine Interface Table
COMPRESSION SECTION	DSECT Name: DMBCPAC	CPAC	— HDAM/HIDAM Variable Length Segment Compression/Expansion Routine
INDEX MAINTENANCE PARAMETERS	DSECT Name: DMBXMPRM	XMPRM	— HDAM/HIDAM User Secondary Index Suppression Routine Interface Table
PHYSICAL SEGMENT DESCRIPTION BLOCK	DSECT Name: DMBPSDB	PSDB	— Physical Segment Description Block
SECONDARY LIST	DSECT Name: DMBSEC	SEC	— Secondary List
FIELD DESCRIPTION BLOCK	DSECT Name: FDB	FDB	— Field Description Block
Tape or DASD I/O Module			

Figure 5-3. General Structure of DMB

PROGRAM SPECIFICATION BLOCK - PSB

A PSB must be created for every user program which will run under DL/I control. The PSB is created in "skeleton" format (principally PCBs only) by PSBGEN. The PSB must be cataloged and link edited into the Core Image Library. The PSB is loaded into main storage by the DL/I Application Control Blocks Creation and Maintenance Utility program and expanded and completed by this utility. The expansion is performed by segment definition in the DBD representing the associated data base. The expanded PSB is link edited into the Core Image Library. The PSB is moved to its execution-time location in the PSB pool by the application control blocks load and relocate module (DLZDBLM0). In expanded final format, the PSB consists of the following parts in the order specified:

1. PSB prefix - of which the most important part is the variable-length PSB list: the address list of the PCBs in the PSB. A dope vector table follows the PSB prefix for PL/I programs.

2. A variable number of data base PCBs. For each data base PCB there is a JCB (job control block) consisting of the following parts:

   - JCB prefix

   - DSG (data set group) table. This table contains entries describing the data bases specifically used for this PCB. There are entries for all logically connected data bases, all primary HIDAM indexes, and a secondary index if used as the processing sequence.

   - Level table. This table provides memory of the last DL/I CALL.

   - SDB (segment description block). This block contains an entry for each segment to which the user has declared himself sensitive in the PCB. The SDB entry describes the sensitive segment.

   - Work area for index maintenance, variable-length segment support, or miscellaneous function. These are allocated only when required (if any user PCB directly or indirectly refers to an index data base).

   - PSB work areas; of variable length depending on the requirements of the PCBs.

GENERAL STRUCTURE

The general structure of the PSB is shown in Figure 5-4.

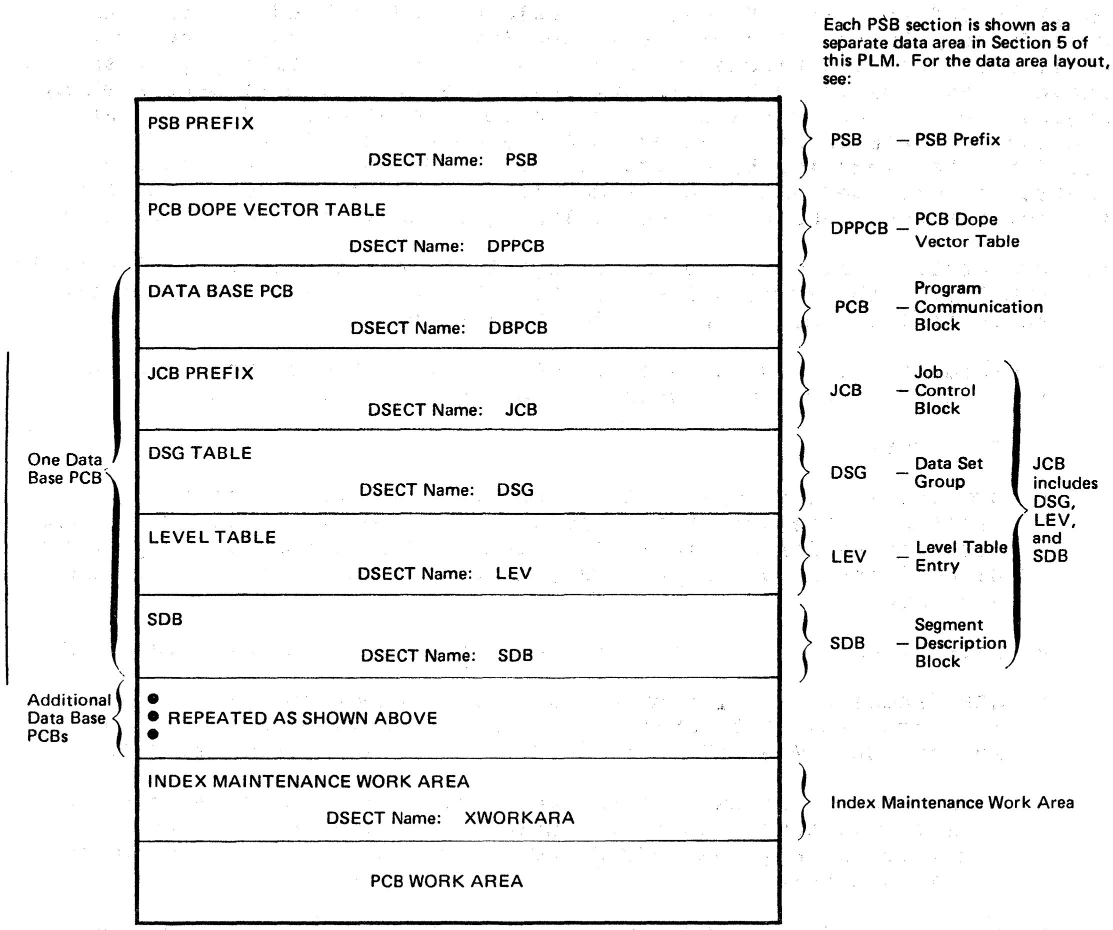


Figure 5-4. General Structure of PSB.

DL/I BUFFER POOL CONTROL BLOCKS

The DL/I buffer pool control blocks provide the control information to manage the entire buffer pool for the DL/I task. The buffer pool control blocks are as follows:

- Buffer Pool Control Block Prefix - This control block contains the statistics and other control information for the entire buffer pool.
- Subpool Information Table - This control block contains information for a specific subpool, including the size of the buffers in the subpool. There is one subpool information table for each subpool allocated.
- DMB Subpool Directory - This control block contains a one-byte subpool number relative to zero for each HDAM or HIDAM data base allocated. The DMB sequence number is used as an offset 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 5-5.

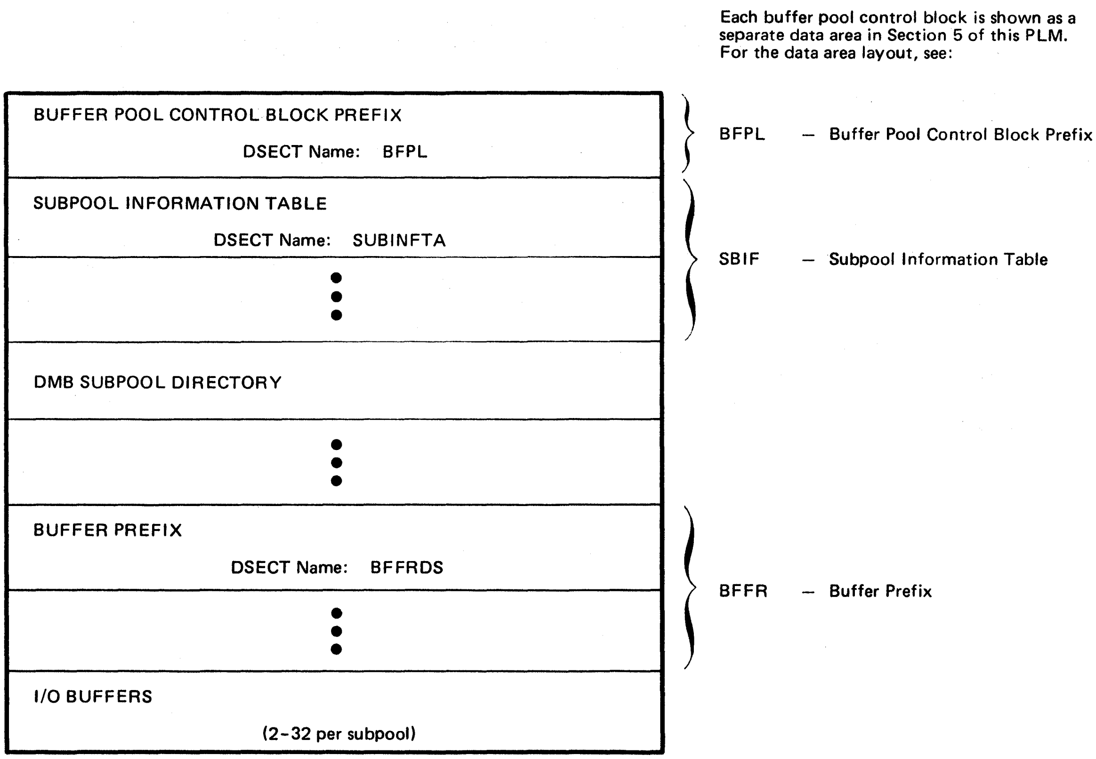


Figure 5-5. General Structure of DL/I Buffer Pool Control Blocks

ACBXT - ACB EXTENSION

DSECT Name: DMBACBXT

The ACB extension is described as part of the general structure and description of the data management block (DMB). The information in ACBXT is repeated for each data set in the DMB.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBACBAD	0(00)	
DMBACBAP	7(07)	
DMBACBDL	6(06)	
DMBACBEX	68(44)	
DMBACBLC	56(38)	
DMBACBLN	80(50)	
DMBACBMN	10(0A)	
DMBACBMX	8(08)	
DMBACBND	80(50)	
DMBACBNM	60(3C)	
DMBACBRP	52(34)	
DMBACBST	0(00)	
DMBACLNO	60(3C)	
*DMBBESDS	46(2E)	40
DMBBFACT	44(2C)	
DMBCICYL	28(1C)	
DMBCINV	4(04)	
*DMBCISPL	35(23)	80
DMBCITRK	30(1E)	
DMBDTFIN	0(00)	(See DTF extension at end of ACBXT)
DMBDTFOT	4(04)	(See DTF extension at end of ACBXT)
DMBECB	12(0C)	
DMBFBASN	72(48)	
*DMBFBA	46(2E)	20
DMBFRSPC	58(3A)	
DMBFRSP1	59(3B)	
DMBHIBLK	16(10)	
DMBHIRBA	36(24)	
DMBINDO	46(2E)	
*DMBIGNOR	34(22)	40
*DMBKEY	46(2E)	80
DMBKEYLE	31(1F)	
DMBLRECL	42(2A)	
*DMBNUSE	34(22)	20
DMBOFLGS	34(22)	
*DMBOPEN	34(22)	10
*DMBPSEQ	35(23)	10
*DMBPUTKY	34(22)	08
DMBRBASN	20(14)	
DMBRKP	32(20)	
DMBRLBLK	24(18)	
DMBSPLCT	48(30)	
DMBVSBFR	40(28)	
DMBVSFLG	35(23)	
*DMBWCHK	46(2E)	08

RECORD LAYOUT - ACBXT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	DMBACBST		Start of ACB extension
0(00)	4	DMBACBAD		Address of corresponding ACB
4(04)	2	DMBCINV		Control interval size
6(06)	1	DMBACBDL		Delta cylinders to scan
7(07)	1	DMBACBAP		Number of root anchor points per control interval (HDAM)
8(08)	2	DMBACBMX		Length of the largest segment in data set
10(0A)	2	DMBACBMN		Length of the smallest segment in data set
12(0C)	4	DMBECB		VSAM ACB event control block (ECB) used by buffer handler (DLZDBH00)
16(10)	4	DMBHIBLK		Highest control interval RBA
20(14)	4	DMBRBASN		RBA of last logical record assigned (HISAM) or relative block number of last control interval assigned (HD). During batch initialization the high-order byte is the buffer size (control interval size/512) indicator
24(18)	4	DMBRLBLK		Relative block number of last control interval written (HD)
28(1C)	2	DMBCICYL		Number of control intervals per cylinder
30(1E)	1	DMBCITRK		Number of control intervals per track
31(1F)	1	DMBKEYLE		Key length of KSDS
32(20)	2	DMBRKP		Relative key position
34(22)	1	DMBOFLGS		Open flags
		DMBIGNOR	40	IGN was specified for workfile on load
		DMBNUSE	20	ACB does not have resolved secondary index entries; workfile must be used
		DMBOPEN	10	The corresponding ACB is open
		DMBPUTKY	08	Simulate not load mode to VSAM
35(23)	1	DMBVSFLG		Flags
		DMBCISPL	80	Control interval split occurred

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		DMBPSEQ	10	Sequential processing is possible for this KSDS
36(24)	4	DMBHIRBA		Highest RBA in present range of extents (HIDAM ESDS only)
40(28)	2	DMBVSBFR		Number of buffers to be used
42(2A)	2	DMBLRECL		Logical record length
44(2C)	2	DMBBFACT		Blocking factor
46(2E)	1	DMBIND0		Permanent indicators
		DMBWCHK	08	Write check option
		DMBFBA	20	FBA device suport
		DMBBESDS	40	Blocked ESDS
		DMBKEY	80	Data set contains keys (Simple HISAM and SHISAM)
47(2F)	1			**Reserved**
48(30)	4	DMBSPLCT		Control interval split count
52(34)	4	DMBACBRP		Address of RPL for this ACB
56(38)	2	DMBACBLC		Log count (HISAM only)
58(3A)	1	DMBFRSPC		Distributed free space parameter
59(3B)	1	DMBFRSP1		Second free space parameter
60(3C)	8	DMBACBNM		Data set name as in ACB
		DMBACLN0		Length of version 1.0
68(44)	4	DMBACBEX		Address of exit list for this ACB
72(48)	2	DMBFBASN		FBA scan value
74(4A)	6			**Reserved**
80(50)	2	DMBACBND		End of ACB extension
		DBMACBLN		Length of ACB extension (DMBACBND minus DMBACBST)

Note: HSAM DMBs have the following DTF extension.

DSECT Name: DMBDTFXT

0(00)	4	DMBDTFIN		Address of HSAM input DTF
4(04)	4	DMBDTFOT		Address of HSAM output DTF

BFFR - BUFFER PREFIX

DSECT Name: BFFRDS

The buffer prefix is described as part of the general structure and description of the DL/I buffer pool control blocks. There is one buffer prefix for each buffer allocated.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
BFFRADDR	12(0C)	
BFFRCIID	0(00)	
BFFRCIRB	0(00)	
BFFRDCB	6(06)	
BFFRDMB	4(04)	
*BFFREXNQ	7(07)	02
BFFRHOLE	30(1E)	
*BFFRLAST	27(1B)	01
BFFRLEN	32(20)	
*BFFRLOCK	27(1B)	40
BFFRLOCU	10(0A)	
*BFFRMT	7(07)	10
BFFRNACB	26(1A)	
BFFRNCII	20(14)	
BFFRNCID	20(14)	
BFFRNDMB	24(18)	
*BFFRNORU	27(1B)	80
BFFRNPSF	28(1C)	
BFFRNPSL	29(1D)	
BFFRNPST	28(1C)	
*BFFRPNNQ	7(07)	01
*BFFRPRED	7(07)	08
BFFRPST	8(08)	
BFFRPSTF	8(08)	
BFFRPSTL	9(09)	
*BFFRREAD	7(07)	20
*BFFRRREL	27(1B)	08
BFFRSW	7(07)	
BFFRSW1	27(1B)	
BFFRUSCT	12(0C)	
BFFRUSID	16(10)	
BFFRWCBW	19(13)	
BFFRWCFW	18(12)	
*BFFRWCH	7(07)	80
*BFFRWERR	7(07)	04
*BFFRWRT	7(07)	40

RECORD LAYOUT - BFFR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	7	BFFRCIID		Control Interval identifier
0(00)	4	BFFRCIRB		Control Interval RBA
4(04)	2	BFFRDMB		DMB Number
6(06)	1	BFFRDCB		ACB Number
7(07)	1	BFFRSW		Flags
		BFFRWCH	80	Buffer on write chain
		BFFRWRT	40	Buffer being written
		BFFRREAD	20	Buffer being read
		BFFRMT	10	Buffer empty
		BFFRPRED	08	Buffer waiting for predecessor being written
		BFFRWERR	04	Buffer has permanent write error
		BFFREXNQ	02	Existing CI ID enqueued
		BFFRPNNQ	01	Pending CI ID enqueued
8(08)	2	BFFRPST		PST prefix numbers for enqueue/dequeue
8(08)	1	BFFRPSTF		PST prefix number of the controlling task
9(09)	1	BFFRPSTL		PST prefix number of the last task in the chain of waiting tasks
10(0A)	2	BFFRLOCU		Log count
12(0C)	1	BFFRUSCT		Use count
12(0C)	4	BFFRADDR		Address of buffer
16(10)	2	BFFRUSID		ID of the users who altered this buffer
18(12)	1	BFFRWCFW		Next lower buffer on the write chain
19(13)	1	BFFRWCBW		Next higher buffer on the write chain
20(14)	7	BFFRNCID		New control interval identifier
20(14)	4	BFFRNCII		New control interval RBA
24(18)	2	BFFRNDMB		New DMB number
26(1A)	1	BFFRNACB		New ACB number

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
27(1B)	1	BFFRSW1		Flags
		BFFRNORU	80	Buffer is not reusable
		BFFRLOCK	40	Buffer locked by logger
		BFFRREL	08	Buffer is released
		BFFRLAST	01	Last buffer prefix for this subpool
28(1C)	2	BFFRNPST		PST prefix numbers for enqueue/dequeue
28(1C)	1	BFFRNPSF		PST prefix number of task which enqueued on new CI ID and is first in the chain
29(1D)	1	BFFRNPSL		PST prefix number of task which enqueued on new CI ID and is last in the chain
30(1E)	2	BFFRHOLE		Length of largest space available in the buffer
32(20)		BFFRLEN		Length of buffer prefix

BFPL - BUFFER POOL CONTROL BLOCK PREFIX

DSECT Name: 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.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
BFPLALTR	28(1C)	
BFPLBKWT	36(24)	
BFPLCHBK	48(30)	
BFPLCHWT	44(2C)	
BFPLCOUT	62(3E)	
*BFPLEXCI	64(40)	00
BFPLID	0(00)	
BFPLIGET	56(38)	
BFPLINCO	96(60)	
BFPLINMA	72(48)	
BFPLINPL	20(14)	
BFPLINRO	88(58)	
BFPLINW1	88(58)	
BFPLINW2	104(68)	
BFPLISTL	52(34)	
BFPLLEN	140(8C)	
BFPLNQW1	64(40)	
BFPLNQW2	68(44)	
BFPLNWBK	40(28)	
BFPLOSWT	32(20)	
*BFPLPECI	64(40)	04
BFPLPRAD	128(80)	
BFPLPSIF	124(7C)	
BFPLPSIL	125(7D)	
BFPLPSI1	120(78)	
BFPLRDCT	24(18)	
BFPLROCO	63(3F)	
BFPLRQCT	16(10)	
BFPLSUBD	132(84)	
BFPLSUIN	136(88)	
*BFPLSUPO	64(40)	08
*BFPLSW00	68(44)	00
*BFPLSW80	68(44)	80
BFPLWERR	60(3C)	
BFPLWERT	61(3D)	

RECORD LAYOUT - BFPL

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	BFPLID		Buffer pool control block ID (BFPL)
4(04)	12			*Reserved*
16(10)	4	BFPLRQCT		Number of requests received by the buffer handler
20(14)	4	BFPLINPL		Number of requests satisfied from buffer pool
24(18)	4	BFPLRDCT		Number of read requests issued
28(1C)	4	BFPLALTR		Number of buffer alter requests received
32(20)	4	BFPLOSWT		Number of writes issued
36(24)	4	BFPLBKWT		Number of blocks written
40(28)	4	BFPLNWBK		Number of new blocks created in pool
44(2C)	4	BFPLCHWT		Number of chained writes issued
48(30)	4	BFPLCHBK		Number of blocks written on write chain
52(34)	4	BFPLISTL		Number of retrieves by key calls
56(38)	4	BFPLIGET		Number of GN calls received
60(3C)	1	BFPLWERR		Number of permanent write error buffers in pool
61(3D)	1	BFPLWERT		Largest number of write error buffers ever in pool
62(3E)	1	BFPLCOUT		Number of rows/columns in matrix currently in use
63(3F)	1	BFPLROCO		Mask showing available rows/columns in matrix
64(40)	4	BFPLNQW1		ENQ/DEQ workarea 1. Byte 0 indicates the following:
		BFPLEXCI	00	ENQ/DEQ existing CI code
		BFPLPECI	04	ENQ/DEQ pending CI code
		BFPLSUPO	08	ENQ/DEQ subpool code
				Bytes 1-3 contain a pointer to the PST prefix numbers of the first and last task waiting for the resource

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
68(44)	4	BFPLNQW2		ENQ/DEQ workarea 2
		BFPLSW00	00	Mask to turn off wait switch
		BFPLSW80	80	Task waiting for matrix space
72(48)	16	BFPLINMA		Interlock detection matrix
88(58)	16	BFPLINW1		Interlock detection workarea 1
88(58)	8	BFPLINRO		
96(60)	8	BFPLINCO		
104(68)	16	BFPLINW2		Interlock detection workarea 2
120(78)	4	BFPLPSI1		Pointer to the PST prefix numbers of the first and last task waiting for matrix space
124(7C)	1	BFPLPSIF		PST prefix number of the first task waiting for matrix space
125(7D)	1	BFPLPSIL		PST prefix number of the last task waiting for matrix space
126(7E)	2			*Reserved*
128(80)	4	BFPLPRAD		Beginning address of the buffer prefix area
132(84)	4	BFPLSUBD		Beginning address of the DMB subpool directory
136(88)	4	BFPLSUIN		Beginning of the subpool information table entries
140(8C)		BFPLLEN		Length of the buffer pool control block prefix

CPAC - HDAM/HIDAM VARIABLE LENGTH SEGMENT COMPRESSION/EXPANSION ROUTINE INTERFACE TABLE

DSECT Name: DMBCPAC

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

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBCPCNM	0(00)	
DMBCPCSG	8(08)	
DMBCPEP	16(10)	
DMBCPFLG	20(14)	
*DMBCPKEY	20(14)	02
DMBCPLNG	26(1A)	
*DMBCPNIT	20(14)	01
DMBCPRES	28(1C)	
*DMBCPSEQ	20(14)	08
DMBCPSGL	24(18)	
DMBCPSQF	21(15)	
DMBCPSQL	22(16)	
*DMBCPVLR	20(14)	04

RECORD LAYOUT - CPAC

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBCPCNM		Segment Name
8(08)	4	DMBCPCSG		Compression routine name
16(10)	4	DMBCPEP		Entry point of compression routine
20(14)	1	DMBCPFLG		Flag byte
		DMBCPSEQ	08	Segment has a sequence field defined
		DMBCPVLR	04	Segment is variable length
		DMBCPKEY	02	Segment has key compression option
		DMBCPNIT	01	Initialization and termination processing required
21(15)	1	DMBCPSQF		Length of key field minus 1
22(16)	2	DMBCPSQL		Offset to sequence field
24(18)	2	DMBCPSGL		Maximum segment length
26(1A)	2	DMBCPLNG		Total length of CSECT - fixed lengths, constants, plus user data
28(1C)	4	DMBCPRES		Reserved for initialization

DACS - HDAM RANDOMIZING ROUTINE INTERFACE TABLE

DSECT Name: DMBDACS

The HDAM randomizing routine interface table is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBDABLK	16(10)	
DMBDABYC	24(18)	
DMBDABYM	20(14)	
DMBDACP	28(1C)	
DMBDAEP	9(09)	
DMBDAKL	8(08)	
DMBDANME	0(00)	
DMBDARAP	14(0E)	
DMBDASZE	12(0C)	

RECORD LAYOUT - DACS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBDANME		Name of address conversion algorithm load module
8(08)	1	DMBDAKL		Root Key length minus 1
9(09)	3	DMBDAEP		Entry point to conversion module
12(0C)	2	DMBDASZE		Size of this DSECT
14(0E)	2	DMBDARAP		Number of root anchor pointers per block
16(10)	4	DMBDABLK		Number of highest block directly addressable
20(14)	4	DMBDABYM		Maximum number of bytes per root before overflow outside of directly addressable area
24(18)	4	DMBDABYC		Current number of bytes consecutively inserted or loaded under root
28(1C)	4	DMBDACP		Result of last address conversion

DDIR - DMB DIRECTORY

DSECT Name: DLZDDIR

The DMB directory contains an entry for every DMB (data management block) that can be accessed under DL/I control. The DMB directory is part of the DL/I nucleus and is created during DL/I system definition for online processing. The start address of the directory (SCDDLIDM) and entry length (SCDDLIDL) are contained in the system contents directory (SCD).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DDIRADDR	8(08)	
*DDIRBAD	19(13)	01
DDIRCNT	12(0C)	
DDIRCODE	18(12)	
DDIRCOD2	19(13)	
DDIRDMBL	13(0D)	
*DDIREXCL	19(13)	10
*DDIREXSD	19(13)	08
*DDIRGRP	19(13)	02
*DDIRHSAM	19(13)	20
*DDIRINOP	18(12)	20
*DDIRKBRQ	18(12)	10
DDIRLEN	24(18)	
*DDIRNDBM	19(13)	80
*DDIRNOUP	18(12)	01
*DDIRNOSC	18(12)	04
*DDIRNRAN	19(13)	40
DDIRNUMB	16(10)	
*DDIROPEN	18(12)	40
DDIRPPST	21(15)	
*DDIRSECL	18(12)	80
DDIRSYM	0(00)	
DDIRVSRT	20(14)	
*DDIRWAIT	18(12)	08
*DDIR1GRP	19(13)	04

RECORD LAYOUT - DDIR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	DDIR		Label to establish entry address
0(00)	8	DDIRSYM		DMB name - converted from DMBNAME supplied during DBDGEN
8(08)	4	DDIRADDR		DMB address
12(0C)	1	DDIRCNT		Number of users scheduled to this DMB
13(0D)	3	DDIRDMBL		Storage required for this DMB
16(10)	2	DDIRNUMB		DMB number of this DMB
18(12)	1	DDIRCODE		DMB code
		DDIRSECL	80	Security locked
		DDIROPEN	40	At least one ACB is opened
		DDIRINOP	20	DMB to be opened during online initialization or during start call
		DDIRKBRQ	10	Buffer pool space required for this KSDS
		DDIRWAIT	08	System task waiting for zero DDIRCNT
		DDIRNOSC	04	Do not schedule this DMB because it is stopped
		DDIRNOUP	01	Do not schedule updates
19(13)	1	DDIRCOD2		DMB code byte 2
		DDIRNDBM	80	DMB not present in library
		DDIRNRAN	40	Requested randomizing module not present in library
		DDIRHSAM	20	This DMB for HSAM
		DDIREXCL	10	This DMB being used exclusively
		DDIREXSD	08	Exclusive control required for scheduling
		DDIR1GRP	04	DMB first in shared index
		DDIRGRP	02	DMB belongs to shared index
		DDIRBAD	01	DMB initialization failed
20(14)	1	DDIRVSRT		R15 VSAM return code
21(15)	3	DDIRPPST		PPST address if DMB is used exclusively
24(18)		DDIRLEN		Length of one DDIR entry

DLZTWAB - TRANSACTION WORK AREA

DSECT Name: DLZTWA

The DLZTWA macro provides the mapping for the batch partition controller's transaction work area. The information is used for communication with:

- DL/I task termination
- CICS/VS
- Batch partition
- Sheduling MPS batch jobs
- Online message module

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
TWABEND	202(CA)	
TWABPC	0(00)	
TWABPCID	4(04)	
*TWABCOK	0(00)	80
TWABPCSV	76(4C)	
TWACALL	40(28)	
TWACCND	192(C0)	
*TWAEOJSW	0(00)	40
TWAMPCE	5(05)	
TWAMPCPT	1(01)	
TWAMPSFG	0(00)	
TWAMPSID	180(B4)	
TWAMSG	148(94)	
TWAMSGID	152(98)	
TWAMSGNO	148(94)	
TWAMSG01	156(9C)	
TWAMSG02	160(A0)	
TWAMSG03	164(A4)	
TWAMSG04	168(A8)	
TWAN1PTR	32(20)	
TWAPARMC	36(24)	
TWAPIDTB	56(38)	
TWAPSBDL	55(37)	
TWAPSBN	44(2C)	
TWAPSBNM	48(30)	
TWAPSW	172(AC)	
TWAPTEND	68(44)	
TWAPTIDF	58(3A)	
TWARCODE	190(BE)	
TWASCHDC	36(24)	
TWAWLIST	8(08)	
TWAXCBDL	16(10)	
TWAXCBN1	24(18)	
TWAXCBN2	20(14)	
TWAXCB2	8(08)	
TWAXCB3	12(0C)	
TWAXNAME	182(B6)	

RECORD LAYOUT - DLZTWAB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
THE FOLLOWING FIELDS ARE USED FOR COMMUNICATING WITH THE DL/I TASK TERMINATION ROUTINE				
0(00)	0	TWABPC		Start of TWABPC
0(00)	1	TWAMPSFG		BPC flag byte:
		TWABPCCK	80	BPC abnormal termination processing completed
		TWAEOJSW	40	EOJ processing reached for MPS batch partition
1(01)	3	TWAMPCPT		Address of MPC partition table
4(04)	1	TWABPCID		Batch partition identifier (F1,F2,...)
5(05)	3	TWAMPCE		Address of specific MPC partition table entry
THE FOLLOWING IS THE BATCH PARTITION CONTROLLER'S CICS/VS WAITM ECB LIST, DELIMITER, AND XECB				
8(08)	0	TWAWLIST		Start of TWAWLIST
8(08)	4	TWAXCB2		Pointer to BPC's XECB (DLZXCBn2)
12(0C)	4	TWAXCB3		Pointer to ABEND XECB (DLZXCBn3)
16(10)	4	TWAXCBDL		ECB list delimiter ('FFFFFFFF')
20(14)	4	TWAXCBN2		XECB for BPC
THE FOLLOWING FIELDS ARE USED FOR COMMUNICATION WITH THE BATCH PARTITION				
24(18)	8	TWAXCBN1		XECB name for batch initialization (DLZXCBn1)
32(20)	4	TWAN1PTR		XECBTAB table entry address for batch initialization's XECB (DLZXCBn1)
THE FOLLOWING FIELDS ARE USED FOR THE BATCH PARTITION CONTROLLER'S DL/I SCHEDULING CALL PARAMETER LIST AND THE PSBNAME TO BE SCHEDULED				
36(24)	0	TWASCHDC		Start of TWASCHDC
36(24)	4	TWAPARMC		Pointer to parameter count
40(28)	4	TWACALL		Pointer to call function
44(2C)	4	TWAPSBN		Pointer to PSB name
48(30)	7	TWAPSBNM		PSB name (PSBNAME)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
55(37)	1	TWAPSBDL		PSB name delimiter

THE FOLLOWING TABLE CONTAINS THE PARTITION IDENTIFIERS USED TO IDENTIFY THE BATCH PARTITION ASSOCIATED WITH MESSAGES DLZ082I, DLZ084I, AND DLZ103I

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
56(38)	20	TWAPIDTB		Start of TWAPIDTB
56(38)	2	Unnamed		Background partition ID (BG)
58(3A)	2	TWAPTIDF		Foreground partition ID (F6)
60(3C)	2	Unnamed		Foreground partition ID (F5)
62(3E)	2	Unnamed		Foreground partition ID (F4)
64(40)	2	Unnamed		Foreground partition ID (F3)
66(42)	2	Unnamed		Foreground partition ID (F2)
68(44)	2	TWAPTEND		Foreground partition ID (F1)
70(46)	2	Unnamed		**Reserved**
72(48)	2	Unnamed		**Reserved**
74(4A)	2	Unnamed		**Reserved**

BATCH PARTITION CONTROLLER REGISTER SAVE AREA

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
76(4C)	72	TWABPCSV		BPC register save area (18 fullwords)

THE FOLLOWING ARE THE PARAMETER LIST POINTERS, PARAMETERS, AND MESSAGE FILLERS PASSED TO THE DL/I ONLINE MESSAGE MODULE (DLZERMSG) FOR ALL BPC MESSAGES

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
148(94)	0	TWAMSG		Start of TWAMSG
148(94)	4	TWAMSGNO		Message number pointer for all BPC messages
152(98)	4	TWAMSGID		Partition ID pointer (for messages DLZ082I, DLZ084I, and DLZ103I)
				BPC module ID pointer (for message DLZ104I)
156(9C)	4	TWAMSG01		Module name pointer (for messages DLZ082I and DLZ084I)
				Termination condition pointer and delimiter (for message DLZ103I)
				CICS ABEND code pointer and delimiter (for message DLZ104I)
160(A0)	4	TWAMSG02		XECBTAB TYPE= pointer (for messages DLZ082I and DLZ084I)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				PSW pointer and delimiter (for message DLZ104I)
164(A4)	4	TWAMSG03		XECBTAB XECB=XECBname pointer (for messages DLZ082I and DLZ084I)
168(A8)	4	TWAMSG04		Return code pointer and delimiter (for messages DLZ082I and DLZ084I)
172(AC)	8	TWAPSW		Program interrupt PSW
180(B4)	2	TWAMPSID		Batch partitiion ID
182(B6)	8	TWAXNAME		XECBTAB XECB=XECBname (DLZXCBnn)
190(BE)	2	TWARCODE		Return code
192(C0)	10	TWACOND		BPC termination condition (abnormally or normally)
202(CA)	4	TWABEND		CICS ABEND completion code (ASRA)

DMB - DMB PREFIX

DSECT Name: DMB

The DMB prefix is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBDALGR	12(0C)	
*DMBHD	6(06)	06
*DMBHI	6(06)	07
*DMBHSAM	6(06)	05
*DMBISAM1	6(06)	02
DMBLDDCB	7(07)	
DMBLENTB	2(02)	
*DMBNDEX	6(06)	08
DMBNREF	12(0C)	
DMBORG	6(06)	
DMBPDATA	8(08)	
DMBPPRLN	16(10)	
DMBPPRND	16(10)	
DMBSECTB	4(04)	
*DMBSHIS	6(06)	01
DMBSIZE	0(00)	
*DMBSSAM	6(06)	04
*DMBV11	0(00)	80

RECORD LAYOUT - DMB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	2	DMBSIZE		DMB size
		DMBV11	80	DL/I version 1.1 or later
2(02)	2	DMBLENTB		Offset from DMB to first PSDB (DMBPSDB)
4(04)	2	DMBSECTB		Offset from DMB to end of PSDBs + 1
6(06)	1	DMBORG		DMB organization
		DMBSHIS	01	Simple HISAM
		DMBISAM1	02	HISAM
		DMBSSAM	04	Simple HSAM
		DMBHSAM	05	HSAM
		DMBHD	06	HDAM
		DMBHI	07	HIDAM
		DMBNDEX	08	Index data base
7(07)	1	DMBLDDCB		ACB number (minus 1) of sequential data set used to write index records on data base load
8(08)	2	DMBPDATA		Length of system data in index data base (protected)
10(0A)	2			*Reserved*
12(0C)	1	DMBNREF		Number of entries in external reference table
12(0C)	4	DMBDALGR		Address of direct algorithm communication table if HDAM (DMBDACS); LRECL number if HSAM
16(10)		DMBPPRND		End + 1 of DMB prefix. This is also the address of the first ACB extension (DMBACBXT)
16(10)		DMBPPRLN		Length of DMB prefix (DMBPPRND minus DMB)

DPPCB - PCB DOPE VECTOR TABLE

DSECT Name: DPPCB

The PCB dope vector table is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DPPCBDBD	0(00)	
DPPCBJCB	32(20)	
DPPCBKFD	52(34)	
DPPCBLEV	8(08)	
DPPCBLKY	44(2C)	
DPPCBPRO	28(18)	
DPPCBSFD	36(24)	
DPPCBSTC	16(10)	
DPPCPNSS	48(30)	

RECORD LAYOUT - DPPCB

Offset Dec(Hex)	Length	Field Name	Meaning
0(00)	4	DPPCBDBD	The address of the location that contains DBPCBDBD
4(04)	2	Maximum Length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
6(06)	2	Current length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
8(08)	4	DPPCBLEV	The address of the location that contains DBPCBLEV
12(0C)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
14(0E)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
16(10)	4	DPPCBSTC	The address of the location that contains DBPCBSTC
20(14)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
22(16)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string
24(18)	4	DPPCBPRO	The address of the location that contains DBPCBPRO
28(1C)	2	Maximum length	Maximum length: Halfword binary number which specifies number of storage units allocated for the string; byte count if character, bit count if bit
30(1E)	2	Current Length	Current length: Halfword binary number which specifies the number of storage units, within the maximum length, currently occupied by the string

Offset Dec(Hex)	Length	Field Name	Meaning
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

DSG - DATA SET GROUP

DSECT Name: DSG

The DSG is described as part of the general structure and description of the program specification block (PSB).

Note: With the exception of the first three characters of each field/flag name (DSG instead of JCB) the layout of the data set group is identical to the layout of the 'DSG Section' of the job control block (JCB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*DSGBLDEL	15(0F)	80
DSGBOFF	12(0C)	
*DSGCOMMD	16(10)	02
*DSGCONST	15(0F)	20
*DSGDATX	16(10)	40
DSGDCBA	0(00)	
DSGDCBNO	6(06)	
DSGDMBNO	4(04)	
DSG DSGLN	28(1C)	
*DSGDSOHD	7(07)	20
*DSGDSOHI	7(07)	10
*DSGDSOHS	7(07)	02
*DSGDSOH1	7(07)	04
*DSGDSOLS	7(07)	80
*DSGDSORI	7(07)	44
*DSGDSOUP	7(07)	01
*DSGDUPS	15(0F)	08
*DSGHDULD	15(0F)	40
DSGHSADD	8(08)	
*DSGHSWLR	15(0F)	01
DSGINDA	7(07)	
DSGINDB	14(0E)	
DSGINDC	15(0F)	
DSGINDG	16(10)	
DSGLROOT	24(18)	
DSGNOSAM	20(14)	
*DSGPADKY	15(0F)	10
*DSGPREM	16(10)	80
*DSGRETD	16(10)	04
*DSGVL	16(10)	08
*DSGXP	16(10)	10

RECORD LAYOUT - DSG

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	DSGDCBA		Address of the ACB extension for this data set (KSDS ACB extension if HISAM)
4(04)	2	DSGDMBNO		DMB number for this DSG
6(06)	1	DSGDCBNO		ACB number of ACB in DMB (KSDS ACB number if HISAM)
7(07)	1	DSGINDA		JCB indicators
		DSGDSOLS	80	This is last DSG in JCB
		DSGDSORI	44	Data set group is root in index
		DSGDSOHD	20	Data set group is HDAM
		DSGDSOHI	10	Data set group is HIDAM
		DSGDSOH1	04	Data set group is HISAM or simple HISAM
		DSGDSOHS	02	Data set group is HSAM or simple HSAM
		DSGDSOUP	01	Data set group is SHSAM or SHISAM
8(08)	4	DSGHSADD		HSAM I/O area after open
12(0C)	2	DSGBOFF		HSAM block size
14(0E)	1	DSGINDB		(Not used in DL/I DOS/VS)
15(0F)	1	DSGINDC		JCB indicators
		DSGBLDEL	80	Delete/replace DSG
		DSGHDULD	40	HD unload is running
		DSGCONST	20	Index data set contains constant
		DSGPADKY	10	Search argument not equal to key length
		DSGDUPS	08	Nonunique secondary index keys
		DSGHSWLR	01	HSAM wrong length record
16(10)	1	DSGINDG		DSG indicators - retrieve's variable length flags
		DSGPREM	80	Segment prefix moved to work area
		DSGDATX	40	Segment completely expanded
		DSGXP	10	Force complete segment expansion
		DSGVL	08	The variable length routine has been entered for segment
		DSGRETD	04	Data return call
		DSGCOMMD	02	Path return call
17(11)	3			**Reserved**
20(14)	4	DSGNOSAM		Retrieve's HSAM ID
24(18)	4	DSGLROOT		RBA of current root
28(1C)		DSGDSGLN		Length of each DSG section of JCB

FDB - FIELD DESCRIPTION BLOCK

DSECT Name: FDB

The field description block (FDB) is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*FDBCHAR	10(0A)	03	
FDBDCENF	10(0A)		
FDBEND	12(0C)		(See XDFLD fields)
*FDBEQOK	10(0A)	20	
FDBFLENG	11(0B)		
*FDBFP	10(0A)	04	
*FDBHEX	10(0A)	01	
*FDBKEY	10(0A)	40	
*FDBLAST	10(0A)	80	
FDBLEN	11(0B)		(See DFLD fields)
FDBOFFCK	8(08)		(See /CK fields)
FDBOFFST	8(08)		
*FDBPACK	10(0A)	02	
*FDBSPEC	10(0A)	10	
FDBSYMBL	0(00)		
FDBSYSLN	10(0A)		(See /CK fields
FDBSYSNM	0(00)		(See /CK fields)
*FDBTYPE	10(0A)	07	
*FDBXDCON	10(0A)	08	(See XDFLD fields)
*FDBXDEQ	10(0A)	01	(See XDFLD fields)
FDBXDFLG	10(0A)		(See XDFLD fields)
FDBXDLEN	12(0C)		(See XDFLD fields)
*FDBXDLST	10(0A)	80	(See XDFLD fields)
FDBXDNM	0(00)		(See XDFLD fields)
FDBXDSEC	8(08)		(See XDFLD fields)
*FDBXDSPC	10(0A)	10	
*FDBXDSSQ	10(0A)	04	(See XDFLD fields)
*FDBXDSSS	10(0A)	20	
*FDBXDSYM	10(0A)	40	
*FDBZD	10(0A)	07	

RECORD LAYOUT - FDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FDBSYMBL		Symbolic name
8(08)	2	FDBOFFST		Field offset from segment beginning
10(0A)	1	FDBDCENF		Flags
		FDBLAST	80	Last FDB for this segment
		FDBKEY	40	This is segment's sequence field
		FDBEQOK	20	Duplicate sequence fields allowed
		FDBSPEC	10	Special FDB (XDFLD, /CK, or /SK)
		FDBTYPE	07	Field format bits
		FDBZD	07	Field is zoned decimal
		FDBFP	04	Field is floating point
		FDBPACK	02	Field is packed decimal
		FDBHEX	01	Field is hexadecimal
		FDBCHAR	03	Field is character
11(0B)	1	FDBFLENG		Executable field length

This describes the /CK system-related field

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00	3	FDBSYSNM		Constant '/CK'
3(03)	5			Remainder of field name
8(08)	2	FDBOFFCK		Offset from beginning of concatenated key
10(0A)	2	FDBSYSLN		Bits 0-3 = X'0001'; Bits 4-15 = length minus 1

This describes the XDFLD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FDBXDNM		FDB Name
8(08)	2	FDBXDSEC		Offset to secondary list for this index
10(0A)	1	FDBXDFLG		Flags
		FDBXDLST	80	Last FDB
		FDBXDSYM	40	Pointer is symbolic
		FDBXDSSS	20	Pointer is contained in SOURCE/SUBSEQ data
		FDBXDSPC	10	Special FDB
		FDBXDCON	08	Constant present
		FDBXDSSQ	04	SUBSEQ present
		FDBXDEQ	01	Index segment same as index source segment
11(0B)	1	FDBXDLEN		Length of search field
12(0C)		FDBEND		End of FDB entry
12(0C)		FDBLEN		Length of FDB entry (FDBEND minus FDBSYMBL)

FER - FIELD EXIT ROUTINE INTERFACE LIST

DSECT Name: FER

The FER (Field Exit Routine Interface List) is used to pass information to the named user-written exit routine whenever a designated field is to be processed.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Code (Char)
FERPCSC	2(02)	
*FERPCSCT	2(02)	B
*FERPCSFE	2(02)	C
*FERPCSNT	2(02)	A
*FERPCSOK	2(02)	
*FERPCSTC	2(02)	D
FERPEC	0(00)	
FERPFNCT	1(01)	
FERPFSBA	28(1C)	
*FERPGET	0(00)	G
*FERPINS	1(01)	I
FERPLEN	80(50)	
FERPPFA	12(0C)	
FERPPFL	10(0A)	
FERPPSA	4(04)	
*FERPPUT	0(00)	P
*FERPREP	1(01)	R
*FERPRET	1(01)	G
*FERPSSA	1(01)	S
FERPUFA	24(18)	
FERPUFL	22(16)	
FERPUSA	16(10)	
FERPUWA	32(20)	
*FERPXDF	1(01)	X

RECORD LAYOUT - FER

Offset Dec(Hex)	Length	Field/Flag Name	Code (Char)	Meaning
0(00)	1	FERPEC		Entry code
		FERPGET	G	Get function
		FERPPUT	P	Put function
1(01)	1	FERPFNCT		Function code
		FERPRET	G	Retrieve segment conversion
		FERPINS	I	Insert
		FERPREP	R	Replace
		FERPSSA	S	Retrieve SSA conversion
		FERPXDF	X	Retrieve SSA conversion for XDFLD
2(02)	1	FERPCSC		Conversion status code
		FERPCSOK		OK
		FERPCSNT	A	Numeric truncation error
		FERPCSCT	B	Character truncation error
		FERPCSFE	C	Format error
		FERPCSTC	D	Type conflict
3(03)	1			**Reserved**
4(04)	4	FERPPSA		Physical segment address (if variable length, points to two byte length field)
8(08)	2			**Reserved**
10(0A)	2	FERPPFL		Physical field length (zero if virtual field)
12(0C)	4	FERPPFA		Physical field address (zero if virtual field)
16(10)	4	FERPUSA		User segment address
20(14)	2			**Reserved**
22(16)	2	FERPUFL		User field length
24(18)	4	FERPUFA		User field address
28(1C)	4	FERPFSBA		FSB address
32(20)	48	FERPUWA		User work area
80(50)	0	FERPLEN		Length of field exit routine interface list

FERT - FIELD EXIT ROUTINE TABLE

DSECT Name: FERT

The FERT (Field Exit Routine Table) is used to hold information about a user-written exit routine.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code (Hex)
*FERTDUMP	20(14)	80
FERTFLAG	20(14)	
FERTLEN	24(18)	
FERTNAME	0(00)	
FERTPRES	16(10)	
FERTRTEP	8(08)	
FERTRTLG	12(0C)	

RECORD LAYOUT - FERT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FERTNAME		Module name
8(08)	4	FERTRTEP		Module entry point
12(0C)	4	FERTRTLG		Module length
16(10)	4	FERTPRES		Pointer to next FERT entry
20(14)	1	FERTFLAG		
		FERTDUMP	80	Control block dumped
21(15)	3			**Reserved**
24(18)	0	FERTLEN		Length of field exit routine table

FSB - FIELD SENSITIVITY BLOCK

DSECT Name: FSB

The FSB (Field Sensitivity Block) is used to hold information about a field which has been defined with a SENFLD statement during PSBGEN.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
FSBCHAIN	28(1C)	
*FSBCHAR	10(0A)	03
*FSBCR	11(0B)	20
*FSBDPF	10(0A)	10
*FSBEQOK	10(0A)	20
FSBFDBP	0(00)	
*FSBFER	16(10)	20
FSBFERTA	24(18)	
FSBFLAG	11(0B)	
FSBFLDNM	0(00)	
*FSBFP	10(0A)	04
*FSBHEX	10(0A)	01
*FSBIV	16(10)	40
FSBIVA	20(14)	
*FSBKEY	10(0A)	40
*FSBLAST	10(0A)	80
FSBLEN	32(20)	
*FSBNR	16(10)	08
*FSBOVF	11(0B)	40
*FSBPACK	10(0A)	02
FSBPCHA	4(04)	
FSBPHYAD	6(06)	
FSBPVLEN	12(0C)	
FSBPVLOC	8(08)	
FSBPVTYP	10(0A)	
*FSBSSA	11(0B)	80
*FSBTYPE	10(0A)	07
*FSBUCHAR	16(10)	03
*FSBUFP	16(10)	04
*FSBUHEX	16(10)	01
*FSBUPACK	16(10)	02
FSBUVLEN	18(12)	
FSBUVLOC	14(0E)	
FSBUVTYP	16(10)	
*FSBUZD	16(10)	07
*FSBVF	16(10)	10
*FSBZD	10(0A)	07

RECORD LAYOUT - FSB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	FSBFLDNM		Field name
0(00)	4	FSBFDBP		FDB address (ACBGEN only)
4(04)	2	FSBPCHA		Physical view chain pointer (ACBGEN only)
6(06)	2	FSBPHYAD		Field physical adjustment factor (ACBGEN only)
8(08)	2	FSBPVLOC		Displacement in physical segment
10(0A)	1	FSBPVTYP		Physical field type
		FSBLAST	80	Last FSB
		FSBKEY	40	Sequence field
		FSBEQOK	20	Duplicate sequence allowed
		FSBDPF	10	Field is in destination parent
		FSBTYPE	07	Field format bits
		FSBZD	07	Field format is zoned decimal
		FSBFP	04	Field format is floating point
		FSBCHAR	03	Field format is character
		FSBPACK	02	Field format is packed decimal
		FSBHEX	01	Field format is binary
11(0B)	1	FSBFLAG		Flags
		FSBSSA	80	Field may be used in an SSA
		FSBOVF	40	Field has subfields
		FSBCR	20	Conversion required
12(0C)	2	FSBPVLEN		Physical field length (executable)
14(0E)	2	FSBUVLOC		Field displacement in user's view
16(10)	1	FSBUVTYP		User's field type
		FSBIV	40	Initial value specified
		FSBFER	20	Field exit routine specified
		FSBVF	10	Field is virtual
		FSBNR	08	Replace prohibited
		FSBUZD	07	User field format is zoned decimal
		FSBUFP	04	User field format is floating point
		FSBUCHAR	03	User field format is character
		FSBUPACK	02	User field format is packed decimal
		FSBUHEX	01	User field format is binary
17(11)	1			**Reserved**
18(12)	2	FSBUVLEN		User's field length (executable)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
20(14)	4	FSBIVA		Pointer to specified initial value
24(18)	4	FSBFERTA		Field exit routine table entry address
28(1C)	4	FSBCHAIN		Chain pointer for ACBGEN
32(20)	0	FSBLEN		Length of FSB entry

JCB - JOB CONTROL BLOCK

DSECT Name: JCB

The JCB is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBAILEX	64(40)	04
*JCBBLDEL	179(B3)	80
JCBBOFF	176(B0)	
JCBCODE	60(3C)	
*JCBCOMMD	180(B4)	02
*JCBCONST	179(B3)	20
*JCBDATX	180(B4)	40
JCBDCBA	164(A4)	
JCBDCBNO	170(AA)	
*JCBDEFDL	60(3C)	40
*JCBDLET	148(94)	02
JCBDMBNO	168(A8)	
*JCBDOPI	64(40)	08
JCBDSGLN	188(BC)	
*JCBDSOHD	171(AB)	20
*JCBDSOHI	171(AB)	10
*JCBDSOHS	171(AB)	02
*JCBDSOH1	171(AB)	04
*JCBDSOLS	171(AB)	80
*JCBDSORI	171(AB)	44
*JCBDSOUP	171(AB)	01
*JCBDUPS	179(B3)	08
*JCBFLS	64(40)	01
*JCBHDULD	179(B3)	40
JCBHSADD	172(AC)	
*JCBHSWLR	179(B3)	01
JCBINDA	171(AB)	
JCBINDB	178(B2)	
JCBINDC	179(B3)	
JCBINDG	180(B4)	
*JCBISRT	148(94)	01
*JCBKEYX	180(B4)	20
JCBLEVND	4(04	
JCBLEVTB	0(00)	
JCBLEV1C	32(20)	
JCBLROOT	188(BC)	
JCBLVC	65(41)	
JCBLVT	64(40)	
JCBMKYL	38(26)	
*JCBMLPOS	60(3C)	08
*JCBNODEQ	148(94)	80
JCBNOSAM	184(B8)	
*JCBNTFD	148(94)	08
*JCBOPEN	61(3D)	80
*JCBORGHD	61(3D)	20
*JCBORGHI	61(3D)	10
*JCBORGHS	61(3D)	02

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBORGH1	61(3D)	04
JCBORGN	61(3D)	
*JCBORGRI	61(3D)	44
*JCBORGSH	61(3D)	05
*JCBORGSS	61(3D)	01
*JCBPADKY	179(B3)	10
JCBPC	66(42)	
*JCBPCHK	148(94)	20
JCBPOP	67(43)	
*JCBPPENQ	148(94)	10
*JCBPREM	180(B4)	80
JCBPRESF	63(3F)	
JCBPREVF	30(1E)	
JCBPREVR	31(1F)	
JCBPRLEN	188(BC)	
*JCBRAP	148(94)	40
*JCBRDREQ	60(3C)	01
JCBRES1	40(28)	
JCBRES2	44(2C)	
JCBRES3	48(30)	
JCBRES4	52(34)	
JCBRES5	56(38)	
*JCBRETD	180(B4)	04
*JCBRETDL	60(3C)	20
*JCBRTIST	60(3C)	02
JCBRWKF	62(3E)	
JCBSDBND	12(0C)	
JCBSDB1	8(08)	
*JCBSGRET	60(3C)	04
JCBSIZE	36(24)	
*JCBSKPG	148(94)	04
JCBSTOR1	68(44)	
JCBSTOR2	72(48)	
JCBSTOR3	76(4C)	
JCBSTOR4	80(50)	
JCBSTOR5	84(54)	
JCBSTOR6	88(58)	
JCBSTOR7	92(5C)	
JCBSTOR8	96(60)	
*JCBSWAP	179(B3)	01
*JCBTAREX	60(3C)	10
*JCBTARPR	60(3C)	80
JCBTRACE	16(10)	
*JCBVL	180(B4)	08
JCBWKR0	100(64)	
JCBWKR1	104(68)	
JCBWKR2	108(6C)	
JCBWKR3	112(70)	
JCBWRK4	116(74)	
JCBWKR5	120(78)	
JCBWKR6	124(7C)	
JCBWKR7	128(80)	
JCBWKR8	132(84)	
JCBWKR9	136(88)	
JCBWKR10	140(8C)	
JCBWKR11	144(90)	
JCBWKR12	148(94)	
JCBWKR13	152(98)	
JCBWKR14	156(9C)	
JCBWKR15	160(A0)	
JCBWK12A	148(94)	
JCBWK12B	149(95)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*JCBXP	180(B4)	10

RECORD LAYOUT - JCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	JCBLEVTB		Address of level table
4(04)	4	JCBLEVND		Address of end of level table + 1
8(08)	4	JCBSDB1		Address of first SDB entry (roots)
12(0C)	4	JCBSDBND		Address of end of SDBs + 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 call not requiring a PCB is not put in the trace table. However, the function code appears in JCBPREVF or JCBPREVR.

Name	Code(Hex)	Meaning
FUNCGU	01	'GU' Get Unique
FUNCGHU	01	"GHU' Get Hold Unique
FUNCGN	03	"GN' Get Next
FUNCHHN	03	'GHN' Get Hold Next
FUNCGNP	04	"GNP' Get Next Within Parent
FUNCGHNP	04	"GHNP' Get Hold Next Within Parent
FUNCDRTY	20	Delete/Replace
FUNCREPL	21	"REPL' Replace
FUNCDLET	22	"DLET" Delete
FUNCISTY	40	"ISRT" Insert
FUNCISRT	41	Insert
FUNCASRT	42	DL/I Utility Insert

The following codes must have a PCB

FUNCCHKP	85	"CHKP" checkpoint
FUNCPCBM	90	PCB Call for MPS

The following codes do not require a PCB

FUNCUNLD	A0	"UNLD" Unload Call
FUNCGSCD	A1	"GSCD" Get SCD Call
FUNCTERM	A3	"TERM" Termination Call

DL/I FUNCTION TYPES

FUNCGNTY	80	Get Next Type
FUNCGUTY	40	Get Unique Type
FUNCPATY	20	Parent Type
FUNCHOTY	08	Hold Type

30(1E)	1	JCBPREVF		Prior function
31(1F)	1	JCBPREVR		Prior return code (right byte)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
32(20)	4	JCBLEV1C		Address of first level table entry in call; Address of lowest level table entry succesfully processed by retrieve
36(24)	2	JCBSIZE		PCB plus JCB size
38(26)	2	JCBMKYL		Maximum length of key feedback area
40(28)	4	JCBRES1		Call characteristics set by call analyzer JCBRES1 = X'80' No SSAs X'40' Qualified SSAs X'20' Unqualified SSAs X'10' Multiple SSAs X'08' Multiple unqualified SSAs X'04' Qualified SSA after an unqualified SSA X'02' Last SSA qualified JCBRES1 + 1 = X'04' Call has C command code X'02' Call has T command code X'01' JCBLEV1C has been filled on this call JCBRES1 + 2 = X'80' Any level qualified on data X'40' Any level had D command code X'20' Qualified SSA follows D command code JCBRES1 + 3 = X'80' Field is not in sublist X'40' Qualification field is in logical parent 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 communications switch
		JCBTARPR	80	DLZPOST update twin pointers only
		JCBDEFDL	40	Re-insert of a deleted segment
		JCBRETDL	20	Return deleted segment for HD unload
		JCBTAREX	10	Reposition for GN (no SSA) with multiple positioning

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		JCBMLPOS	08	Retrieve keeping multiple positions
		JCBSGRET	04	Used in positioning after not found
		JCBRTIST	02	Retrieve positioning for insert
		JCBRDREQ	01	DLZSKPG start at next occurence of segment
61(3D)	1	JCBORGN		Open switch and composite organization of all SDBs in the JCB
		JCBOPEN	80	Open done for all data sets in the JCB
		JCBORGRI	44	Organization is root of index
		JCBORGHD	20	Organization is HDAM
		JCBORGHI	10	Organization is HIDAM
		JCBORGSH	05	Organization is simple HISAM
		JCBORGH1	04	Organization is HISAM
		JCBORGHS	02	Organization is HSAM
		JCBORGSS	01	Organization is simple HSAM
62(3E)	1	JCBRWKF		Retrieve's working function
63(3F)	1	JCBPRESF		Present coded function (see DL/I Function Codes)
64(40)	1	JCBLVT		Switches used in accessing segments via DLZSKPG routine
		JCBDOPI	08	Program isolation is to be done for associated PCB
		JCBALLEX	04	All sensitive segments have exclusive intent
		JCBFLS	01	At least one segment has field level sensitivity (used by call analyzer)
65(41)	1	JCBLVC		Level of segment being searched for by retrieve
66(42)	1	JCBPC		Physical code of segment being searched for by retrieve
67(43)	1	JCBPOP		Parent level for within parent calls
68(44)	4	JCBSTOR1		Insert's use across I/O or calls
72(48)	4	JCBSTOR2		Insert's use across I/O or calls
76(4C)	4	JCBSTOR3		Insert's use across I/O or calls
80(50)	4	JCBSTOR4		Address of last segment read - referenced by label BEGBUF in retrieve

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
84(54)	4	JCBSTOR5		Current segment RBA - referenced by label CURTTR in retrieve
88(58)	4	JCBSTOR6		Retrieve's use across I/O or calls
92(5C)	4	JCBSTOR7		Contains switches for positive check phase - referenced by label KEEPIT in retrieve
96(60)	4	JCBSTOR8		Work area for retrieve
100(64)	4	JCBWKR0		Action modules work area
104(68)	4	JCBWKR1		Action modules work area
108(6C)	4	JCBWKR2		Action modules work area
112(70)	4	JCBWKR3		Action modules work area
116(74)	4	JCBWKR4		Action modules work area
120(78)	4	JCBWKR5		Action modules work area
124(7C)	4	JCBWKR6		Action modules work area
128(80)	4	JCBWKR7		Action modules work area
132(84)	4	JCBWKR8		Action modules work area
136(88)	4	JCBWKR9		Action modules work area
140(8C)	4	JCBWKR10		Action modules work area
144(90)	4	JCBWKR11		Action modules work area
148(94)	4	JCBWKR12		Action modules work area
148(94)	4	JCBWK12A		Program isolation switches (retrieve only)
		JCBNODEQ	80	No dequeue processing; all level table entries empty after CHKP, TERM, etc.
		JCBRAP	40	Root anchor pointer enqueued (HDAM only)
		JCBPCHK	20	DLZPCHK calling DLZPOST (enqueue not required)
		JCBPPENQ	10	DLZKDTL enqueued on physical parent searching on data field
		JCBNTFD	08	DLZPCHK processing not found condition
		JCBSKPG	04	DLZDEQ should release all outstanding enqueues
		JCBDLET	02	ENQ/DEQ required in DLZPCHK due to delete
		JCBISRT	01	Indicates DLZHIDA or DLZHDAM is accessing destination parent during a logical child insert

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
149(95)	3	JCBWK12B		Action modules work area
152(98)	4	JCBWKR13		Action modules work area
156(9C)	4	JCBWKR14		Action modules work area
160(A0)	4	JCBWKR15		Action modules work area
Start of each DSG section of JCB				
164(A4)	4	JCBDCBA		Address of the ACB extension for this data set (KSDS ACB extension if HISAM)
168(A8)	2	JCBDMBNO		DMB number for this DSG
170(AA)	1	JCBDCBNO		ACB number of ACB in DMB (KSDS ACB number if HISAM)
171(AB)	1	JCBINDA		JCB Indicators
		JCBDSOLS	80	This last DSG in JCB
		JCBDSORI	44	Data set group is root in index
		JCBDSOHD	20	Data set group is HDAM
		JCBDSOHI	10	Data set group is HIDAM
		JCBDSOH1	04	Data set group is HISAM or simple HISAM
		JCBDSOHS	02	Data set group is HSAM or simple HSAM
		JCBDSOUP	01	Data set group is SHSAM or SHISAM
172(AC)	4	JCBHSADD		HSAM I/O area after open
176(B0)	2	JCBBOFF		HSAM block size
178(B2)	1	JCBINDB		(Not used in DL/I DOS/VS)
179(B3)	1	JCBINDC		JCB indicators
		JCBBLDEL	80	This DSG belongs to delete/replace
		JCBHDULD	40	HD unload is running
		JCBCONST	20	Index data set contains constant
		JCBPADKY	10	Search argument not equal to key length
		JCBDUPS	08	Non-unique secondary index keys
		JCBHSWLR	01	HSAM wrong length record
180(B4)	1	JCBINDG		JCB indicators - retrieve variable length flags
		JCBPREM	80	Segment prefix moved to work area
		JCBDATX	40	Segment completely expanded
		JCBXP	10	Force complete segment expansion
		JCBVL	08	The variable length routine has been entered for segment
		JCBRETD	04	Data return call
		JCBCOMMD	02	Path return call

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
181(B5)	3			**Reserved**
184(B8)	4	JCBNOSAM		Retrieve HSAM's ID
188(BC)	4	JCBLROOT		RBA of current root
		JCBPRLEN		Length of JCB prefix
		JCBDSGLN		Length of each DSG section of JCB

LEV - LEVEL TABLE ENTRY

DSECT Name: LEV

The level table entry is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*LEVCDB	13(0D)	80
*LEVCOMMC	18(12)	40
*LEVCOMMD	19(13)	04
*LEVCOMMF	19(13)	20
*LEVCOMML	19(13)	10
*LEVCOMMN	19(13)	02
*LEVCOMMQ	19(13)	01
*LEVCOMMT	18(12)	80
*LEVCOMMX	18(12)	20
*LEVCONT	13(0D)	08
*LEVDATA	12(0C)	08
*LEVDATA1	17(11)	04
*LEVDLET	12(0C)	80
*LEVEMPTY	12(0C)	40
LEVEND	36(24)	
*LEVEOD	13(0D)	20
LEVF1	12(0C)	
LEVF2	13(0D)	
LEVF3	17(11)	
LEVF4	18(12)	
LEVF5	19(13)	
*LEVHELD	12(0C)	20
*LEVHIER	12(0C)	10
*LEVISRT	17(11)	80
*LEVKEY1	17(11)	02
*LEVLAST	12(0C)	01
LEVLEN	36(24)	
LEVLEV	0(00)	
*LEVLSW	13(0D)	02
*LEVMEMAC	20(14)	08
*LEVMEMAS	20(14)	02
LEVMEMBR	20(14)	
*LEVMEMEQ	20(14)	80
*LEVMEMGT	20(14)	20
*LEVMEMKY	20(14)	04
*LEVMEMLT	20(14)	40
*LEVMEMNE	20(14)	10
*LEVMEMPL	20(14)	01
*LEVNDB	13(0D)	01
*LEVNFPOS	13(0D)	40
LEVNUPC	16(10)	
LEVNUSDB	28(1C)	
LEVPC	1(01)	
*LEVPFRST	12(0C)	02
*LEVPLAST	12(0C)	04
*LEVPSUDO	17(11)	08
LEVSDB	8(08)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
LEVSEGOF	2(02)	
LEVSSA	32(20)	
*LEVSTOP	13(0D)	04
LEVTTR	4(04)	
LEVUSEOF	14(0E)	

RECORD LAYOUT - LEV

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	LEVLEV		Level number
1(01)	1	LEVPC		Current segment physical code

Note: This portion of the level table, once set by retrieve/insert, is never cleared to zeros; it is only changed as needed.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
2(02)	2	LEVSEGOF		Segment's physical code offset from start of record (relative offset to segment from start of buffer)
4(04)	4	LEVTTR		Relative byte address
8(08)	4	LEVSDB		SDB entry address for current segment physical code in this entry
12(0C)	1	LEVF1		Flags
		LEVDLET	80	Segment at this level newly deleted
		LEVEMPTY	40	This level table entry empty
		LEVHELD	20	Segment at this level in hold status
		LEVHIER	10	Segment at this level in hierarchic path (HISAM only)
		LEVDATA	08	Segment at this level moved to user
		LEVPLAST	04	Segment is last of type for parent
		LEVPFRST	02	Segment is first of type for parent
		LEVLAST	01	This is the last level table for PCB
13(0D)	1	LEVF2		Flags
		LEVCDB	80	Verify enques required in data base of current segment
		LEVNFPOS	40	Level has not found position for higher level
		LEVEOD	20	EOD flag
		LEVCONT	08	The SSA at this level allows retrieve to obtain the next sequential segment
		LEVSTOP	04	Used to determine the setting of LEVCONT by retrieve
		LEVLSW	02	Used by retrieve
		LEVNDB	01	Verify enques required in destination parents data base
14(0E)	2	LEVUSEOF		Offset of segment in user I/O area (PSTUSER)

Note: Fields LEVNUPC through LEVSSA describe the SSA set by the call analyzer for this entry.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
16(10)	1	LEVNUPC		Physical code of requested segment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
17(11)	1	LEVF3		Flags
		LEVISRT	80	Inserting at this level (set by retrieve)
		LEVPSUDO	08	This is a pseudo SSA filling gap
		LEVDATA1	04	SSA qualified on data field
		LEVKEY1	02	SSA qualified on key field
18(12)	1	LEVF4		Flags
		LEVCOMMT	80	T command code - retrieve by direct address
		LEVCOMMC	40	C command code - qualifier is concatenated key
		LEVCOMMX	20	X command code - index maintenance internal call
19(13)	1	LEVF5		Flags
		LEVCOMMF	20	F command code - get first of segment type
		LEVCOMML	10	L command code - get last of segment type
		LEVCOMMD	04	D command code - transfer data this level
		LEVCOMMN	02	N command code - do not replace this level
		LEVCOMMQ	01	Q command code - enqueue segment at this level read only
20(14)	1	LEVMEMBR		Switch for each member
		LEVMEMEQ	80	Operator has = sign
		LEVMEMLT	40	Operator has < sign
		LEVMEMGT	20	Operator has > sign
		LEVMEMNE	10	Operator is not equal (LEVMEMGT + LEVMEMLT)
		LEVMEMAC	08	This member in use - (unqualified in only bit)
		LEVMEMKY	04	Qualification is on key field
		LEVMEMAS	02	See meaning for X'01'
		LEVMEMPL	01	LEVMEMAS + LEVMEMPL = right parenthesis present (always on for DL/I DOS/VS)
21(15)	7			**Reserved**
28(1C)	4	LEVNUSDB		SSAs SDB address
32(20)	4	LEVSSA		SSAs left parenthesis position address
36(24)		LEVEND		End of level table entry
36(24)		LEVLEN		Length of level table entry (LEVEND minus LEVLEV)

MPCPT - MPC PARTITION TABLE

The Master Partition Controller (MPC) partition table is used to pass control information when processing batch partition application programs under multiple partition support (MPS). The MPC partition table resides in the transaction work area. There is one entry for every partition sysgened.

Field Name	Length (bytes)	Description
MPCPARTB	200	Contains one 28 byte entry (see MPC Partition Table entry) for each partition defined during system generation. The last entry is delimited by a full-word of X'FF'.
MPCECBLT	4 (per entry)	This is the CICS WAITM ECB list. It contains one entry for each: • DLZXCB00 (Stop Transaction XECB) - used to stop MPS • DLZXCB01 (Stop Partition XECB) - posted by BPC when it stops • DLZXCBn0 (Start partition XECB) - defined by MPS. Used by batch initialization to notify MPC to start the BPC • DLZXCBn3 (ABEND XECB) - Used for ABEND handling Note: n is the partition indicator. It can be 1 through 7. The last entry is delimited by a fullword of X'FF'.

MPC PARTITION TABLE ENTRY

DSECT Name: MPCPT

There is one MPC partition table entry for every partition defined during system generation.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*MPCADEF	0(00)	02
MPCAXECB	12(0C)	
*MPCCNBPC	20(14)	80
MPCDELIM	0(00)	
*MPCERR	0(00)	40
MPCFLAG	0(00)	
MPCFLAG1	20(14)	
*MPCPACT	0(00)	80
MPCPID	3(03)	
*MPCPSTP	0(00)	10
MPCPTLN	28(1C)	
MPCRC1	1(01)	
MPCRC2	2(02)	
*MPCREBPC	0(00)	01
*MPCSDEF	0(00)	04
MPCSXECB	8(08)	
MPCTCA	4(04)	
*MPCTSTP	0(00)	20
*MPCXECB	0(00)	08

RECORD LAYOUT - MPC

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	MPCDELIM		MPCPT delimiter field
0(00)	1	MPCFLAG		MPC activity flags
		MPCPACT	80	Partition active indicator
		MPCERR	40	Error condition encountered on DL/I scheduling call, or BPC attach failure
		MPCTSTP	20	Stop transaction indicator
		MPCPSTP	10	Stop partition indicator
		MPCXECB	08	XECBs deleted for this partition
		MPCSDEF	04	Start XECB defined
		MPCADEF	02	ABEND XECB defined
		MPCREBPC	01	Reschedule BPC
1(01)	1	MPCRC1		Error return code from TCAFCTR
2(02)	1	MPCRC2		Error return code from TCADLTR
3(03)	1	MPCPID		Partition identifier (F1, F2,...)
4(04)	4	MPCTCA		Address of TCA
8(08)	4	MPCSXECB		Address of stop partition XECB (DLZXCB01)
12(0C)	4	MPCAXECB		Address of partition ABEND XECB (DLZXCBn3)
16(10)	4	Unnamed		**Reserved**
20(14)	1	MPCFLAG1		MPC activity flags
		MPCCNBPC	80	Cancel BPC at stop transaction when MPS batch partition is not active.
21(15)	3	Unnamed		**Reserved**
24(18)	4	Unnamed		**Reserved**
28(1C)		MPCPTLN		Length of partition table entry

PCB - PROGRAM COMMUNICATION BLOCK

DSECT Name: DBPCB

The data management PCB (program communication block) is described as part of the general structure and description of the program specification block (PSB),

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DBPCBDBD	0(00)	
DBPCBJCB	16(10)	
DBPCBKFD	36(24)	
DBPCBLEV	8(08)	
DBPCBLKY	28(1C)	
DBPCBMKL	28(1C)	
DBPCBNSS	32(20)	
DBPCBPRO	12(0C)	
DBPCBSFD	20(14)	
DBPCBSTC	19(0A)	
*DBPCBTKW	16(10)	80

RECORD LAYOUT - PCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DBPCBDBD		DBD Name
8(08)	2	DBPCBLEV		Level feedback
10(0A)	2	DBPCBSTC		Status codes
12(0C)	4	DBPCBPRO		DL/I processing options
16(10)	4	DBPCBJCB		JCB address
		DBPCBTKW	80	Another task waiting for resource owned by this task
20(14)	8	DBPCBSFD		Segment name feedback
28(1C)	4	DBPCBLKY		Maximum length of key feedback area
28(1C)	4	DBPCBMKL		Current length of key feedback area
32(20)	4	DBPCBNSS		Number of sensitive segments in the PCB
36(24)	Var	DBPCBKFD		Key feedback area

PDCA - PROBLEM DETERMINATION CONTROL AREA

DSECT Name: PDCA

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

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PDCACPAC	0(00)	
PDCAFERT	8(08)	
PDCAFLAG	12(0C)	
PDCAMSG	13(0D)	
*PDCASTOP	12(0C)	
PDCAXPRM	4(04)	

RECORD LAYOUT - PDCA

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	4	PDCACPAC		Variable length segment compression routine list pointer
4(04)	4	PDCAXPRM		Secondary index suppression routine list pointer
8(08)	4	PDCAFERT		Field exit routine list
12(0C)	1	PDCAFLAG		PDCA flag byte
		PDCASTOP	80	Stop saving messages
13(0D)	3	PDCAMSG		ABEND code
16(0F)	16			**Reserved**

PDIR - PSB DIRECTORY

DSECT Name: DLZPDIR

The PSB directory contains an entry for every PSB (program specification block) that may run under DL/I control. The PSB directory is part of the DL/I nucleus and is created during DL/I system definition for online processing. The start address of the PSB directory (SCDDLIPS) and the entry length (SCDDLIPL)are contained in the SCD (system contents directory).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PDIRADDR	8(08)	
*PDIRBAD	19(13)	01
*PDIRBPLI	19(13)	08
PDIRCODE	18(12)	
*PDIRDELT	18(12)	02
*PDIRDUPL	18(12)	10
*PDIREM	19(13)	20
PDIREMOT	24(18)	
*PDIREXC	18(12)	40
PDIRLEN	28(1C)	
*PDIRNOSC	19(13)	80
*PDIRNTNT	19(13)	10
PDIROPTC	19(13)	
*PDIRPLI	18(12)	20
PDIRPSBL	12(0C)	
*PDIRSCHD	19(13)	40
PDIRSILA	20(14)	
PDIRSYM	(0(00)	
*PDIRTFAL	18(12)	01
*PDIRUPD	18(12)	80
*PDIRZWA	16(10)	

RECORD LAYOUT - PDIR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	PDIR		Label used to establish address
0(00)	8	PDIRSYM		PSB execution name - converted from name supplied during PSBGEN
8(08)	4	PDIRADDR		PSB address (contains 0 for remote PSB)
12(0C)	4	PDIRPSBL		Storage required for PSB
16(10)	2	PDIRZWA		Storage required for index workarea
18(12)	1	PDIRCODE		PSB code byte
		PDIRUPD	80	This PSB is update sensitive
		PDIREXC	40	This PSB requires DMB exclusive control
		PDIRPLI	20	This PSB for PL/I
		PDIRDUPL	10	This PSB is duplicate
		PDIRDELT	02	This PSB is delete sensitive
		PDIRTFAL	01	PSDB-SDB chaining error detected during online task termination
19(13)	1	PDIROPTC		PSB scheduling codes
		PDIRNOSC	80	Do not schedule this PSB
		PDIRSCHD	40	This PSB is scheduled
		PDIRREM	20	This PSB is remote
		PDIRNTNT	10	This PSB is waiting for intent
		PDIRBPLI	08	DFHTBP using PL/I PSB
		PDIRBAD	01	PSB initialization failed
20(14)	4	PDIRSILA		Address of PSB segment intent list
24(18)	4	PDIREMOT		Address of RPDIR entry for this remote PSB
28(1C)		PDIRLEN		PSB directory entry length

PPST - PST PREFIX

DSECT Name: DLZPPST

The PST prefix contains data required for user task scheduling in a CICS/VS online environment. It also contains a section used by buffer handler for enqueue/dequeue information and another section used for online segment intent scheduling. The PST prefix is logically part of the PST (partition specification table). However, in order to operate more efficiently in a virtual storage environment, all PST prefixes (one for batch) are organized so that they are physically located in one contiguous area.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PPSTA	4(04)	01
*PPSTACT	4(04)	04
*PPSTBF	4(04)	10
PPSTCA	5(05)	
PPSTCB	1(01)	
PPSTCF	0(00)	
PPSTCHAI	28(1C)	
PPSTCW	3(03)	
PPSTECB	2(02)	
PPSTEND	32(20)	
PPSTEXCI	12(0C)	
PPSTID	8(08)	
PPSTIND	4(04)	
*PPSTIO	4(04)	80
PPSTLEN	32(20)	(See segment intent scheduling section)
PPSTMATR	24(18)	
*PPSTMPS	4(04)	08
*PPSTMSDL	4(04)	02
PPSTPECI	16(10)	
PPSTPDIR	12(0C)	(See segment intent scheduling section)
*PPSTSI	4(04)	40
PPSTSUPO	20(14)	
*PPSTTC	4(04)	20
PPSTTC	9(09)	
PPSTTSKP	16(10)	(See segment intent scheduling secion)

RECORD LAYOUT - PPST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	PPSTCF		Prefix chain forward pointer
1(01)	1	PPSTCB		Prefix chain backward pointer
2(02)	1	PPSTECB		POST/WAIT byte of PST ECB
3(03)	1	PPSTCW		PST prefix program isolation wait chain
4(04)	1	PPSTIND		Task schedule and dispatch indicators
		PPSTIO	80	Waiting for I/O
		PPSTSI	40	Cannot schedule due to segment intent conflict
		PPSTTC	20	Cannot schedule - task count limit exceeded
		PPSTBF	10	Task enqueued by buffer handler
		PPSTMPS	08	Indicates MPS task
		PPSTACT	04	This is current task
		PPSTMSDL	02	Scheduled by BPC
		PPSTA	01	Task is scheduled
5(05)	3	PPSTCA		Address of PST
8(08)	1	PPSTID		Task ID
9(09)	3	PPSTTCA		Task TCA address
This section used by buffer handler for enqueue/dequeue				
12(0C)	4	PPSTEXCI		Enqueue/dequeue pointers for existing control interval: Byte 0-1 = buffer number Byte 2-3 = PPST number of task next in chain
16(10)	4	PPSTPECI		Enqueue/dequeue pointers for pending control interval: Byte 0-1 = buffer number Byte 2-3 = PPST number of task next in chain
20(14)	4	PPSTSUPO		Enqueue/dequeue pointer for subpool space: Byte 0-1 = subpool number Byte 2-3 = PPST number of next task in chain
24(18)	4	PPSTMATR		Enqueue/dequeue pointers for interlock detection matrix space: Byte 0-1 = X'00' Byte 2-3 = PPST number of next task in chain

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
28(1C)	4	PPSTCHAI		Enqueue/dequeue pending control interval chain field pointers: Byte 0-1 = buffer number Byte 2-3 = PPST number of next task in chain
32(20)		PPSTEND		End of prefix DSECT

***This section used to online segment intent scheduling**

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
12(0C)	4	PPSTPDIR		Task PDIR entry address
16(10)	1	PPSTTSKP		Task dispatching priority
32(20)	1			Reset to end of prefix DSECT
32(20)		PPSTLEN		Length of PST prefix

PSB - PSB Prefix

DSECT Name: PSB

The PSB prefix is described as part of the general structure and description of the program specification block (PSB)

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSBCODE	29(1D)	
PSBDBOFF	34(22)	
*PSBFLS	29(1D)	01
PSBFRTA	0(00)	
PSBIOASZ	1(01)	
PSBIOAWK	18(24)	
PSBINDEX	28(1C)	
PSBLIST	36(24)	
PSBNDXWK	20(14)	
*PSBPLI	29(1D)	10
PSBPST	12(0C)	
PSBSEGWK	8(08)	
PSBSIZE	30(1E)	
PSBTPOFF	32(20)	
PSBVMID	0(00)	
*PSBV11	0(00)	01
PSBXIOWK	4(04)	
PSBXPCB	16(10)	

RECORD LAYOUT - PSB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	PSBVMID		DOS DL/I version ID
		PSBV11	01	Version 1.1 or later
0(00)	4	PSBFRTA		Field exit routine address. If no entries in table, low order 3 bytes = 0 (used only during initialization)
1(01)	3	PSBIOASZ		Size of the PSB I/O work area whose address is in PSBIOAWK. This field contains a 16-bit logical number.
4(04)	4	PSBXIOWK		Address of index I/O work area or user's version of a segment built by retrieve
8(08)	4	PSBSEGWK		Address of variable length segment work area
12(0C)	4	PSBPST		PST address if PSB is scheduled or active
16(10)	4	PSBXPCB		Address of index PCB
20(14)	4	PSBNDXWK		Address of index maintenance work area or pointer to the field exit parameter list
24(18)	4	PSBIOAWK		Address of I/O work area
28(1C)	1	PSBINDEX		(Not used in DL/I DOS/VS)
29(1D)	1	PSBCODE		PSB flags
		PSBPLI	10	PL/I is source language
		PSBFLS	01	PSB contains field sensitive segment
30(1E)	2	PSBSIZE		PSB size
32(20)	2	PSBTPOFF		(Not used in DL/I DOS/VS)
34(22)	2	PSBDBOFF		Offset from the PSBLIST to first DB PCB
36(24)	Var	PSBLIST		Beginning of PCB list. Note: this field is a list of fullword pointers containing PCB addresses. Last PCB address word has byte 0, bit 0 = 1. List may contain a maximum of 64 addresses. For PL/I programs these pointers are to the dope Vector Tables in which the first word is a pointer to the associated PCB.

<u>PSDB - PHYSICAL SEGMENT DESCRIPTION BLOCK</u>

DSECT Name: DMBPSDB

The PSDB is described as part of the general structure and description of the data management block (DMB)

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBCKL	14(0E)	
*DMBCPT	24(18)	04
*DMBCPTIT	24(18)	01
*DMBCPTKY	24(18)	02
*DMCCTR	7(07)	80
DMBDCB	6(06)	
DMBDL	10(0A)	
DMBDLT	13(0D)	
*DMBDRL	13(0D)	03
*DMBDRP	13(0D)	02
*DMBDRV	13(0D)	01
*DMBEX	16(10)	80
DMBFDBA	16(10)	
DMBFLAG	32(20)	
DMBFSDB	20(14)	
*DMBIFST	12(0C)	10
*DMBIHERE	12(0C)	30
*DMBILST	12(0C)	20
*DMBIRL	12(0C)	03
*DMBIRP	12(0C)	02
*DMBIRV	12(0C)	01
DMBISRT	12(0C)	
*DMBLCEX	32(20)	20
DMBLEV	2(02)	
*DMBLP	7(07)	02
*DMBLPEX	32(20)	40
DMBLST	32(20)	
*DMBLTBK	7(07)	04
*DMBLTFD	7(07)	08
*DMBNXEX	32(20)	10
*DMBPI	24(18)	80
DMBPLEM	36(24)	
*DMBPP	7(07)	10
DMBPPBK	5(05)	
DMBPPFD	4(04)	
DMBPRSZ	8(08)	
DMBPSC	1(01)	
DMBPSDBN	36(24)	
*DMBPTBK	7(07)	20
*DMBPTFD	7(07)	40
DMBPTR	7(07)	
*DMBRRL	13(0D)	0C
*DMBRRP	13(0D)	08
*DMBRRV	13(0D)	04
DMBSC	0(00)	
DMBSCTAB	25(19)	
DMBSGMN	28(1C)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBSGMX	30(1E)	
*DMBUP	16(10)	40
DMBUSE	16(10)	
DMBVLDFG	24(18)	
*DMBVLS	24(18)	04
*DMBXDES	32(20)	04
DMBXNULL	3(03)	
*DMBXPROT	12(0C)	80

RECORD LAYOUT - PSDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	DMBSC		Segment code
			01	Root segment code
1(01)	1	DMBPSC		Parent's segment code
2(02)	1	DMBLEV		Segment level
3(03)	1	DMBXNULL		(Not used in DL/I DOS/VS)
4(04)	1	DMBPPFD		Pointer number in parent to first occurrence of segment for parent
5(05)	1	DMBPPBK		Pointer number in parent to last occurrence of segment for parent
6(06)	1	DMBDCB		ACB number
7(07)	1	DMBPTR		Prefix flags
		DMBCTR	80	Counter present
		DMBPTFD	40	Segment has physical twin forward pointer
		DMBPTBK	20	Segment has physical twin backward pointer
		DMBPP	10	Segment has physical parent pointer
		DMBLTFD	08	Segment has logical twin forward pointer
		DMBLTBK	04	Segment has logical twin backward pointer
		DMBLP	02	Segment has logical parent pointer
8(08)	2	DMBPRSZ		Prefix length of segment
19(0A)	2	DMBDL		Data length of segment
12(0C)	1	DMBISRT		Insert rules
		DMBXPROT	80	System data in index is protected
		DMBIHERE	30	If no key field, insert at current position
		DMBILST	20	If no key field, insert after existing segment
		DMBIFST	10	If no key field, insert before existing segment
		DMBIRL	03	Insert rule is logical
		DMBIRP	02	Insert rule is physical
		DMBIRV	01	Insert rule is virtual
13(0D)	1	DMBDLT		Delete/replace rules
		DMBRRL	0C	Replace rule is logical
		DMBRRP	08	Replace rule is physical
		DMBRRV	04	Replace rule is virtual
		DMBDRL	03	Delete rule is logical
		DMBDRP	02	Delete rule is physical
		DMBDRV	01	Delete rule is virtual

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
14(0E)	2	DMBCKL		Concatenated key length of parent of this segment
16(10)	1	DMBUSE		Code Byte
		DMBEX	80	This PSDB in use exclusively
		DMBUP	40	This PSDB in use for update. Bits 2-7 contain a count of read-only users
16(10)	4	DMBFDBA		Address of FDBs for this segment
20(14)	4	DMBFSDB		Address of first SDB for this segment
24(18)	1	DMBVLDFG		Variable length data flag
		DMBPI	80	Program isolation should be done for this segment
		DMBCPT	08	Segment has compression routine
		DMBVLS	04	Segment is variable length
		DMBCPTKY	02	Compression routine has key expand routine
		DMBCPTIT	01	Compression routine has intialization processing
25(19)	3	DMBSCTAB		Address of segment compaction table
28(1C)	2	DMBSGMN		If variable length segment; minimum length of segment
30(1E)	2	DMBSGMX		If variable length segment; maximum length of segment
32(20)	1	DMBFLAG		Secondary list flag
		DMBLPEX	40	A logical parent exists (segment is a logical child)
		DMBLCEX	20	One or more logical children exists (segment is a logical parent)
		DMBNXEX	10	One or more indexes exist
		DMBXDEX	04	An indexed segment exists
32(20)	4	DMBLST		Address of secondary list for this segment
36(24)		DMBPSDBN		End of one PSDB entry
36(24)		DMBPLEM		Length of each PSDB entry (DMBPSDBN minus DMBSC)

PSIL - PSB INTENT LIST

DSECT Name: DLZPSIL

The PSB intent list is pointed to from the PSB directory and is a list of all the DMBs which may be used by that PSB (program).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PSILBFRI	8(08)	20
*PSILDBEX	8(08)	80
*PSILDBUP	8(08)	40
PSILDIRA	0(00)	
PSILDIRN	4(04)	
PSILDMBN	0(00)	
PSILLNGH	9(09)	
PSILNTNT	8(08)	
PSILSEGD	10(0A)	

RECORD LAYOUT - PSIL

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	PSILDMBN		DMB name for this list entry - overlaid during initialization
0(00)	4	PSILDIRA		Address of DMB directory entry - resolved during initialization
4(04)	2	PSILDIRN		DMB number of this DMB
6(06)	2			**Reserved**
8(08)	1	PSILNTNT		Segment intent descriptor byte
		PSILDBEX	80	PSB contains a PCB which requires exclusive control for this DMB
		PSILDBUP	40	PSB contains a PCB which is update sensitive
		PSILBFRI	20	Buffer pool space required for this KSDS
9(09)	1	PSILLNGH		Length of this entry in list
10(0A)	Var	PSILSEGD		Segment intent bits. Two bits are used for each segment in the DMB and represent the PSB's sensitivity to each PSDB.

Their meanings are:

<u>Bit Meaning</u>
00 PSB not sensitive to segment
01 PSB read only sensitive
10 PSB update sensitive
11 PSB requests exclusive control (HISAM root insert)

The bits are allocated to segments in the following manner:

	BYTE 1								BYTE 2							
BIT	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SEGMENT	4		3		2		1		8		7		6		5	

The second part of the segment intent bits is a mask. It is constructed from the segment intent bits of the first part. Part 2 has the same length as part 1.

PST - PARTITION SPECIFICATION TABLE

DSECT Name: DLZPST

One partition specification table (PST) exists for each task in an online or batch processing partition. All DL/I resources allocated to the task can be located through the PST. The PST also contains pointers to the task I/O area and any segments currently associated with the task.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*DBLCMC	436(1B4)	00
*DBLFSE1	436(1B4)	00
*DBLFSE2	436(1B4)	04
*DBLLASTC	436(1B4)	08
*DBLLGDLT	436(1B4)	60
*DBLNDXC	436(1B4)	80
*DBLNEWBL	436(1B4)	01
*DBLNTCR	436(1B4)	70
*DBLOOPS	436(1B4)	0A
*DBLPHYD	436(1B4)	20
*DBLPHYI	436(1B4)	40
*DBLPHYR	436(1B4)	10
*DBLPHYRO	436(1B4)	02
PSTABIND	72(048)	
PSTACBNM	146(092)	
PSTACCT	92(05C)	
*PSTBATCH	464(1D0)	80
*PSTBDCAL	133(085)	10
*PSTBFALT	132(084)	05
*PSTBFMPT	132(084)	04
PSTBFUSE	160(0A0)	
*PSTBKLCT	132(084)	01
PSTBLKNM	140(08C)	
*PSTBTMPF	132(084)	03
*PSTBTMPF	133(085)	03
PSTBUFFA	156(09C)	
*PSTBYALT	132(084)	06
*PSTBYEND	133(085)	28
*PSTBYLCT	132(084)	02
PSTBYTNM	148(094)	
*PSTCALI	465(1D1)	02
*PSTCANLI	483(1E3)	40
*PSTCHKP	465(1D1)	04
*PSTCLOK	133(085)	00
PSTCLRWT	254(0FE)	
PSTCNVB	475(1DB)	
PSTCODE1	68(044)	
PSTCPLN	180(0B4)	
PSTCTGFL	220(0DC)	
PSTCTGL1	244(0F4)	
PSTCTGL2	247(0F7)	
PSTCTGNM	180(0B4)	
PSTCTGPL	180(0B4)	
PSTCTGRT	248(0F8)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSTCTGWK	244(0F4)	
PSTCURWA	340(154)	
PSTCWKLN	248(0F8)	
PSTDATA	152(098)	
PSTDBPCD	128(080)	
PSTDDLET	120(078)	
PSTDGHN	108(06C)	
PSTDGHNP	112(070)	
PSTDGHU	104(068)	
PSTDGN	96(060)	
PSTDGNP	100(064)	
PSTDGU	92(05C)	
PSTDISRT	116(074)	
PSTDLIWA	44(02C)	
PSTDLIWB	48(030)	
PSTDLIWC	52(034)	
PSTDLIWD	56(038)	
PSTDLIWE	60(03C)	
PSTDLIWF	64(040)	
PSTDLIW0	4(004)	
PSTDLIW1	8(008)	
PSTDLIW2	12(00C)	
PSTDLIW3	16(010)	
PSTDLIW4	20(014)	
PSTDLIW5	24(018)	
PSTDLIW6	28(01C)	
PSTDLIW7	32(020)	
PSTDLIW8	36(024)	
PSTDLIW9	40(028)	
PSTDLROM	348(15C)	
PSTDLTWA	344(158)	
PSTDMBNM	144(090)	
PSTDREPL	124(07C)	
PSTDSGA	136(088)	
*PSTDUMPI	483(1E3)	80
*PSTENDDA	133(085)	24
*PSTEOD	133(085)	2C
*PSTERASE	132(084)	0A
PSTERCD1	466(1D2)	
PSTERCD2	467(1D3)	
PSTERCOD	466(1D2)	
PSTERDT1	468(1D4)	
PSTERDT2	475(1DB)	
PSTERIND	483(1E3)	
*PSTERMSP	72(048)	80
*PSTEXPAD	254(0FE)	40
*PSTFBSPC	132(084)	04
PSTFNCTN	132(084)	
*PSTFRBLK	133(085)	30
*PSTFRSPC	132(084)	02
*PSTGBSPC	132(084)	03
*PSTGETNX	132(084)	0B
*PSTGTDS	133(085)	04
*PSTGTRAP	132(084)	04
*PSTGTSPC	132(084)	01
PSTGVPL	232(0E8)	
PSTGVWKL	232(0E8)	
*PSTHISMR	464(1D0)	10
*PSTINLD	133(085)	34
*PSTINTNT	68(044)	40
*PSTIOERR	133(085)	08
PSTIQPRM	72(048)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*PSTIWAIT	254(0FE)	80
PSTLIPRM	484(1E4)	
*PSTLODU	464(1D0)	40
*PSTLODUH	464(1D0)	20
PSTLOGQ	436(1B4)	
PSTLOGWA	432(1B0)	
PSTMI	76(04C)	
PSTMROCO	177(0B1)	
*PSTMSPUT	132(084)	0E
*PSTNOERR	176(0B0)	40
PSTNORO	564(234)	
*PSTNOSPC	133(085)	0C
*PSTNOTFD	133(085)	14
*PSTNPLSP	133(085)	1C
PSTNUMRO	252(0FC)	
PSTNUMWT	253(0FD)	
*PSTNWBLK	133(085)	18
*PSTOCALL	132(084)	04
*PSTOCBAD	132(084)	80
*PSTOCCLS	132(084)	00
*PSTOCDCB	132(084)	10
*PSTOCDMB	132(084)	01
*PSTOCDSG	132(084)	40
*PSTOCLD	132(084)	20
*PSTOCOPN	132(084)	08
*PSTOCPCB	132(084)	02
PSTOFFST	134(086)	
*PSTOLTW	68(044)	04
PSTPCPGM	448(1C0)	
PSTPCPSB	456(1C8)	
PSTPCT1	464(1D0)	
PSTPCT2	465(1D1)	
*PSTPGUSR	132(084)	07
*PSTPIPIU	133(085)	80
*PSTPISIU	133(085)	40
*PSTPLI	465(1D1)	01
PSTPLIPR	556(22C)	
PSTPOSEL	176(0B0)	
PSTPREAD	00(00)	
PSTPREAR	168(0A8)	
*PSTPRVWT	68(044)	08
PSTPSB	88(058)	
*PSTPUTKY	132(084)	0D
*PSTQDEQ	132(084)	08
*PSTQENQ	132(084)	08
*PSTQLEO	570(238)	00
PSTQLEV	570(238)	
*PSTQLEXC	570(238)	08
*PSTQLUPD	570(238)	04
*PSTQPUR	132(084)	0C
*PSTQRBDC	133(085)	08
*PSTQRDDL	133(085)	04
*PSTQRNSE	133(085)	10
*PSTQROOP	133(085)	02
*PSTQRWR	133(085)	01
*PSTQVER	132(084)	04
PSTRBAL	202(0CA)	
*PSTRDERR	133(085)	08
PSTRETRE	220(0DC)	
PSTRPSTA	578(240)	
PSTRRDF	570(238)	
PSTRRDL	578(240)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
PSTRTCDE	133(085)	
*PSTSABND	72(048)	20
PSTSAVRE	180(0B4)	
*PSTSCALL	68(044)	80
PSTSCDAD	68(044)	
*PSTSCHED	68(044)	10
PSTSDATA	202(0CA)	
PSTSEG	84(054)	
PSTSEGL	80(050)	
PSTSPL	208(0D0)	
*PSTSTLBG	132(084)	0C
*PSTSTLEQ	132(084)	09
PSTSUBNM	172(0AC)	
PSTSUIN	164(0A4)	
PSTSV1	592(250)	
PSTSV2	664(298)	
PSTSV3	736(2E0)	
PSTSV4	808(328)	
PSTSV5	880(370)	
PSTSV6	952(3B8)	
PSTSV7	1024(400)	
PSTSWI	174(0AE)	
PSTSWKAR	180(0B4)	
PSTSWKL	202(0CA)	
*PSTTABND	72(048)	10
PSTTSKID	256(100)	
*PSTUDR	464(1D0)	04
*PSTULU	464(1D0)	02
PSTUSER	76(04C)	
*PSTUSM	464(1D0)	01
*PSTUST	464(1D0)	08
PSTVLSR	246(0F6)	
PSTVSL	202(0CA)	
*PSTWRITE	132(084)	08
PSTWRKD1	312(138)	
PSTWRKD2	316(13C)	
PSTWRKD3	320(140)	
PSTWRKD4	324(144)	
PSTWRKD5	328(148)	
PSTWRKD6	332(14C)	
PSTWRKD7	336(150)	
PSTWRKT1	292(124)	
PSTWRKT2	296(128)	
PSTWRKT3	300(12C)	
PSTWRKT4	304(130)	
PSTWRKT5	308(134)	
PSTWRK1	276(114)	
PSTWRK1	436(1B4)	
PSTWRK2	280(118)	
PSTWRK3	284(11C)	
PSTWRK4	288(120)	
*PSTWROSI	133(085)	20
*PSTXCONM	465(1D1)	80
*PSTXMDLT	132(084)	A0
*PSTXMISR	132(084)	A2
*PSTXMRPL	132(084)	A1
*PSTXMUNL	132(084)	A3
*PSTXPRTM	465(1D1)	40
PSTXPSV1	260(104)	
PSTXPSV2	264(108)	
PSTXPSV3	268(10C)	

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(000)	4	PSTPREAD		Address of this PST prefix
4(004)	4	PSTDLIW0		Action modules work area HD unload (DLZURGU0) return address for retrieve
(008)	4	PSTDLIW1		Action modules work area
12(00C)	4	PSTDLIW2		Action modules work area
16(010)	4	PSTDLIW3		Action modules work area
20(014)	4	PSTDLIW4		Action modules work area
24(018)	4	PSTDLIW5		Action modules work area
28(01C)	4	PSTDLIW6		Action modules work area
32(020)	4	PSTDLIW7		Action modules work area
36(024)	4	PSTDLIW8		Action modules work area
40(028)	4	PSTDLIW9		Action modules work area
44(02C)	4	PSTDLIWA		Action modules work area
48(030)	4	PSTDLIWB		Action modules work area
52(034)	4	PSTDLIWC		Action modules work area
56(038)	4	PSTDLIWD		Action modules work area
60(03C)	4	PSTDLIWE		Action modules work area
64(040)	4	PSTDLIWF		Action modules work area
USER CALL PROCESSING SECTION				
68(044)	1	PSTCODE1		
		PSTSCALL	80	PST for system call
		PSTINTNT	40	Cannot schedule, intent not satisfied
		PSTSCHED	10	OK to complete scheduling
		PSTPRVWT	08	Logger private wait indicator
		PSTOLTW	04	Another task waiting for resource owned by this task. Note: If PSTINTNT and PSTSCHED are both set, DL/I backout is in control.
68(044)	4	PSTSCDAD		Address of SCD
72(048)	4	PSTABIND		Task/system ABEND indicator
		PSTERMSP	80	PUT error message indicator
		PSTSABND	20	System ABEND indicator bit
		PSTTABND	10	Task ABEND indicator bit
72(048)	4	PSTIQPRM		Address of caller's parameter list
76(04C)	4	PSTMI		Return segment indicator

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
76(04C)	4	PSTUSER		Address of user's I/O area
80(050)	4	PSTSEGL		Retrieved segment length
84(054)	4	PSTSEG		Retrieved segment address
88(058)	4	PSTPSB		PDIR entry address
USER TASK STATISTICS				
92(05C)	4	PSTACCT		
92(05C)	4	PSTDGU		Number of GU calls issued
96(060)	4	PSTDGN		Number of GN calls issued
100(064)	4	PSTDGNP		Number of GNP calls issued
104(068)	4	PSTDGHU		Number of GHU calls issued
108(06C)	4	PSTDGHN		Number of GHN calls issued
112(070)	4	PSTDGHNP		Number of GHNP calls issued
116(074)	4	PSTDISRT		Number of ISRT calls issued
120(078)	4	PSTDDLET		Number of DLET calls issued
124(07C)	4	PSTDREPL		Number of REPL calls issued
ACTION MODULES SECTION				
128(080)	4	PSTDBPCB		Address of current PCB
132(084)	1	PSTFNCTN		Function codes
EQUATES FOR BUFFER HANDLER FUNCTION CODES				
		PSTBKLCT	01	Locate relative block number
		PSTBYLCT	02	If HD, locate relative byte number. If HISAM or HIDAM INDEX, read a record by RBA from a KSDS. If HISAM, read a record by RBA from an ESDS.
		PSTGBSPC	03	Get buffer space
		PSTFBSPC	04	Free buffer space
		PSTBFMPT	04	Mark buffers enpty
		PSTBFALT	05	If HD, mark a buffer containing data altered. If HISAM or HIDAM INDEX, write a record by RBA to a KSDS. If HISAM, write a record by RBA to an ESDS
		PSTBYALT	06	Locate a relative byte number and mark buffer altered
		PSTPGUSR	07	Purge all buffers altered by a task
		PSTWRITE	08	Write a new record to HISAM ESDs
		PSTSTLEQ	09	Read a record by key from a KSDS
		PSTERASE	0A	Erase a record in a KSDS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTGETNX	0B	Read the next record in a KSDS
		PSTSTLBG	0C	Read the record containing the first root in a KSDS
		PSTPUTKY	0D	Insert a record by key into a KSDS
		PSTMSPUT	0E	Insert record(s) sequentially into a KSDS
EQUATES FOR OPEN/CLOSE FUNCTION CODES				
		PSTOCDMB	01	Close DMB. Address of DMB in R2
		PSTOCPCB	02	Close PCB. Address of PCB in R2
		PSTOCALL	04	Close all DMBs
		PSTOCCLS	00	Close call. Bit 4 = 0
		PSTOCOPN	08	Open call. Bit 4 = 1
		PSTOCDCB	10	Open/close the DMB in PSTDCBNM. DSG address in PSTDSGA
		PSTOCLD	20	Open for load
		PSTOCDSG	40	Open the DSG in PSTDSGA
		PSTOCBAD	80	Open unsuccessful
EQUATES FOR SPACE MANAGEMENT FUNCTION CODES				
			80	Backout in control
		PSTGTSPC	01	Get space for segment. R5 contains pointer to PSDB
		PSTFRSPC	02	Free space for segment. R5 contains pointer to PSDB
		PSTBTMPF	03	Do bit map update
		PSTGTRAP	04	Get space close to RAP in PSTBYTNM
EQUATES FOR INDEX MAINTENANCE FUNCTION CODES				
		PSTXMDLT	A0	Perform index maintenance for segment to be deleted
		PSTXMRPL	A1	Perform index maintenance for segment to be replaced
		PSTXMISR	A2	Perform index maintenance for segment to be inserted
		PSTXMUNL	A3	Perform index maintenance for segment to be unloaded
EQUATES FOR PROGRAM ISOLATION FUNCTION CODES				
		PSTQENQ	00	Enqueue (Queueing facility)
		PSTQVER	04	Verify (Queueing facility)
		PSTQDEQ	08	Dequeue (Queueing facility)
		PSTQPUR	0C	Purge (Queueing facility)
133(085)	1	PSTRTCDE		Return codes
EQUATES FOR BUFFER HANDLER RETURN CODES				
		PSTCLOK	00	No error occurred
		PSTGTDS	04	RBN is beyond the end of the data set
		PSTIOERR	08	I/O error

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTRDERR	08	Permanent read error
		PSTNOSPC	0C	No space for adds
		PSTBDCAL	10	Illegal call
		PSTNOTFD	14	No record found (retrieve by key)
		PSTNWBLK	18	New block was created in the buffer pool
		PSTNPLSP	1C	Insufficient space in the buffer pool
		PSTWROSI	20	Size of requested buffer exceeds the size of buffers in any subpool
		PSTENDDA	24	End of data set. No record returned
		PSTBYEND	28	Key or RBA higher than the highest key or RBA in the data set
		PSTEOD	2C	End of data set reached on a request issued by open
		PSTINLD	34	Invalid request during data set loading
SPACE MANAGEMENT RETURN CODES				
		PSTFRBLK	30	Block not used due to distributed free space parameter
		PSTBTMPF	03	Bit map update required
EQUATES FOR PROGRAM ISOLATION RETURN CODES				
		PSTQRWR	01	Wait was required
		PSTQROOP	02	Other owners present
		PSTQRDDL	04	Terminated due to deadlock
		PSTQRBDC	08	Terminated due to bad call
		PSTQRNSE	10	Terminated. Insufficient storage
		PSTPISIU	40	Secondary index updated
		PSTPIPIU	80	Primary index updated
134(086)	2	PSTOFFST		Offset to PSTDATA from start of buffer
136(088)	4	PSTDSGA		Address of DSG portion of the JCB
140(08C)	4	PSTBLKNM		Relative block number
144(090)	2	PSTDMBNM		DMB number
146(092)	1	PSTACBNM		ACB number
147(093)	1			**Reserved**
148(094)	4	PSTBYTNM		RBA or relative record number. High order byte contains X'80' if request is for HISAM ESDS
152(098)	4	PSTDATA		Address of requested data
156(09C)	4	PSTBUFFA		Address of buffer prefix

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
BUFFER HANDLER AND SPACE MANAGEMENT SECTION				
160(0A0)	4	PSTBFUSE		Address of the buffer prefix to be used
164(0A4)	4	PSTSUIN		Address of the subpool information table to be used
168(0A8)	4	PSTPREAR		Beginning address of the buffer prefix area for the subpool information table used
172(0AC)	2	PSTSUBNM		Subpool number used during this call
174(0AE)	2	PSTSWI		Work space
			08	HD write in progress
			04	CI in overflow area full
			80	HISAM ESDS is being processed
			02	Request made to the buffer handler by space management
			FF	Purge buffer request completed
176(0B0)	1	PSTPOSEL		Count for position of use chain element
		PSTNOERR	40	No error message
177(0B1)	1	PSTMROCO		Number of the row/column in the interlock detection matrix currently used by this task
178(0B2)	2			**Reserved**
180(0B4)	40	PSTSAVRE		Work area used by buffer handler when processing a request
THIS AREA IS USED BY DLZDCI00 FOR SHOWCAT AND GETVCE FOR FBA SUPPORT				
180(0B4)	40	PSTSWKAR		SHOWCAT work area used by Space Management DLZGGSP0 and Open/Close DLZDLOC0
202(0CA)		PSTSDATA		Location of needed data returned by SHOWCAT
		PSTRBAL		RBA data length (equated to 4)
		PSTVSL		Volume serial number length (equated to 6)
		PSTSWKL		Length of SHOWCAT work area (equated to 64)
246(0F6)		PSTVLSR		Volume serial number save area
208(0D0)		PSTSPL		SHOWCAT parameter list
232(0E8)		PSTGVPL		GETVCE parameter list

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTGVWKL		Length of GETVCE work area (equated to 52)

THE FOLLOWING FIELDS ARE USED BY DL/I OPEN/CLOSE (DLZDLOC0) AND SPACE MANAGEMENT (DLZDHDS0) FOR VSAM CATALOG PARAMETER LIST WHEN PROCESSING AN OUT-OF-SPACE CONDITION FOR HIDAM DATA BASE

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
180(0B4)	40	PSTCTGPL		Area used as the VSAM catalog parameter list (CTGPL) by DLZGGSP0 and DLZDLOC0 to do locate
		PSTCPLN		Length of CTGPL block (equated to 40)
		PSTCTGNM		Number of CTGFL entries (equated to 1)
220(0DC)	32	PSTRETRE		Buffer handler subroutine linkage register (R14) save area when procssing a request

THE FOLLOWING FIELDS ARE USED BY OPEN/CLOSE (DLZDLOC0) AND SPACE MANAGEMENT (DLZDHDS0) FOR VSAM FIELD PARAMETER LIST WHEN PROCESSING AN OUT-OF-SPACE CONDITION FOR HIDAM DATA BASE

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
220(0DC)	24	PSTCTGFL		Area used as the VSAM field parameter list (CTGFL) by DLZGGSP0 and DLZDLOC0 to do locate
244(0F4)	8	PSTCTGWK		VSAM catalog work area
244(0F4)	3	PSTCTGL1		Catalog work area length 1
247(0F7)	1	PSTCTGL2		Catalog work area length 2
248(0F8)	4	PSTCTGRT		VSAM catalog return area for HI-RBA
		PSTCWKLN		Length of catalog work area (equated to 8)

BUFFER HANDLER STATISTICS

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
252(0FC)	1	PSTNUMRO		Number of blocks read on this call
253(0FD)	1	PSTNUMWT		Number of writes issued on this call
254(0FE)	1	PSTCLRWT		Buffer handler switch
		PSTIWAIT	80	IWAIT issued during this call
255(0FF)	1			**Reserved**
256(100)	4	PSTTSKID		Hashed task ID. High-order byte, binary date. Low-order three bytes, assigned in ascending sequence

THE FOLLOWING FIELDS ARE USED AS SAVE AREAS SO THAT THE DMB ECB CAN BE POSTED IF THE TASK IS CANCELED WHILE WAITING FOR I/O COMPLETION

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
260(104)	4	PSTXPSV1		User VSAM save area address

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
264(108)	4	PSTXPSV2		EXCPAD return address
268(10C)	4	PSTXPSV3		EXCPAD parameter list address
272(110)	4			**Reserved**
PST WORK AREAS				
276(114)	4	PSTWRK1		PSTWRKn are work words for buffer handler (DLZDBH00) and data base logger.
280(118)	4	PSTWRK2		
284(11C)	4	PSTWRK3		
288(120)	4	PSTWRK4		
292(124)	4	PSTWRKT1		PSTWRKn is work space preserved across calls to the buffer handler.
296(128)	4	PSTWRKT2		
300(12C)	4	PSTWRKT3		
304(130)	4	PSTWRKT4		
308(134)	4	PSTWRKT5		
THE HIGH-ORDER BYTE OF PSTWRKT4 IS USED TO PASS THE FOLLOWING FUNCTION CODES TO INDEX MAINTENANCE				
			04	Reinsert index
			03	Secondary indexes only
			02	Primary indexes only
			01	Both primary and secondary indexes
312(138)	4	PSTWRKD1		PSTWRKDn is work space for use by DELETE/REPLACE, RETRIEVE, and LOAD/INSERT.
316(13C)	4	PSTWRKD2		
320(140)	4	PSTWRKD3		
324(144)	4	PSTWRKD4		
328(148)	4	PSTWRKD5		
332(14C)	4	PSTWRKD6		
336(150)	4	PSTWRKD7		
340(154)	4	PSTCURWA		Current delete work area
344(158)	4	PSTDLTWA		First delete work area address
348(15C)	84	PSTDLROM		Save and maintenance work area for retrieve
DATA BASE LOG SECTION				
432(1B0)	4	PSTLOGWA		Work area address for log O/P

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
436(1B4)	4	PSTLOGQ		Address of reuse queue QCB in pool
DATA BASE LOG USE OF PSTWRK1				
		PSTWRK1		Physical SDB address. If new block, low-order 2 bytes are call count. High-order byte used for function code
DATA BASE LOG FUNCTION CODES				
		DBLNDXC	80	Index maintenance call
		DBLCMC	00	Bits 1-3 = 0, chain maintenance call
		DBLNTCR	70	Counter maintenance
		DBLLGDLT	60	Delete byte maintenance
		DBLPHYI	40	Insert
		DBLPHYD	20	Physical delete
		DBLPHYR	10	Replace
		DBLOOPS	0A	No data. End of user call
		DBLLASTC	08	Last change for user call
		DBLFSE1	00	Bit 5 = 0, one FSE (if bits 1 or 2 on)
		DBLFSE2	04	Two FSEs (if bits 1 or 2 on)
		DBLPHYRO	02	Qld copy of a replace
		DBLNEWBL	01	New block log call
DATA BASE LOG USE OF PSTWRK2 - PSTWRK4				
Chain maintenance - Old copy of chain pointer (4 bytes). Insert/Delete - Offset and new FSEs (6 or 12 bytes)				
440(1B8)	8			**Reserved**
PARTITION/TASK INFORMATION				
448(1C0)	8	PSTPCPGM		Application program name. If batch UDR, ULR,or ULU; DBD name
456(1C8)	8	PSTPCPSB		PSB name
464(1D0)	1	PSTPCT1		Partition/task option
		PSTBATCH	80	PST is in batch partition
		PSTLODU	40	Load utility
		PSTLODUH	20	Load HDAM DB
		PSTHISMR	10	HISAM data base recovery in process
		PSTUST	08	Statistics utility
		PSTUDR	04	Data base recovery utility
		PSTULU	02	Data base load/unload utility
		PSTUSM	01	Security maintenance utility
465(1D1)	1	PSTPCT2		Program options/information overlaid on every call to the batch program request handler
		PSTXCONM	80	Exclude console message
		PSTXPRTM	40	Exclude printer message
		PSTCHKP	04	User checkpoint call successful
		PSTCALI	02	User's call list is implicit

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
		PSTPLI	01	User program is PL/I
466(1D2)	1	PSTERCOD		Error message codes
466(1D2)	1	PSTERCD1		Error message code byte one
467(1D3)	1	PSTERCD2		Error message code byte two
468(1D4)	7	PSTERDT1		Error message data for ACB or DTF name
475(1DB)	6	PSTCNVB		Doubleword for HD randomizing module
475(1DB)	6	PSTERDT2		Variable error message data
483(1E3)	1	PSTERIND		Error routine indicator
		PSTDUMPI	80	Issue dump after error message put
		PSTCANLI	40	Issue cancel after error message put
484(1E4)	72	PSTLIPRM		Area to build user parameter list and register save area for MPS start and stop calls
556(22C)	8	PSTPLIPR		PL/I region STXIT processor
564(234)	6	PSTNORO		Number of owned resources
570(238)	0	PSTQLEV		Queue request level
		PSTQLEO	00	Read only level
		PSTQLUPD	04	Update level
		PSTQLEXC	08	Exclusive level
570(238)	4	PSTRRDF		Pointer to first RRD
574(23C)	4	PSTRRDL		Pointer to last RRD
578(240)	4	PSTRPSTA		Remote PST (RPST) address. Contains 0 if not scheduled to a remote PSB.
590(244)	12			**Reserved**
REGISTER SAVE AREA				
592(250)	72	PSTSV1		PSTSV1 through PSTSV7 are seven register save areas required for processing DL/I user calls. The convention used in storing registers in these save areas is to begin with R14 and end with R12; that is, R14, R15, R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, and R12.
664(298)	72	PSTSV2		
736(2E0)	72	PSTSV3		
808(328)	72	PSTSV4		
880(370)	72	PSTSV5		
952(3B8)	72	PSTSV6		
1024(400)	72	PSTSV7		

QWA - QUEUING FACILITY WORK AREA

DSECT Name: DLZQWA

The QWA contains information used by the queuing facility module to build control blocks and RDB queue headers. It also contains information used to locate the proper RDB for a particular resource ID.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
QWACPP	16(10)	
*QWADDDF	20(14)	01
QWAFLG1	20(14)	
QWAFLG2	21(15)	
QWAFLG3	22(16)	
QWAFLG4	23(17)	
QWAFPP	12(0C)	
QWAHMLT	32(20)	
QWANCQH	28(1C)	
*QWANPOF	20(14)	02
QWARDBQH	36(24)	
QWAWFD	24(18)	

RECORD LAYOUT - QWA

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	14			Module ID
14(0E)	4	QWAFPP		First page pointer for free block management
18(12)	4	QWACPP		Current page pointer for free block management
22(16)	1	QWAFLG1		First flag byte
		QWADDDF	01	Do deadlock detection
		QWANPOF	02	New Prime owner exists.
23(17)	1	QWAFLG2		Second flag byte
24(18)	1	QWAFLG3		Third flag byte
25(19)	1	QWAFLG4		Fourth flag byte
26(1A)	4	QWAWFD		Work field 1
30(1E)	4	QWANOQH		Number of queue heads
34(22)	4	QWAHMLT		Hashing Multiplier
38(26)	4	QWARDBQH		RDB chain queue headers(one FW/entry)

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.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
RDBLEN	24(18)	
RDBMAXL	8(08)	
RDBNOWN	12(0C)	
RDBPOID	0(00)	
RDBRDBB	4(04)	
RDBRDBF	0(00)	
RDBRID	16(10)	
RDBRRDF	8(08)	
RDBRRDL	12(0C)	
RDBUOID	4(04)	

RECORD LAYOUT - RDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	RDBPOID		Primary owner PST prefix number
0(00)	4	RDBRDBF		RDB forward chain pointer
4(04)	1	RDBUOID		Update owner PST prefix number
4(04)	4	RDBRDBB		RDB backward chain pointer
8(08)	1	RDBMAXL		Maximum level of current owners
8(08)	4	RDBRRDF		Pointer to first RRD
12(0C)	1	RDBNOWN		Current number of owners
12(0C)	4	RDBRRDL		Pointer to last RRD
16(10)	7	RDBRID		Resource ID
23(17)	1			**Reserved**
24(18)		RDBLEN		Length of RDB

RIB - REMOTE INTERFACE BLOCK

DSECT Name: DLZRIB

This DSECT describes remote interface block fields. The RIB is used by DL/I for CICS/VS intersystem communication (ISC) support. It defines fields passed between CICS/VS and DL/I.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*RIBBUFAL	18(12)	40
*RIBCALL	20(14)	40
RIBCHAIN	4(04)	
RIBDLTR	22(16)	
RIBFCTR	21(15)	
*RIBFUNC	20(14)	80
RIBINDEX	16(10)	
RIBIOAWK	8(08)	
RIBISC	18(12)	
RIBISCI	20(14)	
RIBISCO	19(13)	
RIBLEN	24(18)	
*RIBLNKA	20(14)	20
*RIBLNKSH	20(14)	10
RIBPCBAL	0(00)	
*RIBPCBM	18(12)	80
RIBRSET	23(17)	
*RIBSYNC	19(13)	80
RIBUPPER	12(0C)	

RECORD LAYOUT - RIB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RIB		Start of RIB DSECT. This control block follows immediately after the RPST.
0(00)	4	RIBPCBAL		Local PCB address list.
4(04)	4	RIBCHAIN		Remote PSB storage chain.
8(08)	4	RIBIOAWK		Local PSB I/O work area.
12(0C)	4	RIBUPPER		Highest address of caller partition.
16(10)	2	RIBINDEX		PCB index number.
18(12)	1	RIBISC		ISC scheduling duration flags:
		RIBPCBM	80	PCBM scheduling call issued.
		RIBBUFAL	40	RIBIOAWK buffer allocated.
19(13)	1	RIBISCO		ISC outbound flags:
		RIBSYNC	80	Synchronization point issued.
20(14)	1	RIBISCI		ISC inbound flags:
		RIBFUNC	80	Function string invalid.
		RIBCALL	40	User call parameter list invalid.
		RIBLNKNA	20	Link does not exist.
		RIBLNKSH	10	Link is out of service.
21(15)	1	RIBFCTR		ISC response code.
22(16)	1	RIBDLTR		Additional response information.
23(17)	0	RIBRSET		Length of function dependent flags.
23(17)	1	Unnamed		**Reserved**
24(18)		RIBLEN		Length of RIB.

RPCB - REMOTE PCB

DSECT Name: DLZRPCB

This DSECT describes remote PCB fields. The RPCB is an extension of PCB local storage used by DL/I for CICS/VS intersystem communication (ISC) support. RPCBs exist only while a task is scheduled for a data base that is located on some other system. In this case, the address of the RPCB is located four bytes ahead of the PCB.

RECORD LAYOUT - RPCB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPCB		Start of RPCB DSECT.
0(00)	4	RPCBMIOS		Maximum PCB I/O area size.
4(04)	4	RPCBSEGL		Length of last retrieve.
8(08)	1	RPCBFLAG		Flag byte:
		RPCBPATH	80	Previous get hold path call.
9(09)	3	Unnamed		**Reserved**
12(0C)		RPCBLEN		Length of RPCB

RPDIR - REMOTE PSB DIRECTORY

DSECT Name: DLZRPDIR

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

RECORD LAYOUT - RPDIR

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPDIR		Start of RPDIR DSECT
0(00)	4	RPDIRSYS		System name on which remote PSB is defined.
4(04)	8	RPDIRPSB		Name of PSB to use on remote system.
12(0C)		RPDIRLEN		Length of RPDIR

RPST - REMOTE PST

DSECT Name: DLZRPST

This DSECT describes remote PST fields. The RPST is an extension of task local storage used by DLZODP for CICS/VS intersystem communication (ISC) support.

RECORD LAYOUT - RPST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	RPST		Start of RPST DSECT.
0(00)	4	RPSTISC1		ISC parameter 1.
4(04)	4	RPSTISC2		ISC parameter 2.
8(08)	4	RPSTISC3		ISC parameter 3.
12(0C)	4	RPSTISC4		ISC parameter 4.
16(10)	4	RPSTISC5		ISC parameter 5.
20(14)	4	RPSTISC6		ISC parameter 6.
24(18)	1	RPSTATUS		Flag byte.
25(19)	3	RPSTACTA		Program's ACT entry address.
28(1C)		RPSTLEN		Length of RPST.

RRD - RESOURCE REQUEST DESCRIPTOR

DSECT Name: DLZRRD

The RRD (Resource Request Descriptor) is used to maintain a record of all the requests and their current status by one task for a particular resource.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
RRDFLAG	16(10)	
RRDLEN	18(24)	
RRDMAXL	12(0C)	
RRDNQEX	8(08)	
RRDNQRO	0(00)	
RRDNQUP	4(04)	
*RRDOWNF	16(10)	01
*RRDPOWNF	16(10)	04
RRDPSTP	16(10)	
RRDPSTQB	4(04)	
RRDPSTQF	0(00)	
RRDRDBP	20(14)	
RRDRDBQB	12(0C)	
RRDRDBQF	8(08)	
*RRDWAITF	16(10)	02

RECORD LAYOUT - RRD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	RRDNQRO		Number of read-only ownerships for task
0(00)	4	RRDPSTQF		PST queue forward pointer; next RRD for task
4(04)	1	RRDNQUP		Number of exclusive (update) ownerships for task
4(04)	4	RRDPSTQB		PST queue backward pointer; prior RRD for task
8(08)	1	RRDNQEX		Number of exclusive ownerships for task
8(08)	4	RRDRDBQF		RDB queue forward pointer; next RRD for resource
12(0C)	1	RRDMAXL		Current maximum ownership level for resource by task
12(0C)	4	RRDRDBQB		RDB queue backward pointer; prior RRD for resource
16(10)	1	RRDFLAG		Flag byte
		RRDOWNF	01	PST owns resource
		RRDWAITF	02	PST is waiting for resource
		RRDPOWNF	04	PST is prime owner of resource
16(10)	4	RRDRDBP		RDB address for resource
20(14)	4	RRDPSTP		PST address for task
24(18)	4	RRDLEN		Length of RRD

SBIF - SUBPOOL INFORMATION TABLE

DSECT Name: SUBINFTA

The subpool information table is described as part of the general structure and description of DL/I buffer pool control blocks. There is one subpool information table for each subpool allocated.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SUBBFHD	3(03)	
SUBBFNO	2(02)	
SUBBFSIZ	44(2C)	
SUBDMBCT	45(2D)	
*SUBFRSV	3(03)	80
SUBLEN	46(2E)	
SUBNQFI	0(00)	
SUBNQLA	1(01)	
SUBUCHAI	8(08)	
SUBUCPRE	4(04)	
SUBUCSUF	40(28)	

RECORD LAYOUT - SBIF

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	SUBNQFI		PST prefix number of first task in chain for enqueue subpool
1(01)	1	SUBNQLA		PST prefix number of last task in chain for enqueue subpool
2(02)	1	SUBBFNO		Number of buffers in this subpool
3(03)	1	SUBBFHD		HDBFR indicator
		SUBFRSV	80	DMB assigned to this subpool by HDBFR parameter
4(04)	4	SUBUCPRE		Accumulated number of buffers in preceeding subpools
8(08)	32	SUBUCHAI		Buffer use chain
40(28)	4	SUBUCSUF		(Not used in DL/I DOS/VS)
44(2C)	1	SUBBFSIZ		Size of the buffers in this subpool: X'01' = 512 bytes X'02' = 1024 bytes X'03' = 1536 bytes X'04' = 2048 bytes X'05' = 2560 bytes X'06' = 3072 bytes X'07' = 3584 bytes X'08' = 4096 bytes
45(2D)	1	SUBDMBCT		Number of DMBs assigned
46(2E)	0	SUBLEN		Length of subpool information table

SCD - SYSTEM CONTENTS DIRECTORY

DSECT Name: DLZSCD

The DL/I SCD (System Contents Directory) is produced during DL/I system definition for online CICS/VS-DL/I. The SCD is preassembled as part of the DL/I nucleus in the batch DL/I system. The SCD contains major entry pointers for all DL/I facilities.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
CPYRITE	0(00)	
SCD	96(60)	
SCDABEND	200(C8)	
SCDABSAV	288(120)	
SCDACTBA	264(108)	
SCDASE	196(C4)	
SCDATSKC	106(6A)	
SCDBFPL	216(D8)	
SCDBKWRK	352(160)	
SCDCDTA	268(10C)	
SCDCMTCT	384(180)	
*SCDCMTI	284(11C)	40
SCDCMXT	104(68)	
SCDCOMRG	124(7C)	
SCDCPY10	180(B4)	
SCDCSABA	276(114)	
SCDCWRK	336(150)	
SCDCWRKL	340(154)	
SCDDATE	98(62)	
*SCDDBASL	346(15A)	02
SCDDBFA	217(D9)	
SCDDBFPL	216(D8)	
SCDDBLAS	324(144)	
*SCDDBLCJ	346(15A)	20
SCDDBLCL	320(140)	
*SCDDBLD2	346(15A)	10
SCDDBLFW	316(13C)	
SCDDBMPS	304(130)	
SCDDBLNT	148(94)	
*SCDDBLO	346(15A)	80
SCDDBLOP	346(15A)	
*SCDDBLOR	346(15A)	40
*SCDDBLSP	346(15A)	08
SCDDBLSV	328(148)	
*SCDDBLTD	346(15A)	20
SCDDBLWO	332(14C)	
SCDDDBH0	136(88)	
*SCDDELT	284(11C)	20
SCDDHDS0	160(A0)	
*SCDDLARE	144(90)	28
SCDDLICL	168(A8)	
SCDDLICT	144(90)	
SCDDLIDL	232(E8)	
SCDDLIDM	228(E4)	
SCDDLIDN	234(EA)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
SCDDLIDR	152(98)	
SCDDLIIN	156(9C)	
SCDDLIM	97(61)	
SCDDLIPL	224(E0)	
SCDDLIPN	226(E2)	
SCDDLIPS	220(DC)	
SCDDLIRE	140(8C)	
SCDDLIS	272(110)	
SCDDLIUP	276(114)	
SCDDLIV	96(60)	
SCDDLOCT	380(17C)	
SCDDSEHO	172(AC)	
SCDDXMTO	164(A4)	
SCDERRMS	192(C0)	
SCDEXTBA	300(12C)	
SCDFLPC	244(F4)	
*SCDFLSAV	244(F4)	40
*SCDHLRE	284(11C)	08
SCDIWAIT	188(BC)	
*SCDLIPLI	244(F4)	80
SCDLNGTH	392(188)	
SCDLOCOU	348(15C)	
SCDLOWER	108(6C)	
SCDLOWID	120(78)	
SCDLSTAD	292(124)	
SCDMPCPT	296(128)	
*SCDMTI	284(11C)	80
SCDMXTSK	102(66)	
*SCDNABND	284(11C)	01
SCDNAVID	116(74)	
*SCDNDMP	284(11C)	04
*SCDNJNL	284(11C)	01
*SCDNLOGI	284(11C)	02
SCDNTWC	286(11E)	
SCDPDUP	388(184)	
*SCDPI	304(130)	40
SCDPPAB	248(F8)	
SCDPPAF	244(F4)	
SCDPPFB	256(100)	
SCDPPFF	252(FC)	
SCDPPSTL	240(F0)	
SCDPPSTN	242(F2)	
SCDPPSTS	236(EC)	
SCDPRHED	132(84)	
SCDPSTLN	260(104)	
*SCDQFJRN	172(AC)	08
*SCDQFSDC	172(AC)	04
SCDQUEFW	176(B0)	
SCDQUEFO	172(AC)	
SCDREENT	312(138)	
*SCDRELOD	285(11D)	08
SCDREPLN	344(158)	
*SCDRLABN	285(11D)	04
*SCDRLRST	285(11D)	10
*SCDRPSB	304(130)	20
SCDSEQ	342(156)	
SCDSIND	284(11C)	
SCDSIND2	285(11D)	
*SCDSOPLG	285(11D)	01
SCDSPCNT	282(11A)	
*SCDSYACT	285(11D)	40
*SCDSYINT	285(11D)	02

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*SCDSYSAB	285(11D)	80
*SCDSYWAT	285(11D)	20
*SCDTAMOD	368(170)	40
*SCDTBHCL	368(170)	02
*SCDTCPOS	368(170)	10
*SCDTINDX	368(170)	01
SCDTKCNT	280(118)	
SCDTKTRM	204(CC)	
*SCDTOLBH	369(171)	80
*SCDTPITR	369(171)	40
SCDTRACE	356(164)	
SCDTRCNM	360(168)	
*SCDTRETR	368(170)	20
SCDTRFL1	368(170)	
SCDTRFL2	369(171)	
SCDTSKCR	372(174)	
*SCDTUSER	368(170)	80
*SCDTVSAM	368(170)	04
*SCDTWFI	284(11C)	08
*SCDUPD	284(11C)	10
SCDUPPER	112(70)	
SCDUSAVE	244(F4)	
SCDWAIT	262(106)	
*SCDXECB	304(130)	80

RECORD LAYOUT - SCD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	96	CPYRITE		Reserved for copyright information
96(60)	0	SCD		Start of addressable SCD
SYSTEM CONFIGURATION SECTION				
96(60)	1	SCDDLIV		DL/I version number
97(61)	1	SCDDLIM		DL/I release level
98(62)	4	SCDDATE		System date - Julian
102(66)	2	SCDMXTSK		DL/I minimum task count - online
104(68)	2	SCDCMXT		DL/I current maximum task - online
106(6A)	2	SCDATSKC		Active DL/I task counter - online
108(6C)	4	SCDLOWER		Partition lower boundary; address pointer to addressable part of the SCD (batch only)
112(70)	4	SCDUPPER		Partition upper boundary address
116(74)	4	SCDNAVID		Next available task ID
120(78)	4	SCDLOWID		Lowest task ID
124(7C)	4	SCDCOMRG		COMREG address
128(80)	4			**Reserved**
ACTION MODULE ENTRY POINT ADDRESSES				
132(84)	4	SCDPRHED		Entry point of program request handler: Batch = DLZPRHB0 Online = DLZPRH00
136(88)	4	SCDDDBH0		Entry point of buffer handler (DLZDBH00)
140(8C)	4	SCDDLIRE		Entry point of retrieve (DLZDLR00)
144(90)	4	SCDDLICT		Entry point of call analyzer (DLZDLA00)
		SCDDLARE	28	Offset to entry point on return to call analyzer
148(94)	4	SCDDBLNT		Entry point of data base log module (DLZRDBL0) = entry

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				point of log initialization until after initialization
152(98)	4	SCDDLID2		Entry point of delete/replace (DLZDLD00)
156(9C)	4	SCDDLIIN		Entry point of load/insert for retrieve (DLZDDLE0)
160(A0)	4	SCDDHDS0		Entry point of space management (DLZDHDS0)
164(A4)	4	SCDDXMT0		Entry point of index maintenance (DLZDXMT0)
168(A8)	4	SCDDLICL		Entry point of open/close (DLZDLOC0)
172(AC)	4	SCDDSEH0		Entry point of routine to create work files for batch only (DLZDSEH0)
172(AC)	4	SCDQUEF0		Entry point of enqueue/dequeue module for program isolation - online only (DLZQUEF0)
		SCDQFSDC	04	Displacement to SCD address field in DLZQUEF0
		SCDQFJRN	08	Displacement to JRNAD exit address field in DLZQUEF0
176(B0)	4	SCDQUEFW		Enqueue/dequeue work area
180(B4)	4	SCDCPY10		Entry point for field level sensitivity expansion routine (DLZCPY10)
184(B8)	4			**Reserved**
188(BC)	4	SCDIWAIT		Entry point of IWAIT routine: Batch = DLZIWAIT Online = DLZOWAIT
192(C0)	4	SCDERRMS		Entry point of error message routine: Batch = ERRORMSG Online = DLZERMSG
196(C4)	4	SCDASE		Entry point of online schedule and termination (DLZSCHDL)
200(C8)	4	SCDABEND		Entry point of DL/I ABEND routine: Batch = DLZABEND Online = DLZABND0
204(CC)	4	SCDTKTRM		Entry point of online task termination for program request handler (DLZTKTRM)
208(D0)	8			**Reserved**

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
SYSTEM CONTROL BLOCK SECTION				
216(D8)	0	SCDDBFPL		Label for buffer handler
216(D8)	1	SCDBFPL		Number of buffer subpools
217(D9)	3	SCDDBFA		Address of buffer pool control block prefix (DLZBFPL)
220(DC)	4	SCDDLIPS		Address of PSB directory (DLZPDIR)
224(E0)	2	SCDDLIPL		Length of PDIR entries
226(E2)	2	SCDDLIPN		Number of PDIR entries
228(E4)	4	SCDDLIDM		Address of DMB directory (DLZDDIR)
232(E8)	2	SCDDLIDL		Length of DDIR entries
234(EA)	2	SCDDLIDN		Number of DDIR entries
236(EC)	4	SCDPPSTS		Address of PST prefix entries (DLZPPST)
240(F0)	2	SCDPPSTL		Length of PPST entries
242(F2)	2	SCDPPSTN		Number of PPST entries
244(F4)	4	SCDPPAF		Online forward PST prefix active pointer
244(F4)	4	SCDUSAVE		Used for MPS or batch. Contains address of user savearea where DL/I registers are saved.
244(F4)	1	SCDFLPC		Flag byte (used for MPS or batch):
		SCDLIPLI	80	0 = Currently executing in DL/I code (or in a user program that is not written in PL/I). 1 = Currently executing in PL/I code.
		SCDFLSAV	40	0 = User savearea used for STXIT PC. 1 = DL/I savearea used for STXIT PC.
248(F8)	4	SCDPPAB		Online backward PST prefix active pointer
252(FC)	4	SCDPPFF		Online forward PST prefix free pointer (DLZPPSTF)
256(100)	4	SCDPPFB		Online backward PST prefix free pointer (DLZPPSTE)
260(104)	2	SCDPSTLN		Length of PST

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
262(106)	2	SCDWAIT		Number of tasks waiting for CMAX
264(108)	4	SCDACTBA		Address of online application program control table (DLZACTBA)
268(10C)	4	SCDCDTA		Address of current online dispatched task's TCA
272(110)	4	SCDDLIS		Address of first online task suspended
276(114)	4	SCDDLIUP		Address of batch DL/I upper boundary
276(114)	4	SCDCSABA		Address of online CICS CSA
280(118)	2	SCDTKCNT		Count of DL/I tasks assigned PPST
282(11A)	2	SCDSPCNT		Count of suspended tasks due to maximum task
284(11C)	1	SCDSIND		System indicator
		SCDMTI	80	DL/I Maximum task indicator
		SCDCMTI	40	DL/I current maximum task indicator
		SCDDELT	20	Online indicator for PSB has delete sensitivity
		SCDUPD	10	Online indicator for PSB has update sensitivity
		SCDTWFI	08	Task waiting for segment intent
		SCDHLRE	08	High level language reentry indicator STXIT
		SCDNDMP	04	No dump at ABEND
		SCDNLOGI	02	No data base logging to be done
		SCDNABND	01	Batch - no STXIT ABEND to be issued
		SCDNJNL	01	Online - no CICS journal in use
285(11D)	1	SCDSIND2		System flags
		SCDSYSAB	80	System ABEND online
		SCDSYACT	40	System task active
		SCDSYWAT	20	System task waiting
		SCDRLRST	10	HD reload/restart
		SCDRELOD	08	HD reload utility
		SCDRLABN	04	HD reload or reload/restart ABEND is in process
		SCDSYINT	02	Initialization bit
		SCDSOPLG	01	Open records written to CICS journal
286(11E)	2	SCDNTWC		Segment intent wait counter
288(120)	4	SCDABSAV		Pointer to pseudo ABEND save area (DLZABSAV)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
292(124)	4	SCDLSTAD		Address of CICS interface address list (DLZDLIAL)
296(128)	4	SCDMPCPT		Address of MPC partition table
300(12C)	4	SCDEXTBA		Pointer to SCD extension
304(130)	1	SCDDBMPS		Flag Byte
		SCDXECB	80	XECBs defined by MPC
		SCDPI	40	Program isolation active.
		SCDRPSB	20	Remote PSB defined.
305(131)	1			**Reserved**
306(132)	2			**Reserved**
308(134)	4			**Reserved**
DATA BASE CHANGE LOG SECTION				
312(138)	4	SCDREENT		Entry point of log write only
316(13C)	4	SCDDBLFW		Entry point of log force write
320(140)	4	SCDDBLCL		Entry point of log close routine
324(144)	4	SCDDBLAS		Entry point of asynchronous log
328(148)	4	SCDDBLSV		Entry point of log save area
332(14C)	4	SCDDBLWO		Entry point of write log open record
336(150)	4	SCDCWRK		Address of DB log work area
340(154)	2	SCDCWRKL		Length of DB log work area
342(156)	2	SCDSEQ		DB log sequence number
344(158)	2	SCDREPLN		Length of DB log prefix
346(15A)	1	SCDDBLOP		Data base log option byte
		SCDDBLO	80	DB log is open
		SCDDBLOR	40	DB log open required
		SCDDBLTD	20	Disk logging used
		SCDDBLD2	10	Two disk extents used
		SCDDBLSP	08	Pause before extent switch
		SCDDBLCJ	04	CICS journal in use
		SCDDBASL	02	DB asynchronous log required
347(15B)	1			**Reserved**
348(15C)	2	SCDLOCOU		Current log count
350(15E)	2			**Reserved**
352(160)	4	SCDBKWRK		Backout log workarea pointer.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
TRACE SECTION				
356(164)	4	SCDTRACE		Entry point of trace module if present
360(168)	8	SCDTRCNM		Name of trace module
368(170)	1	SCDTRFL1		Trace option byte 1
		SCDTUSER	80	User call interface
		SCDTAMOD	40	Action module trace
		SCDTRETR	20	Retrieve (for GET calls)
		SCDTCPOS	10	Current position information
		SCDTVSAM	04	VSAM interface
		SCDTBHCL	02	Buffer handler interface
		SCDTINDX	01	Requests to index maintenance
369(171)	1	SCDTRFL2		Trace option byte 2
		SCDTOLBH	80	Online trace
		SCDTPITR	40	Program isolation trace
370(172)	2			**Reserved**
STATISTICS SECTION			(Online only)	
372(174)	8	SCDTSKCT		Total number of PSB scheduling calls
380(17C)	4	SCDDLOCT		Program isolation deadlock occurrence count
384(180)	4	SCDCMTCT		Number of times at current maximum task
388(184)	4	SCDPDUP		Number of duplicate PSBs created
392(188)		SCDLNGTH		Length of SCD

SCDEXT - SCD EXTENSION

DSECT Name: SCDEXTDS

The SCD extension is generated in the same manner as the SCD (system contents directory) and is a logical extension of it.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
SCDAPSTR	24(18)		Batch usage
SCDEABEX	4(04)		Batch usage
SCDEABSV	8(08)		Batch usage
SCDEFECB	8(08)		Online usage
SCDEIDNX	24(18)		Online usage
SCDEIDST	20(14)		Online usage
SCDEIDWK	28(1C)		Online usage
SCDELECB	0(00)		Online usage
SCDEMSGT	32(20)		Online usage
SCDEPASS	16(10)		Online usage
SCDEPCEX	12(0C)		Batch usage
SCDEREEN	0(00)		Batch usage
SCDESECB	4(04)		Online usage
SCDETRAN	16(10)		Batch usage
SCDETRSV	20(14)		Batch usage
SCDETRTB	36(24)		Online usage and batch usage
SCDETRTE	40(28)		Online usage and batch usage
SCDETRTS	44(2C)		Online usage and batch usage
SCDEVSEX	12(0C)		Online usage
SCDEXLEN	52(34)		Online usage

RECORD LAYOUT - SCDEXT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
Online Usage of the SCD Extension				
0(00)	4	SCDELECB		Logger I/O ECB
4(04)	4	SCDESECB		System enqueue ECB
8(08)	4	SCDEFECB		System function call ECB
12(0C)	4	SCDEVSEX		Address of VSAM EXCP exit (DLZOVSEX)
16(10)	4	SCDEPASS		Address of system password (DLZPASS)
20(14)	4	SCDEIDST		Address of first PPST ID assigned (DLZIDLST)
24(18)	4	SCDEIDNX		Address of last active PPST ID (DLZIDLST)
28(1C)	4	SCDEIDWK		Address of PPST search table (DLZIDWRK)
32(20)	4	SCDEMSGT		Address of online message module (DLZMMSGT)
36(24)	4	SCDETRTB		Current entry in incore table
40(28)	4	SCDETRTE		End address +1 of trace table
44(2C)	4	SCDETRTS		Start address of trace table
48(30)	4			**Reserved**
52(34)		SCDEXLEN		Length of SCD extension
Batch Usage of SCD Extension				
0(00)	4	SCDEREEN		Address of utility block call entry point
4(04)	4	SCDEABEX		Address of STXIT ABEND routine (DLZAABND)
8(08)	4	SCDEABSV		Address of STXIT ABEND save area
12(0C)	4	SCDEPCEX		Address of STXIT PC routine (DLZPABND)
16(10)	4	SCDETRAN		Address of ABTERM transient area
20(14)	4	SCDETRSV		Address of transient save area
24(18)	4	SCDAPSTR		Application program start address

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
28(1C)	8			(Not used in batch)
36(24)	4	SCDETRTB		Current entry in incore table
40(28)	4	SCDETRTE		End address +1 of trace table
44(2C)	4	SCDETRTS		Start address of trace table
48(30)	4			**Reserved**

SDB - SEGMENT DESCRIPTION BLOCK

DSECT Name: SDB

The segment description block (SDB) is described as part of the general structure and description of the program specification block (PSB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
*SDBALTSC	11(0B)	20
*SDBALTSQ	11(0B)	40
*SDBCISP	11(0B)	04
*SDBCTR	37(25)	80
*SDBDCHG	11(0B)	01
SDBDDIR	12(0C)	
*SDBDPAR	11(0B)	10
SDBDSGA	28(1C)	
SDBEND	60(3C)	
*SDBFLS	56(38)	02
SDBF3	10(0A)	
SDBF4	11(0B)	
*SDBGEN	32(20)	10
SDBKEYFD	40(28)	
SDBKEYLN	24(18)	
*SDBLCH	11(0B)	01
SDBLEN	60(3C)	
SDBLEVEL	8(08)	
*SDBLP	37(25)	02
*SDBLTPK	37(25)	04
*SDBLTFD	37(25)	08
SDBLTN	0(00)	
SDBLTP	0(00)	
SDBNSDB	16(10)	
*SDBORGHD	9(09)	20
*SDBORGHI	9(09)	10
*SDBORGHS	9(09)	02
*SDBORGH1	9(09)	04
SDBORGN	9(09)	
*SDBORGRI	9(09)	44
*SDBORGSH	9(09)	05
*SDBORGSS	9(09)	01
SDBPARA	24(18)	
SDBPCB	39(27)	
SDBPCF	38(26)	
*SDBPCTSP	32(20)	40
SDBPHYCD	12(0C)	
SDBPOSC	48(30)	
*SDBPOSL	11(0B)	02
SDBPOSN	52(34)	
SDBPOSP	44(2C)	
*SDBPP	37(25)	10
*SDBPPST	32(20)	80
*SDBPPTSP	32(20)	C0
SDBPSDB	20(14)	
*SDBPTB	37(25)	20
SDBPTDS	37(25)	

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*SDBPTF	37(25)	40	
SDBPTNO	36(24)		
*SDBSEND	10(0A)	10	
*SDBSENG	10(0A)	80	
*SDBSENI	10(0A)	40	
*SDBSENK	10(0A)	08	
*SDBSENL	10(0A)	01	
*SDBSENP	10(0A)	04	
*SDBSENR	10(0A)	20	
*SDBSENX	10(0A)	02	
*SDBSLC	32(20)	02	
*SDBSLP	32(20)	01	
*SDBSNX	32(20)	04	
*SDBSPP	32(20)	08	
SDBSYM	0(00)		
SDBTARG	33(21)		
SDBTFLG	32(20)		
SDBXFFSB	16(10)		(See SDBXP block at end of SDB)
SDBXFISL	6(06)		(See SDBXP block at end of SDB)
SDBXFL	56(38)		
SDBXFLAG	12(0C)		(See SDBXP block at end of SDB)
SDBXFLEN	16(10)		(See SDBXP block at end of SDB)
SDBXFLN	2(02)		(See SDBXP block at end of SDB)
SDBXFNB	1(01)		(See SDBXP block at end of SDB)
*SDBXFNR	12(0C)	80	(See SDBXP block at end of SDB)
SDBXFSBP	8(08)		(See SDBXP block at end of SDB)
SDBXFUSL	4(04)		(See SDBXP block at end of SDB)
SDBXPANS	56(38)		
SDBXPASF	16(10)		(See SDBXP block at end of SDB)
SDBXPEND	20(14)		(See SDBXP block at end of SDB)
SDBXPFDB	0(00)		(See SDBXP block at end of SDB)
*SDBXPFS	0(00)	02	(See SDBXP block at end of SDB)
SDBXPMSK	4(04)		(See SDBXP block at end of SDB)
*SDBXPRES	56(38)	01	
*SDBXPSI	0(00)	01	(See SDBXP block at end of SDB)
SDBXPSZ	20(14)		(See SDBXP block at end of SDB)
SDBXPTYP	0(00)		(See SDBXP block at end of SDB)
SDBXSQLN	14(0E)		(See SDBXP block at end of SDB)
SDBXSQOF	12(0C)		(See SDBXP block at end of SDB)
SDBXWMSK	8(08)		(See SDBXP block at end of SDB)

RECORD LAYOUT - SDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	SDBSYM		Segment symbolic name
0(00)	4	SDBLTP		Prior segment on logical twin chain
0(00)	4	SDBLTN		Next segment on logical twin chain
8(08)	1	SDBLEVEL		Level of this segment (logical)
9(09)	1	SDBORGN		Organization of data base containing segment
		SDBORGRI	44	This segment is root of index
		SDBORGHD	20	This segment is in a HDAM organization
		SDBORGHI	10	This segment is in a HIDAM organization
		SDBORGSH	05	This segment is in a simple HISAM organization
		SDBORGH1	04	This segment is in a HISAM organization
		SDBORGHS	02	This segment is in an HSAM organization
		SDBORGSS	01	This segment is in a simple HSAM organization
10(0A)	1	SDBF3		Call sensitivity
		SDBSENG	80	Sensitivity is read only
		SDBSENI	40	Sensitivity is insert
		SDBSENR	20	Sensitivity is replace
		SDBSEND	10	Sensitivity is delete
		SDBSENK	08	Sensitivity is key only
		SDBSENP	04	Sensitivity is path only
		SDBSENX	02	Sensitivity is exclusive
		SDBSENL	01	Sensitivity is load
11(0B)	1	SDBF4		Code byte
		SDBALTSQ	40	Secondary index is main processing sequence
		SDBALTSC	20	Secondary index search fields require conversion
		SDBDPAR	10	Field is in destination parent
		SDBCISP	04	Control interval split occurred in HISAM KSDS
		SDBPOSL	02	Position lost
		SDBLCH	01	Field is in logical child
		SDBDCHG	01	Temporary switch for replace; data changed
12(0C)	1	SDBPHYCD		Segment code
12(0C)	4	SDBDDIR		DMB directory address
16(10)	4	SDBNSDB		Next SDB for this PSDB
20(14)	4	SDBPSDB		Address of PSDB

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
24(18)	1	SDBKEYLN		Executable key length of key field
24(18)	4	SDBPARA		Parent SDB (address of PCB for root SDB) or address of prior SDB on 'SDBTARG' chain for generated SDBs (SDBGEN on in SDBTFLG)
28(1C)	4	SDBDSGA		Address of data set group section of JCB for data set containing segment
32(20)	1	SDBTFLG		Logical relationship code
		SDBPPTSP	C0	Segment is physical parent of target of SDBPARA
		SDBPPSP	80	Segment is physical parent of SDBPARA
		SDBPCTSP	40	Segment is physical child of target of SDBPARA
		SDBGEN	10	This SDB is a generated SDB
		SDBSPP	08	Segment is a virtual logical child
		SDBSNX	04	Segment is retrieved via index
		SDBSLC	02	(See bit flag 0001 0010)
		SDBSLP	01	Segment is a logical child

SDBTFLG Bit Flags

1xx0 xxxx	Inverted structure - The segment logically above this one is below it in the physical data base hierarchy. The segment logically above this one is represented by the SDB pointed to in SDBPARA. If SDBPARA points to a SDB for a logical child, this segment could be physically above either the logical child or its destination parent. A generated SDB pointed to by SDBTARG in the logical child's SDB represents the destination parent.
x1x0 xxxx	Logical relation - The segment represented by the SDB pointed to by SDBPARA is a logical child and this segment is either the physical parent or a physical child of its destination parent.
10x0 xxxx	This segment is the physical parent of the segment represented by the SDB identified as SDBPARA.
11x0 xxxx	The segment represented by the SDB pointed to in SDBPARA is a logical child and this segment is the physical parent of its destination parent (SDBTARG).
01x0 xxxx	The segment represented by the SDB pointed to in SDBPARA is a logical child

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
			and this segment is a physical child of its destination parent.	
		xxx0 1xxx	This segment is the logical child in a virtual logical child concatenated segment and SDBTARG point to the logical child's physical parent.	
		xxx0 xxx1	This segment is the logical child in a normal concatenated segment and SDBTARG points to the logical parent.	
		xxx1 xxxx	SDB is a generated SDB.	
		0001 0010	SDB is a generated SDB for an index. If SDBTARG is non-zero, it points to the generated SDB for the index target.	
		0001 0110	SDB is a generated SDB for a HIDAM root segment. SDBTARG points to the SDB for the primary index segment.	
33(21)	3	SDBTARG		Address of the logically related segments SDB
36(24)	1	SDBPTNO		Pointer number of first physical pointer
37(25)	1	SDBPTDS		Physical pointer flag
		SDBCTR	80	This logical parent segment has a counter
		SDBPTF	40	This segment has a physical twin forward pointer
		SDBPTB	20	This segment has a physical twin backward pointer
		SDBPP	10	This segment has a physical parent pointer
		SDBLTFD	08	This segment has a logical twin forward pointer
		SDBLTBK	04	This segment has a logical twin backward pointer
		SDBLP	02	This segment has a logical parent pointer
38(26)	1	SDBPCF		Pointer number in parent to first occurrence of this segment type
39(27)	1	SDBPCB		Pointer number in parent to last occurrence of this segment type
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

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
				In generated SDB for physical destination parent: Byte 0 = Physical segment code of virtual logical child; Bytes 1-3 = virtual logical child's PSDB address
44(2C)	4	SDBPOSP		Previous position
48(30)	4	SDBPOSC		Current position. X'80' in high-order byte = position lost, in conjunction with SDBPOSL in SDBF4
52(34)	4	SDBPOSN		Next position (current position in generated SDBs)
56(38)	1	SDBXFL		SDB expansion flag
		SDBXPRES	01	SDB expansion for secondary index processing sequence is present. (Secondary index is main processing sequence.)
		SDBFLS	02	Segment has field level sensitivity
56(38)	4	SDBXPANS		SDB expansion address
60(3C)		SDBEND		End of SDB entry
60(3C)		SDBLEN		Length of each SDB (SDBEND minus SDBSYM)

SDB EXPANSION BLOCK

DSECT Name: SDBXP

This block is present if indicated in SDB; see field SDBXFL, flag SDBXPRES.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	SDBXPTYP		SDB expansion type
		SDBXPSI	01	SDB expansion is for secondary index
		SDBXPFS	02	SDB expansion is for field sensitivity
0(00)	3	SDBXPFDB		Address of secondary index sequence field FDB
4(04)	4	SDBXPMSK		Mask of XDFLD FDBs allowed in SSAs
8(08)	4	SDBXWMSK		Work area reserved for open/close

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
12(0C)	2	SDBXSQOF		Offset from DBPCBKFD to SUBSEQ area (0 if area not present)
14(0E)	2	SDBXSQLN		Length of SUBSEQ field(s) minus 1
16(10)	4	SDBXPASF		Alternate sequence FSB pointer
20(14)		SDBXPEND		End of SDB expansion block entry
20(14)		SDBXPSZ		Length of one SDB expansion block entry (SDBXPEND minus SDBXP)

SDB EXPANSION BLOCK FOR FIELD SENSITIVITY

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
1(01)	1	SDBXFNB		Number of FSBs
2(02)	2	SDBXFLN		Length of expansion block
4(04)	2	SDBXFUSL		Length of segment in user's view
6(06)	2	SDBXFISL		Insert length of segment
8(08)	4	SDBXFSBP		ACBGEN - first FSB address
12(0C)	1	SDBXFLAG		Flags
		SDBXFNR	80	At least one NOREPL rule
13(0D)	3			**Reserved**
16(10)	0	SDBXFEND		End of SDB expansion block entry
16(10)	0	SDBXFLEN		Length of one SDB expansion block
16(10)	0	SDBXFFSB		Start of first FSB

SEC - SECONDARY LIST

DSECT Name: DMBSEC

The secondary list is described as part of the general structure and description of the DMB. The labels in SEC vary with the type of secondary index entry. See the field description listed by code type in the record layout.

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*DMBEXIT	1(01)	02	(See Code 40)
*DMBEXLOD	1(01)	04	(See Code 40)
*DMBEXTRN	0(00)	40	
DMBFDFLG	1(01)		(See Code 04)
DMBFDOFF	6(06)		(See Code 04)
*DMBFDONE	1(01)	10	(See Code 04)
*DMBFDUSE	1(01)	01	(See Code 04)
*DMBINDXD	0(00)	44	
DMBIPSDB	8(08)		(See Code 64)
DMBISSOF	2(02)		(See Code 64)
DMBISSSC	8(08)		(See Code 64)
DMBNBYTE	4(04)		(See Code 40)
*DMBNXISS	0(00)	60	
*DMBNXXDS	0(00)	64	
DMBSCDE	0(00)		
DMBSECDB	4(04)		(See Code 01)
DMBSECLN	16(10)		(See Code 64)
DMBSECND	16(10)		(See Code 64)
DMBSECNM	8(08)		(See Code 01)
DMBSECSC	4(04)		(See Code 01)
DMBSFCEN	12(0C)		(See Code 08)
DMBSFD	2(02)		(See Code 01)
DMBSFLEN	13(0D)		(See Code 08)
DMBSFLG	1(01)		(See Code 01)
DMBSFLG1	1(01)		(See Code 40)
DMBSFNAM	2(02)		(See Code 08)
DMBSFOFF	10(0A)		(See Code 08)
DMBSFPSC	1(01)		(See Code 08)
DMBSKYLN	1(01)		(See Code 60)
*DMBSLC	0(00)	02	
*DMBSLCF	0(00)	08	
DMBSLCFL	2(02)		(See Code 02)
DMBSLCIR	1(01)		(See Code 02)
*DMBSLP	0(00)	01	
*DMBSND	0(00)	80	
*DMBSNULL	1(01)	01	(See Code 40)
DMBSOFF	2(02)		(See Code 44)
*DMBSOURC	0(00)	20	
*DMBSRCH	0(00)	04	
*DMBSUBSQ	0(00)	24	
*DMBSYMN1	1(01)	04	(See Code 04)
DMBSYMOF	14(0E)		(See Code 44)
*DMBSYM1	1(01)	08	(See Code 04)
*DMBSYSFD	1(01)	02	(See Code 04)
*DMBVKY	1(01)	C'V'	(See Code 01)

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)	
*DMBXDCON	12(0C)	08	(See Code 44)
*DMBXDEQ	12(0C)	01	(See Code 44)
DMBXDFLG	12(0C)		(See Code 44)
*DMBXDLST	12(0C)	80	(See Code 44)
DMBXDPAD	13(0D)		(See Code 44)
DMBXDSC	8(08)		(See Code 44)
DMBXDSDB	4(04)		(See Code 44)
*DMBXDSPC	12(0C)	10	(See Code 44)
DMBXDSSC	4(04)		(See Code 44)
*DMBXDSSQ	12(0C)	04	(See Code 44)
*DMBXDSSS	12(0C)	20	(See Code 44)
*DMBXDSYM	12(0C)	40	(See Code 44)
DMBXITAD	4(04)		(See Code 40)
DMBXNSDB	4(04)		(See Code 60)
DMBXNSSC	4(04)		(See Code 60)
DMBXPSDB	8(08)		(See Code 44)
DMBXSOFF	14(0E)		(See Code 08)

RECORD LAYOUT - SEC

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	1	DMBSCDE		Code byte
		DMBSLP	01	Secondary list describes a logical parent
		DMBSLC	02	Secondary list describes a logical child
		DMBSRCH	04	Secondary list describes index search field(s)
		DMBSLCF	08	Secondary list describes logical twin sequence field
		DMBSOURC	20	Secondary list describes index DDATA field(s)
		DMBSUBSQ	24	Secondary list describes index SUBSEQ field(s)
		DMBEXTRN	40	Secondary list describes index user exit routine
		DMBINDXD	44	Secondary list describes index target segment as seen from index pointer segment
		DMBNXISS	60	Secondary list describes index relationship as seen from index source segment
		DMBNXXDS	64	Secondary list describes index relationship as seen from index target segment. This list is not present if ISS=TARGET
		DMBSND	80	Last entry in secondary list

THE FOLLOWING FIELDS ARE LISTED BY CODE TYPE

CODE 01 - DESCRIBES LOGICAL PARENT

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
1(01)	1	DMBSFLG		
		DMBVKY	C'V'	Key of logical parent is virtual
2(02)	2	DMBSFD		Logical parent key length
4(04)	1	DMBSECSC		Segment code of referenced segment
4(04)	4	DMBSECDB		DDIR address of referenced data base
8(08)	8	DMBSECNM		Segment name of referenced segment

CODE 02 - DESCRIBES LOGICAL CHILD

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
1(01)	1	DMBSLCIR		Logical twin sequence insert rule
2(02)	2	DMBSLCFL		Number of first and last logical child pointers in logical parent prefix

Remaining fields are same as Code 01.

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
CODE 04 - DESCRIBES INDEX SEARCH FIELDS				
1(01)	5	DMBFDFLG		Five 1-byte flags associated with the following FDB offsets
		DMBSYM1	08	First part of symbolic pointer
		DMBSYMN1	04	Not first part of symbolic pointer (middle or last)
		DMBSYSFD	02	This slot for system-related field
		DMBFDUSE	01	This slot in use
		DMBFDONE	10	This entry processed by block builder
6(06)	10	DMBFDOFF		Offset to FDB from first FDB of ISS if this slot is in use. Otherwise, zero.
CODE 08 - DESCRIBES LOGICAL TWIN SEQUENCE FIELD				
1(01)	1	DMBSFPSC		Virtual logical child physical segment code
2(02)	8	DMBSFNAM		FDB field name
10(0A)	2	DMBSFOFF		Offset to field in segment
12(0C)	1	DMBSFCEN		Code byte (same as FDBDCENF in FDB)
13(0D)	1	DMBSFLEN		Executable field length
14(0E)	2	DMBXSOFF		Offset to field in indexed segment
CODE 20 - DESCRIBES DDATA FIELD				
Same fields as Code 04				
CODE 24 - DESCRIBES SUBSEQ FIELD				
Same fields as Code 04				
CODE 40 - DESCRIBES INDEX EXIT ROUTINE				
1(01)	1	DMBSFLG1		Flag byte
		DMBSNULL	01	Null field present
		DMBEXIT	02	Exit routine present
		DMBEXLOD	04	Exit routine has been loaded
2(02)	2			***Reserved***
4(04)	4	DMBNBYTE		If index field equals this byte, bypass indexing
4(04)	4	DMBXITAD		Address of index maintenance parameter CSECT
8(08)	8			***Reserved***

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
CODE 44 - DESCRIBES INDEX TARGET SEGMENT				
1(01)	1	DMBSKYLN		Executable length of key
2(02)	2	DMBSOFF		Offset to PSDB address pointer of index target segment
4(04)	4	DMBXDSSC		Segment code of index target segment
4(04)	4	DMBXDSDB		DDIR address of index target segment
8(08)	4	DMBXDSC		Segment code of index target segment
8(08)	4	DMBXPSDB		PSDB address of index target segment
12(0C)	1	DMBXDFLG		Code byte from associated FDB
		DMBXDLST	80	Last FDB in list
		DMBXDSYM	40	Index pointer is symbolic
		DMBXDSSS	20	Pointer contained in source/subseq data
		DMBXDSPC	10	Special FDB for secondary index
		DMBXDCON	08	Constant present
		DMBXDSSQ	04	SUBSEQ present
		DMBXDSOR	02	
		DMBXDEQ	01	XDS=ISS
13(0D)	1	DMBXDPAD		Padding constant
14(0E)	2	DMBSYMOF		Offset to symbolic pointer indexing segment
CODE 60 - DESCRIBES INDEX FROM ISS				
1(01)	3			Same as code 44
4(04)	1	DMBXNSSC		Segment code of index pointer segment
4(04)	4	DMBXNSDB		DDIR address of index
Remaining fields same as Code 44				
CODE 64 - DESCRIBES INDEX FROM INDEX TARGET				
1(01)	1			Same as code 44
2(02)	2	DMBISSOF		Offset to Code 60 from start of ISS secondary list
4(04)	4			Same as code 60
8(08)	1	DMBISSSC		Segment code of index source segment
8(08)	4	DMBIPSDB		PSDB address of index source segment

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
12(0C)	1			Same as code 44
16(10)		DMBSECND		End of each secondary list entry
16(10)		DMBSECLN		Length of each secondary list entry

UIB - USER INTERFACE BLOCK

DSECT Name: DLIUIB

The user section of this control block is used by extended DL/I call interface support (along with CICS/VS high-level language support). This section contains scheduling and system call status information returned to the user. (Prior to Version 1.4, this information was returned to the user in the TCA.) A system section of the UIB follows the user section. It is used by DL/I as task-local storage. Unlike PST storage, UIB storage is not released at scheduling termination.

RECORD LAYOUT - UIB (USER SECTION)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	0	UIB		Start of UIB DSECT.
0(00)	4	UIBPCBAL		PCB address list.
4(04)	0	UIBRCODE		DL/I return codes.
4(04)	1	UIBFCTR		Return code.
5(05)	1	UIBDLTR		Additional information.
6(06)	2	Unnamed		**Reserved**
8(08)		UIBLEN		Length of UIB (for Assembler language only).

RECORD LAYOUT - UIB (SYSTEM SECTION)

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
8(08)	64	UIBREGSV		Register save area.
72(48)	8	UIBPSB		PCB name on scheduling call.
80(50)	4	UIBFUNC		Call function type.
84(54)	1	UIBFLAG1		UIB Flag.
		UIBSCHD	01	Scheduling call.
		UIBDB	02	Data base call.
		UIBTERM	04	Term call.
		UIBREMOT	80	PSB on remote system.
85(55)	1	UIBFLAG2		UIB flag.
88(58)		UIBSLEN		Length of user and system UIB

XMPRM - HDAM/HIDAM USER SECONDARY INDEX SUPPRESSION ROUTINE INTERFACE TABLE

DSECT Name: DMBXMPRM

This table is described as part of the general structure and description of the data management block (DMB).

ALPHABETIC LIST OF FIELD/FLAG NAMES

Field/Flag Name	Offset Dec(Hex)	Flag Code(Hex)
DMBXMPLN	28(1C)	
DMBXMRES	32(20)	
DMBXMSGN	0(00)	
DMBXMXDN	8(08)	
DMBXMXEP	24(18)	
DMBXMXNM	16(10)	

RECORD LAYOUT - XMPRM

Offset Dec(Hex)	Length	Field/Flag Name	Flag Code(Hex)	Meaning
0(00)	8	DMBXMSGN		Name of indexed segment
8(08)	8	DMBXMXDN		Name of XDFLD
16(10)	8	DMBXMXNM		Name of user exit routine
24(18)	4	DMBXMXEP		Entry point of user exit routine
28(1C)	2	DMBXMPLN		Length of index maintenance parameters
30(1E)	2			***Reserved***
32(20)	4	DMBXMRES		Reserved for initialization

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/1 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
Work File 1
Work File 3

ACCUMULATION HEADER RECORD

This record is used by modules DLZUC350 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	HLENGTH	2	Length of cum header record
2	2	HSPACE	2	Zeros
4	4	HCODE	1	Header record ID X'00'
5	5	HFLG	1	Type of data set and organization Bit 5=0 ESDS data set =1 KSDS data set 6=0 HS data set =1 HD data set
6	6	HLRECL	2	Record length
8	8	HORG	1	Prefix organization code
9	9	HPURDATE	3	Purge date for data base data set
C	12	HPURTIME	4	Purge time for data base data set
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	Block size

ACCUMULATION RECORD

This record is used by modules DLZUC350 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	CLENGTH	2	Length of cum record
2	2	CSPACE	2	Zeros
4	4	CCODE	1	X'50' record identifier
5	5	CFLG	1	Type of data set and organization Bit Meaning 5=0 ESDS =1 KSDS 6=0 HS file =1 HD file
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 of CDATA
1C	28	CDATAID	Var	KSDS prime key or ESDS RBN
		CDATAOL	Var	One or more 4 byte data elements: bytes 0-1 - offset into data set record bytes 2-3 - length of corresponding CDATASEG
		CDATASEG	Var	One or more segment data entries to be moved into data set record.

APPLICATION PROGRAM SCHEDULING RECORD

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

Hex	Dec	Name	Ln	Description
0	0	LENGTH	2	Length of record
2	2	SPACE	2	Binary zero
4	4	LOGFLAG	1	Record type code - X'08'
5	5	SCHDCODE	1	Task ID
8	8	PSBNAME	8	PSB name
E	14	CICSID	3	Packed CICS Transaction ID (online only)

APPLICATION PROGRAM TERMINATION RECORD

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

Hex	Dec	Name	Ln	Description
0	0	PLENGTH	2	Halfword binary length of logical record
2	2	PSPACE	2	Halfword reserved for system use (binary zero)
4	4	ALLOGFLG	1	Identifies this logical record as application program termination record; value is X'07'
5	5	ALPSBNAM	8	PSB name
D	13	ALID	1	TASK ID
E	14	TSKSTAT	40	10 fullwords of Accounting from PSTACCT (online only)
36	54	CICSID	3	Packed CICS transaction I.D. (online only)

CHECKPOINT LOG RECORD

Checkpoint log records are used to restart a job near its point of failure. The records are created and written on the DL/I log (if data base logging is active) if requested by the user via checkpoint calls (CHKP). Each log record contains a user-supplied unique checkpoint identification passed with the CHKP call.

In case of a job failure in a batch environment, the backout utility can be run to backout data base changes occurring since the last checkpoint record was written. For MPS and/or online tasks with CICS/VS dynamic transaction backout active, backout is performed automatically to the last checkpoint when a task fails.

Hex	Dec	Name	Ln	Description
0	0	CHKPLEN	2	Length of log record
2	2	CHKPSPC	2	Blanks/zeros
4	4	CHKPCODE	1	Log record ID

Flag Name	Hex Code	Meaning
CHKPLRID	41	Checkpoint Log record ID

Hex	Dec	Name	Ln	Description
5	5	CHKPPSB	8	Checkpoint PSB name
D	13	CHKPID	8	User checkpoint ID
15	21	CHKPRLEN		Length of checkpoint log record

CHECKPOINT RECORD

This DSECT (RCHKREC) defines the format of the checkpoint records within the unloaded data base for HD reorganization unload/reload utilities.

Hex	Dec	Name	Ln	Description
0	0	RCHKPTID	1	Identifies checkpoint record; Always X'00'
1	1	RCHKNAME	6	Constant for checkpoint record; Always C'CHKPNT'
7	7	RCHKNUM	4	Checkpoint number; 1-9999 (decimal)
B	11		1	Comma, for message to SYSLOG and SYSLST
C	12	RCHKVOL1	6	If tape, file serial number of output volume one at checkpoint time. If DASD - ******.
12	18		1	Comma, for message to SYSLOG and SYSLST
13	19	RCHKVOL2	6	If tape, file serial number of output volume two at checkpoint time. If DASD - ******.
19	25		1	Comma, for message to SYSLOG and SYSLST
1A	26	RCKSEGNM	8	Segment name of root segment in process at checkpoint time

22	34		4	Reserved for future use
26	38	RCHKRECL	2	Length of I/O area needed for GU call at restart time
28	40	RCHKPOSC	4	RBN of current record, if HD organization
2C	44	RCHKPTNR	1	Number of checkpoint records (1 or 2)
2D	45	RCHKEYLN	1	Key length of current segment, if HISAM
2E	46	RCKEYVAL	236	Segment sequence field value, if HISAM
11A	282	Reserved	12	Reserved
126	294	RCHKSEG	4	Total number of segments unloaded
12A	298	RCHKROOT	4	Total number of root segments unloaded
12E	302	RCHKREND	Var	Statistics table

Notes:
- Dummy checkpoint record does not contain statistics table.
- Checkpoint message written to SYSLOG and SYSLST consists of message prefix DLZ381I followed by bytes 1 - 34 of the checkpoint record.

CONTROL DATA SET

Macro DLZUCDS0 contains the DSECT defining format of a control list entry. One or more list entries may be contained in the control list. The control list may spread over one or more control list blocks.

Control Information and Identifier

Hex	Dec	Name	Ln	Description
0	0	LECELCNT	2	Number of 1600 byte records in control data set
2	2	LELSTLOC	2	Displacement to next entry
4	4	LECDSID	20	Identifier: ' CONTROL DATA SET '.
18	24	LEFLG4	1	Flag byte 4:

FLAG Name	Hex Code	Meaning
LESTAT	80	Statistics to be provided
LESUMM	40	Give summary for message DLZ978I

19	25	Unnamed	1	**Reserved**
1A	26	LESRTSZE	2	Maximum work file record length used as SORT size parameter by prefix resolution utility (DLZURG10).

Data Base List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same level)
4	4	LENAME	8	DBD name.
C	12	LESLPTR	4	List entry sublist pointer (to list at next lower level)
10	16	LECRNO	2	Input control card number
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan method option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

Segment List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same level)
4	4	LENAME	8	Logical parent segment name.
C	12	LESLPTR	4	List entry sublist pointer (to list at next lower level)
10	16	LECRNO	2	Input control card number
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan method option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

Hex	Dec	Name	Ln	Description
14	20	LEPSDB	4	PSDB for segment entry
18	24	LELSDB	4	LSDB for segment entry

Secondary List Entry

Hex	Dec	Name	Ln	Description
0	0	LEFPTR	4	List entry forward pointer (to next list element at same lavel)

4	4	LENAME	8	Referenced data base name.
C	12	LEFDLP	2	Length of logical parent concatenated key.
E	14	LEFLG3	1	Flag byte 3:

Flag Name	Hex Code	Meaning
LET23	80	Use type 20/30 records.
LELCSQ	40	Use logical child sequence field.
LENLC	20	No logical child found for logical parent.
LELPCK	02	Use logical parent concatenated key.
LELPOA	01	Use logical parent old address.

F	15	Unnamed	1	**Reserved**
10	16	LEFDLC	2	Position of logical child pointers in prefix
12	18	LELEN	1	Length of list entry
13	19	LEFLG1	1	Flag byte 1:

Flag Name	Hex Code	Meaning
LEF1SOPT	80	User specified scan mehtod option
LEF1SMET	40	If bit 1=0 use SEQ scan method If bit 1=1 use SEG scan method
LEF1S	02	Data base is scanned
LEF1R	01	Data base is reorganized
LEF1I	00	Data base is initially loaded

14	20	LELCSC	1	Logical child's segment code
15	21	LEFLG2	1	Flage byte 2:

Flag Name	Hex Code	Meaning
LECTR	80	Update counter
LELCF	40	Update logical child forward pointer
LELCL	20	Update logical child last pointer
LELP	10	Update logical parent pointer
LELTF	08	Update logical twin forward pointer
LELTB	04	Update logical twin backward pointer
LECUS	02	Counter used this logical child

17	23	Unnamed	2	**Reserved**

DATA BASE LOG RECORD

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

Hex	Dec	Name	Ln	Description
0	0	DLENGTH	2	Length of record
2	2	DSPACE	2	Zero
4	4	DLOGCODE	1	Log record ID X'50' = Data base log record X'51' = Old copy of a replaced segment

5	5	DLOGFLG1	1	<u>Bits</u> 0-3 Task ID 4-7 Count of FSE records present
6	6	DLOGFLG2	1	<u>Bits</u> 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	<u>Bits</u> 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 Reserved 6=1 First log record of a segment 7=1 Last log record of a segment
8	8	DIDLN	2	Length of DDATAID field
A	10	DOFFSET	2	Data offset from beginning of block
C	12	DDATALN	2	Length of DDATA field
E	14	DCCODE	2	DL/I completion code
10	16	DPGMNAME	8	PSB name
18	24	DDBDNAME	8	Data base name from the DMB
20	32	DDSID	1	File identification within the DMB
21	33	DDATE	3	Date - YYDDDF
24	36	DTIME	4	Time - HHMMSS0F
28	40	DSEQ	2	Sequence stamp
2A	42	DDATAID	Var	KSDS - KSDS prime key ESDS - Relative block number

POINTER maintenance record (DDATALN is set to H'4')

DDATA	4	New pointer value
	4	Old pointer value

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

DDATA	2	Segment code and new delete byte
	2	Segment code and old delete byte

PHYSICAL INSERT record (DDATALN is set to segment length)

DDATA	V*	New segment data
DFSEOFF	2	Offset to FSE
DFSE	4	New FSE value If more than one FSE changes, DFSEOFF and DFSE are repeated for each additional one.

PHYSICAL DELETE record (DDATALN is set to segment length)

DDATA	V*	Old segment data
DFSEOFF	2	Offset to FSE
DFSE	4	New FSE value If more than one FSE changes, DFSEOFF and DFSE are repeated for each additional one.

PHYSICAL REPLACE record (DDATALN is set to segment length)

DDATA	V*	Old segment data - DLOGCODE = X'51'
		New segment data - DLOGCODE = X'50'

V* = varies with segment length

DCOUNTER The last four bytes of every log record contain the log record sequence number. Numbers are incremented by one. The sequence number of the first record is one.

DATA RECORD (INPUT)

This record is used as input to module DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	4	ESDS RBA identifier; unused if KSDS
4	4	DSIDIN	1	Character I if KSDS; O if ESDS
5	5	Unnamed	3	Reserved
8	8	Unnamed	Var	KSDS or ESDS physical record image. The first four bytes contain the VSAM relative byte address (RBA) of the next ESDS record containing overflow dependent segments for the root segment. The RBA is zero if no (more) ESDS records follow. The last byte of the data record contains a special physical code X'0'. If the

data base contains only HISAM root segments and ACCESS=SHISAM, the physical code and RBA do not exist.

DATA RECORD (OUTPUT)

This output record is used by module DLZURUP0.

Hex	Dec	Name	Ln	Description
0	0	CONTOUT	4	ESDS RBA identifier; unused if KSDS
4	4	DSIDOUT	1	Character I if KSDS; O if ESDS
5	5	BLNKDOUT	1	(Not used)
6	6	DSRECLN	2	Record size + prefix length
8	8	DATA	Var	KSDS or ESDS physical record image. The first four bytes contain the VSAM relative byte address (RBA) of the next ESDS record containing overflow dependent segments for the root segment. The RBA is zero if no (more) ESDS records follow. The last byte of the data record contains a special physical code X'0'. If the data base contains only HISAM root segments and ACCESS=SHISAM, the physical code and RBA do not exist.

DATE/TIME TABLE

This record is used by modules DLZUCCT0 and DLZUC150.

Hex	Dec	Name	Ln	Description
0	0	TABFLAG1	1	Blank. Used as table delimiter
1	1	TABFLAG2	1	Contains a 0 or 1 to denote routing for the data base in this table
2	2	TABFLAG3	1	Contains flags as follows: Name / Bit / Meaning TABF3N / 0 / Record to LOGOUT if 1 TABF3DT / 1 / Purge date specified
3	3	TABFLAG4	1	Reserved for future use
4	4	TABFLAG5	4	Reserved for future use
8	8	TABFLAG6	8	Contains date/time, if specified

DELETE WORK AREA

This record is used by module DLZDLD00.

Hex	Dec	Name	Ln	Description
0	0	DLTRSCID	7	Resource ID for PI queuing (must be first in WKA)

0	0	DLTRSCRB	4	RBA portion of resource ID
4	4	DLTCHN	8	Chain (prior content PSTWRKD1-2)
4	4	DLTPWAID	4	ID of current work area; DMB number, ACB number, and work area sequence number
4	4	DLTRSCID	3	DMB/ACB number part of resource ID
4	4	DLTDMBNO	2	DMB number
8	8	Unnamed	4	Prior scan exit address (PSTWRKD2)
C	12	DLTWANXT	4	Address of next WKA
10	16	DLTWASW	1	Switch

Flag Name	Hex Code	Meaning
DLTWSBEG	01	First work area in work space
DLTERFLG	02	R-0 record flag required
DLTLRFLG	04	R-0 record flag required due to LP LC counter update
DLTVRFLG	08	Verifies are required
DLTSCFLG	10	Pre-scan was done
DLTIMFLG	20	Index maintenance was done

10	16	DLTWAPRI	4	Address of prior WKA
14	20	DLTDMB	4	DMB address of this WKA
18	24	DLTSPSDB	4	Scan start PSDB
1C	28	DLTLPSDB	4	Scan end PSDB
20	32	DLTSLEV	2	Level at which scan started
22	34	DLTTEMPH	2	Half word temporary save area
24	36	DLTESECL	4	Secondary list address causing exit
28	40	DLTEDMB	4	Exit DMB address
2C	44	DLTEPSDB	4	Prior DMB's PSDB (exit point)
30	48	DLTERBN	4	Exit RBN
34	52	DLTLPKOF	2	Offset from DLTWA to concatenated key
36	54	DLTWASZ	2	Length of this work area
38	56	DLTMID	36	'Middle' of WKA
38	56	DLTPLT	4	Save area for prior L/C on twin chain
3C	60	DLTCLT	4	Save area for current L/C on twin chain
40	64	DLTNLT	4	Save area for next L/C on twin chain
44	68	DLTTEMP1	4	Working register save area (R6)

48	72	DLTTEMP2	4	Working register save area (R7)
4C	76	DLTTEMP3	4	Working register save area (R8)
50	80	DLTTEMP4	4	Working register save area (R9)
54	84	DLTLEVEL	8	Level information beginning
54	84	DLTRFLG	1	Flag byte

Flag Name	Hex Code	Meaning
DLTSVPP	01	Save segment and parents
DLTSVPC	02	Save segment and physical children
DLTLDO	03	Logical delete only
DLTKEYSW	04	Key stored for this level
DLTTEFLG	08	Temporary lock enqueue was done

54	84	DLTPSDB	4	Current PSDB this level
58	88	DLTRBN	4	RBN of segment this level
5C	92	DLTLEVLN	8	Length of level information entry
64	100	DLTMIDLN	36	Length of last half work area
88	136	DLTWALN	92	Length of basic delete work area

DELETE WORK SPACE PREFIX

This record is used by module DLZDLD00.

Hex	Dec	Name	Ln	Description
0	0	DLTBLKNM	4	Block number of buffer (from PSTBLKNM)
4	4	DLTBUFFA	4	Address of buffer prefix (from PSTBUFFA)
8	8	DLTNXTWS	4	Address of next work space
C	12	DLTPRIWS	4	Address of prior work space
10	16	DLTSIZWS	4	Usable size of this space
14	14		4	Reserved

DL/I CONTROL RECORD

This record is used by module DLZDLOC0.

Hex	Dec	Name	Ln	Description
0	0	RECDATCR	3	Creation date - YYDDDF
3	3	RECTIMCR	5	Creation time - HHMMSSTHOF
8	8	RECDATRE	3	Recovery date - YYDDDF
B	11	RECTIMRE	5	Recovery time - HHMMSSTHOF
10	16	RECDATER	3	Reserved

13	19	RECTIMER	5	Reserved
18	24	RECNXRBA	4	Not used
1C	28	RECDOS	3	DL/I component code (DLZ)
1E	31	RECVERS	3	Version and release level
22	34	RECPTF	2	PTF number
24	36	RECLKSDS	4	KSDS record length (HISAM only)
28	40	RECLESDS	4	ESDS record length
2C	44	RECORGAN	1	Data base organization

Name	Character	Meaning
RECHDAM	D	HDAM
RECHIDAM	I	HIDAM
RECHISAM	S	HISAM

2D	45		Var	Reserved to end of control interval

DUMP HEADER RECORD

This record is used by modules DLZUDMP0 and DLZURDB0.

Hex	Dec	Name	Ln	Description
0	0	DHSAMCTL	1	Reserved for future use
1	1	DUMPID	1	Character D
2	2	DCBNOOUT	2	Reserved for future use
4	4	DUMPDBDN	8	Name of the DMB devised from the Data Base Description (DBD)
C	12	DIDDNOUT	8	Contains the name of the key sequenced data set if this is dump of a KSDS data set
14	20	DDATEOUT	4	Julian date in packed decimal - 00YYDDDF
18	24	DTIMEOUT	4	Time in packed decimal - HHMMSS0F
1C	28	DODDNOUT	8	Contains the name of the entry sequenced data set if this is dump of an ESDS data set
24	36	DIBLKOUT	2	Contains KSDS control interval size if this is dump of KSDS data set
26	38	DIRECOUT	2	Contains KSDS record length if dump of KSDS data set
28	40	DOBLKOUT	2	Contains ESDS control interval size if this is dump of ESDS data set
2A	42	DORECOUT	2	Contains ESDS record length if dump of ESDS
2C	44	DKEYLEN	2	Contains KSDS key length if

				dump of KSDS
2E	46	DKEYPOS	2	Contains KSDS relative key positive if dump of KSDS
30	48	DDBDORG	1	Data set organization code

DUMP RECORD PREFIX

This record is used by module DLZUDMP0.

Hex	Dec	Name	Ln	Description
0	0	COUNTOUT	4	ESDS RBA identifier; record count if KSDS
4	4	DSIDOUT	1	Character I if KSDS; O if ESDS
5	5	Reserved	1	Reserved for future use
6	6	DSRECLN	2	Record size + prefix length
8	8	DATA	Var	Physical record image

FILE OPEN RECORD

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

Hex	Dec	Name	Ln	Description
0	0	DLENGTH	2	Length of record
2	2	DSPACE1	2	Binary zero
4	4	DLOGCODE	1	Record type code - X'2F'
5	5	DLOGFLG1	2	Data set organization X'00' = ESDS X'04' = KSDS
7	7	DSPACE2	9	Binary zero
10	16	DPGMNAME	8	Data set filename (ACB)
18	24	DDBDNAME	8	DMB name
20	32	DDSID	1	DSGACBNO (2 if HISAM ESDS; otherwise 1)
21	33	DDATE	3	Binary zero
24	36	DTIME	4	Binary zero
28	40	DCOUNT2F	4	Log record sequence number

HEADER RECORD (INPUT)

This record is used as input for module DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	1	X'FF' header/statistic record identifier

1	1	IDIN	1	Character R
2	2	RECLNOUT	2	Size of output record, including prefix
4	4	DBDNAME	8	Name of the DMB derived from the Data Base Description (DBD)
C	12	DDNAMEI	8	Name of key sequenced data set (KSDS)
14	20	Unnamed	4	Julian date in packed decimal-00YYDDDF
18	24	Unnamed	4	Time in packed decimal-HHMMSSOF
1C	28	DDNAMEO	8	Name of entry sequenced data set (ESDS)
24	36	BLKSIZEI	2	KSDS record length * number of records/control interval
26	38	LRECLI	2	KSDS record length
28	40	BLKSIZEO	2	ESDS record length * number of records/control interval
2A	42	LRECLO	2	ESDS record length
2C	44	Unnamed	1	0; (Not used)
2D	45	KEYLENGI	1	KSDS key length
2E	46	KEYPOSI	2	KSDS relative key position

HEADER RECORD (OUTPUT)

This record is used by module DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	HSAMCTRL	1	X"FF" header/statistic record identifier
1	1	IDOUT	1	Character R
2	2	RECLNOUT	2	Size of output record, including prefix
4	4	DBDOUT	8	Name of the DMB derived from the Data Base Description (DBD)
C	12	IDDNOUT	8	Name of key sequenced data set (KSDS)
14	20	DATEOUT	4	Julian date in packed decimal-00YYDDDF
18	24	TIMEOUT	4	Time in packed decimal-HHMMSSOF
1C	28	ODDNOUT	8	Name of entry sequenced data set (ESDS)
24	36	IBLKSOUT	2	KSDS record length * number of records/control interval
26	38	ILRECOUT	2	KSDS record length
28	40	OBLKSOUT	2	ESDS record length * number of records/control interval

2A	42	OLRECOUT	2	ESDS record length
2C	44	IKEYLENG	2	KSDS key length
2E	46	IKEYPOS	2	KSDS relative key position

INDEX MAINTENANCE WORK AREA

This record is used by module DLZDMXT0.

Hex	Dec	Name	Ln	Description
0	0	XSAVDSGA	4	Save location for caller's DSG
4	4	XSAVPCB	4	Save location for caller's PCB
8	8	XSAVUSER	4	Save location for caller's I/O area
C	12	XSAVIQPR	4	For caller's call list address
10	16	XPHYSPP	4	Save location for physical parent pointer.
14	20	XWORKPCB	4	Save location for XMAINTs PCB
18	24	XWORKSAA	4	Address of SSA built by DLZDXMT0
1C	28	XWORKFNC	4	XMAINTs function code for call
20	32	XDPSDBAD	4	Address of PSDB of indexed segment
24	36	XDSECLST	4	Secondary list of indexed segment
28	40	XDRID	8	Indexed segment ID for enqueue
28	40	XDRBAPTR	4	RBA of indexed segment
2C	44	XDDMBACB	4	DMB and ACB numbers of indexed segment
30	48	XNRID	8	Indexing segment ID for enqueue
30	48	XNRBAPTR	4	RBA of indexing segment
34	52	XNDMBACB	4	DMB and ACB numbers of indexing segment
38	56	XSPSDBAD	4	PSDB of index source segment
3C	60	XSSECLST	4	Secondary list of index source segment
40	64	XSRBAPTR	4	RBA of index source segment
44	68	XNPSDBAD	4	Address of PSDB of indexing segment
48	72	XDSDBAD	4	Index target segment SDB address
4C	76	XSSDBAD	4	Index source segment SDB address
50	80	XPROT	2	Length of protected data

52	82	XRPREFIX	2	Record prefix length
54	84	XSPREFIX	2	Segment prefix length
56	86	XNSEGLEN	2	Length of indexing segment
58	88	XNKEYLEN	2	Sequence field length of index pointer segment
5C	92	STACK1	4	Return address for first level subroutine
60	96	STACK2	4	Return address for second level subroutine
64	100	STACK3	4	Return address for third level subroutine
68	104	XSAVSTC	1	Save status code
69	105		1	*Reserved*
6A	106	XCALLFUN	1	Call attributes byte

Flag Name	Hex Code	Meaning
ISLOAD	80	Load mode
ISASRT	40	ASRT call
ISDLET	20	DLET call
ISISRT	10	ISRT call
ISREPL	08	Function is replace
ISUNLD	02	UNLD call

6B	107	XTSWIT1	1	Temporary switch

Flag Name	Hex Code	Meaning
XNOSUPR	80	No suppression for this index
XOLDSUPR	40	Old segment was suppressed
XPTRONLY	20	PTR to XDS only, no CONCAT key
XISPRIM	10	A primary index was found
XNULLFLD	01	Null value suppression
XEXITRT	02	Exit routine for suppression
XDATACHN	04	XNS changed in a replace call

6E	110	XWORKPUT	2	Begin of record for load

(The rest of this record starts on a fullword boundary)

70	112	XWORKUSR	0	XMAINTs I/O area for call
70	112	XWORKDUM	2	Reserved
72	114	XWORKSEG	0	Start of segment

72	114	XWORKCD	1	Segment code

Flag Name	Hex Code	Meaning
XNSEGC01	01	Segment code of indexing segement

73	115	XWORKDEL	1	Delete byte in indexing segment
74	116	XWORKPTR	4	Pointer in indexing segment
78	120	XWORKKEY	VAR	Area for key in indexing segment

(The SSA for the XMAINT call to the analyzer is created behind the key)

LIST CONTROL BLOCK

This record is used by module DLZUSCH0.

Hex	Dec	Name	Ln	Description
1C	28	ENTLNGTH	2	The length, in bytes, of each entry in the list
1E	30	COMPLOC	2	The offset from the beginning of each entry to the key field
20	32	COMPLNG	2	The length of the key field
22	34	NUMENT	2	The current number of entries in the list
24	36	CHAINLOC	4	The location of the first of a chain of core blocks containing sorted list entries
28	40	CHBACK	4	The location of the last block in the chain
2C	44	ENTBLKSZ	4	The size of each core block used for list entries (includes the chaining fields).

This value is calculated as follows: ENTBLKSZ = 16*ENTLNGTH+8

30	48	LASTLO, LASTHI, 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	Ln	Description
0	0	RGUSEGLV	1	Segment code for this segment
1	1	RGUHSDF	1	HSAM delete flag; always X'80' to denote HD Reorganization Unload Utility
2	2	RGUHDRLN	2	Length of header portion of record
4	4	RGUSEGLN	2	Length of data portion of 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	Ln	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	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	SMIMCHLD	4	Maximum immediate children
C	12	SAIMCHLD	4	Average immediate children
10	16	WKIMCHLD	4	Working entry for above
14	20	SMSBCHLD	4	Maximum subordinate children
18	24	SASBCHLD	4	Average subordinate children
1C	28	WKSBCHLD	4	Working entry for above
20	32	TSEGTYPE	4	Total segments for this type
24	36	SEGLEVEL	1	Segment level
25	37	SEGPHYCD	1	Segment physical code
26	38	TABLEND	2	Table end indicator (X'80')
26	38	TSEGLEN	2	Segment length including prefix
28	40	STATABSZ		Length of each table entry

SHORT SEGMENT TABLE

This record is used by module DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	SEGMDSN0	1	Data set number (not used by DLZURUL0)
1	1	SEGMCODE	1	Physical segment code
2	2	PARSEGCD	1	Physical code of this segment's parent
3	3	SEGMLEVL	1	Segment hierarchical level
4	4	Unnamed	2	Number of logical children and fields (not used by DLZURUL0)
6	6	SEGMLENG	2	Segment length, including prefix

SORTED LIST BLOCK

This record is used by module DLZUSCH0.

Hex	Dec	Name	Ln	Description
0	0	ENCNT	1	The count minus one of the current number of entries in this block (currently, the maximum value for count is 16)
1	1	CHAIN	3	The location of the next sorted list block in the chain. In the last block, this field contains binary zeros.
4	4	BKCHAIN	4	The location of the preceding sorted list block in the chain. In the first block on the chain, this field contains the location of the CHAINLOC field in the list control block.
8	8	ENTRIES	Var	Up to 16 full entries in sorted order. Note: All blocks are the same size regardless of the number of entries contained. Unused space at the end of a block is not zeroed.

SSA FOR GU CALL BY KEY

This record is used by module DLZURGU0.

Hex	Dec	Name	Ln	Description
0	0	KEYSEGNM	8	Name of segment to be retrieved
8	8	KEYCODE	2	'*C' - command code
A	10	KLEFTPAR	1	'(' - left parenthesis
B	11	KEY	1-236	key to be retrieved
-	-	KRITEPAR	2	')' - right parenthesis

SSA FOR GU CALL BY RBA

This record is used by module DLZURGU0.

Hex	Dec	Name	Ln	Description
0	0	RBASEGNM	8	Name of segment to be retrieved
8	8	RBACODE	2	'*T' - command code
A	10	RLEFTPAR	1	'(' - left parenthesis
B	11	RBA	4	RBA to be retrieved
F	15	RRITEPAR	1	')' - right parenthesis

SSA FOR THE XMAINT CALL TO THE ANALYZER

This record is used by module DLZDXMT0.

Hex	Dec	Name	Ln	Description
0	0	XSEGNAME	8	Name of index pointer segment
8	8	XCOMMCOD	2	'*X' - command code
A	10	XLEFTPAR	1	'(' - left parenthesis
B	11	XKEYVALU	VAR	Key value followed by right parenthesis ')'

STATISTICS RECORD

This record is used by modules DLZURUL0 and DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	Unnamed	1	X'FF' header/statistics record identifier
1	1	Unnamed	1	Character S
2	2	Unnamed	2	Number of segment types in data set group (16 bytes per segment type)
4	4	Unnamed	8	Name of the DMB derived from the DBD
C	12	Unnamed	8	KSDS filename
14	20	Unnamed	8	ESDS filename
1C	28	Unnamed	Var	A 16-byte table entry for each segment type in the data base

DESCRIPTION OF VARIABLE LENGTH LAST FIELD OF STATISTICS RECORD WHEN USED AS OUTPUT FOR DLZURUL0.

Hex	Dec	Name	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	TSEGTYPE	4	Total number of segments unloaded

C	12	SEGLEV	1	Segment level
D	13	SEGPCD	1	Segment physical code
E	14	TSEGLN	2	Segment length, including prefix

DESCRIPTICN OF VARIABLE LENGTH LAST FIELD OF STATISTICS RECORD WHEN USED AS INPUT FOR DLZURRL0.

Hex	Dec	Name	Ln	Description
0	0	SEGNAME	8	Segment name
8	8	TOTSEG	4	Total number of segments unloaded
C	12	SEGLEV	1	Segment level
D	13	SEGPCD	1	Segment physical code
E	14	SEGLN	2	Segment length, including prefix

WORK FILE 1

This record is used as the input file for DLZURG10.

Hex	Dec	Name	Ln	Description
0	0	ALENGTH	2	Length of work file 1 record
2	2	ASPACE	2	Two bytes of zeros
4	4	ALTYPE	1	Type of input record

Flag Name	Hex Code	Meaning
ATYPE00	00	Type 00 record
ATYPE01	01	Type 01 record
ATYPE02	02	Type 02 record
ATYPE03	03	Type 03 record
ATYPE10	10	Type 10 record
ATYPE20	20	Type 20 record
ATYPE30	30	Type 30 record
ATYPE40	40	Type 40 record

DL/I Record Type	Use
00	Generated once for each use of a segment as a logical parent
10	Generated once for each use of a segment as a logical child.
20	Generated when a segment used as a logical child contains logical twin forward pointers and when the logical twin chain cannot be resolved by using the logical child's sequence field.
30	Generated when a segment used as a logical child contains logical twin backward pointers and when the logical

				twin chain cannot be resolved by using the logical child's sequence field. 40 Generated once for each time a segment is indexed
5	5	ALFLAG1	1	Flag 1

Flag Name	Hex Code	Meaning
AL1LOAD	80	Set to 1 if ISRT; set to 0 if ASRT
AL1SEQ	40	Set to 1 if sequence field is present
AL1SCAN	20	Set to 1 if record produced by scan program (DLZURGS0)
AL1LPCK	10	Set to 1 if logical parent concatenated key is prsent
AL1SQUN	08	Sequence field is unique
AL1SEQA	04	Set to 1 if root sequence field is present
AL1CONST	02	Constant present in key
AL1SYMB	01	For type 40 record; pointer is symbolic
AL1T23	01	Set to 1 if logical twin pointers are to be resolved by type 20 and 30 records

6	6	ALFLAG2	1	Executable length of sequence field, if present
7	7	ALFLAG3	1	Executable length of indexed field, if present, or executable length of logical parent concatenated key, if present
8	8	ALEVTTR	4	Value of LEVTTR after BYLCT
C	12	ALPDBNAM	8	Data base of logical parent
14	20	ALPSEQ	1	Segment code of logical parent
15	21	ALPCKEY	4	Logical parent's concatenated key
15	21	ALPOADDR	4	Logical parent's old address
19	25	ALCDBNAM	8	Data base of logical child
21	33	ALCSEG	1	Segment code of logical child

FOR TYPE 00 AND 01 RECORDS

22	34	ALCFL	4	Old value of logical child first or logical child last pointer
26	38	ALT0001	1	X'00' or X'01'
27	39	ALPLSGOF	2	Value of logical parent's LEVSEGOF after BYLCT
29	41	ALCCTR	4	Old value of counter field
2D	45	ALPDCB	1	DCB NUMBER FOR LP
(TYPE 01 RECORD ENDS HERE)				
2E	46	ALPSEQA	1	Sequence field and length for root of segment
FOR TYPE 02 RECORDS				
22	34	ALCOAD	4	Logical child old address
26	38	ALT02	1	X'02'
FOR TYPE 10, 20, AND 30 RECORDS				
22	34	ALFIL	1	X'FF'
23	35	ALCSEQ	4	Logical child sequence field
23	35	ALCM	4	If LC has LT pointers and a non-unique sequence field and is being reloaded, ALCM contains the following: For Type 10 - LC's old address For Type 20 - LC's old LT forward pointer For Type 30 - LC's old LT backward pointer Otherwise, ALCM contains the value of LEVSEGOF, with high order bit set to one
27	39	ALT123	1	X'10', or X'20', or X'30'
28	40	ALCDCB	1	DCB number for LC
29	41	ALCSEQA	1	Sequence field and length for root of segment
FOR TYPE 40 RECORDS				
8	8	AILCOA	4	Logical child old address
C	12	AIDBNAM	8	Index data base name
14	20	AIFLDVAL	1	Indexed field value (variable length)
14	20	AISC	1	Index segment's segment code
15	21	AISEQ	1	Index segment's sequence code (if second level and present)
15	21	AISEGN	8	Index segment's name (For level 2 index segments)

15	21	AIFLDN	8	Indexed field name (For level 1 index segments)
1D	29	AISDBN	8	Indexed segment's data base name
25	37	AISSC	1	Indexed segment's segment code
26	38	AILCNA	4	Logical child new address
2A	42	AIDATA	1	Indexed segment data (for source fields)

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

9	9	AISSFN	8	Index segment name or field name
11	17	AISSAID	3	SSA ID and command code
14	20	AISFLDV	1	Indexed segment's indexed field value (variable length)
14	20	AISSEQ	1	Index segment's sequence field value (variable length)
21	33	AXSC	1	Segment code of indexed segment
22	34	AXDDIR	3	DDIR address of indexed data base
25	37	AXLCNA	4	Logical child new address
29	41	AXDATA	1	Index source data

WORK FILE 3

This record is the output file from DLZURG10 and is used as the input file for DLZURGP0.

Hex	Dec	Name	Ln	Description
0	0	CLENGTH	2	Length of work file record
2	2	CSPACE	2	Zeros
4	4	CTYPE	1	Work file record type

Flag Name	Hex Code	Meaning
CTYPE0	00	Type 00 record
CTYPE01	01	Type 01 record
CTYPE1	10	Type 10 record
CTYPE2	20	Type 20 record
CTYPE3	30	Type 30 record
CTYPE4	40	Type 40 record

5	5	CFLAG1	1	Origin of record

Flag Name	Hex Code	Meaning
CF1LOAD	80	Flag on-initial load; Flag off-reorganization
CF1SCAN	20	Record produced by scan
CFILPCK	10	Logical parent concatenated key if present

CF1SEQA	04	Set to 1 if root sequence field present
CF1T0F	02	Set to 1 if matching type 10 record found
CF1T23	01	Set to 1 if logical twin pointer is to be resolved by type 20 and 30 records

FIELDS IN TYPE 0 RECORD

6	6	CLCDBN0	8	Logical child data base name
E	14	CLCSEGN0	1	Logical child segment code
F	15	CLPSEGN0	1	Logical parent segment code
10	16	CLCFRST	4	Logical child first pointer
14	20	CLCDLST	4	Logical child last counter
18	24	CLCDCNT	4	Logical child delta counter
1C	28	CLPDBN0	8	Logical parent data base name

FIELDS IN TYPE 1 RECORD

6	6	CLPDBN1	8	Logical parent data base name
E	14	CLPSEGN1	1	Logical parent segment code
F	15	CLCSEGN1	1	Logical child segment code
10	16	CLTFWD	4	Logical twin forward pointer
14	20	CLTBKWD	4	Logical twin backward pointer
18	24	CLPNWAD1	4	Logical parent new address
1C	28	CLCDBN1	8	Logical child data base name
24	36	CDCB	1	DCB number
25	37	CFIL	1	
26	38	CLEVTTR	4	Contents of LEVTTR after BYLCT
2A	42	CLEVSGOF	2	Contents of LEVSEGOF after BYLCT (high order bit of CLEVSGOF is set to 1 if segment is not in HD)
2C	44	CLCCNT	4	Old value of counter field
30	48	CLSEQ	1	Root sequence field

This section contains two tables that cross-reference DL/I messages and DL/I status codes with the module(s) that originate them.

Additional diagnostic information can be found in the DL/I DOS/VS Diagnostic Guide, SH24-5002.

SYSTEM MESSAGE/MODULE CROSS REFERENCE

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

Message Number	Module	Figure Number
DLZ001I	DLZBNUC0	2-4.2
DLZ002I	DLZBNUC0	2-4.2
DLZ003I	DLZDDLE0	
DLZ004I	DLZDBH00	
	DLZRDBL0	2-16.7
DLZ005I	DLZDBH00	
DLZ006I	DLZOLI00	2-5.4
DLZ007I	DLZDSEH0	2-38
	DLZDXMT0	
DLZ008I	DLZRRC00	2-3.9
DLZ009I	DLZRRC00	2-3.8
DLZ010A	DLZRRC00	
	DLZMPI00	2-21.1
DLZ011I	DLZRRC00	2-3.2
DLZ012I	DLZMPI00	2-21.1
	DLZRRC00	2-3.4, 2-3.7, 2-3.9
DLZ013I	DLZOLI00	2-5.3
DLZ014A	DLZRRC00	
	DLZMPI00	2-21.1
DLZ015I	DLZRRC00	2-3.3, 2-3.9
DLZ016I	DLZDLOC0	
DLZ017I	DLZRRC00	2-3.7
DLZ018I	DLZRRC00	2-3.7
DLZ019I	DLZRRC00	2-3.9
DLZ020I	DLZDLOC0	2-14.1
	DLZRDBL0	2-16.1
DLZ021I	DLZDLOC0	
	DLZRDBL0	2-16.6
DLZ022I	DLZDLOC0	
DLZ023I	DLZDLOC0	2-14.1
DLZ024I	DLZDLOC0	
DLZ025I	DLZDLOC0	2-14.1
DLZ026I	DLZRRC00	2-3.8
DLZ027I	DLZDLOC0	2-14.1
DLZ028I	DLZDLOC0	2-14.1
DLZ029I	DLZOLI00	2-5.3, 2-5.9
DLZ030A	DLZOLI00	2-5.8
DLZ031I	DLZOLI00	2-5.1
DLZ032A	DLZOLI00	2-5.4
	DLZRDBL1	
DLZ033I	DLZODP	2-6.15, 2-6.16
DLZ034I	DLZOLI00	2-5.1
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

Message Number	Module	Figure Number
DLZ047I	DLZOLI00	2-5.3
DLZ048I	DLZOLI00	2-5.3
DLZ049I	DLZOLI00	2-5.3
DLZ050I	DLZOLI00	2-5.1
DLZ051I	DLZOLI00	2-5.1
DLZ052I	DLZOLI00	2-5.5
DLZ053I	DLZOLI00	2-5.5
DLZ054I	DLZOLI00	2-5.5
DLZ055I	DLZOLI00	2-5.4
DLZ056I	DLZOLI00	2-5.4
DLZ057I	DLZOLI00	2-5.5
DLZ058I	DLZOLI00	2-5.6, 2-5.7
DLZ060I	DLZOLI00	2-5.9
DLZ061A	DLZOLI00	2-5.9
DLZ062I	DLZODP	2-6.10
DLZ063I	DLZODP	2-6.2
DLZ064I	DLZOLI00	2-5.1
DLZ065I	DLZODP	2-6.2
DLZ066I	DLZODP	2-6.2
DLZ067I	DLZODP	2-6.2
DLZ068I	DLZODP	2-6.2
DLZ069I	DLZODP	2-6.2
DLZ070I	DLZODP	2-6.2
DLZ071I	DLZOLI00	2-5.2
DLZ072I	DLZOLI00	2-5.3
DLZ073I	DLZOLI00	2-5.3
DLZ074I	DLZOLI00	2-5.3
DLZ075I	DLZRRC00	2-3.9
DLZ076A	DLZRDBL0	2-16.7.
DLZ077I	DLZRDBL0	2-16.1, 2-16.7
DLZ078I	DLZRRC00	2-3.9
DLZ079I	DLZRDBL0	2-16.7
DLZ080I	DLZMSTP0	2-22
DLZ081I	DLZMPI00	2-21.1
DLZ082I	DLZBPC00	2-20.1, 2-20.5
	DLZMPC00	2-19.2, 2-19.4, 2-19.5, 2-19.7, 2-19.8
	DLZMPI00	2-21.1, 2-21.3
DLZ083I	DLZMSTR0	2-18
DLZ084I	DLZBPC00	2-20.2, 2-24.4
	DLZMPC00	2-19.4
	DLZMPI00	2-21.1, 2-21.3
	DLZODP01	2-6.3
DLZ085I	DLZMPI00	2-21.1
DLZ086I	DLZMPC00	2-19.7
DLZ087A	DLZMPI00	2-21.1
DLZ088I	DLZMPC00	2-19.1
DLZ089I	DLZMPI00	2-21.1
DLZ090I	DLZMPI00	2-21.2
DLZ091I	DLZMPI00	2-21.3
DLZ092I	DLZMPI00	2-21.3
DLZ093I	DLZMPC00	2-19.2
DLZ094I	DLZMPC00	2-19.1, 2-19.8
DLZ095I	DLZMPI00	2-21.1
DLZ096I	DLZMPI00	2-21.5
DLZ097I	DLZMSTR0	2-18
DLZ098I	DLZMPI00	2-21.3
DLZ099I	DLZMPI00	2-21.1
DLZ100I	DLZMPI00	2-21.3
DLZ101I	DLZMSTR0	2-18
DLZ102I	DLZMPI00	2-21.3

Message Number	Module	Figure Number
DLZ103I	DLZBPC00	2-20.5
DLZ104I	DLZMPC00	2-19.9
	DLZBPC00	2-20.6
DLZ105I	DLZRRC00	
	DLZBNUC0	2-4.1
	DLZMPI00	
DLZ106I	DLZQUEF0	
DLZ108I	DLZQUEF0	
DLZ120I	DLZTRACE	
DLZ260I	DLZBNUC0	2-4.1
	DLZODP	2-6.10
DLZ261I	DLZBNUC0	2-4.1
	DLZODP	2-6.10
DLZ262I	DLZRRC00	2-3.8
	DLZOLI00	2-5.9
DLZ263I	DLZRRC00	2-3.7
DLZ264I	DLZRDBL1	
DLZ266I	DLZRRC00	2-3.7
	DLZOLI00	2-5.3
DLZ267I	DLZQUEF0	2-23
DLZ268I	DLZDDLE0	
DLZ301I	DLZUDMP0	
	DLZURDB0	
	DLZURGL0	2-32
	DLZURGU0	2-31
	DLZURRL0	
	DLZUC350	
	DLZURUL0	
DLZ302I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZURCC0	2-27.1
DLZ303I	DLZUDMP0	2-25
	DLZURUL0	2-29
DLZ304I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURCC0	2-27.1
DLZ305I	DLZUDMP0	
	DLZURDB0	
	DLZURUL0	
DLZ306I	DLZURUL0	
	DLZURDB0	
	DLZUDMP0	
DLZ307I	DLZURUL0	2-29
	DLZUDMP0	2-25
	DLZURRL0	2-30
	DLZURCC0	2-27.1
DLZ308I	DLZUDMP0	2-25
	DLZURUL0	2-29
DLZ309I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZRDBL0	
DLZ310I	DLZUDMP0	2-25
	DLZURUL0	2-29
	DLZURRL0	2-30
	DLZRDBL0	
	DLZURCC0	2-27.1
DLZ311I	DLZURRL0	
	DLZURGU0	2-31

Message Number	Module	Figure Number
	DLZURGL0	2-32
	DLZLOGP0	
	DLZTPRT0	
DLZ312I	DLZURDB0	
DLZ313I	DLZURDB0	
DLZ314I	DLZURDB0	
DLZ315I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ316I	DLZURDB0	
	DLZUDMP0	
DLZ317I	DLZURDB0	
DLZ318A	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ319I	DLZURUL0	
	DLZURGU0	
	DLZUDMP0	
	DLZURGL0	2-32
	DLZURDB0	
	DLZURRL0	
DLZ320I	DLZURUL0	
	DLZURGU0	
	DLZUDMP0	
DLZ321I	DLZURUL0	
	DLZUDMP0	
	DLZURRL0	
DLZ322I	DLZURDB0	
DLZ323I	DLZURDB0	
DLZ324I	DLZURDB0	
DLZ325I	DLZURDB0	
DLZ326I	DLZURDB0	
DLZ327I	DLZURDB0	
DLZ328I	DLZURDB0	
DLZ330I	DLZURDB0	
DLZ331I	DLZURDB0	
DLZ332I	DLZURDB0	
DLZ333I	DLZURDB0	
DLZ334I	DLZURDB0	
DLZ335I	DLZURDB0	
DLZ336I	DLZURDB0	
DLZ337I	DLZURDB0	
DLZ338I	DLZURDB0	
DLZ339I	DLZURDB0	
DLZ340I	DLZURDB0	
DLZ341I	DLZURDB0	
DLZ342I	DLZBACK0	
	DLZLPCC0	
	DLZURCC0	2-27.1
	DLZUCCT0	
DLZ343I	DLZURDB0	
DLZ345I	DLZURGU0	2-31
	DLZUDMP0	
	DLZURUL0	
DLZ346I	DLZURGU0	
DLZ347I	DLZURGU0	2-31
DLZ348I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ349I	DLZURGU0	2-31
DLZ350I	DLZUDMP0	
DLZ351I	DLZUDMP0	
DLZ352I	DLZURGU0	2-31

Message Number	Module	Figure Number
DLZ353I	DLZURRL0	
DLZ354I	DLZURGL0	2-32
DLZ355I	DLZURGL0	2-32
DLZ356I	DLZURRL0	
DLZ357I	DLZURUL0	
	DLZUDMP0	
DLZ358I	DLZURUL0	
DLZ360I	DLZUCCT0	
DLZ361I	DLZUCCT0	
DLZ362I	DLZUCCT0	
DLZ363I	DLZUCCT0	
DLZ364I	DLZUCCT0	
DLZ365I	DLZUCCT0	
DLZ366I	DLZUCCT0	
DLZ367I	DLZUCCT0	
DLZ369I	DLZUCCT0	
	DLZUC150	
DLZ370I	DLZURGL0	2-32
DLZ371I	DLZUC150	
DLZ372I	DLZURCC0	2-27.1
	DLZLPCC0	
	DLZBACK0	
	DLZUCCT0	
DLZ373I	DLZUC350	
DLZ374I	DLZUC150	
	DLZUC350	
DLZ375I	DLZUC350	
DLZ376I	DLZURGL0	2-32
DLZ377I	DLZURGU0	
DLZ378I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ379I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ380I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ381I	DLZURGU0	2-31
	DLZURGL0	2-32
DLZ382I	DLZURUL0	
DLZ383I	DLZURUL0	
DLZ384I	DLZUCUM0	
DLZ385I	DLZUCUM0	
DLZ387I	DLZURGL0	
DLZ389I	DLZURGL0	2-32
	DLZURRL0	
DLZ390I	DLZUC150	
	DLZLOGP0	
DLZ391I	DLZUDMP0	
	DLZURDB0	
	DLZURUL0	
	DLZURRL0	
	DLZBACK0	
	DLZUC150	
	DLZUC350	
	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
	DLZURGP0	
	DLZUCCT0	
	DLZTPRT0	
DLZ392I	DLZURUL0	

Message Number	Module	Figure Number
	DLZURGU0	2-31
	DLZURRL0	
DLZ393I	DLZURRL0	
DLZ394I	DLZURRL0	
	DLZURDB0	
DLZ395I	DLZBACK0	
DLZ396I	DLZRDBC0	
DLZ397I	DLZRDBC0	
DLZ398I	DLZRDBC0	
DLZ399I	DLZRDBC0	
DLZ400I	DLZURGU0	2-31
DLZ401I	DLZBACK0	
	DLZLPCC0	
	DLZUCCT0	
DLZ402I	DLZBACK0	
	DLZURDB0	
	DLZUC150	
DLZ404I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ405I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ406I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ407I	DLZLPCC0	
	DLZTPRT0	
	DLZURCC0	
DLZ408I	DLZLPCC0	
DLZ409I	DLZLPCC0	
DLZ410I	DLZLPCC0	
DLZ411I	DLZLPCC0	
DLZ412I	DLZLPCC0	
DLZ413I	DLZLPCC0	
DLZ414I	DLZLPCC0	
	DLZURCC0	
	DLZTPRT0	
DLZ415I	DLZLPCC0	
	DLZURCC0	
DLZ416I	DLZLPCC0	2-39.1
DLZ417I	DLZLOGP0	
DLZ418I	DLZLOGP0	
DLZ419I	DLZLOGP0	
DLZ420I	DLZLOGP0	
DLZ421I	DLZLOGP0	
DLZ422I	DLZLOGP0	
DLZ423I	DLZLOGP0	
DLZ424I	DLZLOGP0	
DLZ425I	DLZLOGP0	
DLZ426I	DLZLPCC0	
DLZ427I	DLZLOGP0	
DLZ428I	DLZLOGP0	
DLZ429I	DLZLOGP0	
DLZ430I	DLZLPCC0	
DLZ431I	DLZLPCC0	
DLZ432I	DLZLPCC0	

Message Number	Module	Figure Number
DLZ433I	DLZLPCC0	
DLZ434I	DLZLPCC0	
DLZ440I	DLZTPRT0	
DLZ441I	DLZTPRT0	
DLZ442I	DLZTPRT0	
DLZ443I	DLZTPRT0	
DLZ444I	DLZTPRT0	
DLZ445I	DLZTPRT0	
DLZ476I	DLZDLA00	
DLZ570I	DLZDLBL3	
DLZ571I	DLZUACB0	2-33
DLZ572I	DLZDLBL0	
	DLZDLBL1	
DLZ573I	DLZDLBL0	
	DLZDLBL1	
DLZ583I	DLZUACB0	
DLZ584I	DLZUACB0	
DLZ585I	DLZUACB0	
DLZ587I	DLZUACB0	2-33
DLZ588I	DLZUACB0	2-33
DLZ589I	DLZUACB0	2-33
DLZ772I	DLZDXMT0	
DLZ796I	DLZDLD00	
DLZ797I	DLZDDLE0	
DLZ798I	DLZDLRG0	
	DLZDLRD0	
DLZ799I	DLZDLD00	
	DLZCPY10	
DLZ800I	DLZDLR00	
DLZ801I	DLZDLR00	
DLZ802I	DLZDLD00	
DLZ803I	DLZDLD00	
DLZ804I	DLZDLD00	
DLZ806I	DLZDLD00	
	DLZCPY10	
DLZ807I	DLZDLD00	
DLZ808I	DLZDLD00	
DLZ830I	DLZDHD00	
	DLZGGSP0	
DLZ831I	DLZDHDS0	2-13.5
DLZ841I	DLZDBH00	
DLZ844I	DLZDBH02	
DLZ845I	DLZDBH00	
DLZ847I	DLZDBH00	
DLZ848I	DLZDBH00	
DLZ850I	DLZDDLE0	
DLZ855I	DLZDDLE0	
DLZ860I	DLZDDLE0	
	DLZDXMT0	
DLZ861I	DLZDDLE0	
DLZ862I	DLZDDLE0	
DLZ863I	DLZDDLE0	
DLZ864I	DLZDDLE0	
DLZ868I	DLZDXMT0	
DLZ888I	DLZBACK0	
DLZ894I	DLZBACK0	
	DLZLOGP0	
	DLZURDB0	
	DLZUC150	
DLZ901I	DLZDLBL2	

Message Number	Module	Figure Number
DLZ902I	DLZDLBL2	
DLZ903I	DLZDLBL2	
DLZ904I	DLZDLBL0	
DLZ905I	DLZDLBL0	
	DLZDLBL1	
	DLZDLBL2	
	DLZDLBL3	
DLZ906I	DLZDLBL0	
DLZ907I	DLZDLBL3	
DLZ908I	DLZDLBL3	
DLZ909I	DLZDLBL2	
DLZ910I	DLZDLBL0	
	DLZDLBL1	
DLZ911I	DLZDLBL2	
DLZ912I	DLZDLBL1	
DLZ913I	DLZDLBL1	
DLZ914I	DLZDLBL2	
DLZ915I	DLZDLBL1	
DLZ916I	DLZDLBL1	
DLZ917I	DLZDLBL1	
DLZ918I	DLZDLBL2	
DLZ919I	DLZDLBL2	
DLZ920I	DLZDLBL1	
DLZ921I	DLZDLBL0	
DLZ922I	DLZDLBL1	
DLZ923I	DLZDLBL1	
DLZ924I	DLZDLBL1	
DLZ925I	DLZDLBL1	
DLZ926I	DLZDLBL0	
	DLZDLBL1	
	DLZDLBL2	
	DLZDLBL3	
DLZ927I	DLZDLBL1	
DLZ928I	DLZDLBL1	
DLZ929I	DLZDLBL0	
	DLZDLBL1	
DLZ931I	DLZDLBL1	
DLZ932I	DLZDLBL1	
DLZ933I	DLZDLBL3	
DLZ934I	DLZDLBL2	
DLZ935I	DLZDLBL2	
DLZ936I	DLZDLBL1	
DLZ937I	DLZDLBL1	
DLZ938I	DLZDLBL2	
DLZ939I	DLZDLBL1	
DLZ940I	DLZDLBL2	
DLZ941I	DLZDLBL2	
DLZ942I	DLZDLBL2	
DLZ943I	DLZDLBL2	
DLZ944I	DLZDLBL2	
DLZ945I	DLZDLBL0	
DLZ946I	DLZDLBL2	
DLZ947I	DLZDLBL2	
DLZ948I	DLZDLBL2	
DLZ949I	DLZDLBL2	
DLZ952I	DLZURPR0	
	DLZURGS0	2-35
DLZ953I	DLZURGP0	
DLZ954I	DLZURPR0	2-34
	DLZURGS0	2-35

Message Number	Module	Figure Number
	DLZURG10	2-36
	DLZURGP0	
DLZ955I	DLZURG10	2-36.2, 2-36.4
	DLZURGP0	
DLZ956I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURGP0	
DLZ957I	DLZURGS0	2-35
	DLZURG10	2-36
DLZ958I	DLZURGS0	2-35
	DLZURGP0	
DLZ959I	DLZURGS0	
	DLZURGP0	
DLZ960I	DLZURGP0	
DLZ961I	DLZURPR0	
	DLZURGS0	
	DLZURG10	
DLZ962I	DLZURPR0	2-34
DLZ963I	DLZURPR0	2-34
DLZ964I	DLZURPR0	2-34
DLZ965I	DLZURPR0	2-34
DLZ966I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
	DLZURGP0	
DLZ967I	DLZURGS0	2-35
DLZ968I	DLZURGS0	
	DLZURPR0	
	DLZURG10	2-36
	DLZURGP0	
DLZ969I	DLZURGS0	2-35
DLZ970I	DLZURGS0	2-35
DLZ971I	DLZURGS0	2-35
DLZ972I	DLZURGS0	
DLZ973I	DLZURGS0	
DLZ974I	DLZURGS0	
DLZ975I	DLZURGS0	2-35
DLZ976I	DLZURPR0	2-34
DLZ977I	DLZURG10	2-36.2
DLZ978I	DLZURG10	2-36.2
DLZ979I	DLZURG10	2-36.2
DLZ980I	DLZURG10	2-36.2, 2-36.4
DLZ981I	DLZURG10	2-36.4
DLZ982I	DLZURG10	2-36
	DLZURGP0	
DLZ983I	DLZURGP0	
DLZ984I	DLZURPR0	2-34
	DLZURGS0	2-35
	DLZURG10	2-36
DLZ985I	DLZURPR0	2-34
DLZ989I	DLZURG10	2-36.2
DLZ990I	DLZURGS0	
	DLZURGP0	
	DLZURG10	
DLZ991I	DLZURPR0	

DL/I STATUS CODES/MODULE CROSS REFERENCE

This table cross-references DL/I status codes (in alphabetic order) with the module(s) that can cause that status code to be set. The modules are described in Section 3 of this publication. The status codes are described in Chapter 6 of "DL/I DOS/VS Messages and Codes".

Status Code	Module
AB	DLZDLA00
AC	DLZDLA00
AD	DLZDLA00, DLZODP
AH	DLZDLA00
AI	DLZDLA00, DLZDLD00
AJ	DLZDLA00
AK	DLZDLA00, DLZDLR00
AM	DLZDLA00, DLZDLD00
AO	DLZDLD00, DLZDLR00, DLZDDLE0, DLZCPY10
CA	DLZCPY10
CB	DLZCPY10
CC	DLZCPY10
CD	DLZCPY10
CE	DLZCPY10
DA	DLZDLD00
DJ	DLZDLA00
DX	DLZDLD00
GA	DLZDLR00
GB	DLZDLR00
GE	DLZDLR00
GK	DLZDLR00
GP	DLZDLR00
II	DLZDLR00, 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
V1	DLZDLA00
XD	DLZDLA01
XH	DLZDLA00

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). The entry modules 000 for update and 001 for initial generation of low-level codes have multiple entry points for call statements issued by the user-written application program, that is, a separate entry point for each source language that is supported. All modules have only a single exit point, all lower level modules 002 through 006 are only entered at one point.

All modules assemble and issue DL/I calls. The entry point for DL/I depends on the source language that is identified by the entry point into LLC/CC in DL/I. The language bits in the LLC/CC execution control block (LECB) identify the source language of the application program. If an unexpected status code of DL/I is reported in the appropriate PCB, the error bits in the LECB are turned on, and control is routed back directly to the entry modules 000 or 001.

LLC/CC in DL/I consists of the following modules:

- Module 000 is the entry module for maintenance of low level codes. It passes control to module 002 for execution.
- Module 001 is the entry module for initial generation of low level codes. It passes control to module 002 for execution.
- Module 002 is the common mainline control module. It follows down a hierarchical path of a product structure. For actual explosion, control is passed to module 003. If a particular hierarchical path is exhausted, module 004 is executed to process a parallel path on the same hierarchical level. If all parts on the same level are processed, module 005 steps up one level to identify a parallel path on the higher level. If the original starting level is reached, the complete structure is processed, and control is returned to module 000 or 001. Module 002 also detects loops and executes continuity check recovery in module 006.
- Module 003 explodes a particular part into all its components. Control is passed from and to module 002.
- Module 004 removes the part which has previously been processed from the hierarchical path thus opening a new hierarchical path via the next parent part on the same level. Control is passed from and to module 002.
- Module 005 steps up one level and removes the higher level part from the hierarchical path to open another path. Control is passed from and to module 002. If module 002 is not able to follow a new path on this level, module 005 may be executed repetitively.
- Module 006 handles restoring of old low-level codes if a continuity check is detected. Control is passed to and from module 002.

For a more detailed description, see the relevent HIPO charts at the end of Appendix A.

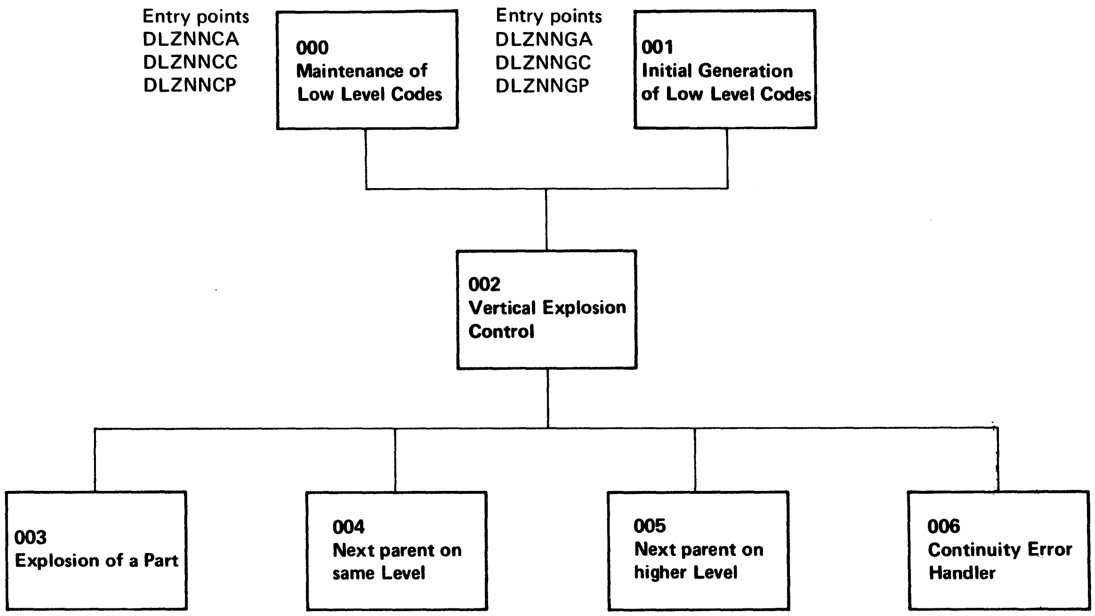


Figure 7-1 Structure of LLC/CC in DL/I

MODIFICATION AIDS

EXTERNAL NAMES

LLC/CC in DL/I uses external names in the directories and libraries of DOS/VS. The following table presents a list of all external names which are used. The user should obtain a DSERV listing to avoid duplicate names.

Type of program	SSL		RL		CIL
	A.books	E.books	Directory entries	Entry points	
Execution program	DLZNN	DLZNN	DLZNN*	DLZNNCA* DLZNNCC* DLZNNCP* DLZNNEC* DLZNNGA* DLZNNGC* DLZNNGP*	
Initialization program for the control data base	DLZNNICT	DLZNNICT			DLZNNICT

* May be modified by the user during customization.

LLC/CC EXECUTION CONTROL BLOCK (LECB)

The LECB of LLC/CC in DL/I is the focal point for all information related to actual operation of the execution program. It consists of 16 bytes which are subdivided into 4 fullwords. An entry point DLZNNEC is provided so that an application program may access the contents of the LECB.

The LECB contains the following information:

1. Identification portion (fullword 0):
   Bytes 0 through 3: C'LECB'=X'D3C5C3C2'
   This identifier facilitates location of the LECB in a main storage dump.

2. Execution control portion (fullword 1):
   Byte 4:

   - Bits 0 through 3: Run type bits
     Bit 0 and bit 1: Reserved
     Bit 2: 1 if IG run
     Bit 3: 1 if U run

   - Bits 4 through 7: Not used

   Byte 5:

   - Bits 0 through 3: Language bits
     Bit 0: Reserved
     Bit 1: 1 if Assembler
     Bit 2: 1 if COBOL
     Bit 3: 1 if PL/I

   - Bits 4 through 7: Not used

   Byte 6: Status byte

   - Bits 0 through 3: Completion bits (mutually exclusive)
     Bit 0: 1 if not completed, abnormal condition encountered
     Bit 1: 1 if component requires no change (U run only)
     Bit 2: 1 if part is already processed (IG run only)
     Bit 3: 1 if part has no components (IG run only, and only if bit 2 is off)

     Besides its function as an indicator, bit 3 also serves to transfer information whether a particular part in an explosion sequence has component parts. Bit 3 is turned off in module 002 before entering module 003. If no component parts are found during the execution of module 003, the bit is turned on. Upon return to module 002, the bit is tested to decide whether module 004 must be called.

   - Bits 4 through 7: Error bits, extending completion bit 0. A single error bit does not reflect a particular error condition, therefore, the hexadecimal representation of the total bit pattern in the status byte has to be analyzed.

     X'80' Parent part not found
     X'81' Component part not found (U run only)
     X'84' Continuity check for parent part
     X'85' Continuity check for any component part
     X'87' Input parameter in error

X'88'	Unexpected DL/I status code for parts data base
X'8A'	Unexpected DL/I status code for control data base
X'8C'	Both error conditions X'84' and X'88'
X'8D'	Both error conditions X'85' and X'88'
X'8E'	Both error conditions X'84' and X'8A'
X'8F'	Both error conditions X'85' and X'8A'

Byte 7: Not used

3. Parameter list portion (fullword 2):

   Bytes 8 through 11: Address constant pointing to the parameter list which has been previously submitted to DL/I by LLC/CC in DL/I. Contents is defined hexadecimal zeros prior to the first run through LLC/CC in DL/I. The address constant is not affected by insertion of locators if the application program is written in PL/I.

4. PCB save area portion (fullword 3):

   Bytes 12 through 15: Address constant pointing to a 64-byte save area for a PCB. This save area is initialized to blanks (X'40'), however, in case of an unexpected DL/I status code, the related PCB is saved into this save area. The PCB is stored left justified. If the length of the PCB exceeds 64 bytes, the exceeding data is truncated.

The contents of the status bytes is externally represented by the return codes of LLC/CC in DL/I.

IG stands for "initial generation of low level codes", U stands for "update of low level codes".

The LECB is located at the very end of the code of LLC/CC in DL/I. Therefore, the last byte of LLC/CC in DL/I may be addressed DLZNNEC+15.

LANGUAGE CONSIDERATIONS

During PSB generation, the source language of application programs using DL/I facilities is defined in the PSBGEN statements. While COBOL is handled like Assembler, the PCB has a different layout if PL/I is specified. Therefore, LLC/CC in DL/I has to use different entry points into DL/I depending on the source language of the invoking user-written application program.

The entry routines of the execution program of LLC/CC in DL/I offer different entry points. The x identifies initial generation mode (G) or update mode (C). Six different entry points are available for transfer of control:

- DLZNNxA and DLZNNxC are the entry points for application programs written in Assembler or COBOL, respectively. No special processing is required.
- DLZNNxP are the entry points for application programs written in the PL/I Optimizer language. Upon entry, the address constants in the parameter list pointing to the locators of the parameters transmitted are replaced by the addresses which are stored in the respective locators.

For each source language, the appropriate language bit in the LLC/CC execution control block (LECB) is set upon entry.

When a DL/I call is issued, the language bits are tested to specify the right entry point in DL/I: ASMTDLI, CBLTDLI, or PLITDLI. If the source language is PL/I, the parameter list is encoded to transfer address constants pointing to locators rather than pointing directly to the parameters.

SAVE AREAS

LLC/CC in DL/I contains a set of save areas which facilitate tracing main storage dumps. The most important save areas are:

- Standard save area, addressed by register 13. Symbolic name is SAVE.
- Return addresses for subroutines, that is, contents of register 14. Symbolic names are CALLSV, PARMJUSV, INSRSAVE, SETUPSV, M002SV through M006SV. Save areas M002SV through M006SV are reset to hexadecimal zeros when the respective modules M002 through M006 are left again.
- Save area for the contents of register 1 when entering LLC/CC in DL/I, that is, address of the parameter list submitted from the application program. Symbolic name is R1SAVE.
- Save area for the leftmost 240 bytes of a PCB if an unexpected DL/I status code is encountered. Symbolic name is PCBSAVE. The address of PCBSAVE is also available in fullword 3 of the LECB.

REGISTER USAGE

R0:	Work register
R1:	Work register, address of parameter lists during parameter transfer
R2:	Address of parameter list when preparing parameter transfer
R5:	Work register
R6:	Address of PCB for parts data base
R7:	Address of PCB for control data base
R8:	Base register
R9:	Second base register
R12:	Reserved
R13:	Address of register save area
R14:	Standard return address
R15:	Standard linkage register

HIPO DIAGRAMS FOR LLC/CC

The following HIPO diagrams describe the seven modules (000-006) of LLC.

Input

INPUT
- PARTS PCB
- CTL.-PCB
- PAR. PART
- COMP. PART

PARTS DB

Processing

This module is entered from an application program via CALL.

[01] Obtain and adjust input data.

[02] Read parent and component part. if not found, go to step 9.

[03] Increment the LLC of the parent part by 1 to obtain the initial LLC. Set actual LLC to initial LLC. if the actual LLC is not higher than the LLC of the component part, no processing is required and control is passed to step 9.

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

[05] Insert dependent segments into the control data base for parent part and for component part.

Output

LLC
- INITIAL
- ACTUAL

CONTROL DB

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

Notes	Routine	Label	Ref
[01] The calling application program uses three different entry points for Assembler, COBOL or PL/I. A parameter list consisting of 6 pointers identifies 6 fields, 4 of them containing input data, 2 of them expecting output data.		DLZNNCA DLZNNCC DLZNNCP	
[05] The original LLC of the component is saved in an UPDMASTR segment. A PARTBEXP segment for continuity check control with a key composed of hexa zeros plus the key of the parent part is inserted. The continuity check itself is explained in note 6 of 002 - VERTICAL EXPLOSION CONTROL. A PARTBEXP segment for explosion control with a key composed of the actual LLC plus key of the component part is inserted.		PARTBEXP	

Notes	Routine	Label	Ref

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-01

Input	Processing	Output
	[06] Replace the old LLC in the component part by the actual LLC.	PARTS DB
	[07] Execute LLC/CC in DL/I.	
	002 VERTICAL EXPLOSION CONTROL 3.1.3	
	[08] Remove all control information from the control data base by deleting the root segment LLCTL with key = X.	CONTROL DB
	[09] Set up return information. Return to calling application program.	OUTPUT: RETURN CD. / LOOP KEY

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref
[09] Return information is obtained from the status bits of the LECB and from the internal loop key field.		DLZNNEC					

000 - MAINTENANCE OF LOW - LEVEL CODES

HIPOMAT 1.1 Diagram - 3.1.1-02

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	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					

001 - INITIAL GENERATION OF LOW - LEVEL CODES HIPOMAT 1.1 Diagram - 3.1.2-01

Input	Processing	Output
	[07] Execute LLC/CC in DL/I.	
	002 VERTICAL EXPLOSION CONTROL 3.1.3	
	[08] Remove all control information from the control data base by deleting the root segment LLCTL with key = X	CONTROL DB
	[09] Set up return information. Return to calling application program.	OUTPUT: RET.-CODE, LOOP KEY

001 - INITIAL GENERATION OF LOW - LEVEL CODES

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

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
[01] Vertical explosion control is performed by means of PARTBEXP segments. Each time a new component part is encountered with a low-level code which needs replacement, a PARTBEXP segment - key = LLC + part key - is created. When going down a product-structure tree, this step of LLC/CC in DL/I identifies a new component part to become a parent part within the recursive process of explosion. Explosion proceeds on a FIFO basis.		PARTBEXP	
[02] During previous explosions, no component part was found requiring the replacement of its current low-level code, or no component part was found at all. Therefore, no segment PARTBEXP was inserted.			
[03] The initial low-level code was established either in module 000 or in module 001, resp.			

Notes	Routine	Label	Ref

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-01

Input

PARTS DB

Processing

04 If a segment is found, read the root segment of the part.

05 Increment the actual LLC by 1 to post into the components of the new part.

06 Perform continuity check and go to step 9. If the check fails, restore old LLC status in 006.

006
ERROR RECOVERY HANDLER
3.1.3.4

07 Upon return from 006, control is passed to step 12.

Output

PART
ACTUAL

LLC
ACTUAL

CONTROL DB

OUTPUT DATA
LOOP KEY
ERROR INFO

002 - VERTICAL EXPLOSION CONTROL   HIPOMAT 1.1 Diagram - 3.1.3-02

Notes	Routine	Label	Ref
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

Input | Processing | Output

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

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 LECB 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.		LECBSNOC					

002 - VERTICAL EXPLOSION CONTROL

HIPOMAT 1.1 Diagram - 3.1.3-03

Input	Processing	Output
PARTS DB	01 Read the first or next component segment of the actual part. If not found, processing is completed and control is passed to step 6.	
LLC ACTUAL	02 Compare actual LLC and LLC in the component. If the actual LLC is not higher, no further processing is required and control is passed back to step 1.	
	03 Save the old LLC of the component in an UFDMASTR segment.	CONTROL DB
	04 Replace the old LLC of the component by the actual LLC.	PARTS DB

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-01

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

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-01

Input	Processing	Output
	[05] Insert a segment PARTBEXP with key composed of actual LLC plus PARTKEY of the component. Go back to step 1. ⟹	CONTROL DB
	[06] Go back to module CC2.	

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-02

Notes	Routine	Label	Ref	Notes	Routine	Label	Ref

003 - EXPLOSION OF A PART

HIPOMAT 1.1 Diagram - 3.1.3.1-02

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

Input

LLC
ACTUAL

CONTROL DB

LLC
ACTUAL

CONTROL DB

PART KEY

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

005 - NEXT PARENT ON HIGHER LEVEL

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

CHART: DLZNN
PAGE 1 OF 1

Input

CONTROL DB

PARTS DB

Processing

01 Read sequentially all UPDMASTR segments.

02 Restore all LLCs of parts referenced by an UPDMASTR segment to its original value.

03 Return to module 002.

Output

PARTS DB

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
SEGM	Allows the user to specify a DL/I segment, its parent segment, the segment length, the segment name, and segment prefix information
LCHILD	Allows the user to define an index relationship or a logical relationship in which a segment will participate.
XDFLD	Allows the user to define secondary indexing relationships.
FIELD	Allows the DL/I user to specify a data field or key field for a segment. The field definition includes the related segment field name, field start position in segment, field length, and field type.
DBDGEN	Causes the segments, fields, and data sets defined in the SEGM, FIELD, and DATASET macro instructions to be generated into an object module.
FINISH	Checks whether a DBDGEN statement was present.

The DBD generation macros utilize a universal set of globals. The COPY book for these globals is in the DOS/VS Source Statement Library and is named DLZDBGLB.

DBDGEN MACRO CALLING SEQUENCE

External Macro	Inner 1	Inner 2
DBD	DLZALPHA	
DATASET	DLZALPHA	
	DLZCKDDN	
	DLZDEVSI	
SEGM	DLZALPHA	
	DLZSOURS	DLZXPARM
		DLZALPHA
		DLZXTDBD
	DLZXPARM	
	DLZSEGPT	
	DLZHIERS	
	DLZXTDBD	
	DLZSETFL	DLZSEGPT
XDFLD	DLZALPHA	
LCHILD	DLZALPHA	
	DLZXTDBD	
	DLZSEGPT	
FIELD		
DBDGEN	DLZSEGPT	
	DLZLRECL	
	DLZSOURS	DLZXPARM
		DLZALPHA
		DLZXTDBD
	DLZXTDBD	
	DLZCAP (See Note)	
FINISH		

Note: Not called if device is FBA.

DBDGEN MACRO – GLOBAL SYMBOL CROSS REFERENCE

GLOBAL SYMBOLS			MACROS																		
NAME	TYPE	SIZE	DATASET	DBD	DBDGEN	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD	FIELD	FINISH	LCHILD	SEGM	XDFLD
ACC	C		U	U	R								R						R	R	
ALIAS	B																		R	U	
BLK	A	10	U								U										
CAPCYL	A				R		U														
CAPTRK	A				R		U														
CDNBR	A		U		U												R			U	
CSB	B				R												R			U	
DBD	B		R	U	R												R		R	R	R
DBDERR	B		S	S	U			S		S	S	S	S	U		S	U	R	S	U	S
DBDTERM	B		R														R		R	U	R
DBN	C			S	R									U		R			R		
DBNAME	C	255			R											U					
DD	A							U													
DDNS	C	255						U													
DEV	C	10	U		R						R										
DEVADR1	C		S		R																
DEVADR2	C		S		R																
DNBR	A		U		R				R												
DSC	A	10			R															U	
DSLKL	A	10			R												U				
DSLSL	A	10			R						R	U									
DSSKL	A	10			R												U				
DSSSL	A	10			R							U									
DS1	C	10	S		R																
DS2	C	10	S								R										
ERROR	B													R	S					R	
EXTDB	A				R											U					
EXTDBN	A				R									R		U			R	R	
F@	A																U			S	U
FBABLK	A	10	S		R																
FBFF	A		S		R																
FD@	A																			S	U
FF	A				R												U			R	U
FLDCH	A	1020			R												U				S
FLDLG	A	1020			R												U				
FLDNM	C	1020			R									R			U			R	U
FLDS#	A	1020			R									R			U				U
FLDSEN	C	1020			R									R						R	S
FLDSQ	B	1020			R												S				
FLDST	A	1020			R												U				
FLDTY	A	1020			R									R			S				
FSPF	A		S		R																
GENCHK	B				S													R			
HDAM	B		S	U	R												R				R
HDB	A		S	S	R																

A = algebraic
B = binary
C = character
R = reference
S = set
U = reference/set

DBDGEN MACRO – GLOBAL SYMBOL CROSS REFERENCE (cont'd)

GLOBAL SYMBOLS			MACROS																		
NAME	TYPE	SIZE	DATASET	DBD	DBDGEN	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD	FIELD	FINISH	LCHILD	SEGM	XDFLD
HDORG	B		U	U	R						R		R				R		R	R	R
HDRBN	A		S	S	R																
HIDAM	B		S	U	R												R		R	R	
HIORG	B		U	U	R							R					R				
HISAM	B		U	U	R																
HSAM	B		S								R									S	
HSEQ	A	16								U											
HSORG	B		U	U	R				R		R						R				
IB	A	255			U														S		
INDLCHD	B				R														S		
INDX	B		U	U	R						R	R					R		R	R	
LCDS#	A	255			R														S	S	
LCFLG	A	255			R														S	S	R
LCLP#	A	255			R														S	S	
LCNM	C	255			R														S	S	
LCPS	C	255			R														S	R	
LCXD	A				R														U	U	U
LEV	A				R															U	
LOGICAL	B		U	U	R									R			R			R	
LP	B												R						R	U	R
MAXCHLD	A			S															R	R	
MAXDMAN	A		R	S																	
MAXFLDS	A			S										R			R				R
MAXSEGS	A			S																R	
NSTRT	A																U			S	
OBLK	A	10	S								U										
OBLKSZ	A	16			U						S										
OLRECL	A	16			U						S										
OREC	A	10	U								R	R									
ORG	A			U	R															S	
OVF	C	10	S		R						R										
PBLKSZ	A	16			U						S										
PLIST	C	100											R	R	S					U	
PLISTK	A	100												R	S					R	
PLRECL	A	16			U						S										
PNBR	A												R		U					U	
QUITB	B		R	R		S								R					R	R	R
RAPS	A		U	S	R																
REC	A	10	U								R	R									
RMN	C		S	S	R																
ROOT	B										R		R				R		R	U	
RTKEY	A				R												S				
S	A				R						R	R	R	R		R	R		R	U	R
S#FLD	A	255			R												U				
S#LC	A	255			R														U	U	R

A = algebraic
B = binary
C = character
R = reference
S = set
U = reference/set

DBDGEN MACRO – GLOBAL SYMBOL CROSS REFERENCE (cont'd)

GLOBAL SYMBOLS			MACROS																		
NAME	TYPE	SIZE	DATASET	DBD	DBDGEN	DLZALPHA	DLZCAP	DLZCKDDN	DLZDEVSI	DLZHIERS	DLZLRECL	DLZSEGPT	DLZSETFL	DLZSOURS	DLZXPARM	DLZXTDBD	FIELD	FINISH	LCHILD	SEGM	XDFLD
S#PC	A	255			R															U	
SCK	A	255															U			U	
SCN	A	10	S		R																
SCRN	C	255			R															S	
SDL	A	255			R						R	R					U			S	
SDS #	A	255			R						R	R								S	
SD1	A	255			R									U							S
SD2	A	255												R							S
SD3	A	255												U							S
SFFLD	A	255			R												U			S	
SFLC	A	255			R															U	R
SFLG1	A	255			R								U						U		
SFLG2	A	255			R								U								
SFLG3	A	255			R															U	
SHISAM	B		U	U	R						R									R	
SHSAM	B		S	U							R									U	
SI	C	255			R														S		
SLC	A				R														U	U	R
SLD	A			S										U							R
SLEV	A	255			R															U	
SLFLD	A																U			S	U
SLSEQ	A																U			S	
SLU	A													R							U
SMINDL	A	255			R												R			S	
SN	C	255			R						R	R		R		R	R		R	U	R
SP#	A	255			R								R						R	S	
SPL	A	255			R						R	R	U						U	U	
SPPP	B	255											U						U		
SPRD	B	255			R															U	
SS	C	255												U							U
SSX	B																U			S	
SVLFLG	A	255			R												R			U	
TRK	A	10	R						S		R	R									
TRK2	A	10							S		R										
TRK3	A	10							S		R										
TRK4	A	10							S		R										
VLC	B												R				R			U	
VV	B												U							U	
XDFLG	A	255			R									R			R				S

A = algebraic
B = binary
C = character
R = reference
S = set
U = reference/set

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	AN AN1 ,FIELD=,CHAR= ALL

This macro tests a specific character position (represented by the CHAR= operand) or all character positions in a specific field (represented by the FIELD= operand) to determine if the character is one of the 39 alphameric characters (A through Z, #, $, @, and 0 through 9). The value range of CHAR is 1 to 255. The default value is 1. The global symbol QUITB is set in the following cases:

- If the positional parameter is not AN, AN1, or ALL and the character is not alphabetic (A through Z, #, $, @).
- If the positional parameter is AN and any chracter is not alphameric (A through Z, #, $, @, or 0 through 9).
- If the positional parameter is AN1 and the first character tested is not alphameric (A through Z, #, $, @, or 0 through 9).
- If the positional parameter is ALL and the first character tested is not alphabetic (A through Z, #, $, @).

DLZCAP MACRO

	DLZCAP	DEVICE, BLOCKSIZ

This macro is called by DBDGEN to calculate the block capacity per track and cylinder provided the blocks do not have keys. These numbers are required to generate some entries within the DTFSD (HSAM) and ACB-extension. The capacities are returned using global arithmetic variables (GBLA). Input values are:

DEVICE: 2314, 3330, 3333, 3340
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	PC,PPC,PLEV

This macro is called by the SEGM macro to validate the hierarchical sequence of segment specifications. The macro maintains a 16-entry table (HSEQ) containing the lowest allowable PC at every level.

The operands are:

PC

specifies segment physical (or sequence) code

PPC

specifies parent physical code

PLEV

specifies parent level

An error message is produced if any of the following conditions exists:

- PC ≠ 1 and PLEV = 0
- PLEV > 14 or PPC > PC
- value of PPC ≠ value of HSEQ table entry represented by PLEV

DLZLRECL MACRO

	DLZLRECL	NUMBER

where NUMBER = 1

This macro is called by DBDGEN to calculate LRECL and BLKSIZE values for the file number specified in the operand field of the macro call.

DLZSEGPT MACRO

	DLZSEGPT	NUMBER

where NUMBER = 1

This macro is called by SEGM, LCHILD, and 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 SEGM330, SEGM340, and SEGM350 if the segment referenced by the operand value violates those rules.

DLZSETFL MACRO

	DLZSETFL	PN,RULES=

This macro processes the POINTER or PTR operand of the SEGM macro and sets the &SFLG1(&S) and &SFLG2(&S) globals to reflect the entered value. The &SFLG1(&S) and &SFLG2(&S) 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 (&SFLG1(&S)) is set as follows:

```
Bit 1 - CTR        If TWINBWD and/or LTWINBWD is specified,
    2 - TWIN       Bit 2 and/or Bit 5 is set on, in
    3 - TWINBWD    addition to Bit 3 and/or Bit 6,
    4 - PARNT      respectively.
    5 - LTWIN
    6 - LTWINBWD
    7 - LPARNT
    8 - NOTWIN
```

Flag Byte 2 (&SFLG2(&S)) 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:

PN

specifies the parent segment number

RULES=

specifies the RULES= operand as specified on the SEGM statement

DLZXPARM MACRO

	DLZXPARM	PARM=,NBR=

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

NBR=

specifies the maximum number of operand values to be allowed in each subparameter

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.

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.

DBD GENERATION CONTROL BLOCK OUTPUT - DBDGEN

The data base description block (DBD) is the result of each data base generation.

- DIAGRAM OF DBDGEN CONTROL BLOCK OUTPUT

GENERAL STRUCTURE:

DIRECTORY
PREFIX
DMANTAB
ACB EXTENSION (SAME AS DMB) (If HSAM or SSAM, DTFs)
SEGTAB
FLDTAB
EXTDBD
LCHILD
SORTAB
INDXTAB
DACT (Same as DMB)
COMPRESSION EXIT CSECTS (same as DMB)
INDEX EXIT CSECTS (same as DMB)

1. DIRECTORY LAYOUT

Hex	Dec	Name	Ln	Description
0	0	AMODLEV	1	Release level (X'00'=1.0, X'11'=1.1)
1	1	APREFIX	3	Address of PREFIX
4	4	ASEGTAB	4	Address of SEGTAB
8	8	AFLDTAB	4	Address of FLDTAB
C	12	ALCHILD	4	Address of LCHILD
10	16	AEXTDBD	4	Address of EXTDBD
14	20	ASORTAB	4	Address of SORTAB
18	24	ARMVTAB	4	Address of DMBDACS
1C	28	AINDXTAB	4	Address of INDXTAB
20	32	ADSGCB	4	Address of ACB extension

2. PREFIX LAYOUT

Hex	Dec	Name	Ln	Description
0	0	PREDBDNM	8	DBD name
8	8	PRENOLEV	2	Number of levels in data base
A	10	PRENOSEG	2	Number of segments
C	12	PREACCES	1	Organization

Name	EQU	Meaning
PRESHIS	X'01'	Simple HISAM
PREISAM1	X'02'	HISAM
PRESSAM	X'04'	Simple HSAM
PREHSAM	X'05'	HSAM
PREHD	X'06'	HDAM
PREHI	X'07'	HIDAM
PRENDEX	X'08'	INDEX

Hex	Dec	Name	Ln	Description
D	13	PRENODSG	1	Number of data sets
E	14	PRENODBD	2	Number of externally referenced data bases
10	16	PRERNDM	8	Randomizing algorithm name
18	24	PRENOLCH	2	Number of logical children
1A	26	PREAP	2	Number of root anchor points
1C	28	DBDPFRBN	4	Maximum relative block number (HD)
20	32	DBDPFBYT	4	Maximum bytes in prime area (HD)

3. DMANTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	PREDD1	8	Input or prime filename
8	8	PREDEV1	4	Device type
C	12	PREID	1	Data set group ID
D	13	PRENSGA	1	Number of segments in data set
E	14	PREDELTA	2	Delta scan cylinders (HD)
10	16	PRELSL	2	Length of longest segment plus prefix
12	18	PRESSL	2	Length of shortest segment plus prefix
14	20	PRELKL	2	Length of longest key
16	22	PRESKL	2	Length of shortest key
18	24	PRELRECL	2	Prime/input record length
1A	26	PREBLKSZ	2	Prime/input block size (control interval)
1C	28	PREOLREC	2	ESDS/output record length
1E	30	PREOBLKS	2	ESDS/output block size (control interval)
20	32	PREDD2	8	ESDS/output filename

4. ACB EXTENSION

See "ACB Extension - ACBXT".

5. SEGTAB LAYOUT

One of these tables exists for each segment.

Hex	Dec	Name	Ln	Description
0	0	SEGDSNO	1	Segment data set number
1	1	SEGPHYCD	1	Segment code
2	2	SEGPARPC	1	Parent segment code
3	3	SEGLEVEL	1	Segment level
4	4	SEGNOLCH	1	Number of logical children
5	5	SEGNOFLD	1	Number of fields
6	6	SEGLENG	2	Segment data length (maximum length if variable length segment)
8	8	SEGFREQ	4	Reserved
C	12	SEGSEGNM	8	Segment name

14	20	SEGFLG1	1	Prefix pointer flag EQU — Meaning X'80' Counter X'40' Physical twin forward X'20' Physical twin backward X'10' Physical parent X'08' Logical twin forward X'04' Logical twin backward X'02' Logical parent X'01' Hierarchical
15	21	SEGFLG2	1	Segment update rules EQU — Meaning Insert rule X'C0' Logical X'80' Physical X'40' Virtual Delete rule X'30' Logical X'20' Physical X'10' Virtual Replace rule X'0C' Logical X'08' Physical X'04' Virtual Physical location of inserts, when no key field X'03' Here (current position) X'02' First X'01' Last
16	22	SEGFLG3	1	X'08' Parent has backward pointers to this segment
17	23	SEGFLG4	1	Number of physical children pointed to directly by this segment
18	24	SEGLCHLD	4	Offset to first LCHILD entry
1C	28	DBDSSN	2	Number of source segments
1E	30	DBDSSOFF	2	Offset to first source segment
20	32	SEGFLDTB	4	Offset to first FLDTAB
24	36	DBDSPFSZ	2	Segment prefix size
26	38	SEGLENGV	2	Minimum segment length (0 if fixed length)
28	40	Reserved	4	Reserved

2C	44	SEGPACOP	1	VL-Compression options

Name	EQU	Meaning
SEGCPRT	X'08'	Segment has compression routine
SEGTYPVL	X'04'	Segment is variable length
SEGPACIT	X'01'	Initialization exit requested for compression routine

2D	45	SEGPACRT	3	Address of compression table

6. FLDTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	FLDNAME	8	Field name
8	8	FLDSTART	2	Start position offset
A	10	FLDFLAG	1	

EQU	Meaning
X'80'	Last field for a SEGTAB
X'40'	Sequence field
X'20'	Multiple sequence fields
X'10'	Special FDB
	Field type
X'01'	Hexadecimal
X'02'	Packed
X'03'	Character
X'04'	Floating point

Hex	Dec	Name	Ln	Description
B	11	FLDLEN	1	Field length
C	12	FLDSNAME	8	Source field name
14	20	FLDSEGTB	4	Pointer to SEGTAB entry

7. EXTDBD LAYOUT

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

8. LCHDTAB LAYOUT

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

Bit	Meaning
0=0	LCHEDBD address is a EXTDBD entry
0=1	LCHEDBD address is a SEGTAB entry

				1-7	Reserved
9	9	LCHEDBD	3	Offset to EXTDBD or SEGTAB entry	
C	12	LCHFLAG	1		
				EQU	Meaning
				X"80"	Last entry for a SEGTAB
				X"40"	Reserved
				X"20"	INDEX entry
				X"10"	Reserved
				X"08"	LP definition
				X"04"	INDEX pointer
				X"02"	SNGL pointer
				X"01"	DBLE pointer
D	13	LCHIBYTE	1	Reserved	
E	14	LCHPRDSG	2	Offset to paired segment	
10	16	LCHFLDNM	8	Indexed field name	

9. SORTAB LAYOUT

Hex	Dec	Name	Ln	Description
0	0	DBDSORNM	8	Source segment name
8	8	DBDSSFLG	1	Source segment flag - reserved
9	9	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	

PSBGEN MACRO — GLOBAL SYMBOL CROSS REFERENCE

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
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	
SFF	A							R	S	R
SN	C	500					R		U	
SP	A	500					R		S	
SPC	A	500					R		S	
SPO	C	500		S			R		S	
SS	A	255				R	R	U	U	U

A = algebraic
B = binary
C = character

R = reference
S = set
U = reference/set

PSBGEN MACRC 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 MACRC

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.

SENFLC MACRO

This is an external macro used to specify sensitive fields within a sensitive segment.

SENSEG MACRO

This is an external macro used to specify sensitive segments in a data base PCB.

VIRFLD MACRO

This is an external macro used to specify fields that exist in the user's view of a sensitive segment, but not in the physical view.

PSB GENERATION CONTROL BLOCK OUTPUT - PSBGEN

1. PSB - PREFIX

Hex	Dec	Ln	Description
0	0	4	Address of SEGTAB
4	4	4	Address of SORTAB
8	8	4	Address of DBREFTAB
C	12	4	Reserved
10	16	4	PST address (prefix size)
14	20	12	Reserved
20	32	1	Reserved
21	33	1	PSB code
22	34	2	PSB prefix size
24	36	2	Reserved
26	38	2	Offset to first DB PCB address
28	40	Var	Address of PCB(s) (one 4-byte address for each PCB)

2. DB PCB

Hex	Dec	Ln	Description
PL/I dope vectors precede PCB if LANG=PL/I			
0	0	8	Data base name
8	8	2	Level feedback
A	10	2	Status code
C	12	4	Processing options
10	16	4	JCB address
14	20	8	Segment name feedback
1C	28	1	Position
1D	29	3	Key feedback length
20	32	2	Number of sensitive segments
22	34	2	Offset to first SENSEG
24	36	Var	Key feedback area

3. SEGTAB ENTRY

Hex	Dec	Ln	Decription
0	0	8	Segment name
8	8	4	Processing options
C	12	1	Flag
D	13	3	PCB address
10	16	2	Offset to parent segment
12	18	2	Offset to FSB list

4. SORTAB ENTRY

Hex	Dec	Ln	Description
0	0	8	Segment name
8	8	1	Flag
9	9	3	Offset to data base entry

5. DBREFTAB ENTRY

Hex	Dec	Ln	Description
0	0	12	Data base name
C	12	4	Reserved

6. FLS TABLE

Hex	Dec	Ln	Description
0	0	4	FSB list address
4	4	4	FSB table address
8	8	4	Field exit routine table address
C	12	4	Field exit routine table length
10	16	4	Initial value table address
14	20	4	Initial value table length

7. FSB LIST ENTRY

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

This section describes the executable processing macros that standardize some processing routines and DSECTS and lists the macros that provide the DSECTs.

DLZBLDL

This macro is used to search the core image libraries to determine if a specified load module is present. Optionally, if the phase is present, the length of it is calculated for the caller. The DOS/VS LOAD macro (TXT=NO) is used to obtain the directory entry information.

OPERANDS

The descriptions and valid parameters for the two keyword operands are as follows:

- PHASE — The name of the phase in the core image library.

  =(reg) — The register specified in parenthesis must point to the 8-byte name (padded with blanks if necessary).

  ='name' — The actual phase name may be specified enclosed in single quotes.

  = label — This is the label of an 8-byte field containing the phase name with any necessary blanks.

  Register 1 is the default which must be loaded with the address of the name.

- LENGTH — Specified if the caller desires the actual length of the load module to be calculated by this macro.

  =(reg) — The register specified in parenthesis will contain the length in binary of the load module as indicated in the directory entry. Register 15 is invalid.

  = label — This is the label of a fullword in the calling program which will contain the length of the found phase on exit.

  If LENGTH is omitted, no length will be calculated.

EXIT CONDITIONS

R15 = 0 — The phase was found and the length, if requested, has been returned.

R15 = 4 — The phase was not found.

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.

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]
```

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:

0DLZNNNI TEXTTEXTTEXTTEXTTEXTTEXTTEXTTEXT

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.

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. There is one entry for every partition defined during system generation, except for the partition where the MPC resides.

DLZTWAB

This macro provides the mapping for the BPC batch partition control information for the DL/I task termination routine under MPS (Multiple Partition Support). This information resides in the BPC's task transaction work area.

DLZXTAB

This macro provides the mapping for the XECBTAB macro DEFINE, DELETE, and CHECK options under MPS (Multiple Partition Support).

DLZXCB1

This macro maps the DLZXCBn1 and the data that follows it. It is used to check data under MPS (Multiple Partition Support).

MACROS USED TO CREATE DSECTS FOR DL/I SYSTEM CONTROL BLOCKS

The following macros are used to generate DSECTS for the DL/I control blocks:

DLZBFFR
DLZBFPL
DLZDDIR
DLZIDLI
DLZPDIR
DLZPPST
DLZPSIL
DLZPST
DLZSCD.

Macros used only by utilities to generate DSECTs:

DLZCKPT
DLZDTF
DLZIDBD
DLZRECO
DLZUCHDR
DLZUCOLD
DLZUCREC
DLZUCUMC
DLZUDHDR
DLZURGUF
DLZURHDR
DLZUSTAT
DLZTRENT.

Miscellaneous macros:

DLZDLIST	Creates parameter list for DLZIDUMP macro
DLZDLP	Log record DSECTs and declarations
DLZHDS0	Work area for DLZDHDS0
DLZIDUMP	IPCS dump hook macro
DLZQUATE	Register equates
DLZSBIF	Work area for DLZDBH00
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.

LY12-5016-5

DOS/VS Logic Manual Printed in U.S.A. LY12-5016-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

DL/I DOS/VS Logic Manual
LY12-5016-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

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

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

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

• What is your occupation? ____________________

• How do you use this publication:

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

Your comments:

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

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

LY12-5016-5

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape — Please Do Not Staple — Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 10 ENDICOTT, N.Y. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

Fold — Fold

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

Your Name ______________________________

Company Name ______________________ *Department* __________

Street Address ______________________________

City ______________________________

State ______________________ *Zip Code* __________

IBM Branch Office serving you ______________________________

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601