SY35-0010-3
File No. S370-30

Systems

OS/VS2 Access Method Services Logic

Release 3.7

Includes Selectable Units:

Supervisor Performance #2 VS2.03.807
Data Management VS2.03.808
MSS Enhancements 5752-824

IBM

Includes Selectable Units:

Supervisor Performance #2 VS2.03.807
Data Management VS2.03.808
MSS Enhancements 5752-824

Fourth Edition (January 1977)

This edition replaces the previous edition (numbered SY35-0010-2) and makes that edition obsolete.

This edition applies to Release 3.7 of OS/VS2 and to any subsequent releases of that system unless otherwise indicated in new editions or technical newsletters.

Significant system changes are summarized under "Summary of Amendments" following the list of illustrations. In addition, miscellaneous editorial and technical changes have been made throughout the publication. Because the technical changes in this edition are extensive and difficult to localize they are not marked by vertical lines in the left margin; the entire edition should be carefully reviewed.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography,* GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of this publication. If the forms have been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150. All comments and suggestions become the property of IBM.

PREFACE

This book describes the internal logic of OS/VS2 Access Method Services routines and provides diagnostic information.9 This information is directed to support personnel and development programmers who require an in-depth knowledge of the program's design, organization, and data areas. It is not required for effective use of Access Method Services.

You should be familiar with general programming techniques, OS/VS2 VSAM concepts and use, TSO concepts and use, and System/370 before reading this book. If you are unfamiliar with these concepts, get and read:

- *OS/VS2 Access Method Services,* GC26-3841, which describes the general syntax of the Access Method Services language, the commands of this processor, and how they are used.
- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which describes the use of VSAM.
- *OS/VS2 TSO Terminal User's Guide,* GC28-0645, which describes how to use TSO commands.

Other books that may be helpful to you are:

- *OS/VS Mass Storage System (MSS) Services Logic,* SY35-0016, which describes the Function Support Routines of Access Method Services for the Mass Storage System.
- *OS/VS2 System Programming Library: Debugging Handbook, Volume 1,* GC28-0708, and *OS/VS2 System Programming Library: Debugging Handbook, Volume 2,* GC28-0709, which describes how to analyze a main storage dump from OS/VS2.
- *OS/VS2 System Programming Library: Service Aids,* GC28-0674, which describes several service aids and programs available under the operating system.
- *OS/VS2 Catalog Management Logic,* SY26-3826, which describes the internal workings of VSAM catalog management and the format of the OS/VS2 system catalog.
- *IBM System/360 Operating System Catalog Management Program Logic Manual,* GY28-6606, which describes the format of OS catalogs.
- *OS/VS2 Virtual Storage Access Method (VSAM) Logic,* SY26-3825, which describes the internal workings of VSAM.
- *OS/VS2 Data Areas,* SYB8-0606, which shows the content of most of the operating system control blocks and tables for OS/VS2.
- *Guide to PL/S-Generated Listings,* GC28-6786, which helps interpret the microfiche listings. The microfiche listings contain both the PL/S and assembly source code. **Note:** *Guide to PL/S-Generated Listings* describes PL/S-1, but Access Method Services is written in PL/S-2.
- *OS/VS2 TSO Command Language Reference,* GC28-0646, which describes TSO commands in detail.
- *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic,* SY28-0650, which describes the logic of the TSO terminal monitor program and service routines.

- *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor,* GC28-0648, which describes TSO interfaces used by the Access Method Services Reader/Interpreter.
- *OS/VS2 System Programming Library: System Generation,* GC26-3792, which describes the use of Access Method Services during system generation.
- *OS/VS2 System Programming Library: Job Management,* GC26-0627, which describes dynamic allocation.

This book is divided into six chapters:

- "Introduction" describes the design philosophy of this processor, and defines terms used later in the book.
- "Method of Operation" describes how the program works. Emphasis is on the flow of data and the technology that is used rather than on the organization of modules.
- "Program Organization" shows how the processor is packaged into load modules. Relationships between the Access Method Services processor and the operating system are given.
- "Microfiche Directory" relates the information in this book to the listings found on microfiche.
- "Data Areas" describes the control blocks and other data areas that are internal to this processor.
- "Diagnostic Aids" shows how to analyze a dump of the processor and how to find specific modules and data areas.

This manual contains information for the following Selectable Units (SUs):

- OS/VS2 MVS Supervisor Performance #2 **(VS2.03.807)**
- OS/VS2 MVS Data Management **(VS2.03.808)**
- OS/VS2 MVS MSS Enhancements **(5752-824)**

In this manual, any references made to an IBM program product are not intended to state or imply that only IBM's program may be used; any functionally equivalent program may be used instead. This manual has references to the following IBM program products:

RACF—Resource Access Control Facility Program Number 5740-XXH

CONTENTS

ILLUSTRATIONS

Figures

Diagrams

SUMMARY OF AMENDMENTS

OS/VS2 MVS Data Management (VS2.03.808)

- A new Access Methods Services command, RESETCAT, is provided for catalog recovery. RESETCAT synchronizes a catalog to the level of the volumes it owns.
- Control block manipulation macros for generating, modifying, testing, and displaying the ACB; RPL, and EXLST control blocks are no longer used during OPEN/CLOSE processing. (Their use in GET/PUT and POINT operations was removed in the previous release.) Access Method Services now processes these VSAM control blocks directly.

OS/VS2 MVS Supervisor Performance #2 (VS2.03.807)

- **Support of Auxiliary Storage Manager (ASM) redesign.** SWAP space data sets can now be defined and preformatted. Also, a new integrity attribute has been added to both SWAP and PAGE spaces.
- **Prose messages for dynamic allocation errors.** The system DAIRFAIL routine is invoked to provide prose error messages for errors received from dynamic allocation.
- **Support of Resource Access Control Facility (RACF).** Changes are included to support protection of VSAM data sets through RACF.
- **TSO Service Routine Linkage.** Interface to TSO service routines has been changed; the CALLTSSR macro replaces the use of LINK and LOAD macros.

OS/VS2 MVS MSS Enhancements (5752-824)

- The System Adapter has been updated to support MSS Release 3. A new Umacro, URACHECK, has been added.

Release 3.7

Miscellaneous maintenance corrections and clarifications have been made throughout the manual. In addition, support has been included for:

- **Revised LISTCAT format.** LISTCAT output is now printed in a new tabular format, making a significant improvement in readability.
- **ALTER Error Checking.** Additional error checking is performed by ALTER to detect incompatibilities between the object to be altered and the attributes specified in the command.
- **Mass Storage System Release 2.** Two new Umacros, ULOCATE and UNNCATALOG, have been added to the System Adapter. A new function, data set services, has been added to the Volume Services routine. Other miscellaneous changes have been made to the System and I/O Adapters to support new MSS enhancements.
- **Mass Storage System REPAIRV Utility.** New functions have been added to the UEXCP macro to support the REPAIRV utility: OPENR opens data

sets, VTOCs, VTOC headers, and staging packs; READCNT, SPACCR, and READKD assist with retrieving data; and WRITEREC and FWRITE assist with writing data.

Enhanced VSAM

Following is a summary of major changes to Access Method Services for the release of enhanced VSAM and includes support for:

- Alternate Indexes
- Relative Record Data Sets
- Spanned Records
- Catalog Recovery
- Reusable Data Sets
- Miscellaneous Enhancements

Each of these affects one or all of the sections in the manual: method of operation, microfiche directory, data areas (mostly for new or changed Function Data Tables—FDTs), and diagnostic aids (in particular, trace and dump points, error codes, and message-to-module cross reference).

Alternate Indexes

Alternate indexes have been added for key-sequenced and entry-sequenced data sets to provide alternate paths through which to gain access to data. They change the method of operation diagrams for DEFINE CLUSTER, ALTER, DELETE, EXPORT, IMPORT and LISTCAT and add diagrams for the new commands, DEFINE ALTERNATEINDEX, DEFINE PATH, and BLDINDEX.

Relative-Record Data Sets

The relative-record data set brings to three the number of VSAM data sets. It changes the method of operation diagram for DEFINE CLUSTER, PRINT, REPRO and I/O Adapter.

Spanned Records

A record in a key-sequenced or entry-sequenced data set is no longer limited by control-interval size, but can span control intervals. Spanned records change the method of operation diagram for DEFINE CLUSTER.

Catalog Recovery

The user can specify when he defines a catalog that a catalog recovery area (CRA) is to be built for it. A CRA contains information that can be used to recover a damaged catalog. Catalog recovery changes the method of operation diagrams for DEFINE MASTERCATALOG and DEFINE USERCATALOG. It adds diagrams for the new commands, LISTCRA, EXPORTRA, and IMPORTRA.

Reusable Data Sets

Data sets defined as reusable can be reused without deleting and redefining them. They change the method of operation diagram for DEFINE CLUSTER.

Miscellaneous Enhancements

Relatively minor changes have been incorporated into several Access Method Services functions. DEFINE includes default key and record size values and supports an exception exit. REPRO permits copy operations into nonempty key-sequenced data sets. IMPORT supports import operations into empty data sets. EXPORT allows a variable blocksize for portable data sets.

Release 3

Access Method Services has been extended to support:

- The IBM 3850 Mass Storage System. Existing function support routines have been changed to process staging parameters. Several Umacros have been added to the adapters.
- The CHKLIST utility program for listing tape data sets that were open at a checkpoint. A function support routine has been added to process the CHKLIST command.
- Catalog unload/reload. The REPRO function support routine has been changed for catalog unload/reload.
- LISTCAT output modifications. The LISTCAT function support routine has been changed to process new parameters for LISTCAT and to print selected information for the TSO user.

These extensions have caused modifications throughout all sections of the book.

New function support routines for the Mass Storage System Access Method Services commands for space management in the Mass Storage System are described in *OS/VS Mass Storage System (MSS) Services Logic.*

INTRODUCTION

OS/VS2 Access Method Services are utility-like functions required to establish and manage VSAM (Virtual Storage Access Method) data sets. Access Method Services allows you to define, print, delete, or copy VSAM data sets, convert ISAM or SAM data sets into VSAM data sets, build alternate indexes, recover data and catalog entries in the event of a catalog failure, alter or list the entries in a VSAM catalog, create portable (or backup) copies, and convert entries in a OS/VS catalog to entries in a VSAM catalog. Features of its logic are:

- The processor is organized into *executable* and *non-executable* modules. An executable module contains instructions that can be performed by the computer. A non-executable module contains nothing that can be performed by the computer. In Access Method Services all descriptive information—such as, command descriptors—and static text— such as, messages—are centralized in non-executable modules. (In Access Method Services, there is generally a one to one correspondence between modules and csects. Consequently, this publication generally discusses modules. A major exception is IDCAMS. For more information on IDCAMS, see "Program Organization.")
- All external interfaces to Access Method Services are isolated in a small set of modules. Changing these modules allows this processor to run with another operating system or with access methods other than those supported by this release of Access Method Services.
- Each module serves just one purpose and is coded to most efficiently accomplish that purpose.

This book does not discuss VSAM or TSO, their concepts or their data areas. For a discussion of VSAM, see the publication OS/VS *Virtual Storage Access Method* (VSAM) *Programmer's Guide* For a discussion of TSO, see the publication *OS/VS2 TSO Terminal User's Guide.*

This book does not discuss the FSRs (Function Support Routines) for the IBM 3850 Mass Storage System. For a description of those FSRs, see the publication *OS/VS Mass Storage System (MSS) Services Logic.*

The Access Method Services processor accepts commands and sometimes input data sets or catalogs. It produces output data sets and/or printed reports. Details of the commands and the use of Access Method Services are found in *OS/VS Access Method Services.*

Requirements

This processor requires OS/VS2 as its operating system. The processor executes as a problem program. Virtual Storage requirements for the processor are found in *OS/VS2 Enhanced System Information.*

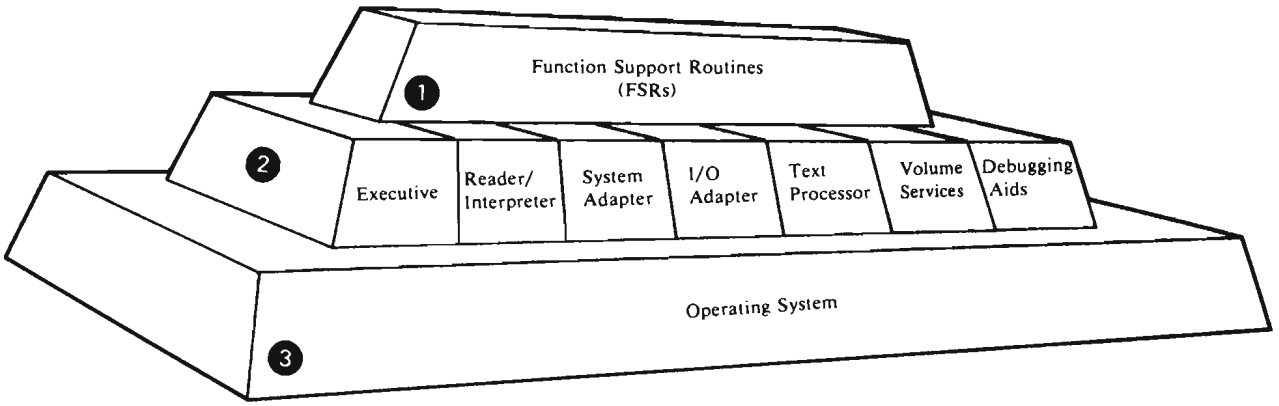
The Access Method Services Processor

Figure 1 describes the structure of the processor. Figures 2 through 4 describe in general how the processor functions. In Figures 1 through 4 paragraph numbers match numbers in the illustrations to coordinate the paragraphs and parts of the illustrations.

Figure 1 shows the executable elements of the Access Method Services processor as they form a structure within the operating system. As shown here, six of the elements form a "substructure" that supports the remaining elements, which form a "superstructure."

Following the flow of logic reveals more of the processor than the structure of executable modules. Figure 2 and the two which follow show the sequence in which modules execute, important internal tables, and how non-executable modules are used.

The System Adapter is the external entry and exit point for Access Method Services. At entry time, the *GDT* (Global Data Table) is built by the System Adapter. The GDT is always passed as a parameter when any internal module is called, and through the GDT can be found the entry point for any service supplied by the substructure. The GDT contains the addresses for the various services provided by the System Adapter, the I/O Adapter, and the Text



1. The superstructure consists of the *FSRs* (Function Support Routines). There is one FSR for each command verb of Access Method Services. Any system interface or I/O function that is required by one of the FSRs is supplied through the substructure. The superstructure is thus insulated from the operating system by the substructure.

2. The substructure consists of the *Executive*, the *Reader/Interpreter*, the *System Adapter*, the *I/O Adapter*, the *Text* and the *Debugging Aids*. The Executive routes control between the other components of Access Method Services—specifically, between the Reader/Interpreter and the FSRs. The Reader/Interpreter translates the commands for Access Method Services into an internal form called the *FDT* (Function Data Table). The System Adapter similarly provides *all* system interfaces for the processor. The I/O Adapter issues *all* I/O operations at the behest of any other routine in Access Method Services. The Text Processor prepares *all* printed materials, whether simple messages or listings, that are required to fulfill a command. The Volume Services provide all the services for processing volume labels and VTOCs for the Mass Storage System. (However, they are not designed for exclusive use by the Mass Storage System.) The Debugging Aids write diagnostic information when requested.

3. The operating system supports the Access Method Services processor, just as the substructure supports the superstructure (the FSRs). However, the FSRs execute in total independence of the actual operating system in which Access Method Services is running. All requests for system services or I/O are made to the substructure, which received the request and issues the appropriate request to the operating system. Thus additional access methods can be easily supported by Access Method Services, by merely augmenting the I/O Adapter appropriately. Access Method Services can be run in a different host operating system by changing the System Adapter and the I/O Adapter to match the new host.

Figure 1. The Structure of the Access Method Services Processor

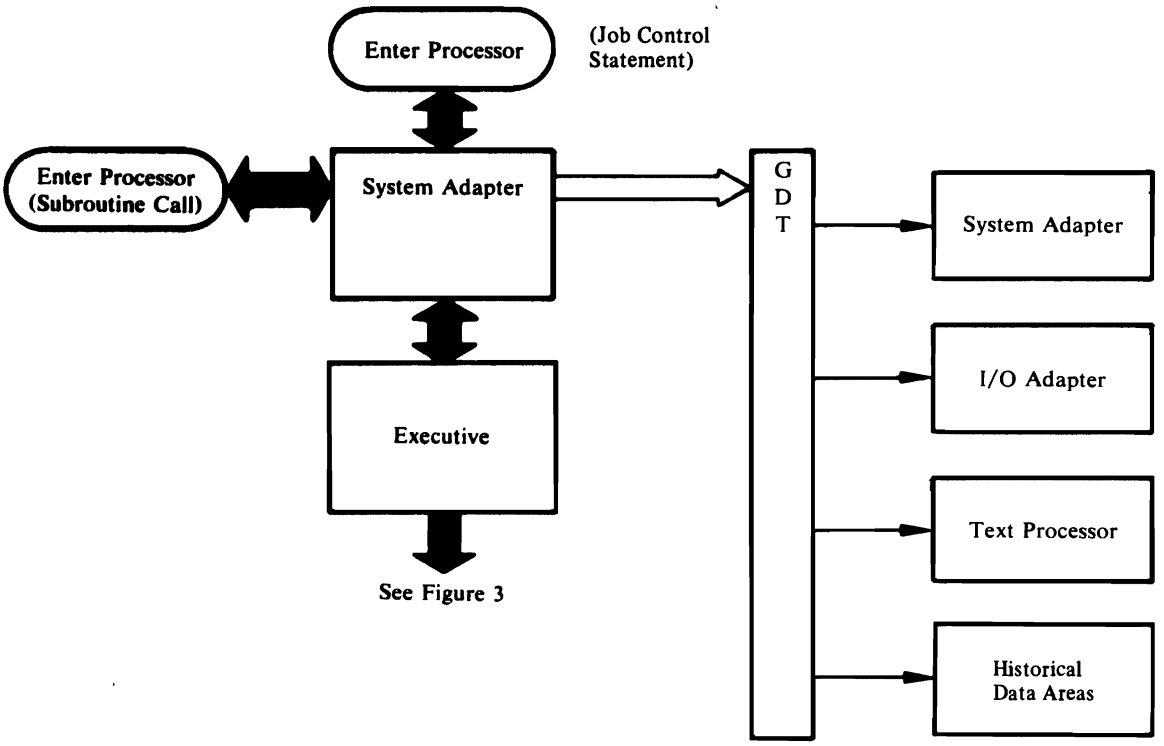


Figure 2. Initialization of Access Method Services

Processor. The GDT also points to historical data areas that are built and maintained by various processor substructure modules.

Control passes from the initialization effected by the System Adapter to the Executive. Figure 3 shows this transfer of control, and details the parsing operation of the processor.

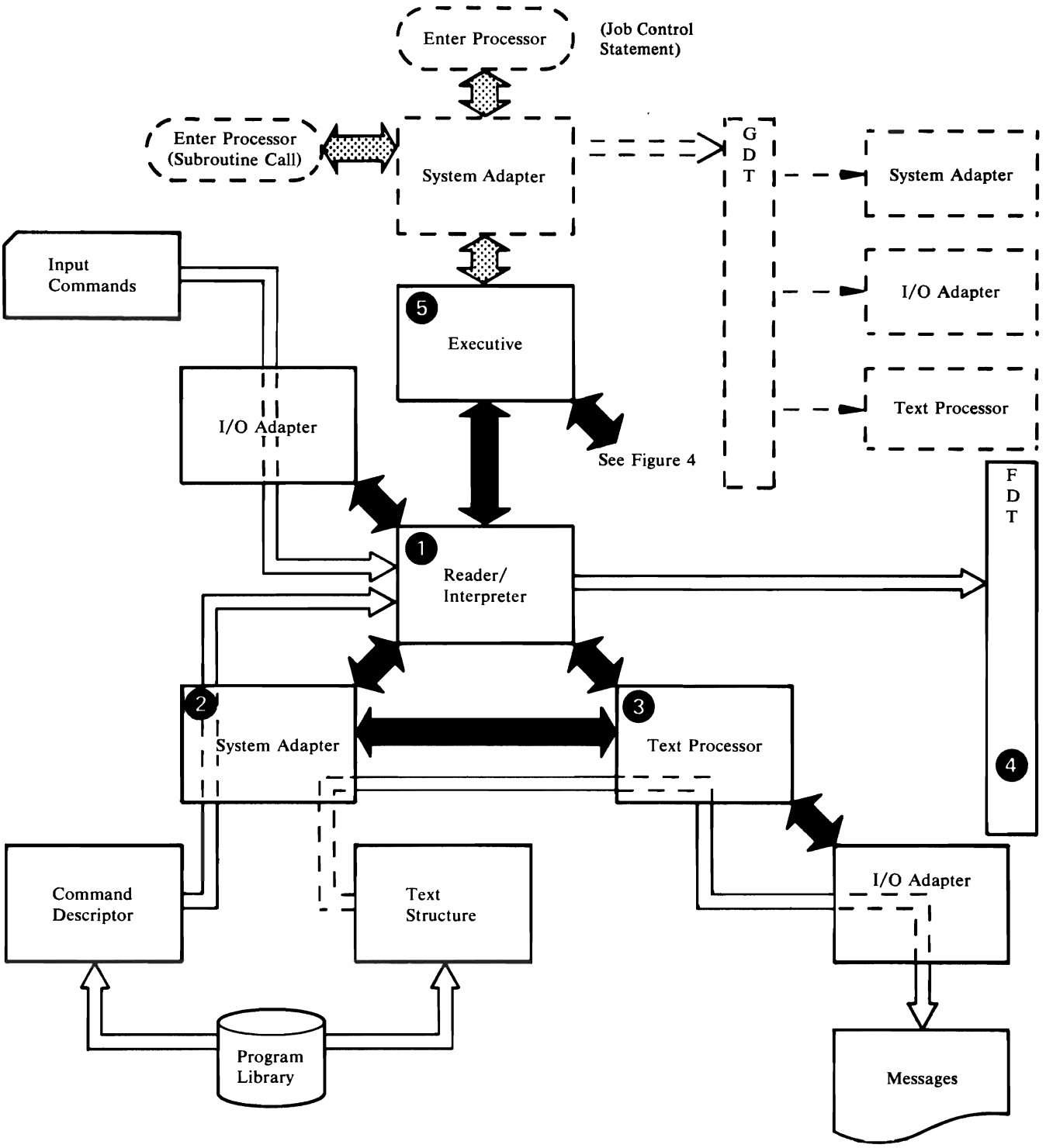


Figure 3 (Part 1 of 2). Reading and Parsing a Command

1. The Executive calls the Reader/Interpreter, which reads a command from the input stream. The I/O Adapter performs the actual read at the behest of the Reader/Interpreter; the address for the "get" service is found in the GDT.

2. To parse the command, the Reader/Interpreter compares it against a special table called a *Command Descriptor.* This Command Descriptor forms a non-executable load module, and is loaded from the program library by a service of the System Adapter. There is a Command Descriptor for each possible verb to be recognized by Access Method Services. This Command Descriptor specifies each possible keyword, its permitted range of values, and any other information that is needed to parse and interpret the command.

3. As a command is parsed, certain messages may be issued. To format these messages, the Text Processor is invoked (again through to GDT). The Text Processor determines the format of printed material and the text of fixed messages by using *Text Structures.* These Text Structures are also non-executable load modules (loaded by the System Adapter when needed), and they describe page layout, static portions of the text, headings, footings and other details of the printed page. Once a line of message is formatted, the I/O Adapter writes the line to the print file.

4. As a command is parsed, the Reader/Interpreter builds an *FDT* (Function Data Table) from the values that it finds. The FDT is an encoded representation of the user's command. The FDT is passed back to the executive as the results of the parse. The Executive in turn passes the FDT to the appropriate FSR for processing.

   Control returns to the Executive, along with the FDT and the name of the FSR needed to process this command. Figure 4 depicts the FSR in action.

Figure 3 (Part 2 of 2). Reading and Parsing a Command

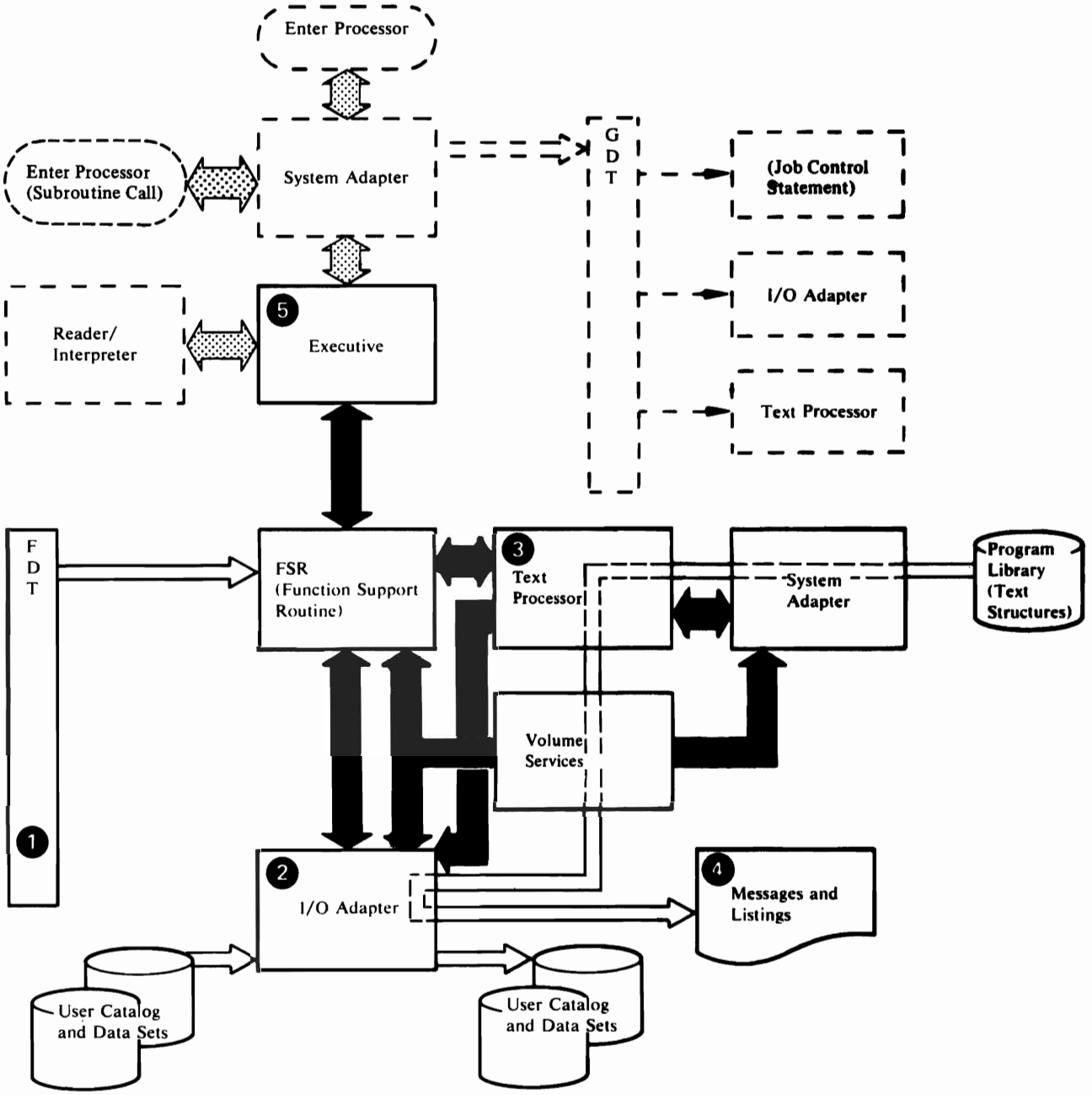


Figure 4 (Part 1 of 2). Performing a Function

1. The command at this point in time is described in the FDT. The FDT is an internal encoding of the original command, in a rigorous format with the values for all possible parameters in a prescribed order.
2. Any data sets or user catalogs required for this particular function are accessed through the I/O Adapter. The address of this service is found in the GDT.
3. Any printed output is prepared by the Text Processor, whose addresses are also found in the GDT. Static text and page layout instructions are found in the Text Structures, which are loaded by the System Adapter.
4. Finally, all output is produced by another of the services of the I/O Adapter.
5. Control returns to the Executive. If more commands remain, the Reader/Interpreter repeats its procedure, followed by the appropriate FSR. Control is routed back and forth between the Reader/Interpreter and the FSRs by the Executive in this fashion until all commands have been processed.

Figure 4 (Part 2 of 2). Performing a Function

Naming Conventions

The Access Method Services processor is named IDCAMS. The names of all modules that form this processor are seven or eight characters long, and begin with the characters IDC. The remaining characters of the name relate to its use. Executable load modules and Command Descriptors have seven-character names, while Text Structures have eight-character names.

The modules of the processor are grouped by their functional relationship. Each of these relationships is indicated by a two-character mnemonic identifier, which appears as characters 4 and 5 of the module name. These identifiers are listed in the following table:

AL	ALTER FSR	MP	IMPORT FSR
AM	Interactive TSO Interface	PM	PARM FSR
BI	BLDINDEX FSR	PR	PRINT FSR
CC	CNVTCAT FSR	RC	EXPORTRA FSR
CD	Command Descriptor	RI	Reader/Interpreter
CK	CHKLIST FSR	RM	IMPORTRA FSR
DB	Debugging Facility	RP	REPRO FSR
DE	DEFINE FSR	RS	RESETCAT FSR (**VS2.03.808**)
DL	DELETE FSR	SA	System Adapter
EX	Executive	TP	Text Processor
IO	I/O Adapter	TS	Text Structure
LC	LISTCAT FSR	VS	Volume Services
LR	LISTCRA FSR	VY	VERIFY FSR
		XP	EXPORT FSR

The remaining characters of a module name indicate the function of that module. Two numeric digits are used for the name of a load module and the entry point of a single-entry module. Two alphabetic characters indicate an entry point in a multiple-entry module. Thus the name "IDCPR01" is the name of the first load module for the PRINT FSR, and "IDCPR01" is the only entry point to that module. "IDCSA02" is the second load module for the System Adapter, and "IDCSAGS" is the entry point in that module for the "get space" service.

Note: The only exception to the naming convention for entry points is the module IDCSA01. It's entry points are IDCSA01 and IDCSATO.

The last two characters of a Command Descriptor are the mnemonic identifier for the FSR for that Command Descriptor. Similarly, Text Structure names end with the FSR mnemonic identifier and a single digit (to allow for multiple Text Structures per FSR). For example the three modules for PRINT are:

IDCPR01	PRINT FSR load module
IDCCDPR	PRINT Command Descriptor
IDCTSPR0	First Text Structure for PRINT

Names for processor-wide data structures and fields are six characters long. The first three characters identify the structure. The last three characters indicate the function of the field. (In this publication, the data areas are often referred to by the first three characters.) Values for a field (for example, a bit in a flag field) have names that are eight characters long. The last two characters of a value indicate the meaning of that value. For example, "IOCDSO" is a field of the I/O Communications Structure that defines the data set organization. One of its bits is named "IOCDSOAM," which means that this bit signifies a VSAM organization.

Local names used internally by only one subcomponent follow no processor-wide conventions.

Character Code Dependencies

Most of the character dependencies of this processor are isolated in the Command Descriptor modules and the Text Structure modules. For example, all input text is translated by referring to the Command Descriptor modules, and all output text is controlled by the Text Structure modules and a parameter defining the output graphics.

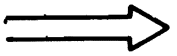
Most of the executable modules of the processor have no character dependencies. However, some modules of the Reader/Interpreter and the Text Processor have character dependencies. Such character dependencies are identified in the prologue of each module.

The character set used at execution time must be equivalent to that used during assembly of the character-dependent modules. The IBM-supplied version of these modules assumes EBCDIC character representations. If a different character representation is to be used during execution, then the character-dependent modules must be re-assembled.

METHOD OF OPERATION

This chapter contains method of operation diagrams for each element within the substructure and superstructure of Access Method Services. Following each diagram is an extended description of the processing steps and the name of the modules and procedures used to perform each step within the diagram. Using these names, you can go either to the chapter "Microfiche Directory" or to the microfiche itself for more information.

The following legend explains the symbols used throughout this chapter.

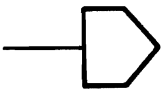


Data flow

Flow of control, entry and exit points

Data flow when existing data has been altered

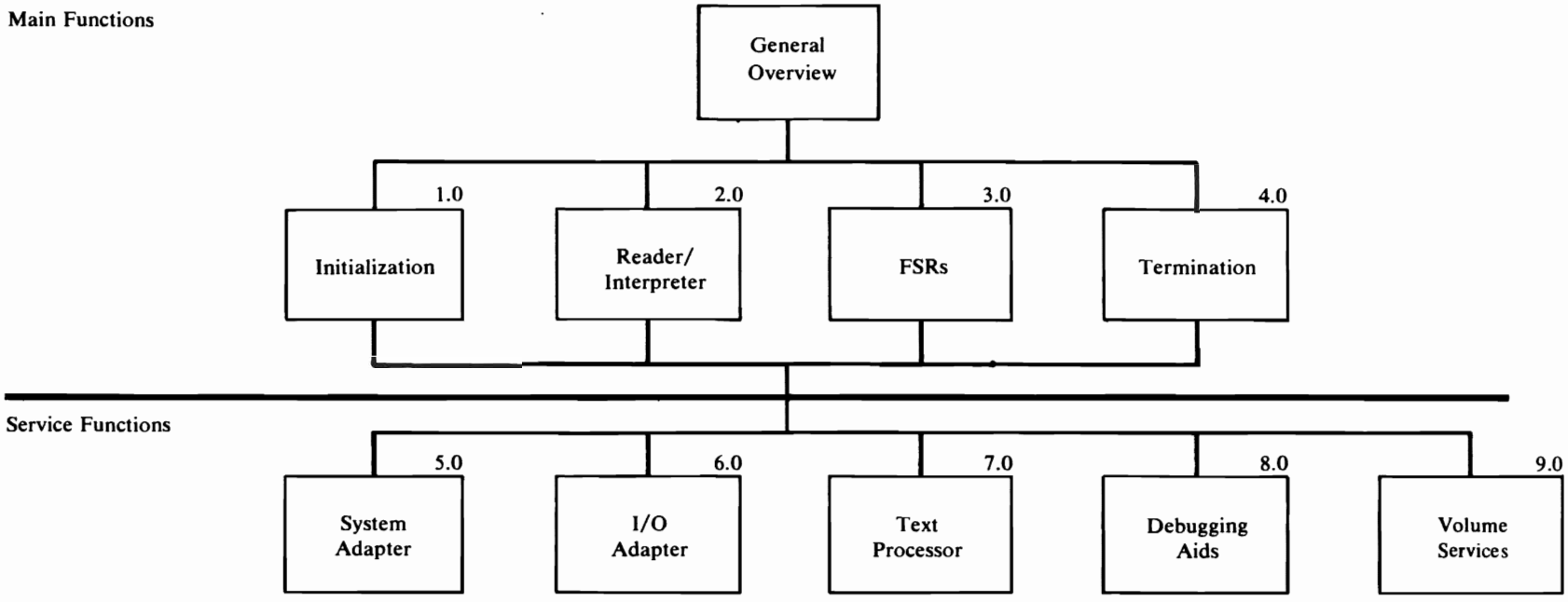
On page connector



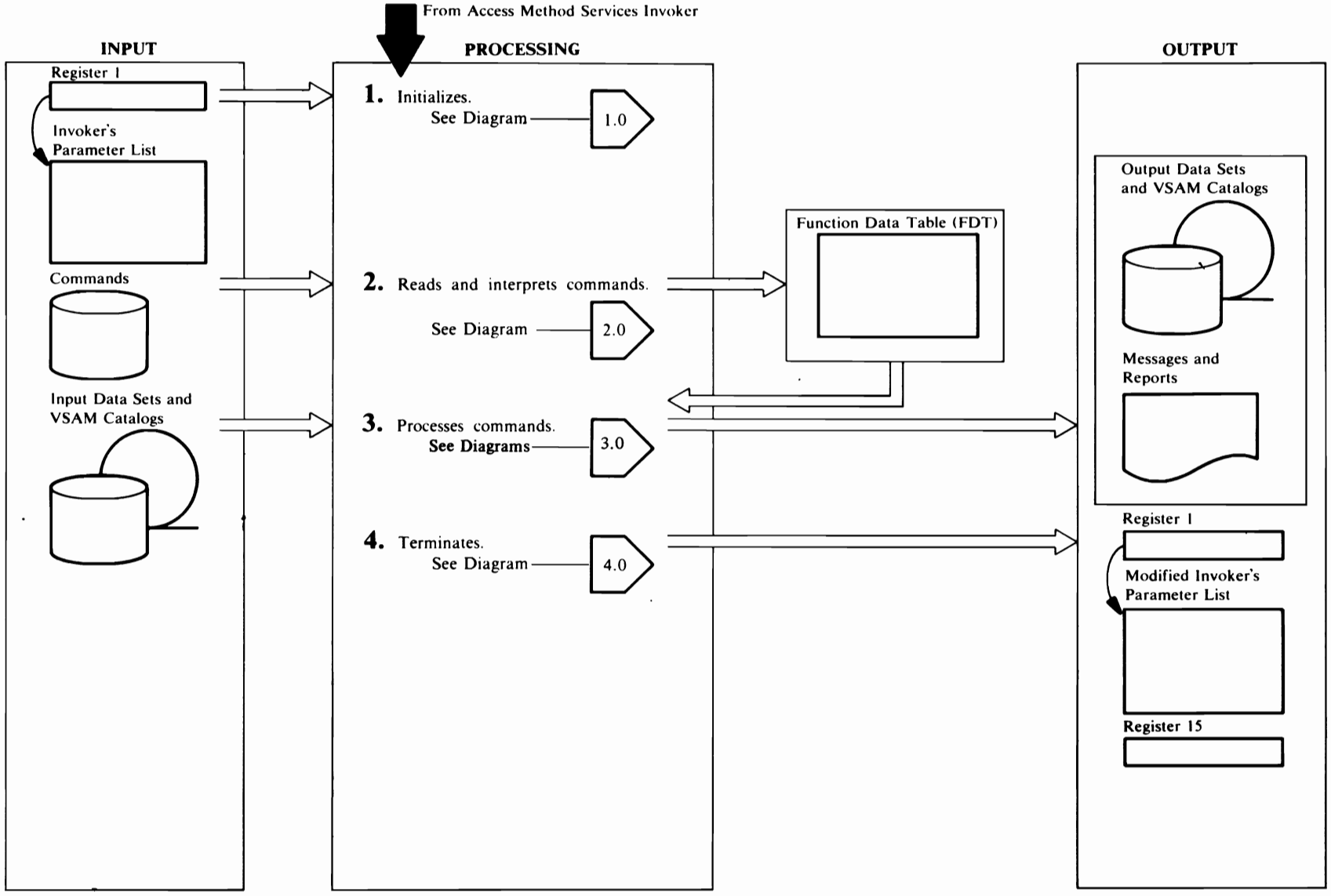
Off page connector

Pointer to more information

Access Method Services Visual Table of Contents



Access Method Services Overview



Initialization Visual Table of Contents

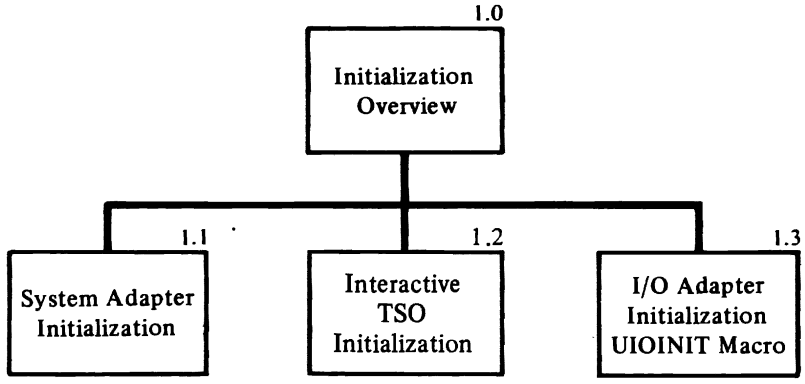
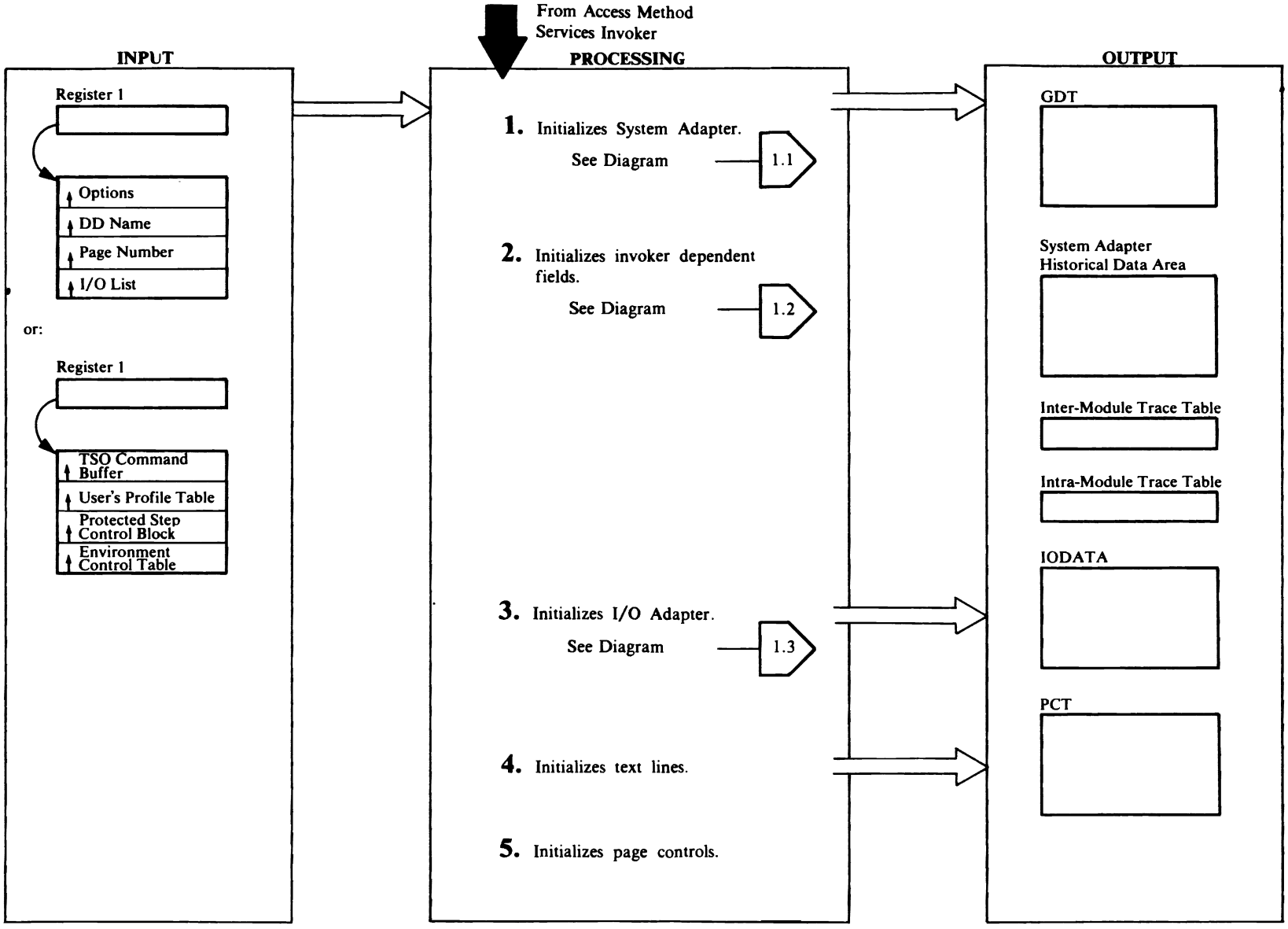


Diagram 1.0 Access Method Services Initialization Overview



Extended Description for Diagram 1.0

Module: IDCSA01, IDCAM01, IDCAM02

Procedure: IDCSA01, IDCAM01, IDCAM02

1. Access Method Services can be invoked with a batched job—either through JCL or a program—or invoked interactively with TSO. A batched job invokes the System Adapter at entry point IDCSA01. When a TSO terminal user uses an Access Method Services command, IDCAM01 or IDCAM02 receive control. IDCAM01 and IDCAM02 contain identical code. IDCAM01 or IDCAM02 invokes the System Adapter at entry point IDCSATO. The System Adapter sets up the GDT, trace tables, and the System Adapter Historical Data Area. The System Adapter obtains storage for modules that are continuously used, such as the System Adapter and the I/O Adapter. Diagram 1.1 shows System Adapter initialization in detail.

Module: IDCSA01

Procedure: IDCSA01

2. The System Adapter initializes invoker dependent fields.
   - If Access Method Services was invoked with a batched job, the System Adapter puts the address of the Reader/Interpreter name 'IDCRI01' in GDTRIP of the GDT.
   - If Access Method Services is invoked interactively with TSO, the System Adapter puts the address of the Reader/Interpreter name 'IDCRI04' in GDTRIP of the GDT. The System Adapter initializes the GDT for processing with interactive TSO. See Diagram 1.2 for more information on interactive TSO initialization.

Module: IDCEX02

Procedure: IDCEX02

3. IDCEX02 issues the UIOINIT macro to cause the I/O Adapter to initialize. The I/O Adapter initializes its Historical Data Area. IDCIOIT saves the addresses of alternative DD name list if supplied by the invoker. Diagram 1.3 shows I/O Adapter initialization in detail.

Module: IDCEX02

Procedure: IDCEX02

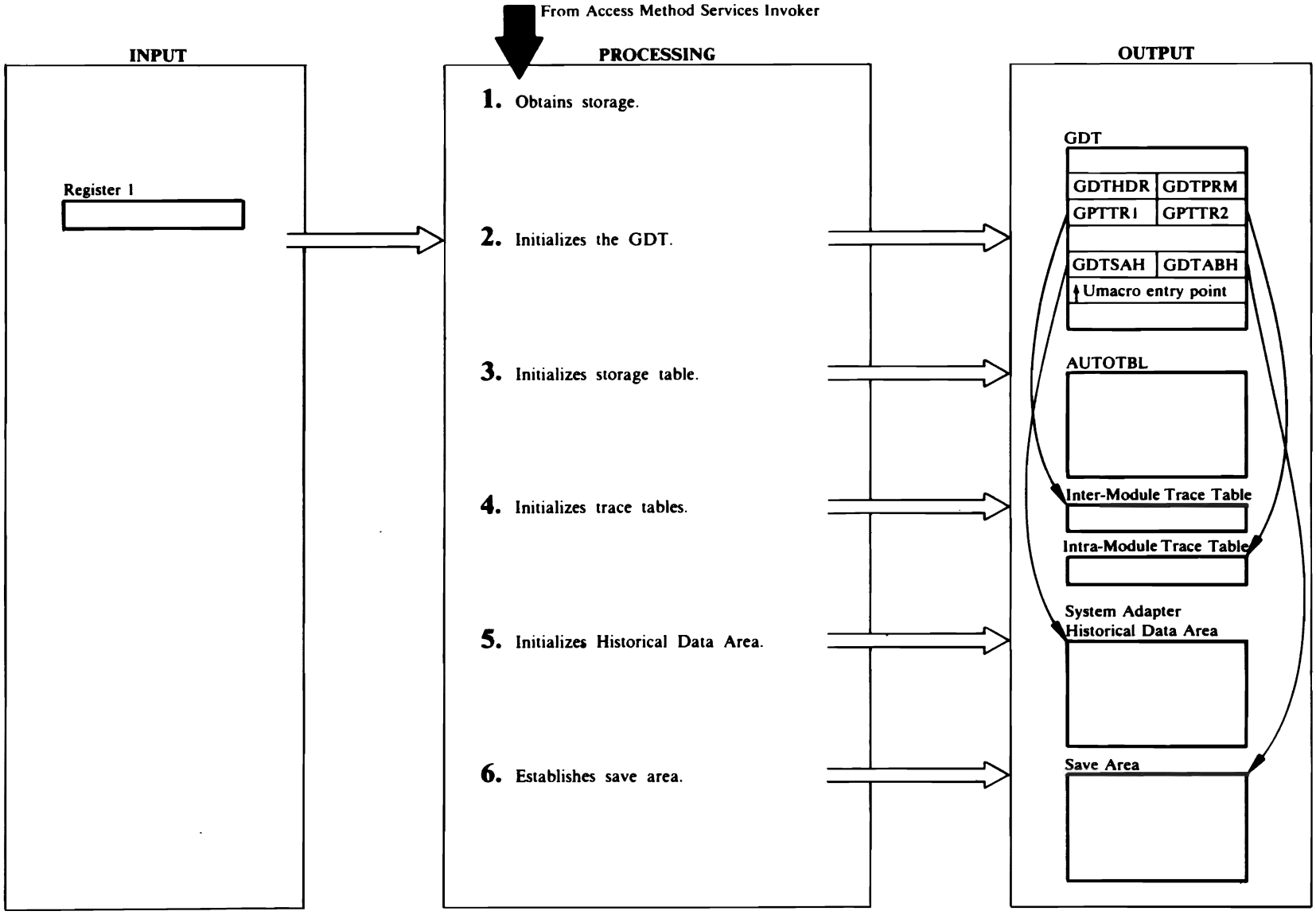
4. IDCEX02 issues a UESTS macro instruction to set up the Print Control Table, PCT. The address for the Text Processor Historical Data Area is in the GDTTPH field of the GDT. Since GDTTPH contains zero, the text processor builds the primary PCT.

Module: IDCEX01, IDCEX02

Procedure: CALLRI, IDCEX02, SCANPARM

5. If the invoker supplied a starting page number in the parameters, IDCEX02 issues a UREST macro instruction to set the page number. IDCEX01 uses the name of the Reader/Interpreter addressed from GDTRIP in the GDT to load the Reader/Interpreter. Control is given to the Reader/Interpreter to process the input as well as any parameters supplied on the EXEC statement that invoked Access Method Services.

Diagram 1.1 System Adapter Initialization



Extended Description for Diagram 1.1

Module: IDCSA01

Procedure: IDCSA01

1. IDCSA01 issues a GETMAIN instruction to obtain space for the following tables:
   - Global Data Table, GDT
   - Inter-Module Trace Table
   - Intra-Module Trace Table
   - System Adapter Historical Data Area
   - Storage Table, AUTOTBL

Module: IDCSA01

Procedure: IDCSA01

2. IDCSA01 puts the characters 'GDTb' in the first four bytes of the GDT. It puts the address of the invoker's parameter list, which is in register 1, in the GDTPRM field of the GDT. IDCSA01 puts the address of the System Adapter Historical Data Area in GDTSAH. It also puts the address of the Inter-Module Trace Table in GDTTR1 and the address of the Intra-Module Trace Table in GDTTR2. IDCSA01 puts the address of the System Adapter save area in GDTABH. Additionally, it puts addresses for the processor-defined macro instructions, called Umacros, in the GDT. All remaining fields of the GDT contain zeros.

Module: IDCSA01

Procedure: IDCSA01

3. Rather than obtaining new storage each time IDCSA02, IDCSA03, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IFCSA10, IDCTP01, IDCIO01, or IDCIO05 is called, the System Adapter issues one GETMAIN macro for each module and saves the storage address in the Storage Table, AUTOTBL. When one of the modules is called, it calls the PROLOG routine that returns the address of the storage obtained for the module during System Adapter initialization. The storage address for IDCSA03, however, is kept in the GDTSPR field of the GDT because IDCSA03 contains the PROLOG routine code and needs to get its storage without using the PROLOG routine. **For VS2.03.807:** IDCSA01 initializes the Load List Block (LLBLK) and puts a pointer to the LLBLK in the System Adapter Historical Area.

Module: IDCSA01

Procedure: IDCSA01

4. IDCSA01 initializes the Inter-Module Trace and Intra-Module Trace Tables to blanks. It places the characters 'bINTERbb' and 'bINTRAbb' before the respective tables. It also puts the characters 'SA01' in the Inter-Module Trace Table and in the save area provided by the Access Method Services invoker.

Module: IDCSA01

Procedure: IDCSA01

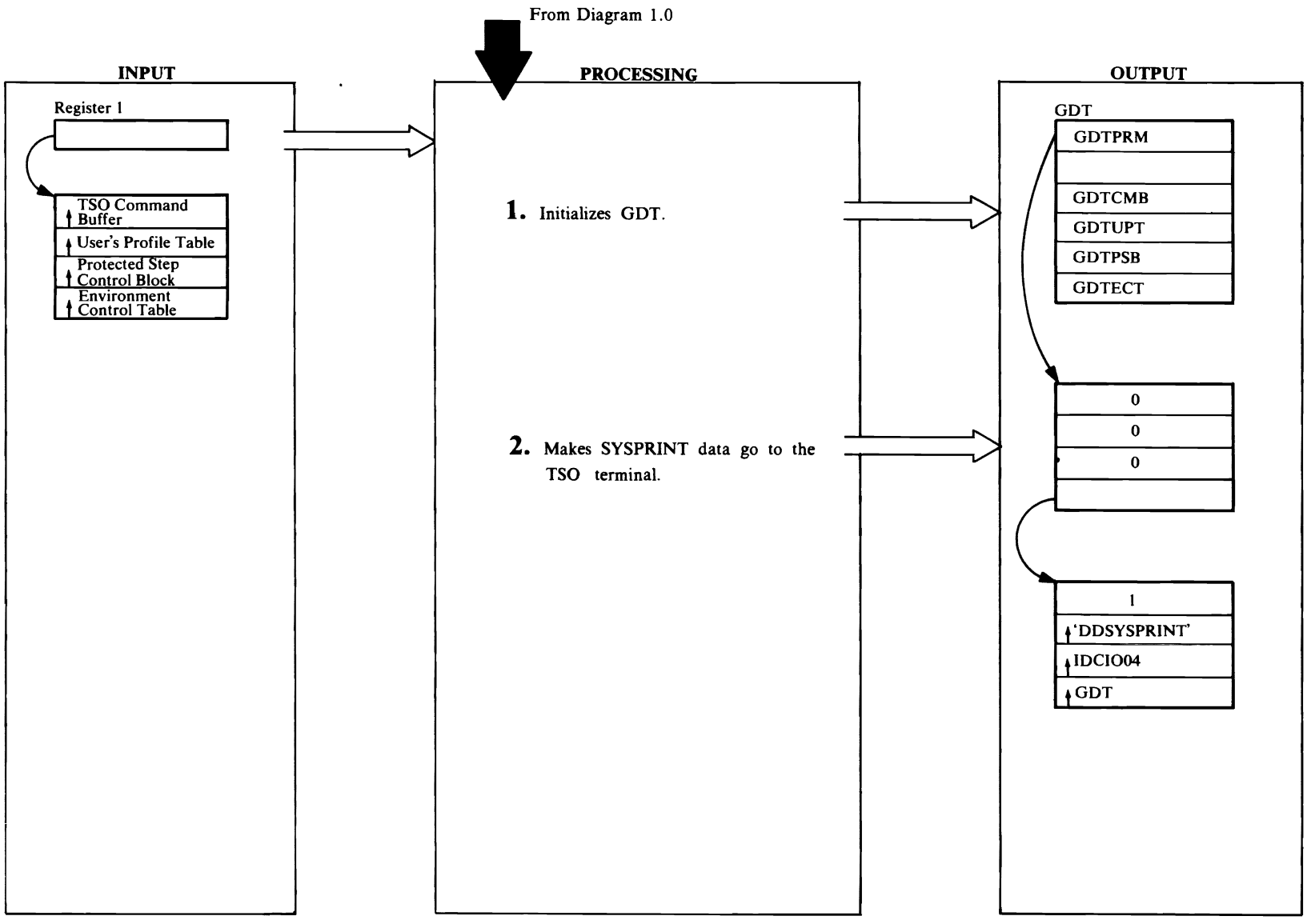
5. IDCSA01 sets the first UGPOOL storage area pointer in the System Adapter Historical Data Area to zero. It sets the last UGPOOL storage area pointer to the address of the first UGPOOL area pointer.

Module: IDCSA01

Procedure: IDCSA01

6. The System Adapter saves the current values of its registers in a save area pointed to by the GDTABH field in the GDT. The UABORT routine uses the register values to establish addressability before processing. Control goes to Diagram 1.0, step 2.

Diagram 1.2 Interactive TSO Initialization



Extended Description for Diagram 1.2

Module: IDCSA01

Procedure: IDCSA01

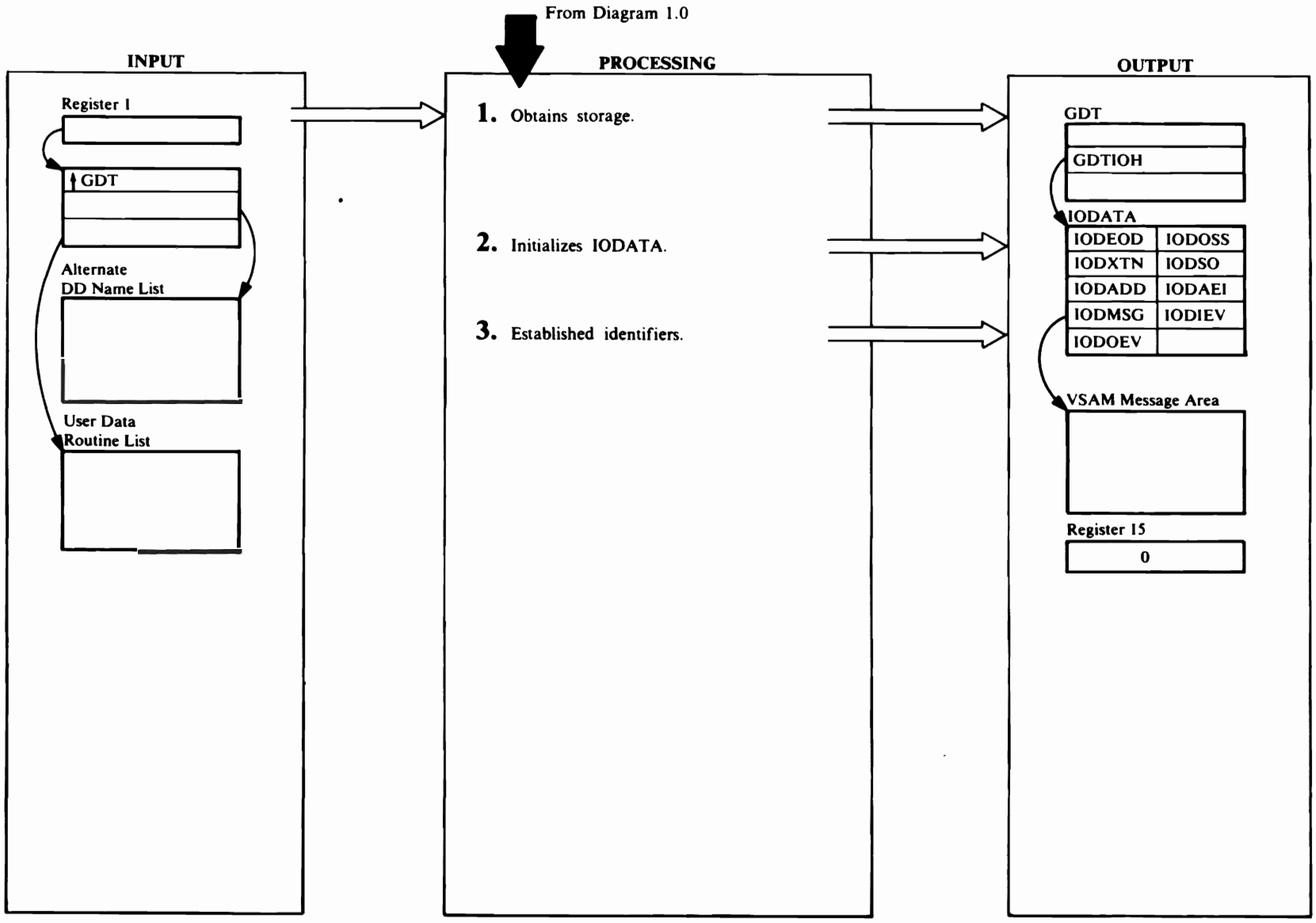
1. IDCSA01 puts the address of the TSO Command Buffer in GDTCMB, the address of the TSO Environment Control Table in GDTECT, the address of the TSO Protected Step Control Block in GDTPSB, and the address of the TSO User's Profile Table in GDTUPT.

Module: IDCSA01

Procedure: IDCSA01

2. In order to force all output written to the SYSPRINT data set to be displayed on the TSO terminal, IDCSA01 sets up IDCIO04 as a user I/O routine. IDCSA01 issues a LOAD macro to load IDCIO04. IDCSA01 puts the address of a parameter list in GDTPRM. The parameter list is provided by the System Adapter and causes all output to SYSPRINT to be given to IDCIO04. Refer to Diagram 6.5.1 for more information on IDCIO04. IDCIO04 writes to the TSO terminal. Control returns to Diagram 1.0, step 3.

Diagram 1.3 I/O Adapter Initialization – UIOINIT Macro



Extended Description for Diagram 1.3

Module: IDCIO01

Procedure: IDCIOIT

1. The I/O Adapter issues a UGPOOL to obtain storage for its Historical Data Area—IODATA, and a VSAM message area. IDCIOIT puts the IODATA address in the GDTIOH field in the GDT. The VSAM message area immediately follows IODATA. VSAM uses the message area to format any error messages it gives to the I/O adapter. If storage is not obtained from either UGPOOL, the I/O Adapter issues a UABORT to terminate the processor.

Module: IDCIO01

Procedure: IDCIOIT

2. The I/O Adapter initializes IODATA. IDCIOIT puts the address of the VSAM message area in the IODMSG field of IODATA. If the Access Method Services invoker supplied DD names for the system data sets—such as SYSIN and SYSPRINT—IDCIOIT puts the address of those DD names in the IODADD field of IODATA. If the invoker supplied a list of his own I/O programs to the address, IDCIOIT puts that address in IOBXTN. IDCIOIT puts the address of the Access Method Services End-of-Data routine in IODEOD. It puts the address for a synad routine for non-VSAM input data sets in IODOSS and the address for a synad routine for non-VSAM output data sets in IODSO. It also puts the address of the End-of-Data routine for VSAM data sets in IODAEI.

Module: IDCIO01

Procedure: IDCIOIT

3. IDCIOIT initializes the IODSID to the characters 'IO00'. The I/O Adapter uses this identifier to keep track of data sets. The first data set the I/O Adapter opens gets the identification 'IO01'. The second data set gets the identification 'IO02', and so on. The identification appears at the beginning of the storage area for each data set. IDCIOIT puts a return code of zero in register 15 and gives control to Diagram 1.0, step 4.

Reader/Interpreter Visual Table of Contents

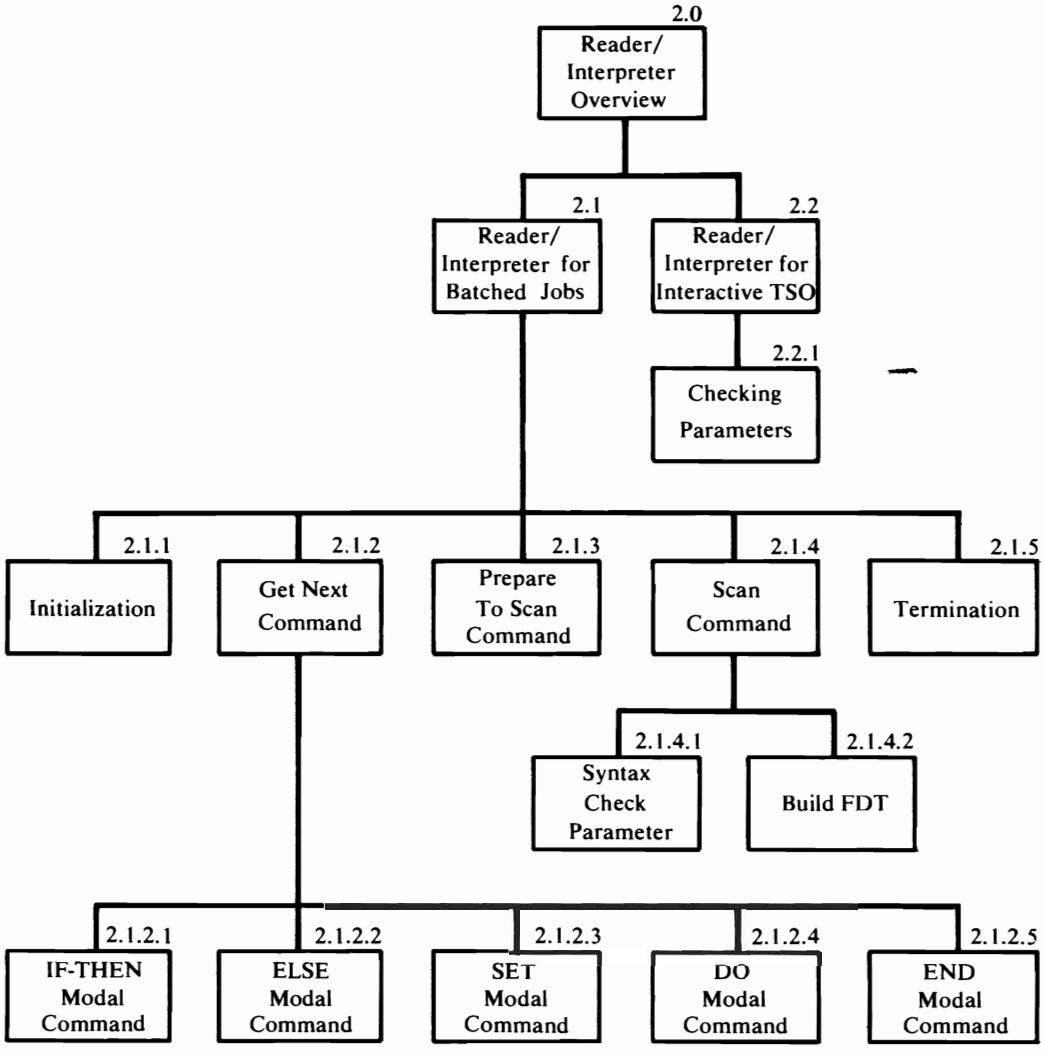
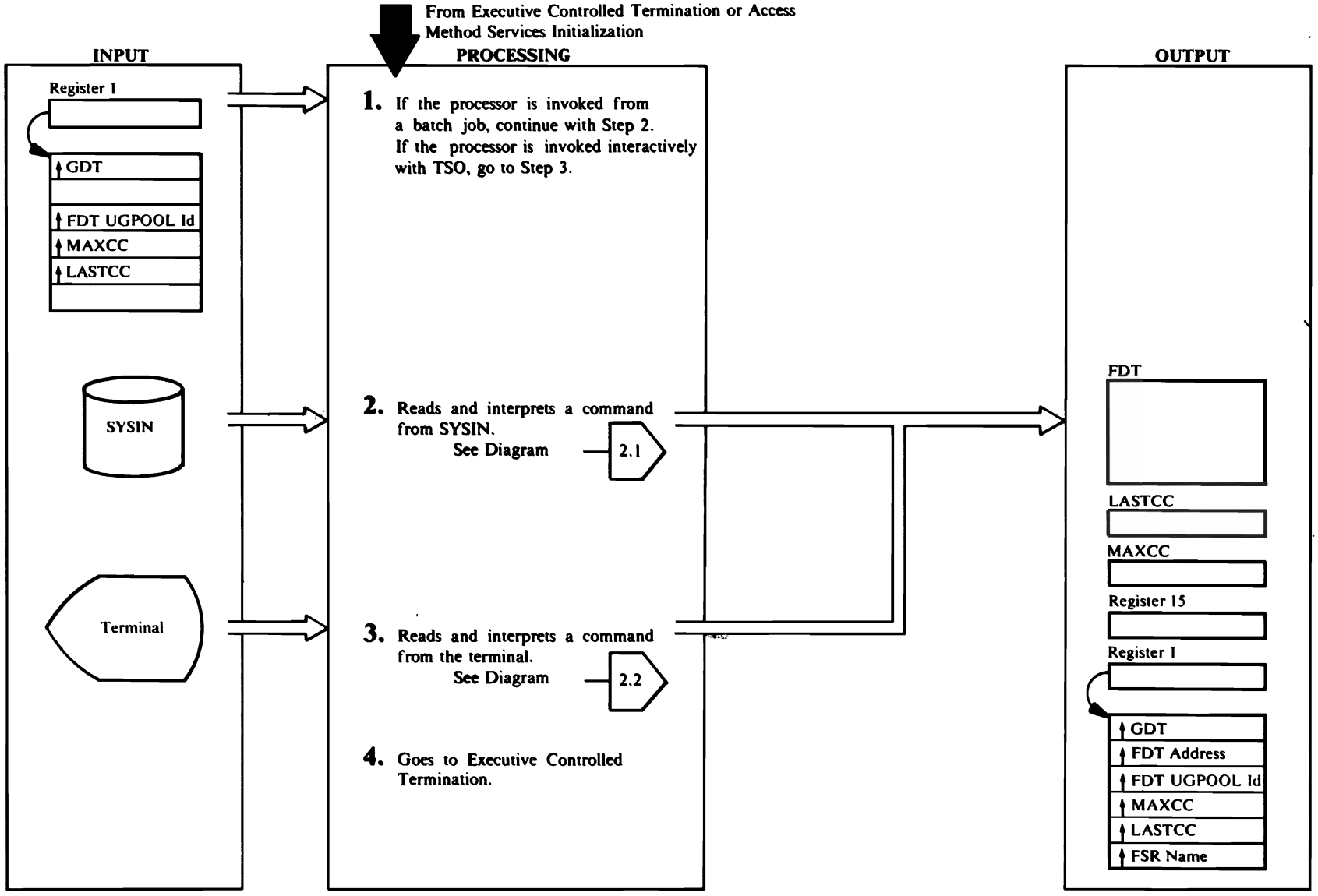


Diagram 2.0 Reader/Interpreter Overview



Extended Description for Diagram 2.0

Module: IDCRI01, IDCRI02, IDCRI03, IDCRI04

Procedure: IDCRI01, IDCRI02, IDCRI03, IDCRI04

1. Access Method Services accomplishes the task of reading and interpreting commands with two Reader/Interpreters. The invocation of Access Method Services determines which Reader/Interpreter is used. If Access Method Services is invoked with a batched job, processing continues with step 2. If Access Method Services is invoked interactively with TSO, processing continues with step 3.

Module: IDCRI01, IDCRI02, IDCRI03

Procedure: IDCRI01, IDCRI02, IDCRI03

2. The Reader/Interpreter for batched jobs gets a command from SYSIN or the PARM field of the EXEC JCL statement and builds a Function Data Table, FDT. Diagram 2.1 shows the Reader/Interpreter for batched jobs in detail. Control goes to step 4.

Module: IDCRI04

Procedure: IDCRI04

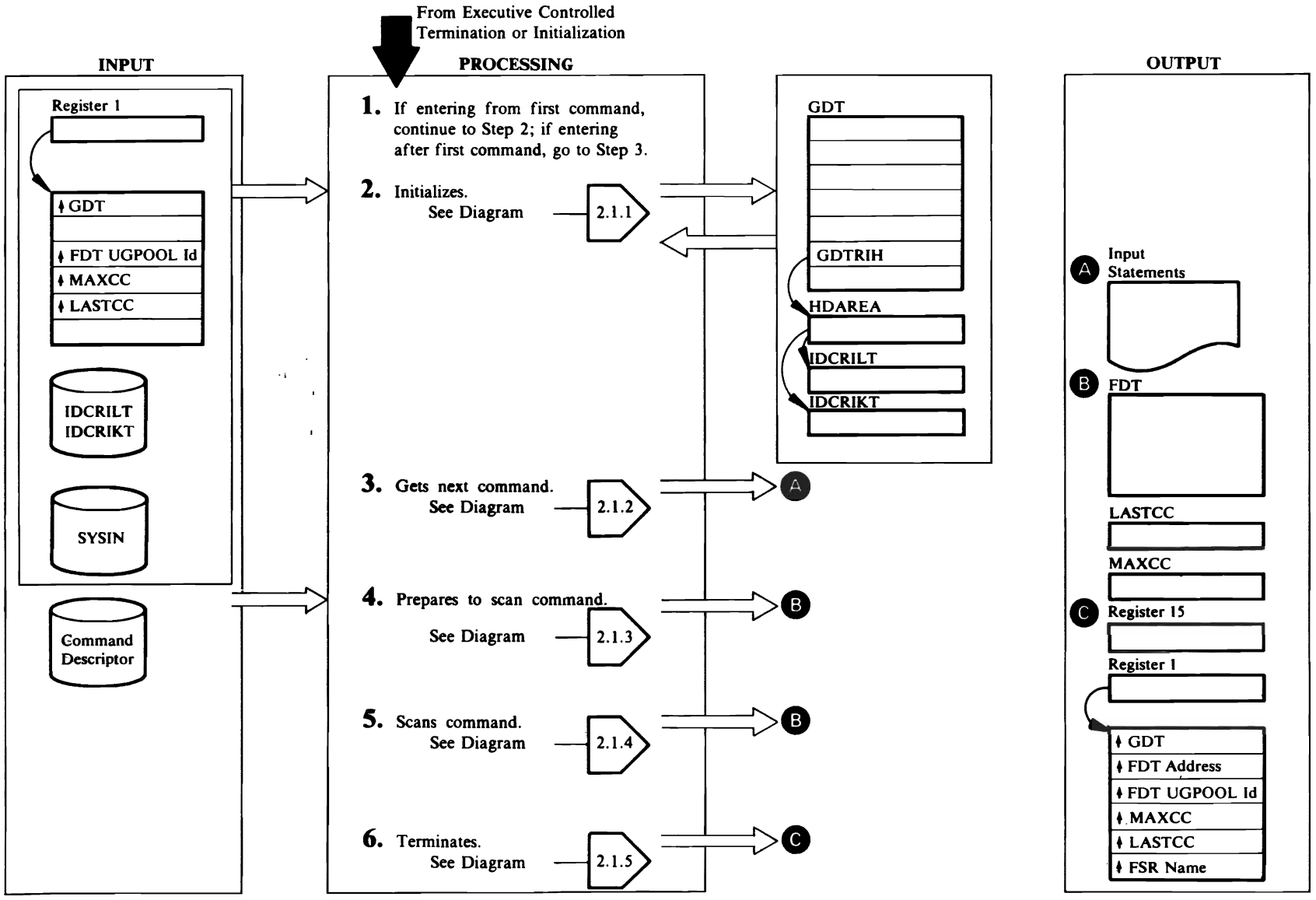
3. The Reader/Interpreter for interactive TSO gets the command from the terminal and builds a FDT. Diagram 2.2 shows the Reader/Interpreter for interactive TSO in detail.

Module: IDCRI01, IDCRI04

Procedure: IDCRI01, IDCRI04

4. When either Reader/Interpreter finishes building the FDT for a command, the Reader/Interpreter gives control to Executive Controlled Termination, Diagram 4.0. The Executive routes the FDT to the proper Function Support Routine, FSR, to enact the command.

Diagram 2.1 Reader/Interpreter for Batched Jobs Overview



Extended Description for Diagram 2.1

Module: IDCRI01

Procedure: RIINIT

1. This Reader/Interpreter receives control if Access Method Services is invoked with a batched job. Access Method Services Initialization, Diagram 1.0, gives control to the Reader/Interpreter, R/I, after the processor is initialized. If this is the first time the R/I receives control, processing continues with step 2. After the first command is processed, Executive Controlled Termination, Diagram 4.0, gives control to the R/I. If this is not the first time the R/I receives control, processing continues with step 3.
2. RIINIT initializes the Reader/Interpreter Historical Data Area, HDAREA. RIINIT loads the command name table, IDCRILT, and the modal command name table, IDCRIKT. RIINIT opens the input data set, SYSIN, and RIINIT prepares the parameters from the EXEC statement for scanning if they exist. Diagram 2.1.1 shows the initialization procedure in detail.

Module: IDCRI01

Procedure: GETNEXT, MODALSET, MODALIF, MODLELSE

3. GETNEXT reads and processes modal commands until a functional command is encountered. The execution of the functional command depends on the results from the modal commands. However, every command is completely checked for syntax errors, whether or not it is executed. Diagram 2.1.2 shows obtaining a command in detail.

Module: IDCRI02

Procedure: IDCRI02

4. IDCRI02 loads the command descriptor for the functional command to be scanned. IDCRI02 initializes the Function Data Table, FDT. Diagram 2.1.3 shows the preparation for command scanning in detail.

Module: IDCRI01

Procedure: SCANCMD, KWDPARM, POSPARM, INREPEAT, BUILDFDT, CONVERT, GETSPACE, DSIDCHK

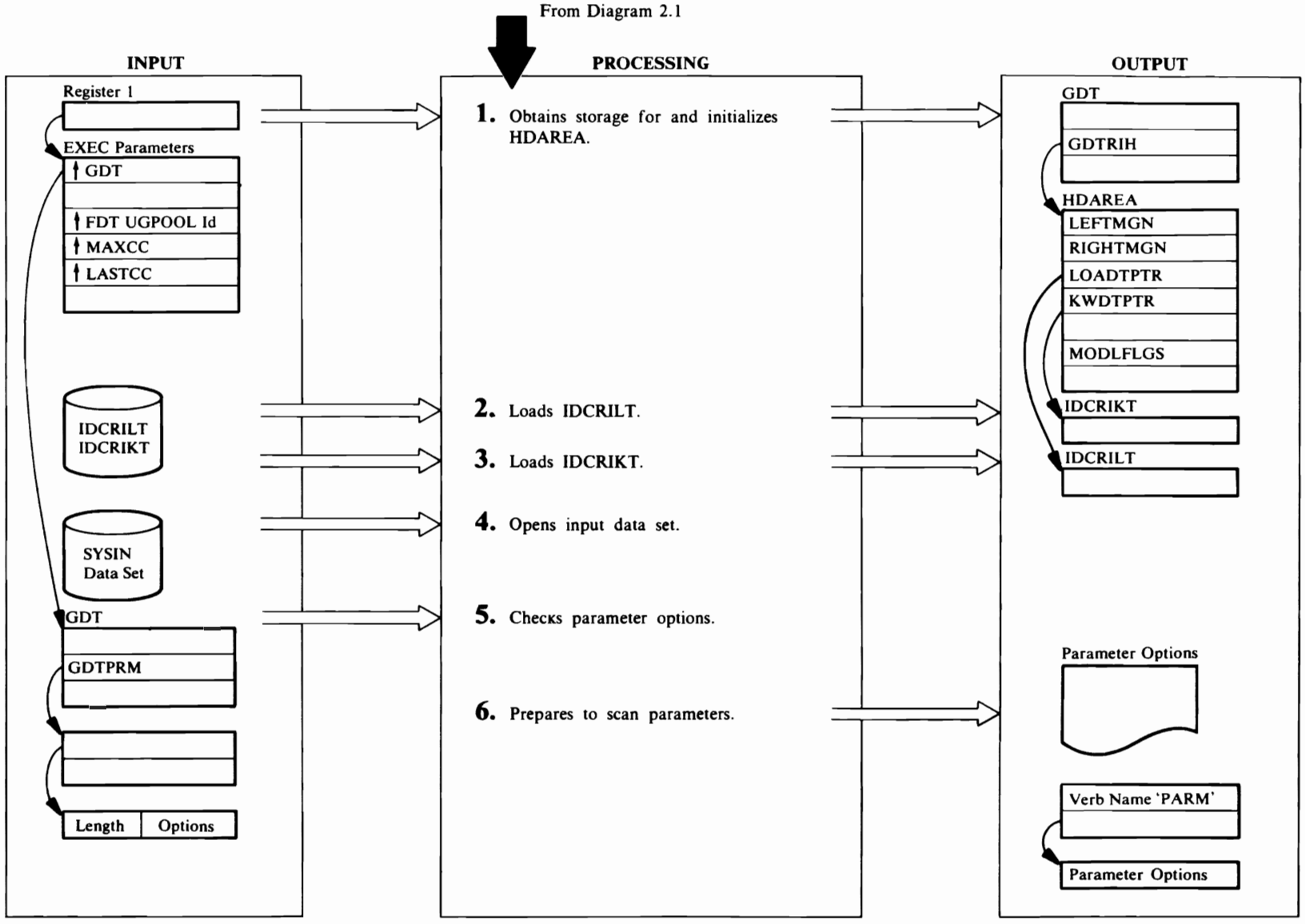
5. SCANCMD and BUILDFDT check the functional command for correctness. If the command is incorrect, ERROR1 or ERROR2 writes an error message. BUILDFDT and INREPEAT complete the FDT for correct commands. Diagram 2.1.4 shows the command scanning in detail.

Module: IDCRI03

Procedure: IDCRI03

6. IDCRI03 deletes the work tables and temporary storage. If the command is to be executed, control is given to Executive Controlled Termination that gives control to the Function Support Routine, FSR, that executes the command. If the command is not to be executed due to syntax errors or due to the results of a modal expression, control returns to step 3 to get the next command. If the error is severe, control returns to Executive Controlled Termination, Diagram 4.0, with an indication that the processor cannot continue. Diagram 2.1.5 shows termination processing in detail.

Diagram 2.1.1 Reader/Interpreter Initialization for Batched Jobs Initialization



Extended Description for Diagram 2.1.1

Module: IDCRI01

Procedure: RIINIT

1. RIINIT obtains storage for HDAREA and sets the left margin field to 2 and the right margin field to 72. A user changes the margins using a PARM command. RIINIT initializes the rest of HDAREA to zero. If RIINIT cannot obtain storage, control is given to Reader/Interpreter termination. Diagram 2.1.5 with an indication that causes the processor to end.

Module: IDCRI01

Procedure: RIINIT

2. RIINIT loads the command name table, IDCRILT, and places the address of IDCRILT in the LOADTPTR field in HDAREA. IDCRILT contains the name of each verb and corresponding command descriptor.

Module: IDCRI01

Procedure: RIINIT

3. RIINIT loads the modal name table, IDCRIKT, and places the address of IDCRIKT in the KWDTPTR field in HDAREA. IDCRIKT contains modal command keyword and verb name symbols, plus the length of each symbol.

Module: IDCRI01

Procedure: RIINIT

4. RIINIT opens the input data set which has a default DD name of SYSIN. If SYSIN cannot be opened, control is given to the Reader/Interpreter termination, Diagram 2.1.5, with an indication that causes the processor to end.

Module: IDCRI01

Procedure: RIINIT

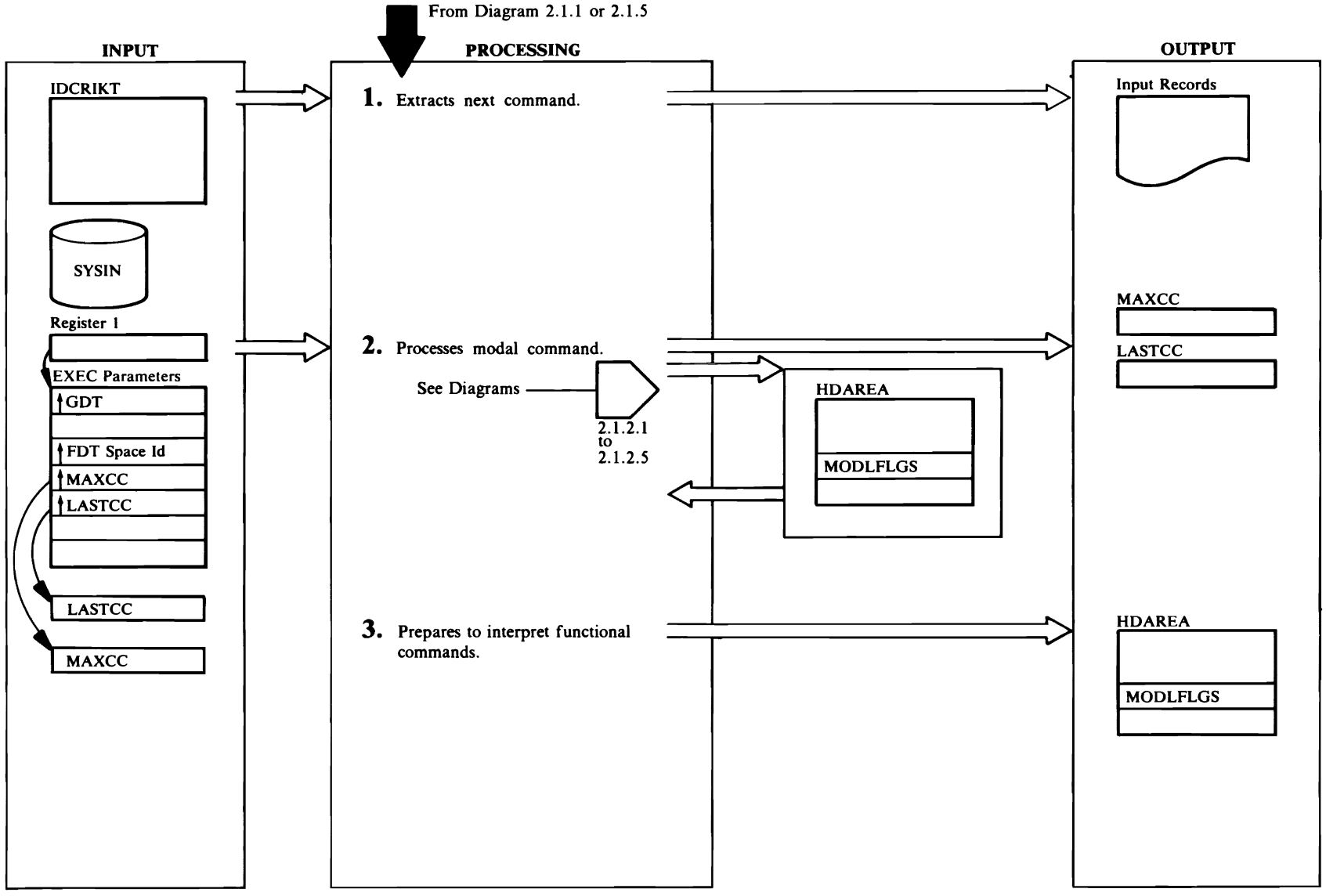
5. The Reader/Interpreter, R/I, checks for parameters supplied before SYSIN is read. The invoker may supply parameters by putting them in the EXEC JCL statement. Parameters may also be supplied through the data the user provides to the processor at the time the user's program invokes Access Method Services. If parameters are supplied, the GDTPRM field of the GDT contains the address of a fullword that contains the address of the parameters. The first two bytes of the parameters is the total length of the parameters. If no parameters are supplied, the length field is zero.

Module: IDCRI01

Procedure: RIINIT

6. The parameters are printed on SYSPRINT and are treated as the parameters for a PARM command. The symbol for PARM in IDCRIKT is supplied as the verb name, and the options are scanned by the R/I as though a PARM command had been encountered in SYSIN. After the pseudo PARM command is executed by the PARM FSR, Executive Controlled Termination gives R/I control to read the first command. Control goes to Diagram 2.1.2 to get the first command.

Diagram 2.1.2 Reader/Interpreter for Batched Jobs Get Next Command



Extended Description for Diagram 2.1.2

Module: IDCRI01

Procedure: GETNEXT, NXTFIELD, NEXTCHAR

1. GETRECRD reads SYSIN to get an input record and writes each input record on SYSPRINT. GETNEXT locates the verb on the input record and checks it against the symbols for the modal verbs IF, ELSE, SET, DO, and END in IDCRIKT. If a match is found, the verb is a correct modal verb, and processing continues to step 2. If a match is not found, the verb is assumed to be a functional verb, and processing goes to step 3.

Module: IDCRI01

Procedure: GETNEXT, MODALIF, MODLELSE, MODALSET

2. GETNEXT sets condition codes and the MODLFLGS field in HDAREA depending on the modal command. Control returns to step 1 to get the next command. The modal commands are shown in detail in the following diagrams:

   IF-THEN, Diagram 2.1.2.1
   ELSE, Diagram 2.1.2.2
   SET, Diagram 2.1.2.3
   DO, Diagram 2.1.2.4
   END, Diagram 2.1.2.5

Module: IDCRI01

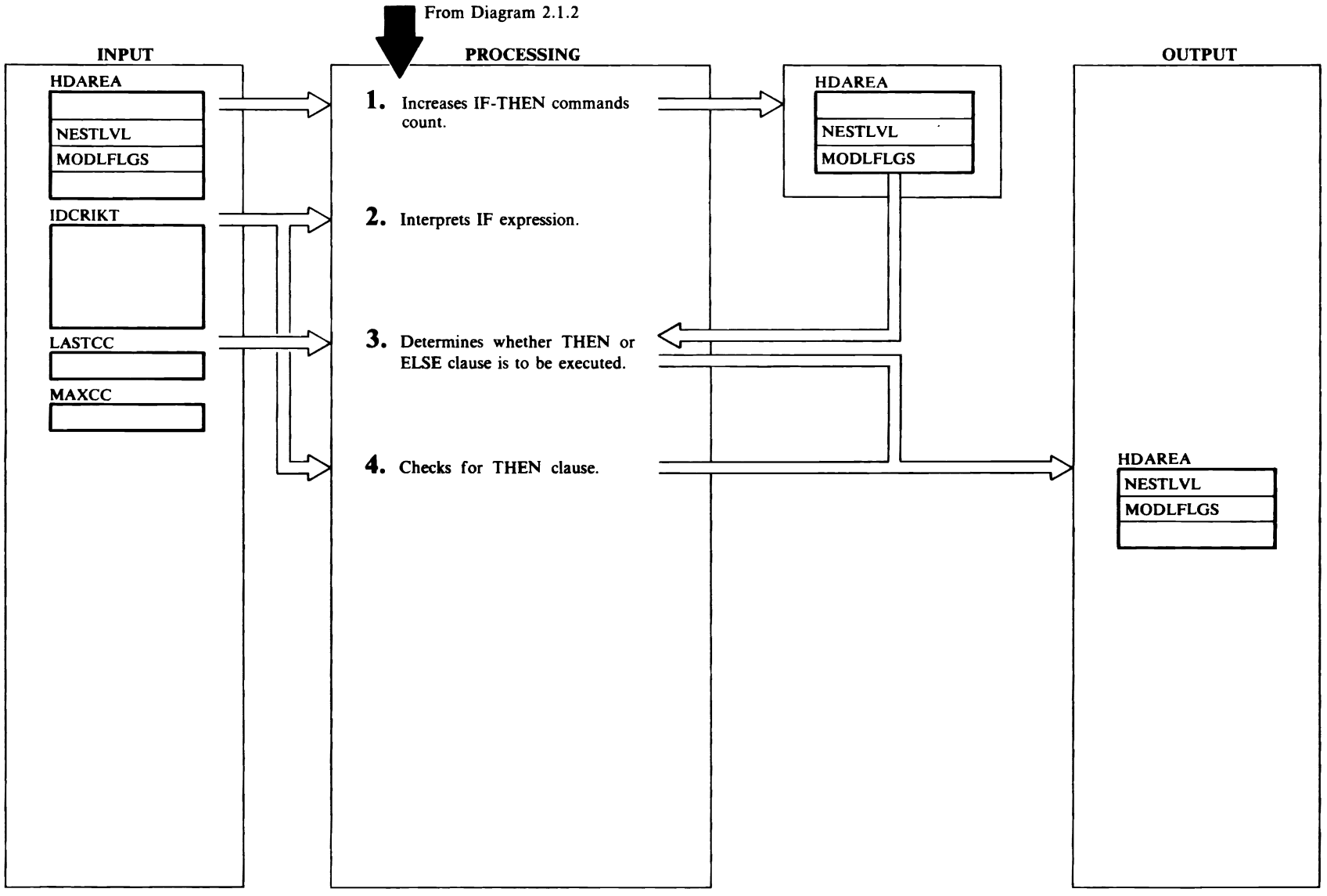
Procedure: GETNEXT

3. GETNEXT checks the MODLFLGS field in HDAREA to determine if the function command should be executed. If the functional command is not to be executed, GETNEXT sets a flag. Every command is completely checked for syntax errors, whether or not it is to be executed. If the functional command finishes an IF - THEN command, GETNEXT subtracts 1 from the number of nested IF - THEN commands and sets MODLFLGS for the finished IF - THEN command to zero. The functional commands are shown in detail in the following diagrams:

   ALTER, Diagram 3.1
   CNVTCAT, Diagram 3.2
   DEFINE, Diagram 3.3
   DELETE, Diagram 3.4
   EXPORT, Diagram 3.5
   IMPORT, Diagram 3.6
   LISTCAT, Diagram 3.7
   PARM, Diagram 3.8
   PRINT, Diagram 3.9
   REPRO, Diagram 3.10
   VERIFY, Diagram 3.11
   CHKLIST, Diagram 3.12
   BLDINDEX, Diagram 3.13
   LISTCRA, Diagram 3.14
   EXPORTRA, Diagram 3.15
   IMPORTRA, Diagram 3.16
   RESETCAT, Diagram 3.17 (**VS2.03.808**)

   Control goes to Diagram 2.1.3 to scan the command.

Diagram 2.1.2.1 IF-THEN Modal Command



Extended Description for Diagram 2.1.2.1

Module: IDCRI01

Procedure: MODALIF

1. The value in the NESTLVL field of HDAREA is used as an index to the MODLFLGS field for the current IF - THEN command and the THEN and ELSE clauses that belong to the IF - THEN. MODALIF adds 1 to the number of nested IF commands in NESTLVL. There is one set of modal flags in HDAREA for each level of IF - THEN commands. The new level of MODLFLGS is initialized to zero. To see if too many IF - THEN commands are nested, MODALIF compares the number of nested IF - THEN commands to the number permitted, 10.

   When a syntax error is detected, MODALIF sets LASTCC to 16, and control is given to Reader/Interpreter termination, Diagram 2.1.5, to cause the Executive to terminate the processor.

Module: IDCRI01

Procedure: MODALIF, PACKCVB, NXTFIELD, NEXTCHAR

2. MODALIF compares the characters following the IF with the symbols for LASTCC and MAXCC in IDCRIKT. MODALIF compares the operator with all possible operators (LT, GT, EQ, NE, GE, LE, =, ¬=, >, <, >=, <=). PACKCVB converts the decimal value following the operator to binary. If any errors are detected, the syntax error procedure in step 1 is followed.

Module: IDCRI01

Procedure: MODALIF

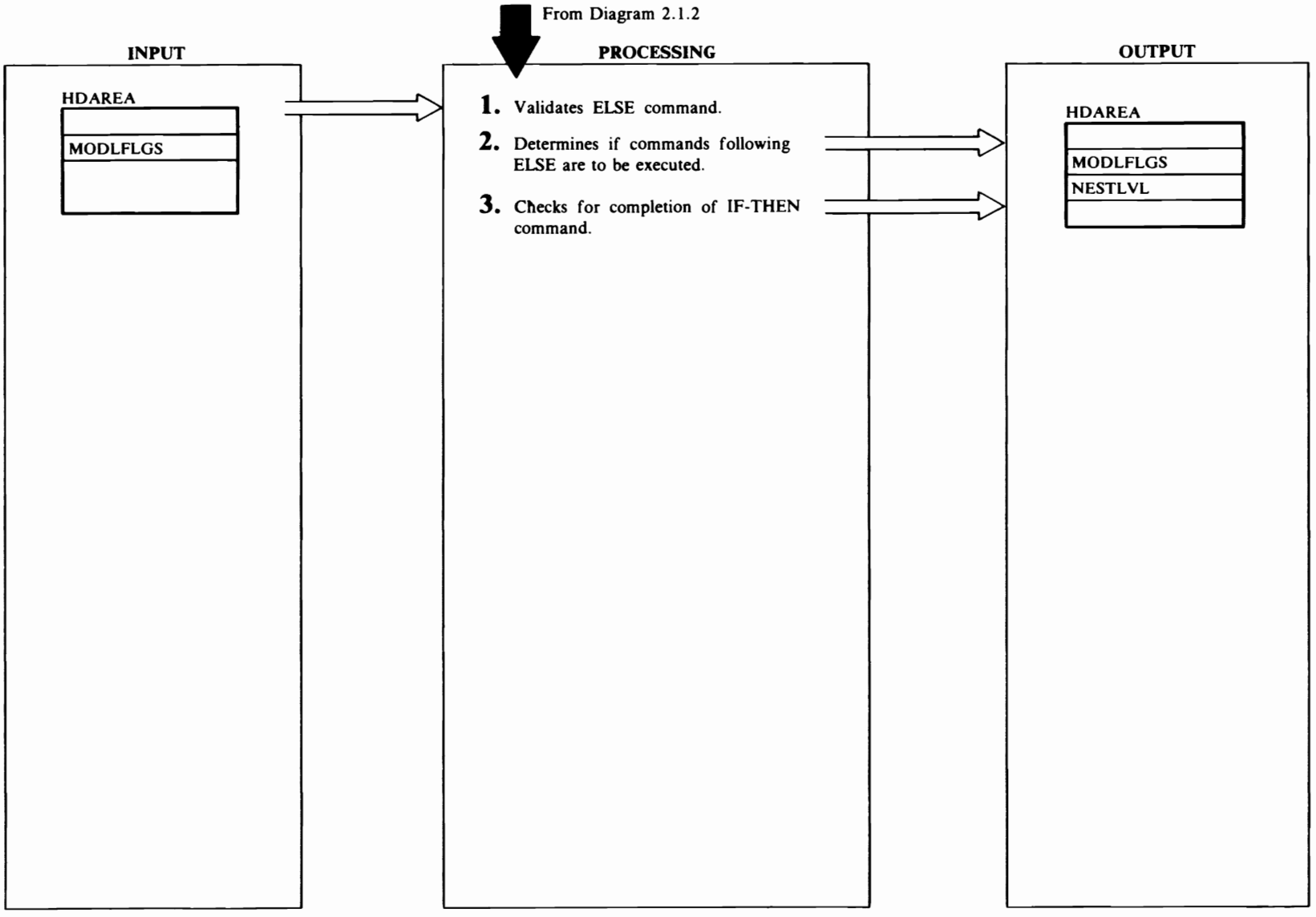
3. MODALIF sets the THENFLAG to 1 to indicate that the THEN clause of the IF - THEN command is being processed. MODALIF compares the value of LASTCC or MAXCC with the number in the IF - THEN command and evaluates it for true or false depending upon the operator. If the result is false, MODALIF sets the SKIPFLAG in HDAREA to 1, indicating that commands in the THEN clause of the IF - THEN command are to be skipped—that is, the Reader/Interpreter is to check only the syntax of the commands in the THEN clause.

Module: IDCRI01

Procedure: MODALIF

4. MODALIF compares the characters following the relational expression with the symbol for THEN in IDCRIKT. An error occurs if THEN does not follow IF, and the syntax error procedure in step 1 is followed. If a terminator follows the THEN keyword, there is a null THEN clause in the current IF - THEN command. Control returns to Diagram 2.1.2 to obtain the next command.

Diagram 2.1.2.2 ELSE Modal Command



Extended Description for Diagram 2.1.2.2

Module: IDCRI01

Procedure: MODLELSE

1. MODLELSE sets the ELSEFLAG in HDAREA for the current IF - THEN command to 1, indicating that the ELSE clause of the IF - THEN command is being processed. The THENFLAG is turned off. An error is caused by an ELSE without a prior IF - THEN, and the syntax error procedure in Diagram 2.1.2.1, step 1 is followed.

Module: IDCRI01

Procedure: MODLELSE

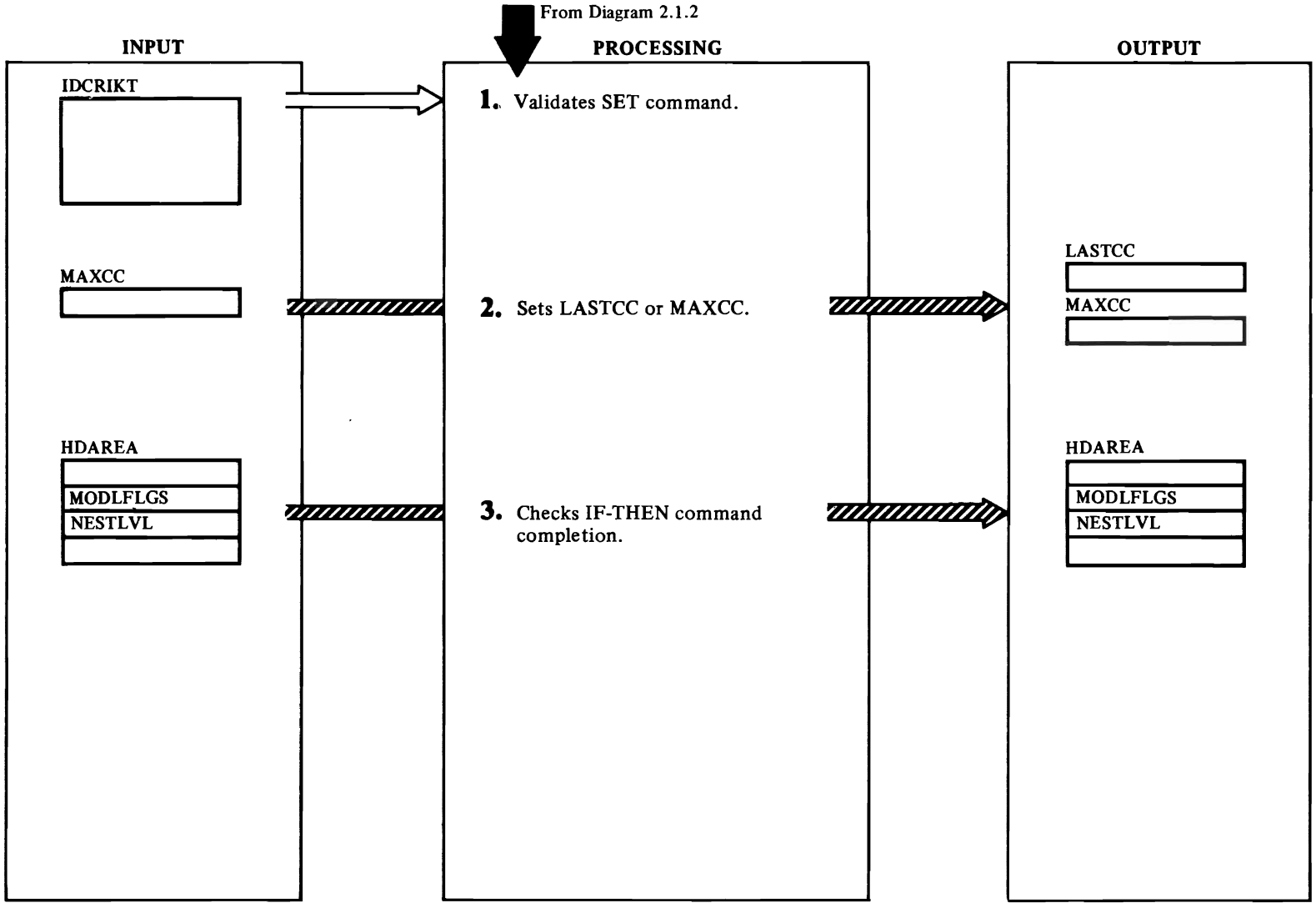
2. SKIPFLAG indicates whether the commands in the ELSE clause of the IF - THEN command should be executed or only checked for syntax errors. If SKIPFLAG is zero, the THEN clause of the IF - THEN command was executed, the ELSE clause should not be executed, and MODLELSE sets SKIPFLAG to 1. If SKIPFLAG is 1, the THEN clause of the IF - THEN command was not executed, the ELSE clause should be executed, and MODLELSE sets SKIPFLAG to zero. However, if the entire IF - THEN - ELSE command is nested within another THEN or ELSE clause that is not being executed, neither the THEN clause nor the ELSE clause of the nested IF - THEN - ELSE command is executed.

Module: IDCRI01

Procedure: MODLELSE, NXTFIELD, NEXTCHAR

3. If a terminator immediately follows ELSE, there are no commands in the ELSE clause of the current IF - THEN command. MODLELSE subtracts 1 from NESTLVL since the IF command is completed. Control is given to Diagram 2.1.2 to obtain the next command, whether or not a terminator follows the ELSE.

Diagram 2.1.2.3 SET Modal Command



Extended Description for Diagram 2.1.2.3

Module: IDCRI01

Procedure: MODALSET, PACKCVB, NXTFIELD, NEXTCHAR

1. MODALSET compares the characters following SET with the symbols for LASTCC and MAXCC in IDCRIKT. MODALSET compares the operator with the symbols EQ and =. PACKCVB converts the decimal value following the operator to binary. If a syntax error is encountered, the processing in Diagram 2.1.2.1, step 1 is followed.

Module: IDCRI01

Procedure: MODALSET

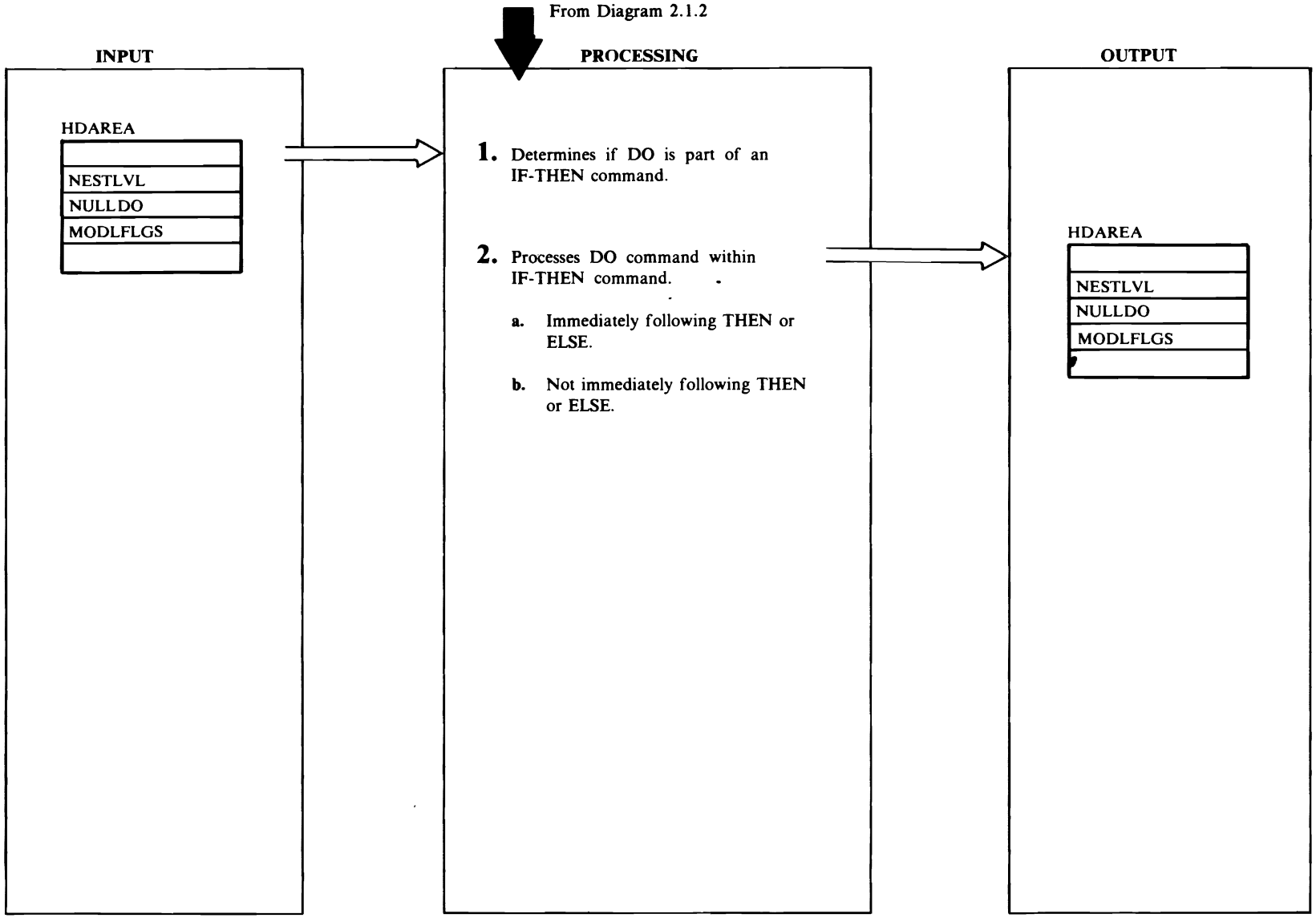
2. MODALSET obtains MAXCC or LASTCC and changes its value to the value specified in the SET command. If the command is SET LASTCC, MODALSET compares MAXCC and LASTCC, and the larger value is put into MAXCC. If the SET command is only being checked for syntax errors, neither MAXCC nor LASTCC is changed.

Module: IDCRI01

Procedure: MODALSET

3. MODALSET determines that the current IF command is finished by checking that the SET command follows an ELSE keyword and that the SET command is not within a DO group. If both of these conditions are met, MODALSET subtracts 1 from NESTLVL in HDAREA, and returns control to Diagram 2.1.2 to obtain the next command.

Diagram 2.1.2.4 DO Modal Command



Extended Description for Diagram 2.1.2.4

Module: IDCRI01

Procedure: GETNEXT, NXTFIELD, NEXTCHAR

1. If a DO command is not part of an IF - THEN command, control returns to Diagram 2.1.2 to obtain the next command. If a DO command is part of an IF - THEN command, processing continues to step 2.

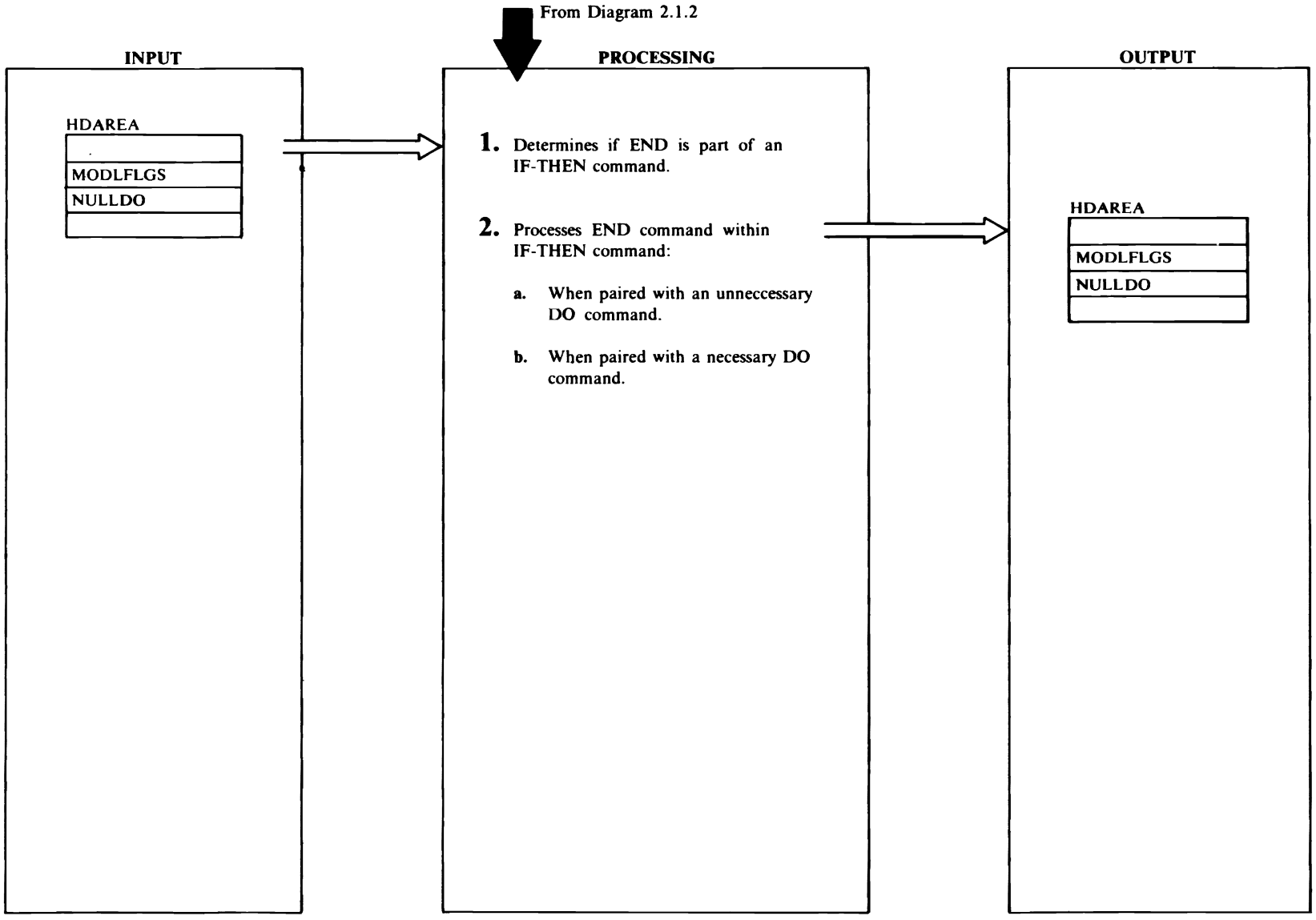
Module: IDCRI01

Procedure: MODALIF, MODELSE, NXTFIELD, NEXTCHAR, GETNEXT

2. a. If a DO command is part of an IF - THEN command and immediately follows a THEN or ELSE keyword, MODALIF or MODLELSE sets DOFLAG to 1. Control returns to Diagram 2.1.2 for the first command of the DO group.

   b. If a DO command is part of an IF - THEN command, but it does not immediately follow a THEN or ELSE keyword, the DO command is unnecessary. GETNEXT increases the NULLDO field in HDAREA by 1, and control returns to Diagram 2.1.2 for the first command of the unnecessary DO group.

Diagram 2.1.2.5 END Modal Command



Extended Description for Diagram 2.1.2.5

Module: IDCRI01

Procedure: GETNEXT

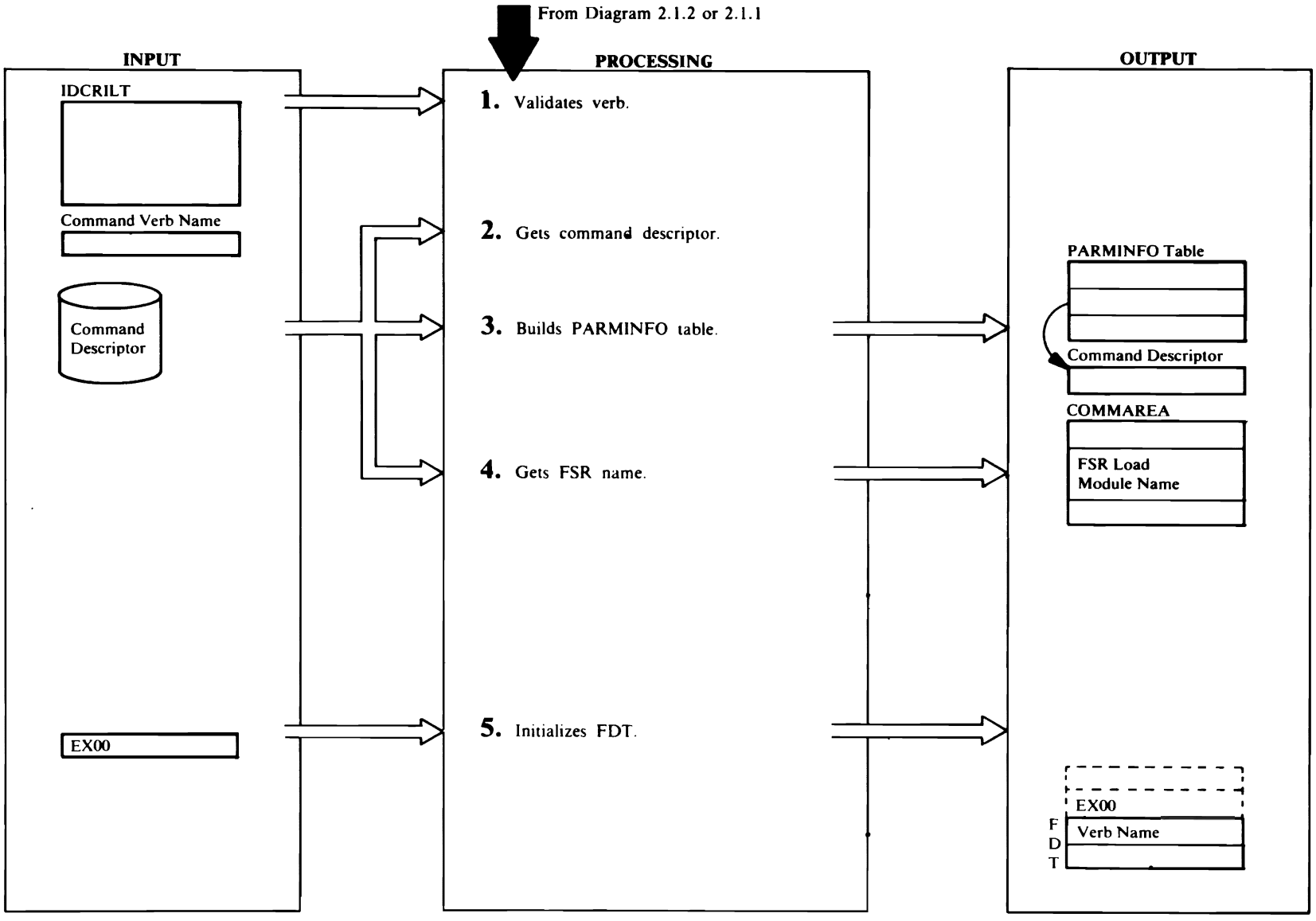
1. GETNEXT checks the NESTLVL field in HDAREA; if NESTLVL contains a zero, no IF – THEN command is being processed, and control returns to Diagram 2.1.2 to obtain the next command. If NESTLVL contains a value other than zero, processing continues with step 2.

Module: IDCRI01

Procedure: GETNEXT

2. An END encountered during the processing of an IF – THEN command must be paired with a DO command. If a DO command has not been found in the current IF - THEN command, the END is processed as a syntax error as in Diagram 2.2.1, step 1.
   a. If the END command is paired with an unnecessary DO command, GETNEXT subtracts 1 from the count in the NULLDO field in HDAREA. Control returns to Diagram 2.2 to obtain the next command.
   b. If an END is paired with a necessary DO command, GETNEXT sets the DOFLAG for the current IF – THEN command to zero. An IF - THEN command is completed if the END is paired with a necessary DO that followed an ELSE. GETNEXT subtracts 1 from the count of nested IF – THEN commands in NESTLVL. Control returns to Diagram 2.1.2 to obtain the next command.

Diagram 2.1.3 Reader/Interpreter Prepare To Scan Command



Extended Description for Diagram 2.1.3

Module: IDCRI02, IDCRI01

Procedure: IDCRI02, ERROR2

1. Reader/Interpreter Initialization, Diagram 2.1.1, gives control to this section only if parameters were present before SYSIN was read. Otherwise, control comes from Diagram 2.1.2. IDCRI02 compares the verb name with the valid functional verb names in IDCRILT. If a match is found, IDCRI02 obtains the name of the verb's command descriptor from the table. If a match is not found, the verb is invalid, and ERROR2 prints a message on SYSPRINT. The remainder of the command is ignored, and control is given to Reader/Interpreter termination, Diagram 2.1.5.

Module: IDCRI02

Procedure: IDCRI02

2. IDCRI02 uses the command descriptor name to load the command descriptor. A command descriptor is a load module describing all the parameters the command may have. Access Method Services defines a parameter as:

   - Positional data—positional parameters cannot have subparameters.
   - Keyword with or without data—keyword parameters may have subparameters.

   Data is a constant or list of constants.

   Some examples of parameters are:

   - *entryname...* in DELETE is a positional parameter.
   - VOLUMES (111111) is one parameter with a keyword VOLUMES and data of '111111'.
   - VOLUMES (111111, 222222) is one parameter with keyword VOLUMES and data of '111111' and '222222'. (111111, 222222) is a list of constants. Each constant is the same thing—that is, a volume serial number.
   - KEYS (5, 40) is 3 parameters—KEYS, *length* with value 5, and *position* with value 40. KEYS is a keyword while *length* and *position* are each positional parameters. (*length*, *position*) is not a list of constants because the second item, *position*, is different from the first, *length*. *length* and *position* are subparameters of KEYS.
   - KEYRANGES ((5, 40), (50, 60), (70, 80)) is 3 parameters—KEYRANGES, *lowkey*, and *highkey*. The subparameters of KEYRANGES, *lowkey* and *highkey* are repeated. In Access Method Services, each repetition of a parameter must be enclosed in parentheses. Since *lowkey* and *highkey* are positional parameters, they must always be in the same relative position. They are repeated as a pair to maintain their position.

Module: IDCRI02, IDCRI01

Procedure: IDCRI02

3. The command descriptor contains an identification number for each parameter the command is permitted to have. Since the sections of the command descriptor that describe the parameters are in no set order, IDCRI02 builds the PARMINFO Table to access information in the order of the parameter identification number. The PARMINFO Table consists of several sections. In the Descriptor Pointer section, the first pointer in the array points to the Command Descriptor section that describes parameter with identification number 1. The second pointer points to the Command Descriptor section that describes parameter with identification number 2, and so on. The PARMFLAG section contains one entry for each parameter identification possible in the command. PARMFLAG is used to keep track of which parameters have been found. When a parameter is found, SETFLAG sets the indicator for the parameter in PARMFLAG.

   In Access Method Services, a subparameter is a parameter that modifies another parameter. For example, in DEFINE SPACE (VOL ...), VOL is a subparameter of SPACE. In this document, the parameter that the subparameter modifies is called its superparameter. In this example, SPACE is the superparameter of VOL. A superparameter, then, is a parameter that is modified by other parameters. For each subparameter, IDCRI02 puts the number of its superparameter in the PARMINFO Table in the Superparameter ID section that the Reader/Interpreter, R/I, uses to determine the relationship among parameters.

Module: IDCRI02

Procedure: IDCRI02

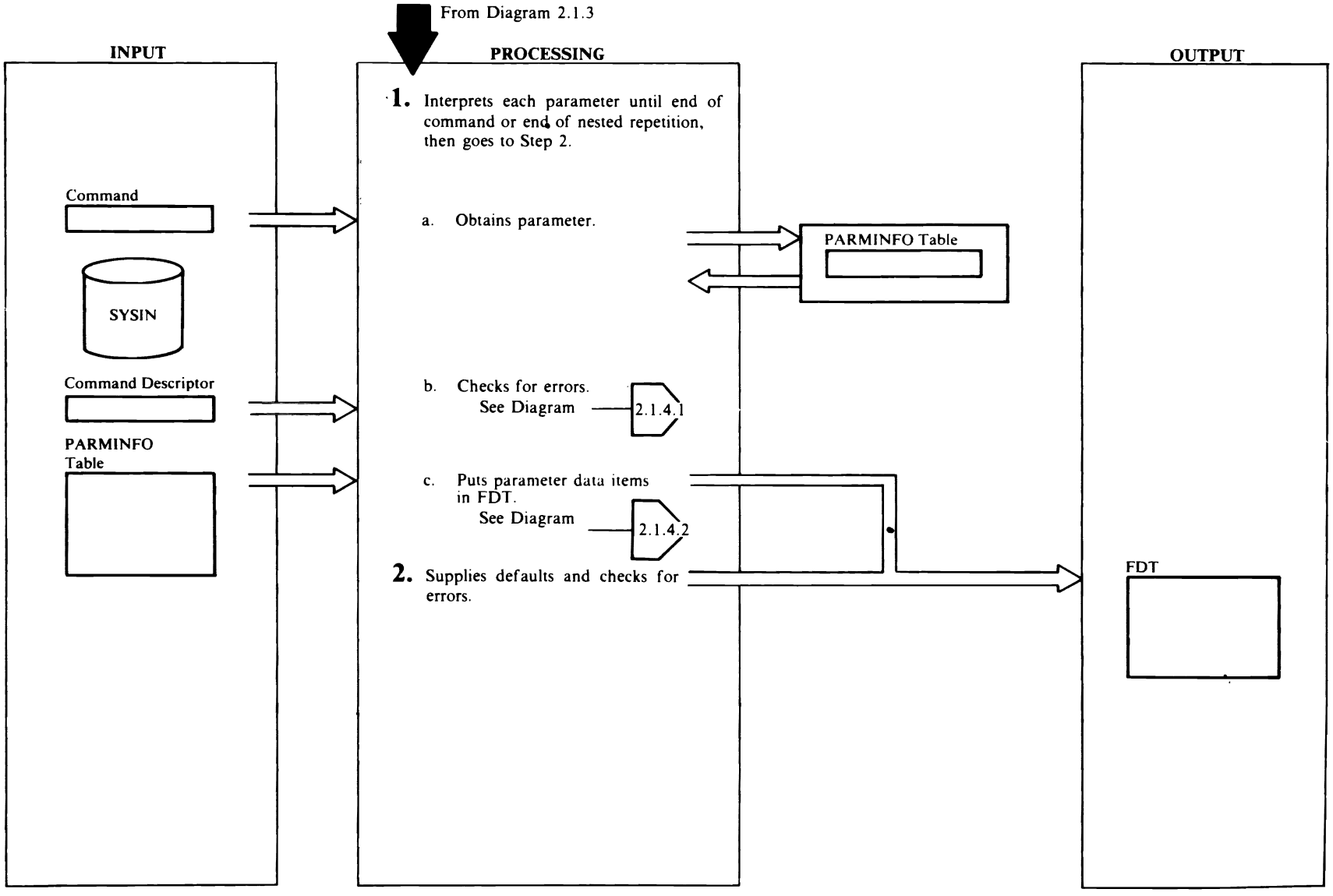
4. IDCRI02 obtains the FSR load module name from the command descriptor and places the name in the FSRLNAME field in COMMAREA. The Executive uses the FSR load module name to load the FSR that executes the command.

Module: IDCRI02

Procedure: IDCRI02

5. IDCRI02 obtains storage for the Function Data Table, FDT. The verb uses eight bytes of storage, and each parameter uses four additional bytes. IDCRI02 obtains more storage for the FDT if any parameter is repeated. The amount of storage for repeated parameters is calculated from the command descriptor. Because IDCRI02 uses a UGPOOL macro instruction to obtain storage, the identifier EX00 precedes the FDT. IDCRI02 initializes the FDT to zero and places the verb name in the first eight bytes. The FDT contains the information from the command that an FSR needs to execute the command. The FDT is the interface between the R/I and the FSRs and consists of a primary array of addresses, one secondary array of addresses for each repeated parameter, and encoded data from the command. Control goes to Diagram 2.1.4.

Diagram 2.1.4 Reader/Interpreter Scan Command



Extended Description for Diagram 2.1.4

Module: IDCRI01

Procedure: BUILDFDT, CONVERT, DSIDCHK, NAMESCAN, SCANCMD, KWDPARM, POSPARM, INREPEAT, GETDATA, GETSIMPL, GETQUOTD, ERROR1, ERROR2, NXTFIELD, NEXTCHAR, GETRECRD

1. If the Reader/Interpreter, R/I, is processing a specified parameter, processing continues with step 1.a. If the R/I is processing the end of a command or the end of a repeated parameter, processing continues with step 2. A parameter set is a parameter repeated as a group. Each repeated parameter set is treated separately from the command and from other repeated parameter sets. PARMFLAG for the parameters in a repetition are reset to zero for each group of repeated parameters in order to start the processing again for the new repeated group of parameters.

   a. SCANCMD extracts a parameter from the input record in storage. If the entire parameter is not in storage, GETRECRD reads SYSIN until all the parameter is in storage.

   b. SCANCMD checks the parameter for syntax errors based upon the information for the parameter in the command descriptor. If errors are found, ERROR1 or ERROR2 writes a message to SYSPRINT and sets LASTCC to 12. The rest of the command is skipped, and control is passed to R/I termination.

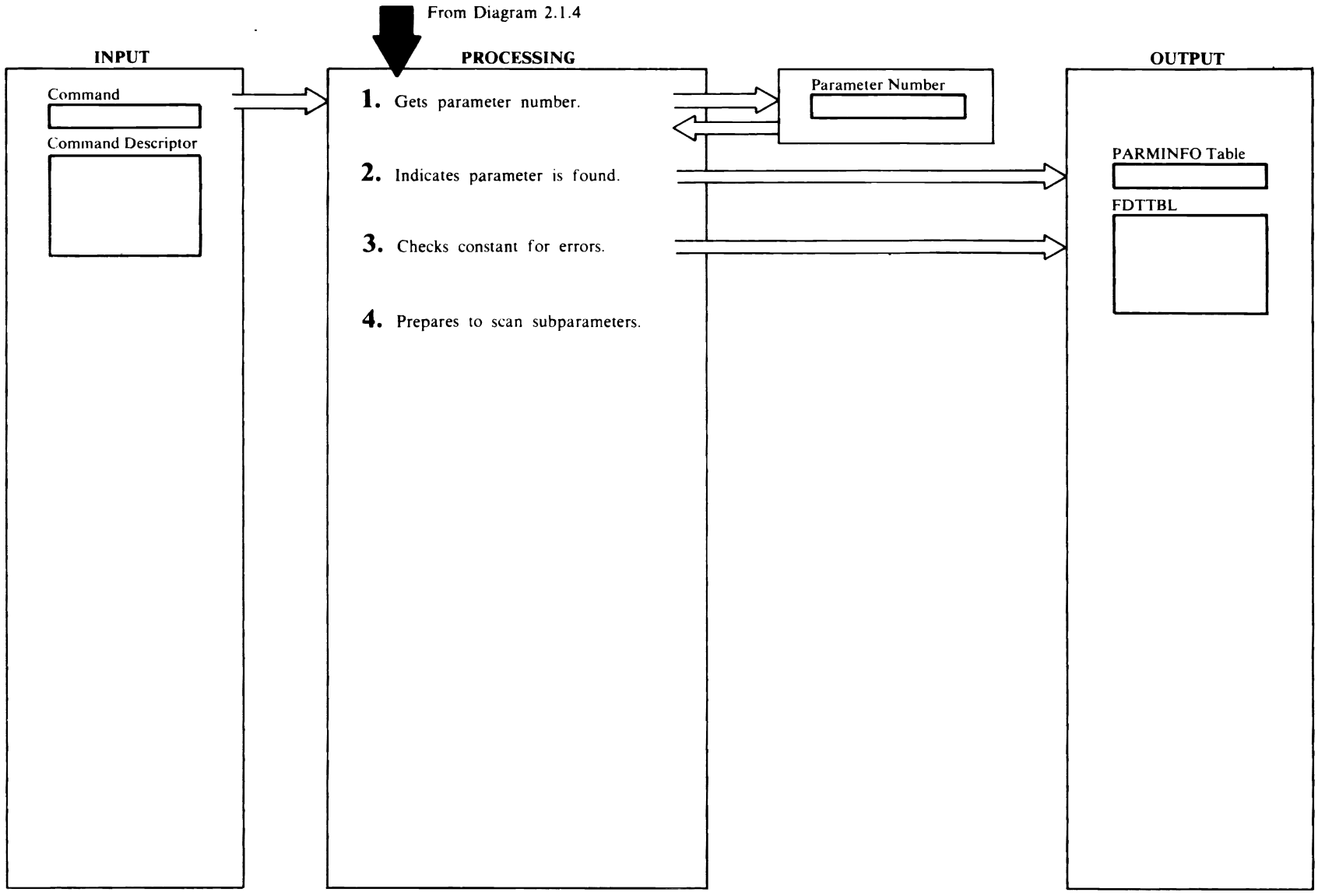
   c. As SCANCMD scans the command, BUILDFDT encodes the command into the FDT in order to describe the command to the FSR that will execute it. The data items are checked for additional errors (errors are processed as described in step 1.b). Parameter scanning continues one parameter at a time until the end of a repeated parameter list is reached or until the command terminator is found. For positional parameters and data belonging to keywords, BUILDFDT checks to ensure that a string does not exceed the allowed length, that a number is not out of range, and that there are not too many elements in a list.

Module: IDCRI01

Procedure: DEFAULTS, SETDFLT, NEEDNOTS

2. The PARMINFO Table is used to access the description of each parameter. If a repeated group of parameters or a command is incomplete, default values are supplied to the FDT. The defaults, which are in the command descriptor, are always supplied whenever an input parameter is omitted, unless the defaults conflict with the input parameters. DEFAULTS and SETDFLT check to ensure that the combination of defaults supplied for the command is meaningful, that is, no parameters that are syntactically correct but logically incorrect. PARMFLAG and the command descriptor are used to make inter-parameter checks for missing keywords and mutually exclusive keywords. If command scanning is not complete, control returns to step 1 to obtain the next parameter.

Diagram 2.1.4.1 Reader/Interpreter for Job Batched Jobs Syntax Check Parameter



Extended Description for Diagram 2.1.4.1

Module: IDCRI01

Procedure: SCANCMD, KWDPARM

1. The identification number is found differently for positional and keyword parameters. For a positional parameter, SCANCMD obtains the number of the parameter from the subparameter ID number list in the current superparameter's descriptor. For a keyword parameter, KWDPARM compares the keyword to every possible keyword permitted in the current level of parameter processing. When a match is found, KWDPARM saves the ID number of the keyword.

Module: IDCRI01

Procedure: SETFLAG

2. SETFLAG uses the ID number of the parameter as an index to the FDT. SETFLAG puts the address of the FDT field in the same FDT field—the FDT field points to itself—to indicate that the parameter has been found. If the parameter has data, the FDT field will be changed later to the address of the data. Also, SETFLAG sets the PARMFLAG value to 1 for this parameter to indicate the parameter has been found in the command.

Module: IDCRI01

Procedure: GETDATA, CONVERT, PACKCVB, DSIDCHK, ERROR2

3. If the parameter is a constant in the case of positional parameters, or if a constant is associated with the parameter in the case of a keyword parameter, GETDATA checks the constant for syntax errors. If an error is encountered, ERROR2 issues a message on SYSPRINT and sets LASTCC to 12. In Access Method Services, a constant is one of the following:

   - dsname/password
   - dsname(membername)/password
   - dname/password
   - 'character string'
   - character string
   - X'hexadecimal digits'
   - decimal digits
   - B'binary digits'

   A list of constants is several constants in the same format following each other. A constant or a list of constants may belong to one parameter.

Module: IDCRI01

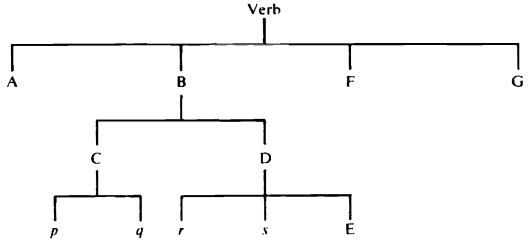
Procedure: SCANCMD

4. If the keyword parameter has subparameters associated with it, SCANCMD processes the subparameters next. For example, if the following command is specified:

   VERB A(*x*) B(C(*p q*) D(*r s* E(*x*))) F G(*x*)

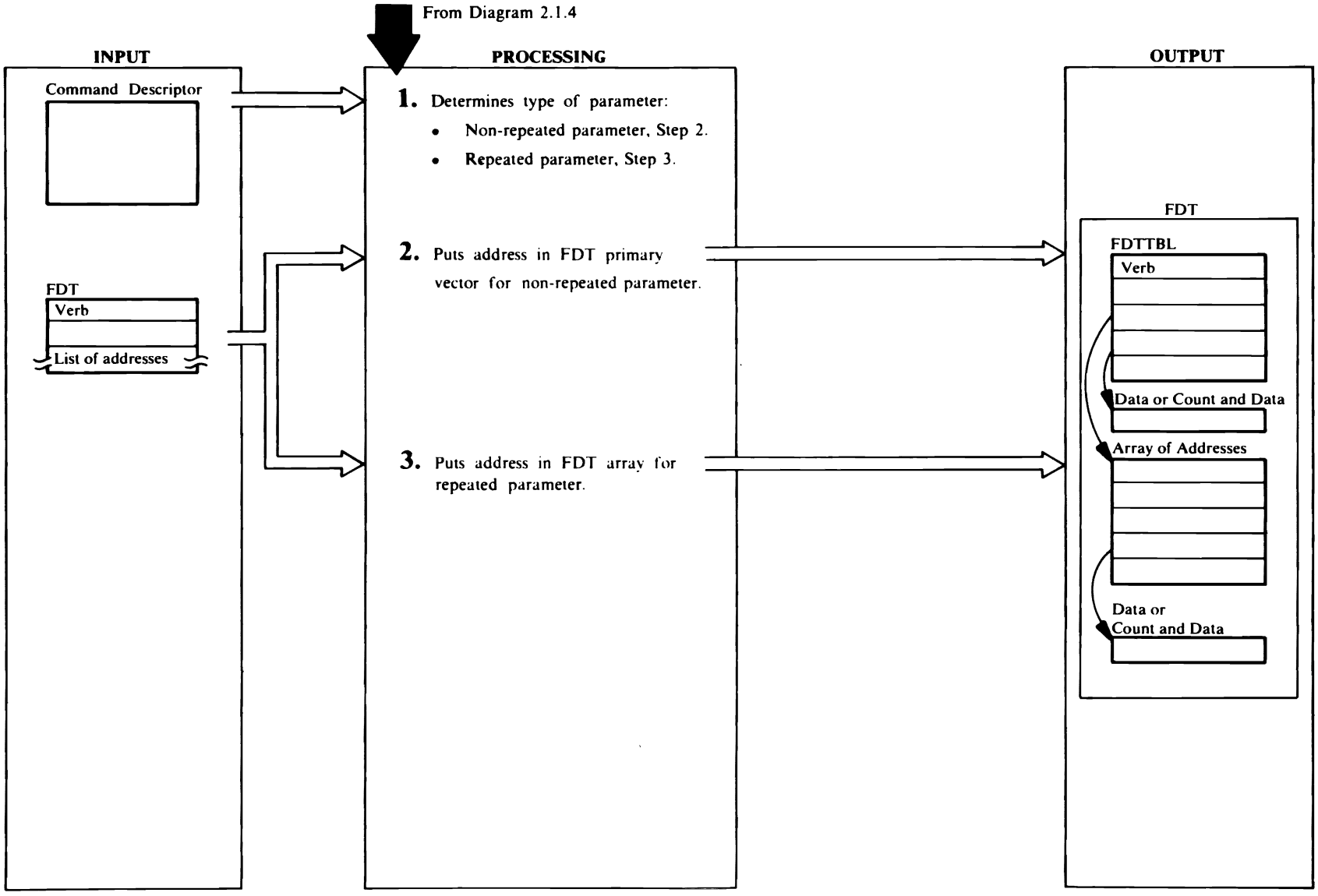
   A, B, C, D, E, F, and G are keyword parameters. *p*, *q*, *r*, and *s* are positional parameters. *x* represents data.

   The command has the following structure for scanning:



   The structure is in levels of parameter dependency. The verb is on level zero. Parameters A, B, F, and G are on level one. When the Reader/Interpreter, R/I, scans level one and finds parameter B, the scanning begins one level lower with parameters C and D on level two. When parameter C is found, the scan again moves one level lower to scan the C subparameters. At the end of the C subparameters, the scan returns to level two to scan the next parameter on level two. At the end of the D subparameters, there are no more parameters on level two, and the scan returns to level one for parameter F. In other words, the parameters are processed in the same order that they appear on the input statement. R/I keeps the level number of the parameter being scanned in PARMLVL. R/I keeps the ID number of the superparameter for the level being scanned in SUPERID. R/I keeps the ID number of the parameter being scanned in PARMID. Control returns to Diagram 2.1.4, step 1.c.

Diagram 2.1.4.2 Reader/Interpreter Build FDT



Extended Description for Diagram 2.1.4.2

Module: IDCRI01

Procedure: PACKCVB, CONVERT, GETSPACE, MORSPACE

1. The parameter type determines how it is encoded into the FDT. If the parameter cannot be repeated, processing continues with step 2; if the parameter can be repeated, processing continues with step 3. Refer to Diagram 2.1.3 for a definition of parameter.
2. A non-repeated parameter is one of the following:
   - A keyword with no data and no repeated subparameters
   - A keyword with no data and repeated subparameters
   - A positional or keyword parameter with a single constant as data
   - A positional or keyword parameter with a list of constants as data

   Each category is encoded differently into the FDT as follows in the same order as above:
   - The address in the FDT points to itself
   - The address in the FDT points to a fullword containing the number of subparameter repetitions
   - The address in the FDT points to the single constant
   - The address in the FDT points to a halfword containing the number of constants and immediately preceding the list of constants

   Character string constants are not changed, but PACKCVB and CONVERT convert numbers and hexadecimal strings to binary before the address is put in the FDT. If a list of constants is found, GETSPACE obtains space for the list when the first constant is processed. MORESPACE obtains additional space, if necessary. In the Reader/Interpreter, R/I, listings, the word *scaler* is interchangeable with the word *constant*. Control returns to Diagram 2.1.4 for the next parameter.

Module: IDCRI01

Procedure: SCANMD, INREPEAT, DEFAULTS, NEEDNOTS

3. Each repeated parameter—postional or keyword— is one of two repetition types.

Repetition Type 1
The repeated parameter is not embedded in another repeated parameter. The *objectname* parameter in the IMPORT command has Type 1 repetition.

Repetition Type 2
The repeated parameter is embedded within another repeated parameter. The *lowkey* parameter in the IMPORT command has Type 2 repetition.

The maximum number of repetitions for a parameter is in the command descriptor for the parameter. The R/I uses the repetition type to insert the addresses of the data associated with the parameter in a secondary FDT array of addresses. The address of the array is put in the primary FDT. For each repetition type the FDT array is different.

Repetition Type 1
The array is one-dimensional and contains one address for each possible occurrence of the parameter.

Repetition Type 2
The array is two-dimensional. There is one row for each possible occurrence of the Type 1 or outer parameter. There is one column for each possible occurrence of the Type 2 or inner parameter.

Consider a command in the following format:

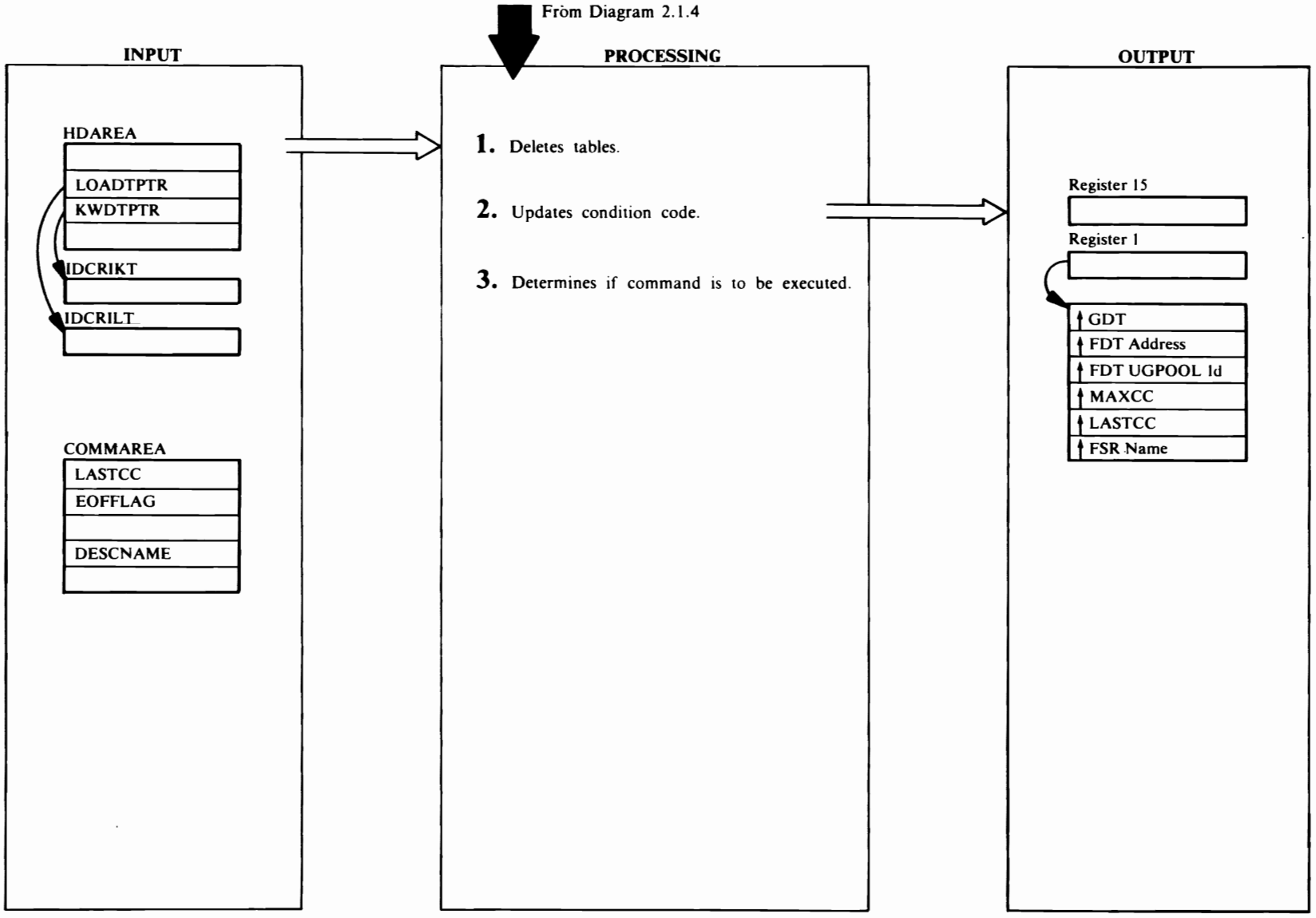
VERB A((B(C D((*x y*) ...)) E) ...) F

The type 1 parameters are B, C, D, and E because the entire parameter set (B(C D((*x y*) ...)) E) can be repeated, but it is not embedded in another repeated parameter.

The Type 2 parameters are *x* and *y* because (*x y*) can be repeated, and it is embedded in another repeated parameter. A one dimensional array is built for each Type 1 parameter, B, C, D, and E, but a two dimensional array is built for each Type 2 parameter, *x* and *y*.

The data from each repetition of a parameter is treated as in step 2, but instead of putting the data address in the primary FDT array, R/I puts the address in the secondary array of addresses for the parameter. In the R/I listings, repetition type is called *repeatedness nesting*. Refer to the examples of FDT in the "Data Areas" chapter. Control returns to Diagram 2.1.4 for the next parameter.

Diagram 2.1.5 Reader/Interpreter for Batched Jobs Termination



Extended Description for Diagram 2.1.5

Module: IDCRI03

Procedure: IDCRI03

1. IDCRI03 deletes the command descriptor table for the current command and temporary storage. If end-of-file or a severe error is encountered, IDCRI03 deletes the command name table, IDCRILT, the modal name table, IDCRIKT, and HDAREA.

Module: IDCRI03

Procedure: IDCRI03

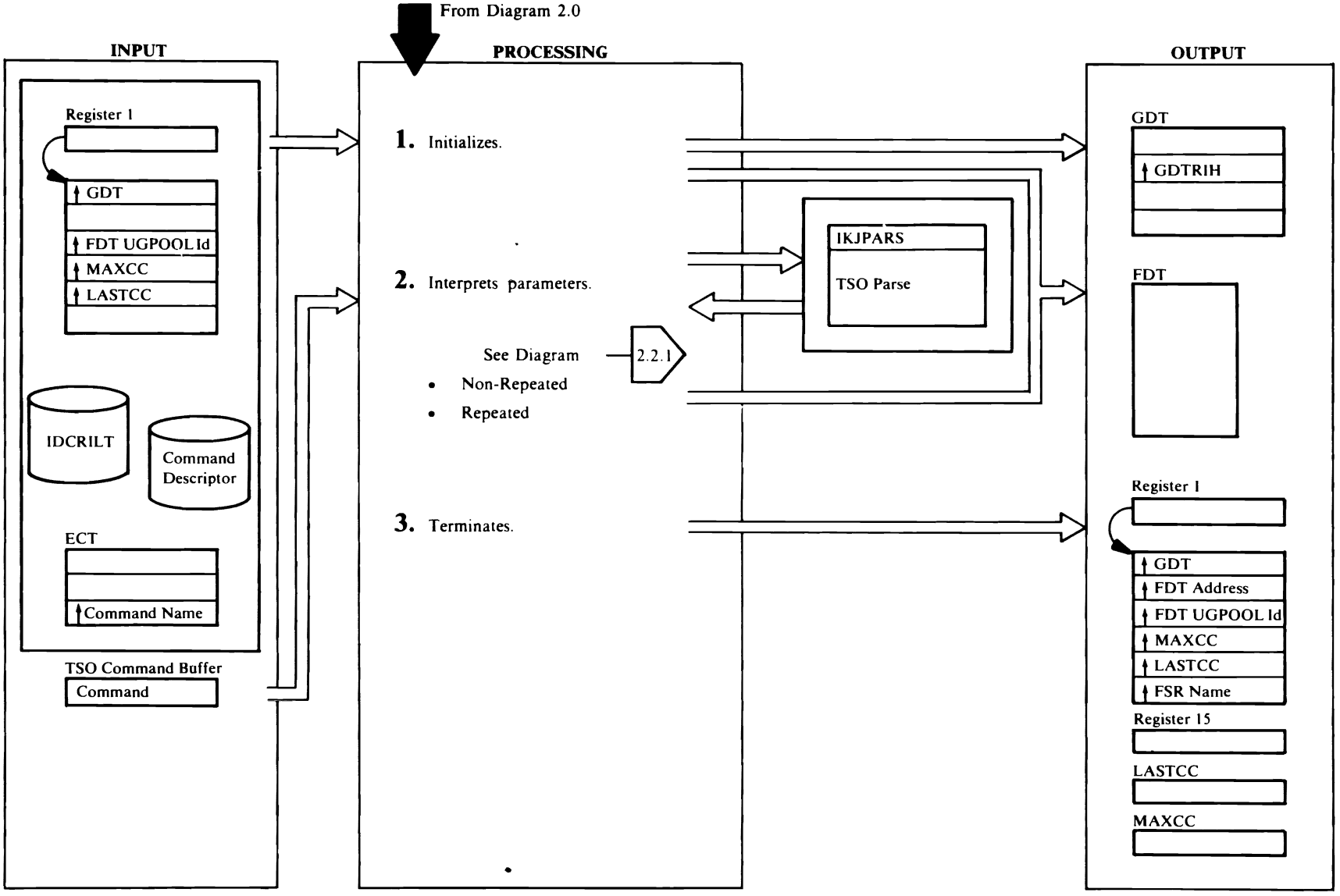
2. If end-of-file is encountered on SYSIN, IDCRI03 sets a flag in COMMAREA, and IDCRI01 puts a non-zero value in register 15, indicating that the Executive is not to call the Reader/Interpreter, R/I, again. If end-of-file has not been encountered and no severe errors were found, IDCRI01 sets register 15 to zero. If an error causes the R/I to terminate all processing, IDCRI03 prints an error message on SYSPRINT. IDCRI03 sets MAXCC to 16 which indicates that the Executive is not to call the R/I again.

Module: IDCRI03, IDCRI01

Procedure: IDCRI03, IDCRI01

3. If the command had errors or was being scanned only for syntax errors due to a modal expression, IDCRI03 releases the FDT and gives control to Diagram 2.1.2 to get the next command from SYSIN. If the command is to be executed or severe errors were encountered, IDCRI01 gives control to Executive Controlled Termination, Diagram 4.0.

Diagram 2.2 Reader/Interpreter for Interactive TSO



Extended Description for Diagram 2.2

Module: IDCRI04

Procedure: IDCRI04, RISETUP

1. GDTRIH is zero before the first call to the Reader/Interpreter, R/I. If GDTRIH is zero, IDCRI04 puts the address of GDTRIH in GDTRIH; if GDTRIH is non-zero, control goes to step 3 to terminate the R/I. RISETUP gets the command name from the Environment Control Table, ECT. TSO builds the ECT before Access Method Services gets control. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor* for information about the ECT. The command name table, IDCRILT, is loaded with a ULOAD macro. RISETUP issues a ULOAD macro to load the command descriptor for the command name from the ECT. A UGPOOL obtains space for the PARMINFO table which will contain information about each parameter, and the RSI table which will contain information about repeated parameters. Refer to Diagram 2.1.3 for the definition of *parameter*.

   The FDT contains the information from the command that an FSR needs to execute the command. The FDT is the interface between the R/I and the FSRs and consists of a primary array of addresses, a secondary array of addresses for each repeated parameter, and encoded data from the command. RISETUP gets space with a UGPOOL macro for the FDT primary array of addresses and arrays of repetition counts. The UGPOOL identification for the FDT is given to the R/I when the R/I is given control. Because RISETUP uses a UGPOOL macro to obtain storage, the identifier 'EX00' precedes the FDT. The verb uses eight bytes, and each parameter uses four additional bytes. More storage is obtained if any parameter is repeated. The amount of storage for repeated parameters is calculated from the command descriptor or from the number of repeated parameters in the command. RISETUP initializes the FDT to zero and places the verb name in the first eight bytes.

Module: IDCRI04

Procedure: MAINSCAN, SUBSCAN, TRNSLATE, NOTPARMS, RESOLVE, DEFAULTS, SETDFLT, NEEDPRMS, ADDPARM

2. The command's parameters are checked and encoded into the FDT in three groups—first, non-repeated parameters; second, repeated parameters of Type 1 repetition; and third, repeated parameters of Type 2 repetition. Refer to Diagram 2.1.4.2 for a definition of *repetition Type*. Within each group, the individual parameters are checked and encoded into the FDT in the order they appear in the command descriptor. As the non-repeated parameters are checked and encoded into the FDT, information is gathered about the repeated parameters with Type 1 repetition. As the repeated parameters with Type 1 repetition are checked and encoded into the FDT, information is gathered about the repeated parameters with Type 2 repetition. After all parameters have been encoded into the FDT, control goes to step 3.

   - For the non-repeated parameter group, MAINSCAN builds a Parse Parameter List, PPL, which points to the primary Parameter Control List, PCL, in the command descriptor and to the TSO command buffer. MAINSCAN then issues a ULINK macro to give control to IKJPARS. **Note: For VS2.03.807** a CALLTSSR macro is issued rather than ULINK. IKJPARS parses the non-repeated parameters and handles syntax errors. The output of IKJPARS is a Parameter Descriptor List, PDL, which contains the encoded non-repeated parameters. The repeated parameters are also in the PDL, but they are not encoded—just a long string of data. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or Command Processor* for more information on IKJPARS and its input and output. If the return code from IKJPARS is non-zero, MAINSCAN sets LASTCC to 12 and gives control to step 3 for R/I termination.

     For each possible parameter in the group, TRNSLATE finds the parameter in the command descriptor. TRNSLATE uses the offset into the PDL in the command descriptor to find the PDE for the parameter. If the parameter exists in the PDL, BUILDFDT encodes it into the FDT from the PDL. Refer to Diagram 2.1.4.2 for a description of how each parameter is encoded into the FDT. The R/I must check the command for missing parameters, default parameters, and incompatible parameters. If NOTPARMS finds two incompatible parameters, RESOLVE issues a UPROMPT macro so the TSO terminal user can choose the parameter. DEFAULTS and SETDFLT supply defaults for any missing parameters that have defaults in the command descriptor. NEEDPRMS checks for any required non-repeated parameters that are missing. ADDPARM issues a UPROMPT macro so the TSO terminal user can supply the missing subparameters. After missing parameters are supplied, IKJPARS parses them as described above, and BUILDFDT encodes them into the FDT. After missing non-repeated parameters are supplied. NOTPARMS and DEFAULTS check the new version of the command. See Diagram 2.2.1 for more information on parameter checking.

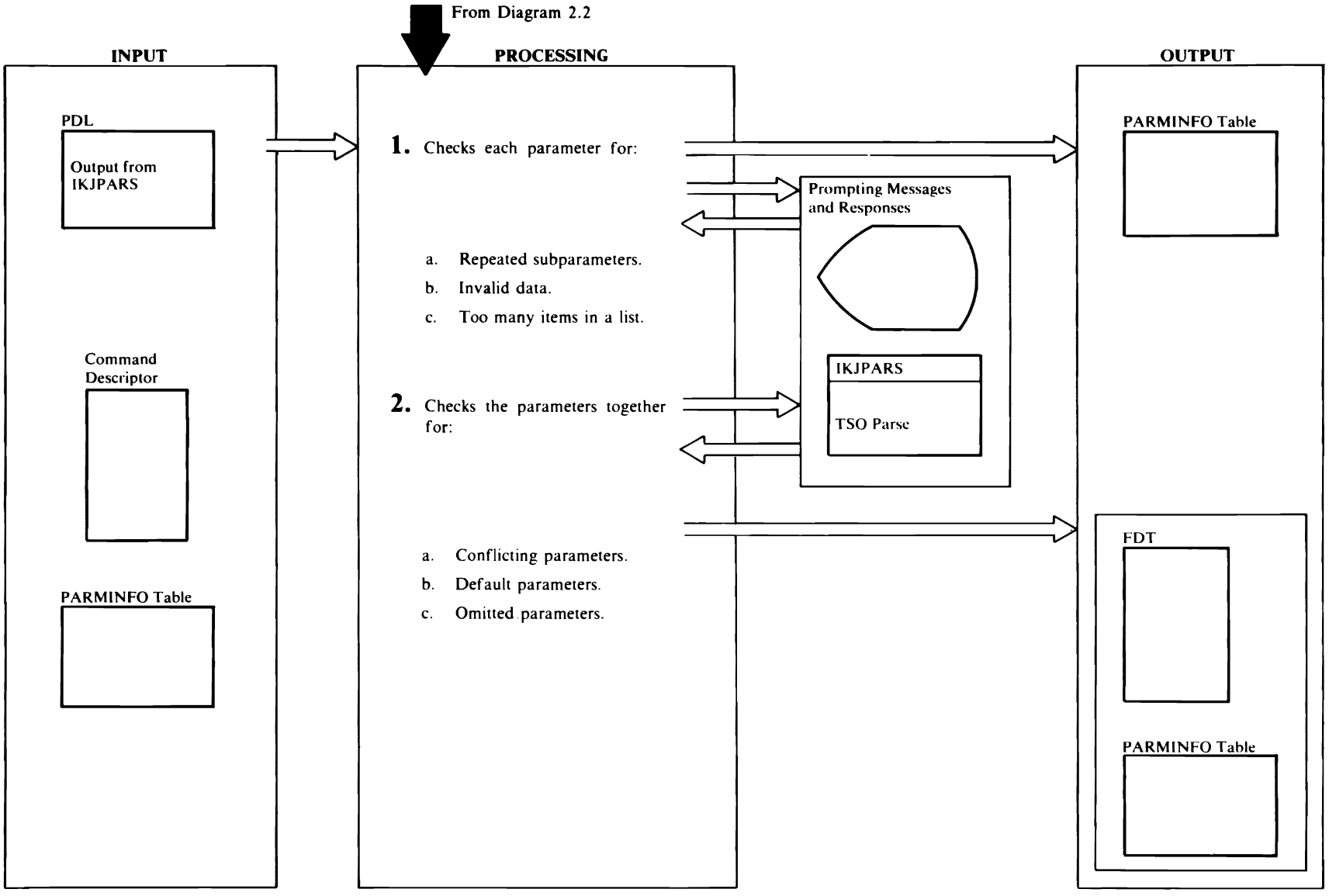
   - Both types of repeated parameters are divided into parameter sets. A parameter set is several parameters repeated together. In the IMPORT command, all the subparameters of OBJECTS are a parameter set. Each (*lowkey highkey*) subparameter pair of the KEYRANGES parameter is a parameter set. First, the parameter sets with Type 1 repetition are checked and encoded into the FDT. Second, the parameter sets with Type 2 repetition are checked and encoded into the FDT. Each parameter set is treated separately from other parameter sets. The following occurs for each parameter set. SUBSCAN gets information about the parameter set from the Repeated Sublist Index, RSI. SUBSCAN uses the information to build a PPL which points to the Parameter Control List, PCL, and to the parameter set. The PCL describes all possible parameters in the parameter set. SUBSCAN issues a ULINK macro to give control to IKJPARS. **Note: For VS2.03.807** a CALLTSSR macro is issued rather than ULINK. IKJPARS parses the parameter set and handles syntax errors. The output from IKJPARS is a PDL which contains the encoded parameter set. If the return code from IKJPARS is non-zero, SUBSCAN sets LASTCC to 12 and gives control to step 3 for R/I termination. The parameter set is encoded into the FDT and checked as described in the preceding paragraph.

Module: IDCRI04

Procedure: RITERM

3. If either GDTRIH or LASTCC is non-zero, RITERM puts a non-zero number in register 15 as a signal to the Executive to terminate Access Method Services. If both GDTRIH and LASTCC are zero, RITERM puts a zero in register 15 so the Executive will give the command to a FSR. RITERM frees all temporary storage. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 2.2.1 Reader/Interpreter for Interactive TSO: Checking Parameters



Extended Description for Diagram 2.2.1

Module: IDCRI04

Procedure: TRNSLATE, FINDPDE, REPLIST, TOOMANY, NEWPARM, BUILDFDT, GETSPACE

1. TRNSLATE checks for each possible parameter either in the group of non-repeated parameters or in the set of repeated parameters. For each possible parameter, FINDPDE obtains the offset of its Parameter Descriptor Entry, PDE, from the command descriptor. The PDE is always at the same offset within the Parameter Descriptor List, PDL. Then TRNSLATE checks the PDE to see if the parameter is coded in the command.

   a. If the parameter contains repeated subparameters, REPLIST saves information about the repeated subparameters in the Repeated Sublist Index table, RSI. RSI contains the address, length, repetition number, and superparameter identification number for each repetition of the subparameters. Refer to Diagram 2.1.3 for a definition of *superparameter*. REPLIST issues a UGPOOL macro to obtain storage for the secondary FDT array of addresses for each repetition of the subparameters. REPLIST also calculates the offset in the FDT for each repetition of the subparameters. If too many repeated subparameters appear for the parameter, TOOMANY issues a UPROMPT macro so the TSO terminal user can choose whether to ignore the extra subparameters or to terminate the command. If the parameter should have repeated subparameters and none appear in the command, NEWPARM issues a UPROMPT macro so the TSO terminal user can supply the missing subparameters. REPLIST saves information about the new subparameters in RSI as if the subparameters had originally appeared in the command. However, no further checking is done on the repeated subparameters at this time.

   b. BUILDFDT checks data belonging to a parameter. If the data is incorrect, NEWPARM issues a UPROMPT so the TSO terminal user can supply new data. BUILDFDT points to a Parameter Control List, PCL, that describes the new data and issues a ULINK macro to give control to the TSO Parse Routine, IKJPARS. **Note: For VS2.03.807** IKJPARS receives control via the CALLTSSR macro rather than ULINK. IKJPARS checks the new data for syntax errors and returns the data encoded in a Parameter Descriptor List, PDL. BUILDFDT puts the correct data in the FDT.

   c. GETSPACE gets storage for the data and places the address of the storage in the FDT. GETSPACE counts the number of constants in a list. If too many constants appear, TOOMANY issues a UPROMPT macro so the TSO terminal user can choose whether to ignore the extra constants or to terminate the command.

Module: IDCRI04

Procedure: NOTPARMS, RESOLVE, DEFAULTS, NEEDPRMS, ADDPARM, SETDFLT

2. The parameters as a group of non-repeated parameters or as a set of repeated parameters are checked for incompatible parameters, default parameters, and missing parameters.

   a. NOTPARMS checks the parameters against the command descriptor. If any two parameters are incompatible, RESOLVE issues a UPROMPT so the TSO terminal user can choose one. RESOLVE removes the not-chosen parameter from the FDT.

   b. If any missing parameter has a default, DEFAULTS checks the command descriptor to find the default. DEFAULTS points to a Parameter Control List, PCL, that describes the default and issues a ULINK macro to give control to the TSO Parse Routine, IKJPARS. **Note: For VS2.03.807** IKJPARS receives control via the CALLTSSR macro rather than ULINK. IKJPARS checks the default and returns the default parameter encoded in a Parameter Descriptor List, PDL. SETDFLT adds each defaulted parameter to the FDT.

   c. After incompatible and default parameters are checked, NEEDPRMS checks for missing parameters. If any parameters are still missing, ADDPARM issues a UPROMPT so the TSO terminal user can supply the missing parameter(s). ADDPARM points to a Parameter Control List, PCL and issues a ULINK macro to give control to the TSO Parse Routine, IKJPARS. **Note: For VS2.03.807** IKJPARS receives control via the CALLTSSR macro rather than ULINK. IKJPARS checks the new parameter for syntax errors and returns the parameter encoded in a Parameter Descriptor List, PDL. The parameter is put in the FDT as described in Diagram 2.1.4.2. If any new parameters are added to the FDT, step 2 is repeated until no new parameters are added by step 2.c.

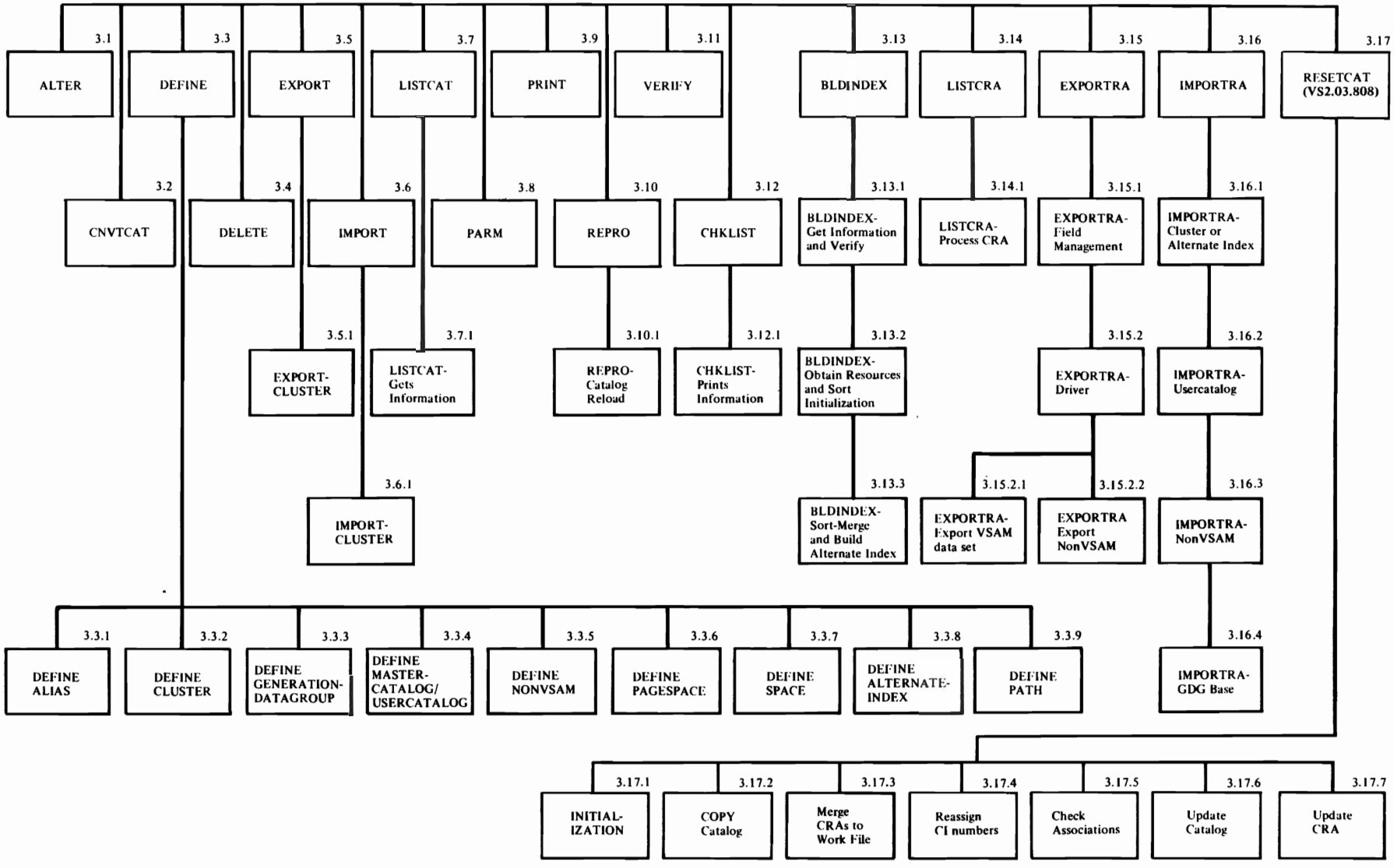
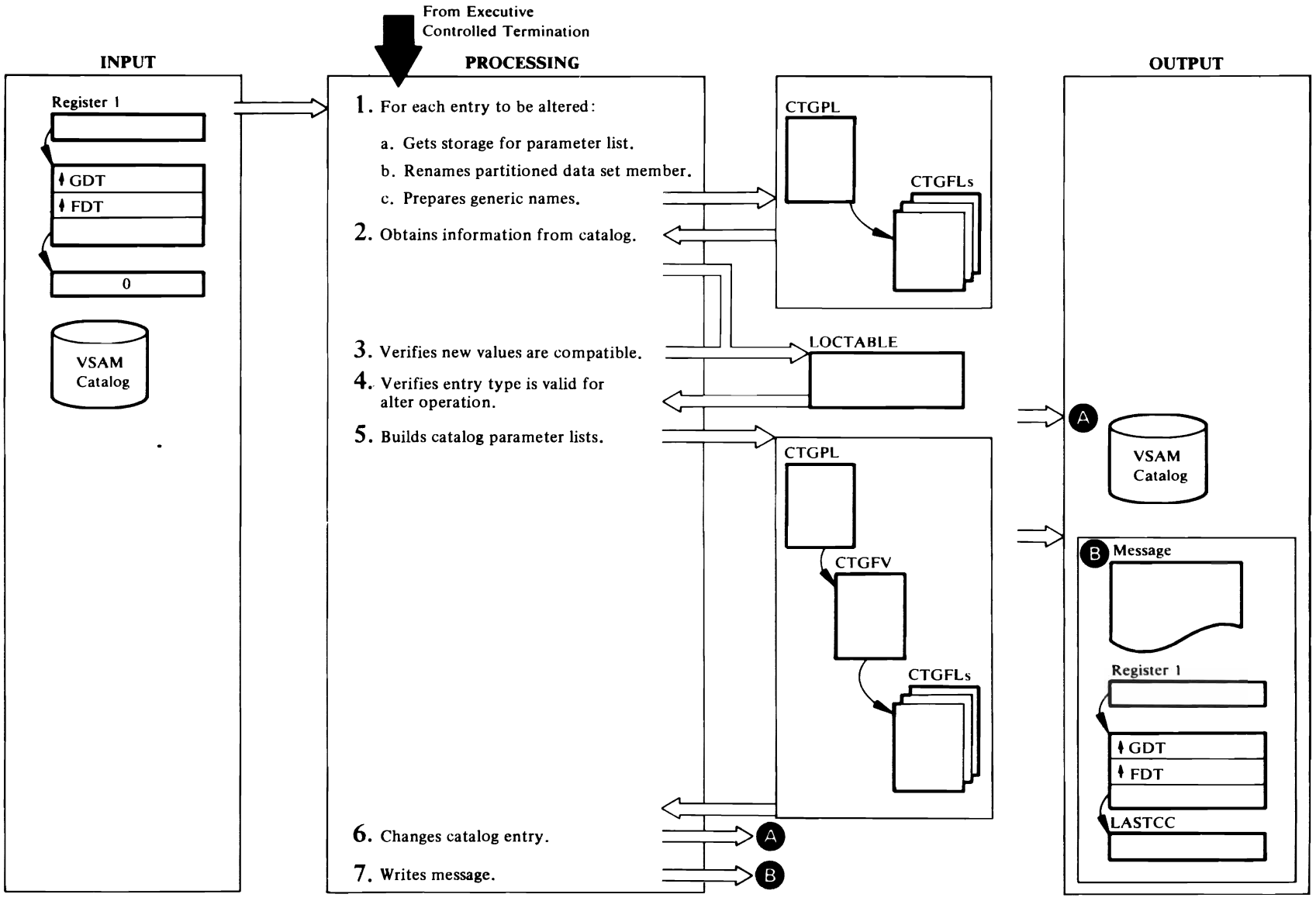

3.1
ALTER
3.3
DEFINE
3.5
EXPORT
3.7
LISTCAT
3.9
PRINT
3.11
VERIFY
3.13
BLDINDEX
3.14
LISTCRA
3.15
EXPORTRA
3.16
IMPORTRA
3.17
RESETCAT (VS2.03.808)
3.2
CNVTCAT
3.4
DELETE
3.6
IMPORT
3.8
PARM
3.10
REPRO
3.12
CHKLIST
3.13.1
BLDINDEX-Get Information and Verify
3.14.1
LISTCRA-Process CRA
3.15.1
EXPORTRA-Field Management
3.16.1
IMPORTRA-Cluster or Alternate Index
3.5.1
EXPORT-CLUSTER
3.7.1
LISTCAT-Gets Information
3.10.1
REPRO-Catalog Reload
3.12.1
CHKLIST-Prints Information
3.13.2
BLDINDEX-Obtain Resources and Sort Initialization
3.15.2
EXPORTRA-Driver
3.16.2
IMPORTRA-Usercatalog
3.6.1
IMPORT-CLUSTER
3.13.3
BLDINDEX-Sort-Merge and Build Alternate Index
3.15.2.1
EXPORTRA-Export VSAM data set
3.15.2.2
EXPORTRA Export NonVSAM
3.16.3
IMPORTRA-NonVSAM
3.3.1
DEFINE ALIAS
3.3.2
DEFINE CLUSTER
3.3.3
DEFINE GENERATION-DATAGROUP
3.3.4
DEFINE MASTER-CATALOG/ USERCATALOG
3.3.5
DEFINE NONVSAM
3.3.6
DEFINE PAGESPACE
3.3.7
DEFINE SPACE
3.3.8
DEFINE ALTERNATE-INDEX
3.3.9
DEFINE PATH
3.16.4
IMPORTRA-GDG Base
3.17.1
INITIAL-IZATION
3.17.2
COPY Catalog
3.17.3
Merge CRAs to Work File
3.17.4
Reassign CI numbers
3.17.5
Check Associations
3.17.6
Update Catalog
3.17.7
Update CRA

Diagram 3.1 ALTER FSR



Extended Description for Diagram 3.1

Module: IDCAL01

Procedure: IDCAL01, MEMRENAM

1. IDCAL01 tests the FDT to see if the *entryname* needs to be qualified. Data set names are qualified if Access Method Services is invoked interactively with TSO. If the *entryname* needs qualification, IDCAL01 issues a UQUAL macro for the name and uses the returned data set name for the remainder of the ALTER FSR. If the *entryname* is qualified by UQUAL and NEWNAME is specified and needs qualification, IDCAL01 puts the trailing qualifier from the *entryname* on the NEWNAME. This modified NEWNAME is used for the remainder of the ALTER FSR.

   a. IDCAL01 gets storage for the catalog parameter list.

   b. IDCAL01 tests the FDT to see if the *entryname* contains a *membername* for a partitioned data set and NEWNAME is specified. If it is not, control continues with step 1.c. MEMRENAM checks to be sure that the partitioned data set name from *entryname* and the partitioned data set name from NEWNAME are identical. The two names can both be aliases, but they must match character for character. If they do not match, control goes to step 7. MEMRENAM builds an OPNAGL. If FILE is not specified, MEMRENAM puts the *entryname* in the OPNAGL. This causes the partitioned data set to be dynamically allocated by the UOPEN macro. MEMRENAM issues a UOPEN to open the partitioned data set. If FILE is specified, MEMRENAM compares the *entryname* with the data set name returned by UOPEN. If the names do not match, control goes to step 7. MEMRENAM then issues a USTOW macro with the new member name to rename a member of the partitioned data set. A UCLOSE macro closes the partitioned data set.

   c. IDCAL01 tests the FDT to determine if the *entryname* is a generic name—that is, it contains an *. If it is a generic name, IDCAL01 issues a UCIR macro to get a list of *entrynames*. Each name in the list is treated as though it had been specified as an *entryname* in the ALTER command. A catalog name returned with the data set names that match the *entryname* is saved and used in calls to VSAM catalog management.

   If NEWNAME is specified, IDCAL01 tests the FDT to determine if the NEWNAME is a generic name. If it is generic, IDCAL01 builds a unique NEWNAME for each *entryname*. If either the *entryname* or the NEWNAME is generic, they must both be generic. The * in this NEWNAME is replaced with whatever characters replaced the * in the *entryname*. These characters will vary for each *entryname* in the list. If the resulting NEWNAME is greater than 44 characters, this *entryname* is not processed and control returns to step 1 for the next *entryname*.

Module: IDCAL01

Procedure: LOCATPRC

2. Due to the arrangement of some information in a VSAM catalog, in order to change part of a field, the entire field must be retrieved and changed. If only NEWNAME, OWNER | NULLIFY (OWNER), TO | FOR | NULLIFY (RETENTION), EXCEPTION EXIT, NOUPGRADE, UPDATE | NOUPDATE, or BUFFERSPACE is specified, control goes to step 5. LOCATPRC builds a CTGPL and CTGFLs which reference the PASSWALL, DSATTR, GDGATTR, AMDSBCAT, ENTYPE, CATACB, CRAVOL, RGATTR, and HURBADS catalog fields. Refer to the list at the end of this description for the contents of the catalog fields obtained with a particular CTGFL. LOCATPRC issues a UCATLG macro to retrieve the information from the catalog. If the return code is zero, LOCATPRC uses the returned information to build a table, LOCTABLE.

   If the return code is 40, the work area for VSAM is too small. LOCATPRC increases the work area and reissues the UCATLG. If the return code is any other non-zero number, the ALTER command is terminated and control goes to step 7.

Module: IDCAL01

Procedure: CHECKPRC

3. Following the primary locate, IDCAL01 will invoke CHECKPRC if any of the following parameters were specified: UPGRADE, KEYS, RECORDSIZE, UNIQUEKEY. CHECKPRC will perform further verification of these parameters which will, in most cases, require additional locates (called 'secondary' locates). Password processing for the primary and secondary locates and for the Alter function itself is handled as follows:

   If KEYS and RECORDSIZE are not specified:

   a. On the primary locate, if a password is supplied, it is referenced from the CPL. The verify master password bit is set.

   b. If UPGRADE is specified, a secondary locate for the data HURBADS is required. If a password is supplied, it is referenced from the CPL. The verify master password bit is turned off. The password (which is that of the cluster level) will be verified as being read level or higher.

   c. On the Alter, if a password is supplied, it is referenced from the CPL. The verify master password bit is turned off.

   If KEYS and/or RECORDSIZE are specified:

   a. On the primary locate, if a password is supplied, it is referenced from the CPL. The verify master password bit is set.

   b. On the secondary locates, if a password is supplied, it is referenced from the CPL. The verify master password bit is turned off. The bypass verification bit is turned on. No verification will take place and the requested information will be returned.

   c. On the Alter, processing is as described in b above.

   If UPGRADE was specified, CHECKPRC verifies that the ENTYPE is a G (alternate index). If UPGRADE was specified, CHECKPRC verifies that the high-used RBA is zero. This latter check will require a locate of the data HURBADS. If UNIQUEKEY was specified when the attribute was previously NONUNIQUEKEY, CHECKPRC verifies that the high-used RBA of the data object is zero and that the data object is associated with an alternate index. If any of these error checks fail, a message is printed and processing is terminated.

Module: IDCAL01

Procedure: CHECKPRC

The major portion of the new CHECKPRC procedure performs the validity checking required to alter the KEYS and/or RECORDSIZE values of an empty data set. This checking requires the following secondary locates, based on the ENTYPE returned from the primary locate:

ENTYPE	Locates	Fields Requested
D	1-C or G association	NAMEDS (a maximum of three associations)
	2-I association C or G	AMDSBCAT
C	1-D association	AMDSBCAT, HURBADS, NAMEDS, ENTYPE, DSATTR, PASSWALL
	2-I association	AMDSBCAT
G	1-D association	AMDSBCAT, HURBADS, NAMEDS, ENTYPE, DSATTR, PASSWALL
	2-I association	AMDSBCAT
R	1-D association of AIX or cluster	AMDSBCAT, HURBADS NAMEDS, ENTYPE, DSATTR, PASSWALL
	2-I association of AIX or cluster	AMDSBCAT

Module: IDCAL01

Procedure: CHECKPRC

If the ENTYPE is none of the above, CHECKPRC returns to IDCAL01 with a terminating condition code. The LOCATE for the index AMDSBCAT will be issued only for a KSDS. CHECKPRC also verifies that the HURBADS is zero. If not, CHECKPRC returns to IDCAL01 with a terminating condition code. If object being altered is a relative record data set, CHECKPRC verifies that the average and maximum record size specified are equal and, if not, returns to IDCAL01 with a terminating condition code. If the ENTYPE returned in the primary locate is C, G or R, CHECKPRC saves the control interval number of the data component which is to be altered.

Module: IDCAL01

Procedure: CHECKPRC

After retrieval of the appropriate AMDSBCATs, the following check is made of the new average and maximum recordsizes and/or new key values.

a. Data Object

AMDRKP + AMDKEYLN ≤ AMDLRECL

or, if the object has the spanned attribute,

AMDRKP + AMDKEYLN ≤ AMDCINV - D.H.R.S

b. DATA object

AMDCINV ≥ AMDRKP + AMDKEYLN + D.R.H.S & AMDCIPCA * (AMDCINV - D.R.H.S) ≥ AMDLRECL

c. Index AMDCINV ≥ max (x,y) where:

X = I.R.H.S + (2 * (AMDKEYLN + 2)) + (3 * AMDCIPCA) + D.R.H.S

Y = I.R.H.S + (8 * AMDCIPCA) + (2 * SQRT (AMDCIPCA)) + D.R.H.S.

I.R.H.S = index record header size = 24

D.R.H.S = data record header size = 7 if non-spanned

D.R.H.S = data record header size = 10 if spanned

If any of these relationships do not hold, CHECKPRC returns to IDCAL01 with a terminating condition code.

If this is an alteration of an ESDS the index validity check is not performed. If this is an alteration of an alternate index, the AMDRKP is a fixed value of X'05'. If relative key position is specified, it applies to the position of the alternate key within the base cluster record.

If the object being altered is a alternate index and the KEYS parameter was specified, a further check will be made which requires retrieving the AMDSB of the base cluster's data component.

Module: IDCAL01

Procedure: CHECKPRC

The table below shows the locates which CHECKPRC issues based on the ENTYPE returned from the primary locate.

ENTYPE	Locates	Fields Requested
D	1-C association of G retrieved in secondary locate	NAMEDS (the first association)
	2-D association of C	AMDSBCAT (the first association)
G	1-C association retrieved in primary locate	NAMEDS
	2-D association of C	AMDSBCAT
R	1-D association of base cluster retrieved in primary locate	AMDSBCAT

Module: IDCAL01

Procedure: CHECKPRC

Using the base cluster's data AMDSB, CHECKPRC verifies the following:

AIX AMDAXRKP + AIX AMDKEYLN ≤ base cluster AMDLRECL

or, if the base cluster has the spanned attribute,

AIX AMDAXRKP + AIX AMDKEYLN ≤ base cluster AMDCINV-D.R.H.S

where D.R.H.S = 10

If either of these conditions are not true, CHECKPRC returns to IDCAL01 with a terminating error.

Assuming no terminating errors have been found, CHECKPRC now sets the appropriate return code to IDCAL01 indicating what situation was encountered. The return code will eventually be passed back to the caller, and a message written. The table below shows the return code value which will be set:

	New values are equal to previous values	New values are not equal to previous values
Previous KEYS and/or RECORDSIZE values were default values	4	0
Previous KEYS and/or RECORDSIZE values were not default values	4	8

If the return code is 0, the alter takes place. If the return code is 4, KEYS and RECORDSIZE will not be altered but alters will take place for any other parameters specified. A return code of 8 will be treated as a terminating condition code. If the

verification of the new values fails, the return code is 12.

Control is returned to IDCAL01.

Module: IDCAL01

Procedure: PARAMCHK

4. If only NEWNAME, OWNER | NULLIFY(OWNER), TO | FOR | NULLIFY(RETENTION), EXCEPTIONEXIT, NOUPGRADE, UPDATE | NOUPDATE, or BUFFERSPACE is specified, control goes to step 5. Otherwise, IDCAL01 passes control to the internal procedure PARAMCHK. PARAMCHK verifies that the parameters specified on the ALTER command are valid for the entry type of the object to be altered. The STAGE | BIND | CYLINDERFAULT, DESTAGEWAIT | NODESTAGEWAIT, WRITECHECK | NOWRITECHECK, INHIBIT | NOINHIBIT, AND SHAREOPTIONS parameters are only allowed for data or index objects. The ERASE | NOERASE, FREESPACE and UNIQUEKEY | NONUNIQUEKEY parameters are only allowed for data objects. If PARAMCHK detects an error control goes to step 7, otherwise, control goes to step 5.

Module: IDCAL01, IDCAL01

Procedure: ALTERPRC, DALCPROC

5. ALTERPRC uses the data from the ALTER command in the FDT and LOCTABLE. ALTERPRC builds a CTGPL, a CTGFV, and several CTGFLs in order to change information in the catalog. Only fields that are specified in the ALTER command are changed in the catalog. If information in a field is not being changed, the CTGFL for the field is not built. If ADDVOLUMES or REMOVEVOLUMES has been specified and the object's catalog is a recoverable catalog, ALTERPRC calls DALCPROC to perform volume allocation. DALCPROC dynamically allocates all volumes in the ADDVOLUME | REMOVEVOLUME lists and additionally allocates the catalog recovery volume for the object being altered. The table below lists the data areas that pass information to VSAM and the keywords whose data is passed.

Module: IDCAL01

Procedure: INDEXPRC

6. IDCAL01 issues a UCATLG macro to change the catalog entry. If KEYS is specified for a KSDS or an alternate index, INDEXPRC builds a CPL, FVT, and AMDSB FPL; then change the catalog entry of the associated index object. If the return code is nonzero, IDCAL01 builds an error conversion table and calls UERROR. UERROR will handle the printing of the error message.

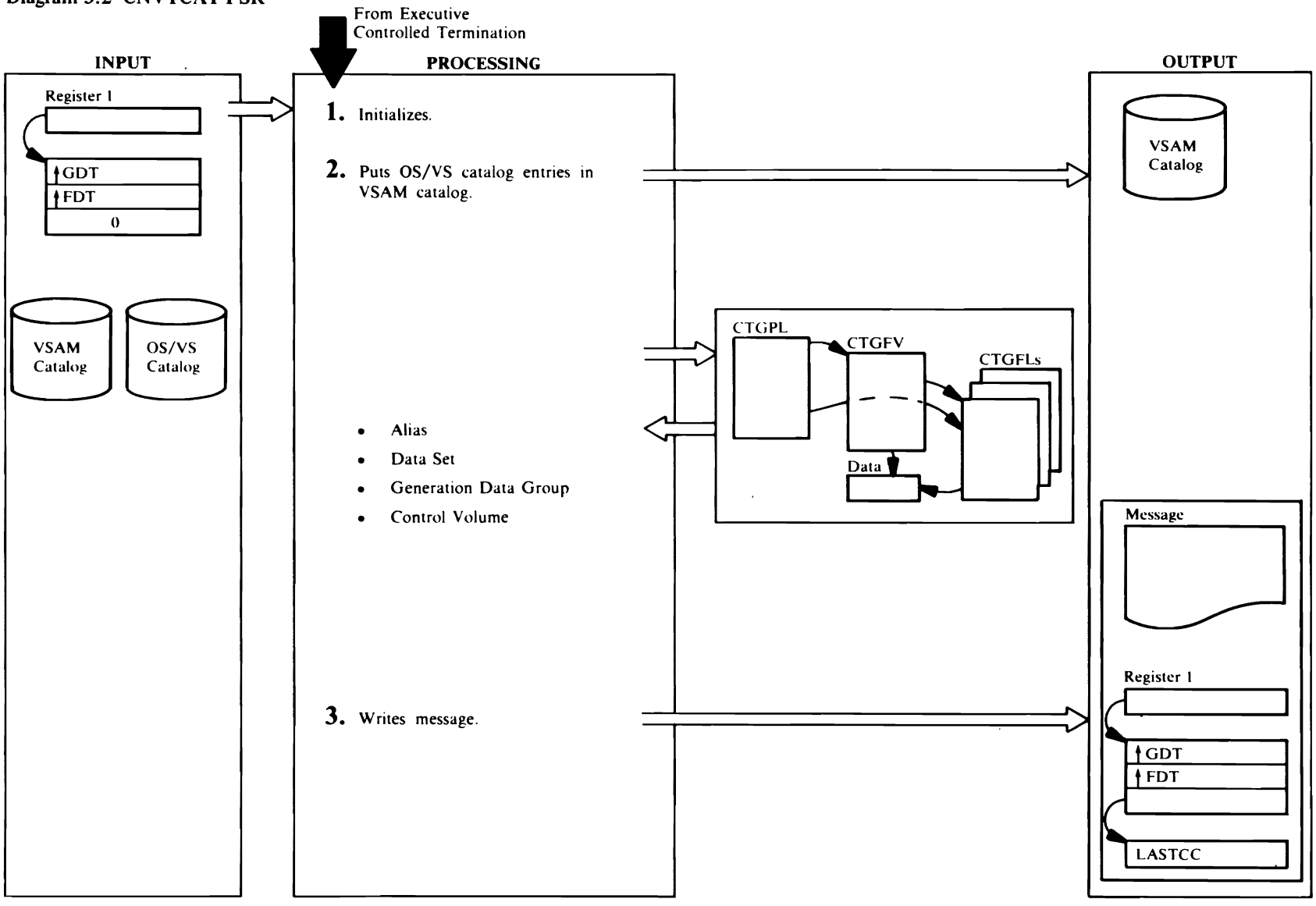
Module: IDCAL01

Procedure: IDCAL01

7. ICDAL01 also writes a message with LASTCC to SYSPRINT. If IDCAL01 opened a VSAM catalog, it closes the catalog with a UCLOSE macro. Control goes to Executive Controlled Termination.

Data Area	Keyword Data	Data Area	Keyword Data
CTGPL	NEWNAME address FILE address ADDVOLUMES address REMOVEVOLUMES address	AMDSBCAT CTGFL	FREESPACE WRITECHECK | NOWRITECHECK STAGE | BIND | CYLINDERFAULT DESTAGEWAIT | NODESTAGEWAIT KEYS RECORDSIZE-maximum UNIQUEKEY | NONUNIQUEKEY
BUFSIZE CTGFL	BUFFERSPACE		
DESTEXDT CTGFL	TO | FOR NULLIFY RETENTION		
DSATTR CTGFL	ERASE | NOERASE SHAREOPTIONS UNINHIBIT | INHIBIT	EXCPEXIT CTGFL	EXCEPTIONEXIT NULLIFY EXCEPTIONEXIT
OWNERID CTGFL	OWNER NULLIFY OWNER	RGATTR CTGFL	UPGRADE | NOUPGRADE UPDATE | NOUPDATE
PASSWALL CTGFL	MASTERPW CONTROLPW UPDATEPW READPW CODE ATTEMPTS AUTHORIZATION NULLIFY for any keywords just listed	LRECL CTGFL	RECORDSIZE-average
		GDGATTR CTGFL	EMPTY | NOEMPTY SCRATCH | NOSCRATCH

Diagram 3.2 CNVTCAT FSR



Extended Description for Diagram 3.2

Module: IDCCC01

Procedure: IDCCC01, CNVTINIT

1. IDCCC01 builds an OPNAGL. If INFILE is specified, IDCCC01 puts the DDname in the OPNAGL. If INDATASET is specified, IDCCC01 puts the *data set name* in the OPNAGL. This causes the UOPEN macro to dynamically allocate the data set. IDCCC01 issues a UOPEN macro to open the input data set. If the input data set does not open successfully, the command terminates, and control goes to step 3.

   CNVTINIT issues a UGPOOL to obtain storage for SYSCTLG index blocks, parameter lists for cataloging in the VSAM catalog, data areas required by the VSAM parameter lists, and an initial VSAM catalog work area required by VSAM for locate requests. If storage is not obtained, the command terminates, and control goes to step 3.

Module: IDCCC01

Procedure: CONVERT, AEPROC, CATALOG, GIPEPROC, CVPEPROC, ERRPROC

2. In an OS/VS catalog, a data set name is not kept as a unit. Rather, each data set name is considered as a series of qualifiers ended with a simple name. Each qualifier is in a separate index level. The CNVTCAT FSR must get each qualifier and form the complete data set name. The CNVTCAT FSR gives the completed data set name to VSAM, and VSAM puts the completed data set name as a unit in the VSAM catalog. Refer to *OS/VS2 Catalog Management Logic* for more information on catalog management. If LIST is specified, each OS/VS catalog entry is written after it is successfully put in the VSAM catalog. OS/VS catalog entries not put in the VSAM catalog are always written in a message.

   To form a data set name, CONVERT starts with an entry in the Volume Index of the OS/VS catalog and follows the chain of index levels until the simple name, the last part of the data set name, is found. CONVERT collects the names of the qualifiers as it searches for the simple name. After the data set name is cataloged in the VSAM catalog, CONVERT gets the next simple name on the last index level. When all the simple names on an index level have been found, CONVERT backs up to the next higher index level and finds any simple names or further index levels. When CONVERT has backed up to the Volume Index, CONVERT reads the next entry in the Volume Index and starts searching through the index levels again.

   When the search for a complete data set name stops, the name may be an alias of another data set, a data set name, a generation data group name, or a pointer to another OS/VS catalog—CVOL. Each type of name is processed as follows:

   - An alias is indicated by a Alias Entry. AEPROC sets an indicator that an alias has been found. An alias is an alternative name for the first qualifier. CONVERT must find the complete data set name just as it does for any data set name. See the next paragraph for more information on completing a data set name. When the data set name is formed, CATALOG issues a UCATLG macro with the alias name and the true data set name to create an alias entry in the VSAM catalog. If the true data set name is the name of a generation data group, the alias is not put in the VSAM catalog. If the alias is a duplicate, the alias is not put in the VSAM catalog, and ERRPROC is called which invokes UERROR to write an error message.
   - If a data set resides on 1 to 5 volumes, the simple name is contained in a Data Set Pointer Entry. If the data set resides on more than 5 volumes, the simple name is found in a Volume Control Block Pointer Entry. CONVERT builds VSAM parameter lists with the information from the OS/VS catalog. CATALOG issues a UCATLG macro to define the data set as a non-VSAM entry in the VSAM catalog. If the return code is zero, control returns to step 2 for the next entry in the OS/VS catalog.

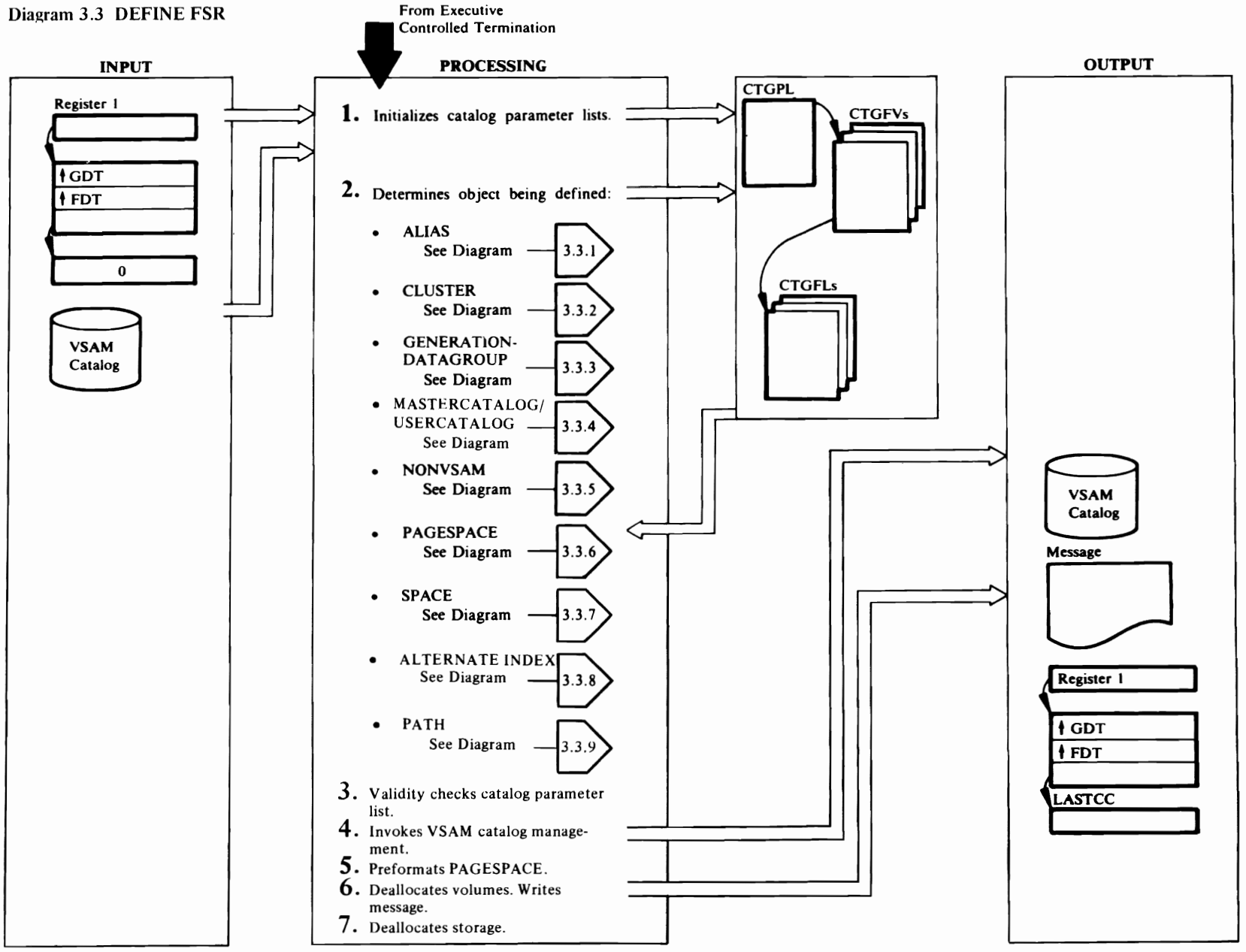
     If the return code indicates that the data set name already exists in the VSAM catalog, CATALOG tests the data set name. If the name begins with 'SYS1.', the data set is not put in the VSAM catalog and a message is written. If the name does not begin with 'SYS1.', CATALOG issues a UCATLG macro to locate more information from the VSAM catalog. If the data set in the VSAM catalog is a VSAM data set, the OS/VS data set is not put in the VSAM catalog and a message is written. If the data set in the VSAM catalog is a non-VSAM data set, it is assumed that the duplicate names represent the same data set. CATALOG compares the volumes assigned to the non-VSAM data set with the volumes assigned to the OS/VS data set. If the volumes match, the OS/VS data set is not put in the VSAM catalog, and a message is written. If the volumes do not match, CATALOG issues a UCATLG macro to change the information in the VSAM catalog about the non-VSAM data set to match the volumes assigned to the OS/VS data set. Control returns to step 2 for the next name in the OS/VS catalog.
   - GIPEPROC builds VSAM parameter lists with the information from the OS/VS catalog. CATALOG issues a UCATLG macro to define the generation data group as a generation data group base entry in the VSAM catalog. GIPEPROC finds information about each generation of the generation data group. For each generation, CATALOG builds a VSAM parameter list with information from the OS/VS catalog. CATALOG issues a UCATLG macro to define the generation as a non-VSAM entry in the VSAM catalog. The same error checks are performed as for a data set. When all the generations have been cataloged in the VSAM catalog, control returns to step 2 for the next name in the OS/VS catalog.
   - CVPEPROC compares the volumes assigned to the OS/VS CVOL entry with each *volser...* from the CVOLEQUATES parameter. If no match is found, the OS/VS CVOL entry is not put in the VSAM catalog, a message is written, and control returns to step 2 for the next name in the OS/VS catalog. If a match is found, CVPEPROC puts the *usercatalog* name associated with the matching *volser...* from the CVOLEQUATES parameter and the name from the OS/VS CVOL entry in a VSAM parameter list. CATALOG issues a UCATLG macro to define the OS/VS CVOL name as an alias to the *usercatalog* name in the VSAM master catalog. Control returns to step 2 for the next name in the OS/VS catalog.

Module: IDCCC01

Procedure: IDCCC01

3. IDCCC01 issues a UFPOOL macro to free all storage obtained for the CNVTCAT FSR. A message with LASTCC is written with a UPRINT macro. IDCCC01 also writes summary messages of the number of OS/VS catalog entries converted and the number of VSAM catalog entries updated. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.3 DEFINE FSR



Extended Description for Diagram 3.3 (Without VS2.03.807)

Module: IDCDE01

Procedure: IDCDE01

1. IDCDE01 issues a UGPOOL macro to obtain storage for a CTGPL, four CTGFVs, and a work area. The CTGPL, CTGFVs, and CTGFLs are used to pass information to VSAM catalog management. The CTGFVs are found through the CTGPL, and the CTGFLs are found through the CTGFVs. Refer to *OS/VS2 Catalog Management* for more information on the CTGPL, CTGFV, and CTGFL. Refer to the "Diagnostic Aids" chapter for an illustration of the DEFINE FSR work area. The characters CATPLIST precede the CTGPL. A call is made to IDCDE02 to establish addressability and obtain automatic storage for the IDCDE02 module and the common service routines. If a *catname* is supplied with a CATALOG parameter IDCDE01 puts the address of the *catname* and the *password* in the CTGPL.
2. IDCDE01 determines the type of DEFINE by testing for the following keywords: ALIAS, CLUSTER, GENERATIONDATAGROUP, MASTERCATALOG, NONVSAM, PAGESPACE, SPACE, USERCATALOG, ALTERNATE INDEX, PATH. The types of DEFINE are shown in detail in the following diagrams:

   ALIAS see Diagram 3.3.1
   CLUSTER see Diagram 3.3.2
   GENERATIONDATAGROUP see Diagram 3.3.3
   MASTERCATALOG see Diagram 3.3.4
   NONVSAM see Diagram 3.3.5
   PAGESPACE see Diagram 3.3.6
   SPACE see Diagram 3.3.7
   USERCATALOG see Diagram 3.3.4
   ALTERNATEINDEX see Diagram 3.3.8
   PATH see Diagram 3.3.9
3. IDCDE01 performs validity checking to ensure:

   KSDS, ESDS, RRDS, and AIX

   - Space parameters have been properly specified. (See Diagram 3.3.2, step 4, and Diagram 3.3.8, step 4)
   - Volumes have been specified in both DATA and INDEX FVTs.
   - If UNIQUE is specified, ensure CTGFVIND has been set and build null volume FVT. If CTGFVIND has not been set, call DALPROC to dynamically allocate the volume.
   - If an ESDS, KSDS or AIX has the REUSABLE attribute make sure it is not unique nor have KEYRANGES been specified. Verify that the REUSABLE attribute is the same at DATA and INDEX levels.
   - If the data AMDSB indicates an RRDS, insure that it does not also indicate spanned.
   - If average and maximum recordsize have not been specified, specify defaults: average for non-spanned=4089, average for spanned=4086, maximum for non-spanned=4089, maximum for spanned=32,600
   - If record size is greater than 32,761 (maximum CI size), ensure that it has the spanned attribute.
   - If KEYRANGES specified, ensure that key values do not exceed maximum key length.

   MASTERCATALOG/USERCATALOG

   - Space parameters have been properly specified (See Diagram 3.3.4, step 5).

   SPACE

   - Space parameters have been properly specified. (see Diagram 3.3.7, step 2)

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

4. IDCDE01 invokes VSAM catalog management by issuing a UCATLG macro. If the return code from catalog management indicates volume allocation is required, DALCPROC calls UALLOC to dynamically allocate the volumes; IDCDE01 issues the UCATLG macro again. If VSAM catalog management generates names for DATA and INDEX components, IDCDE01 prints the names with a UPRINT macro. IDCDE01 also prints volume allocation status. If the return code is non-zero, IDCDE01 calls UERROR to write a message that indicates an error occurred.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

5. If a PAGESPACE is being defined, IDCDE01 pre-formats the newly defined pagespace. Pre-formatting consists of opening and closing the pagespace. IDCDE01 builds an OPNAGL and puts in the name of the pagespace for the data set name. This causes the pagespace to be dynamically allocated when IDCDE01 issues a UOPEN macro. If the return code from UOPEN is non-zero, IDCDE01 terminates by giving control to step 5. If the return code is zero, IDCDE01 issues a UCLOSE to close the pagespace and deallocate it.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

6. If any volumes have been mounted, an ALLAGL exists for each request to mount a volume. DALCPROC repetitively issues UDEALLOC macros until all the volumes have been deallocated—until DALCPROC has used each ALLAGL as a parameter with a UDEALLOC macro. A message with LASTCC is written with a UPRINT macro. IDCDE01 issues a UFPOOL to free all the storage obtained for the DEFINE FSR. Control goes to Executive Controlled Termination, Diagram 4.0.

Module: IDCDE01, IDCDE02

Procedure: FREESTG

7. IDCDE01 calls IDCDE02 at its FREESTG entry point to deallocate automatic storage for IDCDE02. IDCDE01 then invokes the UEPIL macro to free any remaining automatic storage. Control goes to Executive Controlled Termination, Diagram 4.0.

Extended Description for Diagram 3.3

With VS2.03.807

Module: IDCDE01, IDCDE02

Procedure: IDCDE01, IDCDE02

1. IDCDE01 issues a UGPOOL macro to obtain storage for a CTGPL, four CTGFVs, and a work area. The CTGPL, CTGFVs, and CTGFLs are used to pass information to VSAM catalog management. The CTGFVs are found through the CTGPL, and the CTGFLs are found through the CTGFVs. Refer to *OS/VS2 Catalog Management* for more information on the CTGPL, CTGFV, and CTGFL. Refer to the "Diagnostic Aids" chapter for an illustration of the DEFINE FSR work area. The characters CATPLIST precede the CTGPL. A call is made to IDCDE02 to establish addressability and obtain automatic storage for the IDCDE02 module and the common service routines. If a *catname* is supplied with a CATALOG parameter IDCDE01 puts the address of the *catname* and the *password* in the CTGPL. IDCDE02 to format the catalog parameter lists.

Module: IDCDE03

Procedure: IDCDE03

2. IDCDE03 determines the type of DEFINE by testing for the following keywords: ALIAS, CLUSTER, GENERATIONDATAGROUP, MASTERCATALOG, NONVSAM, PAGESPACE, SPACE, USERCATALOG, ALTERNATE INDEX, PATH. The types of DEFINE are shown in detail in the following diagrams:

   ALIAS see Diagram 3.3.1
   CLUSTER see Diagram 3.3.2
   GENERATIONDATAGROUP see Diagram 3.3.3
   MASTERCATALOG see Diagram 3.3.4
   NONVSAM see Diagram 3.3.5
   PAGESPACE see Diagram 3.3.6
   SPACE see Diagram 3.3.7
   USERCATALOG see Diagram 3.3.4
   ALTERNATEINDEX see Diagram 3.3.8
   PATH see Diagram 3.3.9

   IDCDE03 then returns control to IDCDE01.

Module: IDCDE01

Procedure: IDCDE01, INTGCHK, DALCPROC

3. IDCDE01 calls INTGCHK to perform validity checking to insure:

KSDS, ESDS, RRDS, and AIX

- Space parameters have been properly specified.
- Volumes have been specified in both DATA and INDEX FVTs.
- If UNIQUE is specified, ensure CTGFVIND has been set and build null volume FVT. If CTGFVIND has not been set, call DALCPROC to dynamically allocate the volume.
- If an ESDS, KSDS, or AIX has the REUSABLE attribute make sure it is not unique nor have KEYRANGES been specified. Verify that the REUSABLE attribute is the same at DATA and INDEX levels.
- If the data AMDSB indicates an RRDS, ensure that it does not also indicate spanned.
- If average and maximum recordsize have not been specified, specify defaults: average for non-spanned=4089, average for spanned=4086, maximum for non-spanned=4089, maximum for spanned=32,600
- If record size is greater than 32,761 (maximum CI size), ensure that it has the spanned attribute.
- If KEYRANGES specified, ensure that key values do not exceed maximum key length.

For a VSAM cluster, two or three CTGFVs have been built—for cluster information, data information, and index information. If a VOLUME CTGFV has been built, it does not contain any information. The following table shows the possible places where a SPACPARM CTGFL may have been built and the action INTGCHK takes.

For an INDEXED data set:

SPACPARM CTGFL

SPACPARM CTGFL

Cluster	Data	Index	Action
x	x	x	This is an error; IDCDE01 terminates the DEFINE.
x	x		This is an error; IDCDE01 terminates the DEFINE.
x		x	This is an error; IDCDE01 terminates the DEFINE.
	x	x	OK; no action.
x			OK; no action
	x		OK; no action
		x	This is an error; IDCDE01 terminates the DEFINE.
	none		This is an error; IDCDE01 terminates the DEFINE.

For a NONINDEXED data set or a PAGESPACE:

SPACPARM CTGFL

SPACPARM CTGFL

Cluster	Data	Action
x	x	This is an error; IDCDE01 terminates the DEFINE.
x		OK; no action.
	x	OK; no action.
none		This is an error; IDCDE01 terminates the DEFINE.

For a Unique cluster or PAGESPACE, CTGFVIND in the DATA and INDEX CTGFV is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volumes. The *dname* returned by UALLOC is put in CTGFVIND.

For an alternate index two or three CTGFVs have been built—one each for alternate index, data, and index information. If a VOLUME CTGFV has been built, it does not have any information in it because VSAM uses it for a work space. The following table shows the possible places where a SPACPARM CTGFL may have been built and the action INTGCHK takes.

(With VS2.03.807)

SPACPARM CTGFL

Alternate Index	Data	Index	Action
X	X	X	This is an error; IDCDE01 terminates the DEFINE.
X	X		This is an error; IDCDE01 terminates the DEFINE.
X		X	This is an error; IDCDE01 terminates the DEFINE.
	X	X	OK; no action.
X			OK; no action.
	X		OK; no action.
		X	This is an error; IDCDE01 terminates the DEFINE.
none	none	none	This is an error; IDCDE01 terminates the DEFINE.

Control goes to Diagram 3.3, step 4.

USERCATALOG

For USERCATALOG four CTGFVs have been built—one each for cluster information, data information, index information, and volume information. A SPACPARM CTGFL must be specified on the CTGFV for volume information. In addition, INTGCHK checks the other three CTGFVs for a SPACPARM CTGFV.

SPACPARM CTGFL

The following table shows the possible CTGFVs (in addition to the VOLUME CTGFV) where a SPACPARM CTGFL may have been built and the action INTGCHK takes:

SPACPARM CTGFL

Cluster	Data	Index	Action
x	x	x	IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
x	x		IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
x		x	This is an error; IDCDE01 terminates the DEFINE.
x			OK - no action.
none			This is an error; IDCDE01 terminates the DEFINE.

CTGFVIND on the VOLUME CTGFV is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volume. The *dname* returned by UALLOC is put in CTGFVIND. Control goes to Diagram 4.3, step 3. If an error occurs, IDCDE01 writes a message, and control goes to Diagram 3.3, step 6.

SPACE

- Space parameters have been properly specified.

For DEFINE SPACE only a VOLUME CTGFV is built. INTGCHK checks the VOLUME CTGFV to be sure a SPACPARM CTGFL is present. If the space is in units of records, the VOLUME CTGFV must contain the address of a LRECL CTGFL. INTGCHK checks for CTGFVIND the VOLUME CTGFV. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volumes. The *dname* returned by UALLOC is put in CTGFVIND. Control goes to Diagram 3.3, step 4.

Module: IDCDE01

Procedure: IDCDE01, DALPROC

4. IDCDE01 invokes VSAM catalog management by issuing a UCATLG macro. If the return code from catalog management indicates volume allocation is required, DALCPROC calls UALLOC to dynamically allocate the volumes; IDCDE01 issues the UCATLG again. If VSAM catalog management generates names for DATA and INDEX components, IDCDE01 prints the names with a UPRINT macro. IDCDE01 also prints volume allocation status. If the return code is non-zero, IDCDE01 calls UERROR to write a message that indicates an error occurred.

Module: IDCDE01

Procedure: IDCDE01

5. If a PAGESPACE is being defined, IDCDE01 pre-formats the newly defined pagespace. Pre-formatting consists of opening and closing the pagespace. IDCDE01 builds an OPNAGL and puts in the name of the pagespace for the data set name. This causes the pagespace to be dynamically allocated when IDCDE01 issues a UOPEN macro. If the return code from UOPEN is non-zero, IDCDE01 terminates by giving control to step 6. If the return code is zero, IDCDE01 issues a UCLOSE to close the pagespace and deallocate it.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

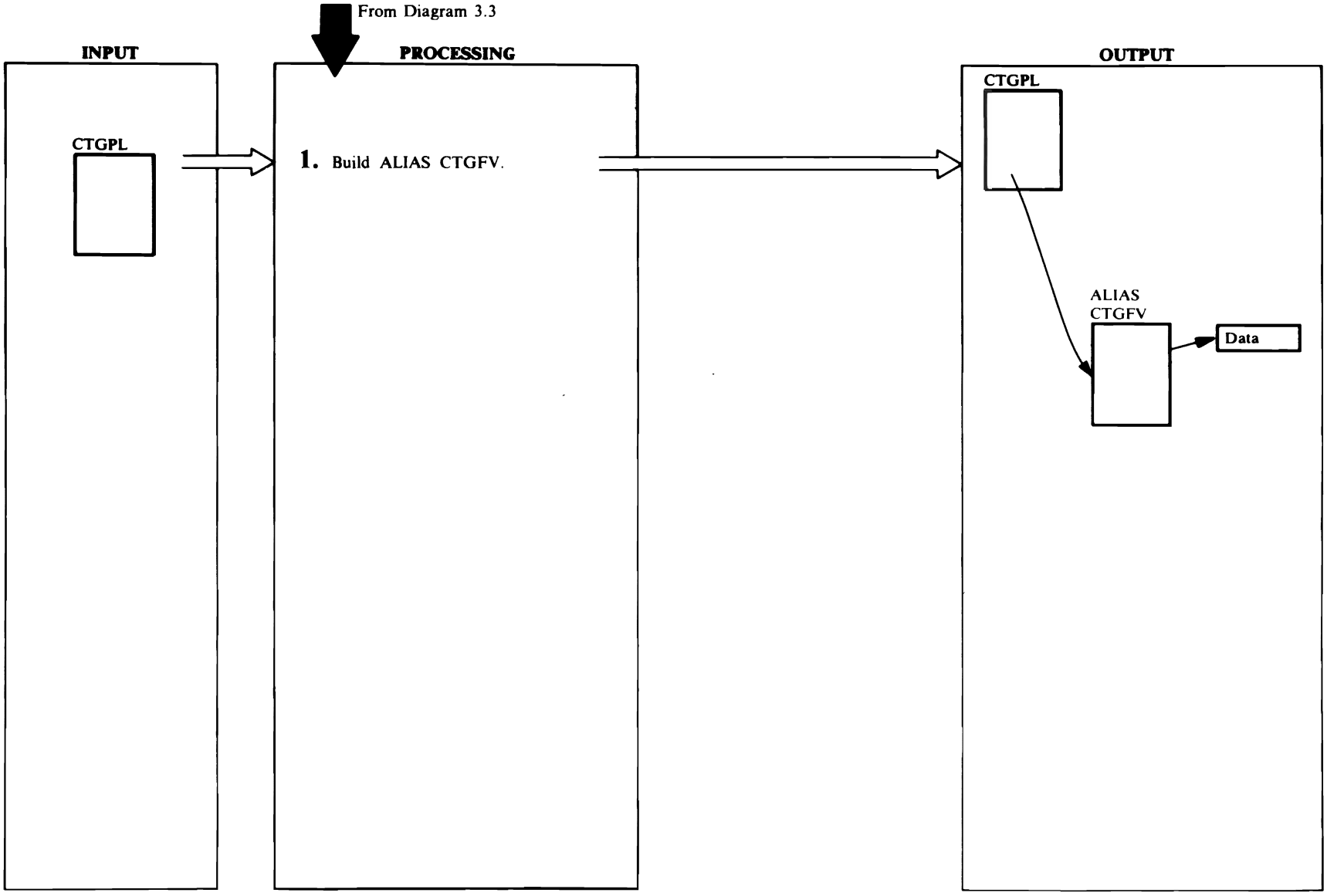
6. If any volumes have been mounted, an ALLAGL exists for each request to mount a volume. DALCPROC repetitively issues UDEALLOC macros until all the volumes have been deallocated—until DALCPROC has used each ALLAGL as a parameter with a UDEALLOC macro. A message with LASTCC is written with a UPRINT macro. IDCDE01 issues a UFPOOL to free all the dynamic storage obtained for the DEFINE FSR.

Module: IDCDE01, IDCDE02

Procedure: FREESTG

7. IDCDE01 calls IDCDE02 at its FREESTG entry point to deallocate automatic storage for IDCDE02. IDCDE01 then invokes the UEPIL macro to free any remaining automatic storage. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.3.1 DEFINE ALIAS



Extended Description for Diagram 3.3.1

Module: IDCDE01, IDCDE02 (**without VS2.03.807**)
IDCDE02, IDCDE03 (**with VS2.03.807**)

Procedure: NVSAMPRC, NAMEPROC

1. NVSAMPRC sets the identification of 'NVSAMFVT' in the eight bytes preceding the area that is usually used for a CLUSTER CTGFV. NVSAMPRC puts the address of the ALIAS CTGFV in the CTGFVT field of the CTGPL. An 'X' is set in the CTGFVTYP field of the CTGPL to indicate that an alias is being defined. NAMEPROC puts the address of *objectname* from NAME in the ALIAS CTGFV. NAMEPROC also puts *entryname/password* from RELATE in the ALIAS CTGFV. Control goes to step 3.

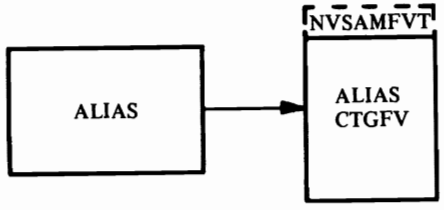
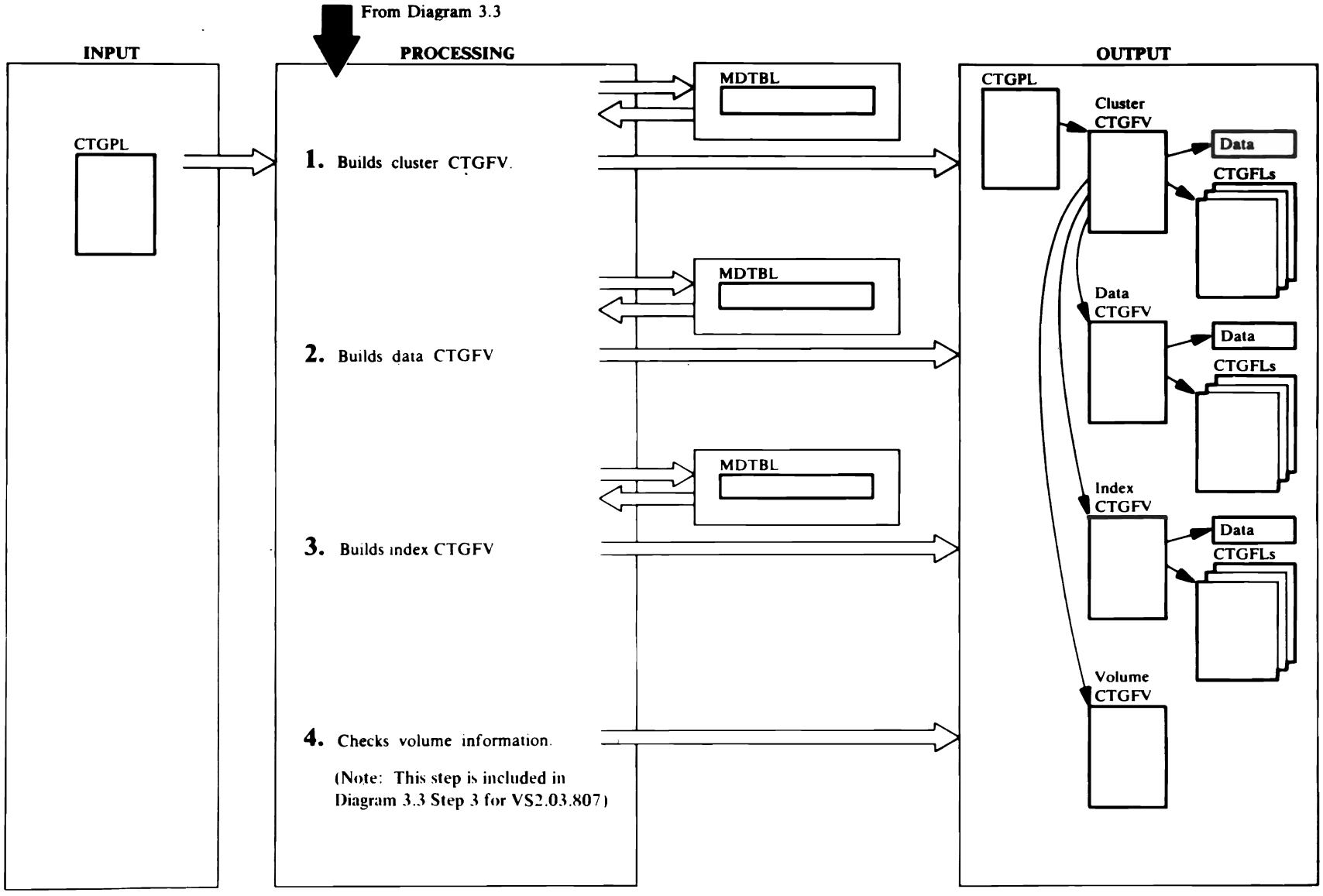


Diagram 3.3.2 Define FSR – Cluster



Extended Description for Diagram 3.3.2

Module: IDCDE01, IDCDEQ2 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: DSETPROC, NAMEPROC, MODELPRC, PROTPROC, ALLCPROC

1. In the DEFINE CLUSTER command, you specify information under three main keywords: CLUSTER, DATA, and INDEX. The DEFINE FSR builds a CTGFV to describe the cluster, data, and index components of the cluster as well as building a VOLUME CTGFV if UNIQUE is specified. Information specified under CLUSTER goes in the CLUSTER CTGFV; information under DATA goes in the DATA CTGFV; and information under INDEX goes in the INDEX CTGFV. Nothing is put in the VOLUME CTGFV. If not enough information is specified under DATA or INDEX to build the DATA or INDEX CTGFV, information from CLUSTER completes the DATA or INDEX CTGFV. If information is duplicated under DATA or INDEX and under CLUSTER—like WRITECHECK—information from DATA or INDEX overrides the information from CLUSTER in the DATA or INDEX CTGFV. The exception is space information from TRACKS, CYLINDERS, or RECORDS, which is never copied from CLUSTER to the DATA or INDEX CTGFV.

   If MODELs are specified, the information in the command overrides the information in a MODEL. A MODEL has one catalog entry to describe its cluster, one entry for its data, and one entry for its index, if the MODEL is a keyed sequence data set. The information in a MODEL's cluster catalog entry is used to build the CLUSTER CTGFV; information in a MODEL's data entry is used to build the DATA CTGFV; and information in the MODEL's index entry is used to build the INDEX CTGFV. The order of precedence when modeling is shown below where 1 takes the highest precedence:

   CLUSTER CTGFV

   1. CLUSTER parameters
   2. Cluster object of CLUSTER model

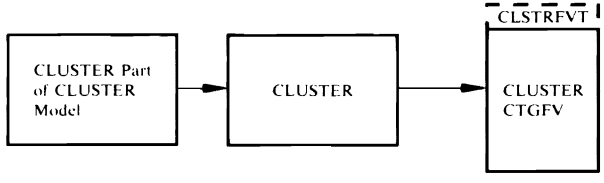
   DATA CTGFV

   1. DATA parameters
   2. DATA model
   3. CLUSTER parameters
   4. Data object of CLUSTER model

   INDEX CTGFV

   1. INDEX parameters
   2. INDEX model
   3. CLUSTER parameters
   4. Index object of CLUSTER model

   If MODEL is specified, MODELPRC issues a UCATLG to retrieve information from the modeled VSAM data set. The information from the cluster catalog entry of the modeled data set is put in a table, MDLTABL, and the Control Interval number for the data and index catalog entries of the modeled data set are saved. MDLTABL contains an address and the length of each field of information returned from the UCATLG. In building the CLUSTER CTGFV, information is obtained from MDLTABL and is then overlaid by information specified in the CLUSTER parameters. DSETPROC sets the identification of 'CLSTRFVT' in the eight bytes before the CLUSTER CTGFV. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC puts the address of *objectname* from NAME in the CLUSTER CTGFV. NAMEPROC builds a DSETEXDT CTGFL with the information from TO | FOR. PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds a OWNERID CTGFL with *ownerid* from OWNER. If *ownerid* is not specified and if Access Method Services has been invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO terminal user's identification for an *ownerid*. PROTPROC builds an OWNERID CTGFL with the TSO terminal user's identification. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. ALLCPROC initializes a pointer in the CLUSTER CTGFV to a work area for the catalog recovery volume serial number.



Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: DSETPROC, NAMEPROC, KEYPROC, MODLEPRC, ALLCPROC, PROTPROC

2. DSETPROC sets the identification of 'DATAFVTb' in the eight bytes preceding the DATA CTGFV. The DEFINE FSR builds the DATA CTGFV with the information specified in CLUSTER parameters. This information is then overlaid by the information specified in the DATA parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the DATA CTGFV is built with information specified in the CLUSTER parameters.

   If MODEL is specified under CLUSTER and MODEL is not specified under DATA, MODELPRC uses the saved Control Interval number for the modeled cluster's data catalog entry. The information from the data catalog entry of the modeled cluster is put in MDLTABL. The DATA CTGFV is built with information from MDLTABL and is then overlaid by the information specified in CLUSTER parameters. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC builds an EXCPEXIT CTGFL with exception exit information. KEYPROC builds a AMDSBCAT CTGFL, and ALLCPROC builds a DSATTR CTGFL, but no information is put in them yet.

   KEYPROC puts the *length* and *position* from KEYS in the AMDSBCAT CTGFL. If no key values are specified, KEYPROC sets up default values. KEYPROC also sets indication in AMDSB if SPANNED has been specified. If NUMBERED has been specified, KEYPROC sets AMDRRDS in the AMDSB field. KEYPROC also puts the address of *((lowkey) (highkey)...)* from KEYRANGES in the DATA CTGFV. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. ALLCPROC also builds a BUFSIZE CTGFL with information from BUFFERSPACE. The following is put in the AMDSBCAT CTGFL:

   ORDERED | UNORDERED
   cipercent and *capercent* from FREESPACE
   size from CONTROLINTERVALSIZE
   WRITECHECK | NOWRITECHECK

maximum from RECORDSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT
length and position from KEYS

PROTPROC puts ERASE | NOERASE, REUSE | NOREUSE, and *crossregion crosssystem* from SHAREOPTIONS in the DSATTR CTGFL.

Protection information is obtained only from the MODEL via MDLTABL in order to provide different protection for CLUSTER and DATA. PROTPROC builds a PASSWALL CTGFL with protection information from the MODEL and builds an OWNERID CTGFL with owner information from the MODEL.

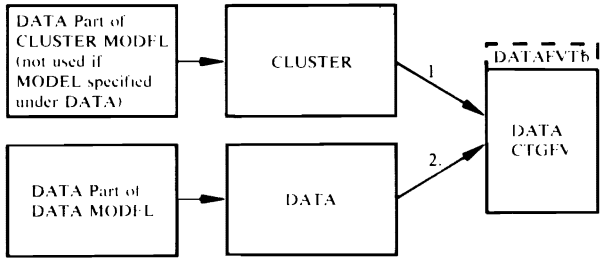
On the second pass, called the explicit pass, if MODEL is not specified under DATA, the information specified in the DATA parameters overlays the information placed in the DATA CTGFV on the implicit pass.

If MODEL is specified under DATA, MODELPRC issues a UCATLG to get information from the data catalog entry of the modeled data set. The information from the data catalog entry of the modeled data set is put in MDLTABL. The information in MDLTABL overlays the information placed in the DATA CTGFV on the implicit pass. Finally, the information in the DATA CTGFV is overlaid with the information specified in the DATA parameters.

NAMEPROC puts the address of *objectname* from NAME in the DATA CTGFV. Using a pointer to the name of the EXCEPTIONEXIT routine, NAMEPROC builds and initializes the EXCPEXIT FPL and references it in the FVT field CTGFVEXT. KEYPROC sets the AMDSPAN flag of AMDATTR to indicate the SPANNED | NONSPANNED option. KEYPROC puts *length* and *position* from KEYS in the AMDSBCAT CTGFL. KEYPROC puts the address of *((lowkey) (highkey)...)* list from KEYRANGES in the DATA CTGFV. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. Note: the volume serial list is not merged with any other volume serial list. ALLCPROC also builds or modifies the SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS; the LRECL CTGFL with *average* from RECORDSIZE; and the BUFSIZE CTGFL with *size* from BUFFERSPACE. PROTPROC builds or modifies the PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds or modifies the OWNERID CTGFL with *ownerid* from OWNER. If *ownerid* is not specified and if Access Method Services has been invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO terminal user's identification for an *ownerid*. PROTPROC builds an OWNERID CTGFL with the TSO terminal user's identification. The following is put in the the AMDSBCAT CTGFL:

ORDERED | UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK | NOWRITECHECK
maximum from RECORDSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT
length and position from KEYS

UNIQUE | SUBALLOCATION and SPEED | RECOVERY are put in the DSATTR CTGFL. ERASE | NOERASE, REUSE | NOREUSE, and *crossregion crosssystem* from SHAREOPTIONS are put in the DSATTR CTGFL.



Module: IDCDE01, IDCDE02 (**without VS2.03.807**) IDCDE02, IDCDE03 (**with VS2.03.807**)

Procedure: DSETPROC, NAMEPROC, KEYPROC, ALLCPROC, MODELPROC, IXOPPROC, PROTPROC

3. An INDEX CTGFV is built if any of the following are true:

INDEXED is specified
NONINDEXED and NUMBERED is not specified
The MODEL under CLUSTER is an indexed data set

In the listings, an *indexed* data set is called a KSDS for Key Sequence Data Set. A *non-indexed* data set is called an ESDS for Entry Sequence Data Set and a *numbered* data set is called an RRDS for Relative Record Data Set.

DSETPROC sets the identification of 'INDEXFVT' in the eight bytes preceding the INDEX CTGFV. The DEFINE FSR builds the INDEX CTGFV with the information specified in the CLUSTER parameters, which is overlaid by the information specified in the INDEX parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the INDEX CTGFV is built with information specified in CLUSTER parameters.

If MODEL is specified under CLUSTER and MODEL is not specified under INDEX, MODELPRC uses the saved Control Interval number for the modeled cluster's index catalog entry. The information from the index catalog entry of the modeled cluster is put in MDLTABL. The INDEX CTGFV is built with information from MDLTABL and is then overlaid by the information specified in the CLUSTER parameters.

NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC also puts the address of *objectname* from NAME in the INDEX CTGFV. Using a pointer to the name of the EXCEPTIONEXIT routine, NAMEPROC builds and initializes the EXCPEXIT FPL and references it in the FVT field CTGFVEXT. KEYPROC builds a AMDSBCAT CTGFL, and ALLCPROC builds a DSATTR CTGFL, but no information is put in them yet. IXOPPROC puts REPLICATE | NOREPLICATE and IMBED | NOIMBED in the AMDSBCAT CTGFL. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the INDEX CTGFV. ALLCPROC also builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
WRITECHECK | NOWRITECHECK
size from CONTROLINTERVALSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT

UNIQUE | SUBALLOCATION and REUSE | NOREUSE is put in the DSATTR CTGFL.

Record size is not indicated because it is always fixed length for the index of a VSAM data set.

Protection information is obtained only from the MODEL via MDLTABL in order to provide different protection at the CLUSTER and INDEX. PROTPROC builds a PASSWALL CTGFL with protection information from the MODEL as well as a OWNERID CTGFL with owner information from the MODEL. PROTPROC sets the appropriate bit of the ATTR1 field of the DSATTR field to indicate REUSE | NOREUSE.

On the second pass, called the explicit pass, if MODEL is not specified under INDEX, the information specified in the INDEX parameters overlays the information placed in the INDEX CTGFV on the implicit pass.

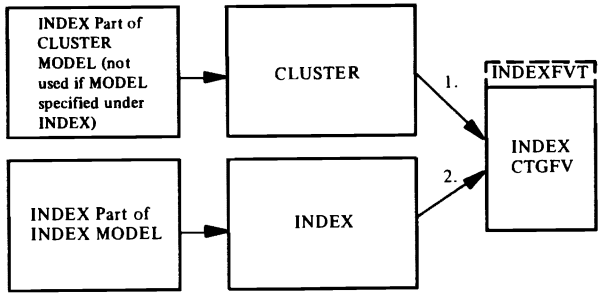
If MODEL is specified under INDEX, MODELPRC issues a UCATLG to get information from the index catalog entry of the modeled data set. The information from the index entry of the modeled data set is put in MDLTABL. The information in MDLTABL overlays the information placed in the INDEX CTGFV on the implicit pass. Finally, the information in the INDEX CTGFV is overlaid with the information specified in the INDEX parameters.

NAMEPROC puts the address of *objectname* from NAME in the INDEX CTGFV. Using a pointer to the name of the EXCEPTIONEXIT routine, NAMEPROC builds and initializes the EXCPEXIT FPL if specified under INDEX. IXOPPROC puts REPLICATE | NOREPLICATE and IMBED | NOIMBED in the AMDSBCAT CTGFL. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the INDEX CTGFV. ALLCPROC also builds or modifies the SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. PROTPROC builds or modifies the PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds or modifies the OWNERID CTGFL with *ownerid* from OWNERID. If *ownerid* is not specified and if Access Method Services has been invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO terminal user's identification for an *ownerid*. PROTPROC builds an OWNERID CTGFL with the TSO terminal user's identification. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
WRITECHECK | NOWRITECHECK
size from CONTROLINTERVALSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT

The following is put in the DSATTR CTGFL:

UNIQUE | SUBALLOCATION
ERASE | NOERASE
REUSE | NOREUSE
crossregion crosssystem from SHAREOPTIONS



Note: If VS2.03.807 is installed, the following step, (step 4), is described in Diagram 3.3, step 3.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

4. For a VSAM data set, two or three CTGFVs have been built— for cluster information, data information, and index information. If a VOLUME CTGFV has been built, it does not contain any information. The following table shows the possible places where a SPACPARM CTGFL, may have been built and the action IDCDE01 takes.

   For an INDEXED data set:

   SPACPARM CTGFL

SPACPARM CTGFL

Cluster	Data	Index	Action
x	x	x	This is an error; IDCDE01 terminates the DEFINE.
x	x		This is an error; IDCDE01 terminates the DEFINE.
x		x	This is an error; IDCDE01 terminates the DEFINE.
	x	x	OK; no action.
x			OK; no action
	x		OK; no action
		x	This is an error; IDCDE01 terminates the DEFINE.
none			This is an error; IDCDE01 terminates the DEFINE.

For a NONINDEXED data set:

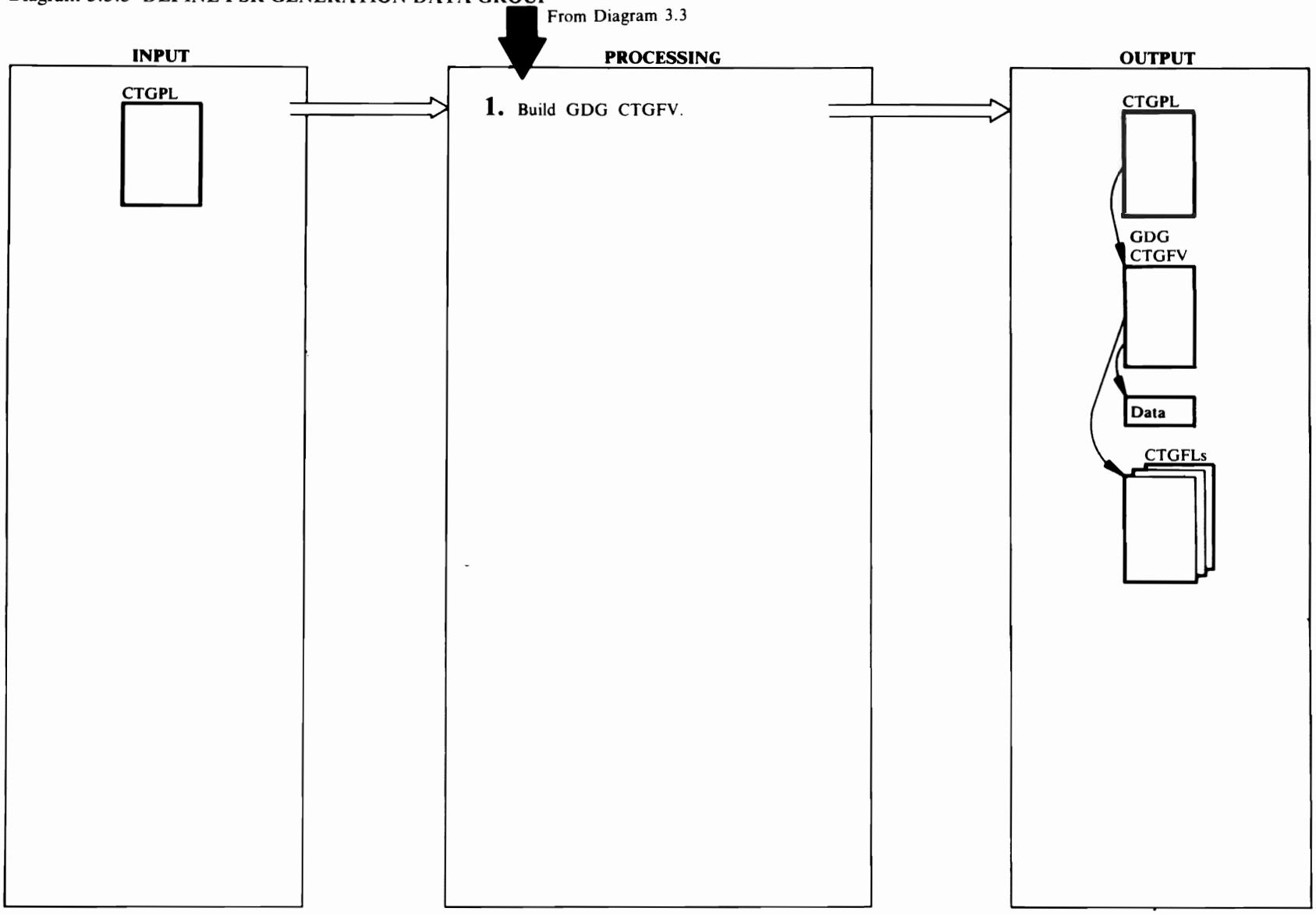
SPACPARM CTGFL

SPACPARM CTGFL

Cluster	Data	Action
x	x	This is an error; IDCDE01 terminates the DEFINE.
x		OK; no action.
	x	OK; no action.
none		This is an error; IDCDE01 terminates the DEFINE.

For a UNIQUE data component, CTGFVIND in the DATA CTGFV is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volumes. The *dname* returned by UALLOC is put in CTGFVIND. This same processing is performed for a UNIQUE index component. If either the data or index component is UNIQUE, IDCDE01 builds a null volume FVT. Control goes to Diagram 3.3, step 3.

Diagram 3.3.3 DEFINE FSR-GENERATION DATA GROUP



Extended Description for Diagram 3.3.3

Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: NVSAMPRC, NAMEPROC, PROTPROC

1. NVSAMPRC sets the identification of 'NVSAMFVT' in the eight bytes preceding the area that is usually used for a CLUSTER CTGFV. NVSAMPRC puts the address of the GDG CTGFV in the CTGFVT field of the CTGPL. NAMEPROC puts the address of *objectname* from NAME in the GDG CTGFV after testing to ensure that the name is less than 35 characters long. A 'B' is set in the CTGFVTYP field of the CTGPL to indicate that a generation data group base is being defined. NAMEPROC puts EMPTY | NOEMPTY, and SCRATCH | NOSCRATCH in the GDGATTR CTGFL and *limit* from LIMIT in the GDGLIMIT CTGFL. PROTPROC builds an OWNERID CTGFL. If Access Method Services is invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO user's identification, which is used as OWNERID when OWNER isn't specified. NAMEPROC issues a UTIME macro to get the creation date, which it puts in a DSETCRDT CTGFL. Information from the TO | FOR parameters goes in the DSETEXDT CTGFL. Control goes to Diagram 3.3, step 3.

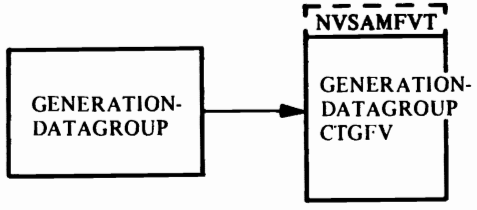
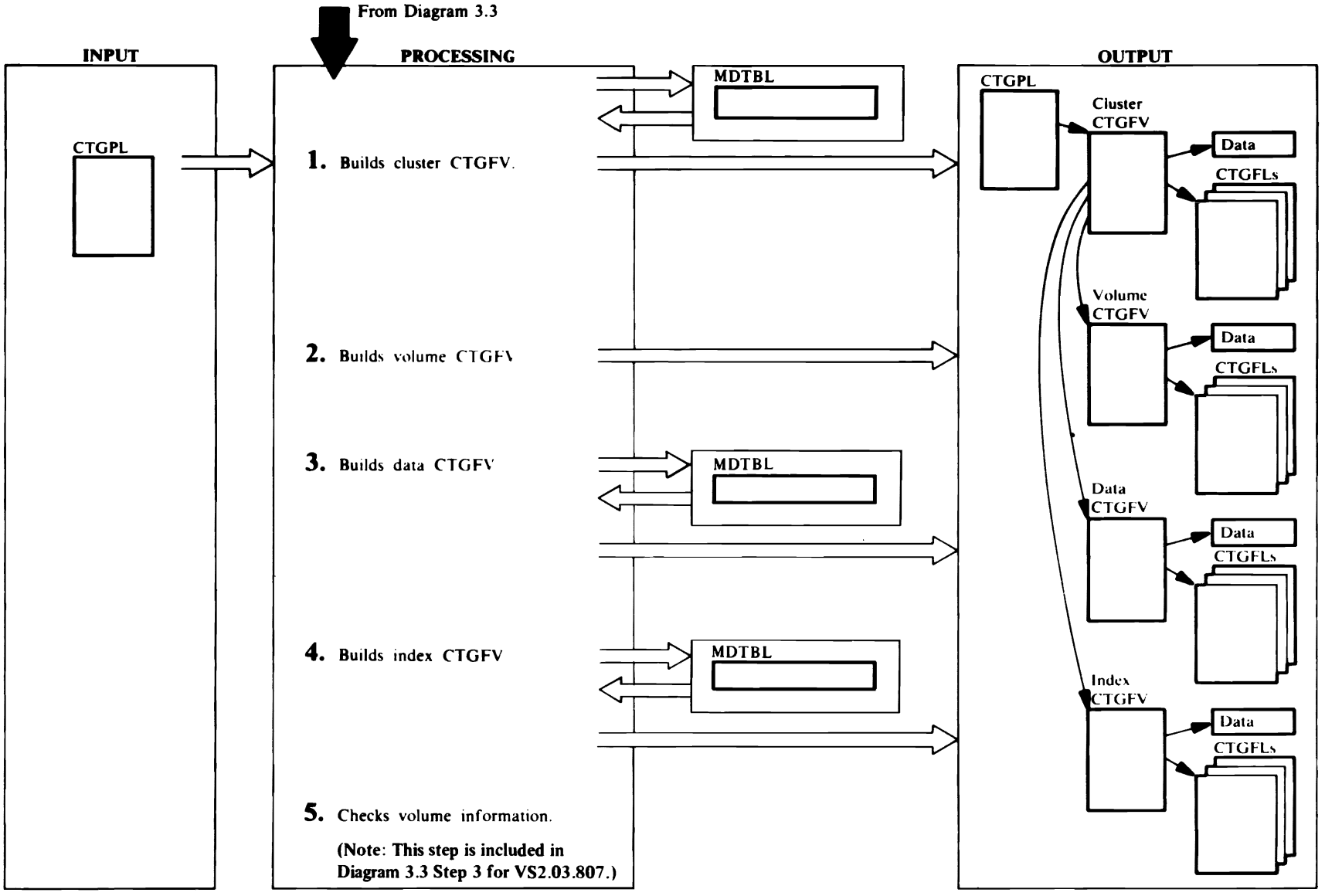


Diagram 3.3.4 DEFINE FSR – MASTERCATALOG/USERCATALOG



Extended Description for Diagram 3.3.4

Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: CTLGPROC, ALLCPROC, NAMEPROC, MODELPRC, PROTPROC

1. Because the master catalog for a VS2 system is created at system-generation time and because only one master catalog is allowed on a system, the DEFINE MASTERCATALOG command results in a usercatalog being created. The mastercatalog-usercatalog descriptions have been merged but it should be remembered that the MODEL information does not apply for DEFINE MASTERCATALOG.

   In the DEFINE USERCATALOG command, you specify information under three main keywords: USERCATALOG, DATA, and INDEX. The DEFINE FSR builds a CTGFV to describe the cluster, data, and index components of the user catalog as well as building a VOLUME CTGFV. Information specified under USERCATALOG goes in the CLUSTER and VOLUME CTGFVs; information under DATA goes in the DATA CTGFV; and information under INDEX goes in the INDEX CTGFV. If not enough information is specified under DATA or INDEX to build the DATA or INDEX CTGFV, information from USERCATALOG completes the DATA or INDEX CTGFV. If information is duplicated under DATA or INDEX and under USERCATALOG—like WRITECHECK—information from DATA or INDEX overrides the information from USERCATALOG in the DATA or INDEX CTGFV. The exception is space information from TRACKS, CYLINDERS, or RECORDS. Space information is never copied from the cluster.

   If a MODEL is specified, the information in the command overrides the information in the MODEL. The MODEL has one catalog entry to describe its cluster, one entry for its data, and one entry for its index. The information in the MODEL's cluster catalog entry is used to build the CLUSTER CTGFV; information in the MODEL's data catalog entry is used to build the DATA CTGFV; and information in the MODEL's index entry is used to build the INDEX CTGFV. The order of precedence when modeling is shown below where 1 has the highest precedence:

   CLUSTER CTGFV

   1. USERCATALOG parameters
   2. Cluster object of model

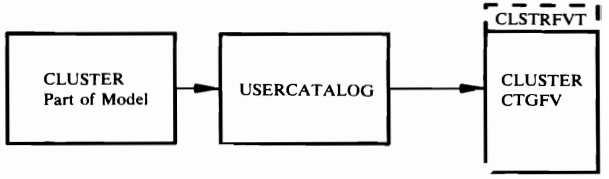
   DATA CTGFV

   1. DATA parameters
   2. USERCATALOG parameters
   3. Data object of model

   INDEX CTGFV

   1. INDEX parameters
   2. USERCATALOG parameters
   3. Index object of model

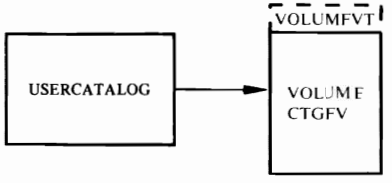
   CTLGPROC sets the identification of 'CLSTRFVT' in the eight bytes before the CLUSTER CTGFV. A U is put in the CTGTYPE field of the CTGPL to indicate that a user catalog is being defined (in the case of a mastercatalog CTGTYPE is set to M.) CTLGPROC puts the address of the *objectname* from NAME in the CLUSTER CTGFV. CTLGPROC checks for a MODEL keyword. If MODEL is specified, MODELPRC issues a UCATLG macro to retrieve information from the modeled user catalog. The information from the cluster catalog entry of the modeled user catalog is put in a table, MDLTABL, and the Control Interval number for the data and index entries of the modeled user catalog are saved. MDLTABL contains an address and the length of each field of information returned from the UCATLG. In building the CLUSTER CTGFV, information is obtained from MDLTABL and is then overlaid by the information specified in the USERCATALOG parameters. NAMEPROC builds a DSETEXDT CTGFL with the information from TO | FOR. PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds an OWNERID CTGFL with *ownerid* from OWNER. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, and RECORDS. ALLCPROC sets on the BIND bit. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. If *ownerid* is not supplied and Access Method Services has been invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO terminal user's identification for an *ownerid*. PROTPROC builds an OWNERID CTGFL with the TSO terminal user's identification.



Module: IDCDE01 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: CTLGPROC, ALLCPROC

2. The DEFINE FSR builds a VOLUME CTGFV with information specified under USERCATALOG. No information is taken from a MODEL for the VOLUME CTGFV. CTLGPROC sets the identification of 'VOLUMFVT' in the eight bytes preceding the VOLUME CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. ALLCPROC puts the address of *volser* from VOLUMES and the address of *dname* from FILE in the VOLUME CTGFV.



Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: CTLGPPROC, NAMEPROC, KEYPROC, ALLCPROC, MODELPRC

3. CTLGPROC sets the identification of 'DATAFVTb' in the eight bytes preceding the DATA CTGFV. The DEFINE FSR builds the DATA CTGFV with the information specified in USERCATALOG parameters. This information is then overlaid by the information specified in the DATA parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the DATA

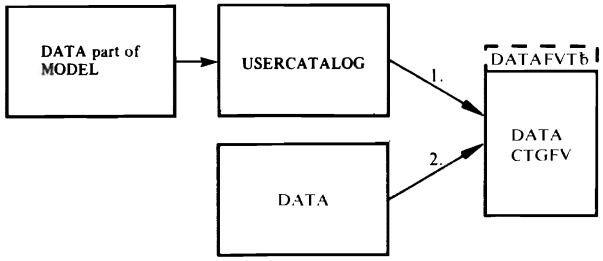
CTGFV is built with information specified in the USERCATALOG parameters.

If MODEL is specified, MODELPRC uses the saved Control Interval number for the modeled user catalog's data entry. The information from the data entry of the modeled user catalog is put in MDLTABL. The DATA CTGFV is built with information from MDLTABL and is then overlaid by the information specified in USERCATALOG parameters.

NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. KEYPROC builds an AMDSBCAT CTGFL, but no information is put in yet. ALLCPROC puts the address of the *volser* from VOLUME and the address of *dname* from FILE in the DATA CTGFV. WRITECHECK | NOWRITECHECK, BIND, and DESTAGEWAIT | NODESTAGEWAIT are put in the AMDSBCAT CTGFL. ALLCPROC builds a BUFSIZE CTGFL with information from BUFFERSPACE. ALLCPROC also builds a DSATTR CTGFL for data set attributes, and, in addition the Recoverable/Not Recoverable attribute is set in DSATTR.

On the second pass, called the explicit pass, the information in the DATA CTGFV from the implicit pass is overlaid by the information specified in the DATA parameters.

ALLCPROC builds a SPACPARM CTGFL for primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. If WRITECHECK | NOWRITECHECK or DESTAGEWAIT | NODESTAGEWAIT is specified under DATA, it is overridden in the AMDSBCAT CTGFL. If BUFFERSPACE is specified under DATA, ALLCPROC builds a BUFSIZE CTGFL or modifies the existing one. ALLCPROC initializes the Recoverable/Not Recoverable attribute in DSATTR.



Module: IDCDE01, IDCDE02 (**without VS2.03.807**)
IDCDE02, IDCDE03 (**with VS2.03.807**)

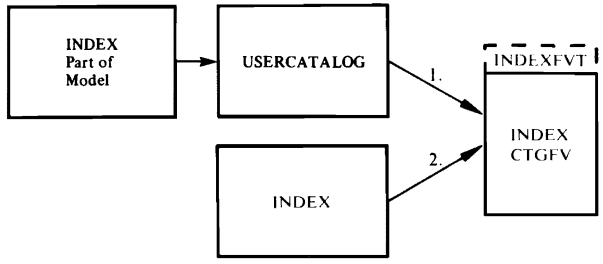
Procedure: CTLGPROC, NAMEPROC, KEYPROC, ALLCPROC, MODELPRC

4. CTLGPROC sets the identification of 'INDEXFVT' in the eight bytes preceding the INDEX CTGFV. The DEFINE FSR builds the INDEX CTGFV with the information specified in USERCATALOG parameters which is overlaid by the information specified in the INDEX parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the INDEX CTGFV is built with information specified in USERCATALOG parameters.

   If MODEL is specified, MODELPRC uses the saved Control Interval number for the modeled user catalog's index entry. The information from the index entry of the modeled user catalog is put in MDLTABL. The INDEX CTGFV is built with information from MDLTABL and then overlaid by the information specified in the USERCATALOG parameters.

   NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. KEYPROC builds a AMDSBCAT CTGFL, but no information is put in yet. ALLCPROC puts the address of the *volser* from VOLUME and the address of *dname* from FILE in the INDEX CTGFV.

   WRITECHECK | NOWRITECHECK, BIND, and DESTAGEWAIT | NODESTAGEWAIT are put in the AMDSBCAT CTGFL. ALLCPROC builds a DSATTR CTGFL for data set attributes, On the second pass, called the explicit pass, the information in the INDEX CTGFV from the implicit pass is overlaid by the information specified in the INDEX parameters. ALLCPROC builds a SPACPARM CTGFL for primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. WRITECHECK | NOWRITECHECK and DESTAGEWAIT | NODESTAGEWAIT are overridden in the AMDSBCAT CTGFL.



Note: If **VS2.03.807** is installed, this step, step 5, is described in Diagram 3.3, step 3.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

5. For USERCATALOG four CTGFVs have been built—one each for cluster information, data information, index information, and volume information. A SPACPARM CTGFL must be specified on the CTGFV for volume information. In addition, IDCDE01 checks the other three CTGFVs for a SPACPARM CTGFV.

   SPACPARM CTGFL

   The following table shows the possible CTGFVs (in addition to the VOLUME CTGFV) where a SPACPARM CTGFL may have been built and the action IDCDE01 takes:

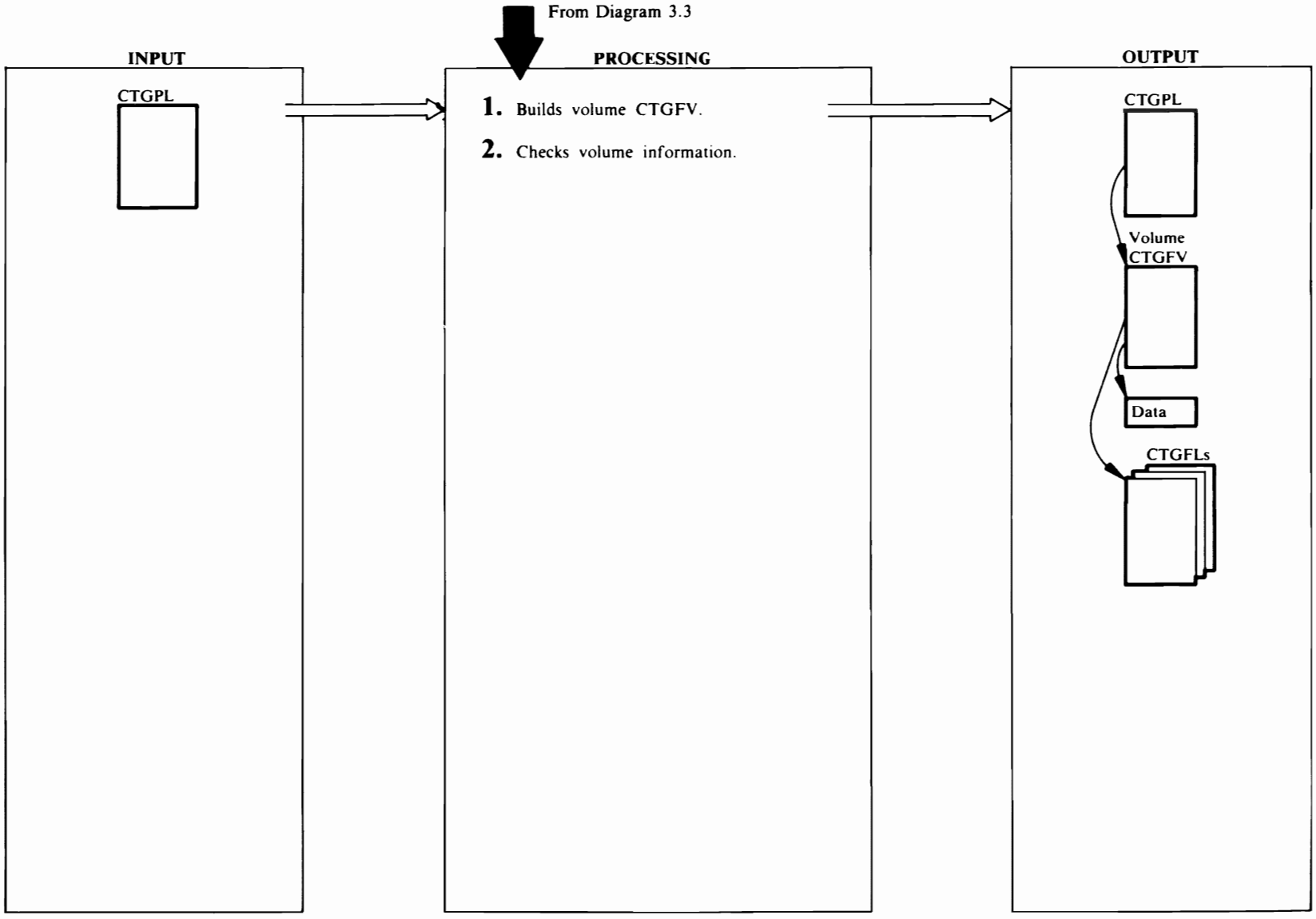
Note: Ignore this page if **VS2.03.807** is not installed.

SPACPARM CTGFL

Cluster	Data	Index	Action
x	x	x	IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
x	x		IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
x		x	This is an error; IDCDE01 terminates the DEFINE.
x			OK - no action.
none			This is an error; IDCDE01 terminates the DEFINE.

CTGFVIND on the VOLUME CTGFV is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volume. The *dname* returned by UALLOC is put in CTGFVIND. Control goes to Diagram 3.3, step 3. If an error occurs, IDCDE01 writes a message, and control goes to Diagram 3.3, step 6.

Diagram 3.3.5 DEFINE FSR – NONVSAM



Extended Description for Diagram 3.3.5

Module: IDCDE01, IDCDE02 (**without VS2.03.807**)
IDCDE02, IDCDE03 (**with VS2.03.807**)

Procedure: NVSAMPRC, ALLCPROC, PROTPROC, NAMEPROC

1. NVSAMPRC sets the identification of 'NVSAMFVT' in the eight bytes preceding the area that is usually used for a CLUSTER CTGFV. An 'A' is set in the CTGFVTYP field of the CTGPL to indicate that a non-VSAM data set is being defined. NVSAMPRC puts the address of the NONVSAM CTGFV in the CTGFVT field of the CTGPL. NAMEPROC puts the address of *objectname* from NAME in the NONVSAM CTGFV. ALLCPROC puts the address of *volser* from VOLUMES in the NONVSAM CTGFV. ALLCPROC builds a DEVTYPE CTGFL for information from DEVICETYPES. If FILESEQUENCENUMBERS is specified, ALLCPROC puts the address of *numbers* from FILESEQUENCENUMBERS in the NONVSAM CTGFV. PROTPROC builds an OWNERID CTGFL. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field. If Access Method Services is invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO user's identification, which is used as OWNERID when OWNER isn't specified. NAMEPROC issues a UTIME macro to get the creation date, which it puts in a DSETCRDT CTGFL. NAMEPROC uses the information from TO | FOR to build a DSETEXDT CTGFL. Control goes to Diagram 3.3, step 3.

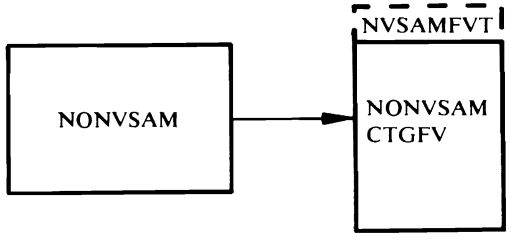
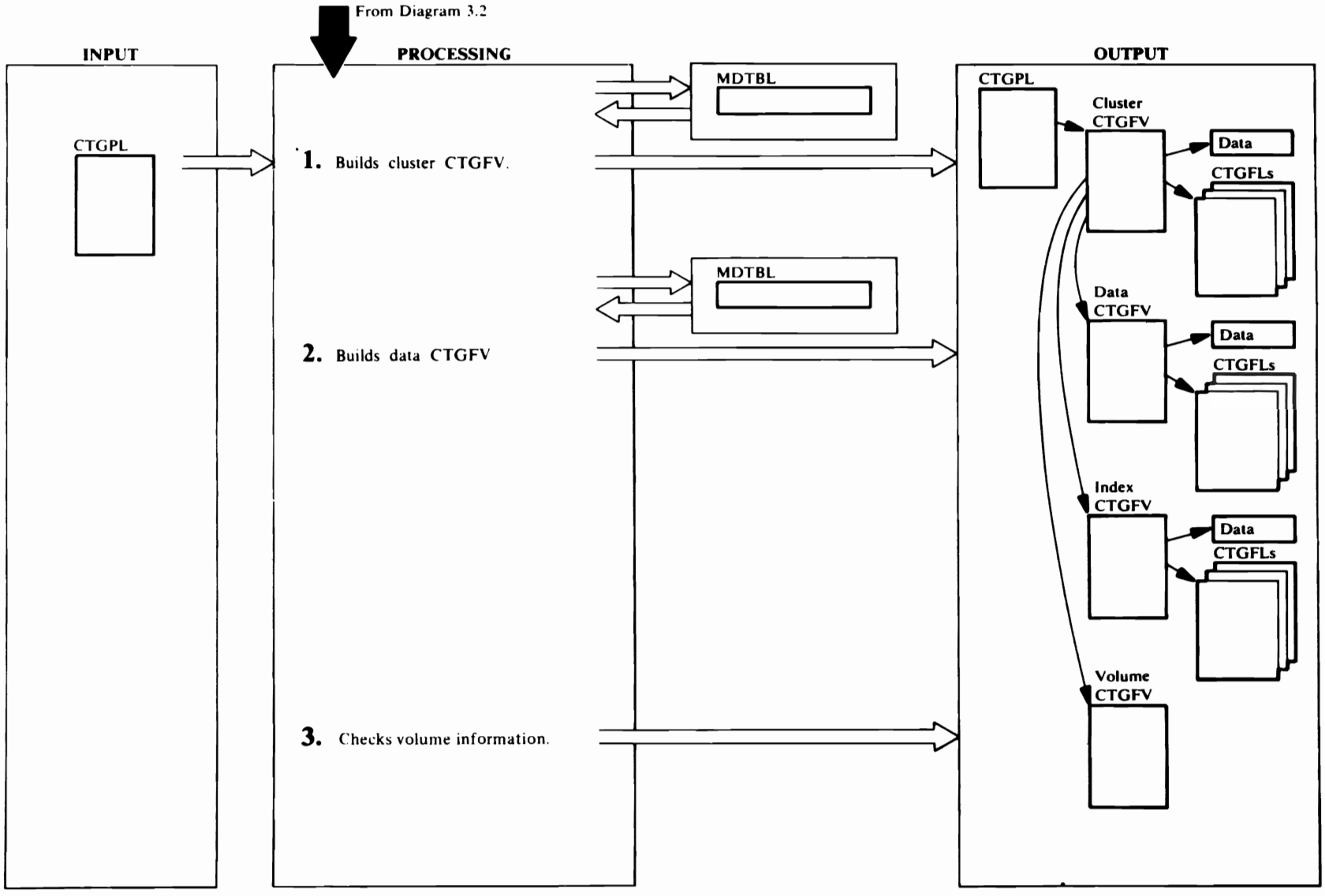


Diagram 3.3.6 DEFINE FSR – PAGESPACE



Extended Description for Diagram 3.3.6

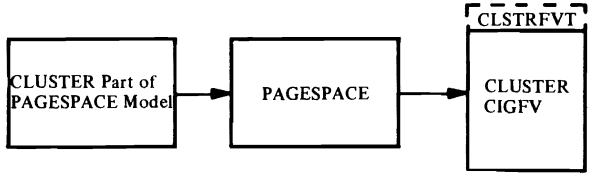
Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: DSETPROC, NAMEPROC, MODELPRC, PROTPROC

1. In the DEFINE PAGESPACE the DEFINE FSR builds CTGFVs for the cluster and a CTGFV for the data component as well as a VOLUME CTGFV if UNIQUE is specified. Information supplied under PAGESPACE is put in the CLUSTER CTGFV and the DATA CTGFV. The VOLUME CTGFV contains zeros.

   If a MODEL is specified, the information in the command overrides the information in the MODEL. The MODEL has one catalog entry to describe its cluster and one entry to describe its data. The information in the MODEL's cluster catalog entry is used to build the CLUSTER CTGFV, and information in the MODEL's data catalog entry is used to build the DATA CTGFV.

   DSETPROC sets the identification of CLSTRFVT in the eight bytes before the CLUSTER CTGFV. A 'P' is set in the CTGTYPE field in the CTGPL to indicate that a pagespace is being defined. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC puts the address of *objectname* from NAME in the CLUSTER CTGFV. DSETPROC checks for a MODEL keyword. If MODEL is specified, MODELPRC issues a UCATLG to retrieve information from the modeled pagespace. The information from the cluster catalog entry and the Control Interval number for the data entry of the modeled pagespace are saved. MDLTABL contains an address and the length of each field of information returned from the UCATLG. For the remainder of the CLUSTER CTGFV, first, the information is obtained from MDLTABL; then, the data from the command overrides information from MDLTABL if the information is specified in the command. NAMEPROC builds a DSETEXTDT CTGFL with the information from TO | FOR. PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds a OWNERID CTGFL with *ownerid* from OWNERID. If *ownerid* is not specified and Access Method Services has been invoked interactively with TSO, PROTPROC issues a UID macro to get the TSO terminal user's identification for an *ownerid*. PROTPROC builds an OWNERID CTGFL with the TSO terminal user's identification.



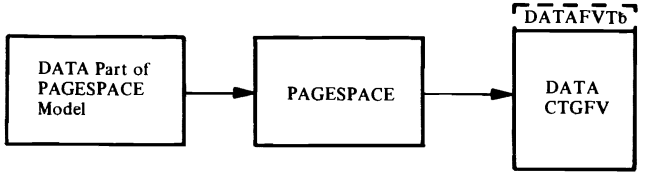
Module: IDCDE01, IDCDE02 **(without VS2.03.807)**
IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: DSETPROC, MODELPRC, NAMEPROC, KEYPROC, ALLCPROC

2. DSETPROC sets the identification of DATAFVTb in the eight bytes preceding the DATA CTGFV. If MODEL is specified in the command, MODELPRC uses the saved Control Interval number for the modeled pagespace's data catalog entry. The information from the data entry of the modeled pagespace is put in MDLTABL. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. For the remainder of this step, first, the information is obtained from MDLTABL; then, the data from the command overrides the information in MDLTABL if information is specified in the command. KEYPROC builds a AMDSBCAT CTGFL, and ALLCPROC builds a DSATTR CTGFL, but no information is put in them yet. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. Note: the volume serial list is not merged with any other volume serial list. ALLCPROC also builds the SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, and RECORDS.

   For users without VS2.03.807:
   UNIQUE | SUBALLOCATION is put in the DSATTR CTGFL. ALLCPROC always turns on track overflow in the DSATTR CTGFL. There is no way to specify track overflow in the command, but it is always turned on for a pagespace.

   For Users with VS2.03.807:
   UNIQUE | SUBALLOCATION is put in the DSATTR CTGFL. If SUBALLOCATION has not been specified, the UNIQUE bit will be turned on. ALLCPROC turns on track overflow in the DSATTR CTGFL, unless SWAP has been specified. Track overflow is always turned on for a NOSWAP pagespace.



   Note: If VS2.03.807 is installed, this step, step 3, is described in Diagram 3.3, step 3.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

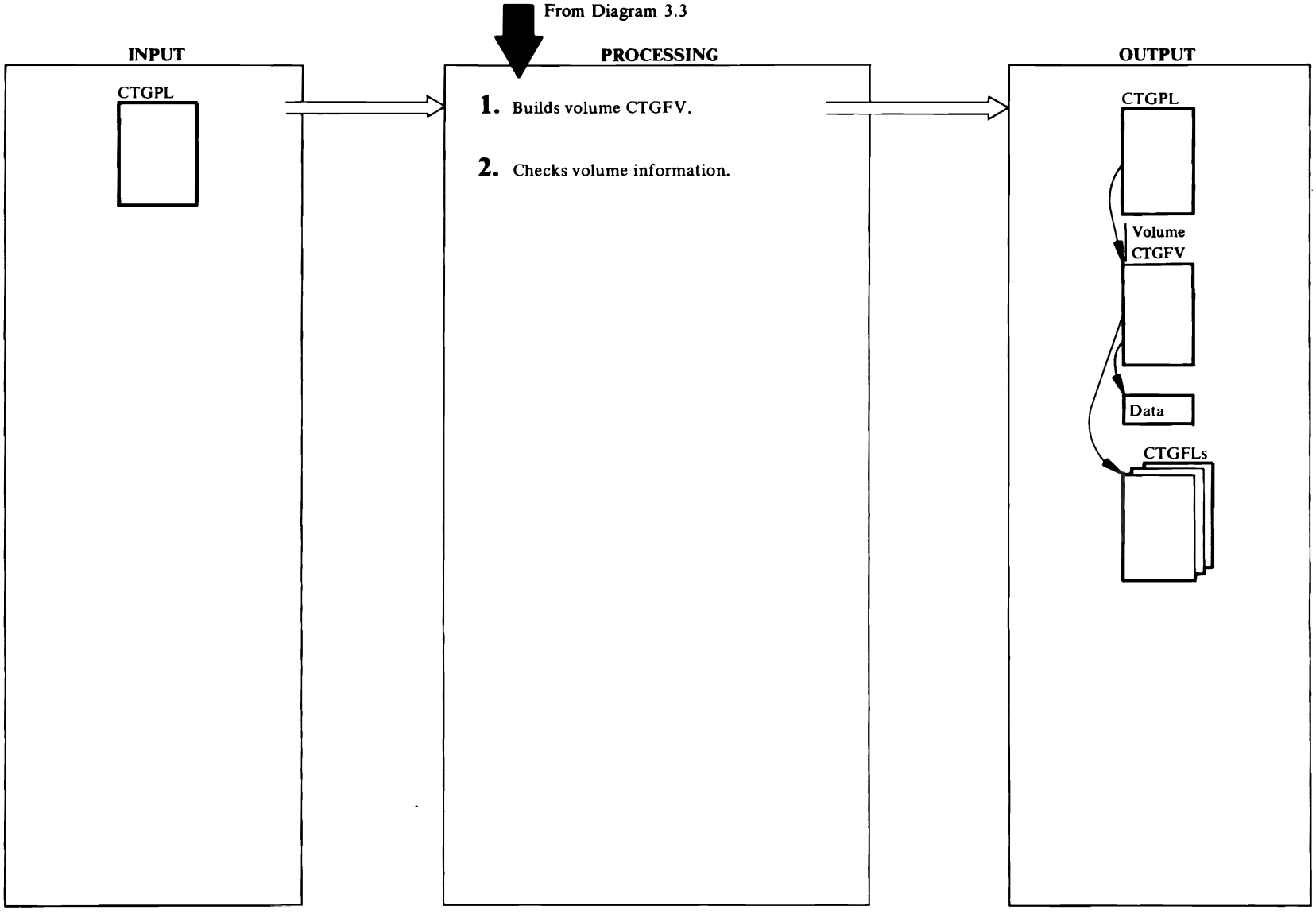
3. IDCDE01 performs validity checking as shown in the following table:

   SPACEPARM CTGFL

Cluster	Data	Action
X	X	This is an error; IDCDE01 terminates the DEFINE.
X		O.K. - no action
	X	O.K. - no action

   For a UNIQUE pagespace, the CTGFVIND is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volume. The *dname* returned by UALLOC is placed in CTGFVIND. IDCDE01 builds a null VOLUME FVT for a UNIQUE pagespace.

Diagram 3.3.7 DEFINE FSR – Space

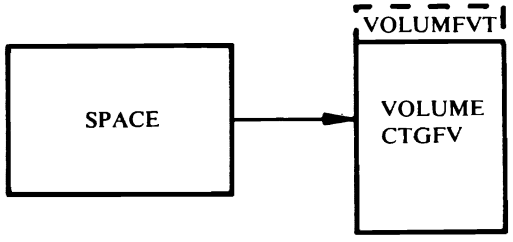


Extended Description for Diagram 3.3.7

Module: IDCDE01, IDCDE02 (**without VS2.03.807**)
IDCDE02, IDCDE03 (**with VS2.03.807**)

Procedure: DSPACPRC, ALLCPROC

1. DSPACPRC sets the identification of 'VOLUMFVT' in the eight bytes preceding the VOLUME CTGFV. The address of the VOLUME CTGFV is put in the CTGPL in the field named CTGFVT because the VOLUME CTGFV is the only CTGFV for a DEFINE SPACE. ALLCPROC puts the address of the *volser* from VOLUMES and the address of *dname* from FILE in the VOLUME CTGFV. ALLCPROC builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, RECORDS. If RECORDS is specified, ALLCPROC builds a LRECL CTGFL with information from RECORDSIZE.



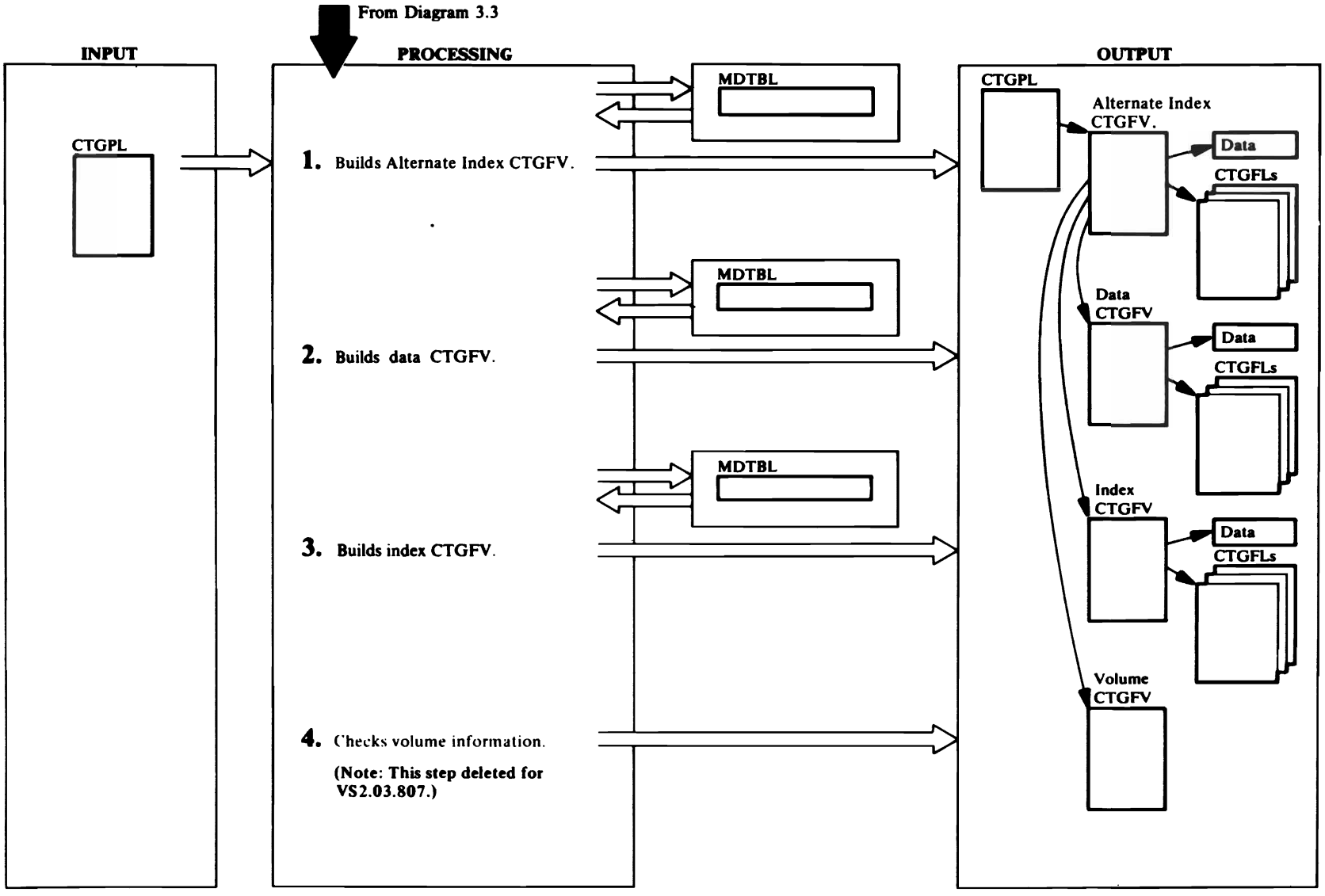
Note: For users with VS2.03.807, this step is described in Diagram 3.3, step 3.

Module: IDCDE01

Procedure: IDCDE01, DALCPROC

2. For DEFINE SPACE, only a VOLUME CTGFV is built. IDCDE01 checks the VOLUME CTGFV to be sure a SPACPARM CTGFL is present. If the space is in units of records, the VOLUME CTGFV must contain the address of a LRECL CTGFL. IDCDE01 checks for a CTGFVIND in the VOLUME CTGFV. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volumes. The *dname* returned by UALLOC is put in CTGFVIND. Control goes to Diagram 3.3, step 4.

Diagram 3.3.8 DEFINE FSR – DEFINE ALTERNATE INDEX



Extended Description for Diagram 3.3.8

Module: IDCDE01, IDCDE02, **(without VS2.03.807)** IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: AIXPROC, NAMEPROC, MODELPRC, PROTPROC, ALLCPROC

1. In the DEFINE AIX command, you specify information under three main keywords: AIX, DATA, and INDEX. The DEFINE FSR builds a CTGFV to describe the alternate index, data, and index components of the alternate index as well as building a VOLUME CTGFV if UNIQUE is specified. Information specified under ALTERNATEINDEX goes in the ALTERNATEINDEX CTGFV; information under DATA goes in the DATA CTGFV; and information under INDEX goes in the INDEX CTGFV. Nothing is put in the VOLUME CTGFV. If not enough information is specified under DATA or INDEX to build the DATA or INDEX CTGFV, information from ALTERNATEINDEX completes the DATA or INDEX CTGFV. If information is duplicated under DATA or INDEX and under ALTERNATEINDEX—like WRITECHECK—information from DATA or INDEX overrides the information from ALTERNATEINDEX in the DATA or INDEX CTGFV. The exception is space information from TRACKS, CYLINDERS, or RECORDS. Space information is never copied from ALTERNATEINDEX.

   If MODELs are specified, the information in the command overrides the information in a MODEL. A MODEL has one catalog entry to describe its alternate index, one entry for its data, and one entry for its index. The information in a MODEL's alternate index catalog entry is used to build the ALTERNATEINDEX CTGFV; information in a MODELS's data entry is used to build the DATA CTGFV; and information in the MODEL's index entry is used to build the INDEX CTGFV. The order of precedence when modeling is shown below where 1 takes the highest precedence:

   ALTERNATEINDEX CTGFV

   1. ALTERNATEINDEX parameters
   2. Cluster object of ALTERNATEINDEX model

   DATA CTGFV

   1. Data parameters
   2. DATA model
   3. ALTERNATEINDEX parameters
   4. Data object of ALTERNATEINDEX model

   INDEX CTGFV

   1. INDEX parameters
   2. INDEX model
   3. ALTERNATEINDEX parameters
   4. Index object of ALTERNATEINDEX model.

   AIXPROC sets the identification of 'AIXFVTbb' in the 8 bytes before the ALTERNATEINDEX CTGFV. AIXPROC checks for a MODEL keyword under ALTERNATEINDEX. If MODEL is specified, MODELPRC issues a UCATLG to retrieve information from the modeled alternate index. The information from the alternate index catalog entry of the modeled data set is put in a table, MDLTABL, and the control interval number for the data and index entries of the modeled data set are saved. MDLTABL contains an address and the length of each field of information returned from the UCATLG. In building the ALTERNATEINDEX CTGFV, information is obtained from MDLTABL and is then overlaid with information specified in the ALTERNATEINDEX parameters. NAMEPROC issues a UTIME macro to get the creation date which is put in an DSETCRDT CTGFL. NAMEPROC puts the address of *objectname* from NAME in the CLUSTER CTGFV. The call to NAMEPROC for initialization of the alternate index level sets up a pointer to the related name and its password, if any, in the CTGFV. ALLCRPOC will set the address of the recovery volume serial work area in the CTGFVWKA field of the alternate index (G) FVT. NAMEPROC builds a DSETEXDT CTGFL with the information from TO | FOR. PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds an OWNERID CTGFL with information from OWNER. The call to PROTPROC in the initialization of the AIX FVT includes an indication as to whether UPGRADE or NOUPGRADE has been specified. PROTPROC builds a RGATTR FPL and initializes it depending upon the information passed by AIXPROC. If neither of these parameters was specified, a default of UPGRADE is set in RGATTR. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, or RECORDS.

Module: IDCDE01, IDCDE02 **(without VS2.03.807)** IDCDE02, IDCDE03 **(with VS2.03.807)**

Procedure: AIXPROC NAMEPROC KEYPROC MODELPRC ALLCPROC PROTPROC

2. AIXPROC sets the identification of DATAFVT in the 8 bytes preceding the DATA CTGFV. The DEFINE FSR builds the DATA CTGFV with the information specified in ALTERNATEINDEX parameters. This information is then overlaid by the information specified in the DATA parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the DATA CTGFV is built with the information specified in the ALTERNATEINDEX parameters.

   If MODEL is specified under ALTERNATEINDEX and MODEL is not specified under DATA, MODELPRC uses the saved control interval number for the data entry of the modeled data set to get information from the data entry. The information from the data entry of the modeled data set is put in MDLTABL. The DATA CTGFV is built with information from MDLTABL and is then overlaid by the information specified in ALTERNATEINDEX parameters. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. The calls to NAMEPROC in the initialization of the DATA FVT for an alternate index includes a pointer to the name of the EXCEPTIONEXIT routine; NAMEPROC builds and initializes the EXCPEXIT FPL and references it in the FVT field CTGFVEXT. KEYPROC builds an AMDSBCAT CTGFL, and ALLCPROC builds a DSATTR CTGFL, but no information is put in them yet. KEYPROC puts the *length* and *offset* from KEYS in the AMDSBCAT CTGFL. If no key values have been specified, KEYPROC sets up defaults. KEYPROC also puts the address of *(lowkey highkey)...* from KEYRANGES in the DATA CTGFV. The calls to KEYPROC in the construction of the DATA FVT of an AIX includes an indication of UNIQUEKEY/NONUNIQUEKEY. KEYPROC

initializes the AMDUNQ flag in the AMDSB to indicate the appropriate condition. KEYPROC sets the AMDRKP field to a fixed value of X'05' and the AMDAXRKP field to the value specified for relative key position. KEYPROC sets the AMDSPAN flag in the AMDSB since all alternate indexes have the spanned attribute. The AMDSB FPL is built by KEYPROC. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. ALLCPROC also builds a BUFSIZE CTGFL with information from BUFFERSPACE. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK | NOWRITECHECK
maximum from RECORDSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT

PROTPROC puts ERASE | NOERASE, REUSE | NOREUSE, and *crosspartition crosssystem* from SHAREOPTIONS in the DSATTR CTGFL.

Protection information is obtained only from the MODEL via MDLTABL in order to provide different protection at the ALTERNATEINDEX and DATA. PROTPROC builds a PASSWALL CTGFL with protection information from the MODEL as well as a OWNERID CTGFL with owner information from the MODEL.

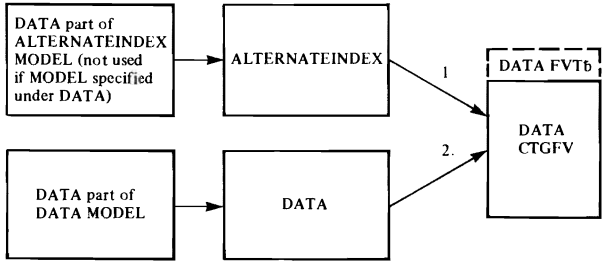
On the second pass, called the explicit pass, if MODEL is not specified under DATA, the information specified in the DATA parameters overlays the information placed in the DATA CTGFV on the implicit pass.

If MODEL is specified under DATA, MODELPRC issues a UCATLG to get information from the data catalog entry of the modeled alternate index. The information from the data entry of the modeled alternate index is put in MDLTABL. The information in MDLTABL overlays the information placed in the DATA CTGFV on the implicit pass. Finally, the information in the DATA CTGFV is overlaid with the information specified in the DATA parameters. NAMEPROC puts the address of *objectname* from NAME in the DATA CTGFV. KEYPROC puts *length* and *offset* from KEYS in the AMDSBCAT CTGFL. KEYPROC puts the address of *(lowkey highkey)...* from KEYRANGES in the DATA CTGFV. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. Note: the volume serial list is not merged with any other volume serial list. ALLCPROC also builds or modifies the SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS; the LRECL CTGFL with *average* from RECORDSIZE; and the BUFSIZE CTGFL with *size* from BUFFERSPACE. PROTPROC builds or modifies the PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION.

PROTPROC also builds or modifies the OWNERID CTGFL with *ownerid* from OWNER. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK | NOWRITECHECK
maximum from RECORDSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESSTAGEWAIT

UNIQUE | SUBALLOCATION and SPEED | RECOVERY are put in the DSATTER CTGFL. ERASE | NOERASE, REUSE | NOREUSE, and *crosspartition crosssystem* from SHAREOPTIONS are put in the DSATTR CTGFL.



Module: IDCDE01, IDCDE02 **(without VS2.03.807)** IDCDE02, IDCDE03, **(with VS2.03.807)**

Procedure: AIXPROC, NAMEPROC, KEYPROC, ALLCPROC, MODELPROC, IXOPPROC, PROTPROC

3. An INDEX CTGFV is always built for an alternate index.

AIXPROC sets the identification of INDEXFVT in the 8 bytes preceding the INDEX CTGFV. The DEFINE FSR builds the INDEX CTGFV with the information specified in ALTERNATEINDEX parameters, which is overlaid by the information specified in the INDEX parameters. Two passes are performed. On the first pass, called the implicit pass, if MODEL is not specified, the INDEX CTGFV is built with the information specified in ALTERNATEINDEX parameters.

If MODEL is specified under ALTERNATEINDEX and MODEL is not specified under INDEX, MODELPRC uses the saved control interval number for the index entry of the modeled alternate index to get information from the index entry. The information from the index entry of the modeled alternate index is put in MDLTABL. The INDEX CTGFV is built with information from MDLTABL and then overlaid by the information specified in the ALTERNATEINDEX parameters. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. The calls to NAMEPROC in the initialization of the DATA and INDEX FVTs for an alternate index includes a pointer to the name of the EXCEPTIONEXIT routine; NAMEPROC builds and initializes the EXCPEXIT FPL and references it in the FVT field CTGFVEXT. KEYPROC builds an AMDSBCAT CTGFL, and ALLCPROC builds a DSATTR CTGFL, but no information is put in them yet. IXOPPROC puts REPLICATE | NOREPLICATE and IMBED | NOIMBED in the AMDSBCAT CTGFL. ALLCPROC puts the address of the *dname* from FILE and the address of *volser* from VOLUMES in the INDEX CTGFV. ALLCPROC also builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
WRITECHECK | NOWRITECHECK
size from CONTROLINTERVALSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT

UNIQUE | SUBALLOCATION and REUSE | NOREUSE is put in the DSATTR CTGFL. Record size is not indicated because it is always fixed length for the index of an alternate index.

Protection information is obtained only from the MODEL via MDLTABL in order to provide different protection at the ALTERNATEINDEX and INDEX. PROTPROC builds a PASSWALL CTGFL with protection information from the MODEL as well as a

OWNERID CTGFL with owner information from the MODEL

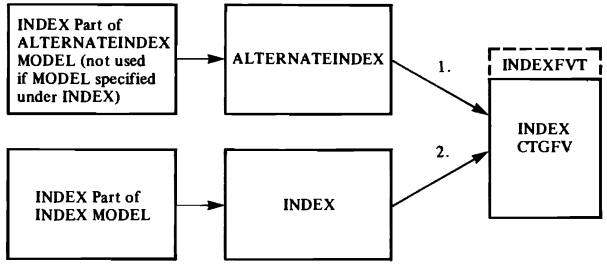
On the second pass, called the explicit pass, if MODEL is not specified under Index, the informatior specified in the INDEX parameters overlays the information placed in the INDEX CTGFV on the implicit pass.

If MODEL is specified under INDEX, MODELPRC issues a UCATLG to get information from the index catalog entry of the modeled alternate index. The information from the index entry of the modeled alternate index is put in MDLTABL. The informatior in MDLTABL overlays the information placed in the INDEX CTGFV on the implicit pass. Finally, the information in the INDEX CTGFV is overlaid with the information specified in the INDEX parameters. NAMEPROC puts the address of *objectname* from NAME in the INDEX CTGFV. IXOPPROC puts REPLICATE | NOREPLICATE and IMBED | NOIMBED in the AMDSBCAT CTGFL. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the INDEX CTGFV. ALLCPROC also builds or modifies the SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, or RECORDS. PROTPROC builds or modifies the PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds or modifies the OWNERID CTGFL with *ownerid* from OWNER. The following is put in the AMDSBCAT CTGFL:

ORDERED | UNORDERED
WRITECHECK | NOWRITECHECK
size from CONTROLINTERVALSIZE
STAGE | BIND | CYLINDERFAULT
DESTAGEWAIT | NODESTAGEWAIT

The following is put in the DSATTR CTGFL:

UNIQUE | SUBALLOCATION
ERASE | NOERASE
REUSE | NOREUSE
crosspartition crosssystem from SHAREOPTIONS



Module: IDCDE01

Procedure: IDCDE01, DALCPROC

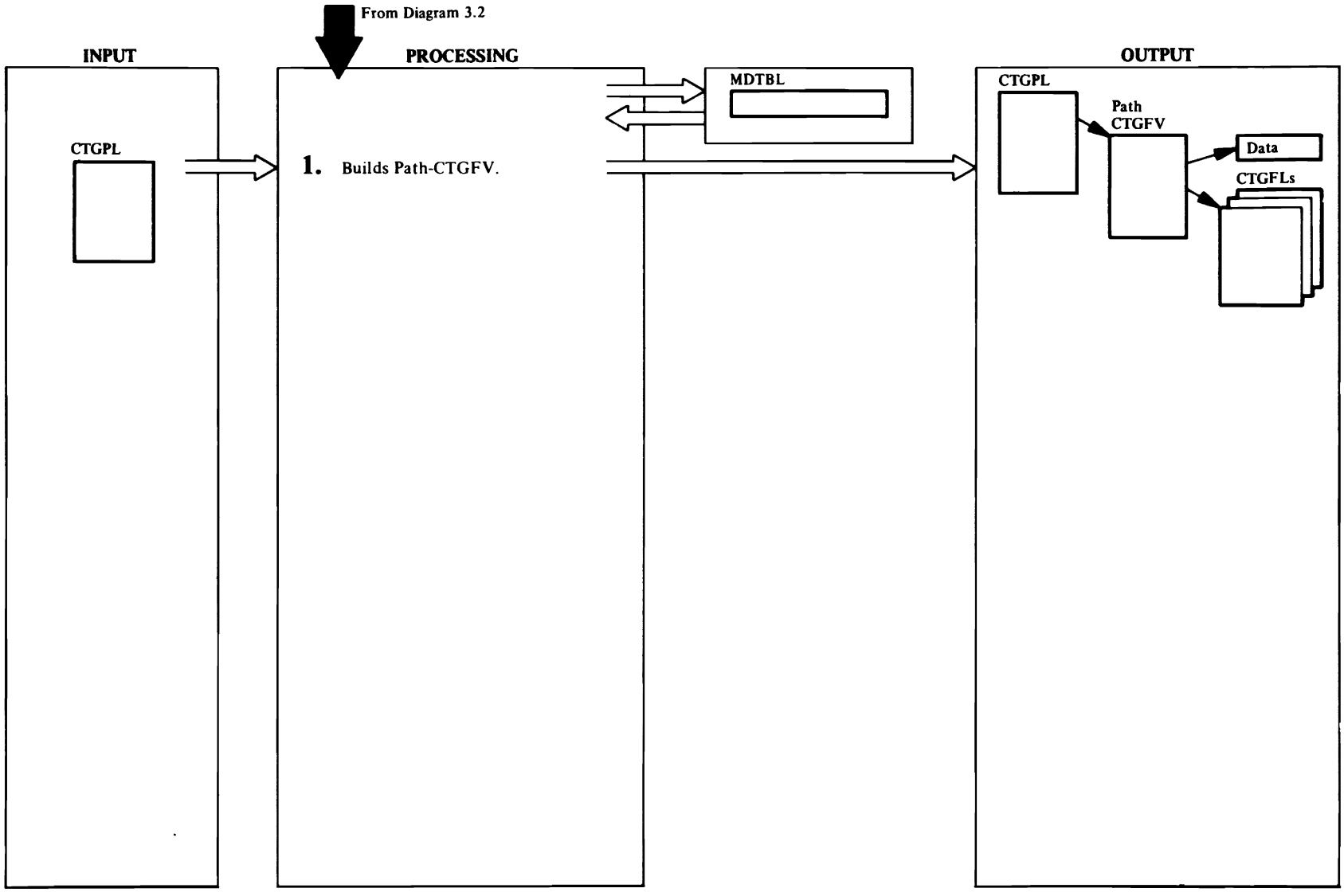
4. **Note: Users with VS2.03.807 should ignore this step.** For an alternate index two or three CTGFVs have been built—one each for alternate index, data, and index information. If a VOLUME CTGFV has been built, it does not have any information in it because VSAM uses it for a work space. The following table shows the possible places where a SPACPARM CTGFL may have been built and the action IDCDE01 takes.

SPACPARM CTGFL

Alternate Index	Data	Index	Action
X	X	X	This is an error; IDCDE01 terminates the DEFINE.
X	X		This is an error; IDCDE01 terminates the DEFINE.
X		X	This is an error; IDCDE01 terminates the DEFINE.
	X	X	OK; no action.
X			OK; no action.
	X		OK; no action.
		X	This is an error; IDCDE01 terminates the DEFINE.
none	none	none	This is an error; IDCDE01 terminates the DEFINE.

For a UNIQUE data component, CTGFV is checked for a *dname* from FILE. If FILE is not supplied, DALCPROC builds an ALLAGL and issues a UALLOC macro to dynamically allocate the volumes. The *dname* returned by UALLOC is put in CTGFVIND. This same processing is performed for a UNIQUE index component. If either the data or index component is UNIQUE, IDCDE01 builds a null volume FVT. Control goes to Diagram 3.2, step 3.

Diagram 3.3.9 DEFINE FSR – DEFINE PATH



Extended Description for Diagram 3.3.9

Module: IDCDE01, IDCDE02 (**without VS2.03.807**)
IDCDE02, IDCDE03 (**with VS2.03.807**)

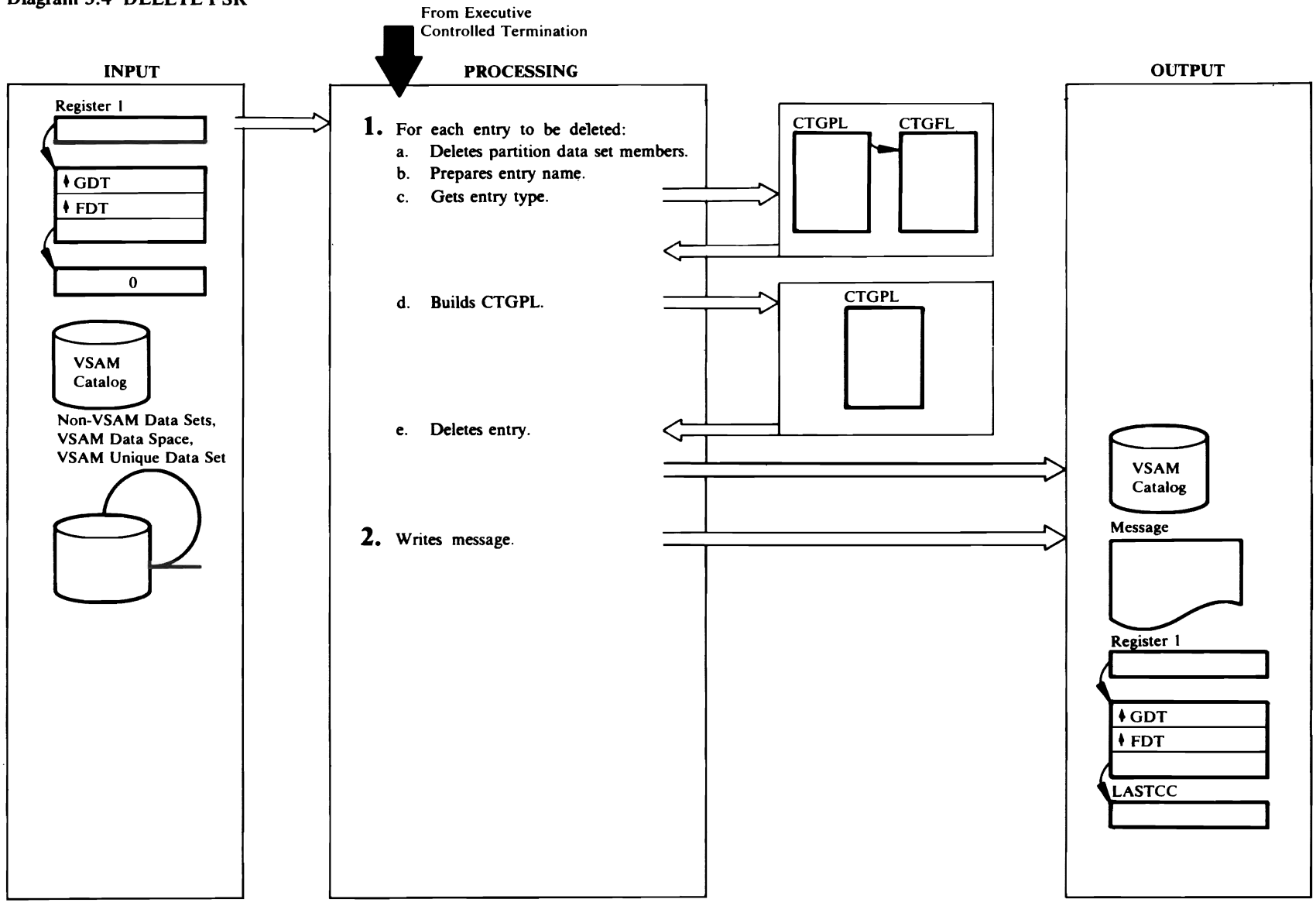
Procedure: PATHPROC, NAMEPROC, MODELPRC, PROTPROC, ALLCPROC

1. In the DEFINE PATH command, you specify information under one main keyword: PATH. The DEFINE FSR builds a CTGFV to describe the path. Information specified under PATH goes in the PATH CTGFV.

   If MODEL is specified, the information in the command overrides the information in a model. A model has one catalog entry to describe its path. The information in a model's path catalog entry is used to build the PATH CTGFV.

   PATHPROC checks for a MODEL keyword under PATH. If MODEL is specified, MODELPRC issues a UCATLG to retrieve information from the modeled VSAM data set. The information from the path catalog entry of the modeled data set is put in a table, MDLTABL. MDLTABL contains an address and the length of each field of information returned from the UCATLG. In building the PATH FVT, information is obtained from the MDLTABL and is then overlaid by information specified in the PATH parameters. PATHPROC sets the identification of PATHFVT in the 8 bytes before the PATH CTGFV. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC puts the address of *objectname* from NAME in the PATH CTGFV. NAMEPROC is supplied with the address necessary to reference the PATHENTRY name and places its address in CTGFVNAM. The password of the PATHENTRY is referenced from CTGFVPWD. NAMEPROC builds a DSETEXDT CTGFL with the information from TO | FOR, PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds an OWNERID CTGFL with information from OWNER. The call to PROTPROC in the construction of the PATH FVT includes the UPDATE | NOUPDATE indication for a path. PROTPROC builds the RGATTR FPL and references it in the PATH FVT field CTGFVUPG. If neither of these parameters was specified, a default of UPDATE is set in the RGATTR. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field of the PATH FVT. The CTGFVTYP field of the PATH FVT is set to R.

Diagram 3.4 DELETE FSR



Extended Description for Diagram 3.4

Module: IDCDL01

Procedure: FINDTYPE, BUILDCPL, CATCALL, IDCDL01, MEMDLETE, PARAMCHK, DELTPROC, ALLOPROC, RC240PRC

1. IDCDL01 checks the FDT to see if the *entryname* needs to be qualified. If the name needs qualification, IDCDL01 issues a UQUAL macro and uses the returned name for the remainder of the DELETE FSR. The following steps are performed for each *entryname* to be deleted. Control goes to step 3 to terminate the command when all *entrynames* have been deleted, or a serious error is encountered.

   a. IDCDL01 checks the FDT to see if the *entryname* contains a *membername* indicating the data set is a partitioned data set, PDS. If it is not, control goes to step 2.b. MEMDLETE checks for errors. MEMDLETE builds an OPNAGL to open the partitioned data set. If FILE is not specified, MEMDLETE puts the *entryname* in the OPNAGL. This causes the UOPEN macro to dynamically allocate the data set. MEMDLETE issues a UOPEN macro to open the partitioned data set. To delete the member name, MEMDLETE issues a USTOW macro. MEMDLETE issues a UCLOSE to close the partitioned data set. No more processing is done on the partitioned data set. Error or status messages are written. Control goes to step 2.a for the next entry.

   b. IDCDL01 also checks the FDT to see if the *entryname* is a generic data set name—that is, it contains an *. For a generic name, IDCDL01 issues a UCIR macro to get a list of data set names. Each name in the list is treated as though it had been specified as an *entryname* in the DELETE command. Any catalog name returned with the data set names that match the *entryname* is saved and used in calls to VSAM catalog management.

   c. If the entry type is not specified with the command, FINDTYPE builds a CTGPL and three CTGFLs in which VSAM returns the entry type, catalog ACB address, and cluster attribute field. FINDTYPE initializes the CTGPL and CTGFLs once for the entire DELETE command, and they are used with each *entryname*. FINDTYPE issues a UCATLG macro to locate the entry type, catalog ACB address and cluster attribute field. If the *entryname* is a pagespace and ERASE is specified, the *entryname* is not deleted; control goes to step 2 for the next *entryname*. If the *entryname* is non-VSAM and NOSCRATCH is not specified, ALLOPROC issues a UALLOC macro to allocate the data set. Control goes to step 2 for the next *entryname*. If the return code is non-zero and not 8, FINDTYPE writes an error message, but the rest of the DELETE command is processed.

      PARAMCHK checks for invalid or insufficient parameters which the Reader/Interpreter cannot check. The Reader/Interpreter cannot do all the necessary parameter checking if the user has not specified the entry type. PARAMCHK checks if no TYPE is specified and the type was found by FINDTYPE, or if the *entryname* is a generic name and type is not specified. If there is an invalid parameter, PARAMCHK writes an error message, but the rest of the DELETE command is processed.

   d. BUILDCPL builds a CTGPL to delete the entry. BUILDCPL initializes the CTGPL once for the entire DELETE command, and it is used over and over for each *entryname*. BUILDCPL puts the following information in the CTGPL: the address of the *entryname*, the address of the *dname*, type of entry if specified on the command, PURGE | NOPURGE, ERASE | NOERASE, FORCE | NOFORCE, SCRATCH | NOSCRATCH, address of a *password* if specified, and the address of the catalog name if CATALOG is specified or the catalog ACB address if FINDTYPE got the catalog ACB address. BUILDCPL also puts the address of a work area needed by VSAM in the CTGPL. BUILDCPL puts the address of the entry name and the address of the entry password in the CTGPL.

      If the entry type is non-VSAM and neither SCRATCH or NOSCRATCH is specified, BUILDCPL sets SCRATCH in the CTGPL. If the type is NONVSAM and the data set is to be scratched, ALLOPROC issues a UALLOC macro to allocate the data set. If the entry was located from the catalog, BUILDCPL puts the entry type in the CTGPL. If the entry is a VSAM data space, ALLOPROC issues a UALLOC macro to mount the volume. If the entry type is a cluster or alternate index with ERASE specified but FILE not specified, ALLOPROC issues a UALLOC macro to allocate the cluster or alternate index.

   e. CATCALL deletes the entryname by issuing a UCATLG macro with the CTGPL built by BUILDCPL. If the length field of the workarea passed to VSAM has a number greater than 4 in the second 2 bytes, catalog has returned a list of deleted objects. CATCALL writes the name of each deleted object in the entry with a UPRINT macro.

      If the VSAM catalog return code is anything other than 40, 72, 76, or 240, control returns to DELTPROC. Return codes of 0, 110 (**VS2.03.807 only**), and 160 indicate that the function completed successfully: all other return codes indicate errors. On a VSAM catalog return code of 160, a message is printed indicating the deletion of data spaces on a volume did not cause the volume entry record to be deleted from the catalog; this occurs when the volume contains non-empty data spaces (or is a candidate volume for a VSAM data set) and the FORCE parameter is not specified. For **VS2.03.807**, a VSAM catalog return code of 110 causes a message to be printed indicating that the profile of a Resource Access Control Facility (RACF) entity could not be deleted because it is not present (reason code 4), or because it is not eligible for deletion (reason code 8).

      On a return code of 40, MORESP calculates the size of the workarea required by VSAM. If the present one is large enough, MORESP clears it. Otherwise, unless the workarea is in PL/S automatic storage, it is freed and a new workarea is gotten. CATCALL then re-issues the UCATLG with the delete CTGPL.

      For a VSAM catalog return code of 72 or 76, catalog management returns the name of an object to be dynamically allocated: this is the last entry in the workarea. ALLOPROC issues a UALLOC for this entry, and CATCALL then re-issues the UCATLG for the delete request.

      If the VSAM catalog return code is 240 and the user specified the CATALOG parameter, RC240PRC determines if the entryname to be deleted is accessible to dynamic allocation by performing listcatalogs to obtain the name of the catalog in which dynamic allocation will find it. Provided that the data set is accessible to allocation, ALLOPROC issues a UALLOC to dynamically allocate it, and CATCALL then re-issues the UCATLG to perform the delete operation.

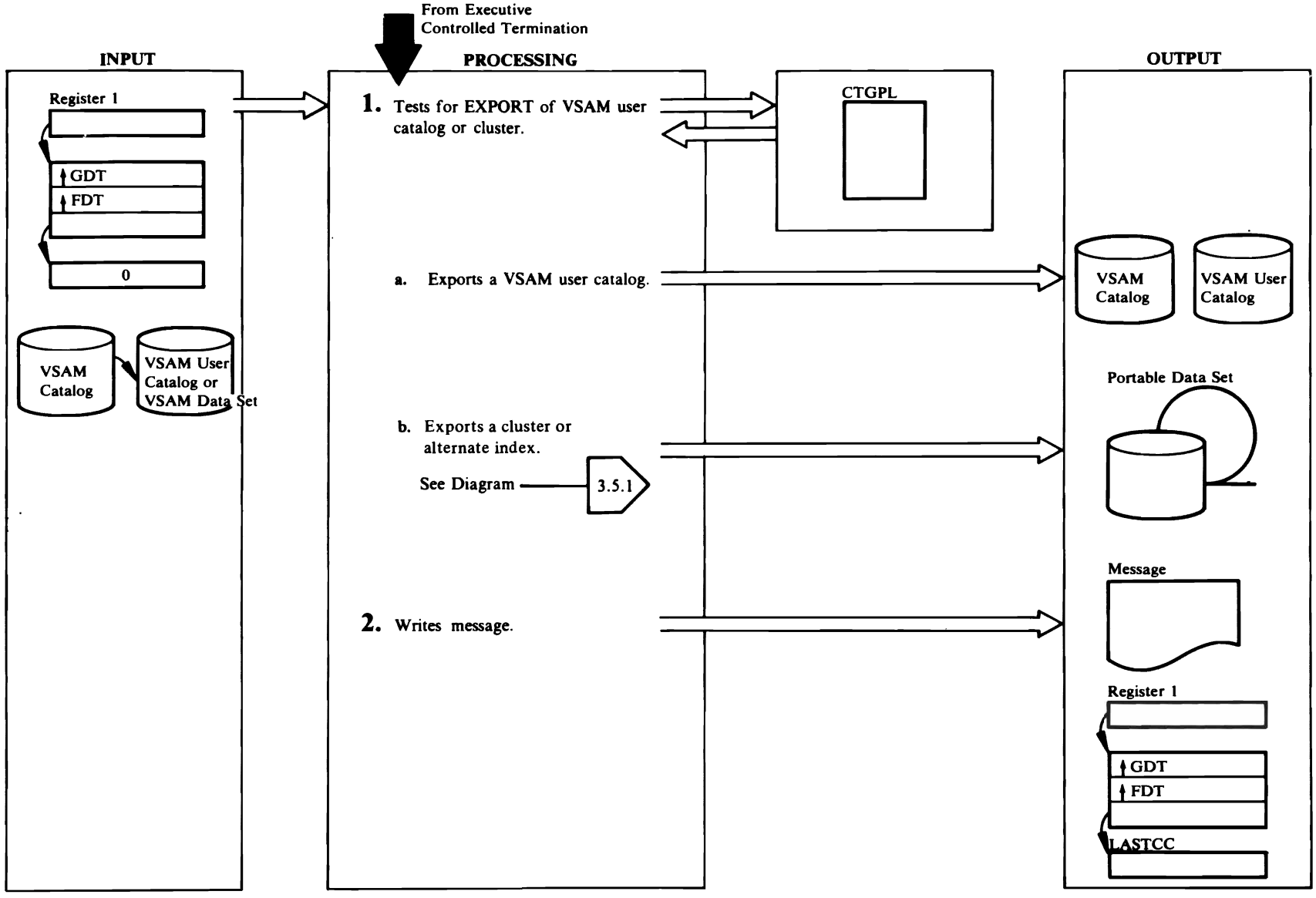
      This step is repeated until the VSAM catalog return code is other than 40, 72, 76, or 240.

Module: IDCDL01

Procedure: IDCDL01

2. IDCDL01 prints a message with LASTCC. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.5 EXPORT FSR



Extended Description for Diagram 3.5

Module: IDCXP01

Procedure: ALTRPROC, IDCXP01, DELTPROC, DSCTPROC, LOCPROC, CTLGPROC, OPENPROC, PUTPROC, CLUSPROC, RFCPROC

1. IDCXP01 tests the FDT for DISCONNECT in the EXPORT command. Step 1.a is done if DISCONNECT is specified, or step 1.b is done if DISCONNECT is not specified.

   a. DELTPROC builds a CTGPL to delete the user catalog entry in the VSAM catalog. DELTPROC issues a UGPOOL for a work area in which VSAM puts deleted names. If a *password* is supplied, LOCPROC puts it in the CTGPL. CTLGPROC deletes the user catalog entry by issuing a UCATLG macro with the CTGPL. If the return code is 40, the work area addressed from the CTGPL is too small. The former work area is released with a UFSPACE, and the returned size of the work area needed is used with a UGPOOL to get another work area. If the new work area is obtained, another UCATLG macro is issued. If the return code from the first UCATLG is non-zero and not 40, or if the return code from the second UCATLG is non-zero, an error message is written by building an error conversion table and issuing the UERROR macro.

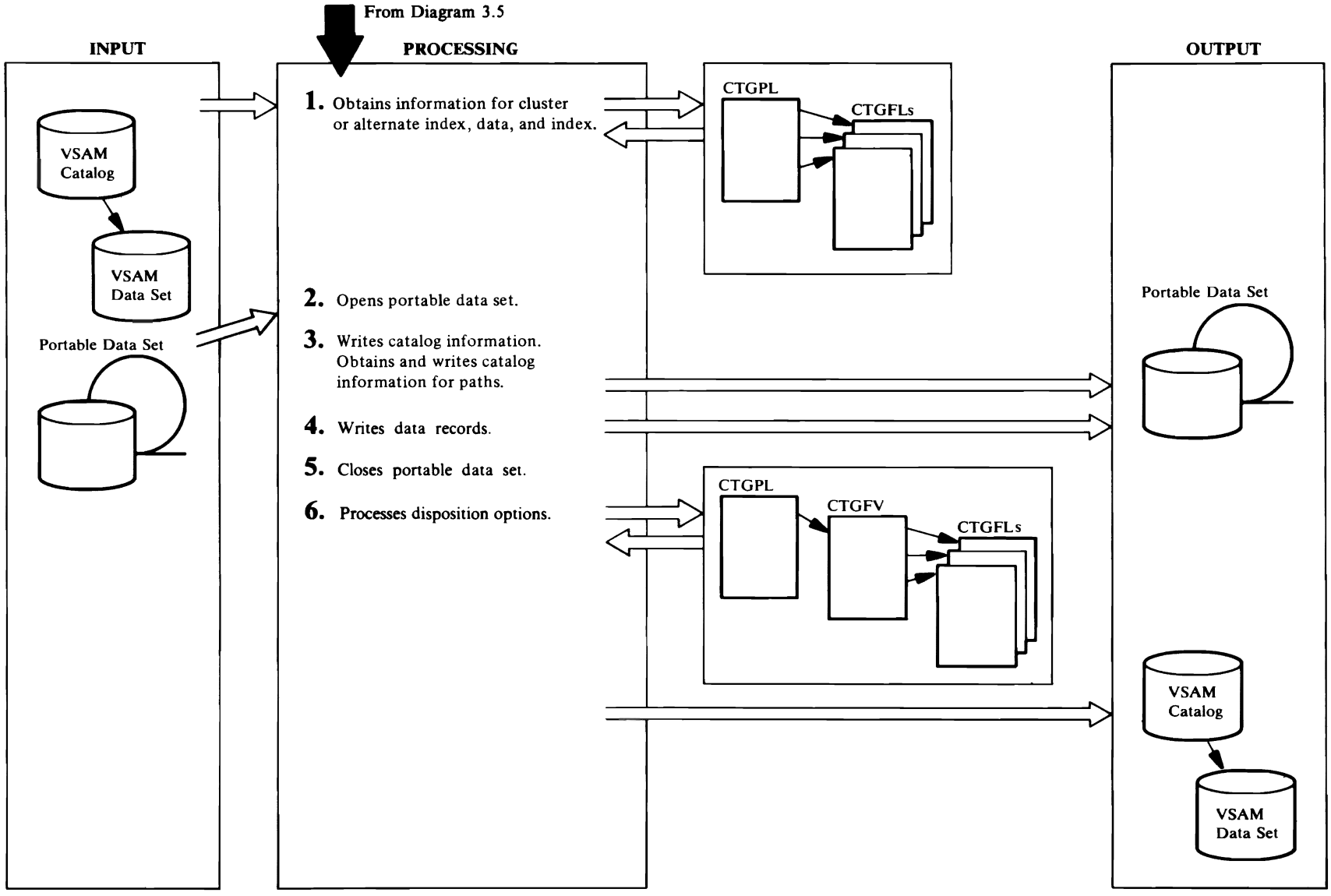
   b. CLUSPROC tests the FDT to see if the cluster or alternate index and the portable data set names need to be qualified. Data set names are qualified if Access Method Services is invoked interactively with TSO. If the data set names need qualification, CLUSPROC issues a UQUAL macro for each name and uses the returned data set names for the remainder of the EXPORT FSR. LOCPROC gets catalog information about the components of the VSAM data set. OPENPROC opens the portable data set for output. PUTPROC writes catalog information and data records on the portable data set. CLUSPROC closes the portable data set and processes the disposition options, TEMPORARY | PERMANENT. Refer to "Appendix A" for a description of the portable data set. Diagram 3.4.1 shows exporting a cluster or alternate index in detail.

Module: IDCXP01

Procedure: IDCXP01

2. IDCXP01 writes a message with LASTCC. Messages listing the exported catalog or VSAM data set are written. IDCXP01 closes any open data sets with the UCLOSE macro. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.5.1 EXPORT FSR – Cluster or Alternate Index



Extended Description for Diagram 3.5.1

Module: IDCXP01

Procedure: LOCPROC, CTLGPROC, IDCXP01

1. For the cluster or alternate index entry of the VSAM data set, LOCPROC builds a CTGPL and CTGFLs to retrieve information from the VSAM catalog. One CTGFL is built for each of the following pieces of information:

   Entry type
   Entry name
   Data set attributes
   Data set owner
   Data set creation date
   Data set expiration date
   Password
   Password prompting
   Password attempts
   User module name
   User module area
   Space information
   Buffer size
   Logical record length
   Low key on volume
   High key on volume
   AMDSB control block
   Cluster attributes
   Type and name of data and index entry
   Exception exit
   Alternate index and Path attributes
   Catalog ACB

   CTLGPROC issues a UCATLG with the CTGPL and CTGFLs to retrieve the information from the catalog. If the work area is too small, CTLGPROC will enlarge it and reissue the UCATLG. If the LOCATE fails for a reason other than that the work area is too small, an error message is written by building an error conversion table and issuing the UERROR macro. This processing occurs for all UCATLG requests issued by CTLGPROC. IDCXP01 tests to be sure that the type of catalog entry is a cluster or alternate index and not a pagespace. If it is not a cluster or alternate index, an error message is written and the VSAM data set is not exported. Information is requested on all the fields even if the information is not available in the cluster entry because VSAM ignores requests for fields that do not apply for this entry.

   LOCPROC builds a CTGPL and CTGFLs for the data entry of the VSAM data set. CTGFLs are built for each piece of information in the above list except the last two, type and name of data and index entry, and catalog ACB. The control interval of the data entry is used to find the data entry. CTLGPROC issues a UCATLG with the CTGPL and CTGFLs to retrieve the information from the catalog. If the work area is too small, CTLGPROC enlarges it and reissues the UCATLG. The returned information is saved.

   The processing in the above paragraph is repeated for the index entry if any.

Module: IDCXP01

Procedure: OPENPROC

   CLUSPROC determines if the object being exported is an alternate index. If so, LOCPROC builds a CTGPL and CTGFLs for the base cluster associated with the alternate index. CTFGLs are built for entry type and entry name. CTLGPROC issues a UCATLG to retrieve this information. The entry name will be written to the portable data set as the related name.

2. OPENPROC builds an OPNAGL. If OUTDATASET is specified, OPENPROC puts the data set name in the OPNAGL. This causes the data set to be dynamically allocated by the UOPEN macro. OPENPROC issues a UOPEN to open the portable data set for output. If the return code is non-zero, an error message is written and the VSAM data set is not exported. Refer to *Appendix A* for a description of the portable data set.

Module: IDCXP01

Procedure: CLUSPROC, PUTPROC, CONTRBL

3. CONTRBL constructs a dictionary for each CTGPL. The CTGFLs contain information returned by VSAM. If a fixed length field has no information, VSAM puts all binary ones in the CTGFL where the information would have been. If a variable length field has no information, VSAM puts zeros in the two-byte length field that precedes the field in the CTGFL where the information would have been. If INHIBITTARGET is specified, a flag is set in the portable data set so IMPORT can process INHIBITTARGET. PUTPROC writes the dictionary followed by the information from each CTGFL—except the CTGFL for cluster attributes. If the length of the dictionary or catalog information is greater than the logical record length for the portable data set, PUTPROC writes the dictionary or catalog information in segments. PUTPROC writes the records with a UPUT macro. Refer to *Appendix A* for the format of the portable data set.

   After the catalog information pertaining to the cluster or alternate index and associated data and index objects has been written to the portable data set, CLUSPROC obtains information regarding all paths which have been defined over the object being exported. For the first path association LOCPROC builds a CTGPL and CTGFLs to retrieve the information from the VSAM catalog. CTGFLs are built for the same pieces of information as for the data and index objects. CTLGPROC issues a UCATLG to retrieve the information which is then written to the portable data set. In addition, the name of the cluster or alternate index being exported and its password are written to the portable data set as the PATHENTRY name and PATHENTRY password. CONTRBL is called to actually construct the portability record. CLUSPROC retrieves information for all the remaining path associations and then writes it to the portable data set using the same CTGPL and CTGFLs which were set up for the first path association. Prior to calling CTLGPROC for each, the work area is cleared and the control interval number of the next associated path is placed in the CTGPL.

Module: IDCXP01

Procedure: RECPROC, LOCPROC, OPENPROC

4. CLUSPROC issues a UALLOC macro to allocate the VSAM data set if it was not allocated via JCL. RECPROC opens the VSAM data set with a UOPEN macro and issues a UCOPY to copy all the records to the portable data set. RECPROC issues a UCLOSE to close the VSAM data set. Following a successful open, RECPROC compares the data set name returned by UOPEN to that specified by the caller as the entry name in the EXPORT command. If the compare is unequal, LOCPROC builds a CTGPL and CTGFLs to perform a LOCATE on the name returned by UOPEN. CTGFLs are built for ENTYPE and NAMEDS. CTLGPROC issues a UCATLG macro. If the ENTYPE returned is not that of a path, an error message is written and the command is terminated. If the ENTYPE is that of a path, a second LOCATE is performed using the control interval number of the pathentry object. A CTGFL is built for ENTNAME by LOCPROC and a UCATLG macro issued by CTLGPROC. If the name returned is not equal to the entry name specified in the EXPORT command, a message is written and the command terminated.

   When exporting a relative record data set, the relative record number of each record written to the portable data set is placed by UCOPY in a 4-byte area immediately preceding the record itself. OPENPROC

triggers this processing by setting the Export/Import flag in the OPNAGL of the input data set.

Module: IDCXP01

Procedure: CLUSPROC

5. CLUSPROC issues a UCLOSE to close the portable data set.

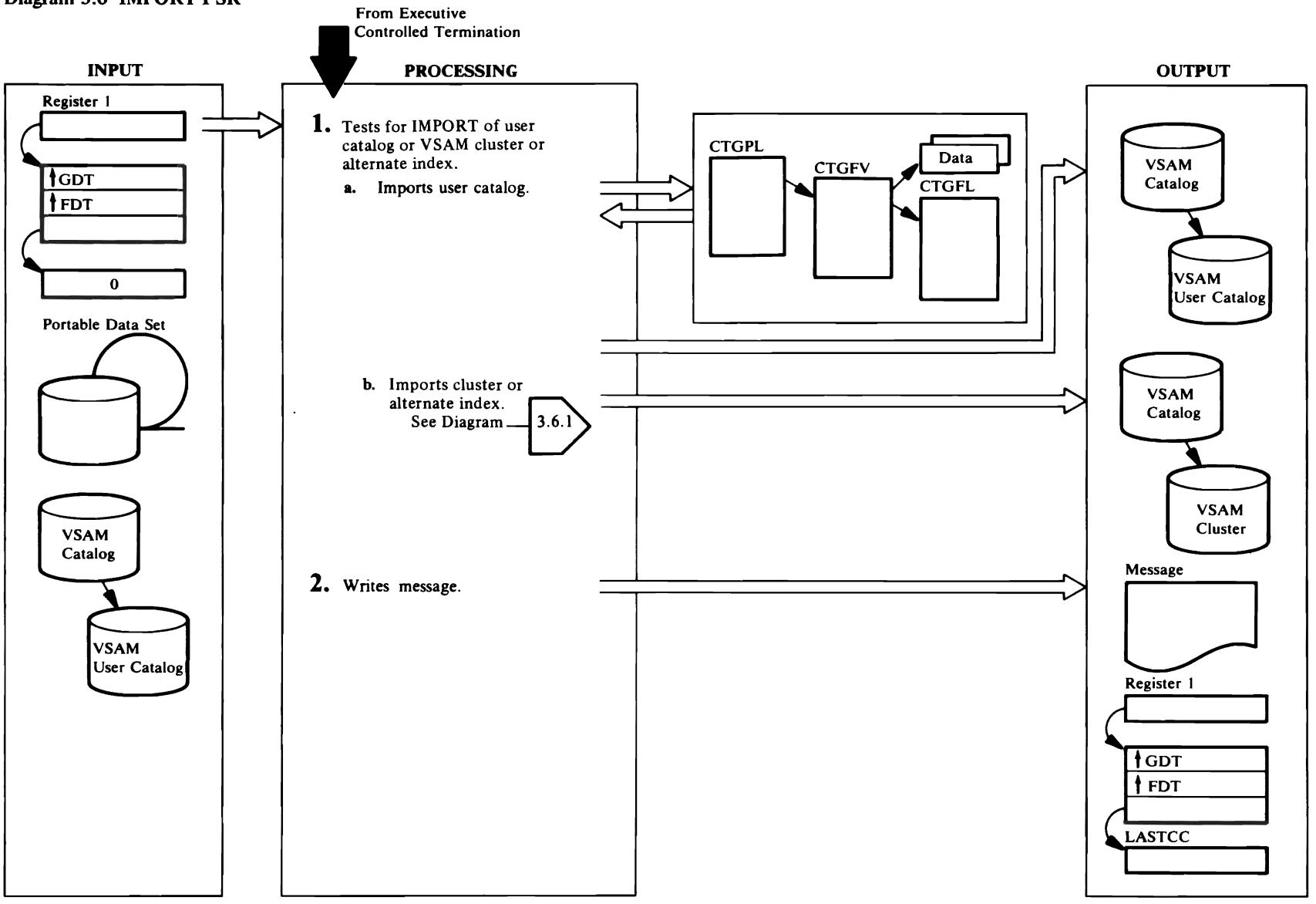
Module: IDCXP01

Procedure: DELTRPOC, CLUSPROC, CTLGPROC, ALTRPROC, MORESP

6. If PERMANENT is specified, DELTPROC builds a CTGPL. If ERASE or PURGE is specified, CLUSPROC changes the CTGPL. CTLGPROC issues a UCATLG to delete the VSAM data set from the VSAM catalog. If the delete fails, an error message is written by building an error conversion table and issuing the UERROR macro. The names of all deleted entries are printed. If the VSAM catalog return code is 40, MORESP is called to get a larger work area. If the VSAM catalog return code is 110, (**VS2.03.807 only**), a message is printed indicating that the profile of a RACF-indicated entity could not be deleted because it was not found or was not eligible for deletion.

   If TEMPORARY is specified, the temporary export field must be turned on in the catalog entry. ALTRPROC modifies the existing CTGPLs, builds a CTGFV, and modifies the existing CTGFLs for the fields that need to be changed in the VSAM catalog. The temporary export flag and, if INHIBITSOURCE is specified, the inhibit update flag is set in the DSATTR CTGFL. An ENTNAME CTGFL for the *entryname* is also built. ALTRPROC places the address of the dname specified in the INFILE parameter, if any, in the CTGFV for catalog recovery purposes. CTLGPROC issues one UCATLG for the data entry and one UCATLG for the index entry if it exists. The data set attributes field does not appear at the cluster or alternate index entry. CLUSPROC issues a UDEALLOC macro to deallocate the cluster or alternate index if it was allocated by EXPORT. Control returns to Diagram 3.5, step 2.

Diagram 3.6 IMPORT FSR



Extended Description for Diagram 3.6

Module: IDCMP01

Procedure: OPENPROC, IDCMP01, CLUSPROC, CNCTPROC, CPLPROC, LVLRPROC, CTLGPROC, RECPROC, ALTRPROC

1. IDCMP01 tests the FDT for the CONNECT keyword in the IMPORT command to determine if a VSAM data set or a VSAM catalog is being imported. If CONNECT is specified, a VSAM user catalog is being imported, and step 1.a is done. If CONNECT is not specified, a VSAM data set is being imported, and step 1.b is done.

   a. The following is repeated for every *objectname* in OBJECTS. **Note:** More than one user catalog can be imported with one IMPORT command. CPLPROC initializes a CTGPL. FVTPROC builds a CTGFV. LVLRPROC builds a DEVTYPE CTGFL from the DEVICETYPES in the command. LVLRPROC builds a volume list from VOLUMES and puts the address of the volume list in the CTGFV. CNCTPROC puts the address of the *objectname* from OBJECTS in the CTGFV. If no *objectname* is specified, an error message is written, and the catalog is not imported. The operation type field in the CTGFV is set to 'A' to indicate a catalog connect. CNCTPROC issues a UCATLG to connect the catalog. If the return code is non-zero, an error message is written by building an error conversion table and invoking the UERROR macro. When all the catalogs have been connected, control goes to step 2.

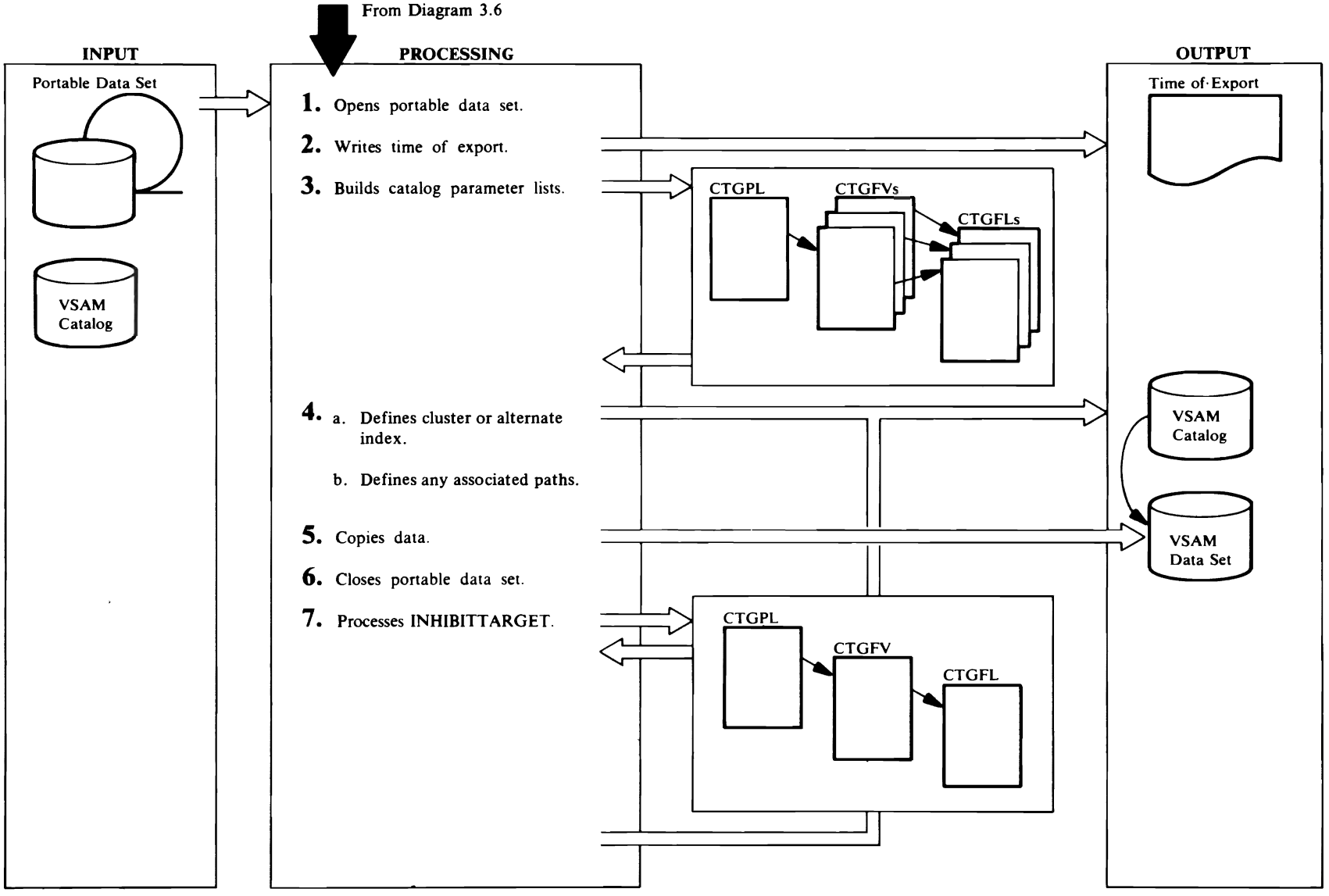
   b. CLUSPROC tests the FDT to see if the portable or target name needs to be qualified. Data set names are qualified if Access Method Services in invoked interactively with TSO. If the data set names need qualification, CLUSPROC issues a UQUAL macro for each name and uses the returned data set names for the remainder of the IMPORT FSR. OPENPROC opens the portable data set. CLUSPROC writes the time of export with a UPRINT macro. CLUSPROC uses the catalog information in the portable data set to "define" the VSAM data set. OPENPROC opens the VSAM data set, and RECPROC copies the data records from the portable data set to the VSAM data set. If INHIBITTARGET was specified when the VSAM data set was exported, ALTRPROC alters the catalog entry for the VSAM data set. Refer to "Appendix A" for the format of the portable data set.

Module: IDCMP01

Procedure: IDCMP01

2. IDCMP01 writes a message with LASTCC. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.6.1 IMPORT FSR – Cluster or Alternate Index



Extended Description for Diagram 3.6.1

Module: IDCMP01

Procedure: OPENPROC, IDCMP01

1. OPENPROC builds an OPNAGL. If INDATASET is specified, OPENPROC puts the data set name in the OPNAGL. This causes the data set to be dynamically allocated by the UOPEN macro. OPENPROC issues a UOPEN macro to open the portable data set. The portable data set was created by an EXPORT command and contains catalog information and data records for the VSAM data set that was exported. Refer to "Appendix A" for the format of a portable data set. If the return code is non-zero, IDCMP01 writes a message. If the portable data set is open, IDCMP01 issues a UCLOSE to close the data set, and the IMPORT command is terminated.

Module: IDCMP01

Procedure: CLUSPROC

2. CLUSPROC gets the first record of the portable data set which contains the time the portable data set was created by the EXPORT FSR. MSGPROC writes the time with a UPRINT macro.

3. The information for catalog parameter lists comes from three places; the portable data set's copy of the previous catalog entry, the IMPORT command, and both the portable data set and the IMPORT command.

   a. CLUSPROC via CPLPROC builds a CTGPL for a define operation. CLUSPROC issues a UGET macro to read the first catalog record in the portable data set. The catalog record contains the size of the data record that follows. FVTPROC builds from 2 to 3 CTGFVs, one each for the cluster or alternate index and its associated data and index entries. BFPLPROC builds CTGFLs with information from the portable data set. The exception is the PASSWALL CTGFL which is built by BPASPROC. If the exported VSAM data set was UNIQUE, IUNIQPRC builds a CTGFV for volume information. No data is put in the volume CTGFV. If the object being imported is an alternate index, the related name (given in the RELATE parameter) is passed via the alternate index (G) FVT. A work area for the return of the catalog recovery volume serial number, if any, is passed via the cluster or alternate index FVT.

   b. If VOLUMES information is not in the portable data set or not specified in the IMPORT command for at least one object (either cluster, data, or index), an error message is written and the IMPORT is terminated. LVLRPROC puts the address of the *volser...* list from VOLUMES in the CTGFV for the *objectname* in the OBJECTS parameter.

Module: IDCMP01

Procedure: CLUSPROC, CPLPROC, FVTPROC, BFPLPROC, BPASPROC, IUNIQPRC, LVLRPROC, RANGPROC

   c. If ORDERED | UNORDERED is specified for a particular *objectname*, CLUSPROC changes the AMDSBCAT CTGFL for the *objectname*. If KEYRANGES is specified for the index object, RANGPROC builds a list of key ranges and puts the address of the key range list in the CTGFV. If NEWNAME is specified for a particular object, CLUSPROC puts the address of the new name in the particular CTGFV. Data from the IMPORT command overrides data from the portable data set. If an exported data or index component was unique, DALCPROC issues a UALLOC to allocate the volumes for that component. **For VS2.03.807 only:** If SAVRAC is specified, CPLPROC will turn on the CTGPROC indicator to indicate that profiles are to be saved and not defined. If the RACF indicator is on in the dictionary record read from the portable data set, FVTPROC will set the CTGFVRON bit to indicate that the RACF indicator should be set.

Module: IDCMP01

Procedure: CTLGPROC, CPLPROC, CLUSPROC, DELTPROC, DUPNPROC

4. a. CTLGPROC issues a UCATLG macro to define the VSAM data set. If the return code is 40, the work area for VSAM catalog management is increased and the UCATLG is reissued. If the return code is 8, a duplicate cluster name exists on the VSAM catalog. CPLPROC builds a CTGPL to locate the catalog entry to determine if the duplicate cluster had a temporary EXPORT done against it or if it is an empty data set. DUPNPROC builds DSATTR, HURBADS and AMDSBCAT CTGFLs to obtain the data set attribute information, the high-used RBA and the AMDSB control block of the data component. If the temporary export flag is not on in either the data or index or the data set is not empty, the IMPORT is not done. If the data set is empty, a check is made to ensure that the data set organization, key length and relative key position in the catalog entry are the same as those which were exported. If any of these factors are different, a message is written and the IMPORT is not done. If **VS2.03.807** is installed and if the data set is empty, a check is made for the INTOEMPTY keyword. If not specified, a message is written and the IMPORT is not done. The maximum LRECL of the cataloged entry is then compared to that of the data set exported. Unless it is greater than or equal to the maximum LRECL of the data set exported, the IMPORT is terminated. Otherwise, control goes to step 4.b. If the temporary export flag is on, CPLPROC builds a CTGPL to delete the duplicate VSAM data set. If ERASE | NOERASE or PURGE | NOPURGE is specified, CPLPROC puts the information in the CTGPL so that VSAM will take the appropriate action. DELTPROC issues a UCATLG macro to delete the object. For **VS2.03.807**, a catalog return code of 110 with a reason code of 4 or 8 will cause a message to be printed indicating that the profile of a RACF-indicated entity could not be deleted. (It could not be found or it was not eligible for deletion.) Then CTLGPROC reissues the UCATLG macro to define the VSAM data set. If the return code from the UCATLG macro is zero, control goes to step 4b. If a recovery volume serial is returned for the define, a UPRINT macro is issued to print it. If the VSAM catalog return code is 140 with a reason code of 40, DALCPROC allocates the volumes required by VSAM catalog management. Then the UCATLG is reissued. If the return code is any non-zero value other than those mentioned previously, CTLPROC issues an error message by building an error conversion table and invoking the UERROR macro.

Module: IDCMP01

Procedure: OPENPROC, RECPROC

   b. If the cluster or alternate index exported had any associated paths defined over it, the catalog entries for these paths were also exported. CLUSPROC processes the catalog information for each path in a manner similar to that described in step 3.a. The PATHENTRY name and password, if any, are passed for the path (R) FVT. The only subparameters of the OBJECTS parameter allowed for path objects are NEWNAME and FILE. If any other subparameter is specified, a new IMPORT message is written and that path is not defined. CTLGPROC issues a UCATLG macro to define each path. If the return code from UCATLG is

nonzero, a message is written by building an error conversion table and invoking the UERROR macro. However,, the IMPORT is not terminated.

5. OPENPROC builds an OPNAGL and issues a UOPEN to open the newly defined VSAM data set. If a password is specified via the OUTFILE or OUTDATASET parameter, this password is passed to UOPEN for use inn building the ACB. Otherwise, the exported master password, if any, is used. RECPROC issues a UCOPY to copy the data from the portable data set to the newly defined VSAM data set.

   When importing a relative record data set, the relative record number of each record on the portable data set is contained in a 4-byte area immediately preceding the record itself. UCOPY processing uses this relative record number in writing the records to the output data set. OPENPROC sets the Export/Import flag in the OPNAGL of the output data set to indicate to UCOPY that this is to be done.

   Following a successful open, RECPROC compares the name specified via the OUTFILE parameter to the name of the object exported. If the compare is unequal, RECPROC builds a CTGPL and CTGFLs and issues a UCATLG macro to loacte the entry type and associations of the name specified via OUTFILE. If the entry type returned is that of a path, RECPROC builds a CTGPL and CTGFL and issues a UCATLG macro to locate the entry name of the pathentry association (alternate index or cluster) and compares the name returned from the Locate to the name of the object exported. If the verification fails, a message is written and the IMPORT is not done.

Module: IDCMP01

Procedure: CLUSPROC

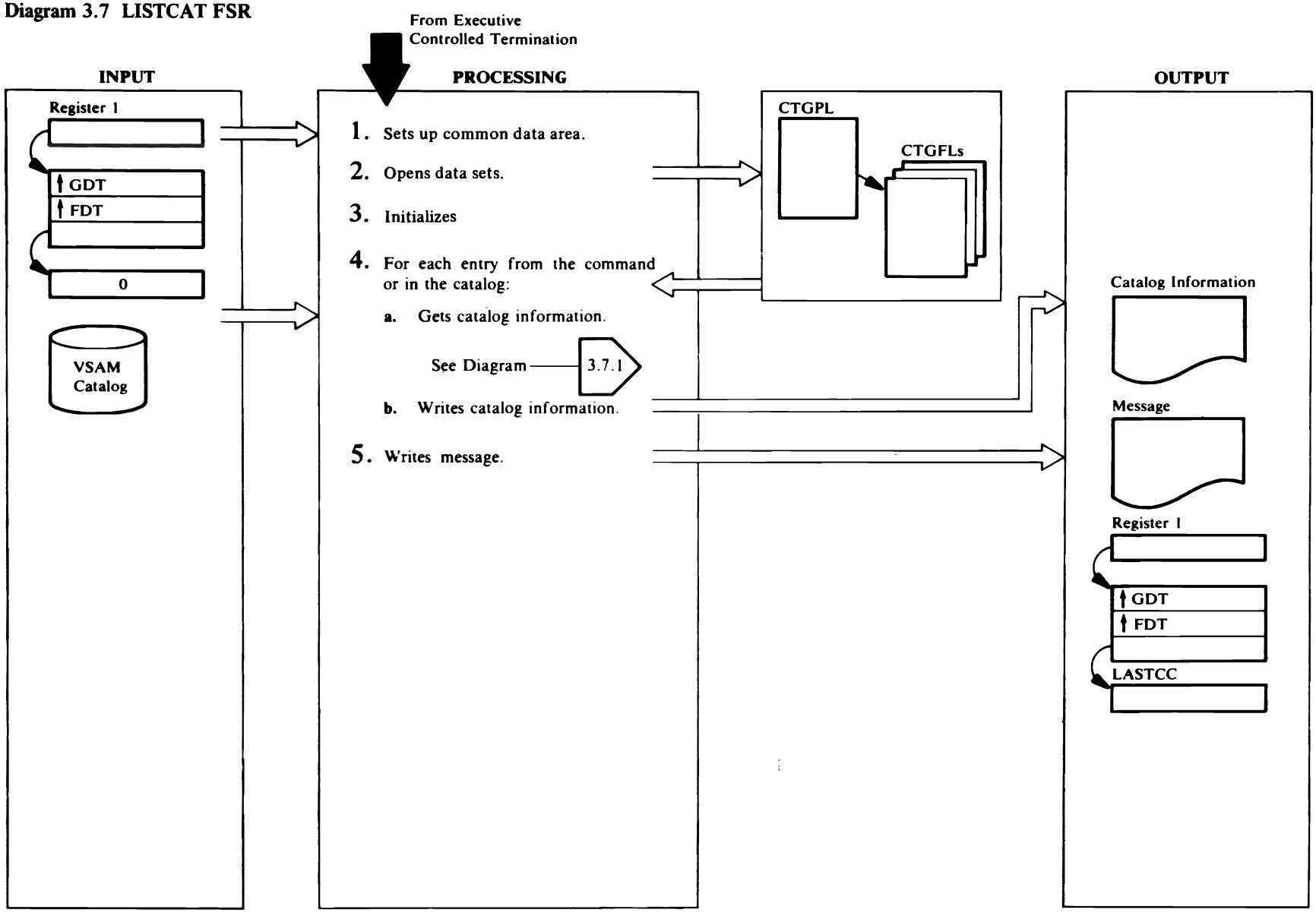
6. CLUSPROC issues a UCLOSE to close the portable data set. If UOPEN allocates a data set, UCLOSE automatically deallocates the data set. IDCMP01 CLUSPROC

Module: IDCMP01

Procedure: ALTRPROC, CPLPROC

7. If INHIBITTARGET was specified when the VSAM data set was exported, the catalog entry must be altered. ALTRPROC builds a CTGFV and a DSATTR CTGFL for the data set attributes field with INHIBITTARGET specified. CPLPROC builds a CTGPL to alter the VSAM data component. CTLGPROC issues a UCATLG macro to alter the VSAM data set to inhibit the VSAM data set. If the VSAM data set has an index component, step 7 is repeated to alter the index component to INHIBITTARGET. After INHIBITTARGET is processed, CLUSPROC issues a UDEALLOC to deallocate the cluster. Control goes to Diagram 3.6, step 2.

Diagram 3.7 LISTCAT FSR



Extended Description for Diagram 3.7

Module: IDCLC01, IDCLC02

Procedure: IDCLC01, IDCLC02

1. Before processing the catalog entries, IDCLC01 links to IDCLC02. IDCLC02 establishes addressibility and initializes an array of 4-byte pointers to point to several different work areas. These work areas are common work areas used by both IDCLC01 and IDCLC02. They are used to store pointers and variables and reside in IDCLC02's automatic storage. The address of the array of pointers is passed back to IDCLC01 in register 15.

Module: IDCLC01

Procedure: INITPROC, DATEPROC, TIMEPROC

2. If OUTFILE is specified, INITPROC builds an OPNAGL and issues a UOPEN to open the alternative output data set. By opening the alternative file first, any LISTCAT error messages appear on the alternative file. If a CATALOG is specified, INITPROC puts the address of the *catname* in the CTGPL to make VSAM open the catalog. If CATALOG is not specified in the LISTCAT command, INITPROC puts the address of 44 blanks in the CTGPL to make VSAM find the catalog and open it.

   If CREATION or EXPIRATION is specified, INITPROC issues a UTIME macro to get the date as an 8-byte microsecond value and a 4-byte packed-decimal value. INITPROC calls DATEPROC and TIMEPROC to calculate the creation and expiration values for the LISTCAT FSR and catalog management to use to determine whether a given entry should be listed. INITPROC issues a warning message if EXPIRATION is specified with an entry type that either has no expiration-date field or contains an expiration-date field that is never initialized.

Module: IDCLC01

Procedure: INITPROC

3. INITPROC issues a UGPOOL macro to obtain storage for the CTGPL, CTGFLs, work areas, and DARGLIST. INITPROC puts the address of a work area for VSAM in the CTGPL. The returned catalog ACB from the UOPEN is put in the CTGPL. Also, if *password* is specified in CATALOG, the address of the *password* is put in the CTGPL. INITPROC determines the number of catalog fields to be obtained for each catalog entry by the specification of NAME, HISTORY, VOLUMES, ALLOCATION, or ALL. Catalog fields are obtained by control blocks named CTGFLs. The following table shows the CTGFLs that are used for each type of catalog entry.

   Note: a PAGESPACE consists of a cluster and data entry.

Module: IDCLC01, IDCLC02

Procedure: ENTPROC, RTEPROC, TIMEPROC, CKDTPROC, ANLTPROC, ANSVPROC, LOCPROC, CDIPROC, AUPROC, VPROC, FPLPROC, ALSPROC, SHORTLST, VOLLIST, LISTPROC

4. If ENTRIES is specified, catalog information is found on each *entryname* in the command. If the *entryname* is a generic name, ENTPROC issues a UCIR macro. The UCIR macro returns a list of names that match the *entryname* except the * has been replaced by a single qualifier. If the *entryname* is unqualified and Access Method Services is invoked interactively with TSO, ENTPROC issues a UQUAL macro to get the fully qualified name which is used instead of the *entryname* specified in the command.

   If LEVEL is specified, ENTPROC issues a UCIR macro to get all the names that match the LEVEL name. one qualifier replaces the * and any number of qualifiers may be added to the end of the name

   If neither ENTRIES nor LEVEL is specified, ENTPROC tests the FDT to determine if Access Method Services is invoked interactively with TSO. If it is, ENTPROC issues a UID macro to get the TSO user's identification. The TSO user's identification is treated like a LEVEL name, and ENTPROC issues a UCIR macro to get all the names that start with the TSO user's identification. If no parameters besides NAME or CATALOG were specified, ENTPROC passes control to SHORTLST, which formats a list of data-set names extracted from the work area filled in by the UCIR macro and issues a UPRINT macro to write the list to the terminal. (The list includes all data-set names that begin with the TSO user's identification and that were found in all catalogs searched.) Processing continues at step 5. If no parameters besides VOLUME or CATALOG were specified, ENTPROC passes control to VOLLIST. After each data-set name, extracted from the UCIR workarea, is located, VOLLIST will format a list of data-set names and for entries that contain volume information, the volume serial numbers. VOLLIST calls LISTPROC to issue a UPRINT macro to write the list to the terminal. (The list includes all data-set names that begin with the TSO user's identification and that were found in all caatalogs searched). Processing continues at Step 5. If parameters besides NAME or VOLUME or CATALOG were specified, processing continues as though Access Method Services were not invoked interactivly with TSO.

Module: IDCLC01, IDCLC02

Procedure: ENTPROC, RTEPROC, TIMEPROC, CKDTPROC, ANLTPROC, ANSVPROC, LOCPROC, CDIPROC, AUPROC, VPROC, FPLPROC, ALSPROC, SHORTLST, VOLLIST, LISTPROC

   If neither ENTRIES nor LEVEL is specified and Access Method Services is not invoked interactively with TSO, information is found for each entry in the catalog.

   a. LOCPROC issues a UCATLG to locate the catalog information for an entry. If a required password is not supplied, VSAM returns the entry type and entry name fields in a work area instead of through the CTGFLs. The catalog ACB is returned the first time information is successfully located in the catalog. LOCPROC saves the catalog ACB and removes the CATACB CTGFL from the list of CTGFLs to be used to locate information on other catalog entries. If only selected names are being listed and CATALOG is not supplied, a bit is set so that VSAM will search all available catalogs. If the selected name is a generic name, the catalog name returned from the UCIR macro is used to locate information. If the entire catalog is being listed, only one catalog is searched for the entire LISTCAT command. VSAM searches the catalogs in the following order: STEPCAT, JOBCAT, Master Catalog, and any user catalog named with the first qualifier of the *entryname*. Diagram 3.7.1 shows getting catalog information in detail.

   b. RTEPROC test the entry type of the catalog entry. If the type is ALTERNATE INDEX, CLUSTER, DATA, INDEX, PATH, or PAGESPACE, CDIPROC formats the information and writes it with a UPRINT macro. If the type is ALIAS, GENERATIONDATAGROUP, NONVSAM, or USERCATALOG, AUPROC formats the information and writes it with a UPRINT macro. If the type is an association of an alternate index, cluster, pagespace, generation data group, or alias, ANLTPROC formats the information and writes it with a UPRINT macro. If the type is SPACE, RTEPROC checks for a date restriction (CREATION specified). If the entry is date-restricted, the calculated creation date is

compared with the time stamp of the volume entry and the DSCB to determine whether to list the entry. If so, VPROC formats the information and writes it with a UPRINT macro. Otherwise processing continues at step 4.a for the next entry. **Note:** Information written for a SPACE entry does not come directly from the catalog because LISTCAT has a special interface with VSAM for all LISTCAT requests. VSAM manipulates information in the catalog to provide the special interface to LISTCAT. If the entry type is a cluster, alternate index, pagespace, generation data group, or alias, RTEPROC determines whether an association—that is, a data entry, index entry, true name entry, or generation data set—is to be listed. If it is, ANSVPROC saves the name of the association and lists the entry. FPLPROC reinitializes the CTGFLs.

The information about the data or index or path is obtained by Control Interval rather than by name. Control returns to step 4.a to locate information about the data or index or path. If RTEPROC was entered at the secondary entry point, RTEASSOC, to locate the associations of an alternate index, cluster, page space, or generation data group, RTEPROC first checks for a NOTUSABLE option request. If NOTUSABLE was requested and the retrieved entry is a data or index entry, a check is made to determine if the entry has been markedd as unusable. If the entry has been marked as unusable, processing continues; otherwise, processing continues at Step 4 for the next entry. RTEPROC then checks for a date restriction (CREATION or EXPIRATION specified). If the entry is date-restricted, CKDTPROC compares the calculated creation and expiration dates with the entry's creation and expiration dates to determine whether to list the entry. If so, control goes to the appropriate format routine to format the information and write it with a UPRINT macro. Otherwise processing continues at step 4.a for the next entry. FPLPROC reinitializes the CTGFLs for the next catalog entry. If the type is not valid, RTEPROC writes a message. Control goes to step 4.a for the next entry. Refer to *OS/VS2 Access Method Services* for a sample listing of LISTCAT output.

Module: IDCLC01, IDCLC02

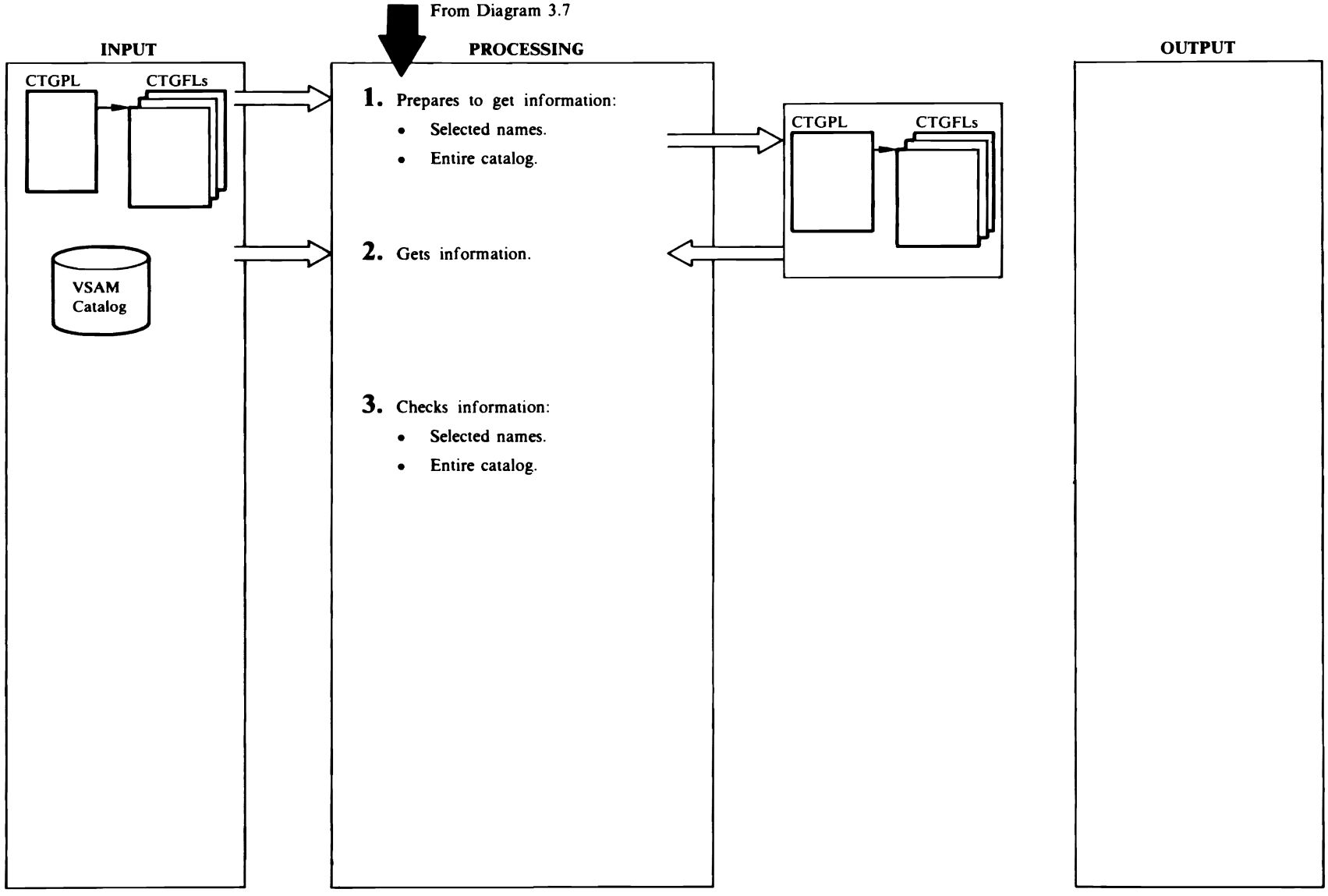
Procedure: IDCLC01, FREESTG

5. IDCLC01 writes a summary of the entries listed and suppressed due to incorrect passwords if Access Method Services is not invoked interactively with TSO. Any storage obtained during the processing of the LISTCATALOG command is released with a UFPOOL macro. IDCLC01 then passes control to FREESTG (in IDCLC02) to free the automatic storage acquired by IDCLC02. IDCLC01 then writes a message containing LASTCC. If an alternative output file was opened by INITPROC, IDCLC01 issues a UCLOSE to close the file. Control goes to Executive Controlled Termination, Diagram 4.1. See below.

   If NAME is specified, INITPROC initializes CTGFLs 2 through 5; if HISTORY, 2 through 9; if VOLUMES, 2 through 11; if ALLOCATION, 2 through 16; if ALL, 2 through 32 (**33 if VS2.03.807**). INITPROC adds the DSDTEXDT CTGFL to the end of the NAME list if EXPIRATION is specified; it adds the DSETCRDT CTGFL to the end if CREATION is specified. INITPROC adds the VOLTSTMP and SPACEHDR CTGFLs to the end of the NAME, VOLUME, and ALLOCATION list if CREATION is specified and volume entries are to be listed. INITPROC adds the DSATTR to the end of the NAME, VOLUME and ALLOCATION list of NOTUSEABLE is specified. INITPROC also adds the CATACB CTGFL to the end of each list if CATALOG is not supplied in the LISTCAT command. If more than one entry type is to be listed, or CREATION or EXPIRATION or NOTUSABLE is specified, INITPROC adds the MULTITYP CTGFL to the beginning of the list of CTGFLs.

CTGFL Name	Entry Type ALIAS	CLUSTER	DATA	INDEX	GDG	NON-VSAM	SPACE	USER-CAT.	ALT. INDEX	PATH	Data in CTGFLS
1. MULTITYP											Identifies multiple catalog types to be listed.
2. ENTYPE	X	X	X	X	X	X	X	X	X	X	Entry type.
3. ENTNAME	X	X	X	X	X	X	X	X	X	X	Entry name.
4. CATTR		X									Indicator for pagespace.
5. NAMEDS	X	X	X	X	X				X	X	DI number and entry type of each association.
6. DSETEXDT		X	X	X	X	X			X	X	Data set expiration date.
7. DSETCRDT		X	X	X	X	X			X	X	Data set expiration date.
8. OWNERID		X	X	X	X	X			X	X	Data set owner.
9. RELCRA	X	X	X	X	X	X	X	X	X	X	VSAM release and catalog recovery information.
10. CATVOL			X	X		X		X			Volume information for data set.
11. VOLDVCHR							X				Volume device character.
12. SPACPARM			X	X							Primary and secondary allocation.
13. HURBADS			X	X							High used RBA.
14. HARBADS			X	X							High allocated RBA.
15. ENTVOL			X	X							Physical description of data set.
16. NOBYTTRK			X								Number of bytes per track on this volume.
17. VOLTSTMP							X				Volume time stamp.
18. SYSEXTDS							X				System allowed extents.
19. NODSPACE							X				Number of data space on volume.
20. NODSET							X				Number of data sets on volume.
21. SPACEHDR							X				Characteristics and statistics of data space.
22. DSDIRECT							X				Data Set directory for a data space.
23. DSPDSCRP							X				Physical description of data space.
24. PASSWALL		X	X	X					X	X	Password (security) information.
25. AMDSBCAT			X	X							AMDSB control block.
26. DSATTR			X	X							Data set attributes and for PAGESPACE, track overflow.
27. BUFSIZE			X	X							Minimum buffer size.
28. LRECL			X	X							Logical record size.
29. GDGLIMIT					X						Number of generations for this GDG.
30. GDGATTR					X						GDG attributes.
31. RGATTR									X	X	AIX and PATH attributes.
32. EXCPEXIT			X	X							Exception exit module name.
For users without VS2.03.807:											
33. CATACB											Catalog ACB address.
For users with VS2.03.807:											
33. SECFLADS		X	X X							XX	RACF security flag.
34. CATACB											Catalog ACB address.

Diagram 3.7.1 LISTCAT FSR – Gets Information



Extended Description for Diagram 3.7.1

Module: IDCLC01

Procedure: ENTPROC, GNXTPROC

1. If only selected names are being listed, control goes to step 1.a; if the entire catalog is being listed, control goes to step 1.b.

   a. ENTPROC puts the address of the *entryname* in the CTGPL. If only SPACE information is to be listed, ENTPROC treats the *entryname* as a six-character volume serial number and extends it to 44 characters by padding on the right with binary zeros. ENTPROC puts the address of the volume serial number in the CTGPL. If *password* is supplied with CATALOG, ENTPROC puts the address of the *password* in the CTGPL. If there is no *password* supplied with CATALOG, and there is a *password* specified with the *entryname*, ENTPROC puts the address of the *password* in the CTGPL. If there is no *entryname* to be listed, control goes to Diagram 3.7, step 4.

   b. GNXTPROC sets the CTGPL to indicate that each catalog entry is to be located by the catalog index rather than by a specific name. For the first entry, GNXTPROC puts the address of 44 blanks in the CTGPL as a starting key in the catalog search for the first catalog entry. After the first entry, GNXTPROC adds one to the key—which is the previously retrieved entry name—to make the new key higher in the collating sequence than the old key.

Module: IDCLC02

Procedure: LOCPROC

2. LOCPROC issues a UCATLG macro with the CTGPL and CTGFLs to locate catalog information about the entry. If the return code indicates a lack of workspace, LOCPROC gets more workspace and tries the UCATLG once more. IDCLC01 LOCPROC

Module: IDCLC01

Procedure: ENTPROC, CKDTPROC, GNXTPROC

3. If only selected names are being listed, control goes to step 3.a; if the entire catalog is being listed, control goes to step 3.b.

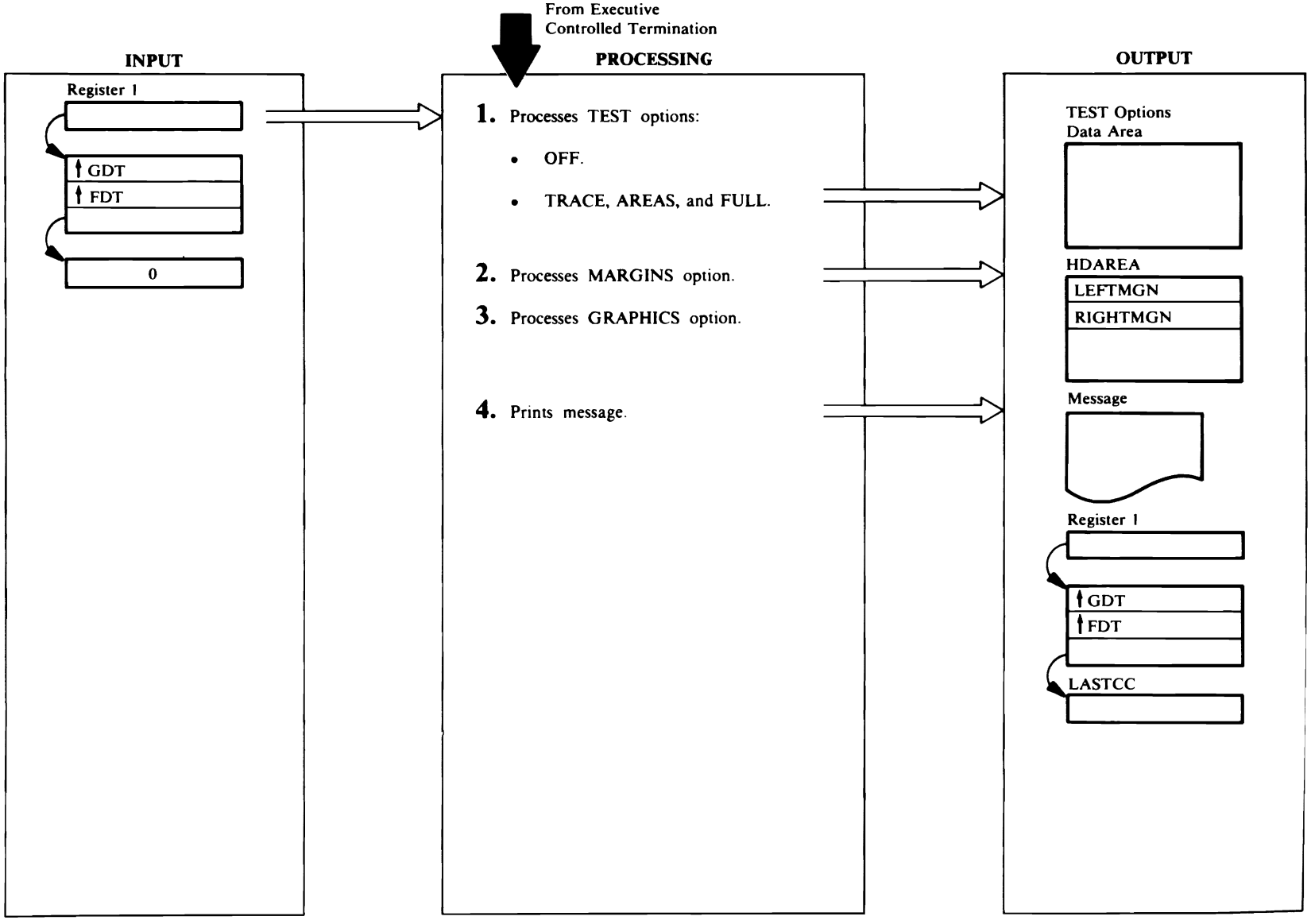
   a. ENTPROC compares the type of entry information returned to the type of information requested in the LISTCAT command. If the entry type matches the type requested in the command, ENTPROC then checks for a date restriction (CREATION or EXPIRATION specified). If the entry is date-restricted, CKDTPROC compares the calculated creation and expiration dates with the entry's creation and expiration dates to determine whether to list the entry. If so, processing continues; otherwise processing continues at Diagram 3.7, step 3.a, for the next *entryname* in the LISTCAT command. If NOTUSABLE was requested and the retrieved entry is a data or index entry, a check is made to determine if the entry has been marked as unusable. If the entry has been marked as unusable, processing continues at Diagram 3.7, step 3.b. Otherwise, processing continues at Diagram 3.7, step 3.a., for the next entryname in the LISTCAT command.

   If the entry type doesn't match the type requested in the command, and the entry is for a cluster, alternate index, pagespace, or generation data group, processing continues at Diagram 3.7, step 3.b.

   If the entry type does not match the type requested in the command and the entry is not a cluster, alternate index, pagespace, or generation data group, or the entry is a cluster or an alternate index and the type specified is not data, index, or path, ENTPROC writes a message, but does not list the entry. If the UCATLG return code is non-zero, ENTPROC also writes a message. Control goes to Diagram 3.7, step 3.a for the next *entryname* in the LISTCAT command.

   b. GNXTPROC saves the name of the retrieved entry to use as a key in locating information for the next entry in the catalog. If the return from the UCATLG macro is zero, control goes to Diagram 3.7, step 3.b. If the return code from UCATLG indicates password verification failure or lack of workspace, GNXTPROC writes a message and control goes to Diagram 3.7, step 3.a for the next entry in the catalog. GNXTPROC checks for end-of-file and unrecoverable errors. When end-of-file or an unrecoverable error is encountered, control goes to Diagram 3.7, step 4 to terminate the LISTCAT command.

Diagram 3.8 PARM FSR



Extended Description for Diagram 3.8

Module: IDCPM01

Procedure: TESTPARM, TESTSAVE

1. If the address of the dump routine is in GDTDBG, a TEST option is currently in effect. TESTPARM frees the Debugging Aids Historical Data Area whose address is in GDTDBH, and it sets GDTDBH to zero.
   a. If the TEST keyword is followed by OFF, TESTPARM deletes the dump routine, IDCDB01, whose address is in GDTDBG, and it sets GDTDBG to zero. Control goes to step 2.
   b. If the TEST keyword is followed by TRACE, AREAS, or FULL, TESTPARM issues a UGSPACE macro to obtain a new Test Option Data Area. TESTSAVE puts the information from the FDT in the new Test Option Data Area. If GDTDBG is zero, TESTPARM issues the ULOAD macro to load dump routine. TESTPARM puts the address of the dump routine in GDTDBG. Although the trace tables record execution since Access Method Services invocation, the earliest time a trace table or dump can be printed is in the Executive prior to the second call to the Reader/Interpreter. This is because the TEST option is not on until the PARM command has been completed.

Module: IDCPM01

Procedure: MARGPARM

2. MARGPARM checks the margins for validity. The left margin must be less than the right margin. If the margins are invalid, MARGPARM sets the left margin to 2 and the right margin to 72, which are the Access Method Services default margins. MARGPARM puts the margin values in the first two halfwords of the Reader/Interpreter Historical Data Area. The margins can be changed with any PARM command, but the margins are most likely to be specified only at Access Method Services invocation via the PARM field of the EXEC JCL statement. IDCPM01 MARGPARM

Module: IDCPM01

Procedure: GRPHPARM

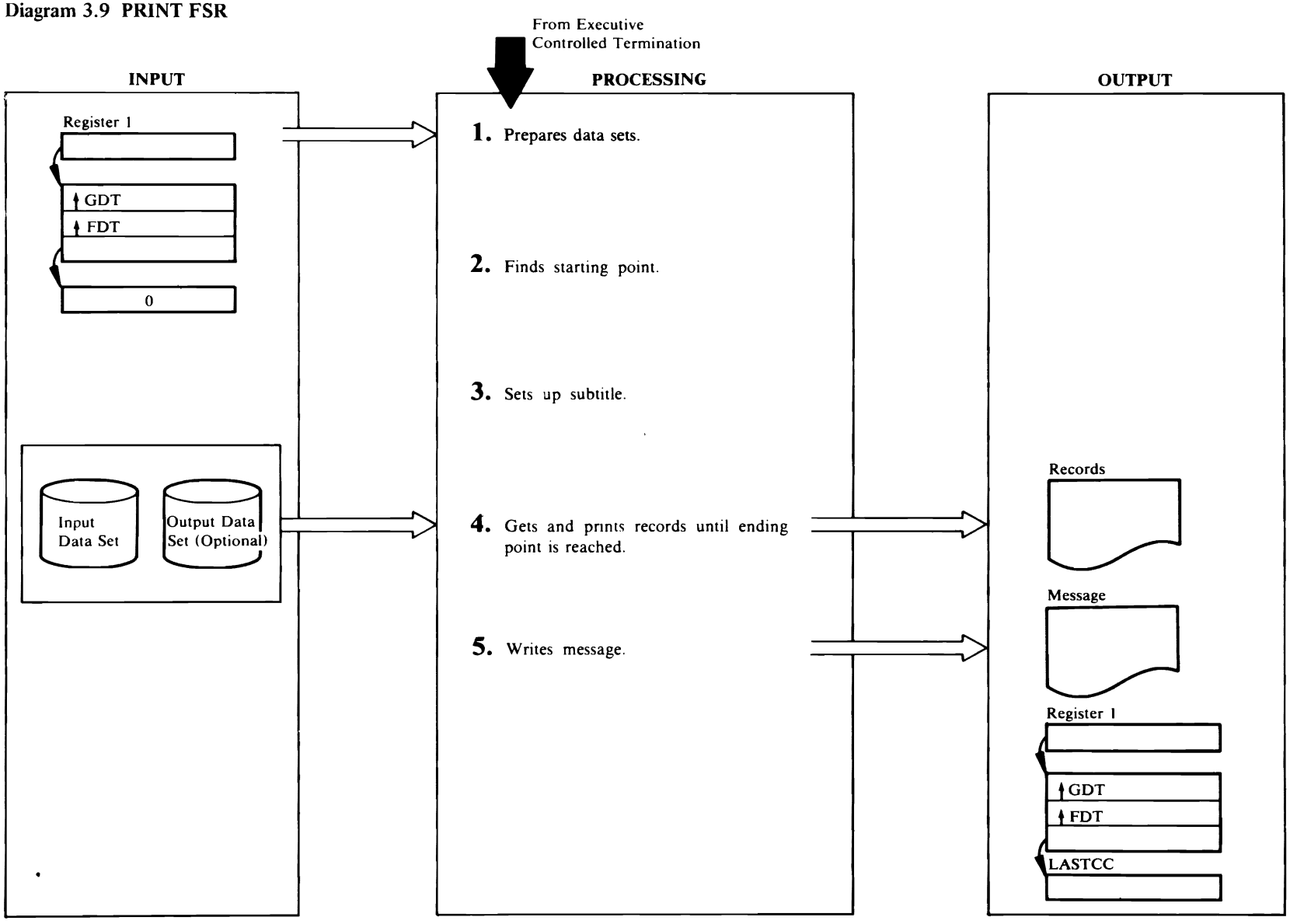
3. GRPHPARM puts the GRAPHICS parameter (CHAIN or TABLE) in a Text Processor Print Control Argument list. GRPHPARM issues a UREST macro for the Text Processor to use the new chain or table with Access Method Services output. The CHAIN parameter specifies one of several graphic character sets available. However, the CHAIN parameter does not specify a particular physical type chain. IDCPM01 GRPHPARM

Module: IDCPM01

Procedure: IDCPM01

4. IDCPM01 prints a message containing LASTCC. Control goes to Executive Controlled Termination, Diagram 4.1.

Diagram 3.9 PRINT FSR



Extended Description for Diagram 3.9

Module: IDCPR01

Procedure: IDCPR01

1. IDCPR01 tests the FDT to see if the input data set name needs to be qualified. Data set names are qualified if Access Method Services is invoked interactively with TSO. If the data set name needs qualification, IDCPR01 issues a UQUAL macro and uses the returned data set name for the remainder of the PRINT FSR. IDCPR01 builds an OPNAGL for the input data set. If the PRINT command specifies a FROMKEY or TOKEY parameter, IDCPR01 opens the data set for key sequence record retrieval. If FROMADDRESS or TOADDRESS is specified, IDCPR01 opens the data set for sequential record retrieval. If the PRINT command specifies FROMNUMBER or TONUMBER, IDCPR01 opens the data set for keyed processing. IDCPR01 puts any ENVIRONMENT parameters in the OPNAGL. The input data set can be a VSAM catalog. If INDATASET is specified, IDCPR01 puts the data set name in the OPNAGL. This causes the data set to be dynamically allocated by the UOPEN macro. IDCPR01 issues a UOPEN macro to open the input data set. If an output data set is specified with the OUTDDVAL keyword, IDCPR01 builds an OPNAGL and issues a UOPEN for the output data set. If the return code from a UOPEN macro is non-zero, IDCPR01 writes a message and terminates the PRINT command.

Module: IDCPR01

Procedure: DELIMSET

2 DELIMSET performs additional validity checking to verify that From/To parameters are consistent with data set organization. If the parameter is invalid, an error message is written. Checks are made for invalid use of

Module: IDCPR01

Procedure: TEXTPSET

FROMADDRESS | TOADDRESS with RRDS and FROMNUM | TONUM with KSDS

If FROMNUMBER is specified, DELIMSET issues a UPOSIT macro to position to the starting relative record number. If SKIP is specified for a VSAM relative record data set, DELIMSET issues a UPOSIT to position to the next relative record number beyond the skip count. A VSAM relative record data set is printed in relative record number order.

If FROMKEY is specified, DELIMSET issues a UPOSIT macro to position to the starting key. If FROMADDRESS is specified, DELIMSET issues a UPOSIT macro to position to the starting address. If SKIP is specified, DELIMSET issues as many UGET macros as there are records to skip. The way the data set is opened determines how the records are skipped. Any data set opened as an ESDS causes records to be printed in chronological order. A keyed data set opened as a KSDS causes records to be printed in key-sequence order. If no starting point is specified, the starting point is the first record in the input data set.

3. If print has not been invoked interactively via TSO, TEXTPSET formats a subtitle line with static text and the input data set name from the IOCSTR. TEXTPSET issues a UPRINT macro to get the static text and insert it into the buffer in which the subtitle line is being built. No printing is done with this UPRINT macro. TEXTPSET issues a UESTA macro to give the subtitle to the Text Processor.

Module: IDCPR01

Procedure: IDCPR01

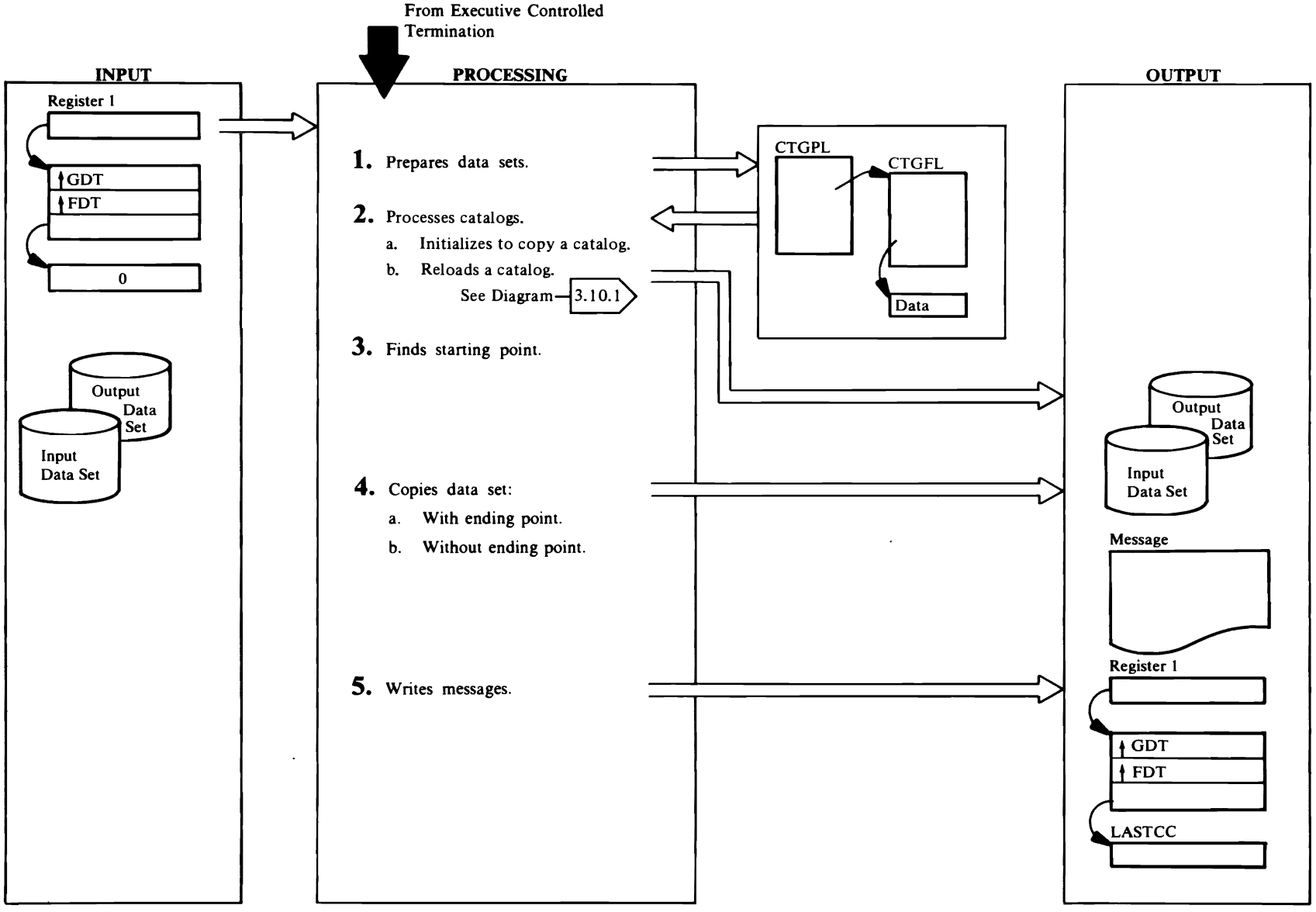
4. The following steps are repeated until the ending point in the input data set is found. If TOKEY is specified, IDCPR01 calculates the key location in the record from information in the IOCSTR. Retrieving records stops when the key in the input record is higher than the value in TOKEY. If TOADDRESS is specified, printing stops when the Relative Byte Address returned by the UGET macro equals the value supplied by TOADDRESS. If COUNTVAL is specified, printing stops when the number of records printed equals the number supplied by COUNTVAL. If TONUMBER is specified, retrieving and printing stops when the relative record number of the input record is higher than the TONUMBER value. If COUNT is specified for a VSAM relative record data set, printing stops when the number of valid relative record slots printed plus the number of invalid slots bypassed exceeds the value supplied by COUNT. If no ending point is specified, printing stops when the last record of the input data set is printed. If no ending point is specified, printing stops when the last record of the input data set is printed.

   a. IDCPR01 issues a UGET to obtain a logical record. If the return code from the UGET macro is non-zero, IDCPR01 checks the return code for a recoverable error. The recoverable errors are duplicate keys, records out of sequence, invalid length records, and I/O errors in the data of a VSAM data set. After a non-recoverable error or 4 recoverable errors, printing stops.

   b. IDCPR01 issues a UPRINT to print the logical record just obtained. A minimum of 3 lines is printed for each logical record from the input data set. The first line printed contains the record identification: key, address, sequence number (nonVSAM except ISAM) or relative record number. The relative record number is printed for a relative record data set and indicates the slot number. Unused slots will be indicated by missing numbers. The second line is blank. The third and following lines contain the logical record from the input data set. The format of the logical records depends on whether HEX, CHARACTER, or DUMP was specified in the command. If an output data set is specified with the OUTDDVAL keyword, IDCPR01 prints the records on that output data set. If the return code from the UPRINT macro is 12 or greater, IDCPR01 will terminate processing: there is no checking for recoverable errors.

5. IDCPR01 writes a message with LASTCC to SYSPRINT. IDCPR01 issues a UCLOSE macro to close the input data set and any output data set other than SYSPRINT. If UOPEN dynamically allocated a data set, UCLOSE deallocates it. SYSPRINT is not closed. Control returns to Executive Controlled Termination, Diagram 4.0.

Diagram 3.10 REPRO FSR



Extended Description for Diagram 3.10

Module: IDCRP01

Procedure: IDCRP01

1. IDCRP01 tests the FDT to see if the input and output data set names should be qualified. Data set names are qualified if Access Method Services is invoked interactively with TSO. If the data set names need qualification, IDCRP01 issues a UQUAL macro for each data set name and uses the returned data set name for the remainder of the REPRO FSR. IDCRP01 builds an OPNAGL for the input data set. If FROMKEY or TOKEY is specified, IDCRP01 opens the input data set for key sequence processing. If FROMADDRESS or TOADDRESS is specified, OPNAGL opens the input data set for sequential record retrieval. If FROMNUMBER or TONUMBER is specified, IDCRP01 opens the input data set for key sequence processing. IDCRP01 also builds an OPNAGL for the output data set, and it puts any ENVIRONMENT parameters in the OPNAGL. If REUSE or REPLACE is specified, IDCRP01 sets the OPNAGL for the output data set to reflect these parameters. UOPEN will open the output data set with the reset option. If INDATASET or OUTDATASET is specified, IDCRP01 puts the data set name in the OPNAGL. This causes the data set to be dynamically allocated by the UOPEN macro. It also issues one UOPEN macro that opens both the input and output data sets. If the return code from the UOPEN macro is non-zero, IDCRP01 writes a message on SYSPRINT and terminates the REPRO command. Following the open of both data sets, IDCRP01 checks for a nonrelative-record input data set together with a nonempty relative record output data set. If this error condition is detected, a message is written on SYSPRINT and the REPRO command is terminated. IDCRP01 IDCRP01

Module: IDCRP01

Procedure: IDCRP01, VERIFYC, RECOVCAT

2. Both the input and the output data sets must be VSAM catalogs in order for the input data set to be copied as a catalog. IDCRP01 tests the input and output data sets by issuing a UCATLG macros to locate information about each data set. A data set is a catalog if the Control Interval Number is X'000002' and the entry type is 'C' or if the entry type is 'U'. Both the control-interval and type 'C' conditions are needed because the object may describe itself in its own catalog—Control Interval number X'000002'entry type is 'C'—or may be a user catalog in the master catalog—entry type 'U'. If the output data set is a catalog, CATRELSW is turned on to initialize for catalog reload. If the input data set is also a catalog, CATRELSW is turned off and CATCOPSW is turned on to initialize for copy catalog. IDCRP01 indicates in the IOCSTR that catalogs are to be copied. If data-set delimiters were specified for either the input or the output data set for catalog reload or copy catalog, a message is printed and termination processing continues with step 5.

   If neither the input nor the output data set is a catalog, processing continues with step 3.

   If catalog copy is initiated, each catalog is checked for the recoverable attribute. An error message is issued if either the source or the target catalogs are recoverable and the REPRO command is terminated. otherwise, processing continues with step 3.

   If catalog reload is initiated, processing continues on diagram 3.10.1.

Module: IDCRP01

Procedure: DELIMSET

3. DELIMSET performs additional validity checking to verify that FROM/TO parameters are consistent with input data set organization. If the parameter is invalid, an error message is written. Checks are made for invalid use of FROMADDRESS | TOADDRESS with relative-record data set and FROMNUM | TONUM with key-sequenced data set. If FROMKEY is specified, DELIMSET issues a UPOSIT macro to position to the starting key. If FROMADDRESS is specified, DELIMSET issues a UPOSIT macro to position to the starting address. If FROMNUMBER is specified, DELIMSET issues a UPOSIT macro to position to the starting relative record number. If SKIP is specified for a VSAM relative-record data set, DELIMSET issues a UPOSIT macro to position to the next relative-record number beyond the skip count. If SKIP is specified for a key-sequenced or entry-sequenced data set, DELIMSET issues as many UGET macros as there are records to skip. The way the data set is opened determines how the records are skipped. Any input data set opened as an ESDS causes records to be read in chronological order. A keyed data set opened as a KSDS causes records to be read in key-sequence order. If no starting point is specified, the starting point is the first record in the input data set.

   When copying from a non-relative-record data set into an empty relative-record data set, records are copied into consecutive relative-record locations. When copying from one relative-record data set to another, records are placed in the same slot in the output data set as they were in the input data set.

Module: IDCRP01

Procedure: IDCRP01

4. a. If an ending point other than the end of the input data set is specified by the TOKEY, TOADDRESS, or COUNT keywords, the following steps are repeated until the ending point is found. If TOKEY is specified, IDCRP01 calculates the key location in the record from information in the IOCSTR. Retrieving records stops when the key in the input record is higher than the value in TOKEY. If TOADDRESS is specified, copying stops when the Relative Byte Address returned by the UGET macro equals the value supplied by TOADDRESS. If COUNTVAL is specified, copying stops when the number of records copied equals the number supplied by COUNTVAL. If TONUMBER is specified, copying stops when the relative-record number of the input record is higher than the TONUMBER value. If COUNT is specified for a VSAM relative-record data set, copying stops when the number of valid relative-record slots copied plus the number of invalid slots bypassed exceeds the value supplied by COUNT.

      - IDCRP01 issues a UGET macro to obtain a logical record from the input data set. If the return code from the UGET is non-zero, It also checks the return code for a recoverable error. The recoverable errors are duplicate keys, records out of sequence, invalid length records, and I/O errors in the data of a VSAM data set. After a non-recoverable error or four recoverable errors, copying stops.
      - IDCRP01 issues a UPUT to write the logical record to the output data set. If the return code from the UPUT macro is non-zero, IDCRP01 checks the return code for a recoverable error. After a non-recoverable error or four recoverable errors, copying stops.

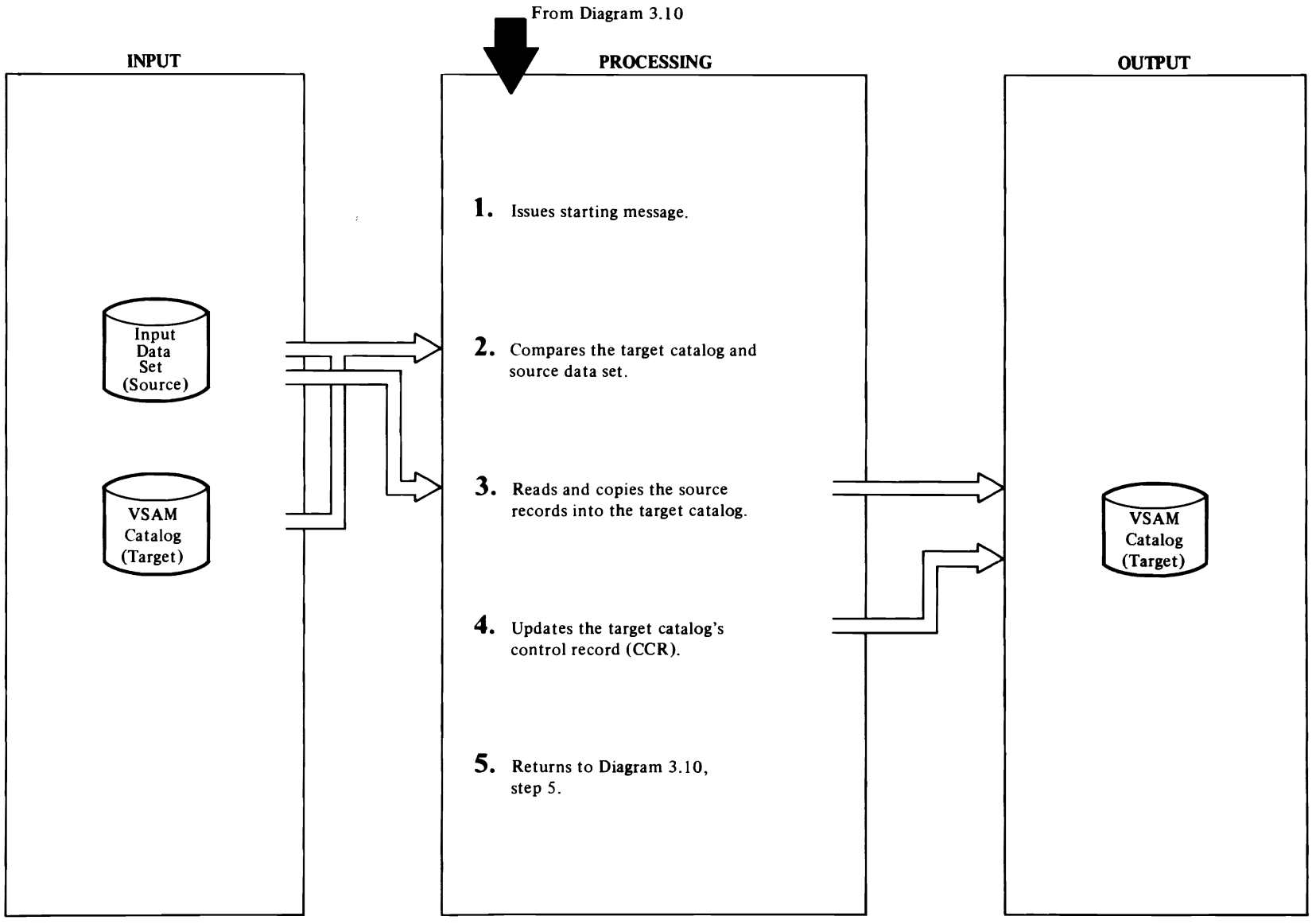
   b. If no ending point is specified in the REPRO command, IDCRP01 issues a UCOPY macro to copy the input data set to the last record. IDCRP01 IDCRP01

Module: IDCRP01

Procedure: IDCRP01

5. IDCRP01 writes a message with LASTCC to SYSPRINT. It also closes the input and output data sets with one UCLOSE macro. If UOPEN dynamically allocated a data set, UCLOSE deallocates it. Control returns to Executive Controlled Termination, Diagram 4.0.

Diagram 3.10.1 REPRO FSR – Catalog Reload



Extended Description for Diagram 3.10.1

Module: IDCRP01

Procedure: IDCRP01

1. The message says that catalog reload had begun.

Module: IDCRP01

Procedure: CATRELOD

2. Additional checks are made at this time by using data from the first 10 records of the input and output data sets. If the data set names do not match, a message is issued, processing is set for termination, and further checks are made. Termination also occurs if the input data set record format does not match a VSAM catalog record format, if there is insufficient space in the output data set, and if the volume serial numbers or the device types do not match. Messages are issued for the corresponding errors. IDCRP01 CATRELOD

Module: IDCRP01

Procedure: CATRELOD, SORSREAD, TARGREAD, GETPAIR, DUMPIT, TRUENAME, CATRANS, CONVRTCI, CATCOMP

3. When all the checks are satisfied, the unloaded catalog is copied into the output data set. Each record is read from the input data set and translated. It is then compared to the target catalog.
   - If a record existed on both backup and target catalogs, the translated backup updates the target.
   - If a record existed only on the backup, then this record is inserted into the target catalog.
   - If a record existed only on the target catalog, then it is processed in one of two ways.
     a. If the target record is a true name record, then it is deleted.
     b. If the target record is a low key range record, then it is made a catalog free record and placed on the free chain.
   - In both cases where the keys are not equal, differences in true name entries between the backup and target catalogs are checked.
     a. If a target true name record exists without a corresponding backup or vice versa, then a message is printed indicating this, provided that not more than 100 messages have been issued. A warning return code of 4 is attached to the message.
     b. At the 101th discrepancy, a message is issued saying that comparison is terminated. The only discrepancies to be printed afterwards will be for volume entries.

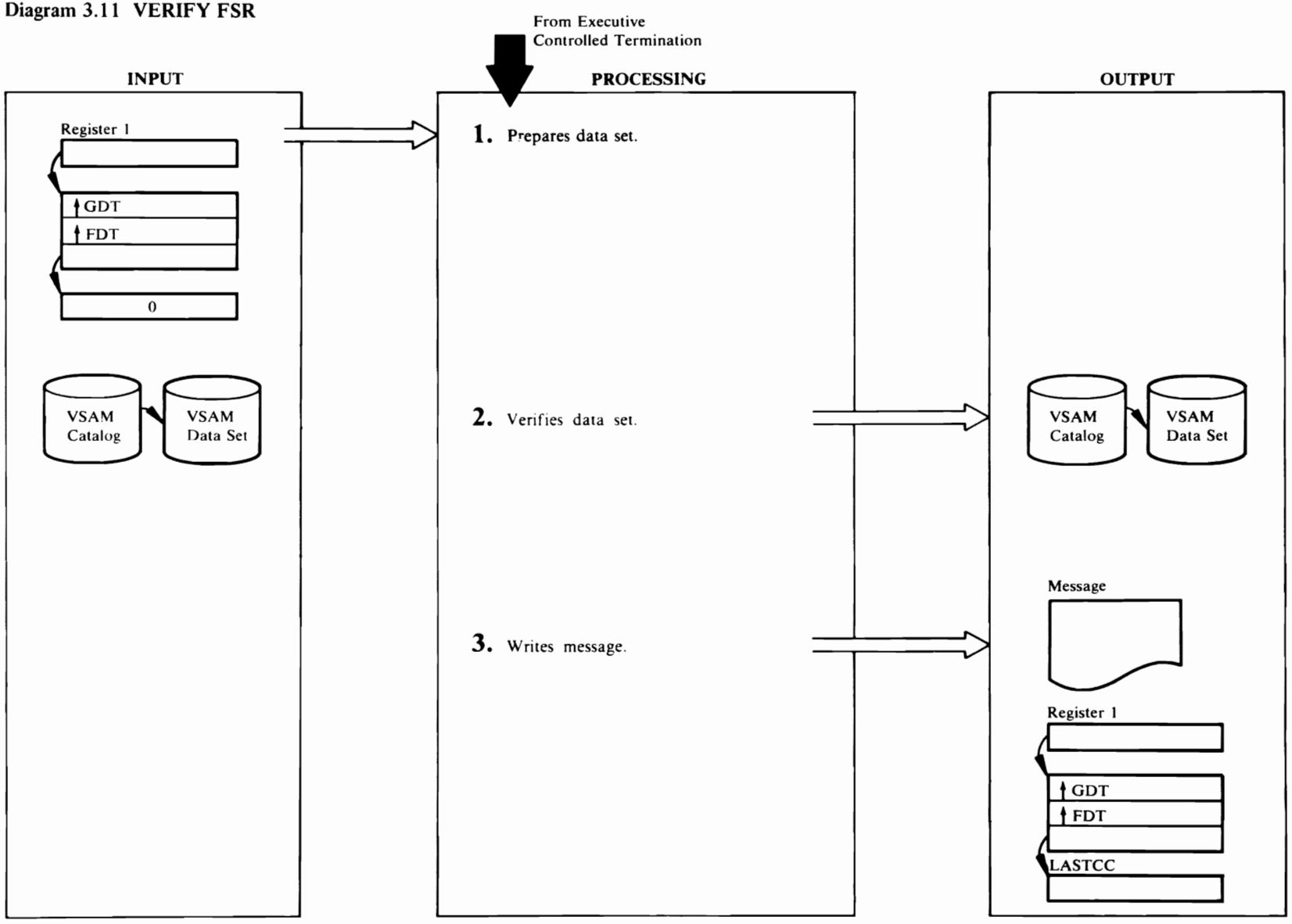
Module: IDCRP01

Procedure: CATRELOD

4. After both backup and target records have been processed sequentially by key to the end-of-file, one more record needs to be updated.
   - The catalog free chain pointers are counted and updated. The RBA fields are cleared so they will be correct for the next open of the catalog and the updated record is written back.

   The number of records copied is the number of backup records read if catalog reload has taken place; otherwise, it is the number of output records written.

Diagram 3.11 VERIFY FSR



Extended Description for Diagram 3.11

Module: IDCVY01

Procedure: OPENPROC, IDCVY01

1. IDCVY01 tests the FDT to see if the VSAM data set name should be qualified. Data set names are qualified if Access Method Services is invoked interactively with TSO. If the data set name needs qualification, IDCVY01 issues a UQUAL macro and uses the returned data set name for the remainder of the VERIFY FSR. If FILE is specified, OPENPROC puts the address of the DD name in the OPNAGL. If DATASET is specified, OPENPROC puts the data set name in the OPNAGL. This causes the data set to be dynamically allocated by the UOPEN macro. OPENPROC builds an OPNAGL to open the VSAM data set for control interval update processing. A UOPEN macro is issued to open the data set. If the open was not successful, LASTCC is set to 12, and control goes to step 3.

Module: IDCVY01

Procedure: IDCVY01, TERMPROC

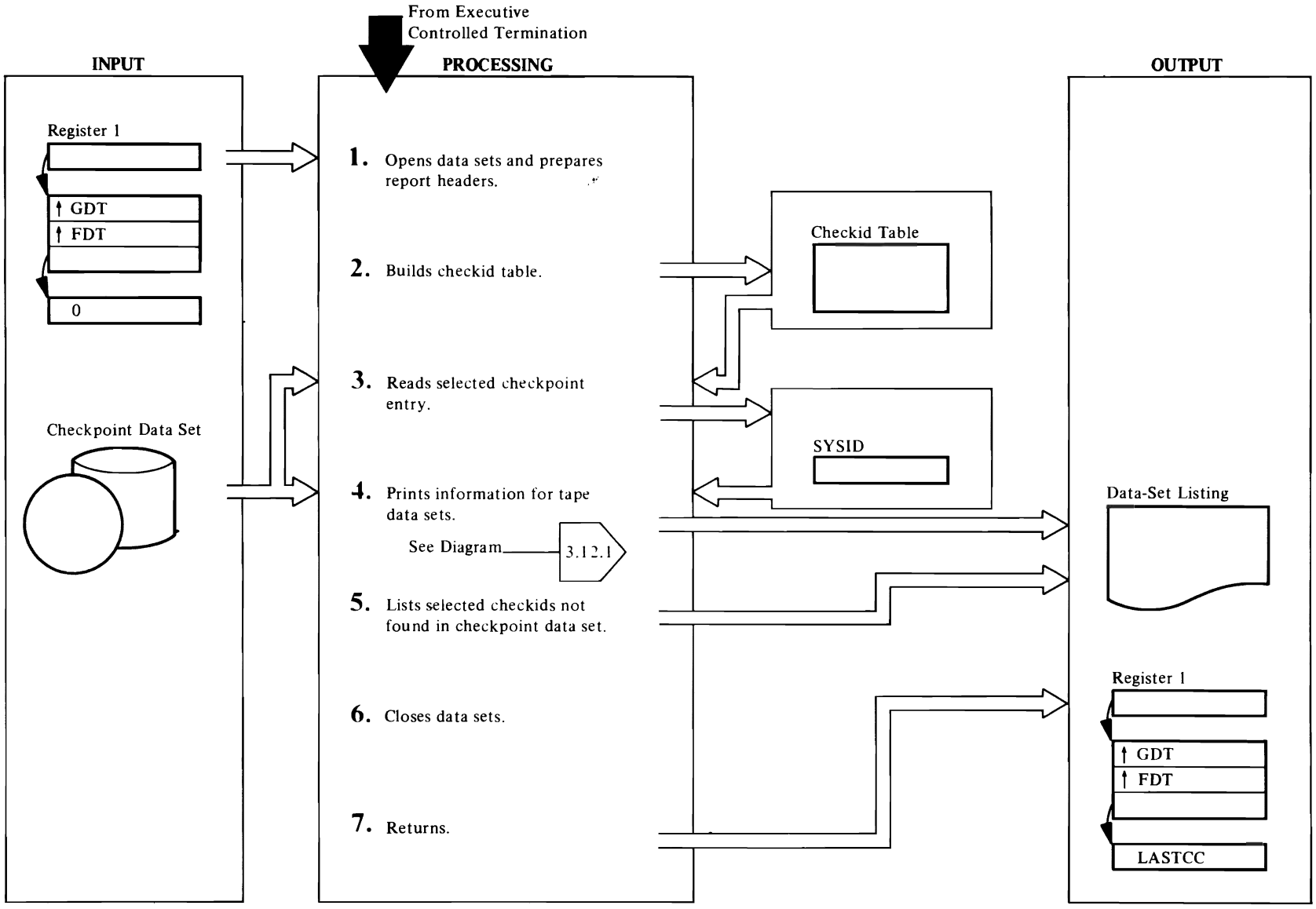
2. IDCVY01 issues a UVERIFY macro to verify the data set. TERMPROC issues a UCLOSE macro to close the data set. If the close was not successful, LASTCC is 4.

Module: IDCVY01

Procedure: IDCVY01

3. IDCVY01 prints a message containing LASTCC. Control goes to Executive Controlled Termination, Diagram 4.0.

Diagram 3.12 CHKLIST FSR



Extended Description for Diagram 3.12

Module: IDCCK01

Procedure: HSKGPROC

1. If OUTFILE was specified, HSKGPROC builds the OPNAGL for the alternate output file and issues a UOPEN macro to open it. HSKGPROC builds the OPNAGL for the INFILE checkpoint data set and issues a UOPEN macro to open it. The report subtitles are retrieved from IDCTSCK0 with a UPRINT macro, the checkpoint data set's name in the IOCSTR is put into the subtitle, and a UESTA macro is issued to prepare the page control table. during the merge phase.

Module: IDCCK01

Procedure: BUILDTAB

2. If CHECKID was specified, BUILDTAB computes the number of bytes of storage required for the checkid table. The table consists of a 2-byte count field (IDCOUNT) and 17-byte table entries. The first byte of a table entry (IDFOUND) indicates whether the checkid indicated in the remaining 16 bytes was found in the checkpoint data set. The number of bytes required for the table is 2 (for IDCOUNT), plus 17 times CHKIDCNT (the number of checkids) from the FDT. Space is acquired with a UGPOOL macro, and the checkids are extracted from the FDT and placed in the checkid table. IDCOUNT is set equal to CHKIDCNT.

   Each entry in the table is matched with each of the other entries. When a duplicate entry is found and IDFOUND for that entry indicates that the checkid was found, the duplicate is printed with a UPRINT macro.

   If CHECKID wasn't specified, a UGPOOL macro is issued for IDCOUNT only, and IDCOUNT is set to -1.

Module: IDCCK01

Procedure: CHRPROC, GETCHR

3. CHRPROC invokes GETCHR to get the next CHR record from the checkpoint data set. GETCHR reads the checkpoint data set until it finds a 400-byte record with the CHR ID '$$/%@/$CHR%@/$//%'. CHRPROC determines whether the user specified this CHR record for processing.

   If IDCOUNT is greater than zero, CHRPROC invokes GETCHR to search the checkid table for a match. GETCHR searches until it finds a match or encounters end-of-file in the checkpoint data set. End-of-file signals the end of CHKLIST processing. When a match is found, IDFOUND is set to indicate it and IDCOUNT is decremented by one. The found checkid is printed with a UPRINT macro.

   If IDCOUNT is less than zero, every CHR is to be processed, and each checkid is printed with a UPRINT macro.

Module: IDCCK01

Procedure: DSDRPROC, DSDRVOLS, GETNEXT, DSDRLIST

4. IDCCK01 invokes DSDRPROC to process the DSDR records for the checkpoint entry whose CHR was just processed (in step 3). GETNEXT reads logical DSDR records and DSDRPROC examines them until DSDRPROC finds a type 1 DSDR record for a tape data set. The dsname, ddname, unit type, and JFCB information are extracted and listed. If a JFCBX is indicated to be present (JFCBEXAD not equal to 0), DSDRVOLS is invoked to process the type 2 DSDR records. When all type 1 DSDR records and related type 2 DSDR records for tape data sets have been processed, DSDRPROC returns to IDCCK01.

   Steps 3 and 4 are repeated until all selected checkids have been processed or end-of-file has been reached in the checkpoint data set.

Module: IDCCK01

Procedure: IDCCK01

5. If CHECKID was specified, the checkid table is searched for entries whose IDFOUND bytes still indicate the checkid wasn't found. A UPRINT macro is issued to print each of these checkids in a "SELECTED CHECKID NOT FOUND" message.

Module: IDCCK01

Procedure: IDCCK01

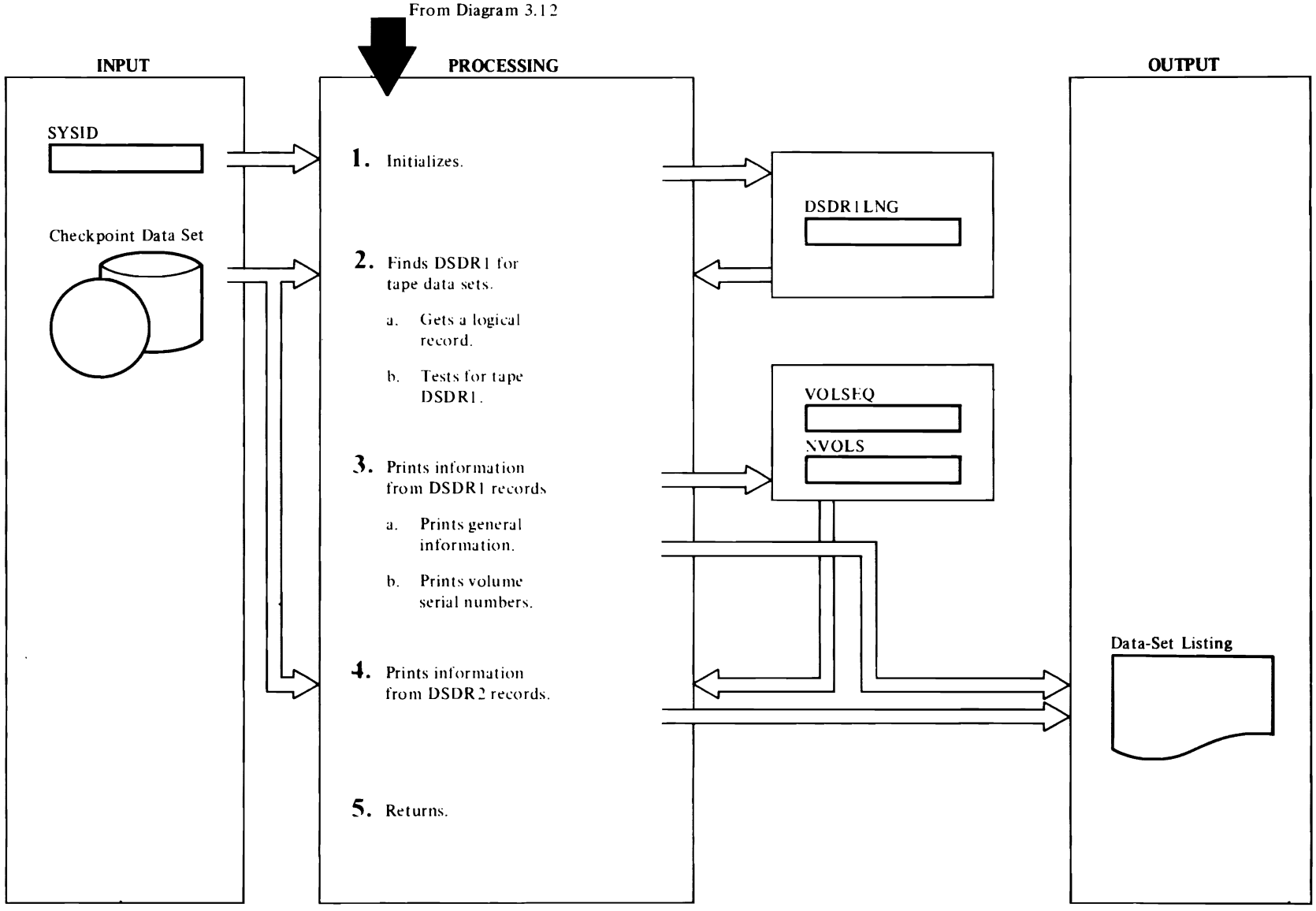
6. Opened data sets are closed with a UCLOSE macro, and acquired space is freed with a UFPOOL macro.

Module: IDCCK01

Procedure: IDCCK01

7. The highest condition code in LASTCC is returned to the Executive by way of a UEPIL macro.

Diagram 3.12.1 CHKLIST FSR – Prints Information



Extended Description for Diagram 3.12.1

Module: IDCCK01

Procedure: DSDRPROC

1. If the VS1 bit is on in the SYSID, DSDR1LNG is set to 191 (length of VS1 type 1 DSDR record); otherwise, DSDR1LNG is set to 195 (length of VS2 type 1 DSDR record). The logical record pointer (LOGRECP) is set to zero.

Module: IDCCK01

Procedure: GETNEXT, DSDRPROC

2. a. Upon entry to GETNEXT, if LOGRECP isn't equal to the address of the input buffer, GETNEXT reads the next block from the checkpoint data set. If the block is 400 bytes long and is a DDNT, type 1 DSDR, or type 2 DSDR record, LOGRECP is set to the address of the input buffer. If the 400-byte record is a CHR record, end-of-entry condition is set, and the next read for a CHR record is inhibited.

   Upon entry to GETNEXT, if LOGRECP is equal to the address of the input buffer, the processing described above is bypassed, and LOGRECP is incremented by SIZE (length of the previously read logical record).

   b. If the record returned by GETNEXT is a DDNT record, LOGRECP is set to zero, and step a is repeated. If the record is a type 2 DSDR record, SIZE is set equal to the length of a type 2 DSDR record, and step a is repeated. For any other record except a type 1 DSDR record, end-of-entry condition is set. If the record is a type 1 DSDR record, SIZE is set to DSDR1LNG. Type 1 DSDR records for data sets other than tape (UCBTBYT3 not equal to X'80') are bypassed, and step a is repeated.

Module: IDCCK01

Procedure: DSDRPROC, DSDRLIST

3. a. The dsname, ddname, unit type, volume sequence number (VOLSEQ), and number of volumes (NVOLS) are extracted from the type 1 DSDR record and placed in the print buffer. A UPRINT macro is issued to print the buffer.

   b. The volume serial numbers in the type 1 DSDR record (maximum of five) are placed in the print buffer. If VOLSEQ is less than or equal to 5, but not equal to 0, DSDRLIST puts an asterisk in the print buffer next to the last volume serial number, and VOLSEQ is set to zero. If VOLSEQ is greater than 5, it is reduced by 5. A UPRINT macro is issued to print the buffer of five (or fewer) volume serial numbers.

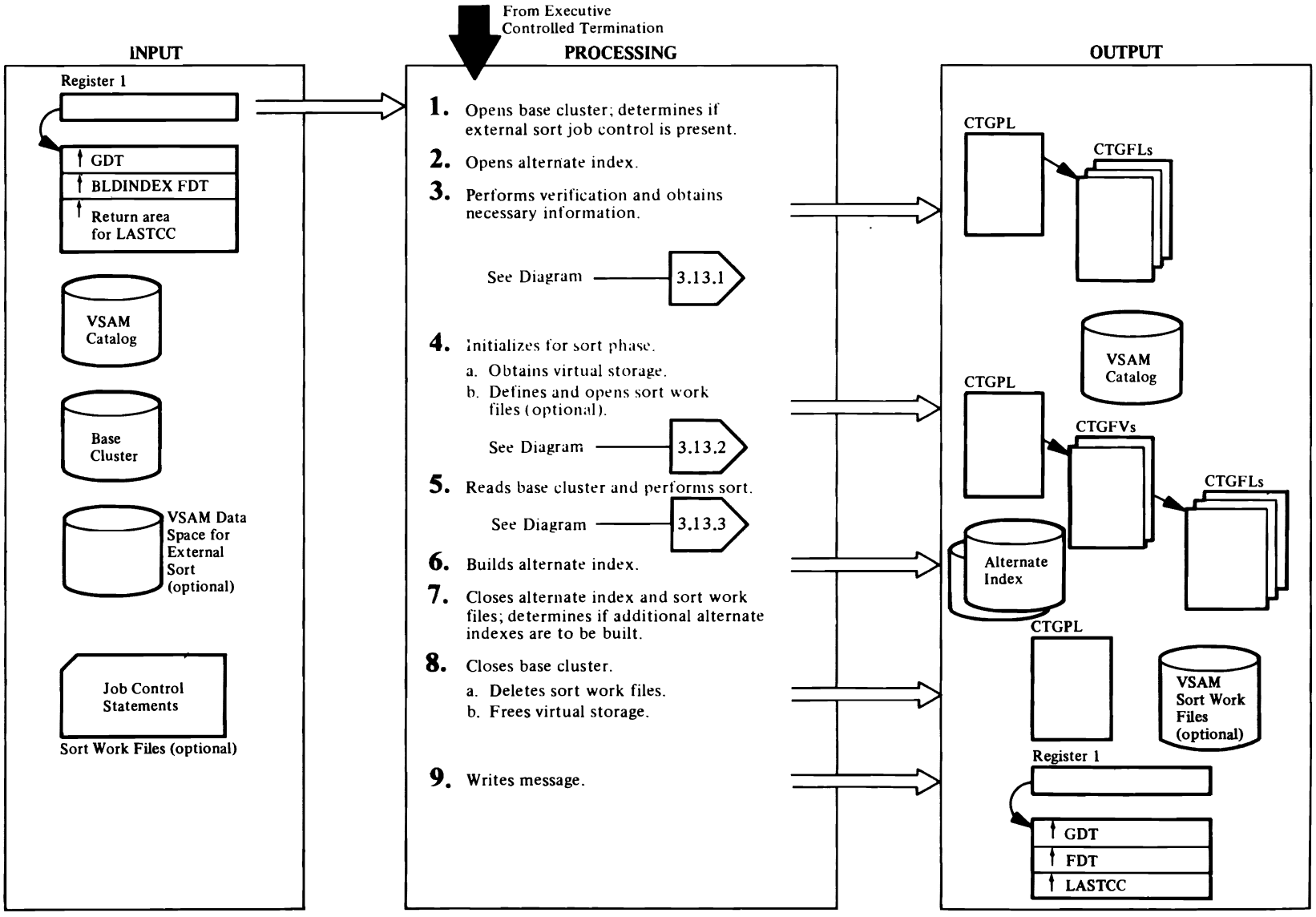
Module: IDCCK01

Procedure: DSDRPROC, DSDRVOLS, GETNEXT, DSDRLIST

4. If NVOLS is greater than 5 and JFCBEXAD isn't equal to 0 (which indicates that a type 2 DSDR record containing the JFCBX is expected), the next logical record is read. If it isn't a type 2 DSDR record, DSDRVOLS issues an error message. A maximum of 15 volume serial numbers are extracted from the type 2 DSDR record, five at a time. For each group of five, DSDRLIST examines VOLSEQ as described in step 3.b and issues a UPRINT macro to print the buffer. When DSDRVOLS has processed all type 2 DSDR records, steps 2 through 4 are repeated until all DSDR records for tape data sets have been processed for the current checkpoint entry.

5. Returns to Diagram 3.12.

Diagram 3.13 BLDINDEX FSR



Extended Description for Diagram 3.13

Module: IDCBI01

Procedure: OPNPROC, JCPROC

1. IDCBI01 calls OPENPROC to build an OPNAGL and issue a UOPEN to open the base cluster for input. If INFILE was specified, OPENPROC indicates the INFILE dname in the OPNAGL. If INDATASET was specified, OPENPROC determines if a TSO environment exists and if the data set name is unqualified. If so, OPENPROC issues the UQUAL macro to qualify the data set name. The data set name is indicated in the OPNAGL. The data set will be dynamically allocated as part of UOPEN processing. OPENPROC indicates input processing in the OPNAGL. UOPEN processing determines if the base cluster is a KSDS or an ESDS and sets a flag in the IOCSTR returned to OPENPROC following the open. This flag will be used by BLDINDEX to determine if alternate index records are to contain prime key pointers or RBA pointers. UOPEN also sets the RPL to keyed sequential processing for a KSDS or addressed sequential processing for an ESDS. If the return code from UOPEN is nonzero, OPENPROC returns to IDCBI01 with LASTCC set to 12 and the BLDINDEX command is terminated.

   OPENPROC checks the high-used RBA of the base cluster returned in the IOCSTR. If the high-used RBA is zero, OPENPROC issues a message returns to IDCBI01 with LASTCC set to 12 and the BLDINDEX command is terminated.

   IDCBI01 calls JCPROC to determine if job control for an external sort has been provided. BLDINDEX will always perform an internal sort if enough virtual core has been provided by the caller. Otherwise, if the caller has provided appropriate job control, BLDINDEX will perform an external sort using two VSAM entry sequenced data sets. Job control consists of DD cards with the following specifications:

DD Name	As provided via the WORKFILES parameter, or defaulted to IDCUT1 and IDCUT2
Data Set Name	Required; can be systemgenerated erated
Volume Serial Numbers	Required; must specify volume(s) containing VSAM data space accessible via a currently available catalog.
Disposition	DISP=OLD required
Unit	Required
Access Method	AMP='AMORG' required

   If the caller has specified the WORKFILES parameter, JCPROC issues a UIOINFO specifying the first dname of that parameter. Otherwise, the UIOINFO specifies a default dname of IDCUT1. The UIOINFO requests a return of the data set name and volume serial number(s). If the return code from UIOINFO is zero, JCPROC issues another UIOINFO requesting the same information for the second dname specified via WORKFILES or the default dname of IDCUT2 if WORKFILES has not been specified. If both UIOINFOs are successful, JCPROC saves the pointers to the information obtained.

Module: IDCBI01

Procedure: MAINPROC, OPENPROC

2. Steps 2 through 7 are performed for each alternate index specified in the OUTFILE or OUTDATASET parameter.

   IDCBI01 calls MAINPROC to control the building of the alternate index. MAINPROC calls OPENPROC to build an OPNAGL and issue a UOPEN for the alternate index. OPENPROC sets a flag in the OPNAGL to indicate that only the alternate index is to be opened. If OUTFILE was specified, OPENPROC indicates the OUTFILE dname in the OPNAGL. If OUTDATASET was specified, OPENPROC determines if a TSO environment exists and if the data set name is unqualified. If so, OPENPROC issues the UQUAL macro to qualify the data set name. The data set name is indicated in the OPNAGL. The data set will be dynamically allocated as part of UOPEN processing. The OPNAGL specifies keyed sequential output processing and specifies open with reset. If the alternate index is nonempty and was defined with the reusable attribute, VSAM OPEN will reset it to an empty condition. If the return code is nonzero OPENPROC sets LASTCC to 8 and returns to MAINPROC where control is passed to Step 7.

Module: IDCBI01

Procedure: MAINPROC, LOCPROC

3. In order to accomplish validity checking and obtain required information, MAINPROC calls LOCPROC to issue VSAM catalog locates. See Diagram 3.13.1.

   On return from LOCPROC, the following information has been obtained to be used in subsequent processing:

Type of base cluster (KSDS or ESDS)	-	returned from UOPEN of base cluster; also in data AMDSB.
Position and length of prime key (if base cluster is a KSDS)	-	in base cluster data AMDSB control block.
Length of alternate–index record	-	in alternate index data AMDSB.
Length of alternate key	-	in alternate index data AMDSB control block.
Position of alternate key in base cluster record	-	in alternate index AMDSB control block.
Unique or nonunique key indicator	-	in alternate index AMDSB control block.
Number of records in the base cluster	-	in base cluster AMDSB control block.

Module: IDCBI01, MAINPROC

Procedure: INITPROC

4. MAINPROC calls INITPROC to obtain resources for building the alternate index. Resources consist of virtual storage for buffers and work areas, virtual storage for the sort and defined and opened sort work files if it is determined that such are required. See Diagram 3.13.2.

Module: IDCBI01

Procedure: MAINPROC, CNTLPROC

5. MAINPROC calls CNTLPROC to read the base cluster and control the sort-merge process. See Diagram 3.13.3.

Module: IDCBI01

Procedure: CNTLPROC, BLDPROC, MERGPROC

6. If an internal sort was performed, CNTLPROC passes each sort record to BLDPROC to build and write the alternate index records. Otherwise, CNTLPROC calls MERGPROC to perform the merge passes and build the alternate index. See Diagram 3.13.3 for BLDPROC and MERGPROC processing.

Module: IDCBI01

Procedure: FINPROC

7. IDCBI01 calls FINPROC to perform cleanup from the alternate index just built. FINPROC tests for an alternate index and sort work files and issues a UCLOSE for any of those data sets which are open. If BLDINDEX processing encounters any errors, FINPROC issues an appropriate error message. Catalog error messages are issued by building an error conversion table and invoking the UERROR macro. FINPROC also issues a UFPOOL to free the sort core, buffers and work areas used in building this alternate index. A message indicating the success or failure of the alternate index build is written. The setting of LASTCC determines the message to be written. If LASTCC from the current build is higher than the maximum value from previous builds, it is saved. LASTCC is cleared for subsequent builds. If the caller of the BLDINDEX has specified multiple alternate indexes, control returns to Step 2.

Module: IDCBI01

Procedure: TERMPROC, DELTPROC

8. IDCBI01 calls TERMPROC to perform final cleanup. TERMPROC issues a UCLOSE to close the base cluster. If sort work files exist, DELTPROC is called to build a CTGPL to delete them.

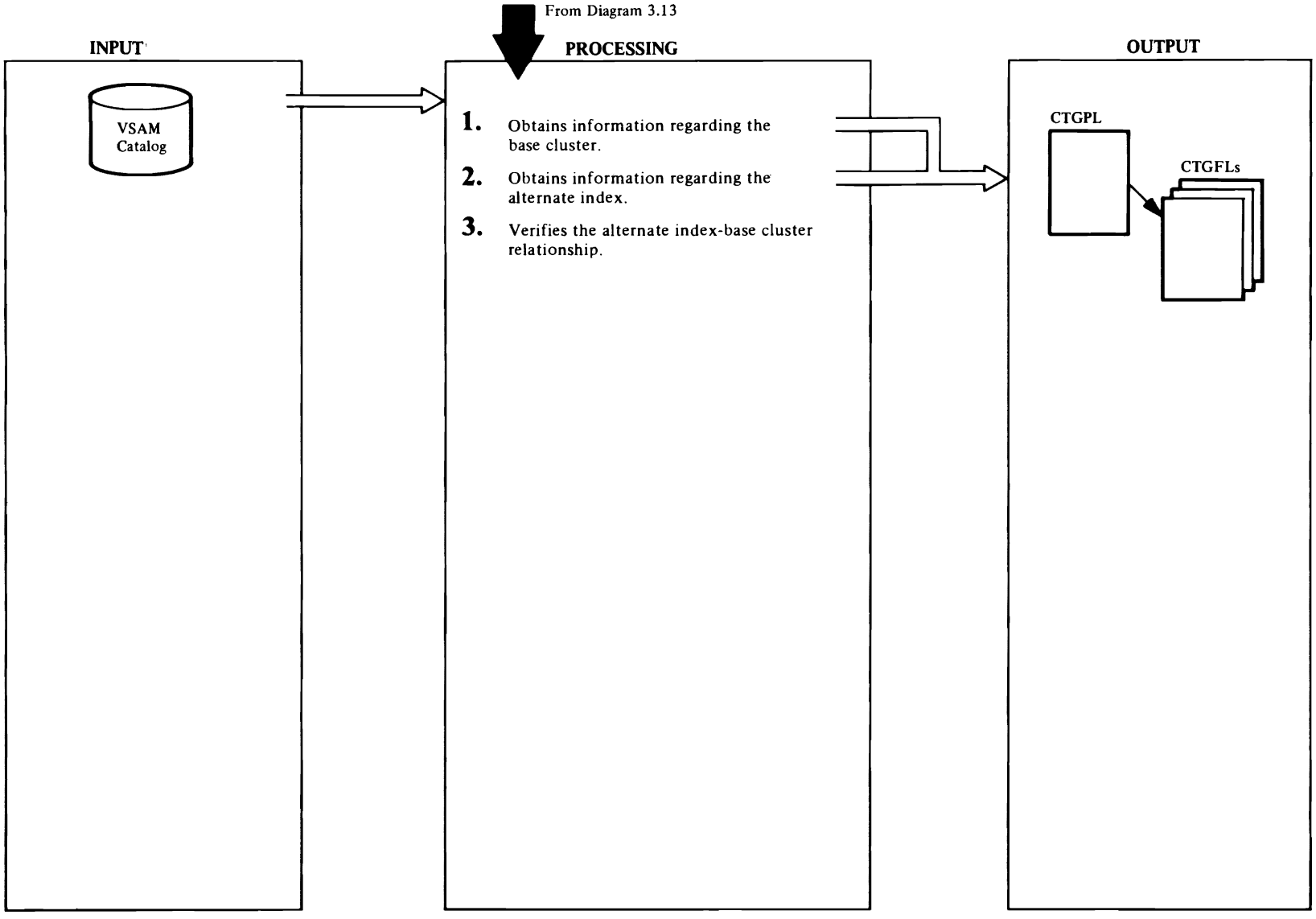
   A UCATLG macro is issued by DELTPROC to delete each sort work file. TERMPROC issues a UFPOOL to free all remaining core obtained via UGPOOL.

Module: IDCBI01

Procedure: TERMPR

9. TERMPROC writes a termination message with the maximum LASTCC encountered. Control returns to Executive controlled termination via IDCBI01.

Diagram 3.13.1 BLDINDEX FSR – Get Information and Verify

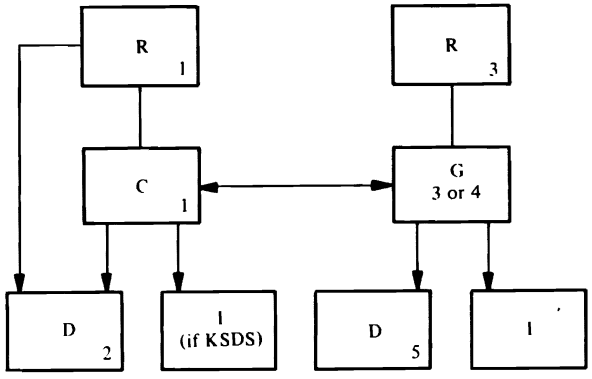


Extended Description for Diagram 3.13.1

Module: IDCBI01

Procedure: LOCPROC, CATPROC

1. The caller of BLDINDEX may specify the alternate index and base cluster names or a path to either. The diagram below shows the relationship of the various objects involved:



R = Path
C = Cluster
G = Alternate Index
D = Data
I = Index

The number in each box indicates which of the locates described below retrieves information for that object. The purpose of this series of locates is:

a. to retrieve the data AMDSB control block of the alternate index and base cluster, and

b. to verify that the alternate index specified by the caller does indeed relate to the base cluster specified.

If the caller specified a path over the alternate index via OUTFILE or OUTDATASET, an additional locate will be required to reach the G object will be required (Locate 4).

The building of the CTGPL and CTGPLs and the issuance of the UCATLG is actually done by CATPROC. LOCPROC makes successive calls to CATPROC to perform these functions. On each entry to CATPROC, the CTGPL and CTGFLs are rebuilt for the specific locate being processed. LOCPROC calls CATPROC for locates 1 and 2 only on the first alternate index being built since these locates are against the base cluster. Appropriate information is saved.

Module: IDCBI01

Procedure: LOCPROC, CATPROC

Locate 1

Locate 1 retrieves the associations of the name specified via INFILE or INDATASET. CATPROC builds a CTGPL for a locate operation. CTGFLs are built for:

ENTYPE - Entry Type

NAMEDS - Type and control interval number of the first three associations

CATACB - Catalog ACB

The entry name used in this locate is the data set name specified by the caller in the DD card pointed to by the INFILE parameter or the name specified in the INDATASET parameter. If the return code from catalog is nonzero, LOCPROC sets a locate error condition, sets LASTCC to 12 and returns control to MAINPROC. MAINPROC returns to IDCBI01 where control is passed to Step 7 (Diagram 3.3).

Note: This same type of error processing follows all subsequent locates except that LASTCC is set to 8 for locates 3, 4, and 5.

If the Entry Type returned by catalog management is an R (path), LOCPROC tests that the first association is a C (base cluster). If the Entry Type is not an R, it must be a C. Otherwise LOCPROC issues a message, sets LASTCC to 12 and returns control to MAINPROC.

Locate 2

CATPROC builds a CTGPL and CTGFLs to retrieve the base cluster data AMDSB.

CTGPL: Entry "name" is the control interval number of the base cluster's data object (D) returned in Locate 1.

CTGFL: ENTYPE - Entry Type

CTGFL: NAMEDS - Type and control interval number of the first three objects associated with the data object

CTGFL: AMDSBCAT - AMDSB control block

The catalog ACB returned from Locate 1 is used in this and all subsequent locates.

LOCPROC saves the first control interval number returned for NAMEDS which is the control interval number of the base cluster object. LOCPROC also moves the AMDSB control block to its own work area.

Module: IDCBI01

Procedure: LOCPROC, CATPROC

2. *Locate 3*

Locate 3 is essentially the same as Locate 1 (minus the catalog ACB address) except that the name specified via OUTFILE or OUTDATASET is used. If the entry type returned by catalog management is an R (path), LOCPROC tests that the first association is a G (alternate index). If the entry type is not an R, it must be a G. Otherwise, LOCPROC issues a message, sets LASTCC to 8 and returns control to MAINPROC.

Locate 4

If the Entry Type from Locate 3 was an R. CATPROC builds a CTGPL and CTGFL to retrieve the alternate index associations.

CTGPL: Entry"name" used is the control interval number of the alternate index (G) associated with the path (R) returned in Locate 3.

ENTYPE: Entry type

CTGFL: NAMEDS—Type and control interval number of the first three objects associated with the alternate index. The entry type returned by catalog management must be a G. Otherwise, LOCPROC issues a message, sets LASTCC to 8, and returns control to MAINPROC.

Module: IDCBI01

Procedure: LOCPROC, CATPROC

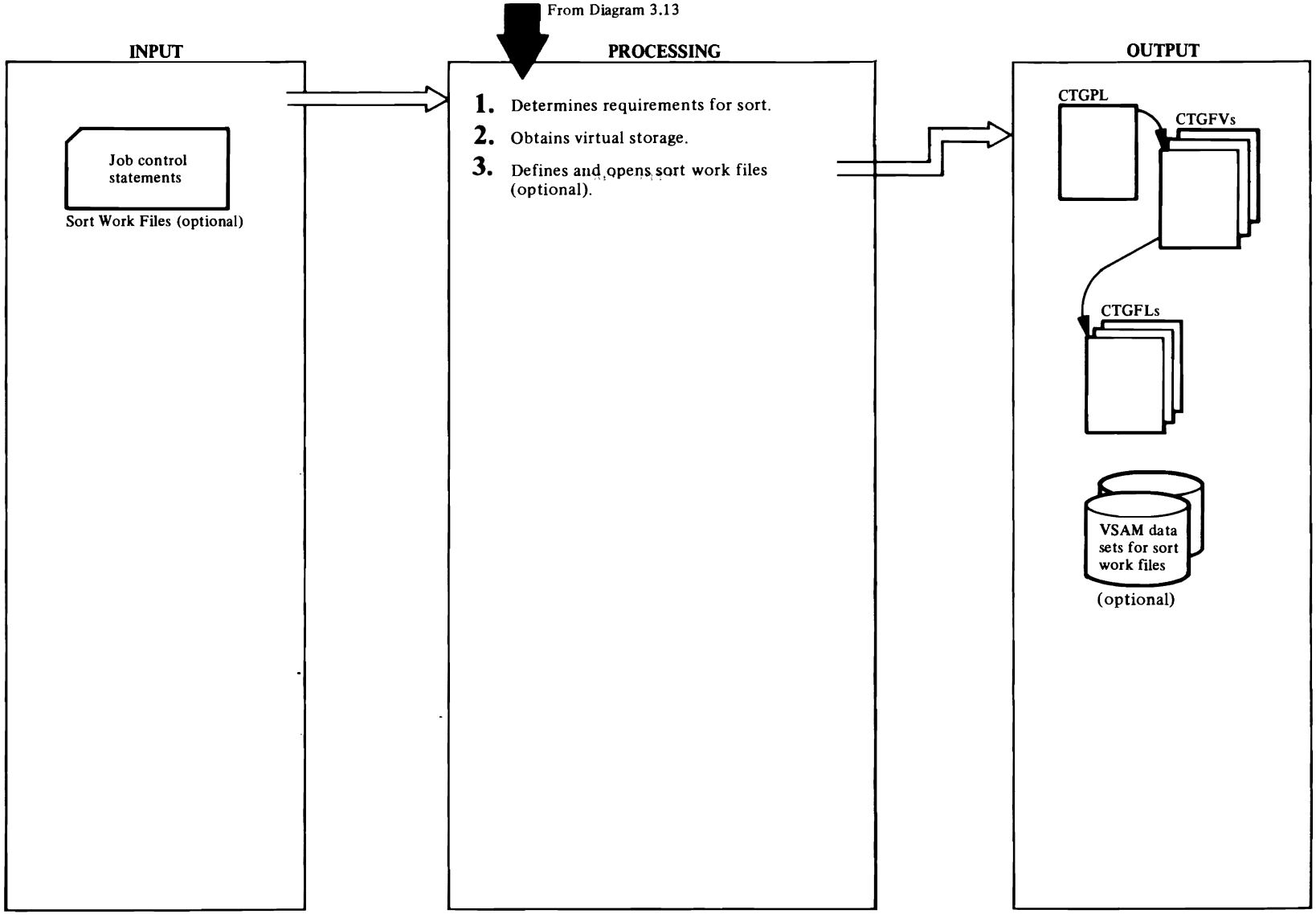
3. LOCPROC must now verify that the alternate index specified by the caller is in fact related to the base cluster specified. LOCPROC compares the control interval number of the base cluster saved from Locate 2 of the control interval number of the third association returned from Locate 3 or 4. This should be, for an alternate index, the control interval number of the related base cluster. If the CI numbers are not equal LOCPROC issues a message, sets LASTCC to 8 and returns control to MAINPROC.

Locate 5

Locate 5 is the same as Locate 2 for the alternate index data AMDSB control block.

Control returns to Diagram 3.3 where control will be passed to Step 4 or Step 7 depending on the setting of LASTCC.

Diagram 3.13.2 BLDINDEX FSR – Obtain Resources and Sort Initialization



Extended Description for Diagram 3.13.2

Module: IDCBI01

Procedure: INITPROC

1. INITPROC issues a UGPOOL macro to obtain virtual core for buffers and work areas, consisting of 1 2K buffer (to be used for output if an external sort is performed), the area required for the CPL/FVT/FPL complex to define the sort work files and the alternate index record output buffer. The first two areas are obtained at this time, even though they may not be used, so that if it is necessary to perform an external sort it will not fail due to lack of virtual storage. If the UGPOOL fails, INITPROC sets LASTCC to 8, issues a message and returns control to IDCBI01, Step 7 (via MAINPROC).

   INITPROC calculates the requirements for both an internal sort and an external sort. If an external sort is performed, the records being sorted are blocked into a block 2048 bytes in length, using a logical record length of 2041 bytes. Blocking and deblocking of sort records within the 2041-byte logical record is accomplished by BLDINDEX.

   The formulas used to determine sort work size are.

   Sort Record Size = Alternate Index Key Length + Prime Key Length (KSDS) or 4 (ESDS)

   $$\text{Number of Records per Block} = \frac{2041}{\text{Sort Record Size}}$$

   $$\text{Total number of 2K Blocks} = \frac{\text{(Number of Records in Base Cluster)}}{\text{(Number of Records per Block)}}$$

   During the first phase of either an internal or external sort, the records being sorted are packed contiguously into a record sort area (RSA). The RSA size is always in increments of 2K so that it can be later used as an input buffer area during the merge phase of an external sort. The initial size of the RSA is calculated as

   Number of Records in Base Cluster * Sort Record Size

   and rounded up to the nearest multiple of 2K. This size is then adjusted as follows:

   a. If the RSA size is less than 4K, it is set at 4K. The number of records in the base cluster is obtained from a statistic maintained in the base cluster AMDSB control block. If this statistic is in error (which can happen if a system failure occurs during a close), it may be necessary to go into an external sort. In this case, space for two input buffers is required.

   b. If the EXTERNALSORT parameter has been specified by the caller of BLDINDEX, the RSA size is set at 32K—the minimum amount of storage which will be used for an external sort during the merge phase.

Module: IDCBI01

Procedure: INITPROC

2. In addition to virtual storage for the RSA, virtual storage for the table (called the "heap") which drives the first phase of the sort is required. This is a table of 4-byte pointers. The amount required is calculated as follows:

   $$\text{RSA Capacity} = \frac{\text{RSA Size}}{\text{Sort Record Size}}$$

   Heap Size = RSA Capacity * 4

   INITPROC issues a UGPOOL for the RSA size plus the heap size. If the UGPOOL fails, the initially calculated RSA size could not be obtained and it will be necessary to perform an external sort. The maximum amount of core used for an external sort is 100K, the minimum 32K. If the maximum amount cannot be obtained, an attempt is made to obtain an intermediate RSA of 60K. INITPROC sets the RSA size to the next lower plateau—100K, 60K, 32K—and loops back to the start of Step 2. If the UGPOOL fails at the lowest plateau (32K), INITPROC sets LASTCC to 8, issues a messages and returns control to IDCBI01, Step 7 (via MAINPROC).

Module: IDCBI01

Procedure: INITPROC, DEFPROC, DELTPROC, OPENPROC

3. If virtual storage was successfully obtained but the amount obtained for the RSA was less than the originally calculated required amount, INITPROC calls DEFPROC to define and open two sort work files to be used during the merge phase of an external sort.

   DEFPROC checks to determine that the caller of BLDINDEX did provide external sort job control. If job control was not provided, DEFPROC sets LASTCC to 8, issues a message, and returns to its caller.

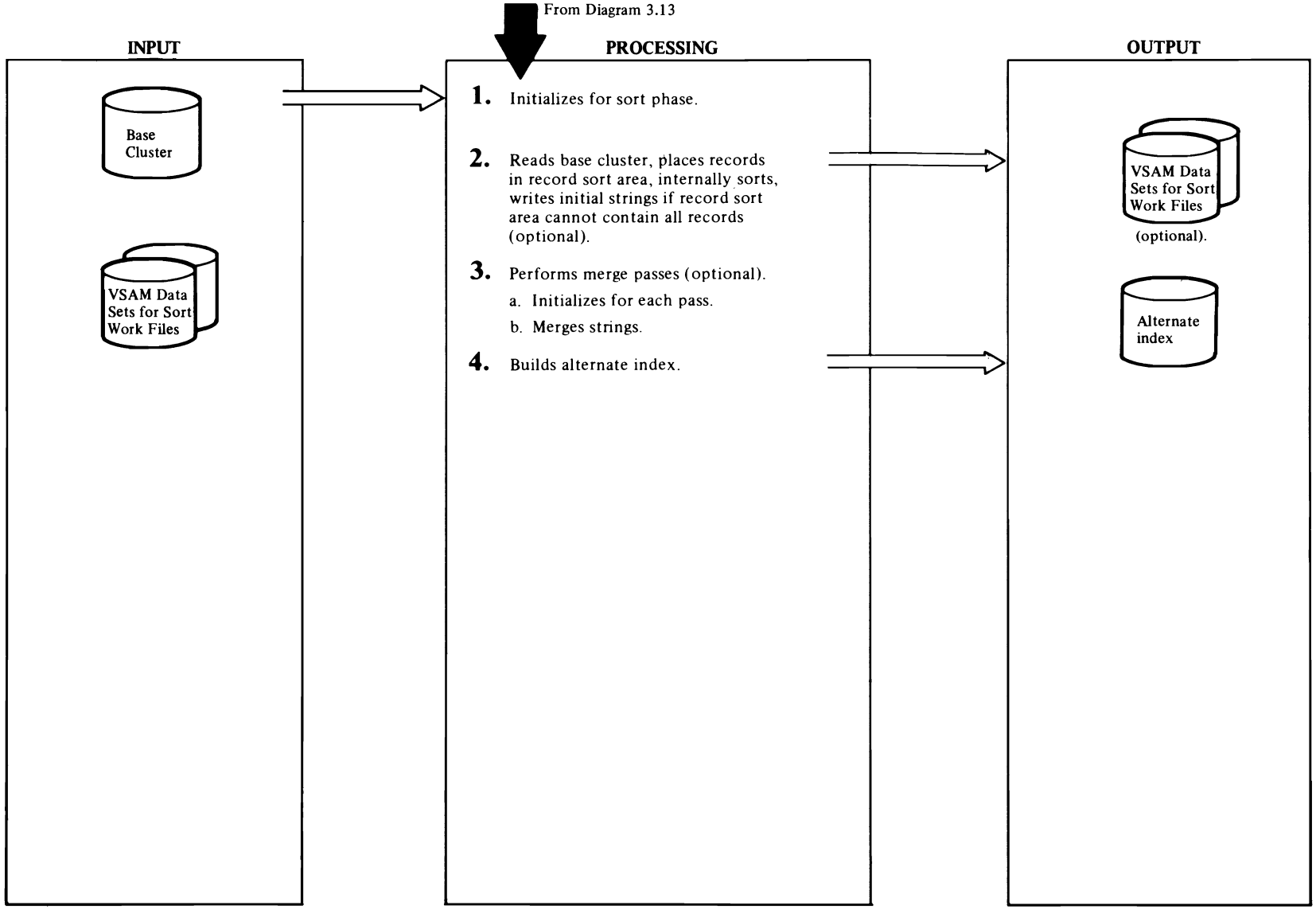
   DEFPROC determines if large enough sort work files exist from a previous sort and, if so, bypasses the define process.

   If external sort work files exist but are not large enough, DEFPROC calls DELTPROC to build a CTGPL to delete each sort work file (specifying the PURGE option).

   If sort work files are to be defined, DEFPROC builds a CTGPL, a cluster CTGFV, a data CTGFV and the required CTGFLs to define the first external sort work file. DEFPROC issues a UTIME macro in order to provide the creation date in the define operation. The cluster FVT references the data set name and the data FVT references the volume serial numbers obtained via UIOINFO from the sort work job control statements. Space allocation is in records: primary, the number of 2K blocks calculated by INITPROC; secondary, 10% of primary. The data set attributes specified are: ESDS, nowritecheck, unordered, speed, suballocation, noerase, reuse, default shareoptions, control interval size of 2048, logical record length of 2041.

   DEFPROC issues a UCATLG macro to define the first work file, makes the necessary changes to the FVTs and issues a UCATLG for the second work file. DEFPROC next calls OPENPROC to build OPNAGL and open the two data sets just defined. The OPNAGLs specify sequential output using control interval processing with user buffers. If the define or open for either of the sort work files fails, DEFPROC sets a define error condition, LASTCC to 8, and returns control to INITPROC. If both sort work files are successfully defined and opened, DEFPROC returns to INITPROC with a flag indicating that an external sort is to be performed. INITPROC returns control to Diagram 3.13 where control will be passed to Step 5 or Step 7 depending on the setting of LASTCC.

Diagram 3.13.3. BLDINDEX FSR – Sort-Merge and Build Alternate Index



Extended Description for Diagram 3.13.3

Module: IDCBI01

Procedure: CNTLPROC

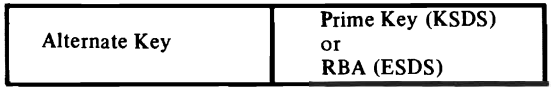
1. CNTLPROC initializes factors which will be used during the sort-merge including pointers to the record sort area (RSA), and the table of pointers which is used during the sort. CNTLPROC also initializes the output buffer with an RDF and CIDF in the event an external sort is performed (the sort work files are processed in control interval mode with user buffers).

Module: IDCBI01

Procedure: CNTLPROC, SORTPROC, BLDPROC, SPILPROC, DEFPROC

2. In a loop CNTLPROC reads each base cluster record and passes it to SORTPROC. SORTPROC performs the function of building the sort records from the base cluster record, placing each record in the RSA and updating the table of pointers (called the'heap') to the records in the RSA. The heap is sorted when the RSA has reached capacity and/or when the last base cluster record has been processed.

   Each sort record is formed by concatenating the prime key of the base cluster (KSDS) or its RBA (ESDS) to the alternate key.



   If the base cluster record is not long enough to contain the alternate key, SORTPROC issues a warning message and sets the current condition code to 4.

   The heap sort consists of two phases. The first phase builds the heap into a tree of nodes having a parent-child relationship. Each parent node has two child nodes and the parent represents a key higher than either of the two children. At the end of the first phase the node at the top of the tree represents the highest key. The second phase removes the top node, places it at the bottom, reduces the heap by 1 and adjusts the parent-child relationships of the remaining nodes. This loop continues until the top of the heap represents the lowest key.

   If enough virtual core was available to contain all the sort records, the sorting takes place after the last base cluster record has been read, after which CNTLPROC passes each record to BLDPROC to build and write the alternate index records (see Step 4). Otherwise, sorting takes place each time the RSA is filled. After the heap is sorted, if the sort was caused by the RSA reaching capacity before end-of-file on the base cluster, SORTPROC calls SPILPROC to write out the records in the RSA in a string of 2K blocks to the external sort work file.

   SPILPROC determines if sort work files have already been defined and opened by INITPROC and, if not calls DEFPROC to perform that function. Normally, SPILPROC will find sort work files already defined and opened. However, if the statistic contained in the base cluster AMDSB control block as to the number of records in the base cluster was erroneously low and the calculated virtual storage for the sort was obtained, INITPROC will not have initialized sort work files. SPILPROC blocks the sort records into the 2K output buffer and issues a UPUT macro to write it. This is performed in a loop until all sort records in the RSA have been written out.

   CNTLPROC calls SORTPROC under the following conditions:

   - After each base cluster record has been read. The address of the record is contained in the IOCSTR of the base cluster.
   - At end-of-file on the base cluster.

Module: IDCBI01

Procedure: CNTLPROC, MERGPROC, BLDPROC

3. After all base cluster records have been read, if the RSA was not large enough to contain all sort records, merge passes must be performed using the two external sort work files. SPILPROC has written out the first strings during the sort phase. During this phase the external sort work file is in create mode. The data set was opened with MACRF=CNV, UBF, OUT, SEQ. PUTs are issued with OPTCD=CNV, SEQ, NUP. Control intervals are written in physical sequence. At the end of the sort phase, CNTLPROC issues a UCLOSE macro to close the output sort work file followed by UOPEN to reopen it. This is necessary to get out of create mode. The second open specifies MACRF=CNV, UBF, DIR, UPD. Subsequently all PUTs will be issued with OPTCD=CNV, DIR, UPD.

   CNTLPROC then calls MERGPROC to control the merge passes. MERGPROC performs the function of merging strings of sort records originally built by SPILPROC using the two external sort work files. The order of merge is normally 16 or less using an area of

32K (the original RSA) for input buffers. In one case, the order of merge will be 2. That is, when the statistic of the number of records in the base cluster AMDSB was so erroneously low that an RSA of 4K was obtained.

In general, the merge is accomplished in the following manner (assuming a 16-way merge) -

- Reading the first 2K block of the first n strings to be merged, where n is 16 if there are 16 or more input strings or where n is the total number of input strings if less than 16.
- Using the first record of each string, build an array in the form of a tree. The tree is made of nodes with a single node at the top. Each parent node has two child nodes and the tree is built so that the record represented by the parent node is lower in value than either child. As the tree-add loop starts, the size of the tree is increased by 1 thus leaving an empty slot at the bottom. The parent of the empty slot is established and if the new record is higher than the parent, it goes into the empty slot at the bottom. However, if the new record is lower, the parent is moved down leaving an empty slot. The parent of the new empty slot is established and the process continues until the new record is found to be higher than the parent at which time it goes into the empty child slot. If the parent is moved from the top of the tree, the new record goes there and the process stops.
- Output the lowest record on the tree. This output will be to BLDPROC (see Step 4) if this is the last or only merge pass or to the output string if this is not the last merge pass.
- Update the tree filling the slot left empty from the step above.
- Get the next record from the same string as the previous lowest record. Output it if it is lower than the current lowest, otherwise add it to the tree.
- Continue this process until all records in all input strings currently being processed have been output.
- Loop until all input strings for this merge pass have been output.
- If more merge passes are required, make the previous output file the next input file and vice versa and repeat the merge passes until the number of input strings is equal to or less than the order of merge.

Module: IDCBI01

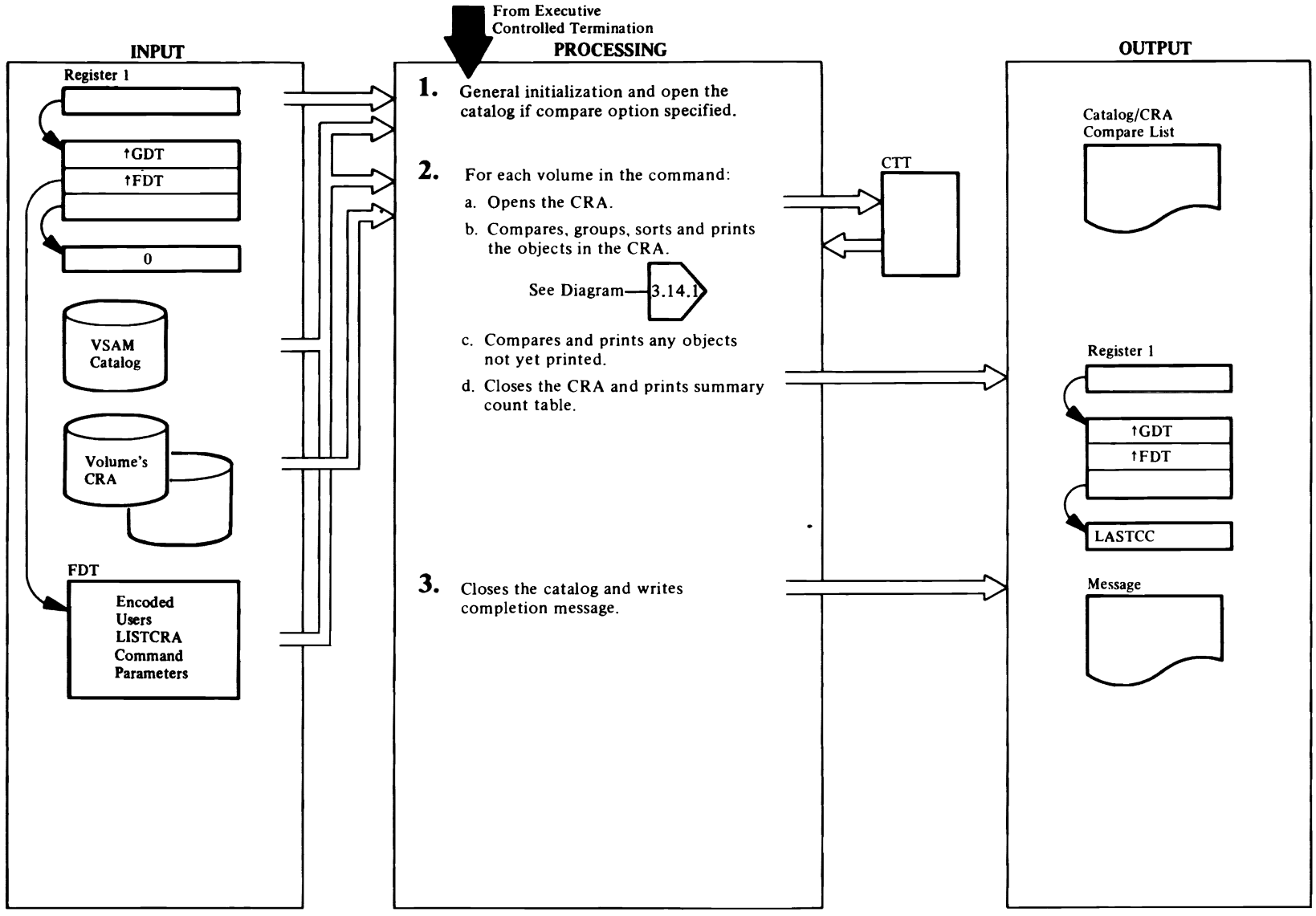
Procedure: BLDPROC

4. BLDPROC is called either from CNTLPROC (if an internal sort was performed) or MERGPROC (on the last merge pass of an external sort). In either case, BLDPROC is passed sorted records one at a time.

   On the first entry to BLDPROC, the IOCSTR for the alternate index is initialized as well as the static portion of the alternate index record. On all subsequent entries, the alternate key of the sort record passed to BLDPROC is compared to the key of the alternate index record being built. If these keys are unequal, the alternate index record is to be written out. BLDPROC determines if the record was too short to contain all the prime key or RBA pointers and, if so, issues a warning message containing the number of excess pointers and sets the current condition code to 4. The record is written with a UPUT macro and the buffer reset for the next record. Before moving the prime key or RBA from the sort record to the alternate index record, BLDPROC checks if the alternate index was defined with the UNIQUEKEY attribute. If so and if the new prime key or RBA is not the first for this alternate index record, BLDPROC issues a warning message and sets the current condition code to 4. Only the first prime key or RBA is placed in the alternate index record. BLDPROC also checks that the record is long enough to contain the new prime key or RBA and, if not, increments an excess pointer counter. If all checks prove successful, the new prime key or RBA is moved to the alternate index record.

   After CNTLPROC passes the last sort record to BLDPROC (internal sort) or receives control back from MERGPROC (external sort), CNTLPROC calls BLDPROC one more time to write out the last alternate index record. Control is then returned to IDCBI01 via MAINPROC—Diagram 3.13, Step 7.

Diagram 3.14 LISTCRA FSR



Extended Description for Diagram 3.14

Module: IDCLR01

Procedure: AATOPLR, INITLZE, CATOPEN, ERROR

1. Routine addresses, the UOPEN argument list and the UIOINFO option byte are initialized in the work area. The alternate print file is UOPENED if possible. If the COMPARE option was specified, a UOPEN is issued for the catalog specified on the control cards. If the OPEN is successful, a UVERIFY is issued and the catalog name is obtained using Access Method Services field management (IDCRC04) and compared to the catalog named specified on the control cards. If there is a match, the volume serial is obtained via IDCRC04 and the catalog is locked out from other users of the system. If the COMPARE option was not specified or the OPEN of the catalog failed, the no compare indicator is set.

Module: IDCLR01, IDCLR02, IDCRC04

Procedure: AATOPLR, CRAOPEN, PRTVOL, INTSORT, MEMSORT, DOVSAM, PRTVSAM, DOOTHER, PRTOTHR, PRTFIFO, GETPRT, PRTCMP, CLENCRA, SUMIT

2. For each of the CRAs specified by a job control card, the following is repeated:

   a. The UOPEN parameter list is set up with the *dname* and the master catalog password and the UOPEN and UVERIFY are issued for the CRA. If they are successful and there is a match on the owning catalog name, the volume serial is obtained from the CRA via IDCRC04 and a UREST is issued to print a subtitle for this CRA. The entire CRA is read to build the CI translate table (CTT) in space gotten by UGPOOL. If the SEQUENTIALDUMP option was specified, all of the printing of the records is done by calls to GETPRT as the CTT is built and steps b. c. and the first part of d. below are skipped.

   b. The CRA volume record and its extensions are optionally compared to the corresponding catalog entry and printed by PRTVOL. The VSAM objects are then sorted into alphabetical order, optionally compared to corresponding catalog entries and printed by INTSORT, MEMSORT, DOVSAM, and PRTVSAM. Next, the nonVSAM objects are sorted, compared, and printed by INTSORT, MEMSORT, DOOTHR, and PRTOTHR. See Diagram 3.14.1.

   c. If either sort fails for lack of memory (from b. above), the objects are compared and/or printed in the order they appear in the CRA by PRTFIFO. Records already processed by the above procedures are skipped. If the object is a VSAM object, PRTVSAM is called and if it is a not, PRTOTHR is called.

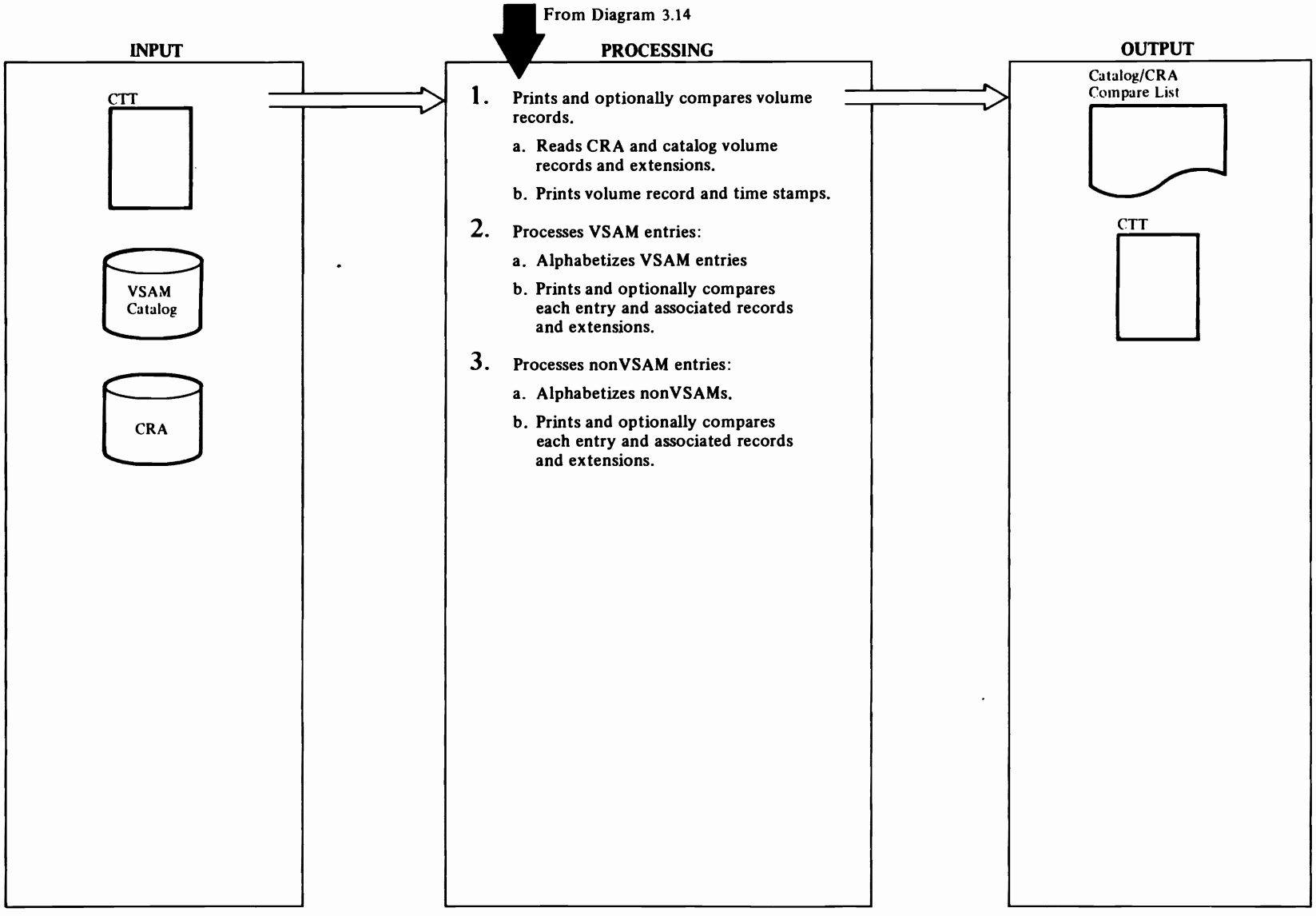
   d. GETPRT is used to get the CRA copy of any other records, and the catalog record, if compare. These are printed and compared by PRTCMP. When all objects have been processed, the CRA is closed by CLENCRA and a summary is printed by SUMIT.

Module: IDCLR01

Procedure: AATOPLR, CLEANUP

3. The UCLOSE macro is issued to close the catalog data set and the UDEQ is issued to release the system lockout from the catalog. The completion code message is printed and the UFPOOL macro is issued to free storage. Control is returned to the caller.

Diagram 3.14.1 LISTCRA FSR – Process CRA



Extended Description for Diagram 3.14.1

Module: IDCLR01, IDCLR02, IDCRC04

Procedure: PRTVOL, SUMIT, GETPRT, VERTEXT, INTVEXT, TCICTCR, BLDVEXT, PRTMCWD, UPRINT, UIOINFO, PRTTIME

1. a. PRTVOL uses GETPRT to read the CRA volume record and IDCRC04 to extract the identifying fields and, if compare, the equivalent information is gotten from the catalog in the same manner. If compare is specified, information is compared and, if not equal, the record is printed and the severest miscompared field is identified by PRTMCWD. If compare is not specified, all records are printed. Horizontal extension records are processed and vertical extension records are checked by VERTEXT and handled in the same way.

   b. The time stamps from the CRA volume record and on the CRA volume and, if compare, in the catalog records are printed by PRTTIME.

Module: IDCLR01, IDCLR02, IDCRC04

Procedure: INTSORT, MEMSORT, DOVSAM, PRTVSAM, GETPRT, VERTEXT, INTVEXT, TCICTCR, BLDVEXT, ADDASOC, INTASOC, PRTMCWD, UPRINT, PRTAAXV, PRTOJVL, CKEYRNG, SUMIT

2. a. The sort of the VSAM entries is initialized by INTSORT which scans the CTT counting the number of VSAM entries, gets storage via UGPOOL for a sort table, initializes dummy first and last entries and then loops through the CTT entries calling IDCRC04 to extract the entry names to be sorted. The MEMSORT procedure orders the entries by adding forward and backward chain pointers to alphabetize.

   b. If compare was specified, the following procedure is passed through twice, the first time comparing only. When a miscompare is detected the procedure is restarted printing everything. From the entries in the sort table an association table is built containing the control intervals of all associated entries. Passing through this table all associated records are printed. For base cluster's AIX associations, only the entries' volumes are printed (to assist in recovery). The horizontal extension records are printed as are the vertical extension records. Throughout, the names of significant items are noted if they miscompared and these are printed.

Module: IDCLR01, IDCLR02, IDCRC04

Procedure: INTSORT, MEMSORT, DOOTHR, PRTOTHR, GETPRT, VERTEXT, INTVEXT, TCICTCR, BLDVEXT, SUMIT, PRTMCWD, UPRINT, PRTOJAL, INTASOC

3. a. The logic and procedures used here are the same as are used in 2a with the exception that nonVSAM entries in the CTT are sorted.

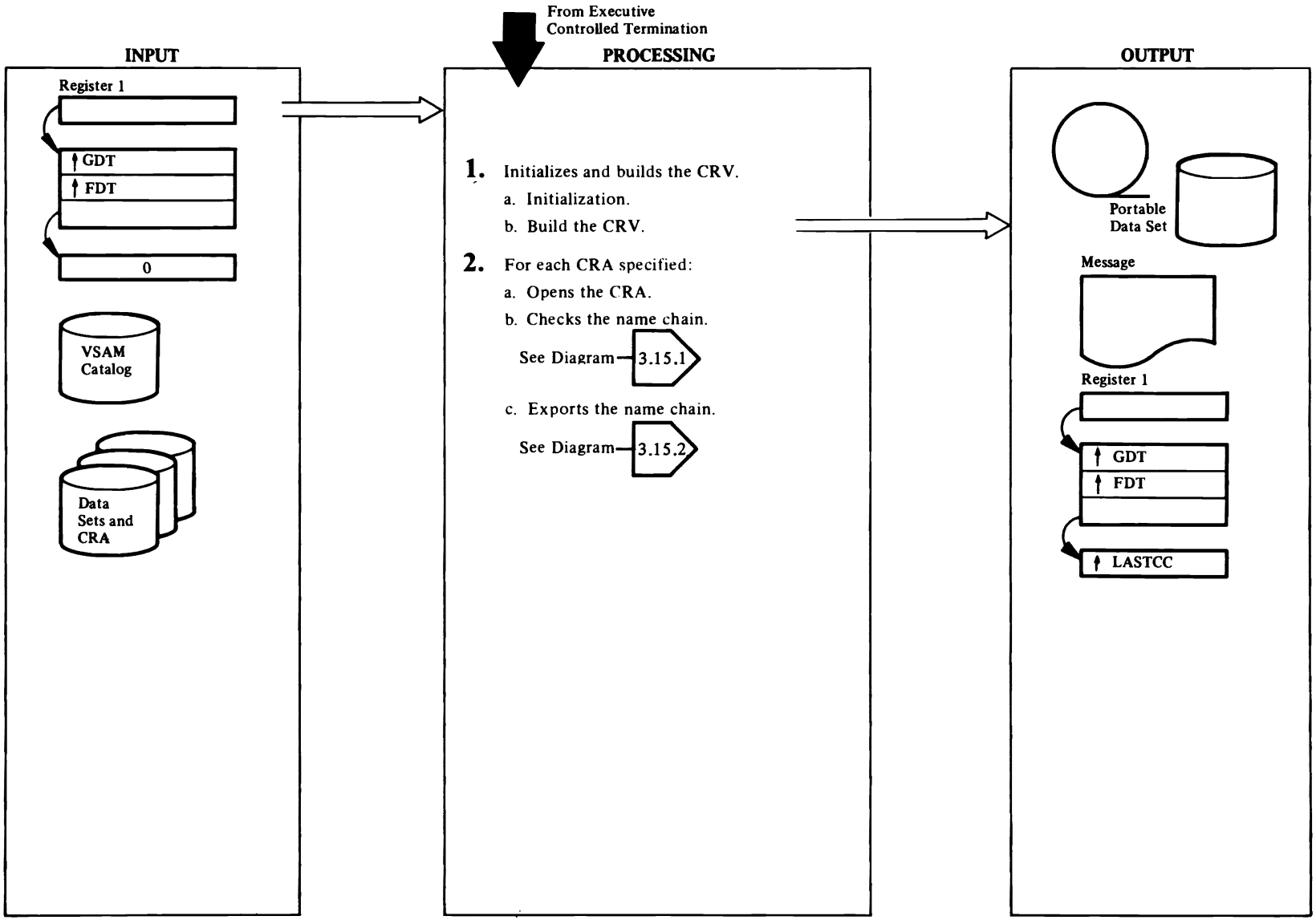
   b. The logic and procedures used here are the same as used in Step 2b except that nonVSAM entries are handled. In addition, PRTOJAL is used to process aliases and logic is included to group generation data groups.

   For all of the above, GETPRT expands further as follows:

   GETPRT, UGET, IDCRC04, BUFSHUF, PRTCMP, UPRINT, PRTOJVL, CKEYRNG, PRTDMP, PRTDMPC.

   GETPRT uses UGET to read the CRA record and the catalog record, if compare. IDCRC04 is used to extract all necessary fields from the record. These are printed and optionally compared by PRTCMP and PRTDMP (if the dump format was specified) and PRTDMPC (if compare was also specified). PRTOJVL is used to print the object's volumes.

Diagram 3.15 EXPORTRA FSR



Extended Description for Diagram 3.15

Module: IDCRC01

Procedure: INIT, SUBSP, BUILDCRV, BUILDNAM, MESSAGE

1. a. SUBSP is called which issues a UGPOOL to obtain storage for the blocks associated with the name chain. This storage is allocated into small blocks to be used later. Storage is then obtained for the buffer pool VGO space, the CRV, the ACC and the VTT.

   b. Each CRA volume is read for the following information: UIOINFO is used to obtain the volume serial numbers and device types which are placed in the CRV. BUILDNAM is called to build the name chain. This procedure calls SUBSP to get a block of storage to be anchored to the CRV. The name pointer is placed in the block as it is read from the CRA.

Module: IDCRC01, IDCRC02, IDCRC03, IDCRC04

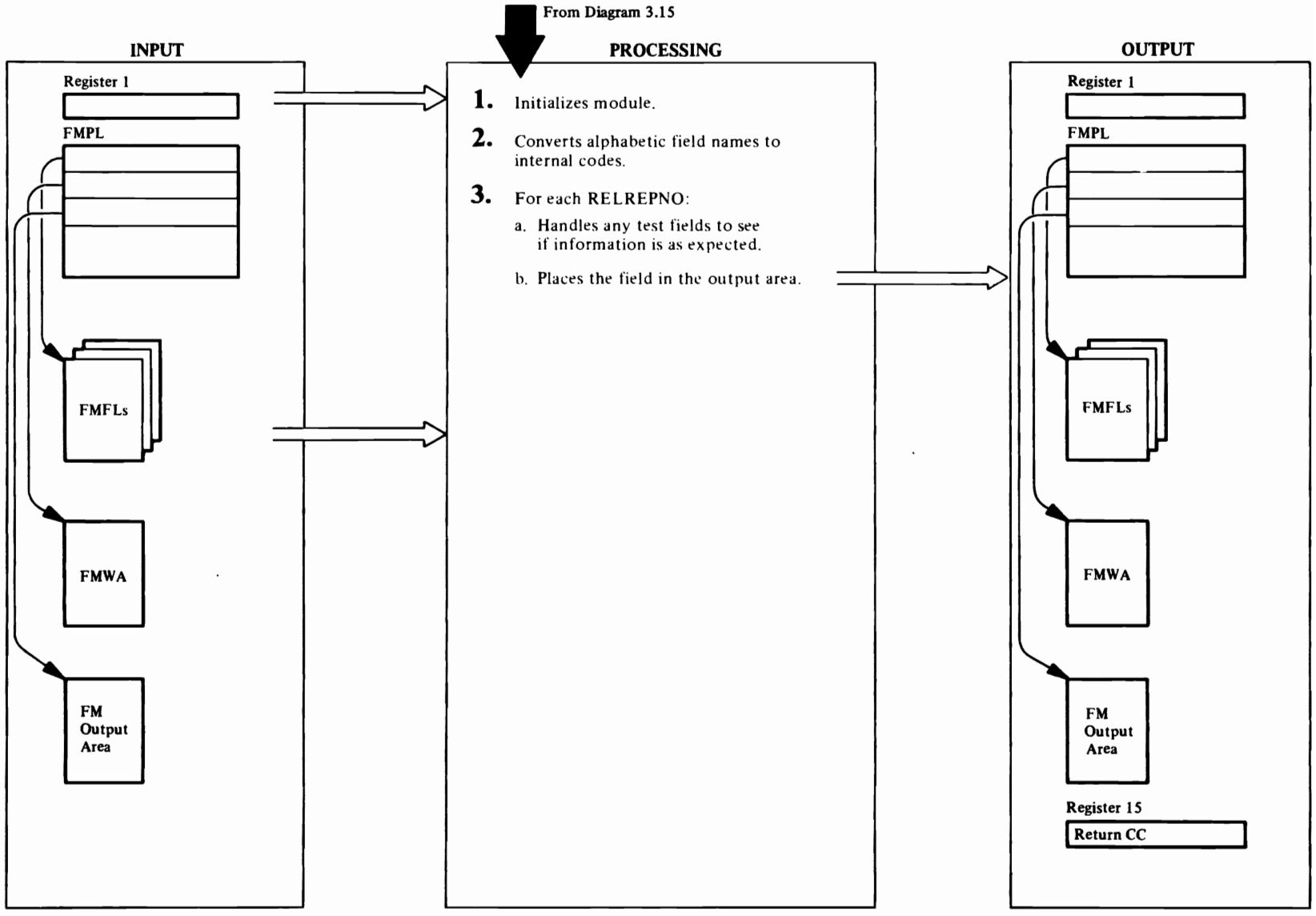
Procedure: OPENCRA, OPEN, TIMESTMP, SCANCRA, NAMETABL, DIRECT, EXTRACT, ERRCK, MESSAGE, COMPNAME, CKCATNm, CKNAMES, DUPNAMCK, SYNCH, OBJVOLCK, CRAOPEN, EXPORTDR, OPENCRA, MESSAGE, ERROR

2. a. OPENCRA initializes the buffer pool pointer required by field managment (IDCRC04). It then calls OPEN which opens the CRA for direct processing and checks it for the correct owning catalog. OPENCRA then issues the UIOINFO macro to get the CRA volume timestamp and place them into the VTT. It then calls SCANCRA to build the catalog CI numbers and places them in the CTT and calls NAMETABL which places the record type and name pointer in the name block. If entries were specified, the name block is marked if a match is found with the input. OPENCRA then calls DIRECT which calls EXTRACT which interfaces with IDCRC04 to obtain the directory information from tthe CRA record. ERRCK calls MESSAGE if an error occurred in this procedure. For IDCRC04 see Diagram 3.15.1.

   b. CKNAMES is called to gather the passwords for the VSAM data sets using EXTRACT, collect the association CI numbers for the VSAM data sets using EXTRACT, determine the largest LRECL for the data sets using EXTRACT, and flagging any object names if they are invalid for this system. DUPNAMCK is called to loop through all the names in the chain checking for duplicates. If one is found, it is marked so that it will not be exported. A message is written indicating the duplicate name. SYNCH is called which checks each entry on the name chain for a CI number, checks the VSAM data sets for a data entry and if there is a data volume index, OBJVOLCK is called which matches the volume serials in the VGOs and VTT, matching the CI and timestamp. If at any time there was an error, ERROR is called to write a message and determine if the process can continue.

   c. EXPORTDR is called which closes the CRA as a data set and opens it as a catalog, then calls MESSAGE to write the "exporting CRA" message. It checks the name chain for the CRA for null entries and nonmatches and marks them not exportable. It initializes the export table and calls IDCRC02 to export the entries. See Diagram 3.15.2 for a description of IDCRC02. When the Export Driver returns, then the completion or error message is printed and processing continues with the next CRA.

Diagram 3.15.1 EXPORTRA FSR – Field Management



Extended Description for Diagram 3.15.1

Module: IDCRC04

Procedure: IDCRC04

1. IDCRC04 is a service routine used by EXPORTRA and LISTCRA to compare and extract data from catalog and CRA records. Upon entry from either IDCRC01 or IDCLR01 it sets up addressability to the work area and initializes the current CI number in the work area for the callers get routine (either IDCRC03 or IDCLR02).

Module: IDCRC04

Procedure: PSCNC, PTRNS

2. PSCNC is called which loops through each field management field list and calls PTRNS which compresses the name into a 4-character ID and places it into the FMFT along with its corresponding dictionary information and supplied group code. The tables are chained according to like group code.

Module: IDCRC04 IDCLR02 IDCRC03

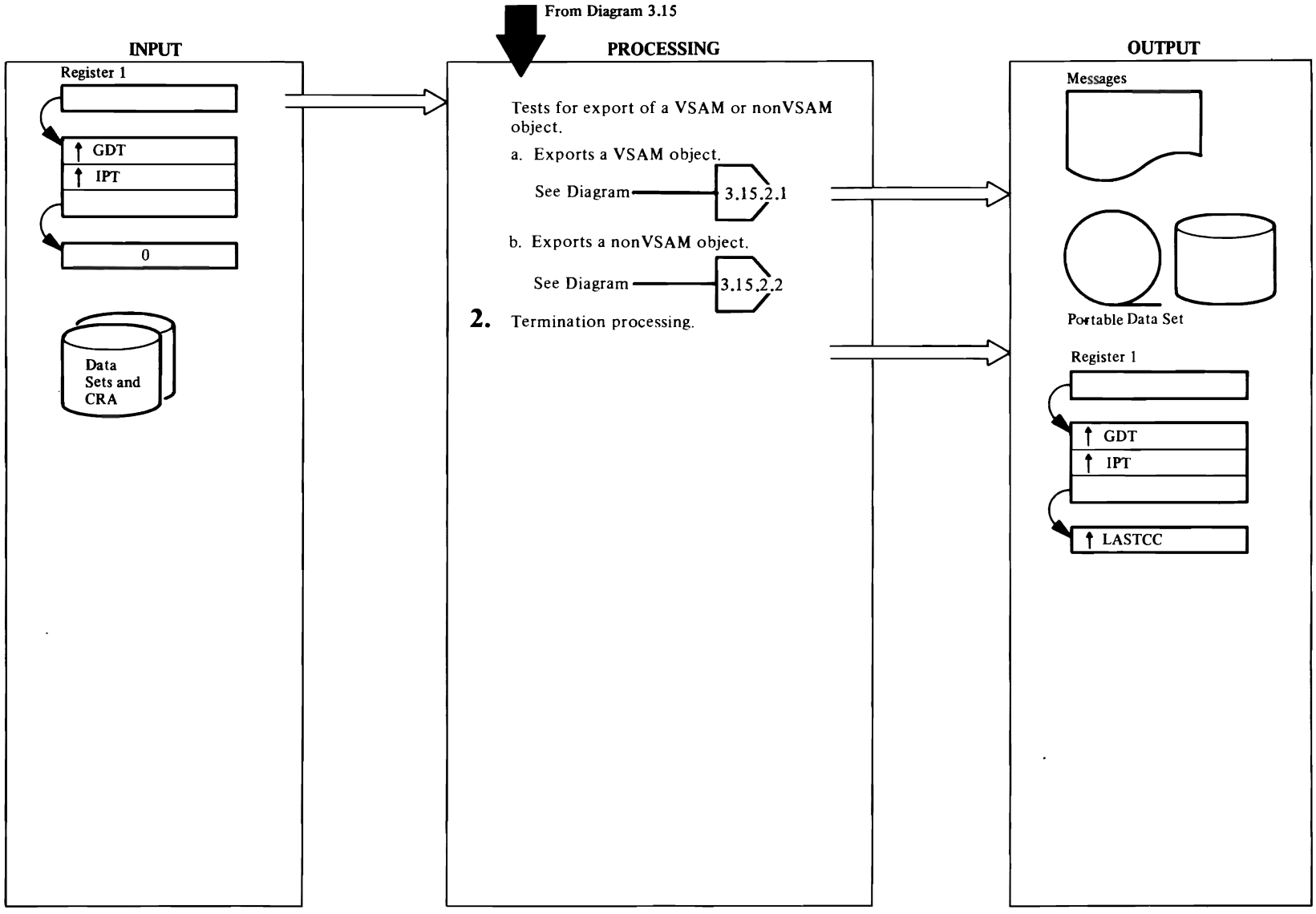
Procedure: PSCNF, PTSTS, PGVAL, PGREC,, PCKLC, PEXPT, PLNRV, PTCMP, PLOC2, PGREP, PSHIN

3. PSCNF is called to process these field tables. It first processes the test field and then the one it is looking for so it may place the data in the output area.

   a. The field lists are tested by looping through all the CI numbers (PGVAL), interfacing with the callers get record routine, either IDCRC03 or IDCLR02 to obtain addressability to the block containing a CI number (PGREC). It then locates the catalog fields within a given record by insuring the requested field actually exists in the group occurrence data (PCKLC) then sets up the address and length of extension pointers as requested via the RELREPNO specified on entry (PEXPT) and extracts the data from the found field and indicates its length (PLNRV). After the data is found, it is compared by PTCMP with the input data and a match or mismatch is indicated.

   b. PLOC2 is the highest-level procedure for placing the data in the output area. This procedure is called by PSCNF if the FMFT is not a test FMFT. It calls PGREP to find the highest non-deleted RELREPNO with the desired group code and saves the address and length of the field which is checked by PGREC. PSHIN checks for enough space in the output area and, if there, moves the field to the output area or moves Fs if non-existent. PGVAL and its subprocedures described above are used to find the fields requested and, after found, PSHIN moves the data to the output area.

Diagram 3.15.2. EXPORTRA FSR – Driver



Extended Description for Diagram 3.15.2

Module: IDCRC02

Procedure: OPENPROC, CLUSPROC, SAVEPROC, RECPROC, PUTPROC, NVSMPROC, GDGPROC, ASOCPROC, ALSPROC

1. IDCRC02 tests the input parameter list for export of a VSAM or nonVSAM object. OPENPROC opens the portable data set for output. If the object to be exported is a VSAM object then step 1.a is done; if it is a nonVSAM object and it is not associated to a GDG, then step 1.b is done. If the object is a GDG, then step 1.c is done.

   a. CLUSPROC gets catalog information for the cluster, data, index and paths from the CRA. SAVEPROC holds the control records containing the catalog information until catalog processing is completed, then writes them to the portable data set. OPENPROC opens the cluster data for input. RECPROC copies the data to the portable data set. PUTPROC writes a software end-of-file to the portable data set.

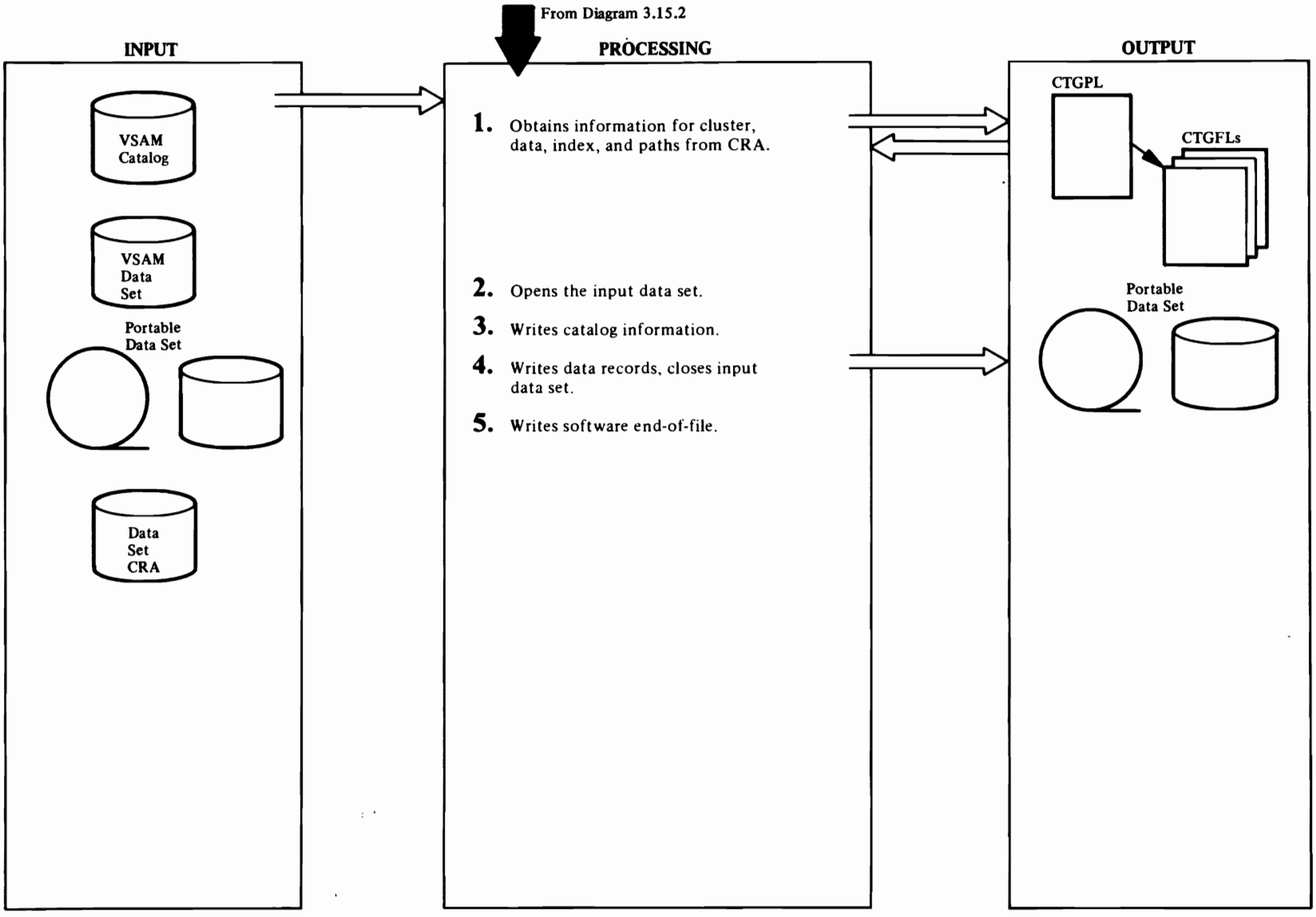
   b. NVSMPROC gets catalog information for the nonVSAM object from the CRA. ALSPROC gets catalog information for any aliases connected with the nonVSAM object. SAVEPROC holds the control records containing catalog information until catalog processing is completed, then writes them to the portable data set. PUTPROC writes a software end-of-file to the portable data set.

   c. GDGPROC gets catalog information for the GDG object from the CRA. ASOCPROC gets catalog information for any nonVSAM associations to the GDG base. ALSPROC gets catalog information for any alias associations to the nonVSAM object. SAVEPROC holds the control records containing catalog information until catalog processing is completed, then writes them to the portable data set. PUTPROC writes a software end-of-file to the portable data set.

Module: IDCRC02

2. IDCRC02 tests return codes from CLUSPROC, NVSMPROC, and GDGPROC. If any alias or path is not exportable, a warning message is issued. The portable data set is then closed if it is the last request or if a severe error occurred.

Diagram 3.15.2.1 EXPORTRA FSR – Export VSAM Data Set



Extended Description for Diagram 3.15.2.1

Module: IDCRC02

Procedure: CTLGPROC, CLUSPROC LOCPROC

1. For the cluster entry of the VSAM data set, LOCPROC builds a CTGPL and CTGFLs to retrieve information from the CRA. A CTGFL is built for the following catalog fields:

   ENTYPE, ENTNAME, DSATTR, OWNERID, DSETCRDT, DSETEXDT, BUFSIZE, LRECL, SPACPARM, PASSWORD, PASSPRMT, USVRMDUL, USERAREC, LOKEYV, HIKEYV, VOLSER, AMDSBCAT, EXCPEXIT, RGATTR, SECFLAGS (**VS2.03.807 only**), NAMEDS and CATACB. CTLGPROC issues a UCATLG with the CTGPL and CTGFLs to retrieve the information from the CRA. CLUSPROC validity checks the catalog entry type and named fields. LOCPROC builds a CTGPL and CTGFLs for the data and index components of the VSAM cluster. CTLGPROC issues a UCATLG to obtain the same catalog information as obtained for the cluster except for the NAMEDS and CATACB fields. Path associations, if present, are processed with the same type of CTGPL and CTGFLs as used for data and index.

Module: IDCRC02

Procedure: OPENPROC

2. OPENPROC opens the VSAM data set for input and verifies the open.

Module: IRCRC02

Procedure: PUTPROC

3. Control records containing catalog information for the cluster, data, index, and paths are written to the portable data set after catalog processing for the object to be exported has been completed.

Module: IDCRC02

Procedure: RECPROC

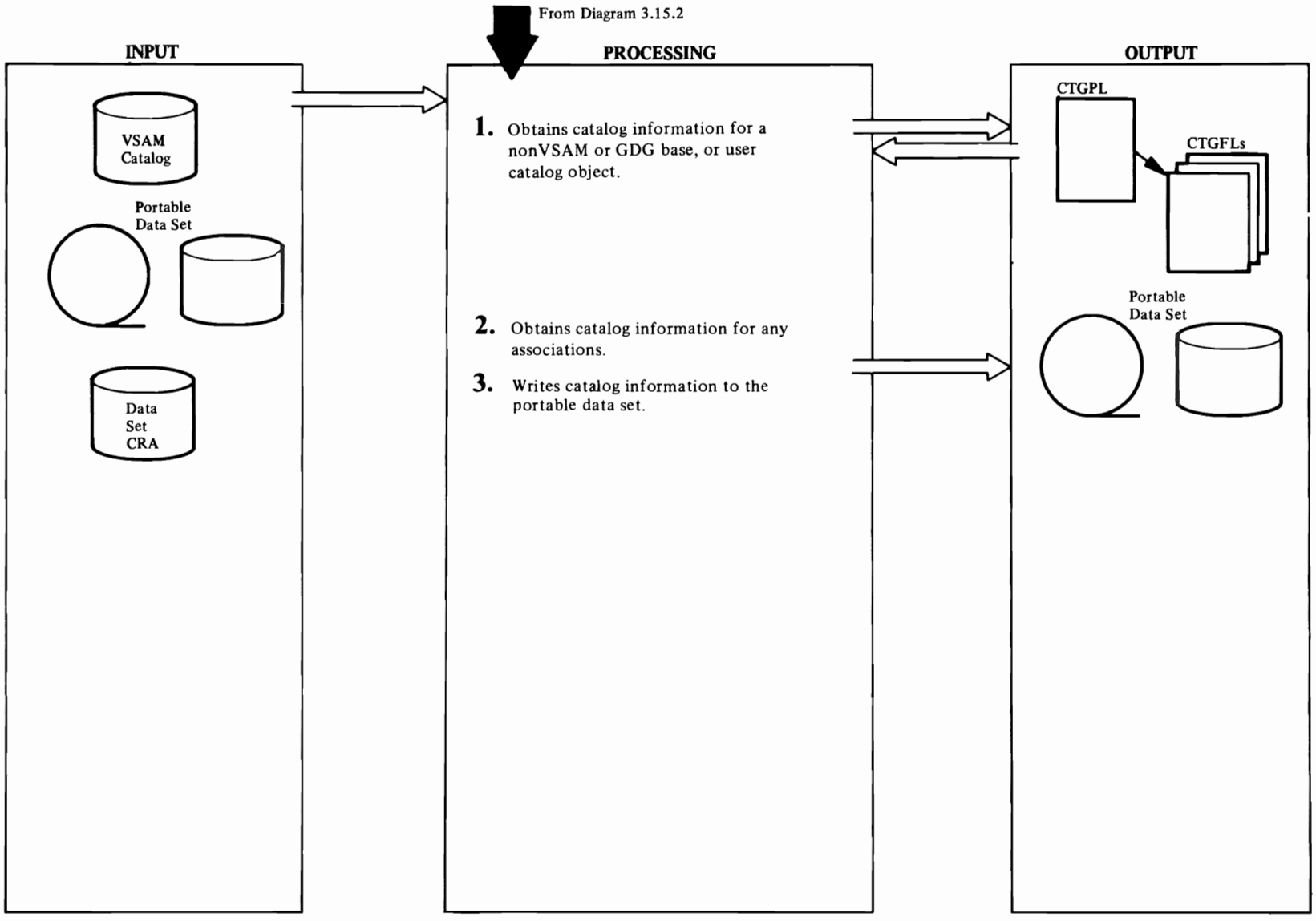
4. RECPROC copies the data to the portable data set and closes the input data set.

Module: IDCRC02

Procedure: CLUSPROC

5. CLUSPROC writes a software end-of-file on the portable data set.

Diagram 3.15.2.2 EXPORTRA FSR – Export NonVSAM



Extended Description for Diagram 3.15.2.2

Module: IDCRC02

Procedure: LOCPROC, CTLGPROC

1. For a nonVSAM or user catalog object, LOCPROC builds a CTGPL and multiple CTGFLs to retrieve catalog information. A CTGFL is built for each of the following fields:

   ENTYPE, ENTNAME, VOLSER, DEVTYP, NAMEDS, CATACB, FILESEQ, OWNERID, DSETCRDT, DSETEXTD

   For a GDG object, LOCPROC builds a CTGPL and multiple CTGFLs to retrieve catalog information. A CTGFL is built for each of the following: ENTYPE, ENTNAME, GDGLIMIT, GDGATTR, OWNERID, DSETCRDT, DSETEXTD, NAMEDS, CATACB.

   CTLGPROC issues a UCATLG with the CTGPL and CTGFLs to retrieve the information from the CRA, and to validity check the ENTYPE and NAMEDS fields.

Module: IDCRC02

Procedure: LOCPROC, CTLGPROC

2. LOCPROC builds a CTGPL and multiple CTGFLs for any alias associations. A CTGFL is built for ENTYPE and ENTNAME catalog fields. CTLGPROC issues a UCATLG to obtain the catalog information.

Module: IDCRC02

Procedure: NVSMPROC, ALSPROC, GDGPROC, ASOCPROC

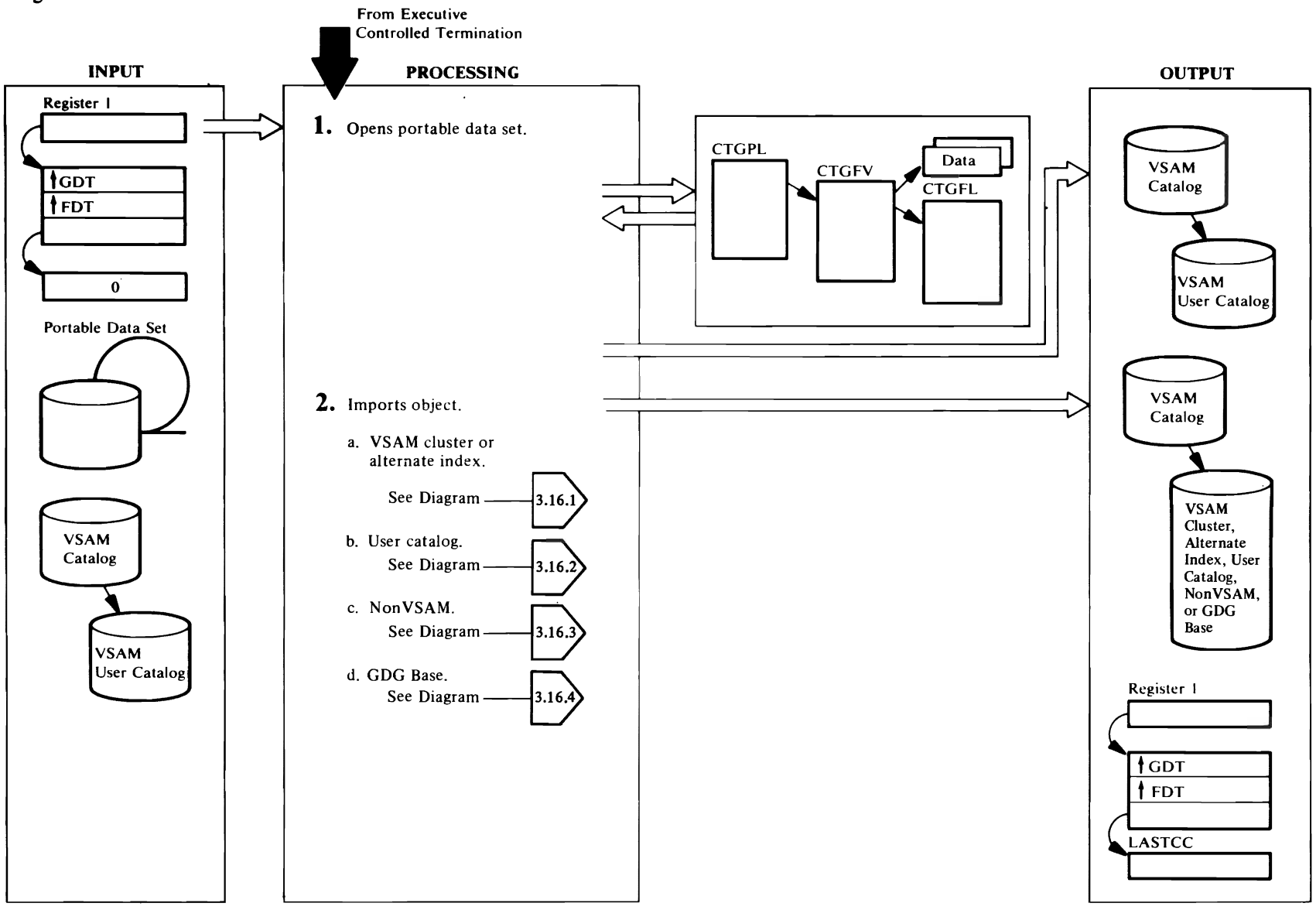
3. NVSMPROC and ALSPROC write control records containing the catalog information to the portable data set after catalog processing is completed.

Module: IDCRC02

Procedure: GDGPROC, NVSMPROC

4. GDGPROC and NVSMPROC write a software end-of-file to the portable data set.

Diagram 3.16 IMPORTRA FSR



Extended Description for Diagram 3.16

Module: IDCRM01

Procedure: OPENPROC

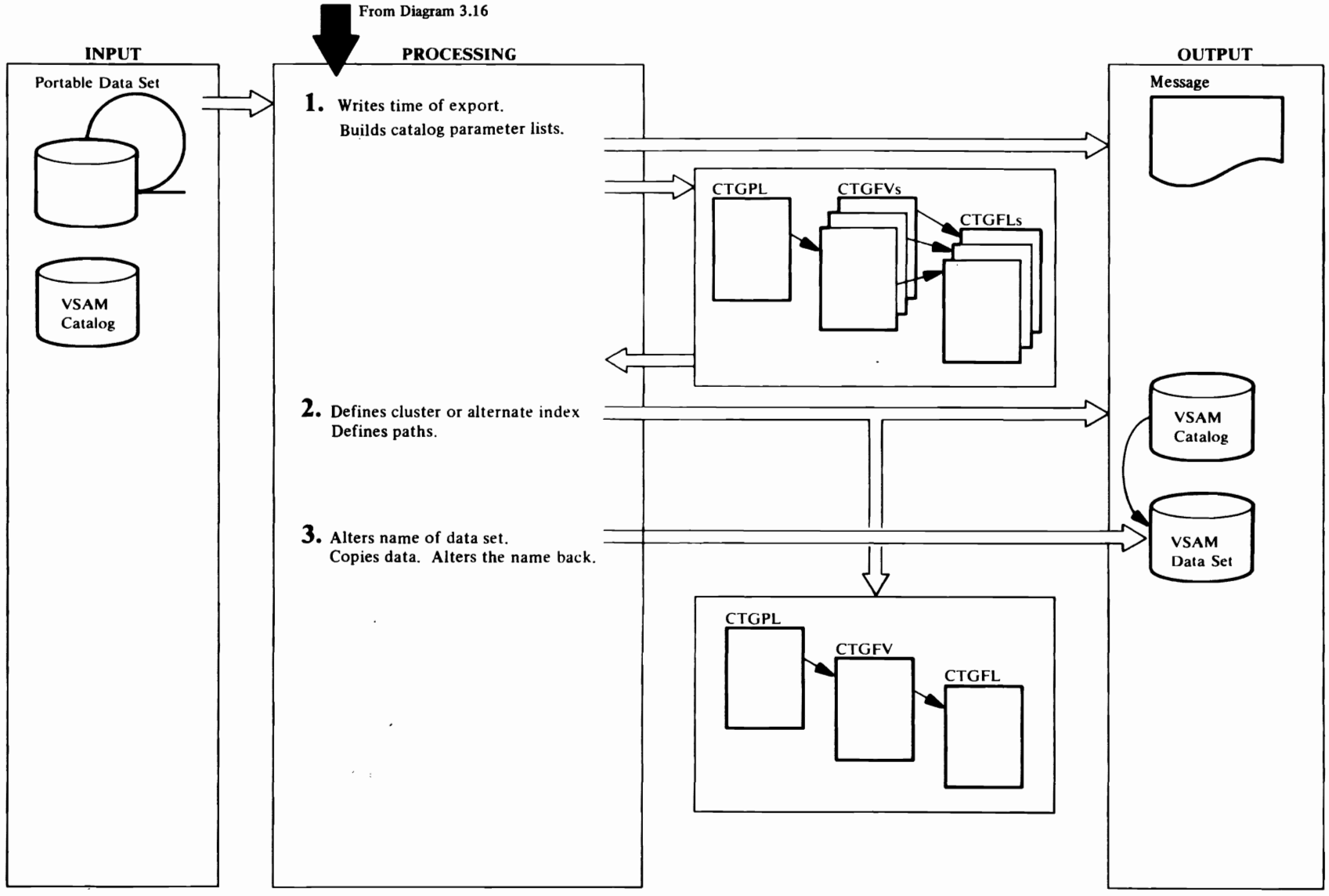
1. If INFILE is specified, IDCRM01 issues a UIOINFO to obtain the data set name coded on the DD job control statement associated with the INFILE parameter. OPENPROC builds an OPNAGL and issues a UOPEN to open the portable data set. OPENPROC then issues a UGET to get the first record of the portable data set, which contains the record size of the data set. If the record size is larger than the record size used to open the portable data set, it is closed and reopened.

Module: IDCRM01

Procedure: CLUSPROC, UCATPROC, NVSMPROC, GDGPROC

2. For each item on the portable data set, IDCRM01 reads a timestamp record and prints a message indicating the time and data of the EXPORTRA operation. On the basis of the timestamp record, one of CLUSPROC, UCATPROC, NVSMPROC, or GDGPROC is called to actually import the object. If the read for a timestamp record should fail, IDCRM01 assumes that an end-of-file has been found on the portable data set and passes control to Executive Controlled Termination.

Diagram 3.16.1 IMPORTRA FSR – CLUSTER or ALTERNATE INDEX



Extended Description for Diagram 3.16.1

Module: IDCRM01

Procedure: CLUSPROC, CPLPROC, GETPROC, FVTPROC, BFPLPROC, BPASPROC, IUNIQPRC

1. CLUSPROC via CPLPROC builds a CTGPL for a define operation. CLUSPROC issues a UGET macro to read the catalog control records and calls GETPROC to read the catalog data records. Control records are read for the cluster or alternate index and their data and index, if any, components. CLUSPROC then calls FVTPROC to build two or three FVTs. FVTPROC in turn calls BFPLPROC to build FPLs for the catalog information on the portable data set. BPASPROC builds an FPL for security information. If the data or index component was originally defined as unique, IUNIQPRC builds a null volume FVT. The OBJECTS list is scanned for volume information about the object to be defined; if found, such information overrides that found on the portable data set. The volumes of the data set are dynamically allocated if necessary. If OUTFILE was not provided by the user, the data set just defined is dynamically allocated.

Module: IDCRM01

Procedure: CTLGPROC, DELTPROC

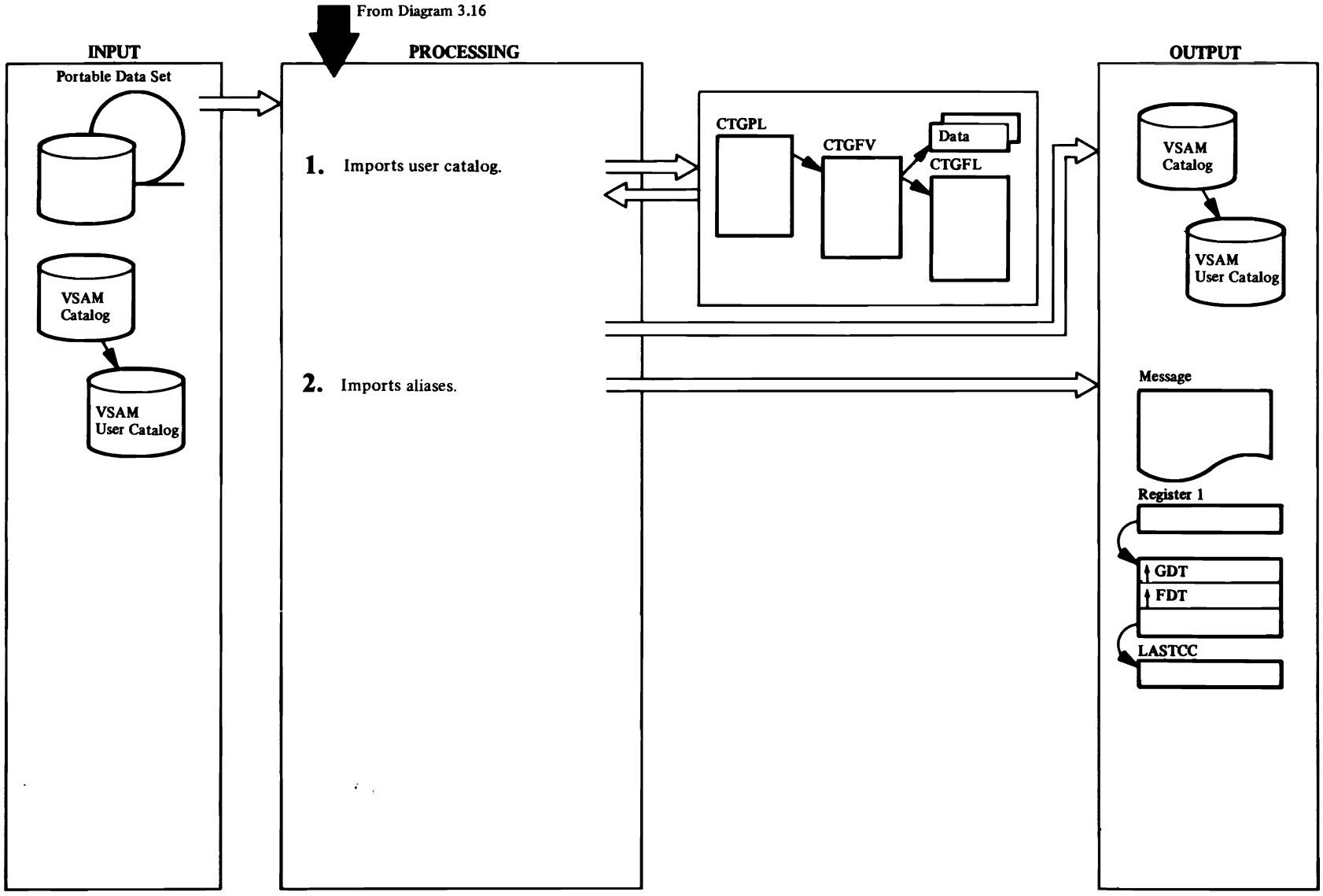
2. CTLGPROC issues a UCATLG macro to invoke VSAM catalog management. If VSAM issues a return code of 8, DELTPROC issues a UCATLG to delete the object from the catalog. Should this operation fail or should the original define fail with a return code other than 88, an error conversion table is built for a delete function. CTLGPROC calls the UERROR macro to print the prose error message and passes control back to IDCRM01 for the next object. an error conversion table is built for a delete function.

Module: IDCRM01

Procedure: ALTRPROC

3. If OUTFILE was specified, ALTRPROC renames the VSAM object to be loaded to the dummy name returned by the UIOINFO. A UOPEN macro is issued to open the VSAM object, and UCOPY is used to copy data records from the portable data set to the VSAM object. UCLOSE closes the VSAM object, and ALTRPROC alters the name of the object just loaded back to that under which it was defined, if necessary. Processing returns to Diagram 3.16, step 2, for the next item on the portable data set.

Diagram 3.16.2 IMPORTRA FSR – USERCATALOG



Extended Description for Diagram 3.16.2

Module: IDCRM01

Procedure: CPLPROC, UCATPROC, GETPROC, LVLRPROC, NFVTPROC, CTLGPROC, CPLPROC, DELTPROC

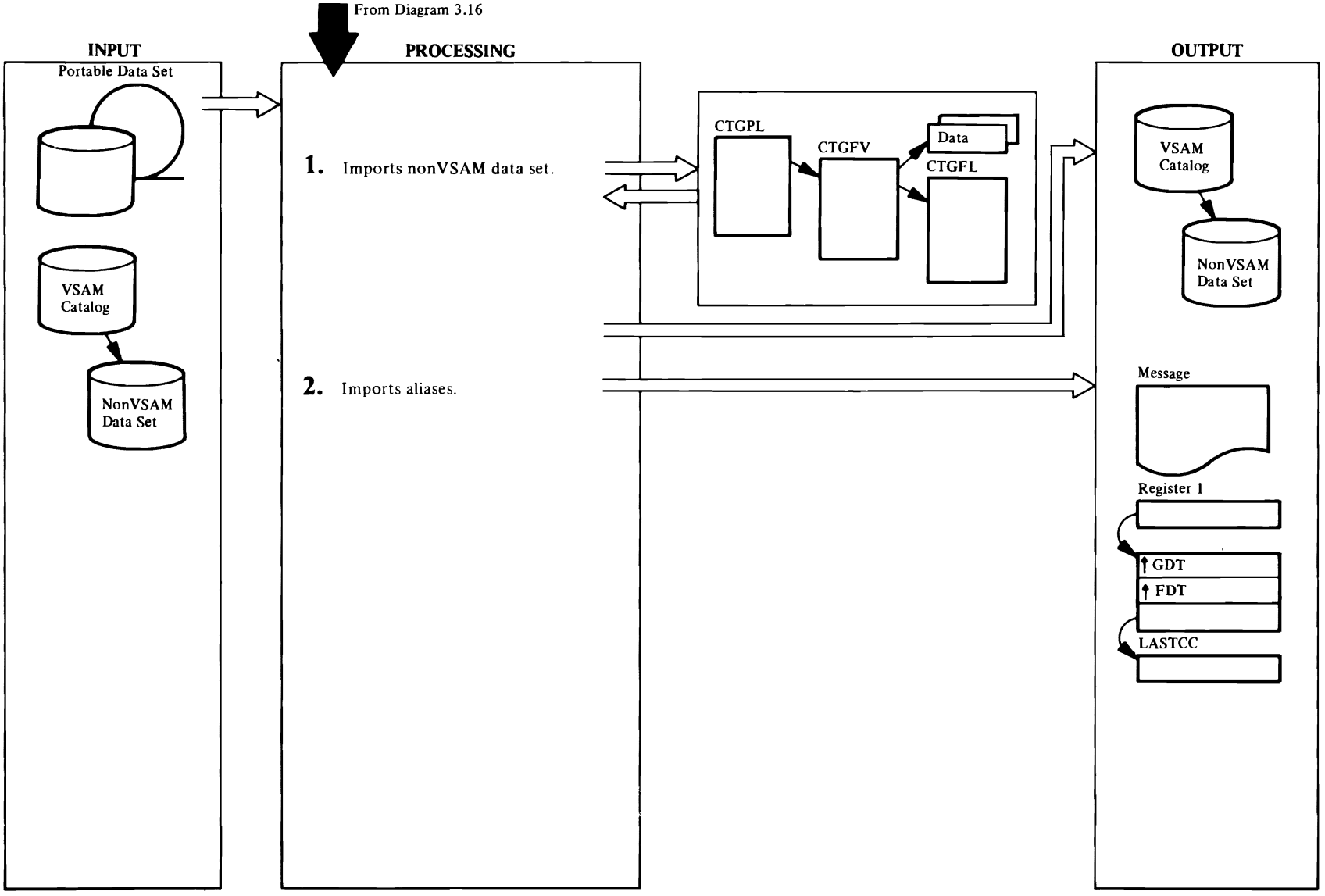
1. CPLPROC builds a CPL to be used to connect the user catalog pointer. UCATPROC then issues a UGET to get the catalog control record and calls GETPROC to obtain the catalog data record. LVLRPROC builds a DEVTYPE FPL and a volume serial list on the basis of information supplied on the portable data set or furnished through the OBJECTS parameter. NFVTPROC builds an FVT for the define. CTLGPROC issues a UCATLG macro to connect the user catalog. If the VSAM catalog return code is 8, then CPLPROC builds a CPL to do a disconnect operation, and DELTPROC actually invokes catalog to perform this operation. Should this succeed, a second attempt is made to connect the user catalog.

Module: IDCRM01

Procedure: ALISPROC, CPLPROC, NFVTPROC, CTLGPROC

2. For each alias item on the portable data set, CPLPROC builds a CPL and NFVTPROC builds an FVT. Then CTLGPROC issues a UCATLG to define the alias object.

Diagram 3.16.3 IMPORTRA FSR – NONVSAM



Extended Description for Diagram 3.16.3

Module: IDCRM01

Procedure: CPLPROC, NVSMPROC, GETPROC, LVLRPROC, NFVTPROC, CTLGPROC, DELTPROC

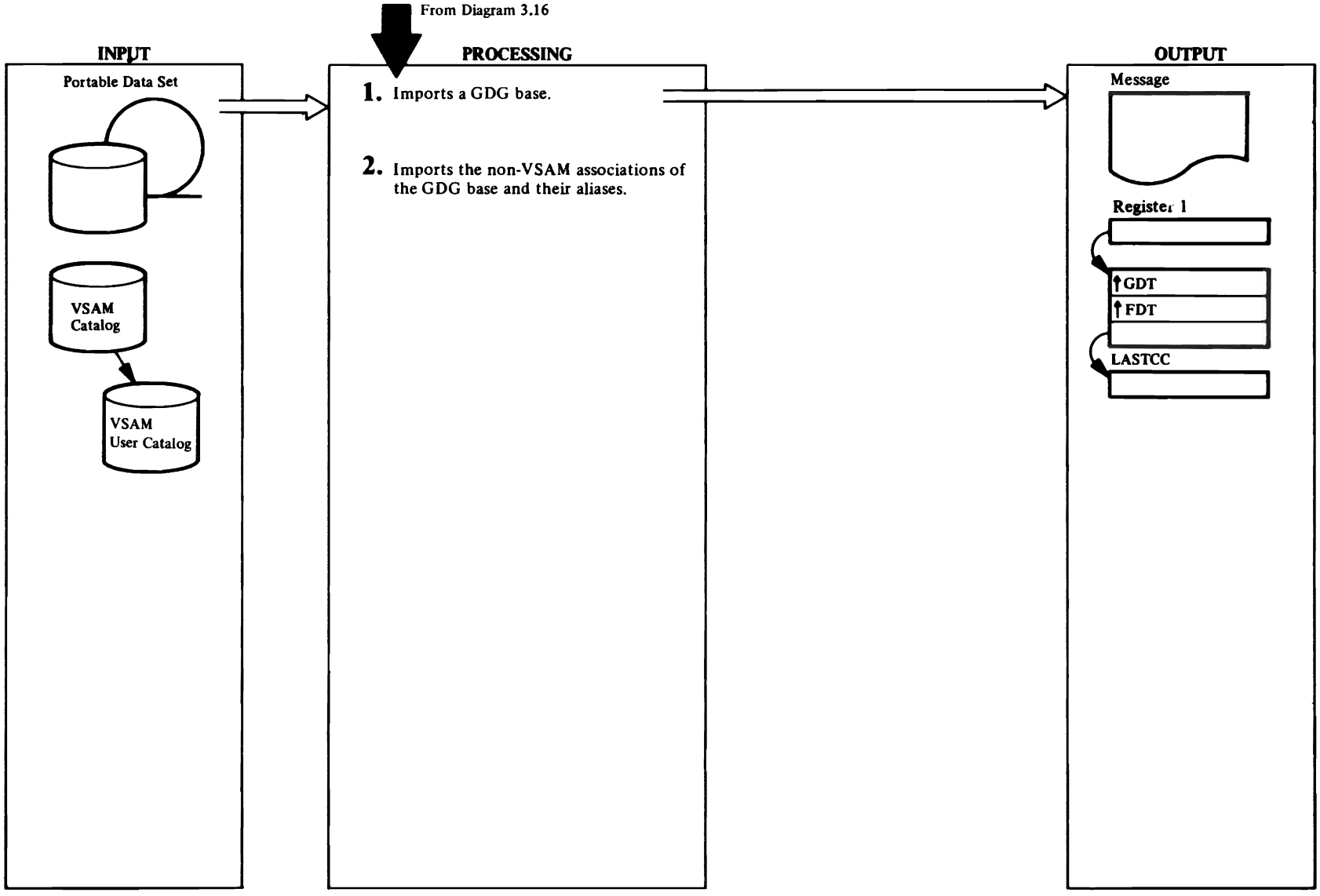
1. CPLPROC builds a CPL to be used to define the nonVSAM data set. NVSMPROC then issues a UGET to get the catalog control record and calls GETPROC to obtain the catalog data record. LVLRPROC builds a DEVTYPE FPL and a volume serial list on the basis of information supplied on the portable data set or furnished through the OBJECTS parameter. NFVTPROC builds an FVT for the define. CTLGPROC issues a UCATLG macro to define the nonVSAM data set. If the VSAM catalog return code is 8, then CPLPROC builds a CPL to do a delete operation, and DELTPROC actually invokes catalog to perform this operation. Should this succeed, a second attempt is made to define the nonVSAM data set.

Module: IDCRM01

Procedure: ALISPROC, CPLPROC, NFVTPROC, CTLGPROC

2. For each alias item on the portable data set, CPLPROC builds a CPL and NFVTPROC builds an FVT. Then CTLGPROC issues a UCATLG to define the alias object.

Diagram 3.16.4 IMPORTRA – GDG BASE



Extended Description for Diagram 3.16.4

Module: IDCRM01

Procedure: GDGPROC, CPLPROC, GETPROC, CTLGPROC, DELTPROC, BFPLPROC

1. CPLPROC builds a CPL to be used to define the GDG base. GCGPROC then issues a UGET to get the catalog control record and calls GETPROC to obtain the catalog data record. NFVTPROC builds an FVT for the define operation, and calls BFPLPROC to build FPLs for the fields exported by EXPORTRA. CTLGPROC issues a UCATLG to macro to define the GDG base. If the VSAM catalog return code is 8, then CPLPROC builds a CPL to do a delete operation, and DELTPROC invokes catalog ko perform this operation. Should this succeed, a second attempt is made to define the GDG base.

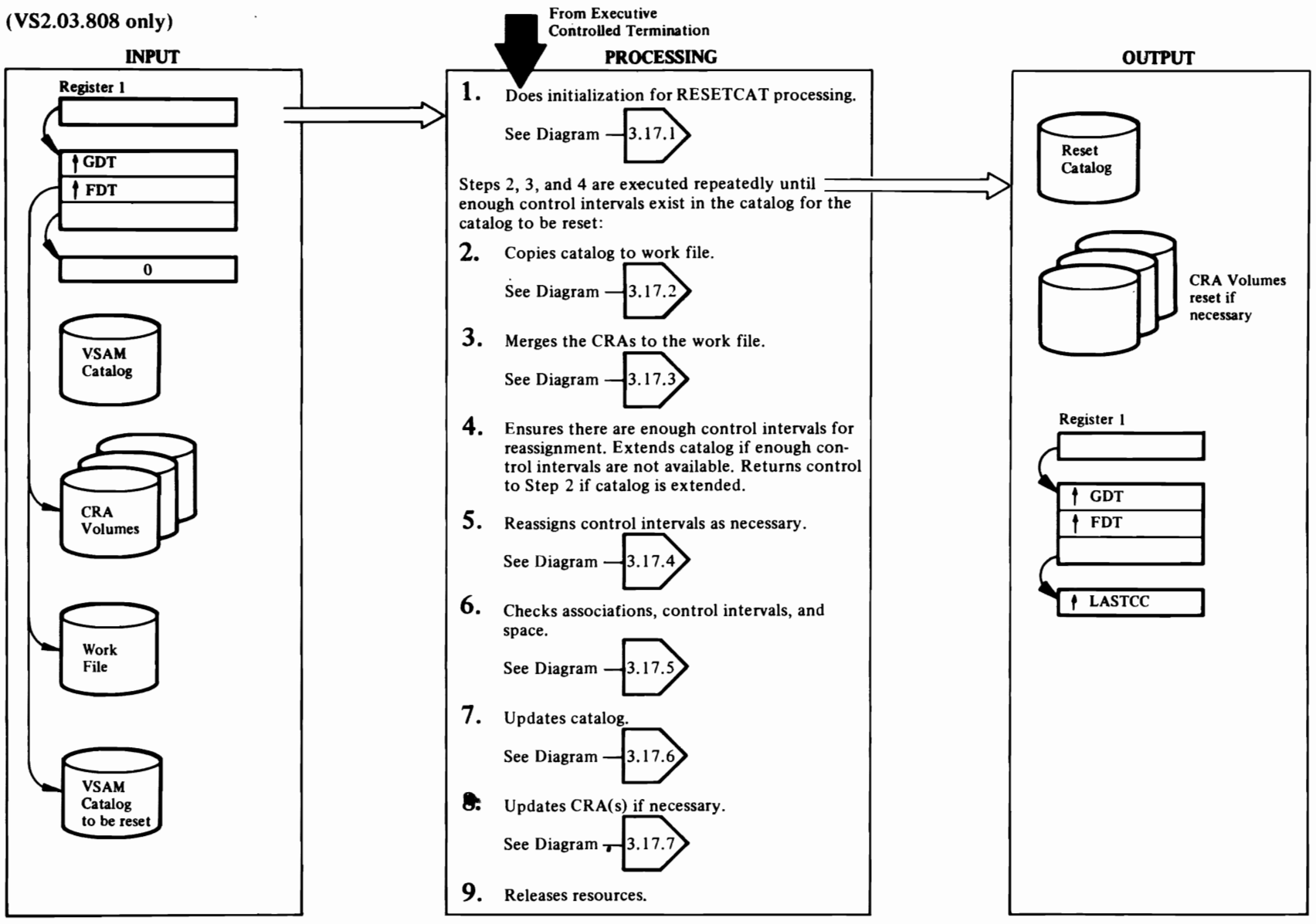
Module: IDCRM01

Procedure: NVSMPROC, ALISPROC

2. For each nonVSAM object associated with the GDG base, NVSMPROC is called. NVSMPROC imports the nonVSAM association, and calls ALISPROC to import any aliases of the nonVSAM.

Diagram 3.17 RESETCAT FSR

(VS2.03.808 only)



Extended Description for Diagram 3.17

(VS2.03.808 only)

Module: IDCRS01, IDCRS06

Procedures: INIT, DSOPEN, CATINIT, WFDEF, CRAMNT

1. INIT is the first procedure called by RESETCAT. It uses the UGPOOL macro to obtain work areas common to all of RESETCAT, and initializes them. The catalog to be reset is opened, verified, and validity checked. Next, exclusive control over the catalog is obtained via the URESERVE macro. The work file is defined and opened. An entry in the RESVOL table is created for each CRA volume identified by the CRAFILES parameter or the CRAVOLS parameter. All volumes specified for reset via CRAVOLS are allocated dynamically. Finally, INIT builds the CIXLT table. The CIXLT table is used to translate a catalog control interval number into a work file relative record number.

The following three steps, Steps 2, 3, and 4, form an interactive loop. These three steps are executed repeatedly until the catalog to be reset has enough control intervals.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: COPYCAT, BLDVLST, SCNRLST, DSCLOSE

2. COPYCAT performs the initial load of the work file from the catalog to be reset. The CIXLT table built by INIT maps every catalog DATA control interval number (CIN) to a relative record number (RRN) slot in the work file. It also indicates whether the control is for the low key range (LKR) or high key range (HKR) portions of the catalog. LKR records from the catalog are written to the work file as normal RRDS records. HKR records are also written to the work file; however, for each HKR record written, a flag is set indicating that control interval will later be reassigned. Dummy records (formatted control intervals with no data in them) are written to the work file to represent that portion of the catalog which extends from the first unformatted free control interval to the LKR high allocated control interval. A table (VOLSERTB) is built from all volume records read from the catalog. Free records and records which belong to a CRA specified for reset are maintained on an "available" chain and an "available" count is kept for these records. When processing is completed, the work file is closed.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: MERGECRA, DSOPEN, SCNRLST, CKERR, PROCCRA, VOLCHK, DSCLOSE

3. MERGECRA merges each reset CRA into the work file. Each CRA is opened. The cluster record is read, and the catalog name is verified. The PROCCRA procedure is called to merge the CRA records into the work file, and the VOLCHK procedure is called to perform the volume consistency check.

Module: IDCRS01, IDCRS05, IDCRS06, IDCRS07

Procedures: ENSURECI, DSCLOSE, CATEOV, CKERR, DSOPEN, CATINIT

4. ENSURECI ensures that there are enough free control intervals for reassignment. If the number of control intervals to be reassigned is less than or equal to the number of control intervals available, a flag, RSENUFCI, is set, indicating that enough control intervals are available for reassignment. However, if the control intervals to be reassigned are greater than the number available, ENSURECI forces the extension of the catalog by performing the following:

   The catalog is closed by calling DSCLOSE. Next, all storage obtained during COPYCAT processing is freed by issuing UFPOOL. The highest formatted work file relative record number is saved in RSWFHURR, and CATEOV is called to extend the catalog by writing free records into the catalog until the catalog has been extended and sufficient control intervals are available for the reset operation. If CATEOV returns with an error condition, CKERR is called to terminate RESETCAT processing.

   After the catalog is successfully extended, DSOPEN is called to re-open and verify the catalog. CATINIT is called to re-establish the catalog's geometry by building the CI to RRN translate table (CIXLT).

Module: IDCRS01, IDCRS05

Procedures: REASSIGN, ADDUPCR

5. The REASSIGN procedure performs control interval (CIN) reassignment. The invalid and duplicate records on the reassign chain are assigned to valid CINs from the available chain. Each record on the reassign chain is read and an "available" record from the available chain is found. The reassign record is copied to the "available" record buffer; the CIN is changed to reflect the CIN of the "available" record. If there is a pointer to a duplicate record (DUPPTR), it is copied from the reassign record's processing field. The "available" record is then updated to reflect the reassigned record. The record whose DUPPTR points to the reassigned record's relative record number is found by following the duplicate record chain. The DUPPTR of this record is changed to reflect the "available" record's CIN. This record is then updated.

Module: IDCRS02, IDCRS03

Procedures: ASSOC, PROCTYPE, VERDSDIR, PROCVOL

6. The ASSOC procedure controls the checking of all control interval numbers (CIN) in all records being reset. This includes CINs in associations and data set directories. ASSOC also controls the checking for any space conflicts of VSAM data sets.

Module: IDCRS01, IDCRS05, IDCRS07

Procedures: UPDCAT, CKERR, ADDUPCR, ENTNMCK, SCNRLST, RENAMEP, UPDCCR, CRAUPCHN, DELTN, ADDTN

7. UPDCAT updates the catalog from the work file. At this point, any records in the work file which do not match the catalog must be written to the catalog. Each valid work file record is read and, if the "update catalog" flag is on, the record is written to the catalog low key range (LKR). True names are deleted from and added to the catalog high key range (HKR) as necessary. If the "update CRA" flag is on, the control interval of the work file record is placed on the CRA update chain. Then the free record chain is rebuilt.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: UPDCRA, SCNRLST, DSOPEN, DSCLOSE, CKERR

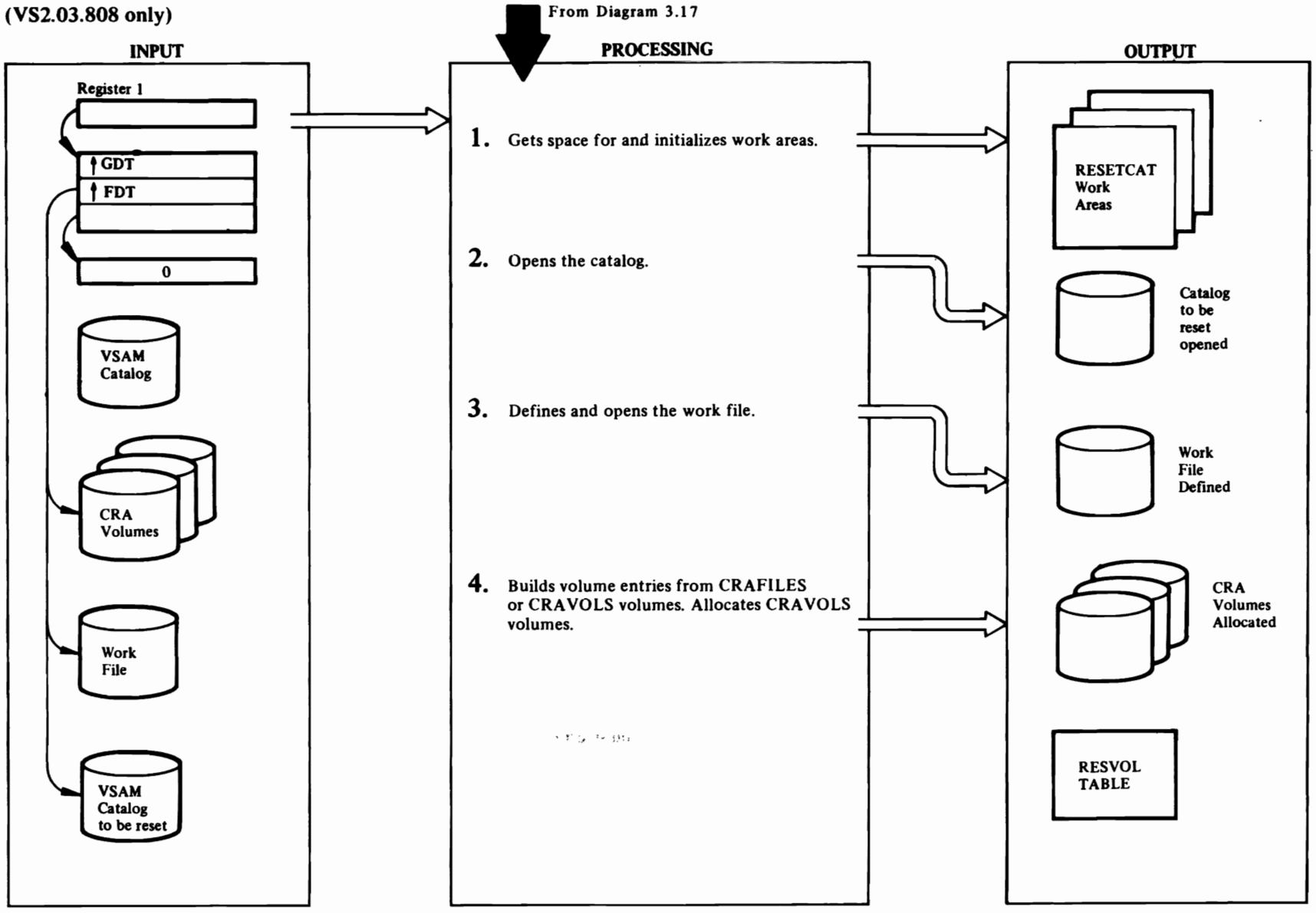
8. UPDCRA updates CRAs from the work file. Each entry in RESVOL (a table containing an entry for each volume whose CRA is required in the reset operation) is obtained. If there are any updates to be made in that CRA, it is opened, updated, and closed. If any free records are placed in the CRA, the CCR record is updated. If the RESVOL entry indicates that the CRA volume was allocated, it is deallocated.

Module: IDCRS01, IDCRS05

Procedures: WRAPUP, CLEANUP, CKERR

9. If RESETCAT processing is successfully completed, WRAPUP is the last procedure called. WRAPUP ensures that all resources obtained by RESETCAT are freed, it prints the message that processing is complete, and then returns control to the system.

Diagram 3.17.1 RESETCAT FSR – Initialization
(VS2.03.808 only)



Extended Description for Diagram 3.17.1

(VS2.03.808 only)

Module: IDCRS01

Procedure: INIT

1. INIT issues the UGPOOL macro to obtain storage for the following work areas:

   - Record Management control blocks (GRAB,BUFFER)
   - Control Blocks for Catalog Management LOCATE macro (CPLs and FPLs)

   The FDT is checked to see if IGNORE is specified; if so, a flag (RSIGNORE) is set in RSWORK.

   After obtaining the above storage, INIT formats the RESETCAT record management control blocks. Control blocks (CPL and FPL) of Catalog Management are also formatted along with certain portions of the main work area.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: INIT, DSOPEN, CKERR

2. DSOPEN is called to open the catalog to be reset. Validity checks are made on the catalog to ensure that it is recoverable and is not the master catalog. CKERR is called if these checks fail.

   Exclusive use of the catalog is ensured by issuing two umacros: USYSINFO and URESERVE. USYSINFO gets the catalog UCB address and URESERVE obtains exclusive use of the ENQ name of the catalog (SYSIGGV2/*catname*). If it is determined that the catalog is being used by someone else (by checking the count in the CAXACT field), CKERR is called to terminate RESETCAT processing.

   DSOPEN is called to perform a VERIFY operation on the catalog; the high used RBA of the catalog is adjusted if necessary. UGPOOL is issued to obtain storage for the CIXLT table.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: INIT, RECMGMT, WFDEF, DSOPEN, CKERR

3. RECMGMT is called (with the GETRCD option) to get control interval zero (CI=0) from the catalog. The high allocation data CI is computed (HARBADS/512) and saved in RSCAHACI.

   The primary and secondary extents of the work file are computed as follows:

   Primary = no. of records currently allocated in the catalog.

   $$\text{Secondary} = \frac{(\text{MAXCI}*2 - \text{primary}) + 125}{126}$$

   where MAXCI = Largest CI number possible for a catalog.

   The WFDEF procedure is called to define the work file. If it is found that the work file is defined in the catalog being reset, CKERR is called.

   DSOPEN is now called to open the work file.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: INIT, CKERR, CATINIT, CRAMNT

4. The RESVOL table is constructed consisting of an entry for each CRA volume supplied by the invoker of RESETCAT with the CRAFILES parameter or CRAVOLS parameter. Each entry consists of fields for volume serial number, device type, and *ddname*. A pointer, RSVOLALL, points to the first entry in the table and each entry is chained to the next. A flag indicates the last 'ALL' entry which is followed by the 'NONE' entries.

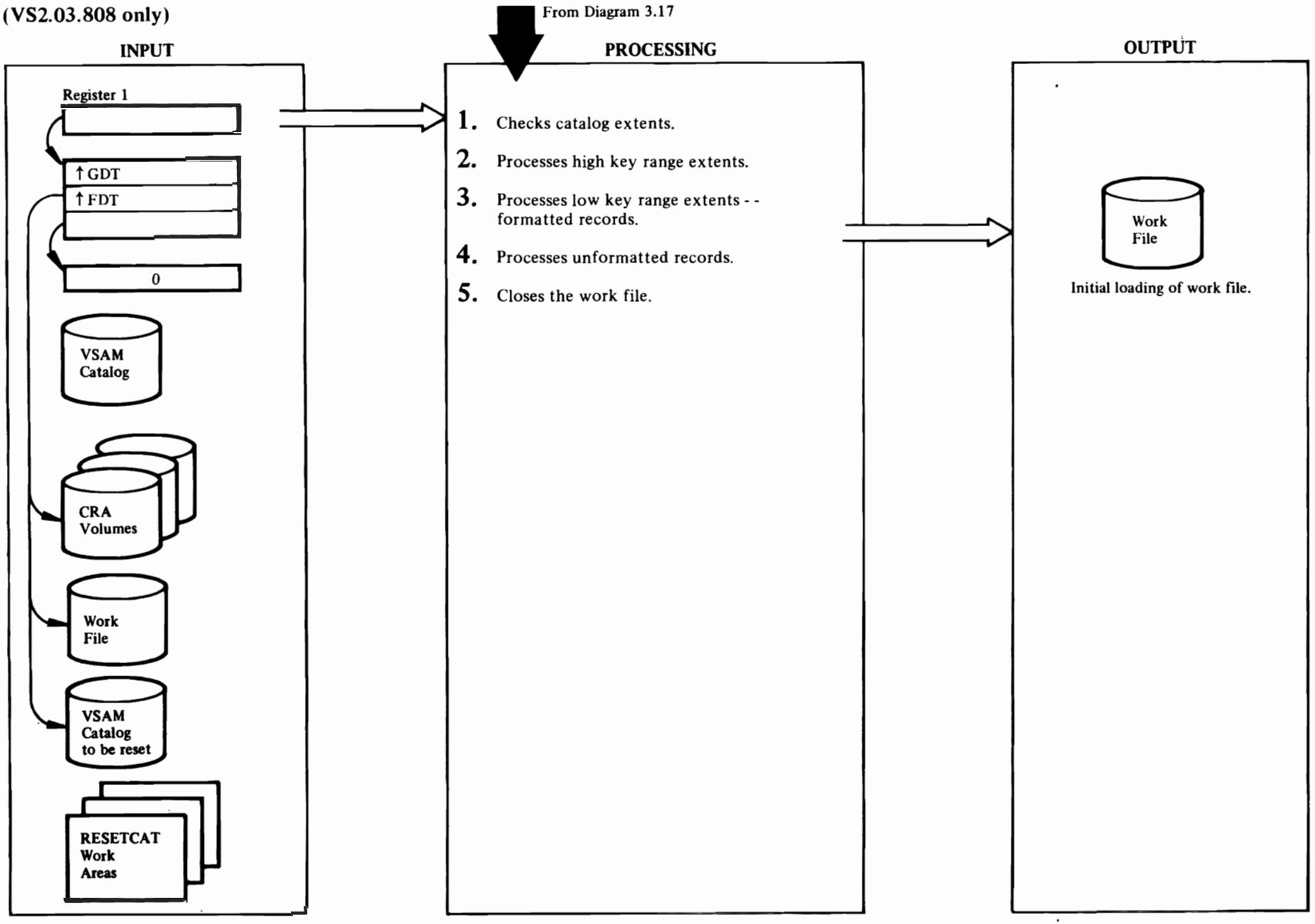
   The FDT is checked to see if CRAVOLS or CRAFILES is specified. If CRAVOLS is specified, CRAMNT is called to allocate the volumes. The "CRA allocated" audit flag is set and the *ddname* of the volume is inserted in the RESVOL table.

   If CRAFILES is specified, the volume serial number of the CRA is obtained via the UIOINFO macro. The volume serial number of the CRA is inserted in the RESVOL entry. If the catalog volume serial number is specified, its RESVOL entry is positioned as the first entry in the list.

   If no CRA is specified, CKERR is called to flag an error condition.

   CATINIT is now called to build the CIXLT table. The CIXLT maps the catalog control intervals to the work file relative record numbers. There is an entry in CIXLT for each catalog extent.

Diagram 3.17.2 RESETCAT FSR – Copy catalog to work file (VS2.03.808 only)



Extended Description for Diagram 3.17.2 (VS2.03.808 only)

Module: IDCRS01

Procedure: COPYCAT

1. The COPYCAT procedure obtains each entry from CIXLT and examines it to see if the first control interval number in the entry is greater than the catalog low key range (LKR) high allocated control interval. If so, it indicates COPYCAT processing is complete and control returns to the main procedure, IDCRS01.

   Another test is made to see if all 127 entries have been processed; if so, control returns to main line IDCRS01 processing.

2. If the CIXLT entry represents a high key range (HKR) extent, a flag is set indicating that this is an "invalid" record in the work file. A dummy record is formatted and written to the work file as follows:

   - If the relative record number (RRN) is greater than the high formatted relative record number in the work file, RECMGMT(ADDRCD) is called to add the record to the work file.
   - If the RRN is not greater, RECMGMT(UPDRCD) is called to update the record in the work file.

3. If the CIXLT entry represents a LKR extent, the record is processed as a formatted record. If the CI of the record is less than the next free unformatted catalog CI, then GETRCD of the RECMGMT procedure is called to read the record from the catalog. The catalog record is moved to the work file buffer. If the record happens to be a free record (not currently used in the catalog), it is placed on the available chain. The count of available records is incremented. If it is not a free record and if it is a volume record, then a VOLSERTB entry consisting of volume serial number and CI number is formatted. BLDVLST is called to add this entry to the VOLSERTB table. In order to check to see if the record is also on a CRA specified for reset, SCNRLST is called. If it is a CRA record, a flag is set indicating that the record is to be deleted. The record is placed on the available chain and the available count is incremented. LKR records are written to the work file as follows:

   - If the RRN is greater that the high formatted RRN, ADDRCD is called to add the record to the work file.
   - If the RRN is not greater, then UPDRCD is called to update the record in the work file.

4. If the CI of the record is equal to or greater than the next free unformatted CI in the catalog, then the "update catalog" flag is set in the work file processing field and a dummy free record is formatted. The dummy record is placed on the available chain and the available count is incremented. If the CI of the record is equal to or greater than the End of Volume unformatted free CI, then the "invalid" flag is set in the work file processing field. A dummy record is formatted. The unformatted dummy record is written to the work file as follows:

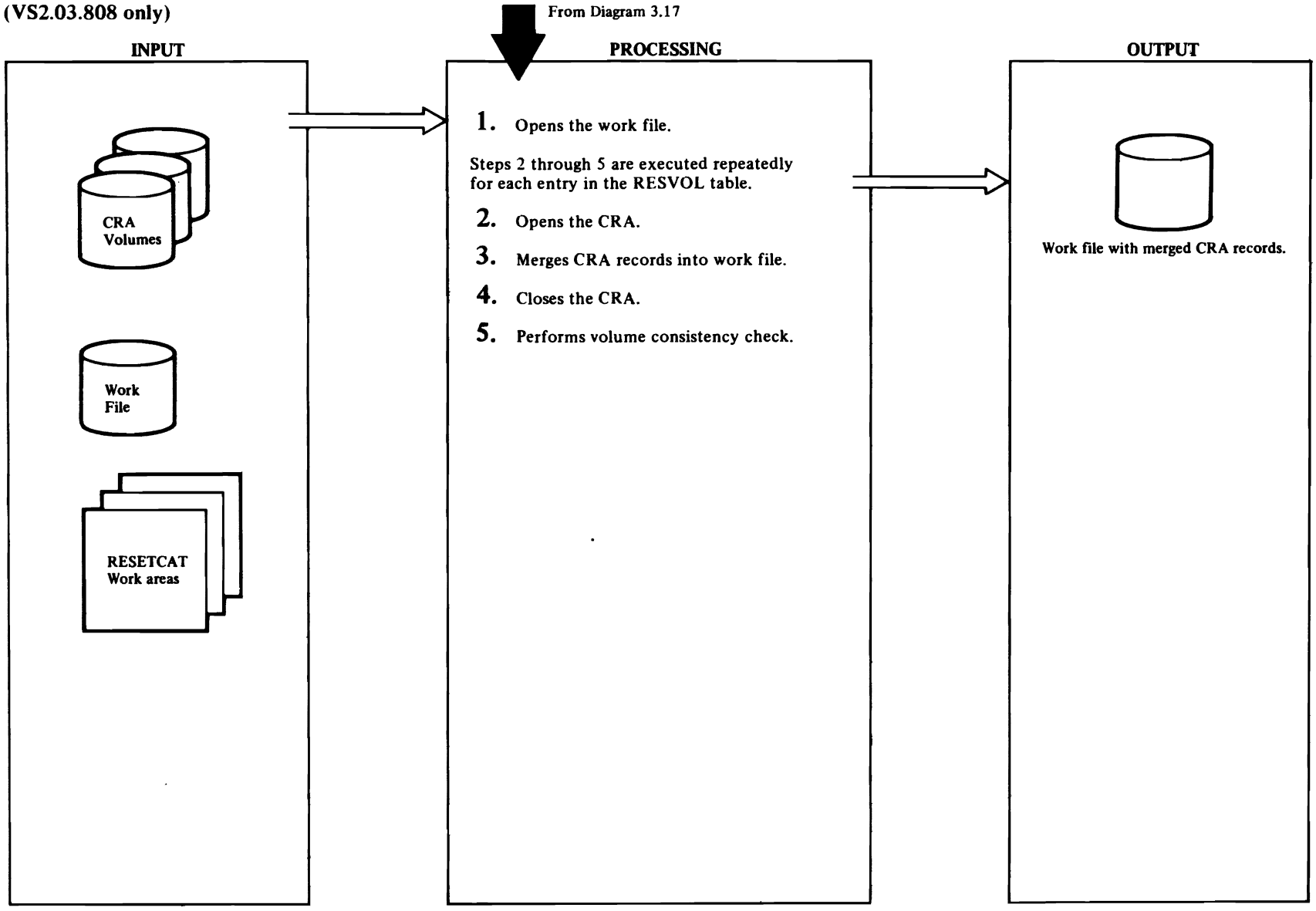
   - If the RRN is greater than the high formatted RRN, then ADDRCD is called to add the record to the work file.
   - If the RRN is not greater, UPDRCD is called to update the record in the work file.

Module: IDCRS01, IDCRS06

Procedures: COPYCAT, DSCLOSE

5. The "work file created" flag is tested; if it is off, DSCLOSE is called to close the work file.

Diagram 3.17.3 RESETCAT FSR – Merge CRS(s) to the work file.
(VS2.03.808 only)



Extended Description for Diagram 3.17.3

(VS2.03.808 only)

Module: IDCRS01, IDCRS06

Procedures: MERGECRA, DSOPEN

1. The "work file open" flag is tested to see if the work file is already open; if off, DSOPEN is called to open the work file.

   Steps 2 through 5 form an iterative loop. These four steps are executed repeatedly for each entry in the RESVOL table.

2. The SCNRLST procedure is called to obtain an entry from the RESVOL table indicating the volume serial number of a CRA specified for the reset operation. If SCNRLST finds that all entries are processed and if the "termination" flag is on, CKERR is called to print an error message and terminate processing. If SCNRLST successfully returns a CRA volume serial number, DSOPEN is called to open this CRA. If open fails, flags are set to terminate processing and to bypass the volume consistency check. If the open is successful, RECMGMT (with GETRCD option) is called to read the CRA cluster record (CI=2). If the CRA entry name is not for the catalog being reset, then CKERR is called to print an error message. Flags are set to terminate processing and to bypass the volume consistency check.

Module: IDCRS01

Procedures: MERGECRA, PROCCRA

3. PROCCRA is called to merge CRA records into the work file.

   Beginning with the volume record, each CRA record is read and merged. The CIN of the volume record is updated/added to VOLSERTB, so that Volume records may be located later. The work file record corresponding to the catalog control interval (CATCI) of each CRA record (except CRA free records) is read. If the work file record is free or available, the CRA record replaces it. If the work file record has already been replaced or if the work file record does not belong to a reset CRA, the CRA record is written to the overflow area and maintained on the duplicate chain for that CATCI. Records written to the overflow or "invalid" areas of the work file are placed on the "reassign chain" and a "reassign count" is kept for these records. Each time a free or available work file record is replaced, the "available" count is decremented.

Module: IDCRS01, IDCRS06

Procedures: MERGECRA, DSCLOSE

4. If the "CRA open" flag is set, DSCLOSE is called to close the CRA. If the close fails, flags are set to terminate processing and to bypass the volume consistency check.

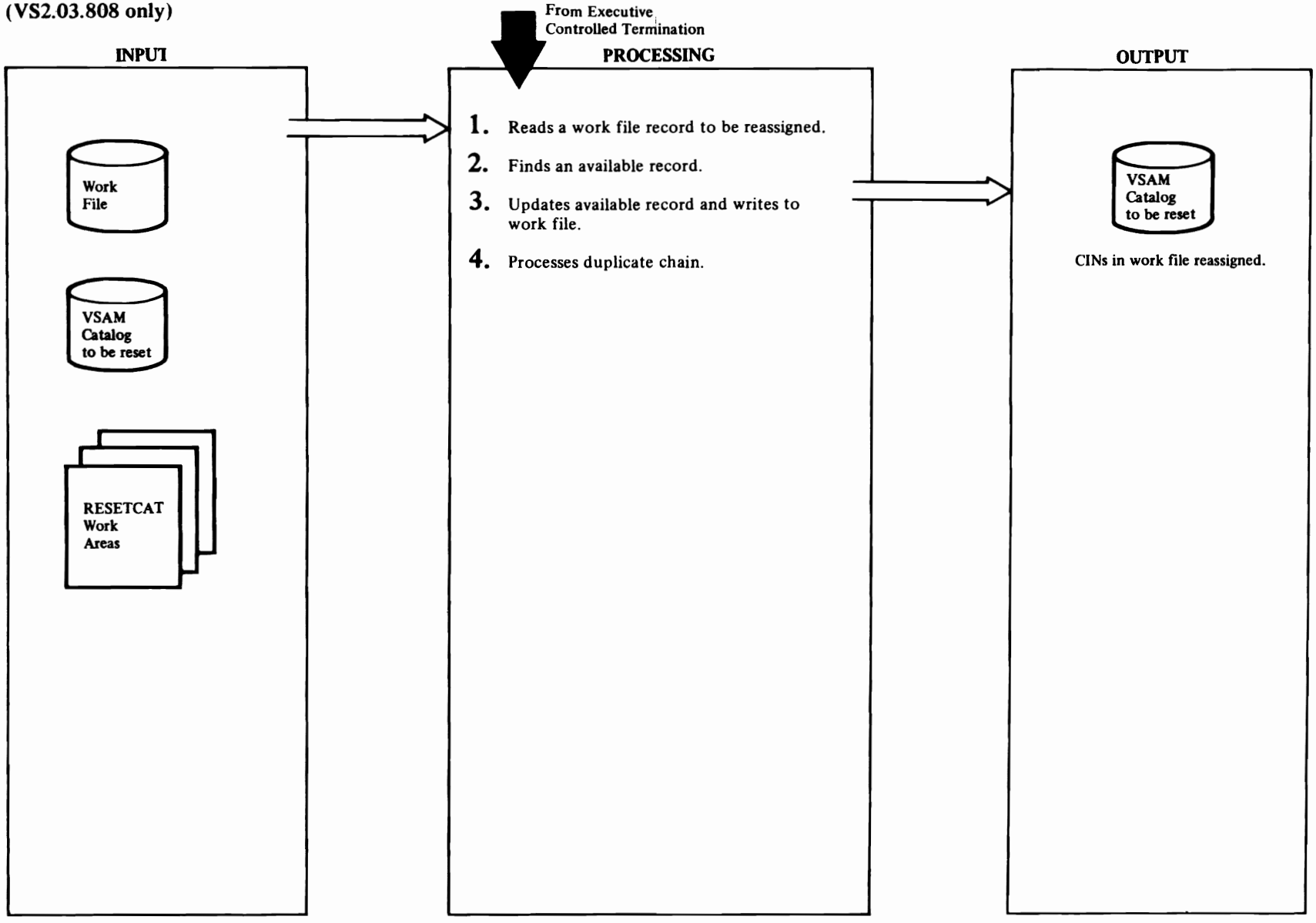
Module: IDCRS01, IDCRS03

Procedures: MERGECRA, VOLCHK

5. If the flag to bypass the volume consistency check is not on, VOLCHK is called to perform the volume consistency check.

   VOLCHK ensures that there is a one-to-one correspondence between each VSAM data space on a volume (format 1 DSCB in the VTOC) and each space header in the volume record for that volume. If a format 1 DSCB does not have a corresponding space header, the format 1 DSCB is scratched. If a space header refers to a non-existent format 1 DSCB, the space header is deleted. If the extents in a space header are not identical to the extents in the corresponding format 1 DSCB, the extents in the space header are corrected.

Diagram 3.17.4 RESETCAT FSR – Reassign CI numbers (VS2.03.808 only)



Extended Description for Diagram 3.17.4 (VS2.03.808 only)

Module: IDCRS01, IDCRS06

Procedures: REASSIGN, RECMGMT

1. Before it reassigns any records, the REASSIGN procedure determines whether any records need to be reassigned. If the reassign count is zero, it means no records need to be reassigned. Control is returns to mainline IDCRS01 processing. Control is also returned if all records on the reassign chain have been read.

   RECMGMT (with GETRCD option) is called to read the next record on the reassign chain. The reassign chain pointer is saved.

Module: IDCRS01, IDCRS06

Procedures: REASSIGN, RECMGMT

2. The next record on the available chain is read via GETRCD. The available chain pointer is saved. If the "replaced from CRA" flag is set, then this record cannot be used, so the next record on the available chain is read until an available record is found.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: REASSIGN, ADDUPCR, RECMGMT

3. The reassign record is moved to the available record buffer. The reassign DUPPTR is copied to the available DUPPTR. Two flags, "replaced from CRA" and "update catalog", are set. ADDUPCR procedure is called to perform CRA update processing. A flag indicating that the record is reassigned is set.

   RECMGMT (with the UPDRCD option) is called to write the updated available record to the work file.

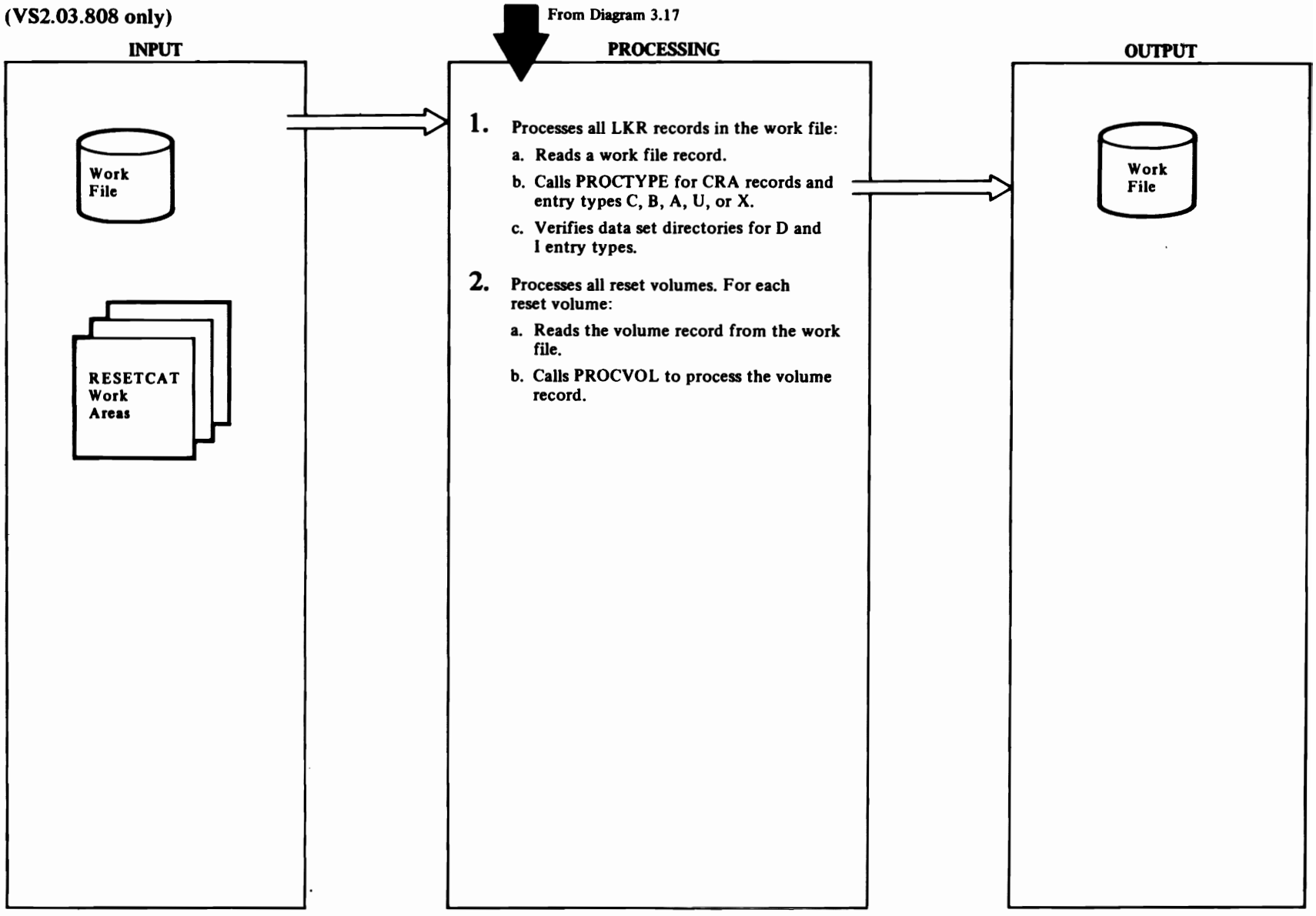
Module: IDCRS01, IDCRS06

Procedures: REASSIGN, RECMGMT

4. The relative record number (RRN) of the reassigned record is saved. RECMGMT (GETRCD) is called to read the record pointed to by the catalog control interval of the reassigned record or the DUPPTR.

   If the DUPPTR does not point to the RRN of the reassigned record, then the next record on the duplicate record chain is read. When the record is found, the DUPPTR is updated to point to the CI of the available record. RECMGMT (UPDRCD) is called to write the record back to the work file.

Diagram 3.17.5 RESETCAT FSR – Check Associations (VS2.03.808 only)



Extended Description for Diagram 3.17.5 (VS2.03.808 only)

Module: IDCRS02, IDCRS06

Procedures: ASSOC, RECMGMT, PROCTYPE, VERDSDIR

1.a. Each work file record is read sequentially up to the high allocated catalog control interval. Each record is checked to see if the "associations checked" flag is on. If it is, control goes to step 2.

b. If the flag is not on and if the record is from a CRA being reset, then, for each C, B, A, U, or X record, the PROCTYPE procedure is called to process control interval numbers.

For a given catalog entry type, PROCTYPE controls the process of scanning a catalog record for control interval numbers. It determines which other records which along with the given record are a part of a set of records. It verifies all control interval numbers in the entire set of records. Control interval numbers are also corrected if necessary.

c. VERDSDIR is called to check data set directories if the entry type is D or I. The VERDSDIR procedure verifies the data set directory entries for VSAM data sets which are not on reset volumes. It specifically looks for multivolume VSAM data sets where the primary volume is not a reset volume but a secondary volume is a a reset volume. VERDSDIR changes work file records to correct error conditions; namely, it marks a volume group occurrence (VGO) unusable when no data set directory exists for that data set.

Module: IDCRS02, IDCRS06

Procedures: ASSOC, RECMGMT

2.a. For each reset volume, the volume record is read from the work file via RECMGMT(GETRCD).

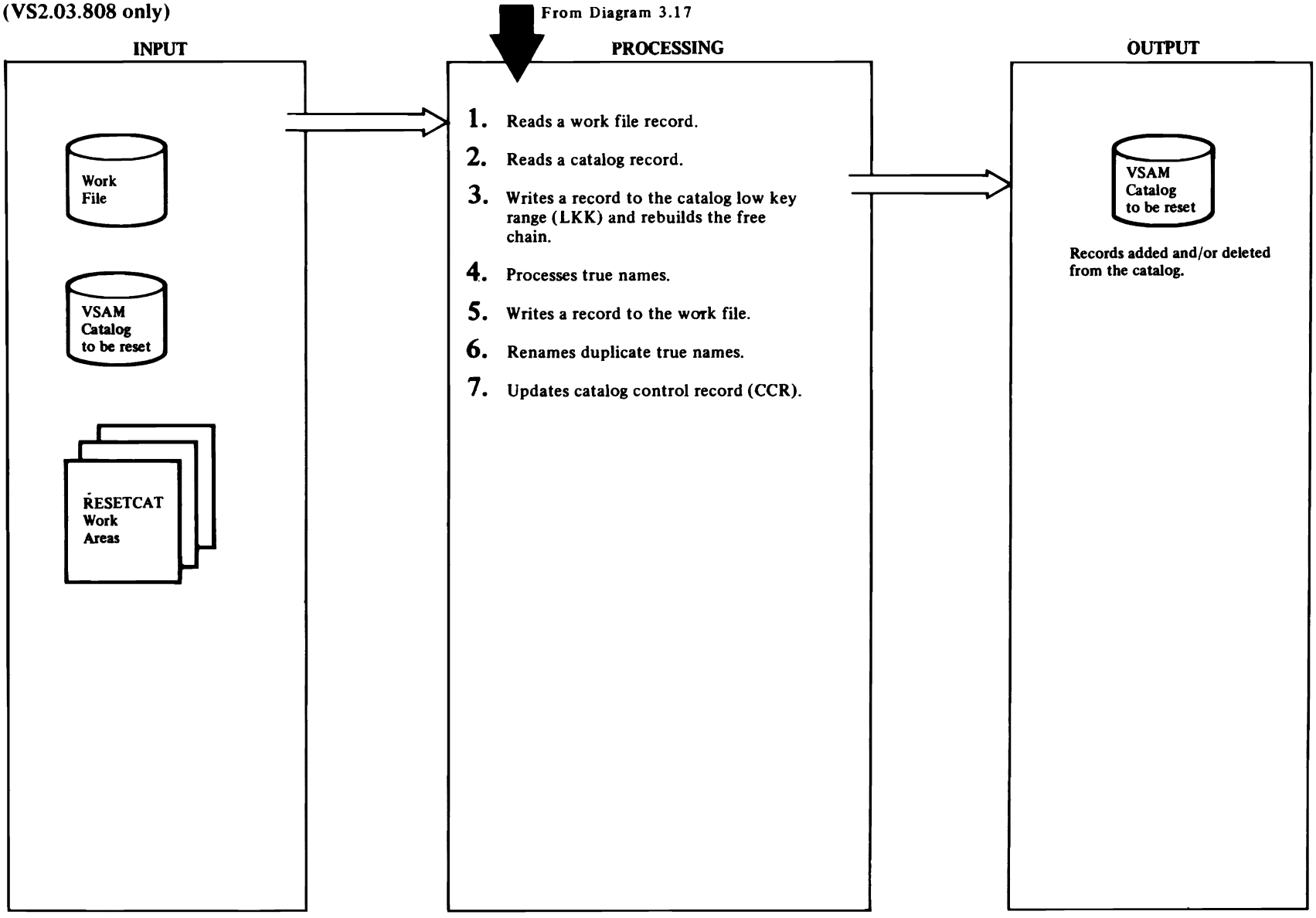
Module: IDCRS02, IDCRS06

Procedures: ASSOC, RECMGMT, PROCVOL PROCTYPE, VERDSDIR

2.b. The PROCVOL procedures is called to process the volume record.

PROCVOL controls the checking of space conflicts for each volume record. PROCVOL calls PROCTYPE to find and verify each control number in a volume record and its extensions. PROCVOL verifies and, if necessary, corrects the volume space bit map.

Diagram 3.17.6 RESETCAT FSR – Update the Catalog (VS2.03.808 only)



Extended Description for Diagram 3.17.6 (VS2.03.808 only)

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: UPDCAT, CKERR, RECMGMT

1. UPDCAT ensures that all CRAs required for updating are available by checking the "update CRA unavailable" flag (RSBADVOL). If the check shows that a CRA is not available, the CKERR routine is called to print a message and terminate RESETCAT processing.

   Each catalog extent in the work file is processed by checking each entry in CIXLT. If the extent represents a HKR, it is ignored. Only LKR extents are considered. For each LKR extent, RECMGMT (GETRCD) is called to read a work file LKR record.

Module: IDCRS01, IDCRS06

Procedures: UPDCAT, RECMGMT

2. For each work file record read, the "update catalog" flag (RSWUPCAT) is tested and if the flag indicates the catalog should be updated, the corresponding catalog record is read via the GETRCD routine.

Module: IDCRS01, IDCRS06

Procedures: UPDCAT, ADDUPCR, RECMGMT

3. After each catalog record is read, the "association checked" flag (RSWASSCK) is tested. If it is not on, the ADDUPCR routine is called to prepare for update CRA processing. The ENTNMCK procedure is called to determine if the catalog record has a true name; if there is a true name, a flag is set and the true name is saved. Next, ENTNMCK is called again to see if the work file record has a true name. If it does, a flag is set.

   If the record is free or the "association checked" flag is off, a deleted free work file record is formatted in the catalog buffer and placed on the free chain; otherwise, the work file record is moved to the catalog LKR buffer. If the control interval number of the record is greater than or equal to the first unformatted free control interval, RECMGMT (ADDRCD) is called to add the record to the LKR. If the CIN is less than the first unformatted free CIN, the UPDRCD option of RECMGMT is called to update the catalog record.

   The free chain is rebuilt.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: UPDCAT, RECMGMT, DELTN, ADDTN

4. If the catalog record has a true name and the work file record does not (or has a true name different from the catalog), then the true name is deleted from the catalog HKR by calling DELTN, provided the CIN is correct.

   If the work file record has a true name and the catalog record does not (or has a true name different from the work file), ADDTN is called to write a true name record. If ADDTN indicates a duplicate record exists, the work file record is placed on the true name chain for a future rename operation (see Step 6). The "write work file" (RSUCTWWF) flag is set.

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: UPDAT, SCNRLST, RECMGMT, CRAUPCHN

5. UPDCAT checks to see if the "update CRA" flag (RSUPCRA) is on. If it is, the SCNRLST routine is called to scan the RESVOL table for the CRA volume serial number. Next, the work file record is placed on the CRA update chain for this CRA volume by the CRAUPCHN procedure. Thee "write work file" flag is set.

   If the "write work file" flag (RSUCTWWF) is on, UPDRCD is called to update the work file record with the true name chain pointer and/or the CRA update pointer.

Module: IDCRS01, IDCRS06, IDCRS07

Procedures: UPDCAT, RECMGMT, RENAMEP, ADDTN

6. After all the catalog LKR extents have been processed, the true name chain is checked. If the chain is not empty, the GETRCD routine of RECMGMT is called to read a work file record on the true name chain. The ADDTN routine is called to add the true name to the catalog HKR. If a duplicate name is detected, then the RENAMEP procedure is called to assign a new name to the true name.

Module: IDCRS01, IDCRS06

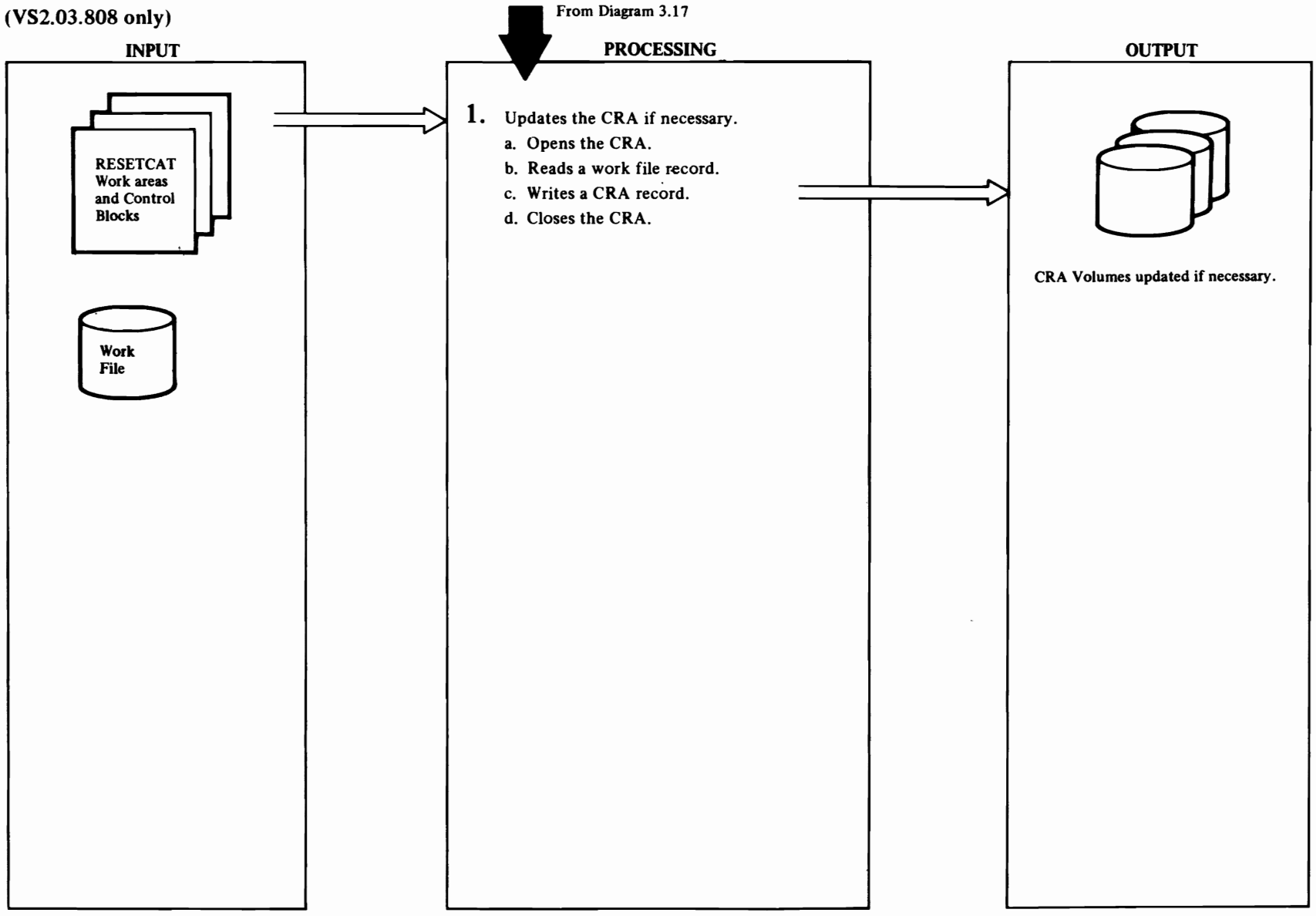
Procedures: UPDCAT, RECMGMT, UPDCCR

7. The GETRCD routine of RECMGMT is called to read the CCR (control interval number 3). The following items in the CCR are updated by UPDCCR:

   - First unformatted free record
   - Count of deleted free records
   - Control interval number of first deleted free record
   - High RBA maintained in the CCR

   After the above items are changed, RECMGMT (with UPDRCD option) is called to write the updated CCR back to the catalog.

Diagram 3.17.7 RESETCAT FSR - Update the CRA (VS2.03.808 only)



Extended Description for Diagram 3.17.7 (VS2.03.808 only)

Module: IDCRS01, IDCRS05, IDCRS06

Procedures: UPDCRA, SCNRLST, RECMGMT, CKERR

1.a. The SCNRLST routine is called to obtain a CRA volume serial number entry from the RESVOL table. A check is made to see if this CRA needs to be updated by checking if the CRA update chain is empty. If the chain is not empty, DSOPEN is called to open and verify the CRA. If the open is successful, the "CRA open" flag is set; if not, the "termination" flag is set.

b. Each record in the CRA update chain is read from the work file RECMGMT (GETRCD). The control interval number of the next record in the chain is saved. If the record just read happens to be a free record, the CRA CCR record needs to be updated. If the CCR has not been read already, RECMGMT (GETRCD) is called to read it. The deleted free record count in the CCR is incremented, and the record is placed on the CRA free chain.

c. The record read from the work file is moved to the CRA buffer. Control interval information is inserted and RECMGMT (UPDRCD) is called to write an updated record in the CRA.

After all records in the CRA update chain have been processed for a specific CRA, RECMGMT (UPDRCD) is called to write the updated CCR record back to the CRA.

d. DSCLOSE is called to close the CRA. If the close fails, the "termination" flag is checked. If it is set, CKERR is called to print an error message and terminate RESETCAT processing. If the termination flag is not set, control returns to the caller.

Termination Visual Table of Contents

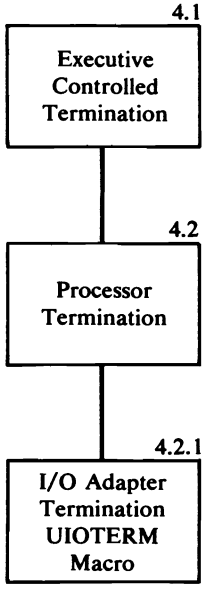
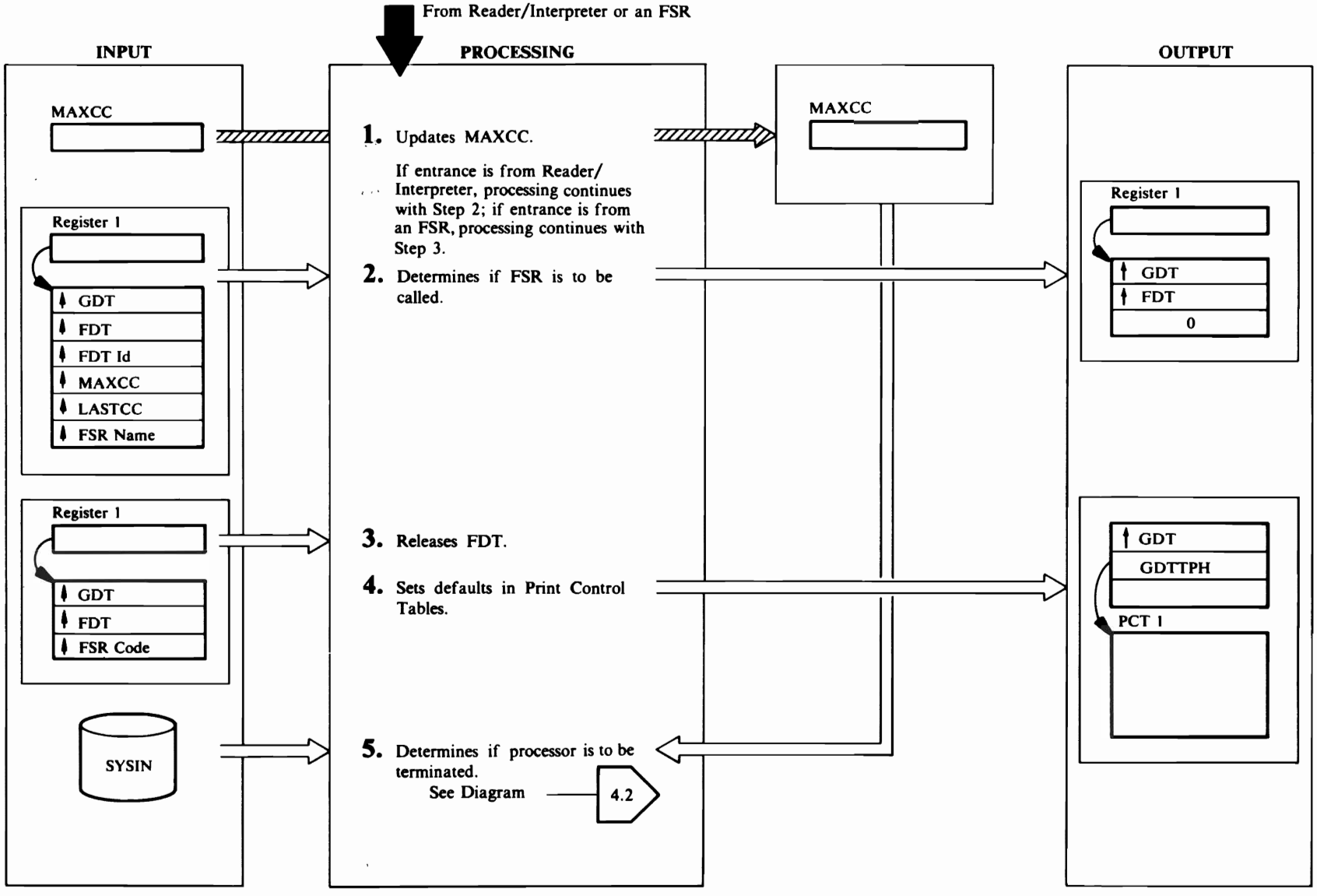


Diagram 4.1 Executive Controlled Termination



Extended Description for Diagram 4.1

Module: IDCEX01

Procedure: MAIN

1. IDCEX01 compares the LASTCC code returned by the FSR or the Reader/Interpreter with MAXCC and puts the greater number in MAXCC. If control is from the Reader/Interpreter, MAXCC has already been properly set. If entrance is from the Reader/Interpreter, processing continues with step 2; if entrance is from an FSR, processing continues with step 3.

Module: IDCEX01

Procedure: MAIN

2. If MAXCC is less than 16, IDCEX01 gives control to an FSR. The Reader/Interpreter passes the FSR name to IDCEX01. If MAXCC is greater than or equal to 16, processing continues with step 5.

Module: IDCEX01

Procedure: CALLFSR

3. IDCEX01 releases storage for the FDT using a UFPOOL macro. The pool identification is EX00, and the FDT is the only data in the pool.

Module: IDCEX01

Procedure: CALLFSR

4. IDCEX01 sets the Print Control Table to Access Method Services default values by issuing a URESET macro instruction.

Module: IDCEX01

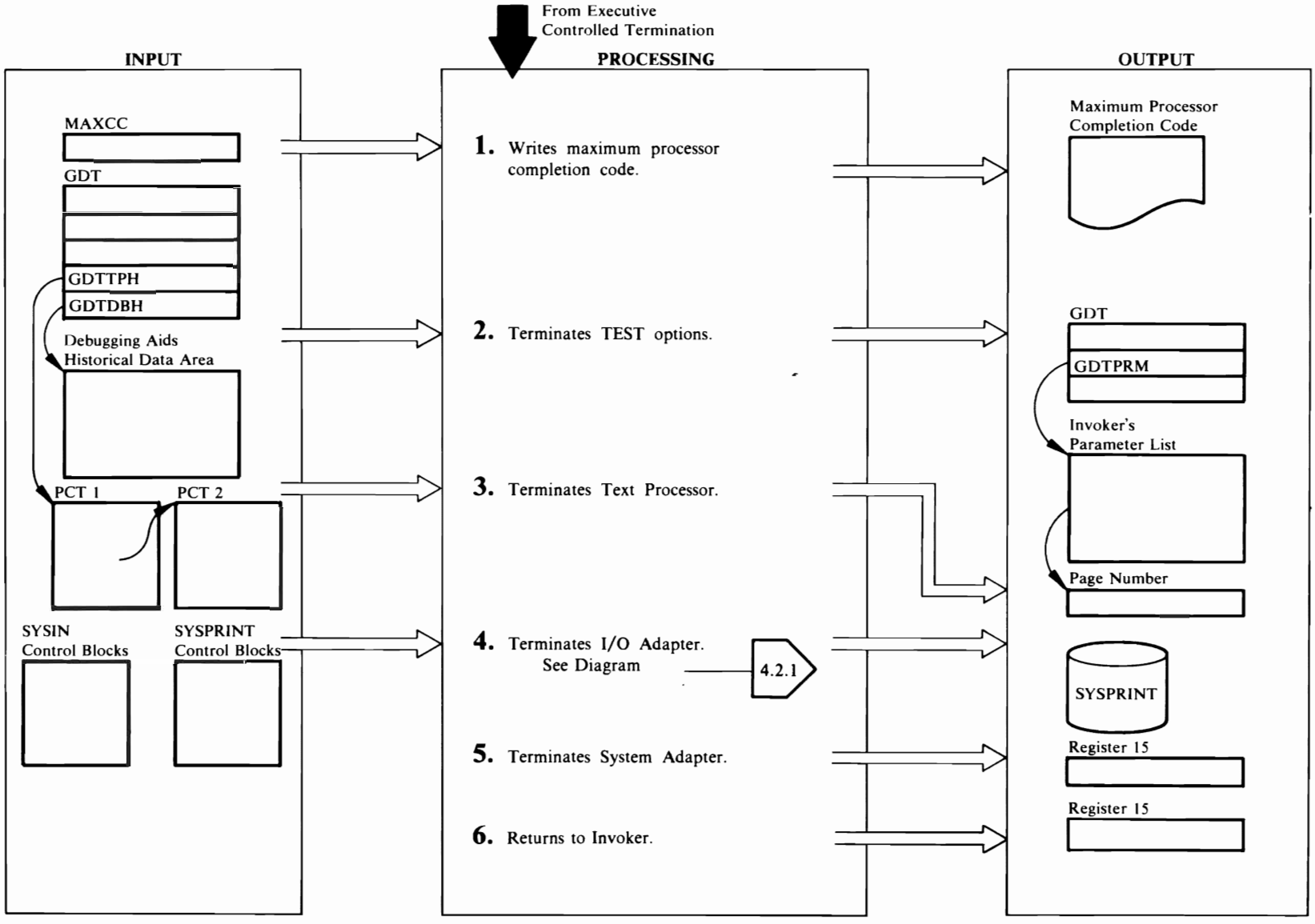
Procedure: MAIN

5. The processor has terminated if one of the following conditions is met:

   - The Reader/Interpreter has detected end-of-file on SYSIN. In this case, the Reader/Interpreter puts a non-zero value in register 15.
   - An error has occurred so that processing cannot continue, and MAXCC contains a value greater than or equal to 16.

   If one of these conditions is met, control is given to Processor Termination, Diagram 4.2. If neither of the two conditions is met, control is given to the Reader/Interpreter, Diagram 2.0, to obtain the next command.

Diagram 4.2 Processor Termination



Extended Description for Diagram 4.2

Module: IDCEX03

Procedure: IDCEX03

1. IDCEX03 prints a message of the maximum processor condition code, MAXCC by using a UPRINT macro.

Module: IDCEX03

Procedure: IDCEX03

2. If TEST options were specified on a PARM command or on the EXEC statement that invoked Access Method Services, IDCPM01 has loaded the Debug Module, IDCDB01. IDCEX03 sets GDTDBG, the address of the Debug Module, to zero after deleting the Debug Module by issuing the UDELETE macro. The address of the Debugging-Aids Historical Data Area is in GDTDBH. IDCEX03 frees the Debugging-Aids Historical Data Area used by the UDUMP macro. It also sets GDTDBH to zero after the area is freed.

Module: IDCEX03

Procedure: IDCEX03, SCANPARM

3. IDCEX03 terminates the Test Processor by issuing a URESET macro. If the invoker of Access Method Services wants the last page number returned, IDCEX03 passes the address of the invoker's page number field to the URESET macro.

Module: IDCEX03

Procedure: IDCEX03

4. IDCEX03 terminates the I/O Adapter by issuing a UIOTERM macro. Diagram 4.2.1 shows I/O Adapter termination in detail.

Module: IDCSA01

Procedure: IDCSA01

5. (For **VS2.03.807**, IDCSA01 searches the Load List Block (LLBLK) and issues a DELETE macro for each module identified in the LLBLK.)

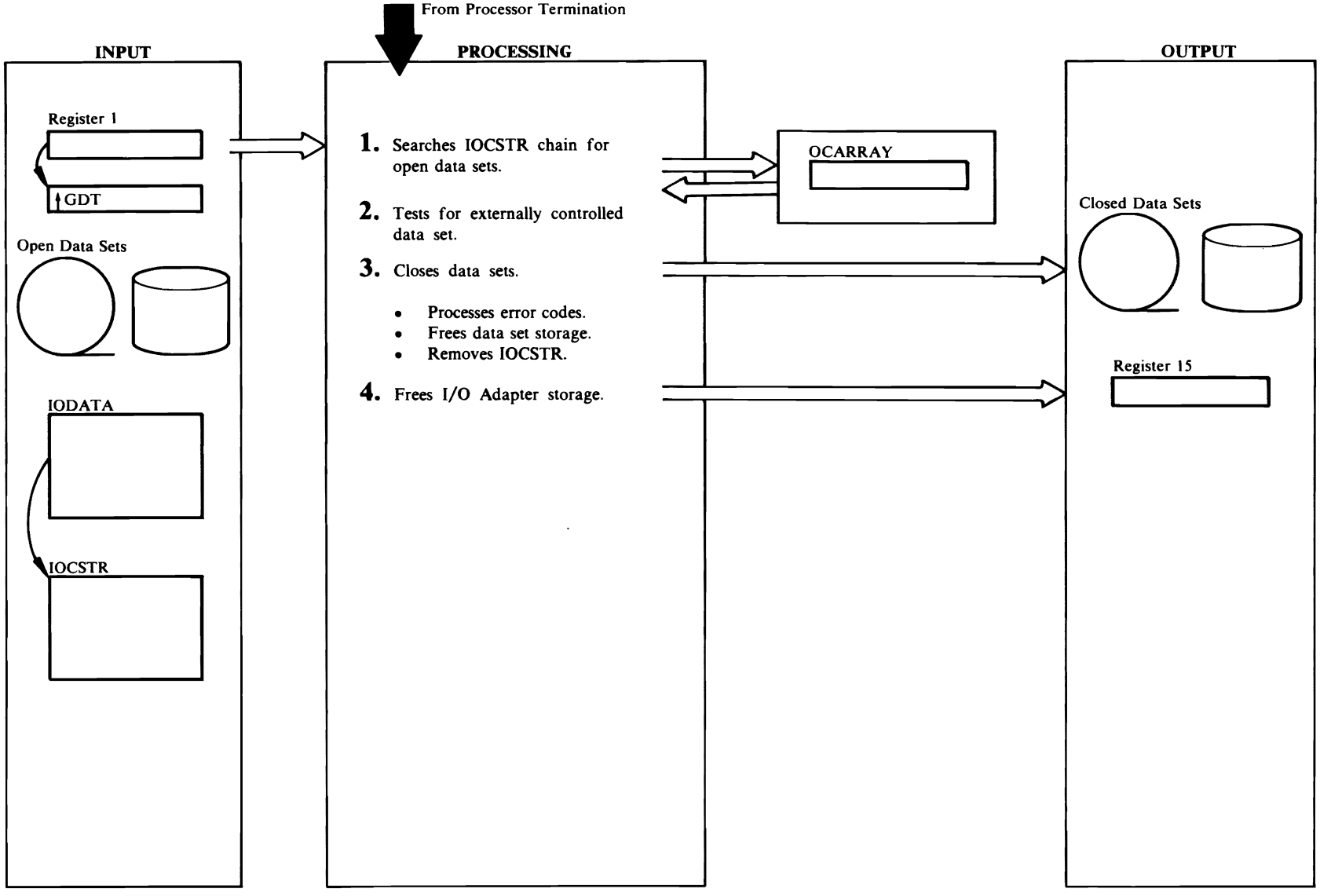
   IDCSA02, IDCSA03, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10, IDCTP01, IDCIO01, and IDCIO05. The Storage Table, AUTOTBL, contains the storage addresses for all of these modules except IDCSA03. The GDT contains the storage address for IDCSA03. IDCSA01 also frees the Inter-Module Trace Table, the Intra-Module Trace Table, the System Adapter Historical Data Area, and the GDT. When the System Adapter receives control, register 15 contains MAXCC. IDCEX01 copied MAXCC into register 15 for the Access Method Services invoker.

Module: IDCSA01, IDCAM01, IDCAM02

Procedure: IDCSA01, IDCAM01, IDCAM02

6. The System Adapter returns to the Access Method Services invoker with a return code in register 15.
   - If Access Method Services is invoked with a batched job, control returns directly to the invoker. If Access Method Services is invoked with JCL, control returns to the operating system. If Access Method Services is invoked with a problem program, control returns to the problem program.
   - If Access Method Services is invoked interactively with TSO, control returns to IDCAM01 or IDCAM02. IDCAM01 and IDCAM02 contain identical code. IDCAM01 or IDCAM02 tests the return code in register 15. If the return code is greater than 8, IDCAM01 or IDCAM02 issues a STACK and a TCLEARQ macro to delete any commands that are waiting to be processed. IDCAM01 or IDCAM02 returns control to TSO Terminal Monitor Program.

Diagram 4.2.1 I/O Adapter Termination – UIOTERM Macro



Extended Description for Diagram 4.2.1

Module: IDCIO01

Procedure: IDCIO01

1. IDCIO01 sets up a loop to close all open data sets, and sets the "close all" option in OCARRAY that permits SYSIN and SYSPRINT to be closed.

Module: IDCIO02

Procedure: CLOSERTN

2. CLOSERTN examines the IOCSTR chain for the address of IOCSTRs to close. For a non-VSAM data set, CLOSERTN sets the address of a SYNAD routine in the DCB to zero and puts the address of a CLOSE exit routine in the DCB. If the data set is not open, IOCFLGOP = 1, CLOSERTN determines if it is user controlled. If so, CLOSERTN passes arguments to the user routine. If the data set is SYSPRINT on a TSO terminal, IDCIO04 does not close the data set, but gives a zero return code because the data set doesn't need to be closed. This check is made for up to the first four IOCSTRs in the IOCSTR chain. Normally, only the SYSIN and SYSPRINT IOCSTRs are in the chain at termination.

Module: IDCIO02, IDCIO04

Procedure: CLOSERTN

3. CLOSERTN issues a CLOSE macro with the address of up to four DCBs or ACBs. If an ABEND occurs during the closing of a non-VSAM data set, the operating system close routine gives control to a CLOSE exit routine that will cause the I/O Adapter to abend. The UABORT macro is issued after control returns from the CLOSE macro. Closing continues with the next data set. The following steps are performed for each data set:

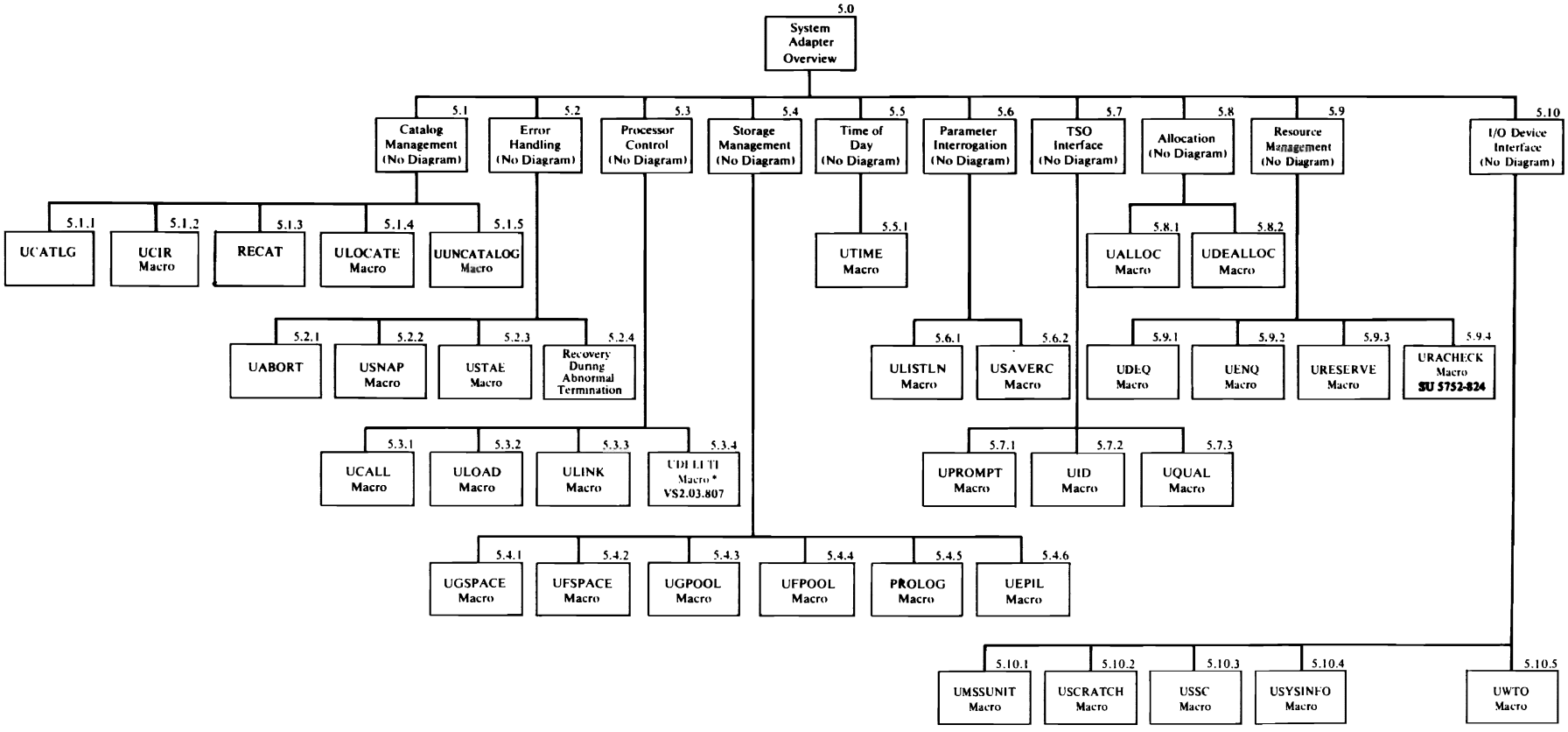
   - For VSAM data sets, CLOSERTN issues a SHOWCB macro to return the ACB error code. If the ACB error code is not zero, BLDOCMSG writes a message. However, since SYSPRINT is the first data set closed, BLDOCMSG issues a UABORT macro. No test is made for non-VSAM data sets.
   - For VSAM data sets, CLOSERTN checks the IOCSEX to see if there are any VSAM control blocks to free. When any length of the ACB, RPL, or EXLST is non-zero, ENVFREE issues a FREEMAIN macro to release the control block. For open non-VSAM data sets, ENVFREE issues a FREEMAIN to free any buffers obtained by the operating system open routines. If the data set has been dynamically allocated, ENVFREE builds an ALLAGL with a disposition of KEEP. ENVFREE then issues a UDEALLOC macro to deallocate the data set.
   - CLOSERTN saves the address of the closed data sets' IOCSTRs and the address of the next IOCSTR in the chain. CLOSERTN issues a UFPOOL macro to free storage obtained for the closed data set. CLOSERTN searches the IOCSTR chain until the IOCSTR that points to the IOCSTR of the closed data set is found. CLOSERTN replaces the address of the closed data set's IOCSTR with the address of the next IOCSTR in the chain.

Module: IDCIO01

Procedure: IDCIOCL

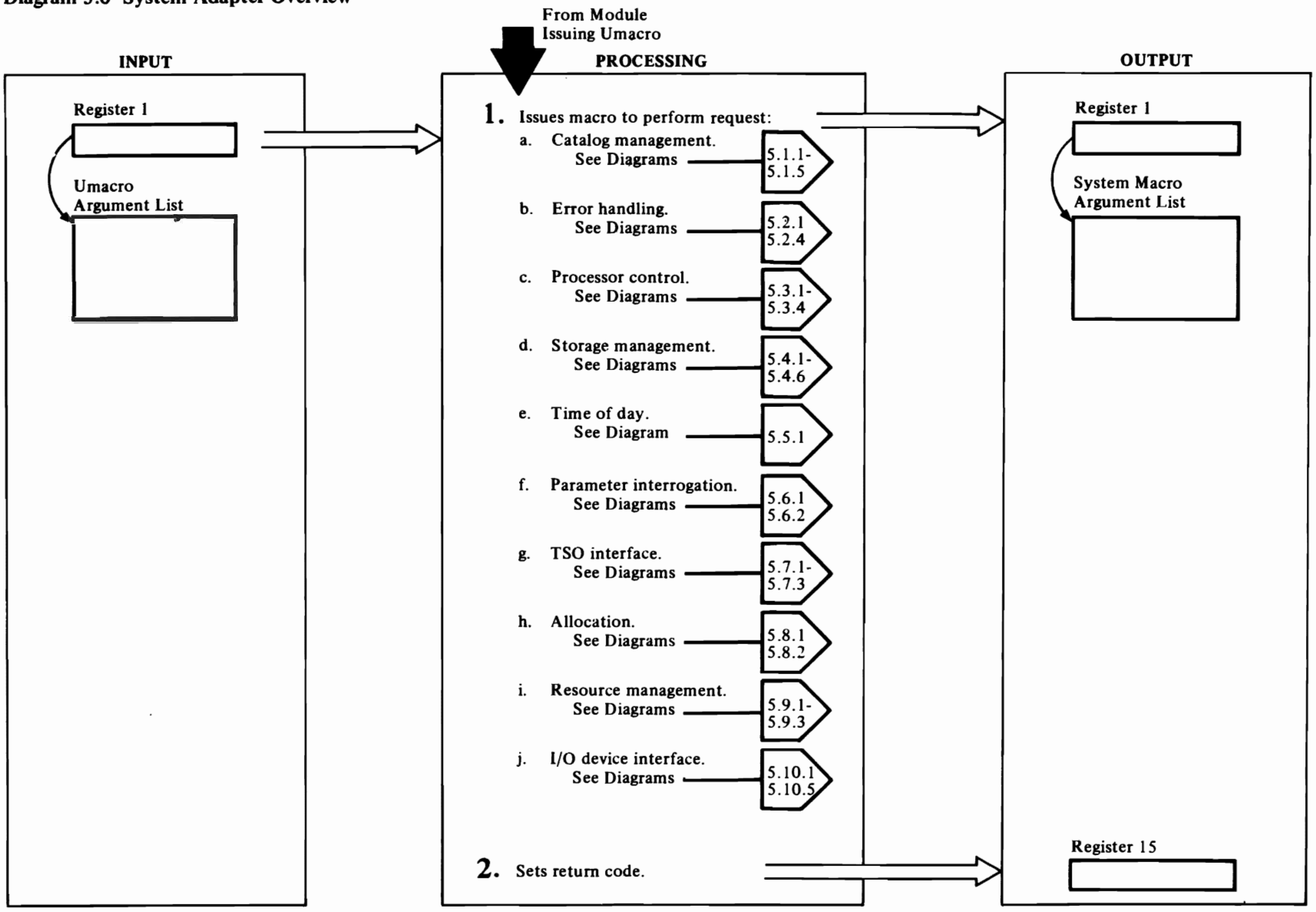
4. Processing returns to step 1 until all data sets have been closed. When all data sets are closed, the IOCSTR chain no longer exists. CLOSERTN issues a UFPOOL macro to free storage obtained by the I/O Adapter. The only storage remaining to be freed is IODATA and the message area for VSAM data sets. IDCIOCL puts a return code in register 15. Control then returns to the module that issued the UIOTERM macro.

System Adapter Visual Table of Contents



*No Diagram if **VS2.03.807** is not installed.

Diagram 5.0 System Adapter Overview



Extended Description for Diagram 5.0

Module: IDCSA01, IDCSA02, IDCSA03, IDCSA05, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10

Procedure: IDCSA01, IDCSA02, IDCSA03, IDCSA05, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10

1. The System Adapter and the I/O Adapter insulate the rest of the processor from the operating system. Whenever the processor wants a service that requires an operating system dependent macro, like GETMAIN or GETVIS, the processor calls the System Adapter with a U macro. Different versions of the System Adapter and I/O Adapter supply code for different operating systems. Except for the System Adapter and the I/O Adapter, the Access Method Services modules are oblivious to the operating system. The System Adapter and the I/O Adapter are tuned for VS or DOS before they are shipped to the customer—that is, before SYSGEN time. The processor listings are either for VS or for DOS, but not both at once. System macros in the listings indicate the operating system the listing represents.

   Types of services provided by the System Adapter:

   a. Whenever information is to be added or deleted from the VSAM catalog, a UCATLG macro is issued. The VSAM CATLG macro cannot be used because, although the VSAM CATLG macro has the same parameters in VS and DOS, the general code is different. The VSAM CATLG macro must be in a program that is assembled under the correct operating system. Whenever information is to be retrieved from a VSAM catalog, a UCATLG or a UCIR macro is issued. The UCATLG macro retrieves information about a particular VSAM catalog entry. The UCIR macro retrieves data set names from a VSAM catalog that match data set name qualifiers. Whenever a data set is to be recataloged in an OS/VS CVOL or a VSAM catalog, a URECAT macro is issued. The ULOCATE macro is issued to retrieve information about a nonVSAM data set from a VSAM catalog or an OS/VS CVOL. Whenever a data set is to be uncataloged and scratched from a VSAM catalog or OS/VS CVOL, a UUNCATLG macro is issued. Diagrams 5.1.1, 5.1.2, 5.1.3, 5.1.4, and 5.1.5 show the UCATLG, UCIR, URECAT, ULOCATE, and UUNCATLG macros in detail.

   b. Error handling is accomplished with UABORT, USNAP, and USTAE. For errors, when processing cannot continue, a UABORT is issued to print a dump and return control to the operating system. For debugging information, a USNAP is issued to print the region and return control to the Access Method Services module that issued the USNAP. Diagrams 5.2.1 and 5.2.2 show the UABORT and USNAP macros in detail.

      A USTAE is issued to establish or cancel an ESTAE environment. Diagram 5.2.3 shows the USTAE macro in detail. Diagram 5.2.4 shows recovery processing during abnormal termination.

   c. Inter-processor module control is accomplished with UCALL and ULOAD. UCALL loads a module and gives control to it. It is used to transfer control from one module to another within Access Method Services. ULOAD just loads a module. It is mainly used for non-executable modules like static text structures.

      Without VS2.03.807: UDELETE was used to delete modules originally loaded by ULOAD. Now UDELETE does not delete modules; the operating system deletes modules when necessary. When a UDELETE macro is uissed, the system adapter just returns to the module issuing the UDELETE. There is no diagram for UDELETE. Control between the processor and a module not a part of the processor is accomplished with a ULINK macro. ULINK loads the module and gives control to the module at a specific entry point. Diagrams 5.3.1 through 5.3.3 show the UCALL, ULOAD and ULINK macros in detail.

      With VS2.03.807: Control between the processor and a module not a part of the processor is accomplished with a ULINK macro. ULINK loads the module and gives control to the module at a specific entry point. A list of modules loaded by ULOAD, UCALL, and ULINK will be maintained in a table called the Load List Block (LLBLK). Each request for a module via ULOAD, UCALL, and ULINK causes a use count for the module to be increased by one. When a UDELETE macro is issued, the use count is decremented by one. See Storage Management below for information of how the macros UGPOOL. UGSPACE, and PROLOG utilize the use count in LLBLK when acquiring virtual storage. Diagram 5.3.1 through 5.3.4 show the UCALL, ULOAD, ULINK and UDELETE macros in detail.

Module: IDCSA01

Procedure: IDCSA01

   d. Storage management is performed with three types macros:

      1. UGSPACE and UFSPACE, shown in Diagrams 5.4.1 and 5.4.2
      2. UGPOOL and UFPOOL, shown in Diagrams 5.4.3 and 5.4.4
      3. PROLOG and UEPIL, shown in Diagrams 5.4.5 and 5.4.6

      The first type is used to obtain large amounts of storage. The caller must remember the address of the storage, and must issue a UFSPACE to release the storage.

      The second type is used for small amounts of storage. The caller does not need to remember the address of each piece because all the pieces can be released with one UFPOOL at the end of the program.

      The third type is used to bypass PL/S-generated GETMAIN and FREEMAIN macros. In a reentrant environment, PL/S generates a GETMAIN macro for all data areas defined in the program, but a GETMAIN doesn't work on DOS. Each Access Method Services routine includes code at the beginning of the routine to replace the GETMAIN. This is the PROLOG code. Control is transferred to the System Adapter that issues the appropriate operating system macro to obtain storage. Instead of issuing a PL/S return statement that uses FREEMAIN, all routines issue a UEPIL macro. The UEPIL macro gives control to the System Adapter. The System Adapter frees storage and gives control to the routine that called the routine issuing the UEPIL. The PL/S-generated code to free storage and to return control is never executed. If **VS2.03.807** is installed, the return code from GETMAIN is checked. If the return code from GETMAIN indicates virtual storage is not available, the Load List Block (LLBLK) is searched and all modules with a use count of zero are deleted. The GETMAIN is then retried.

   e. The time of day is obtained with a UTIME macro, shown in Diagram 5.5.1. Several data formats for the time and date are allowed.

f. Parameter interrogation is performed by the ULISTLN and the USAVERC macros, shown in Diagrams 5.6.1 and 5.6.2.

g. The processor communicates with the TSO terminal user with the UPROMPT macro. The UPROMPT macro gives a message to the TSO terminal user and receives a reply. The UID macro gets the TSO terminal user's identification from the User Profile Table. The UQUAL macro completes a data set name—that is, it fully qualifies the name—by linking to a TSO module. Diagrams 5.7.1 through 5.7.3 show the UPROMPT, UID and UQUAL macros in detail.

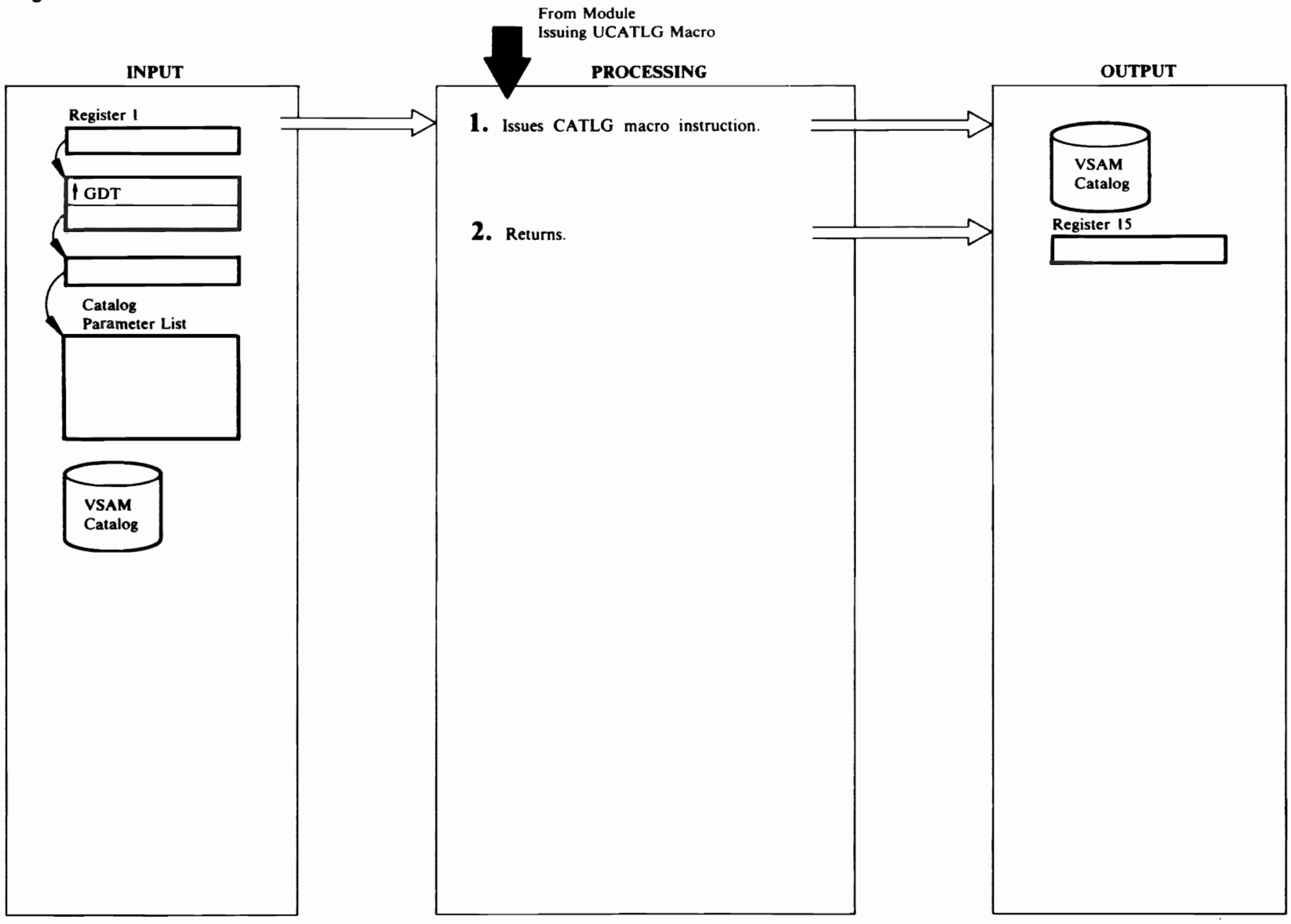
h. Data set allocation and volume mounting are performed by the UALLOC and UDEALLOC macros. UALLOC dynamically allocates a data set or dynamically mounts a volume. Dynamic allocation or mounting means allocating without having a DD statement. UDEALLOC deallocates a data set or dismounts a volume that was allocated with the UALLOC macro. Diagrams 5.8.1 and 5.8.2 show the UALLOC and UDEALLOC macros in detail.

i. Resource management is performed by the UENQ, UDEQ, and URESERVE macros. The UENQ MACRO acquires control of a resource. The URESERVE macro acquires control of an I/O unit. The UDEQ macro releases control of a resource or I/O unit acquired by UENQ or URESERVE. Diagrams 5.9.1 through 5.9.3 show the UDEQ, UENQ, and URESERVE macros in detail.

j. I/O device interface is performed by the USCRATCH, USYSINFO, UWTO, UMSSUNIT, and USSC macros. The USCRATCH macro deletes a data set stored on one or more direct-access volumes. The USYSINFO macro returns information from the system control blocks, such as job and step names or specific information about an I/O unit. The UWTO macro is used to write messages to the console operator. The UMSSUNIT macro mounts or demounts a mass storage volume. The USSC macro performs Mass Storage Control functions. Diagrams 5.10.1 through f.10.5 show the UMSSUNIT, USCRATCH, USSC, USYSINFO, and UWTO macros in detail.

2. At the end of most Umacros, a return code is put in register 15, and control returns to the module that issued the Umacro. The exceptions are UABORT, UCALL, UEPIL, and ULINK.

Diagram 5.1.1 UCATLG Macro



Extended Description for Diagram 5.1.1

Module: IDCSA02

Procedure: IDCSA02

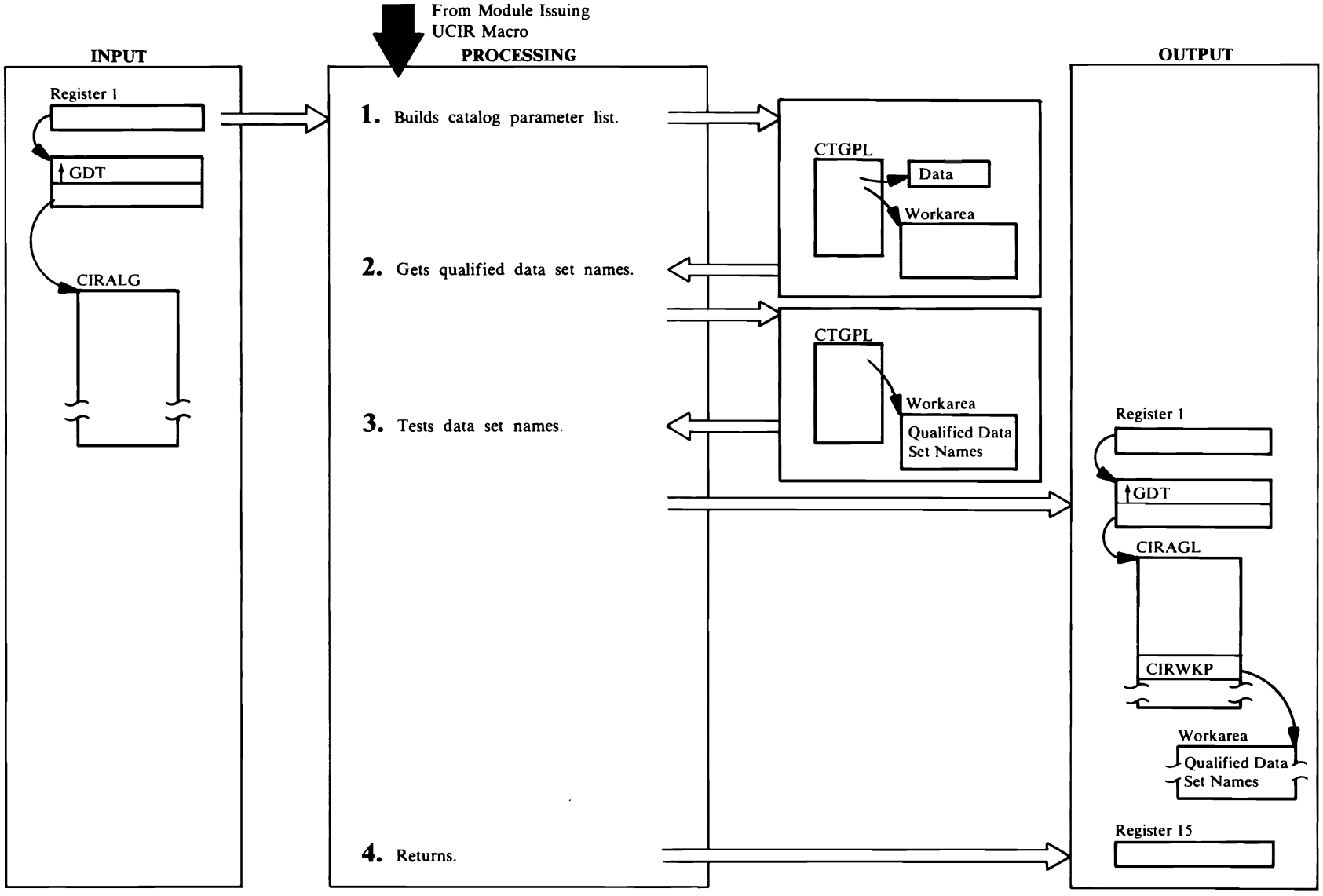
1. IDCSA02 passes the catalog parameter list to VSAM with a CATLG macro.

Module: IDCSA02

Procedure: IDCSA02

2. IDCSA02 puts the return code from VSAM in register 15, and returns control to the module that issued the UCATLG macro.

Diagram 5.1.2 UCIR Macro



Extended Description for Diagram 5.1.2

Module: IDCSA02

Procedure: IDCSA02

1. UCIR builds a CTGPL and sets flags indicating the data set name is supplied, a super locate request, a generic locate request, VSAM data set, and return catalog names. UCIR moves the length of the header qualifier and the header qualifier to a work area. UCIR puts the address of the work area in the CTGPL. If the module that issued the UCIR supplies a catalog name or password in the CIRAGL, UCIR puts the address of the catalog name or password in the CTGPL. The VSAM catalog routines need a work area for the names returned from the catalog. UCIR gets the work area by issuing a UGPOOL macro with the UGPOOL identification in CIRAGL. UCIR initially gets a work area large enough for 180 returned names. After the UGPOOL macro, UCIR puts the address of the work area in the CTGPL.

Module: IDCSA02

Procedure: IDCSA02

2. UCIR issues a UCATLG macro to get the qualified data set names. If the return code is zero, control continues with step 3. If the return code is non-zero and indicates something other than the work area is too small, control continues with step 4. If the return code indicates that the work area is too small, UCIR issues a UFPOOL to free the current work area. Then, UCIR issues a UGPOOL for a work area either twice the size of the last work area or the size VSAM catalog management returns. If UGPOOL does get the storage requested, UCIR puts the address in the CTGPL, and control returns to the beginning of step 2. If UGPOOL cannot get storage, control goes to step 4.

Module: IDCSA02

Procedure: IDCSA02

3. The address of the work area containing the qualified data set names is put in CIRWKP in CIRAGL. The number of data set names is put in the first word of the work area. UCIR performs tests on each qualified data set name to determine if the data set name matches the criteria in CIRAGL. Suppose the VSAM catalog contains the following names:

   1. D26U.A.B.C
   2. D27V.A.B.C
   3. D27V.AMS.AMSIO01.MAC.PGM
   4. D27V.AMS.AMSIO01.MOD.PGM
   5. D27V.AMS.AMSIO02.MOD.PGM
   6. D27V.AMS.DEV
   7. D27V.AMS.MAC
   8. D27V.TEST.DEV

   If CIRAGL specifies the "one" qualifier:

   Case A: If no trailing qualifiers and the leading qualifier is D27V.AMS., then names 6 and 7 are returned by UCIR.

   Case B: If the trailing qualifier is .DEV and the leading qualifier is D27V., then names 6 and 8 are returned by UCIR.

   If CIRAGL specifies the "all" qualifier:

   Case A: If no trailing qualifier and the leading qualifier is D27V.AMS., then names 3,4,5,6, and 7 are returned by UCIR.

   Case B: If the trailing qualifier is .MOD and the leading qualifier is D27V.AMS., then names 4 and 5 are returned by UCIR.

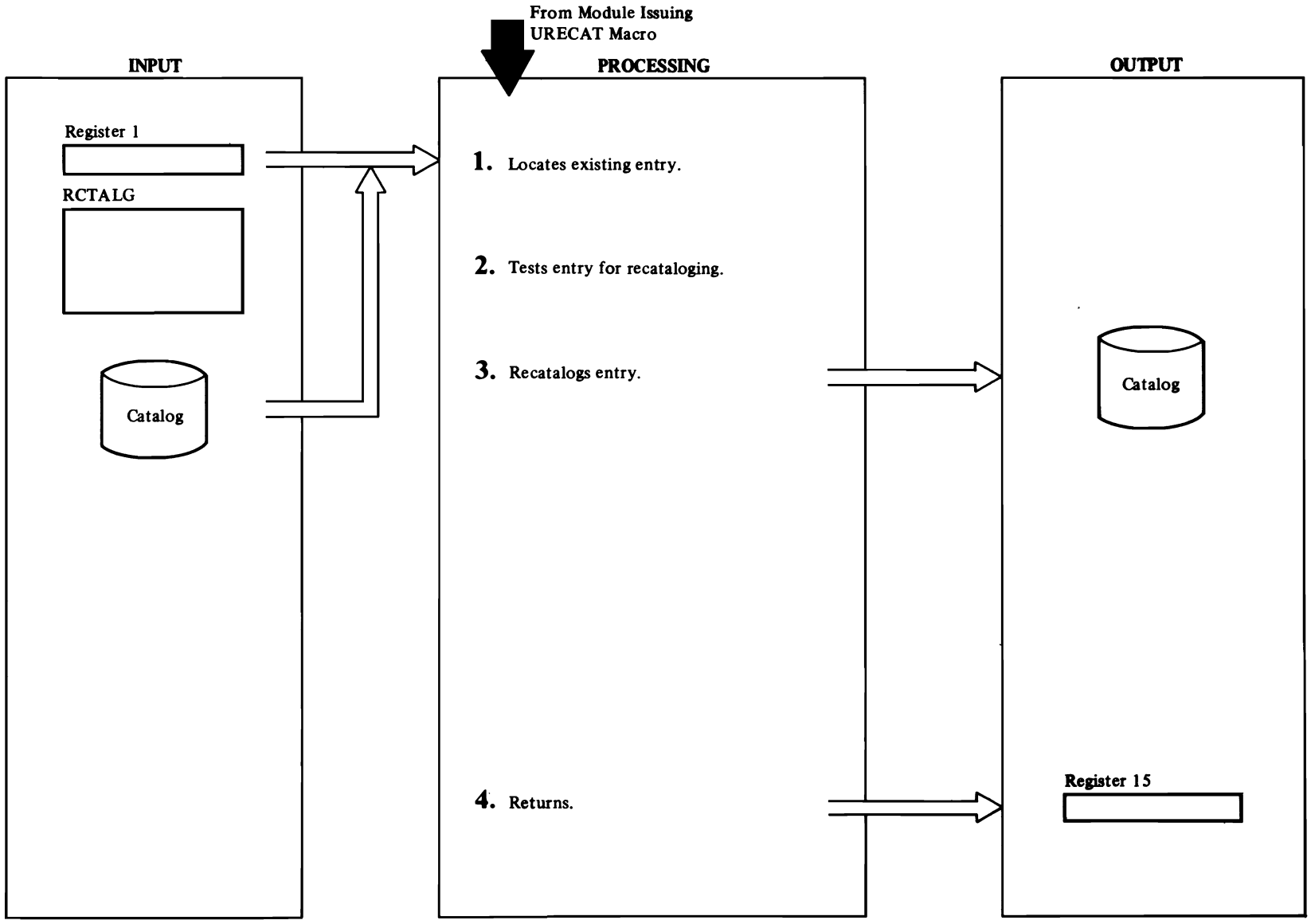
   Only the names that match the criteria are returned in the work area addressed in CIRWKP. Catalog names are always returned because the catalog name can be used to tell which catalog a data set name comes from.

Module: IDCSA02

Procedure: IDCSA02

4. UCIR puts a return code in register 15 and returns control to the module that issued the UCIR macro. The module that issued the UCIR must free the work area addressed by CIRWKP.

Diagram 5.1.3 System Adapter – URECAT Macro



Extended Description for Diagram 5.1.3

Module: IDCSA07

Procedure: IDCSARC GETENT

1. IDCSARC issues a LOCATE macro to locate the data set's entry in an OS/VS or VSAM catalog. If no entry is found with the specified name, no recataloging is performed.

Module: IDCSA07

Procedure: TESTENT

2. TESTENT determines whether the data set can and needs to be recataloged. If the data set resides on 20 or fewer volumes, TESTENT searches the catalog workarea returned by the LOCATE macro for a match of the current device and volume serial number. If no match is found, the data set has already been recataloged or the entry is for a different data set, so no recataloging is performed.

   If the data set resides on more than 20 volumes, TESTENT searches the catalog workarea for a match of the new device and volume serial number. If a match is found, the data set has already been recataloged. If no match is found, the data set needs to be recataloged— TESTENT issues an informational message.

Module: IDCSA07

Procedure: UPDATENT

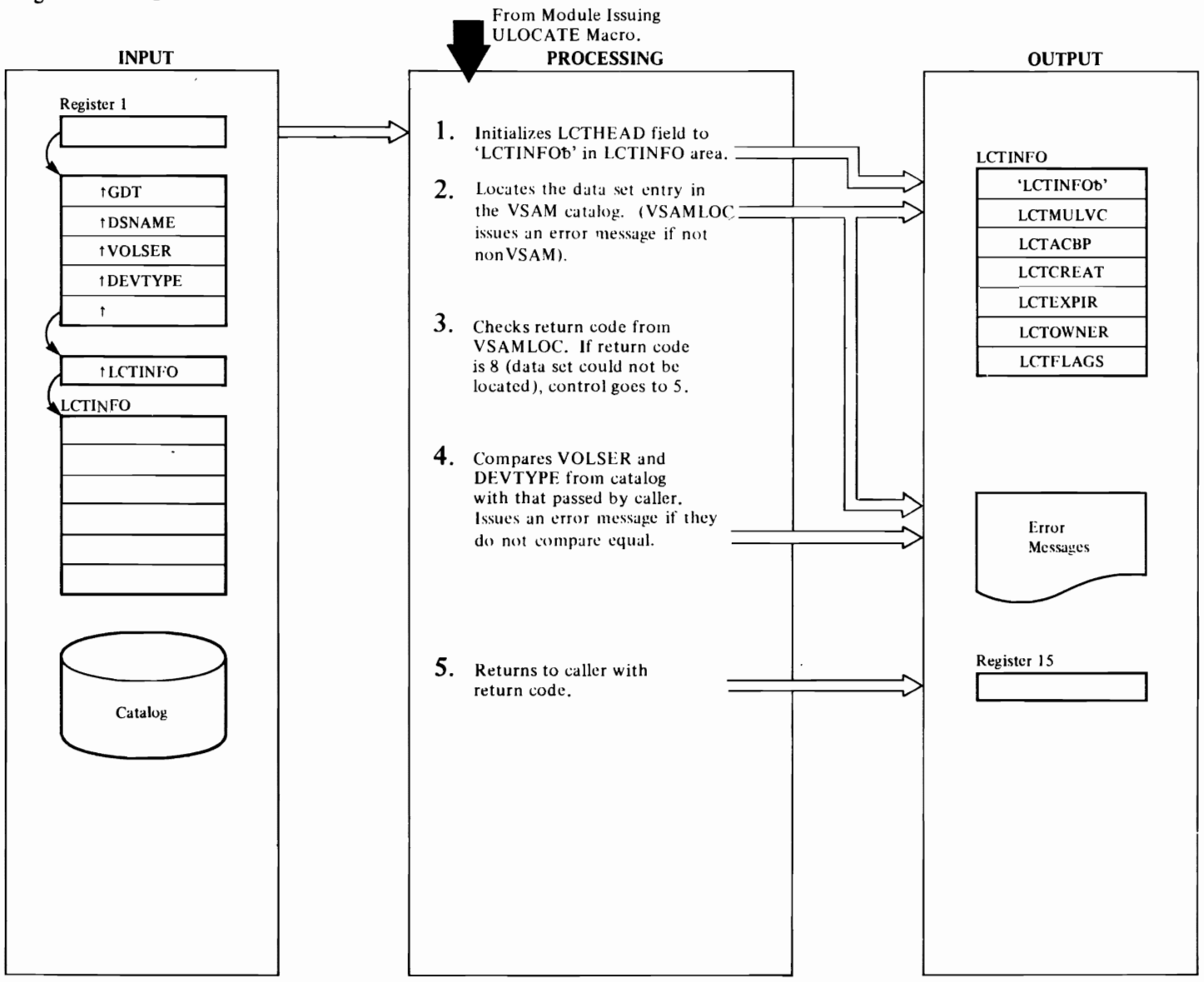
3. UPDATENT updates the device type and/or volume serial number in the catalog workarea. UPDATENT issues a CATALOG macro to recatalog the data set in an OS/VS or VSAM catalog.

Module: IDCSA07

Procedure: IDCSARC

4. If an error occurred, a message is written with a UPRINT IDCSARC returns to the caller with a return code in register 15.

Diagram 5.1.4 U LOCATE M



Extended Description for Diagram 5.1.4

Module: IDCSA07

Procedure: IDCSALC .

1. The pointer to the parameter list is passed in register 1 at entry to IDCSALC. The fifth word of the parameter list points to the address of the LCTINFO area address. The LCTHEAD field is initialized to 'LCTINFO'.

Module: IDCSA07

Procedure: VSAMLOC

2. The VSAMLOC procedure is called to issue a UCATLG macro to locate the data set entry in the VSAM catalog. OS/VS CVOL will also be searched by this VSAM locate request if the data set is not found in the VSAM catalog. VSAMLOC processes return codes as follows:

   0 — Successfully located the data set entry in the catalog. If ENTYPE is not a nonVSAM entry, the VSAMLOC sets a return code of 4 and prints an error message that the data set name was not nonVSAM.
   8 — Data set entry was not located. VSAMLOC returns a return code of 8 to IDCSALC.
   A return code other than 0 or 8 causes VSAMLOC to print an error message via UERROR. VSAMLOC places the following information retrieved from the catalog into the LCTINFO area:

   - Number of volumes
   - Creation date
   - Expiration date
   - Owner name
   - Pointer to catalog ACB
   - LCTVSCAT flag is set

Module: IDCSA07

Procedure: IDCSALC

3. The return code from VSAMLOC is checked. If the code is 8 (data set could not be located), control goes to step 5, otherwise control goes to step 4.

Module: IDCSA07

Procedure: IDCSALC

4. If the data set entry is located, the volume serial number and the device type returned from the catalog are compared with the information passed by the caller. If they do not compare equal, an error message is issued and a return code is set indicating volser or devtype did not compare.

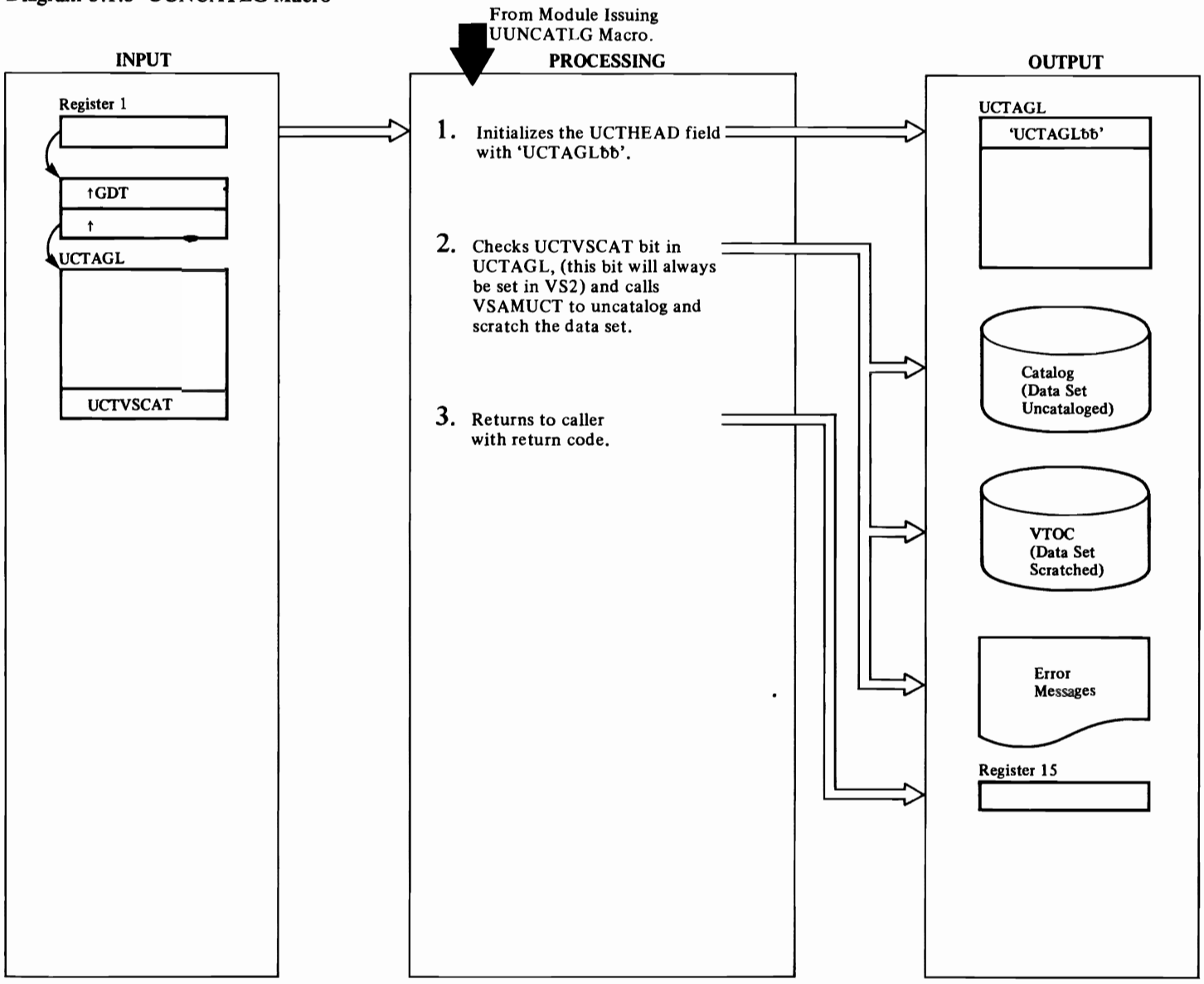
Module: IDCSA07

Procedure: IDCSALC

5. IDCSALC returns control to the caller with the appropriate return code in register 15:

   0 = Successful
   4 = Error on locate request. Either the volume serial number or device type of the caller did not match those in the catalog, or this was not a nonVSAM data set.
   8 = Data set could not be located.

Diagram 5.1.5 UUNCATLG Macro



Extended Description for Diagram 5.1.5

Module: IDCSA07

Procedure: IDCSAUC

1. The UCTHEAD field in the UCTAGL is initialized to 'UCTAGLbb'.

Module: IDCSA07

Procedure: VSAMUCT

2. The UCTVSCAT bit in the UCTAGL is checked to see if a VSAM Delete can be used to uncatalog and scratch the data set. Since this bit will always be set, the VSAMUCT procedure is called. VSAMUCT initializes the VSAM Catalog parameter list, IEZCTGPL, and issues the UCATLG macro to uncatalog and scratch the data set. If the return code from UCATLG is not 0 (successful) or 8 (data set not found), VSAMUCT issues the UERROR macro identifying the catalog error that occurred. Additionally, it issues UPRINT to print error messages to indicate that the data set was not scratched and/or uncataloged. A return code of 0 or 8 causes VSAMUCT to next issue the USCRATCH macro to scratch the data set from the VTOC. **Note:** The USCRATCH macro is issued even though the data set is not found.

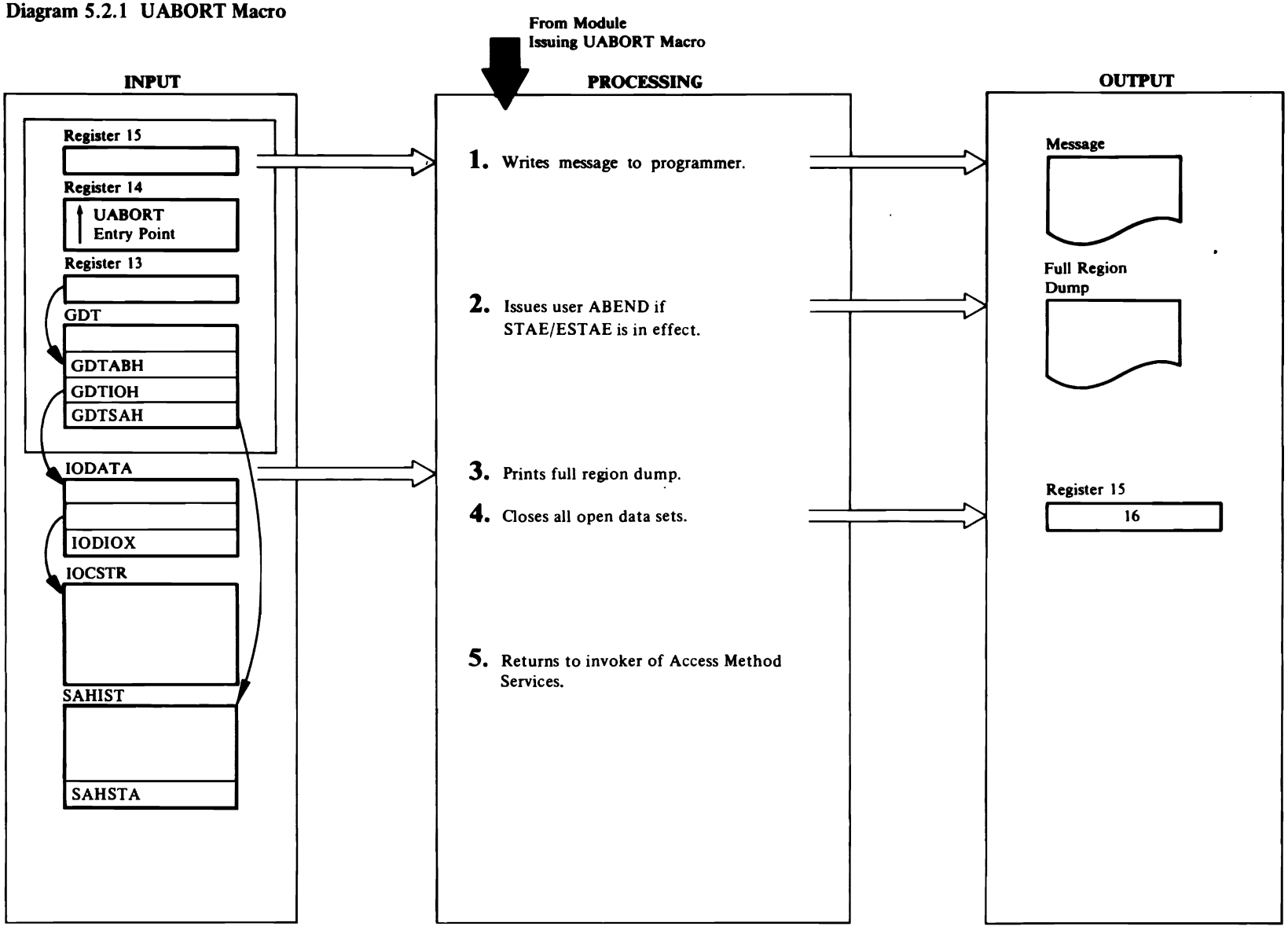
Module: IDCSA07

Procedure: IDCSAUC

3. IDCSAUC returns control to the caller with a return code in register 15, as follows:

   0 = No errors occurred.
   4 = An error occurred in UCATLG or USCRATCH.

Diagram 5.2.1 UABORT Macro



Extended Description for Diagram 5.2.1

Module: IDCSA01

Procedure: IDCSA01

1. The UABORT routine uses the registers saved in the save area pointed to by GDTABH to establish addressability. This is done so the UABORT routine can access storage areas obtained by the System Adapter and remain reentrant. UABORT issues a WTO macro with a routing code of 11 to issue a message to the programmer. If Access Method Services is invoked interactively with TSO, UABORT issues a PUTLINE macro to write the UABORT message on the terminal. The message contains the code given to UABORT in register 15.

Module: IDCSA01

Procedure: IDCSA01

2. IDCSA01 checks the address of the ESTAE parameter list (SAHSTA) to determine whether an ESTAE environment has been established. If so, IDCSA01 issues a user ABEND macro with the user completion code equal to the code given to UABORT in register 15.

Module: IDCSA01

Procedure: IDCSA01

3. IDCSA01 issues the SNAP macro and takes a full region dump with an identification of '00'. Only dumps from the UABORT routine have an identification of '00'.

Module: IDCSA01

Procedure: IDCSA01

4. GDTIOH provides the address of the IODATA. The address of the IOCSTR chain is IODIOC. The UABORT routine goes through the chain of IOCSTRs and tests each one to determine if it is open. The DCB—for non-VSAM data sets—or the ACB—for VSAM data sets—is checked to determine if the data set is open or closed. If the data set is open, IDCSA01 issues a CLOSE macro to close the data set. The processing continues until the end of the chain is reached.

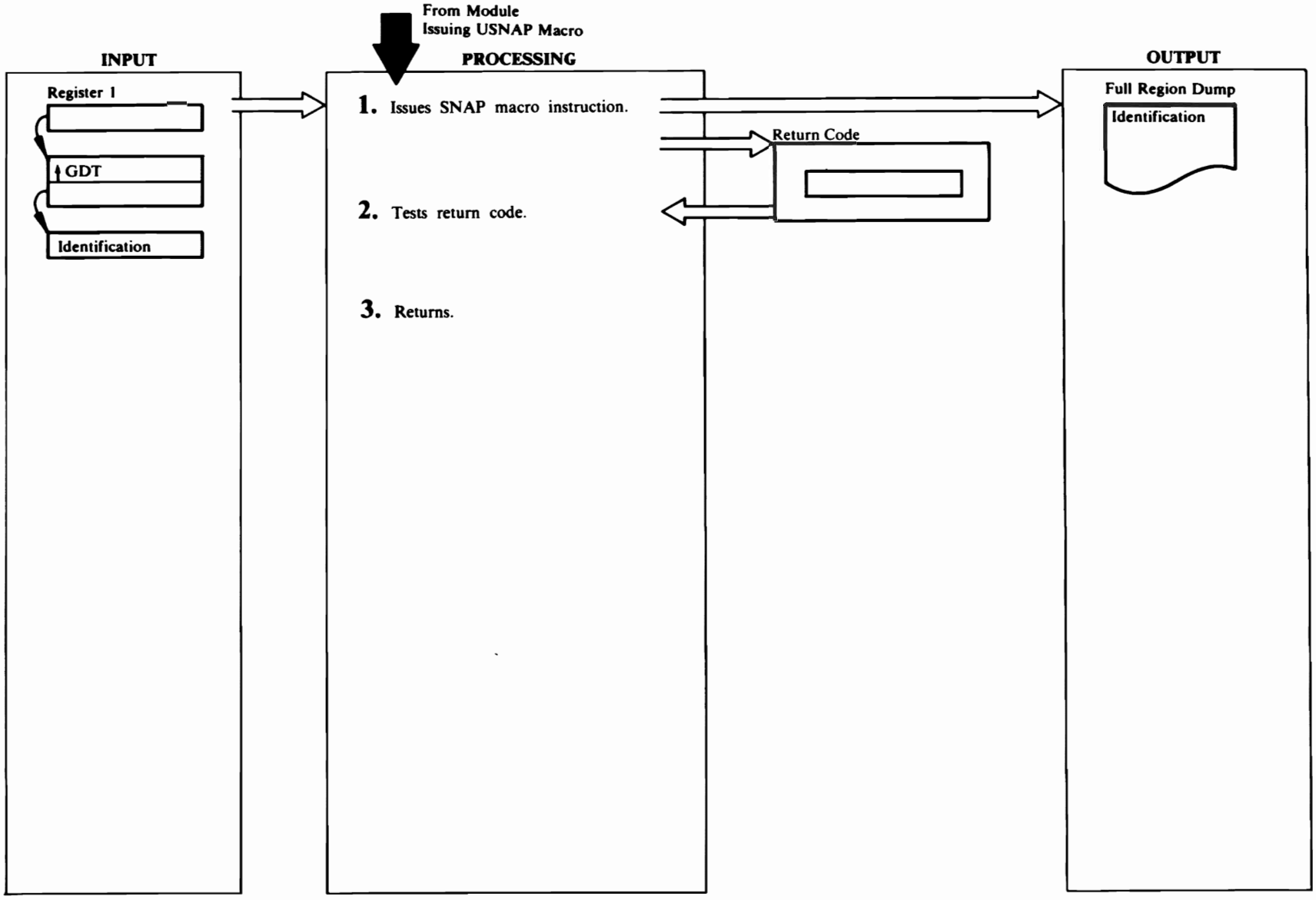
   When the end of the chain is reached, IDCSA01 checks for any open EXCP data sets found during the scan of the I/O Adapter EXCP control block chain (IODIOX). For each open EXCP data set, IDCSA01 issues a UEXCP macro to close the dat set. The processing continues until all open EXCP data sets have been closed.

Module: IDCSA01

Procedure: IDCSA01

5. IDCSA01 returns control to the invoker of Access Method Services with a code of 16 in register 15 to indicate that a catastrophic error has occurred.

Diagram 5.2.2 USNAP Macro



Extended Description for Diagram 5.2.2

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 issues a SNAP macro with the identification provided as the second parameter.

Module: IDCSA02

Procedure: IDCSA02

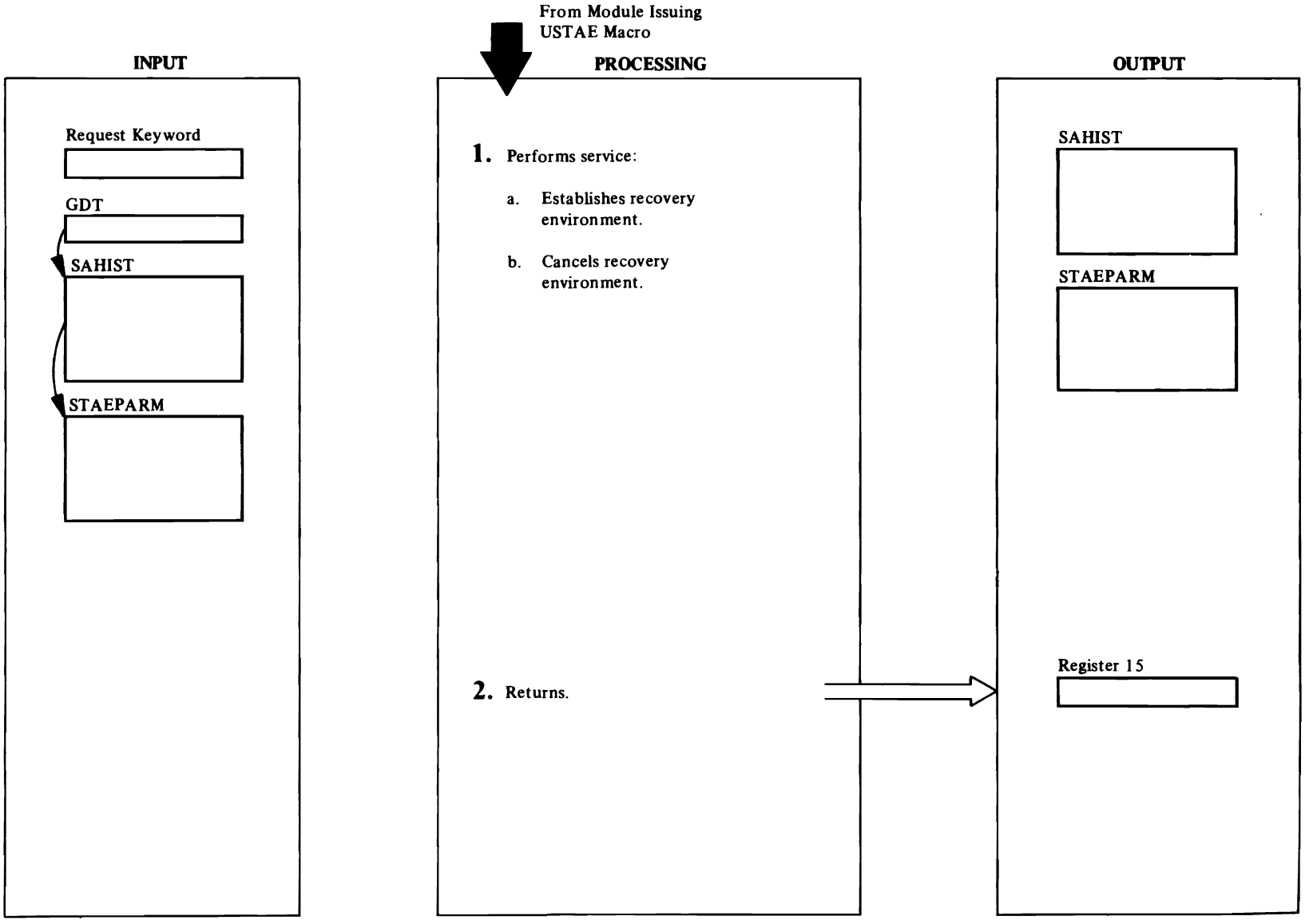
2. If the return code from the SNAP macro is non-zero, IDCSA02 issues a UABORT macro.

Module: IDCSA02

Procedure: IDCSA02

3. IDCSA02 returns control to the module that issued the USNAP macro.

Diagram 5.2.3 System Adapter – USTAE Macro



Extended Description for Diagram 5.2.3

Module: IDCSA10

Procedure: IDCSAST, SETSSTAE, CANSTAE

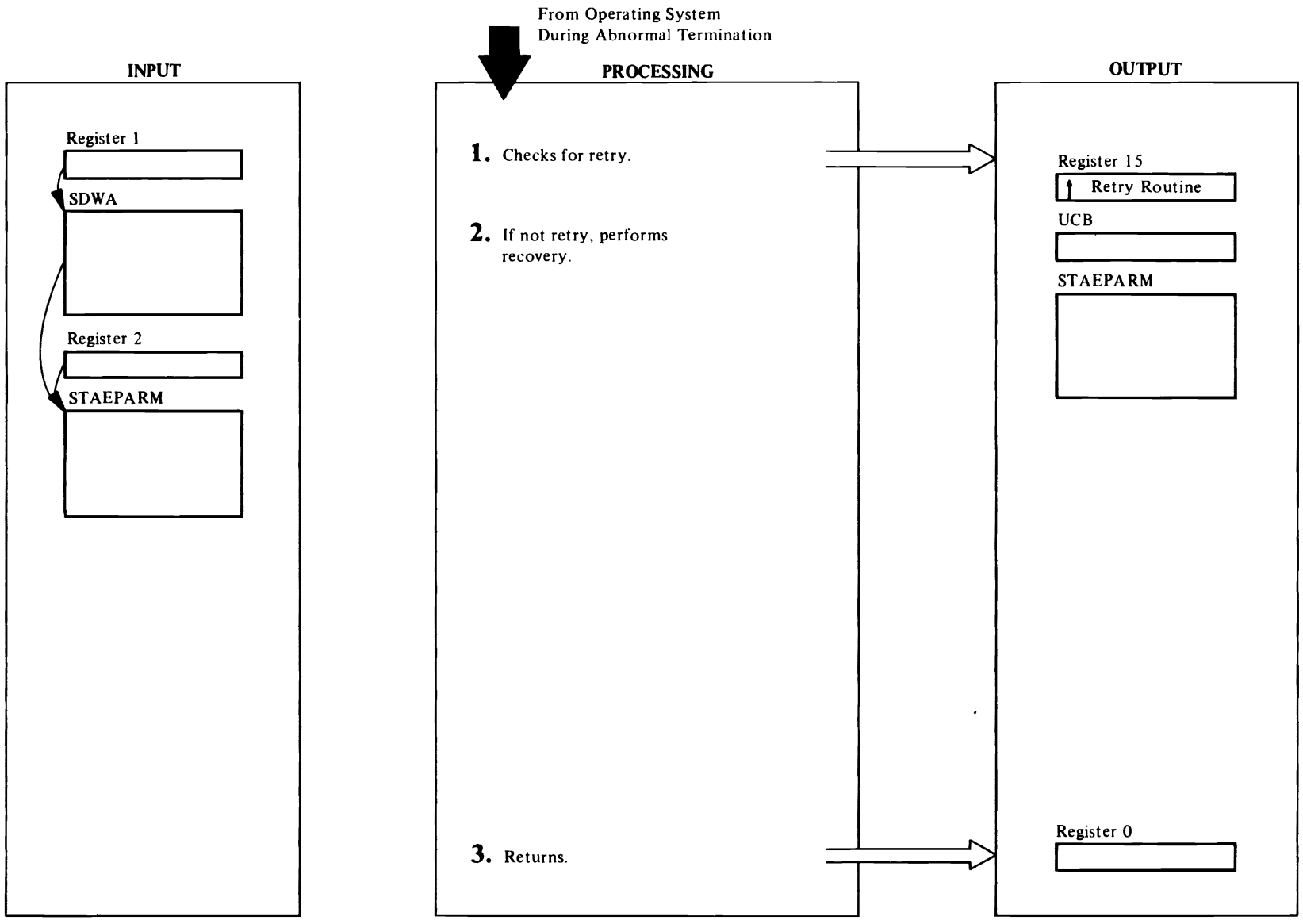
1. IDCSAST checks the request keyword to determine the type of service to perform:
   a. 'SET' indicates a request to establish an ESTAE environment. If the address of the ESTAE parameter is available, SETSTAE uses the established parameter list. Otherwise, it initializes a new ESTAE parameter list and puts its address in the SAHIST data area. SETSTAE issues an ESTAE macro.
   b. 'CANCEL' indicates a request to cancel an ESTAE environment. CANSTAE issues an ESTAE macro to cancel the ESTAE environment.

Module: IDCSA10

Procedure: IDCSAST

2. If any errors occurred, a message is written with the UPRINT macro. IDCSAST returns to the caller with a return code in register 15.

Diagram 5.2.4 System Adapter – Recovery During Abnormal Termination



Extended Description for Diagram 5.2.4

Module: IDCSA10

Procedure: STAEEXIT

1. STAEEXIT checks register 0 to determine whether the work area (SDWA) is available. If not, STAEEXIT doesn't return control to any retry routine. If the work area is available, STAEEXIT checks STAEPARM to determine whether a retry routine has been established by the UEXCP macro and checks the ABEND code in the SDWA to determine whether abnormal termination was caused by an invalid password entered by the operator. STAEEXIT puts the address of the UEXCP retry routine in the SDWA.

Module: IDCSA10

Procedure: RECOVERY

2. RECOVERY performs the recovery operations, depending on the information in STAEPARM:

   If a mass storage volume has been mounted but its UCB has not been posted, RECOVERY demounts the volume by issuing an IDBMNTDE macro.

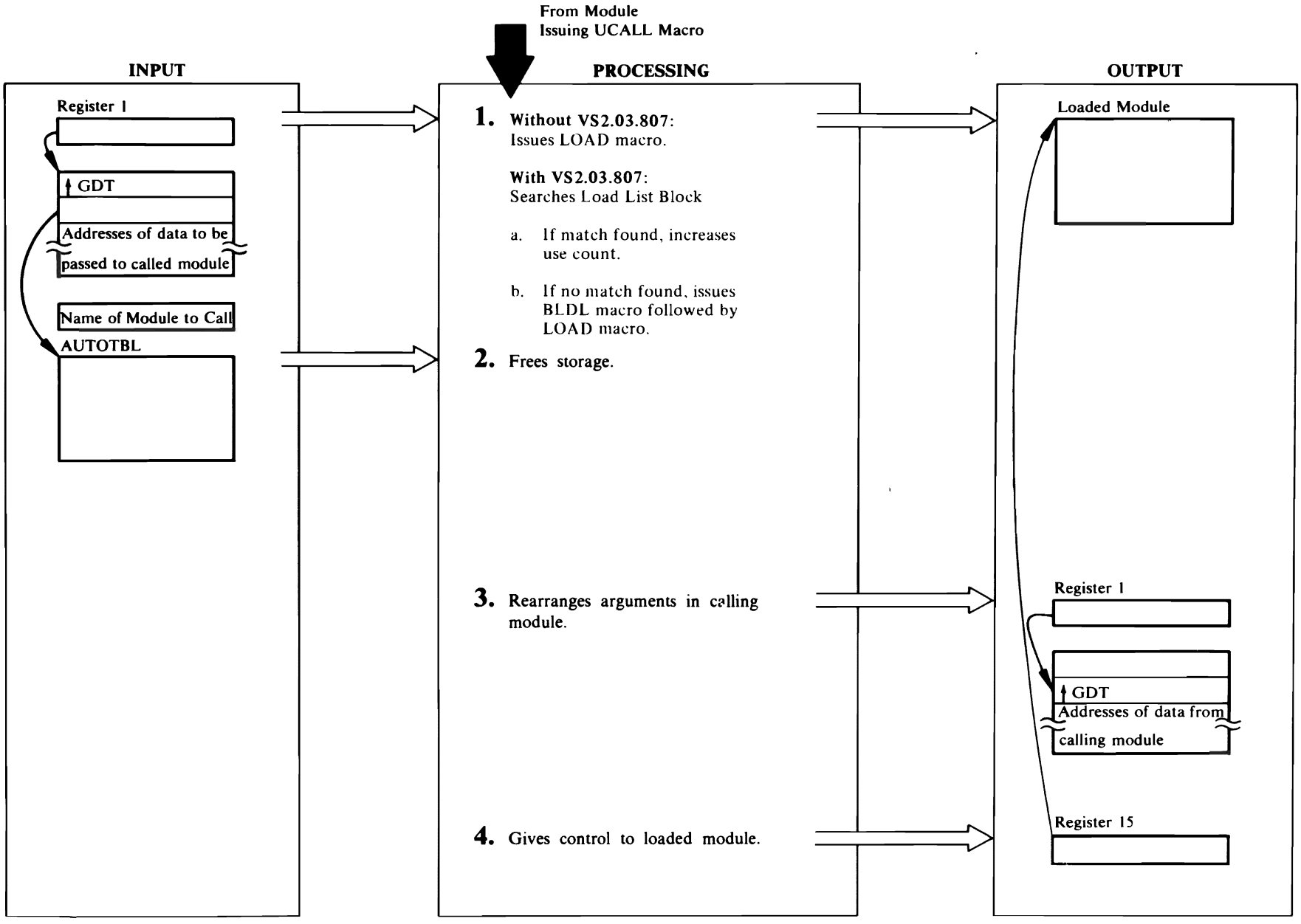
   If a mass storage volume has been demounted but its UCB has not been cleared, RECOVERY constructs SV82LIST and issues SVC 82 to clear the UCB. SVC 82 removes the volume serial number from the UCB and marks the UCB "not ready."

   If the UCB of a real volume has been cleared but the volume is still mounted, RECOVERY issues an IDBMNTDE macro to demount any mass storage volume with the same serial number that is mounted and issues SVC 82 to clear the UCB of the demounted volume. RECOVERY then issues SVC 82 to post the real UCB. SVC 82 restores the volume serial number and marks the UCB "ready." 9 If a volume serial number has been enqueued but that mass storage volume is no longer mounted, RECOVERY issues a DEQ macro. The major name is SYSZVOLS, and the minor name is the volume serial number.

Module: IDCSA10

3. RECOVERY returns to the caller with a return code in register 0. 4 indicates a retry, and zero indicates to continue with ABEND.

Diagram 5.3.1 UCALL Macro



Extended Description for Diagram 5.3.1

(Without VS2.03.807)

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 loads the program named by the UCALL macro with a LOAD macro.

(With VS2.03.807)

Module: IDCSA02

Procedure: IDCSA02, FINDNAME

1. IDCSA02 is entered at the UCALL entry point, and the FINDNAME procedure is called to search the Load List Block (LLBLK) for a match of the name of the module to be loaded. If a match is found, the use count for that module is increased by one. The entry point of the module to be loaded is obtained from the LLBLK entry and processing continues at step 2.

   If a match is not found, a BLDL macro is issued using the module name, followed by a LOAD macro using the BLDL list. The module name, entry point, and size are placed in the next available slot (if one is available) in LLBLK, and the module use count is set to one. The slot is removed from the available slot queue and the next available slot pointer is updated. If a slot is not available, FINDNAME issues a UGSPACE for a new LLBLK, initializes it, and places it on the LLBLK chain before the new module entry is built. Control goes to step 2.

   If the return code from BLDL is non-zero, FINDNAME issues a UABORT 56 and terminates the job. If the return code from UGSPACE is non-zero, FINDNAME issues a UABORT 28 and terminates the job.

Module: IDCSA02

Procedure: IDCSA02

2. IDCSA02 checks the AUTOTBL for the number of outstanding storage requests for IDCSA02. The number is in the STATUS section for IDCSA02. If the number is greater than one, storage other than the storage address in the AUTOTBL has been obtained for IDCSA02. The amount of storage is in the PL/S-generated variable @SIZDATD, and the address is in register 11. IDCSA02 issues a FREEMAIN, and the number in STATUS is decreased by one. If the number in STATUS is one, a FREEMAIN is not issued because the storage is saved for the next time IDCSA02 is given control. The status is reduced by one.

Module: IDCSA02

Procedure: IDCSA02

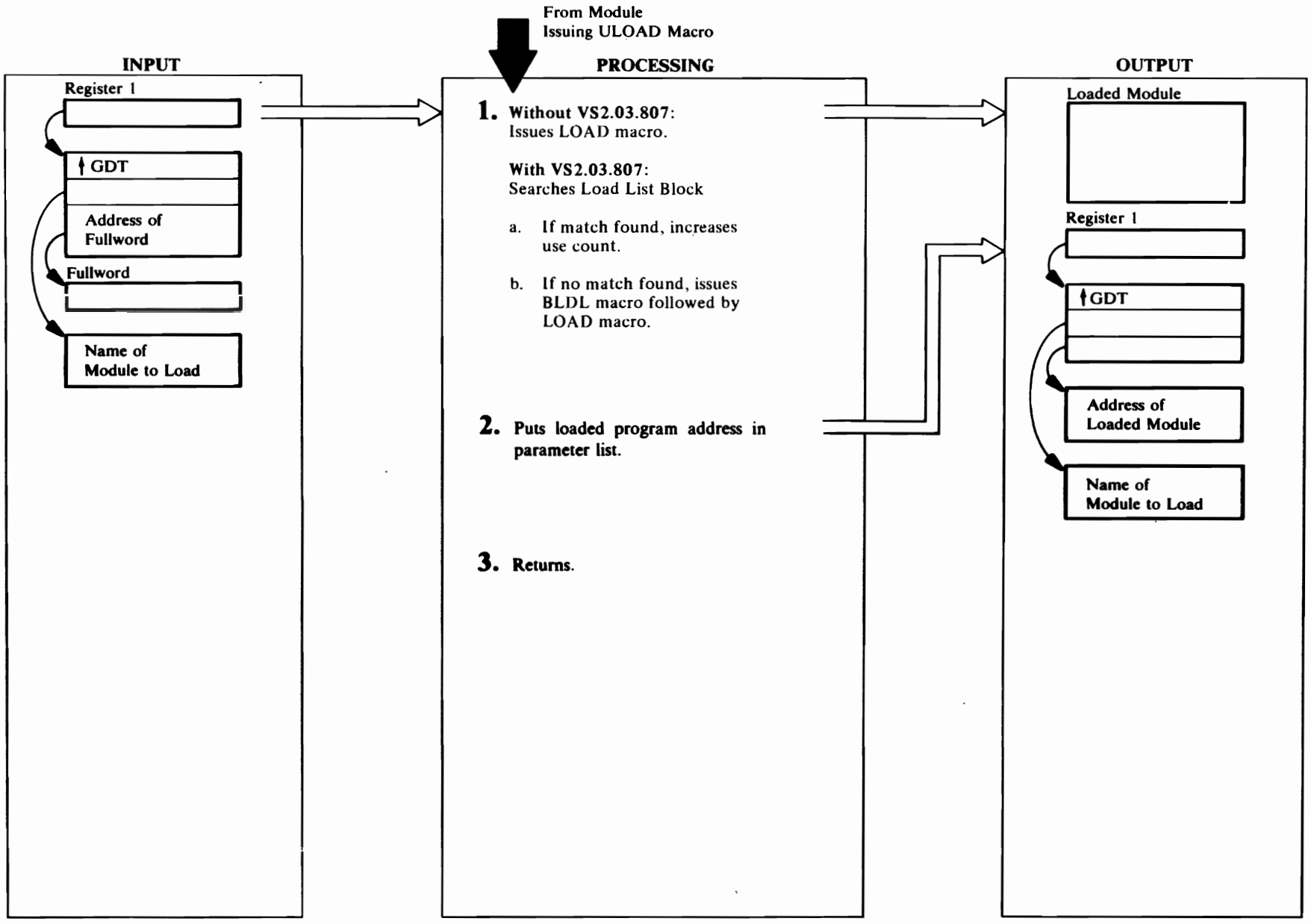
3. IDCSA02 copies the address of the GDT from the first parameter in the calling program to the second parameter in the calling program. IDCSA02 puts the address of the second parameter in the calling program—not the address of the GDT—in register 1. Register 1 now points to a contiguous list of parameters for the called program.

Module: IDCSA02

Procedure: IDCSA02

4. IDCSA02 puts the address of the calling program into register 15. IDCSA02 restores all registers, except 1 and 15, from the calling program's save area. IDCSA02 gives control to the called program.

Diagram 5.3.2 ULOAD Macı



Extended Description for Diagram 5.3.2

Without VS2.03.807

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 issues a LOAD macro using the name of the program given to the ULOAD macro.

(With VS2.03.807)

Module: IDCSA02

Procedure: IDCSA02, FINDNAME

1. IDCSA02 is entered at the ULOAD entry point, and the FINDNAME procedure is called to search the Load List Block (LLBLK) for a match of the name of the module to be loaded. If a match is found, the use count for that module is increased by one. The entry point of the module to be loaded is obtained from the LLBLK entry and processing continues at step 2.

   If a match is not found, a BLDL macro is issued using the module name, followed by a LOAD macro using the BLDL list. The module name, entry point, and size are placed in the next available slot (if one is available) in LLBLK, and the module use count is set to one. The slot is removed from the available slot queue and the next available slot pointer is updated. If a slot is not available, FINDNAME issues a UGSPACE for a new LLBLK, initializes it, and places it on the LLBLK chain before the new module entry is built. Control goes to step 2.

   If the return code from BLDL is non-zero, FINDNAME issues a UABORT 56 and terminates the job. If the return code from UGSPACE is non-zero, FINDNAME issues a UABORT 28 and terminates the job.

Module: IDCSA02

Procedure: IDCSA02

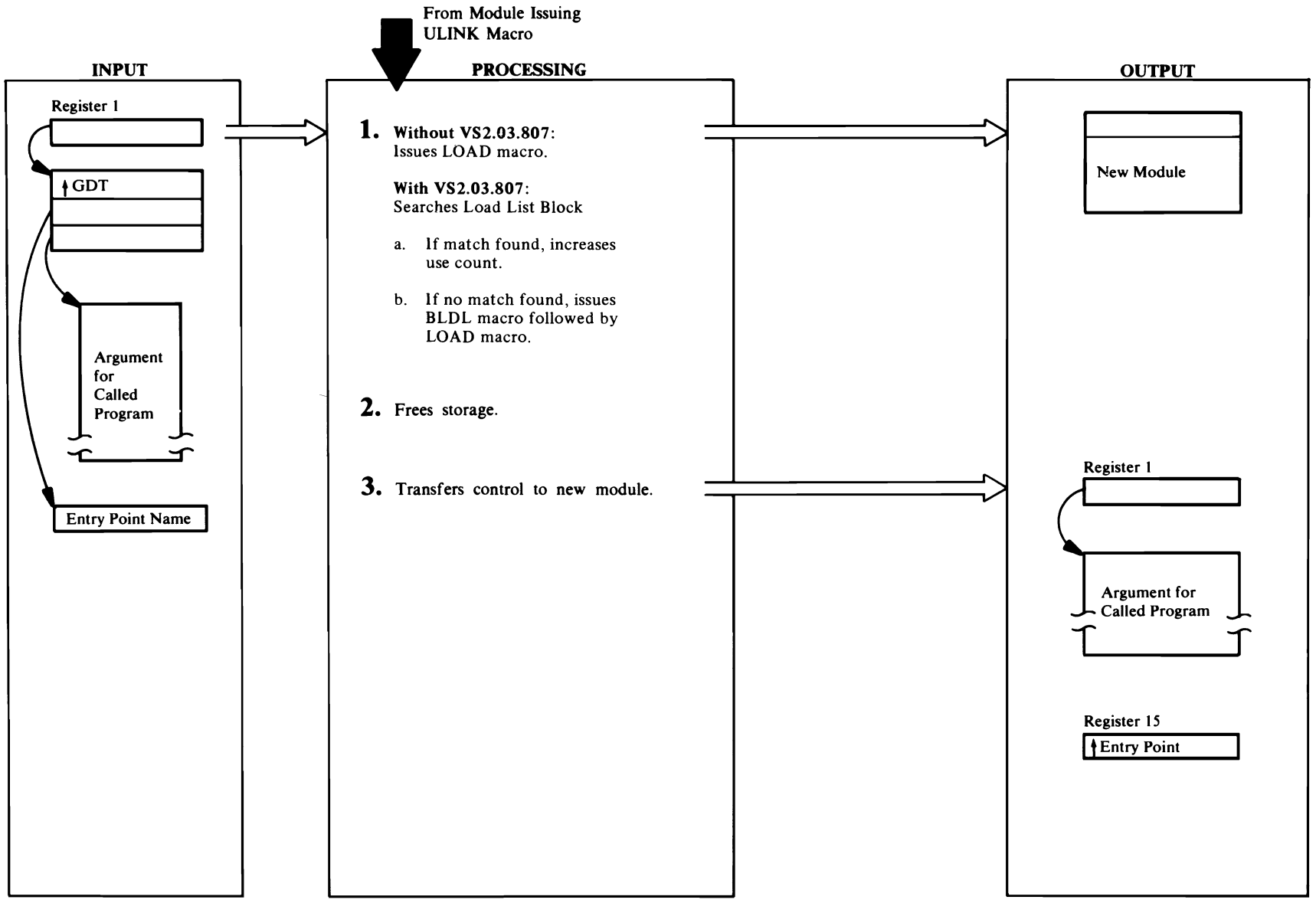
2. IDCSA02 puts the address of the loaded program in the calling program at the address specified with the third parameter.

Module: IDCSA02

Procedure: IDCSA02

3. IDCSA02 returns control to the module that issued the ULOAD macro.

Diagram 5.3.3 ULINK Macro



Extended Description for Diagram 5.3.3

(With VS2.03.807)

Module: IDCSA02

Procedure: IDCSA02

1. ULINK uses the entry point name of the new module to issue a LOAD macro. ULINK saves the address of the new module.

(With VS2.03.807)

Module: IDCSA02

Procedure: IDCSA02, FINDNAME

1. IDCSA02 is entered at the ULINK entry point, and the FINDNAME procedure is called to search the Load List Block (LLBLK) for a march of the name of the module to be loaded. If a match is found, the use count for that module is increased by one. The entry point of the module to be loaded is obtained from the LLBLK entry and processing continues at step 2.

   If a match is not found, a BLDL macro is issued using the module name, followed by a LOAD macro using the BLDL list. The module name, entry point, and size are placed in the next available slot (if one is available) in LLBLK, and the module use count is set to one. The slot is removed from the available slot queue and the next available slot pointer is updated. If a slot is not available, FINDNAME issues a UGSPACE for a new LLBLK, initializes it, and places it on the LLBLK chain before the new module entry is built. Control goes to step 2.

   If the return code from BLDL is non-zero, FINDNAME issues a UABORT 56 and terminates the job. If the return code from UGSPACE is non-zero, FINDNAME issues a UABORT 28 and terminates the job.

Module: IDCSA02

Procedure: IDCSA02

2. The ULINK macro checks AUTOTBL for the number of times module IDCSA02 has been called. If it has been called more than once, ULINK issues a FREEMAIN to free the storage assigned to the last call of IDCSA02. ULINK also decreases the count of calls to IDCSA02 kept in AUTOTBL.

Module: IDCSA02

Procedure: IDCSA02

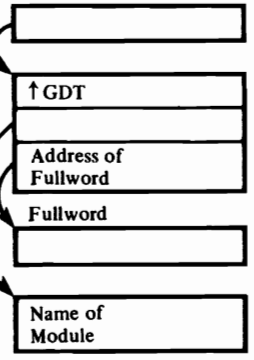
3. If the module that issued the ULINK provided an argument list for the calling program, ULINK puts the address of the argument list in register 1. ULINK does not turn on the high order bit of the last argument in the list. That is the responsibility of the module that issued the ULINK. ULINK restores register 13 so it contains the address of the save area in the module that issued the ULINK. This forces the new module to use the save area provided by the module issuing the ULINK. ULINK puts the return address of the module issuing the ULINK in register 14. When the new module returns control, control does not return to ULINK, but returns to the module issuing the ULINK. ULINK puts the entry point address of the new module in register 15 and gives control to the new module.

Diagram 5.3.4 UDELETE Macro (VS2.03.807 only)

From Diagram 5.0

INPUT

Register 1



PROCESSING

1. Searches for module name in Load List Block (LLBLK).
2. Decrements use count by 1.
3. Returns to caller.

OUTPUT

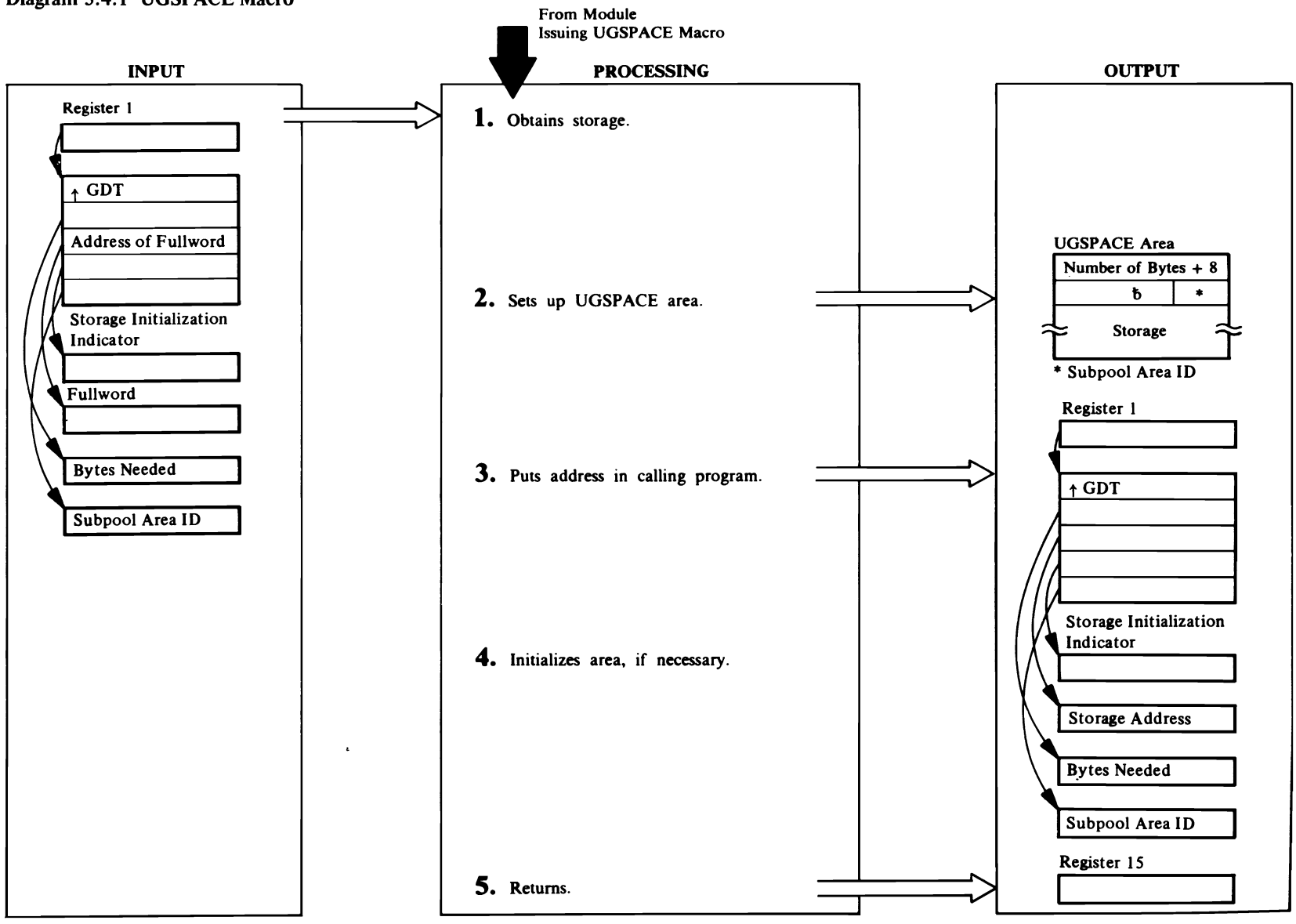
Extended Description for Diagram 5.3.4 (VS2.03.807 only)

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 is entered at the UDELETE entry point. UDELETE searches the LLBLK for the name of the module to be deleted.
2. If the module name is found, the use count for the module is decremented by 1.
3. Control returns to the call of UDELETE.

Diagram 5.4.1 UGSPACE Macro



Extended Description for Diagram 5.4.1

Module: IDCSA02

Procedure: IDCSA02 **(Without VS2.03.807)**
IDCSA02, RELEORE **(With VS2.03.807)**

1. IDCSA02 first checks for a request for a specific subpool area ID. If one is specified, IDCSA02 puts the subpool area ID in the GETMAIN parameter list; otherwise IDCSA02 sets the ID to zero (which is the default subpool area ID). IDCSA02 then issues a GETMAIN for the number of bytes requested plus 8 for the UGSPACE area that precedes each storage area. If the return code from the GETMAIN is non-zero, the address of the storage area is set to zero, and control is given to step 5. However, if **VS2.03.807** is installed, the following additional processing takes place before the address of the storage area is set to zero and control given to step 5:

   The UGSPACE procedure calls the RELECORE procedure. RELECORE searches the active slots in the Load List Block (LLBLK) and issues a DELETE macro for all modules whose use count is zero. UGSPACE then reissues the GETMAIN macro. Now if the return code is still non-zero, the address of the storage area is set to zero, and control is given to step 5.

   If the return code from GETMAIN is zero, control is given to step 2.

Module: IDCSA02

Procedure: IDCSA02

2. IDCSA02 puts the number of bytes in the storage area plus 8 in the first word of the UGSPACE area. IDCSA02 sets the first three bytes of the second word to blanks to distinguish a UGSPACE area from a UGPOOL area. If a subpool area ID was specified, IDCSA02 puts the ID in the fourth byte of the second word; otherwise IDCSA02 sets the fourth byte to zero.

Module: IDCSA02

Procedure: IDCSA02

3. IDCSA02 puts the address of the storage area—not the UGSPACE area—in the calling program at the address specified by the third parameter.

Module: IDCSA02

Procedure: IDCSA02

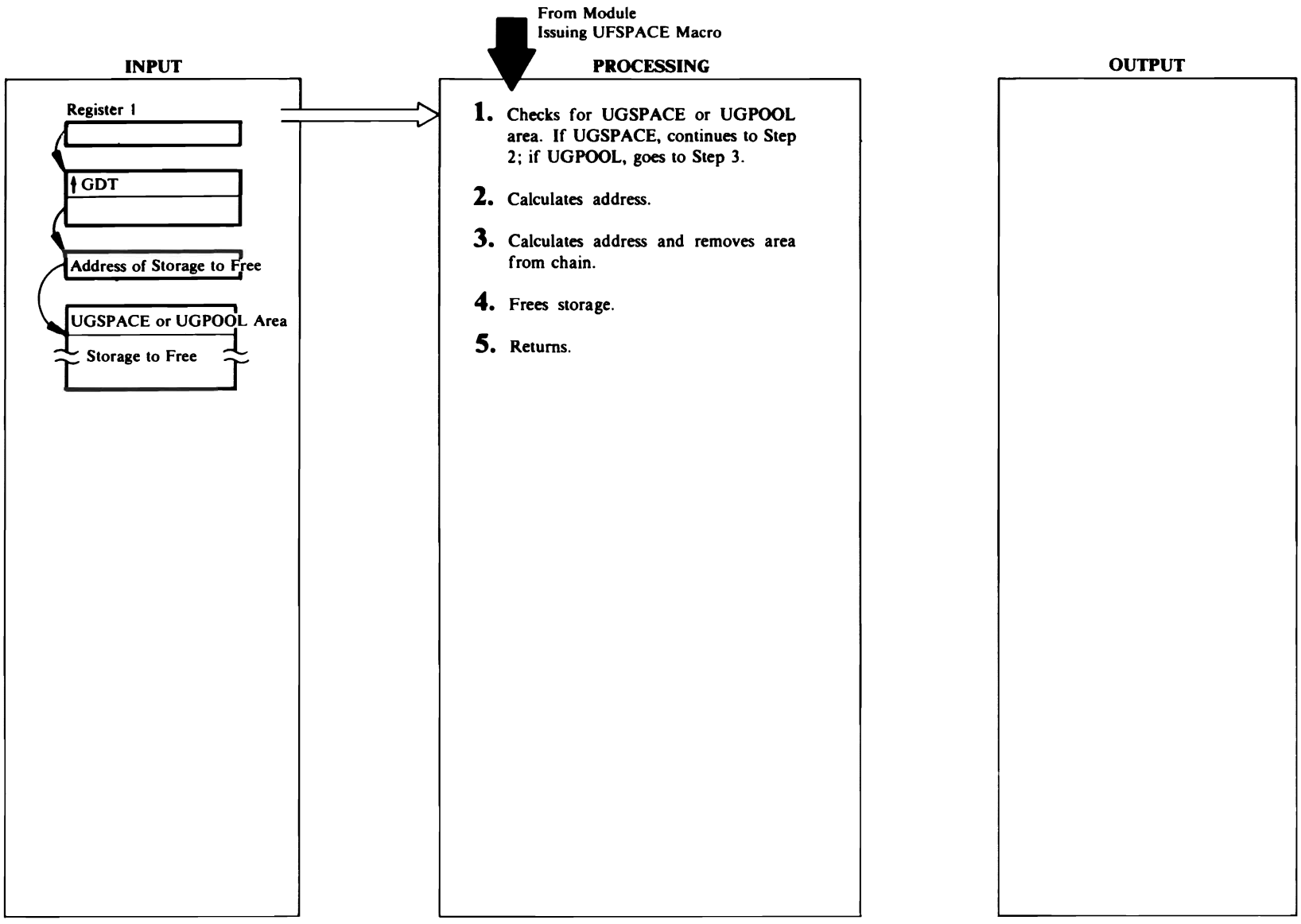
4. If SETZERO or SETBLANK was specified as the fourth parameter, IDCSA02 sets the storage area to zeros or blanks, respectively. If SETZERO or SETBLANK was not specified or NOSET was specified, the storage area is not changed.

Module: IDCSA02

Procedure: IDCSA02

5. IDCSA02 puts a return code in register 15 and returns control to the module that issued the UGSPACE macro.

Diagram 5.4.2 UFSPACE Mac



Extended Description for Diagram 5.4.2

Module: IDCSA02

Procedure: IDCSA02

1. The address of the area to free is used by IDCSA02 to determine if the area was obtained with a UGSPACE or a UGPOOL. If the first three bytes of the fullword at the address minus 4 contains blanks, the area was obtained with a UGSPACE.

Module: IDCSA02

Procedure: IDCSA02

2. If the storage area was obtained with UGSPACE, a UGSPACE area precedes the area. The first word in the UGSPACE area contains the length of the area to free; the fourth byte of the second word contains the subpool area ID. The address of the area to free is calculated by subtracting 8 from the area address.

Module: IDCSA02

Procedure: IDCSA02

3. If the storage area was obtained with a UGPOOL, a UGPOOL area precedes the storage. The length of the area to free is at the third word of the UGPOOL area. The address of the area to free is calculated by subtracting 16 from the area address. The forward and backward chains are updated to remove this area from the chain. If this is the last area in the chain, the address of the last area in the chain in GPLAST in the System Adapter Historical Data area is updated by IDCSA02.

Module: IDCSA02

Procedure: IDCSA02

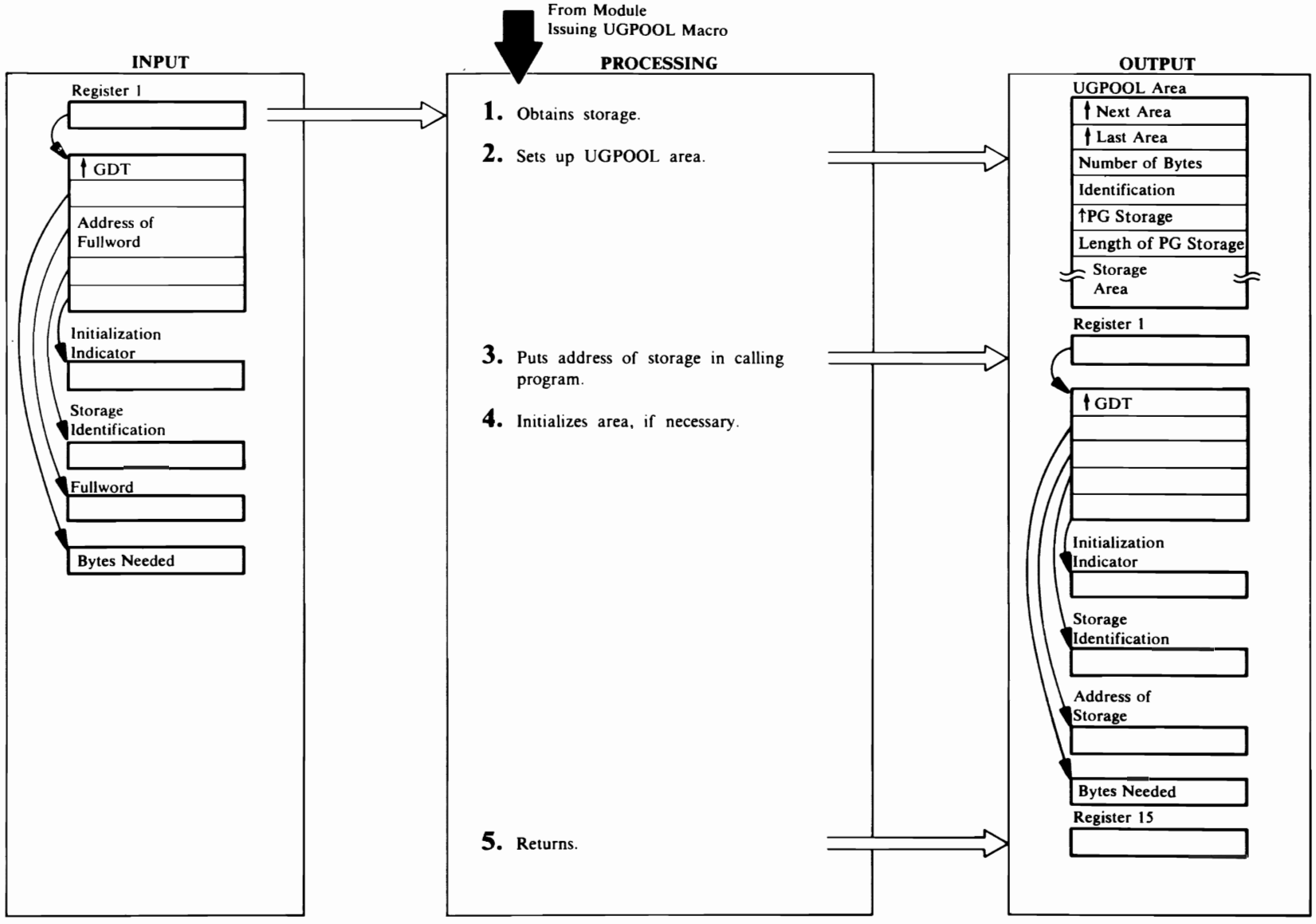
4. A FREEMAIN macro is issued to release the storage plus its UGSPACE or UGPOOL area.

Module: IDCSA02

Procedure: IDCSA02

5. IDCSA02 returns control to the module that issued the UFSPACE macro.

Diagram 5.4.3 UGPOOL Macro



Extended Description for Diagram 5.4.3

Module: IDCSA02, **(Without VS2.03.807)**
IDCSA02, RELECORE **(With VS2.03.807)**

Procedure: IDCSA02

1. IDCSA02 checks for a UGPOOL request for storage on a page boundary (storage identification equal to 'xxPG').

   a. If page-boundary storage was not requested, IDCSA02 issues a GETMAIN for the number of bytes requested, plus 16 for the UGPOOL area. If the return code from the GETMAIN is non-zero, the storage address in the calling program is set to zero, and control is given to step 5. However, if **VS2.03.807** is installed, the following additional processing takes place before the address of the storage area is set to zero and control given to step 5:

      If the return code from the GETMAIN is zero, control is given to step 2.

   b. If page-boundary storage was requested, IDCSA02 issues a GETMAIN, for storage on a page boundary, for the number of bytes requested. If the return code from the GETMAIN is nonzero, the storage address in the calling program is set to zero and processing continues at step 5. However, if **VS2.03.807** is installed, the following additional processing takes place before the address of the storage area is set to zero and control given to step 5:

      The UGSPACE procedure calls the RELECORE procedure. RELECORE searches the active slots in the Load List Block (LLBLK) and issues a DELETE macro for all modules whose use count is zero. UGSPACE then reissues the GETMAIN macro. Now if the return code is still non-zero, the address of the storage area is set to zero, and control is given to step 5.

   c. If the return code from the GETMAIN is zero, IDCSA02 issues a second GETMAIN for a 24-byte UGPOOL area. If the return code from the GETMAIN is nonzero, the storage address in the calling program is set to zero, a FREEMAIN is issued to free the page-boundary storage gotten in step 1.b, and processing continues at step 5.

Module: IDCSA02

Procedure: IDCSA02

2. The new storage area is chained to the other storage areas obtained with UGPOOL. The head of the chain is in GPFIRST and the tail is in GPLAST in the System Adapter Historical Data Area. The new storage area is chained by IDCSA02 to the tail of the list. IDCSA02 sets the forward chain pointer to zero. The backward chain pointer contains the address of the next-to-last area. The identification from the calling module is put in the fourth word of the UGPOOL area. GPLAST is set to the address of the new storage area.

   If page-boundary storage was not requested, IDCSA02 sets the number of bytes in the storage area to the number of bytes requested, plus 16 for the UGPOOL area.

   If page-boundary storage was requested, IDCSA02 sets the number of bytes in the storage area to 24 (size of the UGPOOL area only). IDCSA02 puts the address of the page-boundary storage in the fifth word of the UGPOOL area and puts its length in the sixth word.

Module: IDCSA02

Procedure: IDCSA02

3. IDCSA02 puts the address of the storage area in the calling program at the address specified by the third parameter. The address is either the address, plus 16, of the storage obtained in step 1.a, or the address of the storage obtained in step 1.b.

Module: IDCSA02

Procedure: IDCSA02

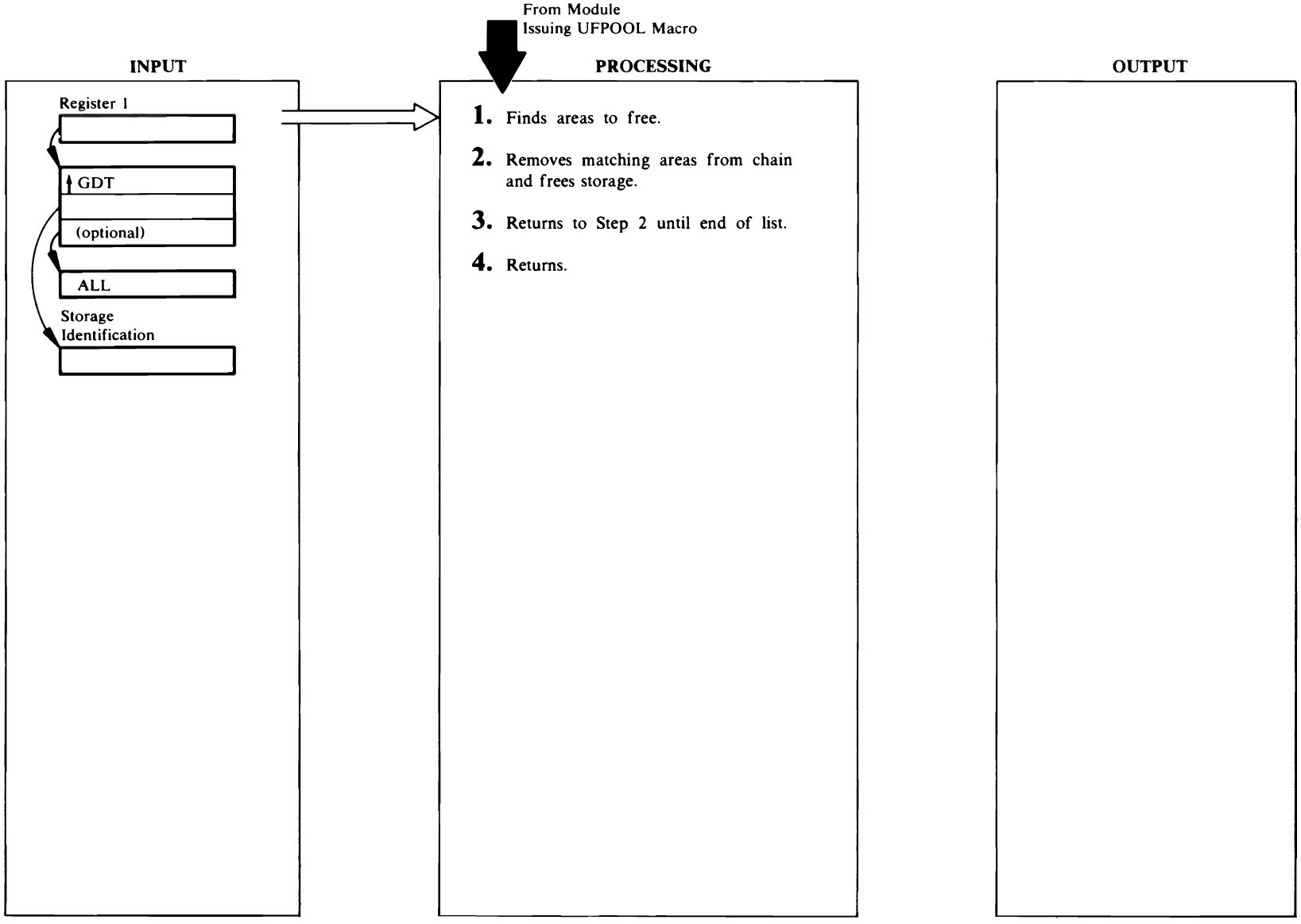
4. If SETZERO or SETBLANK was specified as the fifth parameter, IDCSA02 sets the storage area to zeros or blanks, respectively. If neither SETZERO or SETBLANK is specified, the storage is not changed.

Module: IDCSA02

Procedure: IDCSA02

5. IDCSA02 puts a return code in register 15 and returns control to the module that issued the UGPOOL macro.

Diagram 5.4.4 UFPOOL Macro



Extended Description for Diagram 5.4.4

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 examines the list of UGPOOL areas addressed from GPFIRST to find a match between the storage identifier supplied by the calling program and the identifier in the UGPOOL area. If the calling program specifies ALL as the third parameter, just the first two bytes of the identifiers are compared so that every storage area that matches is freed. If ALL is not specified, IDCSA02 compares four bytes of the identifiers to find the one storage area to be released.

Module: IDCSA02

Procedure: IDCSA02

2. If a match is found, IDCSA02 removes the UGPOOL area from the chain. If the storage to be released was not requested to be on a page boundary, IDCSA02 issues a FREEMAIN macro to release the 16-byte UGPOOL area and its storage area. If it *was* requested to be on a page boundary ('xxPG' storage identification), IDCSA02 issues a FREEMAIN macro to release the page-boundary storage using the address and length contained in the fifth and sixth words of the 24-byte UGPOOL area. IDCSA02 then issues a FREEMAIN macro to release the 24-byte UGPOOL area.

Module: IDCSA02

Procedure: IDCSA02

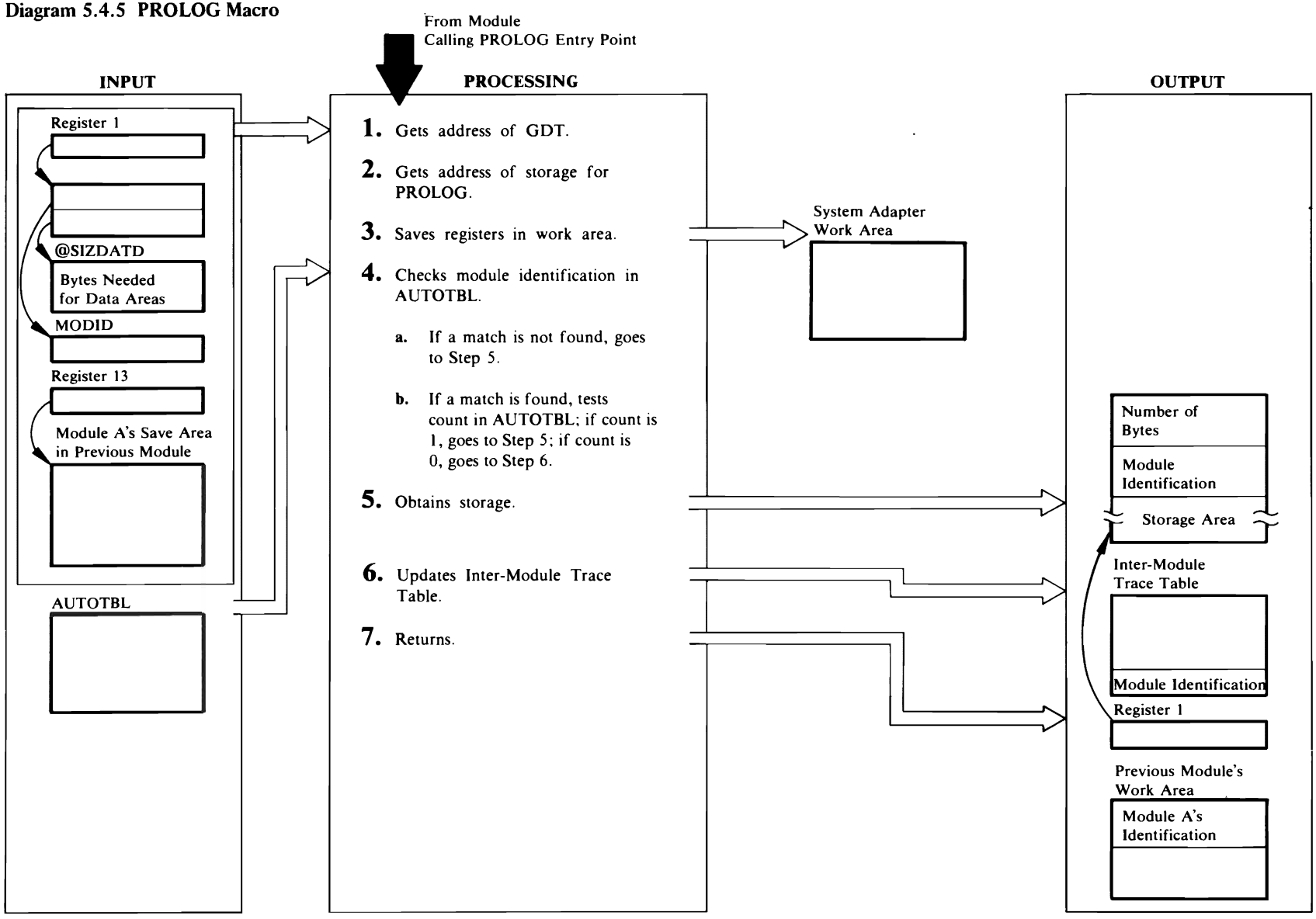
3. If the end of the chain has not been reached, IDCSA02 compares the next UGPOOL area. The entire list is searched for matching identifiers, regardless of whether ALL is specified or not. IDCSA02 returns control to step 2 until the end of the chain is reached.

Module: IDCSA02

Procedure: IDCSA02

4. IDCSA02 returns control to the module that issued the UFPOOL macro.

Diagram 5.4.5 PROLOG Macro



Extended Description for Diagram 5.4.5

Module: IDCSA03

Procedure: IDCSA03

1. The address of the GDT is the first parameter in the call to every Access Method Services module except the call to PROLOG. As an example, let's assume module A gives control to module B. First, module B stores registers in the save area in module A. Second, module B obtains storage for the data in module B. PL/S generates a GETMAIN macro instruction to obtain the storage. But GETMAIN doesn't work on DOS. So, instead of doing a GETMAIN, module B calls PROLOG to get storage for module B's data areas. When module B gets control, register 1 contains the address of a parameter list. By convention within Access Method Services, the first parameter in the parameter list is always the address of the GDT. When PROLOG gets control, register 13 contains the address of the save area in module A. IDCSA03 uses this address to get the address of the GDT.

Module: IDCSA03

Procedure: IDCSA03

2. The address of the storage area PROLOG uses for its data areas is in GDTSPR. IDCSA03 uses this address to establish addressability to the data areas in PROLOG.

Module: IDCSA03

Procedure: IDCSA03

3. Module B's registers are saved in PROLOG because module B doesn't have a save area yet. IDCSA03 chains together the save area in module A and the save area used for module B's registers in PROLOG.

Module: IDCSA03

Procedure: IDCSA03

4. IDCSA03 compares the module identifications in AUTOTBL with the four-character module identification module B passes as the first parameter to PROLOG. If IDCSA03 does not find a match, control goes to step 5. If a match is found, and module B is IDCSA02, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10,IDCIO01, IDCIO05, or IDCTP01, IDCSA01 may have already obtained storage for it. AUTOTBL contains the address of storage already obtained for these modules. IDCSA03 examines the number of times module B has been called. If the number is zero, module B is not using the storage whose address is in AUTOTBL. IDCSA03 does not do a GETMAIN and IDCSA03 gives the storage from AUTOTBL to module B for its data areas. IDCSA03 adds one to the number of times the module is called. If the count is greater than zero, the storage in AUTOTBL is already in use so IDCSA03 must do a GETMAIN. One is added to the number of times the module is called.

Module: IDCSA03

Procedure: IDCSA03

5. If module B did not get storage from AUTOTBL, IDCSA03 issues a GETMAIN for the number of bytes needed. PL/S-2 always puts the number of bytes in a constant called @SIZDATD which is the second parameter to PROLOG. IDCSA03 issues a GETMAIN for the number of bytes in @SIZDATD plus 8 for header information. If the return code from GETMAIN is non-zero, IDCSA03 issues a UABORT macro. However, if **VS2.03.807** is installed, the following occurs before the UABORT is issued:

   IDCSA03 searches the active slots in the Load List Block (LLBLK). It issues a DELETE macro for all modules whose use count is zero. The slots for these modules are placed back on the available slot queue. The GETMAIN macro is then retried. If the return code is still non-zero, IDCSA03 issues the UABORT macro.

   IDCSA03 puts the total length of the storage area in the first word of the header. IDCSA03 puts Module B's identification from MODID in the second word of the header.

Module: IDCSA03

Procedure: IDCSA03

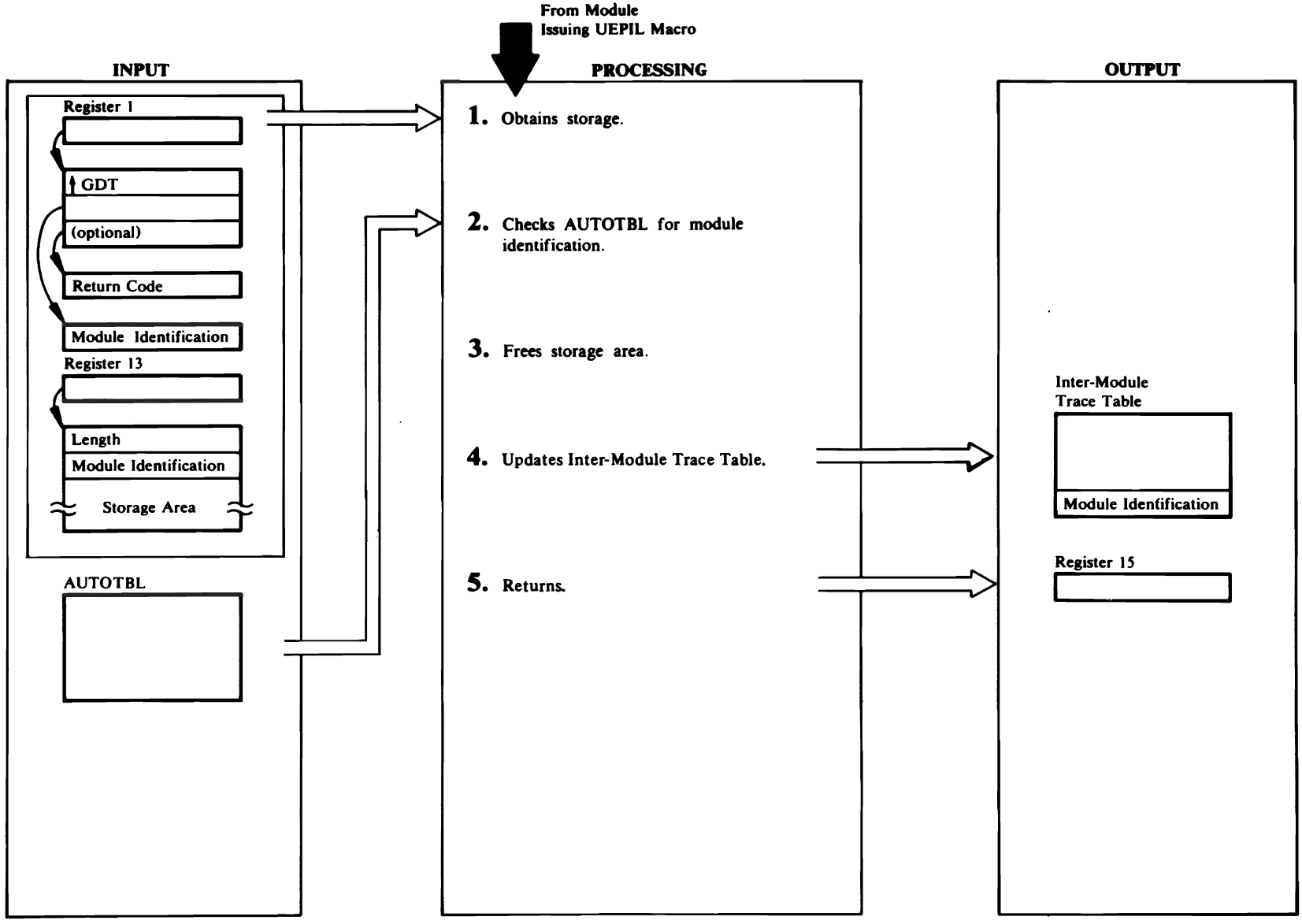
6. IDCSA03 adds module B's identification from MODID to the end of the Inter-Module Trace table. The first, oldest entry in the table is removed.

Module: IDCSA03

Procedure: IDCSA03

7. IDCSA03 puts module B's module identification in the first word of module A's save area. IDCSA03 restores the registers, with the exception of register one, from the work area in PROLOG to be as they were when module B gave control to PROLOG. Register one contains the address of the storage module B uses for its data area. IDCSA03 returns control to module B.

Diagram 5.4.6 UEPIL Macro



Extended Description for Diagram 5.4.6

Module: IDCSA03

Procedure: IDCSA03

1. Let's assume module A gives control to module B. Module B completes its processing and is ready to return control to module A. When module B is compiled on VS, PL/S generates a FREEMAIN for exit code. Rather than having one version of all modules for VS and another for DOS, each module—with a very few exceptions—issues a UEPIL macro to return control. See the chapter "Diagnostic Aids" for an illustration of save areas. The UEPIL bypasses the PL/S-generated FREEMAIN and allows the same module to operate on more than one operating system. When module B is ready to return control to module A, module B issues a UEPIL. UEPIL gets the address of the storage it is to use for data areas from GDTSPR. IDCSA03 saves the address of module B's storage area in register 13. IDCSA03 saves the address of module A's save area, which is obtained from module B's save area, and IDCSA03 sets the forward chain in module A's save area to zero.

Module: IDCSA03

Procedure: IDCSA03

2. IDCSA03 compares module B's module identification against the module identification in AUTOTBL. If a match is not found, control is given to step 3. If IDCSA03 finds a match, the number of times the module has been called is compared to one. If the number is one, IDCSA03 will not issue a FREEMAIN but reduces, by one, the number of times the module has been called. If the number is greater than one, IDCSA03 has acquired storage other than storage from the AUTOTBL, and this storage must be released. IDCSA03 subtracts one from the number of times the module has been called.

Module: IDCSA03

Procedure: IDCSA03

3. IDCSA03 subtracts eight from the address of module B's storage area to get the address of the header information. IDCSA03 issues a FREEMAIN with the length of the storage area as specified in the first word of the header.

Module: IDCSA03

Procedure: IDCSA03

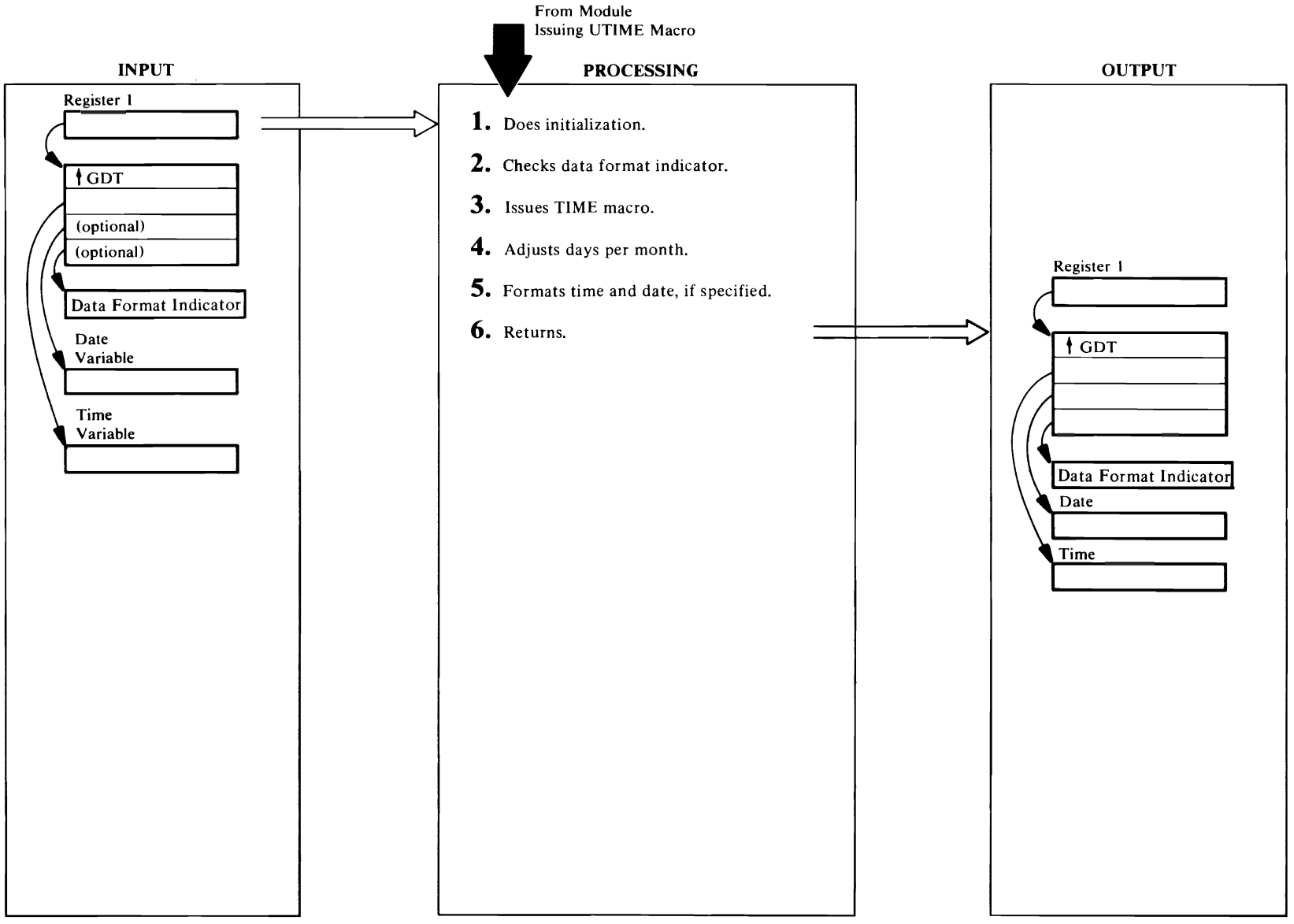
4. IDCSA03 puts the address of module A's save area in register 13. IDCSA03 removes the oldest module identification entry in the Inter-Module Trace table. IDCSA03 adds module A's module identification to the end of the Inter-Module Trace table. IDCSA03 obtains module A's module identification from the first word of the save area where module A saved registers when it was given control.

Module: IDCSA03

Procedure: IDCSA03

5. IDCSA03 restores all registers, except register 15, from module A's save area. Register 15 contains the return code from module B, if module B provides it, or zero. IDCSA03 returns control to module A.

Diagram 5.5.1 UTIME Macro



Extended Description for Diagram 5.5.1

Module: IDCSA02

Procedure: IDCSA02

1. IDCSA02 calculates the number of arguments passed to UTIME. IDCSA02 passes the input parameter list and a variable containing the number of arguments to IDCSA05.

Module: IDCSA05

Procedure: IDCSA05

2. If the caller incorrectly specifies the data format indicator, IDCSA05 issues a UABORT macro.

Module: IDCSA05

Procedure: IDCSA05

3. If the caller specifies FORMAT, IDCSA05 specifies a TIME macro with a DEC option. If HSECOND is specified, IDCSA05 issues a TIME macro with the BIN option. If CLOCK is specified, IDCSA05 issues a STCK instruction. If the caller does not indicate the data format, IDCSA05 issues a TIME macro with a MIC option.

Module: IDCSA05

Procedure: IDCSA05

4. IDCSA05 adjusts the number-of-days-per-month table for leap years. If the year returned by the TIME macro is divisible by four, IDCSA05 sets the number of days in February to 29.

Module: IDCSA05

Procedure: IDCSA05

5. If the caller specifies FORMAT, IDCSA05 formats the time as HH:MM:SS, where HH is hours, MM is minutes, and SS is seconds. The data is in decimal digits. If the date was requested and format specified, IDCSA05 formats the date as MM/DD/YY, where MM is the month, DD is the day, and YY is the year. The date is in decimal digits.

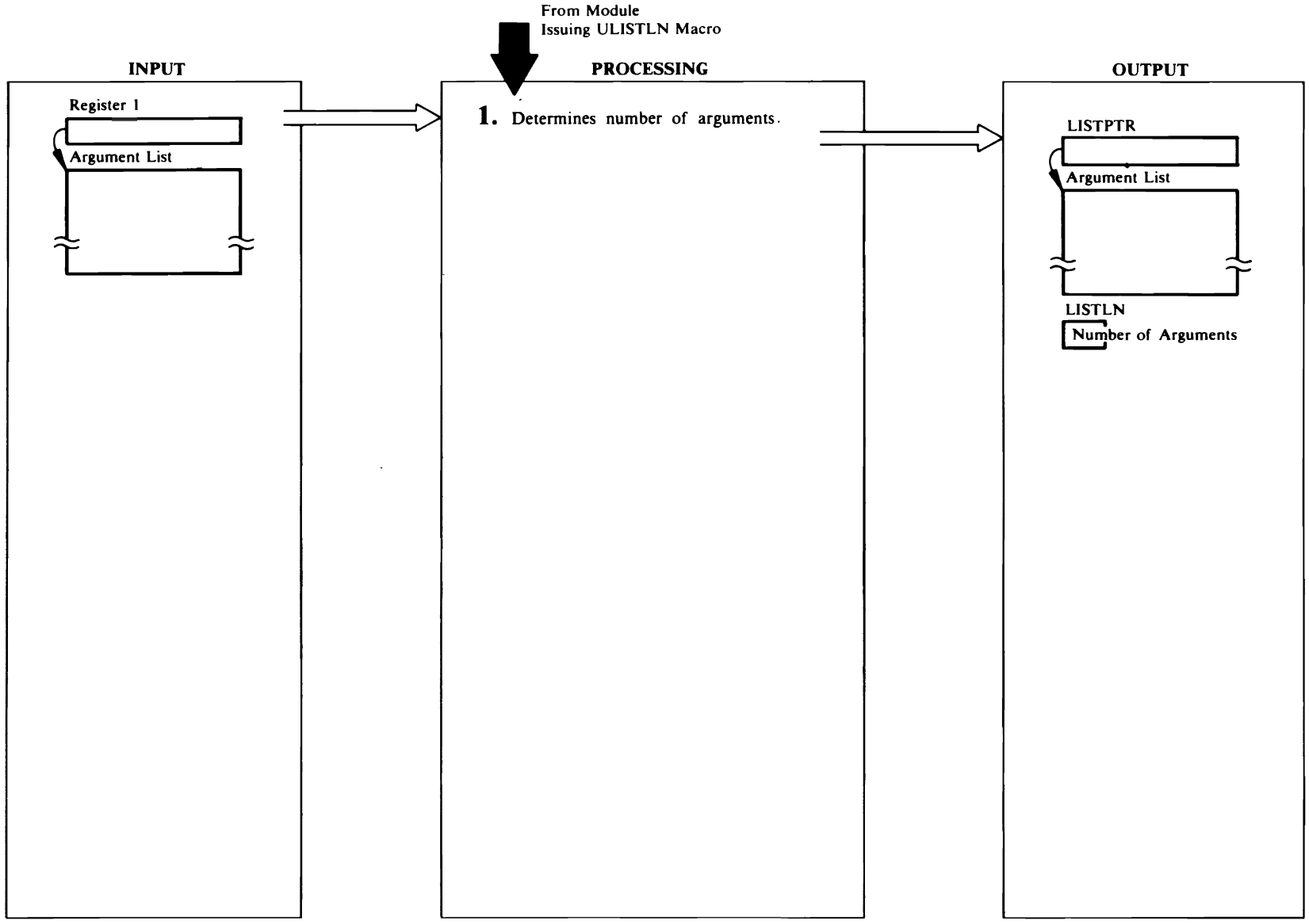
   If HSECOND is specified, IDCSA05 returns the time from the 24-hour clock in hundredths of seconds. If CLOCK is specified, IDCSA05 returns the time from the time-of-day clock in microseconds. If the date is requested and no data format is indicated, or HSECOND or CLOCK is specified, IDCSA05 returns the date in packed-decimal format, 00YYDDDF, where YY is the year, DDD is the day, and F is the sign digit.

Module: IDCSA05, IDCSA02

Procedure: IDCSA05, IDCSA02

6. IDCSA05 moves the time and date to the calling program at the addresses specified by parameters two and three. IDCSA05 returns control to IDCSA02, which returns control to the module that issued the UTIME macro.

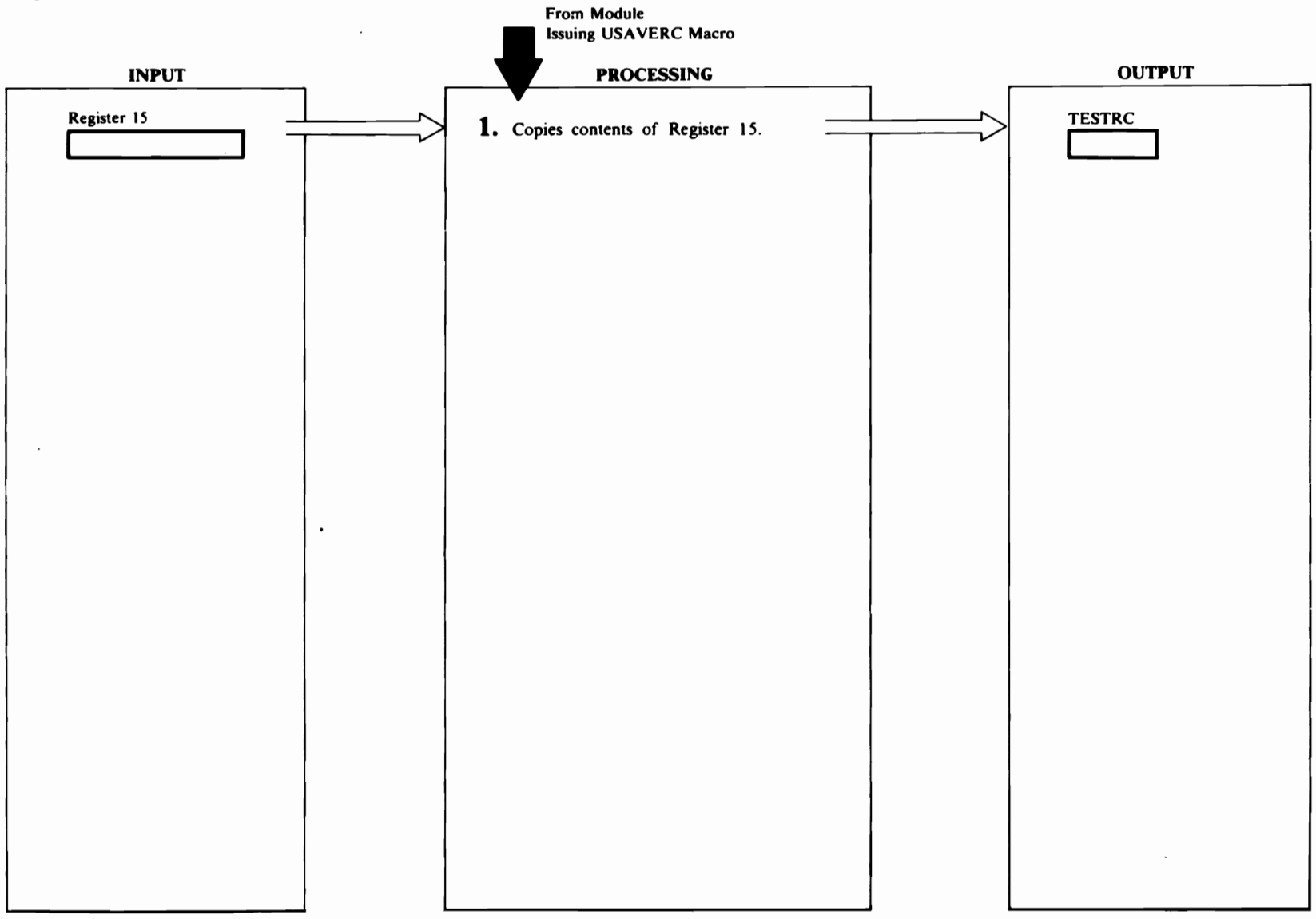
Diagram 5.6.1 ULISTLN Macro



Extended Description for Diagram 5.6.1

1. Unlike most Umacros, ULISTLN generates inline code that performs the function rather than a branch to another module. The code stores the address of the parameter list in register 1 in a fullword named LISTPTR. The code searches the argument list looking for the end of the list. The last argument in the list has a high order bit of one. The number of arguments in the list is put in a byte named LISTLN. If the end of the argument list is not found after 255 arguments, the search stops and LISTLN contains 255. Control continues with the next instruction in the program.

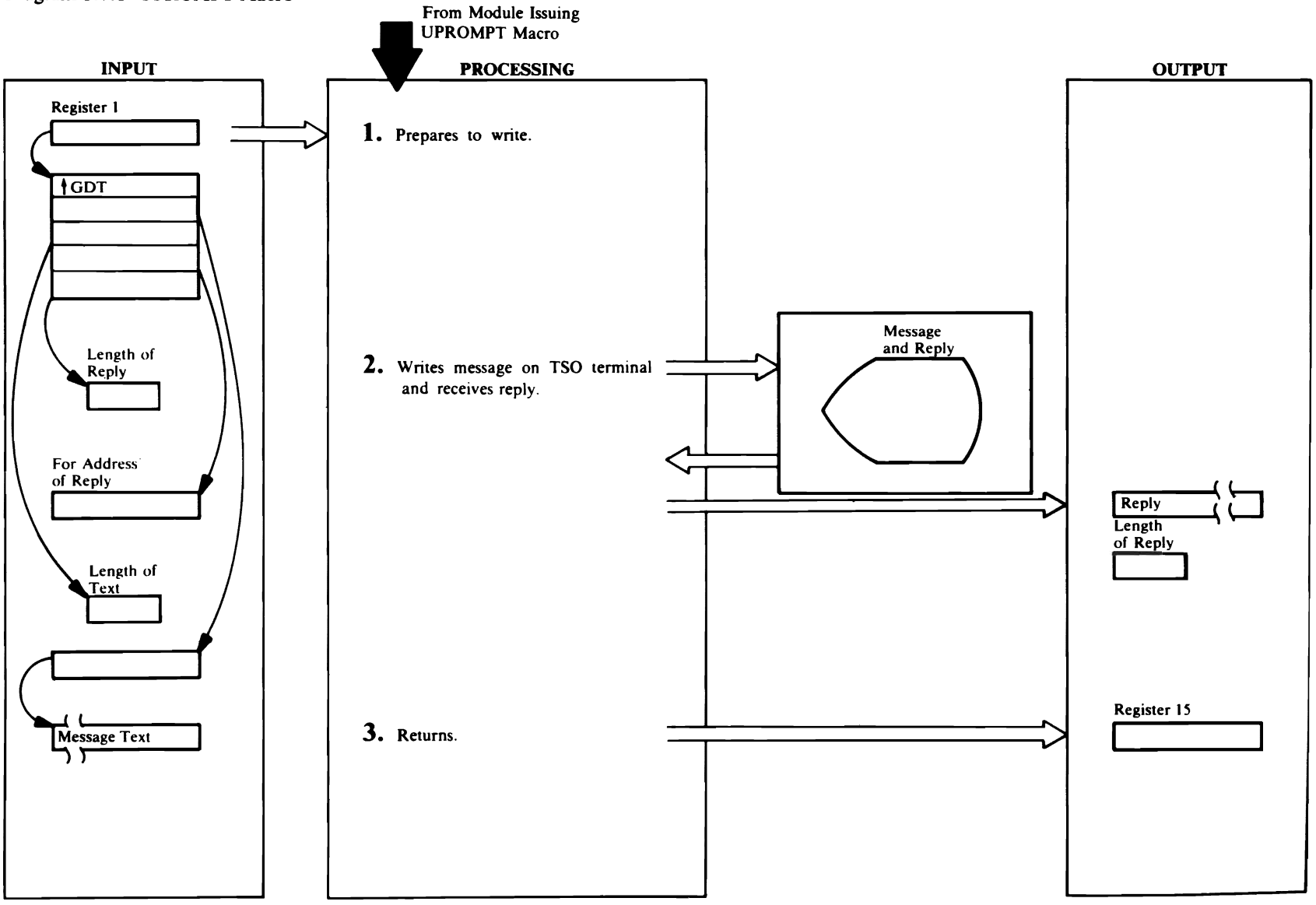
Diagram 5.6.2 USAVERC Macro



Extended Description for Diagram 5.6.2

1. Unlike most Umacros, USAVERC generates inline code that performs the function rather than generating a branch to another module. The code copies the contents of register 15 which must be named RTNREG to a halfword named TESTRC. Control continues with the next instruction in the program.

Diagram 5.7.1 UPROMPT Macro



Extended Description for Diagram 5.7.1

Module: IDCSA02

Procedure: IDCSA02

1. UPROMPT tests GDTECT to be sure Access Method Services is invoked interactively with TSO. If it is not, UPROMPT writes a message and control goes to step 3. UPROMPT checks the length of the message. If the message is longer than 72 characters or less than one character, a UABORT macro is issued. UPROMPT sets up a work area. The first two bytes contain the length of the message plus 4 for the first word. The second two bytes contain zero. The message follows the first four bytes.

Module: IDCSA02

Procedure: IDCSA02

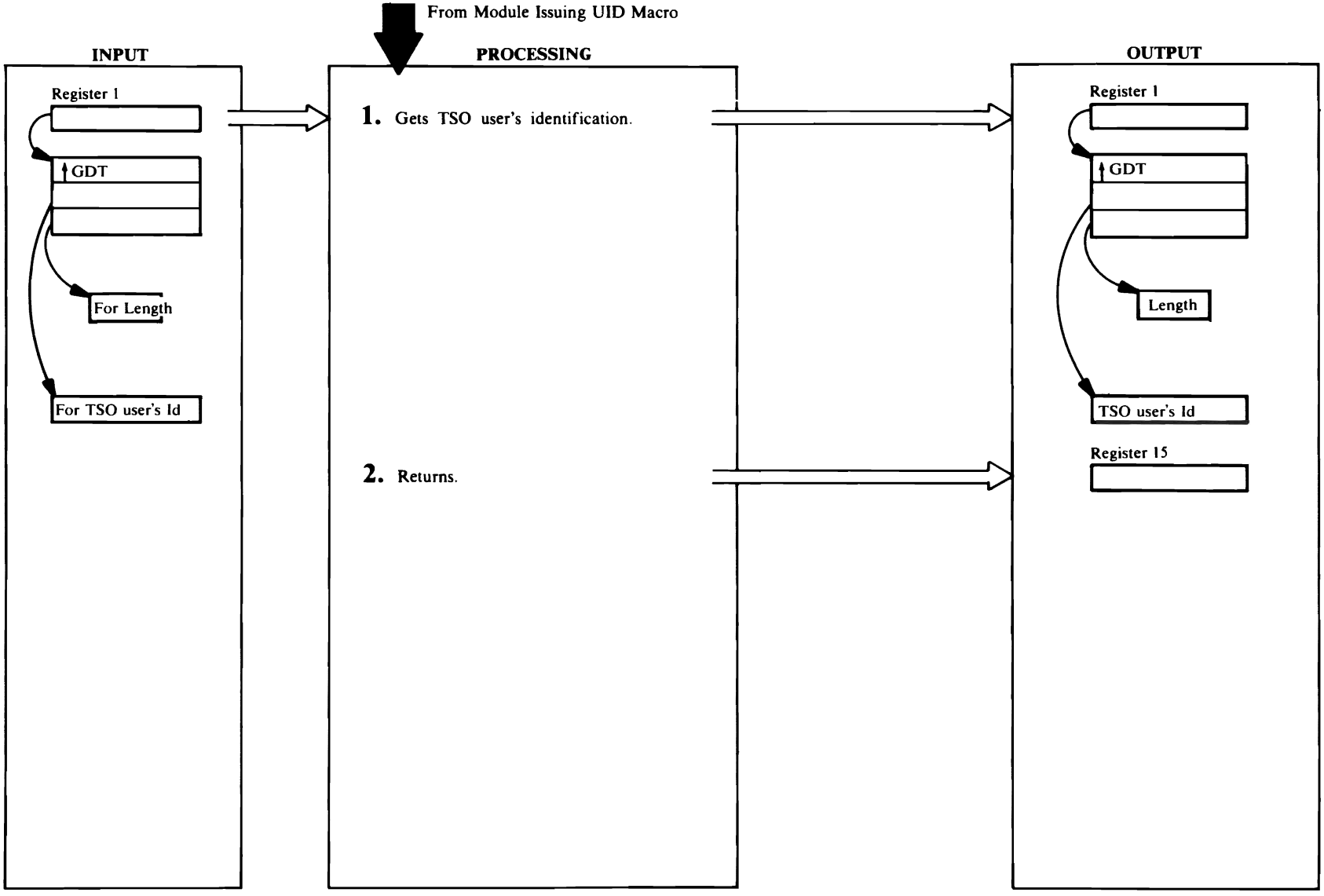
2. UPROMPT issues a PUTGET macro to write the message on the TSO terminal and receive a reply. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor.* If the return code from PUTGET is zero, UPROMPT puts the length of the reply and the text of the reply in the area provided by the module that issued the UPROMPT macro. If the return code from PUTGET indicates a no-prompt situation is in effect, no message is issued and a return code of 8 is put in register 15. For all other non-zero PUTGET return codes, UPROMPT writes a message.

Module: IDCSA02

Procedure: IDCSA02

3. UPROMPT puts a return code in register 15 and returns control to the module that issued the UPROMPT macro.

Diagram 5.7.2 UID Macro



Extended Description for Diagram 5.7.2

Module: IDCSA02

Procedure: IDCSA02

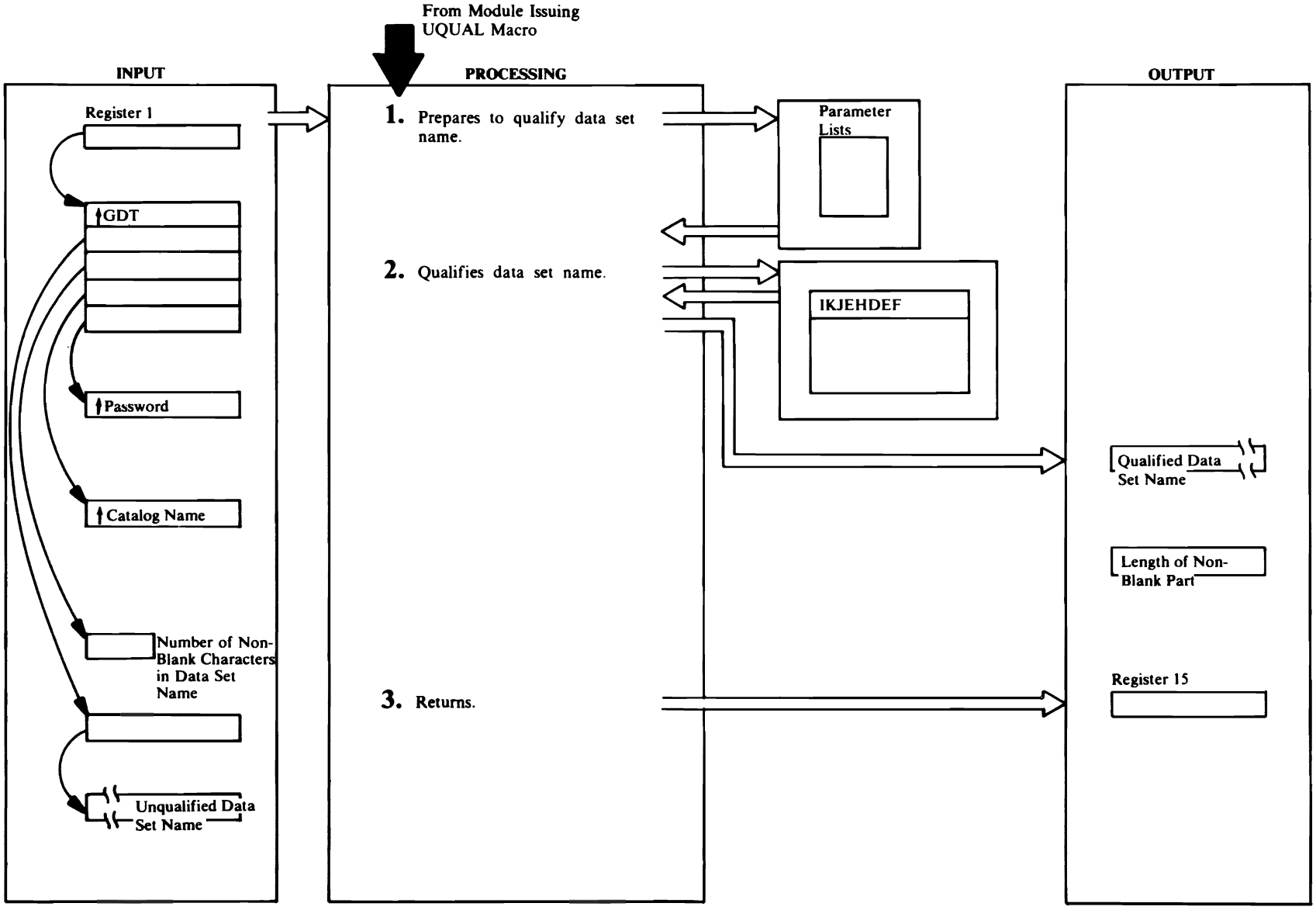
1. UID tests GDTECT to be sure Access Method Services is invoked interactively with TSO. If it is not, UID sets the TSO user's identification to blanks, the length of the TSO user's identification to zero, and puts a 4 in register 15. If Access Method Services is invoked interactively with TSO, UID moves the TSO user's identification from the TSO User Profile Table to the area provided by the module that issued the UID macro. UID puts the number of non-blank characters in the TSO user's identification in the field provided by the module that issued the UID macro. GDTUPT contains the address of the TSO User Profile Table.

Module: IDCSA02

Procedure: IDCSA02

2. UID puts a return code in register 15 and returns control to the module that issued the UID macro.

Diagram 5.7.3 UQUAL Macro



Extended Description for Diagram 5.7.3

Module: IDCSA02

Procedure: IDCSA02

1. UQUAL tests GDTECT to be sure Access Method Services is invoked interactively with TSO. If it is not, NOSUPP writes a message and control goes to step 3. If it is, processing continues as follows. UQUAL builds a TSO Default Parameter Block, DFPB. The DFPB contains the address of the Protected Step Control Block from GDTPCB, the address of the data set name to be qualified, the address of the catalog data set name, and the address of a password. UQUAL also builds a TSO Default Parameter List, DFPL, containing the address of the TSO User Profile Table from GDTUPT, the adddress of the Environment Control Table from GDTECT, the address of an Event Control Block, and the address of the TSO Default Parameter Block just built. See *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic* logic for more information on the DFPB and DFPL.

Module: IDCSA02

Procedure: IDCSA02

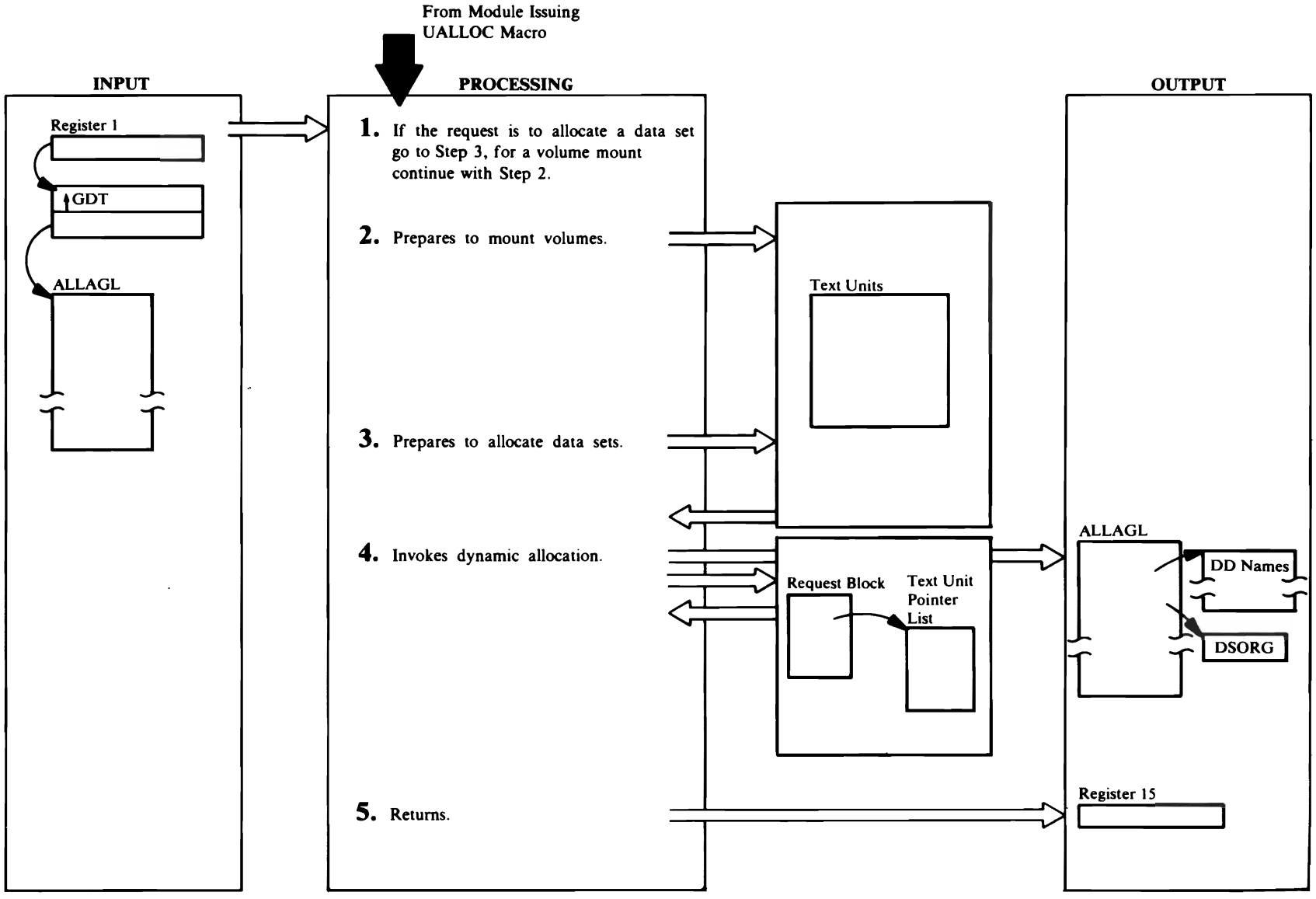
2. UQUAL issues a ULINK macro to load and give control to TSO module IKJEHDEF at entry point IKJDFLT. However, if **VS2.03.807** is installed, UQUAL issues the CALLTSSR macro rather than ULINK. IKJEHDEF qualifies the data set name. See *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic* for more information on IKJEHDEF. If the return code from IKJEHDEF is zero, UQUAL puts the length of the qualified data set name and the qualified data set name in the area provided by the module that issued the UQUAL macro. If the return code is non-zero, UQUAL writes a message.

Module: IDCSA02

Procedure: IDCSA02

3. UQUAL puts a return code in register 15 and returns control to the module that issued the UQUAL macro.

Diagram 5.8.1 UALLOC Macro



Extended Description for Diagram 5.8.1

Module: IDCSA02

Procedure: IDCSA02

1. UALLOC checks the ALLAGL to determine if the UALLOC request is to allocate a data set or to mount volumes. If the request is to allocate a data set, control goes to step 3. If the request is to mount volumes, control continues with step 2.

Module: IDCSA02

Procedure: IDCSA02

2. If PRIVATE volume mounting is requested, UALLOC builds the private volume Text Unit. A unit name from ALLAGL is paired with a volume serial number. If the unit name is the last in the list of unit names, all remaining volume serial numbers are associated with it. If a volume count and a unit count are requested, they too are associated with the last unit name. UALLOC builds a Text Unit with the unit name and volume serial number(s) and optionally builds the Text Units for the volume and unit counts. Control goes to step 4.

Module: IDCSA02

Procedure: IDDCSA02

3. UALLOC builds Text Units with the data set name, data set password, and the data set status and disposition. If a specific device type is requested, UALLOC builds the Text Unit for the unit name.

Module: IDCSA02

Procedure: IDCSA02

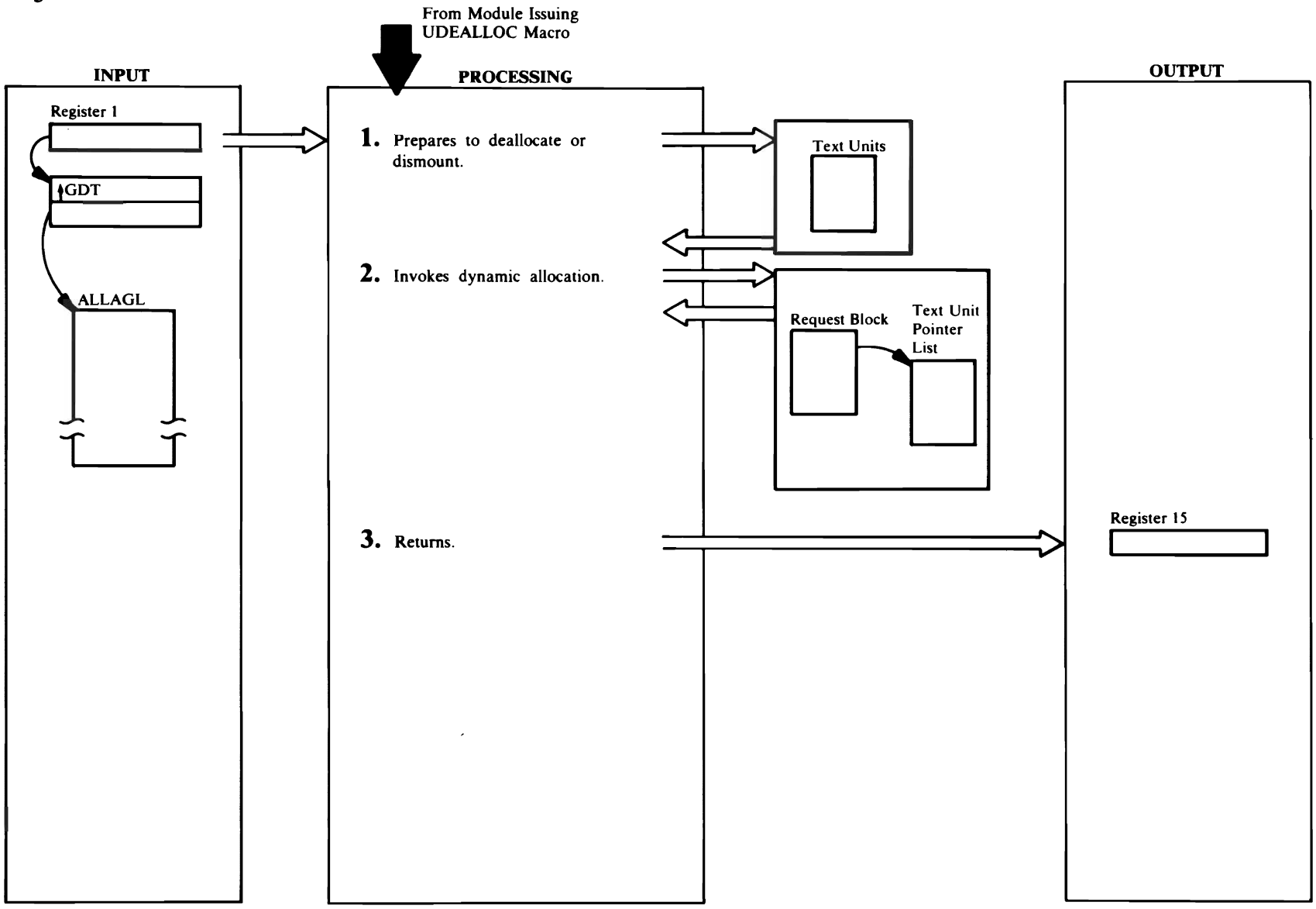
4. UALLOC builds a Text Unit Pointer list containing the addresses of the Text Units. UALLOC creates a Request Block containing the address of the Text Unit Pointer list and issues SVC 99 to invoke dynamic allocation. Refer to *OS/VS2 System Programming Library: Job Management* for more information on dynamic allocation. If the return code is zero, UALLOC saves the returned data set name. If the UALLOC request is for volume mounting and there are more unit names in ALLAGL, control goes to step 2 to prepare to mount the next volume. If the UALLOC request is for volume mounting and there are no more unit names in ALLAGL, UALLOC builds Text Units with the DDname for each unit. UALLOC builds a Text Unit Pointer List and a Request Block, then it issues SVC 99 to get one DDname that describes the concatenated list of volumes. The returned DDname is put in ALLAGL. If the request is for data set allocation, UALLOC puts the data set organization and the returned DDname in ALLAGL.

Module: IDCSA02

Procedure: DYNERR, IDCSA02

5. If the return code from SVC 99 is non-zero, DYNERR writes a message. UALLOC puts a return code in register 15 and returns control to the module that issued the UALLOC macro.

Diagram 5.8.2 UDEALLOC Macro



Extended Description for Diagram 5.8.2

Module: IDCSA02

Procedure: IDCSA02

1. UDEALLOC builds Text Units with information from the ALLAGL. The Text Units contain the data set name or DDname, and the data set disposition from ALLAGL, if specified. The disposition from ALLAGL overrides the current disposition of the data set. If the unallocate option was specified in the ALLAGL, the unallocate option text unit is built. This will unallocate permanently allocated data sets.

Module: IDCSA02

Procedure: IDCSA02, DYNERR

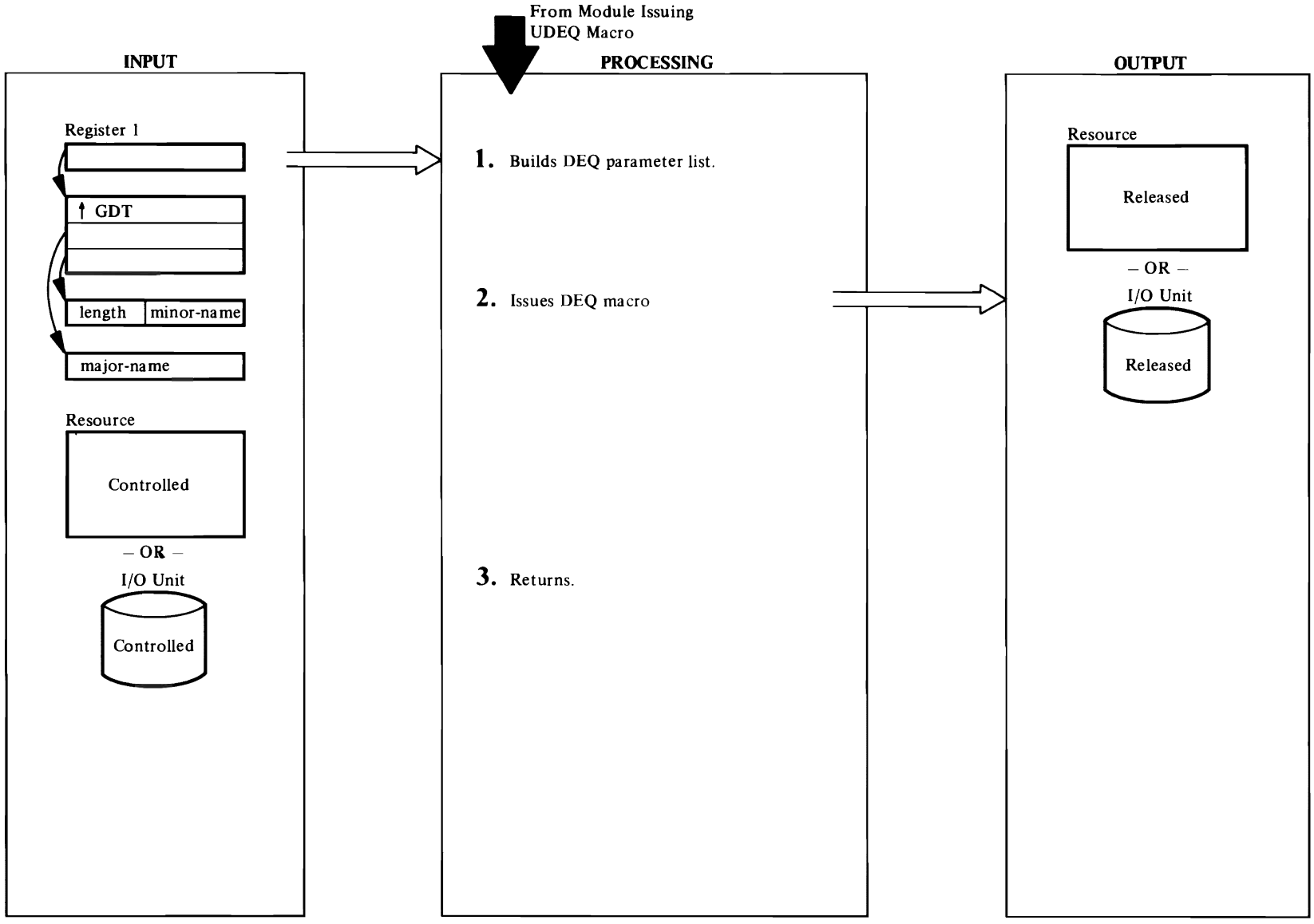
2. UDEALLOC builds a Text Unit Pointer list containing addresses of the Text Units, and a Request Block containing the address of the Text Unit Pointer List. UDEALLOC issues SVC 99 to invoke dynamic allocation. Refer to *OS/VS2 System Programming Library: Job Management* for more information on dynamic allocation. If the return code from SVC 99 is zero, UALLOC checks the information code in the Request Block. If the information code is non-zero, or if the return code from SVC 99 is non-zero, DYNERR writes an error message.

Module: IDCSA02

Procedure: IDCSA02

3. UDEALLOC puts a return code in register 15 and returns control to the module that issued the UDEALLOC macro.

Diagram 5.9.1 System Adapter – UDEQ Macro



Extended Description for Diagram 5.9.1

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 builds the DEQ parameter list and places the minor-name length and the addresses of the parameter list, of the major name, and of the minor name in register variables.

Module: IDCSA08

Procedure: IDCSA08

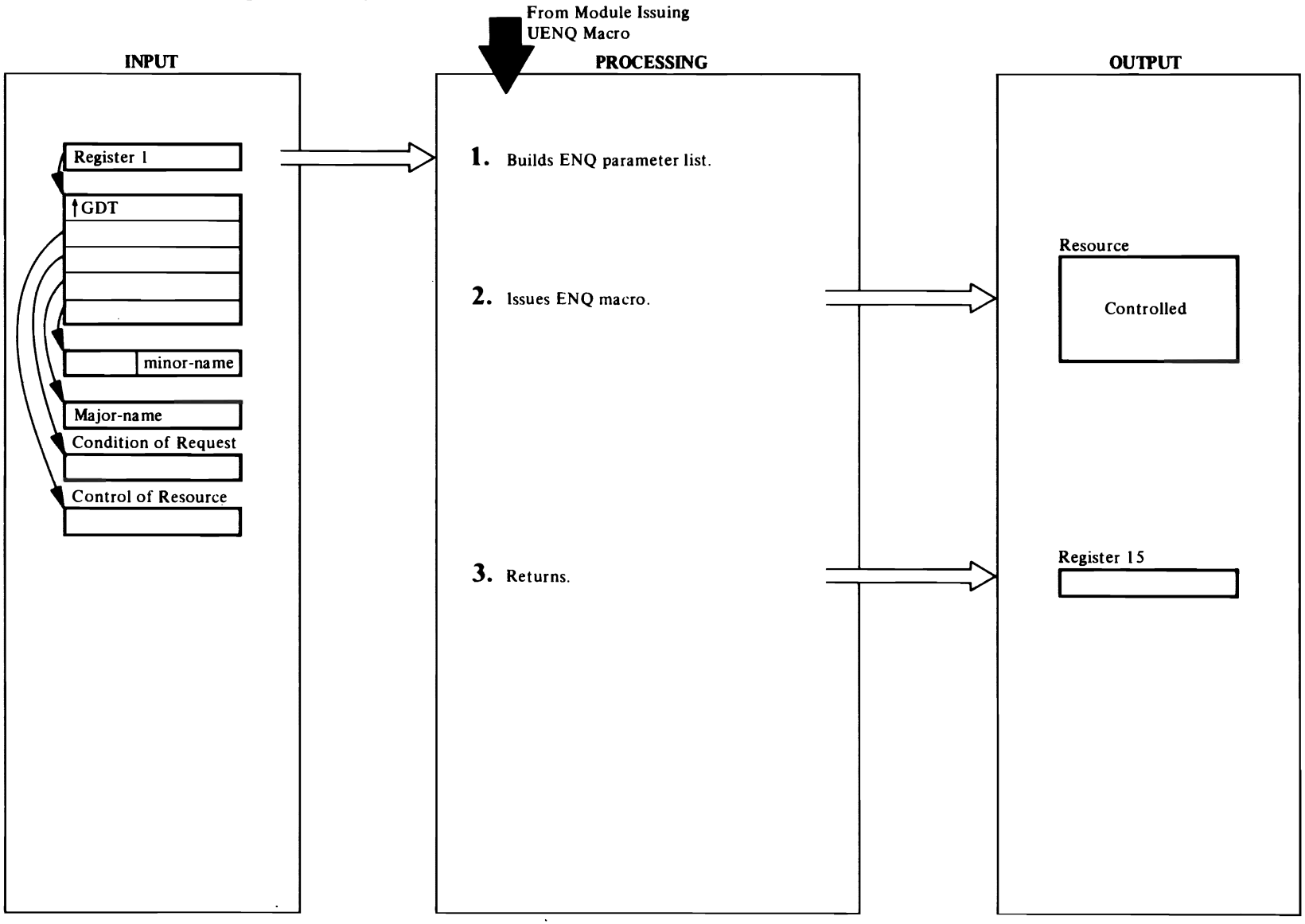
2. IDCSA08 issues a DEQ macro to release control of the resource or I/O unit.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 returns control to the module that issued the UDEQ macro.

Diagram 5.9.2 System Adapter – UENQ Macro



Extended Description for Diagram 5.9.2

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 builds the appropriate ENQ parameter list, depending on the second (SHR or EXCL) and third (NOWAIT or WAIT) arguments supplied by the caller. It places the minor-name length and the addresses of the parameter list, of the major name, and of the minor name in register variables.

Module: IDCSA08

Procedure: IDCSA08

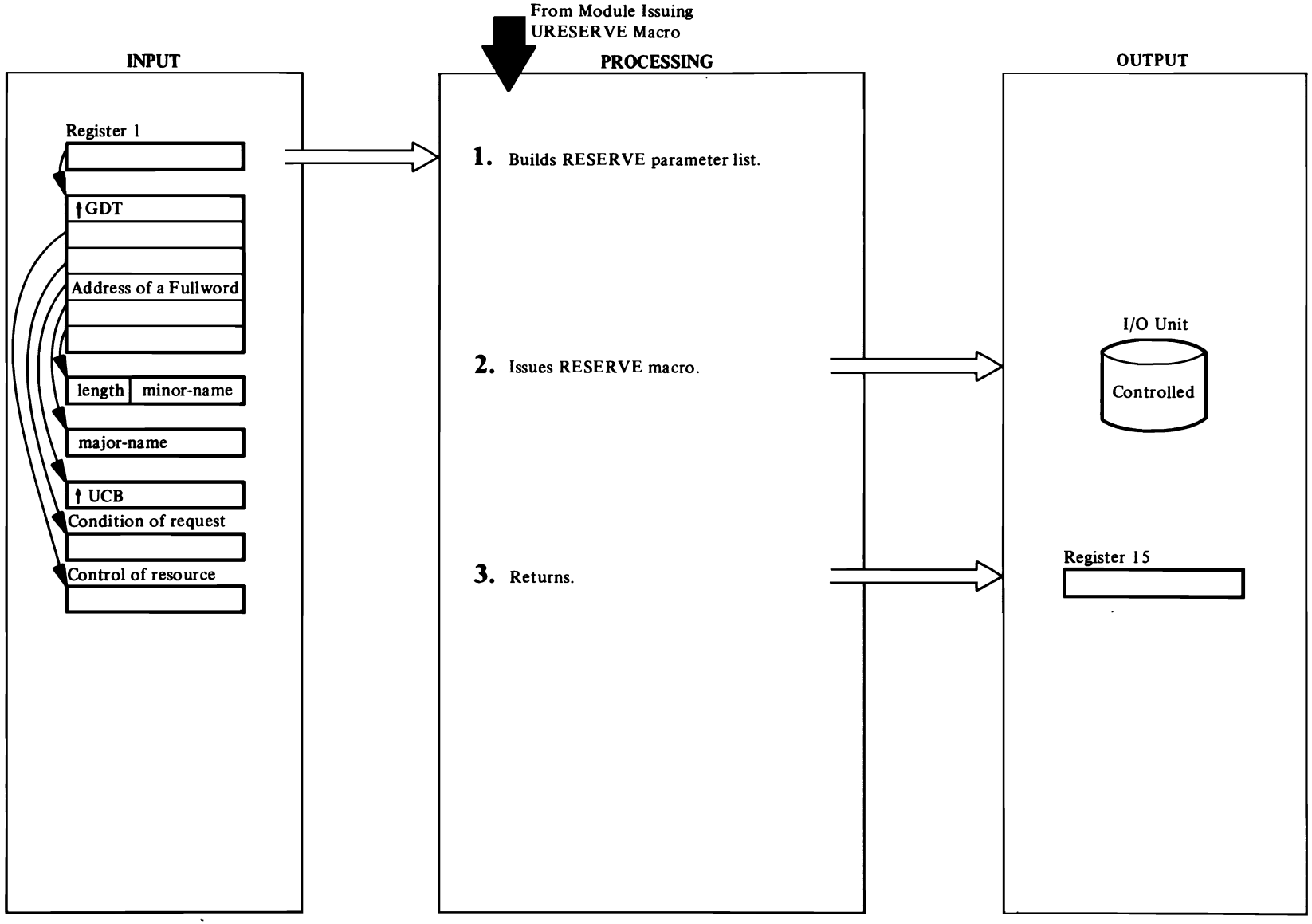
2. IDCSA08 issues an ENQ macro to get control of the resource requested. If the return code in register 15 is zero or if the return code at the address in register 15 is zero or 8, the caller's return code is set to zero. Otherwise the caller's return code is set to 4.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 puts the return code in register 15 and returns control to the module that issued the UENQ macro.

Diagram 5.9.3 System Adapter – URESERVE Macro



Extended Description for Diagram 5.9.3

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 builds the appropriate RESERVE parameter list, depending on the second (SHR or EXCL) and third (NOWAIT or WAIT) arguments supplied by the caller. It places the minor-name length and the addresses of the parameter list, of the major name, of the minor name, and of the UCB pointer in register variables.

Module: IDCSA08

Procedure: IDCSA08

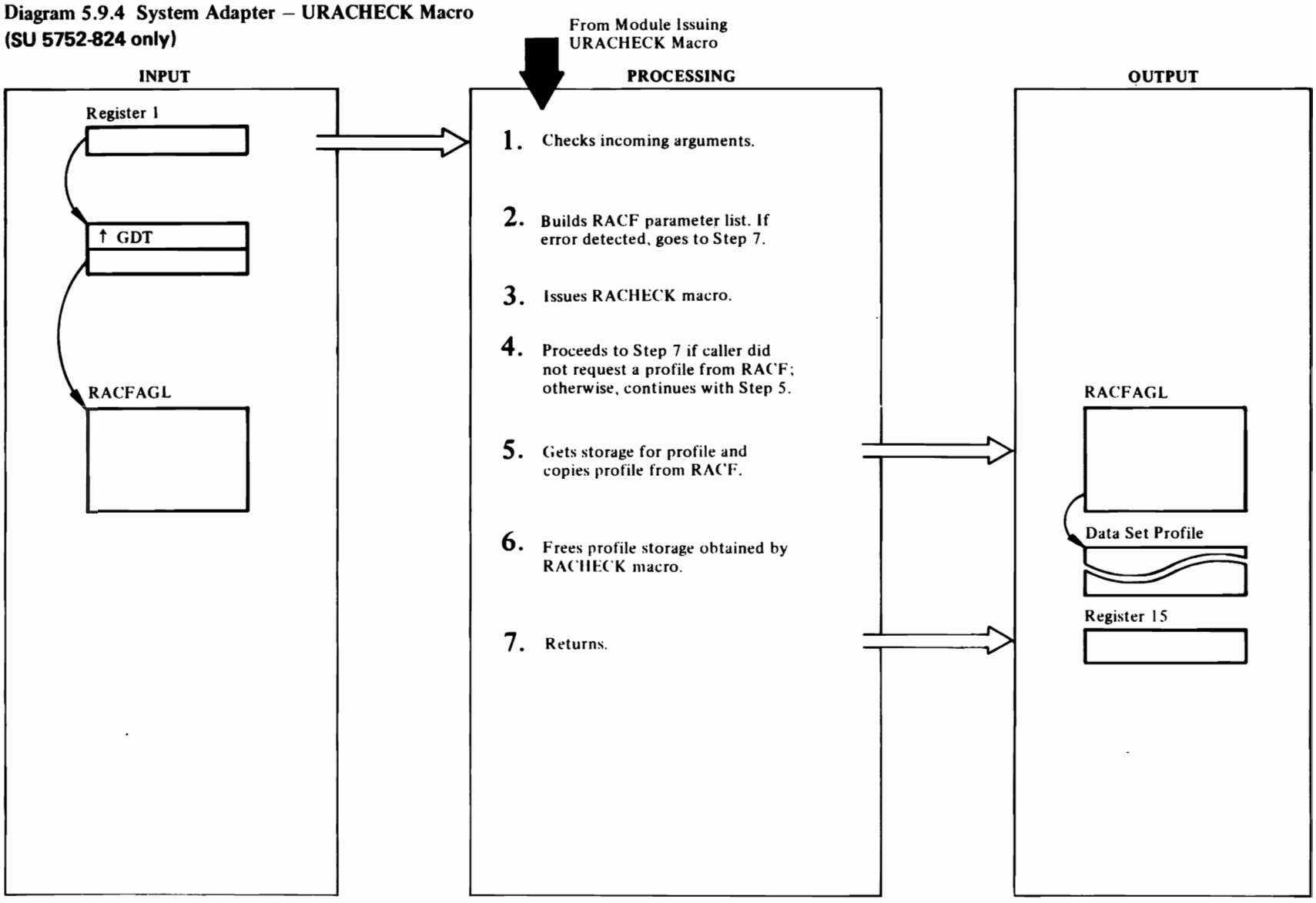
2. IDCSA08 issues a RESERVE macro to reserve the resource requested. If the return code in register 15 is zero or if the return code at the address in register 15 is zero or 8, the caller's return code is set to zero. Otherwise the caller's return code is set to 4.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 puts the return code in register 15 and returns control to the module that issued the URESERVE macro.

Diagram 5.9.4 System Adapter – URACHECK Macro (SU 5752-824 only)



Extended Description for Diagram 5.9.4

(SU 5752-824 only)

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 first checks for invalid incoming arguments. Unless there are exactly two arguments, IDCSA08 issues a UABCRT macro with a code of 40.

Module: IDCSA08

Procedure: IDCSA08

2. IDCSA08 builds the appropriate RACHECK parameter list using the information in the second argument. It places the addresses of classname, and volume serial and the address of either a data set name or a profile of the data set in the RACHECK parameter list. It also sets the appropriate flags in accordance with the caller's request. If any of the flags or addresses passed by the caller are invalid, IDCSA08 issues an error message indicating an invalid URACHECK Parameter List, sets the caller's return code to 12, and continues at STEP 7.

Module: IDCSA08

Procedure: IDCSA08

2. IDCSA08 issues the RACHECK macro to obtain authorization. If the return code is zero, RACF has given the caller authorization to continue. The caller's return code is set to zero. If the return code is not zero, IDCSA08 issues an appropriate message depending on the return code from the RACHECK macro:

   RC=4—No RACF profile on that data set

   RC=8—Invalid RACF authorization.

   IDCSA08 sets the caller's return code to 4 or 8 respectively.

Module: IDCSA08

Procedure: IDCSA08

4. If the caller requested a profile from RACF, IDCSA08 will supply the profile and an address of the profile for either a return code of 0 or 8. If the caller did not request the profile, IDCSA08 continues at Step 7.

Module: IDCSA08

Procedure: IDCSA08

5. IDCSA08 issues a UGSPACE macro to obtain storage for the profile. If the return code from UGSPACE is not zero, IDCSA08 issues an error message indicating that storage could not be obtained. It sets the caller's return code to 8, and continues at Step 7. If the return code from UGSPACE is 0, IDCSA08 then copies the profile provided by RACF. The caller is required to free this space when he no longer needs it.

Module: IDCSA08

Procedure: IDCSA08

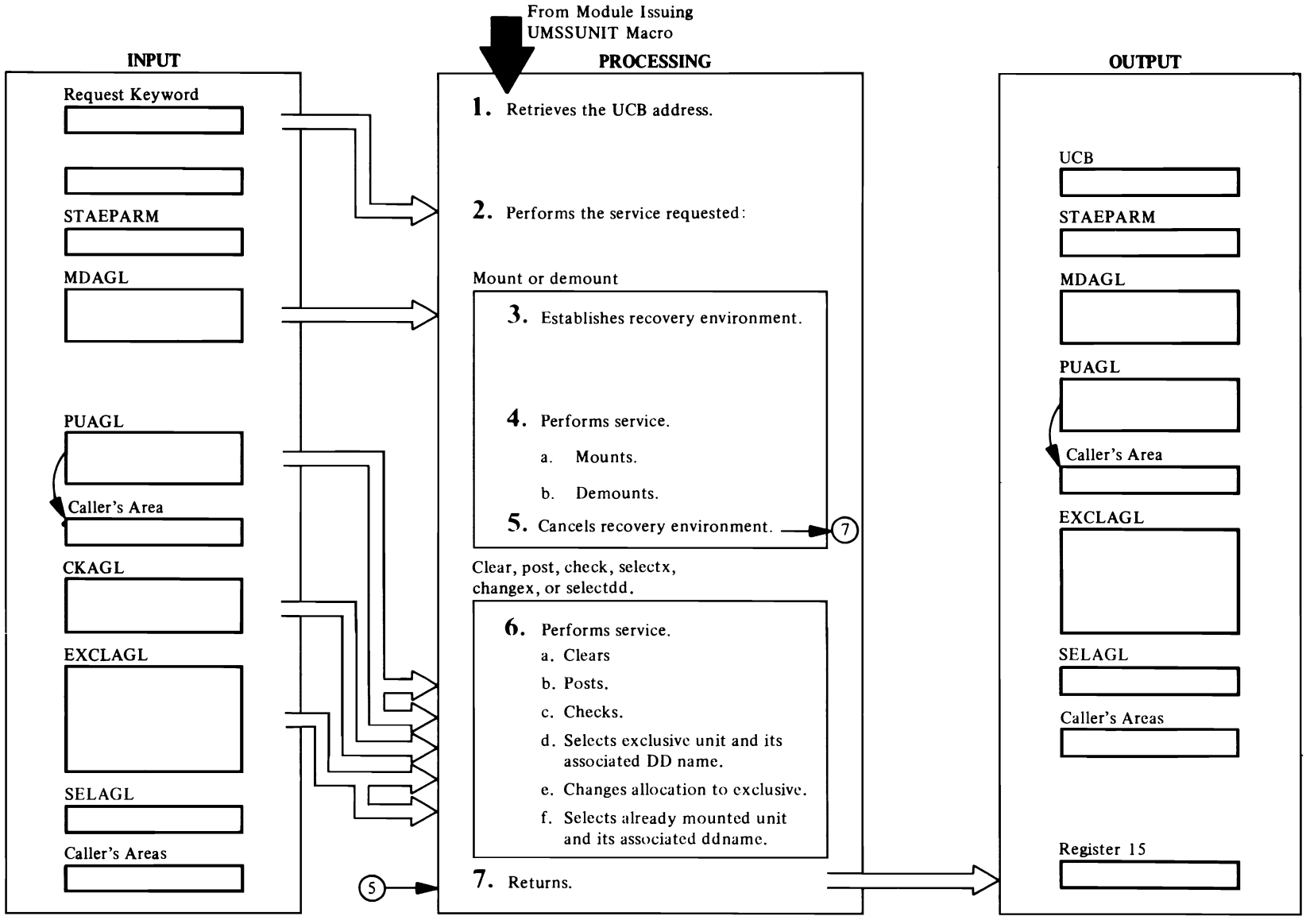
6. IDCSA08 issues a FREEMAIN macro to release the storage obtained for the profile by RACF.

Module: IDCSA08

Procedure: IDCSA08

7. IDCSA08 puts the return code in register 15 and returns control to the module that issued the URACHECK macro.

Diagram 5.10.1 System Adapter – UMSSUNIT Macro



Extended Description for Diagram 5.10.1

Module: IDCSA06

Procedure: IDCSA06

1. IDCSA06 issues a USYSINFO macro if the caller did not provide the UCB address. IDCSA06 puts the UCB address in the caller's area.

Module: IDCSA06

Procedure: IDCSA06

2. IDCSA06 checks the request keyword to determine the type of service to perform: 'MOUNT' indicates a mount request; 'DEMOUNT' indicates a demount request; 'POSTUCB' indicates a UCB post request; 'CHECK' indicates a UCB check request.

Module: IDCSA06

Procedure: MDSETUP

3. MDSETUP issues a USTAE macro to set up a recovery environment.

Module: IDCSA06

Procedure: MOUNTCTL, MOUNTVOL, ENQDEQ, ISSUEMNT, RDLABEL, DEMNTVOL, ISSUEDMT

4. The requested service is performed.

   a. MOUNTCTL checks the UCB to see whether a volume is already mounted. If the requested volume is mounted, processing continues at step 7. If a different volume is mounted, that volume is demounted (see step b).

      If requested, the volume serial number is cleared from the specified UCB of a real volume (see step 6.a). Then MOUNTCTL sets recovery information in STAEPARM to indicate a UCB is cleared but the volume is still mounted.

      ENQDEQ issues an ENQ macro to enqueue on the volume serial number of the volume to be mounted. The major name is SYSZVOLS; the minor name is the volume serial number. ISSUEMNT issues a MOUNT request to MSC (Mass Storage Control) by way of the USSC macro. ISSUEMNT issues an ACQUIRE request to MSC by way of the USSC macro to ensure the mount is complete.

      Unless the volume hasn't been initialized, RDLABEL issues a UEXCP macro to open the VTOC and read the label to retrieve the TTR to the VTOC.

      The UCB for the mounted volume is posted (see step 6.b).

      If any of these actions fails, the volume is demounted (see step b), the UCB cleared (in step b) is posted (see step 6.b), and the volume is dequeued with a DEQ macro. Processing continues at step 5.

   b. DEMNTVOL checks the UCB to ensure that the volume to be demounted is mounted. If it isn't, processing continues at step 5. If the wait option is indicated, ISSUEDMT issues a DEMOUNT request to MSC by way of the USSC macro with the delayed response option.

      The UCB for the demounted volume is cleared even if the demount failed (see step 6.a).

      If requested, ENQDEQ dequeues the volume serial number with a DEQ macro. If requested, the volume serial number is put back into the specified UCB of a real volume (see step 6.b). The volume serial number was removed during the prior mount request.

Module: IDCSA06

Procedure: MDSETUP

5. MDSETUP issues a USTAE macro to cancel the recovery environment unless the volume serial number was removed from a UCB while the volume was still mounted. Processing continues at step 7.

Module: IDCSA06

Procedure: FINDEXCL, SETEXCL, UCBPOST, UCBCHECK, GETEXL, SCANTIOT

6. The requested service is performed.

   a. UCBPOST sets up SV82LIST and issues SVC 82 to clear the specified UCB. SVC 82 removes the volume serial number and VTOC TTR from the UCB and marks the UCB "not ready." The volume serial number and VTOC TTR are returned to the caller's area.

   b. UCBPOST sets up SV82LIST and issues SVC 82 to post the volume serial number and VTOC TTR in the UCB and marks the UCB "ready."

   c. UCBCHECK checks the UCB to determine the status of the unit. It is tested for 3330 direct access, virtual or real device type, or nonsharable status, as requested by the caller in CKAGL. The type of the unit is returned in the caller's area if requested.

d. FINDEXCL issues the UENQ macro to enqueue on the task I/O table (TIOT) with a major name of SYSZTIOT. SCANTIOT scans the TIOT DD entries to find a UCB allocated to this job step for the device on which the volume specified by the caller is mounted. SCANTIOT determines whether a nonsharable unit can be used for the volume if the volume is not mounted but is allocated exclusively to this job step. If the volume specified by the caller is mounted and the UCB is marked nonshareable, SCANTIOT returns the UCB address and DD name to the caller. If the volume is not mounted, FINDEXCL issues an UENQ macro with major name of SYSZVOLS to test whether the volume is allocated exclusively to the job step. If it is FINDEXCL returns the UCB address and DD name of a nonshareable unit found by SCANTIOT. FINDEXCL then issues the UDEQ macro to dequeue on the TIOT.

e. GETEXCL checks whether the UCB indicates the volume is permanently resident or reserved and, if it is, indicates and error. GETEXCL checks whether the unit is already nonshareable and if it is returns to the caller. SETEXCL issues an UENQ macro to the initiator's TCB with major name SYSZVOLS to change the enqueue on the volume mounted on the unit to exclusive. If the UENQ is successful, SETEXCL issues the MODESET macro to enter key 0 and marks the UCB to indicate the unit is nonshareable. SETEXCL then issues the MODESET macro to return to problem-program state.

f. SELECTD issues the UENQ macro to enqueue on the Task I/O Table (TIOT) with a major name of SYSZTIOT. SELECTD next checks to see if a DSAB chain exists; if one does not, the flag VOLFOUND, is set off. If a DSAB chain does exist, the SELESCAN procedure is called to scan the TIOT to find the caller's volume serial number in a UCB. If found, the VOLFOUND flag is set on and the UCB address and the *ddname* are placed in the addressed areas provided by the caller in the SELAGL. SELESCAN then returns control to SELECTD.

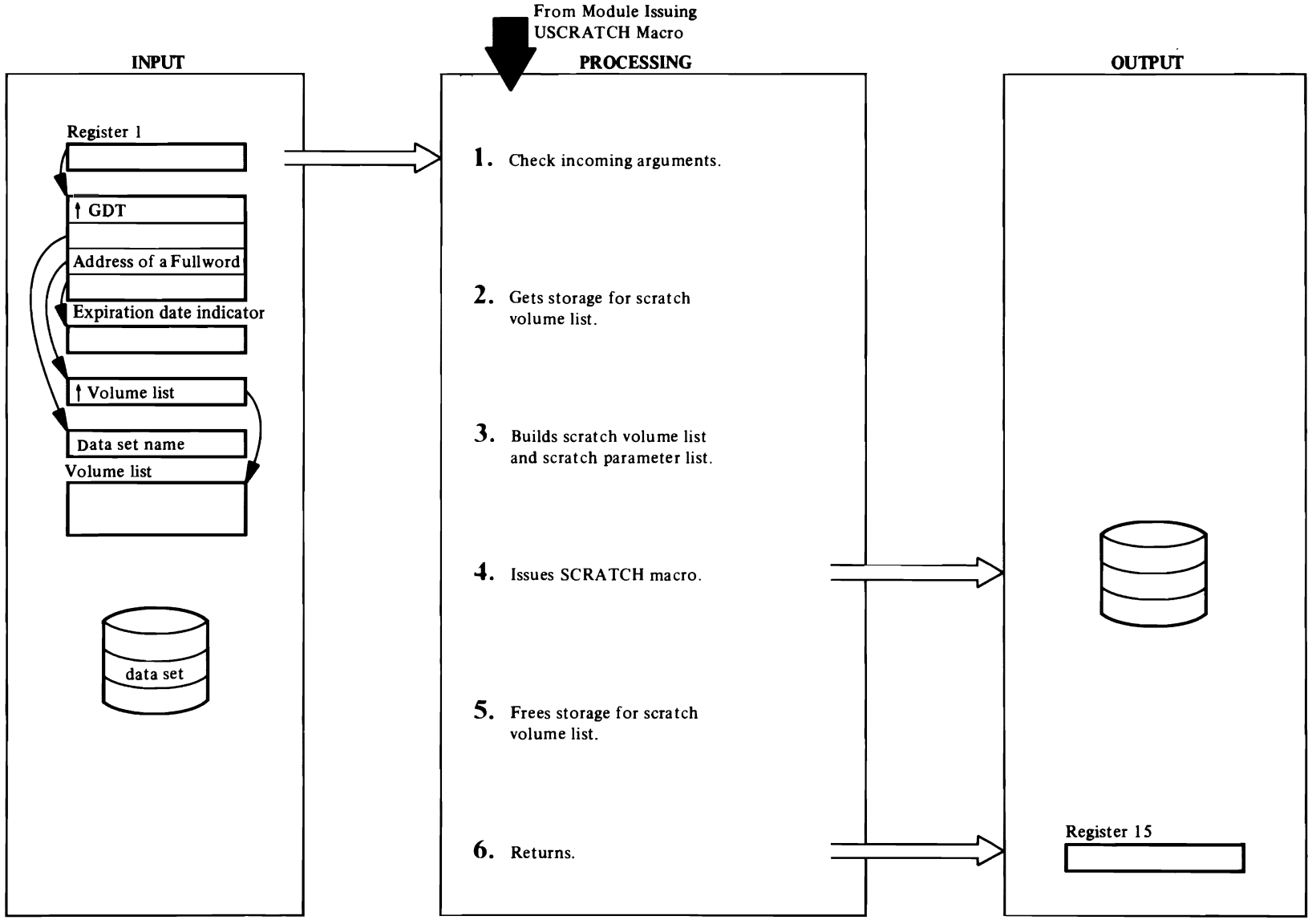
   SELECTD checks VOLFOUND and, if the flag is off, a return code of 4 is set indicating the unit was not mounted. Also, the fields addressed by the SELAGL fields, SELUCBP and SELDDNP are cleared. If VOLFOUND is on, the return code is set to zero, indicating a UCB and *ddname* were found and that the volume is mounted. A UDEQ macro is issued to dequeue the TIOT.

Module: IDCSA06

Procedure: IDCSA06

7. If any errors occurred, a message is written with a UPRINT macro. IDCSA06 returns to the caller with a return code in register 15.

Diagram 5.10.2 System Adapter – USCRATCH Macro



Extended Description for Diagram 5.10.2

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 first checks for invalid incoming arguments. If there aren't exactly three or four arguments, or if the fourth argument isn't OVERRIDE, IDCSA08 issues a UABORT macro with code=40.

Module: IDCSA08

Procedure: IDCSA08

2. If there is more than one volume on the scratch-volume list, IDCSA08 issues a UGSPACE macro to obtain storage for the scratch-volume list. If the return code from UGSPACE is not zero, IDCSA08 issues an error message indicating that storage could not be obtained, sets the caller's return code to 8, and continues at step 6. If there is just one volume on the scratch-volume list, IDCSA08 uses a storage area in its own automatic storage for the scratch-volume list.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 builds the scratch-volume list using the information in the third argument. IDCSA08 then builds the scratch parameter list using the appropriate form of the CAMLST macro expansion. It places the address of the scratch-volume list and the address of the data-set name in the scratch parameter list.

Module: IDCSA08

Procedure: IDCSA08

4. IDCSA08 issues a SCRATCH macro to delete the data set. If the return code is zero, or the scratch status code returned a code of one from the SCRATCH macro, the caller's return code is set to zero and processing continues at step 5. A scratch status code of one means the data set is not on the indicated volume. IDCSA08 issues a message indicating that the data set was not scratched if the return code is not zero or if the scratch status code is not one. IDCSA08 issues a second-level message, depending on the return code and scratch status code returned from the SCRATCH macro:

   RC=4—no volumes containing any part of the data set were mounted.

   RC=8:

      Code=2—password verification failed for the indicated data set.

      Code=3—the data set's date has not expired.

      Code=4—a permanent I/O error occurred.

      Code=5 or 6—the indicated volume could not be mounted.

      Code=7—the specified data set was in use.

      Code=8 (**VS2.03.807 only**)—user does not have the proper RACF authorization to scratch the data set.

   RC=12—the scratch-volume list was invalid—IDCSA08 issues a UABORT macro with code=40.

   IDCSA08 sets the caller's return code to 4.

Module: IDCSA08

Procedure: IDCSA08

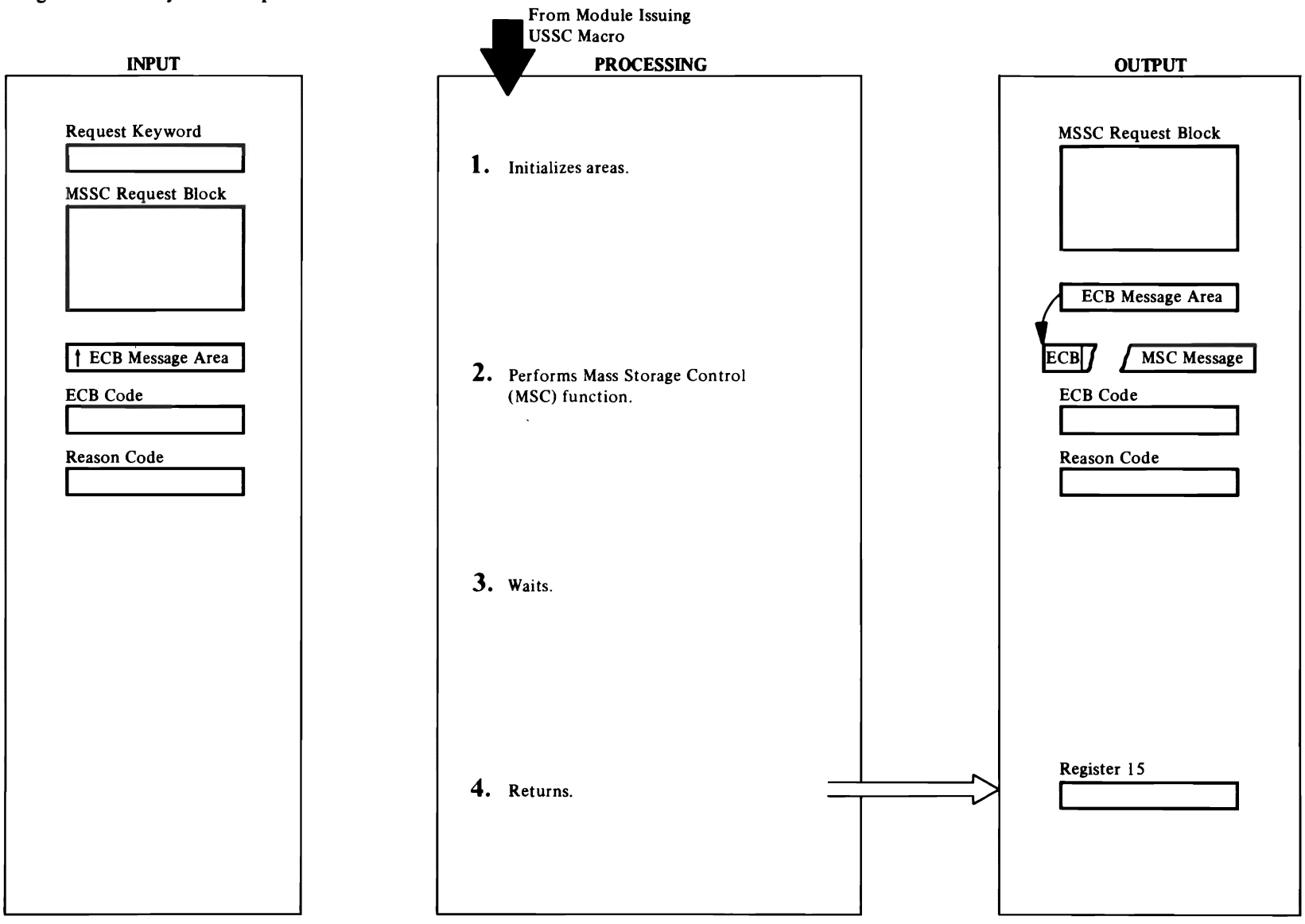
5. If storage was obtained for the scratch-volume list, IDCSA08 releases the storage by issuing a UFSPACE macro.

Module: IDCSA08

Procedure: IDCSA08

6. IDCSA08 puts the return code in register 15 and returns control to the module that issued the USCRATCH macro.

Diagram 5.10.3 System Adapter – USSC Macro



Extended Description for Diagram 5.10.3

Module: IDCSA09

Procedure: IDCSASS CHECKARG

1. IDCSA09 initializes to zero the input areas: reason code, ECB code, and pointer to the ECB message area. CHECKARG checks whether the number of arguments is correct for the request type. If the ECB code argument is present, IDCSASS issues a UGSPACE macro to obtain storage for the ECB and for the MSC (Mass Storage Control) message area.

Module: IDCSA09

Procedure: ISSUEMAC

2. ISSUEMAC issues an MSSC (Mass Storage System Communicator) macro (SVC 126) to do the following according to the request keywords:

   'ACQUIRE' acquires extents on a staging drive.
   'MOUNT' mounts a mass storage volume.
   'DEMOUNT' demounts a mass storage volume.
   'DEFINE' defines a mass storage volume.
   'MOVE' moves a data cartridge or a mass storage volume.
   'TRACEQ' traces MSC activity.
   'COPYTABL' copies the MSC tables.
   'COPYCRTG' copies a data cartridge.
   'COPYVOL' copies a mass storage volume.
   'VVIC' modifies the Mass Storage Volume Inventory data set.
   'TUNE' changes the MSC tuning values.

   For the 'ACQUIRE,' the caller must have initialized the request code and MSSC request block length in the MSSC request block.

   ISSUEMAC specifies an ECB address in the MSSC request block if the ECB code argument was specified.

Module: IDCSA09

Procedure: CHKCODE

3. CHKCODE issues a WAIT macro if an ECB address was specified in the MSSC request block and either the return code from SVC 126 was 0 or the reason code indicates the MSC function was successful but the Mass Storage Volume Inventory data set was not updated. If the ECB completion code indicates that the function was successful, CHKCODE sets the pointer to the ECB message area in the caller's area. If the ECB completion code indicates a failure, CHKCODE puts the ECB completion code in the caller's MSC message area.

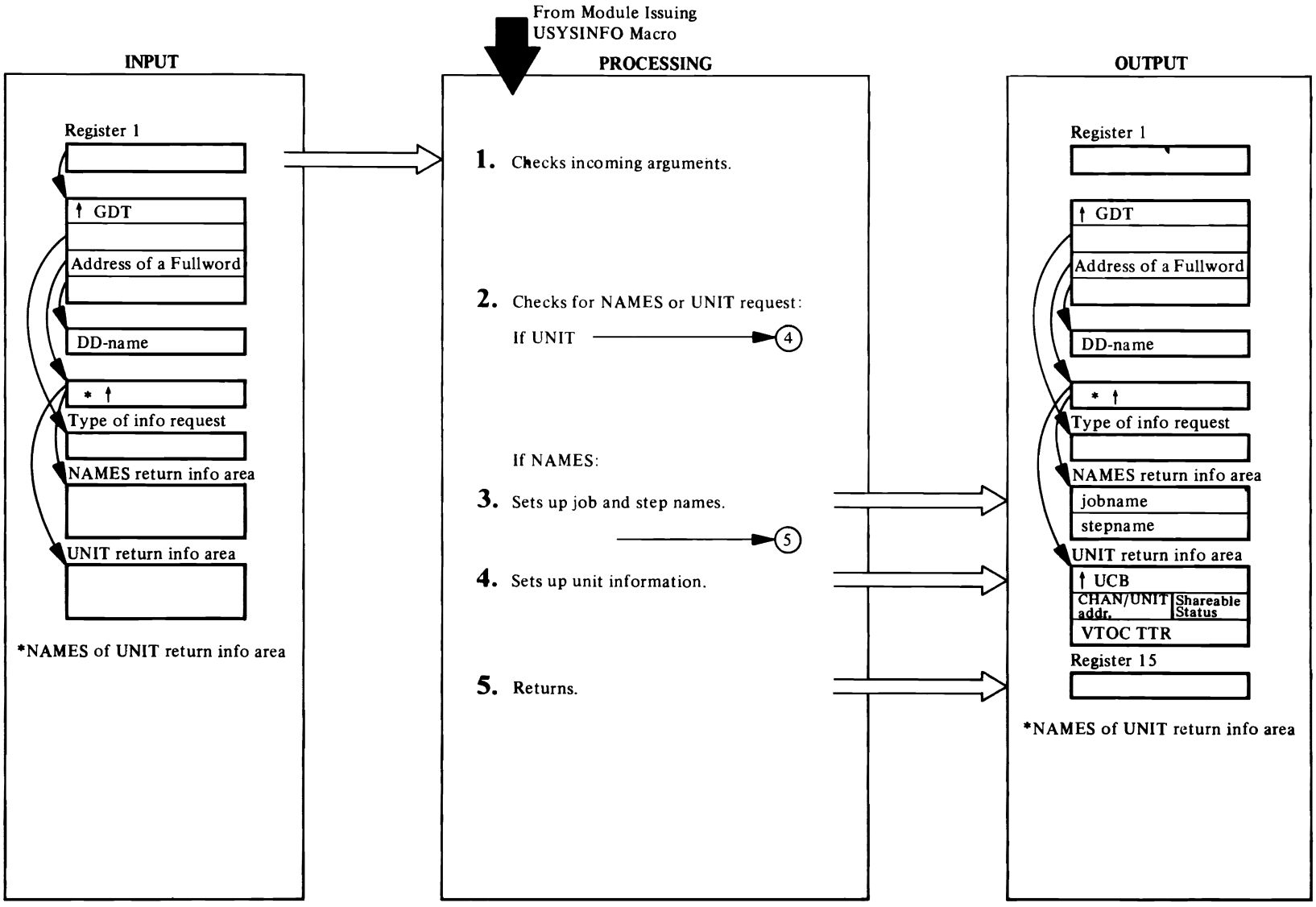
   If the return code from SVC 126 is 4, and the reason code indicates an MSC error, CHKCODE puts the reason code from register 0 into the caller's reason code area. If the reason code indicates an MSVC (Mass Storage Volume Control) error, CHKCODE issues a message that says the function was performed but the Mass Storage Volume Inventory data set was not updated, and the reason code is not put into the caller's reason code area.

Module: IDCSA09

Procedure: IDCSASS

4. IDCSASS issues a UFSPACE macro to free the storage obtained for the ECB and the message area if the area was not used or the caller didn't request it to be returned, and sets the pointer to the caller's ECB message area to zero. If invalid arguments were passed as input, IDCSASS issues a UABORT macro with code=40. Otherwise, IDCSASS returns to the caller with a return code in register 15.

Diagram 5.10.4 System Adapter – USYSINFO Macro



Extended Description for Diagram 5.10.4

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 first checks for invalid incoming arguments. If there aren't exactly three arguments, with NAMES the second argument, or if there aren't exactly four arguments, with UNIT the second argument, IDCSA08 issues a UABORT macro with code=40.

Module: IDCSA08

Procedure: IDCSA08

2. IDCSA08 uses the second argument to determine what information to return to the module that issued USYSINFO. For a NAMES request, IDCSA08 returns the job and step names; for UNIT, IDCSA08 returns specific unit information about the unit identified by way of the DD name supplied as the fourth argument.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 issues an EXTRACT macro to obtain the address of the task I/O table (TIOT). IDCSA08 takes the job and step names from the TIOT and places them in the information area supplied by the caller.

Module: IDCSA08

Procedure: IDCSA08

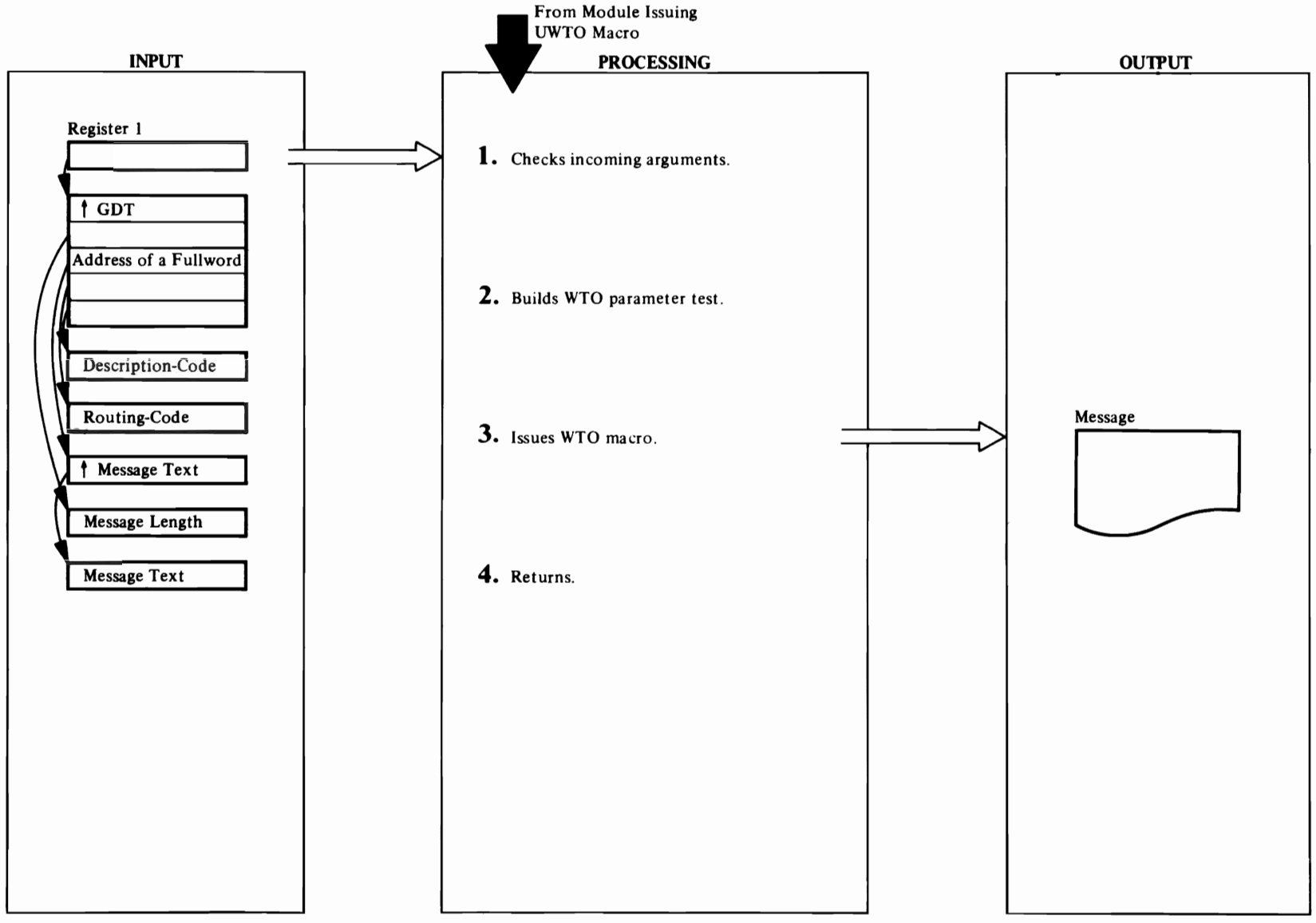
4. IDCSA08 searches the TIOT DD entries to find a DD name that matches the DD name supplied as the fourth argument. If IDCSA08 doesn't find the DD name, it issues an error message and sets the caller's return code to 4. If IDCSA08 finds the DD name, it places the unit control block (UCB) address for that DD entry in the information area supplied by the caller. The UCB address is then used to obtain the binary channel and unit addresses, the indicators about task and system sharing status, and the VTOC TTR pointer. IDCSA08 places this information in the information area supplied by the caller.

Module: IDCSA08

Procedure: IDCSA08

5. IDCSA08 puts the return code in register 15 and returns control to the module that issued the USYSINFO macro.

Diagram 5.10.5 System Adapter – UWTO Macro From Module Issuing



Extended Description for Diagram 5.10.5

Module: IDCSA08

Procedure: IDCSA08

1. IDCSA08 first checks for invalid incoming arguments. If there aren't exactly three or five arguments, IDCSA08 issues a UABORT macro with code=40.

Module: IDCSA08

Procedure: IDCSA08

2. IDCSA08 builds the Write to Operator parameter list. IDCSA08 places the message text and its length in the parameter list. If the routing and descriptor codes are supplied, IDCSA08 places them at the end of the message text and sets the multiple console support (MCS) flags.

Module: IDCSA08

Procedure: IDCSA08

3. IDCSA08 issues a WTO macro.

Module: IDCSA08

Procedure: IDCSA08

4. IDCSA08 returns control to the module that issued the UWTO macro.

I/O Adapter Visual Table of Contents

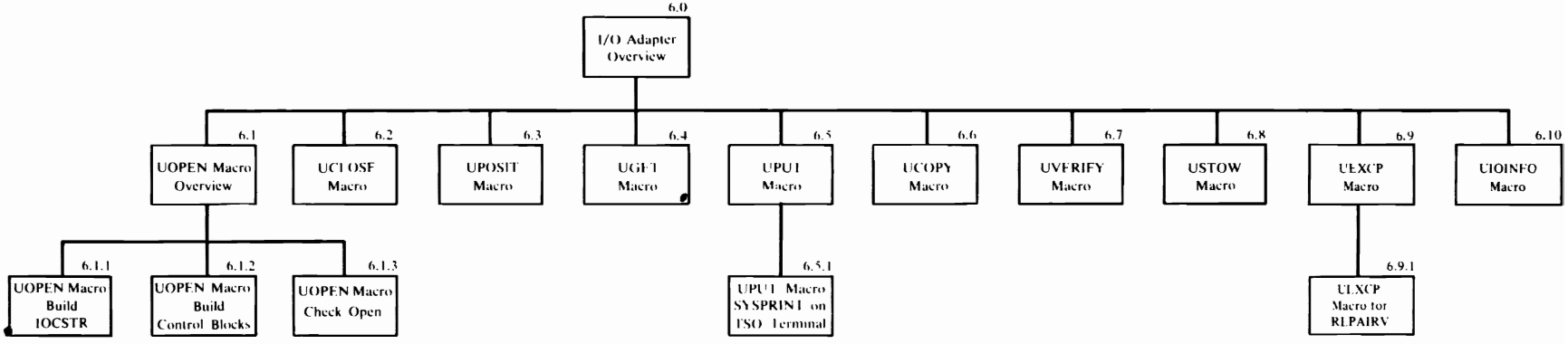
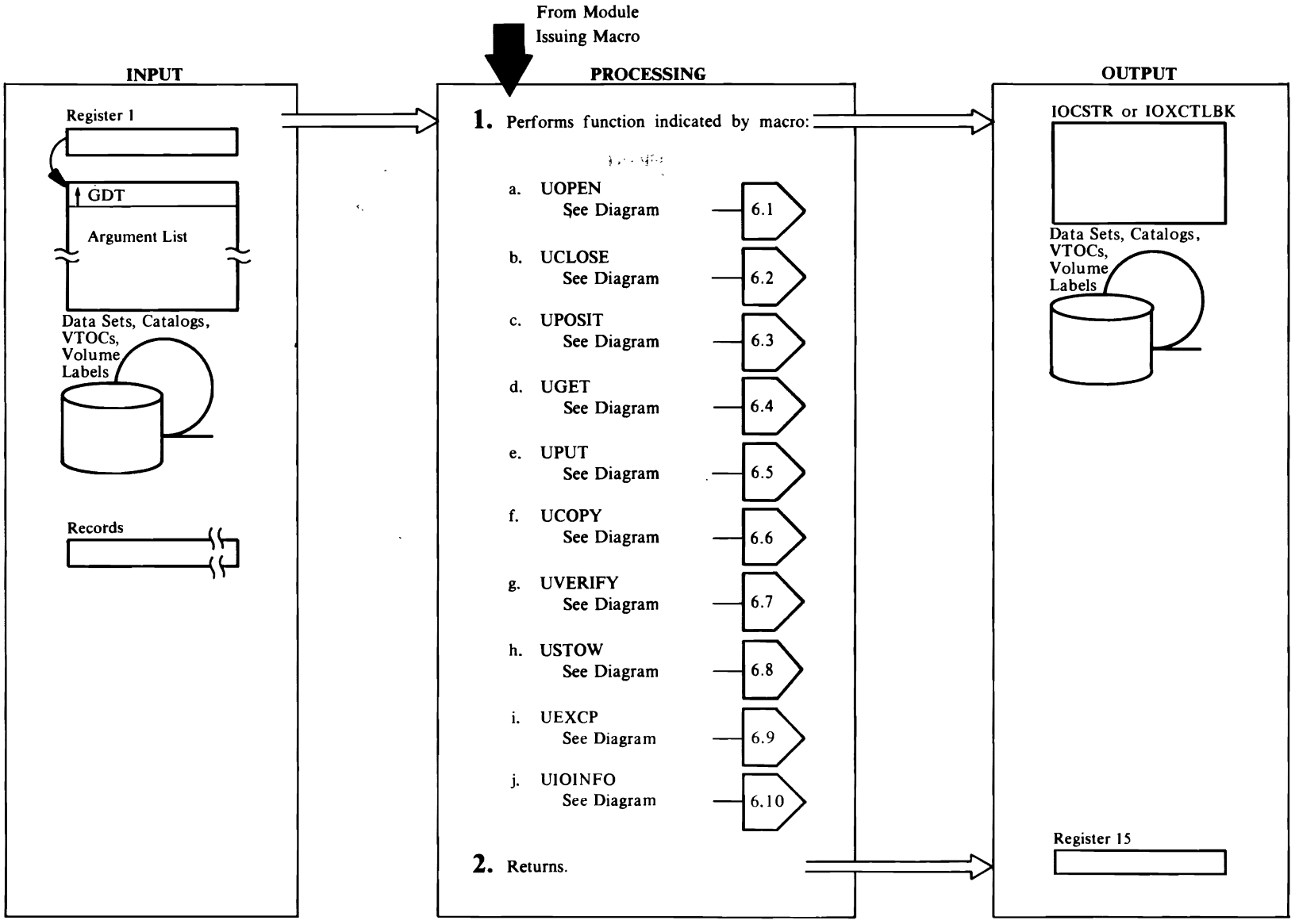


Diagram 6.0 I/O Adapter Overview



Extended Description for Diagram 6.0

Module: IDCIO01, IDCIO02, IDCIO03, IDCIO04, IDCIO05

Procedures: IDCIO01, IDCIO02, IDCIO03, IDCIO04, IDCIO05

1. The I/O operation depends upon the macro:

   a. The UOPEN macro opens from one to four data sets. If DCB parameters are not supplied for the SYSPRINT data set, the default is RECFM=VBA, LRECL=125, and BLKSIZE=629. All ISAM data sets are opened for input and QISAM access method. Diagram 6.1 shows the UOPEN macro in detail.

   b. The UCLOSE macro closes from one to four data sets that were opened by the I/O Adapter. SYSIN and SYSPRINT are not closed with this macro, but at processor termination with the UIOTERM macro. This is to consolidate termination work. Diagram 6.2 shows the UCLOSE macro in detail.

   c. The UPOSIT positions to a record in a data set on a direct access device. The type of positioning depends upon the data set:

      For VSAM data sets, positioning may be by key, relative block address, (RBA) or relative record number.

      For ISAM data sets, positioning is by key only.

      For BSAM or BPAM data sets, positioning is by relative track and record number, TTR*z*.

      Diagram 6.3 shows the UPOSIT macro in detail.

   d. The UGET macro gets a record from a data set opened with a UOPEN macro. If the data set is being processed with keys, the key is returned with the record. For a keyed VSAM data set opened for address processing, the key is in the data, but the address of the key is not returned. For a VSAM data set using blocked processing, a control interval is returned. If a relative-record data set (RRDS) is being processed, a relative record number is returned. Only if a data set is opened for update processing may the record be modified in the buffer. If a data set is opened for update processing, a UPUT must be issued after each UGET even if it is the last UGET before the data set is closed. Update processing is provided by the I/O Adapter when processing a PUT (Replace) request by the REPRO FSR. Diagram 6.4 shows the UGET macro in detail.

   e. The UPUT macro writes records to a data set that was opened with the UOPEN macro. Multiple records can be written with one UPUT. If the data set is VSAM opened for block processing, the record must be a control interval. A UPUT must be issued for each UGET on a data set opened for update. Diagram 6.5 shows the UPUT macro in detail. If the data set is SYSPRINT on a TSO terminal, UPUT gives control to Diagram 6.5.1.

   f. The UCOPY macro copies one data set to another data set if both data sets have been opened with the UOPEN macro. The input data set may be positioned to a starting point with the UPOSIT macro before the copy takes place. The UCOPY copies all records from the input data set starting at the beginning record and continuing until end-of-file or a terminating error. If the output data set has records before the UCOPY, the following applies:

      If the data set is VSAM with keyed sequential record format, the input records are merged with the existing records.

      If the data set is VSAM with entry sequential record format, the input records are added after the existing records.

      If the data set is non-VSAM, the disposition parameter (DISP) on the DD statement controls the placement of the new records. ISAM data sets cannot be used for output with UCOPY.

      The UCOPY macro also copies a VSAM catalog to an empty VSAM catalog if both catalogs have been opened with the UOPEN macro. The entire input catalog is copied.

      Diagram 6.6 shows the UCOPY macro in detail.

   g. The UVERIFY ensures that the address for the end-of-file for the VSAM data set in the VSAM catalog is the same as the end-of-file address on the I/O device. If the two addresses are not identical, the VSAM catalog changes to match the I/O device. The data set must be VSAM opened for control interval output processing. A return code from the UOPEN macro indicates that the data set may need verification.

      The UVERIFY routine checks to see if the data set can be verified before issuing the VSAM VERIFY macro. The FSR should ignore the return code from UOPEN and issue the UVERIFY in all cases except where a zero IOCSTR address is returned from UOPEN. At UOPEN, VSAM just checks the VSAM catalog for information about the data set; it does not check the physical data set. If the UOPEN returns a code saying that there is no data in the data set, the physical data set may or may not have data. Diagram 6.7 shows the UVERIFY macro in detail.

   h. The USTOW macro deletes or renames a member of a partitioned data set that was opened with a UOPEN macro. The partitioned data set must be opened for output and BPAM processing. Diagram 6.8 shows the USTOW macro in detail.

   i. The UEXCP macro provides for EXCP processing of volume labels and data sets. Diagram 6.9 shows the UEXCP macro in detail.

   j. The UIOINFO macro returns information from a JFCB about a data set identified by dname. The macro analyzes an option byte passed by the caller to determine what kind of information is required. The types of information which may be requested are:

      Data-set name
      Volume serial list
      Device type
      Time stamp

      The caller may provide UIOINFO with a work area into which the requested information should be placed or he may provide an UGPOOL ID. In the latter case UIOINFO obtains the required amount of storage. (The caller is responsible for freeing this storage.)

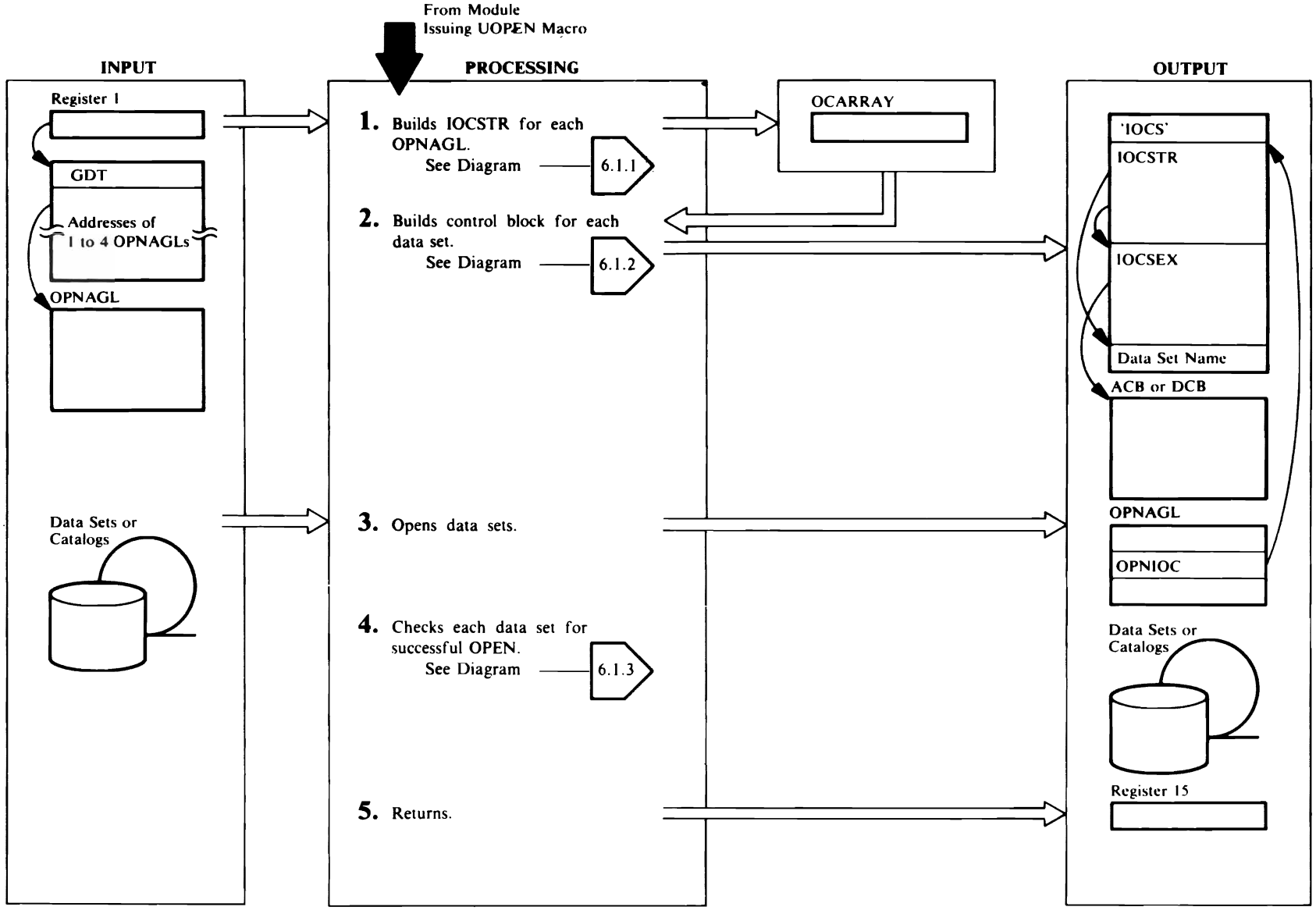
      The data requested is formatted into the return area and control is returned to the caller. Diagram 6.10 shows the UIOINFO macro in detail.

Module: IDCIO01

Procedure: IDCIO01

2. A return code is put in register 15. If the return code is non-zero, a message is written. Control returns to the module that issued the Umacro.

Diagram 6.1 I/O Adapter – UOPEN Macro – Overview



Extended Description for Diagram 6.1

Module: IDCIO01 IDCIO02

Procedures: IDCIOOP, OPENRTN, DSDATA

1. Steps 1 and 2 are repeated for each open argument list, OPNAGL, that the calling module gives to the UOPEN macro via register 1. IDCIOOP builds an internal array, OCARRAY, to describe the open to be performed. OPENRTN increments the identifier in IODSID by 1 to form a unique identifier for the data set. OPENRTN uses the identifier in a UGPOOL macro to obtain storage for an IOCSTR and IOCSEX for the data set and data set name save area. OPENRTN puts the IOCSTR into the chain of IOCSTRs addressed from IODATA in the I/O Adapter Historical Data Area, IODATA. DSDATA gets information about a non-VSAM data set with either the data set name or the DD name. If there is a data set name, the data set is dynamically allocated. If the OPNAGL indicates that the open is for a catalog recovery area (CRA), the DSDATA routine generates a data set name for the CRA, namely, CATALOG.RECOVERY.AREA.VOL.xxxxxx where xxxxxx is the volume serial number of the CRA's first extent. See Diagram 6.1.1.

Module: IDCIO02

Procedure: BUILDACB, BUILDDBK

2. If the data set organization is VSAM, BUILDACB issues a GENCB macro to build an EXLST and an ACB control block. (**Note: VS2.03.808** is installed, the BUILDACB procedure does not issue a GENCB macro to build an EXLST and an ACB control block. It builds them directly.) BUILDACB puts the address and length of the control blocks in the IOCSEX. If the data set organization is non-VSAM. BUILDDBK issues a UGPOOL with the data set identifier from IODSID for storage for the DCB. BUILDDBK moves a model DCB to the storage area and BUILDDBK puts the address of the DCB in IOCSEX in IOCCBA, and initializes the DCB with information for the data set. See Diagram 6.1.2.

Module: IDCIO02

Procedures: OPENRTN, IRTOPEX, ABEXRTN

3. OPENRTN issues one OPEN macro specifying up to four DCBs and/or ACBs. If a modified JFCB is required, OPENRTN issues an OPEN macro, TYPE=J, that specifies up to four DCBs. If the data set is non-VSAM, IRTOPEX exit routine is used to complete the DCB. No exit routine is used in VSAM because all VSAM data sets have been cataloged with the DEFINE command, and VSAM gets all the information from the VSAM catalog. If the OPEN routine detects an ABEND condition while opening a non-VSAM data set, system OPEN passes control to ABEXRTN, an exit routine, which sets IOCINFAE, the abend flag in IOCSEX. OPENRTN attempts an open on all the data sets even if one doesn't open.

Module: IDCIO02

Procedure: OPENRTN, CKNONOP, BUILDRPL

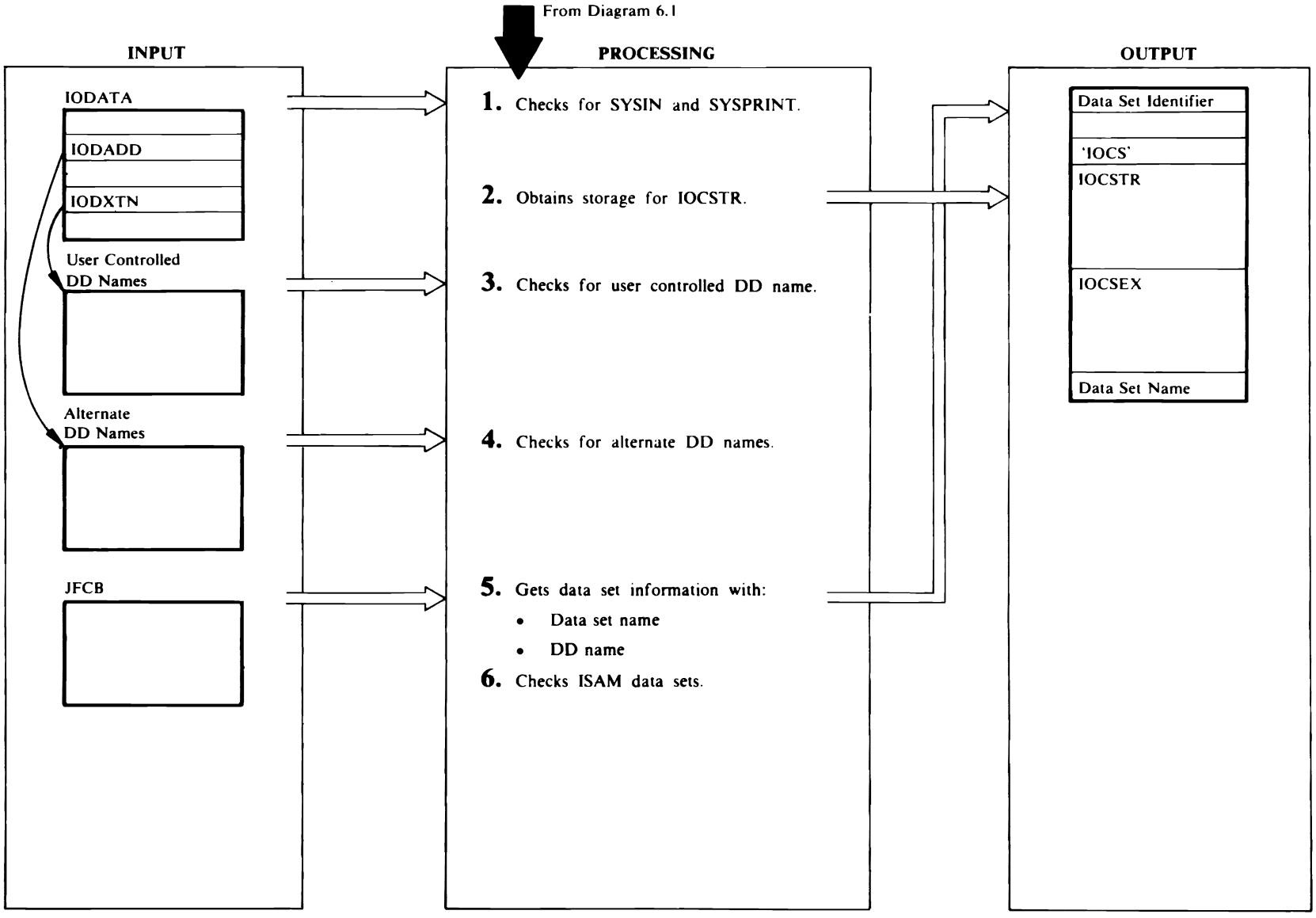
4. (**Note:** If **VS2.03.808** is installed, TESTCB and SHOWCB macros are not issued in this step. Specific fields in the ACB are processed directly.) OPENRTN or CKNONOP tests each data set for a successful open. If the data set is VSAM, OPENRTN issues a TESTCB macro to test the results of the OPEN. If the data set is non-VSAM, CKNONOP checks the open flags in the DCB (DCBOFLGS). If the data set opened successfully, OPENRTN or CKNONOP sets IOCMSGOP in the IOCSTR and IOCFLGOP in the IOCSEX. For a VSAM data set, a second TESTCB is issued to determine if the opened object is a path. If so, keyed processing is assumed; otherwise, if address or control interval processing was not specified in the OPNAGL, OPENRTN issues the TESTCB to determine if the data set has an index. An additional TESTCB is performed to determine if the data set is a Relative Record data set (RRDS). For all VSAM data sets, OPENRTN issues a SHOWCB macro to obtain the data set attributes and BUILDRPL builds a RPL to process the VSAM data set. See Diagram 6.1.3. (**Note:** If **VS2.03.808** is installed, TESTCB and SHOWCB macros are not issued in this step. Specific fields in the ACB are processed directly.)

Module: IDCIO02 IDCIO01

Procedures: DSDATA, BUILDACB, BUILDABK, IRTOPEX, CKNONOP, BUILDRPL, OPENRTN, IDCIOOP

5. If any errors occurred, OPENRTN, DSDATA, BUILDACB, BUILDRPL, or CKNONOP sets a non-zero return code in register 15. IDCIOOP returns control to the module that issued the UOPEN macro.

Diagram 6.1.1 I/O Adapter – UOPEN Macro – Build IOCSTR



Extended Description for Diagram 6.1.1

Module: IDCIO02

Procedure: OPENRTN

1. OPENRTN tests the data set for SYSIN or SYSPRINT.

   SYSIN is tested in two ways:

   - SYSIN is the DD name in the OPNAGL.
   - OPNTYPSI flag in OPNAGL is on.

   SYSPRINT is tested in two ways:

   - SYSPRINT is the DD name in the OPNAGL.
   - OPTYPSO flag in OPNAGL is on.

   If the data set is SYSIN, OPENRTN checks IODICS for an address of an IOCSTR already built for SYSIN. If an IOCSTR is built, SYSIN is already open (or a open was attempted), and OPENRTN returns the address of the IOCSTR for SYSIN in a fullword whose address is in OPNIOC in OPNAGL. No further processing is done on SYSIN. If the data set is SYSPRINT, OPENRTN checks IODOCS for an address of an IOCSTR already built for SYSPRINT. If an IOCSTR is built, SYSPRINT is already open, and OPENRTN returns the address of the IOCSTR for SYSPRINT in a fullword whose address is in OPNIOC in OPNAGL. No further processing is done on SYSPRINT. Processing continues with step 2 if the data set is not open or no open has been attempted.

Module: IDCIO02

Procedure: OPENRTN

2. OPENRTN increments by 1 the data set identifier in IODSID to form a unique identifier for the data set. OPENRTN issues a UGPOOL with the data set identifier to obtain storage for the IOCSTR plus 4 bytes for the characters 'IOCS', the IOCSEX, and the data set name. If storage is not available, PRINTMSG writes a message. OPENRTN chains the new IOCSTR to the last IOCSTR in the chain. If the data set is SYSIN or SYSPRINT, OPENRTN saves the address of the IOCSTR in the I/O Adapter Historical Data Area, IODATA. OPENRTN checks the requested processing of the data set specified in OPNOPT in OPNAGL for input, update or output, and copies into the IOCSTR. Input is the default. The OPNAGL is used to pass information to the I/O Adapter in requesting a data set be opened. Information from the OPNAGL is placed in the IOCSTR and IOCSEX which are then used by the I/O Adapter to control processing of the data set once it is opened. The cross reference at the end of this Extended Description shows how OPNAGL information is transposed into the IOCSTR and IOCSEX.

Module: IDCIO02, IDCIO04

Procedures: OPENRTN, IDCIO04

3. If the invoker of Access Method Services supplied a list of DD names that he wants to control, the address of the list is in IODXTN. If a list exists, OPENRTN compares each entry in the list with the DD name addressed by OPNDDN. If a match is found, OPENRTN puts the address of the user routine in IOCXAD. OPENRTN builds a parameter list for the user routine and puts the address of the parameter list in IOCXPM. OPENRTN gives control to the user routine to do the open. For lack of any information from the user routine about the data set, OPENRTN sets the IOCSTR to indicate the data set is non-VSAM with variable length records and logical record length of 32,760. This does not restrict the type of data sets that can be user controlled. It is just to make the data set appear as something to the FSR that uses it. If the user data set is SYSPRINT on a TSO terminal, IDCIO04 gives a return code of zero because a TSO terminal doesn't need to be opened. If a data set is not user controlled, control continues with step 4.

Module: IDCIO02

Procedure: DSDATA

4. If the invoker of Access Method Services specified a list of alternative DD names, IDCTOIT placed the list address in IODADD at I/O initialization time. The I/O Adapter has a list of DD names for which the invoker can supply alternatives. DSDATA compares the DD name from OPNAGL against the list of names in the I/O Adapter. If a match is found, DSDATA checks the alternative DD name list to see if an alternative DD name has been provided. If a DD name is found, UOPEN uses it instead of the DD name from OPNAGL.

Module: IDCIO02

Procedure: DSDATA

5. The OPNAGL can specify either the data set name or the DD name.

   - If the data set name is specified, DSDATA builds an ALLAGL and issues an UALLOC macro to dynamically allocate the data set. UALLOC returns a DD name for the data set and DSDATA puts the address of DD name in the IOCSTR. If the data set organization is not specified in the OPNAGL, DSDATA puts the data set organization returned by UALLOC in the IOCSTR.

   If OPNOPTBK or OPNOPTKS is not specified, IOCMACCR is set to '1'.

   - If the DD name is provided in the OPNAGL, DSDATA issues a DEVTYPE macro with the DD name in order to obtain device information and to ensure that there is a DD statement with the specified DD name. DSDATA checks the unit type returned by the DEVTYPE macro for unit record and direct access and sets flags in IOCSEX. No indication is set in IOCSEX for magnetic tape. DSDATA issues a RDJFCB macro to obtain more information about the data set. DSDATA uses a QSAM DCB with the RDJFCB macro. If a modified JFCB is required, DSDATA moves the data-set name and the list of volume serial numbers from the OPNAGL to the JFCB and sets a flag in the JFCB to prevent the JFCB from being written. DSDATA generates a data-set name for the CRA. The name generated is: 'CATALOG.RECOVERY.AREA.VOL.xxxxxx ', where xxxxxx is the volume serial number for the first CRA extent. DSDATA checks the JFCB to determine if the data set is identified by DD * or SYSOUT=A. If the data set organization is not specified in the OPNAGL, DSDATA puts the data set organization returned by the JFCB in the IOCSTR. The data set name returned by RDJFCB is saved.

Module: IDCIO01 IDCIO02

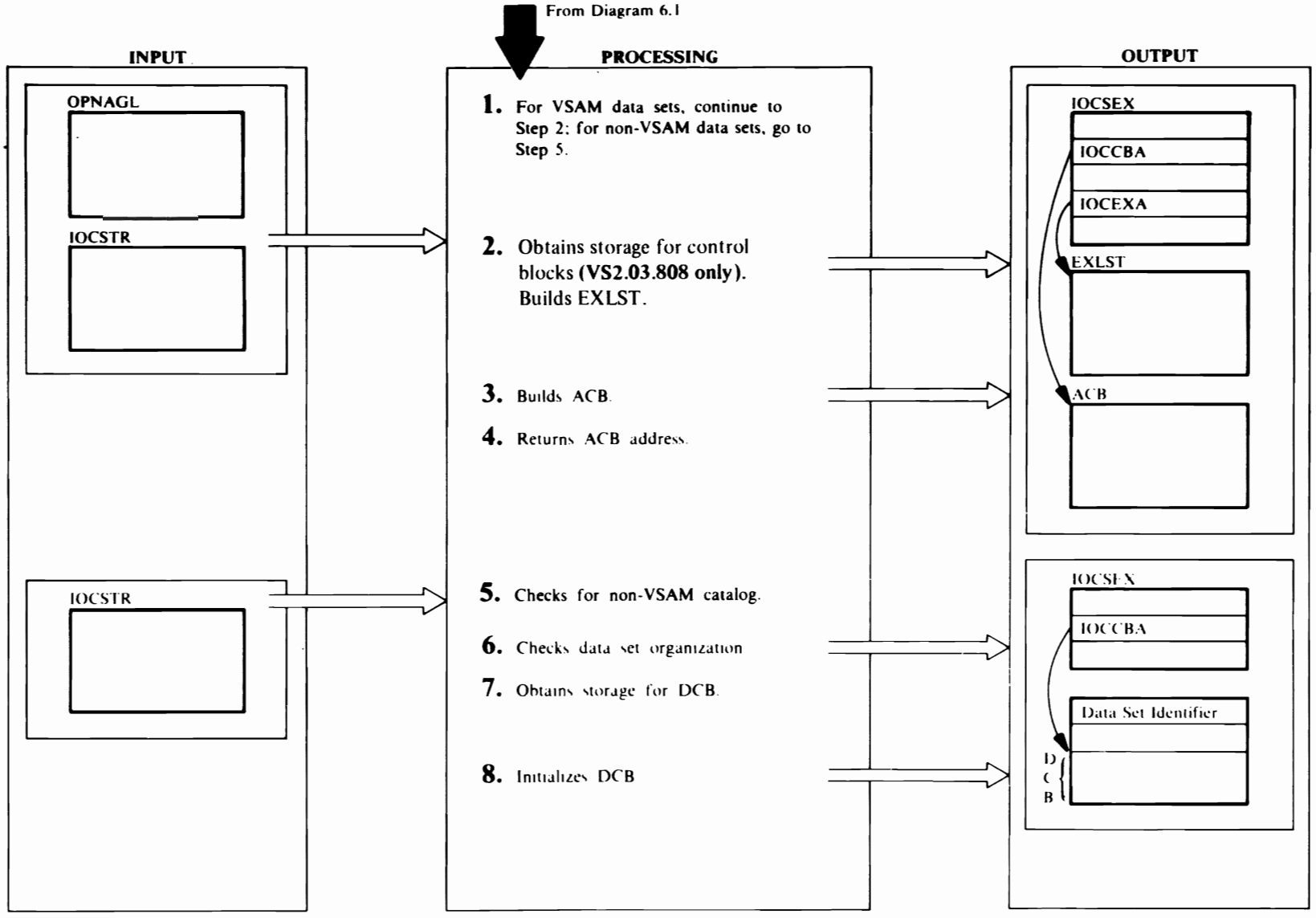
Procedures: IDCIOOP, OPENRTN, DSDATA

6. An ISAM data set can only be opened for input with the QISAM access method. If anything else is specified in the OPNAGL, the ISAM data set is not opened.

OPNAGL IOCSTR/IOCSEX Cross Reference Table

OPNAGL	IOCSTR/IOCSEX	Description
OPNOPTIN	IOCMACIN = '1'	Input processing
OPNOPTOT	IOCMACOT = '1'	Output processing
OPNOPTUP	IOCMACUP = '1'	Update processing
OPNOPTBK	IOCMACBK = '1'	Control Interval processing
OPNOPTKS	IOCMACCR = '0'	Keyed processing
OPNOPTCR	IOCMACCR = '1'	Addressed processing
OPNOPTDR	IOCMACDR = '1'	Direct processing
OPNOPTSK	IOCMACSK = '1'	Skip sequential processing
OPNMODRS	Not required	Open reusable data set with reset
OPNMODAX	Not required	Open alternate index of path only
OPNMODUB	IOCMODUB = '1'	User buffers
OPNMODRP	IOCMODRP = '1'	Replace processing
OPNTYPXM	IOCMODXM = '1'	Export/Import
OPNTYPCI	IOCINFCT = '1'	Open catalog
OPNTYPRA	IOCRCVRA = '1'	Open catalog recovery area
OPNTYPRV	IOCRCVXM = '1'	Recovery bit for VSAM

Diagram 6.1.2 UOPEN Macro Build Control Blocks



Extended Description for Diagram 6.1.2 (Without VS2.03.808)

Module: IDCIO02

Procedure: BUILDACB

1. For a VSAM data set, continue with step 2; for a non-VSAM data set, go to step 5.
2. If a modified JFCB is to be used for opening a VSAM catalog, BUILDACB obtains storage and copies the JFCB previously read and modified by DSDATA. BUILDACB issues a GENCB macro to build an EXLST control block. Only the EODAD exit will be taken if GETVSAM encounters an end-of-file. LERAD and SYNAD exits are specified, but inactive. BUILDACB puts the address and length of the EXLST control block in IOCEXA and IOCEXL, respectively.

Module: IDCIO02

Procedure: BUILDACB

3. BUILDACB issues a GENCB macro to build an ACB control block. ACB parameters set as a result of information contained in the IOCSTR/IOCSEX or OPNAGL.

Bit Referenced	ACB MACRF =
IOCMACIN = '1'	IN
IOCMACOT = '1'	OUT
IOCMACUP = '1'	OUT
IOCMACBK = '1'	CNV
IOCMACCR = '0'	KEY
IOCMACCR = '1'	ADR
IOCMACDR = '1'	DIR
IOCMACSK = '1'	SKP
IOCMODUB = '1'	UBF
OPNMODAX = '1'	AIX
OPNMODRS = '1'	RST

If either OPNOPTDR or OPNOPTSK is not set, MACRF=SEQ is specified.

If IOCRCVRA='1', BUILDACB specifies the CRA=UCRA option for opening a catalog recovery area. If the data set is to be opened as a catalog, BUILDACB specifies the CATALOG OPEN option. If a password is supplied, BUILDACB gives it to the GENCB macro. BUILDACB puts each password in an array of passwords to save the passwords until OPEN time and puts a pointer to the password in the ACB. If a modified JFCB is required, BUILDACB gives the address of the JFCB to the GENCB macro. If IOCRCVRA='1', the third parameter passed to UOPEN is not an address of an OPNAGL; rather it is an address passed by EXPORTRA. The contents of this address must be inserted into the ACBUAPTR field of the ACB. If the type of processing is specified in the OPNAGL, BUILDACB uses it. BUILDACB requests address processing if the data set organization— indexed or non-indexed—is not known. The VSAM open routine will fill in the correct organization. If the organization is not specified, address is set as the default because the GENCB macro defaults to keyed, and VSAM gives an error if the data set is not keyed.

Module: IDCIO02

Procedure: BUILDACB

4. BUILDACB puts the address and length of the ACB in IOCCBA and IOCCBL, respectively. If OPNMODRC in the OPNAGL is 1, BUILDACB puts the address of the ACB in IOCCBP in the IOCSTR.

Module: IDCIO02

Procedure: BUILDDBK

5. If the non-VSAM data set is being opened as a catalog, BUILDDBK does not build control blocks for the data set; the data set is not opened.

Module: IDCIO02

Procedure: BUILDDBK

6. If the data set organization is ISAM, BUILDDBK uses an QISAM DCB. If the data set organization is BPAM, BUILDDBK uses a BSAM DCB and changes the data set organization to partitioned. If the data set is not BPAM and is being opened for blocked processing, BUILDDBK uses a BSAM DCB. If the data set is none of the above, BUILDDBK uses a QSAM DCB.

Module: IDCIO02

Procedure: BUILDDBK

7. If a modified JFCB is required, BUILDDBK adds the address of the JFCB to the exit list; obtains a work area; copies the old exit list, the JFCB exit list, and the JFCB into the work area; and resets the DCBEXLST address to point to the work area. BUILDDBK issues a UGPOOL macro with the data set identifier for storage for the DCB. BUILDDBK puts the address and length of the DCB in IOCCBA and IOCCBL, respectively. If OPNMODRC is one, BUILDDBK puts the address of the DCB in IOCCBP.

Module: IDCIO02

Procedure: BUILDDBK

8. BUILDDBK moves the DCB into the storage obtained by the UGPOOL macro. BUILDDBK moves the DD name and address of an exit list containing the address of the OPEN abend routine and the OPEN exit routine address in the DCB. For an output data set, BUILDDBK puts the address of an output SYNAD Routine in the DCB. If the data set is a BSAM data set and is being opened for update, BUILDDBK obtains a work area for the DECB and puts the address of the work area in IOCDEC. For an input data set, BUILDDBK puts the address of an End-of-Data routine and the address of an input SYNAD Routine in the DCB. If the data set is input and not ISAM, BUILDDBK changes the error option in the DCB from abend to skip. BUILDDBK puts any record format and logical record length specified in the OPNAGL in the DCB. If a blocksize is specified in the OPNAGL, BUILDDBK saves it in the IOCSTR. Subsequently, in the open exit routine, this value will be placed in the DCB if blocksize has not been provided by other means.

Extended Description for Diagram 6.1.2 (With VS2.03.808)

Module: IDCIO02

Procedure: BUILDACB

1. For a VSAM data set, continue with step 2; for a non-VSAM data set, go to step 5.
2. If a modified JFCB is to be used for opening a VSAM catalog, BUILDACB obtains storage and copies the JFCB previously read and modified by DSDATA. BUILDACB issues a UGPOOL to obtain storage for the three VSAM control blocks: EXLST, ACB, and RPL. If OPNSTRNO is zero, BUILDACB obtains storage for one RPL; otherwise the value of OPNSTRNO determines the number of RPLs required. If the return code from UGPOOL is non-zero, BUILDACB sets an error condition code and terminates UOPEN processing.

   BUILDACB first builds an EXLST control block by issuing the EXLST macro. Only the EODAD exit will be taken if GETVSAM encounters an end-of-file. LERAD and SYNAD exits are specified, but inactive. BUILDACB puts the pointer to the EODAD exit routine into the exit list. BUILDACB puts the address and length of the EXLST control block in IOCEXA and IOCEXL, respectively.

Module: IDCIO02

Procedure: BUILDACB

3. BUILDACB builds an ACB control block by issuing the ACB macro. The ACB macro generates the following attributes for the MACRF field: IN, SEQ, ADDR. These attributes are overridden with information contained in the IOCSTR/IOCSEX or OPNAGL.

Bit Referenced	ACB MACRF =
IOCMACOT = '1'	OUT
IOCMACUP = '1'	OUT
IOCMACBK = '1'	CNV
IOCMACCR = '0'	KEY
IOCMACDR = '1'	DIR
IOCMACSK = '1'	SKP
IOCMODUB = '1'	UBF
OPNMODAX = '1'	AIX
OPNMODRS = '1'	RST

   If the type of processing is specified in the OPNAGL, BUILDACB uses it. BUILDACB requests address processing if the data set organization— indexed or non-indexed—is not known. The VSAM open routine will fill in the correct organization. If the organization is not specified, address is set as the default because VSAM defaults to indexed, and gives an error if the data set is not indexed.

   If the data set is to be opened as a catalog, BUILDACB specifies the CATALOG OPEN option. If a password is supplied, BUILDACB puts each password in an array of passwords to save the passwords until OPEN time and puts a pointer to the password in the ACB. If a modified JFCB is required, BUILDACB puts the address of the JFCB in the ACB control block.

   If IOCRCVRA='1', BUILDACB specifies the CRA=UCRA option for opening a catalog recovery area. If IOCRCVRA='1', the third parameter passed to UOPEN is not an address of an OPNAGL; rather it is an address passed by EXPORTRA. The contents of this address must be inserted into the ACBUAPTR field of the ACB. If the value of OPNSTRNO is greater than 1, BUILDACB moves the value of OPNSTRNO to the ACB.

Module: IDCIO02

Procedure: BUILDACB

4. BUILDACB puts the address and length of the ACB in IOCCBA and IOCCBL, respectively. If OPNMODRC in the OPNAGL is 1, BUILDACB puts the address of the ACB in IOCCBP in the IOCSTR.

Module: IDCIO02

Procedure: BUILDDBK

5. If the non-VSAM data set is being opened as a catalog, BUILDDBK does not build control blocks for the data set; the data set is not opened.

Module: IDCIO02

Procedure: BUILDDBK

6. If the data set organization is ISAM, BUILDDBK uses an QISAM DCB. If the data set organization is BPAM, BUILDDBK uses a BSAM DCB and changes the data set organization to partitioned. If the data set is not BPAM and is being opened for blocked processing, BUILDDBK uses a BSAM DCB. If the data set is none of the above, BUILDDBK uses a QSAM DCB.

Module: IDCIO02

Procedure: BUILDDBK

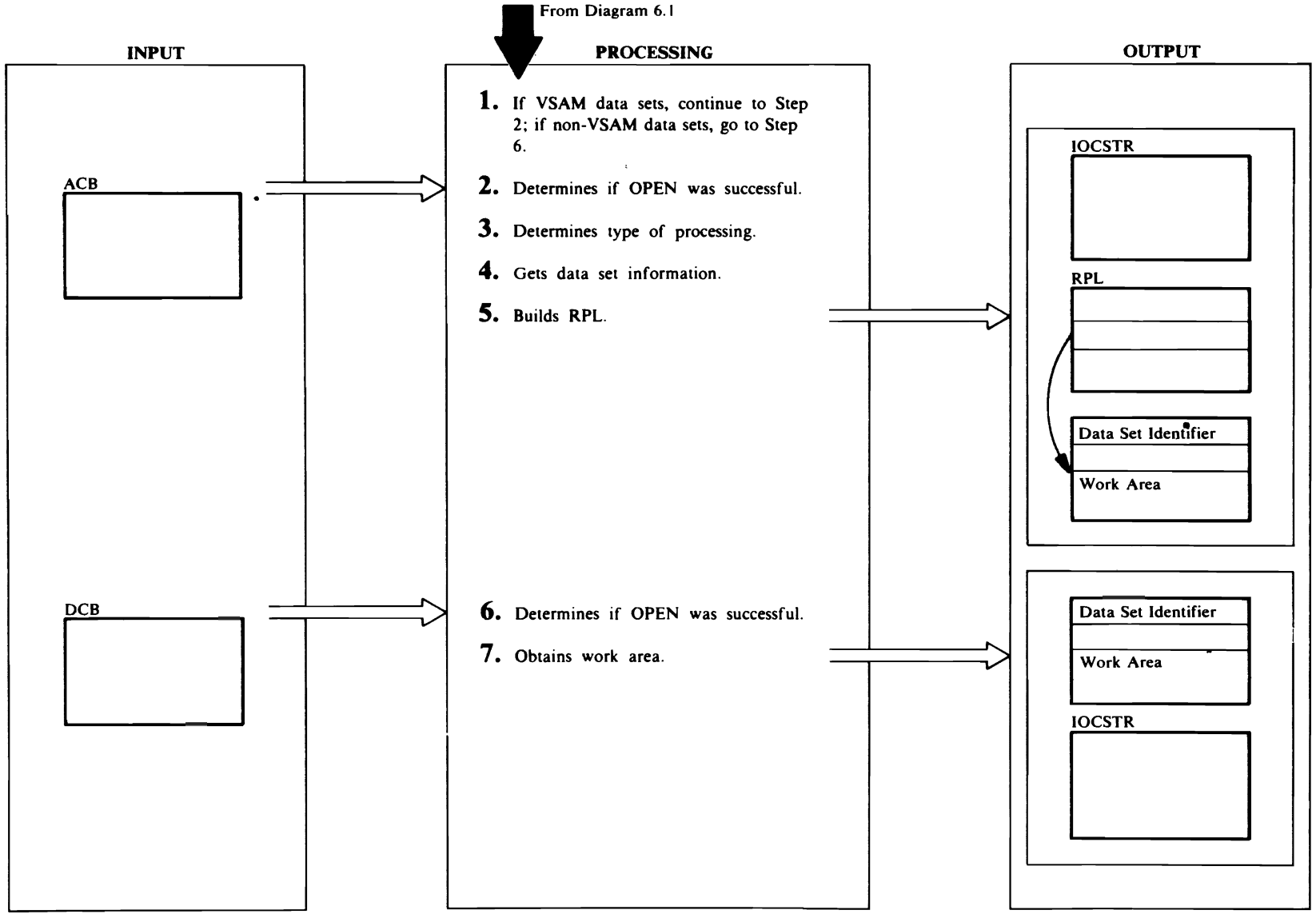
7. If a modified JFCB is required, BUILDDBK adds the address of the JFCB to the exit list; obtains a work area; copies the old exit list, the JFCB exit list, and the JFCB into the work area; and resets the DCBEXLST address to point to the work area. BUILDDBK issues a UGPOOL macro with the data set identifier for storage for the DCB. BUILDDBK puts the address and length of the DCB in IOCCBA and IOCCBL, respectively. If OPNMODRC is one, BUILDDBK puts the address of the DCB in IOCCBP.

Module: IDCIO02

Procedure: BUILDDBK

8. BUILDDBK moves the DCB into the storage obtained by the UGPOOL macro. BUILDDBK moves the DD name and address of an exit list containing the address of the OPEN abend routine and the OPEN exit routine address in the DCB. For an output data set, BUILDDBK puts the address of an output SYNAD Routine in the DCB. If the data set is a BSAM data set and is being opened for update, BUILDDBK obtains a work area for the DECB and puts the address of the work area in IOCDEC. For an input data set, BUILDDBK puts the address of an End-of-Data routine and the address of an input SYNAD Routine in the DCB. If the data set is input and not ISAM, BUILDDBK changes the error option in the DCB from abend to skip. BUILDDBK puts any record format and puts a pointer to the passwrod in the ACB logical record length specified in the OPNAGL in the DCB. If a blocksize is specified in the OPNAGL, BUILDDBK saves it in the IOCSTR. Subsequently, in the open exit routine, this value will be placed in the DCB if blocksize has not been provided by other means.

Diagram 6.1.3 UOPEN Macro Check OPEN



Extended Description for Diagram 6.1.3

(Without VS2.03.808)

Module: IDCIO02

Procedure: OPENRTN

1. The OPEN is checked differently for a VSAM or non-VSAM data set. For a VSAM data set, control goes to step 2; for a non-VSAM data set, control goes to step 6.
2. For VSAM data sets, OPENRTN issues a TESTCB to find out if the open was successful. If the open was successful, OPENRTN sets flags in the IOCSTR and IOCSEX to indicate that the data set can be used and that it must be closed when finished.

Module: IDCIO02

Procedure: OPENRTN

3. OPENRTN issues a second TESTCB to determine if the opened object is a path. If a path has been opened, keyed processing is assumed. If REPLACE processing has been specified for a path, PRINTMSG writes an error message. If the opened object is not a path and if the IOCSTR does not specify control interval or address processing, the type of processing is determined by a TESTCB macro. The TESTCB macro tests the index portion of the data set. If there is an index portion, keyed processing will be used. If there is no index portion, the TESTCB error routine receives control from TESTCB and sets the type of processing to address processing. If the TESTCB indicates the data set has no index, OPENRTN issues a TESTCB specifying ATRB=RRDS. If the test is successful, OPENRTN sets IOCMACCR='0' (keyed) and IOCMACRR='1'. Thus, for a

KSDS	IOCMACCR = 0,	IOCMACRR = 0
ESDS	IOCMACCR = 1,	IOCMACRR = 0
RRDS	IOCMACCR = 0,	IOCMACRR = 1

Module: IDCIO02

Procedure: OPENRTN PRINTMSG

4. OPENRTN issues a SHOWCB macro to obtain the ACB error code, logical record length or control interval, key length, and relative key position. Only the required fields are shown. If the data set did not open, PRINTMSG writes a message. If the data set did open successfully, OPENRTN moves the ACB information to the IOCSTR.

Module: IDCIO02

Procedure: BUILDRPL

5. For any VSAM data set that is open, BUILDRPL builds a request parameter list (RPL). Input work areas are required if the data set is opened for input or update processing. BUILDRPL issues a UGPOOL macro with the data set identification to obtain storage for the maximum length record or one control interval for control interval processing. If IOCMODUB='1', the BUILDRPL procedure of IDCIO02 will not issue a UGPOOL to obtain storage for an I/O area for input or update processing. In subsequent UGET requests the FSR will indicate its own buffers in IOCWORK.

   If IOCMODXM='1' and IOCMACRR='1', indicating EXPORT/IMPORT and RRDS, BUILDRPL will get an extra four bytes for the work area (IOCWKA) if the data set is input (IOCMACIN='1'). The extra four bytes will be utilized in later UCOPY processing for exporting a relative record data set. The work area address specified in the GENCB macro for the RPL is the input work area plus 4 (IOCWKA+4). BUILDRPL issues a GENCB macro to generate the RPL. BUILDRPL gives to the GENCB macro the address of the ACB, options, work area address, maximum length of a data record, message area address, and message area length. If IOCMACRR='1', the OPTCD will indicate 'KEY'. If the RRDS is to be processed for output, IOCMACOT='1' or IOCMACUP='1', OPTCD will indicate 'SKP'. This will cause output RRDS to be processed in skip sequential mode.

Module: IDCIO02

Procedure: BUILDRPL

   Information indicated in IOCSTR/IOCSEX which is reflected in the RPL OPTCD field:

IOCSTR/IOCSEX	RPL OPTCD =
IOCMACUP='1'	UPD
IOCMACUP='0'	NUP
IOCMACDR='1'	DIR
IOCMACSK='1'	SKP
IOCMACCR='0'	KEY
IOCMACCR='1'	ADR
IOCMACBK='1'	CNV

   If either IOCMACDR or IOCMACSK is not set, OPTCD=SEQ is specified.

   If no space is available for the work area or the RPL, BUILDRPL sets an error return code, PRINTMSG writes a message, and OPENRTN turns off the open flag in the IOCSTR. Every VSAM data set processed by the I/O Adapter uses the same message area where VSAM formats a SYNAD message if an I/O error is detected.

Module: IDCIO02

Procedure: CKNONOP

6. For non-VSAM data sets, CKNONOP checks the DCB open flags to determine if the OPEN was successful. CKNONOP puts the address of an exit list containing the address of the EOV abend routine in the DCB. If the OPEN was successful, CKNONOP sets flags in the IOCSTR and IOCSEX to indicate that the data set can be used and that it must be closed when finished. If the data set was not opened successfully, CKNONOP checks IOCINFAE to find out if the open abended. PRINTMSG writes a message.

Module: IDCIO02

Procedures: OPENRTN, BUILDRPL, PRINTMSG, CKNONOP, BLDOCMSG

7. A work area the size of a logical record plus the key is needed for an ISAM data set with unblocked fixed length records and relative key position of zero. A work area the size of a physical block is needed for a BSAM or BPAM data set open for input. CKNONOP issues a UGPOOL macro with the data set identifier to obtain storage for the work area. CKNONOP puts the address of the work area in IOCDAD. The key and data will be moved from the buffer to the work area when a UGET macro is issued.

Extended Description for Diagram 6.1.3 (With VS2.03.808)

Module: IDCIO02

Procedure: OPENRTN

1. The OPEN is checked differently for a VSAM or non-VSAM data set. For a VSAM data set, control goes to step 2; for a non-VSAM data set, control goes to step 6.
2. For VSAM data sets, OPENRTN checks the ACBOPEN flag to find out if the open was successful. If the open was successful, OPENRTN sets flags in the IOCSTR and IOCSEX to indicate that the data set can be used and that it must be closed when finished.

Module: IDCIO02

Procedure: OPENRTN

3. OPENRTN makes another check to determine if the opened object is a path. If a path has been opened, keyed processing is assumed. If REPLACE processing has been specified for a path, PRINTMSG writes an error message. If the opened object is not a path and if the IOCSTR does not specify control interval or address processing, the type of processing is determined by checking the index portion of the data set. If there is an index portion, keyed processing will be used. If there is no index portion, the type of processing is set to address processing. If the data set has no index, OPENRTN next checks the ACB to see if the data set is RRDS; if so, OPENRTN sets IOCMACCR='0' (keyed) and IOCMACRR='1'. Thus, for a

KSDS	IOCMACCR = 0,	IOCMACRR = 0
ESDS	IOCMACCR = 1,	IOCMACRR = 0
RRDS	IOCMACCR = 0,	IOCMACRR = 1

Module: IDCIO02

Procedure: OPENRTN PRINTMSG

4. OPENRTN obtains the ACB error code, logical record length or control interval, key length, and relative key position. Only the required fields are obtained. If the data set did not open, PRINTMSG writes a message. If the data set did open successfully, OPENRTN moves the ACB information to the IOCSTR.

Module: IDCIO02

Procedure: BUILDRPL

5. For any VSAM data set that is open, BUILDRPL builds a request parameter list (RPL). Input work areas are required if the data set is opened for input or update processing. BUILDRPL issues a UGPOOL macro with the data set identification to obtain storage for the maximum length record or one control interval for control interval processing. If IOCMODUB='1', the BUILDRPL procedure of IDCIO02 will not issue a UGPOOL to obtain storage for an I/O area for input or update processing. In subsequent UGET requests the FSR will indicate its own buffers in IOCWORK.

   If IOCMODXM='1' and IOCMACRR='1', indicating EXPORT/IMPORT and RRDS, BUILDRPL will get an extra four bytes for the work area (IOCWKA) if the data set is input (IOCMACIN='1'). The extra four bytes will be utilized in later UCOPY processing for exporting a relative record data set. The work area address specified for the RPL is the input work area plus 4 (IOCWKA+4). If no space is available for the work area BUILDRPL sets an error return code, PRINTMSG writes a message, and OPENRTN turns off the open flag in the IOCSTR.

   BUILDRPL generates an RPL via the RPL macro and initializes it with the address of the ACB, options, work area address, maximum length of a data record, message area address, and message area length. If IOCMACRR='1', the OPTCD will indicate 'KEY'. If the RRDS is to be processed for output, IOCMACOT='1' or IOCMACUP='1', OPTCD will indicate 'SKP'. This will cause output RRDS to be processed in skip sequential mode.

Module: IDCI02

Procedure: BUILDRPL

   The RPL macro generates KEY, SEQ, NUP for the OPTCD field. These attributes are overridden with information indicated in IOCSTR/IOCSEX which is as follows:

IOCSTR/IOCSEX	**RPL OPTCD =**
IOCMACUP='1'	UPD
IOCMACDR='1'	DIR
IOCMACSK='1'	SKP
IOCMACCR='1'	ADR
IOCMACBK='1'	CNV

   The length of the RPL times ACBSTRNO is stored in the IOCLRP field of the IOCSTR extension (IOCSEX). If ACBSTRNO is greater than 1, the first RPL is copied to each additional RPL area.

Module: IDCIO02

Procedure: CKNONOP

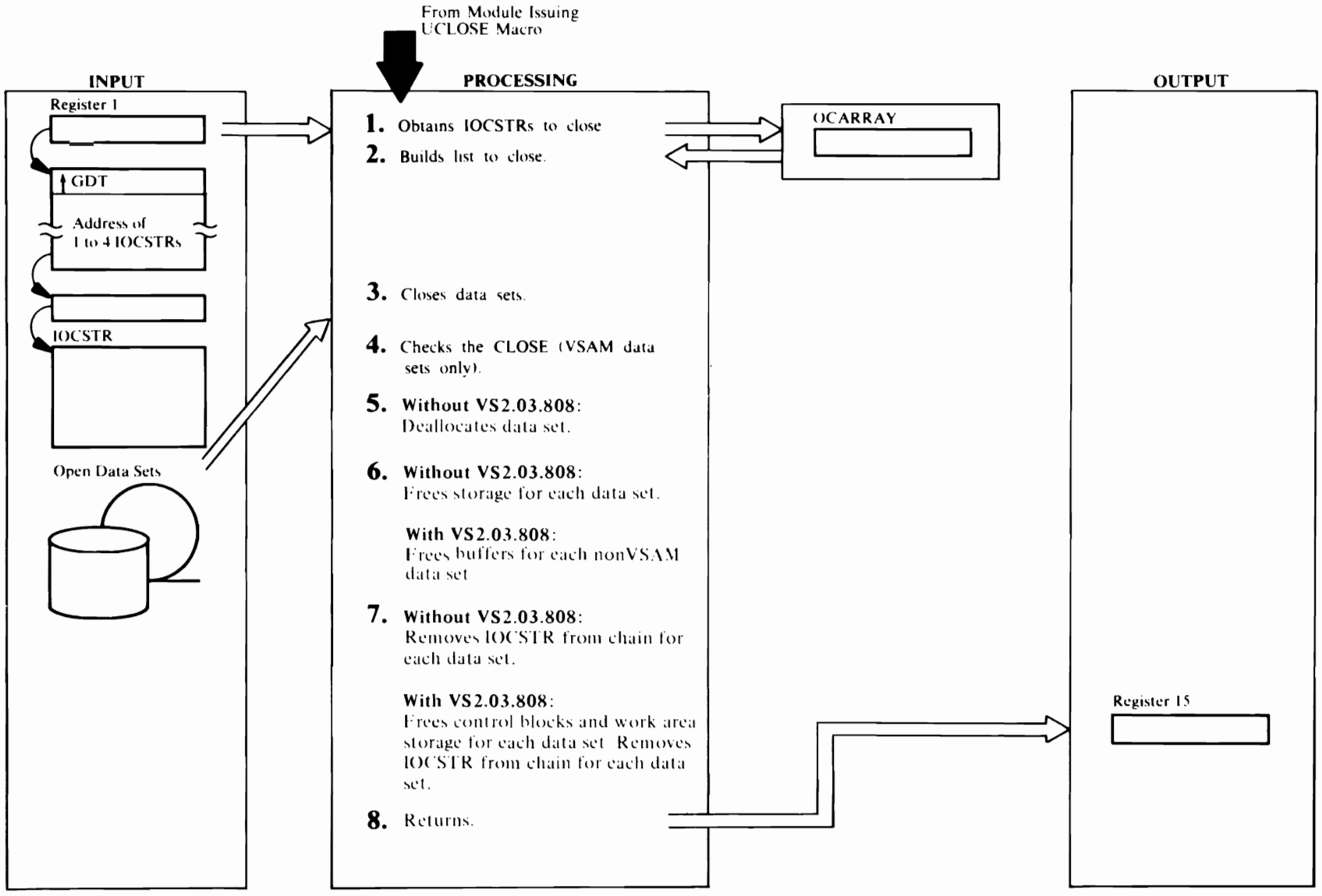
6. For non-VSAM data sets, CKNONOP checks the DCB open flags to determine if the OPEN was successful. CKNONOP puts the address of an exit list containing the address of the EOV abend routine in the DCB. If the OPEN was successful, CKNONOP sets flags in the IOCSTR and IOCSEX to indicate that the data set can be used and that it must be closed when finished. If the data set was not opened successfully, CKNONOP checks IOCINFAE to find out if the open abended. PRINTMSG writes a message.

Module: IDCIO02

Procedures: OPENRTN, BUILDRPL, PRINTMSG, CKNONOP, BLDOCMSG

7. A work area the size of a logical record plus the key is needed for an ISAM data set with unblocked fixed length records and relative key position of zero. A work area the size of a physical block is needed for a BSAM or BPAM data set open for input. CKNONOP issues a UGPOOL macro with the data set identifier to obtain storage for the work area. CKNONOP puts the address of the work area in IOCDAD. The key and data will be moved from the buffer to the work area when a UGET macro is issued.

Diagram 6.2 I/O Adapter UCLOSE Macro



Extended Description for Diagram 6.2

Module: IDCIO01, IDCIO02

Procedure: IDCIOCL BUILDDBK

1. IDCIOCL puts the addresses of IOCSTRs in OCARRAY. Even if the address is zero, it is put in OCARRAY. The address will be zero if a UOPEN was issued against a data set, but the IOCSTR could not be built. BUILDDBK sets the type of operation to 'close' in OCATYP in OCARRAY.

Module: IDCIO02

Procedure: CLOSERTN

2. Only a maximum of four data sets are closed with any one UCLOSE macro. CLOSERTN examines OCARRAY for the addresses of IOCSTRs to close. If the address of an IOCSTR is not zero and CLOSE ALL is not requested, CLOSERTN checks the data set for SYSIN or SYSPRINT. If the data set is SYSIN or SYSPRINT, CLOSERTN does not close the data sets because they are needed until processor termination. For any other non-zero IOCSTR, CLOSERTN saves the address. And, if the DCB or ACB is opened, CLOSERTN saves the address of the control block in preparation for closing. For a non-VSAM data set, CLOSERTN sets the address of a SYNAD routine to zero in the DCB, and CLOSERTN puts the address of a close exit list, containing the CLOSE abend routine address, in the DCB. If the data set is not open, IOCFLGOP=1, CLOSERTN makes a check to determine if it is user controlled. If it is user controlled, CLOSERTN passes arguments to the user routine. CLOSERTN continues the above checking until:

   - IDCIO01 specifies CLOSE ALL in OCARRAY and CLOSERTN has found four IOCSTR addresses. This happens during I/O termination.
   - IDCIO01 does not specify CLOSE ALL in OCARRAY, and CLOSERTN has checked all IOCSTR addresses in OCARRAY.

Module: IDCI002

Procedure: CLOSERTN, ABEXRTN

3. CLOSERTN issues a CLOSE macro with the address of up to four DCBs or ACBs. If an abend occurs during the closing of an non-VSAM data set, a close exit routine gets control and sets flag IOCINFAE, indicating that an abend has occurred. The system CLOSE continues processing with the next data set to close.

Module: IDCIO02

Procedure: CLOSERTN

4. For VSAM data sets, CLOSERTN issues a SHOWCB macro to return the ACB error code. If the ACB error code is non-zero, PRINTMSG writes a message. No tests are made for non-VSAM data sets except to write a message and set an error code if the DCB caused a CLOSE abend. (Note: **VS2.03.808** is installed, the SHOWCB macro is not issued. CLOSERTN checks the ACB error code directly.)

Module: IDCIO02

Procedure: ENVFREE

5. If the data set has been dynamically allocated, ENVFREE builds an ALLAGL with a disposition of KEEP. ENVFREE then issues a UDEALLOC macro to deallocate the data set.

Module: IDCIO02

Procedure: ENVFREE

6. **Without VS2.03.808:** For VSAM data sets, ENVFREE checks the IOCSEX to see if there are any VSAM control blocks to free. ENVFREE compares the length fields for the ACB, RPL, and EXLST against zero. When any length is non-zero, ENVFREE issues a FREEMAIN against the control block. For non-VSAM data sets that were open, ENVFREE issues a FREEPOOL macro to free buffers obtained by the OPEN routines.

   With VS2.03.808: For non-VSAM data sets that were open, ENVFREE issues a FREEPOOL macro to free buffers obtained by the OPEN routines. For VSAM data sets, the storage for the ACB, RPL, and exit list control blocks is freed in Step 7 along with the IOCSTR and all other storage having the same IOCSID.

Module: IDCIO02

Procedure: CLOSERTN

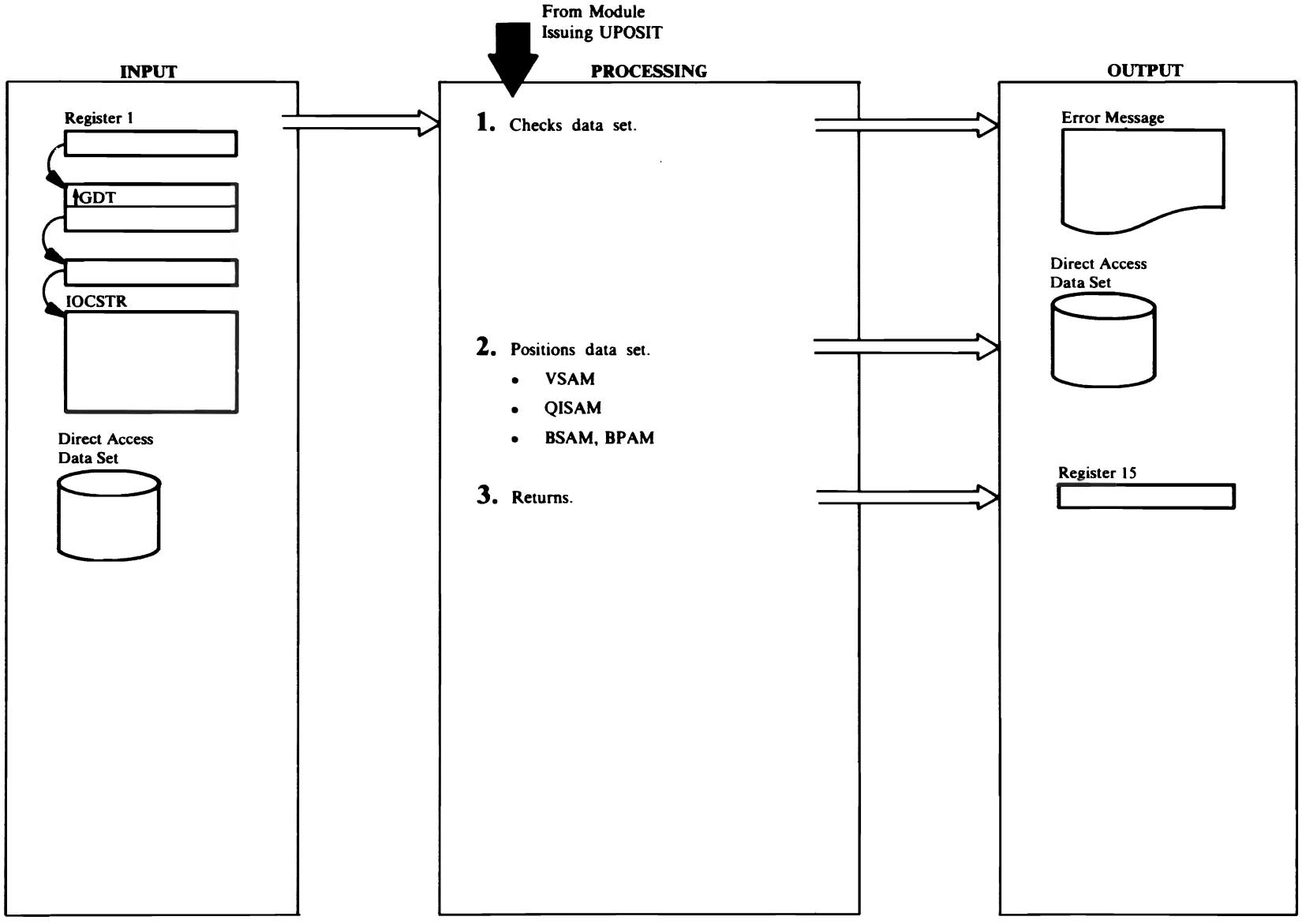
7. CLOSERTN saves the address of the IOCSTR that was closed and the address of the next IOCSTR in the chain. CLOSERTN issues a UFPOOL to free all storage obtained for the data set that is closed. CLOSERTN passes the IOCSID field to UFPOOL, which identifies all storage obtained for the data set. CLOSERTN searches the IOCSTR chain until the IOCSTR is found that points to the closed IOCSTR. CLOSERTN replaces the address of the closed IOCSTR with the address of the next IOCSTR in the chain.

Module: IDCIO01

Procedure: IDCIOCL

8. IDCIOCL puts a return code in register 15 and returns control to the module that issued the UCLOSE.

Diagram 6.3 I/O Adapter UPOSIT Macro



Extended Description for Diagram 6.3

Module: IDCIO03

Procedure: ICDIO03, PRINTMSG

1. If the IOCSTR address is zero or the data set is not open (IOCMSGOP=0), IDCIO03 issues a UABORT macro. A user controlled data set cannot be positioned. If the data set is open for processing (IOCMSGOP=1), and the data set is user controlled (IOCFLFEX=1), IDCIO03 sets a return code of zero, and control goes to step 3. If the data set is open for QSAM, PRINTMSG writes an error message and control goes to step 3. Whenever any error messages are written, IDCIO03 turns off the open for processing indicator, IOCMSGOP, so that no more I/O operations except close are permitted against the data set.

Module: IDCIO03

Procedures: PTAMDS, PRINTMSG, PTISDS

2. IDCIO03 positions the data set depending upon the access method for the data set:

   (**Note:** If **VS2.03.808** is installed- PTAMDS does not issue MODCB or SHOWCB macros. Fields in the RPL are processed directly.)

   - For a VSAM data set, PTAMDS issues a MODCB macro to place the POINT argument in the RPL. VSAM uses the POINT argument in the RPL to position to the requested record. If the data set is open for adddress processing, PTAMDS puts the address of the Relative Byte Address (RBA) in IOCRBA into MODCB. If the data set is RRDS (IOCMACRR='1'), the MODCB ARG parameter is set to contain the address of the relative record number which is contained in IOCREL. If control interval processing is specified (IOCMACBK='1'), the MODCB ARG parameter is set to contain the address of the RBA which is contained in IOCRBA. Otherwise, PTAMDS puts the address of the key in IOCKYA into MODCB. If the key length of the requested record is greater than the key length for the data set, PRINTMSG writes an error message, and PTAMDS does not position to the requested record. PTAMDS expands every key to 256 bytes by adding binary zeros on the right. PTAMDS issues another MODCB to inactivate the End-of-Data routine in the EXLST control block. This is done to prevent the End-of-Data routine from getting control if the record positioned to is beyond the end of the data set. If the End-of-Data routine receives control, an abend would occur. PTAMDS issues the POINT macro to position to the record with the key or the next higher key. PTAMDS issues a MODCB macro to reactivate the End-of-Data exit routine. If the return code from the POINT macro is 12, an I/O error has occurred, and a message is formatted in the message area by VSAM. PRINTMSG prints the error message. If the return code from the POINT macro is 8, a logic error has occurred, and PTAMDS issues a SHOWCB macro. If the results from the SHOWCB macro indicate that no record was found or there was repositioning beyond the end-of-file, PTAMDS sets a return code. For all other logic errors, PRINTMSG writes a message containing the VSAM RPL error code. For **VS2.03.808**, the "suppress messages" flag, IOCMSGSM, is checked. If it is set, PRINTMSG does not write the message.

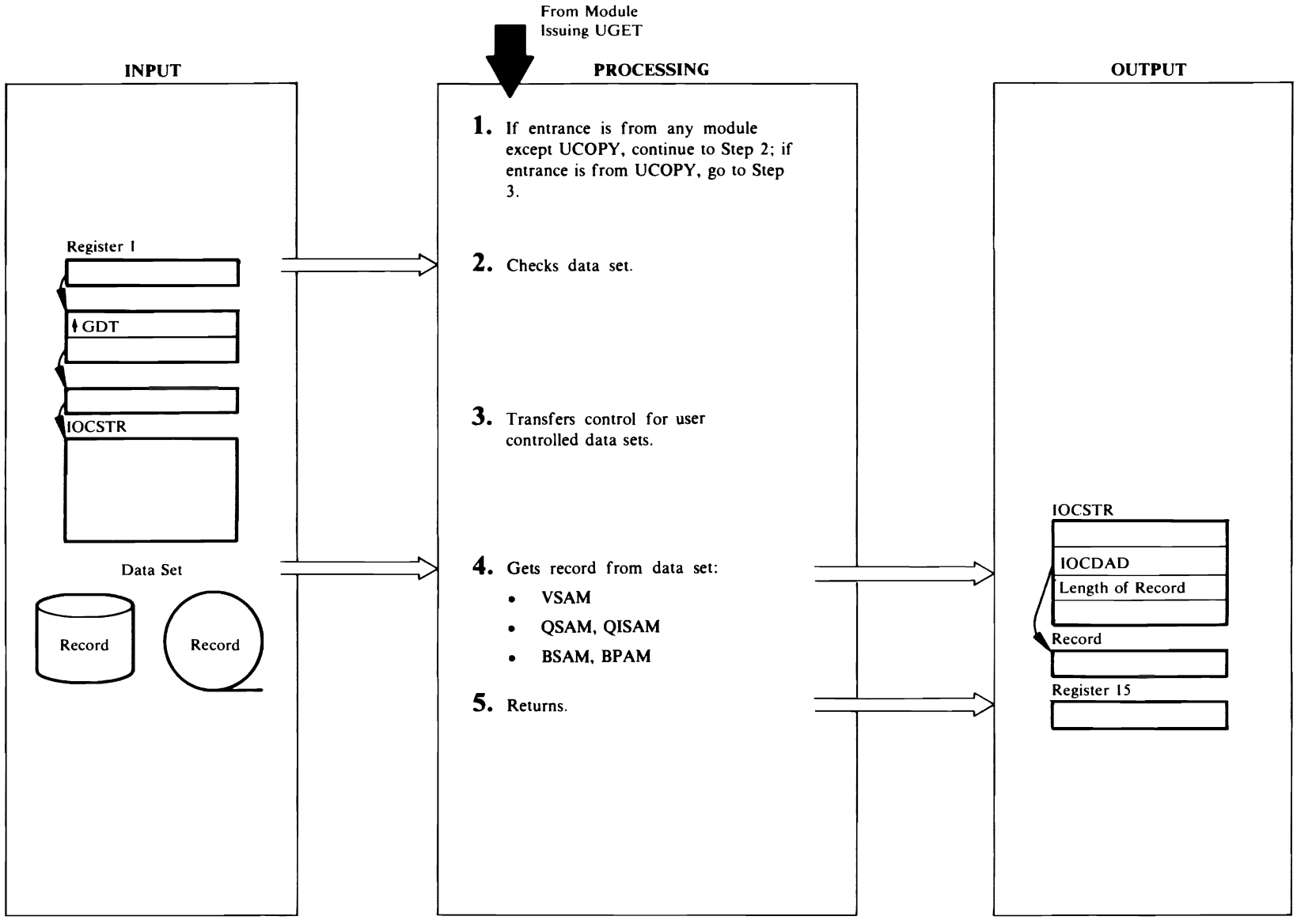
   - For a non-VSAM data set open for QISAM, PTISDS does not position the record if the length supplied is greater than the key length for the data set. For valid key lengths, PTISDS does the positioning. PTISDS sets the SYNAD address in the DCB to the address of a routine to handle "no record found" situations. PTISDS expands the key to 256 bytes by padding on the right with binary zeros. If a SETL has been issued against the DCB, PTISDS issues a ESETL macro. PTISDS issues a SETL macro to position to the record with the key or next higher key. If the positioning is beyond the end of the data set, the SYNAD routine receives control from SETL, sets a return code, and puts the address of the input SYNAD routine in the DCB. If no errors were encountered, PTISDS puts the address of the input SYNAD routine in the DCB.
   - For a non-VSAM data set open for BSAM or BPAM, PTISDS issues a POINT macro to position to the TTR*z* addressed by IOCTTR. If *z* is zero, the exact TTR is positioned to; if *z* is one, the next physical block after the TTR is positioned to. No error checking is done after the POINT macro because errors cannot be detected until the next READ or WRITE on the data set.

Module: IDCIO01

Procedure: IDCIOPO

3. IDCIOPO puts a return code in register 15 and returns control to the module that issued the UPOSIT.

Diagram 6.4 I/O Adapter – UGET Macro



Extended Description for Diagram 6.4

Module: IDCIO01

Procedure: IDCIOGT

1. If entrance is from any module except UCOPY, control goes to step 2. If entrance is from UCOPY, control goes to step 3.
2. If the address of the IOCSTR is zero or the data set is not open for processing, (IOCMSGOP=0), IDCIOGT issues a UABORT macro to terminate processing. If end-of-file has been encountered, (IOCFLGEF=1), on an input data set, IDCIOGT returns control to the module issuing the UGET. This check allows more than one module to issue UGETs on the same data set, and both modules will get end-of-file indications by a return code.

Module: IDCIO01

Procedure: GETEXT

3. If the data set is user controlled, GETEXT passes an argument list to the user routine so the user routine can perform the I/O operation. GETEXT tests the return code from the user routine. If the return code is zero, GETEXT moves the address and length of the data record just read to the IOCSTR, and GETEXT increments the count of successful UGETs. If the return code is end-of-file, GETEXT sets the end-of-file flag in the IOCSTR and GETEXT sets the return code to end-of-file. If the return code is 12 indicating that no more I/O operations can be performed against the data set, GETEXT turns off the open-for-processing flag, IOCMSGOP. For any other return code except 0 and end-of-file, GETEXT sets a return code of 4. IDCIOGT returns control to the module issuing the UGET.

Module: IDCIO01

Procedures: GETVSAM, CHANGE (**VS2.03.808 only**), VSAMERR, PRINTMSG, GETNONVS, IRSISYN

4. IDCIOGT reads the data depending upon the access method for the data set:
   - For a VSAM data set, If IOCMACCP='1', indicating a change in processing modes, the appropriate change is made in the RPL. The following IOCSTR settings specified by the issuer of UGET are reflected in the RPL:

IOCSTR	RPL OPTCD =
IOCCHPSQ	SEQ
IOCCHPDR	DIR

If **VS2.03.808** is installed, the following additional IOCSTR settings are reflected in the RPL:

IOCHPSK	SKP
IOCCHPKS	KEY
IOCCHPCR	ADR
IOCCHPBK	CNV
IOCCHPKG	KGE
IOCCHPKE	KEQ
IOCCHPUP	UPD
IOCCHPNU	NUP

GETVSAM (or **CHANGE if VS2.03.808)** will set all change processing flags to '0', and the IOCSTR will be changed to reflect the new processing option.

If the data set is RRDS, (IOCMACRR='1', RPLARG is set to the address of IOCREL so that VSAM will return the relative record number to UGET.

If user buffer is specified (IOCMODUB='1'), the caller has placed the address of the input work area in IOCWORK. This address will be placed in the RPL work area field.

For OPTCD=CNV or ADR with DIR or SKP, the caller has placed an RBA in IOCRBA. The address of IOCRBA will be placed in the RPLARG field. In this situation, the RBA will not be moved to IOCRBA following the GET.

For OPTCD=KEY with DIR or SKP, the caller has placed the address of the key in IOCKYA and its length in IOCKYL. RPLARG is set equal to IOCKYA and RPLKEYLN is set equal to IOCKYL.

GETVSAM issues a GET macro in the move mode and specifies the address of the RPL built when the data set was opened. If end-of-file is encountered, the VSAM EODAD exit routine sets the end-of-file flag in the IOCSTR and sets the return code to indicate end-of-file. GETVSAM tests the return code from GET. If the return code indicates a logic error, a logic error code is in the RPL. If the return code indicates an I/O error, VSAM has formatted a message in the message area and put a code in the RPL. If the return code is zero, the VSAM GET routine has read the record or control interval. GETVSAM moves the record address, record length, and RBA (if GET SEQ) to the IOCSTR. The record length is increased by four if a relative record data set is being exported. This 4-byte area is used by UCOPY to store the relative record number. If the data set is being processed by key, GETVSAM places the address of the key in the record just read in the IOCSTR. If the return code from the GET is non-zero, VSAMERR obtains the error code from the RPL, and PRINTMSG writes a message. However, if **VS2.03.808** is installed, the "suppress messages" flag, IOCMSGSM, set by the UGET caller is checked. If it is on, the call to VSAMERR to print logical error messages is bypassed.

- For a non-VSAM data set open for block processing, GETNONVS issues a READ macro and moves the length of the record read to the IOCSTR. For a non-VSAM data set open for QSAM or QISAM, GETNONVS reads the data as a logical record. GETNONVS issues a GET in the locate mode specifying the DCB address. GET spanned record processing is the same as unspanned because UOPEN sets BUFTEK=A in the DCB, and GET puts all the pieces of the record together before returning to the I/O Adapter. If an I/O error occurs during the GET, the non-VSAM input SYNAD routine issues a SYNADAF and a SYNADRLS macro to release the SYNADAF message after the message has been moved to a save area. IRSISYN turns off the open-for-processing flag (IOCMSGOP so that no more processing can be done against the data set. If end-of-file is encountered, the I/O Adapter end-of-file routine turns on the end-of-file flag in the IOCSTR and sets a return code. After the GET, if no exit routines were called, the GET was successful, and the GETNONVS puts the address of the data portion of the record (logical record address plus four for the variable length records) in the IOCSTR. GETNONVS puts the length of the data portion of the record in the IOCSTR. If the input IOCSEX indicates a catalog recovery area for import (IMPORTRA), the GETNONVS routine strips off the 4-byte header record prepended to it when the record was exported via EXPORTRA (see UPUT Diagram 6.5). For ISAM data with key position other than zero, the key is included in the data portion of the record. For ISAM data sets with fixed unblocked records and a relative key position of zero, GETNONVS moves the key and data to a work area so they will be contiguous. GETNONVS puts the address of the key in the IOCSTR for all ISAM data sets.
- For a non-VSAM data set open for BSAM or BPAM, the record is read as a physical block. If the data set is open for update, the DECB is copied

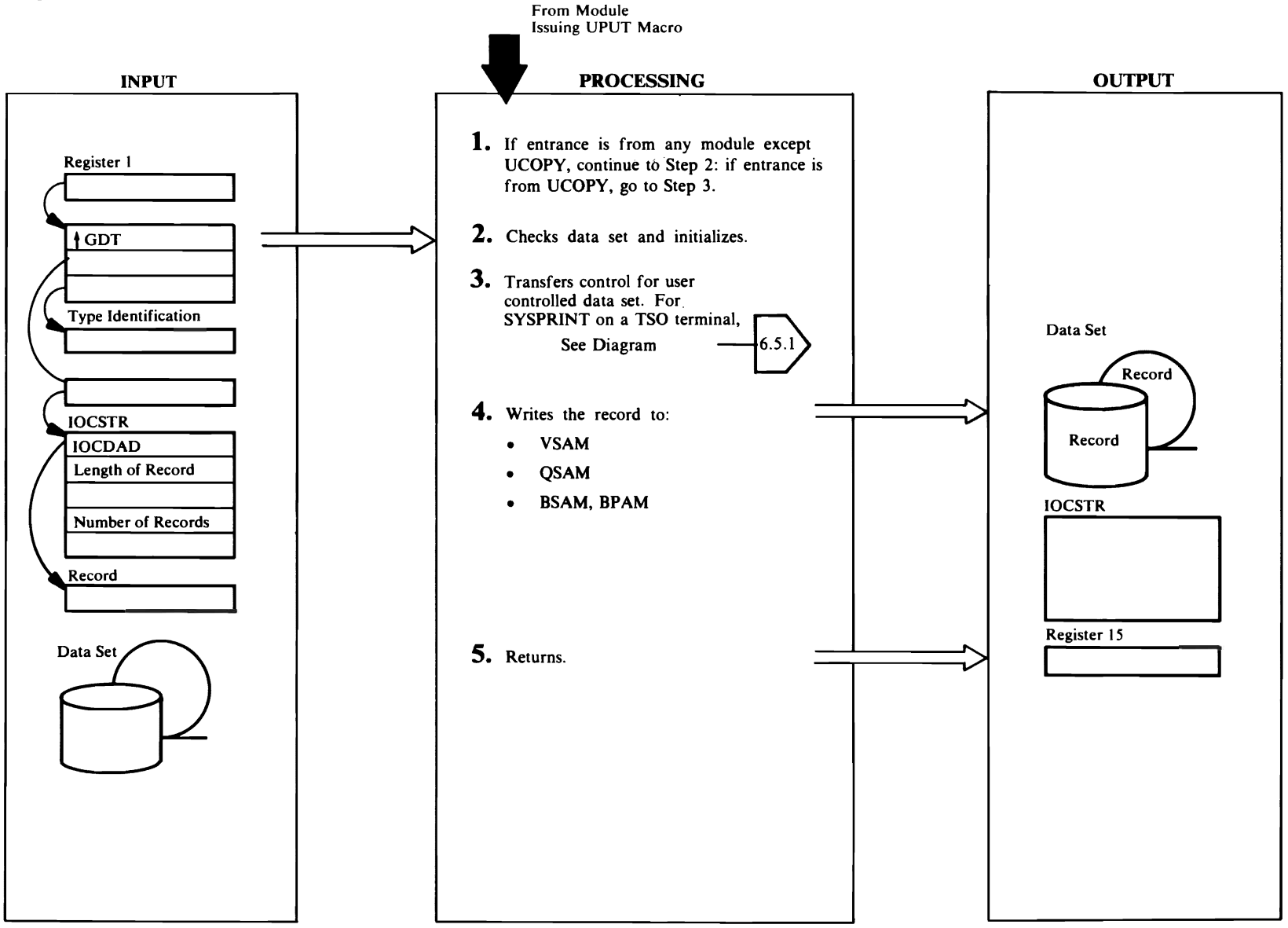
into the work area whose address is in IOCDEC. GETNONVS issues a READ macro with the 'S' option to read the physical block. GETNONVS then issues a CHECK macro to test the READ. If an I/O error occurred, IRSISYN issues SYNADAF and SYNADRLS macros to format an error message and turns off the open-for-processing flag, IOCMSGOP. If end-of-file occurred, IROSEOD turns on the end-of-file flag, IOCFLFEF. If the Operating System End-of-Volume routines detect an error during End-of-Volume, IRIEVAB turns IOCMSGOP, formats a message, and gives a code to the End-of-Volume routines that causes the End-of-Volume routines to ignore the error and return to the I/O Adapter. If the READ is successful, GETNONVS updates the physical block length in IOCDLN. For fixed length, unblocked record format, GETNONVS moves the length of the block just read from the DCB to IOCDLN. For variable or spanned record formats, GETNONVS gets the block length from the Block Descriptor Word preceding the physical block. For undefined and fixed blocked record formats, GETNONVS obtains IOCDLN by subtracting the residual count in the IOB from the blocksize in the DCB. If the physical block has keys, the key immediately precedes the data, and GETNONVS sets IOCDLN to the length of the physical block plus the key.

Module: IDCIO01

Procedure: IDCIOGT

5. IDCIOGT puts a return code in register 15 and returns control to the module that issued the UGET.

Diagram 6.5 I/O Adapter – UPUT Macro



Extended Description for Diagram 6.5

Module: IDCIO01

Procedure: IDCIOPT

1. If entrance is from any module except UCOPY, processing continues with step 2. If entrance is from UCOPY, processing continues with step 3.
2. IDCIOPT uses the type identification to determine whether or not the record is a message. An omitted identification or an identification of zero indicates a data record. A non-zero value indicates a message is to be written. If the address for the IOCSTR is zero, or the open-for-processing flag, IOCMSGOP, is off, IDCIOPT issues a UABORT macro. If IOCPNM is zero, only one record is written with this UPUT, and the length of the record is assumed to be in IOCDLN. If IOCPNM is non-zero, one or more records are written with the UPUT. IOCDLN contains the total length of all the records, and each record is preceded by a two byte length field for that record. IDCIOPT sets IOCPNM to one if it was initially zero. For multiple inputs, IDCIOPT puts the length of the first record in IOCDLN, and IDCIOPT puts the address of the data for the first record in IOCDAD.

Module: IDCIO01

Procedure: PUTEXT

3. If the data set is user controlled, PUTEXT constructs an argument list. PUTEXT gives control to the user routine addressed in IOCXAD. If the data set is SYSPRINT on a TSO terminal, see Diagram 6.5.1 for more details. If the return code from the user routine is zero, PUTEXT increments the number of successful UPUTs in IOCRRN. If the return code is 12, PUTEXT turns off the open-for-processing flag, IOCMSGOP, so that no processing can be done against this data set. PUTEXT returns control to step 2 for the next record.

Module: IDCIO01

Procedures: IDCIOPT, PUTVSAM, VSAMERR, • CHANGE (**VS2.03.808 only**), PRINTMSG, PUTNONVS, IRSOSYN, PUTREP

4. IDCIOPT writes the record depending upon the access method for the data set:
   - For a VSAM data set if **VS2.03.808** is installed, PUTVSAM checks to see if IOCMACER is set by the caller of UPUT; if so, PUTVSAM issues the ERASE macro with a pointer to the RPL. If this case, a UGET for update must previously have been issued by te caller. If IOCMACEN is set by the UPUT caller, PUTVSAM issues the ENDREQ macro with a pointer to the RPL.

Again, with **VS2.03.808** only, if any IOCSTR flag indicating a change in processing modes has been set by the caller, the CHANGE procedure makes the appropriate change in the RPL. The following IOCSTR settings specified by the issuer of UPUT are reflected in the RPL:

IOCSTR	RPL OPTCD=
IOCCHPSQ	SEQ
IOCCHPDR	DIR
IOCCHPSK	SKP
IOCCHPCR	ADR
IOCCHPBK	CNV
IOCCHPKG	KGE
IOCCHPKE	KEQ
IOCCHPUP	UPD
IOCCHPNU	NUP

CHANGE will set all change processing flags to '0', and the IOCSTR will be changed to reflect the new processing option.

PUTVSAM puts the record length and address in the RPL. IF IOCMACRR='1'. indicating a PUT to an RRDS, the RPLARG field in the RPL is set to the address of IOCREL. If OPTCD=CNV.DIR, RPLARG field is set to the address of IOCRBA.

If user buffers are specified, (IOCMODUB=1), the output area address in the RPL is obtained from IOCWORK rather than IOCDAD.

PUTVSAM issues a PUT macro to write the record. The record may be a logical record or a control interval. If the return code from the PUT is zero, PUTVSAM increments the number of successful UPUTs in IOCRRN. If the return code is non-zero, VSAMERR gets the error code from the RPL. If the error code indicates a logic error, VSAMERR determines if it is a duplicate record or a record-out-of-sequence, PRINTMSG writes the appropriate message. Otherwise, the error is assumed to be an I/O error, and VSAM has formatted an error message in the message area. PRINTMSG writes the message. If the error code for an I/O error is greater than 4, VSAMERR turns off the open-for-processing flag, IOCMSGOP. However, if **VS2.03.808** is installed, the 'suppress messages' flag, IOCMSGSM, set the UPUT caller is checked. If it is on, the call to VSAMERR to print logical error messages is bypassed.

PUTVSAM will provide replace processing under the following conditions:

- A return code from PUT indicating a logical error (08)
- RPL feedback code indicating duplicate record.

Replace processing specified by caller (IOCMODRP=1)

In the PUTREP routine, which handles replace processing, IOCWKA is checked to determine if an input work area exists. If not, a UGPOOL is issued to obtain an input work area. The RPL is modified to permit update processing. A GET for update is issued followed by a PUT. The IOCSTR for the PUT will reference the address of the original PUT record in IOCDAD. After the PUT, the RPL is reset for no update processing. PUTVSAM returns control to step 2 for the next record.

For a non-VSAM data set open for QSAM, the record is written as a logical record. For a non-VSAM data set open for block processing, PUTNONVS issues a WRITE macro. For the SYSPRINT data set, PUTNONVS compares the record length to the maximum and truncates the record if it is longer than the maximum. If the record format is fixed and the record length is not exactly the logical record length, PRINTMSG writes an error message, and the record is ignored. For variable length records, PUTNONVS sets the length of the record and record descriptor word in the DCB. If the output IOCSEX indicates export of a catalog recovery area (IOCRCVM='1'), a 4-byte header must be prefixed to each record of the portable data set. The header consists of 4 bytes of binary zeros. However, if the data-length (IOCDLN) and the data pointer (IOCDAD) in the IOCSTR are both zero, then the 4-byte "header" is written as a software end-of-file and consist of X'00008000'. PUTNONVS issues a PUT macro in the LOCATE mode to write the record. If an I/O error is detected during the PUT macro, IRSOSYN issues SYNADAF and SYNADRLS macros to get a message formatted by QSAM. If the error option field in the DCB does not indicate that errors are to be accepted, IRSOSYN turns off the open-for-processing

flag, IOCMSGOP. PUTNONVS increments the number of successful UPUTs if an error message ID has not been set by an error exit routine. PUTNONVS moves the record into the area returned by the PUT macro, and, if necessary, PUTNONVS builds a record descriptor word RDW. PUTNONVS returns control to step 2 for the next record.

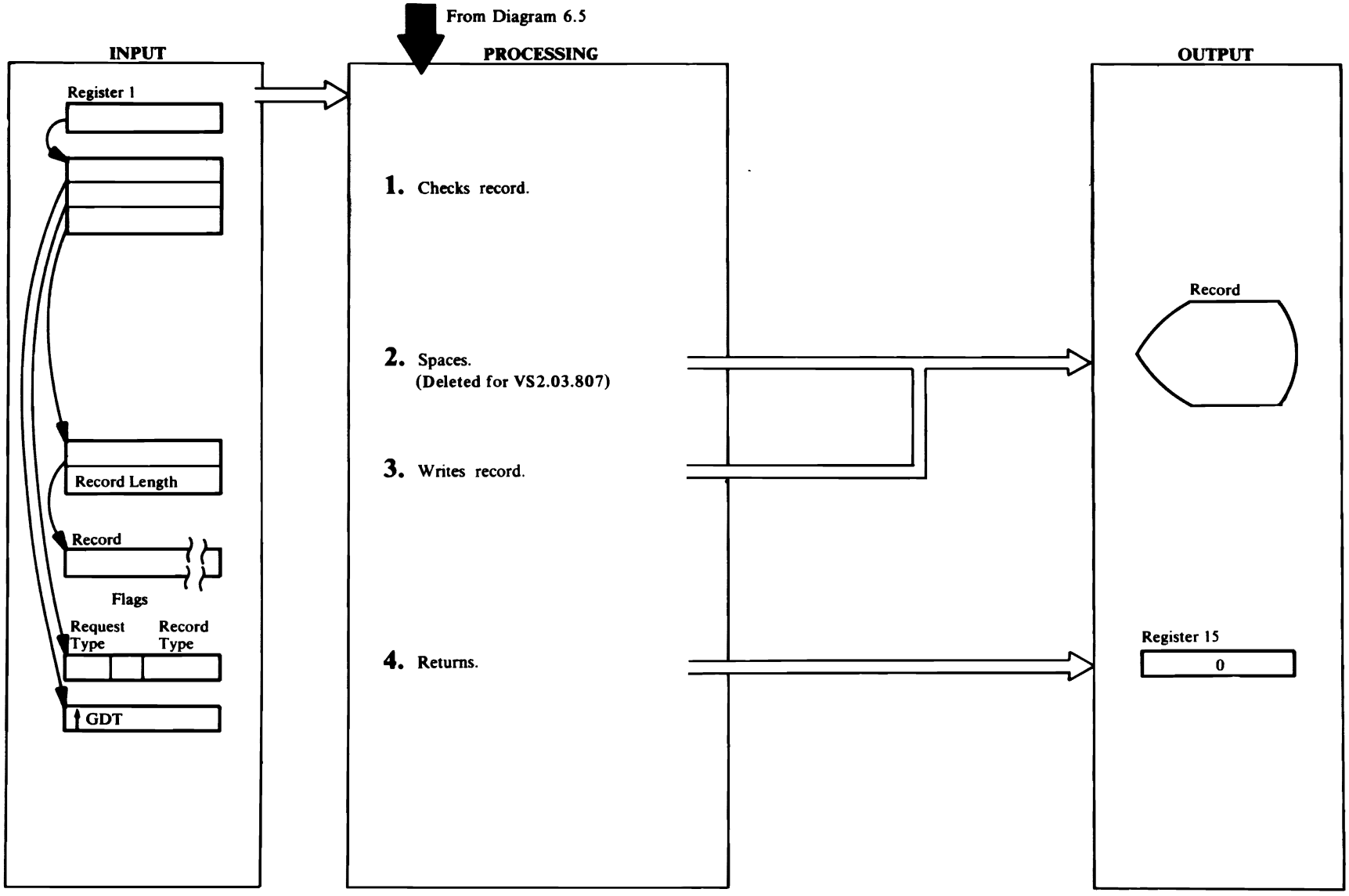
For a non-VSAM data set open for BSAM or BPAM, the record is written as a physical block. If the data set is open for update, the READ DECB, which was saved in the work area whose address is in IOCDEC, is used for the WRITE. If the length of the physical block is longer than the maximum block length, the block is not written; an error message is written on SYSPRINT, and control goes to step 5. PUTNONVS puts the length of the key, if any, and the length of the physical block in the DCB. The key must be the first part of the physical block. If the record format is fixed or variable, PUTNONVS issues a WRITE macro with the 'S' option to write the physical block. If the record format is undefined, PUTNONVS issues a WRITE macro without the 'S' option to write the physical block. PUTNONVS issues a CHECK macro to test the WRITE. If the Operating System End-of-Volume routines detect an error condition during End-of-Volume, IROEVAB turns off the IOCMSGOP, formats a message, and gives a code to the End-of-Volume routines that causes the End-of-Volume routines to ignore the error and return to the I/O Adapter. If the CHECK macro detects an I/O error, IRSOSYN issues SYNADAF and SYNADRLS macros to format a message. If the DCB error option field indicates errors are not to be accepted, IRSOSYN turns off the open-for-processing flag. PUTNONVS returns to step 2 for the next physical block.

Module: IDCIO01

Procedure: IDCIOPT

5. When all the records have been written, IDCIOPT puts a return code in register 15 and returns control to the module that issued the UPUT.

Diagram 6.5.1 I/O Adapter SYSPRINT on A TSO Terminal



Extended Description for Diagram 6.5.1

Module: IDCIO04

Procedure: IDCIO04

1. IDCIO04 checks the record type in the last two bytes of the flags field. Refer to the "Program Organization" chapter for more information on arguments passed to a user I/O routine. If the record type is less than zero, the line is not printed on the terminal, IDCIO04 sets the return code to zero, and control goes to step 3. If the record type is zero or greater, IDCIO04 checks the ASA control character in the first byte of the record to write. If the ASA control character indicates "skip to a new page," the record is not written on the terminal, IDCIO04 sets the return code to zero, and control goes to step 4. If the ASA control character does not indicate "skip to a new page," the record will be written on the TSO terminal.

Module: IDCIO04

Procedure: IDCIO04

(Without VS2.03.807)

2. IDCIO04 checks the ASA control character for the number of lines to space. Single spacing is the default if the ASA control character does not indicate double or triple spacing. IDCIO04 issues a PUTLINE macro to write a blank line for each line to be skipped. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor* for more information about the PUTLINE macro and writing to a TSO terminal.

Module: IDCIO04

Procedure: IDCIO04

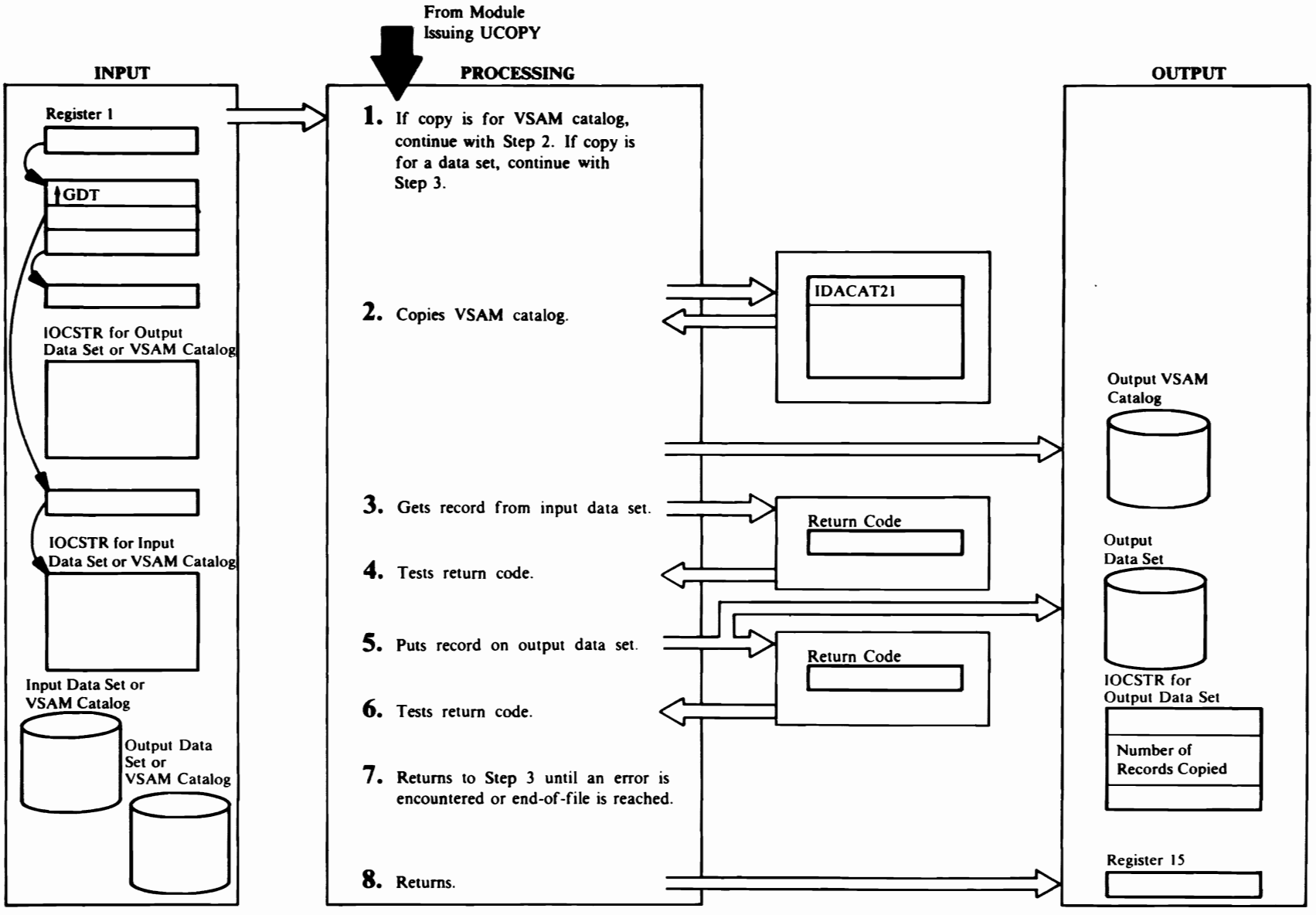
3. IDCIO04 again checks the record type. If the record type is zero, the line is written as a data line. IDCIO04 uses the data form of the PUTLINE Parameter List when the PUTLINE macro is issued. If the record type is greater than zero, the line is written as a message line. IDCIO04 uses the message form of the PUTLINE Parameter List when the PUTLINE macro is issued. If the message is multilevel, IDCIO04 adds the MULTLVL option to the message form of the PUTLINE macro. (**Note:** If **VS2.03.807** is installed, IDCIO04 places a 't' at the end of the primary segment and, if the message is multi-segment, IDCIO04 assembles the segments into a single line by removing the headers from the second and subsequent segments.) IDCIO04 issues a PUTLINE macro to write the line on the TSO terminal. If the return code from the PUTLINE macro is 16, PUTLINE cannot get enough storage; IDCIO04 issues a UABORT macro with a UABORT code of 28. If the return code is not 16, IDCIO04 sets its return code to zero.

Module: IDCIO04

Procedure: IDCIO04

4. IDCIO04 puts its return code in register 15 and returns control to Diagram 6.5.

Diagram 6.6 I/O Adapter UCOPY Macro



Extended Description for Diagram 6.6

Module: IDCIO01

Procedure: IDCIO01

1. If IOCMACCC in the input IOCSTR is on, UCOPY is to copy a VSAM catalog; control continues with step 2. If IOCMACCC is off, UCOPY is to copy a data set; control continues with step 3.

Module: IDCIO01

Procedure: COPYCAT

2. COPYCAT issues a LOAD macro to load the VSAM copy catalog module, IDACAT21. COPYCAT gives control to IDACAT21 to copy the input VSAM catalog to the output VSAM catalog. When IDACAT21 returns, COPYCAT checks the return code from IDACAT21. If the return code is zero, COPYCAT sets the UCOPY return code to zero. If the return code is non-zero, COPYCAT sets the UCOPY return code to four, and writes an error message with the error code in register 0. Control goes to step 8.

Module: IDCIO01

Procedure: IDCIOCO

3. IDCIOCO obtains a record from the input data set by calling the procedures used for a UGET macro. The UGET procedure returns control to this point in the UCOPY routine. Arguments to the UGET procedures are set up as though a UGET had been issued.

Module: IDCIO01

Procedure: IDCIOCO, PRINTMSG

4. IDCIOCO tests the return code from the UGET procedures. If the return code is zero, the UGET procedure read the record successfully. If the output IOCSTR indicates RRDS (IOCMACRR=1) and the input IOCSTR indicates nonRRDS (IOCMACRR=0), an incremental counter is maintained. This counter is incremented by one each time a record is successfully retrieved from the nonRRDS. This count is placed in the output IOCREL prior to UPUTing the record. If the return code indicates end-of-file, the copying stops. If the return code indicates an error, IDCIOCO increments the number of errors for UCOPY. If the UGET routine has set a message, PRINTMSG writes it. Processing continues with the next input record if the number of errors is less than four, and the open-for-processing flag, IOCMSGOP, is on. If the number of errors is four or IDCMSGOP is off, IDCIOCO turns off IOCMSGOP, and control goes to step 6.

Module: IDCIO01

Procedure: IDCIOCO

5. IDCIOCO moves the length and address of the record just read from the input IOCSTR to the output IOCSTR. If the input and output IOCSTR both indicate RRDS, IOCREL is moved from the input IOCSTR to the output IOCSTR before issuing the UPUT. This will result in exact recreation of the correlation between the relative record number in the input and output RRDS.

   If the input IOCSTR indicates IOCMACRR='1' and the input IOCSEX indicates IOCMODXM='1', this is an EXPORT of an RRDS. It is required that the relative record number be carried in the portable data set. The relative record returned in IOCREL when the record is retrieved is placed in the 4-byte field immediately preceding the record. The RRDS record plus the 4-byte field is then written to the portable data set.

   If the output IOCSTR indicates IOCMACRR='1' and the output IOCSEX indicates IOCMODXM='1', this is an IMPORT of an RRDS. Records retrieved from the portable data set have the relative record number prepended to the RRDS record. This relative record number is moved to the output IOCREL. The address of the beginning of the RRDS record is set to its logical beginning (the address of the retrieved record +4) and the length of the record to be written is reduced by 4 bytes. IDCIOCO writes the record by calling the same procedures used for the UPUT macro. IDCIOCO sets up the arguments to the procedures as though a UPUT had been issued. The UPUT procedure returns control to this point in the UCOPY routine.

Module: IDCIO01

Procedure: IDCIOCO

6. IDCIOCO tests the return code from the UPUT procedures. If the return code is zero, the UPUT procedure wrote the record successfully. If the return code indicates an error, IDCIOCO increments the number of errors for the UCOPY.

Module: IDCIO01

Procedure: IDCIOCO, PRINTMSG

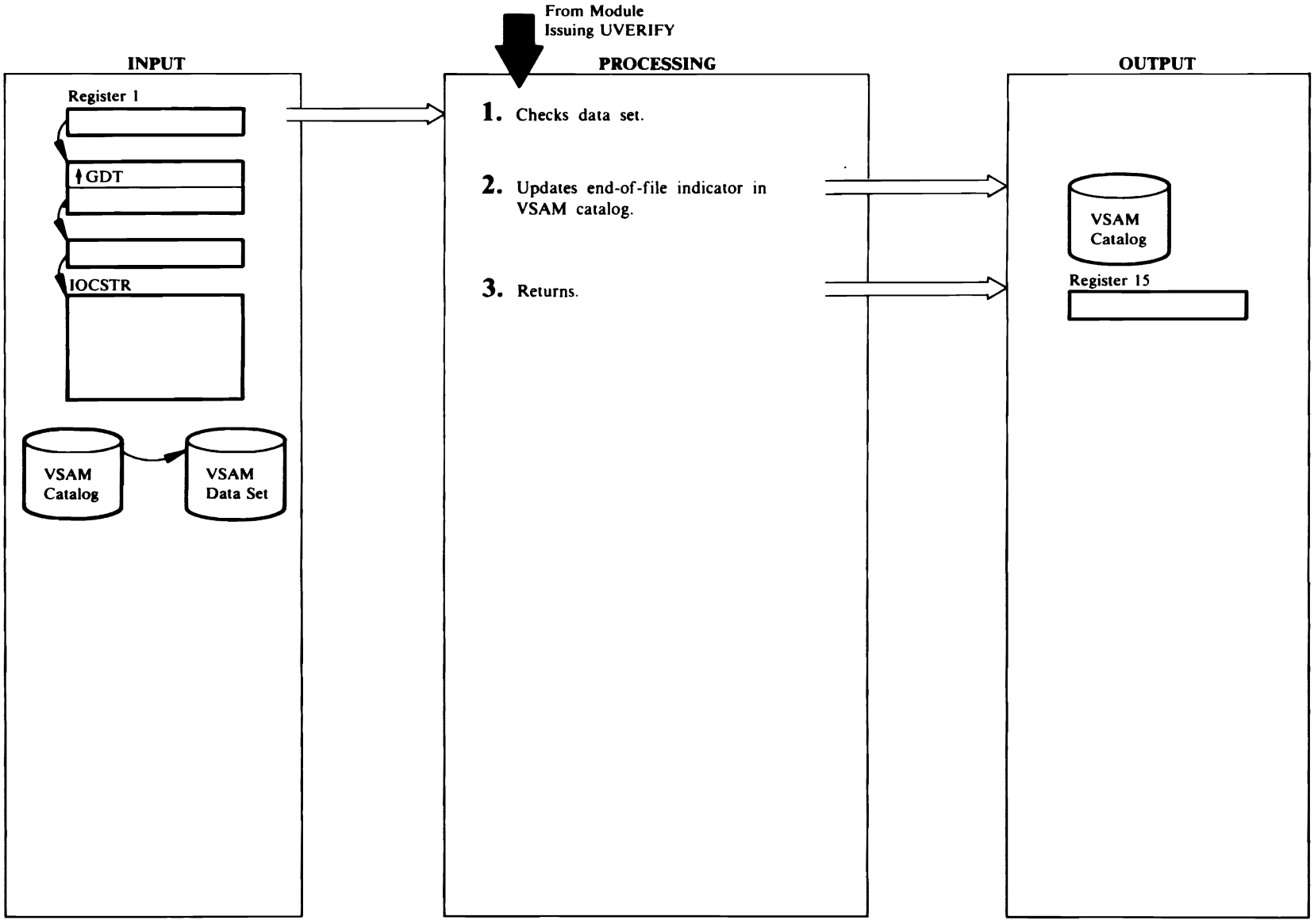
7. Control returns to step 3 if the number of errors is less than four; the open-for-processing flag, IOCMSGOP, is on. PRINTMSG writes a message if the message has been formatted. If the number of errors is four, IDCIOCO turns off IOCMSGOP, and control goes to step 6.

Module: IDCIO01

Procedure: IDCIOCO

8. IDCIOCO puts a return code in register 15 and returns control to the module that issued the UCOPY.

Diagram 6.7 I/O Adapter UVERIFY Macro



Extended Description for Diagram 6.7

Module: IDCIO01

Procedure: IDCIOVY

1. The second argument is assumed to be a valid IOCSTR address. The UVERIFY does not continue if:
   - The data set is not VSAM.
   - No RPL has been built for a VSAM data set.
   - The VSAM data set is not open.

   No error message is written for the last two conditions because messages have been written at open.

Module: IDCIO01

Procedure: IDCIOVY

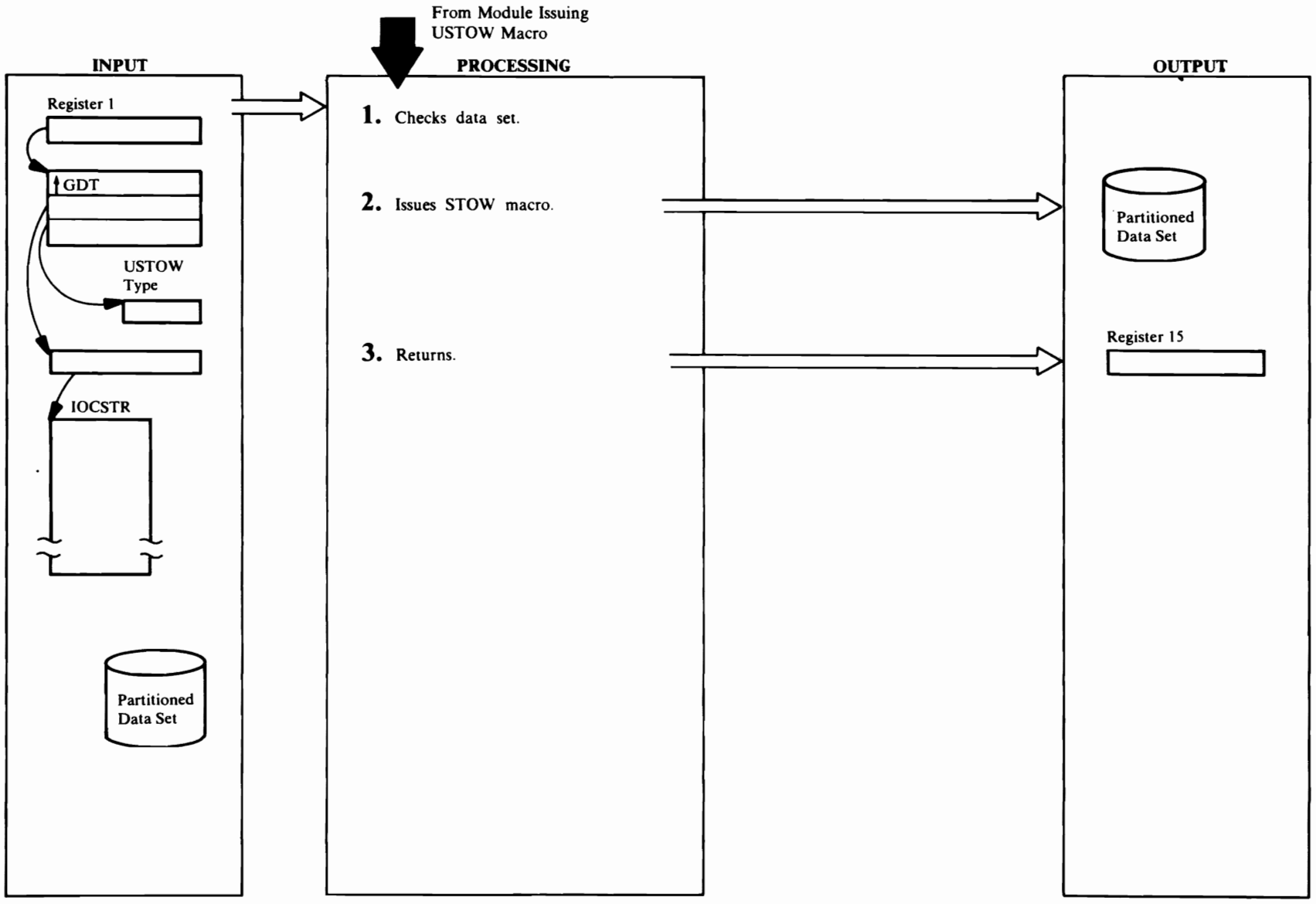
2. IDCIOVY issues a VERIFY macro.

Module: IDCIO01

Procedure: IDCIOVY, VSAMERR, BLDOCMSG, PRINTMSG

3. If the return code is not zero, VSAMER obtains the error code from the RPL. If the error is a logic error, PRINTMSG writes a message. If the error is an I/O error, VSAM has formatted a message in the message area and put a return code in the RPL. If the error code is not 4, which indicates that the error occurred in the data, VSAMERR turns off the open-for-processing flag, IOCMSGOP. IDCIOVY puts a return code in register 15 and returns control to the module that issued the UVERIFY.

Diagram 6.8 I/O Adapter USTOW Macro



Extended Description for Diagram 6.8

Module: IDCIO03

Procedure: STOWRTN

1. STOWRTN checks the IOCSTR for the partitioned data set. If the address for the IOCSTR is zero or if the open-for-processing flag, IOCMSGOP in the IOCSTR, is not on, STOWRTN issues a UABORT macro to terminate Access Method Services.

Module: IDCIO03

Procedure: STOWRTN

2. If the USTOW type is a 'D', a member of the partitioned directory is to be deleted. The address of the member name and the DCB for the partitioned data set are obtained from the IOCSTR. STOWRTN issues a STOW macro with the delete option.

   If the USTOW type is an 'R', a member of the partitioned directory is to be renamed. The address of the current member name, the new member name, and the DCB for the partitioned data set are obtained from the IOCSTR. STOWRTN issues a STOW macro with the rename option.

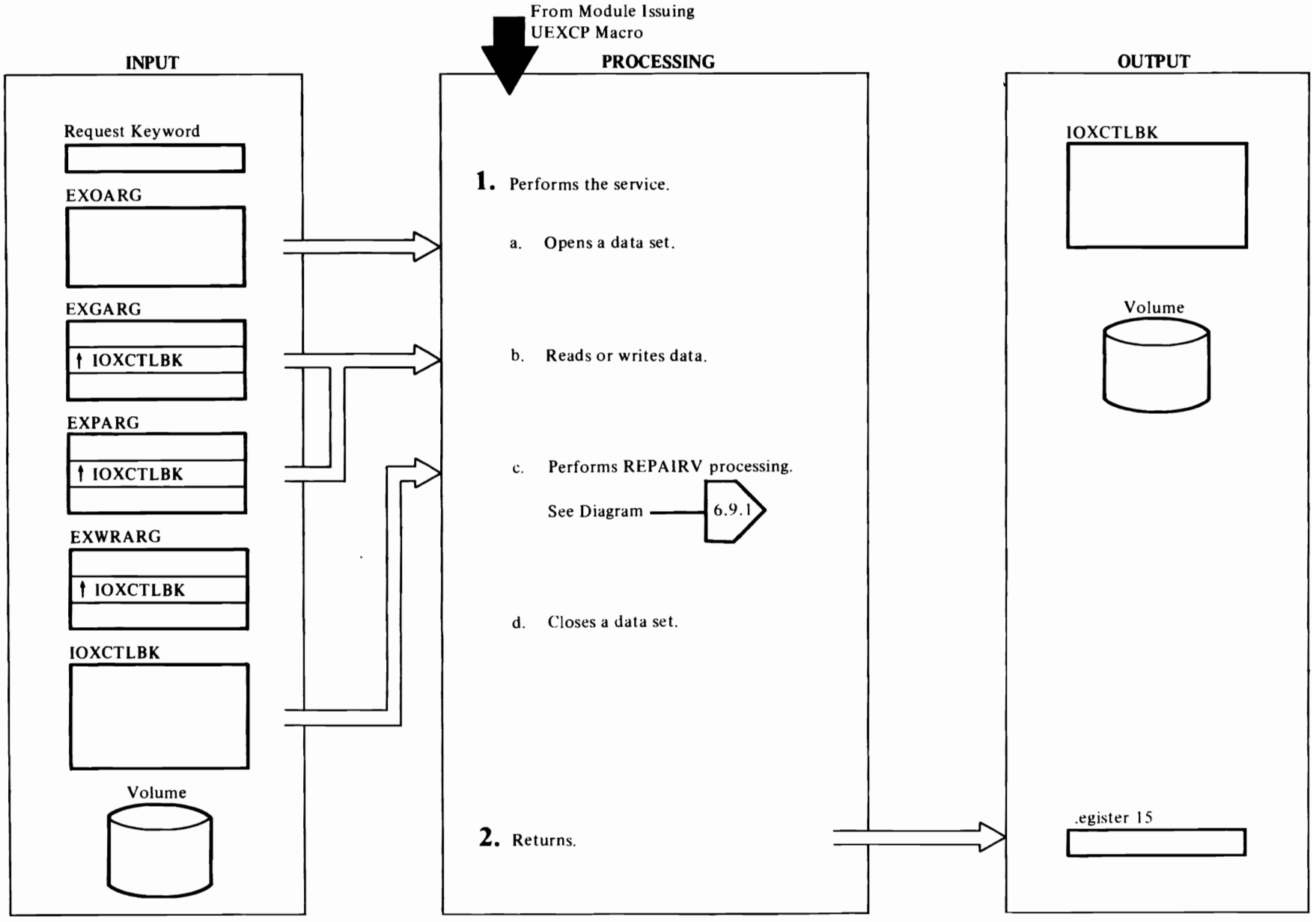
   STOWRTN checks the return code from the STOW macro. If the return code is zero, processing continues with step 3. If the return code is non-zero, a message is written. If the return code is 4, the name to be added to the partitioned data set directory already exists in the directory. If the return code is 8, the member name to be renamed could not be found. If the return code is 12, a permanent I/O error occurred, a return code of 20 indicates the DCB is not open, and a return code of 24 indicates insufficient storage to process the STOW.

Module: IDCIO03

Procedure: STOWRTN

3. STOWRTN puts the return code from the STOW macro in register 15 and returns control to the module that issued the USTOW.

Diagram 6.9 I/O Adapter – UEXCP Macro



Extended Description for Diagram 6.9

Module: IDCIO05

Procedure: IDCIO05, OPENPROC, OPNNEW, OPNTAB, OPNLAB, OPNPASS, OPNVTOC, PUTGET, BLDBLK, READJFCB, BLDCCWP1, BLDCCWP2, BLDCCWG, ISSUEXCP, CLOSEPRC, CLOSESTD, CLOSENEW, SYNAD, CHECKOPN, RETRY, OCABEND

1. IDCIO05 checks the request keyword to determine the type of service to perform:

   'OPEN' indicates a request to open a data set.

   'OPENR' indicates a request to open a data set, VTOC, VTOC header, or staging pack for REPAIRV processing.

   'PUT' indicates a request to write data.

   'GET' indicates a request to read data.

   'CLOSE' indicates a request to close a data set.

   'READCNT' indicates a request to read the count field of all data records on a track.

   'READKD' indicates a reques to read the key and data fields of all data records on a track.

   'SPACCR' indicates a request to bypass the defective count of a data record, so that the rest of the record can be read.

   'WRITEREC' indicates a request to rewrite the key and data fields of a data record.

   'FWRITE' indicates a request to write the count, key, and data fields of one or more new data records.

   a. The EXOARG contains the type of open to be performed:

   Open to write an uninitialized volume—OPNNEW issues SVC 82 to create a DEB to allow writing on the entire volume. Because a VTOC does not exist for a new volume, the standard OPEN cannot be used.

   Open to read the MSC tables—OPNTAB issues an OPEN macro to open the data set for output. OPNTAB puts the DEB extents in the caller's area.

   Open to read and write the volume label—OPNLAB modifies the JFCB (Job File Control Block) and issues an OPEN macro, TYPE=J, to open the VTOC for output. OPNLAB issues a MODESET macro to enter supervisor state, modifies the extents in the DEB to allow access to cylinder 0, then issues a MODESET macro to return to problem-program state.

   Open to verify passwords—OPNPASS sets the address of IDCIO05's retry routine in STAEPARM and, if a recovery environment has not been established, issues a USTAE macro to establish it. The retry routine gets control if the operator doesn't supply the correct password. If **SU 5752-824** is installed, the retry routine gets control if the operator doesn't supply the correct password *or* if RACF authorization is invalid. OPNPASS modifies the volume serial number in the JFCB, if requested, and issues an OPEN macro, TYPE=J, for input or output, as requested. OPNPASS then issues a USTAE macro to cancel the recovery environment unless it had already been established.

   Open to read VTOC DSCBs—OPNVTOC issues the USTAE macro to establish a STAE/ESTAE recovery environment in order to recover from an OPEN ABEND. OPNVTOC initializes the open parameter list, the DCB, and the JFCB. (The JFCB is modified as follows: the data set name field is set to X'04's and the caller's volume serial number is inserted in the volume serial number field in the JFCB). OPNVTOC then issues OPEN TYPE=J to open the VTOC. If a return code of zero is returned, the CHECKOPN procedure is called to make further checks, and the DCBOFLWR bit is set to prevent the CLOSE SVC from writing an EOF when the VTOC is closed. OPNVTOC then reissues the USTAE macro with CANCEL option to cancel the recovery environment.

   For each type of open, the address of the IOXCTLBK is returned in the caller's area.

   b. BLDCCWP1 builds CCWs to write "COUNT KEY DATA" for a write request on a new volume. Otherwise, BLDCCWP2 builds CCWs to write "KEY DATA."

      BLDCCWG builds CCWs to read "KEY DATA " for a read request.

      ISSUEXCP issues an EXCP macro to handle the required I/O and issues a WAIT macro for the I/O to complete.

   c. For a DCB opened for a new volume, CLOSENEW builds the SV82LISt and issues SVC 82 to free the DEB; otherwise, CLOSESTD issues a CLOSE macro. CLOSEPRC issues a UFSPACE macro to free the IOXCTLBK.

Module: IDCIO05

Procedure: IDCIO05

2. If an error occurred, a message is written with a UPRINT macro. IDCIO05 returns to the caller with a return code in register 15.

Diagram 6.9.1 I/O Adapter - UEXCP Macro for REPAIRV Processing

From Diagram 6.9

INPUT

GDT

EXWRARG

PROCESSING

1. Performs the requested REPAIRV processing:

   READCNT – reads the count field of each data record on a track.

   OPENR – opens a data set, VTOC, VTOC header, or a staging pack.

   READKD – reads the count, key, and data fields of each data record on a track.

   SPACCR – bypasses a defective count field so that the key and data fields of a record can be read.

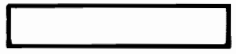
   WRITEREC – rewrites the key and data fields of a record.

   FWRITE – writes one or more new records on a track.

OUTPUT

EXWRARG

Register 15



Extended Description for Diagram 6.9.1

Module: IDCIO05

Procedure: OPENR, READCNT, READKD, SPACCR, WRITEREC, FWRITE, SETCCW, SETKDCCW, ADJCCW, OPNVTH

Open for REPAIRV processing— OPENR can be used to open a data set or a VTOC for REPAIRV processing, and to make an entire pack look like a VTOC for REPAIRV staging error processing. OPENR calls BLDBK to initialize IOXCTLBLK and read the JFCB. For non-VTOC header processing, an OPEN, TYPE=J, macro is issued to open the data set or VTOC for EXCP processing. OPENR calls CHECKOPN to ensure the open was successful. If a staging pack is to be opened, OPENR modifies the DEB to restrict access to the requested extent.

For VTOC header or VSAM data set processing, OPNVTH is called to do a pseudo OPEN (SVC 82) against the impaired VTOC header, and the DEB extent is modified to limit access to the first track of the VTOC.

When a data set, VTOC, VTOC header, or staging area is opened for REPAIRV processing, the REPAIRV utility accesses the data set using the following keywords. IDCIO05 checks the request keyword to determine the type of service to perform:

READCNT: READCNT reads the count field of each record on the track specified with cylinder EXCC and track EXHH. The count fields are read into the buffer contained in RIOAREA (pointed to by EXWIOAR). The number of count fields read (excluding the count field of record R0) is saved in EXCCWCNT. If READCNT is being called for hte first time, READCNT calls SETCCW to build a CCW channel program that reads the count fields. Next, READCNT puts the caller— specified seek address and CCW channel program address in the IOB. READCNT then calls ISSUEXCP to execute the channel program and check for successful completion. Before returning to the caller, READCNT determines the record number of the last count field read and saves it in EXCCWCNT.

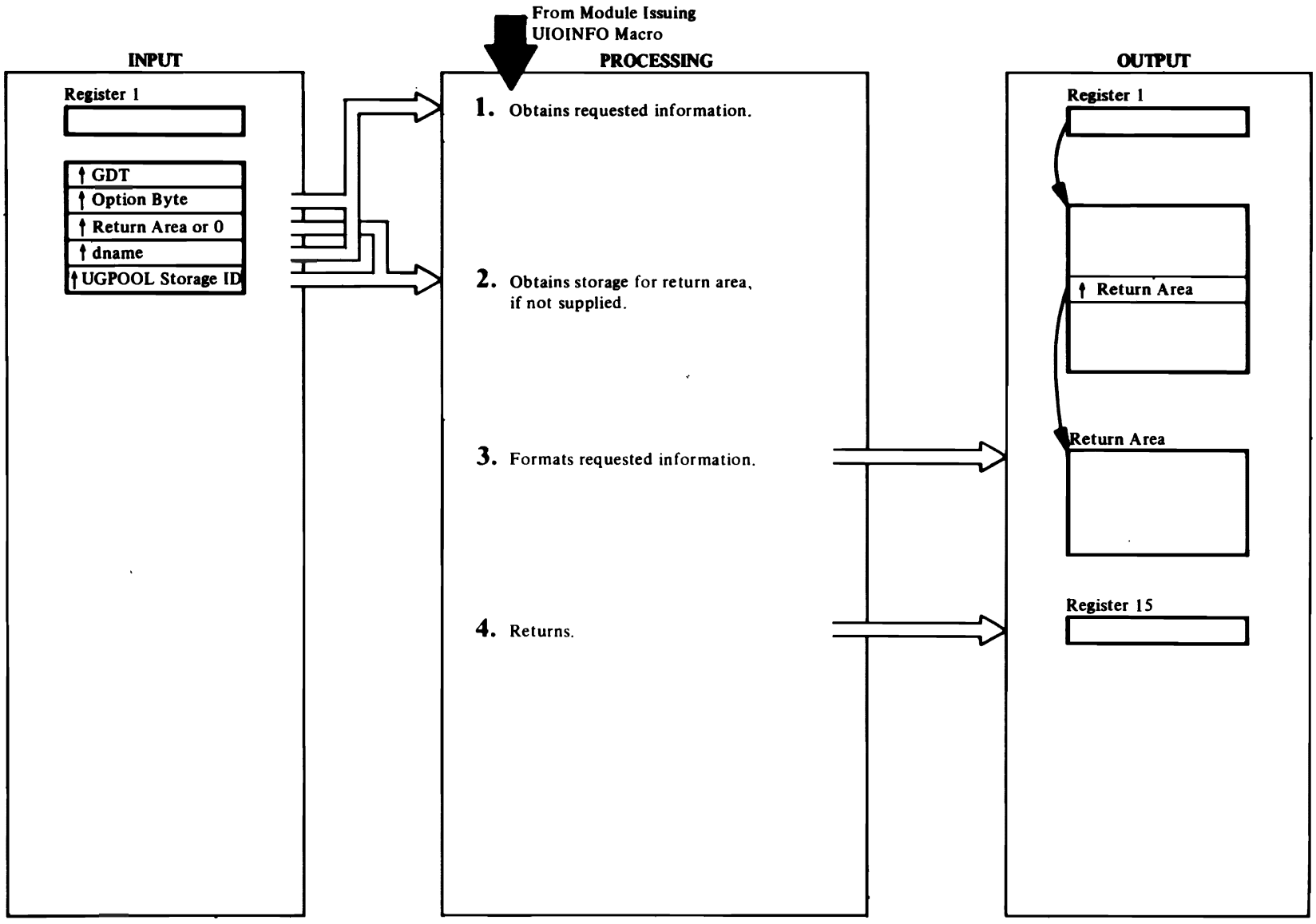
READKD): READKD reads the count, key, and data fields of each record on the track specified with cylinder EXCC and track EXHH. If the count field of a record is defective, only the key and data fields are read. The length of the key and data fields is obtained from the count fields read into RIOAREA buffer with READCNT (if a count field is defective, the key and data length can be obtained from the previous Space Count CCW.) READKD reads the count field, key field (if it exists), and the data field into the buffer in the RIOAREA (pointed to by EXWIOAR). If READKD is being called for the first time, READKD calls SETKDCCW to build a CCW channel program that reads the count, key, and data fields of each record on a track. If READKD is not being called for the first time, READKD calls ADJCCW to modify the previously built CCW channel program so that it will begin reading at the point where the previous READKD request stopped. Next, READKD puts the caller—specified seek address and CCW channel program address in the IOB. READKD then calls ISSUEEXP to execute the channel program and check for successful completion. READKD determines the number of CCWs actually executed and saves it in EXCCWCNT (ADICCW uses this value when READKD is next issued.) Before returning to the caller, READKD determines the location of each data field and saves this in LOCTAB (pointed to be EXLOCPTR).

SPACCR: SPACCR adjusts a CCW channel program built for READCNT to bypass a defective count field so that the record's key and data fields can be read. If the request is to bypass record R0, SPACCR changes the channel program to search for record R0 (SEARCH ID EQUAL) instead of to read it. If the request is to bypass a record other than record R0, SPACCR changes the channel program so that the SPACE COUNT command replaces the READ COUNT command for the record whose count field is defective. EXCCWCNT contains the record number whose count field is defective.

WRITEREC: WRITEREC writes the data from the buffer in WIOAREA (pointed to by EXWIOAR). The record whose key and data fields are to be rewritten is specified with cylinder EXCC, track EXHH, and record number EXRECNUM. EXRWFUN indicates the type of write operation to be performed: Write KD, to write the key and data fields of a record contained entirely on the track, or WRITE SPECIAL, to write the key and data fields of a record contained on the track and on one or more subsequent tracks. WRITEREC builds a CCW channel program to locate and rewrite the record, then to read the record with no data transfer. Next, WRITEREC puts the caller—specified seek address and CCW channel program address in the IOB. WRITEREC then calls ISSUEXCP to execute the channel program and check for successful completion.

FWRITE: FWRITE writes one or more records on a track, beginning with the caller-specified record and continuing for the caller-specified number of records, WIOAREA contains the number of records. WIOAREA contains the number of records to be written, as well as the data to be written for each record. The location of the first record to be written is specified with cylinder EXCC, track EXHH, and record number EXRECNUM. Each record is written using a WRITE CKD command to write the record's count, key, and data fields, unless EXRWFUN indicates the last record written should be a WRITE SPECIAL. If the requested starting record is record R0, a WRITE DATA command is used to write record RO's data field. FWRITE builds a CCW channel program to locate the point at which writing is to begin, to write each record, then to locate the starting point point again and read (with no data transfer) the records just written. Next, FWRITE puts the caller-specified seek address and CCW channel program address in the IOB. FWRITE then calls ISSUEEXCP to execute the channel program and check for successful completion.

Diagram 6.10 I/O Adapter – UIOINFO Macro



Extended Description for Diagram 6.10

Module: IDCIO03

Procedure: DSINFO

1. UIOINFO analyzes the option byte passed by the caller and determines what kind of information is required. (Bit 0 indicates device type; bit 1 indicates volume serial number(s); bit 2 indicates data-set name; bit 3 indicates suppress error message, bit 4 indicates format-4 time stamp.) To get a list of volume serial numbers or data-set name, UIOINFO must obtain job-control information. UIOINFO issues a RDJFCB macro using a QSAM DCB. If the return code from RDJFCB is nonzero, processing continues at step 4.

   If device type is requested, UIOINFO issues a DEVTYPE macro with the dname provided by the caller.

   If time stamp information is requested, UIOINFO issues the OBTAIN macro to read the format-4 DSCB. Old and new time stamps information is extracted from the DSCB.

Module: IDCIO03

Procedure: DSINFO

2. All of the information that UIOINFO obtained in step 1 is placed in IDCIO03's automatic storage work area. During this process, UIOINFO calculates the actual length of the data to be passed back to the caller. The caller can either pass a return area to UIOINFO or pass a UGPOOL storage ID. If the caller has passed a return area, UIOINFO checks the first two bytes (which indicate the length of the return area) to see whether the return area is large enough. If not, UIOINFO puts the size required in the third and fourth bytes, and processing continues at step 4.

   If the caller has passed a UGPOOL storage ID, UIOINFO issues a UGPOOL macro with that ID for the required storage. The caller is responsible for freeing this storage.

Module: IDCIO03

Procedure: DSINFO

3. UIOINFO formats the requested information in the return area.

Module: IDCIO03

Procedure: DSINFO

4. UIOINFO sets a return code in register 15 and returns to the caller.

Text Processor Visual Table of Contents

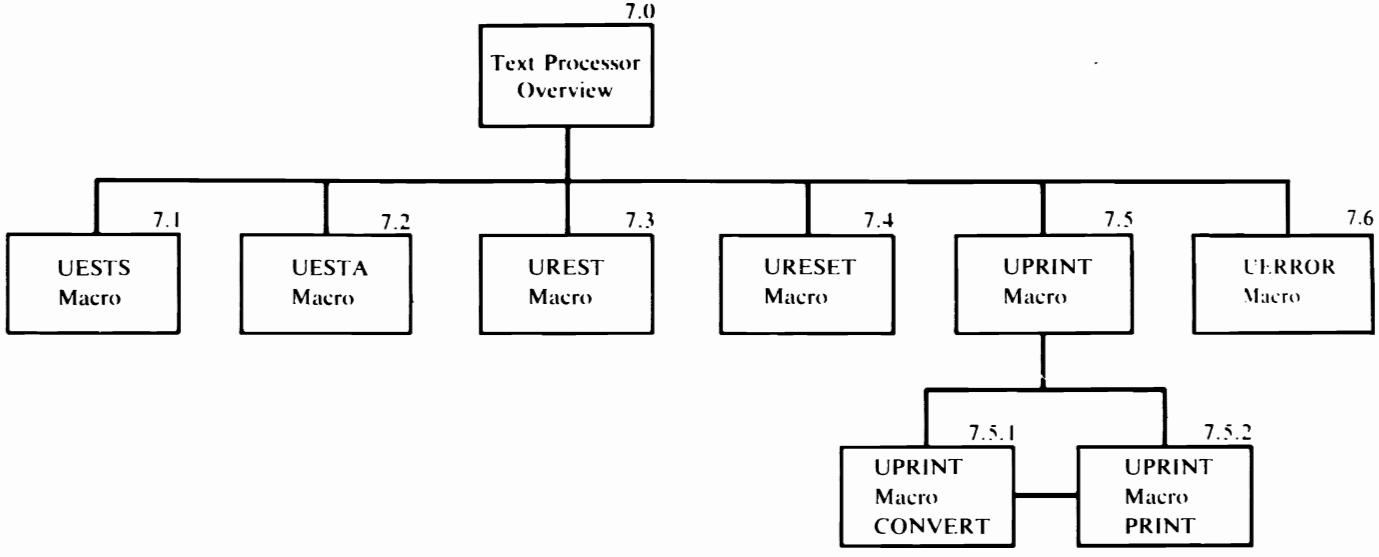
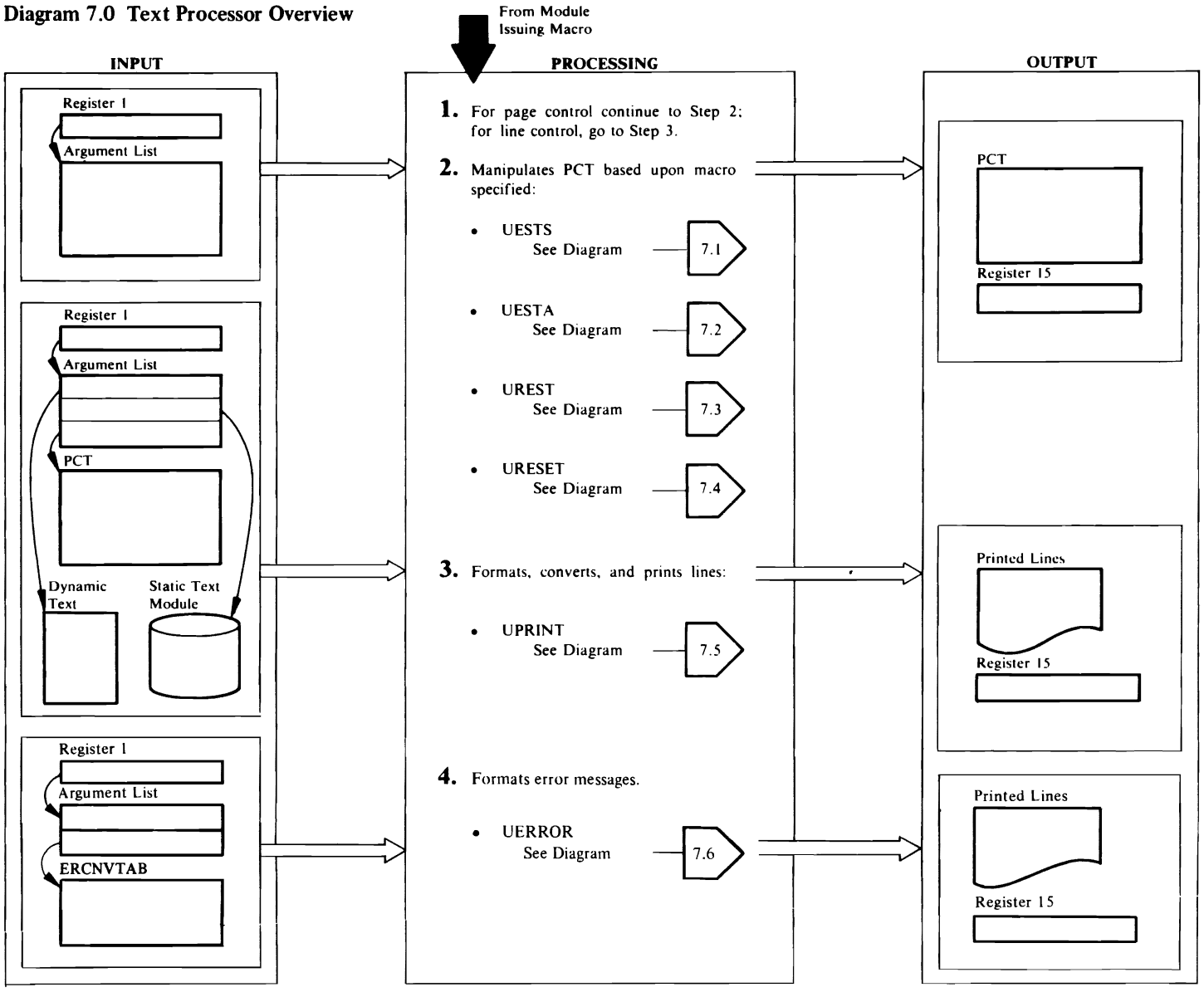


Diagram 7.0 Text Processor Overview



Extended Description for Diagram 7.0

Module: IDCTP01

Procedure: IDCTP01

1. For page control continue with step 2; for line control go to step 3, for error-message formatting go to step 4.
2. The page control macros use the argument list to change the Print Control Table, PCT. The page control macros are:

   UESTS, which establishes the PCT with data from a static text module.

   UESTA, which establishes the PCT with data from storage.

   UREST, which changes the PCT after a UESTS or UESTA macro has been issued.

   URESET, which sets Access Method Services defaults in the PCT.

   Each page printed by Access Method **Services has** three sections:

   1. 0 to 3 subtitles
   2. Header lines and Data lines
   3. 0 to 3 footing lines

   The title section contains the main title line and from zero to three subtitle lines. All lines in the title section are printed at the top of each page. The main title line is the first line on each page followed by subtitle lines. The header and data section contains any header and data lines. The header lines are kept in static text modules and are printed on page overflow conditions. The footing section contains from zero to three lines printed at the bottom of each page. At least one vertical space precedes them. More vertical spaces can appear depending upon the control characters in the first footing line. A new page results from any of the page control macros, a page eject on a line, or a request to print a line that would cause more lines on a page than specified. If there is not enough space on a page for all the header lines and one data line, none are printed. A page is ejected, and title and header lines are printed on the next page. Footing lines are always printed on each page. Vertical spacing is done before the a line is printed.

   The page control macros give the facility to change the following items in the PCT.

Item	Default	Limits
Main title line	1	1
Page number location	107	1 to line width minus field length
Time-of-day location	75	1 to line width minus 8 for field length
Date location	91	1 to line width minus 8 for field length
Subtitle line	no subtitles	0 to 3 lines
Footing line	no footing	0 to 3 lines
Line width	120	133 maximum
Page depth	54	999 maximum
Default vertical Space character	1 vertical space	1,2, or 3 vertical spaces
Translate table for print chain	Standard tables **Module:** IDCTP01	

Procedure: IDCTP01

3. The UPRINT macro formats data within a line, converts data to a printable form, and prints the line or lines. Refer to the "Diagnostic Aids" chapter for formatting examples. IDCTP01 uses the PCT to format the line and the page. The line to be printed is described by two kinds of input: static text and dynamic text. Static text—unchanging data and format structures—resides in a module called a static text module. Dynamic text is any changing data and format structures that reside in storage. Format structures, FMTLIST, describe how the line is to be formatted. The types of formatting are:

   Vertical spacing
   Inserting data into a line
   Extracting fields from a block of data in storage
   Extracting data from a static text module
   Defining default data
   Repeating any of the above actions

   The types of conversion are:

   Binary to hexadecimal
   Binary to hexadecimal with apostrophe
   Binary to dump
   Binary to decimal
   Packed decimal to unpacked decimal
   EBCDIC, no translation

   The types of vertical spacing are:

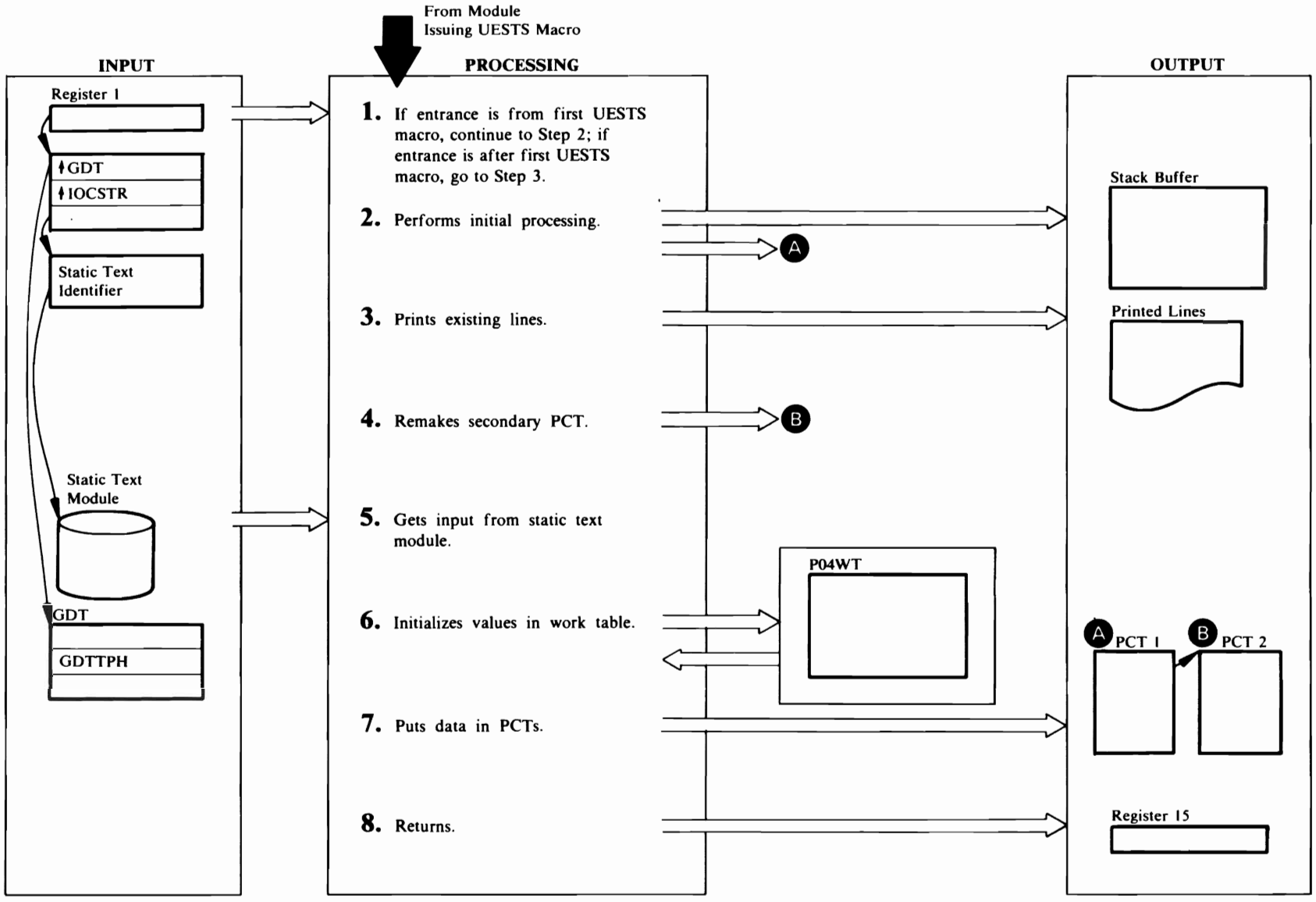
   Absolute spacing— The line is printed at a given line number on the page. If data has been printed at that line number, the page is ejected, and the line is printed at the first data line number on the next page. If the line number is within the title section or header lines, the line is printed at the line number immediately following the header lines. If the line number is within the footing section, the page is ejected, and the line is printed immediately following the header lines on the next page.

   Relative spacing— The line is printed at a number of vertical spaces counted from the last printed line. If there is not enough room on the page to print the line, the page is ejected, and the line is printed after the title section and header lines on the following page.

   Eject— The line is printed after the title section and header lines on the following page.

4. The UERROR macro verbalizes numeric catalog return codes and instigates multi-level message requests to the UPRINT macro. Formatting and printing of the multi-level message is handled by the UPRINT macro.

Diagram 7.1 UESTS Macro



Extended Description for Diagram 7.1

Module: IDCTP01 IDCTP04

Procedures: ESTSCONT INITPCT STACKPUT

1. If entrance is from the first UESTS macro, processing continues with step 2. If entrance is after the first UESTS macro has been issued, processing continues with step 3.
2. ESTSCONT tests the GDTTPM to determine if this is the first UESTS macro issued. If GDTTPH in the GDT is not zero, a PCT already exists, and control is given to step 3. The first time a UESTS macro is issued, the GDTTPH is zero which means that no PCT exists. When no PCT exists, INITPCT obtains and initializes a PCT. INITPCT issues a UGSPACE macro for the primary PCT. UGSPACE puts the address of the primary PCT in GDTTPH. (The GDT refers to the PCT as the Text Processor Historical Data Area.) The Text Processor (TP) uses two Print Control Tables—a primary PCT and a secondary PCT. Each PCT has the same fields. The primary PCT contains default values. INITPCT creates it during processor initialization, and deletes it at processor termination. It exists throughout Access Method Services processing. The secondary PCT contains current values which are different from the default values in the primary PCT. INITPCT creates it and deletes it many times during Access Method Services processing. The address of the secondary PCT is in the primary PCT. When the Text Processor uses a PCT, if the secondary PCT exists, it is used instead of the primary PCT.

   Rather than writing each line as it is completed, the Text Processor saves time by putting completed lines in an area of storage called the stack buffer. When the stack buffer is full, STACKPUT writes it. ESTSCONT issues a UGSPACE macro for storage for the stack buffer and puts the address of the stack buffer in the fields PCTBUF and PCTBNL in the primary PCT. ESTSCONT opens the System output data set with a UOPEN macro. Control is given to step 4.

Module: IDCTP04

Procedure: STACKFL

3. Because controls governing the writing (like page depth and line width) are changing, the lines formatted under the current control values must be written before the control values change. STACKFL writes the stack buffer with a UPUT macro.

Module: IDCTP04

Procedure: INITPCT

4. If a secondary PCT exists—that is, PCTSPP in the primary PCT is not zero—INITPCT releases the secondary PCT with a UFPOOL macro. INITPCT copies some data from the secondary PCT to the primary PCT before the secondary PCT is freed. INITPCT issues a UGPOOL macro for a secondary PCT. INITPCT sets the identification, PCTIDN, in the secondary PCT to 'PCT2', and sets the PCTSPP field to zero.

Module: IDCTP05

Procedure: IDCTP05

5. If a static text module is used once, it is likely that it will be used again on the next call to the Text Processor. Rather than loading and deleting a static text module each time it is used, the static text module is kept in storage until a different static text module is needed. The address of the static text module in storage is kept in PCTSTM in the PCT. The static text identification passed by the calling program to the Text Processor as input is used to reference the appropriate module. IDCTP05 concatenates the first three bytes of the static text identification with 'IDCTS' to form the module name. IDCTP05 compares the module name to the name of the static text module in storage in PCTSTM. If the names don't match, IDCTP05 deletes the static text module in storage with a UDELETE macro, and IDCTP05 loads the requested static text module with a ULOAD macro. IDCTP05 puts the name of the loaded module in PCTSTM and the address of the module in the field PCTSME in the PCT. If a secondary PCT exists, it is used; otherwise, the primary PCT is used.

   IDCTP05 uses the low-order byte of the static text identification as an index to obtain the correct static text entry. IDCTP05 copies the entry from the static text module into storage that IDCTP05 obtains with a UGSPACE macro. This is done so the static text entry is available if the static text module is deleted.

Module: IDCTP04

Procedure: P04SETUP

6. P04SETUP puts data from the static text entry into a work table. P04SETUP uses the work table to make the input from UESTS, UESTA, and UREST into the same format.

Module: IDCTP04

Procedure: PCTSETUP

7. PCTSETUP forces a page overflow so the next line will start on a new page. If no secondary PCT exists, PCTSETUP initializes the primary PCT with the minimum values needed to control a page, which are:
   - A translate table for a print chain
   - A page number increment
   - A line number where the first line is printed
   - A line number where the last line is printed

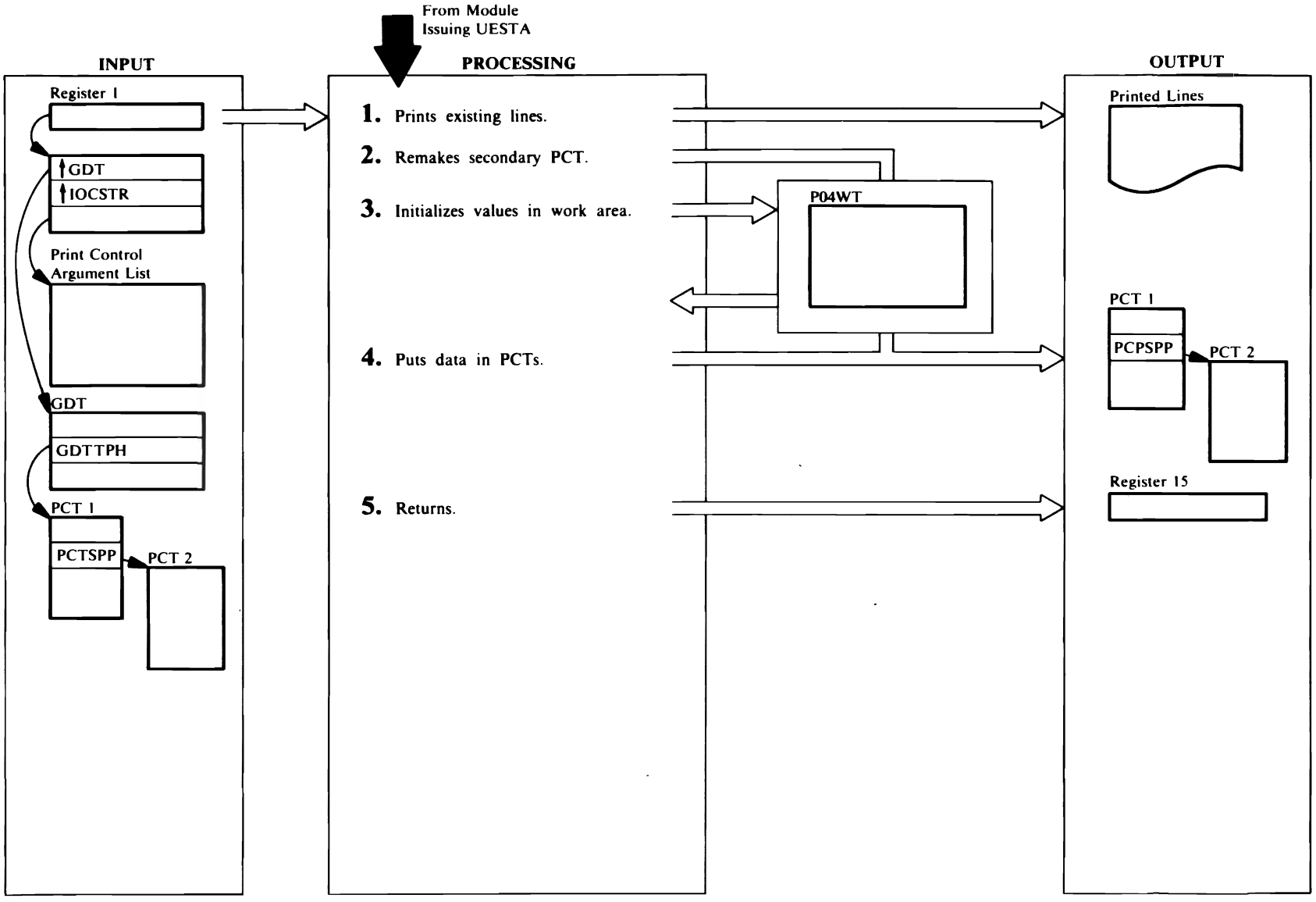
   For initializing either the primary PCT or the secondary PCT, PCTSETUP verifies the input data and puts it into the appropriate PCT.

Module: IDCTP04

Procedure: ESTSCONT

8. ESTSCONT deletes the storage for the static text entry with a UFSPACE macro. ESTSCONT puts a return code in register 15, and control returns to the module that issued the UESTS macro.

Diagram 7.2 UESTA Macro



Extended Description for Diagram 7.2

Module: IDCTP04

Procedures: ESTACONT INITPCT

1. A primary PCT must exist. If it does not, ESTACONT issues a UABORT macro. Because controls governing the writing (like page depth and line width) are changing, the lines formatted under the current control values must be written before the control values change. INITPCT writes the stack buffer with a UPUT macro.

Module: IDCTP04

Procedure: INITPCT

2. If a secondary PCT exists—that is PCTSPP in the primary PCT is not zero—INITPCT releases the secondary PCT with a UFPOOL macro. INITPCT issues a UGPOOL macro for a new secondary PCT. INITPCT sets the identification, PCTIDN, in the secondary PCT to 'PCT2', and INITPCT sets the PCTSPP field to zero. UGPOOL puts the address of the new secondary PCT in the field PCTSPP in the primary PCT. INITPCT copies all the data in the primary PCT into the secondary PCT. INITPCT copies some data from the secondary PCT to the primary PCT before the secondary PCT is deleted.

Module: IDCTP04

Procedure: P04SETUP

3. P04SETUP puts data from the input into a work table. PCTSETUP uses the work table to make the input from UESTS, UESTA, and UREST into the same format.

Module: IDCTP04

Procedure: PCTSETUP

4. PCTSETUP forces a page overflow so the next line will start on a new page. If no secondary PCT exists, PCTSETUP first initializes the primary PCT with the minimum values needed to control a page, which are:

   - A translate table for a print chain
   - A page number increment
   - A first page number
   - A line number where the first line is printed
   - A line number where the last line is printed

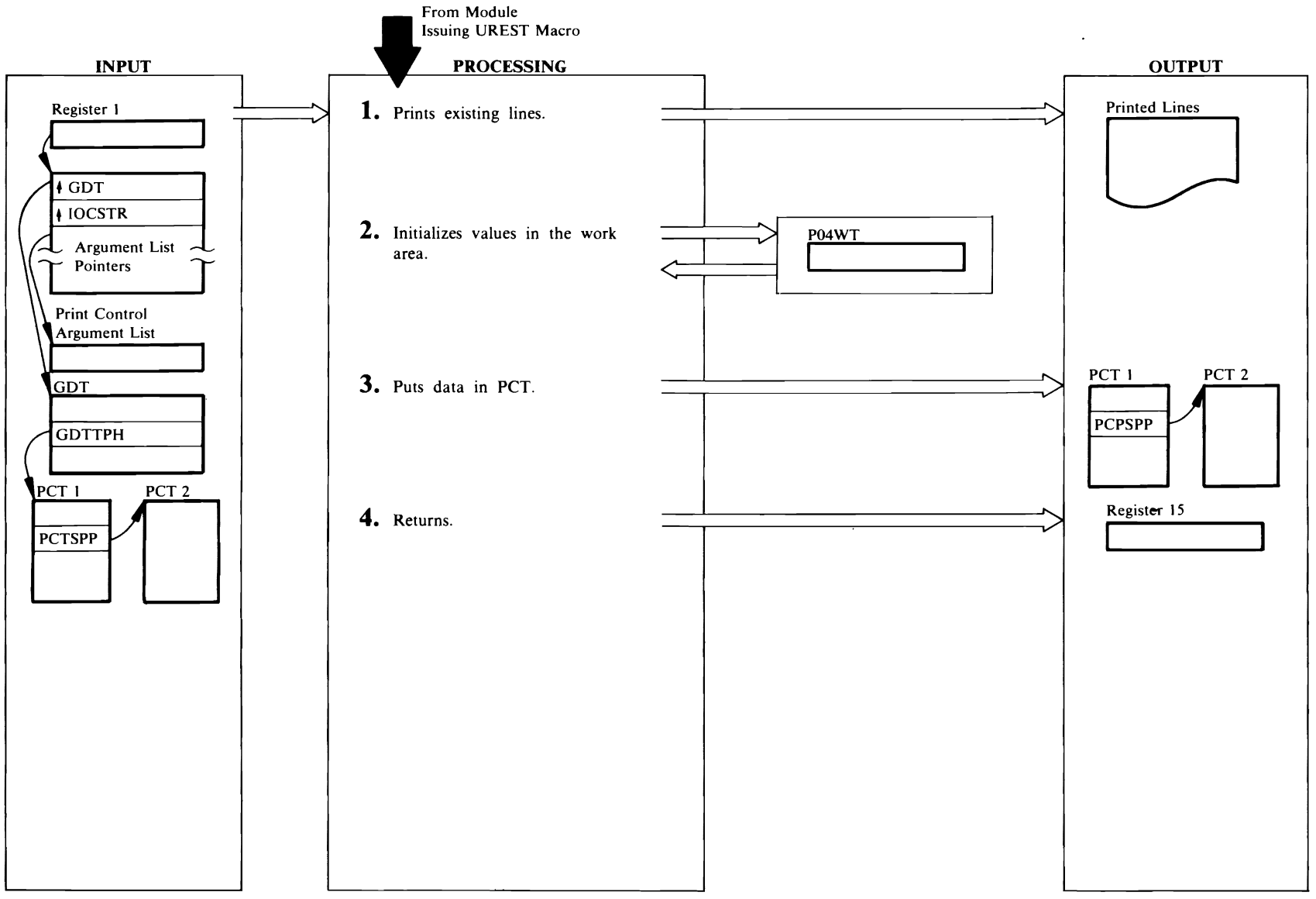
   For initializing either the primary PCT or the secondary PCT, PCTSETUP verifies the data in the work table and puts it into the appropriate PCT.

Module: IDCTP04

Procedure: ESTACONT

5. ESTACONT puts a return code into register 15, and control returns to the module that issued the UESTA macro.

Diagram 7.3 UREST Macro



Extended Description for Diagram 7.3

Module: IDCTP04

Procedure: RETCONT, STACKFL

1. A primary PCT must exist. If it does not, RESTCONT issues a UABORT macro. Because controls governing the writing (like page depth and line width) are changing, the lines formatted under the current control values must be written before the control values change. STACKFL writes the stack buffer with a UPUT macro.

Module: IDCTP04

Procedure: P04SETUP

2. P04SETUP puts data from the input into a work table, P04WT. PCTSETUP uses the work table to make the input from UESTS, UESTA, and UREST into the same format.

Module: IDCTP04

Procedure: RESTCONT PCTSETUP

3. The UREST macro allows the user to change any combination of the following:

   - Subtitle lines
   - Footing lines
   - Line width
   - Page depth
   - Default space character
   - Translate table
   - Starting page number

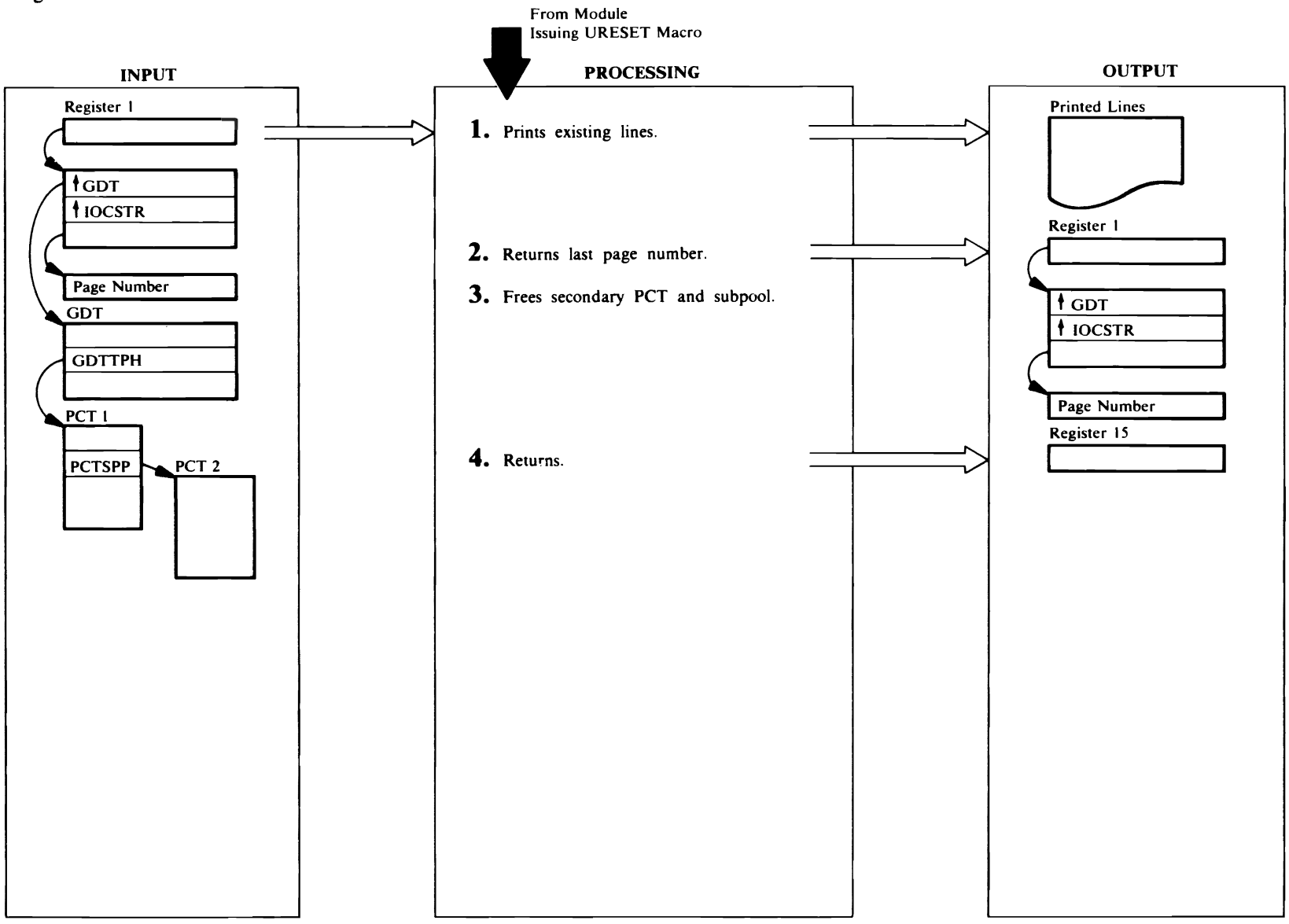
   A value of zero in any of the parameter lists causes the item to be reset to the Access Method Services default. RESTCONT evaluates the input parameter list. If the secondary PCT exists, PCTSETUP modifies it. Otherwise, PCTSETUP modifies the primary PCT.

Module: IDCTP04

Procedure: RESTCONT

4. RESTCONT puts a return code into register 15, and control returns to the module that issued the UREST macro.

Diagram 7.4 URESET Macro



Extended Description for Diagram 7.4

Module: IDCTP04

Procedure: RESETCON STACKFL

1. A primary PCT must exist. If it does not, RESETCON issues a UABORT macro. If a secondary PCT exists, RESETCON forces a page overflow so the next line will begin on a new page. Because controls governing the writing (like page depth and line width) are changing, the lines formatted under the current control values must be written before the control values change. STACKFL writes the stack buffer with a UPUT macro.

Module: IDCTP04

Procedure: RESTCON

2. If the Invoker of Access Method Services requested that the last page number be passed, RESETCON converts the current page number to binary and places it in the invoker's parameter list.

Module: IDCTP04

Procedure: RESETCON

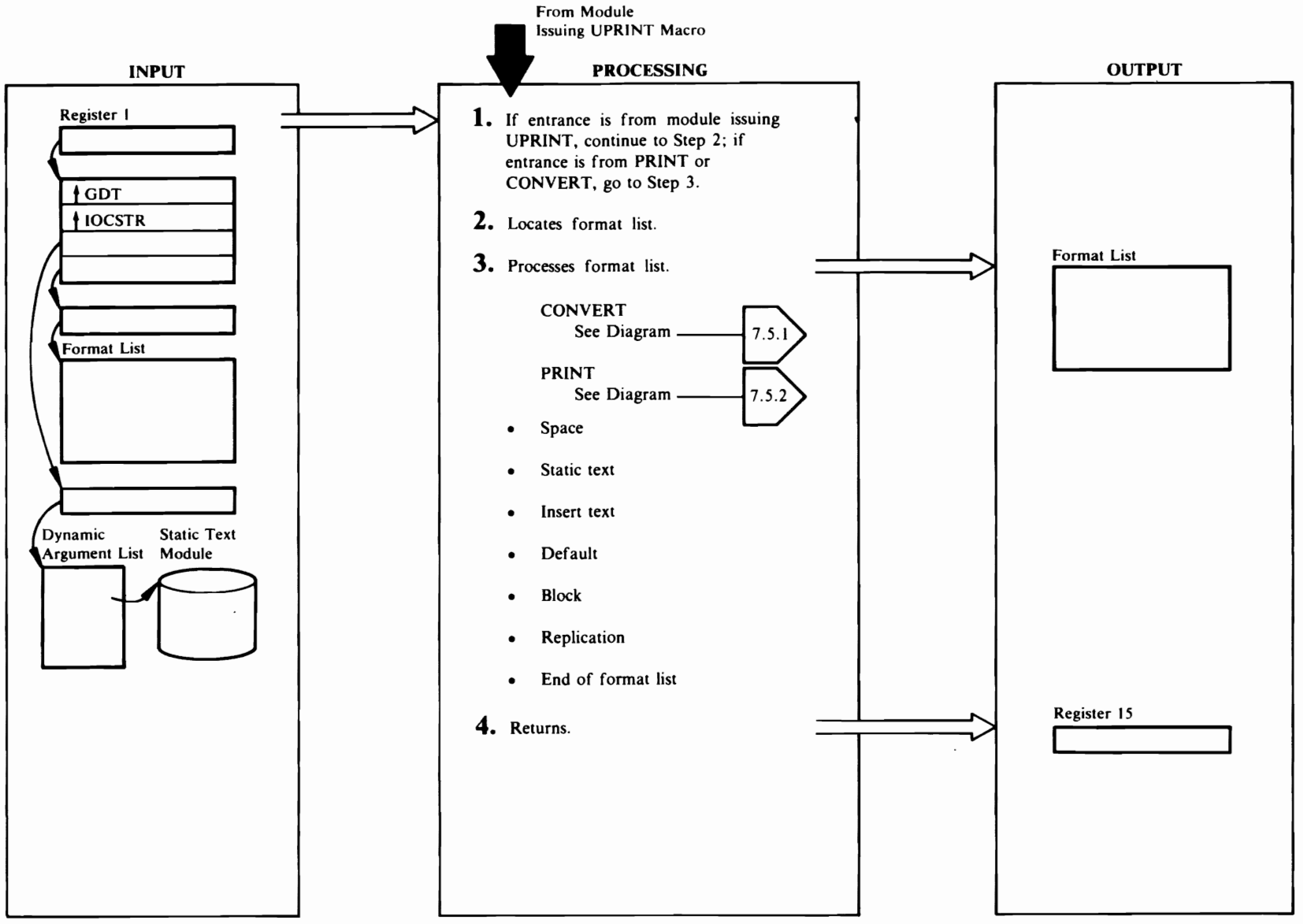
3. Before the secondary PCT is deleted, RESETCON copies some data into the primary PCT. One UFPOOL macro releases the secondary PCT, subtitle lines, footing lines, and any static text entries addressed from the secondary PCT in PCTSQP because everything was obtained with subpool identification 'TP01'. RESETCON sets the address of the secondary PCT to zero in the primary PCT in PCTSPP. This resets all page control values to the values contained in the primary PCT.

Module: IDCTP04

Procedure: RESETCON

4. RESETCON puts a return code into register 15, and control returns to the module that issued the URESET macro.

Diagram 7.5 UPRINT



Extended Description for Diagram 7.5

Module: IDCTP01, IDCTP05

Procedure: IDCTPPR, IDCTP05

1. If entrance is from a module issuing a UPRINT macro, continue with step 2; if entrance is from PRINT, Diagram 7.5.2, or CONVERT, Diagram 7.5.1, go to step 3.
2. The format list, FMTLIST, and Print Control Table, (PCT), must be found. If a secondary PCT exists, IDCTPPR uses it; otherwise, IDCTPPR uses the primary PCT. The format list, FMTLIST, can be in one of three locations:
   - In the FSR
   - In a list of static text entries chained from the PCT
   - In a static text module

   If the format list is in the FSR, DARGSTID in the Dynamic Argument List, DARGLIST, is zero. The calling program gives the address of the FMTLIST to UPRINT as the fourth argument.

   IDCTPPR compares the static text identification in DARGSTID against the static text identification of each entry addressed from the Print Control Table in field PCTSQP. If a match is found, IDCTPPR uses that FMTLIST in the static text entry as input to UPRINT. If a match is not found, IDCTPPR must obtain the FMTLIST from a static text module.

   IDCTP05 concatenates the name of the static text module in DARGSMOD with the characters 'IDCTS' and compares it with the name of the static text module in storage. The name of the static text module currently in storage is kept in PCTSTM in the PCT. If the names do not match, IDCTP05 deletes the module named in PCTSTM with a UDELETE macro, and IDCTP05 loads the module named in DARGSMOD with a ULOAD macro. IDCTP05 puts the name and address of the newly loaded module in the PCT. IDCTP05 finds the particular static text entry by using DARGSENT as an index to the static text module. IDCTP05 copies everything in the static text entry after the length field and puts the static text identification and the address of the next entry in the list at the beginning of each entry on the list. IDCTP05 then chains the copy into the list of static text entries addressed from PCTSQP so it will be readily available when it is used again. See "Debugging a Formatting Problem" in the chapter "Diagnostic Aids" for a discussion of static text entries.

Module: IDCTP01

Procedure: IDCTPPR, SPACE, STATIC, INSERT, BLOCK, REDO,

3. IDCTPPR takes action on the format list substructures in FMTLIST depending upon the structure type. IDCTPPR processes substructures in the order of their appearance in the FMTLIST. If the high order bit in FMTFLGS is on, this substructure is the last in the FMTLIST. The line buffer is a work area where each line is formatted. If there is formatted data in the line buffer, IDCTPPR calls LINEPRT to write the line. See Diagram 7.5.2. See "Debugging a Formatting Problem" for examples of FMTLIST in "Diagnostic Aids" chapter.

   Types of substructures:

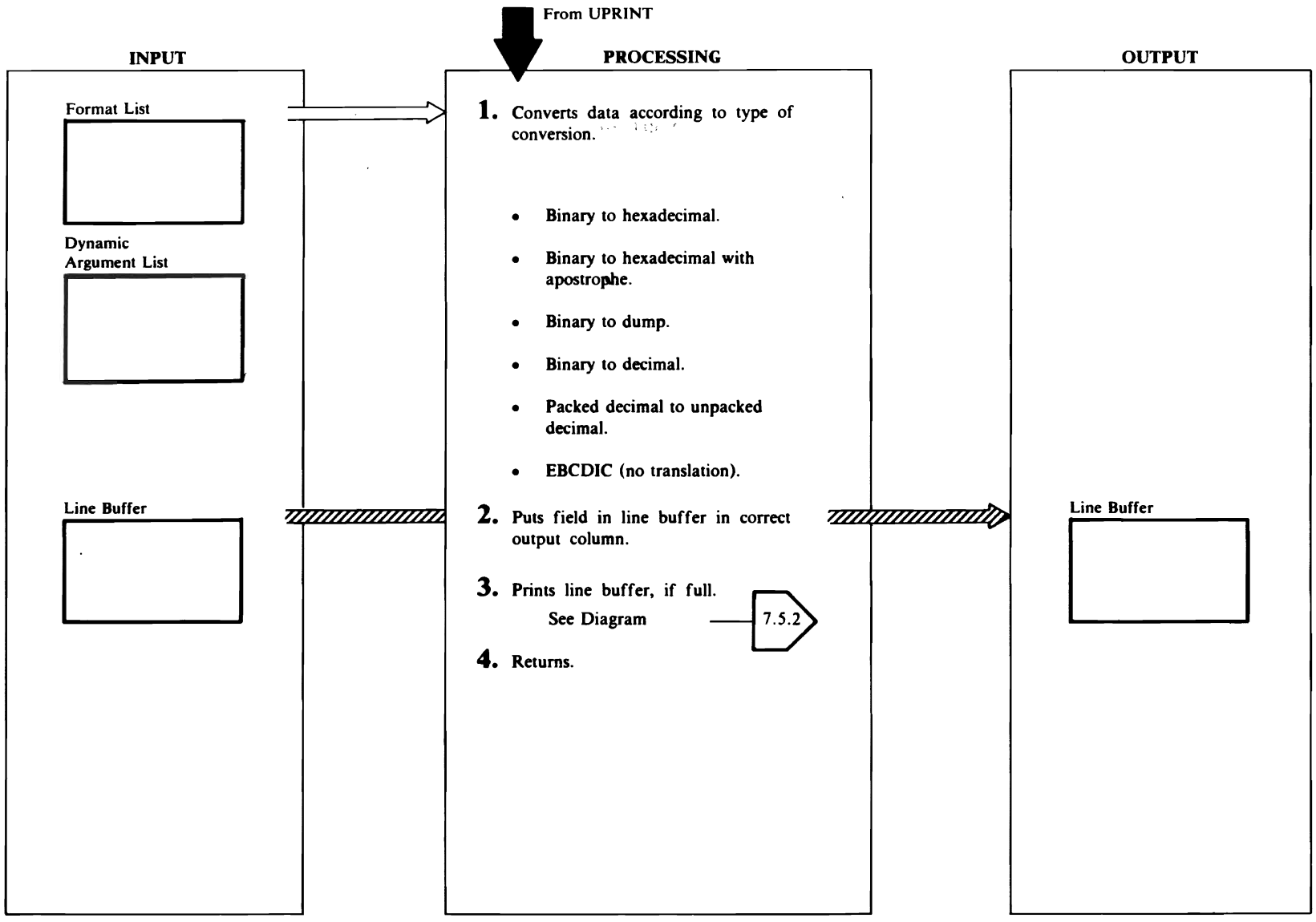
   - Space— If this is the first substructure in the FMTLIST, SPACE saves the spacing-type character from the FMTLIST for LINEPRT, and control returns to step 3 for the next substructure. If the space substructure is not the first substructure in the FMTLIST, SPACE transfers control to PRINT. After control returns from PRINT, the new spacing-type character is saved for the next line. For more information on PRINT, see Diagram 7.5.2. Control returns to step 2 for the next substructure.
   - Static text— STATIC passes the address of the input data, length of input data, type of conversion, position in the output line, and length of output field to ICONVERT. See Diagram 7.5.1.
   - Insert data— INSERT compares the insert reference number in FMTRFNO against every DARGINS field in the Dynamic Data List. If the same number is found in DARGINS, INSERT gives the following information to CONVERT: the length in DARGINL, the address in DARGDTM, the type of conversion from FMTCNVF, the output field length from FMTOLEN, and the position for the field in the output line from FMTOCOL. See Diagram 7.5.1. If the same number is not found in any DARGINS, INSERT ignores the insert-data substructure, and control returns to step 3 for the next substructure. If the next substructure is a default-text substructure, INSERT processes the default structure.
   - Default text— If a default-text substructure does not immediately follow an insert substructure that does not have a matching reference number in DARGINS, INSERT ignores the default-text substructure, and control returns to step 3 for the next substructure. INSERT uses the default-text substructure instead of a matching DARGINS to describe input for an insert-data substructure. INSERT takes the values for input and output from the default-text substructure only. Nothing is taken from the insert substructure. Control is given to CONVERT. See Diagram 7.5.1.
   - Block format— BLOCK obtains input information from DARGDBP and DARGILP. BLOCK adds the offset count in FMTIOFF to the address in DARGDBP to get the address of the input data. If the input length in FMTILEN is zero or 32,767, BLOCK uses the input length in DARGILP. If the length in FMTOLEN is zero or 32,767, the output length is the length of the converted input data. All this data is given to CONVERT. See Diagram 7.5.1.
   - Replication— REDO compares the reference number in FMTRFNO against every DARGREP field. If the same number is not found, REDO ignores the replication substructure and control returns to step 3 for the next substructure. If the same number is found in DARGREP, REDO uses the count in DARGPCT for loop control to set up the number of times the following substructures are repeated. REDO obtains the number of substructures to repeat from FMTRBC. At the end of each time through the substructures, REDO calls PRINT to print a line because the output positions for each field are unchanging. See Diagram 7.5.2. REDO saves the value in FMTRIO and adds it to each address of block data in the substructures being repeated.

Module: IDCTP01

Procedure: IDCTPPR

4. IDCTPPR puts a return code in register 15 and returns control to the module that issued the UPRINT macro.

Diagram 7.5.1 UPRINT Macro – Convert



Extended Description for Diagram 7.5.1

Module: IDCTP01

Procedure: CONVERT, BHCONV, BHDCONV, BDCONV, PUPCONV, EBCDIC

1. CONVERT checks the conversion type from FMTCNVT and converts the field accordingly. Types of conversion:

   Binary to hexadecimal— BHCONV converts bytes of binary data to their equivalent printable hexadecimal. printable hexadecimal. BHCONV prints two characters for each byte. The maximum input length is 32,767. If the length of the converted data is greater than the length of the output field, BHCONV truncates the data on the right. If the length of the converted data is less than the length of the output field, BHCONV does not change the remaining fields to the right. If the converted data extends beyond one line, BHCONV continues the data on the next line.

   Binary to hexadecimal with apostrophe— BHCONV converts bytes of binary data to their equivalent BHCONV prints two characters for each byte. The output is preceded by X' and followed by a single quote. The maximum input length is ((line width - starting position)/2) - 3. If the length of the converted data is greater than the length of the output field, BHCONV truncates the data on the right. If the length of the converted data is less than the length of the output field, BHCONV does not change remaining fields to the right of the trailing apostrophe. If the converted data extends beyond one line, BHCONV truncates the data on the right.

   Binary to dump— BHDCONV converts bytes of binary data to their equivalent printable hexadecimal. BHDCONV prints two characters for each byte. This type of conversion forces the output to begin on a new line. IDCTPPR is called to put the current line in the stack buffer prior to calling CONVERT again. See Diagram 7.5. BHDCONV formats the output line like a standard ABEND dump with: relative addresses on the left of the page, eight segments in the center, and a 32-byte EBCDIC translation with non-printable characters replaced by periods on the right of the page. The output starts in column one and BHDCONV uses 32 bytes of input per line. The maximum input length is 32,767.

   Binary to decimal— BDCONV converts bytes of binary data to their equivalent packed decimal, then calls PUPCONV for further conversion to unpacked decimal. Sign suppression, leading zero suppression, and left alignment can be used. The input length is one to four bytes, and the maximum output length is 16 bytes including the sign. If the length of the converted number is greater than the length of the output field, BDCONV truncates the number on the left. If the converted number extends beyond one line, PUPCONV truncates the number on the right.

   Packed decimal to unpacked decimal— PUPCONV converts bytes of packed decimal data to their equivalent printable unpacked decimal. Sign suppression, leading zero suppression, and left alignment can be used. Eight bytes is the maximum input length, and 16 bytes including sign is the maximum output length. If the length of the converted number is greater than the length of the output field, PUPCONV truncates the number on the left. If the converted number extends beyond one line, PUPCONV truncates the number on the right..

   EBCDIC, no translation— EBCDIC assumes the input is in printable EBCDIC and no conversion is done. If right alignment is specified, the EBCDIC character string is aligned to the right in the print field. The print column specified is added to the print field length to determine the last printable position. Unwanted blanks following a non-blank character can be eliminated by specifying blank suppression on the following field. If blank suppression is specified on an EBCDIC field, EBCDIC moves that field left into the prior EBCDIC field so there is only one blank between the two fields. Blank suppression can be specified only on fields that immediately follow EBCDIC fields. The maximum input length is 32,767. If the output extends beyond one line, EBCDIC prints additional lines.

Module: IDCTP01

Procedure: CONVERT, BHCONV, BHDCONV, BDCONV, PUPCONV, EBCDIC

2. The conversion routines put the converted data in the line buffer in the correct column. FMTOCOL in the FMTLIST specifies the output column. If blank suppression is on (FMTCNVF = X'0010'), the output column is in PCTAPC in the PCT, and FMTOCOL is an offset from the output column in PCTAPC. In this case, the conversion routines find the output column by adding the value in PCTAPC to the value in FMTOCOL. The output column for each field is calculated separately from other fields. Output fields may overlap due to specification of output columns in FMTOCOL.

Module: IDCTP01

Procedure: CONVERT, BHCONV, BHDCONV, PUPCONV, EBCDIC

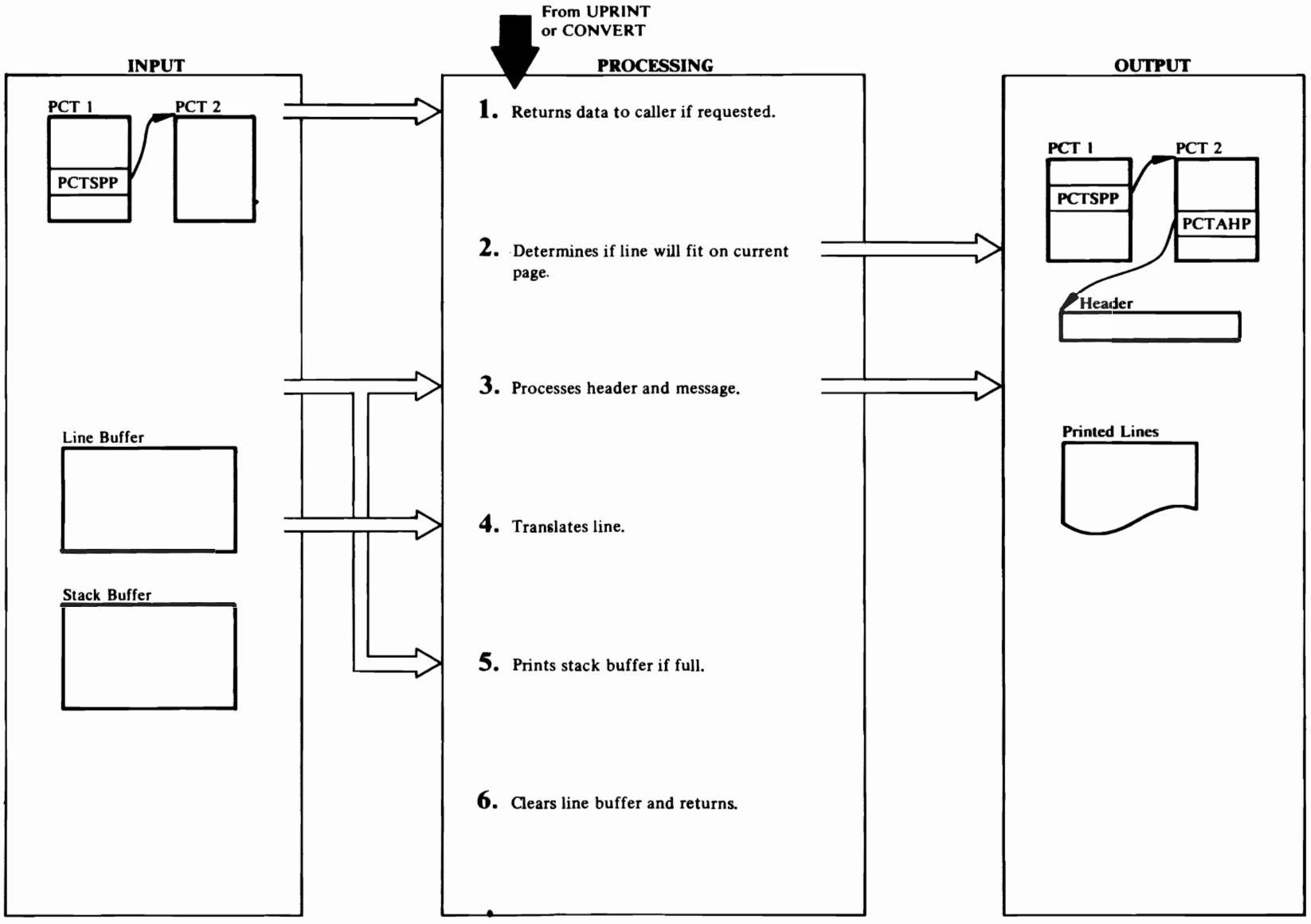
3. When the line buffer is full or a new line is to start, the conversion routines call LINEPRT to print the line. See Diagram 7.5.2.

Module: IDCTP01

Procedure: CONVERT, BHCONV, BHDCONV, PUPCONV, EBCDIC

4. When all the data specified by the FMTLIST substructure is converted, control returns to the caller in Diagram 7.5. See Diagram 7.5.

Diagram 7.5.2 UPRINT Macro – PRINT



Extended Description for Diagram 7.5.2

Module: IDCTP01

Procedure: LINEPRT LINERET

1. LINEPRT tests the return-area pointer in the argument list. If it is not zero, LINERET puts the formatted line in the return area without checking for or setting page-related indicators such as carriage control or headings. LINERET returns only as many characters as allowed by the return-area length.

Module: IDCTP01

Procedure: LINEPRT, STACKPUT

2. LINEPRT tests the print data set supplied with the UPRINT macro to determine if it is a change from the current print file. If the print data sets are changing, STACKPUT writes the stack buffer with a UPUT macro. Then LINEPRT puts the page number and next line number for the new print data set in PCTCPN and PCTNLI, respectively. LINEPRT puts the page number and next line number for the old print data set in PCTSPN and PCTSNL for the standard print data set or in PCTAPN and PCTANL for an alternative print data set. LINEPRT compares the current line number from PCTNLI with the pagesize in PCTPPD to determine if the current line with its spacing will fit on the current page. If the line will not fit, LINEPRT ejects a page, and LINEPRT prints all title lines on the new page. If the vertical spacing is more than three lines, LINEPRT writes blank lines until it is within three lines of the desired line number and the spacing character can handle spacing.

Module: IDCTP01

Procedure: LINEPRT (**Without VS2.03.807**)
LINEPRT, SEGMENT (**With VS2.03.807**)

3. LINEPRT tests the flags in the static text entry to determine if this static text entry describes a header line or a message.

   a. If it is a header line, LINEPRT puts the address of the translated header line in PCTAHP so it can be written again when a page overflows as well as when they are first given to the Text Processor. Unless all header lines, spaces, and one data line will fit on a page, a page overflow occurs, and LINEPRT ejects a page. The number is in HSDP in the static text entry. A UGPOOL is done for storage for the kept header line. Once a header is given to UPRINT, it can only by removed by another header, UESTS, UESTA, or URESET macro.

   b. **Without VS2.03.807:** If it is a message, LINEPRT writes the stack buffer with a UPUT macro.

      With VS2.03.807: If it is a message line, SEGMENT segments any messages over 72 characters in length into multiple message lines of 72 characters or less, then writes the stack buffer with a UPUT macro.

Module: IDCTP01

Procedure: LINEPRT

4. LINEPRT translates the formatted line using the translate table supplied for the print chain and addressed from PCTTRP. In Access Method Services translate tables, all non-printable bit combinations are changed to periods.

Module: IDCTP01,

Procedure: LINEPRT STACKPUT

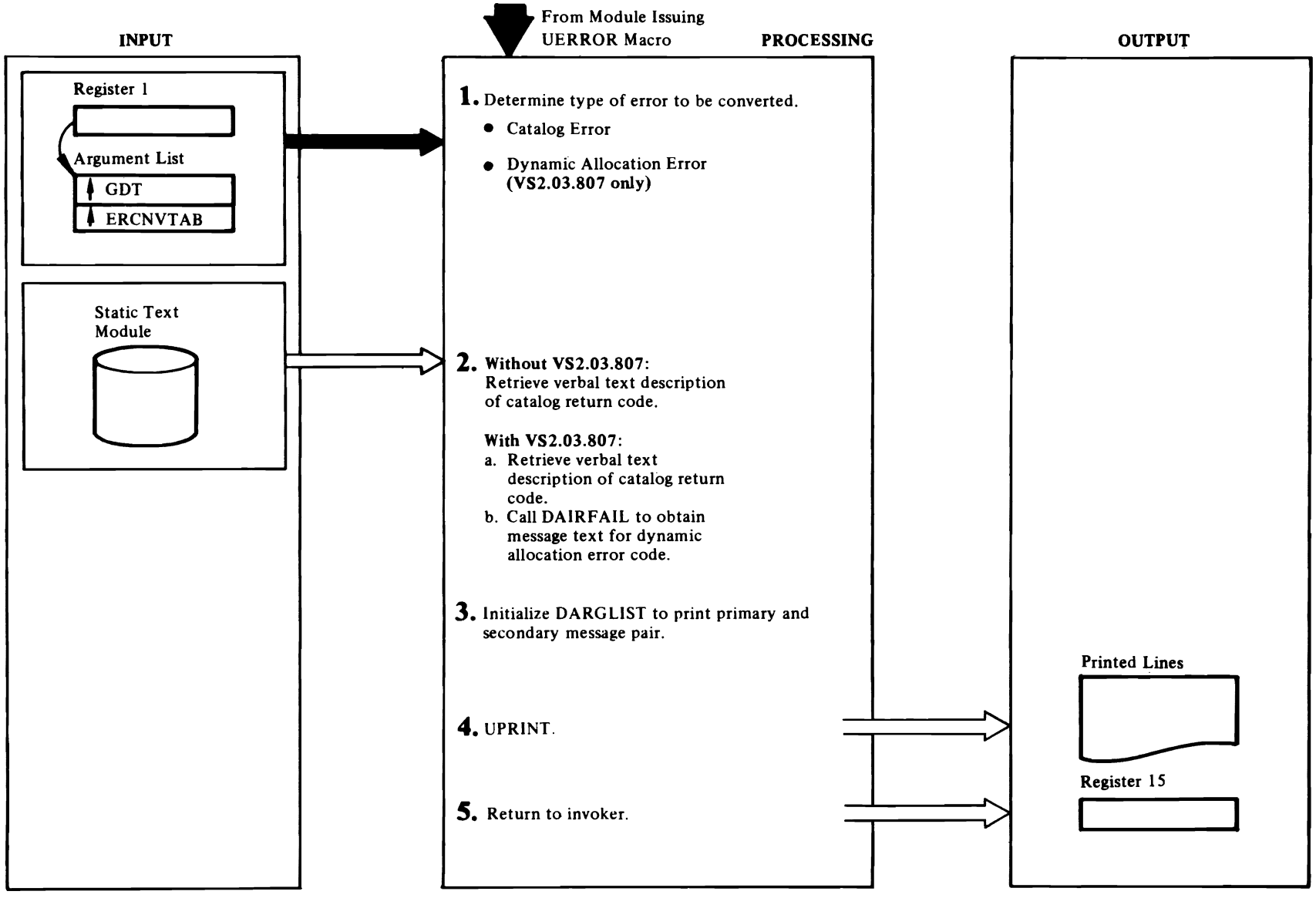
5. LINEPRT puts the translated line preceded by a two-byte length field in the stack buffer. When the stack buffer is full, STACKPUT issues a UPUT against the entire buffer. Lines in the stack buffer are in variable format with as many trailing blanks removed as possible. The minimum line size is 10 bytes. If the line is a message, STACKPUT issues a UPUT against the message alone. This is done because all messages go to the standard SYSPRINT data set. STACKPUT passes an identification number with the UPUT macro. The identification number for all data lines is zero and for messages is the message number. If Access Method Services is invoked interactively with TSO and the message is not to be printed on the TSO terminal, STACKPUT passes the two's complement of the message as the identification number. Therefore, STACKPUT must issue a separate UPUT for each message. If an alternative data set is being processed, there is no way to keep messages for the standard data set until ready to print, because there is only one stack buffer. With TSO, all data is printed immediately instead of being placed in the stack buffer.

Module: IDCTP01

Procedure: LINEPRT

6. LINEPRT fills the line buffer with blanks and control returns to the caller, FORMAT or CONVERT.

Diagram 7.6. UERROR MACRO



Extended Description for Diagram 7.6 (With VS2.03.807)

Module: IDCTP06

Procedure: IDCTP06

1. The Error Conversion Table (ERCNVTAB) indicates the type of error to be converted. The only allowable errors are catalog errors and dynamic allocation errors. IDCTP06 first checks the catalog management error type bit, ERCATLG; if it is on, control is passed to the CATERCNV procedure to process catalog error messages. If ERCATLG is not on, IDCTP06 checks the dynamic allocation error bit, ERDYNAL; if it is on, control is passed to the DAERCNV procedure to process dynamic allocation error messages.

Module: IDCTP06

Procedure: CATERCNV, DAERCNV

2. a. The CATERCNV procedure retrieves the verbal text description from the UERROR static text module (IDCTSTP6). CATERCNV uses the numeric catalog error code to index the appropriate verbal text entry in the static text module. The UPRINT macro is used to return the verbal text.

   b. DAERCNV builds a DAIRFAIL parameter list and invokes the system routine DAIRFAIL (IKJEFF18) via the ULINK macro. DAIRFAIL will analyze the dynamic allocation return code and will return to DAERCNV the text of a primary message or the text of a primary and a secondary message in buffer spaces provided by DAERCNV. A test of the field DFBUFL2 will indicate if a secondary message has been returned by DAIRFAIL. A length of zero in DFBUFL2 indicates no secondary message.

Module: IDCTP06

Procedure: CATERCNV, DAERCNV

3. The DARGLIST is initialized to print the primary and secondary message pair for a catalog error. Dynamic allocation errors will cause DARGLIST to be initialized for a primary message or a primary and a secondary message, depending on the message texts returned from DAIRFAIL. In a batch environment, both messages are issued to the SYSPRINT data set; in a TSO environment, the secondary message is issued to the TSO second level informational message chain. (The I/O Adapter handles these environmental considerations.)

Module: IDCTP06

Procedure: IDCTP06

4. Print the message pair via the Text Processor UPRINT macro.

Module: IDCTP06

Procedure: IDCTP06

5. Control is returned to the issuer of the UERROR macro.

Extended Description for Diagram 7.6 Without VS2.03.807

Module: IDCTP06

Procedure: IDCTP06

1. The Error Conversion Table (ERCNVTAB) indicates the type of error to be converted. The only allowable error is a catalog error.

Module: IDCTP06

Procedure: CATERCNV

2. Retrieve the verbal text description from the UERROR static text module (IDCTSTP6). CATERCNV uses the numeric catalog error code to index the appropriate verbal text entry in the static text module. The UPRINT macro is used to return the verbal text.

Module: IDCTP06

Procedure: CATERCNV

3. The DARGLIST is initialized to print the primary and secondary message pair. In a batch environment, both messages are issued to the SYSPRINT data set; in a TSO environment, the secondary message is issued to the TSO second level informational message chain. (The I/O Adapter handles these environmental considerations.)

Module: IDCTP06

Procedure: IDCTP06

4. Print the message pair via the Text Processor UPRINT macro.

Module: IDCTP06

Procedure: IDCTP06

5. Control is returned to the issuer of the UERROR macro.

Debugging Aids Visual Table of Contents

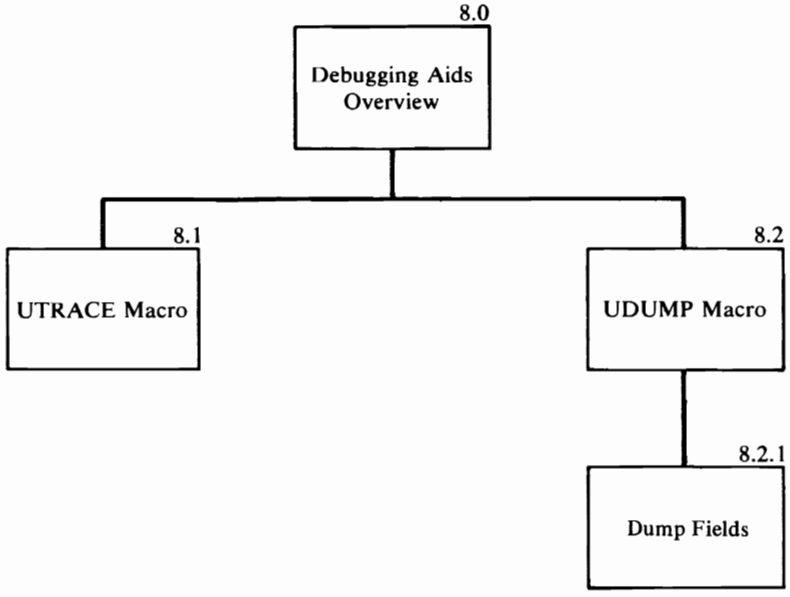
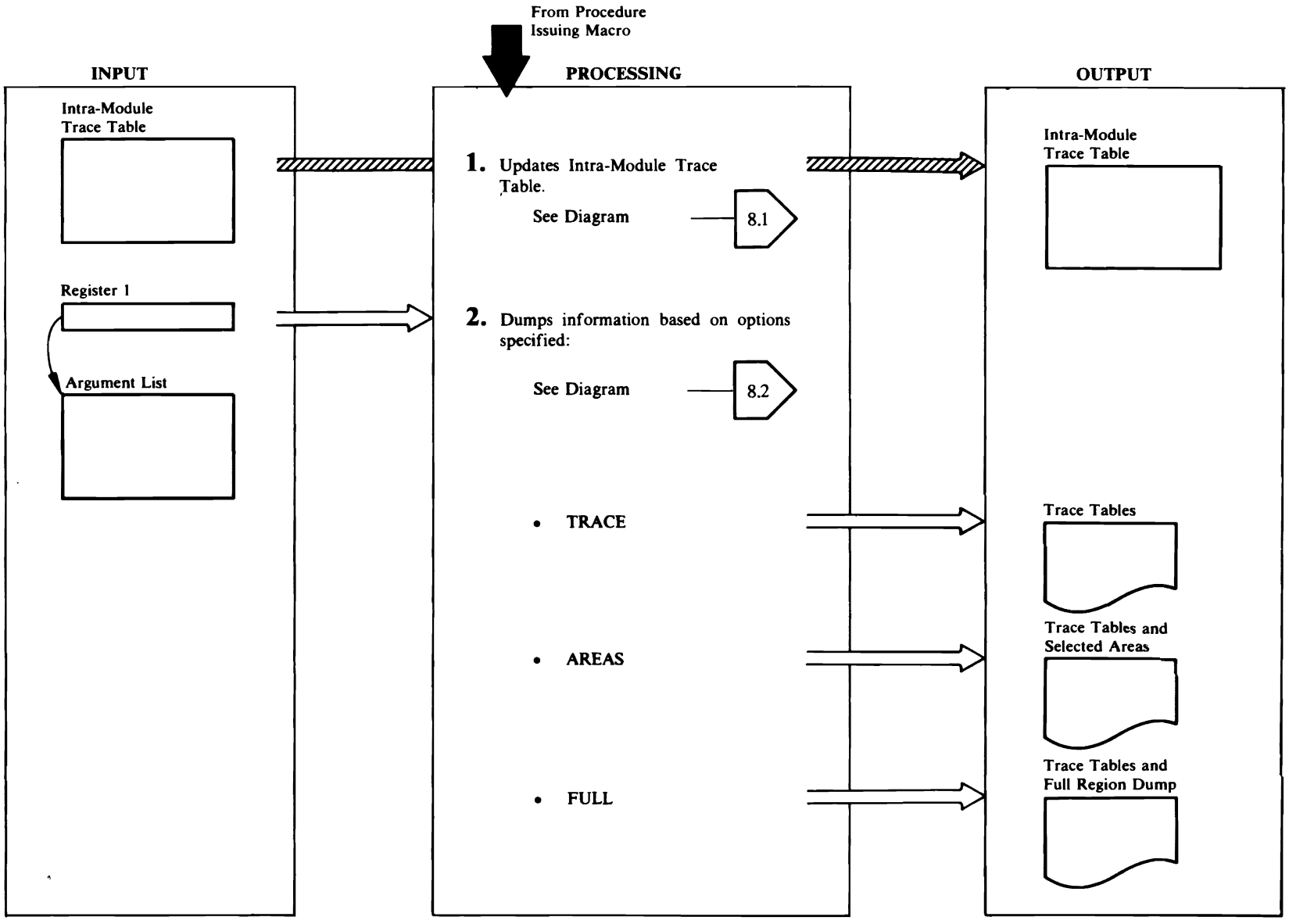


Diagram 8.0 Debugging Aids Overview



Extended Description for Diagram 8.0

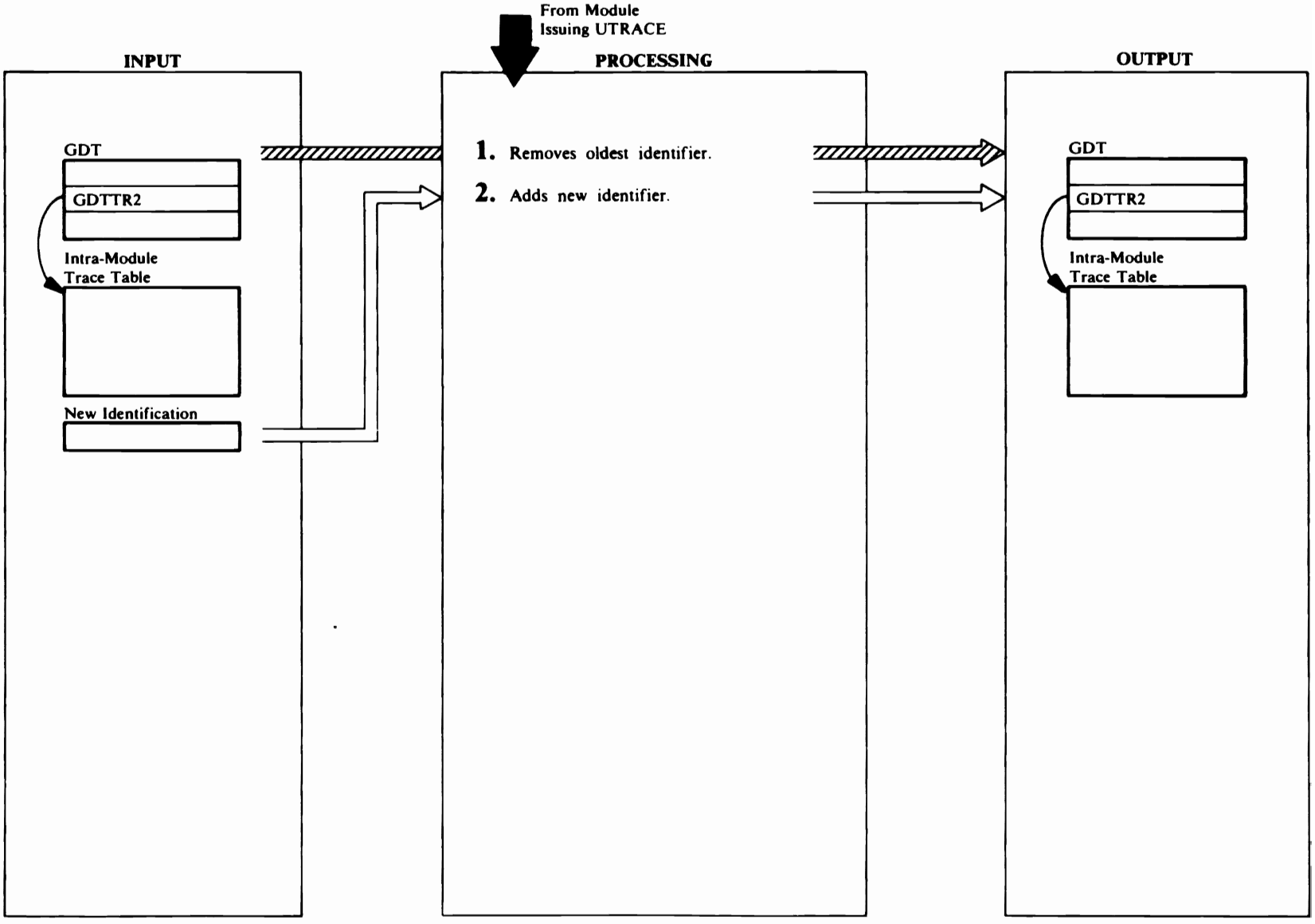
Module: IDCDB01

Procedure: IDCDB01

1. When a module issues a UTRACE macro instruction, the PL/S compiler generates in-line code that updates the Intra-Module Trace Table. Diagram 4.1 shows the UTRACE macro instruction in detail. Processing continues with the statement following the UTRACE macro.
2. The output of the UDUMP macro instruction depends upon the TEST keyword options specified either in the PARM command or from the EXEC statement.
   - If TRACE is specified, UDUMP prints the Inter- and Intra-Module Trace Tables each time a UDUMP macro is executed.
   - If AREAS is specified, UDUMP prints the Inter- and Intra-Module Trace Tables and items given to the UDUMP macro only for the areas specified.
   - If FULL is specified, UDUMP prints Inter- and Intra-Module Trace Tables and a full region dump only for the dump identifiers specified.

   Diagram 4.2 shows the UDUMP macro instruction in detail. Control returns to the module issuing the UDUMP macro.

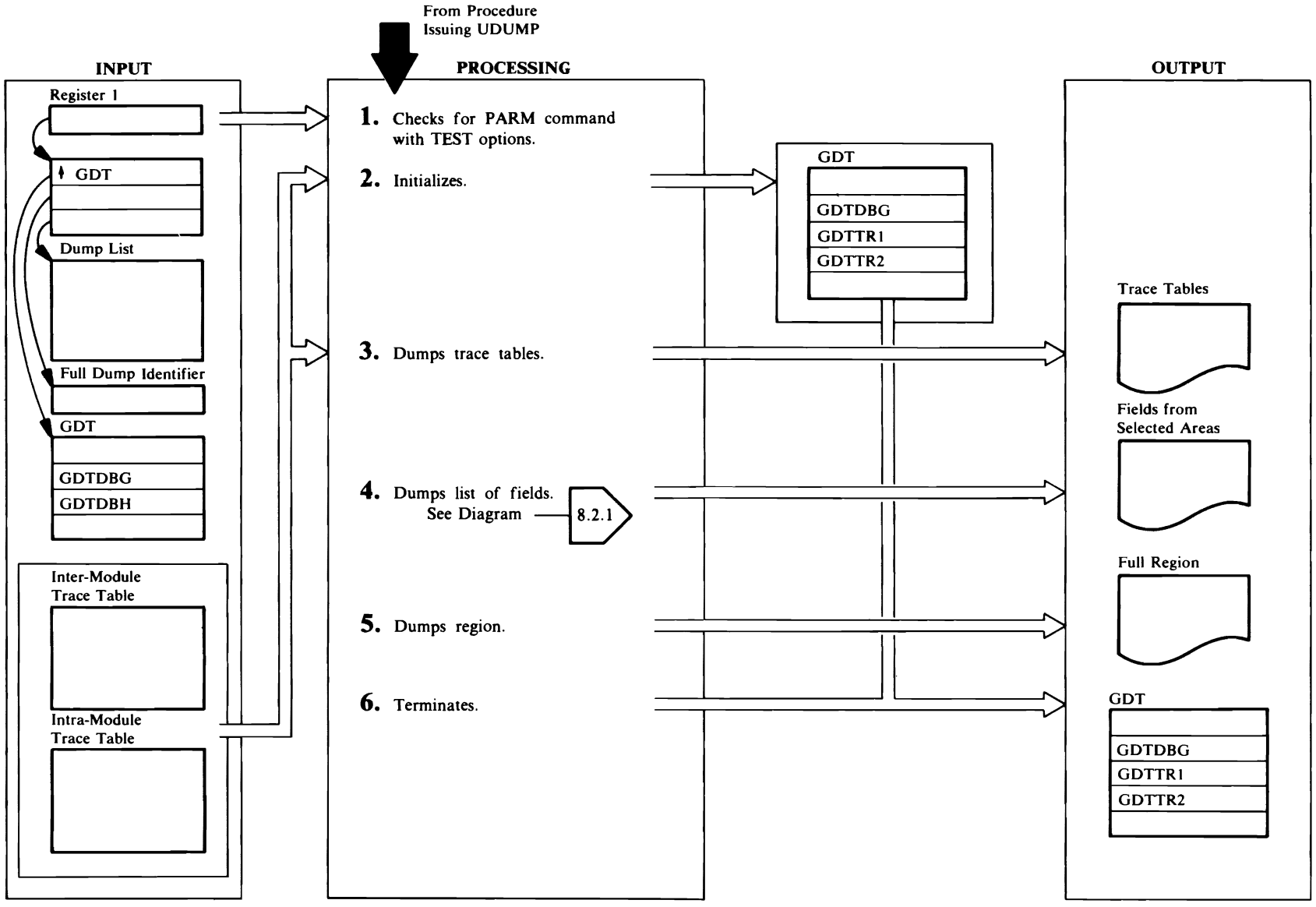
Diagram 8.1 UTRACE Macro



Extended Description for Diagram 8.1

1. The in-line code generated by the UTRACE macro gets the address of the Intra-Module Trace Table from the GDTTR2 field in the GDT. The in-line code shifts the Intra-Module Trace Table left so that the oldest identifier at the beginning of the table is lost.
2. The module provides the UTRACE macro with the new identifier to add to the Trace Table. The generated inline code puts the new identifier at the end of the Trace Table. The new identifier is 4 bytes long; the first two characters are characters 4 and 5 of the module name; the last two characters are assigned by the module. The identifier may either be four characters in quotes or the address of four characters. Control continues with the next instruction.

Diagram 8.2 UDUMP Macro



Extended Description for Diagram 8.2

Module: IDCPM01

Procedure: IDCPM01

1. The PARM command with the TEST keyword must be specified in order for any dumping to take place, or the TEST keyword must be specified in the PARM field of the EXEC statement. The PARM FSR, IDCPM01, has loaded the dump routine, IDCDB01, and has put the address of the dump routine in the GDT, if dumping is to take place when the UDUMP macro is issued. If GDTDBG is non-zero, control goes to Step 2. If GDTDBG is zero, the dump routine is not loaded; no dumping takes place; and control remains in the module issuing the UDUMP macro.

Module: IDCDB01

Procedure: IDCDB01

2. IDCDB01 obtains the calling module identifier from the last entry in the Inter-Module Trace Table. It issues a UTRACE macro to put the caller's module identification in the Intra-Module Trace Table. Both the Inter-Module and the Intra-Module Trace Tables are saved so that the Trace Tables will not be updated during the dumping operation, and the information in the Trace Tables at the time the UDUMP was issued is preserved. IDCDB01 turns off the TEST options by saving the address of the dump routine and setting GDTDBG to zero. This prevents any dumps during the processing of the current dump operation. IDCDB01 also issues a ULISTLN macro to get the number of arguments passed via the UDUMP macro. If there are three arguments, IDCDB01 has received a list of items to dump.

Module: IDCDB01

Procedure: IDCDB01

3. IDCDB01 uses the Test Option Data Area, whose address is in GDTDBH, to determine whether or not to print the Trace Tables. The Trace Tables are printed if any one of the following conditions is present:

   - TESTRACE contains a non-zero value, indicating that the Trace Tables are to be printed each time UDUMP is executed.
   - IDCDB01 compares the calling module identifier from the Inter-Module Trace Table with the module identifiers in the AREANAME. If a match is found, it prints the Trace Tables.
   - IDCDB01 compares the full dump identifier provided by the module issuing the UDUMP macro with the full dump identifiers in FDUMPID. If a match is found, it prints the Trace Tables.

Module: IDCDB01, IDCDB02

Procedure: IDCDB01 IDCDB02

4. If three arguments are given to the UDUMP macro, the third is a list of areas to be dumped. IDCDB02 converts and prints each item in the list. If the calling module identifier from the Inter-Module Trace Table matches a name in AREANAME, IDCDB01 invokes IDCDB02 to process the list. Otherwise, the list is ignored. Diagram 8.2.1 shows dumping fields in detail.

Module: IDCDB01

Procedure: IDCDB01

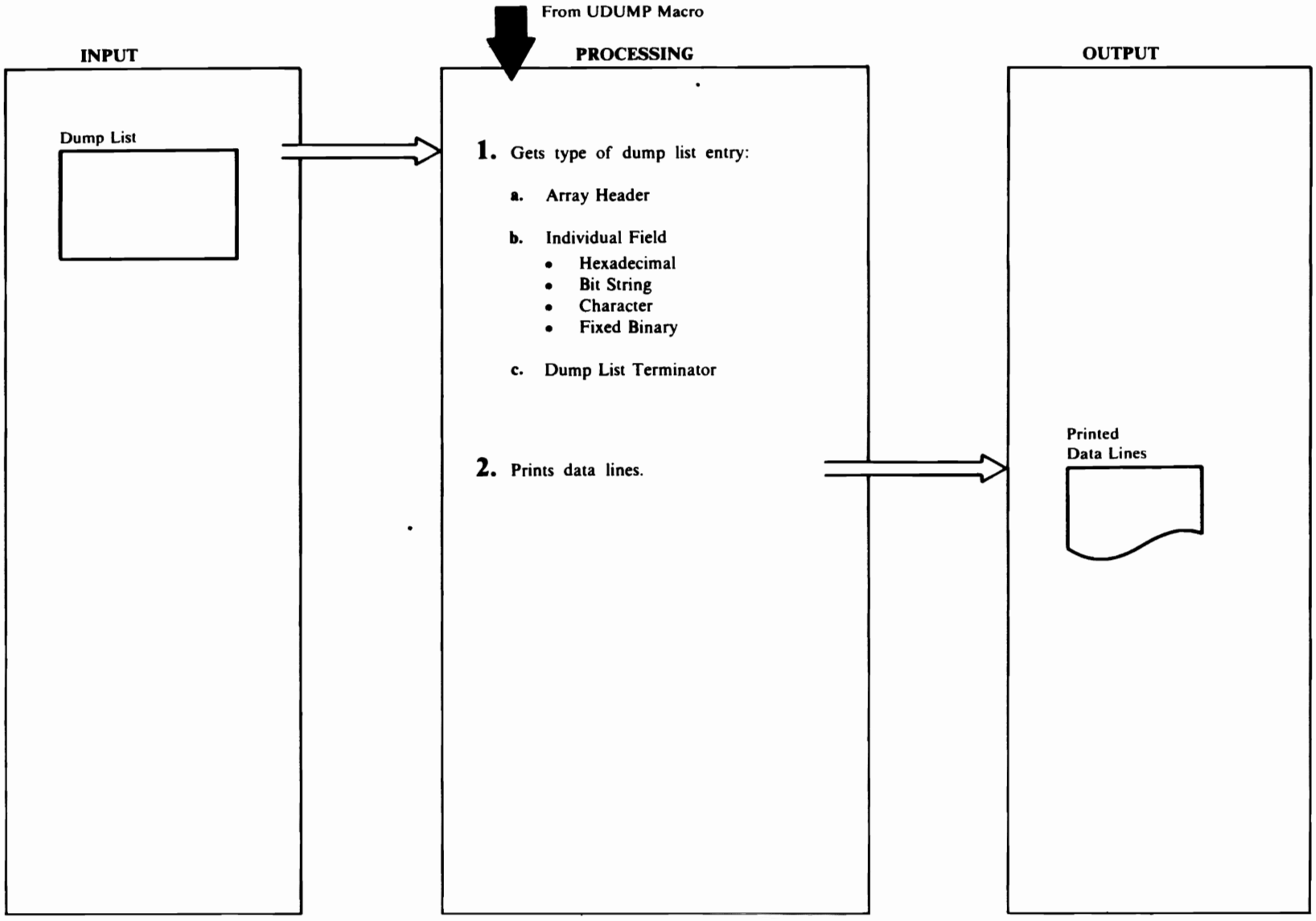
5. IDCDB01 compares the full dump identifier provided by the module issuing the UDUMP macro with full dump identifiers in FDUMPID. If no match is found, processing continues with step 6. IDCDB01 adds 1 to REALBEG and checks the number with FDUMPBEG to determine if the current pass is within the dumping range. If it is, IDCDB01 compares REALCNT with FDUMPCNT to determine if all the dumps requested have been given. If they have not, IDCDB01 adds 1 to SNAPID and issues a USNAP macro to dump the region. UPRINT writes a message stating the full dump identifier (SNAPID).

Module: IDCDB01

Procedure: IDCDBO01

6. IDCDB01 puts the address of the Trace Tables in GDTTR1 and GDTTR2 and resets the TEST options by placing the address of the dump routine in GDTDBG. Control returns to the module that issued the UDUMP macro.

Diagram 8.2.1 UDUMP Macro Dump Field



Extended Description for Diagram 8.2.1

Module: IDCDB02

Procedure: ARRAYHDR, IDCDB02, NAMEFLD, ITEMDUMP, HCONVERT, BCONVERT, CCONVERT, FCONVERT

1. IDCDB02 processes each entry in the Dump List until the end of the list is reached.

   a. If the type in the Dump List is 'A', the entry is an Array Header. If there is any formatted dump data in the line, ARRAYHDR issues a UPRINT to print the line. Each array begins on a new line, and an Array Header cannot occur within the elements of another array. If an Array Header does occur within the elements of another array, UPRINT prints an error message, the Array Header is ignored, and the following field entries are processed as though the Array Header had not been in the Dump List. A UPRINT macro prints the name of the array from the Dump List. ARRAYHDR obtains the looping array control from the Dump List. The number of bytes in each input element of the array is used to address the elements of the array.

   b. If the type in the Dump List is H, B, C, or F, NAMEFLD formats the name of each field in the line. If the field is part of an array, NAMEFLD adds a subscript of the element number to the field name. NAMEFLD also checks the input data type and converts and formats the data as follows:

      - Type H – HCONVERT converts hexadecimal data to printable form and prints two characters per byte of input; each four bytes of input is converted and followed by a blank.
      - Type B – BCONVERT converts bit string data to printable form and prints eight characters followed by a blank per byte of input. The printed output is enclosed in quotes.
      - Type C – CCONVERT converts character input to printable form and prints one character per byte of input. The printed output is an unbroken string of characters enclosed in quotes.
      - Type F – FCONVERT converts fixed binary data to printable decimal. Leading zeros are suppressed. If the input is 2 or 4 bytes long, FCONVERT prints a sign; no sign is printed if the input is 1 or 3 bytes long.

   c. If the first byte of the Dump List entry is X'FF', IDCDB02 terminates processing of the list. Control returns to the main dump routine, IDCDB01.

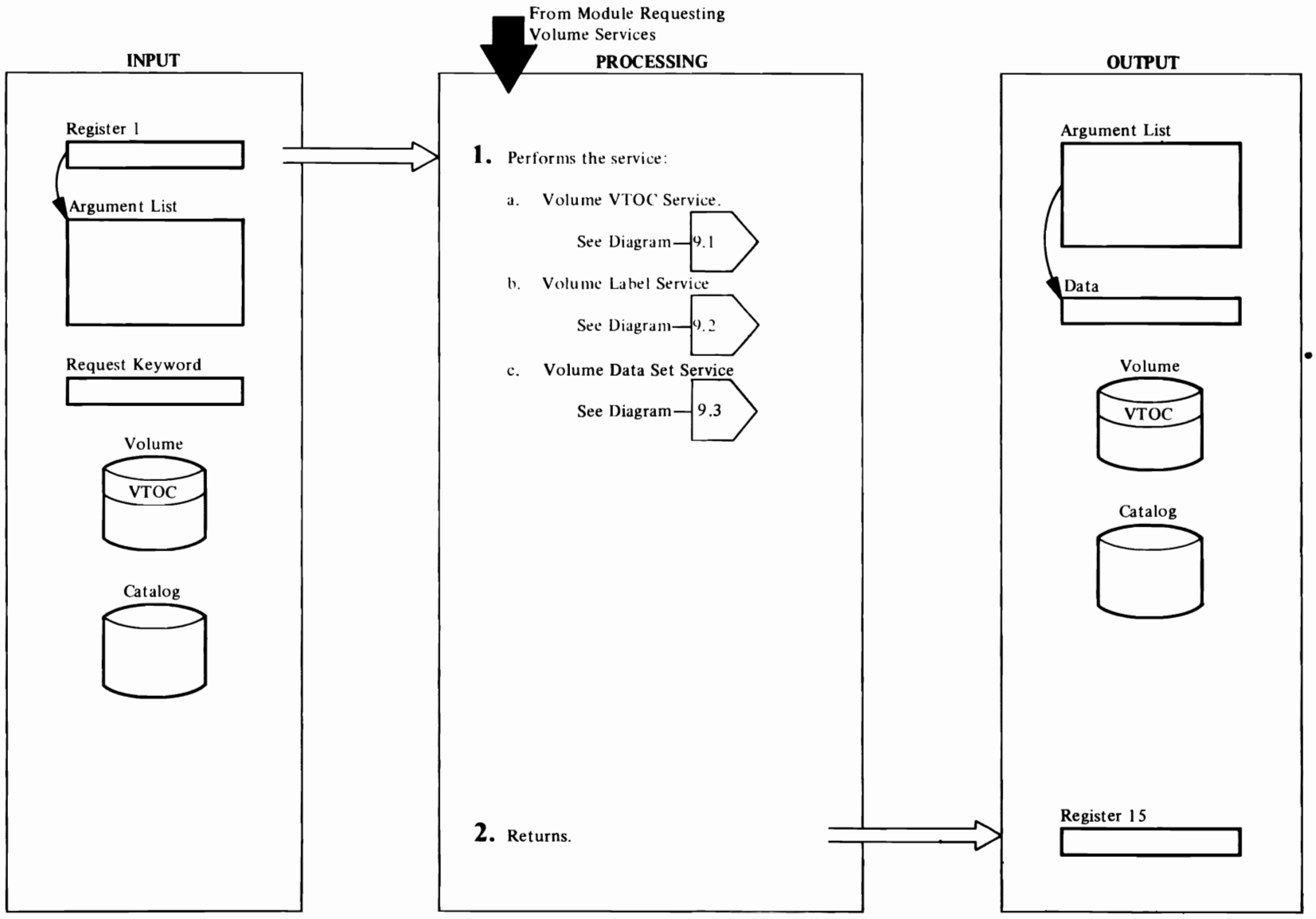
Module: IDCDB02

Procedure: ITEMDUMP

2. IDCDB02 logically divides the page into four columns. A maximum of four different fields may be printed on a line. Each printed field is preceded by its name from the Dump List entry and an equal sign. As soon as one line of data is formatted, a UPRINT macro prints the line.

Volume Services Visual Table of Contents

Diagram 9.0 Volume Services Overview



Extended Description for Diagram 9.0

Module: IDCVS01, IDCVS02, IDCVS03

Procedure: IDCVS01, IDCVS02, IDCVS03

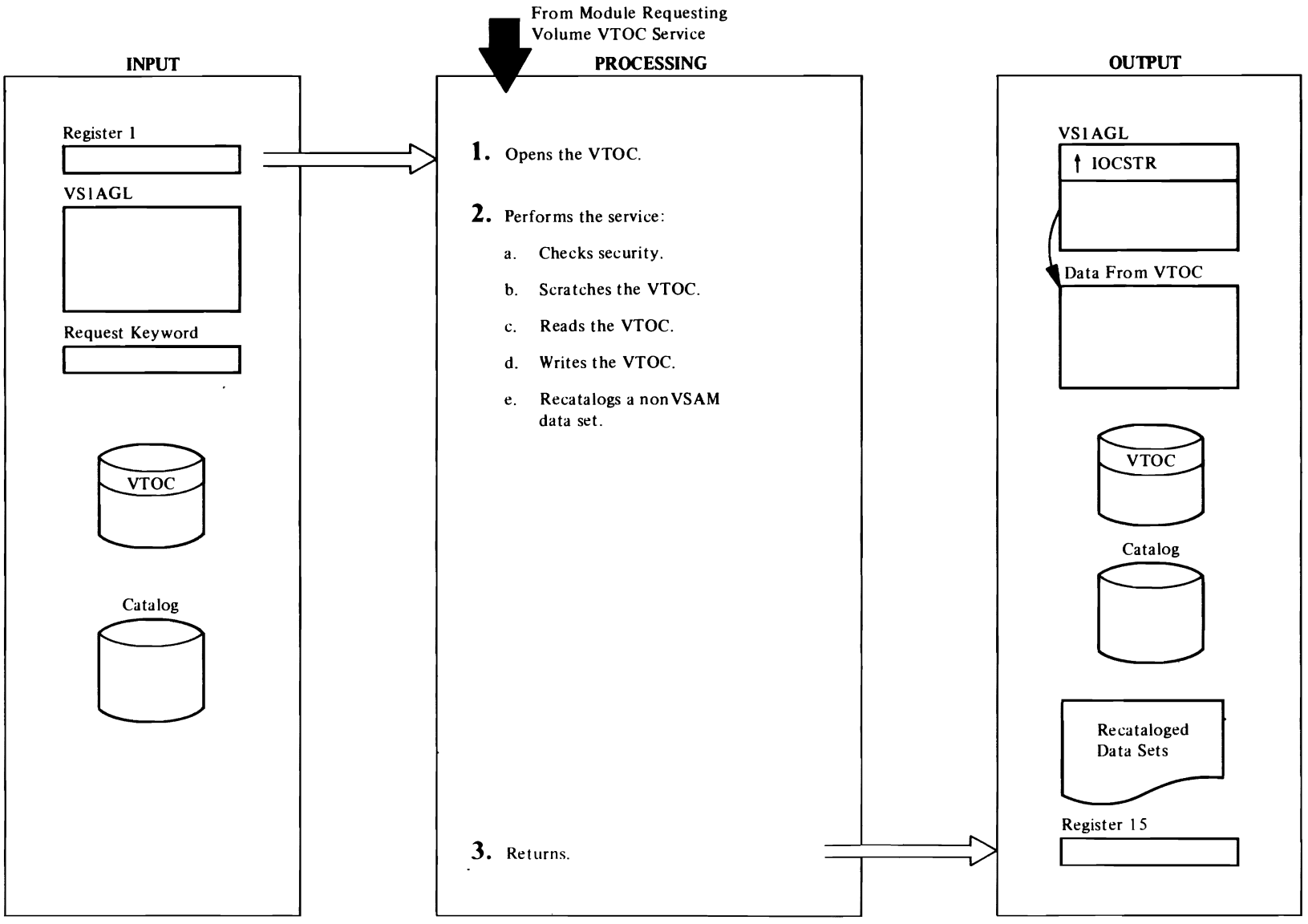
1. The service to be performed depends on the module that is called:
   a. IDCVS01 performs volume VTOC service. Diagram 9.1 shows volume VTOC service in detail.
   b. IDCVS02 performs volume label service. Diagram 9.2 shows volume label service in detail.
   c. IDCVS03 performs volume data set service. Diagram 9.3 shows volume data set service in detail.

Module: IDCVS01, IDCVS02, IDCVS03

Procedure: IDCVS01 IDCVS02, IDCVS03

2. If an error occurred, a message is printed. A return code is set in register 15 and control is returned to the caller.

Diagram 9.1 Volume Services – Volume VTOC Service



Extended Description for Diagram 9.1

(With SU 5752-824)

Module: IDCVS01

Procedure: IDCVS01 VTOCOPEN

1. VTOCOPEN issues a UOPEN macro to open the VTOC (if it's not already open) for BSAM update processing and issues a URESERVE macro to reserve the unit. VTOCOPEN puts the address of the IOCSTR in VS1AGL.

Module: IDCVS01

Procedure: FMT4CHK, FMT4READ, CATCHK, CATOPEN, FMT1CHK, VSAMCHK, NVSAMCHK, EXPIRCHK, FMT4SCR, FMT1READ, VTOCGET, VTOCPUT, RECATLOG

2. IDCVS01 checks the request keyword to determine the type of service to perform:

   'SECCHECK' indicates a request to check security.
   'SCRVTOC' indicates a request to scratch the VTOC.
   'GETVTOC' indicates a request to read the VTOC.
   'PUTVTOC' indicates a request to write the VTOC.
   'RECATLG' indicates a request to recatalog a nonVSAM data set.

   a. FMT4READ issues a UGET macro to read the Format 4 DSCB. FMT4CHK checks it to determine whether a VSAM catalog owns the volume. FMT4CHK indicates ownership status in VS1AGL. If VS1AGL indicates that VSAM ownership isn't allowed, FMT4CHK verifies that the volume is owned by VSAM.

      CATCHK verifies the VSAM catalog RACF authorization password. If the catalog dname or name was provided in VS1AGL, CATOPEN issues UALLOC and UOPEN macros to open the VSAM catalog for a CONVERTV command. CATCHK constructs a CTGPL and a CTGFL to locate the volume entry and verify RACF alter access or the master password of the catalog. CATCHK indicates in VS1AGL whether the volume entry says that the user catalog is on the volume.

      FMT1READ issues a UGET macro to read each Format 1 DSCB to verify data-set security and processes the VSAM data spaces and nonVSAM data sets as follows:

      VSAM data spaces—if VS1AGL requests it and if the catalog access was not authorized, then VSAMCHK issues a UEXCP macro to open each VSAM data space for input. To verify RACF or password authorization, VSAMCHK uses the dname for the volume to open each data space and uses the data-space name to override the data-set name. If catalog access was authorized, this checking is bypassed.

      NonVSAM data sets—if VS1AGL indicates that no nonVSAM data sets are allowed, FMT1CHK verifies that there are no nonVSAM data sets. If VS1AGL requests a check for read access, NVSAMCHK issues a UEXCP macro to open each RACF protected or each read-password-protected data set for input. If VS1AGL requests a check for write access, NVSAMCHK or EXPIRCHK issues a UEXCP macro to open each RACF protected, write-password-protected, or unexpired data set for output. If the data set being opened is password-protected, NVSAMCHK finds a DD statement for the data set from the list of password DD statements passed in VS1AGL and uses that DD statement to open the data set. If the data set is unexpired, but is neither RACF protected nor password-protected EXPIRCHK uses the DD statement for the volume to open the data set, overriding the data-set name.

   b. For a Format 4 or Format 1 DSCB, FMT4READ or FMT1READ issues a UGET macro to read the DSCB. If the volume was owned by VSAM, the Format 4 and Format 1 DSCBS are modified to show the volume is no longer owned by VSAM. FMT1SCR turns off security indicators in the Format 1 DSCBs to bypass VSAM security routines when the DSCB is scratched.

      For each Format 1 DSCB, FMT1SCR issues a USCRATCH macro to remove the DSCB from the VTOC.

   c. FMT4READ issues a UGET macro to read the Format 4 DSCB. If requested, VTOCGET retrieves information about alternate tracks or VSAM ownership.

   d. FMT4READ issues a UGET macro to read the Format 4 DSCB. If requested, VTOCPUT updates the alternate-track information in the DSCB. If requested, VTOCPUT issues a UTIME macro and updates the VSAM time stamp in the DSCB.

      VTOCPUT issues a UPUT macro to write the updated Format 4 DSCB.

   e. For each nonVSAM data set: FMT1READ issues a UGET macro to read the Format 1 DSCB.

      RECATALOG issues a URECAT macro to recatlog the data set with the new device type and volume serial number specified in VS1AGL. If requested, RECATLOG issues a UPRINT macro to list recataloged data sets.

Module: IDCVS01

Procedure: IDCVS01

3. If an error occurred, a message is written with the UPRINT macro. IDCVS01 returns to the caller with a return code in register 15.

Extended Description for Diagram 9.1

(Without SU 5752-824)

Module: IDCVS01

Procedure: IDCVS01 VTOCOPEN

1. VTOCOPEN issues a UOPEN macro to open the VTOC (if it's not already open) for BSAM update processing and issues a URESERVE macro to reserve the unit. VTOCOPEN puts the address of the IOCSTR in VS1AGL.

Module: IDCVS01

Procedure: FMT4CHK, FMT4READ, CATCHK, CATOPEN, FMT1CHK, VSAMCHK, NVSAMCHK, EXPIRCHK, FMT4SCR, FMT1READ, VTOCGET, VTOCPUT, RECATLOG

2. IDCVS01 checks the request keyword to determine the type of service to perform:

   'SECCHECK' indicates a request to check security.
   'SCRVTOC' indicates a request to scratch the VTOC.
   'GETVTOC' indicates a request to read the VTOC.
   'PUTVTOC' indicates a request to write the VTOC.
   'RECATLG' indicates a request to recatalog a nonVSAM data set.

   a. FMT4READ issues a UGET macro to read the Format 4 DSCB. FMT4CHK checks it to determine whether a VSAM catalog owns the volume. FMT4CHK indicates ownership status in VS1AGL. If VS1AGL indicates that VSAM ownership isn't allowed, FMT4CHK verifies that the volume is owned by VSAM.

      CATCHK verifies the VSAM catalog password. If the catalog dname or name was provided in VS1AGL, CATOPEN issues UALLOC and UOPEN macros to open the VSAM catalog for a CONVERTV command. CATCHK constructs a CTGPL and a CTGFL to locate the volume entry and verify the master password of the catalog. CATCHK indicates in VS1AGL whether the volume entry says that the user catalog is on the volume.

      FMT1READ issues a UGET macro to read each Format 1 DSCB to verify data-set security and processes the VSAM data spaces and nonVSAM data sets as follows:

      VSAM data spaces—if VS1AGL requests it and if the catalog password was specified incorrectly, then VSAMCHK checks VSAM master password, and, if the password wasn't specified correctly, issues a UEXCP macro to open each VSAM data space for input. VSAMCHK uses the dname for the volume to open each data space and uses the data-space name to override the data-set name. If the password was specified correctly, this checking is bypassed.

      NonVSAM data sets—if VS1AGL indicates that no nonVSAM data sets are allowed, FMT1CHK verifies that there are no nonVSAM data sets. If VS1AGL requests a check for read access, NVSAMCHK issues a UEXCP macro to open each read-password-protected data set for input. If VS1AGL requests a check for write access, NVSAMCHK or EXPIRCHK issues a UEXCP macro to open each write-password-protected or unexpired data set for output. If the data set being opened is password-protected, NVSAMCHK finds a DD statement for the data set from the list of password DD statements passed in VS1AGL and uses that DD statement to open the data set. If the data set is only unexpired, but not password-protected, EXPIRCHK uses the DD statement for the volume to open the data set, overriding the data-set name.

   b. For a Format 4 or Format 1 DSCB, FMT4READ or FMT1READ issues a UGET macro to read the DSCB. If the volume was owned by VSAM, the Format 4 and Format 1 DSCBS are modified to show the volume is no longer owned by VSAM. FMT1SCR turns off security indicators in the Format 1 DSCBs to bypass VSAM security routines when the DSCB is scratched.

      For each Format 1 DSCB, FMT1SCR issues a USCRATCH macro to remove the DSCB from the VTOC.

   c. FMT4READ issues a UGET macro to read the Format 4 DSCB. If requested, VTOCGET retrieves information about alternate tracks or VSAM ownership.

   d. FMT4READ issues a UGET macro to read the Format 4 DSCB. If requested, VTOCPUT updates the alternate-track information in the DSCB. If requested, VTOCPUT issues a UTIME macro and updates the VSAM time stamp in the DSCB.

      VTOCPUT issues a UPUT macro to write the updated Format 4 DSCB.

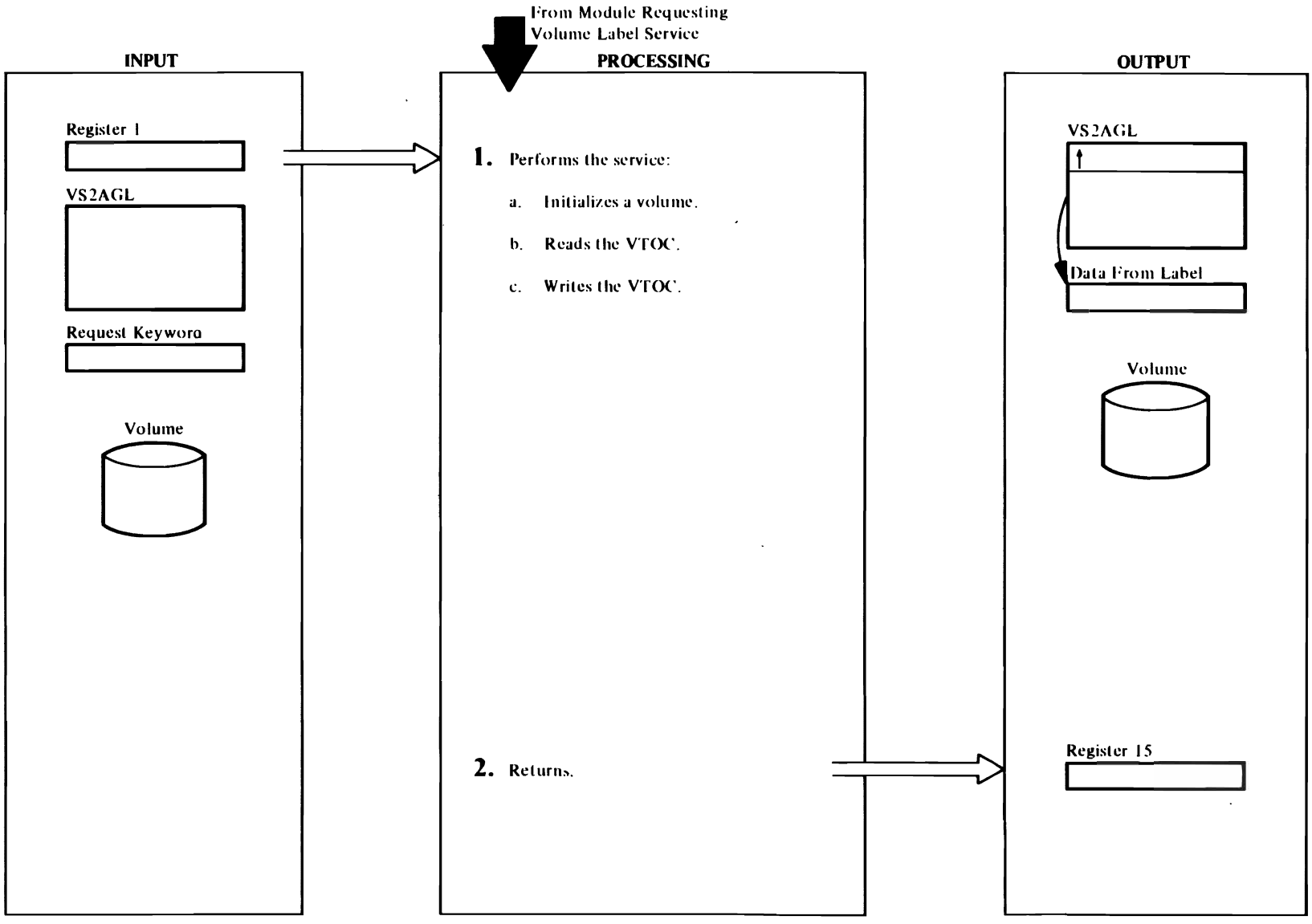
   e. For each nonVSAM data set: FMT1READ issues a UGET macro to read the Format 1 DSCB. RECATALOG issues a URECAT macro to recatlog the data set with the new device type and volume serial number specified in VS1AGL. If requested. RECATLOG issues a UPRINT macro to list recataloged data sets.

Module: IDCVS01

Procedure: IDCVS01

3. If an error occurred, a message is written with the UPRINT macro. IDCVS01 returns to the caller with a return code in register 15.

Diagram 9.2 Volume Services – Volume Label Service



Extended Description for Diagram 9.2

Module: IDCVS02

Procedure: IDCVS02, INITVOLM, GETLAB, PUTLAB

1. IDCVS02 checks the request keyword to determine the type of service to perform:

   'INITVOL' indicates a request to initialize a volume.
   'GETLABEL' indicates a request to read the Volume Label.
   'PUTLABEL' indicates a request to write the Volume Label.

   a. INITVOLM issues a UEXCP macro to create a DEB for the volume. It puts the address of the IOXCTLBK in VS2AGL. It writes the VTOC, starting on cylinder 0, track 1: it writes one Format 4 DSCB, one Format 5 DSCB, and 37 Format 0 DSCBs on the first track, and fills the rest of the tracks in the VTOC with Format 0 DSCBs. It writes the volume label record and IPL 1 and 2 on cylinder 0, track 0.

   b. GETLAB issues a UEXCP macro to open the VTOC to gain access to the label. It issues a URESERVE macro to reserve the unit. GETLAB issues a UEXCP macro to read the label and places the volume serial number and/or owner name in an area pointed to by VS2AGL. It issues a UEXCP macro to close the VTOC.

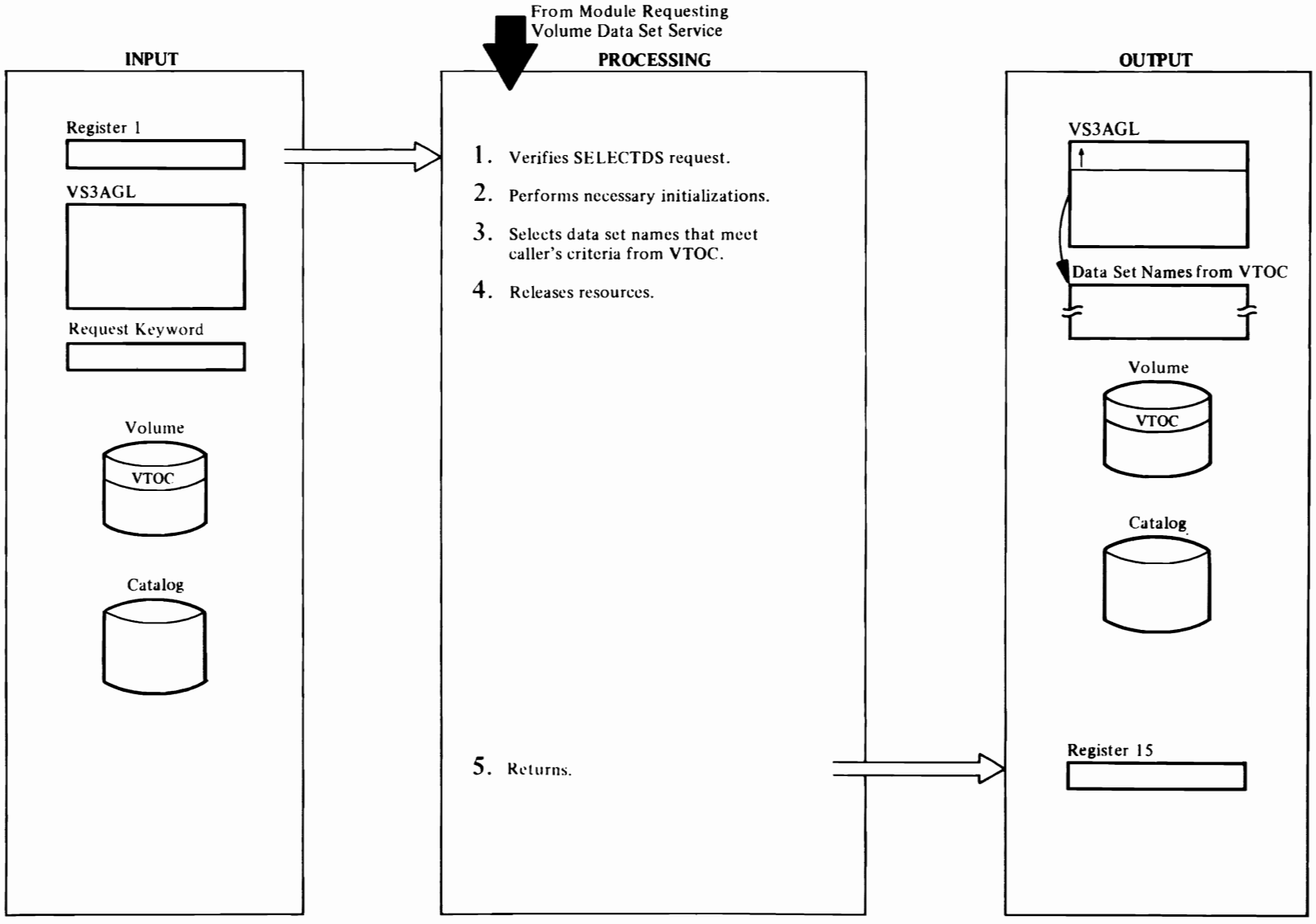
   c. PUTLAB issues a UEXCP macro to open the VTOC to gain access to the label. It issues a URESERVE macro to reserve the unit. PUTLAB issues a UEXCP macro to read the label, updates the volume serial number and/or owner name with information from the caller, and issues a UEXCP macro to write the Volume Label. It issues a UEXCP macro to close the VTOC.

Module: IDCVS02

Procedure: IDCVS02

2. If any errors occurred, a message is written with a UPRINT macro. IDCVS02 returns to the caller with a return code in register 15.

Diagram 9.3 Volume Services – Volume Data Set Service



Extended Description for Diagram 9.3

Module: IDCVS03

Procedure: IDCVS03

1. The request code by the caller is interrogated. If it is not 'SELECTDS', the UABORT macro is issued with code 40 (invalid parameters).

Module: IDCVS03

Procedure: INITIAL, OPENVTOC, GETBLK

2. Initialization for the module is performed by calling the INITIAL procedure. INITIAL calls the OPENVTOC procedure to open the VTOC. OPENVTOC obtains the UCB address of the device via the USYSINFO macro and then enqueues the VTOC by issuing the URESERVE macro. INITIAL next calls the GETBLK routine to determine and acquire the amount of storage required for the selected data set array. GETBLK also gets space for the track I/O buffer.

Module: IDCVS03

Procedure: SELECTOR, GETFMT1, CRITERIA, USAGE, STATUS

3. The SELECTOR procedure is called to search the data set names in the volume's VTOC and to select those which meet the caller's criteria. SELECTOR calls the GETFMT1 procedure to retrieve a volume format 1 DSCB from the VTOC. A return code of 12 indicates the entire VTOC has been processed. Control goes to step 4 if a return code of 12 is encountered. The CRITERIA routine is called to insert information into the data set element array if the data set meets the criteria.

   The USAGE routine is called to furnish the data set element array with information about the organization and size of the data set.

   The STATUS routine is called to furnish the data set element array with information about data set names needed for STATUS reports.

Module: IDCVS03

Procedure: CLEANUP

4. The CLEANUP procedure is called to release resources no longer required by IDCVS03. The UDEQ macro is issued to release the previous URESERVE on the VTOC.

   The UEXCP macro with CLOSE option is issued to close the VTOC. UEXCP will not be issued if the pointer to IOXCTLBK is zero, indicating the VTOC was not successfully opened.

   Space obtained for the track I/O buffer is freed by issuing the UFSPACE macro. UFSPACE is not issued if the pointer to the buffer is zero, indicating no storage was obtained.

Module: IDCVS03

Procedure: IDCVS03

5. Control is returned to the caller with a return code as follows:

   0 = Successful
   4 = Error
   8 = Non-terminating error

PROGRAM ORGANIZATION

This chapter describes the organization of the Access Method Services processor: the physical packaging of routines into load modules.

The final authorities for any program are the compiler and assembly listings for that program. This chapter complements those listings, and assumes that they are at hand. You should have them available for any in-depth analysis. This chapter directs you to a specific module of the processor; the listings for that module provide further detail. The next chapter, "Microfiche Directory," can help you relate the listings to this book.

Overall Organization

The processor consists of executable modules, organized into eight general areas, and non-executable modules (Command Descriptors and Text Structures). As described in the "Introduction," seven of these areas form a substructure that provides services and control for the remaining area. This substructure is made up of the Executive, the System Adapter, the I/O Adapter, the Text Processor, Volume Services, the Reader/Interpreter, and Debugging Aids. The eighth area consists of the Function Support Routines (FSRs), of which there are currently eleven for VSAM, one for checkpoint/restart, and 19 for the Mass Storage System—one for each functional command supported by the processor. The processor—except modules IDCAM01 and IDCAM02—resides in system data set SYS1.LINKLIB. Two modules, IDCAM01 and IDCAM02, reside in the TSO command library data set, SYS1.CMDLIB. Access Method Services command names (such as DELETE), which can be used interactively via TSO, have been linkedited as ALIASES of IDCAM01 or IDCAM02. When a TSO terminal user uses an Access Method Services command, IDCAM01 or IDCAM02 gives control to Access Method Services at entry point IDCSATO. The CHKLIST, PARM, MODAL commands, and Mass Storage System commands cannot be invoked by way of TSO.

Several modules are link-edited together into one load module (named IDCAMS with aliases of IDCSATO), which is loaded when the processor is invoked. This load module is the *root segment* and consists of:

IDCEX01	Executive main module
IDCEX02	Executive initialization, called by IDCEX01
IDCEX03	Executive termination, called by IDCEX01
IDCIO01	I/O Adapter main module
IDCIO02	I/O Adapter Open/Close, called by IDCIO01
IDCIO03	I/O Adapter positioning, called by IDCIO01
IDCIO05	I/O Adapter EXCP I/O processing module
IDCSA01	System Adapter initialization/termination module
IDCSA02	System Adapter services module
IDCSA03	System Adapter prologue/epilogue module
IDCSA05	System Adapter time module, called by IDCSA02
IDCSA06	System Adapter Mount/Demount/PostUCB/Check module
IDCSA07	System Adapter recatalog module
IDCSA08	System Adapter system services module
IDCSA09	System Adapter MSSC call macro
IDCSA10	System Adapter ESTAE environment module

IDCTP01	Text Processor print module, called by any module
IDCTP04	Text Processor Print Control Table control, called by any module.
IDCTP05	Text Processor Text Structure loading, called by IDCTP01 or IDCTP04
IDCTP06	Text Processor Error conversion module

The following modules are loaded by the system when their services are required.

IDCDB01	Dump routine, called by any module
IDCDB02	Symbolic dump, called by IDCDB01
IDCIO04	I/O Adapter's write module for the SYSPRINT data set in an interactive TSO environment, called by IDCIO01
IDCVS01	Volume Services VTOC routine, called by the FSRs
IDCVS02	Volume Services label routine, called by the FSRs
IDCVS03	Volume Data Set routine called by the FSRs

The Reader/Interpreter and the FSRs are alternately called and deleted by the Executive (IDCEX01) to perform their duties. There are two Reader/Interpreters. The invokation of Access Method Services determines which Reader/Interpreter is called. If Access Method Services is invoked from a batched job, the Reader/Interpreter for batched jobs is entered at IDCRI01, which in turn calls IDCRI02 and IDCRI03. If Access Method Services is invoked interactively with TSO, the Reader/Interpreter for interactive TSO is entered at IDCRI04. The FSRs are named as follows:

VSAM		CHECKPOINT/RESTART	
IDCAL01	ALTER	IDCCK01	CHKLIST
IDCBI01	BLDINDEX		
IDCCC01	CNVTCAT		
IDCDE01	DEFINE		
IDCDL01	DELETE		
IDCXP01	EXPORT		
IDCMP01	IMPORT		
IDCLC01	LISTCAT		
IDCRL01	LISTCRA		
IDCPM01	PARM		
IDCPR01	PRINT		
IDCRC01	EXPORTRA		
IDCRM01	IMPORTRA		
IDCRP01	REPRO		
IDCRS01	RESETCAT	**(VS2.03.808 only)**	
IDCVY01	VERIFY		

The Mass Storage System FSRs are described in *OS/VS Mass Storage System (MSS) Services Logic.*

System Macros and Services Used by Access Method Services

All requests for services from the operating system are issued by either the System Adapter or the I/O Adapter. Figure 5 lists all system and I/O macros issued by the processor, along with the issuing module's name and the label at the point of issue. These labels all begin with "L," contain a mnemonic for the macro, and end with a single digit. Thus they are easy to locate with the cross-reference table of the listing.

The adapters provide the operating-system services listed in Figure 5 to the rest of the processor.

Non-operating-system services are also provided by the adapters and by the Text Processor. Requests for services are represented in the listings by a call

to the appropriate service-module entry point. Requests for Volume Services are generated with the UCALL macro.

The Global Data Table (GDT) contains a branch vector to the various entry points in the adapters which provide these services. A routine obtains a service by loading the appropriate entry point address into a register and performing a BALR. Standard linkage is used: register 1 points to a list of argument addresses, register 13 points to a save area, register 14 contains the return address, and register 15 contains the entry point address. The exception is the call to SAABT: register 1 is not used, register 13 contains the address of a save area in the System Adapter, register 14 contains the address of SAABT and register 15 contains an abort code.

Macro	Module	Label
ABEND	IDCSA01	LABEND1
BLDL (VS2.03.807)	IDCSA02	LBLDL1
CALLTSSR (VS2.03.807)	IDCRI04	—
	IDCSA01	—
CAMLST (VS2.03.808)	IDCRS07	CLISTR CLISTS
CATLG	IDCSA02	LCATLG1
	IDCSA07	LCATLG1
CLOSE	IDCIO02	LCLOSE1
	IDCIO05	LCLOSE1
	IDCSA01	LCLOSE1 LCLOSE2
	IDCSA02	LCLOSE3
CHECK	IDCIO01	LCHECK1 LCHECK2
DELETE (VS2.03.807)	IDCSA01 IDCSA02 IDCSA03	LDEL3 LDEL2 LDEL1
DEQ	IDCSA06	DEQ1 LDEQ2
	IDCSA08	LDEQ1
	IDCSA10	DEQ1
DEVTYPE	IDCIO02	LDEVT1
	IDCIO03	LDEVT1
ENDREQ	IDCRP01	—
ENQ	IDCSA06	LENQ1 LENQ2 LENQ3 LENQ4 LENQ5
	IDCSA08	LENQ1
ERASE	IDCRP01	—
ESETL	IDCIO03	LESETL1
ESTAE	IDCSA10	LSTAE2
EXCP	IDCIO05	LEXCP1
EXTRACT	IDCSA08	LEXTR1

Figure 5 (Part 1 of 5). System and I/O Macros Used by Access Method Services

Macro	Module	Label
FREEMAIN	IDCIO02	LFREM1 LFREM2 LFREM3
	IDCIO05	LFREM1
	IDCSA01	LFREM3 LFREM4 LFREM5 LFREM6 LFREM14 LFREM15 LFREM16 LFREM17 LFREM18 LFREM19
	IDCSA02	LFREM7 LFREM8 LFREM9 LFREM10 LFREM11 LFREM12 LFREM13
	IDCSA03	LFREM1 LFREM2
	IDCSA08 **(SU 5752-84)**	LFREM1
FREEPOOL	IDCIO02	LFREEP1
GENCB **(Deleted for VS2.03.808)**	IDCIO02	LGEN1 LGEN2 LGEN3
GET	IDCIO01	LGET1 LGET2 LGET3
	IDCRS07 **(VS2.03.808)**	—
GETMAIN	IDCSA01	LGETM2
	IDCSA02	LGETM3 LGETM3X **(VS2.03.807)** LGETM4 LGETM4X **(VS2.03.807)** LGETM5 LGETM5X **(VS2.03.807)**
	IDCSA03	LGETM1 LGETM1X **(VS2.03.807)**
ICBACREL	IDCSA09	LACQ1
ICBCOTB	IDCSA09	LCOTB1 LTBLR1
ICBCOVC	IDCSA09	LCOVC1 LCOVC2

Figure 5 (Part 2 of 5). System and I/O Macros Used by Access Method Services

Macro	Module	Label
ICBDEFV	IDCSA09	LDEFV1
ICBMCRT	IDCSA09	LMCRT1
ICBMNTDE	IDCSA09	LMNTDE1 LMNTDE2 LMNTDE3
	IDCSA10	LICBMNT1
ICBNULLC	IDCSA09	LNULL1
ICBQUERY	IDCSA09	LRMSC1 LRMSCT1 LSA1 LRLAB1 LCUA1
ICBTRACE	IDCSA09	LTRACE1
ICBTUNE	IDCSA09	LTUNE1 LTUNE2
ICBVVIC	IDCSA09	LVVIC1
ICBDEFV	IDCSA09	LDEFV1
LINK	IDCAM01	LLINK1
	IDCAM02	LLINK2
LOAD	IDCIO01	LLOAD1
	IDCSA02	LLDA1 LLDA2 LLDA3
	IDCIO05	LRVLOAD
LOCATE	IDCSA07	LLOCAT1
MODCB **(Deleted for VS2.03.808)**	IDCIO03	LMOD1 LMOD02 LMOD03
MODESET	IDCIO05	LMODE1 LMODE2 LRVMODE1 LRVMODE2 LRVMODE3 LRVMODE4
	IDCRS07 **(VS2.03.808)**	CATSUPR CATPROB FRCSUPR FRCPROB RENMSUP1 RENMPRB1 SCRSUPR SCRPROB
	IDCSA06	LMODE1 LMODE2

Figure 5 (Part 3 of 5). System and I/O Macros Used by Access Method Services

Macro	Module	Label
OBTAIN	IDCIO03	LOBT1
OPEN	IDCIO02	LOPENJ
		LOPEN1
	IDCIO05	LOPEN1 LOPEN2 LOPEN3 LOPEN4 LOPEN5 LOPEN6 LRVOPN1 LRVOPN2
	IDCSA01	LOPEN1
	IDCSA02	LOPEN2
POINT	IDCIO03	LPOINT1 LPOINT2
PUT	IDCIO01	LPUT1 LPUT2 LPUT3
	IDCRS07 **(VS2.03.808)**	—
PUTGET	IDCSA02	LPUTG1
PUTLINE	IDCIO04	LPUTL1 LPUTL2 LPUTL3 LPUTL4
	IDCSA01	LPUTL1
RACHECK **(SU 5752-824)**	IDCSA08	LRACK1
RDJFCB	IDCIO02	LRDJF1
	IDCIO03	LRDJF1
	IDCIO05	LRDJF1
READ	IDCIO01	LREAD1
RENAME **(VS2.03.808)**	IDCRS07	RENMCML
RESERVE	IDCSA08	LRSVR1
RETURN	IDCIO05	LRETURN1
	IDCRS07 **(VS2.03.808)**	SCRCML
SCRATCH	IDCSA08	LSCRT1
SETL	IDCIO03	LSETL1
SETRP	IDCSA10	LSETRP1

Figure 5 (Part 4 of 5). System and I/O Macros Used by Access Method Services

Macro	Module	Label
SHOWCB (Deleted for VS2.03.808)	IDCIO02	LSHOW1 LSHOW2
	IDCIO03	LSHOW1
SNAP	IDCSA01	LSNAP1
	IDCSA02	LSNAP2
STACK	IDCAM01	LSTACK1
	IDCAM02	LSTACK2
STOW	IDCIO03	LSTOW1 LSTOW2
SVC 82	IDCIO05	L82SVC1 L82SVC2 LRV82SVC
	IDCSA06	L82SVC1 L82SVC2
	IDCSA10	L82SVC1
SVC 99	IDCSA02	L99SVC1 L99SVC2 L99SVC3 L99SVC4
SYNADAF	IDCIO01	LSYNF1 LSYNF2 LSYNF3 LSYNF4 LSYNF5 LSYNF6 LSYNF7
	IDCIO05	LSYNF1
SYNADRLS	IDCIO01	LSYNF1 LSYNR2
	IDCIO05	LSYNR1
TCLEARQ	IDCAM01	LTCLRQ1
	IDCAM02	LTCLRQ2
TESTCB (Deleted for VS2.03.808)	IDCIO02	LTESTCB1 LTESTCB2 LTESTCB3 LTESTCB4
TIME	IDCSA05	LTIME1 LTIME2 LTIME3
VERIFY	IDCIO01	LVRFY1
WAIT	IDCIO05	LWAIT1
	IDCSA09	LWAIT1
WRITE	IDCIO01	LWRITE1
WTO	IDCSA01	LWTO1
	IDCSA08	LWT02

Figure 5 (Part 5 of 5). System and I/O Macros Used by Access Method Services

Internal Services Provided for Processor Modules

Figure 6 contains a list of the services provided by the adapters and the Text Processor, the appropriate module name in each case, and the entry point name. Calls to the services are generated by macros defined by Access Method Services. The macros are collectively called Umacros. The listings contain only the calling sequence and not the Umacro. This publication discusses the Umacros in order to combine the calling sequence with the service performed as a function. The rightmost column lists the arguments that may be included with each of these Umacros. These arguments represent the addresses of the named items. When the argument is preceded by the symbol ↑, then it is the address of a fullword pointer to the named item. Brackets ([]) indicate an optional argument.

Service	Module	Entry Point	Description	Arguments
PROLOG	IDCSA03	IDCSAPR	Initialize a routine on entry; get storage.	module identification size of storage for module
UABORT	IDCSA01	SAABT	Handle unrecoverable error condition while processing.	UABORT code (in register 15)
UALLOC	IDCSA02	IDCSAAL	Allocate a data set or mount a volume.	GDT ALLAGL
UCALL	IDCSA02	IDCSACL	Load an executable module, if necessary, and pass control to it.	GDT entry point name [list of arguments for called module]
UCATLG	IDCSA02	IDCSACA	Catalog request.	GDT ↑catalog parameter list
UCIR	IDCSA02	IDCSACR	Obtain fully-qualified data set names from a VSAM catalog.	GDT CIRAGL
UCLOSE	IDCIO01	IDCIOCL	Close one or more data sets.	GDT OPNAGL [...]
UCOPY	IDCIO01	IDCIOCO	Copy a data set.	GDT ↑input IOCSTR ↑output IOCSTR
UDEALLOC	IDCSA02	IDCSADL	Deallocate a data set or dismount a volume.	GDT ALLAGL
UDELETE	IDCSA02	IDCSADE	Without **VS2.03.807:** Takes no action. With **VS2.03.807**: Decrements loaded module use count.	GDT module name
UDEQ	IDCSA08	IDCSADQ	Release a resource or I/O unit.	GDT major resource name minor resource name
UDUMP	IDCDB01	IDCDB01	Print diagnostic output and storage dump.	GDT dump identifier [↑symbolic dump list]
UENQ	IDCSA08	IDCSANQ	Acquire control of a resource.	GDT 'SHR' | 'EXCL' 'NOWAIT' | 'WAIT' major resource name minor resource name
UEPIL	IDCSA03	IDCSAEP	Free storage on exit from a routine.	GDT module identifier [return code]

Figure 6 (Part 1 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
UERROR	IDCTP06	IDCTPER	Without **VS2.03.807**: Verbalize Catalog Error Messages With **VS2.03.807**: Verbalize Catalog and Dynamic Allocation Error Messages	GDT ERCNVTAB
UESTA	IDCTP04	IDCTPEA	Establish a PCT, Print Control Table, from information in storage.	GDT alternate IOCSTR or zero for SYSPRINT PCARG
UESTS	IDCTP04	IDCTPES	Establish a PCT, Print Control Table, from information in Text Structures.	GDT alternate IOCSTR or zero for SYSPRINT Text Structure identification
UEXCP	IDCIO05	IDCIO05	Open data set for EXCP.	GDT 'OPEN' EXOARG
			Close data set.	GDT 'CLOSE' ↑IOXCTLBK
			Read a record.	GDT 'GET' EXGARG
			Write a record.	GDT 'PUT' EXPARG
			REPAIRV: Open a data set, VTOC, VTOC header, or staging volume.	GDT 'OPENR' EXOARG
			REPAIR: Read count, key, and data of all records on a track.	GDT 'READKD' EXWRARG
			REPAIRV: Read the count field of all records on a track.	GDT 'READCNT' EXWRARG
			REPAIRV: Bypass a defective count field.	GDT 'SPACCR' EXWRARG
			REPAIRV: Write a record's key and data field.	GDT 'WRITEREC' EXWRARG
			REPAIRV: Write multiple records on a track.	GDT 'FWRITE' EXWRARG

Figure 6 (Part 2 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
UFPOOL	IDCSA02	IDCSAFP	Release a named pool of storage.	GDT pool identification ['ALL']
UFSPACE	IDCSA02	IDCSAFS	Release unnamed storage or specific subpool of storage.	GDT address of storage to free
UGET	IDCIO01	IDCIOGT	Read a record.	GDT ↑IOCSTR
UGPOOL	IDCSA02	IDCSAGP	Allocate a named pool of storage and optionally initialize it.	GDT size of storage to obtain returned storage address pool identification ['SETZERO' | 'SETBLANK']
UGSPACE	IDCSA02	IDCSAGS	Allocate unnamed storage or specific subpool of storage, and optionally initialize it.	GDT size of storage to obtain returned storage address ['SETZERO' | 'SETBLANK' | 'NOSET', subpool id]
UID	IDCSA02	IDCSAID	Obtain a terminal user's identification.	GDT returned TSO user's identification returned number of non-blank characters in TSO user's identification
UIOINFO	IDCIO01	IDCIOSI	Obtain data-set information.	GDT option byte ↑work area dname [UGPOOL ID]
UIOINIT	IDCIO01	IDCIOIT	Initialize the I/O Adapter.	GDT [↑DDnames list or zero] [↑external routine list or zero]
UIOTERM	IDCIO01	IDCIOTM	Close all data sets that were opened with UOPEN and free all storage still used by the I/O Adapter.	GDT
ULINK	IDCSA02	IDCSALK	Link to a module.	GDT module name to link to [arguments for called module]
ULISTLN	Inline code	None	Copies the contents of register 1 into a fullword named LISTPTR and puts the number of arguments addressed by register 1 in a byte named LISTLN.	
ULOAD	IDCSA02	IDCSALD	Load a module, if necessary; but do not pass control to it.	GDT module name returned loaded module address

Figure 6 (Part 3 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
ULOCATE	IDCSA07	IDCSALC	Locate a nonVSAM data set entry in the catalog.	GDT ↑Data set name ↑Volume serial number ↑Device type ↑LCTINFO
UMSSUNIT	IDCSA06	IDCSA06	Mount a mass storage volume.	
				GDT 'MOUNT' MDAGL
			Demount a mass storage volume.	GDT 'DEMOUNT' MDAGL
			Post a UCB.	GDT 'POST' PUAGL
			Check a UCB.	GDT 'CHECK' CKAGL
			If the volume is already allocated exclusively to the job step, select a DD statement and UCB that can be used to mount and process the volume.	GDT 'SELECTX' EXCLAGL
			Change the allocation of a unit and volume mounted on that unit to exclusive.	GDT 'CHANGEX' EXCLAGL
			Select an already mounted unit and its associated ddname.	GDT 'SELECTD' SELAGL
UOPEN	IDCIO01	IDCIOOP	Open one or more data sets.	GDT OPNAGL [...]
UPOSIT	IDCIO01	IDCIOPO	Position to a logical record or physical block.	GDT ↑IOCSTR
UPRINT	IDCTP01	IDCTPPR	Format and usually write one or more lines.	GDT alternate IOCSTR or zero for SYSPRINT ↑DARGLIST [↑FMTLIST]
UPROMPT	IDCSA02	IDCSAPT	Obtain information from terminal user.	GDT ↑text for terminal number bytes in of text for terminal ↑area for terminal user's reply number of bytes in terminal user's reply
UPUT	IDCIO01	IDCIOPT	Write a record or physical block.	GDT ↑IOCSTR [ID code]

Figure 6 (Part 4 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
UQUAL	IDCSA02	IDCSAQL	Obtain a fully-qualified data set name in an interactive TSO environment.	GDT ↑unqualified data set name number of non-blank characters in unqualified data set name [↑catalog name or zero] [↑catalog password or zero]
URACHECK (VS2.08.824)	IDCSA08	IDCSARK	Request Resource Access Control Facility (RACF) authorization.	GDT RACFAGL
URECAT	IDCSA07	IDCSARC	Recatalog a nonVSAM data set.	GDT RCTAGL
URESERVE	IDCSA08	IDCSARV	Acquire control of an I/O unit.	GDT 'SHR' | 'EXCL' 'NOWAIT' | 'WAIT' ↑UCB major resource name minor resource name
URESET	IDCTP04	IDCTPRE	Re-initialize PCT, Print Control Table, for the next function.	GDT alternate IOCSTR or zero for SYSPRINT invoker's page number field
UREST	IDCTP04	IDCTPRS	Modify an existing PCT, Print Control Table	GDT alternate IOCSTR or zero for SYSPRINT arg1 arg2 arg n
USAVERC	Inline code	None	Copies the low order half of register 15 into a halfword named TESTRC.	
USCRATCH	IDCSA08	IDCSASC	Delete a data set stored on one or more volumes.	GDT data-set name ↑volume list ['OVERRIDE']
USNAP	IDCSA02	IDCSASN	Call for a SNAP dump.	GDT SNAP dump identification number
USSC	IDCSA09	IDCSASS	Issue MSSC macros and optionally wait for a response.	GDT 'ACQUIRE' | 'MOUNT' | 'DEMOUNT' | 'DEFINE' | 'MOVE' | 'RELINQ' 'TRACEQ' | 'COPYTABL' | 'COPYCRTG' | 'COPYVOL' | 'VVIC' | 'TUNE'
USTAE	IDCSA10	IDCSAST	Establish or cancel an ESTAE environment.	GDT 'SET' | 'CANCEL'

Figure 6 (Part 5 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
USTOW	IDCIO01	IDCIOST	Delete or rename a member of a partitioned data set.	GDT ↑IOCSTR 'D' | 'R'
USYSINFO	IDCSA08	IDCSASI	Return job and step name or unit information.	GDT 'NAMES' | 'UNIT' ↑return area [dname]
UTIME	IDCSA02	IDCSATI	Get date and time of day.	GDT field for returned time [field for returned date] ['FORM' | 'HSEC' | 'KLOK'
UTRACE	Inline code	None	Adds the current identification to the Inter-Module Trace Table.	
UUNCATLG	IDCSA07	IDCSAUC	Uncatalog and scratch a nonVSAM data set.	GDT UCTAGL
UVERIFY	IDCIO01	IDCIOVR	Issue VSAM VERIFY macro.	GDT ↑IOCSTR
UWTO	IDCSA08	IDCSAWO	Write message to the console operator.	GDT message length ↑message text [routine code] [description code]

Volume Services

VTOC

Service	Module	Entry Point	Description	Arguments
GETVTOC	IDCVS01	IDCVS01	Retrieve alternate track and VSAM ownership fields from the VTOC.	GDT 'GETVTOC' VS1AGL
PUTVTOC	IDCVS01	IDCVS01	Update the VSAM time stamp and alternate track fields on the VTOC.	GDT 'PUTVTOC' VS1AGL
RECATLG	IDCVS01	IDCVS01	Recatalog each nonVSAM data set found on the VTOC	GDT 'RECATLG' VS1AGL
SCRVTOC	IDCVS01	IDCVS01	Scratch all data sets from the VTOC and set VSAM ownership field to zero.	GDT 'SCRVTOC' VS1AGL
SECCHECK	IDCVS01	IDCVS01	Perform security checking for both VSAM and nonVSAM data sets	GDT 'SECCHECK' VS1AGL

Figure 6 (Part 6 of 7). Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
GETLABEL	IDCVS02	IDCVS02	Retrieve owner name and/or volume serial number from the volume label.	GDT _GETLABEL' VS2AGL
INITVOL	IDCVS02	IDCVS02	Initialize a volume by writing the VTOC and volume label.	GDT 'INITVOL' VS2AGL
PUTLABEL	IDCVS02	IDCVS02	Update the owner name and/or volume serial number in the volume label.	GDT 'PUTLABEL' VS2AGL
SELECTDS	IDCVS03	IDCVS03	Searches 3330 or 3330V for the data sets that will meet criteria as specified by the user.	GDT 'SELECTDS' VS3AGL

Figure 6 (Part 7 of 7). Internal Services Provided for Processor Modules

Processor Invocation

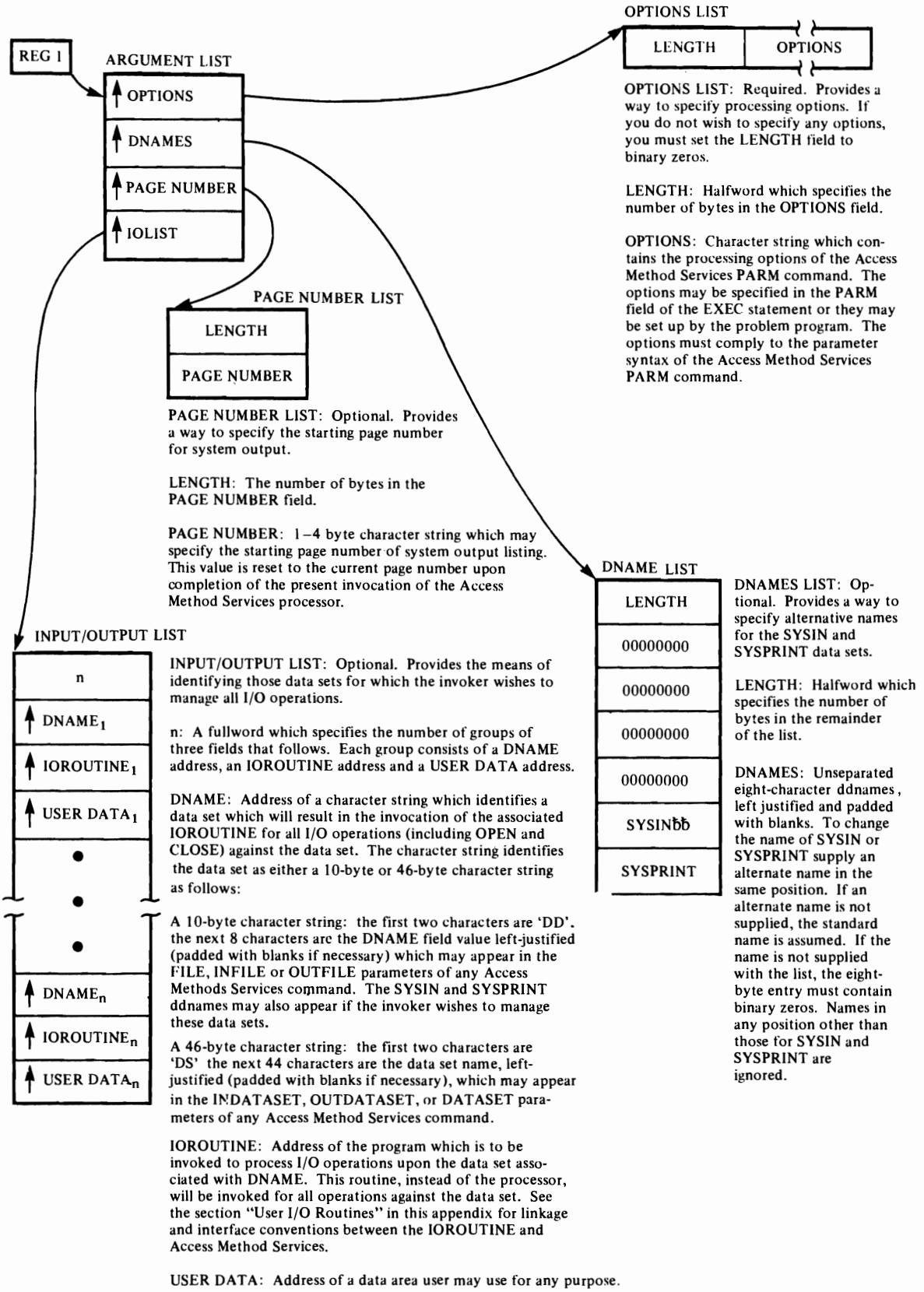
There are 2 ways to invoke the Access Method Services processor:

- A Batched Job—When you use a batched job to invoke the processor, you can give control to the processor either with JCL (// EXEC PGM=IDCAMS) or with a subroutine call to the processor from a program. The processor normally resides in the system data set SYS1.LINKLIB. The register contents and the processor's argument list are the same no matter how you give the processor control. The register contents and argument list are set by either the operating system if you give the processor control with JCL or the calling program if you give the processor control with a subroutine call.

  The register contents comply with standard linkage conventions; that is, register 1 contains the address of the argument list, register 13 contains the address of a save area, register 14 the address of the return location, and register 15 contains the address of the entry point IDCAMS in module IDCSA01 of the System Adapter. On exit from the Access Method Services processor, register 15 contains the value of MAXCC (see the section "Processor Condition Codes" below).

  The argument list for a batched job is shown in Figure 7, can be a maximum of four fullword addresses pointing to strings of data. The last address in the list contains a '1' in the sign field. The first three possible strings of data begin with a two-byte length field. A null element in the list is indicated by either an address of zero or a length of zero.

- Interactively with TSO—When you use interactive TSO to invoke the processor, you can give control to the processor each time you type any Access Method Service command except PARM and modal commands. PARM and modal commands—IF-THEN, ELSE, DO, END, and SET—are not allowed with interactive TSO. The TSO Terminal Monitor Program (TMP) gives control to IDCAM01 or IDCAM02 which reside in the TSO command library data set, SYS1.CMDLIB. IDCAM01 and IDCAM02 have aliases of all the verbs and verb abbreviations permitted



OPTIONS LIST: Required. Provides a way to specify processing options. If you do not wish to specify any options, you must set the LENGTH field to binary zeros.

LENGTH: Halfword which specifies the number of bytes in the OPTIONS field.

OPTIONS: Character string which contains the processing options of the Access Method Services PARM command. The options may be specified in the PARM field of the EXEC statement or they may be set up by the problem program. The options must comply to the parameter syntax of the Access Method Services PARM command.

PAGE NUMBER LIST: Optional. Provides a way to specify the starting page number for system output.

LENGTH: The number of bytes in the PAGE NUMBER field.

PAGE NUMBER: 1–4 byte character string which may specify the starting page number of system output listing. This value is reset to the current page number upon completion of the present invocation of the Access Method Services processor.

INPUT/OUTPUT LIST: Optional. Provides the means of identifying those data sets for which the invoker wishes to manage all I/O operations.

n: A fullword which specifies the number of groups of three fields that follows. Each group consists of a DNAME address, an IOROUTINE address and a USER DATA address.

DNAME: Address of a character string which identifies a data set which will result in the invocation of the associated IOROUTINE for all I/O operations (including OPEN and CLOSE) against the data set. The character string identifies the data set as either a 10-byte or 46-byte character string as follows:

A 10-byte character string: the first two characters are 'DD'. the next 8 characters are the DNAME field value left-justified (padded with blanks if necessary) which may appear in the FILE, INFILE or OUTFILE parameters of any Access Methods Services command. The SYSIN and SYSPRINT ddnames may also appear if the invoker wishes to manage these data sets.

A 46-byte character string: the first two characters are 'DS' the next 44 characters are the data set name, left-justified (padded with blanks if necessary), which may appear in the INDATASET, OUTDATASET, or DATASET parameters of any Access Method Services command.

IOROUTINE: Address of the program which is to be invoked to process I/O operations upon the data set associated with DNAME. This routine, instead of the processor, will be invoked for all operations against the data set. See the section "User I/O Routines" in this appendix for linkage and interface conventions between the IOROUTINE and Access Method Services.

USER DATA: Address of a data area user may use for any purpose.

DNAMES LIST: Optional. Provides a way to specify alternative names for the SYSIN and SYSPRINT data sets.

LENGTH: Halfword which specifies the number of bytes in the remainder of the list.

DNAMES: Unseparated eight-character ddnames, left justified and padded with blanks. To change the name of SYSIN or SYSPRINT supply an alternate name in the same position. If an alternate name is not supplied, the standard name is assumed. If the name is not supplied with the list, the eight-byte entry must contain binary zeros. Names in any position other than those for SYSIN and SYSPRINT are ignored.

Figure 7. Argument List for Processor Invocation with a Batched Job

with interactive TSO. IDCAM01 and IDCAM02 contain identical code. The TMP gives IDCAM01 or IDCAM02 an argument list as shown in Figure 9. For more information on the TMP, refer to *OS/VS2 TSO Terminal Monitor Program and Service Routines Logic*. IDCAM01 or IDCAM02 uses a LINK macro to give control to the processor at entry point IDCSATO. The processor—except modules IDCAM01 and IDCAM02—normally resides in the system data set SYS1.LINKLIB.

The register contents comply with standard linkage conventions; that is register 1 contains the address of the argument list, register 13 contains the address of a save area, register 14 the address of the return location in IDCAM01 or IDCAM02, and register 15 contains the address of entry point IDCSATO in module IDCSA01 of the System Adapter. On return to the TMP, register 15 contains the value of MAXCC (see the section "Processor Condition Codes" below).

The argument list is the same as the argument list the TMP gave IDCAM01 or IDCAM02—shown in Figure 8.

Processor Condition Codes

The processor's condition code is LASTCC, which can be interrogated in the command stream with modal commands. The possible values and their meanings are in the following table. The table illustrates the value of LASTCC.

Code	Meaning
0	The function was executed as directed and expected. Informational messages may have been issued.
4	Some annoyance in executing the complete function was met, but it was possible to continue. The results might not be exactly what the user wants, but no permanent harm appears to have been done by continuing. A warning message was issued.
8	A function could not perform all that was asked of it. The function was completed, but specific details were bypassed.
12	The entire function could not be performed.
16	Severe error or problem encountered. Remainder of command stream is flushed and processor returns condition code 16 to the operating system.

The LASTCC condition code is reflected in its related message numbers. The first numeric character of the message number equals the condition code

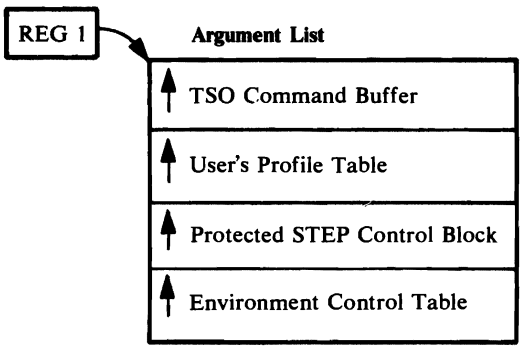


Figure 8. Argument List for Processor Invoked Interactively with TSO

divided by 4. MAXCC, which can also be interrogated in the command stream, is the highest value of LASTCC thus far encountered.

User I/O Routines

If the user has supplied his own I/O routine, the I/O Adapter invokes the user routine. Again, standard linkage is used. Figure 9 shows the arguments passed to the user routine. Each field begins on a fullword boundary.

When writing a user I/O routine, the user must be aware of three things. First, the processor handles the user data set as if it were a nonVSAM data set that contains variable length unblocked records (maximum length—32,760 bytes) with a physical sequential organization. The processor does not test for a JCL statement for the data set. Therefore, the name can be anything. Second, the processor formats data in various ways. The user must know what the format is so that the user's routine can be coded to handle the correct type of input and format the correct type of output. (See "Diagnostic Aids" for more information). Third, each user supplied I/O routine must handle any error messages and provide to the processor a return code in register 15. The processor uses the return code to determine what it is to do next.

The permissible codes are:

- 0 — Operation successful.
- 4 — End of data for a GET operation.
- 8 — Error occurred during a GET/PUT operation but continue processing.
- 12 — (1) Do not allow any further calls (except for CLOSE) to this routine.

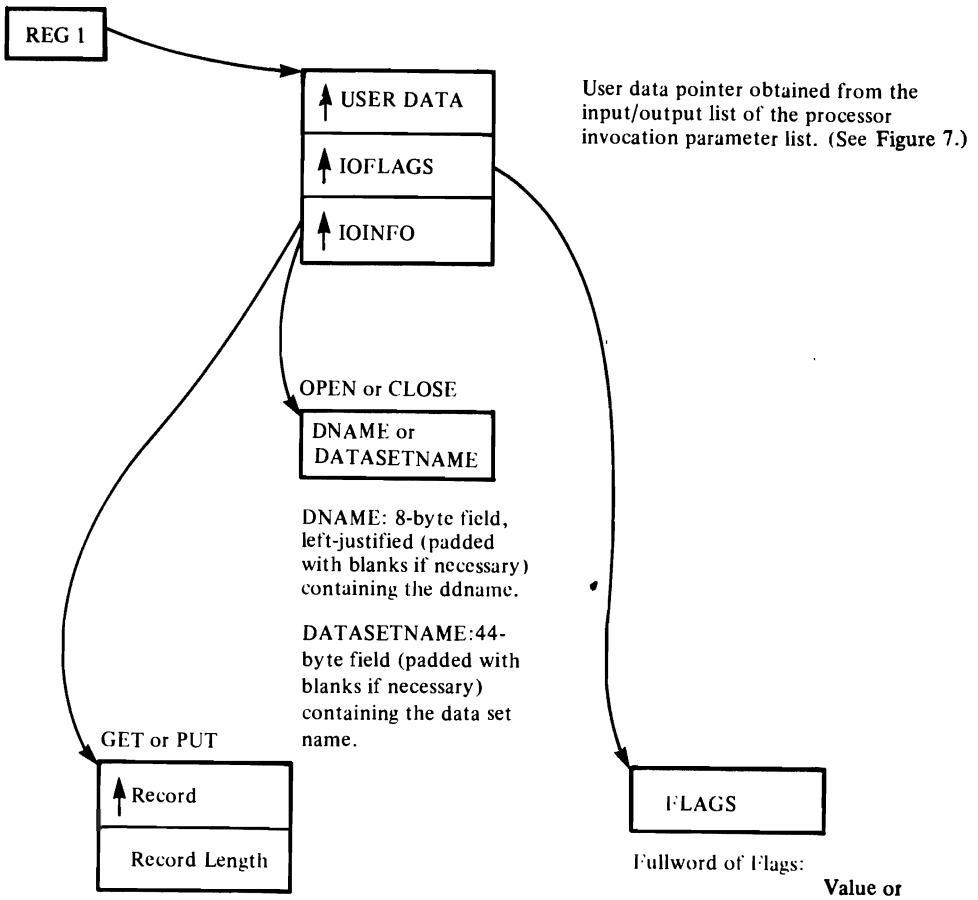


Figure 9. Arguments Passed to and from User I/O Routine

Overall Control Flow

Figures 10 and 11 illustrate the overall control flow through the processor. If the processor is invoked with a batched job, Figure 10 shows the control flow. If the processor is invoked interactively with TSO, Figure 11 shows the control flow. Entry and exit are through IDCSA01. IDCEX01 is the main controller; it alternates control between the Reader/Interpreter and the FSRs to process each command. When all commands are processed or a severe error has occured, IDCEX01 gives control to IDCEX03. After IDCEX03 completes, IDCEX01 returns to IDCSA01.

All modules in Figures 10 and 11 call the modules in Figure 12 for services (like writing a record). The addresses of the entry points to the service modules are kept in the GDT. All modules in Figure 12 also call each other for services.

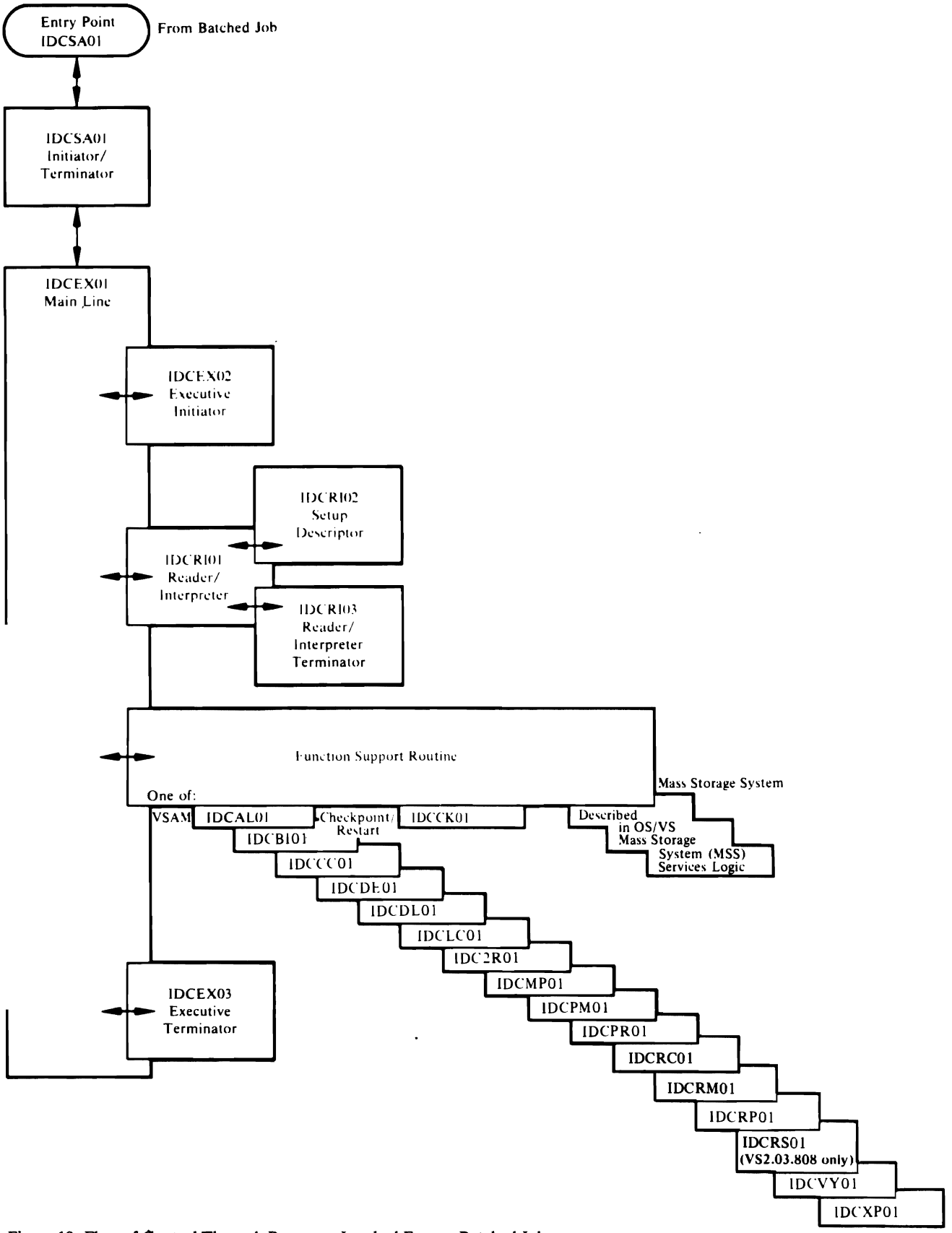


Figure 10. Flow of Control Through Processor Invoked From a Batched Job

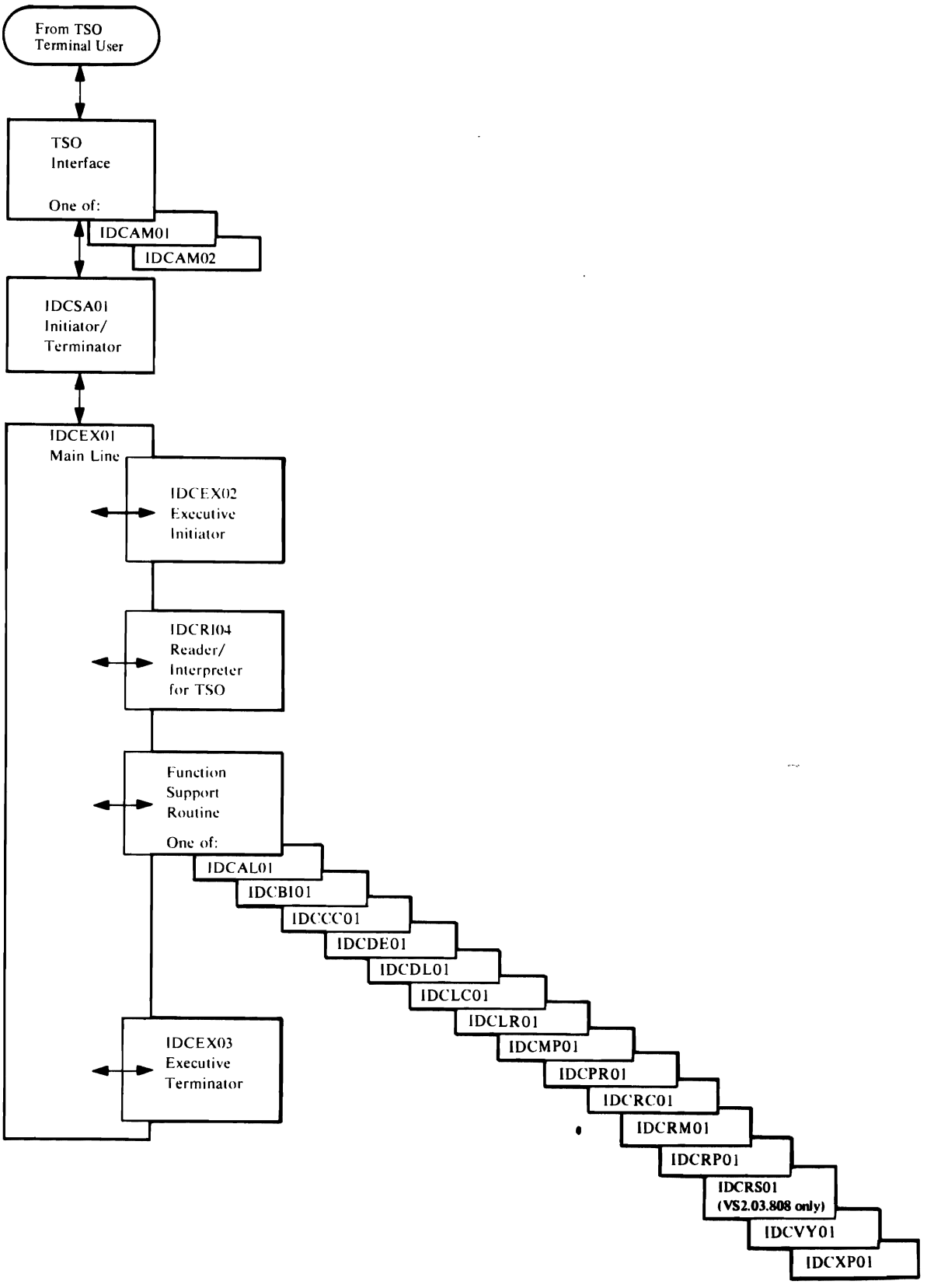


Figure 11. Flow of Control Through Processor Invoked Interactively with TSO

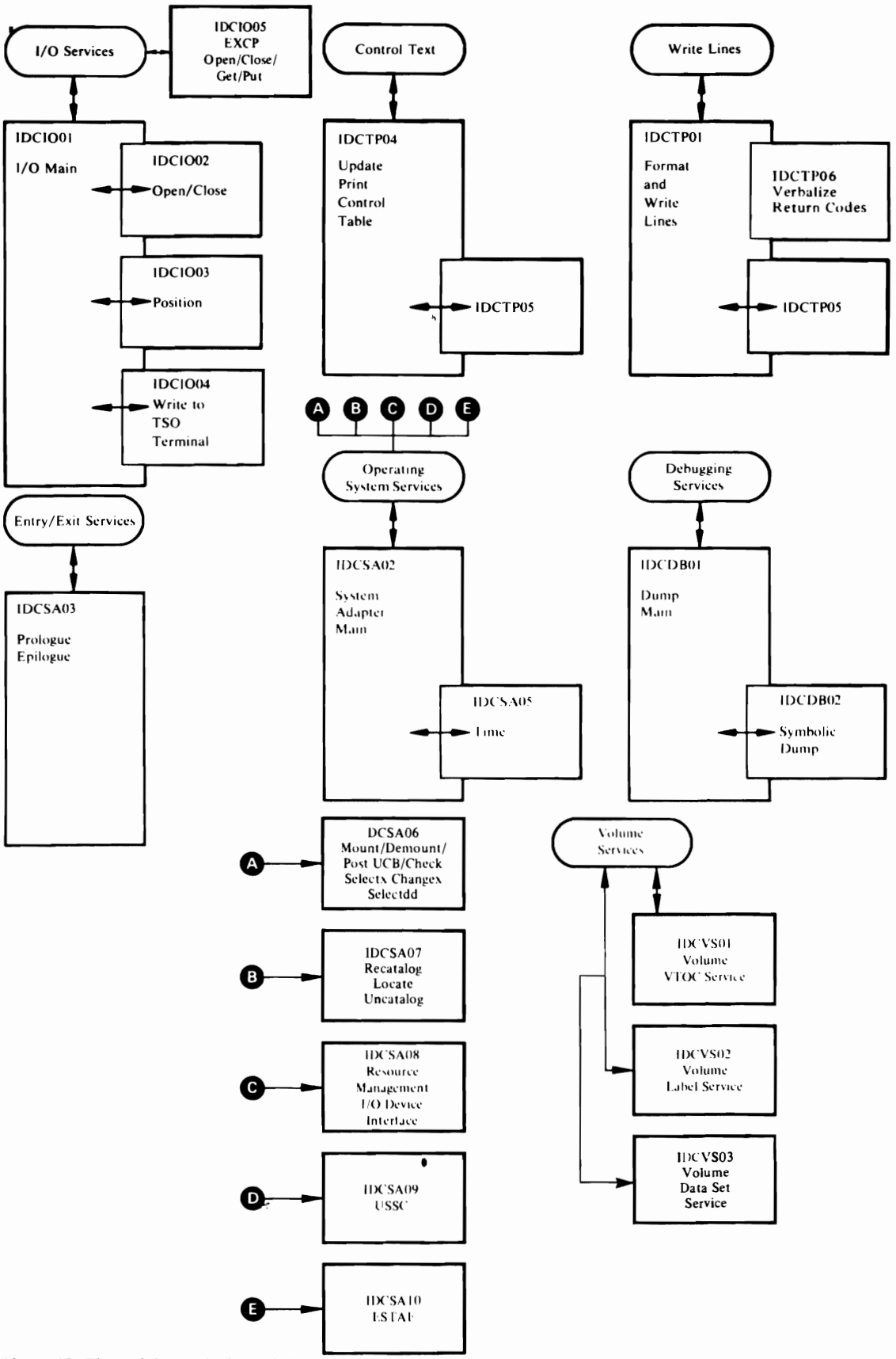


Figure 12. Flow of Control Through Services

MICROFICHE DIRECTORY

This chapter contains a directory to the microfiche listings for all modules of the processor. This directory describes the contents of each module by function and label, allowing you to quickly find any desired code.

The processor is written in PL/S II, a high-level, IBM proprietary system language. Listings that are produced for microfiche consist of the PL/S II source code, a cross-reference and attribute table, and the assembly code. See the IBM publication *Guide to PL/S II* for a more detailed explanation of PL/S II and its listings.

Each module is designed with "GOTO-less" code; that is, there are no explicit GOTOs or branches. All conditional phrases are contained within IF-THEN-ELSE clauses and DO-WHILE clauses of PL/S II. All loops are controlled by DO statements. Extensive use of closed subroutines (procedures) is made.

The microfiche for each module begins with the PL/S II portion, which contains all commentary and is the most readable form of the program. All data areas are defined at the beginning of the listing. IF-THEN-ELSE clauses and DO-loops are indented to denote levels of logic. The cross-reference and attribute table shows each use of each data area. The assembly listing is keyed back to the PL/S II source statement numbers.

The listings are extensively commented. Each module begins with a prologue commentary that lists all standard information for that module. Each internal procedure has a small prologue to further describe its function.

Note: The listings use CPL, FVT, and FPL instead of CTGPL, CTGFV, and CTGFL, respectively. See *OS/VS2 Virtual Storage Access Method (VSAM) Logic* for a description of these data areas. The listings also refer to TSO data areas such as: PCL, PDE, PDL, CPPL. See *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor* for a description of TSO data areas.

In the following tables, the module name appears in the first (leftmost) column. The second column contains an entry-point label, the label of an internal procedure (subroutine), or the label of data used externally—that is, by another module. The third column differentiates between entry points (EP), procedures (PR), and data used externally (DE).

CSECT/Load Module Name	Label	Use	Descriptions
IDCAL01			ALTER FSR; modifies an existing catalog entry. Translates the encoded command parameters into the necessary catalog parameter lists and calls IDCSACA for a catalog request (UCATLG macro).
	IDCAL01	EP	Only entry point to this module.
	ALTERPRC	PR	Builds the VSAM catalog management interface for the alter request.
	CHECKPRC	PR	Verifies that new values are compatible.
	DALCPROC	PR	Manages dynamic allocation.
	INDEXPRC	PR	Builds the VSAM catalog management interface to alter the index component if keys has been specified.

CSECT/Load Module Name	Label	Use	Descriptions
IDCAL01 (continued)	LOCATPRC	PR	Locates catalog fields which must be altered in context. LOCATRPC only locates fields which contain multiple attributes. Since the user may wish to change only one of several attributes, the original field serves as the basis for alternation.
	MEMRENAM	PR	Renames members of a partitioned data set.
	PARAMCHK	PR	Verifies that parameters specified on the command are valid for the type of object to be altered.
IDCAMS		EP	Root segment for Access Method Services; consists of IDCEX01, IDCEX02, IDCEX03, IDCIO01, IDCIO02, IDCIO03, IDCIO05, IDCSA01, IDCSA03, IDCSA04, IDCSA05, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10, IDCTP01, IDCTP04, IDCTP05, and IDCTP06. See this chapter for a description of these modules.
IDCAM01			Interfaces with interactive TSO. If Access Method Services is invoked interactively with TSO, TSO gives control to this module or IDCAM02. IDCAM01 gives control to the System Adapter at entry point IDCSATO. Modules IDCAM01 and IDCAM02 contain identical code. See the chapter "Program Organization" for more information.
	IDCAM01	EP	Only entry point to this module. This module has the name of every functional command—except PARM and CNVTCAT—as an alias. IDCAM01 resides in the TSO command library, SYS1.CMDLIB.
IDCAM02			Interfaces with interactive TSO. If Access Method Services is invoked interactively with TSO, TSO gives control to this module or IDCAM01. IDCAM02 gives control to the System Adapter at entry point IDCSATO. Modules IDCAM02 and IDCAM01 contain identical code. See the chapter "Program Organization" for more information.
	IDCAM02	EP	Only entry point to this module. This module has the name of the CNVTCAT command as an alias. IDCAM02 resides in the TSO command library, SYS1.CMDLIB.
IDCBI01			BLDINDEX FSR; build one or more alternate indexes over a defined, nonempty base cluster.
	IDCBI01	EP	Only entry point to this module.
	OPENPROC	PR	Opens the data sets required by the BLDINDEX FSR - base cluster, alternate index and, optionally, sort work files - by issuing UOPEN.
	JCPROC	PR	Issues the UIOINFO macro to determine if caller supplied sort work job control; obtains data set name and volume serial.
	MAINPROC	PR	Controls the build process for one alternate index by calling OPENPROC, LOCPROC, INITPROC, CNTLPROC.
	FINPROC	PR	Closes alternate index, sort work files, and issues alternate index final status message.
	TERMPROC	PR	Closes base cluster, frees resources, and prints termination message.
	LOCPROC	PR	Controls sequence of catalog locates to obtain information regarding base cluster and alternate index; verifies relationship.

CSECT/Load Module Name	Label	Use	Descriptions
	CATPROC	PR	Constructs CPL and PLs for catalog locate and calls VSAM catalog management via UCATLG.
	DEFPROC	PR	Constructs CPL, FVTs and FPLs and calls VSAM catalog management to define sort work files; opens defined files.
	DELTPROC	PR	Constructs CPL and calls VSAM catalog management to delete sort work files.
	INITPROC	PR	Determines resources required for building alternate index and obtains core for work areas and sorting.
	CNTLPROC	PR	Controls actual build by reading base cluster and calling SORTPROC and MERGPROC or BLDPROC to perform sort-merge and write alternate index records.
	SORTPROC	PR	Constructs records; performs the entire internal sort or the initial sort phase of an external sort.
	SPILPROC	PR	Writes out initial strings to first sort work file in an external sort.
	BLDPROC	PR	Builds and writes the alternate index records from the sequenced sort records.
	MERGPROC	PR	Performs the merge passes of an external sort.
IDCCC01			CNVTCAT FSR; converts OS/VS catalog entries to VSAM catalog entries.
	IDCCC01	EP	Only entry point for this module. Opens and closes catalogs.
	AEPROC	PR	Processes SYSCTLG alias entries (AE).
	ALTRLSTS	PR	Builds a CTGPL, CTGFV, and CTGFLs for altering a VSAM catalog entry.
	CATALOG	PR	Invokes VSAM catalog management to define, locate, and alter the converted SYSCTLG entries.
	CONTINUE	PR	Obtains a SYSCTLG continuation index block at the same index level as the last block.
	CONVERT	PR	Determines type of entry in SYSCTLG data set.
	CVPEPROC	PR	Processes SYSCTLG control volume pointer entries (CVPE).
	CVTINIT	PR	Initializes scan of SYSCTLG, obtains storage, and prepares SYSPRINT subtitle line.
	DEFNLSTS	PR	Builds a CTGPL, CTGFV, and CTGFLs for defining a VSAM catalog entry.
	DSPEPROC	PR	Processes SYSCTLG data set pointer entries (DSPE).
	ERRPROC	PR	Builds error conversion table and invokes UERROR for all VSAM catalog (SVC26) errors.
	GIPEPROC	PR	Processes SYSCTLG generation index pointer entries (GIPE).
	ILEPROC	PR	Processes SYSCTLG index link entries (ILE).
	LOCTLSTS	PR	Builds a CTGPL and CTGFLs for locating a VSAM catalog entry.
	POPUP	PR	Goes to a higher level SYSCTLG index block than the last index block.
	PUSHDOWN	PR	Obtains and verifies a lower level SYSCTLG index block.
	VCBPPROC	PR	Processes SYSCTLG volume control block pointer entries (VCBPE) and volume control blocks (VBC).
	VOLINDEX	PR	Obtains and verifies SYSCTLG volume index.
IDCCDAL			Command Descriptor for ALTER command.
IDCCDBI			Command Descriptor for BLDINDEX verb.
IDCCDCC			Command Descriptor for CNVTCAT command.

CSECT/Load Module Name	Label	Use	Descriptions
IDCCDCK			Command Descriptor for CHKLIST command.
IDCCDDE			Command Descriptor for DEFINE command.
IDCCDDL			Command Descriptor for DELETE command.
IDCCDLC			Command Descriptor for LISTCAT command.
IDCCDLR			Command Descriptor for LISTCRA verb.
IDCCDMP			Command Descriptor for IMPORT command.
IDCCDPM			Command Descriptor for PARM command.
IDCCDPR			Command Descriptor for PRINT command.
IDCCDRC			Command Descriptor for EXPORTRA verb.
IDCCDRM			Command Descriptor for the IMPORTRA verb.
IDCCDRP			Command Descriptor for REPRO command.
IDCCDRS (**VS2.03.808** only)			Command Descriptor for RESETCAT command.
IDCCDVY			Command Descriptor for VERIFY command.
IDCCDXP			Command Descriptor for EXPORT command.
IDCCK01			CHKLIST FSR: lists identifying information for tape data sets open at the time of a checkpoint.
	IDCCK01	EP	Only entry point to this module.
	HSKGPROC	PR	Opens files and sets up subheadings.
	BUILDTAB	PR	Builds table of selected checkpoint IDs.
	CHRPROC	PR	Locates the CHR records for entries to be processed.
	GETCHR	PR	Reads the next CHR record.
	DSDRPROC	PR	Extracts and prints the tape data-set information from DSDR records.
	DSDRVOLS	PR	Extracts and prints volume serial numbers from type 2 DSDR records.
	GETNEXT	PR	Locates the next sequential logical DSDR record in the checkpoint data set.
	DSDRLIST	PR	Prints volume sequence numbers and indicates volume mounted at checkpoint.
IDCDB01			Provides a dump (UDUMP macro).
	IDCDB01	EP	Only entry point to this module.
IDCDB02			Provides a symbolic dump.
	IDCDB02	EP	Only entry point to this module.
	ARRAYHDR	PR	Processes any array header elements (TYPE='A') in the dump list.
	BCONVERT	PR	Converts the value of the current dump item to binary representation.
	CCONVERT	PR	Converts the value of the current dump item to character representation.
	FCONVERT	PR	Converts the value of the current dump item to fixed-integer representation.
	HCONVERT	PR	Converts the value of the current dump item to hexadecimal representation.
	ITEMDUMP	PR	Processes any individual dump list elements.
	NAMEFLD	PR	Inserts the symbolic name of the dump element into the proper position of the output line.

CSECT/Load Module Name	Label	Use	Descriptions
IDCDE01 (without **VS2.03.807**)			DEFINE FSR; creates an entry in a VSAM catalog.
	IDCDE01	EP	Only entry point to this module.
	AIXPROC	PR	Oversees the construction of the CTGPL, CTGFV, and CTGFL for defining a VSAM alternate index.
	CTLGPROC	PR	Oversees the construction of the CTGPL, CTGFV, and CTGFL for defining a VSAM master or user catalog.
	DALCPROC	PR	Allocates and deallocates volumes.
	DSETPROC	PR	Oversees the construction of VSAM data sets and PAGESPACE data sets.
	DSPACPRC	PR	Oversees the construction of the VSAM catalog interface for defining VSAM data spaces.
	MODELPRC	PR	Handles the retrieval of model objects to be used in defining components of VSAM user catalogs and clusters.
	NVSAMPRC	PR	Oversees the construction of the VSAM catalog interface for defining a non-VSAM data set, ALIAS, or generation data group into a VSAM catalog.
	PATHPROC	PR	Oversees the construction of the VSAM catalog. Interface for defining a VSAM path object.
IDCDE02 (without **VS2.03.807**)			Contains the DEFINE FSR third level routines which process parameters specified on the command.
	IDCDE02	EP	Initializes registers and obtains storage.
	ALLCPROC	EP	Initializes several allocation parameters in the CTGFL and CTGFV.
	FREESTG	EP	Manages storage deallocation for IDCDE02 CSECT.
	IXOPPROC	EP	Initializes index options.
	KEYPROC	EP	Initializes the record management control block and the key range "pseudo-field" in the CTGFL.
	NAMEPROC	EP	Initializes the data set creation and expiration dates in the CTGFL and the object name in the CTGFV. NAMEPROC handles the RELATE parameter and exception exit. For generation data groups NAMEPROC initializes the SCRATCH | NOSCRATCH, EMPTY | NOEMPTY, and LIMIT parameters.
	PROTPROC	EP	Initializes the security and owner identification fields and the SHAREOPTIONS and ERASE | NOERASE flags in the CTGFL.
IDCDE01 **(with VS2.03.807)**			DEFINE FSR; creates an entry in a VSAM catalog.
	IDCDE01	EP	Only entry point to this module.
	DALCPROC	PR	Allocates and deallocates volumes.
	INTGCHK	PR	Validity checks completed parameter list before catalog invocation.

CSECT/Load Module Name	Label	Use	Descriptions
IDCDE02 (with VS2.03.807)			Contains third level routines which process parameters specified on the command.
	IDCDE02	EP	Establishes addressability and allocates automatic storage for third level routines.
	ALLCPROC	EP	Initializes several allocation parameters in the CTGFL and CTGFV.
	FREESTG	EP	Manages storage deallocation for IDCDE02 CSECT.
	IXOPPROC	EP	Initializes index options.
	KEYPROC	EP	Initializes the record management control block and the key range "pseudo-field" in the CTGFL.
	MODELEPC	EP	Handles the retrieval of model objects to be used in defining components of VSAM user catalogs and clusters.
	NAMEPROC	EP	Initializes the data set creation and expiration dates in the CTGFL and the object name in the CTGFV. NAMEPROC handles the RELATE parameter and exceptionexit. For generation data groups NAMEPROC initializes the SCRATCH | NOSCRATCH, EMPTY | NOEMPTY, and LIMIT parameters.
	PROTPROC	EP	Initializes the security and owner identification fields and the SHAREOPTIONS and ERASE | NOERASE flags in the CTGFL.
IDCDE03 (with VS2.03.807)			Determines the type of object being defined and calls the appropriate internal procedure.
	IDCDE03	EP	Only entry point to this module.
	AIXPROC	PR	Oversees the construction of the CTGPL, CTGFV, and CTGFL for defining a VSAM alternate index
	CTLGPROC	PR	Oversees the construction of the CTGFL, CTGFV, and CTGFL for defining a VSAM master or user catalog.
	DSETPROC	PR	Oversees the construction of VSAM data sets and PAGESPACE data sets.
	DSPACPRC	PR	Oversees the construction of the VSAM catalog interface for defining VSAM data spaces.
	NVSAMPRC	PR	Oversees the construction of the VSAM catalog interface for defining a non-VSAM data set, ALIAS, or generation data group into a VSAM catalog.
	PATHPROC	PR	Oversees the construction of the VSAM catalog interface for defining a VSAM path object.
IDCDL01			DELETE FSR; deletes a catalog entry from the VSAM catalog.
	IDCDL01	EP	Only entry point to this module.
	ALLOPROC	PR	Dynamically allocates and unallocates data sets and volumes.
	BUILDCPL	PR	Constructs the CTGPL from parameters and specifies in the DELETE command and indicates in the FDT.
	CATCALL	PR	Calls VSAM catalog management to delete a catalog entry.
	FINDTYPE	PR	Locates the entry to be deleted in order to determine its type when type is not specified in command.
	MEMDLETE	PR	Deletes partitioned data set members from the partitioned data set directory.
	MORESP	PR	Obtains a larger work area for catalog management.

CSECT/Load Module Name	Label	Use	Descriptions
	PARAMCHK	PR	Checks for invalid type specification and other command errors.
	RC240PRC	PR	Does validity checking when the user specifies the CATALOG parameter and unites the FILE parameter.
IDCEX01			Main-line for Executive; routes control through processor.
	IDCEX01	EP	Only entry point to this module; entered from IDCSA01.
	CALLFSR	PR	Calls the FSR named by the Reader/Interpreter.
	CALLRI	PR	Calls the Reader/Interpreter.
	MAIN	PR	Alternates control between the Reader/Interpreter and the FSR for each command.
IDCEX02			Initializes the processor.
	IDCEX02	EP	Only entry point to this module.
	SCANPARM	PR	Scans processor invocation parameter list.
IDCEX03			Terminates processing.
	IDCEX03	EP	Only entry point to this module.
	SCANPARM	PR	Scans invoker's parameter list to return next available page number.
IDCIO01			Supplies all I/O services to the processor. At each of the following entry points, IDCIO01 converts the service request to the appropriate system macros.
	IDCIOCL	EP	Close 1 to 4 data sets by calling IDCIO02 (UCLOSE macro).
	IDCIOCO	EP	Copies a data set (UCOPY macro).
	IDCIOGT	EP	Reads a record (UGET macro).
	IDCIOIT	EP	First call to I/O Adapter; initializes the adapter for subsequent calls.
	IDCIOOP	EP	Opens 1 to 4 data sets by calling IDCIO02 (UOPEN macro).
	IDCIOPO	EP	Positions to a specified record or physical block in a data set by calling IDCIO03 (UPOSIT macro).
	IDCIOPT	EP	Writes a record (UPUT macro).
	IDCIOSI	EP	Retrieves data-set information by calling IDCIO03 (UIOINFO macro).
	IDCIOST	EP	Stows a member name in a partitioned data set directory (USTOW macro).
	IDCIOTM	EP	Last call to the I/O Adapter; closes any data sets still open (UIOTERM macro).
	IDCIOVY	EP	Verifies a VSAM data set (UVERIFY macro).
	BLDAMSG	PR	Prepares an error message.
	CHANGE (**VS2.03.808**)	PR	Handle change of processing mode for RPL.
	COPYCAT	PR	Copies a VSAM catalog.
	GETEXT	PR	Calls a user routine to get a data record.
	GETNONVS	PR	Reads a non-VSAM data set.
	GETVSAM	PR	Gets a logical record from a VSAM data set.
	IDCIOS1	DE	Amount of storage IDCIO01 needs. Used by IDCSA01.
	IRAMEOD	PR	End-of-data-set exit routine for VSAM data sets.
	IRIEVAB	PR	End-of-volume abend exit routine for non-VSAM input data sets.
	IROEVAB	PR	End-of-volume abend exit routine for non-VSAM output data sets.

CSECT/Load Module Name	Label	Use	Descriptions
IDCIO01 (continued)	IROSEOD	PR	End-of-data-set exit routine for non-VSAM data sets.
	IRSISYN	PR	Exit routine for input errors on a QSAM data set.
	IRSOSYN	PR	Exit routine for output errors on a QSAM data set.
	PRINTMSG	PR	Prints a message.
	PUTEXT	PR	Calls a user routine for output.
	PUTNONVS	PR	Writes to a non-VSAM data set.
	PUTREP	PR	Handles PUT (Replace) processing.
	PUTVSAM	PR	Puts a logical record to a VSAM data set.
	VSAMERR	PR	Builds VSAM error message argument list.
IDCIO02			Open/Close routine. This routine opens or closes 1 to 4 data sets with one call.
	IDCIO02	EP	Only entry point to this module.
	ABEXTRN	PR	"Abend" during open or close.
	BLDOCMSG	PR	Prepares an error message.
	BUILDACB	PR	Builds ACB and EXLST for a VSAM data set to be opened.
	BUILDDBK	PR	Builds a DCB for a non-VSAM data set to be opened.
	BUILDRPL	PR	Builds RPL for a VSAM data set and gets workareas for input buffers.
	CKNONOP	PR	Checks that a non-VSAM data set was opened successfully.
	CLOSERTN	PR	Closes data sets that were opened by the I/O Adapter.
	DSDATA	PR	Checks for alternate DD name and issues the DEVTYPE and RDJFCB macros.
	ENVFREE	PR	Free storage used for a data set; system areas, buffers, control blocks, etc.
	IRTOPEX	PR	Ensures that valid SYSIN and SYSOUT DCB parameters exist and sets DCB values in IOCSTR.
	OPENRTN	PR	Opens data sets.
	PRINTMSG	PR	Calls Text Processor to print error message.
IDCIO03			Positions to a logical record or physical block. Alters a partitioned data set directory.
	IDCIO03	EP	Only entry point to this module.
	BLDAMSG	PR	Prepares error message.
	IRISSYN	PR	Processes I/O errors encountered during ISAM SETL.
	PRNTMSG	PR	Prints message.
	PTAMDS	PR	Points to VSAM logical record.
	PTISDS	PR	Positions to ISAM logical record by using a SETL macro.
	STOWRTN	PR	Stows a member name in a partitioned data set directory.
	DSINFO	PR	Retrieves data-set information.
IDCIO04			Prints the SYSPRINT data set on a TSO terminal. This module is used when Access Method Services is invoked interactively with TSO.
	IDCIO04	EP	Only entry point to this module.

CSECT/Load Module Name	Label	Use	Descriptions
IDCIO05			Performs the steps necessary to open, close, read, and write volumes using the EXCP access method.
	IDCIO05	EP	Primary entry point to this module. Receives control from caller and calls OPENPROC, CLOSEPRC, and PUTGET subroutines and returns to caller.
	OPENPROC	PR	Determines the type of open required and calls the following subordinate routine: BLDBLK, OPNNEW, OPNPASS, OPNTAB, or OPNLAB.
	OPENR	PR	Opens a data set, VTOC, VTOC header, or staging volume for REPAIRV processing.
	PUTGET	PR	Calls BLDCCWG, BLDCCWP1, BLDCCWP2, or ISSUEXCP and frees CCW storage.
	CLOSEPRC	PR	Determines the type of close required and calls the following subordinate routines: CLOSESTD, CLOSENEW, and FREEBLK.
	OPNPASS	PR	Issues USTAE macro to get up or remove ESTAE environment, issues OPEN, TYPE=J, and calls CHECKOPN.
	OPNTAB	PR	Issues OPEN macro and calls CHECKOPN.
	OPNLAB	PR	Issues Open, TYPE=J, MODESET (supervisor state), calls CHECKOPN, and issues MODESET (problem-program state).
	OPNNEW	PR	Issues SVC 82.
	BLDBLK	PR	Issues UGSPACE, initializes IOXCTLBK, and calls READJFCB.
	BLDCCWP1	PR	Builds CCWs to write "COUNT KEY DATA."
	BLDCCWP2	PR	Builds CCWs to write "KEY DATA."
	BLDCCWG1	PR	Builds CCWs to read "KEY DATA."
	ISSUEXCP	PR	Issues EXCP and WAIT macros and test for errors.
	ADJCCW	PR	Modifiers a CCW string to begin reading at a different location on the track.
	FWRITE	PR	Writes the count, key, and data portions of multiple record on a track beginning at a specified record.
	OPNVTH	PR	Opens a defective VTOC header or a VSAM data set using a pseudo open (SVC82).
	READCNT	PR	Reads the count field of a specified record.
	READKD	PR	Reads the key and data fields of a specified record.
	SETCCW	PR	Builds a CCW string to read the count fields on a track.
	SETKDCCW	PR	Builds a CCW string to read the count field, the key field (if it exists), and the data field of all records on a track.
	SPACCR	PR	Bypass the read count CCW for a specified record.
	WRITEREC	PR	Rewrites the key field and data field of a specified record on a track.
	CLOSESTD	PR	Issues CLOSE and UPRINT macro to print error messages.
	CLOSENEW	PR	Issues SVC 82.
	CHECKOPN	PR	Determines whether OPEN ABEND exit was entered or DCBCFLAGS indicates DCB not opened and issues UPRINT macro to print error messages.
	SYNAD	PR	Issues SYNADAF and SYNADRLS macros and issues UPRINT macro to print error messages.
	OCABEND	EP	Sets IOXABEND flag in IOXCTLBK. Entered by OPEN/CLOSE DCB exit.

CSECT/Load Module Name	Label	Use	Descriptions
IDCIO05 (continued)	READJFCB	PR	Issues RDJFCB macro and necessary error message.
	RETRY	EP	Reestablish addressability, issues invalid password message, and returns to label AFTEROPN with a return code of 4. Entered by systems during ABEND.
IDCLC01			LISTCAT FSR; produces a listing of all or part of a VSAM catalog. This module initializes and manages the routing of VSAM catalog entries.
	IDCLC01	EP	Only entry point to this module.
	ENTPROC	PR	Manages the request for specific entries or generic lists of entries from the catalog. If Access Method Services is invoked interactively with TSO, ENTPROC makes a generic list of entries with the TSO user's identification at the beginning of each entry.
	GNXTPROC	PR	Manages the request for all or a specified subset of the catalog entry types in alphameric sequence.
	INITPROC	PR	Interrogates the FDT and initializes the catalog and DADSM parameter lists and workareas.
	RTEPROC	PR	Routes control to the appropriate formatting procedure. Then routes control for formatting the associated data sets in a cluster, alternate index, pagespace or GDG grouping.
	DATEPROC	PR	Calculates the creation and expiration LISTCAT options to be used to test the catalog entry's creation and expiration dates.
	TIMEPROC	PR	Converts the time of day, either from a UTIME macro or from the catalog volume time stamp, to a packed-decimal format.
	CKDTPROC	PR	Tests the creation and expiration LISTCAT options against the catalog entry's creation and expiration dates.
IDCLC02			This module locates, formats, and lists the VSAM catalog entries.
	IDCLC02	EP	This entry point is used to establish addressability, acquire automatic storage and initialize the common data area pointers.
	ALSPROC	PR	Gets the aliases chained off the entry by control interval number. Calls LISTPROC to list the aliases.
	ANLTPROC	PR	Gets the associated entry names using the control interval number and calls LISTPROC to list the entry names.
	ANSVPROC	EP	Gets the list of associated entry types and control interval numbers from the VSAM catalog management workarea.
	AUPROC	EP	Repetitively builds the Text Processor Dynamic Data Argument List for formatting and listing the VSAM catalog fields for non-VSAM data sets, user catalogs, generation data groups, and aliases. Repeatdly calls LISTPROC to print the data.
	CDIPROC	EP	Formats the VSAM catalog data for alternate index, cluster, pagespace, data, index and path entries. Builds the Text Processor argument list and calls LISTPROC to print the data.
	ERRPROC	EP	Completes the Dynamic Data Argument List with either an Access Method Services or catalog return code, when required. Issues the UPRINT

CSECT/Load Module Name	Label	Use	Descriptions
			macro to list the informational or error messages. Issues UERROR macro to list VSAM catalog (SVC26) error messages. Zeros out the Dynamic Data Argument List when finished.
	FPLPROC	EP	Re-initializes the string of CTGFLs prior to each catalog locate request by using the original CTGFL.
	FREESTG	EP	Issues a UEPIL umacro to free the automatic storage acquired by IDCLC02.
	LISTPROC	EP	Issues the UPRINT macro and zeros out the Dynamic Data Area Argument List when finished.
	LOCPROC	EP	Issues VSAM catalog locate request and obtains additional catalog work space if required. After the first successful locate, sets the catalog ACB information in the CTGPL and establishes the LISTCAT subtitle with the catalog name.
	VPROC	EP	Repetitively builds the Text Processor Dynamic Data Argument List for formatting and listing the VSAM catalog fields for a volume record entry. Repeatedly calls LISTPROC to print the data.
	SHORTLST	EP	Produces an abbreviated list of catalog entry names from the UCIR work area for the TSO user who uses the LISTCAT default parameters.
	VOLLIST	EP	Produces an abbreviated list of catalog entry names and volume serial numbers from the UCIR work area for the TSO user who uses LISTCAT with the volume parameter.
IDCLR01	AATOPLR	EP	Only entry point to this module—Top control segment.
	ADDASOC	PR	Add an association to association table.
	BUFSHUF	PR	Moves record from last general buffer to "home" buffer requested, for this record type.
	BLDVEXT	PR	Builds the vertical extension table.
	CATOPEN	PR	Opens the catalog data set and ENQs on it.
	CKEYRNG	PR	Checks the data object for key range. If yes prints high key.
	CLEANUP	PR	Closes the catalog, DEQs from it, and prints condition codes.
	CLENCRA	PR	Closes the CRA and frees storage associated.
	CRAOPEN	PR	Opens the CRA and calls the procedure to build the CTT.
	CTTBLD	PR	Reads CRA control record, gets storage for CTT, scans CRA and builds CTT. Controls SEQUENTIAL DUMP.
	DOOTHR	PR	Goes through SORTTBL forward chain containing nonVSAM names and calls PRTOTHR to print the objects.
	DOVSAM	PR	Goes through SORTTBL forward chain for VSAM names and calls PRTVSAM to print them.
	ERROR	PR	Using entry subscript for error table, prints the error message, continues or aborts according to last condition code.
	GETPRT	PR	Gets copy of CRA record, calls IDCRC04 to obtain fields requested, and, if COMPARE, gets the catalog record.
	INITLZE	PR	Initializes switches, adapter parameter list, IDCRC04 parameter list, opens the alternate output file, and gets table space.
	INTASOC	PR	Initializes an association table for a base object.

CSECT/Load Module Name	Label	Use	Descriptions
IDCLR01 (continued)	INTSORT	PR	Gets storage for sort table, builds the entries in it from the CTT for the object type specified.
	INTVEXT	PR	Initializes VEXTTBL by calling IDCRC04 requesting extension pointers and places them in the table.
	MEMSORT	PR	Adds forward and backward pointers in sort table.
	PRTAAXV	PR	Prints associated AIXs volumes.
	PRTCMP	PR	Prints and/or compares information in CRA for one entry.
	PRTDMP	PR	Prints unformatted CRA record. If compare, calls PRTDMPC to print corresponding catalog information and underscore miscompares.
	PRTDMPC	PR	Prints unformatted catalog record corresponding to CRA record being printed. The miscompares are underscored.
	PRTFIFO	PR	Print CRA without sorting using the same procedures as if sorting.
	PRTMCWD	PR	Prints miscompare message indicating most severe fields in error.
	PRTOJAL	PR	Print alias(s) associated with an object.
	PRTOJVL	PR	Print volumes and high keys associated with an object.
	PRTOTHR	PR	Print and/or compare all nonVSAM objects and their extensions.
	PRTTIME	PR	Print time stamps of volumes after converting them to MM/DD/YY HH/MM/SS.
	PRTVOL	PR	Print and/or compare volume record and its extensions.
	PRTVSAM	PR	Print and/or compare VSAM structures and associated records.
	SUMIT	PR	Sum or print number of objects processed.
	TCICTCR	PR	Translate control interval from catalog to CRA.
	VERTEXT	PR	Initializes VERTEXT, loops through the extensions and prints them.
IDCLR02		EP	Formats the buffer pool and reads CRA and catalog records.
IDCMP01			IMPORT FSR; re-constructs a VSAM cluster or VSAM user catalog from a portable data set created by IDCXP01. If the portable data set is a cluster or alternate index, IMPORT FSR issues a UCATLG macro to add the necessary entries to the VSAM catalog, and issues a UCOPY macro to copy the portable data set to the VSAM cluster. If the portable data set is a VSAM catalog, IMPORT FSR connects the VSAM catalog by issuing a UCATLG macro.
	IDCMP01	EP	Only entry point to this module.
	ALTRPROC	PR	Constructs a CTGPL and CTGFV for the catalog alter interface.
	BFPLPROC	PR	Constructs a CTGFL from dictionary and workarea information.
	BPASPROC	PR	Constructs PASSWALL CTGFL and moves information into PASSWALL.
	CLUSPROC	PR	Reads catalog and data records from the portable data set. Uses catalog information plus information from the command to perform a catalog define for the cluster. Copies data into the cluster.
	CNCTPROC	PR	Connects one or more user catalogs.
	CPLPROC	PR	Constructs a CTGPL to be used for a catalog define, alter, delete, or locate.

CSECT/Load Module Name	Label	Use	Descriptions
	CTLGPROC	PR	Invokes VSAM catalog management with a CTGPL.
	DALCPROC	PR	Allocates and deallocates volumes.
	DUPNPROC	PR	This procedure is called when a duplicate entry name is found in the catalog when trying to define the cluster being imported. DUPNPROC performs a locate to see if the catalog entry has the temporary export flag on. If the temporary export flag is on, DUPNPROC deletes the existing cluster so the imported cluster can be defined.
	FVTPROC	PR	Constructs CTGFVs and CTGFLs from information in the dictionary.
	GETPROC	PR	Gets a data record and moves it into a buffer. Reconstructs the original record if it has been segmented.
	IUNIQPRC	PR	Checks the DSATTR field in the CTGFV to see if the cluster being defined is a unique cluster. If so, IUNIQPRC supplies a null space (volume) CTGFV to define the cluster.
	LVLRPROC	PR	Constructs a CTGFL for DEVTYPE lists and a list of volume serial numbers.
	MSGPROC	PR	Issues a UPRINT to print messages.
	MVDAPROC	PR	Moves data from one storage location to another.
	OPENPROC	PR	Opens the VSAM cluster for input or opens the portable data set for output.
	RANGPROC	PR	Processes all information about key ranges.
	RECPROC	PR	Copies the data from the portable data set to the VSAM cluster being imported. RECPROC issues a UCOPY macro to copy the portable data set to the cluster, and issues a UCLOSE macro to close the cluster.
IDCPM01			PARM FSR; establishes or changes the processor parameters. Processor parameters (TEST, MARGINS, and GRAPHICS) can be established through the PARM field of the EXEC card. TEST appears in the area addressed in GDTDBH. MARGINS appears as the first two halfwords in the area addressed in GDTRIH. GRAPHICS is recorded in the PCT.
	IDCPM01	EP	Only entry point to this module.
	GRPHPARM	PR	Establishes the translate table specified in the GRAPHICS option.
	MARGPARM	PR	Processes the MARGINS parameter. The left and right margin values are placed into the Reader/Interpreter Historical Data Area to be used by the Reader/Interpreter when processing the next command.
	TESTPARM	PR	Resets the previous TEST option, if necessary. Processes new TEST parameters. Obtains and initializes the Test Option Data Area.
	TESTSAVE	PR	Moves TEST parameters from the FDT to the Test Option Data Area to be used by the Access Method Services dump routine.

CSECT/Load Module Name	Label	Use	Descriptions
IDCPR01			PRINT FSR; prints the contents of a data set in EBCDIC, hexadecimal, or dump format. PRINT FSR establishes page layout by issuing a UESTA macro, and prints lines of data by issuing a UPRINT macro.
	IDCPR01	EP	Only entry point to this module.
	DELIMSET	PR	Establishes the boundaries for printing a subset of the input data set.
	TEXTPSET	PR	Communicates the page layout and record layout for the listing to the Text Processor.
IDCRC01		EP	This is the highest level of control and the only entrypoint to this module. The function loops through the CRAs opening them, writes them and their associated objects to the portability data set and closes them.
	BUILDCRV	PR	Obtains space for CRV, ACC, and VTT, obtains volume and device type information on CRAs, and constructs the name chain for all entries in the CRAs.
	BUILDNAM	PR	Builds the name chain extension block of storage.
	CHKCATNM	PR	Reads a CRA record and checks the owning catalog, then issues an ENQ on the owning catalog.
	CKNAMES	PR	Gathers passwords for VSAM data sets, collects the association CI numbers and determines the largest logical record length.
	COMPNAME	PR	Compresses the blanks from the right of the object name and places it in the space obtained in the procedure SUBSP.
	DIRECT	PR	Gets space and reads in the directory.
	DUPNAMCK	PR	Scans the name chain for duplicate names and prints message if one is found.
	ERRCK	PR	If an error is considered severe, the catalog is closed and the error message is printed.
	EXPORTDR	PR	Prints start of export of CRA message, calls IDCRC02 to export and prints completion message.
	EXTRACT	PR	Sets up the FMPL and calls IDCRC04 to extract data fields from CRA records.
	INIT	PR	Calls SUBSP to obtain storage and then initializes the buffer pool.
	MESSAGE	PR	Handles the printing of all messages.
	NAMETABL	PR	Checks the name on the CRA record and if it is a cluster, AIX, nonVSAM or catalog connector, it builds the name into the name chain.
	OBJVOLCK	PR	Checks the time stamp and CI on the volumes with that of the CRA for each object.
	OPEN	PR	Builds the OPNAGL and issues the open for the CRA. It then checks the owning catalog name for the major owning catalog.
	OPENCRA	PR	Calls procedures to open the CRA, get its time stamp, build the name table and the directory entry.
	SCANCRA	PR	Reads the catalog record, gets storage for CTT and loops all CRA records putting CI numbers in the CTT and calls NAMETABL to build the name table.
	SUBSP	PR	Handles the obtaining and allocation of small pieces of storage associated with the name table from one large block.

CSECT/Load Module Name	Label	Use	Descriptions
	SYNCH	PR	Checks the entire name chain for entries specified in the input. It also checks for valid associations, CIs, and volumes.
	TERM	PR	Dequeues from owning catalog, closes the portability data set, and releases storage.
	TIMESTMP	PR	Reads the volume time stamp using UIOINFO and places it in the volume timestamp table.
IDCRC02			Creates a portable data set of VSAM clusters, catalog information for nonVSAM, and associated aliases for both.
	IDCRC02	EP	Only entry point to this module.
	ALSPROC	PR	Obtains catalog information for alias associations of nonVSAM data sets.
	ASOCPROC	PR	Obtains catalog information for generation data sets associated with generation data groups.
	CLUSPROC	PR	Obtains catalog information and data for VSAM clusters.
	CONTROL	PR	Builds control records containing catalog information.
	CTLGPROC	PR	Invokes catalog management with a CTGPL for Locate.
	LOCPROC	PR	Builds a CTGPL and multiple CTGFLs for catalog locates.
	MVDAPROC	PR	Moves data in storage from one location to another and clears work area storage.
	NVSMPROC	PR	Gets catalog information for nonVSAM data sets.
	OPENPROC	PR	Opens the VSAM cluster for input and the portable data set for output.
	PRNTPROC	PR	Prints messages for association errors.
	PUTPROC	PR	Writes a control record containing catalog information to the portable data set.
	RECPROC	PR	Copies the data for a VSAM cluster to the portable data set.
	SAVEPROC	PR	Saves control records containing catalog information until processing for that object's catalog information is complete and then writes all records to the portable data set.
IDCRC03		EP	Handles format of buffer pool and reading of catalog or CRA records.
IDCRC04		EP	This is the only entry point to this module.
	PCKLC	PR	Insures the requested catalog field exists in a group occurrence being processed.
	PEXPT	PR	Sets up address and length of extension pointers as per argument passed.
	PGREC	PR	Obtains addressability to the desired CI block.
	PGREP	PR	Finds highest non-deleted RELREPNO with desired group code.
	PGVAL	PR	Find the field and extract the requested data.
	PLNRV	PR	Locate non-replicated values.
	PLOCZ	PR	Locate field and dictionary information.
	PLVAL	PR	Locate fixed or variable length field in physical record and group occurrence.
	PSCNC	PR	Loops through all FMFLs to convert names to internal notation.
	PSCNF	PR	Moves requested data to area specified by caller.
	PSHIN	PR	Inserts the data found into requested field.
	PTCMP	PR	Compares sub-fields between input data and "found" data.
	PTRNS	PR	Format and build compressed name table, insure group codes if special name obtained from caller.

CSECT/Load Module Name	Label	Use	Descriptions
IDCRC01 (continued)	PTSTS	PR	Tests for existence of field and if there, places dictionary information into work area.
IDCRIKT			Modal command verb and keyword table used by the Reader/Interpreter.
IDCRILT			Command Name Table for Command Descriptors used by the Reader/Interpreter.
IDCRI01			Reader/Interpreter used when Access Method Services is invoked with a batched job. Its functions are: 1. On first entry only, loads a table of Command Descriptor load module names and a table of modal command verbs; initializes the Reader/Interpreter Historical Data Area; and obtains PARM options if options exist in the PARM field of the JCL statement. 2. Scans the command for a verb. 3. Handles modal commands (IF, ELSE, DO, END, and SET) to determine which command to process next. 4. Having found a functional command verb, invokes IDCRI02 to find and load the appropriate Command Descriptor module and to initialize the FDT. 5. Scans parameters by using the Command Descriptor to check syntax and semantics and to build the FDT. 6. Invokes IDCRI03 for clean-up activity following each function command. Returns to IDCEX01 if the functional command is to be executed—that is, if it contains no errors detected by the Reader/Interpreter.
	IDCRI01	EP	Only entry point to this module.
	BUILDFDT	PR	Places constants into FDT and converts the constants, if necessary.
	BYPASTRM	PR	Prepares to obtain next verb name.
	CONVERT	PR	Converts EBCDIC to binary, decimal, or hexadecimal.
	DEFAULTS	PR	Selects possible defaults for parameters.
	DSIDCHK	PR	Checks data set name item for adherence to naming conventions.
	DSPLCALC	PR	Calculates offset into an array of pointers or counts.
	ERROR1	PR	Processes an error whose message doesn't change.
	ERROR2	PR	Processes an error that requires variable data to be inserted into the message.
	ERRSETUP	PR	Makes special preparations to print semantic error message.
	GETDATA	PR	Prepares to get constant or list of constants from the command.
	GETNEXT	PR	Gets the next functional command verb name and a pointer to its parameters. Interprets modal commands.
	GETQUOTD	PR	Gets a constant without enclosing apostrophes from the command.
	GETRECRD	PR	Reads the next input record and prints it.
	GETSIMPL	PR	Gets an unquoted constant from the command.
	GETSPACE	PR	Allocates space for the FDT.

CSECT/Load Module Name	Label	Use	Descriptions
	INREPEAT	PR	Repetition of a subparameter list has ended; prepares for another subparameter list repetition.
	KWDPARM	PR	Processes a keyword parameter after finding it in the Command Descriptor.
	MODALIF	PR	Processes IF modal command.
	MODALSET	PR	Processes SET modal command.
	MODLELSE	PR	Processes ELSE modal command.
	MORSPACE	PR	Allocates additional space for a list of constants in the FDT.
	NAMESCAN	PR	Checks naming restrictions.
	NEEDNOTS	PR	Checks parameters to ensure that certain semantic requirements have not been violated. Checks for mutually exclusive parameters, and required parameters.
	NEXTCHAR	PR	Gets the next character from the command.
	NXTFIELD	PR	Gets the next field from the command.
	PACKCVB	PR	Converts EBCDIC string to fullword binary number.
	POSPARM	PR	Processes a positional parameter.
	RINIT	PR	Initializes the Reader/Interpreter.
	SETDFLT	PR	Adds default parameter to the FDT.
	SCANCMD	PR	Controls scanning commands and building the FDT.
	SCANENDS	PR	Finds left and right margins of the command in the input record just read.
	SCANSEP	PR	Scans past the next syntactic separator (comma, blanks, and/or comments).
	SETFLAG	PR	Indicates that a particular parameter was found in the command or was implied by defaults.
	SKIPCMD	PR	Bypasses remainder of current command.
IDCRI02			Searches the table of Command-Descriptor modules for the name of the module that corresponds to the current command, and then loads that module. Initializes the FDT.
	IDCRI02	EP	Only entry point to this module.
IDCRI03			Reader/Interpreter functional command termination. Frees working space and deletes unneeded modules.
	IDCRI03	EP	Only entry point to this module.
IDCRI04			Reader/Interpreter used when Access Method Services is invoked interactively with TSO. Its functions are: 1. Loads the Command Descriptor for the command. 2. Invokes the TSO parse service routine to syntax check the parameters. 3. Converts the output from the TSO parse service routine and builds the FDT.
	IDCRI04	EP	Only entry point to this module.
	ADDPARM	PR	Asks TSO terminal user to supply required parameters not originally supplied.
	BUILDFDT	PR	Checks for errors in data and puts data in the FDT.
	DEFAULTS	PR	Chooses parameters that can have defaults.
	FAILSPAC	PR	Handles the "no available storage" error.
	FINDPDE	PR	Finds the Parameter Descriptor Entry (PDE) for a parameter.
	GETSPACE	PR	Obtains storage for the FDT.

CSECT/Load Module Name	Label	Use	Descriptions
IDCRI04 (continued)	MAINSCAN	PR	Directs processing of non-repeated parameters.
	MSGKWD	PR	Obtains the keyword most closely associated with a parameter. The keyword is used in an error message.
	NEEDPRMS	PR	Checks for required parameters that are missing.
	NEWPARM	PR	Asks for correct data from a TSO terminal user.
	NOTPARMS	PR	Checks for conflicting parameters.
	REPLIST	PR	Saves information about repeated subparameters and initializes FDT secondary arrays.
	RESOLVE	PR	Asks TSO terminal user if the Reader/Interpreter can ignore some parameters.
	RISETUP	PR	Obtains a Command Descriptor load module and initializes the FDT and other tables.
	RITERM	PR	Releases storage and terminates the Reader/Interpreter.
	SETDFLT	PR	Puts defaults in FDT.
	SUBSCAN	PR	Directs processing of repeated parameters.
	TOOMANY	PR	Asks TSO terminal user if the Reader/Interpreter may ignore excessive data.
	TRNSLATE	PR	Translates output from TSO parse routine into the FDT.
IDCRM01		EP	Only entry point to this module.
	ALISPROC	PR	Reads data records and defines alias entries.
	ALTRPROC	PR	Constructs the CPL and FVT to be used to alter the names of the objects.
	BFPLPROC	PR	Constructs the skeleton FPL or constructs the FPL from the dictionary and work area information passed by EXPORTRA on the portable volume.
	BPASPROC	PR	Constructs passwall FPL.
	CLUSPROC	PR	Reads catalog and data records from the portability volume and defines the object copy.
	CPLPROC	PR	Constructs the catalog parameter list to be used for UCATLG operations.
	CTLGPROC	PR	Invokes VSAM catalog management to perform operation indicated in CPL.
	DALCPROC	PR	Dynamically allocates and unallocates volumes.
	DELTPROC	PR	Performs all delete operations using catalog management.
	FVTPROC	PR	Constructs FVT and FPLs from information in dictionary passed as an argument.
	GETPROC	PR	Gets a data record via UGET, reconstructs it and places it in the buffer.
	GDGPROC	PR	Defines GDG base entries.
	IUNIQPRC	PR	Checks to see if data set being defined is a unique data set. Builds a null volume FVT if required.
	LVLRPROC	PR	Constructs the volume list and DEVTYPE FPL.
	MVDAPROC	PR	Moves data from one location in storage to another as specified by input arguments.
	NFVTPROC	PR	Constructs the FVT and FPLs for nonVSAM objects.
	NVSMPROC	PR	Reads catalog and data records from the portability data set and performs the define of nonVSAM entries.
	OPENPROC	PR	Performs all opens of VSAM objects for output or the portability data set for input.
	RANGPROC	PR	Processes key range information building the RANGES list.
	RECPROC	PR	Copy data from portability data set to VSAM cluster.

CSECT/Load Module Name	Label	Use	Descriptions
	UCATPROC	PR	Reads catalog and data records from portable volume and performs a define of user catalog pointers.
IDCRP01			REPRO FSR; copy a SAM, ISAM, or VSAM data set to a SAM or VSAM data set; unload or reload catalogs. Data set types are determined at open time, when IDCIOOP is called (UOPEN macro). When records are skipped at the beginning, a series of UGETs is issued until the required record is reached. When records are skipped at the end, a series of UGETs and PUTs is issued. When the copy is to the end of the data set, then a single call is made to IDCIOCP (UCOPY macro), which copies the data set from the first record to be copied through the end of the data set. The UPOSIT macro is employed to position to a FROMKEY or FROMADDRESS starting point.
	IDCRP01	EP	Only entry point to this module.
	DELIMSET	PR	Establishes the boundaries for copying a subset of the input data set.
	CATRELOD	PR	Checks for sufficient space, matching names for target and backup catalogs, and for agreement with volume serial number and device types.
	SORSREAD	PR	Reads a record from the backup catalog during a catalog reload.
	TARGREAD	PR	Reads a record from the target catalog during a catalog reload.
	GETPAIR	PR	Reads a record from both the backup and target catalogs for the initial checking performed before a catalog reload begins.
	DUMPIT	PR	Activated by the PARM test function in order to trace all I/O for catalog reload.
	TRUENAME	PR	Maps the RBA boundaries of the backup truename ranges.
	CATRANS	PR	Locate and translate control interval numbers from source catalog to target catalog.
	CNVRTCI	PR	Converts control interval numbers from source catalog values to target catalog values.
	RECOVCAT	PR	Checks catalog recoverable attribute during copy catalog.
IDCRS01 **(VS2.03.808)**			RESETCAT FSR; synchronize a catalog with the CRA(s) of its owned volumes.
	IDCRS01	EP	Only entry point to this module.
	AERROR	PR	Exit if not enough storage is available to establish automatic storage for RESETCAT modules.
	CATINIT	PR	Initialize RESETCAT's description of the catalog.
	CLEANUP	PR	Ensure all resources are freed.
	COPYCAT	PR	Copy the catalog to the workfile.
	INIT	PR	Perform the main initializations of RESETCAT.
	MERGECRA	PR	Merge and reset CRA into the workfile.
	PROCCRA	PR	Process the records of the current CRA.
	REASSIGN	PR	Perform control interval reassignment.
	UPDCAT	PR	Update the catalog from the workfile.
	UPDCRA	PR	Update the CRAs from the workfile.
	WRAPUP	PR	Handle clean-up operations after successful RESETCAT processing.

CSECT/Load Module Name	Label	Use	Descriptions
IDCRS02 (VS2.03.808)			Performs various checking functions.
	ASSOC	PR	Does association checking.
	CINALTER	PR	Alter control interval numbers in catalog records.
	LOCDIT	PR	Locates a specific control interval number in a catalog record.
	PROCCI	PR	Ensure that a control interval number is in the list of control interval numbers for records being processed.
	PROCTYPE	PR	Scan a catalog record for control interval numbers.
	SCANCI	PR	Scan record for control intervals.
	SETCI	PR	Update the workfile to reflect new control interval numbers for reassigned CINs.
	VERA	PR	Verify aliases for nonVSAM and GDG associations.
	VERC	PR	Verify associations for clusters.
	VERDSDIR	PR	Verify initial space claims.
	VERCI	PR	Verify associations on a set of records.
	VERG	PR	Verify associations for alternate indexes.
	VERR	PR	Verify associations for PATHs.
	VERU	PR	Verify associations for user catalogs.
	VERX	PR	Verify the alias chain.
IDCRS03 (VS2.03.808)			Contains procedures for controlling space.
	CATRCDSU	PR	Establish base record offsets for catalog low key range records.
	CHKBITS	PR	Compare bits in the bit map.
	CHKDSDIR	PR	Check a data set directory entry against a data or index component.
	CHKUNQ	PR	Check extents for unique data spaces.
	GETFIT	PR	Get a free entry in tables for ASSOC procedure.
	GETNEXTE	PR	Translate an index into a table into a virtual address.
	GETTAB	PR	Get and initialize a table for ASSOC procedure.
	MARKUNUS	PR	Mark a volume group occurrence (VGO) unusable.
	PROCVOL	PR	Resolve space conflicts.
	SETBMAP	PR	Check space conflicts for data or index type catalog entries.
	VERB	PR	Verify associations for GDG base records.
	VLNRESET	PR	Verify space requested from objects being reset against non-reset volumes.
	VLRESET	PR	Verify space requested from objects being reset against reset volumes.
	VOLCHK	PR	Volume consistency routine.
IDCRS04 (VS2.03.808)			Performs field management processing.
	DELGO	PR	Delete a group occurrence.
	FIND	PR	Locate requested information from a set of catalog records.
	MODGO	PR	Modify a group occurrence.
IDCRS05 (VS2.03.808)			Association processing.
	ADDTN	PR	Add a true name to the catalog.
	ADDUPCR	PR	Prepare for update CRA processing.
	BLDRLST	PR	Add an entry to the reset volume table.
	BLDVLST	PR	Add an entry to the volume serial table.
	CKERR	PR	Print an error message.

CSECT/Load Module Name	Label	Use	Descriptions
	CRAUPCHN	PR	Add a workfile record to a specific "update CRA" chain.
	DELTN	PR	Delete a true name from the catalog.
	ENTNMCK	PR	Determine if a catalog record has a valid entry name.
	GENNAME	PR	Generate a true name.
	GETVIA	PR	Get a record by control interval number via a specific CRA.
	SCNRLST	PR	Obtain the next CRA volser entry.
	SCNVLST	PR	Scan the list of volumes.
IDCRS06 **(VS2.03.808)**			Handles I/O functions; defines and deletes the workfile.
	CRAMNT	PR	Request CRA allocation.
	CRADMNT	PR	Request CRA deallocation.
	DSCLOSE	PR	Close a VSAM data set.
	DSOPEN	PR	Open a VSAM data set.
	RECMGMT	PR	Perform I/O requests.
	WFDEF	PR	Define the workfile for RESETCAT processing.
	WFDEL	PR	Delete the workfile.
IDCRS07 **(VS2.03.808)**			This module contains system dependent code designed specifically for RESETCAT functions.
	CATEOV	PR	Extend the catalog.
	CNVTCCHH	PR	Convert CCHH to TTnn.
	ENSURECI	PR	Ensure that there are enough control intervals for reassignment.
	EOVPANCI	PR	Format catalog free records until the catalog is extended.
	EOVPCCCR	PR	Update and write the CCR.
	EOVPCHAC	PR	Get the high allocated control interval numbers for the Low Key Range (LKR) and High Key Range (HKR) of the catalog.
	EOVPRBAP	PR	Build a table of high RBA field pointers for record management control blocks.
	EOVPRCCR	PR	Read the catalog control record (CCR) and update the high allocated control intervals in the record management control blocks.
	EOVPWFLR	PR	Write a deleted free record to the catalog.
	EOVPXIO	PR	Perform I/O for the catalog.
	RENAMEP	PR	Rename duplicate true name entries.
	UPDCCR	PR	Update the catalog control recd (CCR).
IDCSA01			Entry and exit module for the Access Method Services processor. Interfaces between the operating system and the processor. Creates the GDT and calls IDCEX01.
	IDCSATO	EP	Entry point to this module if Access Method Services is invoked interactively with TSO.
	IDCSA01	EP	Entry point to this module if Access Method Services is invoked with a batched job.
	GETCORE	PR	Issues GETMAIN to allocate storage.
IDCSA02			Supplies all operating system services to the processor, except prologue and epilogue. At each of the following entry points, IDCSA02 converts the service request to the appropriate system macros.
	IDCSAAL	EP	Allocates a data set or mounts a volume (UALLOC macro).
	IDCSACA	EP	Issues the VSAM CATLG macro (UCATLG macro).

CSECT/Load Module Name	Label	Use	Descriptions
IDCSA02 (continued)	IDCSACL	EP	Loads an executable module and branches to it (UCALL macro).
	IDCSACR	EP	Obtains a fully-qualified data set name from a VSAM catalog (UCIR) macro.
	IDCSADE	EP	Returns to caller (UDELETE macro).
	IDCSADL	EP	Deallocates a data set or dismounts a volume (UDEALLOC macro).
	IDCSAFP	EP	Frees pool, releases pooled storage (UFPOOL macro).
	IDCSAFS	EP	Frees space, releases pooled or non-pooled storage (UFSPACE macro).
	IDCSAGP	EP	Gets pool, a request for pooled storage (UGPOOL macro).
	IDCSAGS	EP	Gets space, a request for non-pooled storage (UGSPACE macro).
	IDCSAID	EP	Obtains a terinal user's identification (UID macro).
	IDCSALD	EP	Loads a module but does not branch to it (ULOAD macro).
	IDCSALK	EP	Loads and gives control to a non-Access Method Service module (ULINK macro).
	IDCSAPT	EP	Obtains information from a TSO terminal user (UPROMPT macro).
	IDCSAQL	EP	Obtains a fully-qualified data set name if Access Method Services is invoked interactively with TSO (UQUAL macro).
	IDCSASN	EP	Provides a dump (USNAP macro).
	IDCSATI	EP	Gets date and time of day by calling IDCSA05 (UTIME macro).
	COREINIT	PR	Initializes an area of storage to binary zeros or blanks.
	DYNERR	PR	Prints a dynamic allocation error message.
	FINDNAME **(VS2.03.807)**	PR	Finds or loads module requested and returns entry point.
	NOSUPP	PR	Prints an error message if a function is not supported if Access Method Services is invoked with a batched job.
	RELECORE **(VS2.03.807)**	PR	Searches Load List Block and deletes modules not in use.
	IDCSAS2	DE	Amount of storage IDCSA02 needs. Used by IDCSA01.
IDCSA03			Prologue and epilogue for all routines. This module is called at entry to and exit from all other modules.
	IDCSAEP	EP	Epilogue entry point, releases storage (UEPIL macro).
	IDCSAPR	EP	Prologue entry point, acquires storage (PROL macro).
	GETCORE	PR	Gets requested amount of storage.
	IDCSAS3	DE	Amount of storage IDCSA03 needs. Used by IDCSA01.
IDCSA05			Gets date and time of day (invoked by IDCSA02).
	IDCSA05	EP	Only entry point to this module.

CSECT/Load Module Name	Label	Use	Descriptions
IDCSA06			Performs the following functions; MOUNT, which performs necessary setup so the MSC can mount a virtual volume. DEMOUNT, which performs necessary setup so the MSC can demount a virtual volume. POST, which issues SVC 82 to update or clear the UCB with the volume serial number and TTR to the VTOC. CHECK, which checks to determine whether the UCB is a 3330V or 3330-1 or is allocated for exclusive use.
	IDCSA06	EP	Only entry point to this module.
	MDSETUP	PR	Setup routine for a MOUNT or DEMOUNT function. Issues ESTAE macro through the IDCSA10 module.
	MOUNTCTL	PR	Controlling Mount routine. Determines whether the volume is mounted or must be mounted. Demounts volume if necessary to mount specified volume.
	MOUNTVOL	PR	Second mount controlling routine. Performs setup to call subordinate routines to mount the 3330V volume.
	DEMNTVOL	PR	Controlling demount routine. Performs setup to call subordinate routines to demount the 3330V volume.
	UCBPOST	PR	Builds the required parameter list and issues SVC 82 to post or clear the UCB with volume label and TTR to the VTOC.
	UCBCHECK	PR	Checks a specified UCB to determine whether it is 3330 direct-access, virtual, or real device. Also checks for exclusive control.
	ISSUEMNT	PR	Issues the USSC macro to mount a volume and if successful issues USSC again to acquire cylinder 0 of the volume.
	ISSUEMNT	PR	Issues the USSC macro to demount a volume.
	ENQDEQ	PR	Issues the ENQ or DEQ macro for calls from MOUNTVOL or DEMNTVOL, respectively.
	SSCMSG	PR	Prints all USSC macro error messages.
IDCSA07			This module recatalogs a nonVSAM data set.
	IDCSARC	EP	Only entry point to this module. Calls GETENT.
	GETENT	PR	Initializes the CAMLST parameter list and issues the LOCATE macro pointing to the CAMLST to locate a catalog entry.
	TESTENT	PR	Tests whether there is a need to recatalog the data set and whether recataloging is supported.
	UPDATENT	PR	Updates the catalog entry with a new device type and/or volume serial number.
IDCSA08			Supplies system-related services to the processor. At each of the following entry points, IDCSA08 converts a request to the appropriate system macro.
	IDCSADQ	EP	Releases a resource (UDEQ macro).
	IDCSANQ	EP	Acquires control of a resource (UENQ macro).
	IDCSARK (SU 5752-824)	EP	Acquires RACK protection (URACHECK macro).
	IDCSARV	EP	Acquires control of an I/O unit (URESRVE macro).
	IDCSASC	EP	Deletes direct-access data sets (USCRATCH macro).
	IDCSASI	EP	Returns system control-block information (USYSINFO macro).

CSECT/Load Module Name	Label	Use	Descriptions
IDCSA08 (continued)	IDCSAWO	EP	Writes messages to console operator (UWTO macro).
IDCSA09			Issues an MSSC macro, and optionally waits on a response. The caller supplies a keyword indicating the macro to the issued, a pointer to the MSSC request block, and a word in which the MSSC reason code can be returned.
	IDCSASS	EP	Only entry point to this module. Directs program execution. Obtains space for an ECB and message if required, frees the area if an error occurred or the caller did not supply a pointer variable to return the ECB message area address.
	CHECKARG	PR	Determines the macro to be issued. Checks that the argument list is valid for that macro.
	ISSUEMAC	PR	Issues the execute form of the requested MSSC macro. Saves the return code and reason code.
	CHECKCODE	PR	Waits on an ECB, if necessary, for the MSSC macro to complete. Returns error codes to caller. If MSC function was successful but an error occurred updating the mass storage volume inventory, prints a message.
IDCSA10			Performs ABEND recovery. When called, this module establishes an ESTAE environment of cancels one. When called by Recovery Termination Management (RTM) performs either cleanup before the ABEND or attempts a retry.
	IDCSA10	EP	Controlling routine. Receives control from the caller through the USTAE macro. Determines the type of request and invokes the proper routine.
	SETSTAE	PR	Issues ESTAE macro to establish an ESTAE environment, builds STAEPARM, puts STAEPARM address into SAHIST and prints error messages.
	CANSTAE	PR	Issues ESTAE macro to cancel the ESTAE environment, removes the STAEPARM address from SAHIST.
	STAEEXIT	PR	Sets up addressability, determines whether it can retry, sets up retry, or calls recovery to cleanup before the ABEND.
	RECOVERY	PR	Performs cleanup before ABEND. Items cleaned up are: 1. A volume incompletely mounted or demounted. 2. A UCB cleared of volser and of the TTR of the VTOC.
IDCTP01			Text Processor: formats output. This module includes all conversion routines and controls the printing of each line.
	IDCTPPR	EP	Prints one or more lines (UPRINT macro).
	BDCONV	PR	Converts binary data to unpacked decimal characters.
	BHCONV	PR	Converts binary data to hexadecimal characters with or without enclosing apostrophes.
	BHDCONV	PR	Converts binary data to hexadecimal-dump format.
	BLOCK	PR	Prepares to print a block of data.
	CONVERT	PR	Converts data and puts it in a line.

CSECT/Load Module Name	Label	Use	Descriptions
	EBCDIC	PR	Moves EBCDIC characters to a line.
	ERROR	PR	Processes error condition.
	IDCTPS1	DE	Amount of storage IDCTP01 needs. Used by IDCSA01.
	INSERT	PR	Inserts data into pre-defined format, or uses static text when inserted data is missing and default data is needed.
	LINEPRT	PR	Controls title lines, headings, spacing, and translates data lines.
	LINERET	PR	Returns formatted lines to the caller.
	PUPCONV	PR	Converts packed-decimal data to unpacked-decimal characters.
	REDO	PR	Initiates data replication.
	SEGMENT **(VS2.03.807)**	PR	Segments messages into 72-character lines.
	STACKPUT	PR	Stacks data lines. Does a UPUT on the line when the stack is full, a message is to be printed, or the print data set is changed.
	SPACE	PR	Spaces a line.
	STATIC	PR	Prepares static text for a line.
IDCTP04			Initializes and modifies PCT; sets up all page controls, defines headings and footings, and defines format of page. Each of the following entry points represents a service provided by the Text Processor.
	IDCTPEA	EP	Establishes a PCT with data from storage (UESTA macro).
	IDCTPES	EP	Establishes a PCT with data from a static text module (UESTS macro).
	IDCTPRE	EP	Re-initializes Text Processor for the next request (URESET macro).
	IDCTPRS	EP	Modifies an existing PCT (UREST macro).
	ESTACONT	PR	Gets space for PCT and initializes it with data from storage (UESTA macro).
	ESTSCONT	PR	Gets space for PCT and initializes it with data from a static text module (UESTS macro).
	INITPCT	PR	Gets and initializes PCT.
	PCTSETUP	PR	Verifies and initializes the PCT.
	P04SETUP	PR	Prepares a work table for PCT initialization.
	RESETCON	PR	Re-initializes Text Processor for next request, returns page number, and clears the PCT.
	RESTCONT	PR	Initializes working table for modifiying existing PCT (UREST macro).
	STACKFL	PR	Prints lines in stack buffer.
IDCTP05			Reads Text Structures for either IDCTP01 or IDCTP04.
	IDCTP05	EP	Only entry point to this module.
IDCTP06			Converts catalog error messages to verbal text.
	IDCTPER	EP	Only entry point to this module.
	CATERCNV	PR	Translates numeric code and sets up multilevel message.
	DAERCNV **(VS2.03.807)**	PR	Invokes DAIRFAIL routine and sets up multi-line messages for dynamic errors.
IDCTSAL0			Text Structure for ALTER messages.
IDCTSBI0			Text Structure for BLDINDEX message.
IDCTSCC0			Text Structure for CNVTCAT messages.
IDCTSCK0			Text Structure for CHKLIST messages

CSECT/Load Module Name	Label	Use	Descriptions
IDCTSDE0			Text Structure for DEFINE messages.
IDCTSDL0			Text Structure for DELETE messages.
IDCTSEX0			Text Structure for Executive messages.
IDCTSIO0			Text Structure for I/O Adapter messages.
IDCTSLC0			Text Structure for LISTCAT listing.
IDCTSLC1			Text Structure for LISTCAT messages.
IDCTSLR0			Text Structure for LISTCRA listing.
IDCTSLR1			Text Structure for LISTRA messages.
IDCTSMP0			Text Structure for IMPORT messages.
IDCTSPR0			Text Structure for PRINT listings.
IDCTSRC0			Text Structure for EXPORTRA messages.
IDCTSRI0			Text Structure for batch Reader/Interpreter messages from IDCRI01, IDCRI02, and IDCRI03.
IDCTSRI1			Text Structure for Reader/Interpreter messages from IDCRI04.
IDCTSRS0 **(VS2.03.808)**			Text Structure for RESETCAT messages.
IDCTSSAO **(SU 5752-824)**			Text Structure for System Adapter messages.
IDCTSTP0			Text Structure for Text Processor print chain definitions.
IDCTSTP1			Text Structure for Text Processor messages.
IDCTSTP6			Text Structure for text processor UERROR messages.
IDCTSUV0			Text Structure for common messages (universal messages) form any module.
IDCTSXP0			Text Structure for EXPORT messages.
IDCVS01			Performs the following functions: Security check, Scratch VTOC, Recatalog nonVSAM data sets on the VTOC, Retrieve information from the VTOC, and Update Retrieve information from the VTOC, and Update information on the VTOC.
	IDCVS01	EP	Only entry point to this module. Controlling routine, receives control from UCALL macro. Determines type of service and calls the routine.
	SECCHECK	PR	Calls procedures to open the VTOC. Checks the Format-4 DSCB and Format-1 DSCB for security.
	VTOCOPEN	PR	Opens the VTOC for BSAM update processing and reserves the unit if requested.
	FMT4CHK	PR	Checks the Format-4 DSCB for security. Calls procedures to read Format-4 DSCB and to verify the VSAM catalog password.
	FMT1CHK	PR	Checks the Format-1 DSCB for security. Calls procedures to read Format-1 DSCB and open data sets.
	FMT4READ	PR	Reads and tests for a Format-4 DSCB.
	FMT1READ	PR	Reads and tests for a Format-1 DSCB.
	VSAMCHK	PR	Opens a VSAM data set.
	NVSAMCHK	PR	Opens a nonVSAM data set.

CSECT/Load Module Name	Label	Use	Descriptions
	EXPIRCHK	PR	Opens an unexpired data set.
	CATCHK	PR	Determines whether the volume is owned by a VSAM Catalog.
	CATOPEN	PR	Opens a VSAM catalog and verifies passwords.
	SCRATCH	PR	Scratches the VTOC. Calls procedures to open the VTOC, clear the Format-4 DSCB, and scratch the Format-1 DSCBs.
	FMT4SCR	PR	Clears the VSAM ownership flag in the Format-4 DSCB. Calls a procedure to read the Format-4 DSCB.
	FMT1SCR	PR	Scratches all Format-1 DSCBs. Calls a procedure to read the Format-1 DSCB.
	VTOCGET	PR	Retrieves information from the VTOC. Calls procedures to open the VTOC and read the Format-4 DSCB.
	VTOCPUT	PR	Updates information in the VTOC. Calls procedures to open the CVTOC and read the Format-4 DSCB.
	RECATALOG	PR	Recatalogs all nonVSAM data sets. Calls procedures to open the VTOC and read Format-1 DSCBs.
IDCVS02			Performs the I/O processing necessary to initialize a new 3330V volume with IPL records, volume-label records, and a variable-size empty VTOC. Also retrieves from and places information into the volume-label record (cylinder 0, track 0, record 3).
	IDCVS02	EP	Only entry point to this module. Controlling routine; receives control from the UCALL macro. Determines the type of service and calls the proper rouine.
	INITVOLM	PR	Issues the UEXCP macros of OPEN and PUT to initialize new 3330V volumes and calls the subordinate routines FMTTRK) and FMTTRK1.
	FMTTRKO	PR	Prepares the three records that are written to cylinder 0, track 0.
	FMTTRK1	PR	This routine prepares the 39 records (DSCBs) that are written to cylinder 0, track 1.
	GETLAB	PR	Opens the VTOC, reads the label, returns the information requested, and closes the VTOC.
	PUTLAB	PR	Opens the VTOC, reads the label, updates the label, returns the information requested, and closes the VTOC.
ICDVS03			Performs the search of a volume VTOC for eligible data sets, based on the user's selection criteria passed in the VS3AGL parameter list.
	IDCVS03	EP	Only entry point to this module. Controlling routine; receives control via the UCALL macro. It determines which type of service is being requested and invokes the proper routine.
	INITIAL	PR	Controlling routine for initialization. Calls procedures to open the VTOC and get storage.
	SELECTOR	PR	Searches the data set names in the volume's VTOC and selects those which meet the caller's criteria.
	CLEANUP	PR	Closes the volume VTOC and frees storage.
	CRITERIA	PR	Inserts certain information into the data set element array for data sets which meet the caller's criteria.

CSECT/Load Module Name	Label	Use	Descriptions
IDCVY01			VERIFY FSR; checks a VSAM data set against its catalog entries and corrects any discrepancies by calling IDCIOVR (UVERIFY macro).
	IDCVY01	EP	Only entry point to this module.
	OPENPROC	PR	Opens the VSAM data set.
	TERMPROC	PR	Closes the VSAM data set.
IDCXP01			EXPORT FSR; creates a portable copy of a VSAM cluster, alternate index or catalog. If the input data set is a VSAM cluster or alternate index, EXPORT FSR copies the cluster. If the input data set is a VSAM catalog, EXPORT FSR disconnects the catalog.
	IDCXP01	EP	Only entry point to this module.
	ALTRPROC	PR	Constructs the CTGPL and CTGFV for a catalog alter so that the data set attributes catalog field (DSATTR) can be modified.
	CLUSPROC	PR	Gets catalog information and data for a cluster or alternate index. Processes the disposition options. If PERMANENT is specified in the command, the cluster is deleted. If TEMPORARY is specified in the command, the temporary export flag is turned on by issuing a catalog alter.
	CONTRBL	PR	Writes catalog information to the portable data set.
	CTLGPROC	PR	Invokes VSAM catalog management with a CTGPL.
	DELTPROC	PR	Constructs a CTGPL to delete a cluster or alternate index or to disconnect a user catalog. Calls catalog to delete a cluster or alternate index.
	DSCTPROC	PR	Disconnects a user catalog.
	LOCPROC	PR	Builds a CTGPL and multiple CTGFLs for catalog locate. CTGFLs are used to locate catalog information to be exported.
	MORESP	PR	Obtains a larger catalog workarea.
	MVDAPROC	PR	Moves data into buffers.
	OPENPROC	PR	Opens a VSAM cluster for input or opens the portable data set for output.
	PUTPROC	PR	Writes a record with catalog information to the portable data set.
	RECPROC	PR	Copies the data from the VSAM cluster to the portable data set, record by record.

DATA AREAS

The data areas in this chapter are described in four columns, which are interpreted as follows:

Offset: The numeric address of the field relative to the beginning of the area. The first number is the offset in decimal, followed (in parentheses) by the hexadecimal equivalent.

Bytes and Bit Pattern: The size (number of bytes) of the field and its alignment relative to the fullword boundary. A *v* indicates variable length.

Examples:

4	A four-byte field beginning on a word boundary.
. . 3	A three-byte field beginning on a halfword boundary and running into the next word.

This column also shows the bit patterns of a byte when they are significant (as in a flag byte). When the column is used to show the state of the bits (0 or 1) in a flag byte, it is shown as follows:

....	The eight bit positions (0-7) in a byte. For ease of scanning, the high-order (leftmost) four bits are separated from the low-order four bits.
x...	A reference to bit 0.
1...	Bit 0 is on.
0...	Bit 0 is off.
.... ..xx	A reference to bits 6 and 7.

Bit settings that are significant are shown and described. Bit settings that are not shown are considered to be reserved and set to zero.

Field Name: A name that identifies the field and appears in the assembly listings. A sub-field or value name is indented from the field's name. An * indicates the field is not named.

Description: Content, Meaning, Use: A description of the use of the field.

ALLAGL
Allocation Argument List

The Allocation Argument List is passed whenever a UALLOC or UDEALLOC macro is issued. It contains information needed to dynamically allocate or deallocate data sets or to mount or dismount volumes.

Created by	Modified by	Used by	Size
All routines	IDCSA02	IDCSA02	31

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	ALLDSN	For data set allocation or deallocation this contains the address of a 44 byte data set name, left-justified and padded with blanks. This field is not used for volume mounting or dismounting.
4 (4)	8	ALLDDN	After successful data set allocation or volume mounting, this field contains the DDname for the data set or volume. The DDname is left-justified and padded with blanks. If several volumes were mounted, the DDname is the first of several concatenated DD statements For data set deallocation and volume dismounting and, if ALLDSN is zero, this field contains the DDname describing the data set or volume. The DDname is left-justified and padded with blanks. It may be the first of concatenated DD statements.
12 (C)	4	ALLULP	For volume mounting this field contains the address of a table.
The table contains:			
	2	*	Number of entries in the following array.
Each entry consists of:			
	2	*	Number of non-blank characters in the following field.
	8	*	Unit name or device type left-justified and padded with blanks.
16 (10)	4	ALLVLP	For volume mounting this field contains the address of a table.
The table contains:			
	2	*	Number of entries in the following array:
Each entry consists of:			
	2	*	Number of non-blank characters in the following field.
	6	*	Volume serial number left justified and padded with blanks. Volume serials in this array are matched one to one with the units in the array in ALLULP. The last unit is matched with all the unmatched volume serials.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			This field is used only for volume mounting.
20 (14)	4	ALLPWD	For allocation of password protected data sets, this field contains the address of a 8 byte password, left-justified and padded with blanks. If ALLPWD contains zero, no password is supplied.
24 (18)	1	ALLSTS	For data set allocation this field means the following data set status:
	.... 1...	ALLSTSSR	Status of SHR.
	.... .1..	ALLSTSNW	Status of NEW.
	.... ..1.	ALLSTSMD	Status of MOD.
	1	ALLSTSOD	Status of OLD.
25 (19)	. 1	ALLDSP	For data set allocation this field indicates the current data set disposition. For data set deallocation this field indicates a disposition that overrides any other disposition.
	.... 1...	ALLDSPKP	Disposition of KEEP.
	.... .1..	ALLDSPDE	Disposition of DELETE.
	.... ..1.	ALLDSPCG	Disposition of CATLG.
	1	ALLDSPUN	Disposition of UNCATLG.
	0		No overriding disposition.
26 (1A)	. . 2	ALLORG	For data set allocation the UALLOC puts the data set organization in this field:
	. . 1		
	1...	ALLORGIS	Indexed sequential.
	.1..	ALLORGPS	Physical sequential.
	..1.	ALLORGDA	Direct access.
	...1	ALLORGCX	Telecommunications line group.
	.... 1...	ALLORGCQ	Direct-access message queue.
	.... .1..	ALLORGMQ	Data control block for message transfer.
	.... ..1.	ALLORGPO	Partitioned organization.
	1	ALLORGUN	Unmoveable data set.
27 (1B)	. . . 1	ALLORG (cont.)	
	1...	ALLORGGS	Graphic data control block.
	.... 1...	ALLORGVS	VSAM.
	0		Dynamic allocation cannot determine the data set organization.
28 (1C)	1	ALLOPT	Indicates the type of request either data set allocation or volume mounting, and allocation option for private mounting and deallocate option to unallocate.
	1...	ALLOPTVL	Request for volume mounting.
	.1..	ALLOPTDS	Request for data set allocation.
	..1.	ALLOPTPV	Request for private volume.
	...1	ALLOPTUN	Request for unallocate option during deallocation.
29 (1D)	.1	ALLVLCNT	Volume count.
30 (1E)	. . 1	ALLUNCNT	Unit count.

BUFS
Buffer Pool Control Block

The Buffer Pool Control Block is used by EXPORTRA to control I/O buffers. It is passed from IDCRC01 through field management (IDCRC04) to IDCRC03.

Created by	Modified by	Used by	Size
IDCRC01	IDCRC03	IDCRC03	28

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	4	BUFPOOL	Address of first buffer.
4(4)	4	BUFPL	Address of chain of buffers.
8 (8)	4	BUFIOCS	Address of the IOCSTR.
12 (C)	4	BUFGDT	Address of the GDT.
16 (10)	4	BUFCTT	Address of the CTT.
20 (14)	4	BUFWKARA	Address of the work area.
24 (18)	2	BUFSIZE	Size of buffer pool.
26 (1A)	.2	BUFSWS	Indicator Flags.
	1...	BUFORMAT	1=Buffer pool formatted
			0=Buffer pool not formatted.
	.xxx xxxx	*	Reserved.
	xxxx xxxx	*	Reserved.

CIRAGL
Catalog Interface Argument List

The Catalog Interface Argument List is required each time a UCIR macro is issued. The CIRAGL contains information about the qualifiers for which data set names are returned.

Created by	Modified by	Used by	Size
All routines	IDCSA02	IDCSA02	32

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	CIRTYP	Type of UCIR request:
	1...		"One" qualifier function. This function returns data set names whose qualifiers match the qualifiers provided in CIRAGL and have only one qualifier between the header and trailer qualifiers supplied in CIRAGL.
	.1..		"All" qualifier function. This function returns all data set names whose qualifiers match the qualifiers provided in CIRAGL.
1 (1)	3	*	Reserved.
4 (4)	2	CIRHLN	Contains the length of the header qualifiers.
6 (6)	2	CIRTLN	Contains the length of the trailer qualifiers. If no trailer qualifiers are supplied, this field must contain zero.
8 (8)	4	CIRHDR	Address of the header qualifier. The header qualifier must end with a period.
12 (C)	4	CIRTLR	Address of trailer qualifier. The trailer qualifier must start with a period.
16 (10)	4	CIRWKP	Address of a fullword where the UCIR macro puts the address of the workarea containing the qualified data set names.
20 (14)	4	CIRCAT	If data set names are to be found only in one catalog, this field contains the address of a 44 byte catalog name. If data set names are to be found in more than one catalog, this field contains zero.
24 (18)	4	CIRPWD	If the catalog addressed in CIRCAT is password protected, this field may contain the address of a 8 byte password. If no password is supplied for a password protected catalog, the operator will be prompted for it.
28 (1C)	4	CIRPID	Contains a 4 byte UGPOOL identifier that is used to obtain storage for the returned workarea. The caller must free this workarea.

CKAGL
Check UCB Argument List

The CKAGL is passed when a UMSSUNIT is issued for a CHECK request. It defines a request to interrogate a specified UCB for information.

Created by	Modified by	Used by	Size
Calling routine	IDCSA06	IDCSA06	29

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	CKHEAD	Set with 'CHECK' by IDCSA06. Set with 'CHECKERR' if an invalid option is set in CKFLAGS.
8(8)	4	CKUCBPTR	Address of an area containing the UCB address or zeros if the UCB address is unknown. If the UCB address is zeros, on return the UCB address will be put in the area.
12(C)	8	CKDDNAME	DDNAME of the volume.
20(14)	4	CKDATYPE	Address of the four-byte area to be posted with the characters 'VIRT' or 'REAL' if the field CKRETTYP is set. The area is set only when the UCB is a 3330.
24(18)	4	CKLABELP	Address of the six-byte volume serial number if CKDYALOC is set.
28(1C)	1	CKFLAGS	Flags indicating the type of test.
	1...	CKTESTVT	Indicates that the request is for a 3330 virtual UCB. This field and CKRETTYP are mutually exclusive.
	.1..	CKRETTYP	Indicates that the test is for a 3330 Model I (real) or 3330V (virtual) UCB and to return the type to the area addressed by CKDATYPE.
	..1.	CKDMTABL	Indicates that the test is for a real or virtual 3330 UCB that is demountable, using the CKDDNAME to locate the UCB pointer.

Command Descriptor

There is a Command Descriptor for each verb supported by this processor. The Command Descriptor is a load module that contains directions for parsing the command, performing semantic checking, and building an FDT from the commands. The name of the load module for each verb is found in a directory, which is itself a load module named IDCRILT.

The name of each load module and the corresponding verb, as supplied by IBM, is as follows:

VSAM		Checkpoint/Restart	
IDCCDAL	ALTER	IDCCDCK	CHKLIST
IDCCDBI	BLDINDEX		
IDCCDCC	CNVTCAT		
IDCCDDE	DEFINE		
IDCCDDL	DELETE		
IDCCDLC	LISTCAT		
IDCCDLR	LISTCRA		
IDCCDMP	IMPORT		
IDCCDPM	PARM		
IDCCDPR	PRINT		
IDCCDRC	EXPORTRA		
IDCCDRM	IMPORTRA		
IDCCDRP	REPRO		
IDCCDRS	RESETCAT (**VS2.03.808**)		
IDCCDVY	VERIFY		
IDCCDXP	EXPORT		

Each Command Descriptor consists of several variable-length data areas. The data areas are divided into two groups. The first group is used to build the Function Data Table, FDT. If Access Method Services is invoked with a batched job, IDCRI01 also uses the first group to parse commands. The second group is used to parse a command if Access Method Services is invoked interactively with TSO.

The first group is described in the following table and consists of two main data areas. The first data area, Verb Data Area, names the FSR load module to use for this command. Appendages to the Verb Data Area define positional parameters, default parameters, groups of needed parameters, and groups of incompatible parameters. The second data area, Parameter Data Area, describes parameters the command may have. If several parameters have the same syntax attributes, one Parameter Data Area can describe the parameters. Appendages to each Parameter Data Area define constants and data for the parameter(s). The Command Descriptor assigns an identification number, ID, to each parameter. the command may have. Both Reader/Interpreters use the ID numbers to reference the Parameter Data Area that describes a parameter.

The second group of data areas in a Command Descriptor consists of Parameter Control Lists, PCL. The PCLs are arranged in the Command Descriptor as follows: First is the PCL that describes all the non-repeated parameters. Second is one PCL for each set of repeated parameters. Third is one PCL for each set of parameters that may be prompted for. IDCRI04 passes the PCLs to TSO so TSO can parse the command. TSO returns a Parameter Descriptor List, PDL. Each parsed parameter is described by its own section of the PDL called the Parameter Descriptor Entry, PDE. For a description of the PCL, PDL, and PDE refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor.*

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCRI01 IDCRI04	Variable

Verb Data Area

A Command Descriptor always begins with the Verb Data Area. This data area names the FSR for this command, gives the total number of parameters and provides offsets to other data areas in the Command Descriptor.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	DESCID	Descriptor identification, contains the last four letters of the Command Descriptor module name. For example, 'CDAL' for the Alter Command Descriptor, IDCCDAL.
4 (4)	2	PCLDSPL1	Number of bytes from the beginning of the Verb Data Area to the first Parameter Control List.
6 (6)	. . 2	VDATALEN	Number of halfwords in Verb Data Area (used to compute the address of the first Parameter Data Area).
6 (6)	2	PARMCNT	Number of Parameter Data Areas in this Command Descriptor.
10 (A)	. . 2	MAXID	Largest parameter ID number that is used in this Command Descriptor.
12 (C)	8	LOADNAME	Load module name of FSR that processes this command.
20 (14)	1	POSDSPL	Number of halfwords from the beginning of the Verb Data Area to Positional Parameter appendage of the Verb Data Area.
21 (15)	. 1	DGRPDSPL	Number of halfwords from the beginning of the Verb Data Area to Default Parameter appendage of the Verb Data Area.
22 (16)	. . 1	VNGRPDSP	Number of halfwords from the beginning of the Verb Data Area to Needed Parameters appendage of the Verb Data Area.
23 (17)	. . . 1	NTGRPDSP	Number of halfwords from the beginning of the Verb Data Area to Incompatible Parameters appendage of the Verb Data Area.

Positional Parameter Appendage

This appendage contains the parameter ID number of each positional parameter that is not a subparameter of other parameters. This appendage may follow the Verb Data Area or any Verb Data Area appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	VPOSCNT	Number, *n*, of ID numbers that follow:
2 (2)	2×*n*	VPOSID*n*	List of ID numbers for positional parameters.

Default Parameter Appendage

This appendage contains the parameter ID number of each default parameter. The parameter IDs are grouped into arrays. The first parameter in each array is the default if none of the parameters in that array is supplied in the command. This appendage may follow the Verb Data area or any Verb Data Area appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	DGRPTOT	Number of arrays that follow.
Each array contains:			
	2	DGRPCNT	Number, *n*, of ID numbers that follow:
	2×*n*	DGRPID*n*	List of ID numbers.

Needed Parameters Appendage

This appendage contains the parameter ID number of any necessary parameter that is not a subparameter of another parameter. The parameter IDs are grouped into arrays. At least one of the parameters in each array must be supplied through the command. This appendage may follow the Verb Data Area or any Verb Data Area appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	VNGRPTOT	Number of arrays that follow:
Each array contains:			
	2	VNGRPCNT	Number, *n*, of ID numbers that follow:
	2×*n*	VNGRPID*n*	List of ID numbers.

Incompatible Parameters Appendage

This appendage contains the parameter ID numbers for each parameter in groups of incompatible parameters. The parameter IDs are grouped into arrays. Only one parameter in each array may be supplied through the command.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	NTGRPTOT	Number of arrays that follow:
Each array contains:			
	2	NTGRPCNT	Number, *n*, of ID numbers that follow:
	2×*n*	NTGRPID*n*	List of ID numbers.

Parameter Data Area

The Parameter Data Area follows the Verb Data Area, and descibes the syntax and subparameters of a parameter. Usually there is one Parameter Data Area for each parameter. However, one Parameter Data Area can describe several parameters if the parameters have the same syntax and data.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	PDEFLEN	Number of halfwords in this Parameter Data Area including appendages.
1 (1)	3	OCCURNUM	Number of times this parameter can be repeated in the command.
4 (4)	1	IDDSPL	Number of halfwords from the beginning of this Parameter Data area to the ID Appendage.
5 (5)	1	KWDDSPL	Number of halfwords from the beginning of this Parameter Data area to the Keyword Appendage.
6 (6)	1	NOTDSPL	Number of halfwords from the beginning of this Parameter Data area to the Conflicting Parameters Appendage.
7 (7)	1	NGRPDSPL	Number of halfwords from the beginning of this Parameter Data area to the Necessary Parameters Appendage.
8 (8)	1	PDEDSPL	Number of halfwords from the beginning of this Parameter Data area to the Prompt Appendage.
9 (9)	1	KWDGRPID	Identification number of a TSO keyword needed when this parameter is part of a group of mutually exclusive parameters.
10 (A)	1	*	Reserved.
11 (B)	1	FLAGS	Flags:
	1...	SCLRDATA	Indicates the user supplies data with this parameter.
	.1..	LEVEL1	Indicates this parameter is not a subparameter.
	..1.	REPEATED	Indicates the user may repeat the subparameters of this parameter.
	...1	SCALAR	Indicates the user supplies a single constant with this parameter.
	.... 1...	LIST	Indicates the user may supply several "like" constants with this parameter.
	.... .1..	DEFAULT	Indicates this parameter has a default value.
	.... ..1.	SUBLIST	Indicates this parameter has subparameters.
	x	*	Reserved.

No Constant Appendage

This appendage follows the above section if the parameter has subparameters. In other words, if SUBLIST=1, this appendage immediately follows the FLAGS field described above.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
12 (C)	2	PCLDSPL2	Number of bytes from the beginning of this Command Descriptor to a PCL describing this parameter's subparameters.
14 (E)	1	SUBDSPL	Number of halfwords from the beginning of this Parameter Data Area to the Subparameter Appendage.
15 (F)	1	REPMAX	Maximum times this parameter's subparameters may be repeated in the command.

Constant Appendage

This appendage follows the basic Parameter Data area if the parameter has constants. In other words, if SCLRDATA=1 this appendage immediately follows the FLAGS field described above.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
12 (C)	4	HIVALUE	The greatest value a number constant may have.
16 (10)	4	LOWVALUE	The least value a number constant may have.
20 (14)	1	MAXLNGTH	The maximum length of the constant after any conversion.
21 (15)	1	LISTMAX	Maximum number of times this constant may be repeated in a list of subparameters.
22 (16)	1	*	Reserved.
23 (17)	1	CFLAG	Flags:
	1...	NUMBER	Indicates the constant is a number.
	.1..	ANYSTRNG	Indicates the constant is a character string.
	..1.	DSNAM	Indicates the constant is a data set name.
	...1	GENERIC	Indicates the constant is a generic data set name.
	.... 1...	VOLID	Indicates a volume serial number may replace a data set name.
	.... .1..	USERID	Indicates a prefix of the TSO user's indentification must be added to the data set name if Access Method Services is invoked interactively with TSO.
	.... ..1.	PWORDOPT	Indicates the character string or data set name may be followed by a password.
	x	*	Reserved.

Default Data Appendage

This appendage follows the Constant Appendage if the parameter data has a default constant. In other words, if DEFAULT=1, this appendage immediately follows the CFLAGS field described above.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
24 (18)	1	DEFLTLEN	Length of following field.
25 (19)	*v*	DEFLTVAL	Default constant as it would appear in the command.

ID Appendage

This appendage contains the offset from the beginning of the primary Parameter Data List, PDL, to the Parameter Data Entry, PDE, for each parameter this Parameter Data Area describes. This appendage may follow any other Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	IDCOUNT	Number of sets of two fields that follow. There is a set of fields for each parameter.
Each set contains:			
	2	IDNUM	Parameter ID number.
	2	PDEOFST1	Offset from the beginning of the primary PDL to the PDE for this parameter.

Keyword Appendage

This appendage contains every keyword for each parameter this Parameter Data Area describes. This appendage may follow any other Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	KWDCOUNT	Number of sets of fields that follow. There is a set of two fields for each keyword.
Each set contains:			
0 (0)	1	KWDLEN	Length of the following keyword.
1 (1)	*v*	KWDITEM	Keyword.

Conflicting Parameters Appendage

This appendage contains the parameter ID of each parameter that may not appear with the parameters this Parameter Data Area describes. This appendage may follow any Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	NOTCOUNT	Number *n* of parameter IDs that follow.
2 (2)	2×*n*	NOTID*n*	List of IDs of conflicting parameters.

Necessary Parameters Appendage

This appendage contains the parameter IDs of parameters that must appear with the parameters this Parameter Data Area describes. The parameters are grouped into arrays. One parameter in each array must appear. This appendage may follow any other Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	NGRPTOT	Number of arrays that follow:
Each array contains:			
0 (0)	2	NGRPCNT	Number, *n*, of ID numbers that follow.
	2×*n*	NGRPID*n*	List of parameter ID numbers for necessary parameters.

Prompt Appendage

This appendage contains an offset from the beginning of the prompt PDL to the PDE for prompting information needed by parameters this Parameter Data Area describes. This appendage may follow any other Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PDECNT	Number of sets of fields that follow. There is a set of 3 fields for each PDL offset for each parameter.
Each set contains:			
	2	PDEPRMID	Contains the parameter ID of the parameter in the prompt PDL.
	2	PDEPCLID	Contains the parameter ID of the parameter whose subparameters have been prompted for.
	2	PDEOFST2	Number of bytes from the beginning of the prompt PDL to the PDE for this parameter.

Subparameter Appendage

This appendage contains all the subparameter IDs. This appendage may follow any other Parameter Data appendage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	SUBCOUNT	Number of sets of fields that follow. There is a set of two fields for each subparameter.
Each set contains:			
	2	PARMTYPE	Identifies this subparameter as positional, 'P', or keyword, 'K'.
	2	SUBID	Subparameter ID.

IDCRILT
Command Name Table

IDCRILT contains a table of all verbs and verb abbreviations and their Command Descriptor load module names.

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCRI02 IDCRI04	290

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	LNAMECNT	Number, *n*, of table entries.
2 (2)	16×*n*	TABLE*n*	*n* table entries.
Each entry contains the following:			
	8	TBLVERB	Verb or verb abbreviation.
	8	TBLLNAME	Corresponding Command Descriptor load module name for this verb.

CRA Access Parameter List

The CRA Access Parameter List provides VSAM catalog management with information necessary to access the CRA as a catalog. It is pointed to by the ACB when the UCRA bit in the ACB is on for the OPEN of a CRA by EXPORTRA. The CRA Access Parameter List consists of three control blocks. The ACB points directly to the ACC (Access Method Services/Catalog Communication Table) which in turn points to the CTT (CRA Access Translate Table) and the VTT (CRA Volume Timestamp Table).

Created by	Modified by	Used by	Size
IDCRC01	None	VSAM Catalog Management	Variable

Access Method Services/Catalog Communication Table (ACC) Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	ACCTRANT	Address of the CRA Access Translate Table (CTT).
4 (4)	1	*	Reserved.
5 (5)	.3	ACCDSNCI	Control Interval number used when LOCATEs are performed via true names.
8 (8)	4	ACCVOLTT	Address of the Volume Timestamp Table.

CRA Access Translate Table (CTT) Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	CTTENTNO	Number of entries in the table.
4 (4)	4xn	CTTENTRY	Variable number (n) of 4-byte entries.
	1	CTTENTYP	Type of CRA record.
	.3	CTTCATCI	Catalog control interval number of the CRA control interval for this entry.

CRA Volume Timestamp Table (VTT) Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	VTTENTNO	Number of entries in the table.
4 (4)	14xn	VTTENTRY	Variable number (n) of 14-byte entries.
	6	VTTVOLSR	Volume serial number for the timestamp of this entry.
	8	VTTTMSTP	The timestamp that is in the format 4 DSCB on this volume.

Dump List

The Dump List tells the UDUMP macro which areas to dump. The Dump List consists of entries that describe the individual fields. If one or more fields are to be repeated, they can be described as an array where each group of fields is an element in the array. In such cases, the array is preceded by a Dump List entry called an array header. The array header causes the fields to be repeated. The end of the Dump List is indicated by an entry called the dump list terminator.

Individual entries are printed as *name=data*. Each field in an array is printed as *name(n)=data*. The array name is printed before the array elements. All arrays start on a new line.

Created by	Modified by	Used by	Size
All routines	All routines	IDCDB02	Variable

Individual Field Entry

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	DMPITMNM	Name to be printed with the field. The name is aligned left and padded with blanks.
8 (8)	4	DMPITMPT	Address of field to be dumped.
12 (C)	2	DMPITMLN	Number of bytes to dump. For hexadecimal, bit, or character strings the number is from 1 to 256. For fixed binary, the number is from 1 to 4.
14 (E)	. . 1	DMPITMTP	One character indicating the type of data in field: H – Hexadecimal printed as two characters per byte. B – Bit string printed as eight characters per byte. C – Character printed as one character per byte. F – Fixed binary printed as a signed number for halfwords or fullwords or as an unsigned number for one or three bytes. Leading zeros are suppressed.
15 (F)	. . . 1	*	Reserved.

Array Header Entry

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	DMPARYNM	Name to be printed at the start of the array. The name is aligned left and padded with blanks.
8 (8)	2	DMPARYSZ	Number of bytes in each input element of the array. The number can be from 1 to 32,767.
10 (A)	. . 2	DMPARYIC	Number of following individual items that are in the array. The number can be from 1 to 32,767.
12 (C)	2	DMPARYEX	Number of times to repeat the individual fields. The number can be from 1 to 99.
14 (E)	. . 1	DMPARTYTP	Array header type—contains 'A'.
15 (F)	. . . 1	*	Reserved.

Dump List Terminator Entry

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	*	End of dump list indicator—contains X'FF'.

DARGLIST
Dynamic Data List

The dynamic data argument list describes variable data to be printed. It is always an argument for a print request (UPRINT macro).

Created by	Modified by	Used by	Size
Calling routine	None	IDCTP01	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	DARGDBP	Contains the address of block of data referred to by format list or zero.
4 (4)	4	DARGRETP	Zero if printing is to occur; non-zero if no printing is to occur. If non-zero, contains the address of the area in which the formatted print lines are to be returned from the Text Processor and not printed. Spacing control characters aren't returned. The data is truncated to the length (DARGRETL) of the provided area if necessary. Data will be returned to the specified location.
8 (8)	4	DARGSTID	Zero if a format list is also passed as a parameter. If non-zero, contains the Text Structure identification (STID) for static text element to be used as the format list.
Each DARGSTID contains:			
	3	DARGSMOD	Last three characters of the text-structure module name.
	. . . 1	DARGSENT	Static text entry.
12 (C)	2	DARGILP	Length of block whose address is in DARGDBP.
14 (E)	. . 2	DARGCNT	Number of insert and replication elements contained in DARGARY.
16 (10)	2	DARGRETL	Length of the return-data area—that is, DARGRETP.
18 (12)	. . 1	DARGIND	Offset to add to the print column in the format list (FMTOCOL).
19 (13)	. . . 1	*	Reserved.
20 (14)	8×*n*	DARGARY*n*	Group array in one of two formats. The following fields are repeated *n* times, where *n* = DARGCNT.
For insert data each array contains:			
	2	DARGINS	Insert reference number.
	. . 2	DARGINL	Input data length of the field pointed to by DARGDTM.
	4	DARGDTM	Dynamic data pointer, address of field to use for this insert.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
For replication data each array contains:			
	2	DARGREP	Replication reference number.
	2	DARGPCT	Replication count, number of times to replicate a series of format substructures (FMTLIST). This field is not used for replication structures.

ERCNVTAB
Error Conversion Table

The Error Conversion Table is passed whenever a UERROR macro is issued. It contains the information necessary to convert numeric error codes into prose messages.

Created by	Modified by	Used by	Size
All routines	None	IDCTP06	32

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	ERTYPE	Type of error code to be converted.
	1...	ERCATLG	VSAM catalog management error.
	.1..	EROSCAT	OS/VS Catalog error—used in OS/VS1 only.
	..1.	EROYNAL	Dynamic Allocation error **(VS2.03.807)**
1(1)	.1	EROPER	VSAM catalog operation being performed when error occurred. Only one operation type allowed per UERROR invocation.
	1...	ERCATLC	CMS Locate.
	.1..	ERCATDE	CMS Define.
	..1.	ERCATDL	CMS Delete.
	...1	ERCATAL	CMS Alter.
2(2)	..1	EROSOPER	OS/VS Catalog operation being performed—used in VS1 only.
3(3)	...1	*	Reserved.
4(4)	4		Reserved.
8(8)	4		Reserved.
12(C)	4	ERDSNM	Address of data set name or volume serial number associated with the Catalog Management request. The data set name is contained in a 44 byte field padded with blanks; the volume serial number is contained in a 44 byte field padded with binary zeros.
16(10)	4	ERCATRC	Catalog Management return code.
16(10)	4	ERDYNRC	Dynamic allocation return code. **(VS2.03.807)**
20(14)	4	ERCPLPT	Address of Catalog Parameter List (CTGPL) issued that resulted in error condition.
20(14)	4	ERDARBPT	Address of SVC 99 request block issued which resulted in a dynamic allocation error condition. **(VS2.03.807)**
24(18)	4		Reserved.
28(1C)	4		Reserved.

STAEPARM
ESTAE Argument List

The STAEPARM is passed to the ESTAE exit routine in IDCSA10 when an ABEND occurs. It defines whether retry is to be attempted and what recovery must be performed.

Created by	Modified by	Used by	Size
IDCSA10	IDCIO05 IDCSA10 IDCSA06	IDCIO05 IDCSA10	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	STAHEAD	Set with 'STAEPARM' by IDCSA10, when an ESTAE environment is established.
8(8)	4	STAVUCB	Address of the UCB where the mass storage volume is mounted or was demounted. If STAVMNT or STAVDMNT flag is set, this field must be initialized.
12(C)	6	STAVVOL	Volume serial number of the mass storage volume if STAVMNT or STAVENQ is set.
18(12)	..2	*	Reserved.
20(14)	4	STARUCB	Address of the UCB for the real 3330 unit which is cleared if STACLEAR flag is set.
24(18)	6	STARVOL	Volume serial number cleared from the UCB if STACLEAR is set.
30(1E)	..2	*	Reserved.
32(20)	4	STARTTR	TTR of the VTOC cleared from the UCB if STACLEAR is set.
36(24)	4	STARTSAV	Address of the save area for the retry routine's registers.
40(28)	4	STAEXSAV	Address of the registers at the time the ESTAE environment was established. Used by ESTAE routine to restore registers.
44(2C)	4	STARTADD	Address of the retry routine entry point.
48(30)	4	STATCB	Contains the initiator's TCB. Used to dequeue a volume.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
52(34)	1	STARCVY	Recovery flags:
	1...	STAVMNT	Indicates that a mass storage volume is mounted.
	.1..	STAVDMNT	Indicates that a mass storage volume was demounted from a unit, but the UCB has not been marked "NOT READY" yet.
	..1.	STAVENQ	Indicates that an enqueue was done on a volume serial number in STAVVOL. This bit is set off when a dequeue is done.
	...1	STAVPOST	Indicates that the mass storage volume's serial number and VTOC TTR were posted in the UCB following the mount.
	.... 1...	STAVCLEAR	Indicates that the VTOC TTR and volume serial number were removed from a real 3330 UCB, but the volume is still mounted.
53(35)	.1	STARETRY	Retry flags:
	1...	STAIO05	Indicates that the retry routine for IDCIO05 is to be invoked at ABEND.
	.1..	STAOFF	Indicates that the ESTAE environment was cancelled as a result of an ABEND.

EXCLAGL
Exclusive Control Argument List

The EXCLAGL is passed when a UMSSUNIT macro is issued with a select or change request. It defines a request to either (1) select a DD statement and UCB that can be used for the volume if the volume is already allocated exclusively to the job step or (2) change the allocation of a unit and volume to exclusive.

Created by	Modified by	Used by	Size
Calling routine	IDCSA06	IDCSA06	23

Select Request

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	EXCLHEAD	Set by IDCSA06 with 'SELECTXb'.
8 (8)	4	EXCLUCBP	Address of a 4-byte area in which IDCSA06 returns the address of the UCB that can be used to mount the volume.
12 (C)	4	EXCLDDP	Address of an 8-byte area in which IDCSA06 returns the DD name that can be used to open the VTOC on the volume.
16 (10)	6	EXCLVOL	Volume serial number of the volume that is being processed by the caller.
22 (16)	1	EXCLFLAG	Ignored.

Change Request

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	EXCLHEAD	Set by IDCSA06 with 'CHANGEXb'.
8 (8)	4	EXCLUCBP	Address of an area that contains the UCB address, or zeros if the UCB address is unknown. If EXCLUCBP contains zeros, IDCSA06 puts the UCB address in the area upon return.
12 (C)	4	EXCLDDP	Address of the DD name for the volume being processed.
16 (10)	6	EXCLVOL	The volume serial number of the volume whose allocation is to be changed to exclusive, or zeros. If the EXCLMNT bit is set, this field must be specified.
22 (16)	1	EXCLFLAG	Option bits:
	1...	EXCLMNT	Indicates the allocation of the volume is to be changed to exclusive only if the volume is mounted on the unit. When this bit is zero, the allocation of any volume mounted on the unit is changed to exclusive.

EXGARG
EXCP GET Argument List

The EXGARG is passed when UEXCP is issued with a GET request. It defines a request to read a data set.

Created by	Modified by	Used by	Size
Calling routine	IDCIO05	IDCIO05	32

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	EXGHEAD	Set with 'GETbbbbbb' by IDCIO05.
8(0)	4	EXGCTLBK	Address of the IOXCTLBK created during the open.
12(C)	4	EXGCCHH1	Address of an area that contains the physical CCHHR (search) of the first records to be read. This field is required only when the EXOTAB open option is selected.
16(10)	4	EXGCCHH2	Address of an area that contains the CCHHR (seek) in the count field of the first record to be read.
20(14)	4	EXGRECNO	The number of records to be read. The value cannot exceed the number of records on one track.
24(1B)	4	EXGDATAP	Address of the area into which the records are read. Each record is read and stored contiguously.
28(1C)	1	EXGKEYLN	Key length of the records being read.
29(1D)	.1	*	Reserved.
30(1E)	..2	EXGDATAL	Data length of the records being read.

EXOARG
EXCP OPEN Argument List

The EXOARG is passed when a UEXCP is issued with an open request. It defines a request to open a data set.

Created by	Modified by	Used by	Size
Calling routine	IDCIO05	IDCIO05	34

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	EXOHEAD	Set with 'OPENbbbb' by IDCIO05.
8(8)	4	EXODDN	Address of eight-byte DD name.
12(C)	4	EXODSN	Address of 44-byte data-set name.
16(10)	4	EXOVSN	Address of eight-byte volume serial number.
20(14)	4	EXOUCB	Address of the UCB.
24(18)	4	EXOEXT	Address of ten-byte extent return area. Upon return, the area contains the beginning and ending CCHHR of the first extent.
28(1C)	4	EXOCTLBK	Address of four-byte area for return of IOXCTLBK address built by I/O adapter.
32(20)	1	EXOPT	EXCP open options:
	1...	EXOTAB	Open for MSC tables. This option is mutually exclusive of the other open options.
	.1..	EXOLAB	Open for the VTOC to read or write the volume label. This option is mutually exclusive of the other open options.
	..1.	EXOPASS	Open to check for password protected data sets. This option is mutually exclusive of other open options.
	...1	EXONEW	Open to initialize the volume with IPL records, label record, and VTOC. This option is mutually exclusive of the other open options.
	.... 1...	EXOVTOC	Open to read the VTOC DSCBs. This option is mutually exclusive of other open options.
	.... .1..	EXOREP	OPENR is specified: open is being done for a REPAIRV function. This option is mutually exclusive of the other open options.
	xx	*	Reserved.
33(21)	.1	EXFLG	Open flags.
	1...	EXOREAD	Read-only access.
	...1	EXORVS	The user wants to open a VSAM data set for REPAIRV processing. This bit is an option of OPENR.
	.... 1...	EXOSPK	Open is for an entire staging pack. The user wants to be able to read an entire staging pack taken offline from the 3850. This is an option of OPENR.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
	.... .1..	EXORDS	The user wants to open a data set for a repair function. This is an option of OPENR.
	.... ..1.	EXORVT	The user wants to open a data set for a repair function. This is an option of OPENR.
	1	EXOVTH	The user wants to open a VTOC header for a repair function. This is an option of OPENR.
	.xx.	*	Reserved.

EXPARG
EXCP PUT Argument List

The EXPARG is passed when UEXCP is issued with a PUT request. It defines a request to write a data set.

Created by	Modified by	Used by	Size
Calling routine	IDCIO05	IDCIO05	20

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	EXPHEAD	Set with 'PUTbbbbbb' by IDCIO05.
8(8)	4	EXPCTLBK	Address of the IOXCTLBK built during the open.
12(C)	4	EXPDATA	Address of the data-to-write block. EXPDATAB structure is used to initialize this block.
16(10)	4	EXPCCHHR	Address of an area that contains the CCHHR where the data will be written.

EXPDATAB
PUT Data Block

The EXPDATAB is passed when UEXCP is issued with a PUT request. It defines a request to write a data set. The EXPARG points to EXPDATAB.

Created by	Modified by	Used by	Size
Calling routine	IDCIO05	IDCIO05	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	4	EXPRECNO	Number of records to be written.
4(4)	8×*n*	EXPARRAY*n*	Data-to-write array. There is an array entry for each record to be written.
Each entry contains:			
	4	EXPDATAP	Address of the record to be written.
	1	EXKEYLN	Key length of the record being written. Required for each record.
	.1	*	Reserved.
	..2	EXPDATAL	Data length of the record being written. Required for each record.

FMPL
Field Management Parameter List

The Field Management Parameter List is passed whenever module IDCRC04 is called within EXPORTRA and LISTCRA. It contains information and pointers which enable IDCRC04 to extract data from records within the catalog or CRA.

Created by	Modified by	Used by	Size
IDCRC01 IDCLR01	IDCRC04	IDCRC04	Variable

Field Management Parameter List Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMPLFLNO	Number of FMFL pointers.
1 (1)	.3	FMPLBCIN	Control interval number of the base record.
4 (4)	4	FMPLGRTN	Address of the GET routine.
8 (8)	4	FMPLWKAR	Address of the field management work area.
12 (C)	4	FMPLUPTR	Value passed to user GET routine at Input/Output processing time.
16 (10)	1	FMPLRTCD	Return code from a call to IDCRC04.
17 (11)	.1	*	Reserved.
18 (12)	..2	FMPLENTH	Length of the output area provided by caller.
20 (14)	4	FMPLOAR	Address of the output area.
24 (18)	4xn	FMPLFMFL	Array of variable number (n) of 4-byte FMFL pointers.

Field Management Field List (FMFL) Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMFLDLNO	Number of length/data pairs passed by caller.
1 (1)	.1	FMFLTSTC	Compare test condition code.
2 (1)	..1	FMFLGRPC	Field group code supplied by caller.
3 (1)	...1	FMFLINDS	FMFL indicator flags.
	xxxx xxx.	*	Reserved.
	1	FMFLSUCC	Bit indicating success of test. 0=test is successful. 1=test is unsuccessful.
4 (4)	4	FMFLWKAR	Work area for field management.
8 (8)	4	FMFLDNAM	Pointer to 8-byte field name.
12 (C)	4	FMFLTCHN	Address of next test FMFL.
16 (10)	8xn	FMFLDATA	Variable number (n) Length/Data pointer pairs.
	4.	FMFLENTH	4-byte length of supplied data.
	.4	FMFLADDR	4-byte address supplied data.

FMTLIST
Format List

The format list defines the format of printed output. This list consists of several substructures, each identified by its flag byte. Format lists exist in the Text Structures, where they are referenced by STID numbers (Static Text Identifiers). Optionally, they may be passed as an argument of the UPRINT macro, in which case the DARGLIST argument does not furnish a STID.

Created by	Modified by	Used by	Size
Calling routine	None	IDCTP01	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flags:
	1...	FMTEOLF	End of structure.
	.1..	FMTSCF	Space control.
	..1.	FMTIDF	Insert data.
	...1	FMTBDF	Block data.
	.... 1...	FMTREPF	Replication.
	.... .1..	FMTSTF	Static text.
	.... ..1.	FMTDFF	Default data.
	1	FMTHDF	Header line.

Interpretation of each substructure of the format list depends on the value of FMTFLGS. Each of the possible substructures is shown below.

Spacing

The spacing substructure of the format list specifies the line spacing or carriage control to use while printing. The default spacing is used only when a line is not immediately preceded by a spacing substructure. A spacing substructure imbedded in an entry causes the previously formatted data to be printed and signals the start of a new line.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'40' or X'41'.
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTSPF	Space factor: if FMTSPT is equal to 'A', this is the absolute line number to use for printing this line. If FMTSPT is equal to 'R', this is the number of spaces to take before printing. Page overflow results in printing on the first line of the next page.
4 (4)	1	FMTSPT	Spacing type: 'A' signifies absolute line number in FMTSPF, and 'R' signifies relative line number. 'E' signifies page eject.
5 (5)	. 1	*	Reserved.

Insert Data

The insert-data substructure refers to data defined in the dynamic data argument structure, and identified by reference number. This represents variable data to be inserted into the printed line.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'20' or X'A0'. (X'A0' also denotes end-of-structure.)
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTRFNO	Insert reference number: identification number for dynamic data insert that defines the input data to be used for formatting.
4 (4)	2	*	Reserved.
6 (6)	. . 2	FMTOCOL	Print line column for beginning of this field, or if FMTBS is equal to one, the offset from the column indicated by field PCTAPC.
8 (8)	2	FMTOLEN	Output field length. If FMTOLEN is equal to zero or 32,767, then the full, converted input length is used.
10 (A)	. . 1	FMTCNVF	Flags to define conversion and formatting to be done:
	1...	FMTBH	Byte to printable, hexadecimal representation.
	.1..	FMTBHA	Byte to hexadecimal, preceded by X' and followed by a single quote.
	..1.	FMTBHD	STAndard dump format. FMTOCOL and FMTOLEN are ignored.
	...1	FMTBD	Binary to unpacked decimal characters.
	.... 1...	FMTPU	Packed to unpacked decimal characters.
11 (B)	. . . 1	FMTCNVF (cont.)	Conversion flags (continued).
	1...	FMTZS	SUPpress leading zeros by replacing with blanks.
	.1..	FMTAL	Aligned left; the high-order non-zero digit is put in first print column as specified by FMTOCOL.
	..1.	FMTSS	Suppress signs.
	...1	FMTBS	Suppress all trailing blanks but one of the preceding field; add the offset in FMTOCOL to the value in PCTAPC for the print column.
	.... 1...	FMTAR	Align EBCDIC character strings to the right. The print column is added to the print field length to determine the last printable position.

Default Text

The default-text substructure is only used when it immediately follows an insert-data substructure. When examining the insert structure, the value in DARGINS is compared to the value in FMTRFNO. If no match is found, the next format structure is examined to determine whether it is a default structure. If the flag FMTDFF is on in this next structure, the structure is used. In all other cases, it is skipped over.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'02' or X'82'. (X'82' also denotes end-of-structure.)
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTILEN	Length of the default text.
4 (4)	2	FMTIOFF	Offset from the beginning of the format structures to the default text (which follows the format structures).
6 (6)	. 2	FMTOCOL	Print line column, same as for insert substructure.
8 (8)	2	FMTOLEN	Output field length, same as for insert substructure.
10 (A)	. . 2	FMTCNVF	Conversion flags, same as for insert substructure.

Block Format

The block format substructure of the format list defines a block of variable data from which fields are extracted for printing.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'10' or X'90'. (X'90' also denotes end-of-structure.)
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTILEN	Length of the input field. If FMTILEN is zero or if FMTILEN is greater than DARGILP minus FMTIOFF then the input length in DARGILP is used.
4 (4)	2	FMTIOFF	Offset from the beginning of the input data block at which this field begins. The beginning of the data block is in DARGDBP.
6 (6)	. . 2	FMTOCOL	Print line column, same as for insert substructure.
8 (8)	2	FMTOLEN	Output field length, same as for insert substructure.
10 (A)	. . 2	FMTCNVF	Conversion flags, same as for insert substructure.

Replication

The replication substructure defines substructures of the format list that are to be repeated. The replication substructure always precedes the first substructure to be repeated.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'08'. (May not have end-of-list flag on.)
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTRFNO	Reference number to identify the dynamic argument that contains the replication count.
4 (4)	2	FMTRBC	Number of substructures that follow that are to be replicated.
6 (6)	. . 2	FMTRIO	Offset to add to all offsets contained in block-format substructures being replicated to access the input fields.

Static Text

The static text substructure defines data from the Text Structures to be placed in the printed line.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	FMTFLGS	Flag byte: X'04' or X'84'. (X'84' also denotes end-of-structure.)
1 (1)	. 1	*	Reserved.
2 (2)	. . 2	FMTSTL	Length of static text field.
4 (4)	2	FMTSTO	Offset to static text which follows format structures.
6 (6)	. . 2	FMTOCOL	Print line column, same as for insert substructure.
8 (8)	2	FMTOLEN	Output field length, same as for insert substructure.
10 (A)	..2	FMTCNVF	Conversion flags, same as for insert substructure.

FDT
Function Data Table

The Function Data Table is an encoded representation of a command. The Reader/Interpreter constructs the FDT from information found in the command. All defaults are resolved; no conflicts are allowed among the values of an FDT.

The FDT is not one structure, but rather several small structures that are pointed to by a primary vector of addresses, called the FDTTBL. For a parameter that appears in a repeated subparameter list, a secondary vector of addresses results. Figure 13 illustrates this vector and the various small structures to which it points.

The FDT primary vector, FDTTBL, is variable in length. It consists of the command's verb as an 8-byte EBCDIC string, followed by a variable number of fullword pointers. The number of pointers depends on the specific command. There is one pointer per parameter defined in the Command Descriptor. If a pointer is reserved or is not used because the respective parameter has not been specified, the pointer contains zero.

The FDTTBL points to data areas in one of three formats depending upon the input provided by the parameter. If there is more than one data field, an array of data fields is generated. The array is preceeded by a count of the array elements. The count is in a fullword for an array of Number Data Areas and in a halfword for an array of any other data areas. The array consists of one element for each data field supplied as input to the parameter. Every element in the array has the same format—one of the three formats shown below. In the following formats the last 3 characters of the field name are as shown. The first characters may vary and are indicated by *.

Created by	Modified by	Used by	Size
IDCRI01 IDCRI04	None	FSR	Variable

Number Data Area

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	*VAL	Contains the input number in fixed-point binary.

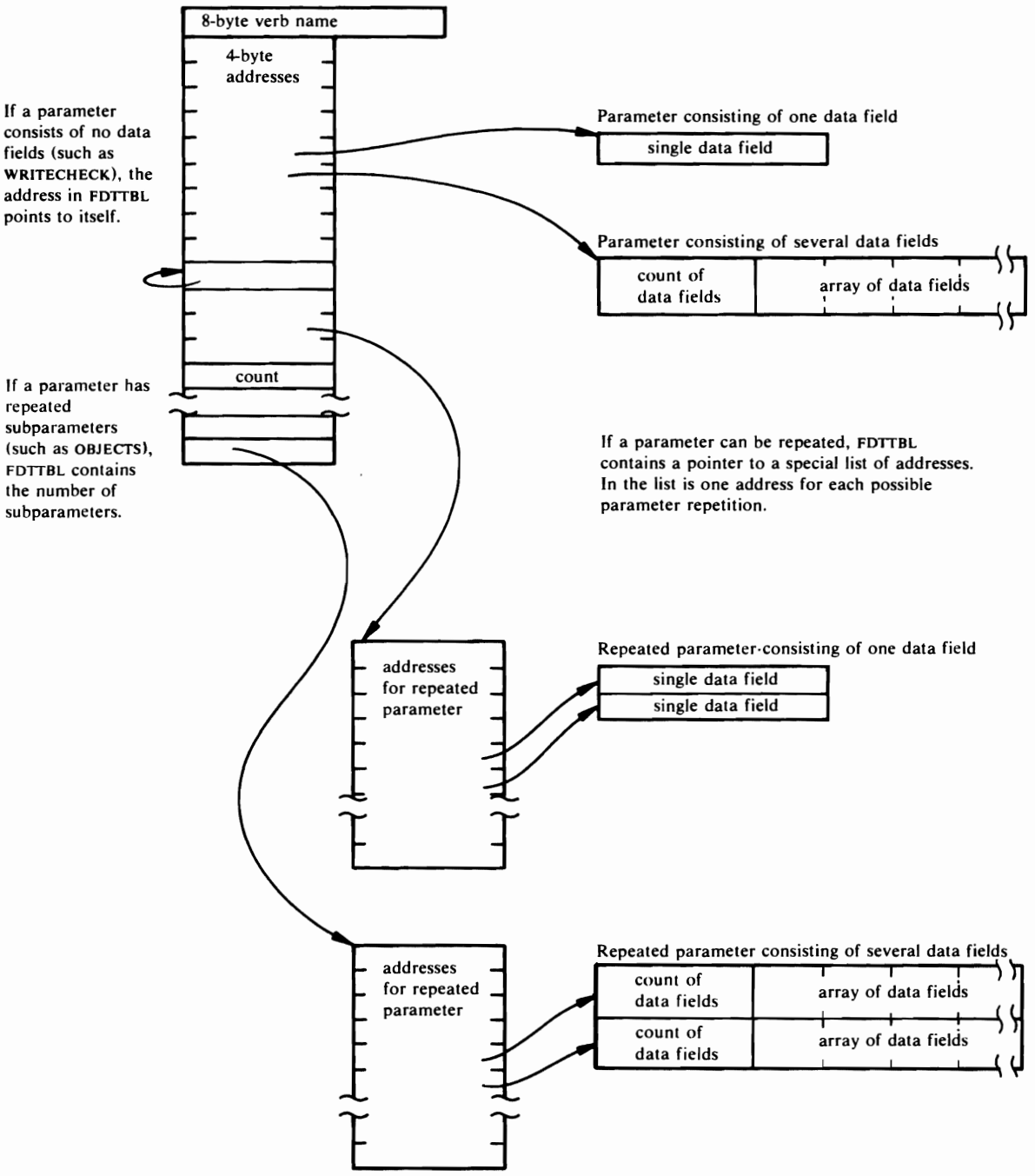


Figure 13. FDT (Function Data Table)

String Data Area

For a character string or hexadecimal string with or without a password the format is:

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	*PLN	Number of characters in the following password. This field does not exist if a password is not allowed with the string.
1 (1)	8	*PAS	Password, if supplied, left-justified and padded with blanks. This field does not exist if a password is not allowed with the string.
9 (9)	1	*LEN	Number of characters in the following field.
10 (A)	*v*	*VAL	Character string left-justified and padded with blanks. The string does not contain deliminters. Double apostrophes are converted to single apostrophes and hexadecimal input is converted to EBCDIC.

Data Set Name or Data Area

For a data set name or generic data set name all with or without a password and member name, the format is:

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	*PLN	Number of characters in the following password.
1 (1)	8	*PAS	Password, if supplied, left-justified and padded with blanks.
9 (9)	1	*POS	Position of any * in the data set name.
10 (A)	1	*FLG	Flag byte:
	1...	*FUQ	This flag is used only if Access Method Services is invoked interactively with TSO. A '1' means the data set name is to be qualified according to TSO naming conventions. Refer to *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor*. '0' means the data set name is to be used without qualification—that is, without any additions.
11 (B)	1	*MLN	Number of characters in the following member name.
12 (C)	8	*MEM	Member name, if supplied, left-justified and padded with blanks.
20 (14)	1	*LEN	Number of characters in the following field.
21 (15)	*v*	*VAL	Data set name or generic data set name in EBCDIC.

FDTs for Specific Commands

The FDT for each command is shown in two different ways in the following sections. First, there is a table relating the pointers to the parameters in the command. Any omitted fields in this table contain zero. Second there is the FDT description as it is used by the FSR for the command.

ALTER FDT

Offset	*Content*	
0 (0)	A L T E	R b b b
8 (8)	†*entryname/password*	†CATALOG
16 (10)	†*catname/password*	†*dname*
24 (18)	†NEWNAME	†FILE
32 (20)	0	†MASTERPW
40 (28)	†CONTROLPW	†UPDATEPW
48 (30)	†READPW	†CODE
56 (38)	†ATTEMPTS	†AUTHORIZATION
64 (40)	†*entrypoint*	†*string*
72 (48)	0	†TO
80 (50)	†FOR	†OWNER
88 (58)	†ERASE	†NOERASE
96 (60)	†SHAREOPTIONS	0
104 (68)	†NULLIFY	†MASTERPW
112 (70)	†CONTROLPW	†UPDATEPW
120 (78)	†READPW	0
128 (80)	†FREESPACE	†*cipercent*
136 (88)	†*capercent*	†WRITECHECK
144 (90)	†NOWRITECHECK	†BUFFERSPACE
152 (98)	†ADDVOLUMES	†REMOVEVOLUMES
160 (A0)	0	†INHIBIT
168 (A8)	†UNINHIBIT	†OWNER
176 (B0)	†CODE	†RETENTION
184 (B8)	†AUTHORIZATION	†MODULE
192 (C0)	†STRING	†*crossregion*
200 (C8)	†*crosssystem*	†EMPTY
208 (D0)	†NOEMPTY	†SCRATCH
216 (D8)	†NOSCRATCH	0
224 (E0)	†EXCEPTIONEXIT	†KEYS
232 (E8)	†*length*	†*offset*
240 (F0)	†RECORDSIZE	†*average*
248 (F8)	†*maximum*	†UNIQUEKEY
256 (100)	†NONUNIQUEKEY	†UPGRADE
264 (108)	†NOUPGRADE	†UPDATE
272 (110)	†NOUPDATE	†EXCEPTIONEXIT
280 (118)	†STAGE	†BIND
288 (120)	†CYLINDERFAULT	†NODESTAGEWAIT
296 (128)	†DESTAGEWAIT	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—ALTERbbb.
8 (8)	4	NTRY	Address of information supplied through the *entryname/password* parameter.
12 (C)	4	CAT	Address of this pointer itself if the CATALOG parameter is supplied.
16 (10)	4	CATLG	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
20 (14)	4	CATDN	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
24 (18)	4	NEWNM	Address of information supplied through the NEWNAME parameter.
28 (1C)	4	INDD	Address of information supplied through the FILE parameter.
32 (20)	4	*	Reserved—contains zero.
36 (24)	4	MASTR	Address of information supplied through the MASTERPW parameter.
40 (28)	4	CNTVL	Address of information supplied through the CONTROLPW parameter.
44 (2C)	4	UPDAT	Address of information supplied through the UPDATEPW parameter.
48 (30)	4	READ	Address of information supplied through the READPW parameter.
52 (34)	4	CODNM	Address of information supplied through the CODE parameter.
56 (38)	4	ATTP	Address of information supplied through the ATTEMPTS parameter.
60 (3C)	4	AUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied.
64 (40)	4	USVR	Address of information supplied through the *entrypoint* supparameter of the AUTHORIZATION parameter.
68 (44)	4	USAR	Address of information supplied through the *string* subparameter of the AUTHORIZATION parameter.
72 (48)	4	*	Reserved—contains zero.
76 (4C)	4	TO	Address of information supplied through the TO parameter.
80 (50)	4	FOR	Address of information supplied through the FOR parameter.
84 (54)	4	OWNER	Address of information supplied through the OWNER parameter.
88 (58)	4	ERASE	Address of this pointer itself if the ERASE parameter is supplied.
92 (5C)	4	NERAS	Address of this pointer itself if the NOERASE parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
96 (60)	4	SHARE	Address of this pointer itself if the SHAREOPTIONS parameter is supplied.
100 (64)	4	*	Reserved—contains zero.
104 (68)	4	NULLF	Address of this pointer itself if the NULLIFY parameter is supplied.
108 (6C)	4	NMSTR	Address of this pointer itself if the MASTERPW subparameter of the NULLIFY parameter is supplied.
112 (70)	4	NCNTV	Address of this pointer itself if the CONTROLPW subparameter of the NULLIFY parameter is supplied.
116 (74)	4	NUPDT	Address of this pointer itself if the UPDATEPW subparameter of the NULLIFY parameter is supplied.
120 (78)	4	NREAD	Address of this pointer itself if the READPW subparameter of the NULLIFY parameter is supplied.
124 (7C)	4	*	Reserved—contains zero.
128 (80)	4	FSPAC	Address of this pointer itself if the FREESPACE parameter is supplied.
132 (84)	4	FSPCI	Address of information supplied through the *cipercent* subparameter of the FREESPACE parameter.
136 (88)	4	FSPCA	Address of information supplied through the *capercent* subparameter of the FREESPACE parameter.
140 (8C)	4	WRTCK	Address of this pointer itself if the WRITECHECK parameter is supplied.
144 (90)	4	NWTCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied.
148 (94)	4	BUFSZ	Address of information supplied through the BUFFERSPACE parameter.
152 (98)	4	ADDVL	Address of information supplied through the ADDVOLUMES parameter.
156 (9C)	4	REMVL	Address of information supplied through the REMOVEVOLUMES parameter.
160 (A0)	4	*	Reserved—contains zero.
164 (A4)	4	INHIB	Address of this pointer itself if the INHIBIT parameter is supplied.
168 (A8)	4	UNHIB	Address of this pointer itself if the UNINHIBIT parameter is supplied.
172 (AC)	4	NOWNR	Address of this pointer itself if the OWNER subparameter of the NULLIFY parameter is supplied.
176 (B0)	4	NCDNM	Address of this pointer itself if the CODE subparameter of the NULLIFY parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
180 (B4)	4	NRETN	Address of this pointer itself if the RETENTION subparameter of the NULLIFY parameter is supplied.
184 (B8)	4	NAUTH	Address of this pointer itself if the AUTHORIZATION subparameter of the NULLIFY parameter is supplied.
188 (BC)	4	NMDNM	Address of this pointer itself if the MODULE subparameter of the AUTHORIZATION parameter is supplied.
192 (C0)	4	NSTRG	Address of this pointer itself if the STRING subparameter of the AUTHORIZATION parameter is supplied.
196 (C4)	4	SHAR1	Address of information supplied through the *crossregion* subparameter of the SHAREOPTIONS parameter.
200 (C8)	4	SHAR2	Address of information supplied through the *crosssystem* subparameter of the SHAREOPTIONS parameter.
204 (CC)	4	GDGEM	Address of this pointer itself if the EMPTY parameter is supplied.
208 (D0)	4	GDGNE	Address of this pointer itself if the NOEMPTY parameter is supplied.
212 (D4)	4	GDGSC	Address of this pointer itself if the SCRATCH parameter is supplied.
216 (D8)	4	GDGNS	Address of this pointer itself if the NOSCRATCH parameter is supplied.
220(DC)	4	*	Reserved—contains zeros.
224 (E0)	4	EEXT	Address of information supplied through the *mname* subparameter of the EXCEPTIONEXIT parameter.
228 (E4)	4	KEY	Address of this pointer itself if the KEYS parameter has been supplied.
232 (E8)	4	KEYLN	Address of information supplied through the *length* subparameter of the KEYS parameter.
236 (EC)	4	KEYPS	Address of information supplied through the *offset* subparameter of the KEYS parameter.
240 (F0)	4	RECSZ	Address of this pointer itself if the RECORDSIZE parameter has been supplied.
244 (F4)	4	AREC	Address of information supplied through the *average* subparameter of the RECORDSIZE parameter.
248 (F8)	4	MREC	Address of information supplied through the *maximum* subparameter of the RECORDSIZE parameter.
252 (FC)	4	UNQK	Address of this pointer itself if the UNIQUEKEY parameter has been supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
256 (100)	4	NUNQK	Address of this pointer itself if the NONUNIQUEKEY parameter has been supplied.
260 (104)	4	UPG	Address of this pointer itself if the UPGRADE parameter has been supplied.
264 (108)	4	NUPG	Address of this pointer itself if the NOUPGRADE parameter has been supplied.
268 (10C)	4	UPD	Address of this pointer itself if the UPDATE parameter has been supplied.
272 (110)	4	NUPD	Address of this pointer itself if the NOUPDATE parameter has been supplied.
276 (114)	4	NEEXT	Address of this pointer itself if the EXCEPTIONEXIT subparameter of the NULLIFY parameter has been supplied.
280(118)	4	STAGE	Address of this pointer itself if the STAGE parameter is supplied.
284(11C)	4	BIND	Address of this pointer itself if the BIND parameter is supplied.
288(120)	4	CYLF	Address of this pointer itself if the CYLINDERFAULT parameter is supplied.
292(124)	4	NSTGW	Address of this pointer itself if the NODESTAGEWAIT parameter is supplied.
296(128)	4	STGW	Address of this pointer itself if the DESTAGEWAIT parameter is supplied. subparameter of the NULLIFY parameter has been supplied.

BLDINDEX FDT

Offset	*Content*	
0 (0)	**B L D I**	**N D E X**
8 (8)	†INFILE	0
16 (10)	†OUTFILE	0
24 (18)	†CATALOG	†WORKFILES
32 (20)	†*dname1*	†*dname2*
40 (28)	†EXTERNALSORT	†INTERNALSORT

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb—BLDINDEX
8 (8)	4	IFILE	Address of informationsupplied through the INFILE parameter.
12 (C)	4	IDS	Address of the information supplied through the INDATASET parameter.
16 (10)	4	OFILE	Address of information supplied through the OUTFILE parameter.
20 (14)	4	ODS	Address of the information supplied through the OUTDATASET parameter.
24 (18)	4	CAT	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
28 (1C)	4	WFILE	Address of this pointer itself if the WORKFILES parameter has been supplied.
32 (20)	4	WFLE1	Address of information supplied through the *dname1* subparameter of the WORKFILES parameter.
36 (24)	4	WFLE2	Address of information supplied through the *dname2* subparameter of the WORKFILES parameter.
40 (28)	4	ESORT	Address of this pointer itself if the EXTERNALSORT parameter has been supplied.
44 (2C)	4	ISORT	Address of this pointer itself if the INTERNALSORT parameter has been supplied.

CHKLIST FDT

Offset	*Content*	
0 (0)	C H K L	I S T
8 (8)	INFILE	OUTFILE
16 (10)	CHECKID	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	FDTVERB	Verb aligned left and padded with blanks—'CHKLIST␣'.
8(8)	4	IFILE	Address of information supplied through the INFILE parameter.
12(C)	4	OFILE	Address of information supplied through the OUTFILE parameter.
16(10)	4	CHKID	Address of information supplied through the CHECKID parameter.

CNVTCAT FDT

Offset	*Content*	
0 (0)	C N V T	C A T b
8 (8)	†INFILE	†INDATASET
16 (10)	†CATALOG	†*catname/password*
24 (18)	†*dname*	†CVOLEQUATES
32 (20)	†*catname*	†*volser*
40 (28)	†*catname/password*	†LIST
48 (30)	†NOLIST	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—CNVTCATb.
8 (8)	4	IFILE	Address of information supplied through the INFILE parameter.
12 (C)	4	IDS	Address of information supplied through the INDATASET parameter.
16 (10)	4	CAT	Address of this pointer itself if the CATALOG parameter is supplied.
20 (14)	4	CATNM	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
24 (18)	4	CATDN	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
28 (1C)	4	CVEQU	Address of a count of subparameters supplied through the CVOLEQUATES parameter.
32 (20)	4	CVECNPTR	Address of information supplied through the *catname* subparameter of the CVOLEQUATES parameter.
36 (24)	4	CVEVSPTR	Address of information supplied through the *volser* subparameter of the CVOLEQUATES subparameter.
40 (28)	4	MRCAT	Address of information supplied through the MASTERCATALOG parameter.
44 (2C)	4	LIST	Address of this pointer itself if the LIST parameter is supplied or defaulted.
48 (30)	4	NLIST	Address of this pointer itself if the NOLIST parameter is supplied.

DEFINE FDT

There are eight illustrations relating the pointers of the FDT to the FSR parameters in the following order: alias, cluster, generation data group, master catalog, non-VSAM data sets, page spaces, spaces for VSAM data sets, and user catalog.

DEFINE ALIAS

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
48 (30)	0	†ALIAS
80 (50)	0	†NAME
768 (300)	0	†RELATE
1352 (548)	0	

DEFINE ALTERNATEINDEX

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
32 (20)	0	†DATA
40 (28)	†INDEX	0
56 (38)	0	†ALTERNATEINDEX
88 (58)	0	†NAME
96 (60)	†NAME	0
128 (80)	†MODEL	†*entryname/password*
136 (88)	†*catname/password*	†*dname*
144 (90)	†MODEL	†*entryname/password*
152 (98)	†*catname/password*	†*dname*
160 (A0)	0	0
168 (A8)	†MASTERPW	†MASTERPW
176 (B0)	0	0
184 (B8)	†CONTROLPW	†CONTROLPW
192 (C0)	0	0
200 (C8)	†UPDATEPW	†UPDATEPW
208 (D0)	0	0
216 (D8)	†READPW	†READPW
224 (E0)	0	0
232 (E8)	†CODE	†CODE
240 (F0)	0	0
248 (F8)	†ATTEMPTS	†ATTEMPTS
256 (100)	0	0
272 (110)	†AUTHORIZATION	†*entrypoint*

Offset	*Content*	
	D E F I	N E b b
280 (118)	†*string*	†AUTHORIZATION
288 (120)	†*entrypoint*	†*string*
296 (128)	0	0
304 (130)	0	0
312 (138)	0	0
320 (140)	†OWNER	†OWNER
328 (148)	0	0
336 (150)	0	†SHAREOPTIONS
344 (158)	†*crossregion*	†*crosssystem*
352 (160)	†SHAREOPTIONS	†*crossregion*
360 (168)	†*crosssystem*	0
368 (170)	0	†ERASE
376 (178)	†NOERASE	0
384 (180)	0	0
392 (188)	†KEYS	†*length*
400 (190)	†*position*	0
408 (198)	0	†REPLICATE
416 (1A0)	†NOREPLICATE	0
424 (1A8)	0	†IMBED
432 (1B0)	†NOIMBED	0
440 (1B8)	0	0
448 (1C0)	†FILE	†FILE
456 (1C8)	0	0
472 (1D8)	†VOLUMES	†VOLUMES
480 (1E0)	0	0
488 (1E8)	0	†KEYRANGES
496 (1F0)	†*lowkey*	†*highkey*
512 (200)	0	0
520 (208)	†ORDERED	†UNORDERED
528 (210)	†ORDERED	†UNORDERED
536 (218)	0	†SUBALLOCATION
544 (220)	†SUBALLOCATION	0
552 (228)	†UNIQUE	†UNIQUE
560 (230)	0	0
576 (240)	0	0
584 (248)	0	0
600 (258)	0	†TRACKS
608 (260)	†CYLINDERS	†RECORDS
616 (268)	†*primary*	†*secondary*
624 (270)	†TRACKS	†CYLINDERS
632 (278)	†RECORDS	†*primary*
640 (280)	†*secondary*	0

Offset	*Content*	
	D E F I	N E b b
664 (298)	↑RECORDSIZE	↑*average*
672 (2A0)	↑*maximum*	0
680 (2A8)	0	0
688 (2B0)	0	↑WRITECHECK
696 (2B8)	↑NOWRITECHECK	↑WRITECHECK
704 (2C0)	↑NOWRITECHECK	0
712 (2C8)	0	↑SPEED
720 (2D0)	↑RECOVERY	0
728 (2D8)	0	0
736 (2E0)	↑FREESPACE	↑*cipercent*
744 (2E8)	↑*capercent*	0
752 (2F0)	0	↑BUFFERSPACE
760 (2F8)	0	↑CONTROLINTERVALSIZE
768 (300)	↑CONTROLINTERVALSIZE	0
1024 (400)	0	0
1032 (408)	0	0
1048 (418)	0	0
1056 (420)	↑EXCEPTIONEXIT	↑EXCEPTIONEXIT
1064 (428)	↑NUMBERED	↑REUSE
1072 (430)	↑REUSE	↑REUSE
1080 (438)	↑NOREUSE	↑NOREUSE
1088 (440)	↑NOREUSE	↑SPANNED
1096 (448)	↑SPANNED	↑NONSPANNED
1104 (450)	↑NONSPANNED	0
1208 (4B8)	0	0
1216 (4C0)	0	0
1248 (4E0)	0	0
1256 (4E8)	0	0
1288 (508)	0	0
1296 (510)	0	0
1320 (528)	0	↑*primary*
1328 (530)	↑*secondary*	↑*primary*
1336 (538)	↑*secondary*	↑*primary*
1344 (540)	↑*secondary*	↑*primary*
1352 (548)	↑*secondary*	
1360 (550)	↑MODEL	↑*entryname/password*
1368 (558)	↑*catname/password*	↑*dname*
1376 (560)	↑MASTERPW	↑CONTROLPW
1384 (568)	↑UPDATEPW	↑READPW
1392 (570)	↑CODE	↑ATTEMPTS
1400 (578)	↑AUTHORIZATION	↑*entrypoint*
1408 (580)	↑*string*	↑TO

Offset	*Content*	
	D E F I	N E b b
1416 (588)	†FOR	†OWNER
1424 (590)	†SHAREOPTIONS	†*crosspartition*
1432 (598)	†*crosssystem*	†ERASE
1440 (5A0)	†NOERASE	†KEYS
1448 (5A8)	†*length*	†*offset*
1456 (5B0)	†REPLICATE	†NOREPLICATE
1464 (5B8)	†IMBED	†NOIMBED
1472 (5C0)	†FILE	†VOLUMES
1480 (5C8)	†KEYRANGES	†*lowkey*
1488 (5D0)	†*highkey*	†ORDERED
1496 (5D8)	†UNORDERED	†SUBALLOCATION
1504 (5E0)	†UNIQUE	†TRACKS
1512 (5E8)	†*primary*	†*secondary*
1520 (5F0)	†CYLINDERS	†*primary*
1528 (5F8)	†*secondary*	†RECORDS
1536(600)	†*primary*	†*secondary*
1544 (608)	†RECORDSIZE	†*average*
1552 (610)	†*maximum*	†WRITECHECK
1560 (618)	†NOWRITECHECK	†SPEED
1568 (620)	†RECOVERY	†FREESPACE
1576 (628)	†*cipercent*	†*capercent*
1584 (630)	†BUFFERSPACE	†CONTROLINTERVALSIZE
1592 (638)	†RELATE	†EXCEPTIONEXIT
1600 (640)	†REUSE	†NOREUSE
1608 (648)	†UNIQUEKEY	†NONUNIQUEKEY
1616 (650)	†UNIQUEKEY	†NONUNIQUEKEY
1624 (658)	†UPGRADE	†NOUPGRADE
1728 (6C0)	0	†STAGE
1736 (6C8)	†STAGE	†STAGE
1744 (6D0)	0	†BIND
1752 (6D8)	†BIND	†BIND
1760 (6E0)	0	†CYLINDERFAULT
1768(6E8)	†CYLINDERFAULT	†CYLINDERFAULT
1784 (6F8)	0	†NODESTAGEWAIT
1792 (700)	†NODESTAGEWAIT	†NODESTAGEWAIT
1808 (710)	0	†DESTAGEWAIT
1816 (718)	†DESTAGEWAIT	†DESTAGEWAIT

DEFINE CLUSTER

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	↑CATALOG	↑*catname/password*
16 (10)	↑*dname*	0
24 (18)	0	↑CLUSTER
32 (20)	0	↑DATA
40 (28)	↑INDEX	0
72 (48)	↑NAME	0
88 (58)	0	↑NAME
96 (60)	↑NAME	↑INDEXED
104 (68)	↑NONINDEXED	0
112 (70)	↑MODEL	↑*entryname/password*
120 (78)	↑*catname/password*	↑*dname*
128 (80)	↑MODEL	↑*entryname/password*
136 (88)	↑*catname/password*	↑*dname*
144 (90)	↑MODEL	↑*entryname/password*
152 (98)	↑*catname/password*	↑*dname*
160 (A0)	0	↑MASTERPW
168 (A8)	↑MASTERPW	↑MASTERPW
176 (B0)	0	↑CONTROLPW
184 (B8)	↑CONTROLPW	↑CONTROLPW
192 (C0)	0	↑UPDATEPW
200 (C8)	↑UPDATEPW	↑UPDATEPW
208 (D0)	0	↑READPW
216 (D8)	↑READPW	↑READPW
224 (E0)	0	↑CODE
232 (E8)	↑CODE	↑CODE
240 (F0)	0	↑ATTEMPTS
248 (F8)	↑ATTEMPTS	↑ATTEMPTS
256 (100)	0	↑AUTHORIZATION
272 (110)	↑AUTHORIZATION	↑*entrypoint*
280 (118)	↑*string*	↑AUTHORIZATION
288 (120)	↑*entrypoint*	↑*string*
296 (128)	0	↑TO
304 (130)	0	↑FOR
312 (138)	0	↑OWNER
320 (140)	↑OWNER	↑OWNER
328 (148)	↑SHAREOPTIONS	↑*crossregion*
336 (150)	↑*crosssystem*	↑SHAREOPTIONS
344 (158)	↑*crossregion*	↑*crosssystem*

Offset	*Content*	
	D E F I	N E b b
352 (160)	†SHAREOPTIONS	†*crossregion*
360 (168)	†*crosssystem*	†ERASE
368 (170)	†NOERASE	†ERASE
376 (178)	†NOERASE	†KEYS
384 (180)	†*length*	†*position*
392 (188)	†KEYS	†*length*
400 (190)	†*position*	†REPLICATE
408 (198)	†NOREPLICATE	†REPLICATE
416 (1A0)	†NOREPLICATE	†IMBED
424 (1A8)	†NOIMBED	†IMBED
432 (1B0)	†NOIMBED	0
440 (1B8)	†FILE	0
448 (1C0)	†FILE	†FILE
456 (1C8)	0	†VOLUMES
472 (1D8)	†VOLUMES	†VOLUMES
480 (1E0)	†KEYRANGES	†*lowkey*
488 (1E8)	†*highkey*	†KEYRANGES
496 (1F0)	†*lowkey*	†*highkey*
512 (200)	†ORDERED	†UNORDERED
520 (208)	†ORDERED	†UNORDERED
528 (210)	†ORDERED	†UNORDERED
536 (218)	†SUBALLOCATION	†SUBALLOCATION
544 (220)	†SUBALLOCATION	†UNIQUE
552 (228)	†UNIQUE	†UNIQUE
560 (230)	0	†TRACKS
576 (240)	†CYLINDERS	0
584 (248)	0	†RECORDS
600 (258)	0	†TRACKS
608 (260)	†CYLINDERS	†RECORDS
616 (268)	†*primary*	†*secondary*
624 (270)	†TRACKS	†CYLINDERS
632 (278)	†RECORDS	†*primary*
640 (280)	†*secondary*	0
648 (288)	†RECORDSIZE	0
656 (290)	†*average*	†*maximum*
664 (298)	†RECORDSIZE	†*average*
672 (2A0)	†*maximum*	0
680 (2A8)	†WRITECHECK	0
688 (2B0)	†NOWRITECHECK	†WRITECHECK
696 (2B8)	†NOWRITECHECK	†WRITECHECK

Offset	*Content*	
	D E F I	N E b b
704 (2C0)	↑NOWRITECHECK	↑SPEED
712 (2C8)	↑RECOVERY	↑SPEED
720 (2D0)	↑RECOVERY	↑FREESPACE
728 (2D8)	↑*cipercent*	↑*capercent*
736 (2E0)	↑FREESPACE	↑*cipercent*
744 (2E8)	↑*capercent*	0
752 (2F0)	↑BUFFERSPACE	↑BUFFERSPACE
760 (2F8)	↑CONTROLINTERVALSIZE	↑CONTROLINTERVALSIZE
768 (300)	↑CONTROLINTERVALSIZE	0
1024 (400)	0	↑*entrypoint*
1032 (408)	↑*string*	0
1048 (418)	0	↑EXCEPTIONEXIT
1056 (420)	↑EXCEPTIONEXIT	↑EXCEPTIONEXIT
1064 (428)	↑NUMBERED	↑REUSE
1072 (430)	↑REUSE	↑REUSE
1080 (438)	↑NOREUSE	↑NOREUSE
1088 (440)	↑NOREUSE	↑SPANNED
1096 (448)	↑SPANNED	↑NONSPANNED
1104 (450)	↑NONSPANNED	0
1208 (4B8)	0	↑*primary*
1216 (4C0)	↑*secondary*	0
1248 (4E0)	0	↑*primary*
1256 (4E8)	↑*secondary*	0
1288 (508)	0	↑*primary*
1296 (510)	↑*secondary*	0
1320 (528)	0	↑*primary*
1328 (530)	↑*secondary*	↑*primary*
1336 (538)	↑*secondary*	↑*primary*
1344 (540)	↑*secondary*	↑*primary*
1352 (548)	↑*secondary*	
1728 (6C0)	↑STAGE	0
1736 (6C8)	↑STAGE	↑STAGE
1744 (6D0)	↑BIND	0
1752 (6D8)	↑BIND	↑BIND
1760 (6E0)	↑CYLINDERFAULT	0
1768 (6E8)	↑CYLINDERFAULT	↑CYLINDERFAULT
1784 (6F8)	↑NODESTAGEWAIT	0
1784 (6F8)	↑NODESTAGEWAIT	0
1792 (700)	↑NODESTAGEWAIT	↑NODESTAGEWAIT
1808 (710)	↑DESTAGEWAIT	0
1816 (718)	↑DESTAGEWAIT	↑DESTAGEWAIT

DEFINE GENERATIONDATAGROUP

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
56 (38)	†GENERATIONDATAGROUP	0
88 (58)	†NAME	0
776 (308)	†EMPTY	†NOEMPTY
784 (310)	†LIMIT	†SCRATCH
792 (318)	†NOSCRATCH	0
1352 (548)	0	0
1824 (720)	0	†TO
1832 (728)	0	†FOR
1840 (730)	0	†OWNER

DEFINE MASTERCATALOG

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	†MASTERCATALOG
32 (20)	0	†DATA
40 (28)	†INDEX	0
64 (40)	0	†NAME
88 (58)	0	†NAME
96 (60)	†NAME	0
160 (A0)	†MASTERPW	0
176 (B0)	†CONTROLPW	0
192 (C0)	†UPDATEPW	0
208 (D0)	†READPW	0
224 (E0)	†CODE	0
240 (F0)	†ATTEMPTS	0
256 (100)	†AUTHORIZATION	0
264 (108)	†*entrypoint*	†*string*
296 (128)	†TO	0
304 (130)	†FOR	0
312 (138)	†OWNER	0
432 (1B0)	0	†FILE
456 (1C8)	†VOLUMES	0
560 (230)	†TRACKS	0
568 (238)	0	†CYLINDERS
584 (248)	†RECORDS	0
600 (258)	0	†TRACKS
608 (260)	†CYLINDERS	†RECORDS

Offset	*Content*	
	D E F I	N E b b
616 (268)	†*primary*	†*secondary*
624 (270)	†TRACKS	†CYLINDERS
632 (278)	†RECORDS	†*primary*
640 (280)	†*secondary*	0
672 (2A0)	0	†WRITECHECK
680 (2A8)	0	†NOWRITECHECK
688 (2B0)	0	†WRITECHECK
696 (2B8)	†NOWRITECHECK	†WRITECHECK
704 (2C0)	†NOWRITECHECK	0
744 (2E8)	0	†BUFFERSPACE
752 (2F0)	0	†BUFFERSPACE
1104 (450)	0	†RECOVERABLE
1112 (458)	0	†RECOVERABLE
1120 (460)	†NOTRECOVERABLE	0
1128 (468)	†NOTRECOVERABLE	0
1200 (4B0)	0	†*primary*
1208 (4B8)	†*secondary*	0
1240 (4D8)	0	†*primary*
1248 (4E0)	†*secondary*	0
1280 (500)	0	†*primary*
1288 (508)	†*secondary*	0
1320 (528)	0	†*primary*
1328 (530)	†*secondary*	†*primary*
1336 (538)	†*secondary*	†*primary*
1344 (540)	†*secondary*	†*primary*
1352 (548)	†*secondary*	
1776 (6F0)	†NODESTAGEWAIT	0
1800 (708)	†DESTAGEWAIT	0

DEFINE NONVSAM

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
48 (30)	†NONVSAM	0
80 (50)	†NAME	0
464 (1D0)	0	†VOLUMES
504 (1F8)	†DEVICETYPES	†FILESEQUENCENUMBER
1352 (548)	0	
1824 (720)	†TO	0
1832 (728)	†FOR	0
1840 (730)	†OWNER	0

DEFINE PAGESPACE

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
32 (20)	†PAGESPACE	0
72 (48)	0	†NAME
888 (378)	†MODEL	†*entryname/password*
896 (380)	†*catname/password*	†*dname*
904 (388)	†MASTERPW	†CONTROLPW
912 (390)	†UPDATEPW	†READPW
920 (398)	†CODE	†ATTEMPTS
928 (3A0)	†AUTHORIZATION	†*entrypoint*
936 (3A8)	†*string*	†TO
944 (3B0)	†FOR	†OWNER
960(3B8)	0	†ERASE
968 (3C8)	†NOERASE	†FILE
976 (3D0)	†VOLUMES	†SUBALLOCATION
984 (3D8)	†UNIQUE	†TRACKS
992 (3E0)	†CYLINDERS	†RECORDS
1000(3E8)	†SWAP (**VS2.03.807**)	†NOSWAP (**VS2.03.807**)
1232 (4D0)	0	†*primary*
1240 (4D8)	†*secondary*	0
1264 (4F0)	0	†*primary*
1272 (4F8)	†*secondary*	0
1312 (520)	0	†*primary*
1320 (528)	†*secondary*	0
1352 (548)	0	

DEFINE PATH

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	†CATALOG	†*catname/password*
16 (10)	†*dname*	0
64 (40)	†PATH	0
1640 (668)	0	†NAME
1648(570)	†MODEL	†*entryname/password*
1656 (578)	†*catname/password*	†*dname*
1664 (680)	†MASTERPW	†CONTROLPW
1672 (688)	†UPDATEPW	†READPW
1680 (690)	†CODE	†ATTEMPTS
1688 (698)	†AUTHORIZATION	†*entrypoint*
1696 (6A0)	†*string*	†TO
1704 (6A8)	†FOR	†OWNER
1712 (6B0)	†FILE	†UPDATE
1720 (6B8)	†NOUPDATE	†PATHENTRY

DEFINE SPACE

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	↑CATALOG	↑*catname/password*
16 (10)	↑*dname*	0
40 (28)	0	↑SPACE
440 (1B8)	0	↑FILE
464 (1BE)	↑VOLUMES	0
568 (238)	↑TRACKS	0
576 (240)	0	↑CYLINDERS
592 (250)	↑RECORDS	0
640 (280)	0	↑CANDIDATE
648 (288)	0	↑RECORDSIZE
1008 (3F0)	↑*average*	↑*maximum*
1216 (4C0)	0	↑*primary*
1224 (4C8)	↑*secondary*	0
1272 (4F8)	0	↑*primary*
1280 (500)	↑*secondary*	0
1296 (510)	0	↑*primary*
1304 (518)	↑*secondary*	0
1352 (548)	0	

DEFINE USERCATALOG

Offset	*Content*	
0 (0)	D E F I	N E b b
8 (8)	↑CATALOG	↑*catname/password*
16 (10)	↑*dname*	0
24 (18)	↑USERCATALOG	0
32 (20)	0	↑DATA
40 (28)	↑INDEX	0
104 (68)	0	↑MODEL
600 (258)	0	↑TRACKS
608 (260)	↑CYLINDERS	↑RECORDS
616 (268)	↑*primary*	↑*secondary*
624 (270)	↑TRACKS	↑CYLINDERS
632 (278)	↑RECORDS	↑*primary*
640 (280)	↑*secondary*	0
688 (2B0)	0	↑WRITECHECK
696 (2B8)	↑NOWRITECHECK	↑WRITECHECK
704 (2C0)	↑NOWRITECHECK	0
752 (2F0)	0	↑BUFFERSPACE
792 (318)	0	↑NAME
800 (320)	↑MASTERPW	↑CONTROLPW

Offset	*Content*	
	D E F I	N E b b
808 (328)	†UPDATEPW	†READPW
816 (330)	†CODE	†ATTEMPTS
824 (338)	†AUTHORIZATION	†*entrypoint*
832 (340)	†*string*	†TO
840 (348)	†FOR	†OWNER
848 (350)	†FILE	†VOLUMES
856 (358)	†TRACKS	†CYLINDERS
864 (360)	†RECORDS	0
872 (368)	0	†WRITECHECK
880 (370)	†NOWRITECHECK	†BUFFERSPACE
1016 (3F8)	†*entrypoint*	†*catname/password*
1024 (400)	†*dname*	0
1112 (458)	†RECOVERABLE	†RECOVERABLE
1120 (460)	0	†NOTRECOVERABLE
1128 (468)	†NOTRECOVERABLE	0
1256 (4E8)	0	†*primary*
1264 (4F0)	†*secondary*	0
1304 (518)	0	†*primary*
1312 (520)	†*secondary*	0
1320 (528)	0	†*primary*
1328 (530)	†*secondary*	†*primary*
1336 (538)	†*secondary*	†*primary*
1344 (540)	†*secondary*	†*primary*
1352 (548)	†*secondary*	
1776 (6F0)	0	†NODESTAGEWAIT
1792 (700)	†NODESTAGEWAIT	†NODESTAGEWAIT
1800 (708)	0	†DESTAGEWAIT
1816 (718)	†DESTAGEWAIT	†DESTAGEWAIT

DEFINE FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—DEFINEbb.
8 (8)	4	CAT	Address of this pointer itself if the CATALOG parameter has been supplied.
12 (C)	4	CATLG	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
16 (10)	4	CATDN	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
20 (14)	4	MCAT	Address of this pointer itself if the MASTERCATALOG parameter has been supplied—that is, if you are defining a master catalog.
24 (18)	4	UCAT	Address of this pointer itself if the USERCATALOG parameter is supplied—that is, if you are defining a USERCATALOG.
28 (1C)	4	CLST	Address of this pointer itself if the CLUSTER parameter is supplied—that is if you are defining a CLUSTER.
32 (20)	4	PGSP	Address of this pointer itself if the PAGESPACE parameter is supplied—that is, if you are defining a PAGESPACE.
36 (24)	4	DATAA	Address of this pointer itself if the DATA parameter is supplied.
40 (28)	4	INDEX	Address of this pointer itself if the INDEX parameter is supplied.
44 (2C)	4	SPACE	Address of this pointer itself if the SPACE parameter is supplied—that is, if you are defining a VSAM data space.
48 (30)	4	ALIEN	Address of this pointer itself if the NONVSAM parameter is supplied—that is, if you are defining a non-VSAM data set.
52 (34)	4	ALIAS	Address of this pointer itself if the ALIAS parameter is supplied—that is, if you are defining an alias.
56 (38)	4	GENDG	Address of this pointer itself if the GENERATIONDATA- GROUP parameter is supplied—that is, if you are defining a generation data group.
60 (3C)	4	AIX	Address of this pointer itself is the ALTERNATEINDEX parameter is supplied—that is, if you are defining an alternate index.
64 (40)	4	PATH	Address of this pointer itself if the PATH parameter is supplied—that is, if you are defining a path.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
68 (44)	4	METRY	Address of information supplied through the NAME parameter if NAME is supplied under MASTERCATALOG.
72 (48)	4	CETRY	Address of information supplied through the NAME parameter if NAME is supplied under CLUSTER.
76 (4C)	4	PETRY	Address of information supplied through the NAME parameter if NAME is supplied under PAGESPACE.
80 (50)	4	AETRY	Address of information supplied through the NAME parameter if NAME is supplied under NONVSAM.
84 (54)	4	XETRY	Address of information supplied through the NAME parameter if NAME is supplied under ALIAS.
88 (58)	4	GETRY	Address of information supplied through the NAME parameter if NAME is supplied under GENERATIONDATA- GROUP.
92 (5C)	4	DETRY	Address of information supplied through the NAME parameter if NAME is supplied under DATA.
96 (60)	4	IETRY	Address of information supplied through the NAME parameter if NAME is supplied under INDEX.
100 (64)	4	CINDX	Address of this pointer itself if INDEXED is supplied under CLUSTER.
104 (68)	4	CNIDX	Address of this pointer itself if NONINDEXED is supplied under CLUSTER.
108 (6C)	4	UMODL	Address of this pointer itself if MODEL is supplied under USERCATALOG.
112 (70)	4	CMODL	Address of this pointer itself if the MODEL parameter is supplied under CLUSTER.
116 (74)	4	CENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under CLUSTER.
120 (78)	4	CMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under CLUSTER.
124 (7C)	4	CMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under CLUSTER.
128 (80)	4	DMODL	Address of this pointer itself if the MODEL parameter is supplied under DATA.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
132 (84)	4	DENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under DATA.
136 (88)	4	DMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under DATA.
140 (8C)	4	DMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under DATA.
144 (90)	4	IMODL	Address of this pointer itself if MODEL is supplied under INDEX.
148 (94)	4	IENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under INDEX.
152 (98)	4	IMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under INDEX.
156 (9C)	4	IMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under INDEX.
160 (A0)	4	MMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under MASTERCATALOG.
164 (A4)	4	CMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under CLUSTER.
168 (A8)	4	DMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under DATA.
172 (AC)	4	IMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under INDEX.
176 (B0)	4	MCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under MASTERCATALOG.
180 (B4)	4	CCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under CLUSTER.
184 (B8)	4	DCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under DATA.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
188 (BC)	4	ICINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under INDEX.
192 (C0)	4	MUPDT	Address of information supplied through the UPDATEPW if UPDATEPW is supplied under MASTERCATALOG.
196 (C4)	4	CUPDT	Address of information supplied through the UPDATEPW if UPDATEPW is supplied under CLUSTER.
200 (C8)	4	DUPDT	Address of information supplied through the UPDATEPW if UPDATEPW is supplied under DATA.
204 (CC)	4	IUPDT	Address of information supplied through the UPDATEPW if UPDATEPW is supplied under INDEX.
208 (D0)	4	MREAD	Address of information supplied through the READPW parameter if READPW is supplied under MASTERCATALOG.
212 (D4)	4	CREAD	Address of information supplied through the READPW parameter if READPW is supplied under CLUSTER.
216 (D8)	4	DREAD	Address of information supplied through the READPW parameter if READPW is supplied under DATA.
220 (DC)	4	IREAD	Address of information supplied through the READPW parameter if READPW is supplied under INDEX.
224 (E0)	4	MCODE	Address of information supplied through the CODE parameter if CODE is supplied under MASTERCATALOG.
228 (E4)	4	CCODE	Address of information supplied through the CODE parameter if CODE is supplied under CLUSTER.
232 (E8)	4	DCODE	Address of information supplied through the CODE parameter if CODE is supplied under DATA.
236 (EC)	4	ICODE	Address of information supplied through the CODE parameter if CODE is supplied under INDEX.
240 (F0)	4	MATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under MASTERCATALOG.
244 (F4)	4	CATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under CLUSTER.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
248 (F8)	4	DATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under DATA.
252 (FC)	4	IATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under INDEX.
256 (100)	4	MAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under MASTERCATALOG.
260 (104)	4	CAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under CLUSTER.
264 (108)	4	MEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under MASTERCATALOG.
268 (10C)	4	MSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under MASTERCATALOG.
272 (110)	4	DAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under DATA.
276 (114)	4	DEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under DATA.
280 (118)	4	DSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under DATA.
284 (11C)	4	IAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under INDEX.
288 (120)	4	IEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if the AUTHORIZATION parameter is supplied under INDEX.
292 (124)	4	ISTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if the AUTHORIZATION parameter is supplied under INDEX.
296 (128)	4	MTO	Address of information supplied through the TO parameter if TO is supplied under MASTERCATALOG.
300 (12C)	4	CTO	Address of information supplied through the TO parameter if TO is supplied under CLUSTER.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
304 (130)	4	MFOR	Address of information supplied through the FOR parameter if FOR is supplied under MASTERCATALOG.
308 (134)	4	CFOR	Address of information supplied through the FOR parameter if FOR is supplied under CLUSTER.
312 (138)	4	MOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under MASTERCATALOG.
316 (13C)	4	COWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under CLUSTER.
320 (140)	4	DOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under DATA.
324 (144)	4	IOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under INDEX.
328 (148)	4	CSHAR	Address of this pointer itself if the SHAREOPTIONS parameter is supplied under CLUSTER.
332 (14C)	4	CSHR1	Address of information supplied through the *crossregion* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under CLUSTER.
336 (150)	4	CSHR2	Address of information supplied through the *crosssystem* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under CLUSTER.
340 (154)	4	DSHAR	Address of this pointer itself if the SHAREOPTIONS parameter is supplied under DATA.
344 (158)	4	DSHR1	Address of information supplied through the *crossregion* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under DATA.
348 (15C)	4	DSHR2	Address of information supplied through the *crossregion* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under DATA.
352 (160)	4	ISHAR	Address of this pointer itself if the SHAREOPTIONS parameter is supplied under INDEX.
356 (164)	4	ISHR1	Address of information supplied through the *crossregion* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under INDEX.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
360 (168)	4	ISHR2	Address of information supplied through the *crosssystem* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under INDEX.
364 (16C)	4	CERAS	Address of this pointer itself if the ERASE parameter is supplied under CLUSTER.
368 (170)	4	CNERS	Address of this pointer itself if the NOERASE parameter is supplied under CLUSTER.
372 (174)	4	DERAS	Address of this pointer itself if the ERASE parameter is supplied under DATA.
376 (178)	4	DNERS	Address of this pointer itself if the NOERASE parameter is supplied under DATA.
380 (17C)	4	CKEY	Address of this pointer itself if the KEYS parameter is supplied under CLUSTER.
384 (180)	4	CKYLN	Address of information supplied through the *length* subparameter of KEYS if KEYS is supplied under CLUSTER.
388 (184)	4	CKYPS	Address of information supplied through the *position* subparameter of KEYS if KEYS is supplied under CLUSTER.
392 (188)	4	DKEY	Address of this pointer itself if the KEYS parameter is supplied under DATA.
396 (18C)	4	DKYLN	Address of information supplied through the *length* subparameter of KEYS if KEYS is supplied under DATA.
400 (190)	4	DKYPS	Address of information supplied through the *position* subparameter of KEYS if KEYS is supplied under DATA.
404 (194)	4	CREPL	Address of this pointer itself if the REPLICATE parameter is supplied under CLUSTER.
408 (198)	4	CNREP	Address of this pointer itself if the NOREPLICATE parameter is supplied under CLUSTER.
412 (19C)	4	IREPL	Address of this pointer itself if the REPLICATE parameter is supplied under INDEX.
416 (1A0)	4	INREP	Address of this pointer itself if the NOREPLICATE parameter is supplied under INDEX.
420 (1A4)	4	CIMBD	Address of this pointer itself if the IMBED parameter is supplied under CLUSTER.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
424 (1A8)	4	CNIBD	Address of this pointer itself if the NOIMBED parameter is supplied under CLUSTER.
428 (1AC)	4	IIMBD	Address of this pointer itself if the IMBED parameter is supplied under INDEX.
432 (1B0)	4	INIBD	Address of this pointer itself if the NOIMBED parameter is supplied under INDEX.
436 (1B4)	4	MINDD	Address of information supplied through the FILE parameter if FILE is supplied under MASTERCATALOG.
440 (1B8)	4	CINDD	Address of information supplied through the FILE parameter if FILE is supplied under CLUSTER.
444 (1BC)	4	SINDD	Address of information supplied through the FILE parameter if FILE is supplied under SPACE.
448 (1C0)	4	DINDD	Address of information supplied through the FILE parameter if FILE is supplied under DATA.
452 (1C4)	4	IINDD	Address of information supplied through the FILE parameter if FILE is supplied under INDEX.
456 (1C8)	4	MVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under MASTERCATALOG.
460 (1CC)	4	CVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under CLUSTER.
464 (1D0)	4	SVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under SPACE.
468 (1D4)	4	AVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under NONVSAM.
472 (1D8)	4	DVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under DATA.
476 (1DC)	4	IVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under INDEX.
480 (1E0)	4	CRANG	Address of a count of subparameters supplied through the KEYRANGES parameter if KEYRANGES is supplied under CLUSTER.
484 (1E4)	4	CRGLOPTR	Address of information supplied through the *lowkey* subparameter of KEYRANGES if KEYRANGES is supplied under CLUSTER.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
488 (1E8)	4	CRGHIPTR	Address of information supplied through the *highkey* subparameter of KEYRANGES if KEYRANGES is supplied under CLUSTER.
492 (1EC)	4	DRANG	Address of a count of subparameters supplied through the KEYRANGES parameter if KEYRANGES is supplied under DATA.
496 (1F0)	4	DRGLOPTR	Address of information supplied through the *lowkey* subparameter of KEYRANGES if KEYRANGES is supplied under DATA.
500 (1F4)	4	DRGHIPTR	Address of information supplied through the *highkey* subparameter of KEYRANGES if KEYRANGES is supplied under DATA.
504 (1F8)	4	ADEVT	Address of information supplied through the DEVICETYPES parameter if DEVICETYPES is supplied under NONVSAM.
508 (1FC)	4	AFSNO	Address of information supplied through the FILESEQUENCENUMBER parameter if FILESEQUENCENUMBER is supplied under NONVSAM.
512 (200)	4	CORDR	Address of this pointer itself if the ORDERED parameter is supplied under CLUSTER.
516 (204)	4	CUORD	Address of this pointer itself if the UNORDERED parameter is supplied under CLUSTER.
520 (208)	4	DORDR	Address of this pointer itself if the ORDERED parameter is supplied under DATA.
524 (20C)	4	DUORD	Address of this pointer itself if the UNORDERED parameter is supplied under DATA.
528 (210)	4	IORDR	Address of this pointer itself if the ORDERED parameter is supplied under INDEX.
532 (214)	4	IUORD	Address of this pointer itself if the UNORDERED parameter is supplied under INDEX.
536 (218)	4	CSUBA	Address of this pointer itself if the SUBALLOCATION parameter is supplied under CLUSTER.
540 (21C)	4	DSUBA	Address of this pointer itself if the SUBALLOCATION parameter is supplied under DATA.
544 (220)	4	ISUBA	Address of this pointer itself if the SUBALLOCATION parameter is supplied under INDEX.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
548 (224)	4	CUNIQ	Address of this pointer itself if the UNIQUE parameter is supplied under CLUSTER.
552 (228)	4	DUNIQ	Address of this pointer itself if the UNIQUE parameter is supplied under DATA.
556 (22C)	4	IUNIQ	Address of this pointer itself if the UNIQUE parameter is supplied under INDEX.
560 (230)	4	MTRKS	Address of this pointer itself if the TRACKS parameter is supplied under MASTERCATALOG.
564 (234)	4	CTRKS	Address of this pointer itself if the TRACKS parameter is supplied under CLUSTER.
568 (238)	4	STRKS	Address of this pointer itself if the TRACKS parameter is supplied under SPACE.
572 (23C)	4	MCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under MASTERCATALOG.
576 (240)	4	CCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under CLUSTER.
580 (244)	4	SCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under SPACE.
584 (248)	4	MRCDS	Address of this pointer itself if the RECORDS parameter is supplied under MASTERCATALOG.
588 (24C)	4	CRCDS	Address of this pointer itself if the RECORDS parameter is supplied under CLUSTER.
592 (250)	4	SRCDS	Address of this pointer itself if the RECORDS parameter is supplied under SPACE.
596 (254)	8	*	Reserved—contains zeros.
604 (25C)	4	DTRKS	Address of this pointer itself if the TRACKS parameter is supplied under DATA.
608 (260)	4	DCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under DATA.
612 (264)	4	DRCDS	Address of this pointer itself if the RECORDS parameter is supplied under DATA.
616 (268)	4	DTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under DATA.
620 (26C)	4	DTKSC	Address of information supplied through the *secondary* subparameter

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			of TRACKS if TRACKS is supplied under DATA.
624 (270)	4	ITRKS	Address of this pointer itself if the TRACKS parameter is supplied under INDEX.
628 (274)	4	ICYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under INDEX.
632 (278)	4	IRCDS	Address of this pointer itself if the RECORDS parameter is supplied under INDEX.
636 (27C)	4	ITKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under INDEX.
640 (280)	4	ITKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under INDEX.
644 (284)	4	SCAND	Address of this pointer itself if the CANDIDATE parameter is supplied under SPACE.
648 (288)	4	CRSIZ	Address of this pointer itself if the RECORDSIZE parameter is supplied under CLUSTER.
652 (28C)	4	SRSIZ	Address of this pointer itself if the RECORDSIZE parameter is supplied under SPACE.
656 (290)	4	CARSZ	Address of information supplied through the *average* subparameter of RECORDSIZE if RECORDSIZE is supplied under CLUSTER.
660 (294)	4	CMRSZ	Address of information supplied through the *maximum* subparameter of RECORDSIZE if RECORDSIZE is supplied under CLUSTER.
664 (298)	4	DRSIZ	Address of this pointer itself if the RECORDSIZE parameter is supplied under DATA.
668 (29C)	4	DARSZ	Address of information supplied through the *average* subparameter of RECORDSIZE if RECORDSIZE is supplied under DATA.
672 (2A0)	4	DMRSZ	Address of information supplied through the *maximum* subparameter of RECORDSIZE if RECORDSIZE is supplied under DATA.
676 (2A4)	4	MWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under MASTERCATALOG.
680 (2A8)	4	CWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under CLUSTER.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
684 (2AC)	4	MNWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under MASTERCATALOG.
688 (2B0)	4	CNWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under CLUSTER.
692 (2B4)	4	DWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under DATA.
696 (2B8)	4	DNWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under DATA.
700 (2BC)	4	IWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under INDEX.
704 (2C0)	4	INWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under INDEX.
708 (2C4)	4	CSPED	Address of this pointer itself if the SPEED parameter is supplied under CLUSTER.
712 (2C8)	4	CRECV	Address of this pointer itself if the RECOVERY parameter is supplied under CLUSTER.
716 (2CC)	4	DSPED	Address of this pointer itself if the SPEED parameter is supplied under DATA.
720 (2D0)	4	DRECV	Address of this pointer itself if the RECOVERY parameter is supplied under DATA.
724 (2D4)	4	CFSPC	Address of this pointer itself if the FREESPACE parameter is supplied under CLUSTER.
728 (2D8)	4	CCIFS	Address of information supplied through the *cipercent* subparameter of FREESPACE if FREESPACE is supplied under CLUSTER.
732 (2DC)	4	CCAFS	Address of information supplied through the *capercent* subparameter of FREESPACE if FREESPACE is supplied under CLUSTER.
736 (2E0)	4	DFSPC	Address of this pointer itself if the FREESPACE parameter is supplied under DATA.
740 (2E4)	4	DCIFS	Address of information supplied through the *cipercent* subparameter of FREESPACE if FREESPACE is supplied under DATA.
744 (2E8)	4	DCAFS	Address of information supplied through the *capercent* subparameter of FREESPACE if FREESPACE is supplied under DATA.
748 (2EC)	4	MBFSZ	Address of information supplied through the BUFFERSPACE

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			parameter if BUFFERSPACE is supplied under MASTERCATALOG.
752 (2F0)	4	CBFSZ	Address of information supplied through the BUFFERSPACE parameter if BUFFERSPACE is supplied under CLUSTER.
756 (2F4)	4	DBFSZ	Address of information supplied through the BUFFERSPACE parameter if BUFFERSPACE is supplied under DATA.
760 (2F8)	4	CCINV	Address of information supplied through the CONTROLINTERVALSIZE parameter if CONTROLINTERVALSIZE is supplied under CLUSTER.
764 (2FC)	4	DCINV	Address of information supplied through the CONTROLINTERVALSIZE parameter if CONTROLINTERVALSIZE is supplied under DATA.
768 (300)	4	ICINV	Address of information supplied through the CONTROLINTERVALSIZE parameter if CONTROLINTERVALSIZE is supplied under INDEX.
772 (304)	4	ALREL	Address of information supplied through the RELATE parameter if RELATE is supplied under ALIAS.
776 (308)	4	GENEM	Address of this pointer itself if the EMPTY parameter is supplied under GENERATIONDATA GROUP.
780 (30C)	4	GENNE	Address of this pointer itself if the NOEMPTY parameter is supplied under GENERATIONDATA GROUP.
784 (310)	4	GENLM	Address of information supplied through the LIMIT parameter if LIMIT is supplied under GENERATIONDATA. GROUP.
788 (314)	4	GENSC	Address of this pointer itself if the SCRATCH parameter is supplied under GENERATIONDATA GROUP.
792 (318)	4	GENNS	Address of this pointer itself if the NOSCRATCH parameter is supplied under GENERATIONDATAGROUP.
796 (31C)	4	UETRY	Address of information supplied through the NAME parameter if NAME is supplied under USERCATALOG.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
800 (320)	4	UMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under USERCATALOG.
804 (324)	4	UCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under USERCATALOG.
808 (328)	4	UUPDT	Address of information supplied through the UPDATEPW parameter if UPDATEPW is supplied under USERCATALOG.
812 (32C)	4	UREAD	Address of information supplied through the READPW parameter if READPW is supplied under USERCATALOG.
816 (330)	4	UCODE	Address of information supplied through the CODE parameter if CODE is supplied under USERCATALOG.
820 (334)	4	UATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under USERCATALOG.
824 (338)	4	UAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under USERCATALOG.
828 (33C)	4	UEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under USERCATALOG.
832 (340)	4	USTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under USERCATALOG.
836 (344)	4	UTO	Address of information supplied through the TO parameter if TO is supplied under USERCATALOG.
840 (348)	4	UFOR	Address of information supplied through the FOR parameter if FOR is supplied under USERCATALOG.
844 (34C)	4	UOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under USERCATALOG.
848 (350)	4	UINDD	Address of information supplied through the FILE parameter if FILE is supplied under USERCATALOG.
852 (354)	4	UVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under USERCATALOG.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
856 (358)	4	UTRKS	Address of this pointer itself if the TRACKS parameter is supplied under USERCATALOG.
860 (35C)	4	UCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under USERCATALOG.
864 (360)	4	URCDS	Address of this pointer itself if the RECORDS parameter is supplied under USERCATALOG.
868 (364)	8	*	Reserved—contains zeros.
876 (36C)	4	UWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under USERCATALOG.
880 (370)	4	UNWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under USERCATALOG.
884 (374)	4	UBFSZ	Address of information supplied through the BUFFERSPACE parameter if BUFFERSPACE is supplied under USERCATALOG.
888 (378)	4	PMODL	Address of this pointer itself if the MODEL parameter is supplied under PAGESPACE.
892 (37C)	4	PENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under PAGESPACE.
896 (380)	4	PMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under PAGESPACE.
900 (384)	4	PMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under PAGESPACE.
904 (388)	4	PMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under PAGESPACE.
908 (38C)	4	PCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under PAGESPACE.
912 (390)	4	PUPDT	Address of information supplied through the UPDATEPW parameter if UPDATEPW is supplied under PAGESPACE.
916 (394)	4	PREAD	Address of information supplied through the READPW parameter if READPW is supplied under PAGESPACE.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
920 (398)	4	PCODE	Address of information supplied through the CODE parameter if CODE is supplied under PAGESPACE.
924 (39C)	4	PATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under PAGESPACE.
928 (3A0)	4	PAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under PAGESPACE.
932 (3A4)	4	PEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under PAGESPACE.
936 (3A8)	4	PSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under PAGESPACE.
940 (3AC)	4	PTO	Address of information supplied through the TO parameter if TO is supplied under PAGESPACE.
944 (3B0)	4	PFOR	Address of information supplied through the FOR parameter if FOR is supplied under PAGESPACE.
948 (3B4)	4	POWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under PAGESPACE.
952 (3B8)	12	*	Reserved—contains zeros.
964(3C4)	4	PERAS	Address of this pointer itself if ERASE parameter has been supplied under PAGESPACE.
968(3C8)	4	PNERS	Address of this pointer itself if the NOERASE parameter has been supplied under PAGESPACE.
972 (3CC)	4	PINDD	Address of information supplied through the FILE parameter if FILE is supplied under PAGESPACE.
976 (3D0)	4	PVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under PAGESPACE.
980 (3D4)	4	PSUBA	Address of this pointer itself if the SUBALLOCATION parameter is supplied under PAGESPACE.
984 (3D8)	4	PUNIQ	Address of this pointer itself if the UNIQUE parameter is supplied under PAGESPACE.
988 (3DC)	4	PTRKS	Address of this pointer itself if the TRACKS parameter is supplied under PAGESPACE.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
992 (3E0)	4	PCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under PAGESPACE.
996 (3E4)	4	PRCDS	Address of this pointer itself if the RECORDS parameter is supplied under PAGESPACE.
1000 (3E8)	8	*	Reserved—contains zeros. **(without VS2.03.807)**
1000 (3E8)	4	PSWAP	Address of this pointer itself if the SWAP parameter is supplied under CLUSTER. **(VS2.03.807)**
1004 (3EC)	4	PNSWP	Address of this pointer itself if the NOSWAP parameter is supplied under PAGESPACE. **(VS2.03.807)**
1008 (3F0)	4	SARSZ	Address of information supplied through the *average* subparameter of RECORDSIZE if RECORDSIZE is supplied under SPACE.
1012 (3F4)	4	SMRSZ	Address of information supplied through the *maximum* subparameter of RECORDSIZE if RECORDSIZE is supplied under SPACE.
1016 (3F8)	4	UENAM	Address of information supplied through the *entrypoint* subparameter of MODEL if MODEL is supplied under USERCATALOG.
1020 (3FC)	4	UMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under USERCATALOG.
1024 (400)	4	UMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under USERCATALOG.
1028 (404)	4	CEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under CLUSTER.
1032 (408)	4	CSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under CLUSTER.
1036(40C)	16	*	Reserved—contains zeros.
1052(41C)	4	CEEXT	Address of information supplied through the EXCEPTIONEXIT parameter if EXCEPTIONEXIT is supplied under CLUSTER.
1056(420)	4	DEEXT	Address of information supplied through the EXCEPTIONEXIT parameter if EXCEPTIONEXIT is supplied under DATA.
1060(424)	4	IEEXT	Address of information supplied through the EXCEPTIONEXIT parameter if EXCEPTIONEXIT is supplied under INDEX.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1068(42C)	4	CNUMD	Address of this pointer itself if NUMBERED is supplied under CLUSTER.
1072(430)	4	CRUS	Address of this pointer itself if REUSE is supplied under CLUSTER.
1076(434)	4	DRUS	Address of this pointer itself if REUSE is supplied under DATA.
1080(438)	4	IRUS	Address of this pointer itself if REUSE is supplied under INDEX.
1084(43C)	4	CNRUS	Address of this pointer itself if NOREUSE is supplied under CLUSTER.
1088(440)	4	DNRUS	Address of this pointer itself if NOREUSE is supplied under DATA.
1092(444)	4	INRUS	Address of this pointer itself if NOREUSE is supplied under INDEX.
1096(448)	4	CSPND	Address of this pointer itself if SPANNED is supplied under CLUSTER.
1100(44C)	4	DSPND	Address of this pointer itself if SPANNED is supplied under DATA.
1104(450)	4	CNSPD	Address of this pointer itself if NONSPANNED is supplied under CLUSTER.
1108(454)	4	DNSPD	Address of this pointer itself if NONSPANNED is supplied under DATA.
1112(458)	4	MRVBL	Address of this pointer itself if RECOVERABLE is supplied under MASTERCATALOG.
1116(45C)	4	URVBL	Address of this pointer itself if RECOVERABLE is supplied under USERCATALOG.
1120(460)	4	DRVBL	Address of this pointer itself if RECOVERABLE is supplied under DATA.
1124(464)	4	MNRVL	Address of this pointer itself if NOTRECOVERABLE is supplied under MASTERCATALOG.
1128(468)	4	UNRVL	Address of this pointer itself if NOTRECOVERABLE is supplied under USERCATALOG.
1132(46C)	4	DNRVL	Address of this pointer itself if NOTRECOVERABLE is supplied under DATA.
1136(470)	68	*	Reserved—contains zeros.
1204 (4B4)	4	MTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under MASTERCATALOG.
1208 (4B8)	4	MTKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under MASTERCATALOG.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1212 (4BC)	4	CTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under CLUSTER.
1216 (4C0)	4	CTKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under CLUSTER.
1220 (4C4)	4	STKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under SPACE.
1224 (4C8)	4	STKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under SPACE.
1228 (4CC)	4	UTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under USERCATALOG.
1232 (4D0)	4	UTKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under USERCATALOG.
1236 (4D4)	4	PTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under PAGESPACE.
1240 (4D8)	4	PTKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under PAGESPACE.
1244 (4DC)	4	MCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under MASTERCATALOG.
1248 (4E0)	4	MCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under MASTERCATALOG.
1252 (4E4)	4	CCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under CLUSTER.
1256 (4E8)	4	CCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under CLUSTER.
1260 (4EC)	4	UCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under USERCATALOG.
1264 (4F0)	4	UCLSC	Address of information supplied through the *secondary* subparameter

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			of CYLINDERS if CYLINDERS is supplied under USERCATALOG.
1268 (4F4)	4	PCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under PAGESPACE.
1272 (4F8)	4	PCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under PAGESPACE.
1276 (4FC)	4	SCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under SPACE.
1280 (500)	4	SCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under SPACE.
1284 (504)	4	MRCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under MASTERCATALOG.
1288 (508)	4	MRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under MASTERCATALOG.
1292 (50C)	4	CRCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under CLUSTER.
1296 (510)	4	CRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under CLUSTER.
1300 (514)	4	SRCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under SPACE.
1304 (518)	4	SRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under SPACE.
1308 (51C)	4	URCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under USERCATALOG.
1312 (520)	4	URCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under USERCATALOG.
1316 (524)	4	PCRPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under PAGESPACE.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1320 (528)	4	PRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under PAGESPACE.
1324 (52C)	4	DCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under DATA.
1328 (530)	4	DCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under DATA.
1332 (534)	4	DRCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under DATA.
1336 (538)	4	DRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under DATA.
1340 (53C)	4	ICLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under INDEX.
1344 (540)	4	ICLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under INDEX.
1348 (544)	4	IRCPR	Address of information supplied through the *primary* subparameter of RECORDS if CYLINDERS is supplied under INDEX.
1352 (548)	4	IRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if CYLINDERS is supplied under INDEX.
1356 (54C)	4	GETRY	Address of information supplied through the NAME parameter if NAME is supplied under ALTERNATEINDEX.
1360 (550)	4	GMODL	Address of this pointer itself if MODEL is supplied under ALTERNATEINDEX.
1364 (554)	4	GENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under ALTERNATEINDEX.
1368 (558)	4	GMDCT	Address of information supplied through the *CATNAME/password* subparameter of MODEL if MODEL is supplied under ALTERNATEINDEX.
1372 (55C)	4	GMDNM	Address of information supplied through the *dname* subparameter of

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			MODEL if MODEL is supplied under ALTERNATEINDEX.
1376 (560)	4	GMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under ALTERNATEINDEX.
1380 (564)	4	GCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under ALTERNATEINDEX.
1384 (568)	4	GUPDT	Address of information supplied through the UPDATEPW parameter if UPDATEPW is supplied under ALTERNATEINDEX.
1388 (56C)	4	GREAD	Address of information supplied through the READPW parameter if READPW is supplied under ALTERNATEINDEX.
1392 (570)	4	GCODE	Address of information supplied through the CODE parameter if CODE is supplied under ALTERNATEINDEX.
1396 (574)	4	GATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under ALTERNATEINDEX.
1400 (578)	4	GAUTH	Address of this pointer itself if the AUTHORIZATION parameter is supplied under ALTERNATEINDEX.
1404 (57C)	4	GEPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if the AUTHORIZATION parameter is supplied under ALTERNATEINDEX.
1408 (580)	4	GSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under ALTERNATEINDEX.
1412 (584)	4	GTO	Address of information supplied through the TO parameter if TO is supplied under ALTERNATEINDEX.
1416 (588)	4	GFOR	Address of information supplied through the FOR parameter if FOR is supplied under ALTERNATEINDEX.
1420 (58C)	4	GOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under ALTERNATEINDEX.
1424 (590)	4	GSHAR	Address of this pointer itself if the SHAREOPTIONS parameter is supplied under ALTERNATEINDEX.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1428 (594)	4	GSHR1	Address of information supplied through the *crosspartition* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under ALTERNATEINDEX.
1432 (598)	4	GSHR2	Address of information supplied through the *crosssystem* subparameter of SHAREOPTIONS if SHAREOPTIONS is supplied under ALTERNATEINDEX.
1436 (59C)	4	GERAS	Address of this pointer itself if the ERASE parameter is supplied under ALTERNATEINDEX.
1440 (5A0)	4	GNERS	Address of this pointer itself if the NOERASE parameter is supplied under ALTERNATEINDEX.
1444 (5A4)	4	GKEY	Address of this pointer itself if the KEYS parameter is supplied under ALTERNATEINDEX.
1448 (5A8)	4	GKYLN	Address of information supplied through the *length* subparameter of KEYS if KEYS is supplied under ALTERNATEINDEX.
1452 (5AC)	4	GKYPS	Address of information supplied through the *offset* subparameter of KEYS if KEYS is supplied under ALTERNATEINDEX.
1456 (5B0)	4	GREPL	Address of this pointer itself if the REPLICATE parameter is supplied under ALTERNATEINDEX.
1460 (5B4)	4	GNREP	Address of this pointer itself if the NOREPLICATE parameter is supplied under ALTERNATEINDEX.
1464 (5B8)	4	GIMBD	Address of this pointer itself if the IMBED parameter is supplied under ALTERNATEINDEX.
1468 (5BC)	4	GNIBD	Address of this pointer itself if the NOIMBED parameter is supplied under ALTERNATEINDEX.
1472 (5C0)	4	GINDD	Address of information supplied through the FILE parameter if FILE is supplied under ALTERNATEINDEX.
1476 (5C4)	4	GVSER	Address of information supplied through the VOLUMES parameter if VOLUMES is supplied under ALTERNATEINDEX.
1480 (5C8)	4	GRANG	Address of a count of subparameters supplied through the KEYRANGES parameter if KEYRANGES is supplied under ALTERNATEINDEX.
1484 (5CC)	4	GRGLO	Address of information supplied through the *lowkey* subparameter of KEYRANGES if KEYRANGES is supplied under ALTERNATEINDEX.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1488 (5D0)	4	GRGHI	Address of information supplied through the *highkey* subparameter of KEYRANGES if KEYRANGES is supplied under ALTERNATEINDEX.
1492 (5D4)	4	GORDR	Address of this pointer itself if the ORDERED parameter is supplied under ALTERNATEINDEX.
1496 (5D8)	4	GUORD	Address of this pointer itself if the UNORDERED parameter is supplied under ALTERNATEINDEX.
1500 (5DC)	4	GSUBA	Address of this pointer itself if the SUBALLOCATION parameter is supplied under ALTERNATEINDEX.
1504 (5E0)	4	GUNIQ	Address of this pointer itself if the UNIQUE parameter is supplied under ALTERNATEINDEX.
1508 (5E4)	4	GTRKS	Address of this pointer itself if the TRACKS parameter is supplied under ALTERNATEINDEX.
1512 (5E8)	4	GTKPR	Address of information supplied through the *primary* subparameter of TRACKS if TRACKS is supplied under ALTERNATEINDEX.
1516 (5EC)	4	GTKSC	Address of information supplied through the *secondary* subparameter of TRACKS if TRACKS is supplied under ALTERNATEINDEX.
1520 (5F0)	4	GCYLD	Address of this pointer itself if the CYLINDERS parameter is supplied under ALTERNATEINDEX.
1524 (5F4)	4	GCLPR	Address of information supplied through the *primary* subparameter of CYLINDERS if CYLINDERS is supplied under ALTERNATEINDEX.
1528 (5F8)	4	GCLSC	Address of information supplied through the *secondary* subparameter of CYLINDERS if CYLINDERS is supplied under ALTERNATEINDEX.
1532 (5FC)	4	GRCDS	Address of this pointer itself if the RECORDS parameter is supplied under ALTERNATEINDEX.
1536 (600)	4	GRCPR	Address of information supplied through the *primary* subparameter of RECORDS if RECORDS is supplied under ALTERNATEINDEX.
1540 (604)	4	GRCSC	Address of information supplied through the *secondary* subparameter of RECORDS if RECORDS is supplied under ALTERNATEINDEX.
1544 (608)	4	GRSIZ	Address of this pointer itself if the RECORDSIZE parameter is supplied under ALTERNATEINDEX.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1548 (60C)	4	GARSZ	Address of information supplied through the *average* subparameter of RECORDSIZE if RECORDSIZE is supplied under ALTERNATEINDEX.
1552 (610)	4	GMRSZ	Address of information supplied through the *maximum* subparameter of RECORDSIZE if RECORDSIZE is supplied under ALTERNATEINDEX.
1556 (614)	4	GWCK	Address of this pointer itself if the WRITECHECK parameter is supplied under ALTERNATEINDEX.
1560 (618)	4	GNWCK	Address of this pointer itself if the NOWRITECHECK parameter is supplied under ALTERNATEINDEX.
1564 (61C)	4	GSPED	Address of this pointer itself if the SPEED parameter is supplied under ALTERNATEINDEX.
1568 (620)	4	GRECV	Address of this pointer itself if the RECOVERY parameter is supplied under ALTERNATEINDEX.
1572 (624)	4	GFSPC	Address of this pointer itself if the FREESPACE parameter is supplied under ALTERNATEINDEX.
1576 (628)	4	GCIFS	Address of information supplied through the *cipercent* subparameter of FREESPACE if FREESPACE is supplied under ALTERNATEINDEX.
1580 (62C)	4	GCAFS	Address of information supplied through the *capercent* subparameter of FREESPACE if FREESPACE is supplied under ALTERNATEINDEX.
1584 (630)	4	GBFSZ	Address of information supplied through the BUFFERSPACE parameter if BUFFERSPACE is supplied under ALTERNATEINDEX.
1588 (634)	4	GCINV	Address of information supplied through the CONTROLINTERVALSIZE parameter if CONTROLINTERVALSIZE is supplied under ALTERNATEINDEX.
1592 (638)	4	GREL	Address of information supplied through the RELATE parameter if RELATE is supplied under ALTERNATEINDEX.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1596 (63C)	4	GEEXT	Address of information supplied through the EXCEPTIONEXIT parameter if EXCEPTIONEXIT is supplied under ALTERNATEINDEX.
1600 (640)	4	GRUS	Address of information supplied through the REUSE parameter if REUSE is supplied under ALTERNATEINDEX.
1604 (644)	4	GNRUS	Address of information supplied through the NOREUSE parameter if NOREUSE is supplied under ALTERNATEINDEX.
1608 (648)	4	GUNQK	Address of information supplied through the UNIQUEKEY parameter if UNIQUEKEY is supplied under ALTERNATEINDEX.
1612 (64C)	4	GNUQK	Address of information supplied through the NONUNIQUEKEY parameter if NONUNIQUEKEY is supplied under ALTERNATEINDEX.
1616 (650)	4	DUNQK	Address of information supplied through the UUNIQUEKEY parameter if UNIQUEKEY is supplied under DATA.
1620 (654)	4	DNUQK	Address of information supplied through the NONUNIQUEKEY parameter if NONUNIQUEKEY is supplied under DATA.
1624 (658)	4	GUPG	Address of information supplied through the UPGRADE parameter if UPGRADE is supplied under ALTERNATEINDEX.
1628 (65C)	4	GNUPG	Address of information supplied through the NOUPGRADE parameter if NOUPGRADE is supplied under ALTERNATEINDEX.
1632 (660)	12	*	Reserved—contains zeros.
1644 (66C)	4	RETRY	Address of information supplied through the NAME parameter if NAME is supplied under PATH.
1648 (670)	4	RMODL	Address of this pointer itself if the MODEL parameter is supplied under PATH.
1652 (674)	4	RENAM	Address of information supplied through the *entryname/password* subparameter of MODEL if MODEL is supplied under PATH.
1656 (678)	4	RMDCT	Address of information supplied through the *catname/password* subparameter of MODEL if MODEL is supplied under PATH.
1660 (67C)	4	RMDNM	Address of information supplied through the *dname* subparameter of MODEL if MODEL is supplied under PATH.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1664 (680)	4	RMSTR	Address of information supplied through the MASTERPW parameter if MASTERPW is supplied under PATH.
1668 (684)	4	RCINT	Address of information supplied through the CONTROLPW parameter if CONTROLPW is supplied under PATH.
1672 (688)	4	RUPDT	Address of information supplied through the UPDATEPW parameter if UPDATEPW is supplied under PATH.
1676 (68C)	4	RREAD	Address of information supplied through the READPW parameter if READPW is supplied under PATH.
1680 (690)	4	RCODE	Address of information supplied through the CODE parameter if CODE is supplied under PATH.
1684 (694)	4	RATTP	Address of information supplied through the ATTEMPTS parameter if ATTEMPTS is supplied under PATH.
1688 (698)	4	RAUTH	Address of information supplied through the AUTHORIZATION parameter if AUTHORIZATION is supplied under PATH.
1692 (69C)	4	REPNM	Address of information supplied through the *entrypoint* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under PATH.
1696 (6A0)	4	RSTRG	Address of information supplied through the *string* subparameter of AUTHORIZATION if AUTHORIZATION is supplied under PATH.
1700 (6A4)	4	RTO	Address of information supplied through the TO parameter if TO is supplied under PATH.
1704 (6A8)	4	RFOR	Address of information supplied through the FOR parameter if FOR is supplied under PATH.
1708 (6AC)	4	ROWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under PATH.
1712 (6B0)	4	RINDD	Address of information supplied through the FILE parameter if FILE is supplied under PATH.
1716 (6B4)	4	RUPD	Address of information supplied through the UPDATE parameter if UPDATE is supplied under PATH.
1720 (6B8)	4	RNUPD	Address of information supplied through the NOUPDATE parameter if NOUPDATE is supplied under PATH.
1724 (6BC)	4	RPENT	Address of information supplied through the PATHENTRY parameter

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			if PATHENTRY is supplied under PATH.
1728 (6C0)	4	CSTAG	Address of information supplied through the STAGE parameter if STAGE is supplied under CLUSTER.
1732 (6C8)	4	GSTAG	Address of information supplied through the STAGE parameter if STAGE is supplied under ALTERNATEINDEX.
1736 (6C8)	4	DSTAG	Address of information supplied through the STAGE parameter if STAGE is supplied under DATA.
1740 (6CC)	4	ISTAG	Address of information supplied through the STAGE parameter if STAGE is supplied under INDEX.
1744 (6D0)	4	CBIND	Address of information supplied through the BIND parameter if BIND is supplied under CLUSTER.
1748 (6D4)	4	GBIND	Address of information supplied through the BIND parameter if BIND is supplied under ALTERNATEINDEX.
1752 (6D8)	4	DBIND	Address of information supplied through the BIND parameter if BIND is supplied under DATA.
1756(6DC)	4	IBIND	Address of information supplied through the BIND parameter if BIND is supplied under INDEX.
1760(6E0)	4	CCYLF	Address of information supplied through the CYLINDERFAULT parameter if CYLINDERFAULT is supplied under CLUSTER.
1764(6E4)	4	GCYLF	Address of information supplied through the CYLINDERFAULT parameter if CYLINDERFAULT is supplied under ALTERNATEINDEX.
1768(6E8)	4	DCYLF	Address of information supplied through the CYLINDERFAULT parameter if CYLINDERFAULT is supplied under DATA.
1772(6EC)	4	ICYLF	Address of information supplied through the CYLINDERFAULT parameter if CYLINDERFAULT is supplied under INDEX.
1776(6F0)	4	MNSTW	Address of information supplied through the NODESTAGEWAIT parameter if NODESTAGEWAIT is supplied under MASTERCATALOG.
1780(6F4)	4	UNSTW	Address of information supplied through the NODESTAGEWAIT parameter if NODESTAGEWAIT is supplied under USERCATALOG.
1784(6F8)	4	CNSTW	Address of information supplied through the NODESTAGEWAIT

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			parameter if NODESTAGEWAIT is supplied under CLUSTER.
1788(6FC)	4	GNSTW	Address of information supplied through the NODESTAGEWAIT parameter if NODESTAGEWAIT is supplied under ALTERNATEINDEX.
1792(700)	4	DNSTW	Address of information supplied through the NODESTAGEWAIT parameter if NODESTAGEWAIT is supplied under DATA.
1796(704)	4	INSTW	Address of information supplied through the NODESTAGEWAIT parameter if NODESTAGEWAIT is supplied under INDEX.
1800(708)	4	MSTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under MASTERCATALOG.
1804(70C)	4	USTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under USERCATALOG.
1808(710)	4	CSTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under CLUSTER.
1812(714)	4	GSTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under ALTERNATEINDEX.
1816(718)	4	DSTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under DATA.
1820(71C)	4	ISTGW	Address of information supplied through the DESTAGEWAIT parameter if DESTAGEWAIT is supplied under INDEX.
1824(720)	4	ATO	Address of information supplied through the TO parameter if TO is supplied under NONVSAM.
1828(724)	4	BTO	Address of information supplied through the TO parameter if TO is supplied under GENERATION-DATAGROUP.
1832(728)	4	AFOR	Address of information supplied through the FOR parameter if FOR is supplied under NONVSAM.
1836(72C)	4	BFOR	Address of information supplied through the FOR parameter if FOR is supplied under GENERATION-DATAGROUP.

Define FDT Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
1840(730)	4	AOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under NONVSAM.
1844(734)	4	BOWNR	Address of information supplied through the OWNER parameter if OWNER is supplied under GENERATION- DATA GROUP.

DELETE FDT

Offset	*Content*	
0 (0)	D E L E	T E ƀ ƀ
8 (8)	†*entryname/password*	†CATALOG
16 (10)	†*catname/password*	†*dname*
24 (18)	†FILE	†PURGE
32 (20)	†NOPURGE	†ERASE
40 (28)	†NOERASE	0
48 (30)	†CLUSTER	†SPACE
56 (38)	†USERCATALOG	†MASTERCATALOG
64 (40)	†NONVSAM	†SCRATCH
72 (48)	†NOSCRATCH	†PAGESPACE
80 (50)	†GENERATIONDATAGROUP	†ALIAS
88 (58)	†AIX	†PATH
96 (60)	†FRC	†NFRC

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—DELETEƀƀ.
8 (8)	4	NTRY	Address of information supplied through the *entryname/password* parameter.
12 (C)	4	CATLG	Address of this pointer itself if the CATALOG parameter is supplied.
16 (10)	4	CAT	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
20 (14)	4	CATDD	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
24 (18)	4	INDD	Address of information supplied through the FILE parameter.
28 (1C)	4	PURGE	Address of this pointer itself if the PURGE parameter is supplied or defaulted.
32 (20)	4	NOPUR	Address of this pointer itself if the NOPURGE parameter is supplied.
36 (24)	4	ERASE	Address of this pointer itself if the ERASE parameter is supplied.
40 (28)	4	NOERA	Address of this pointer itself if the NOERASE parameter is supplied.
44 (2C)	4	*	Reserved—contains zero.
48 (30)	4	CLUST	Address of this pointer itself if the CLUSTER parameter is supplied.
52 (34)	4	SPACE	Address of this pointer itself if the SPACE parameter is supplied.
56 (38)	4	UCAT	Address of this pointer itself if the USERCATALOG parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
60 (3C)	4	MCAT	Address of this pointer itself if the MASTERCATALOG parameter is supplied.
64 (40)	4	ALIEN	Address of this pointer itself if the NONVSAM parameter is supplied.
68 (44)	4	SCR	Address of this pointer itself if the SCRATCH parameter is supplied.
72 (48)	4	NSCR	Address of this pointer itself if the NOSCRATCH parameter is supplied.
76 (4C)	4	PGSPC	Address of this pointer itself if the PAGESPACE parameter is supplied.
80 (50)	4	GDG	Address of this pointer itself if the GENERATIONDATA- GROUP parameter is supplied.
84 (54)	4	ALIAS	Address of this pointer itself if the ALIAS parameter is supplied.
88 (58)	4	AIX	Address of this pointer itself if the ALTERNATE INDEX parameter has been supplied.
92 (5C)	4	PATH	Address of this pointer itself if the PATH parameter has been supplied.
96 (60)	4	FRC	Address of this pointer itself if the FORCE parameter has been supplied.
100 (64)	4	NFRC	Address of this pointer itself if the NOFORCE parameter has been supplied.

EXPORT FDT

Offset	*Content*	
0 (0)	**E X P O**	**R T b b**
8 (8)	†*entryname/password*	†INFILE
16 (10)	†OUTFILE	†*dname*
24 (18)	0	†TEMPORARY
32 (20)	†PERMANENT	†INHIBITSOURCE
40 (28)	†INHIBITTARGET	†ERASE
48 (30)	†NOERASE	†PURGE
56 (38)	†NOPURGE	†DISCONNECT
64 (40)	†NOINHIBITSOURCE	†NOINHIBITTARGET
72 (48)	0	†OUTDATASET
80 (50)	0	0
88(58)	0	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—EXPORTbb.
8 (8)	4	ENT	Address of information supplied through the *entryname/password* parameter.
12 (C)	4	INDD	Address of information supplied through the INFILE parameter.
16 (10)	4	OUT	Address of this pointer itself if the OUTFILE parameter is supplied.
20 (14)	4	OUTDD	Address of information supplied through the *dname* subparameter of the OUTFILE parameter.
24 (18)	4	ENVIR	Reserved in OS/VS—contains zero.
28 (1C)	4	TEMP	Address of this pointer itself if the TEMPORARY parameter is supplied.
32 (20)	4	PERM	Address of this pointer itself if the PERMANENT parameter is supplied.
36 (24)	4	INHBS	Address of this pointer itself if the INHIBITSOURCE parameter is supplied.
40 (28)	4	INHBT	Address of this pointer itself if the INHIBITTARGET parameter is supplied.
44 (2C)	4	ERASE	Address of this pointer itself if the ERASE parameter is supplied.
48 (30)	4	NOERS	Address of this pointer itself if the NOERASE parameter is supplied.
52 (34)	4	PURGE	Address of this pointer itself if the PURGE parameter is supplied.
56 (38)	4	NPRG	Address of this pointer itself if the NOPURGE parameter is supplied.
60 (3C)	4	DISCT	Address of this pointer itself if the DISCONNECT parameter is supplied.
64 (40)	4	NINHS	Address of this pointer itself if the NOINHIBITSOURCE parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
68 (44)	4	NINHT	Address of this pointer itself if the NOINHIBITTARGET parameter is supplied.
72 (48)	4	*	Reserved—contains zero.
76 (4C)	4	OUTDS	Address of information supplied through the OUTDATASET parameter.
80 (50)	4	*	Reserved—contains zero.
84 (54)	4	PDEV	Reserved in OS/VS—contains zero.
88 (58)	4	BLKSZ	Reserved in OS/VS.

EXPORTRA FDT

Offset	*Content*	
0 (0)	**E X P O**	**R T R A**
8 (8)	↑FORCE	↑NOFORCE
16 (10)	↑OUTFILE	↑*CRA count*
24 (18)	↑*dname*	↑ALL
32 (20)	↑NONE	↑ENTRIES
40 (28)	↑INFILE	↑MASTERPW
48 (30)	0	0
56 (38)	↑*dname*	↑*entryname*
64 (40)	↑*dname*	0

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb—EXPORTRA
8 (8)	4	FRC	Address of this pointer itself if the FORCE parameter has been supplied.
12 (C)	4	NFRC	Address of this pointer itself if the NOFORCE parameter has been supplied.
16 (10)	4	OUT	Address of this pointer itself if the OUTFILE parameter has been supplied.
20 (14)	4	CRACNT	Count of the number of catalog recovery areas (CRAs) that were provided in the EXPORTRA command.
24 (18)	4	CRADDPTR	Address of an array of pointers. Each pointer points at the *dname* for the CRA it relates to in the order that they appear in the EXPORTRA command.
28 (1C)	4	ALLNTPTR	Address of an array of pointers. Each pointer points to itself if ALL was specified for the related CRA.
32 (20)	4	NONEPTR	Address of an array of pointers. Each pointer points to itself if NONE was specified for the related CRA or was defaulted for the CRA.
36 (24)	4	ENTREPTR	Address of an array of counts. Each count indicates the number of entries

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			if ENTRIES was specified for the related CRA. Zero indicates that ENTRIES was not specified.
40 (28)	4	IFILEPTR	Address of an array of pointers. Each pointer points to the *dname* subparameter of the INFILE subparameter, i.e., the *dname* to be used for a CRA if ALL was specified for that CRA.
44 (2C)	4	MRPW	Address of information supplied through the *password* subparameter of the MASTERPW parameter.
48 (30)	4	ENVIR	Reserved in OS/VS—contains zeros.
52 (34)	4	PDEV	Reserved in OS/VS—contains zero.
56 (38)	4	OUTDD	Address of information supplied through the *dname* subparameter of the OUTFILE parameter.
60 (3C)	4	ENTNMPTR	Address of an array of pointers. Each pointer points to the *entryname* subparameter of the ENTRIES subparameter, i.e., all of the entry names in each CRA specified.
64 (40)	4	ENTDNPTR	Address of an array of pointers. Each pointer points to the *dname* subparameter of the ENTRIES subparameter to be used to export the associated entry name in ENTNMPTR.
68 (44)	4	BLKSZ	Reserved in OS/VS—contains zero.

IMPORT FDT

Offset	*Content*	
0 (0)	I M P O	R T b b
8 (8)	†INFILE	†OUTFILE
16 (10)	†OBJECTS	†*objectname*
24 (18)	†NEWNAME	†FILE
32 (20)	†VOLUMES	†KEYRANGES
40 (28)	†DEVICETYPES	†ORDERED
48 (30)	†UNORDERED	†*lowkey*
56 (38)	†*highkey*	†CONNECT
64 (40)	†*dname*	0
72 (48)	†PURGE	†NOPURGE
80 (50)	†ERASE	†NOERASE
96 (60)	0	†INDATASET
104 (68)	†OUTDATASET	0
136 (88)	†CATALOG	† IMPTY (VS2.03.807)
144 (9C)	† SRAC (VS2.03.807)	†NSRAC (VS2.03.807)

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—IMPORTbb.
8 (8)	4	IN	Address of this pointer itself if the INFILE parameter is supplied.
12 (C)	4	OUTDD	Address of information supplied through the OUTFILE parameter.
16 (10)	4	OBJTS	Address of a count of subparameters supplied through the OBJECTS parameter.
20 (14)	4	OBJNMPTR	Address of information supplied through the *objectname* subparameter of the OBJECTS parameter.
24 (18)	4	NEWNMPTR	Address of information supplied through the NEWNAME subparameter of the OBJECTS parameter.
28 (1C)	4	OBJFLPTR	Address of information supplied through the FILE subparameter of the OBJECTS parameter.
32 (20)	4	LISTVPTR	Address of information supplied through the VOLUMES subparameter of the OBJECTS parameter.
36 (24)	4	RANGEPTR	Address of a count of *lowkey highkey* pairs supplied through the KEYRANGES subparameter of the OBJECTS parameter.
40 (28)	4	DEVTPTR	Address of information supplied through the DEVICETYPES subparameter of the OBJECTS parameter.
44 (2C)	4	ORDPTR	Address of information supplied through the ORDERED subparameter of the OBJECTS parameter.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
48 (30)	4	UNORDPTR	Address of information supplied through the UNORDERED subparameter of the OBJECTS parameter.
52 (34)	4	LOWKYPTR	Address of information supplied through the *lowkey* subparameter of the KEYRANGES parameter.
56 (38)	4	HIKEYPTR	Address of information supplied through the *highkey* subparameter of the KEYRANGES parameter.
60 (3C)	4	CON	Address of this pointer itself if the CONNECT parameter is supplied.
64 (40)	4	INDD	Address of information supplied through the *dname* subparameter of the INFILE parameter.
68 (44)	4	ENV	Reserved in OS/VS—contains zero.
72 (48)	4	PRG	Address of this pointer itself if the PURGE parameter is supplied.
76 (4C)	4	NPRG	Address of this pointer itself if the NOPURGE parameter is supplied.
80 (50)	4	ERAS	Address of this pointer itself if the ERASE parameter is supplied.
84 (54)	4	NERAS	Address of this pointer itself if the NOERASE parameter is supplied.
88 (58)	4	BLKSZ	Reserved in OS/VS—contains zero.
92 (5C)	4	PDEV	Reserved in OS/VS—contains zero.
96 (60)	4	RCSZE	Reserved in OS/VS—contains zero.
100 (64)	4	INDS	Address of information supplied through the INDATASET parameter.
104 (68)	4	OUTDS	Address of information supplied through the OUTDATASET parameter.
108 (6C)	28	*	Reserved—contains zeros.
136 (88)	4	CAT	Address of inforamtion supplied through the CATALOG parameter.
140 (8C) **(VS2.03.807)**	4	IMPTY	**Address of this pointer itself if the INTOEMPTY parameter is supplied.**
144 (90) **(VS2.03.807)**	4	SRAC	**Address of this pointer itself if the SAVRAC parameter is specified.**
148 (94) **(VS2.03.807)**	4	NSRAC	**Address of this pointer itself if the NOSAVRAC parameter is specified.**

IMPORTRA FDT

Offset	*Content*	
0 (0)	I M P 0	R T R A
8 (8)	†INFILE	†OUTFILE
16 (10)	†OBJECTS	†*object name*
24 (18)	0	0
32 (20)	†VOLUMES	0
40 (28)	†DEVICETYPE	0
48 (30)	0	0
56 (38)	0	0
64 (40)	†*dname*	0
72 (48)	0	0
80 (50)	0	0
88 (58)	0	0
96 (60)	0	†INDATASET
104 (68)	0	0
112 (70)	0	0
120 (78)	0	0
128 (80)	0	0
136 (88)	†CATALOG	†SRAC (VS2.03.807)
144 (9C)	† NSRAC (VS2.03.807)	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb—IMPORTRA.
8 (8)	4	IN	Address of this pointer itself if the INFILE parameter has been supplied.
12 (C)	4	OUTDD	Address of information supplied through the OUTFILE parameter.
16 (10)	4	OBJTS	Address of the count of objects supplied through the OBJECTS parameter.
20 (14)	4	OBJNMPTR	Address of information supplied through the *name* subparameter of the OBJECTS parameter.
24 (18)	8	*	Reserved—contains zeros.
32 (20)	4	LISTVPTR	Address of information supplied through the VOLUMES subparameter of the OBJECTS parameter.
36 (24)	4	*	Reserved—contains zeros.
40 (28)	4	DEVTPTR	Address of information supplied through the DEVICETYPE subparameter of the OBJECTS parameter.
44 (2C)	20	*	Reserved—contain zeros.
64 (40)	4	INDD	Address of information supplied through the *dname* subparameter of the INFILE parameter.
68 (44)	4	ENV	Reserved in OS/VS. Contains zeros.
72 (48)	16	*	Reserved—contains zeros.
88 (58)	4	BLKSZ	Reserved in OS/VS. Contains zeros.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
92 (5C)	4	PDEV	Reserved in OS/VS. Contains zeros.
96 (60)	4	*	Reserved—contains zeros.
100 (64)	4	INDS	Address of the input data set name.
104 (68)	32	*	Reserved—contains zeros.
136 (88)	4	CAT	Address of information supplied through the CATALOG parameter.
140 (8C) **(VS2.03.807)**	4	SRAC	Address of this location itself if the SAVRAC parameter is specified.
144 (90) **(VS2.03.807)**	4	NSRAC	Address of this location itself if the NOSAVRAC parameter is specified.

LISTCAT FDT

Offset	*Content*	
0 (0)	L I S T	C A T b
8 (8)	↑CATALOG	↑OUTFILE
16 (10)	↑ENTRIES	0
24 (18)	↑CLUSTER	↑DATA
32 (20)	↑INDEX	↑SPACE
40 (28)	↑NONVSAM	↑USERCATALOG
48 (30)	↑*catname/password*	↑*dname*
56 (38)	0	↑NAME
64 (40)	↑ALL	↑VOLUME
72 (48)	↑ALLOCATION	↑ALIAS
80 (50)	↑GENERATIONDATAGROUP	↑PAGESPACE
88 (58)	↑LEVEL	↑ALTERNATEINDEX
96 (60)	↑PATH	↑NOTUSABLE
104 (68)	↑CREATION	↑EXPIRATION
112 (70)	↑HISTORY	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—LISTCATb.
8 (8)	4	CAT	Address of this pointer itself if the CATALOG parameter is supplied.
12 (C)	4	OUTDD	Address of information supplied through the OUTFILE parameter.
16 (10)	4	ENT	Address of information supplied through the ENTRIES parameter.
20 (14)	4	*	Reserved—contains zero.
24 (18)	4	CLUST	Address of this pointer itself if the CLUSTER parameter is supplied.
28 (1C)	4	DATUM	Address of this pointer itself if the DATA parameter is supplied.
32 (20)	4	INDEX	Address of this pointer itself if the INDEX parameter is supplied.
36 (24)	4	SPACE	Address of this pointer itself if the SPACE parameter is supplied.
40 (28)	4	ALIEN	Address of this pointer itself if the NONVSAM parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
44 (2C)	4	UCAT	Address of this pointer itself if the USERCATALOG parameter is supplied.
48 (30)	4	CATNM	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
52 (34)	4	CATDD	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
56 (38)	4	*	Reserved—contains zero.
60 (3C)	4	NAME	Address of this pointer itself if the NAME parameter is supplied.
64 (40)	4	FALL	Address of this pointer itself if the ALL parameter is supplied.
68 (44)	4	VOL	Address of this pointer itself if the VOLUME parameter is supplied.
72 (48)	4	ALLOC	Address of this pointer itself if the ALLOCATION parameter is supplied.
76 (4C)	4	ALIAS	Address of this pointer itself if the ALIAS parameter is supplied.
80 (50)	4	GDG	Address of this pointer itself if the GENERATIONDATA- GROUP parameter is supplied.
84 (54)	4	PGSPC	Address of this pointer itself if the PAGESPACE parameter is supplied.
88 (58)	4	LVL	Address of information supplied through the LEVEL parameter.
92 (5C)	4	AIX	Address of this pointer itself if the ALTERNATEINDEX parameter has been supplied.
96 (60)	4	PATH	Address of this pointer itself if the PATH parameter has been supplied.
100 (64)	4	NUSE	Address of this pointer itself if the NOTUSABLE parameter has been supplied.
104(68)	4	CREAT	Address of information supplied through the CREATION parameter.
108(6C)	4	EXPIR	Address of information supplied through the EXPIRATION parameter.
112(70)	4	HIST	Address of this pointer itself if the HISTORY parameter is supplied.

LISTCRA FDT

Offset	*Content*	
0 (0)	L I S T	C R A ƀ
8 (8)	†INFILE	†COMPARE
16 (10)	†NOCOMPARE	†DUMP
24 (18)	†NAME	†CATALOG
32 (20)	†*catname/password*	†*dname*
40 (28)	†MASTERPW	†SEQUENTIALDUMP

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—LISTCRAƀ.
8 (8)	4	IFILE	Address of information supplied through the *dname* subparameter of the INFILE parameter.
12 (C)	4	CMPR	Address of this pointer itself if the COMPARE parameter has been supplied.
16 (10)	4	NCMPR	Address of this pointer itself if the NOCOMPARE parameter has been supplied.
20 (14)	4	DUMP	Address of this pointer itself if the DUMP parameter has been supplied.
24 (18)	4	NAME	Address of this pointer itself if the NAME parameter has been supplied.
28 (1C)	4	CAT	Address of this pointer itself if the CATALOG parameter has been supplied.
32 (20)	4	CATNM	Address of information supplied through the *catname/password* subparameter of the CATALOG parameter.
36 (24)	4	CATDN	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
40 (28)	4	MRPW	Address of information supplied through the *password* subparameter of the MASTERPW parameter.
44 (32)	4	SDUMP	Address of this pointer itself if the SEQUENTIALDUMP parameter has been supplied.

PARM FDT

Offset	*Content*	
0 (0)	P A R M	b b b b
8 (8)	†TEST	†OFF
16 (10)	†TRACE	†AREAS
24 (18)	†FULL	†*dumpid*
32 (20)	†*count1*	†*count2*
40 (28)	†GRAPHICS	†CHAIN
48 (30)	†TABLE	†MARGINS
56 (38)	†*leftmargin*	†*rightmargin*
64 (40)	†AN	†HN
72 (48)	†PN	†QN
80 (50)	†RN	†SN
88 (58)	†TN	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—PARMbbbb.
8 (8)	4	TEST	Address of this pointer itself if the TEST parameter is supplied.
12 (C)	4	TOFF	Address of this pointer itself if the OFF parameter is supplied.
16 (10)	4	TRACE	Address of this pointer itself if the TRACE parameter is supplied.
20 (14)	4	AREA	Address of information supplied through the AREAS parameter.
24 (18)	4	FULL	Address of a count of subparameters supplied through the FULL parameter.
28 (1C)	4	FIDPTR	Address of information supplied through the *dumpid* subparameter of the FULL parameter.
32 (20)	4	BEGINPTR	Address of information supplied through the *count1* subparameter of the FULL parameter.
36 (24)	4	COUNTPTR	Address of information supplied through the *count2* subparameter of the FULL parameter.
40 (28)	4	GRAPH	Address of this pointer itself if the GRAPHICS parameter is supplied.
44 (2C)	4	CHAIN	Address of information supplied through the CHAIN parameter.
48 (30)	4	TABLE	Address of information supplied through the TABLE parameter.
52 (34)	4	MARG	Address of this pointer itself if the MARGINS parameter is supplied.
56 (38)	4	LMARG	Address of information supplied through the *leftmargin* subparameter of the MARGINS parameter.
60 (3C)	4	RMARG	Address of information supplied through the *rightmargin* subparameter of the MARGINS parameter.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
64 (40)	4	CHNAN	Address of this pointer itself if the AN subparameter of the CHAIN parameter is supplied.
68 (44)	4	CHNHN	Address of this pointer itself if the HN subparameter of the CHAIN parameter is supplied.
72 (48)	4	CHNPN	Address of this pointer itself if the PN subparameter of the CHAIN parameter is supplied.
76 (4C)	4	CHNQN	Address of this pointer itself if the QN subparameter of the CHAIN parameter is supplied.
80 (50)	4	CHNRN	Address of this pointer itself if the RN subparameter of the CHAIN parameter is supplied.
84 (54)	4	CHNSN	Address of this pointer itself if the SN subparameter of the CHAIN parameter is supplied.
88 (58)	4	CHNTN	Address of this pointer itself if the TN subparameter of the CHAIN parameter is supplied.

PRINT FDT

Offset	*Content*	
0 (0)	P R I N	T ƀ ƀ ƀ
8 (8)	†INFILE	0
16 (10)	†FROMKEY	†FROMADDRESS
24 (18)	†SKIP	†TOKEY
32 (20)	†TOADDRESS	†COUNT
40 (28)	†*dname/password*	†INDATASET
48 (30)	0	†HEX
56 (38)	†CHARACTER	†DUMP
64 (40)	0	†ENVIRONMENT
72 (48)	†RECORDFORMAT	†BLOCKSIZE
80 (50)	†RECORDSIZE	0
88 (58)	†HINDEXDEVICE	†PRIMEDATADEVICE
96 (60)	†FIXUNB	†FIXBLK
104 (68)	†VARUNB	†VARBLK
112 (70)	†SPNUNB	†SPNBLK
120 (78)	†UNDEF	†FROMNUMBER
128 (80)	†TONUMBER	

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—PRINTƀƀƀ.
8 (8)	4	INDN	Address of this pointer itself if the INFILE parameter is supplied.
12 (C)	4	OUTDD	Address of information supplied through the OUTFILE parameter.
16 (10)	4	FMKYC	Address of information supplied through the FROMKEY parameter.
20 (14)	4	FMRBA	Address of information supplied through the FROMADDRESS parameter.
24 (18)	4	SKIP	Address of information supplied through the SKIP parameter.
28 (1C)	4	TOKYC	Address of information supplied through the TOKEY parameter.
32 (20)	4	TORBA	Address of information supplied through the TOADDRESS parameter.
36 (24)	4	COUNT	Address of information supplied through the COUNT parameter.
40 (28)	4	INPDD	Address of information supplied through the *dname/password* subparameter of the INFILE parameter.
44 (2C)	4	INDS	Address of information supplied through the INDATASET parameter.
48 (30)	4	*	Reserved—contains zero.
52 (34)	4	FHEX	Address of this pointer itself if the HEX parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
56 (38)	4	FCHAR	Address of this pointer itself if the CHARACTER parameter is supplied.
60 (3C)	4	FDUMP	Address of this pointer itself if the DUMP parameter is supplied.
64 (40)	4	*	Reserved—contains zero.
68 (44)	4	IENV	Reserved in OS/VS—contains zero.
72 (48)	4	IRFMT	Reserved in OS/VS—contains zero.
76 (4C)	4	IBKSZ	Reserved in OS/VS—contains zero.
80 (50)	4	IRCSZ	Reserved in OS/VS—contains zero.
84 (54)	4	*	Reserved—contains zero.
88 (58)	4	IHDEV	Reserved in OS/VS—contains zero.
92 (5C)	4	IPDEV	Reserved in OS/VS—contains zero.
96 (60)	4	IFUNB	Reserved in OS/VS—contains zero.
100 (64)	4	IFBLK	Reserved in OS/VS—contains zero.
104 (68)	4	IVUNB	Reserved in OS/VS—contains zero.
108 (6C)	4	IVBLK	Reserved in OS/VS—contains zero.
112 (70)	4	ISUNB	Reserved in OS/VS—contains zero.
116 (74)	4	ISBLK	Reserved in OS/VS—contains zero.
120 (78)	4	IUNDF	Reserved in OS/VS—contains zero.
124 (7C)	4	FMNUM	Address of information supplied through the FROMNUMBER parameter.
128 (80)	4	TONUM	Address of information supplied through the TONUMBER parameter.

REPRO FDT

Offset	*Content*	
0 (0)	**R E P R**	**O b b b**
8 (8)	†INFILE	†OUTFILE
16 (10)	†FROMKEY	†FROMADDRESS
24 (18)	†SKIP	†TOKEY
32 (20)	†TOADDRESS	†COUNT
40 (28)	†*dname/password*	†*dname/password*
48 (30)	†INDATASET	†OUTDATASET
56 (38)	†FROMNUMBER	†TONUMBER
64 (40)	0	†ENVIRONMENT
72 (48)	†RECORDFORMAT	†BLOCKSIZE
80 (50)	†RECORDSIZE	0
88 (58)	†HINDEXDEVICE	†PRIMEDATADEVICE
96 (60)	†FIXUNB	†FIXBLK
104 (68)	†VARUNB	†VARBLK
112 (70)	†SPNUNB	†SPNBLK
120 (78)	†UNDEF	0
128 (80)	0	†ENVIRONMENT
136 (88)	†RECORDFORMAT	†BLOCKSIZE
144 (90)	†RECORDSIZE	0
152 (98)	†HINDEXDEVICE	†PRIMEDATADEVICE
160 (A0)	†FIXUNB	†FIXBLK
168 (A8)	†VARUNB	†VARBLK
176 (B0)	†SPNUNB	†SPNBLK
184 (B8)	†UNDEF	0
192 (C0)	0	†Dummy
200 (C8)	†REPLACE	†NOREPLACE
208 (D0)	†REUSE	†NOREUSE

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—REPRObbbb.
8 (8)	4	INDN	Address of this pointer itself if the INFILE parameter is supplied.
12 (C)	4	OUTDN	Address of this pointer itself if the OUTFILE parameter is supplied.
16 (10)	4	FMKYC	Address of information supplied through the FROMKEY parameter.
20 (14)	4	FMRBA	Address of information supplied through the FROMADDRESS parameter.
24 (18)	4	SKIP	Address of information supplied through the SKIP parameter.
28 (1C)	4	TOKYC	Address of information supplied through the TOKEY parameter.
32 (20)	4	TORBA	Address of information supplied through the TOADDRESS parameter.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
36 (24)	4	COUNT	Address of information supplied through the COUNT parameter.
40 (28)	4	INPDD	Address of information supplied through the *dname/password* subparameter of the INFILE parameter.
44 (2C)	4	OUTDD	Address of information supplied through the *dname/password* subparameter of the OUTFILE parameter.
48 (30)	4	INDS	Address of information supplied through the INDATASET parameter.
52 (34)	4	OUTDS	Address of information supplied through the OUTDATASET parameter.
56 (38)	12	*	Reserved—contains zeros.
68 (44)	4	IENV	Address of this pointer itself if the ENVIRONMENT subparameter of the INFILE parameter is supplied.
72 (48)	4	IRFMT	Reserved in OS/VS—contains zero.
76 (4C)	4	IBKSZ	Reserved in OS/VS—contains zero.
80 (50)	4	IRCSZ	Reserved in OS/VS—contains zero.
84 (54)	4	*	Reserved—contains zero.
88 (58)	4	IHDEV	Reserved in OS/VS—contains zero.
92 (5C)	4	IPDEV	Reserved in OS/VS—contains zero.
96 (60)	4	IFUNB	Reserved in OS/VS—contains zero.
100 (64)	4	IFBLK	Reserved in OS/VS—contains zero.
104 (68)	4	IVUNB	Reserved in OS/VS—contains zero.
108 (6C)	4	IVBLK	Reserved in OS/VS—contains zero.
112 (70)	4	ISUNB	Reserved in OS/VS—contains zero.
116 (74)	4	ISBLK	Reserved in OS/VS—contains zero.
120 (78)	4	IUNDF	Reserved in OS/VS—contains zero.
124 (7C)	8	*	Reserved—contains zeros.
132 (84)	4	OENV	Reserved in OS/VS—contains zero.
136 (88)	4	ORFMT	Reserved in OS/VS—contains zero.
140 (8C)	4	OBKSZ	Reserved in OS/VS—contains zero.
144 (90)	4	ORCSZ	Reserved in OS/VS—contains zero.
148 (94)	4	*	Reserved—contains zero.
152 (98)	4	OHDEV	Reserved in OS/VS—contains zero.
156 (9C)	4	OPDEV	Reserved in OS/VS—contains zero.
160 (A0)	4	OFUNB	Reserved in OS/VS—contains zero.
164 (A4)	4	OFBLK	Reserved in OS/VS—contains zero.
168 (A8)	4	OVUNB	Reserved in OS/VS—contains zero.
172 (AC)	4	OVBLK	Reserved in OS/VS—contains zero.
176 (B0)	4	OSUNB	Reserved in OS/VS—contains zero.
180 (B4)	4	OSBLK	Reserved in OS/VS—contains zero.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
184 (B8)	4	OUNDF	Reserved in OS/VS—contains zero.
188 (BC)	8	*	Reserved—contains zeros.
196 (C4)	4	IDUMY	Address of this pointer itself if the DUMMY subparameter of the ENVIRONMENT parameter is supplied for the input data set.
200(C8)	4	REP	Address of this pointer itself if the REPLACE parameter has been supplied.
204(CC)	4	NREP	Address of this pointer itself if the NDREPLACE parameter has been supplied.
208(D0)	4	RUS	Address of this pointer itself if the REUSE parameter has been supplied.
212(D4)	4	NRUS	Address of this pointer itself if the NOREUSE parameter has been supplied.

RESETCAT FDT
(VS2.03.808 only)

Offset	*Content*	
0 (0)	**R E S E**	**T C A T**
8 (8)	†CATALOG	†catname/password
16 (10)	†dname	†MASTERPW
24 (18)	†WORKFILE	†WORKCAT
32 (20)	†IGNORE	†NOIGNORE
40 (28)	†CRAFILES count	†dname
48 (30)	†ALL	†NONE
56 (38)	†CRAVOLUMES count	†volser
64 (40)	†devtype	†dname/password

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb—RESETCAT
8 (8)	4	CAT	Address of this parameter itself if the CATALOG parameter has been supplied.
12 (C)	4	CATNM	Address of the information supplied through the *catname/password* subparameter of the CATALOG parameter.
16 (10)	4	CATDN	Address of information supplied through the *dname* subparameter of the CATALOG parameter.
20 (14)	4	MRPW	Address of information supplied through the *password* subparameter of the MASTERPW parameter.
24 (18)	4	WFDN	Address of this parameter itself if the WORKFILE parameter is supplied.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
28 (1C)	4	WCAT	Address of the information supplied through the *catname/password* subparameter of the WORKCAT parameter.
32 (20)	4	IGN	Address of this parameter itself if the IGNORE parameter is supplied.
36 (24)	4	NIGN	Address of this parameter itself if the NOIGNORE parameter is supplied.
40 (28)	4	CFILE	Count of the number of CRAs that specified through the CRAFILES parameter.
44 (2C)	4	CRADNPTR	Address of an array of pointers. Each pointer points at a *dname* for the CRA it relates to in the order that they appear in the CRAFILES parameter.
48 (30)	4	ALLPPTR	Address of an array of pointers. Each pointer points to itself if ALL was specified for the related CRA in the CRAFILES parameter.
52 (34)	4	NONEPTR	Address of an array of pointers. Each pointer points to itself if NONE was specified for the related CRA in the CRAFILES parameter.
56 (38)	4	CVOL	Address of this parameter itself if the CRAVOLUMES parameter is supplied.
60 (3C)	4	CRAVLPTR	Address of an array of pointers. Each pointer points to the information supplied by the *volser* subparameter of the CRAVOLUMES parameter.
64 (40)	4	CRADVPTR	Address of an array of pointers. Each pointer points to the information supplied by the *devtype* subparameter of the CRAVOLUMES parameter.
68 (44)	4	WFILE	Address of the information supplied by the *dname/password* subparameter of the WORKFILE parameter.

VERIFY FDT

Offset	*Content*	
0 (0)	V E R I	F Y b b
8 (8)	↑FILE	↑DATASET

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	FDTVERB	Verb aligned left and padded with blanks—VERIFYbb.
8 (8)	4	IN	Address of information supplied through the FILE parameter.
12 (C)	4	INDAT	Address of information supplied through the DATASET parameter.

GDT
Global Data Table

The GDT is the directory for the services and data areas of the processor. It contains a branch vector for the services provided by the System Adapter, the I/O Adapter, and the Text Processor. It also points to the invoker's parameter list, trace tables, and historical tables. The GDT is always the first parameter passed to any routine. The GDT is contained in the storage associated with module IDCSA01.

Created by	Modified by	Used by	Size
IDCSA01	All service routines	All routines	276

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	GDTHDR	Global Data Table header; contains 'GDTb'.
4 (4)	4	GDTPRM	Address of parameter list from invoker of the processor.
8 (8)	4	GDTTR1	Address of Inter-Module Trace Table.
12 (C)	4	GDTTR2	Address of Intra-Module Trace Table.
16 (10)	4	GDTDBH	Address of Debugging-Aids historical area.
20 (14)	4	GDTSTH	Reserved—contains zero.
24 (18)	4	GDTRIH	Address of Reader/Interpreter historical area.
28 (1C)	4	GDTTPH	Address of Text Processor historical area.
32 (20)	4	GDTSAH	Address of System Adapter historical area.
36 (24)	4	GDTIOH	Address of I/O Adapter historical area.
40 (28)	4	GDTDBG	Address of entry point for dump routine, IDCDB01, (UDUMP macro).
44 (2C)	4	GDTSTC	Reserved—contains zero.
48 (30)	4	GDTPRT	Address of entry point to print, IDCIOPR, (UPRINT macro).
52 (34)	4	GDTESS	Address of entry point to establish PCT from Text Structure, IDCTPES, (UESTS macro).
56 (38)	4	GDTESA	Address of entry point to establish PCT from storage, IDCTPEA, (UESTA macro).
60 (3C)	4	GDTRST	Address of entry point to modify PCT, IDCTPRS, (UREST macro).
64 (40)	4	GDTRES	Address of entry point to reset PCT, IDCTPRE, (URESET macro).
68 (44)	4	GDTCAL	Address of entry point to call, IDCSACL, (UCALL macro).
72 (48)	4	GDTGSP	Address of entry point to get space, IDCSAGS, (UGSPACE macro).

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
76 (4C)	4	GDTFSP	Address of entry point to free storage, IDCSAFS, (UFSPACE macro).
80 (50)	4	GDTGPL	Address of entry point to get storage, IDCSAGP, (UGPOOL macro).
84 (54)	4	GDTFPL	Address of entry point to free storage, IDCSAFP, (UFPOOL macro).
88 (58)	4	GDTLOD	Address of entry point to load module, IDCSALD, (ULOAD macro).
92 (5C)	4	GDTDEL	Address of entry point to delete module, IDCSADE, (UDELETE macro).
96 (60)	4	GDTPRL	Address of entry point for prologue, IDCSAPR.
100 (64)	4	GDTEPL	Address of entry point for epilogue, IDCSAEP, (UEPIL macro).
104 (68)	4	GDTTIM	Address of entry point for time, IDCSATI, (UTIME macro).
108 (6C)	4	GDTIIO	Address of entry point for I/O initialization, IDCIOIT, (UIOINIT macro).
112 (70)	4	GDTTIO	Address of entry point for I/O termination, IDCIOTM, (UIOTERM macro).
116 (74)	4	GDTRIP	Address of Reader/Interpreter name—either 'IDCRI01' or 'IDCRI04'.
120 (78)	4	GDTTOH	Reserved—contains zero.
124 (7C)	4	GDTOPN	Address of entry point to open data sets, IDCIOOP, (UOPEN macro).
128 (80)	4	GDTCLS	Address of entry point to close data sets, IDCIOCL, (UCLOSE macro).
132 (84)	4	GDTGET	Address of entry point to get a logical record, IDCIOGT, (UGET macro).
136 (88)	4	GDTPUT	Address of entry point to put a logical record, IDCIOPT, (UPUT macro).
140 (8C)	4	GDTPOS	Address of entry point to position to a logical record, IDCIOPO, (UPOSIT macro).
144 (90)	4	GDTCPY	Address of entry point to copy logical records, IDCIOCO, (UCOPY macro).
148 (94)	4	GDTCAT	Address of entry point for manipulating VSAM catalog, IDCSACA, (UCATLG macro).
152 (98)	4	GDTABT	Address to abort, SAABT in IDCSA02, (UABORT macro).
156 (9C)	4	GDTABH	Address of UABORT register save area.
160 (A0)	4	*	Reserved—contains zero.
164 (A4)	4	GDTSNP	Address of entry point to snap dump, IDCSASN, (USNAP macro).
168 (A8)	4	GDTSPR	Address of IDCSA03's storage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
172 (AC)	4	GDTVFY	Address of entry point to VERIFY data set, IDCIOVY, (UVERIFY macro).
176 (B0)	4	GDTCMB	Address of TSO command buffer obtained from TSO.
180 (B4)	4	GDTUPT	Address of TSO user profile table obtained from TSO.
184 (B8)	4	GDTPSB	Address of TSO protected step block obtained from TSO.
188 (BC)	4	GDTECT	Address of TSO environment table obtained from TSO.
192 (C0)	4	GDTUID	Address of entry point to obtain TSO user's identification, IDCSAID, (UID macro).
196 (C4)	4	GDTPMT	Address of entry point to obtain information from TSO user IDCSAPT, (UPROMPT macro).
200 (C8)	4	GDTCIR	Address of entry point to get fully-qualified data set names IDCSACR, (UCIR macro).
204 (CC)	4	GDTLNK	Address of entry point to link to module IDCSALK, (ULINK macro).
208 (D0)	4	GDTALC	Address of entry point to allocate a data set or mount a volume, IDCSAAL, (UALLOC macro).
212 (D4)	4	GDTDLC	Address of entry point to deallocate a data set or dismount a volume, IDCSADL, (UDEALLOC macro).
216 (D8)	4	GDTQAL	Address of entry point to construct a fully-qualified data set name in TSO, IDCSAQL, (UQUAL macro).
220 (DC)	4	GDTSTW	Address of entry point to stow a member name of a partitioned data set, IDCIOST, (USTOW macro).
224(E0)	4	GDTSSC	Address of entry point to perform Mass Storage Control (MSC) functions, IDCSA06 (USSC macro).
228(E4)	4	GDTENQ	Address of entry point to acquire control of a resource, IDCSANQ (UENQ macro).
232(E8)	4	GDTRSV	Address of entry point to acquire control of an I/O unit, IDCSARV (URESERVE macro).
236(EC)	4	GDTDEQ	Address of entry point to release a resource IDCSADQ (UDEQ macro).
240(F0)	4	GDTSFO	Address of entry point to obtain system control-block information, IDCSASI (USYSINFO macro).
244(F4)	4	GDTWTO	Address of entry point to write a message to the console operator, IDCSAWO (UWTO macro).
248(F8)	4	GDTSCR	Address of entry point to delete a direct-access data set, IDCSASC (USCRATCH macro).

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
252(FC)	4	GDTUNT	Address of entry point to mount, demount, post, or check a volume, IDCSA06 (UMSSUNIT macro).
256(100)	4	GDTRCT	Address of entry point to recatalog a nonVSAM data set, IDCSARC (URECAT macro).
260(104)	4	GDTIFO	Address of entry point to obtain data-set information, IDCIOSI (UIOINFO macro).
264(108)	4	GDTEXP	Address of entry point to open, close, read, or write a volume with EXCP, IDCIO05 (UEXCP macro).
268(10C)	4	GDTSTA	Address of entry point to perform ABEND recovery, IDCSA10 (USTAE macro).
272(110)	4	GDTERR	Address of entry point to convert numeric return codes, (UERROR macros).
276(114)	4	GDTUNC	Address of entry point to uncatalog and scratch a data set, IDCSAUC (UUNCATLG macro).
280(118)	4	GDTLOC	Address of entry point to locate a VSAM data set, IDCSALC (ULOCATE macro).
284(11C)	4	GDTRCK	Address of entry point to obtain RACF authorization, IDCSA08 (URACHECK macro).

IPT
Input Parameter Table

The Input Parameter Table is a parameter list passed by IDCRC01 to IDCRC02 within EXPORTRA. It is an array of five pointers. Each object pointed to is described after the IPT pointers.

Created by	Modified by	Used by	Size
IDCRC01	IDCRC02	IDCRC02	20

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	4		Address of control block describing the object to be exported.
4(4)	4		Address of control block describing the output (portable) data set.
8(8)	4		Address of the input data set name.
12(C)	4		Address of the output data set name.
16(10)	4		Address of the environment parameter.
Description of control block describing object to be exported.			
0(0)	1	OBJTYP	Type of object.
1(1)	.3	OBJVAL	The catalog control interval number of the entry.
4(4)	4	RESINP	Reserved
8(8)	1	OBJPLN	Password length.
9(9)	8	OBJPAS	Password
Description of control block describing output (portable) data set.			
0(0)	4	OUTLEN	Maximum record length of data component.
4(4)	4	SAVOIOCS	Pointer to output IOCS.
8(8)	4	USBKSZ	User supplied output blocksize.
12(C)	4	RESOUTP	Reserved.
16(10)	1	OUTFLGS	Status of output data set.
	1...	OPNFLG	This flag is on if output data set is open.
	.1..	ENDFLG	This flag is on if this is the last request.
	..1.	EMPTYDS	This flag is on if the object contains no data records.

The third pointer in the IPT points to an 8-byte input dname.

The fourth pointer in the IPT points to an 8-byte output dname.

The fifth pointer in the IPT points to an 8-byte field describing the prime data device (PDEV subparameters).

IODATA
I/O Adapter Historical Area

The I/O Adapter historical area is pointed to by GDTIOH. It is built on the first call to the I/O Adapter (UIOINIT Umacro), and contains information that is common to all modules of the I/O Adapter.

Created by	Modified by	Used by	Size
IDCIO01	IDCIO01	IDCIO01 IDCIO02	68

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	IODIOC	First IOCSTR in chain.
4 (4)	4	IODMSG	Address of an area for VSAM messages.
8 (8)	4	IODADD	Address of the alternate DD list.
12 (C)	4	IODXTN	Address of the external I/O routine list.
16 (10)	4	IODSID	Identifier containing:
	2	IODMID	Module identifier.
	. . 2	IODINC	Pool identifier.
20 (14)	4	IODIEV	Address of input End-of-Volume exit routine.
24 (18)	4	IODOEV	Address of output End-of-Volume exit routine.
28 (1C)	4	*	Reserved—contains zero.
32 (20)	4	IODEOD	Address of end-of-data routine for non-VSAM data sets.
36 (24)	4	IODOSS	Non-VSAM input SYNAD routine address.
40 (28)	4	IODOSO	Non-VSAM output SYNAD routine address.
44 (2C)	4	IODICS	Address of Access Method Services input IOCSTR.
48 (30)	4	IODOCS	Address of the Access Method Services output IOCSTR.
52 (34)	4	IODIOX	Address of the EXCP control-block chain.
56 (38)	4	*	Reserved—contains zero.
60 (3C)	4	IODAEI	Address of VSAM EODAD routine.
64 (40)	4	*	Reserved—contains zero.

IOCSTR
Input/Output Communications Structure

An IOCSTR exists for each open data set, or for any on which an open has been attempted. It contains all information about the data set that may be required by the processor. An IOCSTR is built at open time, and a pointer to the IOCSTR is returned to the requester of the open, in the OPNIOC field of the OPNAGL. A UGPOOL area immediately precedes the IOCSTR. The UGPOOL area contains the identifier assigned to the data set by the I/O Adapter. All other requests for I/O service include this IOCSTR as one of the parameters for the request.

Created by	Modified by	Used by	Size
IDCIO02	All routines	All routines	68

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
–4 (–4)	4	*	Always contains 'IOCS'.
0 (0)	4	IOCDAD	Address of data area.
4 (4)	4	IOCDLN	Length of data record.
8 (8)	4	IOCTRN	Transmission length: LRECL for logical processing or control interval for block processing.
12 (C)	1	IOCKYL	Key length in bytes.
13 (D)	. 3	IOCRKP	Relative key position, value assumes VSAM or ISAM meaning.
16 (10)	1	IOCDSO	Data set organization:
	1...	IOCDSOAM	VSAM data set.
	.1..	IOCDSOPS	Non-VSAM sequential data set (QSAM or BSAM).
	..1.	IOCDSOIS	Indexed sequential (ISAM) data set.
	...1	IOCDSOPO	Partitioned (BPAM) data set.
17 (11)	. 1	IOCRFM	Non-VSAM record format:
	1...	IOCRFMFX	Fixed-length records.
	.1..	IOCRFMVR	Variable-length records, not spanned.
	..1.	IOCRFMUN	Undefined-length records.
	...1	IOCRFMSF	Spanned records.
	.... 1...	IOCRFMBK	Blocked records.
18 (12)	. . 1	IOCMAC	Macro form used:
	1...	IOCMACIN	Input processing.
	.1..	IOCMACOT	Output processing.
	..1.	IOCMACUP	Update processing.
	...0	IOCMACCR	Keyed sequence (VSAM).
	...1		Entry sequence (VSAM).
	.... 0...	IOCMACBK	Logical records (QSAM or QISAM).
	.... 1...		Blocks or control intervals (BSAM, BPAM, or VSAM).
	.... .0..	IOCMACDR	Sequential data set.
	.... .1..		Direct data set.
	.... ..1.	IOCMACCC	Copy catalog.
19(13)	...1	IOCMAC2	
	1...	IOCMACSK	Skip sequential processing.
	.1..	IOCMACAS	Asynchronous processing.
	..1.	IOCMACRR	Relative record processing.
	...1	IOCMACCP	Change processing.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
	.... 1...	IOCMACEN	PUT-ENDREQ processing (**VS2.03.808**)
	.... .1..	IOCMACPA	Reprocessing flag.
	.... ..1.	IOCMACER	PUT-ERASE processing (**VS2.03.808**)
	1	IOCMACNT	NOTE processing (**VS2.03.808**)
20(14)	1	IOCCHP	Change processing mode:
	1...	IOCCHPSQ	Change to sequential.
	.1..	IOCCHPDR	Change to direct.
	..1.	IOCCHPSK	Change to skip sequential.
	...1	IOCCHPKS	Change to keyed.
	.... 1...	IOCCHPCR	Change to addressed.
	.... .1..	IOCCHPBK	Change to control interval.
	.... ..1.	IOCCHPUP	Change to update (**VS2.03.808**).
	1	IOCCHPNU	Change to no update (**VS2.03.808**)
21 (15)	. 1	IOCMSG	Message flags:
	1...	IOCCHPKE	Change to key equal (**VS2.03.808**).
	.1..	IOCCHPKG	Change to key greater than or equal (**VS2.03.808**).
	..1.	IOCMSGOP	Data set is open for processing.
	...1	IOCMSGOE	VSAM open error.
	.... 1...	IOCMSGCE	VSAM close error.
	.... .1..	IOCMSGAE	VSAM error other than open or close.
	.... ..1.	IOCMSGSM	Suppress logical error messages (**VS2.03.808**)
22(16)	..6	IOCVOLSR	Volume serial number of the opened data set.
28(1C)	4	IOCHURBA	High used RBA.
32 (20)	4	IOCDSN	Address of data set name. The data set name usually follows the IOCSTR extension.
36 (24)	4	IOCCBP	Address of ACB or DCB if OPNMODRC is set.
40 (28)	4	*	Either IOCRBA, IOCTTR, or IOCVRC:
	4	IOCRBA	VSAM record relative byte address RBA).
	4	IOCTTR	Track and record number (TTR) for UPOSIT macro.
	4	IOCVRC	VSAM error codes.
44 (2C)	4	*	Either IOCMEM or IOCKYA:
	4	IOCMEM	Address of member name for BPAM data set.
	4	IOCKYA	Address of key.
48 (30)	4	*	Either IOCNWM or IOCPTL with IOCPNM:
	4	IOCNWM	Address of new member name for BPAM data set.
	2	IOCPTL	Length of key supplied for position request.
	. . 2	IOCPNM	Number of stacked puts.
52 (34)	4	IOCRRN	Relative record number.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
56(38)	4	IOCWORK	Address of input work area.
60(3C)	4	IOCREL	Relative record number.
64 (40)	4	IOCEXT	IOCSTR extension address.

IOCSEX
Input/Output Communications Structure Extension

The IOCSTR Extension is built immediately after the IOCSTR. However, for flexibility and to make the IOCSTR easily extensible, field IOCEXT points to the IOCSEX.

Created by	Modified by	Used by	Size
IDCIO02	IDCIO01	IDCIO01	45

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	IOCCBA	Address of ACB or DCB.
4 (4)	4	*	Either IOCRPL or IOCDEC:
	4	IOCRPL	Address of VSAM RPL.
	4	IOCDEC	Address of update DECB.
8 (8)	2	IOCCBL	Length of ACB or DCB.
10 (A)	. . 2	IOCLRP	Length of RPL.
12 (C)	4	IOCWKA	Address of input workarea.
At decimal displacements 16 and 20, one of the two following sets of fields appears:			
Set one:			
16 (10)	4	IOCXAD	User routine address.
20 (14)	4	IOCXPM	User routine parameter address.
Set two:			
16 (10)	4	IOCEXA	VSAM exit list address.
20 (14)	2	IOCEXL	VSAM exit list length.
22 (16)	. . 2	*	Reserved—contains zero.
The data area then continues as follows.			
24 (18)	4	IOCNIO	Address of next IOCSTR in chain.
28 (1C)	4	IOCSID	Storage pool identifier.
32 (20)	1	IOCFLG	Extension flags:
	1...	IOCFLGEX	User controlled data set.
	.1..	IOCFLGDF	Data set defined by JCL.
	..1.	IOCFLGEF	End-of-file on user data set.
	...1	IOCFLGIO	DD * or SYSOUT data set.
	.... 1...	IOCFLGOP	Data set is open.
	.... .x..	IOCFLGOE	Reserved—contains zero.
	.... ..1.	IOCFLGSP	Access Method Services print data set.
33 (21)	. 1	IOCDEV	Device type flags:
	1...	IOCDEVDA	Direct access.
	.1..	IOCDEVMT	Magnetic tape.
	..1.	IOCDEVUR	Unit record.
34 (22)	. . 1	IOCINF	Information flags:
	1...	IOCINFPT	Point has been issued.
	.1..	IOCINFAE	ABEND exit taken.
	..x.	IOCINFND	ReServed—contains zero.
	...x	IOCINFQX	Reserved—contains zero.
	.... 1...	IOCINFAC	ANSI control character.
	.... .x..	IOCINFDO	Reserved in OS/VS—contains zero.
	.... ..1.	IOCINFCT	Opened as a catalog.
	x	IOCINFR1	Reserved—contains zero.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
35 (23)	. . . 1	IOCMOD	Additional information flags:
	1...	IOCMODPD	Process dummy (deleted) records.
	.1..	IOCMODRR	Return the RPL address.
	..1.	IOCMODDY	Dynamically allocated data set.
	...1	IOCMODRG	DSORG specified.
	.... 1...	IOCMODUB	User buffering.
	.... .1..	IOCMODXM	Export/Import.
	.... ..1.	IOCMODRP	Replace Processing.
	1	IOCMODEX	Exclusive Control.
36 (24)	4	IOCDLM	Reserved in OS/VS—contains zero. **(without VS2.03.808)**
		IDOCDDN	Pointer to DDname **(with VS2.03.808)**
40 (28)	2	IOCDNM	Reserved in OS/VS—contains zero.
42 (2A)	. . 2	*	Reserved—contains zero.
44 (2C)	1	IOCRCV	Recovery Flags.
	1...	IOCRCVXM	VSAM recovery flag.
	.1..	IOCRCVRA	Open CRA.

IOXCTLBK
Input/Output Control Block for EXCP

The IOXCTLBK is built when a data set is opened with UEXCP. It is used throughout all processing until the data set is closed with UEXCP.

Created by	Modified by	Used by	Size
IDCIO05	IDCIO05	IDCIO05	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	4	IOXCHAIN	Contains either 0 or the address of the previous IOXCTLBK.
4(4)	8	IOXID	Identifies the control block in a dump with the characters 'IOXCTLBK'.
12(C)	4	IOXWRITE	Contains the number of new records written.
16(10)	4	IOXREADS	Contains the number of records read.
20(14)	2	IOXACODE	Contains the ABEND code for OPEN/CLOSE.
22(16)	..2	IOXRCODE	Contains the return code for OPEN/CLOSE.
24(18)	1	IOXTYPE	Contains the type of EXCP open request: X'10' for volume label, X'20' for MSC tables, X'30' for password checking, and X'40' for new volume.
25(19)	.1	IOXFLGS	Flags:
	1...	IOXOPN	DCB opened.
	.1..	IOXABEND	Password abend exit entered.
	..1.	IOXCLOSE	DCB closed.
	...1	IOXSYNAD	SYNADerror occurred.
26(1A)	..2	*	Reserved—contains zeros.
28(1C)	8	IOXDDN	Eight-byte DD name.
36(24)	4	IOXDCBID	Identifies the DCB in a dump with the characters 'DCBb'.
40(28)	v I	IOXDCB	Contains the DCB for the data set.
	4	IOXECBID	Identifies the control block in a dump with the characters 'ECBb'.
	8	IOXECB	Contains the code posted by the I/O supervisor and the address used by the I/O Supervisor.
	4	IOXIOBID	Identifies the control block in a dump with the characters 'IOBb'.
	v	IOXIOB	Contains the I/O control block used by EXCP.
	4	IOXJFCBI	Identifies the job file control block (JFCB) in a dump with the characters 'JFCB'.
	200	IOXJFCB	Contains the job file control block (JFCB).

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
40(28) (Cont.)	4	IOXEXID	Identifies the control block in a dump with the characters 'XLST'.
	1	*	Reserved—contains zeros.
	.3	IOXJFCBP	Address of the altered job file control block (JFCB).
	4	*	Reserved—contains zeros.
	4	IOXUCBP	Address of the UCB.
	4	IOXCCWID	Identifies the control block in a dump with the characters 'CCWb'.
	1000	IOXCCW5	Array containing the CCW's.
	400	IOXCOUNT	Array containing record count fields for records to be written.

Inter-Module Trace Table

The Inter-Module Trace Table contains information on the flow of control between modules. The table is pointed to by GDTTR1. The oldest identifier is at the beginning of the table. The latest identifier is at the end of the table. Each time a UPROL or UEPIL macro is issued the oldest identifier is removed and the new identifier is added at the end. A UPROL adds the identifier of the current module. A UEPIL adds the identifier of the module to which control is being returned. The UDUMP macro prints the table on SYSPRINT.

Created by	Modified by	Used by	Size
IDCSA01	UEPIL UPROL macros	IDCDB01	100

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
–6 (–6)	6	*	Table identification 'INTERb'.
0 (0)	100	*	Inter-Module Trace Table with 20 entries.
Each entry contains the following:			
	4	*	Identifier provided by module issuing UEPIL or UPROL macros. The identifier is the last four characters of the module name.
	1	*	Blank 'b'.

Intra-Module Trace Table

The Intra-Module Trace Table contains information on the flow of control within modules. The table is pointed to by GDTTR2. The oldest identifier is at the beginning of the table. The latest identifier is at the end of the table. Each time a UTRACE is issued the oldest identifier is removed and the new identifier is added at the end. The UDUMP macro prints the table on SYSPRINT.

Created by	Modified by	Used by	Size
IDCSA01	UTRACE macro	IDCDB01	100

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
-6 (-6)	6	*	Table identification 'INTRA␣'.
0 (0)	100	*	Intra-Module Trace Table with 20 entries.
Each entry contains the following:			
	4	*	Identifier provided by module issuing UTRACE. The first two characters are the mnemonic identifier which are characters 4 and 5 of the module name. For example, EX refers to the Executive.
	1	*	Blank '␣'.

LLBLK
Load List Block
(VS2.03.807 only)

The Load List Block contains an entry for each module loaded by UCALL, ULOAD, and ULINK. It is used to control the loading and deleting of modules used by Access Method Services.

Created by	Modified by	Used by	Size
IDCSA01 IDCSA02	IDCSA02 IDCSA03	IDCSA02 IDCSA03	392

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	LLFSTSLT	Address of first available slot; zero if no slots available.
4 (4)	4	LLNXTBLK	Pointer to next Load List Block.
8 (8)	16	LLSLOT	Load List Slot (24 per block).
Each available inactive slot contains the following:			
8 (8)	4	LLNXTSLT	Pointer to next available slot; zero if last slot in block.
12 (C)	4	LLNOMOD	Contains binary zeros.
Each active slot contains the following:			
8 (8)	8	LLNAME	Name of loaded module in EBCDIC.
16 (10)	4	LLADDR	Entry point or address of loaded module.
20 (14)	1	LLUSECTR	Module use count; if zero, module is not in use.
21 (15)	3	LLMODSZ	Size of module.

LCTINFO
Locate Data Set Return Information Area

LCTINFO is an area a caller passes when a ULOCATE macro is issued. Information about a nonVSAM data set is returned to the caller in this area.

Created by	Modified by	Used by	Size
Calling routine	IDCSA07	IDCSA07	33

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	LCTHEAD	Initialized with 'LCTINFO' by ULOCATE.
8 (8)	4	LCTMULVC	Number of volumes for this data set entry in the catalog.
12 (C)	4	LCTACBP	Pointer to the ACB.
16 (10)	4	LCTCREAT	VSAM creation date
20 (14)	4	LCTEXPIR	VSAM expiration date
24 (18)	8	OCTOWNER	Catalog entry owner name.
32 (20)	1	LCTFLAGS	Return flags.
	1...	LCTVSCAT	VSAM catalog entry.

IDCRIKT
Modal Verb and Keyword Symbol Table

Load module IDCRIKT contains the Modal Verb and Keyword Symbol Table, which acts as the "Command Descriptor" for the modal commands.

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCRI01	90

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	PARMSMLN	Length of PARM character string.
1 (1)	. 9	PARMSYM	PARM character string.
10 (A)	. . 1	SETSMLN	Length of SET character string.
11 (13)	. . . 9	SETSYM	SET character string.
20 (14)	1	IFSMLN	Length of IF character string.
21 (15)	. 9	IFSYM	IF character string.
30 (1E)	. . 1	THENSMLN	Length of THEN character string.
31 (1F)	. . . 9	THENSYM	THEN character string.
40 (28)	1	ELSESMLN	Length of ELSE character string.
41 (29)	. 9	ELSESYM	ELSE character string.
50 (32)	. . 1	DOSMLN	Length of DO character string.
51 (33)	. . . 9	DOSYM	DO character string.
60 (3C)	1	ENDSMLN	Length of END character string.
61 (30)	. 9	ENDSYM	END character string.
70 (46)	. . 1	LSTCCLN	Length of LASTCC character string.
71 (47)	. . . 9	LSTCCSYM	LASTCC character string.
80 (50)	1	MAXCCLN	Length of MAXCC character string.
81 (51)	. 9	MAXCCSYM	MAXCC character string.

MDAGL
Mount/Demount Argument List

The MDAGL is passed when a UMSSUNIT macro is issued with a mount or demount request. It defines a request to mount or demount a mass storage volume.

Created by	Modified by	Used by	Size
Calling routine	IDCSA06	IDCSA06	31

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	MDHEAD	Set by IDCSA06 with 'MOUNTbbb' for a mount request or 'DEMOUNTb' for a demount request.
8(8)	4	MDUCBPTR	Address of an area containing the UCB address or zeros if the UCB address is unknown. If the UCB address is zeros, the UCB address will be put in the area upon return.
12(C)	8	MDDDNAME	DD name of the volume.
20(14)	4	MDPUAGL	Address of an initialized PUAGL argument list if bits MDPOST or MDCLEAR are set.
24(18)	6	MDLABEL	Volume serial number recorded in the MSC tables.
30(1E)	..1	MDFLAGS	Mount/demount options:
	1...	MDNEWVOL	Indicates that the volume to be mounted does not have a volume label or a VTOC. This field is ignored on a demount request.
	.1..	MDCLEAR	Indicates that the caller wants to clear the volume serial from a UCB prior to the mount function. This field is ignored on a demount request.
	..1.	MDPOST	Indicates that the caller wants to post the volume serial number in a UCB after the demount function. This field is ignored on a mount request.
	...1	MDWAIT	Indicates that the caller wants a demount with delayed response to be issued. This field is ignored on a mount request.
	.... 1...	MDENQ	Indicates that an enqueue was done by IDCSA06 as part of a mount request.
	.... .1..	MDDEQ	Indicates that the volume should be dequeued when demounted.

OPNAGL
Open Argument List

The OPNAGL defines a request to open a data set. The address of the OPNAGL is passed as a parameter to the I/O Adapter from any routine that requires the open function.

Created by	Modified by	Used by	Size
Routine that requests an open	IDCIO02	IDCIO02	48

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	OPNOPT	Open options (determine data set usage):
	1...	OPNOPTIN	Input data set.
	.1..	OPNOPTOT	Output data set.
	..1.	OPNOPTUP	Update mode of processing.
	...1	OPNOPTBK	Block processing.
	.... 1...	OPNOPTKS	Keyed processing.
	.... .1..	OPNOPTCR	Addressed processing.
	.... ..1.	OPNOPTDR	Direct processing.
	1	OPNOPTSK	Skip sequential processing.
1 (1)	. 1	OPNRFM	Non-VSAM output record format. Required in DOS/VS; optional OS/VS:
	1...	OPNRFMFX	Fixed.
	.1..	OPNRFMVR	Variable.
	..1.	OPNRFMUN	Undefined.
	...1	OPNRFMSF	Spanned.
	.... 1...	OPNRFMBK	Blocked.
2 (2)	. . 1	OPNTYP	Data set type:
	1...	OPNTYPSI	System input (SYSIN) is to be opened. OPNIOC is the only other required field.
	.1..	OPNTYPSO	System output (SYSPRINT) is to be opened. OPNIOC is the only other required field.
	..1.	OPNTYPCI	Catalog to be opened.
	...1	OPNTYPXM	Export/import data set to be opened.
	.... 1...	OPNTYPRA	Catalog Recovery Area.
	.... .1..	OPNTYPEX	Data set is to be opened for exclusive control.
	.... ..1.	OPNTYPRV	VSAM recovery processing.
	1	OPNTYPSY	Bypass security checking (**VS2.03.807**)
3 (3)	. . . 1	OPNMOD	Open modifiers:
	1...	OPNMODPD	Process ISAM dummy (deleted) records.
	.1..	OPNMODAC	ANSI control character.
	..1.	OPNMODRC	Return control block address.
	...1	OPNMODRR	Return RPL address.
	.... 1...	OPNMODAX	Open alternate index.
	.... .1..	OPNMODRS	Open with reset.
	.... ..1.	OPNMODUB	User buffering.
	1	OPNMODRP	Replace processing.
4 (4)	4	OPNIOC	Address of pointer of IOCSTR. This field is always present. After a successful open, the pointer contains

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			the address of the IOCSTR built by the I/O Adapter.
8 (8)	4	OPNDDN	Address of eight-byte DD name (not present when SYSIN or SYSPRINT is being opened but required at all other times).
12 (C)	4	OPNPWA	Address of an optional eight-byte password, used only with VSAM data sets.
16 (10)	4	OPNDSN	Address of 44-byte data set name.
20 (14)	4	*	Reserved—contains zero.
24 (18)	4	OPNDEVDT	Address of data device type.
28 (1C)	4	OPNDEVIX	Address of index device type.
32 (20)	4	OPNREC	Logical record length, optional.
36 (24)	4	OPNBLK	Block size, optional.
40 (28)	1	OPNKYL	Key length.
41 (29)	. 1	OPNDSO	Data set organization:
	1...	OPNDSOAM	VSAM.
	.1..	OPNDSOPS	non-VSAM.
	..1.	OPNDSOIS	Indexed sequential (ISAM).
	...1	OPNDSOPO	Partitioned (BPAM).
42 (2A)	. . 1	OPNOPT2	Second option byte.
	1...	OPNOPTAS	Asynchronous processing.
	..1.	OPNOPTJM	Modify JFCB.
43 (2B)	...1	*	Reserved—contains zeros. (without **VS2.03.808**)
		OPNSTRNO	Number of strings. (with **VS2.03.808**)
44 (2C)	4	OPNVOL	Pointer to volume serial number.

OCARRAY
Open Close Address Array

The Open Close Address Array is used to pass the address of the OPNAGL or IOCS for up to four data sets at once from IDCIO01 to IDCIO02. It is used within the I/O Adapter.

Created by	Modified by	Used by	Size
IDCIO01	None	IDCIO02	20

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	OCATYP	Type of operation: 1 means open, 2 means close.
1 (1)	. 1	OCAOPT	Options:
	1...	OCAOPTCA	Close all open data sets.
2 (2)	. . 1	OCANUM	Number of data sets to open.
3 (3)	. . . 1	*	Reserved—contains zero.
4 (4)	4	OCADDR*1*	Address of first OPNAGL for open or address of first IOCSTR for close.
8 (8)	4	OCADDR*2*	Address of second OPNAGL for open or address of second IOCSTR for close.
12 (C)	4	OCADDR*3*	Address of third OPNAGL for open or address of third IOCSTR for close.
16 (10)	4	OCADDR*4*	Address of fourth OPNAGL for open or address of fourth IOCSTR for close.

OPRARG
Positioning Argument List

OPARG contains the address of the IOCSTR for a data set that is to be positioned or for a partitioned data set whose directory is to be altered. It is used within the I/O Adapter.

Created by	Modified by	Used by	Size
IDCIO01	None	IDCIO03	12

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	OPRTYP	Type of operation: 1 indicates positioning; 2 indicates alter a partitioned data set directory; 3 indicates return information (UIOINFO).
1 (1)	. 1	OPRPNO	Number of arguments passed to UIOINFO.
2 (2)	. . 1	OPROPT	Option byte.
3 (3)	. . . 1	*	Reserved—contains zeros.
4 (4)	4	OPRICS	Address of input IOCSTR (the data set to be positioned).
8 (8)	4	OPROCS	Address of output IOCSTR (the data set to be positioned).

PUAGL
Post UCB Argument List

The PUAGL is passed when a UMSSUNIT macro is issued with a post request. It defines a request to post a specific UCB.

Created by	Modified by	Used by	Size
Calling routine	IDCSA06	IDCSA06	20

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	PUHEAD	Set with 'POSTUCBb' by IDCSA06.
8(8)	4	PUUCBPTR	Address of the UCB TO BE posted.
12(C)	4	PUTTRPTR	Address of the four-byte TTR to the VTOC to be posted in the UCB.
16(10)	4	PULABELP	Address of the six-byte volume serial number to be posted in the UCB.

PCARG
Print Control Argument List

The Print Control Argument List is used to build a PCT (Print Control Table). This list is an argument of the UESTS macro or the UESTA macro, used to establish a PCT. The list is in a static text module or in storage.

Created by	Modified by	Used by	Size
Calling routine	None	IDCTP04	33

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	PCMTLP	If PCARG is in a static text module, this is an offset from the beginning of the PCARG to a main title line, fully-formatted. If PCARG is in storage, this is the address of a main title line, fully-formatted.
4 (4)	4	PCSTLP	If PCARG is in a static text module, this is an offset from the beginning of the PCARG to one, two, or three contiguous, fully-formatted lines for the subtitle. If PCARG is in storage, this is the address of subtitle lines. The first byte of each line contains the spacing character (0, 1, 2, or 3), and the number of lines is found in PCSTLC.
8 (8)	4	PCFLP	If PCARG is in a static text module, this is an offset from the beginning of the PCARG to one, two, or three contiguous, fully-formatted footing lines. If PCARG is in storage, this is the address of footing lines. The first byte of each line contains the spacing character (0, 1, 2, or 3), and the number of lines is found in PCFLC.
12 (C)	4	PCPCP	If PCARG is in a static text module, this is an offset from the beginning of the PCARG to a 256-byte print chain translate table. If PCARG is in storage, this is the address of a 256-byte print chain translate table.
16 (10)	2	PCPNL	Print column number where the page number field begins.
18 (12)	. . 2	PCPTL	Time field location.
20 (14)	2	PCPDL	Date field location.
22 (16)	. . 2	PCMTLC	Number of lines at PCMTLP.
24 (18)	2	PCSTLC	Number of lines at PCSTLP.
26 (1A)	. . 2	PCFLC	Number of lines at PCFLP.
28 (1C)	2	PCLW	Print line width.
30 (1E)	. . 2	PCPD	Page depth.
32 (20)	1	PCDSC	Default space character, used when space character is not given; invalid, or on overflow. Valid values are 1, 2, or 3.

PCT
Print Control Table

The Print Control Table contains the current page specifications for printing: page width and depth, pointers to heading and footing lines, etc. One PCT, called the *primary* PCT, contains the default values established at processor initialization time. An optional PCT, called the *secondary* PCT, contains page specifications that are unique to a particular FSR, and is cleared between commands. Both PCTs have the same format.

Created by	Modified by	Used by	Size
IDCTP04	IDCTP05 IDCTP01	IDCTP01	112

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	PCTIDN	Identification field: the primary PCT contains "PCT1" in this field; the secondary PCT contains "PCT2".
4 (4)	4	PCTFLG	Action flags:
	1...	PCTH1F	A new header is being entered. This bit is set by IDCTP05 and reset by IDCTP01 as soon as the first header line is printed.
	.1..	PCTH2F	More than one header line is to be saved. This bit is set when the first line is printed by IDCTP01 and reset when the last line has been printed. The count in PCTHLC controls this bit.
	..1.	PCTHAF	A header has been set up.
	...1	PCTLLM	Last line was a message.
	.... 1...	PCTAPF	Alternative print file flag.
8 (8)	4	PCTSPP	Address of secondary PCT. This field is ignored in the secondary PCT.
12 (C)	4	PCTIOC	Address of IOCSTR to be used with UPUT macro.
16 (10)	2	PCTCPN	Current page number on active data set.
18 (12)	. . 2	PCTNLI	Next absolute line number on the current page of active data set.
20 (14)	4	PCTIOS	Address of IOCSTR for SYSPRINT.
24 (18)	2	PCTSPN	Current page number on standard data set.
26 (1A)	. . 2	PCTSNL	Next absolute line number on the current page of standard data set.
28 (1C)	4	PCTIOP	Address of IOCSTR for alternative print data set.
32 (20)	2	PCTAPN	Current page number on alternative data set.
34 (22)	. . 2	PCTANL	Next absolute line number on the current page of alternative data set.
36 (24)	8	PCTSTM	Name of the Static Text module presently in virtual storage.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
44 (2C)	4	PCTSME	Entry point for Static Text module presently in virtual storage.
48 (30)	4	PCTSQP	Address of queue of format structures that are retained until the completion of the function or the issuance of a URESET.
52 (34)	4	PCTAHP	Address of the last header line that was used, needed on an overflow.
56 (38)	4	PCTMLP	Address of main title lines, already fully formatted.
60 (3C)	4	PCTSLP	Address of subtitle lines, already fully formatted.
64 (40)	4	PCTTRP	Address of translate table.
68 (44)	4	PCTPLW	Print line width for the output device.
72 (48)	2	PCTMLC	Number of main title lines.
74 (4A)	. . 2	PCTSLC	Number of subtitle lines.
76 (4C)	4	PCTFLP	Address of footing lines, already fully formatted.
80 (50)	2	PCTFLC	Number of footing lines.
82 (52)	. . 1	PCTHLC	Number of heading lines.
83 (53)	. . . 1	PCTHSC	Total number of lines consumed by the currently active header and the first data line.
84 (54)	2	PCTPNL	Page number location in the main title line.
86 (56)	. . 2	PCTPMN	Signals that this is a message. Before writing a message it contains −1. During writing a message it contains the message number.
88 (58)	2	PCTAPC	"Floating" print column number, used with blank suppression.
90 (5A)	. . 2	PCTPPD	Total number of lines and spaces that may be printed on one page.
92 (5C)	2	PCTDSC	Default space count, used for overflow or in place of an invalid spacing request.
94 (5E)	2	PCTPNI	Page number increment, added to PCTCPN at each page eject.
96 (60)	2	PCTFDL	Absolute line number for the first data line on each page.
98 (62)	. . 2	PCTLDL	Absolute line number of the last data line.
100 (64)	2	PCTFLN	Absolute line number for the first footing line.
102 (66)	. . 2	PCTLNM	Lines in print stack.
104 (68)	4	PCTBUF	Buffer address.
108 (6C)	4	PCTBNL	Address in buffer for next line.

RACFAGL
RACF URACHECK Argument List (VS2.03.824 only)

The RACFAGL defines the URACHECK parameter list and is passed by the caller who issues the URACHECK Umacro to tell IDCSA08 what dataset to RACHECK.

Created by	Modified by	Used by	Size
Calling Routine	None	IDCSA08	20

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	RACFFIG1	Flag byte one.
	...1....	RACFDSTV	Data set type: 0 = NonVSAM data set 1 = VSAM data set.
	.... .1..	RACFLOGF	No logging on RACF failures.
	.... ..1.	RACFLOGN	No logging at all.
	1	RACFCSA	Return profile in storage request.
1 (1)	1	RACFFIG2	Flag byte two.
	1... ...	RACFTALT	Requested alter attribute.
	.... 1...	RACFTCTI	Requested control attribute.
	.... .1..	RACFTUPD	Requested update attribute.
	.... ..1.	RACFTRD	Requested read attribute.
2 (2)	1	RACFFIG3	Flag byte three.
	..1.	RACFPRF	Profile address giver.
3 (3)	1	RACFFIG4	Flag Byte four. Reserved - Contains zeros.
4 (4)	4	RACFENT	Address of data set name.
8 (8)	4	RACFPROF	Address of resource profile in storage. Present when RACFPRF is on.
12 (C)	4	RACFCIN	Address of CLASS name.
16 (10)	4	RACFVOLS	Address of volume serial number.

COMMAREA
Reader/Interpreter Communication Area

The COMMAREA is only used within the Reader/Interpreter for batched jobs to pass information between Reader/Interpreter modules.

Created by	Modified by	Used by	Size
IDCRI01	IDCRI01 IDCRI02 IDCRI03	IDCRI01 IDCRI02 IDCRI03	55

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	RECRDPTR	Address of the beginning of the record currently being scanned.
4 (4)	4	FDTADDR	Address of the primary pointer vector for the FDT.
8 (8)	4	DESCPTR	Address of the Command Descriptor currently being used.
12 (C)	4	WORKPTR	Address of local workarea.
16 (10)	2	RISTATUS	Internal error code for the Reader/Interpreter; set to non-zero if an error is discovered.
18 (12)	2	SCANINDX	Offset into the current record of the last character that was extracted.
20 (14)	2	SCNLIMIT	Location of the final character in the current record that may be scanned.
22 (16)	2	LASTCC	Last processor condition code.
24 (18)	2	MAXCC	Maximum processor condition code.
26 (1A)	8	FSRLNAME	FSR load module name to be invoked if this command is executed.
34 (22)	4	POOLID	Storage area identification code for all space used for the FDT.
38 (26)	8	VERBNAME	Verb from the current input command.
46 (2E)	8	DESCNAME	Module name for the current Command Descriptor.
54 (36)	1	*	Miscellaneous flags:
	1...	GOODCMD	Current command is valid; have Executive invoke the FSR.
	.1..	EOFOK	End of input stream may legitimately occur.
	..1.	OPTSFLAG	Current command came from parameter options specified by the invoker of Access Method Services.
	...1	SCANONLY	Current command is being scanned only for syntax errors.
	.... 1...	SKIPPAST	Current command has just been bypassed.

HDAREA
Reader/Interpreter Historical Area

The Reader/Interpreter Historical Area is created and initialized on the first call to the Reader/Interpreter for batched jobs. It contains information that must be saved across commands, such as input source margins and table locations.

Created by	Modified by	Used by	Size
IDCRI01	IDCRI01 IDCRI02 IDCPM01	IDCRI01 IDCRI02	46

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	LEFTMGN	Leftmost column to use in the input statement. Default to column 2.
2 (2)	. . 2	RIGHTMGN	Rightmost column to use in the input statement. Default to column 72.
4 (4)	4	LOADTPTR	Address of the Command Name table, IDCRILT.
8 (8)	4	KWTBLPTR	Address of modal command verb table, IDCRIKT.
12 (C)	4	ADDRIOCS	Address of IOCSTR for input data set.
16 (10)	1	NESTLVL	IF-THEN nesting level where current command appears.
17 (11)	. 2×*n*	MODLFLGS *n*	Modal flags. A set of modal flags is used for each level of IF-THEN nesting. *n* is the number in NESTLVL.
Each set contains the following:			
	1	NULLDO	Number of unneeded "DO" commands for which no matching "END" commands have been encountered at the current NESTLVL.
	. 1	*	Flags:
	1...	DOFLAG	Current command is part of a "DO" group.
	.1..	THENFLAG	Current commands are associated with a true "IF" condition.
	..1.	ELSEFLAG	Current commands are associated with a false "IF" condition.
	...1	SKIPFLAG	Current commands are to be only checked for proper syntax.

EXWRARG
REPAIRV Argument List

The EXWRARG is passed when UEXCP is issued with a REPAIRV request. It defines a request to repair a 3850 data set.

Created by	Modified by	Used by	Size
Calling routine	IDCIO05	IDCIO05	40

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	EXRWRES	Reserved.
8(8)	4	EXRWCTBL	Address of the open I/O control blocks (IOCTLBLK) that were created in the OPEN call.
12(C)	4	EXRWIOAR	Address of an I/O area: for READCNT and READKD, the read I/O area (RIOAREA) for WRITEREC and FWRITE, the write space area (WIOAREA)
16(10)	4	EXRWDARE	Address of the data read (set when the REPAIRV function's routine exits.)
10(14)	4	EXLOCPTR	Address of the location table, set when the routine exists
24(18)	2	EXCCWCNT	The number of count fields read by READCNT when the routine exits, or The number of data fields read by READKD when the routine exits, or The record number of the record to be operated on by SPACCR. This field is not used by WRITEREC or FWRITE.
26(IA)	5	EXRWCHR	Address of the record to be operated on:
26(IA)	2	EXCC	Cylinder address
28(IC)	2	EXHH	Track address
30(IE)	1	EXRECNUM	Record number
31(IF)	1		Reserved.
32(20)	1	EXRWFUN	For WRITEREC only–the type write operation: 2—WRITE SPECIAL 3—WRITE KEY DATA
32(21)	3		Reserved.
36(24)	4	EXRWKDLN	For SPACE only—address of an area that contains the key length and data length of the record whose count field is to be bypassed. If EXRWKDLN is zero, SPACCR gets the length information from the count field previously read in by READCNT, and saves the length in the location table entry for the requested record.

RCTAGL
Recatalog Argument List

The RCTAGL is passed when a URECAT macro is issued. It defines a request to recatalog a nonVSAM data set.

Created by	Modified by	Used by	Size
Calling routine	IDCSA07	IDCSA07	76

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	RCTHEAD	Set with 'RCTAGLbbb' by IDCSA07.
8(8)	2	RCTOPT	Recatalog options:
	1...	RCTDEV	Indicates that the device type is to be changed. RCTNDEV contains the new device type.
	.1..	RCTVOL	Indicates that the volume serial number is to be changed. RCTNVOL contains the new volume serial number.
10(A)	..4	RCTODEV	Contains the current device type for the data set.
14(E)	..4	RCTNDEV	Contains the new device type for the data set. This field is ignored if RCTDEV is not set.
18(12)	..6	RCTOVOL	Contains the current volume serial number for the data set.
24(18)	6	RCTNVOL	Contains the new volume serial number for the data set. This field is ignored if RCTVOL is not set.
30(1E)	..44	RCTDSET	Contains the data-set name of the data set to be recataloged.

AUTOTBL
Storage Table

The Storage table contains the address for storage areas for modules IDCIO01, IDCIO05, IDCSA02, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10, and IDCTP01. The modules almost always use the same storage instead of getting and freeing storage each time the module gets control.

Created by	Modified by	Used by	Size
IDCSA01	IDCSA02 IDCSA03	IDCSA02 IDCSA03	108

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	108	AUTOARAY	There are 9 sets of the following 4 fields. One set for each module IDCIO01, IDCIO05, IDCSA02, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10, and IDCTP01.
Each set contains the following:			
	4	AREAID	4 byte CSECT identification for the module using this set of 12 bytes. The field contains 'IO01', 'IO05', 'SA02', 'SA06', 'SA07', 'SA08', 'SA09', 'SA10', or 'TP01'.
	2	STATUS	Indicator of the number of storage areas being used for this module. The field contains: 0 – no storage being used. 1 – only the storage area addressed in PTR1 is being used. >1 – the number is a count of storage areas in use for this module. There are more storage areas than are addressed in PTR1.
	. . 2	ASIZE	Number of bytes the module uses.
	4	PTR1	Address of the first storage area for this module.

SELAGL
Selecting a DDname Argument List

The SELAGL is passed when a UMSSUNIT macro is issued with a SELECTDD request. It defines a request to select a DD statement and UCB that can be used for the volume if the volume is already mounted.

Created by	Modified by	Used by	Size
Calling routine	IDCSA06	IDCSA06	23

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	SELHEAD	Set by IDCSA06 with 'SELECTDD'.
8 (8)	4	SELUCBP	Address of the 4-byte area, provided by the caller, in which IDCSA06 returns the address of the UCB associated with the specified volume.
12 (C)	4	SELDDNP	Address of an 8-byte area provided by the caller in which IDCSA06 will return the name of the DD statement that was used in mounting the volume.
16 (10)	6	SELVOL	Volume serial number of the volume used in the DD selection search. This volume serial number is passed by the caller.
22 (16)	1	SELFLAGS	Option flags

SV82LIST
SVC 82 Argument List

The SVC82LIST is passed when SVC 82 is issued. It contains information needed to create a DEB, post a UCB, clear a UCB, or free a DEB.

Created by	Modified by	Used by	Size
Calling routine	None	IGC0008B IGC0408B	16

Creating a DEB

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	SV82CODE	Contains the service request code for the type of service requested. A X'20' for a request to create a DEB.
1(1)	.3	SV82UCBP	Address of the UCB.
4(4)	4	SV82DCBP	DCB address used with the create DEB request.

Posting a UCB

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	SV82CODE	Contains the service request code for the type of service requested. A X'23' for a request to post a UCB.
1(1)	.3	SV82UCBP	Address of the UCB.
4(4)	4	SV82VOLP	Address of the volume serial number to be placed in the UCB on a post UCB request.
8(8)	4	SV82TTRP	Address of the VTOC TTR0 posted to the UCB on a post UCB request.

Clearing a UCB

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	SV82CODE	Contains the service request code for the type of service requested. A X'22' for a request to clear a UCB.
1(1)	.3	SV82UCBP	Address of the UCB.
4(4)	4		* Reserved—contains zeros.
8(8)	4		* Reserved—contains zeros.
12(C)	4	SV82WRKP	Address of a 10-byte area into which the current volume serial and current VTOC TTR0 are returned on the clear UCB request.

Freeing a DEB

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	SV82CODE	Contains the service request code for the type of service requested. A X'21' for a request to free a DEB.
1(1)	.3	SV82UCBP	Address of the UCB.
4(4)	4		* Reserved—contains zeros.
8(8)	4		* Reserved—contains zeros.
12(C)	4	SV82DEBP	DEB address on a free DEB request.

SAHIST
System Adapter Historical Area

The System Adapter's historical area is pointed to by the field GDTSAH. It contains information that is shared between System Adapter modules.

Created by	Modified by	Used by	Size
IDCSA01	IDCSA02 IDCSA03 IDCSA10	IDCIO05 IDCSA02 IDCSA03 IDCSA06 IDCSA10	16 (without **VS2.03.807**) 20 (with **VS2.03.807**)

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	GPFIRST	First UGPOOL storage area pointer.
4 (4)	4	GPLAST	Last UGPOOL storage area pointer.
8 (8)	4	AUTOPTR	Address of AUTOTBL.
12 (C)	4	SAHSTA	Address of ESTAE argument list (STAEPARM).
16(10)	4	LLBLKPTR	Pointer to first Load List Block. **(VS2.03.807)**

TEST Option Data Area

The TEST Option Data Area is used to gather debugging information requested by a PARM command with TRACE, AREAS, or FULL options. The TEST Options Data Area is three tables. The first table, TESTDATA, is present if any PARM command with TRACE, AREAS, or FULL has been executed. The address of TESTDATA is in GDTDBH.

The second table, AREADATA, exists if a PARM command with an AREAS option has been executed. If AREADATA exists, it immediately follows TESTDATA.

The third table, FULLDATA, exists if a PARM command with a FULL option has been executed. If FULLDATA exists, it immediately follows AREADATA, or if AREADATA does not exist, FULLDATA immediately follows TESTDATA.

Created by	Modified by	Used by	Size
IDCPM01	IDCPM01 IDCDB01	IDCPM01 IDCDB01	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
TESTAREA:			
0 (0)	4	AREAPTR	Address of areas identifier table, AREADATA. Zero indicates the table does not exist.
4 (4)	4	FULLPTR	Address of full dump table FULLDATA. Zero indicates the table does not exist.
8 (8)	2..	SNAPID	Number of last full region dump.
10 (A)	..2	TESTRACE	A non-zero value means print the trace tables each time a UDUMP macro is issued. A zero value means print the trace tables only for modules specified in AREAS and FULL options.
AREADATA:			
0 (0)	4	AREAINDX	Number, *j*, of entries in areas identification array. One entry exists for each area identifier specified in the PARM command.
4 (4)	2×*j*	AREADUMP*j*	Areas identifier array containing *j* entries.
Each entry contains the following:			
	2	AREANAME	Two character module identifier where information is gathered. If there is an odd number of area names, two bytes are added to the end of the array.
FULLDATA:			
0 (0)	4	FULLINDX	Number, *k*, of entries in Full Region Dump Array. One entry exists for each full dump.
4 (4)	12×*k*	FULLDUMP*k*	Full Region Dump Array containing *k* entries.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
Each entry contains the following:			
	4	FDUMPID	Four character module identifier where dump is taken.
	2..	FDUMPBEG	Number of the pass through the dump point when dumping is to begin—between 1 and 32,767.
	..2	FDUMPCNT	Number of dumps to take—between 1 and 32,767.
	2..	REALBEG	Current number of passes through this dump point.
	..2	REALCNT	Number of dumps already taken at this dump point.

Text Structure

Text Structures are load modules that contain text (messages and static text items) and format information to use while preparing printed output. This information can be default page dimensions or layout, message text, headings for listings, and similar directions that are used by the Text Processor. There are 22 Text Structure modules, as named in the following table along with the function associated with each. Some FSRs use Text Structures from other FSRs.

IDCTSAL0	ALTER
IDCTSBI0	BUILDINDEX
IDCTSCC0	CNVTCAT
IDCTSCK0	CHKLIST
IDCTSDE0	DEFINE
IDCTSDL0	DELETE
IDCTSEX0	Executive
IDCTSIO0	I/O Adapter
IDCTSLC0	LISTCAT
IDCTSLC1	LISTCAT (messages)
IDCTSLR0	LISTCRA
IDCTSLR1	LISTCRA (messages)
IDCTSMP0	IMPORT/IMPORTRA
IDCTSPR0	PRINT/REPRO
IDCTSRC0	EXPORTRA
IDCTSRI0	Reader/Interpreter for batched jobs
IDCTSRI1	Reader/Interpreter for interactive jobs
IDCTSRS0	RESETCAT (**VS2.03.808 only**)
IDCTSTP0	Text Processor (print chains)
IDCTSTP1	Text Processor (messages)
IDCTSTP6	Text Processor (UERROR messages)
IDCTSUV0	Universal (any module)
IDCTSXP0	EXPORT

A Text Structure consists of an index and text entries. The index is simply a list of halfword displacements from the beginning of the Text Structure to the beginning of the text entry being indexed. The Text Structure identification number is used as the index number. A halfword count of the number of entries precedes the index.

Note: An index entry of -1 indicates that the corresponding text entry is nonexistent.

All text entries contain heading fields and one of the following:

- A format list as described under FMTLIST immediately followed by any static text such as messages referenced by the format list.
- A print control argument list as described under PCARG immediately followed by any static text such as title lines and translate tables referenced by the print control argument list.
- Character code tables which support the GRAPHICS parameter of the PARM command.

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCTP01 IDCTP05	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	INDEX	Number, *n*, of entries in this index.
2 (2)	2×*n*	INDEX*n*	Offset to the appropriate text entry.

Text Entry

The following description shows only the header fields of each text entry. For the remainder of the description, see FMTLIST or PCARG. The text entry begins at offset 2 × *n* + 2 from the beginning of the Text Structure module.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	TXT	Length in bytes of the text entry that follows (not including these header fields).
2 (2)	1	FLG	Flag byte:
	1...		Message entry.
	.1..		Header entry.
	..1.		Secondary message entry.
	...1		Do not transmit this entry to a TSO terminal.
	.... 1...		Secondary message entry.
3 (3)	1	*	Reserved.
The following two fields only exist if this is a text entry for a header line:			
4 (4)	2	HDLI	The number of printable header lines.
6 (6)	2	HDSP	The number of page lines occupied by header lines, intervening blank lines, and the first line of printed data.

UGPOOL Area

When the UGPOOL Umacro is used, an area of storage is allocated to the user and this UGPOOL area is linked into a chain with other areas allocated by UGPOOL. Each such area is preceded by the first 16 bytes, as shown here.

For a page boundary request, the user's allocated area is not preceded by a UGPOOL area. A 24-byte UGPOOL area is linked into the chain and it points to the user's page boundary area.

Created by	Modified by	Used by	Size
IDCSA02	None	IDCSA02	24

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	GPFORWRD	Address of next UGPOOL area.
4 (4)	4	GPBACK	Address of last UGPOOL area.
8 (8)	4	GPLEN	Number of bytes requested plus 16. (For a page boundary request GPLEN = 24.)
12 (C)	4	GPID	Area identification code.
16 (10)	4	GPADRPG	Address of area on a page boundary.
20 (14)	4	GPLENPG	Length of area on a page boundary.

UGSPACE Area

When the UGSPACE Umacro is used, an area of storage is allocated for the user of the Umacro. Each such area is preceded by eight bytes of control information, as shown here.

Created by	Modified by	Used by	Size
IDCSA02	None	IDCSA02	8

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	GSLEN	Number of bytes requested plus 8.
4 (4)	4	GSID	'ƀƀƀ' in first three bytes for UGSPACE area. The last byte is the subpool-ID.

UIOINFO Option Byte and Return Area

The UIOINFO option byte tells IDCIO02 the information desired by the caller who issues a UIOINFO macro.

Created by	Modified by	Used by	Size
Calling routine	None	IDCIO02	1

UIOINFO Option Byte Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	IOINFOPT	Option byte:
	1...	IOINFDVT	Return 8-byte device type.
	.1..	IOINFVOL	Return up to five 6-byte volume serial numbers.
	..1.	IOINFDSN	Return 44-byte data-set name.
	...1	IOINFSUP	Suppress error message.
	.... 1...	IOINFTMS	Return time stamp from the Format 4 DSCB.
	.... .1..	IOINFLUB	not used in VS2.

UIOINFO Return Area Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	4		Header.
			Bytes:
			0-1 Length of entire area (including header).
			2-3 Length of all data returned (including header).

Data returned for each type of information requested is placed consecutively in the work area. The format for the different types of information is shown below:

Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
48		Data set name.
		Bytes:
		0-1 Identifier—X'0001'.
		2-3 Length of data returned.
		4-47 Data set name.
n		Volume serial number list (variable).
		Bytes:
		0-1 Identifier—X'0002'.
		2-3 Length of data returned.
		4-9 First volume serial number.
		. .
		. .
		. .
		n+1-n+6 Last volume serial number.

Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
12		Device type. Bytes: 0-1 Identifier—X'0003'. 2-3 Length of data returned. 4-7 Device type code. 8-11 Maximum block size for device
20		Time stamp. Bytes: 0-1 Identifier—X'0004' 2-3 Length of data returned. 4-11 New time stamp. 12-19 Old time stamp.

UCTAGL
UNCATALOG Argument List

The UCTAGL is passed when a UUNCATLG macro is issued. It defines a request to uncatalog and scratch a nonVSAM data set.

Created by	Modified by	Used by	Size
Calling routine	IDCSA07	IDCSA07	69

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	UCTHEAD	Initialized with 'UCTAGL' by UUNCATLG.
8 (8)	44	UCTDSN	Data set name of nonVSAM data set to be uncataloged and scratched.
52 (34)	4	UCTACBP	Pointer to catalog ACB.
56 (38)	4	UCTVOLP	Pointer to list of volume serial numbers.
60 (3C)	8	UCTDD	ddname of volume on which data set resides.
68 (44)	1 1...	UCTFLAGS UCTVSCAT	UUNCATLG option flags Data set can be uncataloged with a VSAM catalog request. (This bit is always set.)

Unit Table

The unit table is passed by a module that issues a USYSINFO Umacro. IDCSA08 returns in it the information for a unit from the system control blocks.

Created by	Modified by	Used by	Size
Calling routine	IDCSA08	Calling routine	11

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	UNITUCB	Address of the UCB.
4 (4)	2	UNITADR	Channel and unit address:
	1	UNITCHA	Channel address.
	1	UNITUA	Unit address.
6 (6)	..1	UNITSTAT	Status of the unit:
	1...	UNITISHR	Sharable within the system.
	.1..	UNITXSHR	Sharable across systems.
7 (7)	...4	UNITVTOC	Address of the VTOC—TTR0.

UREST Arguments

Any combination of the following structures can be passed to UREST as arguments. The UREST macro changes default items in the Print Control Table. The structures determine which items UREST will change.

Created by	Modified by	Used by	Size
All routines	None	IDCTP01	Variable

PCRST—Change Subtitle Lines

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRSST	Structure identifier; contains 'ST'.
2 (2)	. . 2	PCRSTLC	Number of subtitle lines provided. The maximum is three.
4 (4)	4	PCRSTLP	Address of from one to three contiguous, fully formatted subtitle lines. The number of bytes in each line is the line width plus one for the spacing character. The spacing character is first in each line and must be 1, 2, or 3.

PCRLWS—Change Line Width

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRLWT	Structure identifier; contains 'LW'.
2 (2)	. . 2	PCRLW	New line width in decimal.

PCRPDS—Change Page Depth

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRPDT	Structure identifier; contains 'PD'.
2 (2)	. . 2	PCRPD	New page depth in decimal.

PCRFTS—Change Footing Lines

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRFT	Structure identifier; contains 'FT'.
2 (2)	. . 2	PCRFLC	Number of footing lines provided. The maximum is three.
4 (4)	4	PCRFLP	Address of from one to three contiguous, fully formatted footing lines. The number of bytes in each line is the line width plus one for the spacing character. The spacing character is first in each line and must be 0, 1, 2, or 3.

PCRDSCS—Change Default Spacing Character

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRDSCT	Structure identifier; contains 'SC'.
2 (2)	. . 1	PCRDSC	New default space character. Must be the character 1, 2, or 3.

PCRPCS—Change Translate Table

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRPCT	Structure identifier; contains 'PC'.
2 (2)	. . 2	PCRPCC	If the request is for a print chain provided by Access Method Services, this field contains the characters for the print chain identification as in the GRAPHICS parameter of the PARM command. Otherwise, it contains zero.
4 (4)	4	PCRPCP	Address of a load module name. The load module consists solely of a 256-byte translate table. If the request is for a standard print chain, this field contains zero.

PCRINP—Change Initial Page Number

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	PCRPNT	Structure identifier; contains 'PN'.
2 (2)	. . 2	*	Reserved.
4 (4)	4	PCRPNP	Address of page number field. The first two bytes of the page number field contain the number (from 1 to 4 in binary) of following bytes that contain the page number. The page number is one to four bytes in EBCDIC.

USCRATCH Volume List

The USCRATCH volume list is passed by the caller who issues a USCRATCH Umacro to tell IDCSA08 what data sets to delete.

Created by	Modified by	Used by	Size
Calling routine	None	IDCSA08	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	2	UVOLCNT	The number of arrays that follow (each array identifies a volume on which the data set resides).
Each entry contains:			
	10×*n*	UVOLENT*n*	Volume identifier. The following fields are repeated n times, where n = UVOLCNT.
	4	UVOLDEV	Device type.
	6	UVOLVOL	Volume serial number.

VS1AGL
Volume VTOC Service Argument List

The VS1AGL is passed whenever module IDCVS01 is called. It contains information needed for one of the following functions: Security Checking, Scratching the VTOC, Retrieving VTOC Fields, Updating VTOC Fields, and Recataloging a NonVSAM Data Set.

The parameter list is described separately for each function.

Created by	Modified by	Used by	Size
Calling routine	IDCVS01	IDCVS01	74

Security Checking

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VS1HEAD	Set with 'SECCHECK' by IDCVS01.
8(8)	4	VSUCBP	If a RESERVE is requested, this field must contain the address of a 4-byte area containing either zeros or the UCB address. The VSFILEP field must contain the address of the DD statement that IDCVS01 uses to obtain the UCB address. IDCVS01 fills in the field if the request is successful. If a RESERVE is not requested this field is ignored.
12(C)	4	VSVOLP	Address of a 6-byte volume serial number of the volume to be processed. The serial number must be the one known to the MSC.
16(10)	4	VSFILEP	Address of the 8-byte name of the DD statement to be used to open the VTOC.
20(14)	4	VSIOP	Upon return, this field contains the address of the IOCSTR for the VTOC data set.
24(18)	4	VSNUMATP	Ignored.
28(1C)	4	VSNXTATP	Ignored.
32(20)	4	VSCATP	Address of a 44-byte field containing the name of the VSAM catalog that owns the volume. If VSACBP is specified, this field is ignored. If the catalog name is unknown and VSMASTER is set, this field contains zeros.
36(24)	4	VSACBP	Address of the catalog ACB. If this field is specified the VSCATDDP field is ignored. If the ACB is unknown and VSMASTER is set this field must contain zeros.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
40(28)	4	VSCATDDP	Address of the 8-byte name of the DD statement used to open the VSAM catalog. If VSOPCAT is specified, this field contains the address of a catalog DD field. That field contains the catalog DD name or blanks if the catalog DD name is unknown.
44(2C)	4	VSPSWDP	Address of the 8-byte VSAM catalog password. If VSMASTER is set and the password is not supplied this field is set to zero.
48(30)	4	VSPFILEP	Address of the list of file names used when prompting for passwords. If VSREAD or VSWRITE are set and no file names are specified this field contains zeros.
52(34)	4	VSSERP	Ignored.
56(38)	4	VSDEVP	Ignored.
60(3C)	4	VSSECOPT	Security Checking options:
	1...	VSNOVSAM	Indicates that the volume cannot be owned by a VSAM volume.
	.1..	VSMASTER	Indicates that the master password of the owning VSAM catalog will be verified, and if it fails, all VSAM data set master passwords will be verified.
	..1.	VSNONONV	Indicates that the volume must not contain any nonVSAM data sets.
	...1	VSREAD	Indicates that the read password of each nonVSAM password protected data set will be verified.
	.... 1...	VSWRITE	Indicates that the write password of each nonVSAM password protected data set will be verified.
	.... .1..	VSUCTEST	Indicates that the VSUCAT flag be set upon return indicating whether the volume contains a VSAM user catalog. This flag is interrogated only if VSMASTER is set.
	.... ..1.	VSOPCAT	Indicates that the VSAM catalog will be opened. If this flag is set, VSCATDDP must contain the address of a field containing either blanks or the name of the catalog DD. If IDCVS01 opened the catalog, the catalog DD name is returned in the field set to blanks.
	1	VSUCMAST	Indicates that the master password of the owning VSAM user catalog must be verified.
64(40)	4	VSODEVP	Ignored.
68(44)	1	VSSECOPT	Ignored.
69(45)	.1	VSVTOPT	Ignored.
70(46)	..1	VSCATOPT	Ignored.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
71(47)	...1	VSRETURN	This field is set upon return only if Security Checking was successful:
	1...	VSVSAM	This bit is set to 1 if the volume is owned by a VSAM catalog. It is set to 0 if the volume is not owned by a VSAM catalog.
	.1..	VSUCAT	If VSUCTEST is set, this bit is set to 0 if the VSAM user catalog is not on the volume. It is set to 1 if it is on the volume.
72(48)	1	VSMSG	Message options:
	1...	VSFROMV	Indicates that the words "FROM VOLUME" are put into any error message indicating the FROM volume is in error.
	.1..	VSTOV	Indicates that the words "TO VOLUME" are put into any error message indicating the TO volume is in error.
73(49)	.1	VSRESOPT	Reserve options:
	1...	VSRES	Indicates that the volume is to be reserved prior to reading or updating the VTOC.
	.1..	VSREAL	Indicates that the RESERVE be done on a real volume. If this bit is off, the volume is virtual.

Scratching the VTOC

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VSHEAD	Set with 'SCRVTOCb' by IDCVS01.
8(8)	4	VSUCBP	Same as Security Checking.
12(c)	4	VSVOLP	Same as Security Checking.
16(10)	4	VSFILEP	Same as Security Checking.
20(14)	4	VSIOP	Address of the IOCSTR for the VTOC data set. This field must be specified if the VTOC IS ALREADY OPEN. If the VTOC is not open it contains zeros. Upon return it contains zeros if it is not opened and the IOCSTR address if it is opened.
24(18)	4	VSNUMATP	Ignored.
28(1C)	4	VSNXTATP	Ignored.
32(20)	4	VSCATP	Ignored.
36(24)	4	VSACBP	Ignored.
40(28)	4	VSCATDDP	Ignored.
44(2C)	4	VSPSWDP	Ignored.
48(30)	4	VSPFILEP	Ignored.
52(34)	4	VSSERP	Ignored.
56(38)	4	VSDEVP	This field must contain the address of a 4-byte area containing the device type of the volume whose VTOC is to be scratched.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
60(3C)	4	VSOSERP	Ignored.
64(40)	4	VSODEVP	Ignored.
68(44)	1	VSSECOPT	Ignored.
69(45)	.1	VSVTOPT	Ignored.
70(46)	..1	VSCATOPT	Ignored.
71(47)	...1	VSRETURN	Ignored.
72(48)	1	VSMSG	Same as Security Checking.
73(49)	.1	VSRESOPT	Same as Security Checking.

Retrieving VTOC Fields

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VSHEAD	Set with 'GETVTOCb' by IDCVS01.
8(8)	4	VSUCBP	Same as Security Checking.
12(C)	4	VSVOLP	Same as Security Checking.
16(10)	4	VSFILEP	Same as Security Checking.
20(14)	4	VSIOP	Same as Scratching the VTOC.
24(18)	4	VSNUMATP	If VSALTTRK is set, this field contains the address of a 2-byte fixed field, where the number of alternate tracks is returned by IDCVS01.
28(1C)	4	VSNXTATP	If VSALTTRK is set, this field contains the address of a 4-byte fixed field, where the CCHH of the next alternate track is returned by IDCVS01.
32(20)	4	VSCATP	Ignored.
36(24)	4	VSACBP	Ignored.
40(28)	4	VSCATDDP	Ignored.
44(2C)	4	VSPSWDP	Ignored.
48(30)	4	VSPFILEP	Ignored.
52(34)	4	VSSERP	Ignored.
56(38)	4	VSDEVP	Ignored.
60(3C)	4	VSOSERP	Ignored.
64(40)	4	VSODEVP	Ignored.
68(44)	1	VSSECOPT	Ignored.
69(45)	.1	VSVTOPT	VTOC field options:
	1...	VSTIME	Ignored.
	.1..	VSALTTRK	Indicates that the number of alternate tracks and the address of the next alternate track should be retrieved. VSNUMATP and VSNXTATP must contain the address of the return area.
	..1.	VSVSFLAG	Indicates that the VSAM ownership be returned. The VSVSAM bit in field VSRETURN is set by IDCVS01.
70(46)	..1	VSCATOPT	Ignored.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
71(47)	...1	VSRETURN	This field is modified if the function was successful and VSVSFLAG was set:
	1...	VSVSAM	This bit is set to 1 if the volume is owned by a VSAM catalog. It is set to 0 if the volume is not owned by a VSAM catalog.
	.1..	VSUCAT	Ignored.
72(48)	1	VSMSG	Same as Security Checking.
73(49)	.1	VSRESOPT	Same as Security Checking.

Updating VTOC Fields

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VSHEAD	Set with 'PUTVTOCb' by IDCVS01.
8(8)	4	VSUCBP	Same as Security Checking.
12(C)	4	VSVOLP	Same as Scratching the VTOC.
16(10)	4	VSFILEP	Same as Security Checking.
20(14)	4	VSIOP	Same as Scratching the VTOC.
24(18)	4	VSNUMATP	If VSALTTRK is set, this field contains the address of a 2-byte fixed field which contains the CCHH of the next alternate track.
28(1C)	4	VSNXTATP	If VSALTTRK is set, this field must contain the address of a 4-byte fixed field which contains the CCHH of the next alternate track.
32(20)	4	VSCATP	Ignored.
36(24)	4	VSACBP	Ignored.
40(28)	4	VSCATDDP	Ignored.
44(2C)	4	VSPSWDP	Ignored.
48(30)	4	VSPFILEP	Ignored.
52(34)	4	VSSERP	Ignored.
56(38)	4	VSDEVP	Ignored.
60(3C)	4	VSOSERP	Ignored.
64(40)	4	VSODEVP	Ignored.
68(44)	1	VSSECOPT	Ignored.
69(45)	.1	VSVTOPT	VTOC field options:
	1...	VSTIME	Indicates that the VSAM time stamp is updated with the current time of day.
	.1..	VSALTTRK	Indicates that the number of alternate tracks and the address of the next alternate track should be updated. VSNUMATP and VSNXATP contain the address of the information.
	..1.	VSVSFLAG	Ignored.
70(46)	..1	VSCATOPT	Ignored.
71(47)	...1	VSRETURN	Ignored.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
72(48)	1	VSMSG	Same as Security Checking.
73(49)	.1	VSRESOPT	Same as Security Checking.

Recataloging a NonVSAM Data Set

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VSHEAD	Set with 'RECATLGb' by IDCVS01.
8(8)	4	VSUCBP	Same as Security Checking.
12(C)	4	VSVOLP	Same as Security Checking
16(10)	4	VSFILEP	Same as Security Checking.
20(14)	4	VSIOP	Same as Scratching the VTOC.
24(18)	4	VSNUMATP	Ignored.
28(1C)	4	VSNXTATP	Ignored.
32(20)	4	VSCATP	Ignored
36(24)	4	VSACBP	Ignored.
40(28)	4	VSCATDDP	Ignored.
44(2C)	4	VSPSWDP	Ignored.
48(30	4	VSPFILEP	Ignored.
52(34)	4	VSSERP	If the VSSERIAL bit is set, this field must contain the address of a 6-byte character field that contains the new volume serial number.
56(38)	4	VSDEVP	If VSDEVICE is set, this field must contain the address of the new 4-byte device code obtained from the UCBTYP field.
60(3C)	4	VSOSERP	Contains the address of the current 6-byte volume serial number.
64(40)	4	VSODEVP	Contains the address of the current 4-byte device code obtained from the UCBTYP field.
68(44)	1	VSSECOPT	Ignored.
69(45)	.1	VSVTOPT	Ignored.
70(46)	..1	VSCATOPT	Recatalog options:
	1...	VSDEVICE	Indicates that the device type is changed. If this bit is set VSDEVP must contain the address of the new device type.
	.1..	VSSERIAL	Indicates that the serial number is changed. If this bit is set VSSERP must contain the address of the new serial number.
	..1.	VSLIST	Indicates that the names of all data sets successfully recataloged be listed.
71(47)	...1	VSRETURN	Ignored.
72(48)	1	VSMSG	Same as Security Checking.
73(49)	.1	VSRESOPT	Same as Security Checking.

VS2AGL
Volume Label Service Argument List

The VS2AGL argument list is passed whenever IDCVS02 is called. It contains information needed for one of the following functions:

- The INITVOL function defines a request to have a mass storage volume intialized for use by the operating system.
- The GETLABEL function defines a request to retrieve one or more of the following fields from the volume label: owner or volume serial number.
- The PUTLABEL function defines a request to update one of the following fields in the volume label: owner or volume serial number.

The parameter list is described separately for each function.

Created by	Modified by	Used by	Size
Calling routine	IDCVS02	IDCVS02	45

Initializing Mass Storage Volumes

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VS2HEAD	Set with 'INITVOLb' by IDCVS02.
8(8)	4	VSUCBPTR	Address of the UCB.
12(C)	4	VSVTOCSZ	Number of tracks for the e VTOC.
16(10)	4	VSOWNPTR	Address of the owner name to be placed in the volume label.
20(14)	4	VSVOLPTR	Address of the volume serial number to be placed in the volume label.
24(18)	4	VSIOBKPT	Address of the area in which the I/O control block address is returned. If I/O control block address is nonzero when control is returned, the UEXCP (CLOSE) macro must be issued to close the DCB. The I/O control block address must be initialized to zero before a request.
28(1C)	1	VSLABOPT	Ignored.
29(1D)	.6	VSVOLUME	Ignored.
35(23)	...1	VS2MSG	Ignored.
36(24)	1	VS2RESOP	Ignored.
37(25)	.8	VSDDNAME	DD name for the volume.

Retrieving Volume Label Fields

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VS2HEAD	Set with 'GETLABEL' by IDCVS02.
8(8)	4	VSUCBPTR	Address of a 4-byte area containing the UCB address, or zeros if the UCB address is unknown. If the UCB address is zero, the UCB address will be put in the 4-byte area upon return.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
12(C)	4	VSVTOCSZ	Ignored.
16(10)	4	VSOWNPTR	Address of the area for the owner name to be returned if VSOWNER is set.
20(14)	4	VSVOLPTR	Address of an area for the volume serial number to be returned if VSVOLSER is set.
24(18)	4	VSIOBKPT	Ignored.
28(1C)	1	VSLABOPT	Label options:
	1..	VSOWNER	Indicates that the owner name is to be returned.
	.1..	VSVOLSER	Indicates that the serial number is to be returned.
29(1D)	.6	VSVOLUME	Volume serial number used when opening the VTOC for label processing. It must be the serial number known to the MSC.
35(23)	...1	VS2MSG	Ignored.
36(24)	1	VS2RESOP	Reserve options:
	1...	VS2RES	Indicates that the volume is to be reserved prior to reading or updating the label.
	.1..	VS2REAL	Indicates that the RESERVE is done on a real volume. If this bit is off, the volume is virtual.
37(25)	.8	VSDDNAME	DD name for the volume.

Updating Volume Label Fields

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VS2HEAD	Set with 'PUTLABEL' by IDCVS02.
8(8)	4	VSUCBPTR	Same as Retrieving Volume Label Fields.
12(C)	4	VSVTOCSZ	Ignored.
16(10)	4	VSOWNPTR	Address of owner name to be put in the volume if VSOWNER is set.
20(14)	4	VSVOLPTR	Address of the volume serial number to be placed in the volume label if VSVOLSER is set.
24(18)	4	VSIOBKPT	Ignored.
38(1C)	1	VSLABOPT	Label options:
	1...	VSOWNER	Indicates that the owner is to be updated.
	.1..	VSVOLSER	Indicates that the serial number is updated.
29(1D)	.6	VSVOLUME	Same as Retrieving Volume Label Fields.
35(23)	...1	VS2MSG	Message options:
	1...	VS2TOV	Indicates that error messages contain the words "TO VOLUME".

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
36(24)	1	VS2RESOP	Same as Retrieving Volume Label Fields.
37(25)	.8	VSDDNAME	DD name for the volume.

VS3AGL
Volume Data Set Service Argument List

The VS3AGL is passed whenever IDCVS02 is called. It contains information to perform a SELECTDS function. This function searches the VTOC for specific data sets. The data set names selected in accordance with the caller's criteria are returned in a buffer.

Created by	Modified by	Used by	Size
Calling routine	IDCSV03	IDCSV03	59

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	8	VS3HEAD	Set with 'SELECTDS' by IDCVS03.
8(8)	4	VS3LISTP	Pointer to the data set name array.
12(C)	4	VS3EXDSP	Pointer to an array of data sets to be excluded from the selection search.
16(10)	4	VS3LEVP	Pointer to data set name qualifier.
20(14)	2	VS3LEVLN	Length of qualifier.
22(16)	2	VS3AST	Displacement to asterisk within the qualifier pointed to in VS3LEVP.
24(18)	4	VS3EXPIR	Pointer to expiration date.
28(1C)	4	VS3CREAT	Pointer to creation date.
32(20)	4	VS3UCBP	Pointer to the UCB.
36(24)	4	VS3DEV	Device type of volume.
40(28)	4	VS3POOL	Storage pool ID for data set list (VSDBLOCK).
44(2C)	6	VS3VOL	Volume serial number.
50(32)	8	VS3DD	ddname for opening the VTOC.
58(3A)	1	VS3FLAGS	Flags.
	1...	VS3STAT	Return only VSDSTAT portion of VSDBLOCK. This bit is mutually exclusive with VS3USAGE and VS3SCR.
	.1..	VS3USAGE	Return only VSDUSAGE portion of VSDBLOCK. This bit is mutually exclusive with VS3STAT and VS3SCR.
	..1.	VS3SCR	Return only VSDSCR portion of VSDBLOCK. This bit is mutually exclusive with VS3STAT and VS3USAGE.
	...1	VS3UNCAT	Return only data sets not cataloged.
	.... 1...	VS3SYSNM	Return only data sets with system generated names.

DIAGNOSTIC AIDS

This chapter explains the diagnostic aids provided for Access Method Services,

explains how to find key areas in a dump, and offers suggestions for isolating different types of problems. Before attempting to diagnose a problem with the aids in this chapter, you should be familiar with *OS/VS2 System Programming Library: Debugging Handbook*. This and other publications that may be helpful are listed in the preface to this book.

Several large figures referred to throughout this chapter are located at the end of the chapter.

Four major diagnostic aids are provided by the processor:

- Trace tables, which provide a trace of the flow of control between modules and within modules.
- Dump points, which provide the facility to dump selected areas of virtual storage and take a full region dump.
- The Test option, which you can set to print out the trace tables or to obtain dumps at selected points if Access Method Services is invoked with a batched job.
- ABORT codes and full region dumps, which are produced when the processor detects an unrecoverable condition.

Trace Tables

The processor maintains two trace tables during each execution: the Inter-Module Trace Table, which records the flow of control *between* modules, and the Intra-Module Trace Table, which records the flow of control *within* modules.

You can find the trace tables in any full region dump, you can print them using the Test option, or you can display them on a TSO terminal. The section "Reading a Dump" explains how to find the tables in a dump; the section "The Test Option" explains how to print them; the section "The TSO TEST Command" explains how to display them on a TSO terminal.

Inter-Module Trace Table

The Inter-Module Trace Table begins with the characters INTER and contains the IDs of the last twenty modules that had control. The module IDs are the last four characters of the module name. For example, if the trace looks like this:

```
INTER ... SA01 EX01 RI01 RI02
```

then you know that IDCRI02 had control at the time of the dump.

The Inter-Module Trace Table is updated by the System Adapter not only as each module is entered, but also upon return from a module. Thus, if RI01 calls TP01 which calls IO01 and then returns back to RI01, the trace table looks like this:

```
INTER ... RI01 TP01 IO01 TP01 RI01
```

Intra-Module Trace Table

The Intra-Module Trace Table begins with the characters INTRA and contains the last twenty trace points encountered within modules. Each module has trace points placed at key locations, for example, at the start of procedures and around calls to other modules.

The IDs of the trace points consist of four characters: the first two characters are the mnemonic identifier of the module being traced, and the last two characters identify a specific point within the module. (The mnemonic identifiers are listed in the section "Naming Conventions" in the chapter "Introduction."

The section "Trace and Dump Points to Module Cross Reference" in this chapter contains a list of all the trace points, identifies the module and procedure in which the trace point occurs, and explains the situation at the trace point. For example, if the Intra-Module Trace Table looks like this:

```
INTRA ... SAGS IOOP SACL SAGP
```

then, using this list, you would know that the last trace point encountered was at the start of the routine in module IDCSA02 that processes a UGPOOL macro request.

During the time the Test option is on, the dumping routine (IDCDB01) places dump points in the Intra-Module Trace Table; thus, the trace table contains all the dump points encountered as well as the trace points. All the dump points you may find in the Intra-Module Trace Table, in addition to the trace points, are explained in the section "Trace and Dump Points to Module Cross Reference" in this chapter.

Trace points within a module can be found by examining the microfiche listings for occurrences of the UTRACE macro; the UTRACE macro sets the trace IDs into the trace table. The expansion of the UTRACE macro for trace ID DLLC looks like this:

```
OLDERID2 = NEWERID2;
NEWID2 = 'DLLC';
```

Dump Points

Each module has built-in dump points that invoke diagnostic dumping routines if the Test option is in effect. The dump points, set up by the UDUMP macro, have been placed at key locations in each module (for example, around calls to other processor and non-processor modules). Each dump point specifies the information that can be dumped at that point. Some dump points allow symbolic dumping of selected areas of virtual storage (for example, parameter lists or return codes); all dump points allow dumping of the full region and printing of the trace tables.

Dump points can be found by examining microfiche listings for occurrences of the UDUMP macro. The expansion of the UDUMP macro for the dump point DLVL looks like this:

```
IF GDTDBG = NULLPTR
  THEN;
  ELSE
     CALL IDCB010(GDTTBL,'DLVL');
```

Only the trace tables and the full region can be dumped at this point because only two parameters, the GDTTBL and the dump ID, are passed to the dumping routine.

The section "Module to Dump Points Cross Reference" in this chapter contains a list of all the dump points within each module, indicates what information can be dumped and explains the situation at the dump point. The section "Test Option" in this chapter explains how to take a full region dump.

Dumping Selected Areas of Virtual Storage

Certain Access Method Services modules have the dumping of selected areas of virtual storage built in. Dumping of these selected areas occurs at a dump point as described above. The areas dumped vary with each dump point and are identified with descriptive codes. The list in the section "Module to Dump Points Cross Reference" in this chapter indicates which modules contain dumps of selected areas and the footnotes to that list describe the areas dumped.

Dump points at which selected areas are printed can be found by examining the microfiche listings for occurrences of the UDUMP macro. The expansion is as described above for a full region dump except that the address of a parameter list describing the areas to be dumped is passed to the dumping routing as a third parameter.

Dumping of selected areas can occur with or without a full region dump in addition, as described in the section "Test Option" in this chapter.

Test Option

If you invoked Access Method Services in a batched job, you can use the Test option to activate the printing of diagnostic output at selected points within Access Method Services. The Test option is controlled by the TEST keyword as explained in the following section "TEST Keyword".

The Test option provides you with the ability to print:

- The Inter-Module and Intra-Module Trace Tables. The format and interpretation of these tables are described in the section "Trace Tables" in this chapter.
- Selected areas of virtual storage. The facility for dumping selected areas of virtual storage is described in the section "Dump Points" in this chapter.
- Full region dump. The facility for taking a full region dump is described in the section "Dump Points" in this chapter.

Each variation of the Test option provides an additional level of information. The possible variations are: (1) print the trace tables only; (2) print the trace tables and selected areas of virtual storage; (3) print the trace tables and selected areas of virtual storage and take a full region dump.

TEST Keyword

You can enter the TEST keyword either in the PARM field of the EXEC card that invokes the processor, or on a PARM command. By using the PARM command, you can turn the Test option on and off or change the Test option for different function commands.

The format of the TEST keyword and its subparameters is:

PARM TEST({[TRACE]
[AREAS(*ID-list* **)...)]**
[FULL((*dumplist* **)...)]|**
[OFF]})

where the subparameters are defined as follows:

TRACE specifies that the inter-module and intra-module trace tables are to be printed at every dump point encountered.

AREAS names the modules for which selected areas are to be printed, *in addition* to the trace tables. The trace tables are printed at each dump point encountered within the named modules; if a dump point specifies selected areas to be dumped, these areas are printed also. *ID-list* is a string of two-character mnemonic identifiers separated by commands and/or blanks. The mnemonic identifiers are listed in the section "Naming Conventions" in the chapter "Introduction". The mnemonic identifier, however, for the dump points within System Adapter dump points is ZZ. The maximum number of identifiers is 10. For example, AREAS(EX,PR) specifies that selective dumping is to occur in the Executive modules and the PRINT FSR.

FULL names the dump points at which full region dumps are to be produced, *in addition* to the selected areas and the trace tables. The trace tables and selected areas are produced each time the dump point is encountered; a full region dump is produced as specified in *dumplist*. *dumplist* consists of a string of triplets enclosed in parentheses. The maximum number of triplets is 10. Each triplet is of the form:

(*ident* [*begin* [*count*]])

where the arguments of the triplet are defined as follows:

ident is a four-character dump point. The dump points are identified in UDUMP macros and are listed in the module to Dump Points Cross Reference list.

begin specifies the iteration through the named dump point at which you wish the full region dump to be produced. For example, a *begin* value of 2 specifies that a full region dump is not to be produced until the second encounter of the dump point. The default value is 1, and the maximum is 32,767.

count specifies the number of times the full region dump is to be produced, once the value of *begin* has been satisfied. The default value is 1, and the maximum is 32,767.

For example, FULL((EX1F,4,2),(AL01)) specifies that one full region dump is to be produced the fourth time that point EX1F is encountered, another full region dump is to be produced the fifth time the point is encountered, and one full region dump is to be produced the first time that point AL01 is encountered. Trace tables and any selected areas are to be printed each time dump points EXIF and AL01 is encountered. If the FULL keyword is used, then an AMSDUMP DD statement must be provided. for example:

```
//AMSDUMP DD SYSOUT=A
```

OFF turns off the Test option. No further dumping of trace tables, selected areas, or region will occur until another PARM command specifies one of the other subparameters. This subparameter must occur alone; it may not be coded with any other subparameter of the TEST keyword.

Each time a PARM command is specified, the TEST parameters override the TEST parameters in effect from the previous PARM command.

Figure 14 shows a section of the output from the command:

```
PARM TEST ( FULL (LCTP,2,1) )
```

The trace tables and the selected area, DARGLIST, are printed each time the dump point LCTP is encountered. A full region dump is produced the second time that dump point LCTP is encountered.

How to Use the Test Option

If a problem occurs and you have no idea which modules are involved, run the job again with the TRACE keyword. From the Inter-Module Trace Table you should be able to tell the modules involved. The TRACE keyword, however, produces a large amount of output.

If you suspect which modules are involved, you can rerun the job with the AREAS keyword and specify the identifiers of several suspected modules. You will obtain trace output for only the specified modules.

Once you know the procedure within a module that has caused the problem, select the dump points at which you would like a full dump (using the Module to Dump Points Cross Reference list or by examining the microfiche for dump points), and rerun the job with the FULL keyword. The AREAS and FULL keywords can be used in combination to obtain trace tables and selected areas throughout several modules, but a full region dump only at selected points.

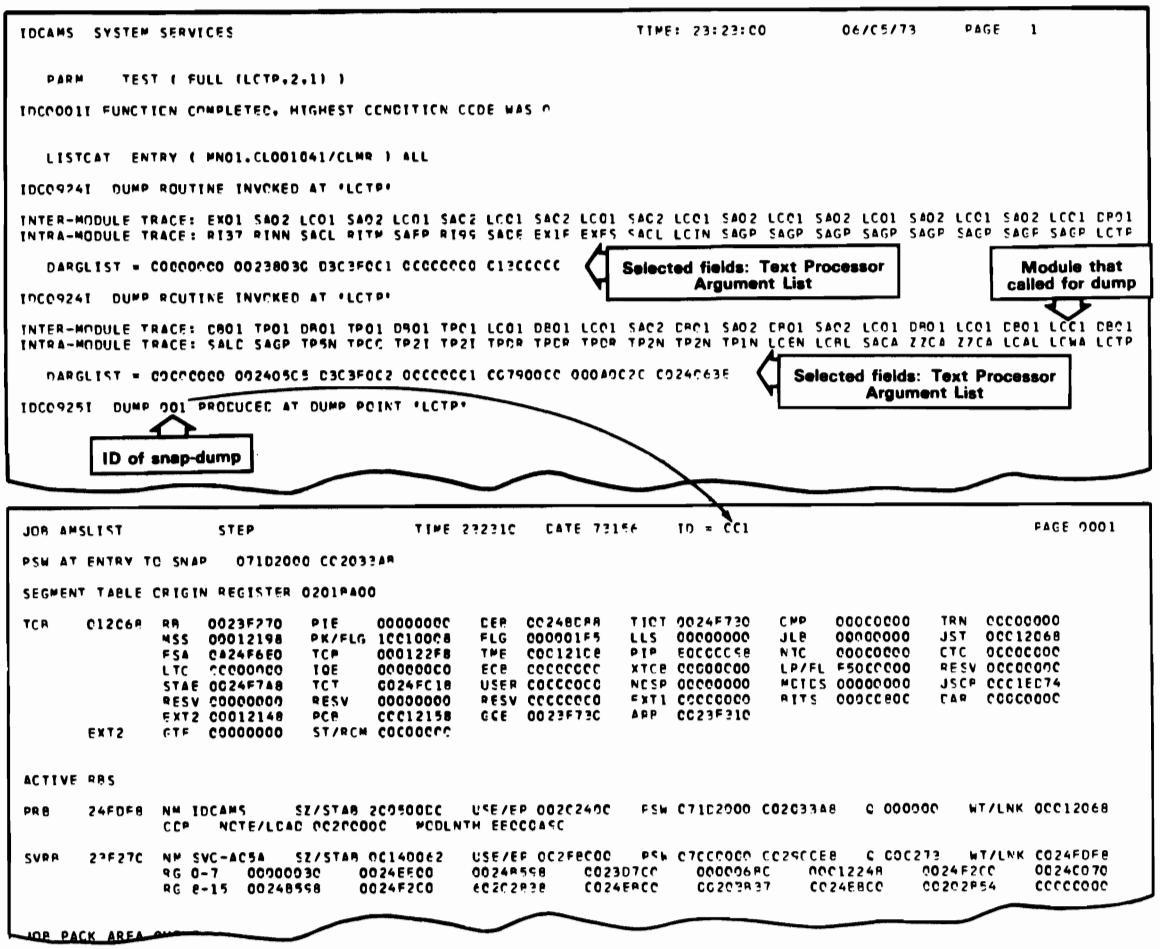


Figure 14. Example of Test Option Output

Trace and Dump Points to Module Cross Reference

The following list contains all trace and dump points, identifies the containing module and procedure and explains the situation at the trace or dump point. When the test option is set, both the trace and dump points are placed in the Intra-Module Trace Table. The trace tables are printed with all variations of the Test option as explained in the section "TEST Keyword."

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
AA00	IDCSA09	ISSUEMAC	dump	After issuing SVC 126.
AA01	IDCSA09	CHKCODE	dump	After delayed response for Mass Storage Control order.
ALMR	IDCAL01	MEMRENAM	trace	Start of procedure that renames members of partitioned data sets.
AL01	IDCAL01	IDCAL01	dump	Before calling the catalog to alter an object.
			trace	Start of ALTER FSR.
AL02	IDCAL01	IDCAL01	dump	End of ALTER FSR.
AL03	IDCAL01	LOCATPRC	dump	After calling the catalog to locate an object.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
AL04	IDCAL01	IDCAL01	dump	Before issuing ALTER request for index object if KEYS specified.
AL31	IDCAL01	LOCATPRC	trace	Start of procedure that locates the entry to be altered.
AL41	IDCAL01	ALTERPRC	trace	Start of procedure that alters the catalog entry.
AL51	IDCAL01	CHECKPRC	trace	Entry to CHECKPRC.
			dump	After locating data component of the alternate index for which UPGRADE has been specified.
AL52	IDCAL01	CHECKPRC	dump	After locating associated cluster or alternate index of the data object specified on ALTER command.
AL53	IDCAL01	CHECKPRC	dump	After locating associated index component.
AL54	IDCAL01	CHECKPRC	dump	After locating the data component of the path's base cluster.
AL55	IDCAL01	CHECKPRC	dump	After locating the cluster component of the alternate index's base cluster.
AL56	IDCAL01	CHECKPRC	dump	After locating the data component of the alternate index's base cluster.
AL61	IDCAL01	INDEXPRC	trace	On entry to INDEXPRC.
AL71	IDCAL01	DALCPROC	trace	Start of procedure that allocates and deallocates volumes.
AL81	IDCAL01	PARAMCHK	trace	On entry to parameter checking procedure.
ALMR	IDCAL01	MEMRENAM	trace	Start of procedure that renames PDS members.
BIB1	IDCBI01	BLDPROC	trace	First entry to procedure that builds and writes the alternate index records.
BIC1	IDCBI01	CNTLPROC	trace	Start of procedure that controls reading base cluster, sorting and writing alternate index.
BIC2	IDCBI01	CNTLPROC	dump	After completion of sort if an internal sort; after completion of sort phase and before merge passes if an external sort.
BIDL	IDCBI01	DELTPROC	trace	Start of procedure that deletes sort work files.
			dump	After return from UCATLG to delete each sort work file.
BID1	IDCBI01	DEFPROC	trace	Start of procedure that defines sort work files.
BID2	IDCBI01	DEFPROC	dump	After return from UCATLG to define each sort work file.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
BIF1	IDCBI01	FINPROC	trace	Start of procedure that closes alternate index and prints status message.
BII1	IDCBI01	INITPROC	trace	Start of procedure that obtains resources for building alternate index.
BII2	IDCBI01	INITPROC	dump	After obtaining or failing to obtain sort core.
BIJ1	IDCBI01	JCPROC	trace	Start of procedure that issues UIOINFO to obtain sort work file job control data.
BIJ2	IDCBI01	JCPROC	dump	After return from each call to UIOINFO.
BIL1	IDCBI01	LOCPROC	trace	Start of procedure that controls catalog locates to obtain information about the base cluster and alternate index.
BIL2	IDCBI01	CATPROC	dump	After return from UCATLG for each locate request.
BIM1	IDCBI01	MERGPROC	trace	Start of procedure that performs the merge passes of an external sort.
BIM2	IDCBI01	MERGPROC	trace	Start of each merge pass of an external sort.
BIM3	IDCBI01	MERGPROC	dump	After the tree of nodes has been initialized for each merge pass of an external sort.
BIM4	IDCBI01	MERGPROC	dump	After processing one set of strings during the merge pass of an external sort.
BIP1	IDCBI01	OPENPROC	trace	Start of procedure that opens data sets.
BIP2	IDCBI01	OPENPROC	dump	After return from UOPEN to open a data set.
BISP	IDCBI01	SPILPROC	trace	Start of procedure that writes out a sorted string in the sort phase of an external sort.
BISR	IDCBI01	SORTPROC	dump	Before sorting the records in the record sort area.
BI01	IDCBI01	IDCBI01	trace	Start of BLDINDEX FSR.
BI02	IDCBI01	MAINPROC	trace	Start of procedure that controls building of one alternate index.
BI03	IDCBI01	MAINPROC	dump	After return from procedure which locates information about the base cluster and alternate index.
BI04	IDCBI01	MAINPROC	dump	After the alternate index has been built; before close.
CCAC	IDCCC01	ALTRLSTS	dump	After building parameter lists to alter a VSAM catalog.
CCAL	IDCCC01	ALTRLSTS	trace	Before building parameter lists to alter a VSAM catalog.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
CCAP	IDCCC01	AEPROC	trace	Ready to process an 'AE' catalog entry from an OS/VS catalog.
CCAT	IDCCC01	CATALOG	trace	Ready to build VSAM parameter lists and alter the nonVSAM entry in the VSAM catalog.
CCAV	IDCCC01	ALTRLSTS	dump	After building parameter lists to alter a VSAM catalog.
CCCE	IDCCC01	CONTINUE	trace	Ready to get a block at the same index level as the last block from an OS/VS catalog.
CCCI	IDCCC01	CNVTINIT	trace	Ready to initialize for scanning OS/VS catalog.
CCCN	IDCCC01	CONTINUE	dump	After obtained block at the same index level as the last block from an OS/VS catalog.
CCCP	IDCCC01	CVPEPROC	trace	Ready to process a 'CVPE' catalog entry from an OS/VS catalog.
CCCT	IDCCC01	CATALOG	trace	Ready to build VSAM argument lists and define converted OS/VS catalog entry in the VSAM catalog.
CCCV	IDCCC01	CONVERT	trace	Ready to convert OS/VS catalog entries to VSAM catalog entries.
CCDC	IDCCC01	DEFNLSTS	dump	After building VSAM argument lists for define.
CCDL	IDCCC01	DEFNLSTS	trace	Ready to build VSAM argument lists for define.
CCDP	IDCCC01	DSPEPROC	trace	Ready to process a 'DSPE' catalog entry from an OS/VS catalog.
CCDV	IDCCC01	DEFNLSTS	dump	After building VSAM argument lists for define.
CCER	IDCCC01	ERRPROC	trace	Start of procedure that invokes UERROR when a catalog error (SVC26) is encountered.
CCGP	IDCCC01	GIPEPROC	trace	Ready to process a 'GIPE' catalog entry from an OS/VS catalog.
CCIC	IDCCC01	CNVINIT	dump	After initializing for scanning of OS/VS catalog.
CCIE	IDCCC01	PUSHDOWN	dump	No 'ICE' entry found on a lower level scanning in the OS/VS catalog.
CCIP	IDCCC01	ILEPROC	trace	Ready to process a 'ILE' catalog entry from an OS/VS catalog.
CCLC	IDCCC01	LOCTLSTS	dump	After building parameter lists to locate information in the VSAM catalog.
CCLL	IDCCC01	LOCTLSTS	trace	Before building parameter lists to locate information in the VSAM catalog.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
CCLT	IDCCC01	CATALOG	trace	Ready to build VSAM argument lists to locate information in the VSAM catalog about a duplicate entry.
CCLV	IDCCC01	CATALOG	dump	After VSAM returns volume information on duplicate entry.
CCPD	IDCCC01	PUSHDOWN	dump	After obtaining block at the same index level as the last block from the OS/VS catalog.
CCPH	IDCCC01	PUSHDOWN	trace	Ready to obtain a block at a lower index level than the last block from the OS/VS catalog.
CCPP	IDCCC01	POPUP	trace	Ready to return to higher level index block in OS/VS catalog.
CCPU	IDCCC01	POPUP	dump	Ready to continue scan of higher level index block in OS/VS catalog.
CCSE	IDCCC01	CONVERT	dump	Invalid entry type in the OS/VS catalog.
CCSL	IDCCC01	CONVERT	dump	Finished converting OS/VS catalog entries to VSAM catalog entires.
CCVE	IDCCC01	VOLINDEX	dump	No 'VICE' entry in the OS/VS catalog.
CCVI	IDCCC01	VOLINDEX	dump	First block of volume index is obtained from the OS/VS catalog.
CCVP	IDCCC01	VCBPPROC	trace	Ready to process a 'VCBPE' and 'VCB' entry from the OS/VS catalog.
CCVX	IDCCC01	VOLINDEX	trace	Ready to get first volume index block from the OS/VS catalog.
CKBD	IDCCK01	BUILDTAB	trace	Start of procedure that builds the checkid table.
CKCK	IDCCK01	IDCCK01	trace	Start of CHKLIST FSR.
CKCP	IDCCK01	CHRPROC	trace	Start of procedure that processes CHR records.
CKDI	IDCCK01	DSDRLIST	trace	Start of procedure that prints the volume serial numbers.
CKDP	IDCCK01	DSDRPROC	trace	Start of procedure that processes DSDR records.
CKDV	IDCCK01	DSDRVOLS	trace	Start of procedure that processes type 2 DSDR records.
CKGC	IDCCK01	GETCHR	trace	Start of procedure that reads CHR records.
CKGN	IDCCK01	GETNEXT	trace	Start of procedure that locates the next logical DSDR record.
CKHS	IDCCK01	HSKGPROC	trace	Start of housekeeping procedure.
CK10	IDCCK01	IDCCK01	dump	Before calling CHR procedure.
CK20	IDCCK01	IDCCK01	dump	Upon return from CHR procedure.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
CK30	IDCCK01	DSDRPROC	dump	After reading a type 1 DSDR record.
CK40	IDCCK01	DSDRPROC	dump	Before processing a type 1 DSDR record for a tape data set.
CK50	IDCCK01	DSDRVOLS	dump	Before processing a type 2 DSDR record.
CP14	IDCRP01	VERIFYC	trace	When either the source or target catalog cannot be verified during a reload.
DB2A	IDCDB02	ARRAYHDR	trace	Start of procedure that processes an array header dump element.
DB2B	IDCDB02	BCONVERT	trace	Start of procedure the converts a dump item to binary representation.
DB2C	IDCDB02	CCONVERT	trace	Start of procedure that converts a dump item to character representation.
DB2F	IDCDB02	FCONVERT	trace	Start of procedure that converts a dump item to fixed representation.
DB2H	IDCDB02	HCONVERT	trace	Start of procedure that converts a dump item to hex representation.
DB2I	IDCDB02	ITEMDUMP	trace	Start of procedure that processes an individual dump list element.
DB2N	IDCDB02	NAMEFLD	trace	Start of procedure that processes the dump element symbolic name.
DE01	IDCDE01	IDCDE01	dump	Before calling the catalog to define an object.
DE02	IDCDE01	IDCDE01	dump	End of DEFINE FSR, before completion message is issued.
DE03	IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	dump	After calling the catalog to locate a model object.
DE04	IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	dump	End of procedure that built the model table.
DE11	IDCDE01	IDCDE01	trace	Start of DEFINE FSR.
DE20	IDCDE03 (with **VS2.03.807**)	IDCDE03	trace	Entry to IDCDE03.
DE21	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	CTLGPROC	trace	Start of procedure that defines a master or user catalog.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
DE22	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	DSETPROC	trace	Start of procedure that defines a
DE23	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	DSPACPRC	trace	Start of procedure that defines a data space.
DE24	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	NVSAMPRC	trace	Start of procedure that defines a nonVSAM data set.
DE25	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	AIXPROC	trace	Start of procedure that defines an alternate index.
DE26	IDCDE01 (without **VS2.03.807**) IDCDE03 (with **VS2.03.807**)	PATHPROC	trace	Start of procedure that defines a path
DE27	IDCDE01	DALCPROC	trace	Start of procedure that allocates and deallocates volumes.
DE30	IDCDE02	IDCDE02	trace	Entry to IDCDE02.
DE31	IDCDE02	NAMEPROC	trace	Start of procedure that builds CTGFLs with name, date, and exception exit information.
DE32	IDCDE02	ALLCPROC	trace	Start of procedure that builds CTGFLs for allocation information.
DE33	IDCDE02	KEYPROC	trace	Start of procedure that builds CTGFLs for key range and AMDSBCAT information.
DE34	IDCDE02	PROTPROC	trace	Start of procedure that builds CTGFLs for protection and RGATTR information.
DE35	IDCDE02	IXOPPROC	trace	Start of procedure that initializes index fields in the AMDSBCAT.
DE36	IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	trace	Start of procedure that locates the model object entry.
DE37	IDCDE02	FREESTG	dump	End of IDCDE02 CSECT

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
DLBC	IDCDL01	BUILDCPL	trace	Start of procedure that builds the CTGPL for the delete request.
DLBG	IDCDL01	IDCDL01	dump	Start of DELETE FSR.
DLCT	IDCDL01	CATCALL	trace	Start of procedure that calls the catalog with a delete request.
DLLC	IDCDL01	FINDTYPE	trace	Start of procedure that locates the type of the entry to be deleted.
DLMS	IDCDL01	MORESP	trace	Entry to MORESP.
DLPC	IDCDL01	PARAMCHK	trace	Start of procedure that checks for invalid parameters.
DLLB	IDCDL01	FINDTYPE	dump	Before calling the catalog to locate the entry type.
DLLA	IDCDL01	FINDTYPE	dump	After calling the catalog to locate the entry type.
DLDB	IDCDL01	CATCALL	dump	Before calling the catalog to delete an entry.
DLDA	IDCDL01	CATCALL	dump	After calling the catalog to delete an entry.
DLMD	IDCDL01	MEMDLETE	trace	Start of procedure that deletes PDS members.
DLAL	IDCDL01	ALLOPROC	trace	Start of procedure that dynamically allocates a data set or volumes.
DLAC	IDCDL01	RC240PRC	trace	Start of procedure that processes a VSAM catalog return code of 240 when the CAT parameter is coded.
EXFS	IDCEX01	CALLFSR	dump	Before each call to an FSR.
EXIF	IDCEX01	CALLFSR	trace	Before each call to an FSR.
EXIM	IDCEX01	MAIN	trace	Before calling the Reader/Interpreter for the first time.
EXIR	IDCEX01	CALLRI	trace	Before each call to the Reader/Interpreter.
EXMN	IDCEX01	IDCEX01	dump	All Reader/Interpreter and FSR processing is complete.
EXRI	IDCEX01	CALLRI	dump	Before each call to the Reader/Interpreter.
EX2S	IDCEX02	SCANPARM	trace	Before processing the caller's parameter list.
EX3S	IDCEX03	SCANPARM	trace	Before processing the caller's parameter list.
IOAB	IDCIO05	OCABEND	dump	Start of OPEN/CLOSE ABEND routine that sets flag in IOXCTLBK.
			trace	Start of procedure that sets IOXABEND flag in IOXCTLBK.
IOAC	IDCIO02	BUILDACB	dump	After ACB and EXLST have been built, at end of procedure.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
			trace	Start of procedure that builds the ACB and EXLST.
IOAJ	IDCIO05	ADJCCW	trace	Start of procedure that adjusts a CCW channel program.
IOBB	IDCIO05	BLDBLK	trace	Start of procedure that issues UGSPACE, calls READJFCB, and writes IOXCTLBK.
IOCK	IDCIO05	CKOPN	trace	Start of procedure that tests OPEN ABEND and issues UPRINT macros.
IOCL	IDCIO01	IDCIOCL	trace	Start of routine that closes data set.
IOCL	IDCIO05	CLOSEPRC	trace	Start of procedure that closes data sets.
IOCN	IDCIO05	CLOSENEW	trace	Start of procedure that issues SVC 82 to free the DEB.
IOCP	IDCIO01	IDCIOCO	trace	Start of routine that copies a data set.
IOCS	IDCIO05	CLOSESTD	trace	Start of procedure that issues CLOSE.
IOCT	IDCIO05	READCNT	trace	Start of procedure that reads the count field of each record on a track.
IOCW	IDCIO05	SETCCW	trace	Start of procedure that builds the CCW channel program to read the count field of each record.
IOC1	IDCIO05	READCNT	trace	Beginning of routine that determines the number of record read.
IODS **(VS2.03.808)**	IDCIO02	DSDATA	dump	After obtaining data set information from the JFCB.
IOEG	IDCIO01	GETEXT	dump	End of procedure that gets a record from the user routine.
			trace	Start of procedure that gets a record from the user routine.
IOEP	IDCIO01	PUTEXT	dump	After control returns from an external user routine.
			trace	Before record is passed to an external user routine.
IOEX	IDCIO05	IDCIO05	dump	End of IDCIO05 module.
IOFB	IDCIO05	FWRITE	trace	Before calling UEXCP to do I/O.
IOFW	IDCIO05	FWRITE	trace	Beginning of procedure that writes multiple records on a track.
IOGR	IDCI001	PUTREP	dump	After the GET for update.
IOGT	IDCIO01	IDCIOGT	trace	Beginning of routine that gets a data record from a data set.
IOG1 (Deleted for **VS2.03.808**)	IDCIO02	BUILDACB	trace	Before the GENCB macro is issued for the EXLST.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
IOG1 (Deleted for VS2.03.808)	IDCIO05	BLDCCWG1	trace	Start of procedure that builds CCWs to read key and data.
IOG2 (Deleted for VS2.03.808)	IDCIO02	BUILDACB	trace	Before the GENCB macro is issued for the ACB.
IOG3 (Deleted for VS2.03.808)	IDCIO02	BUILDRPL	trace	Before the GENCB macro is issued for the RPL.
IOIF	IDCIO03	DSINFO	trace	Entry to UIOINFO processing.
IOIN	IDCIO05	ICIOO05	dump	Start of IDCIO05 module that performs steps necessary to open, close, read, or write volumes.
IOIT	IDCIO01	IDCIOIT	trace	Start of initialization routine.
IOKD	IDCIO05	READKD	trace	Start of procedure that reads the count, key, and data fields of each record on a track.
IOKS	IDCIO05	SETKDCCW	trace	Start of procedure that builds the CCW channel program to read the count, key, and data fields of each record on a track.
IOK2	IDCIO05	READKD	trace	Start of procedure that calculates the number of records to read.
IOK3	IDCIO05	SETKDCCW	trace	Start of procedure that builds read count-key-data CCWs.
IOK4	IDCIO05	SETKDCCW	trace	Start of procedure bypasses bad count fields.
IOM1 (Deleted for VS2.03.808)	IDCIO03	PTAMDS	trace	Before MODCB macro is issued to put the POINT argument in the RPL.
IOM2 (Deleted for VS2.03.808)	IDCIO03	PTAMDS	trace	Before MODCB macro is issued to make the end-of-data routine inactive.
IOM3 (Deleted for VS2.03.808)	IDCIO03	PTAMDS	trace	Before the MODCB macro is issued to make the end-of-data routine active.
IOOC	IDCIO05	OCABEND	trace	Start of Open/Close ABEND routine.
IOOE	IDCIO05	OPENPROC	trace	Start of procedure that determines type of open.
IOOG	IDCIO01	GETNONVS	trace	Start of procedure to get a nonVSAM logical record.
IOOL	IDCIO05	OPNLAB	trace	Start of procedure that opens VTOC to read or write volume label.
IOON	IDCIO05	OPNNEW	trace	Start of procedure that issues SVC 82 to create a DEB.
IOOP	IDCIO01	IDCIOOP	trace	Start of routine that opens data sets.
IOOP	IDCIO05	OPNPASS	trace	Start of procedure that opens data set to verify passwords or expiration dates.
IOOR	IDCIO05	OPENR	trace	Start of procedure that opens data sets, VTOCs, and staging packs for REPAIRV.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
IOOT	IDCIO03	PTISDS	trace	Before SETL macro is issued.
IOOT	IDCIO05	OPNTAB	trace	Start of procedure that opens data set to copy the MSC tables.
IOOV	IDCIO05	OPNVTOC	trace	Start of procedure that opens the VTOC.
IOOW	IDCIO01	PUTNONVS	trace	Start of procedure to write a nonVSAM logical record.
IOPG	IDCIO05	PUTGET	trace	Start of procedure that frees CCW storage.
IOPL	IDCIO01	PUTREP	trace	Entry to PUT (Replace) routine.
IOPO	IDCIO01	IDCIOPO	trace	Start of routine that positions to a data record in an opened VSAM or ISAM data set.
IOPO	IDCIO03	IDCIO03	dump	After positioning is complete, before returning control to IDCIOPO.
IOPR	IDCIO01	PUTREP	dump	After the PUT for update.
IOPT	IDCIO01	IDCIOPT	trace	Start of routine that writes data records to an opened data set.
IOP1	IDCIO05	BLDCCWP1	trace	Start of procedure that builds CCWs to write count, key, and data.
IOP2	IDCIO05	BLDCCWP2	trace	Start of procedure that builds CCWs to write key and data.
IORJ	IDCIO05	READJFCB	trace	Start of procedure that issues RDJFCB and prints error messages.
IORP	IDCIO02	BUILDRPL	dump	After RPL is built, at end of procedure.
IORT	IDCIO05	RETRY	trace	Start of procedure that issues a message that password is invalid.
IOR1	IDCIO05	OPENR	dump	Start of procedure that opens data sets, VTOCs, and staging packs for REPAIRV.
IOR2	IDCIO05	READCNT	dump	Start of procedure that reads the count field of each record on a track.
IOR3	IDCIO05	READKD	dump	Start of procedure that reads the count, key, and data fields of each record on a track.
IOR4	IDCIO05	SPACCR	dump	Start of procedure that spaces over a defective count field on a track.
IOR5	IDCIO05	SETCCW	dump	Start of procedure that builds the CCW channel program to read the count field of each record on a track.
IOR6	IDCIO05	SETKDCCW	dump	Start of procedure that builds the CCW channel program to read the count, key, and data fields of each record on a track.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
IOR7	IDCIO05	ADJCCW	dump	Start of procedure that adjusts a CCW channel program.
IOSN	IDCIO05	SYNAD	trace	Start of procedure that issues SYNADAF, UPRINT, and SYNADRLS macros.
IOSO	IDCIO03	PTISDS	trace	End of routine that positions to an ISAM or BSAM record.
IOSP	IDCIO05	SPACCR	trace	Start of procedure that spaces over a defective count field on a track.
IOSR	IDCIO03	STOWRTN	trace	Before STOW macro is issued.
IOST	IDCIO01	IDCIOST	trace	Entry to the STOW routine.
IOST (Deleted for **VS2.03.808**)	IDCIO03	PTAMDS	trace	After SHOWCB macro is issued to retrieve the RPL error code.
IOSY	IDCIO05	SYNAD	dump	Start of SYNAD routine that issues SYNADAF, SYNADRLS, and UPRINT macros.
IOS1 (Deleted for **VS2.03.808**)	IDCIO02	OPENRTN	trace	Before SHOWCB macro is issued for the ACB.
IOS1 (Deleted for **VS2.03.808**)	IDCIO03	PTAMDS	trace	Before SHOWCB macro is issued to retrieve the RPL error code.
IOS2	IDCIO01	GETNONVS	trace	Start of SYNAD routine for nonVSAM read error.
IOS2 (Deleted for **VS2.03.808**)	IDCIO02	CLOSERTN	trace	Before SHOWCB macro is issued to retrieve the ACB error code.
IOS4	IDCIO01	PUTNONVS	trace	Start of SYNAD routine for nonVSAM put error.
IOTM	IDCIO01	IDCIOTM	trace	Start of termination routine that closes all data sets and frees space.
IOTO (**Deleted for VS2.03.808**)	IDCIO02	OPENRTN	trace	Before TESTCB to see if data set is a path opened for replace processing.
IOT1 (Deleted for **VS2.03.808**)	IDCIO02	OPENRTN	trace	Before TESTCB macro is issued to test if the ACB is open.
IOT2 (Deleted for **VS2.03.808**)	IDCIO02	OPENRTN	trace	Before the TESTCB macro is issued to determine if the data set has an index.
IOT3 (Deleted for **VS2.03.808**)	IDCIO02	OPENRTN	trace	Before TESTCB macro is issued to determine if data set is RRDS.
IOUO	IDCIO01	IDCIOSI	trace	Entry to UIOINFO entry processing.
IOVE	IDCIO01	GETVSAM	trace	Start of end-of-file exit routine for a VSAM file.
IOVG	IDCIO01	GETVSAM	dump	End of procedure that gets a record or control interval from a VSAM data set.
			trace	Start of procedure that gets a record or control interval from a VSAM data set.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
IOVH	IDCIO05	OPNVTH	trace	Start of procedure that performs a pseudo open on a VTOC header or a VSAM data set.
IOVP	IDCIO01	PUTVSAM	dump	End of procedure that writes a VSAM record.
			trace	Start of procedure that writes a VSAM record.
IOVR **(VS2.03.808)**	IDCIO01	VSAMERR	dump	After detection of a VSAM I/O error.
IOVT	IDCIO03	PTAMDS	trace	Start of procedure that positions to a VSAM record or control interval.
IOVY	IDCIO01	IDCIOVY	dump	After VERIFY macro is issued.
			trace	After VERIFY macro is issued.
IOWB	IDCIO05	WRITEREC	trace	Before calling UEXCP to perform I/O.
IOWR	IDCIO05	WRITEREC	trace	Beginning of procedure which writes a record on a track.
IOW1	IDCIO03	STOWRTN	trace	After STOW macro is issued.
IOXC	IDCIO05	ISSUEXCP	trace	Start of procedure that issues EXCP and WAIT macros and tests for errors.
IO0O	IDCIO03	DSINFO	dump	After OBTAIN macro is issued.
IO00	IDCIO03	DSINFO	dump	After RDJFCB macro is issued.
IO01	IDCIO03	DSINFO	dump	After DEVTYPE macro is issued.
IO02	IDCIO03	DSINFO	dump	After formatting work area.
IO05	IDCIO05	IDCIO05	trace	Start of IDCIO05 module.
IO1C	IDCIO02	CLOSERTN	dump	Before CLOSE macro is issued.
IO1O	IDCIO02	OPENRTN	dump	Before OPEN macro is issued.
IO20	IDCIO02	OPENRTN	dump	After OPEN macro is issued.
IO21 **(VS2.03.808)**	IDCIO02	OPENRTN	dump	At completion of all UOPEN processing.
IO2C **(VS2.03.808)**	IDCIO02	CLOSERTN	dump	At completion of all UCLOSE processing.
IO30	IDCIO02	OPENRTN	dump	After OPEN TYPE=J macro is issued.
LCAL	IDCLC02	LOCPROC	dump	After calling the catalog to locate an entry.
LCAP	IDCLC02	AUPROC	dump	Start of procedure that lists information about alias, nonVSAM, user catalog, or generation data group.
LCAS	IDCLC02	ASLPROC	trace	Start of procedure that locates and prints alias names.
LCAU	IDCLC02	AUPROC	trace	Start of procedure that formats catalog fields for a nonVSAM, alias, BDG, or user catalog entry.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
LCCD	IDCLC02	CDIPROC	dump	Start of procedure that lists information about cluster, AIX, pagespace, data, index, or path.
LCBL	IDCLC02	LOCPROC	dump	Before calling the catalog to locate an entry.
LCCK	IDCLC01	CKDTPROC	trace	Start of procedure that tests the creation and expiration LISTCAT options against the catalog entry's creation and expiration date fields.
LCCL	IDCLC02	CDIPROC	trace	Start of procedure that formats catalog fields for a cluster, AIX, data, index, or path entry.
LCDT	IDCLC01	DATEPROC	trace	Start of procedure that calculates the creation and expiration LISTCAT options to be used in the testing of the catalog entry's creation and expiration date fields.
LCEN	IDCLC01	ENTPROC	trace	Before retrieving each entry in a list of entries.
LCER	IDCLC02	ERRPROC	trace	Start of procedure that issues messages.
LCFP	IDCLC02	FPLPROC	trace	Start of procedure that reinitializes CTGFLs for each locate request.
LCLA	IDCLC02	ANLTPROC	trace	Start of procedure that lists catalog entry associations.
LCLO	IDCLC02	LOCPROC	dump	Ready to do a catalog locate for information.
LCIN	IDCLC01	INITPROC	trace	Start of procedure that initializes the catalog parameter list and work areas.
LCLT	IDCLC02	LISTPROC	trace	Start of procedure that prints catalog data.
LCMG	IDCLC02	ERRPROC	dump	Before UPRINT macro is issued to print a message.
LCNX	IDCLC01	GNXTPROC	trace	Before retrieving each entry when processing a full catalog.
LCOJ	IDCLC01	ENTPROC	dump	After preparing to locate information about a name.
LCRA	IDCLC01	RTEPROC	dump	Processing catalog information for associations of a cluster, AIX, pagespace, or generation data group.
LCRP	IDCLC01	RTEPROC	dump	Start of procedure that determines which procedure will list the catalog entry.
LCRT	IDCLC01	RTEPROC	trace	Start of procedure that directs the retrieved entry to the proper formatting procedure.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
LCR2	IDCLC01	RTEPROC	trace	Start of section of procedure that processes associations of a cluster, or AIX
LCSA	IDCLC02	ANSVPROC	trace	Start of procedure that retrieves the list of types and CI numbers.
LCSH	IDCLC02	SHORTLST	trace	Start of procedure that formats an abbreviated list of catalog entry names from the UCIR workarea.
LCTM	IDCLC01	TIMEPROC	trace	Start of procedure that converts the time of day to a packed decimal format.
LCTP	IDCLC02	LISTPROC	dump	Before UPRINT macro is issued to print catalog data.
LCVL	IDCLC02	VPROC	trace	Start of procedure that formats catalog fields of a space entry.
LCVO	IDCLC02	VOLLIST	trace	Start of procedure that formats an abbreviated list of catalog entrynames and volume serials from the UCIR workarea.
LCWA	IDCLC02	LOCPROC	dump	After calling the catalog to locate an entry.
LC02	IDCLC02	IDCLC02	dump	When IDCLC02 is called the first time to establish addressability.
LC98	IDCLC02	FREESTG	dump	End of LISTCAT FSR, before freeing storage in IDCLC02.
LC99	IDCLC01	IDCLC01	dump	End of LISTCAT FSR, before freeing storage in IDCLC01.
LRAA	IDCLR01	AATOPLR	dump	Entry point for IDCLR01
LRAD	IDCLR01	ADDASOC	dump	Start of procedure that adds an association to the association table.
LRBL	IDCLR01	BLDVEXT	dump	Start of procedure that builds virtual extension table.
LRBU	IDCLR01	BUFSHUF	dump	Start of procedure that moves a record to its "home" buffer.
LRCA	IDCLR01	CATOPEN	dump	Start of procedure that prepares to open the catalog.
LRCK	IDCLR01	CKEYRNG	dump	Start of procedure that checks for keyrange.
LRCR	IDCLR01	CRAOPEN	dump	Start of procedure that opens the CRA.
LRCT	IDCLR01	CTTBLD	dump	Start of procedure that builds CI translate table.
LRC1	IDCLR01	CLEANUP	dump	Start of procedure that cleans up before exit.
LRC2	IDCLR01	CLENCRA	dump	Start of procedure that closes the CRA and prints completion message.
LRDO	IDCLR01	DOOTHR	dump	Start of procedure that controls printing nonVSAM information.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
LRDV	IDCLR01	DOVSAM	dump	Start of procedure that controls printing VSAM information.
LRER	IDCLR01	ERROR	dump	Start of procedure that handles errors.
LRGE	IDCLR01	GETPRT	dump	Start of procedure that gets and print records.
LRIA	IDCLR01	INTASOC	dump	Start of procedure that initializes association tables.
LRIN	IDCLR01	INITLZE	dump	Start of procedure that initializes the FSR.
LRIS	IDCLR01	INTSORT	dump	Start of procedure that initializes the sort table.
LRIV	IDCLR01	INTVEXT	dump	Start of procedure that initializes the virtual extension table.
LRME	IDCLR01	MEMSORT	dump	Start of procedure that sorts the entries in sort table.
LRPA	IDCLR01	PRTAAXV	dump	Start of procedure that prints associated AIXs and volumes.
LRPC	IDCLR01	PRTCMP	dump	Start of procedure that prints and compares information.
LRPD	IDCLR01	PRTDMP	dump	Start of procedure that prints dump if specified.
LRPE	IDCLR01	PRTDMPC	dump	Start of procedure that prints dump of catalog record and underscores miscompares.
LRPF	IDCLR01	PRTFIFO	dump	Start of procedure that prints CRA in order of CI number.
LRPJ	IDCLR01	PRTOJAL	dump	Start of procedure that prints object aliases.
LRPK	IDCLR01	PRTOJVL	dump	Start of procedure that prints an object's volumes.
LRPM	IDCLR01	PRTMCWD	dump	Start of procedure that prints miscompare words.
LRPO	IDCLR01	PRTOTHR	dump	Start of procedure that prints nonVSAM objects.
LRPT	IDCLR01	PRTTIME	dump	Start of procedure that prints timestamps.
LRPV	IDCLR01	PRTVSAM	dump	Start of procedure that prints VSAM structures.
LRPW	IDCLR01	PRTVOL	dump	Start of procedure that prints volume records.
LRSM	IDCLR01	SUMIT	dump	Start of procedure that prints number of entries processed.
LRTC	IDCLR01	TCICTCR	dump	Start of procedure that translates the catalog CI to the CRA.
LRVE	IDCLR01	VERTEXT	dump	Start of procedure that handles vertical extension records.
LRZY	IDCLR01	ERROR	dump	After error message has been printed.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
LRZZ	IDCLR01	ERROR	dump	After error that forced an ABORT of this execution.
LR02	IDCLR02	IDCLR02	dump	Entry point for module that gets a record for Recovery Field management routine.
MPBF	IDCMP01	FPLPROC	trace	Start of procedure that constructs a CTGFL.
MPBG	IDCMP01	IDCMP01	trace	Start of IMPORT FSR.
MPCP	IDCMP01	CLUSPROC	trace	Start of procedure that imports a cluster or alternate index.
MPCT	IDCMP01	CLUSPROC	trace	Before processing information from the portable data set to define a cluster or alternate index.
MPDA	IDCMP01	DALCPROC	trace	Start of procedure that dynamically allocates volumes.
MPDC	IDCMP01	DELTPROC	dump	After the first UCATLG.
MPDD	IDCMP01	DELTPROC	dump	After the second UCATLG.
MPDL	IDCMP01	DELTPROC	trace	Entry to DELTPROC.
MPDN	IDCMP01	DUPNPROC	trace	Start of procedure to process a duplicate entry found in the catalog.
MPFN	IDCMP01	IDCMP01	dump	End of IMPORT FSR, prior to closing data sets.
MPFV	IDCMP01	FVTPROC	trace	Start of procedure that constructs a CTGFV and CTGFLs.
MPLV	IDCMP01	LVLRPROC	trace	Start of procedure that constructs CTGFLs for device and volume information.
MPMG	IDCMP01	MSGPROC	trace	Start of procedure that issues messages.
MPOP	IDCMP01	OPENPROC	trace	Start of procedure that opens either the portable data set or the newly defined data set.
MPPS	IDCMP01	BPASPROC	trace	Start of procedure that constructs the PASSWALL CTGFL for protection information.
MPPT	IDCMP01	CLUSPROC	trace	After imported cluster or alternate index has been successfully defined and the contents of the portable data set copied into the new cluster or alternate index.
MPSP	IDCMP01	CTLGPROC	trace	Start of procedure that calls the catalog to locate, alter, or define an entry.
MPUC	IDCMP01	CNCTPROC	trace	Start of procedure that connects a user catalog.
MPUQ	IDCMP01	IUNIQPRC	trace	After a data or index has been found to be unique.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
MPZZ	IDCMP01	CTLGPROC	dump	Before and after calling the catalog to locate, alter, or define an entry.
PMGP	IDCPM01	GRPHPARM	trace	Start of procedure that processes the graphics option.
PMMG	IDCPM01	MARGPARM	trace	Start of procedure that processes the margins option.
PMTP	IDCPM01	TESTPARM	trace	Start of procedure that initializes the TEST option.
PMTS	IDCPM01	TESTSAVE	trace	Start of procedure that initializes the Test Option Data Area.
PR01	IDCPR01	IDCPR01	dump	End of PRINT FSR.
PR11	IDCPR01	IDCPR01	trace	Start of PRINT FSR.
PR18	IDCPR01	IDCPR01	trace	Before termination processing.
PR21	IDCPR01	TEXTPSET	trace	Start of procedure that sets up the text processor interface.
PR31	IDCPR01	DELIMSET	trace	Start of procedure that establishes the beginning and ending delimiters of the data set to be printed.
RCAC	IDCSA07	GETENT	dump	After calls to the catalog to locate an entry.
RCAC	IDCSA07	UPDATENT	dump	After calls to the catalog to recatalog an entry.
RCBG	IDCSA07	IDCSARC	dump	At entry point to IDCSARC procedure.
RCEX	IDCSA07	IDCSARC	trace	Start of IDCSARC procedure that calls GETENT.
RCGE	IDCSA07	GETENT	trace	Start of GETENT procedure that initializes the CAMLST parameter list.
RCND	IDCSA07	IDCSARC	dump	At return to caller.
RCTE	IDCSA07	TESTENT	trace	Start of TESTENT procedure that tests whether there is a need to recatalog.
RCUE	IDCSA07	UPDATENT	trace	Start of UPDATENT procedure that updates the catalog entry.
RC01	IDCRC02	IDCRC02	trace	Start of main procedure.
RC02	IDCRC02	IDCRC02	dump	Start of main procedure.
RC03	IDCRC02	IDCRC02	trace	Return in main procedure from procedures which processed catalog information for objects. Start of termination processing.
RC04	IDCRC02	IDCRC02	dump	Return in main procedure from procedures which processed catalog information for objects. Start of termination processing.
RC05	IDCRC02	CLUSPROC	trace	Start of procedure which processes VSAM objects.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RC06	IDCRC02	CLUSPROC	dump	Start of procedure which processes VSAM objects.
RC07	IDCRC02	CLUSPROC	trace	Before routine which calls LOCPROC for data and index processing.
RC09	IDCRC02	CLUSPROC	trace	Start build of timestamp information for portability data set.
RC11	IDCRC02	CLUSPROC	trace	Start of processing for path associations for VSAM objects.
RC13	IDCRC02	LOCPROC	trace	Start of procedure which builds CPL and FPL's for catalog locate functions.
RC15	IDCRC02	CTLGPROC	trace	Start of procedure which issues catalog locates.
RC16	IDCRC02	CTLGPROC	dump	Start of procedure which issues catalog locates.
RC17	IDCRC02	OPENPROC	trace	Start of procedure to open input and output data sets.
RC19	IDCRC02	PUTPROC	trace	Start of procedure which writes control records to the output data set.
RC21	IDCRC02	RECPROC	trace	Start of procedure which copies the data from the input data set to the output data set.
RC23	IDCRC02	MVDAPROC	trace	Start of procedure which moves control record information in core and clears work areas in core.
RC25	IDCRC02	CONTRBL	trace	Start of procedure which builds control record information.
RC27	IDCRC02	NVSMPROC	trace	Start of procedure which processes nonVSAM objects.
RC28	IDCRC02	NVSMPROC	dump	Start of procedure which processes nonVSAM objects not associated to GDG's.
RC29	IDCRC02	NVSMPROC	trace	Before timestamp processing for nonVSAM objects not associated to GDG's.
RC31	IDCRC02	SAVEPROC	trace	Start of procedure which saves control record information and writes control information to the output data set.
RC33	IDCRC02	ALSPROC	trace	Start of procedure which processes catalog information for alias associations for nonVSAM objects.
RC35	IDCRC02	GDGPROC	trace	Start of procedure which processes catalog information for GDG's.
RC36	IDCRC02	GDGPROC	dump	Start of procedure which processes catalog information for GDG's.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RC37	IDCRC02	GDGPROC	trace	Before build of timestamp information for GDG's.
RC39	IDCRC02	ASOCPROC	trace	Start of procedure which processes catalog information for nonVSAM objects associated with GDG's.
RC40	IDCRC02	ASOCPROC	dump	Start of procedure which processes catalog information for nonVSAM objects associated with GDG's.
RC42	IDCRC02	PRNTPROC	trace	Start of procedure which prints error messages for associations.
RC79	IDCRC01	TERM	both	Before special processing to terminate request (closing output data set).
RC80	IDCRC01	INIT	both	Before initializing to begin processing.
RC81	IDCRC01	BUILDCRV	both	Before building the CRV.
RC82	IDCRC01	EXPORTDR	both	Before looping down name chain to call IDCRC02 to export data sets.
RC83	IDCRC01	SYNCH	both	Before scanning the name chain for a CRA to check it.
RC84	IDCRC01	OBJVOLCK	both	Before checking synchronization of an entry across multiple volumes.
RC85	IDCRC01	DUPNAMCK	both	Before checking the name chain for duplicates.
RC86	IDCRC01	BUILDNAM	both	Before constructing a block for the name chain.
RC87	IDCRC01	COMPNAME	both	Before compressing a name for the name list.
RC88	IDCRC01	SUBSP	both	Before allocating space for the name chain.
RC89	IDCRC01	MESSAGE	both	Before printing any message from IDCRC01.
RC90	IDCRC01	EXTRACT	both	Before using internal Field Management to get information from CRA.
RC91	IDCRC01	OPENCRA	both	Before opening or closing or CRA and doing all other work (e.g. Build CTT).
RC92	IDCRC01	OPEN	both	Before the opening of the CRA.
RC93	IDCRC01	CKCATNM	both	Before checking owning catalog name of CRA being opened.
RC94	IDCRC01	TIMESTMP	both	Before obtaining format 4 timestamp for CRA being opened.
RC95	IDCRC01	SCANCRA	both	Before scanning CRA to build the CTT table.
RC96	IDCRC01	ERRCK	both	After opening a CRA.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RC97	IDCRC01	NAMETABL	both	Before marking or adding a name to the name chain.
RC98	IDCRC01	DIRECT	both	Before obtaining the directory for a volume.
RC99	IDCRC01	CKNAMES	both	Before gathering information on name in name list from CRA.
RIBT	IDCRI01	BYPASTRM	dump	Start of procedure that bypasses the remainder of the current modal or null command.
RICV	IDCRI01	CONVERT	dump	Start of procedure that converts a constant from EBCDIC to binary or hexadecimal.
RIDC	IDCRI01	DSPLCALC	dump	Start of procedure that calculates the position within a secondary FDT vector in which to place an FDT pointer.
RIDF	IDCRI01	DEFAULTS	dump	Start of procedure that adds default parameters to the FDT.
RIEX	IDCRI01	IDCRI01	dump	Start of Reader/Interpreter module.
RIE1	IDCRI01	ERROR1	dump	Start of procedure that issues a message without inserted text.
RIE2	IDCRI01	ERROR2	dump	Start of procedure that issues a message with inserted text.
RIGN	IDCRI01	GETNEXT	dump	Start of procedure that scans the input command.
RIGQ	IDCRI01	GETQUOTD	dump	Start of procedure that scans a quoted constant.
RIGR	IDCRI01	GETRECRD	dump	Start of procedure that obtains the next input record.
RIID	IDCRI01	DSIDCHK	trace	Check restrictions on a data set name and place in FDT.
RIIR	IDCRI01	INREPEAT	dump	Start of procedure that scans a repeated parameter set.
RIMC	IDCRI01	MORSPACE	dump	Start of procedure that allocates more FDT space for a list of constants.
RIME	IDCRI01	MODLELSE	dump	Start of procedure that scans an ELSE modal command.
RIMI	IDCRI01	MODALIF	dump	Start of procedure that scans an IF modal command.
RIMS	IDCRI01	MODALSET	dump	Start of procedure that scans a SET modal command.
RINN	IDCRI01	NEEDNOTS	dump	Start of procedure that checks the input command for conflicting or missing parameters.
RINS	IDCRI01	NAMESCAN	dump	Start of procedure that checks data set names.
RIPC	IDCRI01	PACKCVB	dump	Start of procedure that converts a decimal constant into a binary fullword.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RIPP	IDCRI01	POSPARM	dump	Start of procedure that scans a positional parameter.
RISC	IDCRI01	SCANCMD	dump	Start of procedure that scans the input command parameters.
RISD	IDCRI02	IDCRI02	dump	Start of module that prepares to scan a command parameter set.
RISE	IDCRI01	SCANENDS	dump	Start of procedure that checks the input record for a continuation delimiter and determines the scanning limits of the record.
RISF	IDCRI01	SETFLAG	dump	Start of procedure that notes the occurrence of a parameter in the FDT.
RISK	IDCRI01	SKIPCMD	dump	Start of procedure that bypasses the remainder of a function command.
RIST	IDCRI01	SETDFLT	dump	Start of procedure that puts parameter defaults in the FDT.
RITM	IDCRI03	IDCRI03	dump	Start of module that performs command termination functions.
RI01	IDCRI01	SCANCMD	trace	Start of scanning for a parameter.
RI02	IDCRI01	SCANCMD	trace	Scanning a first-level parameter.
RI03	IDCRI01	SCANCMD	trace	Scanning a subparameter.
RI04	IDCRI01	GETNEXT	trace	Modal command other than ELSE within an IF.
R105	IDCRI01	GETNEXT	trace	Found a functional command.
RI09	IDCRI01	KWDPARM	trace	Found a keyword subparameter.
RI10	IDCRI04	RISETUP	trace	Start of procedure that prepares to process a command.
RI11	IDCRI01	GETDATA	trace	Start of extracting a scalar value.
RI12	IDCRI01	GETDATA	trace	Extract a character string.
RI12	IDCRI04	MAINSCAN	trace	Start of procedure that processes non-repeated parameters.
RI14	IDCRI04	SUBSCAN	trace	Start of procedure that processes repeated parameters.
RI16	IDCRI02	IDCRI02	trace	Prior to loading the command descriptor.
RI16	IDCRI04	TRNSLATE	trace	Start of procedure that translates the PDL into the FDT.
RI17	IDCRI02	IDCRI02	trace	Beginning of the code sequence to build the PARMINFO table.
RI18	IDCRI04	FINDPDE	trace	Start of procedure that finds the PDE offset into the PDL for the current parameter.
RI20	IDCRI04	REPLIST	trace	Start of procedure that saves repeated parameter information.
RI22	IDCRI04	BUILDFDT	trace	Start of procedure that builds FDT data substructure.
RI24	IDCRI01	CONVERT	trace	Start converting a binary number.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RI24	IDCRI04	GETSPACE	trace	Start of procedure that gets space for the FDT data structures.
RI26	IDCRI04	NEWPARM	trace	Start of procedure that gets a new value for the parameter.
RI27	IDCRI01	CONVERT	trace	Start converting a hexadecimal number.
RI28	IDCRI04	TOOMANY	trace	Start of procedure that handles error of excessive subparameters or data.
RI30	IDCRI01	CONVERT	trace	Change converted digits into a binary fullword.
RI30	IDCRI04	NOTPARMS	trace	Start of procedure that checks for conflicting parameters.
RI32	IDCRI04	RESOLVE	trace	Start of procedure that resolves conflicting parameters.
RI34	IDCRI04	DEFAULTS	trace	Start of procedure that selects parameter defaults.
RI35	IDCRI01	INREPEAT	trace	Loop to reset parameter occurrence flags for possible parameters in the sublist.
RI36	IDCRI01	INREPEAT	trace	End of last repeated sublist.
RI36	IDCRI04	SETDFLT	trace	Start of procedure that puts parameter defaults in the FDT.
RI38	IDCRI04	NEEDPRMS	trace	Start of procedure that checks for missing parameters.
RI40	IDCRI04	ADDPARM	trace	Start of procedure that prompts for missing parameters.
RI42	IDCRI04	MSGKWD	trace	Start of procedure that finds a keyword to put in an error message.
RI44	IDCRI01	SETDFLT	trace	Found that default is allowable; ready to put in FDT.
RI44	IDCRI04	FAILSPAC	trace	Start of procedure that prints "no space" error message.
RI45	IDCRI01	SETDFLT	trace	Move a defaulted unquoted constant to FDT.
RI46	IDCRI04	RITERM	trace	Start of procedure that terminates the Reader/Interpreter for TSO.
RI49	IDCRI01	NXTFIELD	trace	Extract a field from input (verb, keyword, or scalar).
RI50	IDCRI01	NXTFIELD	trace	Extract a keyword field.
RI51	IDCRI01	NXTFIELD	trace	Extract a quoted scalar.
RI56	IDCRI01	NEXTCHAR	trace	End-of-file already found in input.
R157	IDCRI01	NEXTCHAR	trace	Extract first character of a new command.
RI59	IDCRI01	NEXTCHAR	trace	End-of-file found while looking for next character.
RI60	IDCRI01	SCANENDS	trace	Skip leading blanks and comments if preceding record indicated continuation.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RI61	IDCRI01	SCANENDS	trace	Bypass a leading comment.
RI62	IDCRI01	SCANENDS	trace	Bypass leading blanks.
RI66	IDCRI01	DSPLCALC	trace	Calculate displacement into the FDT for a parameter in a first-level repeated parameter list.
RI99	IDCRI03	IDCRI03	trace	End of IDCRI03.
RMAL	IDCRM01	ALISPROC	trace	Entry to ALISPROC.
RMAT	IDCRM01	ALTRPROC	trace	Entry to ALTRPROC.
RMBF	IDCRM01	BFPLPROC	trace	Entry to BFPLPROC.
RMBG	IDCRM01	IDCRM01	trace	Entry to IDCRM01.
RMCE	IDCRM01	CLUSPROC	trace	Exit from CLUSPROC.
RMCL	IDCRM01	CPLPROC	dump	After the CPL has been built.
RMCP	IDCRM01	CLUSPROC	trace	Entry to CLUSPROC.
RMCT	IDCRM01	CLUSPROC	trace	Begin reading of cluster or alternate index information from the portable data set.
RMDA	IDCRM01	DALCPROC	trace	Entry to DALCPROC.
RMDC	IDCRM01	DELTPROC	dump	After the first UCTALG in DELTPROC.
RMDD	IDCRM01	DELTPROC	dump	After the second UCATLG in DELTPROC.
RMDL	IDCRM01	DELTPROC	trace	Entry to DELTPROC.
RMDG	IDCRM01	GDGPROC	trace	A duplicate GDG entry has been found.
RMDN	IDCRM01	NVSMPROC	trace	Duplicate nonVSAM entry found.
RMDU	IDCRM01	UCATPROC	trace	Duplicate user catalog found.
RMDV	IDCRM01	CLUSPROC	trace	A duplicate VSAM entry has been found.
RMEL	IDCRM01	IDCRM01	trace	End of the loop for importing objects.
RMFN	IDCRM01	IDCRM01	dump	Termination of IDCRM01.
RMFV	IDCRM01	FVTPROC	trace	Entry to FVTPROC.
RMGD	IDCRM01	GDGPROC	trace	Entry to GDGPROC.
RMLV	IDCRM01	LVLRPROC	trace	Entry to LVLPROC.
RMOP	IDCRM01	OPENPROC	trace	Entry to OPENPROC.
RMNF	IDCRM01	NFVTPROC	trace	Entry to NFVTPROC.
RMNV	IDCRM01	NVSMPROC	trace	Entry to NVSMPROC.
RMPL	IDCRM01	CPLPROC	trace	Entry to CPLPROC.
RMPS	IDCRM01	BPASPROC	trace	Entry to BPASPROC.
RMPT	IDCRM01	CLUSPROC	trace	Beginning of path definition sequence.
RMRG	IDCRM01	RANGPRC	trace	Entry to RANGPROC.
RMSP	IDCRM01	CTLGPROC	trace	Entry to CTLGPROC.
RMUC	IDCRM01	UCATPROC	trace	Entry to UCATPROC.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RMUQ	IDCRM01	IUNIQPRC	trace	A unique data or index component has been detected.
RMZZ	IDCRM01	CTLGPROC	dump	Before and after the UCATLG in CTLGPROC.
RPCI	IDCRP01	CNVRTCI	dump	On exit from procedure that translates control interval numbers on the backup catalog.
RPD1	IDCRP01	CATRELOD	dump	At the end of all reload error checking before any updates have been done to the target catalog.
RPIO	IDCRP01	DUMPIT	dump	After read or write to backup or target catalog.
RPRV	IDCRP01	RECOVCAT	dump	After initialization of locate parameter list to determine catalog recoverable attribute
RPTU	IDCRP01	TRUENAME	dump	On exit from procedure, having built truename range table.
RPT1	IDCRP01	CATRELOD	trace	Start of procedure that performs catalog reload.
RPT2	IDCRP01	TRUENAME	trace	Start of procedure that the RBA boundaries of the backup truename ranges.
RPT3	IDCRP01	CATRANS	trace	On entry to procedure that locates control interval numbers to be translated.
RPT4	IDCRP01	CNVRTCI	trace	On entry to procedure that converts control interval numbers from the backup catalog.
RPT5	IDCRP01	CATCOMP	trace	On entry to procedure that compares truename records.
RPT6	IDCRP01	VERIFYC	trace	On entry to procedure that issues VERIFY against a catalog.
RP01	IDCRP01	IDCRP01	dump	End of REPRO FSR.
RP12	IDCRP01	IDCRP01	trace	After all data sets have not been opened successfully.
RP13	IDCRP01	IDCRP01	trace	Start of loop that copies the data set by issuing UGET and UPUT macros.
RP18	IDCRP01	IDCRP01	trace	After all records have been copied to output data set.
RP21	IDCRP01	DELIMSET	trace	Start of procedure that sets up the beginning and ending delimiters of the input data set.
Note: Trace and Dump points 'RSAD' through 'RS01' are for **VS2.03.808** only.				
RSAD	IDCRS05	ADDUPCR	trace	Upon entry to routine which updates the CRA for a particular period.
RSAE	IDCRS01	AERROR	trace	On entry to routine that exists if enough storage is not available to establish automatic storage required for RESETCAT modules.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSAS	IDCRS02	ASSOC	trace	On entry to routine that initiates association checking.
RSAT	IDCRS05	ADDTN	trace	On entry to routine that adds a true name to the catalog.
RSA1	IDCRS02	ASSOC	dump	At end of procedure that initiates association checking.
RSA2	IDCRS05	ADDUPCR	dump	At end of procedure that prepares for update CRA processing.
RSBR	IDCRS05	BLDRLST	trace	On entry to routine that adds an entry to the reset volume table.
RSBV	IDCRS05	BLDVLST	trace	On entry to routine that adds an entry to the volume serial table.
RSB1	IDCRS05	BLDVLST	dump	End of procedure that adds an entry to the volume serial table.
RSB2	IDCRS05	BLDRLST	dump	At end of procedure that adds an entry to the reset volume table.
RSCA	IDCRS02	CINALTER	trace	On entry to routine that alters control interval numbers in catalog records.
RSCC	IDCRS07	CNVTCCHH	trace	On entry to routine that converts CCHH to TTnn.
RSCD	IDCRS06	CRADMNT	trace	On entry to routine that requests deallocation of a CRA volume.
RSCE	IDCRS07	CATEOV	trace	On entry to routine that extends the catalog.
RSCH	IDCRS03	CHKDSDIR	trace	On entry to routine that checks a data set directory entry against a DATA or INDEX component.
RSCI	IDCRS01	CATINIT	trace	On entry to routine that initializes RESETCAT's description of the catalog.
RSCK	IDCRS05	CKERR	trace	On entry to routine that prints a message if one is associated with the error message given.
RSCL	IDCRS01	CLEANUP	trace	On entry to routine that ensures all RESETCAT resources are freed.
RSCM	IDCRS06	CRAMNT	trace	On entry to routine that requests CRA deallocation.
RSCO	IDCRS01	COPYCAT	trace	On entry to procedure that copies the catalog to the workfile.
RSCR	IDCRS05	CRAUPCHN	trace	On entry to routine that adds a workfile record to a specific "update CRA" chain.
RSCU	IDCRS03	CATRCDSU	trace	On entry to routine that establishes base record offsets for catalog low key range records.
RSC1	IDCRS01	CATINIT	dump	End of procedure that builds CIN to RRN table.
RSC2	IDCRS01	COPYCAT	dump	End of procedure that copies the catalog to the workfile.

Note: Trace and Dump points on this page are for **VS2.03.808** only.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSC3	IDCRS01	CLEANUP	dump	Before freeing the resources used by RESETCAT.
RSC4	IDCRS05	CKERR	dump	Before RESETCAT FSR is terminated due to an error.
RSC5	IDCRS06	CRAMNT	dump	End of procedure that requests CRA allocation.
RSC6	IDCRS06	CRADMNT	dump	End of procedure that requests CRA deallocation.
RSC7	IDCRS07	CATEOV	dump	At conclusion of routine that extends the catalog.
RSDC	IDCRS06	DSCLOSE	trace	On entry to procedure that closes a VSAM data set.
RSDE	IDCRS04	DELGO	trace	On entry to routine that deletes a group occurrence.
RSDO	IDCRS06	DSOPEN	trace	On entry to procedure that opens VSAM data sets.
RSDT	IDCRS05	DELTN	trace	On entry to procedure that deletes a true name from the catalog.
RSD1	IDCRS06	DSOPEN	dump	End of procedure that opens a VSAM data set.
RSD2	IDCRS06	DSCLOSE	dump	End of procedure that closes a VSAM data set.
RSD3	IDCRS04	DELGO	dump	End of procedure that deletes a group occurrence.
RSEN	IDCRS05	ENTNMCK	trace	On entry to routine that determines if a catalog record has a valid entry name.
RSES	IDCRS01	ENSURECI	trace	On entry to routine that ensures there are enough free CIs for reassignment.
RSE1	IDCRS05	ENTNMCK	dump	End of procedure that determines if a record has a true name.
RSE2	IDCRS01	ENSURECI	dump	A start of procedure prior to ensuring enough free CIs.
RSFI	IDCRS04	FIND	trace	On entry to routine that locates requested information from a set of catalog records.
RSFR	IDCRS07	FRCRCCR	trace	On entry to routine that forces reading of the CCR by Catalog Management.
RSF1	IDCRS04	FIND	dump	End of routine that finds one or all group occurrences.
RSGE	IDCRS05	GENNAME	trace	On entry to routine that generates a true name.
RSGF	IDCRS03	GETFIT	trace	On entry to routine that gets a free entry in tables for ASSOC.
RSGN	IDCRS03	GETNEXTE	trace	On entry to routine that translates an index into a table into a virtual address.
RSGT	IDCRS03	GETTAB	trace	On entry to routine that gets and initializes a table for *ASSOC*.

Note: Trace and dump points on this page are for **VS2.03.808** only.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSGV	IDCRS03	GETVIA	trace	On entry to routine that gets a record by control interval number via a specific CRA.
RSG1	IDCRS03	GETVIA	dump	End of procedure that locates records in the workfile.
RSIN	IDCRS01	INIT	trace	On entry to routine which performs the main initializations for RESETCAT.
RSI1	IDCRS01	INIT	dump	End of procedure that initializes data areas and obtains resources.
RSME	IDCRS01	MERGCRA	trace	On entry to routine that merges each reset CRA into the workfile.
RSMO	IDCRS04	MODGO	trace	On entry to procedure that modifies a group occurrence.
RSMU	IDCRS03	MARKUNUS	trace	On entry to routine that marks a Volume Group Occurrence (VGO) unusable.
RSM1	IDCRS01	MERGCRA	dump	End of procedure that merges and resets CRA into the workfile.
RSM2	IDCRS04	MODGO	dump	End of procedure that modifies a group occurrence.
RSPC	IDCRS02	PROCTYPE	trace	On entry to routine that scans a catalog record for CINs.
RSPI	IDCRS02	PROCCI	trace	On entry to routine that ensures a CIN is in the list of CINs for records being processed.
RSPR	IDCRS01	PROCCRA	trace	On entry to routine that processes the records of the current CRA.
RSPV	IDCRS03	PROCVOL	trace	On entry to routine that resolves space conflicts.
RSP1	IDCRS01	PROCCRA	dump	End of procedure that merges the records of a reset CRA into the workfile.
RSP2	IDCRS03	PROCVOL	dump	Before freeing resources used by PROCVOL routine.
RSP3	IDCRS02	PROCTYPE	dump	After processing a set of records for associations.
RSP4	IDCRS02	PROCCI	dump	End of procedure that ensures that a CIN is in the list of CINs.
RSRC	IDCRS06	RECMGMT	trace	On entry to routine that performs all I/O operations for RESETCAT.
RSRE	IDCRS01	REASSIGN	trace	On entry to routine that performs control interval reassignment.
RSRN	IDCRS07	RENAMEP	trace	On entry to routine that renames duplicate true name entries.
RSR1	IDCRS01	REASSIGN	dump	End of procedure that assigns new CINs to records on the reassign chain.
RSR2	IDCRS06	RECMGMT	dump	End of procedure that performs all I/O requests.

Note: Trace and dump points on this page are for **VS2.03.808** only.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSR4	IDCRS07	RENAMEP	dump	Before freeing resources used by the RENAMEP procedure.
RSSB	IDCRS03	SETBMAP	trace	On entry to routine that checks space conflicts for D or I type catalog entries.
RSSC	IDCRS02	SCANCI	trace	On entry to routine that scans records for control intervals.
RSSE	IDCRS02	SETCI	trace	On entry to routine that updates the workfile to reflect new CINs for reassigned CINs.
RSSR	IDCRS05	SCNRLST	trace	On entry to routine that obtains the next CRA volser entry for reset.
RSST	IDCRS03	SETBITS	trace	On entry to routine that maps extents to a bit map.
RSSV	IDCRS05	SCNVLST	trace	On entry to routine that scans through the list of volumes.
RSS2	IDCRS02	SETCI	dump	End of procedure that updates the workfile records from the associations tables.
RSS3	IDCRS03	SETBITS	dump	At end of procedure that sets up a single bit map.
RSS5	IDCRS05	SCNVLST	dump	End of procedure that locates an entry in the volume serial table.
RSS6	IDCRS05	SCNRLST	dump	End of procedure that locates an entry in the reset volume table.
RSUA	IDCRS03	UNALLOC	trace	On entry to routine which unallocates suballocated space from temporary space maps.
RSUC	IDCRS01	UPDCRA	trace	On entry to routine which updates the CRAs from the workfile.
RSUP	IDCRS01	UPDCAT	trace	On entry to routine which updates the catalog from the workfile.
RSUR	IDCRS07	UPDCCR	trace	On entry to procedure which updates the CCR for the catalog.
RSU1	IDCRS01	UPDCAT	dump	End of procedure that updates the catalog from the workfile.
RSU2	IDCRS01	UPDCRA	dump	End of procedure that updates the CRAs from the workfile.
RSVB	ICDRS03	VERB	trace	On entry to routine which verifies associations for GDG base records.
RSVC	IDCRS02	VERC	trace	On entry to routine which verifies associations for clusters.
RSVE	IDCRS02	VERDSDIR	trace	On entry to routine which verifies the data set directory entries for VSAM data sets not on reset volumes.
RSVG	IDCRS02	VERG	trace	On entry to routine which verifies associations for AIXs.

Note: Trace and dump points on this page are for **VS2.03.808** only.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSVN	IDCRS03	VLNRESET	trace	On entry to routine which verifies space requested from objects being reset against non-reset volumes.
RSVO	IDCRS01	VOLCHK	trace	On entry to volume consistency routine (VOLCHK).
RSVP	IDCRS02	VERR	trace	Upon entry to routine which verifies associations for PATHs.
RSVR	IDCRS02	VERCI	trace	On entry to routine which checks validity of each CIN found in a set of records.
RSVS	IDCRS03	VLRESET	trace	On entry to routine which verifies space requested against reset volumes.
RSVU	IDCRS02	VERU	trace	On entry to routine which verifies associations for user catalogs.
RSVX	IDCRS02	VERX	trace	On entry to routine which verifies alias associations.
RSV1	IDCRS03	VOLCHK	dump	End of procedure that checks Format 1 DSCBs against space headers.
RSV2	IDCRS02	VERDSDIR	dump	After verifying initial space claims.
RSV3	IDCRS02	VERCI	dump	After verifying associations on a set of records.
RSV4	IDCRS03	VERB	dump	Before freeing resources used by procedure which verifies GDG data sets.
RSWF	IDCRS06	WFDEF	trace	Upon entry to routine which defines an RRDS as a workfile for RESETCAT processing.
RSWL	IDCRS06	WFDEL	trace	On entry to routine which deletes the workfile.
RSWR	IDCRS01	WRAPUP	trace	On entry to routine which handles clean-up operations after successful RESETCAT processing.
RSW2	IDCRS06	WFDEF	dump	Before the UCATLG work area is freed.
RSW3	IDCRS06	WFDEL	dump	End of procedure that deletes the workfile.
RS00	IDCRS01	IDCRS01	dump	End of RESETCAT FSR.
RS01	IDCRS01	IDCRS01	trace	Upon entry to main RESETCAT module.
SAAL	IDCSA02	IDCSA02	trace	Start of UALLOC macro processing.
SABB **(VS2.03.807)**	IDCSA02	FINDNAME	dump	After a new Load List Block has been built.
SACA	IDCSA02	IDCSA02	trace	Start of routine that processes UCATLG macro.

Note: Trace and dump points through RS01 are for **VS2.03.808** only.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
SACA	IDCSA09	CHECKARG	trace	Start of CHECKARG procedure that determines the macro to be issued and checks for errors.
SACC	IDCSA09	CHKCODE	trace	Start of CHKCODE procedure that returns error codes to the caller.
SACK	IDCSA06	UCBCHECK	trace	Start of routine that checks for UCB.
SACL	IDCSA02	IDCSA02	trace	Start of routine that processes UCALL macro.
SACR	IDCSA02	IDCSA02	trace	Start of UCIR macro processing.
SADE	IDCSA02	IDCSA02	trace	Start of routine that processes UDELETE macro.
SADL	IDCSA02	IDCSA02	trace	Start of UDEALLOC macro processing.
SADM	IDCSA06	DEMNTVOL	trace	Start of procedure that calls ISSUEDMT and UCBPOST.
SADQ	IDCSA08	IDCSA08	trace	Start of routine that processes UDEQ macro.
SAD1 **(VS2.03.807)**	IDCSA01	IDCSA01	trace	During termination processing before all loaded modules are deleted.
SAD2 **(VS2.03.807)**	IDCSA01	IDCSA01	dump	During termination processing after all loaded modules have been deleted.
SAD3 **(VS2.03.807)**	IDCSA02	RELECORE	trace	Before loaded modules with zero use count are deleted.
SAD4 **(VS2.03.807)**	IDCSA02	RELECORE	dump	After deletion of loaded modules with zero use count.
SAED	IDCSA06	ENQDEQ	trace	Start of procedure that enqueues and dequeues.
SAEX	IDCSA06	IDCSA06	dump	End of IDCSA06 module.
SAFE	IDCSA06	FINDEXCL	trace	Start of procedure that determines whether the volume is already allocated exclusively to the job step.
SAFP	IDCSA02	IDCSA02	trace	Start of routine that processes UFPOOL macro.
SAFS	IDCSA02	IDCSA02	trace	Start of routine that processes UFSPACE macro.
SAGE	IDCSA06	GETEXCL	trace	Start of procedure that determines whether the allocation of a unit and volume can be changed to exclusive.
SAGP	IDCSA02	IDCSA02	trace	Start of routine that processes UGPOOL macro.
SAGS	IDCSA02	IDCSA02	trace	Start of routine that processes UGSPACE macro.
SAID	IDCSA02	IDCSA02	trace	Start of UID macro processing.
SAID	IDCSA06	ISSUEDMT	trace	Start of routine that calls USSC macro to demount volume.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
SAIM	IDCSA06		trace	Start of routine that issues USSC macro to mount volume.
SAIN	IDCSA06	IDCSA06	dump	Start of IDCSA06 module that performs MOUNT, DEMOUNT, POST, and CHECK.
SALC	IDCSA07	IDCSALC	dump	At entry to load module.
SALD	IDCSA02	IDCSA02	trace	Start of routine that processes ULOAD macro.
SALE	IDCSA07	IDCSALC	dump	At exit from load module.
SALK	IDCSA02	IDCSA02	trace	Start of procedure that processes ULINK macro.
SALT	IDCSA07	IDCSALC	trace	At entry to load module.
SAMA	IDCSA09	ISSUEMAC	trace	Start of ISSUEMAC procedure that issues the execute form of the requested macro.
SAMC	IDCSA06	MOUNTCTL	trace	Start of control procedure that processes mount requests.
SAMT	IDCSA06	MOUNTVOL	trace	Start of procedure that calls ISSUEMNT, RDLABEL, and UCBPOST.
SANQ	IDCSA08	IDCSA08	trace	Start of routine that processes UENQ macro.
SAPR	IDCSA06	MDSETUP	trace	Start of procedure that sets up for a mount or demount request.
SAPT	IDCSA02	IDCSA02	trace	Start of procedure that processes UPROMPT macro.
SAQL	IDCSA02	IDCSA02	trace	Start of UQUAL macro processing.
SARD	IDCSA06	RDLABEL	trace	Start of routine that issues the UEXCP macro to read volume label.
SARV	IDCSA08	IDCSA08	trace	Start of routine that processes URESERVE macro.
SAR1 (SU 5752-824)	IDCSA08	IDCSA08	dump	After the RACHECK parameter list has been built; before the RACHECK macro is issued.
SAR2 (SU 5752-824)	IDCSA08	IDCSA08	dump	Upon return from the RACHECK macro.
SASC	IDCSA08	IDCSA08	trace	Start of routine that processes USCRATCH macro.
SASC	IDCSA06	SELESCAN	trace	At entry to SELESCAN procedure which scans the TIOT.
SASD	IDCSA06	SELECTD	trace	At entry to SELECTD procedure which searches to see if the caller's volume is mounted.
SASE	IDCSA06	SETEXCL	trace	Start of procedure that changes the enqueue on a volume from shared to exclusive and marks the UCB non-shareable.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
SASI	IDCSA08	IDCSA08	trace	Start of routine that processes USYSINFO macro.
SASN	IDCSA02	IDCSA02	trace	Start of routine that processes USNAP macro.
SASS	IDCSA09	IDCSA09	dump	Start of USSC macro.
			trace	Start of USSC macro.
SAST	IDCSA06	SCANTIOT	trace	Start of procedure that scans the TIOT to find a UCB and DD statement that can be used to process the volume. Start of USSC macro.
SATI	IDCSA02	IDCSA02	trace	Start of routine that processes UTIME macro.
SAUC	IDCSA07	IDCSAUC	dump	Start of UUNCATLG macro.
			trace	Start of UUNCATLG macro.
SAUE	IDCSA07	IDCSAUC	dump	Exit from UUNCATLG macro.
SAUP	IDCSA06	UCBPOST	trace	Start of procedure that builds parameter list and issues SVC 82 to post or clear a UCB.
SAVL	IDCSA07	VSAMLOC	trace	At entry to VSAMLOC procedure.
SAVU	IDCSA07	VSAMUCT	trace	At entry to procedure that issues VSAM DELETE.
SAWO	IDCSA08	IDCSA08	trace	Start of routine that processes UWTO macro.
SA05	IDCSA05	IDCSA05	trace	Before the TIME macro is issued.
SA06	IDCSA06	IDCSA06	trace	Start IDCSA06 module.
STBG	IDCSA10	IDCSAST	dump	At start of module that establishes or cancels an ESTAE environment.
STCS	IDCSA10	CANSTAE	trace	At entry to CANSTAE procedure that issues an ESTAE macro to cancel the ESTAE environment.
STEN	IDCSA10	IDCSAST	dump	At end of module.
STEX	IDCSA10	STAEEXIT	trace	At entry to ESTAEEXIT procedure that determines whether it can retry.
STRY	IDCSA10	RECOVERY	trace	At entry to RECOVERY procedure that cleans up before ABEND.
STSS	IDCSA10	SETSTAE	trace	At entry to SETSTAE procedure that issues an ESTAE macro to establish an ESTAE environment.
STST	IDCSA10	IDCSAST	trace	At entry to USTAE macro that establishes or cancels an ESTAE environment.
TPCC	IDCTP01	IDCTP01	trace	Before the call to the CONVERT routine is issued.
TPEA	IDCTP06	IDCTP06	dump	Start of procedure that processes UERROR macro.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
TPEB	IDCTP06	IDCTP06	dump	Before converted message is printed via the UPRINT macro.
TPER	IDCTP01	ERROR	dump	Start of procedure that prints a text processor error message.
TPE1	IDCTP06	IDCTP06	trace	Start of procedure that processes UERROR macro.
TPE2	IDCTP06	CATERCNV	trace	Entry point to routine that converts catalog error messages to prose.
TPE4 **(VS2.03.807)**	IDCTP06	DAERCNV	trace	On entry to dynamic allocation error conversion routine.
TPE5 **(VS2.03.807)**	IDCTP06	DAERCNV	trace	Before call to DAIRFAIL service routine.
TPIN	IDCTP01	IDCTPPR	dump	At end of phase; the format structure for a UPRINT macro has been processed.
TPMS **(VS2.03.807)**	IDCTP01	LINEPRT	dump	On return from segment routine.
TPSI	IDCTP01	IDCTPPR	dump	After initialization of text processor parameters.
TP2I	IDCTP01	CONVERT	dump	Start of procedure that converts data to a printable form.
TP2N	IDCTP01	CONVERT	dump	End of procedure that converts data to a printable form.
TP3I	IDCTP01	LINEPRT	dump	Start of procedure that formats pages and prints titles, headings, footings, and other lines requested.
TP3N	IDCTP01	LINEPRT	dump	End of procedure that prints lines.
TP4A	IDCTP04	ESTACONT	dump	End of procedure that processes the UESTA macro.
TP4R	IDCTP04	RESTCONT	dump	End of procedure that processes UREST macro.
TP4S	IDCTP04	ESTSCONT	dump	End of procedure that processes
TP5E	IDCTP05	IDCTP05	trace	Start of procedure to get a static text module.
				UESTS macro.
TP5I	IDCTP05	IDCTP05	dump	Start of phase that loads the static text phase.
TP5N	IDCTP05	IDCTP05	dump	End of phase that loads the static text phase.
TPXX	IDCTP01	STACKPUT	dump	Before call to UPUT.
TPX1	IDCTP01	LINEPRT	dump	After return from procedure that puts print lines into output buffer.
VSBG	IDCVS01	IDCBS01	dump	Start of IDCVS01 module.
VSBG	IDCVS02	IDCVS02	dump	Start of IDCVS02 module.
VSCC	IDCVS01	CATCHK	trace	Start of CATCHK procedure that initializes the catalog parameter list.
VSCG	IDCVS03	CATLG	trace	Start of CATLG procedure.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
VSCO	IDCVS01	CATOPEN	trace	Start of CATOPEN procedure that issues a UOPEN macro to open the catalog.
VSCR	IDCVS03	CREATION	trace	Start of CREATION procedure that checks data set creation date.
VSCT	IDCVS03	CRITERIA	trace	Start of CRITERIA procedure that checks if data set meets caller's criteria.
VSCU	IDCVS03	CLEANUP	trace	At entry to CLEANUP procedure that releases resources.
VSEC	IDCVS01	EXPIRCHK	trace	Start of EXPIRCHK procedure that opens an unexpired nonVSAM data set.
VSEL	IDCVS03	EXCLUDE	trace	At entry to EXCLUDE procedure which excludes data sets which caller does not want included in data set array list.
VSEX	IDCVS03	EXPIRATN	trace	At entry to procedure that checks expiration date.
VSF0	IDCVS02	FMTTRK0	trace	Start of FMTTRK routine that prepares the three records written to cylinder 0, track 0.
VSF1	IDCVS02	FMTTRK1	trace	Start of FMTTRK routine that prepares the 39 DSCBs written to cylinder 0, track 1.
VSF1	IDCVS03	GETFMT1	trace	Start of procedure that reads a format 1 DSCB from the VTOC (GETFMT1).
VSGB	IDCVS03	GETBLK	trace	Start of GETBLK procedure which computes and obtains storage for IDCVS03.
VSGL	IDCVS02	GETLAB	trace	Start of procedure that reads VTOC label.
VSIN	IDCVS02	INITVOLM	trace	Start of procedure that issues EXCP and calls FMTTRK1 and FMTTRK0.
VSIT	IDCVS03	INITIAL	trace	Start of initialization procedure (INITIAL).
VSMN	IDCVS02	IDCVS02	trace	Start of IDCVS02 module.
VSNC	IDCVS01	NVSAMCHK	trace	Start of NVSAMCHK procedure that issues a UEXCP macro to open nonVSAM data sets.
VSND	IDCVS01	IDCVS01	dump	End of IDCVS01 module.
VSOP	IDCVS03	OPENVTOC	trace	At entry to routine that opens the VTOC.
VSOT	IDCVS02	IDCVS02	dump	End of IDCVS02 module.
VSPI	IDCVS02	PUTLAB	trace	Start of procedure that writes VTOC label.
VSQL	IDCVS03	QUAL	trace	Start of routine which processes data set names with qualifiers (QUAL).

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
VSRC	IDCVS01	RECATLOG	trace	Start of RECATLOG procedure that recatalogs nonVSAM data sets.
VSSC	IDCVS01	SCRATCH	trace	Start of SCRATCH procedure that calls VTOCOPEN and FMT4SCR procedures.
VSSE	IDCVS01	SECHECK	trace	Start of SECHECK procedure that ensures that a volume is empty or has the correct password.
VSSN	IDCVS03	SYSNAMES	trace	Start of SYSNAMES procedure which checks for system names.
VSSR	IDCVS03	SELECTOR	trace	Start of SELECTOR procedure that selects eligible data sets from the VTOC.
VSST	IDCVS03	STATUS	trace	Start of STATUS routine which provides status information for the data sets selected.
VSTK	IDCVS03	TRKREAD	trace	Start of routine that reads a track into the track I/O buffer.
VSUS	IDCVS03	USAGE	trace	Start of USAGE routine that determines the organization and size of a data set.
VSVC	IDCVS01	VSAMCHK	trace	Start of VSAMCHK procedure that issues a UEXCP macro to open VSAM data sets.
VSVG	IDCVS01	VTOCGET	trace	Start of VTOCGET procedure that calls the VTOCOPEN and FMTR4READ procedures.
VSVP	IDCVS01	VTOCPUT	trace	Start of VTOCPUT procedure that calls the VTOCOPEN and FMT4READ procedures.
VSVT	IDCVS01	VTOCOPEN	trace	Start of VTOCPUT procedure that opens the VTOC.
VS01	IDCVS01	IDCVS01	trace	Start of IDCVS01 procedure that performs VTOC services.
VS03	IDCVS03	IDCVS03	dump	At entry to load module.
VS1C	IDCVS01	FMT1CHK	trace	Start of FMT1CHK procedure that checks for Format 1 DSCB.
VS1R	IDCVS01	FMT1READ	trace	Start of FMT1READ procedure that reads the Format 1 DSCB.
VS1S	IDCVS01	FMT1SCR	trace	Start of FMT1SCR procedure that scratches the Format 1 DSCB.
VS3X	IDCVS03	IDCVS03	dump	At exit from load module.
VS4C	IDCVS01	FMT4CHK	trace	Start of FMT4CHK procedure that checks the Format 4 DSCB format for VSAM catalog ownership.
VS4R	IDCVS01	FMT4READ	trace	Start of FMT4READ procedure that reads the Format 4 DSCB.
VS4S	IDCVS01	FMT4SCR	trace	Start of FMT4SCR procedure that scratches the Format 4 DSCB.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
VYBG	IDCVY01	IDCVY01	dump	Start of VERIFY FSR.
VYCL	IDCVY01	TERMPROC	trace	Start of procedure that closes the data set that was verified.
VYND	IDCVY01	IDCVY01	dump	End of VERIFY FSR.
VYOP	IDCVY01	OPENPROC	trace	Start of procedure that opens the VSAM data set to be verified.
VYST	IDCVY01	IDCVY01	trace	Start of VERIFY FSR.
XPAO	IDCXP01	CLUSPROC	trace	Before retrieving from the catalog the entries associated with the cluster or alternate index being exported.
XPAP	IDCSP01	ALTRPROC	trace	Start of procedure that modifies the CTGPL to set the temporary export flag on.
XPBG	IDCXP01	IDCXP01	trace	Start of EXPORT FSR.
XPCP	IDCXP01	CLUSPROC	trace	Before retrieving the catalog entry for the object to be exported.
XPCR	IDCXP01	CONTRBL	trace	Before constructing control records for the portable data set.
XPCW	IDCXP01	CONTRBL	trace	Before writing control records to the portable data set.
XPDP	IDCXP01	DELTPROC	trace	Start of procedure that sets up the CTGPL to delete a cluster or disconnect a user catalog.
XPED	IDCXP01	IDCXP01	trace	End of EXPORT FSR.
XPFN	IDCXP01	IDCXP01	dump	End of EXPORT FSR, before data sets are closed and space freed.
XPLP	IDCXP01	LOCPROC	trace	Start of procedure that builds the CTGPL and CTGFLs for a locate request.
XPMS	IDCXP01	MORESP	trace	Entry to MORESP.
XPOP	IDCXP01	OPENPROC	trace	Start of procedure that opens either the portable data set or the cluster or alternate index to be exported.
XPPM	IDCXP01	CLUSPROC	trace	Before processing the permanent or temporary export option.
XPPP	IDCXP01	PUTPROC	trace	Start of procedure that writes a record to the portable data set.
XPRP	IDCXP01	RECPROC	trace	Entry to RECPROC.
XPSP	IDCXP01	CTLGPROC	trace	Start of procedure that calls the catalog for a locate, alter, or delete request.
XPTM	IDCXP01	CLUSPROC	trace	Before calling the procedure to alter the CTGPL to set the temporary export flag.
XPUC	IDCXP01	DSCTPROC	trace	Start of procedure that disconnects a user catalog.

Trace and Dump Points to Module or CSECT Cross Reference

Trace or Dump Point	Module or CSECT	Procedure	Type	Situation at Dump or Trace Point
XPWC	IDCXP01	CLUSPROC	trace	Before writing the catalog information to the portable data set.
XPZY	IDCXP01	DELTPROC	dump	Just after the UCATLG macro.
XPZZ	IDCXP01	CTLGPROC	dump	After calling the catalog to locate, alter, or delete an entry.
XP01	IDCXP01	IDCXP01	dump	Start of EXPORT FSR.
ZZAL	IDCSA02	IDCSA02	dump	Before and after each time UALLOC invokes dynamic allocation.
ZZCA	IDCSA02	IDCSA02	dump	Before and after CATLG macro is issued to invoke catalog management routines.
ZZCR	IDCSA02	IDCSA02	dump	Start of procedure that processes UCIR macro and just before the UCIR macro returns to the module that issued the UCIR.
ZZDL	IDCSA02	IDCSA02	dump	Before and after the UDEALLOC macro invokes dynamic allocation.
ZZSC	IDCSA09	ISSUEMAC	dump	Before issuing SVC126.

Module to Dump Points Cross Reference

The following list contains the dump points within each module and procedure, indicates what information can be dumped at each point (either a full dump or selected areas), and explains the situation at the dump point. As explained in the section "TEST Keyword" in this chapter, full region dumps are taken at all dump points in this list. Selected areas can be printed with either the AREAS or FULL variation of the Test option. Details of the selected areas are given in the footnotes following the list.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCAL01	CHECKPRC	AL51	dump	After locating data component of the alternate index for which UPGRADE has been specified.
		AL52	dump	After locating associated cluster or the alternate index of the data object specified on ALTER command.
		AL53	dump	After locating associated index component.
		AL54	dump	After locating the data component of the path's base cluster.
		AL55	dump	After locating the cluster component of the alternate index's base cluster.
		AL56	dump	After locating the data component of the alternate index's base cluster.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCAL01 (continued)	IDCAL01	AL01	dump	Before calling the catalog to alter an object.
		AL02	dump	End of ALTER FSR.
		AL04	dump	Before issuing ALTER request for index objects if KEYS specified.
	LOCATPRC	AL03	dump	After calling the catalog to locate an object.
IDCBI01	CATPROC	BIL2	dump	After return from UCATLG for each locate request.
	CNTRLPRC	BIC2	dump	After completion of sort if an internal sort. After completion of sort phase and before merge passes if an external sort.
	DEFPROC	BID2	dump	After return from UCATLG to define each sort work each sort work file.
	DELTPROC	BIDL	dump	After return from UCATLG to delete each sort work file.
	INITPROC	BII2	dump	After obtaining or failing to obtain sort storage.
	JCPROC	BIJ2	dump	After return from each call to UIOINFO.
	MAINPROC	BI03	dump	After return from procedure which locates information about the base cluster and alternate index.
		BI04	dump	After the alternate index has been built, before CLOSE.
	MERGPROC	BIM3	dump	After the tree has been initialized for each merge pass of an external sort.
		BIM4	dump	After processing one set of strings during the merge pass of an external sort.
	OPENPROC	BIP2	dump	After return from UOPEN to open a data set.
	SORTPROC	BISR	dump	Before sorting the records in the record sort area.
IDCCC01	ALTRLSTS	CCAC	selected areas 1	After building parameter lists to alter a VSAM catalog.
		CCAV	selected areas 2	After building parameter lists to alter a VSAM catalog.
	CATALOG	CCLV	selected areas 3	After request to locate volume information on a duplicate entry.
	CNVTINIT	CCIC	dump	After initializing for scanning of OS/VS catalog.
	CONTINUE	CCCN	selected areas 4	After obtaining block at the same index level as the last block from an OS/VS catalog.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	CONVERT	CCSE	dump	Invalid entry type in the OS/VS catalog.
		CCSL	dump	Finished converting OS/VS catalog entries to VSAM catalog entries.
	DEFNLSTS	CCDC	selected areas 5	After building a VSAM parameter list for a define.
		CCDV	selected areas 6	After building a VSAM parameter list for a define.
	LOCTLSTS	CCLC	selected areas 7	After building parameter lists to locate information in the VSAM catalog.
	POPUP	CCPU	dump	Ready to continue scan of higher level index block.
	PUSHDOWN	CCIE	dump	No Index Control Entry found on a lower level scan in the OS/VS catalog.
		CCPD	selected areas 8	After obtaining a block at the same index level as the last block from the OS/VS catalog.
	VOLINDEX	CCVE	dump	No Volume Index Control Entry in the OS/VS catalog.
		CCVI	selected areas 9	First block of volume index is obtained from the OS/VS catalog.
IDCCK01	DSDRPROC	CK30	dump	After reading a type 1 DSDR record.
		CK40	dump	Before processing a type 1 DSDR record for a tape data set.
	DSDRVOLS	CK50	dump	Before processing a type 2 DSDR record.
	IDCCK01	CK10	dump	Before calling CHR procedure.
		CK20	dump	Upon return from CHR procedure.
IDCDE01	IDCDE01	DE01	dump	Before calling the catalog to define an object.
		DE02	dump	End of DEFINE FSR, before completion message is issued.
IDCDE02 **(VS2.03.807)**	MODELPRC	DE03	dump	After calling the catalog to locate a model object.
		DE04	dump	End of procedure that built the model table.
IDCDE02	FREESTG	DE37	dump	End of IDCDE02 CSECT.
IDCDL01	CATCALL	DLDB	dump	Before calling the catalog to delete an entry.
	CATCALL	DLDA	dump	After calling the catalog to delete an entry.
	FINDTYPE	DLLB	dump	Before calling the catalog to locate the entry type.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCDL01 (continued)	FINDTYPE	DLLA	dump	After calling the catalog to locate the entry type.
	IDCDL01	DLBG	dump	Start of DELETE FSR.
		DLND	dump	End of DELETE FSR, before data sets are closed and the completion message is issued.
IDCEX01	CALLFSR	EXFS	dump	Before each call to an FSR.
	CALLRI	EXRI	dump	Before each call to the Reader/Interpreter.
	IDCEX01	EXMN	dump	All Reader/Interpreter FSR processing is complete.
IDCIO01	GETEXT	IOEG	dump	End of procedure that gets a record from the user routine.
	GETVSAM	IOVG	dump	End of procedure that gets a record or control interval from a VSAM data set.
	IDCIOVY	IOVY	dump	After VERIFY macro is issued.
	PUTEXT	IOEP	dump	After control returns from an external user routine.
	PUTNONVS	IO2P	dump	After writing a spanned record.
	PUTREP	IOPR	dump	After the PUT for update.
		IOGR	dump	After the GET for update.
	PUTVSAM	IOVP	dump	End of procedure that writes a VSAM record.
	VSAMERR **(VS2.03.808)**	IOVR	dump	After detection of a VSAM I/O error.
IDCIO02	BUILDACB	IOAC	dump	After ACB and EXLST have been built, at end of procedure.
	BUILDRPL	IORP	dump	After RPL is built, at end of procedure.
	CLOSERTN	IO1C	dump	Before CLOSE macro is issued.
		IO2C **(VS2.03.808)**	dump	At end of all UCLOSE processing.
	DSDATA **(VS2.03.808)**	IODS	dump	After obtaining data set information from the JFCB.
	OPENRTN	IO1O	dump	Before OPEN macro is issued.
		IO20	dump	After OPEN macro is issued.
		IO21 **(VS2.03.808)**	dump	After all UOPEN processing.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCIO03	DSINFO	IO00	dump	After RDJFCB macro is issued.
		IO01	dump	After return from DEVTYPE.
		IO02	dump	After formatting the work area.
	IDCIO03	IOPO	dump	After positioning is complete, before returning control to IDCIOPO.
	STOWRTN	IOW1	dump	Before STOW macro is issued.
		IOW2	dump	After STOW macro is issued.
IDCIO05	ADJCCW	IOR7	dump	Start of procedure that adjusts a CCW channel program.
	IDCIO05	IOIN	dump	Start of IDCIO05 module.
		IOEX	dump	End of IDCIO05 module. routine.
	OCABEND	IOAB	dump	Start of OPEN/CLOSE ABEND routine.
	OPENR	IOR1	dump	Start of procedure that opens data sets, VTOCs, and staging packs for REPAIRV.
	READCNT	IOR2	dump	Start of procedure that reads the count field of each record on a track.
	READKD	IOR3	dump	Start of procedure that reads the count, key, and data fields of each record on a track.
	SETCCW	IOR5	dump	Start of procedure that builds the CCW channel program to read the count, key, and data fields of each record on a track.
	SETKDCCW	IOR6	dump	Start of procedure that builds the CCW channel program to read the count, key, and data fields of each record on a track.
	SPACCR	IOR4	dump	Start of procedure that spaces over a defective count field on a track.
	SYNAD	IOSY	dump	Start of SYNAD routine.
IDCLC01	ENTPROC	LCOJ	selected areas 12	After preparing to locate information about a name.
	IDCLC01	LC99	dump	End of LISTCAT FSR, before freeing storage in IDCLC01.
	RTEPROC	LCRA	selected areas 19	Processing catalog information for associations of a cluster, pagespace, or generation data group, or AIX.
		LCRP	selected areas 20	Start of procedure that determines which procedure will list the catalog entry.
IDCLC02	AUPROC	LCAP	selected areas 10	Start of procedure that lists information about alias, nonVSAM, user catalog, or generation data group.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCLC02 (continued)	CDIPROC	LCCD	selected areas 11	Start of procedure that lists information about cluster, pagespace, data, or index, AIX and PATH.
	ERRPROC	LCMG	selected areas 13	Before UPRINT macro is issued to print a message.
	FREESTG	LC98	dump	End of LISTCAT FSR, before freeing IDCLC02's automatic storage.
	LDCLC02	LC02	dump	When IDCLC02 is called the first time to establish addressability.
	LISTPROC	LCTP	selected areas 14	Before UPRINT macro is issued to print catalog data.
	LOCPROC	LCAL	selected areas 15	After calling the catalog to locate an entry.
		LCBL	selected areas 16	Before calling the catalog to locate an entry.
		LCLO	selected areas 17	Ready to do a catalog locate for information.
		LCWA	selected areas 18	After calling the catalog to locate an entry.
IDCLR01	AATOPLR	LRAA	dump	Entry point for IDCLR01.
	ADDASOC	LRAD	dump	Start of procedure that adds an association to the association table.
	BLDVEXT	LRBL	dump	Start of procedure that builds vertical extension tables.
	BUFSHUF	LRBU	dump	Start of procedure that moves a record to its buffer.
	CATOPEN	LCRA	dump	Start of procedure that prepares to open the catalog.
	CKEYRNG	LRCK	dump	Start of procedure that checks for keyrange.
	CLEANUP	LRC1	dump	Start of procedure that cleans up before exit.
	CLENCRA	LRC2	dump	Start of procedure that closes the CRA and prints miscompare message.
	CRAOPEN	LRCR	dump	Start of procedure that opens the CRA.
	CTTBLD	LRCT	dump	Start of procedure that builds CI translate table.
	DOOTHR	LRDO	dump	Start of procedure that looks for nonVSAM information.
	DOVSAM	LRDV	dump	Start of procedure that looks for VSAM information.
	ERROR	LRER	dump	Start of procedure that handles errors.
	GETPRT	LRGE	dump	Start of procedure that gets and prints records.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	INITLZE	LRIN	dump	Start of procedure that initializes the FSR.
	INTASOC	LRIA	dump	Start of procedure that initializes association tables.
	INTSORT	LRIS	dump	Start of procedure that initializes the sort table.
	INTVEXT	LRIV	dump	Start of procedure that initializes the vertical extension table.
	MEMSORT	LRME	dump	Start of procedure that sorts the entries in sort table.
	PRTAAXV	LRPA	dump	Start of procedure that prints associated AIXs and volumes.
	PRTCMP	LRPC	dump	Start of procedure that prints compare information.
	PRTDMP	LRPD	dump	Start of procedure that prints dump if specified.
	PRTDMPC	LRPE	dump	Start of procedure that prints dump of catalog record and underscores miscompares.
	PRTFIFO	LRPF	dump	Start of procedure that prints CRA in order of CI number.
	PRTMCWD	LRPM	dump	Start of procedure that prints miscompare words.
	PRTOJAL	LRPJ	dump	Start of procedure that prints object aliases.
	PRTOJVL	LRPK	dump	Start of procedure that prints an object's volumes.
	PROTHR	LRPO	dump	Start of procedure that prints nonVSAM objects.
	PRTTIME	LRPT	dump	Start of procedure that prints timestamps.
	PRTVOL	LRPW	dump	Start of procedure that prints volume records.
	PRTVSAM	LRPV	dump	Start of procedure that prints VSAM structures.
	SUMIT	LRSM	dump	Start of procedure that prints number of entries processed.
	TCICTCR	LRTC	dump	Start of procedure that translates the catalog CI to the CRA.
	VERTEXT	LRVE	dump	Start of procedure that handles vertical extension records.
IDCLR02	IDCLR02	LR02	dump	Entry point for module that gets a record for Recovery Field management routine.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCMP01	CTLGPROC	MPZZ	dump	Before and after calling the catalog to locate, alter, or define an entry.
	DELTPROC	MPDC	dump	After the first UCATLG.
		MPDD	dump	After the second UCATLG.
	IDCMP01	MPFN	dump	End of IMPORT FSR, prior to closing data sets.
IDCPR01	IDCPR01	PR01	dump	End of PRINT FSR.
IDCRC01	CKNAMES	RC99	dump	Before gathering information on name in name list from CRA.
	DIRECT	RC98	dump	Before obtaining a directory for a volume.
	NAMETABL	RC97	dump	Before marking or adding a name to the name chain.
	ERRCK	RC96	dump	After opening a CRA.
	SCANCRA	RC95	dump	Before scanning CRA to build the CTT table.
	TIMESTMP	RC94	dump	Before obtaining format 4 timestamp for CRA being opened.
	CKCATNM	RC93	dump	Before checking owning catalog name of CRA being opened.
	OPEN	RC92	dump	Before the opening of the CRA.
	OPENCRA	RC91	dump	Before opening or closing a CRA and doing all other work (e.g. Build CTT).
	EXTRACT	RC90	dump	Before using internal Field Management to get information from CRA.
	MESSAGE	RC89	dump	Before printing any message from IDCRC01.
	SUBSP	RC88	dump	Before allocating space for the name chain.
	COMPNAME	RC87	dump	Before compressing a name for the name list.
	BUILDNAM	RC86	dump	Before constructing a block for the name chain.
	DUPNAMCK	RC85	dump	Before checking the name chain for duplicates.
	OBJVOLCK	RC84	dump	Before checking Sync. of entry across multiple volumes.
	SYNCH	RC83	dump	Before scanning the name chain for a CRA to check it.
	EXPORTDR	RC82	dump	Before looping down name chain to call IDCRC02 to export data sets.
	BUILDCRV	RC81	dump	Before building the CRV.
IDCRC01	INIT	RC80	dump	Before initializing to begin processing.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	TERM	RC79	dump	Before special processing to terminate request (closing output data set.
IDCRC02	ASOCPROC	RC40	dump	Start of procedure which processes nonVSAM objects associated with GDG's
	CLUSPROC	RC06	dump	Start of procedure which processes VSAM objects.
	CTLGPROC	RC16	dump	Start of procedure which issues catalog locates.
	GDGPROC	RC36	dump	Start of procedure which processes GDG objects.
	IDCRC02	RC02	dump	Start of main procedure.
	IDCRC02	RC04	dump	Before termination processing.
	NVSMPROC	RC28	dump	Start of procedure which processes nonVSAM objects not associated with GDG's.
IDCRI01	BYPASTRM	RIBT	dump	Start of procedure that bypasses the remainder of the current modal or null command.
	CONVERT	RICV	dump	Start of procedure that converts a constant from EBCDIC to binary or hexadecimal.
	DEFAULTS	RIDF	dump	Start of procedure that adds default parameters to the FDT.
	DSPLCALC	RIDC	dump	Start of procedure that calculates the position within a secondary FDT vector in which to place an FDT pointer.
	ERROR1	RIE1	dump	Start of procedure that issues a message without inserted text.
	ERROR2	RIE2	dump	Start of procedure that issues a message with inserted text.
	GETNEXT	RIGN	dump	Start of procedure that scans the input command.
	GETQUOTD	RIGQ	dump	Start of procedure that scans a quoted constant.
	GETRECRD	RIGR	dump	Start of procedure that obtains the next input record.
	IDCRI01	RIEX	dump	Start of Reader/Interpreter module.
	INREPEAT	RIIR	dump	Start of procedure that scans a repeated parameter set.
	MODALIF	RIMI	dump	Start of procedure that scans an IF modal command.
	MODALSET	RIMS	dump	Start of procedure that scans a SET modal command.
	MODLELSE	RIME	dump	Start of procedure that scans an ELSE modal command.
IDCRI01	MORSPACE	RIMC	dump	Start of procedure that scans an FDT space for a list of constants.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCRI01 (continued)	NAMESCAN	RINS	dump	Start of procedure that checks data set names.
	NEEDNOTS	RINN	dump	Start of procedure that checks the input command for conflicting or missing parameters.
	PACKCVB	RIPC	dump	Start of procedure that converts a decimal constant into a binary fullword.
	POSPARM	RIPP	dump	Start of procedure that scans a positional parameter.
	SCANCMD	RISC	dump	Start of procedure that scans the input command parameters.
	SCANENDS	RISE	dump	Start of procedure that checks the input record for a continuation delimiter and determines the scanning limits of the record.
	SETDFLT	RIST	dump	Start of procedure that puts defaults in the FDT.
	SETFLAG	RISF	dump	Start of procedure that notes the occurrence of a parameter in the FDT.
	SKIPCMD	RISK	dump	Start of procedure that bypasses the remainder of a function command.
IDCRI02	IDCRI02	RISD	dump	Start of module that prepares to scan a command parameter set.
IDCRI03	IDCRI03	RITM	dump	Start of module that performs command termination functions.
IDCRM01	CPLPROC	RMCL	dump	After the CPL has been built.
	CTLGPROC	RMZZ	dump	Before and after the UCATLG in CTLGPROC.
	DELTPROC	RMDC	dump	After the first UCATLG in DELTPROC.
		RMDD	dump	After the second UCATLG in DELTPROC.
	IDCRM01	RMFN	dump	Termination of IDCRM01.
IDCRP01	IDCRP01	RP01	dump	End of REPRO FSR.
	CATRELOD	RPD1	dump	At the end of all reload error checking before any updates have been done to the target catalog.
	CNVRTCI	RPCI	selected areas 21	On exit from procedure that translates control interval numbers on the backup catalog.
	DUMPIT	RPIO	selected areas 22	After read or write to backup or target catalog.
	RECOVCAT	RPRV	selected areas 23	After setting up locate for recoverable catalog attributes.
	TRUENAME	RPTU	selected areas 24	On exit from procedure having built truename range table.
IDCRS01 **(VS2.03.808)**	CATINIT	RSC1	dump	End of procedure that builds CIN to RRN table.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCRS01 (continued)	COPYCAT	RSC2	dump	End of procedure that copies the catalog to the workfile.
	CLEANUP	RSC3	dump	Before freeing the resources used by RESETCAT.
	ENSURECI	RSE2	dump	At start of procedure prior to ensuring enough free control intervals.
	INIT	RSI1	dump	End of procedure that initializes data area and obtains resources.
	MERGECRA	RSM1	dump	End of procedure that merges and resets CRA into the workfile.
	PROCCRA	RSP1	dump	End of procedure that merges the records of a reset CRA into the workfile.
	REASSIGN	RSR1	dump	End of procedure that assigns new control interval numbers to records on the reassign chain.
	UPDCAT	RSU1	dump	End of procedure that updates the catalog from the workfile.
	IDCRS01	RS00	dump	End of RESETCAT FSR.
IDCRS02 **(VS2.03.808)**	ASSOC	RSA1	dump	End of procedure that initiates association checking.
	PROCTYPE	RSP3	dump	After processing a set of records for associations.
	PROCCI	RSP4	dump	End of procedure that ensures that a control interval number is in the list of control interval numbers.
	SETCI	RSS2	dump	End of procedure that updates the workfile records from the associations tables.
	VERDSDIR	RSV2	dump	After verifying initial space claims.
	VERCI	RSV3	dump	After verifying associations on a set of records.
IDCRS03 **(VS2.03.808)**	GETVIA	RSG1	dump	End of procedure that locates records in the workfile.
	PROCVOL	RSP2	dump	Before freeing resources used by PROCVOL routine.
	SETBITS	RSS3	dump	At end of procedure that sets up a single bit map.
	VOLCHK	RSV1	dump	End of procedure that checks Format 1 DSCBs against space headers.
	VERB	RSV4	dump	Before freeing resources used by procedure which verifies GDG data sets.
IDCRS04 **(VS2.03.808)**	DELGO	RSD3	dump	End of procedure that deletes a group occurrence.
	FIND	RSF1	dump	End of routine that finds one or all group occurrences.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCRS04 (continued)	MODGO	RSM2	dump	End of procedure that modifies a group occurrence.
IDCRS05 **(VS2.03.808)**	ADDUPCR	RSA2	dump	End of procedure that prepares for update CRA processing.
	BLDVLST	RSB1	dump	End of procedure that adds an entry to the volume serial table.
	BLDRLST	RSB2	dump	End of procedure that adds an entry to the reset volume table.
	CKERR	RSC4	dump	Before RESETCAT terminates due to an error.
	ENTNMCK	RSE1	dump	End of procedure that determines if a record has a true name.
	SCNVLST	RSS5	dump	End of procedure that locates an entry in the volume serial table.
	SCNRLST	RSS6	dump	End of procedure that locates an entry in the reset volume table.
IDCRS06 **(VS2.03.808)**	CRAMNT	RSC5	dump	End of procedure that requests CRA allocation.
	CRADMNT	RSC6	dump	End of procedure that requests CRA deallocation.
	DAOPEN	RSD1	dump	End of procedure that opens a VSAM file.
	DSCLOSE	RSD2	dump	End of procedure that closes a VSAM file.
	RECMGMT	RSR2	dump	End of procedure that performs all I/O requests.
	WFDEF	RSW2	dump	Before the UCATLG work area is freed.
	WFDEL	RSW3	dump	End of procedure that deletes the workfile.
IDCRS07 **(VS2.03.808)**	CATEOV	RSC7	dump	At conclusion of routine that extends the catalog.
	RENAMEP	RSR4	dump	Before freeing resources used by the RENAMEP procedure.
	UPDCCR	RSU2	dump	End of procedure that updates the CRAs from the workfile.
IDCSA01 **(VS2.03.807)**	IDCSA01	SAD2	dump	During termination processing; after all loaded modules have been deleted. have been deleted.
IDCSA02 **(VS2.03.807)**	FINDNAME	SABB	dump	After a new Load List Block has been built.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCSA02	IDCSA02	ZZAL	dump	Before and after each time the UALLOC macro invokes dynamic allocation.
		ZZCA	dump	Before and after CATLG macro is issued to invoke catalog management routines.
		ZZCR	dump	Start of procedure that processes UCIR macro and just before the UCIR macro returns to the module that issued the UCIR.
		ZZDL	dump	Before and after the UDEALLOC macro invokes dynamic allocation.
	RELECORE **(VS2.03.807)**	SAD4	dump	After deletion of loaded modules with zero use count.
IDCSA06	IDCSA06	SAEX	dump	End of IDCSA06 module.
		SAIN	dump	Start of IDCSA06 module.
IDCSA07	IDCSARC	RCBG	dump	Start of IDCSARC procedure.
		RCND	dump	End of IDCSARC procedure.
	UPDATENT	RCAC	selected areas 25	After calls to the catalog.
	IDCSALC	SALC	dump	Start of IDCSALC procedure.
		SALE	dump	End of IDCSALC procedure.
	IDCSAUC	SAUC	dump	Start of IDCSAUC procedure.
		SAUE	dump	End of IDCSAUC procedure.
IDCSA08 **(SU 5752-824)**	IDCSA08	SAR1	dump	After the RACHECK parameter list has been built; before the RACHECK macro is issued.
		SAR2	dump	Upon return from theRACHECK macro.
IDCSA09	CHKCODE	AA01	dump	After delayed response for Mass Storage control order.
	IDCSA09	SASS	dump	Start of USSC macro.
	ISSUEMAC	AA00	dump	After issuing SVC 126.
		ZZSC	dump	Before issuing SVC 126.
IDCSA10	IDCSAST	STBG	dump	At start of module that established or cancels an ESTAE environment.
		STEN	dump	At end of module.
IDCTP01	CONVERT	TP2I	dump	Start of procedure that converts data to a printable form.
		TP2N	dump	End of procedure that converts data to a printable form.
	ERROR	TPER	dump	Start of procedure that prints a text processor error message.
	IDCTPPR	TPS1	dump	After initialization of text processor parameters.

Module or CSECT to Dump Points Cross Reference

Module or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
IDCTP01 (continued)		TPIN	dump	At end of phase; the format structure for a UPRINT macro has been processed.
	LINEPRT **(VS2.03.807)**	TPMS	dump	On return from segment routine.
	LINEPRT	TPX1	dump	After return from procedure that puts print lines into output buffer.
		TP3I	dump	Start of procedure that formats pages and prints titles, headings, footings, and other lines requested.
		TP3N	dump	End of procedure that prints lines.
	STACKPUT	TPXX	dump	Before call to UPUT.
IDCTP04	ESTACONT	TP4A	dump	End of procedure that processes the UESTA macro.
	ESTSCONT	TP4S	dump	End of procedure that processes the UESTS macro.
	RESTCONT	TP4R	dump	End of procedure that processes the UREST macro.
IDCTP05	IDCTP05	TP5I	dump	Start of phase that loads the static text phase.
		TP5N	dump	Start of phase that loads the static text phase.
IDCTP06	IDCTP06	TPEA	dump	Start of procedure that processes UERROR macro.
		TPEB	dump	Before converted message is printed via the UPRINT macro.
IDCVS01	IDCVS01	VSBG	dump	Start of IDCVS01 module.
		VSND	dump	End of IDCVS01 module.
IDCVS02	IDCVS02	VSBG	dump	Start of IDCVS02 module.
		VSOT	dump	End of IDCVS02 module.
IDCVS03	IDCVS03	VS8G	dump	Entry to load module.
		VS3X	dump	Exit from load module.
IDCVY01	IDCVY01	VYBG	dump	Start of VERIFY FSR.
		VYND	dump	End of VERIFY FSR.
IDCXP01	IDCXP01	XPFN	dump	End of EXPORT FSR, before data sets are closed and space freed.
		XP01	dump	Start of EXPORT FSR.
	CTLGPROC	XPZZ	dump	After calling the catalog to locate, alter, or delete an entry.
	DELTPROC	XPZY	dump	Just after the UCATLG macro.

Selected Areas Notes: The following list describes the selected areas printed at the specified dump points. On the printed output, the area title precedes each area dumped.

Dump Point		Area Title	Area Description
1.	CCAC	CPLPTR	VSAM catalog parameter list address
		CPL	VSAM catalog parameter list
		FVTPTR	VSAM catalog field vector table address
		FVT	VSAM field vector table
		DEVTPTR	VSAM field parameter list address (device type FPL)
		DEVTFPL	VSAM field parameter list (device type FPL)
		FSEQPTR	VSAM field parameter list address (file sequence FPL)
		FSEQFPL	VSAM field parameter list (file sequence FPL)
2.	CCAV	DEVTPTR	Address of device code list
		DEVTLIST	Device code list (referenced by FPL)
		VOLPTR	Address of volume serial number list
		VOLLIST	Volume serial number list (referenced by FVT)
		FILEPTR	Address of file sequence list
		FILELIST	File sequence number list (referenced by FPL)
3.	CCLV	ENTYPPTR	Address of catalog entry type returned by VSAM catalog
		ENTYPLST	Catalog entry type returned by VSAM catalog
		CATVLPTR	Address of catalog volume information returned by VSAM catalog
		CATVLLST	Catalog volume information returned by VSAM catalog
4.	CCCN	BLKLEVEL	OS/VS catalog level index
		BLOCKPTR	OS/VS catalog index block pointer
		BLOCK	OS/VS catalog index block
5.	CCDC	Same as CCAC	
6.	CCDV	Same as CCAV	
7.	CCLC	CPLPTR	VSAM catalog parameter list address
		CPL	VSAM catalog parameter list
		ENTYPPTR	VSAM field parameter list address (entry type FPL)
		ENTYPFPL	VSAM field parameter list (entry type FPL)
		CATVLPTR	VSAM field parameter list address (volume information FPL)
		CATVLFPL	VSAM field parameter list (volume information FPL)
8.	CCPD	Same as CCCN	
9.	CCVI	Same as CCCN	
10.	LCAP	FLAGS	Internal flags and switches used in IDCLC01 (refer to the microfiche listing data area, FLAGS, for mapping)
		ASSOC#	Number of associations of the catalog entry being processed
		HOLDETYP	Type of catalog entry being processed
11.	LCCD	Same as LCAP	
12.	LCOJ	CATNAME	Catalog name

Dump Point	Area Title	Area Description
12. LCOJ (continued)	OBJCNT	Number of entries being processed **Note:** Multiple entries can be processed when a generic name is supplied.
	OBJTYP	Entry type currently being processed
	OBJNM	Name of entry being processed
	CACBFLG	Flag indicating that VSAM should return the catalog ACB pointer
	CTGOPTN2	Second option byte indicator in CTGPL
	FLAGS	Internal flags and switches (refer to microfiche listing data area, FLAGS, for mapping)
	TYPECNT	Number of entry types specified by the user
	TYPELIST	Types of entry types specified by the user
13. LCMG	ERRDARG	Text processor argument list (DARGLIST) used for printing messages
14. LCTP	DARGLIST	Text processor argument list (DARGLIST) used for printing the catalog data
15. LCAL	CATRC	VSAM catalog return code
	CTGENT	VSAM locate key (either the entry name or the CI number)
	CTGPSWD	User-supplied password
	CTGPL	VSAM catalog parameter list
	CTGFL array	
	FPL(1)	VSAM field parameter list **Note:** The number of FPLs (nn) varies with the amount of catalog information requested (i.e., NAME, HISTORY, VOL, etc.).
	FPL(nn)	HISTORY, VOL, etc.).
	MULTIFPL	VSAM field parameter list if a special function FPL is required
16. LCBL	Same as LCAL	
17. LCLO	FLAGS	Internal flags and switches (refer to microfiche listing data area, FLAGS, for mapping)
18. LCWA	CTGWKAPT	Work area address of VSAM returned catalog fields
	CTGWKA array	
	WKA(1)	VSAM returned catalog fields. **Note:** This work area is dumped as an array of 256-byte blocks and the last block less than 256 bytes is indicated as WKAEND.
	WKAEND	
19. LCRA	Same as LCAP	
20. LCRP	Same as LCAP	
21. RPCI	OLDCI#	CI number of backup catalog record to be converted
	NEWCI#	Converted CI number in the target catalog (i.e., OLDCI# converted to NEWCI#)
22. RPIO	DLOUTREC	A record in the high key range of the target catalog which was deleted because it did not exist in the backup catalog
	FUPOTREC	A record in the low key range of the target catalog which was converted to a free record because it did not exist in the backup catalog

Dump Point	Area Title	Area Description
	INSOTREC	A record inserted into the target catalog because it existed in the backup catalog but not in the target catalog
	UPOUTREC	A record used to update the target catalog because the same record existed in both the backup and the target catalogs
	RDCCREC	Catalog control record of the target catalog before it was updated
	UPCCREC	Catalog control record of the target catalog after it was updated with results of the reload operation
	RDINPREC	A record from the backup catalog before any action is taken
	RDOUTREC	A record from the target catalog before any action is taken
	2ND-HALF	The second half of the record printed just above
23. RPRV	DSATRPL	If both input and output are catalogs, the CPL used to locate the catalog DSATTR field to determine if it has the recoverable attribute **Note:** The first printed is for the output catalog; the second is for the input catalog.
	DSATRFPL	The FPL chained off the CPL described for DSATRPL
24. RPTU	SORSTABL	A table which maps the extents of the high key range of the backup catalog **Note:** Each entry maps one extent and contains: Word 1—High RBA of the extent Word 2—Number of CIs in the extent The table is used to convert a CI number in the backup catalog to the appropriate CI number for the target catalog (see 'RPCI' above).
	TARGTABL	Same as SORSTABL for the target catalog
25. RCAC	WKAREA	Catalog work area

The TSO TEST Command

If you invoked Access Method Services interactively with TSO, you can use the TSO TEST command to diagnose a problem. Refer to OS/VS2 TSO *Command Language Reference* for a complete description of the TSO TEST command. In order to execute Access Method Services with the TEST command, enter the following after logging on at your terminal:

```
TEST 'SYS1.CMDLIB
(Access Method Services command name)'CP
```

TSO responds with:

```
ENTER COMMAND FOR CP
```

Now, enter:

```
Access Method Services command with its parameters
```

TSO responds with:

```
TEST
```

Now, Access Method Services is running under the TSO TEST command, and you can enter any TEST subcommand.

How to Use the TSO TEST Command

You must use the microfiche listings with the TSO TEST command.

If a problem occurs and you have no idea which modules are involved, run the command under TSO TEST and stop execution at the beginning of each module. Each Access Method Services module begins with a PROL macro which calls module IDCSA03 at entry point IDCSAPR. Using the microfiche listings for IDCSA03, find the offset from the beginning of IDCSA03 to IDCSAPR. Enter:

```
AT IDCSA03.IDCSA03.+offsett-to-IDCSAPRD
```

When execution stops at the beginning of each module, you can display the trace tables with the LIST subcommand. From the Inter-Module Trace Table you should be able to tell the modules involved.

If you suspect which modules are involved, rerun the Access Method Services command under TSO TEST and stop execution at the beginning of each module involved by using an AT subcommand. Each time execution stops you can examine the trace tables and storage to further isolate the problem.

Once you know the areas within a module that have caused the problem, select a dump or trace point where you want to stop execution. Using the microfiche listing for the module, find the offset from the beginning of the module to the dump or trace point. Enter an AT subcommand for each point where you want to stop execution.

Even though you don't have a dump, refer to the section on "Reading a Dump" to find data areas in storage.

ABORT Codes

Whenever an unrecoverable error is detected by the processor, the routine that detects the error issues a UABORT macro. The System Adapter then issues a Write-To-Programmer message (IDC4999I) with an ABORT code indicating the error and produces a snap dump if the //AMSDUMP DD card is specified. If you invoked Access Method Services with a batched job, you supply the //AMSDUMP DD card in the JCL used to invoke Access Method Services. If you invoked Access Method Services interactively with TSO, you must supply the //AMSDUMP DD card in the procedure you specified in the PROC parameter of the LOGON command. Refer to *OS/VS2 TSO Command Language Reference* for more information on the LOGON command. A snap dump produced through a UABORT macro has an ID = 000.

Module IDCSA01, procedure PRNTERR, detects the ABORT condition. One of the following ABORT codes is issued along with message IDC4999I.

ABORT Codes

ABORT Code	Module or CSECT	Procedure	Situation that Caused ABORT
24(18)	IDCTP01	IDCTP01	The pointer to the Print Control Table in the GDT is not set.
	IDCTP04	RESETCON	The pointer to the Print Control Table in the GDT is not set.
28(1C)	IDCIO01	IDCIOIT	Storage was not available for the I/O Adapter historical area and message area.
	IDCIO02	BLDOCMSG	A message that sufficient storage was not available could not be issued because the SYSPRINT data set is not open.
	IDCIO04	IDCIO04	PUTLINE returned a return code greater than 4.
	IDCSA01	GETCORE	Storage was not available for the automatic storage required for IDCSA02, IDCSA03, IDCSA06, IDCSA07, IDCSA08, IDCSA09, IDCSA10, IDCIO01, IDCIO05, or IDCTP01.
	IDCSA02	IDCSA02	Storage was not available for a dynamic volume serial list or a dynamic DD name list.
	IDCSA03	GETCORE	Storage was not available for a GETMAIN request.
	IDCTP01	LINEPRT	Storage unavailable for page header line.
	IDCTP01	ERROR	Storage unavailable for saving conversion/print argument list.
	IDCTP05	IDCTP05	Storage unavailable for loading static text formatting structure.
	IDCTP04	ESTSCONT	Storage unavailable for line stack buffer.
	IDCTP04	PCTSETUP	Storage unavailable for print chain translate table.
	IDCTP04	PCTSETUP	Storage unavailable for main title line.
	IDCTP04	PCTSETUP	Storage unavailable for subtitle line.
	IDCTP04	PCTSETUP	Storage unavailable for page footing lines.
	IDCTP04	INITPCT	Storage unavailable for Print Control Table.

ABORT Codes

ABORT Code	Module or CSECT	Procedure	Situation that Caused ABORT
32(20)	IDCIO01	IDCIOGT	The pointer to the IOCSTR is zero, or the open flag in the IOCSTR is not set, indicating that the data set to be accessed has not been opened successfully.
		IDCIOPT	The pointer to the IOCSTR is zero, or the open flag in the IOCSTR is not set, indicating that the data set to be accessed has not been opened successfully.
	IDCIO03	IDCIO03	The pointer to the IOCSTR is zero, or the open flag in the IOCSTR is not set, indicating that the data set to be accessed has not been opened successfully.
36(24)	IDCIO02	BLDOCMSG	The SYSPRINT data set could not be opened, or the SYSPRINT data set has already been closed and a message cannot be issued.
	IDCTP01	STACKPUT	Unable to transmit data to output file.
40(28)	IDCIO01	IDCIOCL	The length of the UCLOSE argument list is invalid. The length must be greater than 1 and less than 6.
		IDCIOOP	The length of the UOPEN argument list is invalid. The length must be greater than 1 and less than 6.
		IDCIOPT	The length of the UPUT argument list is invalid. The length must be greater than 1 and less than 4.
		IDCIOSI	The length of the UIOINFO argument list is invalid. The length must be greater than 3 and less than 6.
	IDCSA02	IDCSA02	The argument list of a UGSAPCE, UGPOOL, UFPOOL, UALLOC, or UCIR macro is invalid.
	IDCSA05	IDCSA05	The argument list for the UTIME macro is invalid.
	IDCSA08	IDCSA08	The argument list of a USYSINFO, URACHECK (SU 5752-824), USCRATCH, or UWTO macro is invalid.
44(2C)	IDCSA02	IDCSA02	The SNAP macro was not successful. Either the DCB for the dump data set was not Open or was invalid, or not enough storage was available for the dump.
56(38) (VS2.03.807)	IDCSA02	IDCSA02	The BLDL macro failed to find a required Access Method Services module.

ABORT Codes

ABORT Code	Module or CSECT	Procedure	Situation that Caused ABORT
60(3C) **(Deleted for VS2.03.808)**	IDCIO02	BUILDACB	The return code from the VSAM GENCB macro for the EXLST or ACB was not zero, and the reason code in register zero indicates a problem other than storage not available.
		BUILDRPL	The return code from the VSAM GENCB macro for the RPL was not zero, and the reason code in register zero indicates a problem other than storage not available.
		CLOSERTN	The return code from the VSAM SHOWCB macro for the ACB error code was not zero.
		OPENRTN	The return code from either the VSAM SHOWCB macro for the ACB attributes or the VSAM TESTCB macro for the index attribute or ACB open flag was not zero.
	IDCIO03	PTAMDS	The return code from either the VSAM SHOWCB macro for the RPL error code or the VSAM MODCB macro for the RPL or the EXLIST was not zero.
72(48) **(VS2.03.808)**	IDCRS05	CKERR	An internal RESETCAT error occurred. This situation should not occur in a debugged program.

You can find UABORT macros by examining the microfiche listings. The expansion of a UABORT macro for an ABORT code of 28 looks like this:

```
RESPECIFY(REG13,REG14,REG15) RSTD;
REG15 = 28;
REG14 = GDTABT;
REG13 = GDTABH;
GEN(BR    REG14);
RESPECIFY(REG13,REG14,REG15)UNRSTD;
```

Reading a Dump

This section describes how to find modules and data areas belonging to the processor in a full region dump, either a snap dump or an ABEND dump. If you are debugging under the TSO TEST command, you can use this section with the microfiche listings to find data in storage.

Snap dumps are produced by the processor on two different occasions. If the Test option is on and the FULL keyword is specified, the processor produces as many snap dumps as requested, at the points requested. The IDs of these dumps start with ID = 001. If an ABORT condition occurs, the processor produces a snap dump with an ID = 000.

All executable modules and certain data areas belonging to the processor are preceded by an EBCDIC character string to identify the module or data area. Modules are preceded by the full module name, for example, IDCTP01b. (The date of compilation, in character form, follows the module name.) Data areas are preceded by a four-byte identifier, either specific to the data area, or for the storage area in which it is built. For example, the GDT is preceded by the characters GDTb. The FDT is built in storage owned by the Executive, and it is found in the storage areas preceded by the characters EX00.

How to Find the Module and Registers

The best way to determine which module caused the dump and to find the registers of that module varies according to the type of dump you have.

In an ABEND dump, standard methods explained in *OS/VS2 System Programming Library: Debugging Handbook* should be used.

In a snap dump caused by an ABORT condition, the last entry in the Inter-Module Trace Table identifies the module that issued the UABORT macro. Message IDC4999I identifies the ABORT code set in the UABORT macro. Once you know the ABORT code and the module that issued the UABORT macro, you can use the previous list of ABORT codes to determine the internal procedure that issued the UABORT macro and the situation that caused the procedure to issue the macro. The last entry in the Intra-Module Trace Table may be a trace point within the module that issued the UABORT macro.

The registers at the time that the UABORT macro was issued are not saved by the processor and cannot be found in a dump.

If you have a snap dump produced at a dump point, the trace tables printed along with the dump tell you at what point the dump occurred. The next to the last ID in the Inter-Module Trace Table identifies the module that issued the UDUMP macro; the last ID in the Intra-Module Trace Table identifies the exact dump point at which the dump was produced. You can use the trace tables printed along with the dump to trace the flow of control before the dump point. These trace tables are better to use for this purpose than the trace tables in the dump because the printed trace tables do not contain all the trace points encountered while producing the dump. The trace tables in the dump have been filled with dump-related trace points.

You can find the registers at the time the UDUMP macro was issued in the save area where IDCDB01 saved the caller's registers. The first word of this save area contains the characters DB01.

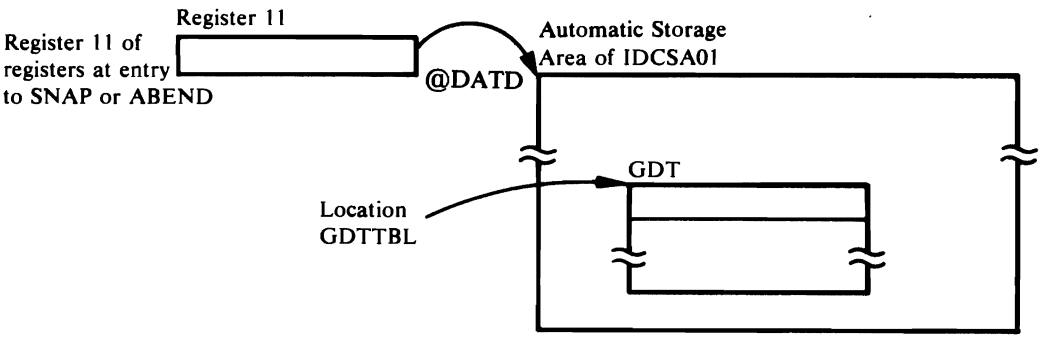
Figure 19, Part 1, illustrates how to find the module that caused the dump and its registers in a snap dump produced through the Test option. In this example, module IDCSA02 called for a dump at the dump point 'ZZCA'. Module IDCDB01 saved the registers of module IDCSA02 in the latter's save area.

How to Find the GDT

The Global Data Table (GDT) is preceded by the identifier GDTb (see Figure 19, Part 4) so you may be able to find it by scanning down the right side of the dump. A more systematic way of finding the GDT depends upon the type of dump you have. Figure 15 shows the two methods of finding the GDT and is referred to in the following paragraphs.

In a snap dump produced as the result of an ABORT condition, you must use Method 1 shown in Figure 15. The GDT is contained in the System Adapter's (IDCSA01) automatic storage area. Register 11 of the registers at entry to SNAP points to the automatic storage area of IDCSA01. The GDT is located at offset GDTTBL in the storage area; you must examine the microfiche listing for IDCSA01 to find the offset from the start of the automatic storage area at @DATD to location GDTTBL. Add the offset of location GDTTBL to the contents of register 11 to obtain the address of the GDT.

Method 1.



Method 2.

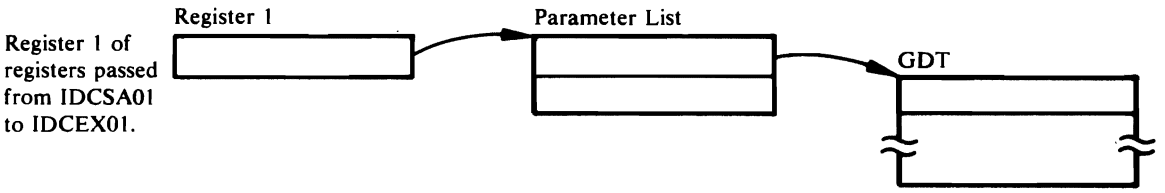


Figure 15. How to Find the GDT

In an ABEND dump, if the ABEND occurred after the call to IDCSA01 but before IDCSA01 calls IDCEX01, then you must again use Method 1. Add the contents of register 11 of the registers at entry to ABEND to the offset of GDTTBL, to find the address of the GDT.

If the ABEND occurred after IDCSA01 called IDCEX01, use Method 2 shown in Figure 15. The address of the GDT was passed as a parameter from IDCSA01 to IDCEX01. You must find the save area where IDCEX01 saved the registers belonging to IDCSA01. Register 1 in this save area contains the address of a parameter list. The first word in the parameter list contains the address of the GDT.

In a snap dump produced as a result of the Test option, you can most easily find the GDT using Method 2. Find the save area where IDCEX01 saved the registers belonging to IDCSA01. Register 1 in this save area contains the address of a parameter list. The first word in the parameter list contains the address of the GDT.

The GDT is the "anchor" for all areas of the processor. In the GDT are found pointers to the trace tables, to the historical areas, and to the entry points of the System Adapter, the I/O Adapter, and the Test Processor.

Figure 19 shows the GDT as it appears in a dump. Part 1 of Figure 19 shows the registers belonging to IDCSA01 and saved by IDCEX01. Register 1 points to the parameter list. Part 4 of Figure 19 shows the parameter list and the GDT.

How to Find Save Areas

The first word of the standard save area for processor modules contains the ID of the module that saved its caller's registers in that save area. (The module ID is the last four characters of the module name.) For example, if the first word of the save area contains DE01, then you would know that IDCDE01 saved its caller's registers in this area. The remainder of the save area is set up following standard register saving conventions.

The save area chain normally appears in the formatted area of a dump. In addition, the start of the Access Method Services save area chain can be found in GDTABH in the Global Data Table (GDT). You must examine the microfiche listing of IDCSA01 to find the offset of location GDTABH.

Figure 19 shows the save areas at the start of a dump.

How to Find the Trace Tables

The trace tables can easily be found once you have found the GDT. The third word of the GDT (including the GDT identifier) points to the Inter-Module Trace Table; the fourth word of the GDT points to the Intra-Module Trace Table.

Several areas in a dump may look as if they contain the trace tables; however, these areas may simply be areas used in constructing the trace tables.

Figure 19, Part 4, shows how the trace tables appear in a dump. Note that the last (twentieth) trace point in the Intra-Module Trace Table is SASN; IO01 is not part of the trace table.

Note: If, in the Inter-Module Trace Table, the sequence SA02 SA02 occurs, the second SA02 is really the ID for module IDCIO02.

How to Find the FDT

You can find the Function Data Table (FDT) for an FSR after the FSR has received control by finding the save area in which the FSR saved the registers belonging to IDCEX01. The first word of this save area contains the module ID of the FSR, for example, PR01 for the PRINT FSR. The preceding save area in the save area trace contains EX01 in the first word. Register 1 in the save area where the FSR saved registers contains the address of a parameter list. The second word of that parameter list contains the address of the FDT.

All FDTs are built by the Reader/Interpreter in a UGPOOL storage area obtained by the Executive; the UGPOOL area has an ID of EX00. The first two words of the FDT contain the name of the command.

Figure 19 shows how an FDT looks in a dump. Part 1 of Figure 19 shows the registers belonging to IDCEX01 and saved by IDCDL01. Register 1 points to the parameter list. Part 3 of Figure 19 shows the parameter list and the FDT.

How to Find Automatic Storage Areas

The automatic storage area for a module is that storage area obtained for the module whenever the module is entered; dynamic storage areas, on the other hand, are those storage areas obtained by the module as it is executing. All automatic storage areas, as well as dynamic storage areas, are obtained by the System Adapter.

The automatic storage area for most processor modules is preceded by an eight-byte header. The first four bytes contain the number of bytes in the automatic storage area (including the eight-byte header), and the last four bytes contain the module ID. However, for commonly called modules, namely, IDCIO01, IDCSA02, IDCSA03, and IDCTP01, no header precedes the storage area, unless the module has been called recursively. On recursive calls (that is, the module has been called again within the original call), the storage area that is obtained is preceded by an eight-byte header.

The best way to find the automatic storage area for a module depends upon the module.

The address of the automatic storage area for module IDCSA03 is kept in the GDT.

The addresses of the automatic storage areas for modules IDCIO01, IDCIO05, IDCSA02, IDCSA06, IDCSA07, IDCSA08, IDCSA09,IDCSA10, and IDCTP01 are kept by the System Adapter in the AUTOTBL. Figure 16 shows the format of the AUTOTBL and how to find it. However, if any of these modules have been called recursively, indicated by a use count in the AUTOTBL greater than one, another automatic storage area has been obtained. You must find the second and third storage areas using the module's data register or save area register as explained in the next paragraphs.

Figure 19, Part 4, shows how the System Adapter Historical Area and AUTOTBL appear in a dump.

To find the automatic storage area for any module, you can examine the microfiche listings to find which register has been used by the compiler as the data register. This register, usually register 11, points to the automatic storage area.

For all processor modules, the first item in the automatic storage area is the save area. Thus, you can also use register 13, which contains the address of the save area, to find the automatic storage area belonging to that module.

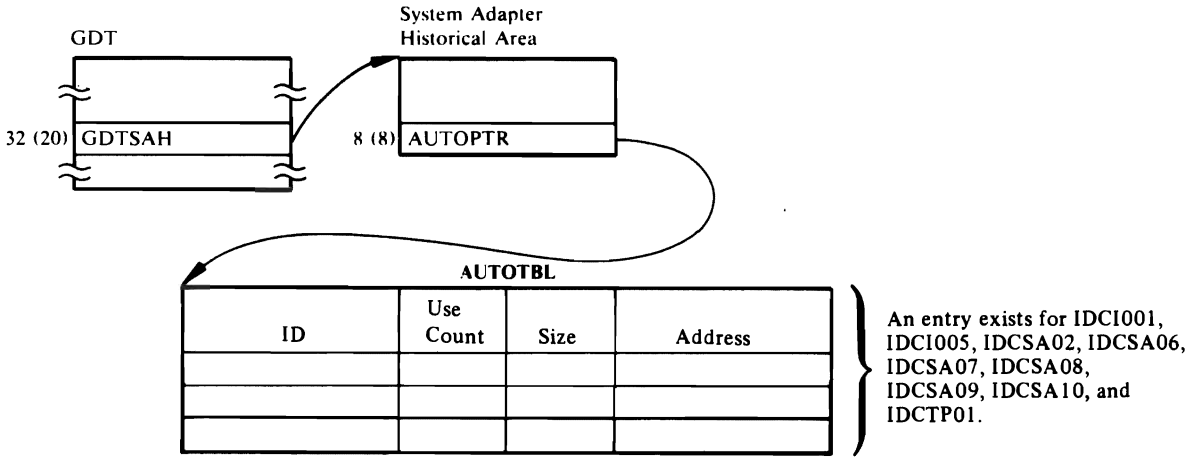
Figure 17 shows the automatic storage area for module IDCEX01. Module IDCEX01 has called IDCDL01; therefore, module IDCDL01 has saved the registers belonging to IDCEX01 in the save area.

Figure 19, Part 2, shows an automatic storage area as seen in a dump.

How to Find Dynamic Storage Areas

The dynamic storage area is that area obtained by the module as it is necessary; the automatic storage area, on the other hand, is that storage area obtained for the module whenever the module is entered. All dynamic storage areas, as well as all automatic storage areas, are obtained by the System Adapter. A module obtains storage areas dynamically by issuing either a UGSPACE or a UGPOOL macro.

To find a storage area obtained via a UGSPACE macro, you must examine the microfiche listings to see where the module has saved the address of that



Field	*Contents*
ID	Module ID, 'IO01', 'IO05', 'SA02', 'SA06', 'SA07', 'SA08', 'SA09', 'SA10' or 'TP01'
Use Count	Number of automatic storage areas obtained for the module: 0 – no storage area being used 1 – the storage area whose address is in this table is the only storage area being used >1 – another storage area has been obtained for the module
Size	Number of bytes in automatic storage area
Address	Address of automatic storage area

Figure 16. Format of AUTOTBL

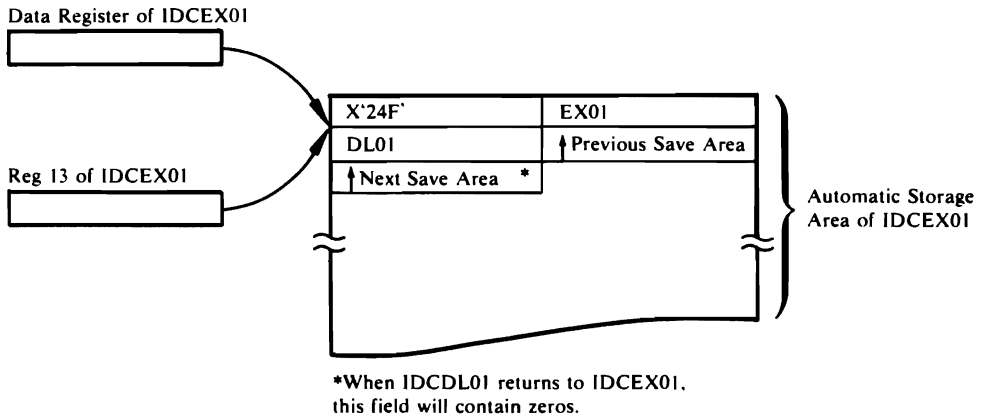


Figure 17. Example of an Automatic Storage Area

particular storage area. To find a storage area obtained via a UGPOOL macro, you can again examine the microfiche listings or you can follow the UGPOOL storage chain maintained by the System Adapter.

Figure 18 shows how to find the chain of UGPOOL areas from the System Adapter's historical area.

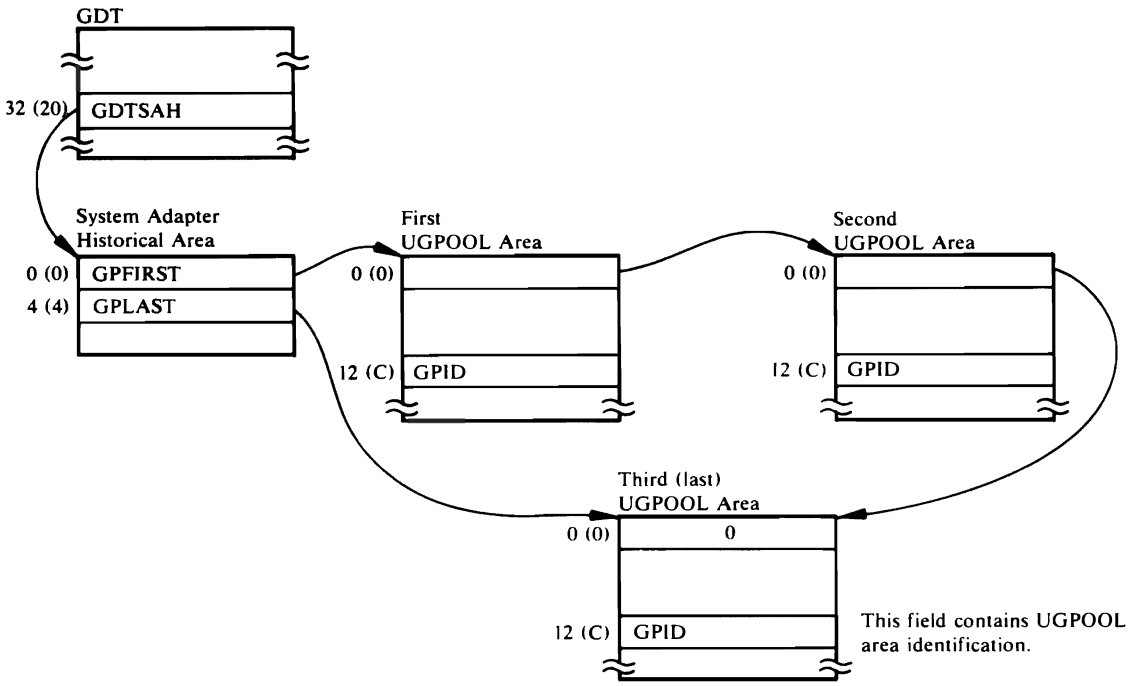


Figure 18. UGPOOL Area Chain

Contents of UGPOOL Areas

The following list contains the UGPOOL IDs used by different modules when they obtain storage. The list also contains the name of the internal procedure that issues the UGPOOL macro, and the contents stored in the UGPOOL area.

Figure 19 shows the UGPOOL chain as it appears in a dump. Part 4 of Figure 19 shows the start of the chain in the GDT. Part 3 shows the chain.

Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCAL01	AL00	ALTERPRC	One of the following: PASSWALL field or volume list.
		IDCAL01	CTGPL, CTGFV, and CTGFLs. Catalog entry names returned from a generic locate. UGPOOL area is obtained by IDCSA02(UCIR).
		INDEXPRC	CTGPL, CTGFV, CTGFL and IDAAMDSB field.
		LOCATPRC	Catalog work area for locate requests.
		DALCPROC	Volume list and ALLAGL parameter list.
IDCBI01	BI01	JCPROC	Area obtained by UIOINFO to contain sort work file data set name and volume serial list; passed back to JCPROC.
	BIPG	INITPROC	One 2048 byte buffer, followed by area for define CTGPL, CTGFVs, and CTGFLs followed by alternate index record output buffer; area starts on page boundary.
	BIPG	INITPROC	Record sort area followed by table which controls the sort.

Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCCC01	CC00	CATALOG	A new VSAM catalog work area if VSAM return code indicates insufficient space.
		CNVTINIT	All of the following: OS/VS SYSCTLG index blocks for 22 index levels, VSAM parameter lists, data addressed by the VSAM parameter lists, VSAM catalog work area.
IDCCK01	CK00	BUILDTAB	Area where table of checkpoint IDs is built.
IDCDE01	DE00	DALCPROC	List of volumes to be mounted and ALLAGL parameter list.
		IDCDE01	CTGPL, and CTGFVs.
IDCDE02	DE00	ALLCPROC	One of the following: volume list, file sequence list, device type list, or CTGFLs (DSATTR, LRECL, BUFSIZE, SPACPARM, DEVTYP, FILESEQ).
		KEYPROC	One or both of the following: AMDSBCAT CTGFL and IDAAMDSB field, or key range list.
		MODELPRC	One of the following: CTGPL and CTGFLs used to locate a model object, or catalog locate work area.
		NAMEPROC	Creation, expiration date and exception exit CTGFLs and GDG attributes.
		PROTPROC	PASSWALL CTGFL, OWNERID CTGFL, PASSWALL field, User Authorization Record, RGATTR CTGFL and RGATTR field.
IDCDL01	DL00	DCDL01	Catalog entry names returned from a generic locate. UGPOOL area is obtained by IDCSA02(UCIR).
	DL01	MORESP	Larger VSAM catalog management services work area if necessary.
IDCIO01	IO00	IDCIOIT	I/O Adapter historical area and VSAM message area.
	IO*nn*	PUTREP	Work area where VSAM moves records during GET. The UGPOOL ID is the same as the ID for the associated IOCSTR.
IDCIO02 **(VS2.03.808)**	10*nn*	BUILDACB	ACB, RPL, EXLST for a VSAM data set. The UGPOOL ID is the same ID as the associated IOCSTR.

Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCIO02	IO*nn*	BUILDDBK	JFCB and exit list if OPENJ processing; DCB; DECB if BSAM update processing. The UGPOOL ID is the same as the ID for the associated IOCSTR.
		BUILDRPL	Work area where VSAM moves records during GET. The UGPOOL ID is the same as the ID for the associated IOCSTR.
		CKNONOP	Work area used to assemble an ISAM data set with fixed records and non-imbedded keys. The UGPOOL ID is the same as the ID for the associated IOCSTR.
		OPENRTN	IOCS prefix, IOCSTR, IOCSEX, and data set name. Each data set that is opened is assigned a unique UGPOOL ID, starting with IO01; the next data set that is opened is assigned an ID of IO02. All areas associated with a data set have the save UGPOOL IDs.
IDCIO03		DSINFO	Area in which data set name, volume serial numbers, device type, and/or Format-4 time stamp is returned to the caller if an area is not supplied by the caller. The UGPOOL ID is supplied by the caller.
IDCLC01	LC00	INITPROC	Main CTGPL used for all locate requests except when locating the entry names of associated entries. This area also contains a save area for the CTGPL.
	LC01	INITPROC	All CTGFLs, followed by the CTGFL save area.
	LC02	INITPROC	Catalog work area referenced by the main CTGPL.
	LC03	INITPROC	CTGPL used to locate entry names of associated entries. This area also contains a save area for the CTGPL.
	LC04	INITPROC	Catalog work area referenced by the CTGPL used to locate entry names of associated entries.
	LC05	INITPROC	String of control interval numbers and types of associated entries for cluster, AIX, GDG and pagespace.
	LC06	INITPROC	Text processor argument list.
	LC07	INITPROC	Abbreviations used in catalog listing, loaded from static text module.
	LC10	ENTPROC	Catalog entry names returned from a generic locate. UGPOOL area is obtained by IDCSA02(UCIR).
	LC11	INITPROC	String of control interval numbers and types of associated entries for data, index, path and nonVSAM.
IDCLC02	LC08	LOCPROC	Larger catalog work area. UGPOOL LC02 is released.
	LC09	CDIPROC	Larger area for string of control interval numbers and types of associated entries. UGPOOL LC05 or LC11 is released.

Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCMP01	MP01	BFPLPROC	CTGFL.
		BPASPROC	PASSWALL CTGFL.
		CLUSPROC	Buffer to read data records from the portable data set.
		CTLGPROC	Larger catalog work area.
		DALCPROC	List of volumes to be mounted.
		DELTPROC	Larger VSAM catalog management services work area if necessary.
		LVLRPROC	One of the following: volume list for define, or DEVTYPE CTGFL.
		RANGPROC	Range list.
		FVTPROC	FVT and pointers to FPLs.
IDCRC02	RC02	IDCRC02	Output buffer for header record.
		CLUSPROC	Output buffer for control record.
		LOCPROC	Buffer for FPL's and catalog work area.
		CTLGPROC	Buffer for larger catalog work area..
		NVSMPROC	Output buffer for control record.
		SAVEPROC	Buffer for catalog control records.
		GDGPROC	Output buffer for control record.
		ASOCPROC	Output buffer for control record.
IDCRI01	EX00	GETSPACE	FDT—data substructures.
		MORSPACE	FDT—data list substructures.
		SCANCMD	FDT—secondary pointer vectors.
	RI*nn*	INREPEAT	FDT—temporary space for secondary pointer vectors. *nn* is the ID of the parameter associated with the secondary pointer vector.
IDCRI02	EX00	IDCRI02	One of the following: Reader/Interpreter tables, or FDT.
	RI*nn*	IDCRI02	FDT—temporary space for secondary pointer vectors. *nn* is the ID of the parameter associated with the secondary pointer vector.
IDCRI04	EX00	GETSPACE	FDT data substructures.
		REPLIST	FDT secondary address vectors.
		RISETUP	FDT primary address vector and secondary count vectors.
		SETDFLT	Default parameter work area.
		SUBSCAN	Work area for input command.
	RI04	REPLIST	Additional space for tables used only by the Reader/Interpreter.
		RISETUP	Tables used only by the Reader/Interpreter.

Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCRM01	RM01	ALISPROC	Catalog data record buffer.
		BFPLPROC	Obtain one or two FPLs.
		BPASPROC	Contain PASSWALL field information.
		CLUSPROC	Buffer area for data record containing catalog locate area. Also volume list.
		CPLPROC	Catalog parameter list.
		CTLGPROC	Larger catalog work area.
		DALCPROC	List of volumes to be mounted.
		DELTPROC	Larger catalog work area.
		FVTPROC	FVT and pointers to FPLs.
		GDGPROC	Storage for data record.
		LVLRPROC	Volume serial list. DEVTYP FPL and associated device type lists. List of FILESEQUENCE numbers and associated FPL.
		NFVTPROC	FVT and total number of FPLs.
		NVSMPROC	Buffer for data record.
		RANGPROC	Storage for range list.
		UCATPROC	Storage for data record.
IDCRS01 **(VS2.03.808)**	RS01	IDCRS01	Automatic storage for modules IDCRS01-IDCRS07.
	RS01	INIT	Work area used for umacro parameter lists, record access blocks, and control interval translate table.
	RS03	INIT	Area obtained by UIOINFO for catalog data set information.
IDCRS03 **(VS2.03.808)**	RS10	GETTAB	Tables obtained as needed for association checking.
	RS11	PROCVOL	Work areas used for bit maps.
	RS12	VERB	Work area used for GDG association checking.
IDCRS04 **(VS2.03.808)**	RS04	NINIT	Work area used for FIND processing.
	RS04	NXPND	Extension to FIND work area.
IDCRS05 **(VS2.03.808)**	RS01	BLDRLST	RESVOL table.
	RS02	BLDVLST	VOLSERTB table.
IDCRS06 **(VS2.03.808)**	RS03	WFDEF	Work area used for UCATLG parameter list to define work file, area obtained by UIOINFO for work file data set information.
IDCRS07 **(VS2.03.808)**	RS03	RENMSETV	CAMLST for RENAME.
	RS03	SCRATCHP	Volume list for SCRATCH.
IDCTP01	TP03	LINEPRT	Header line.
IDCTP04	TP01	INITPROC	Secondary Print Control Table.
		PCTSETUP	One of the following: Print Control Table, subtitle lines, or footing lines.
IDCTP05	TP01	IDCTP05	Entry from a static text format structure.

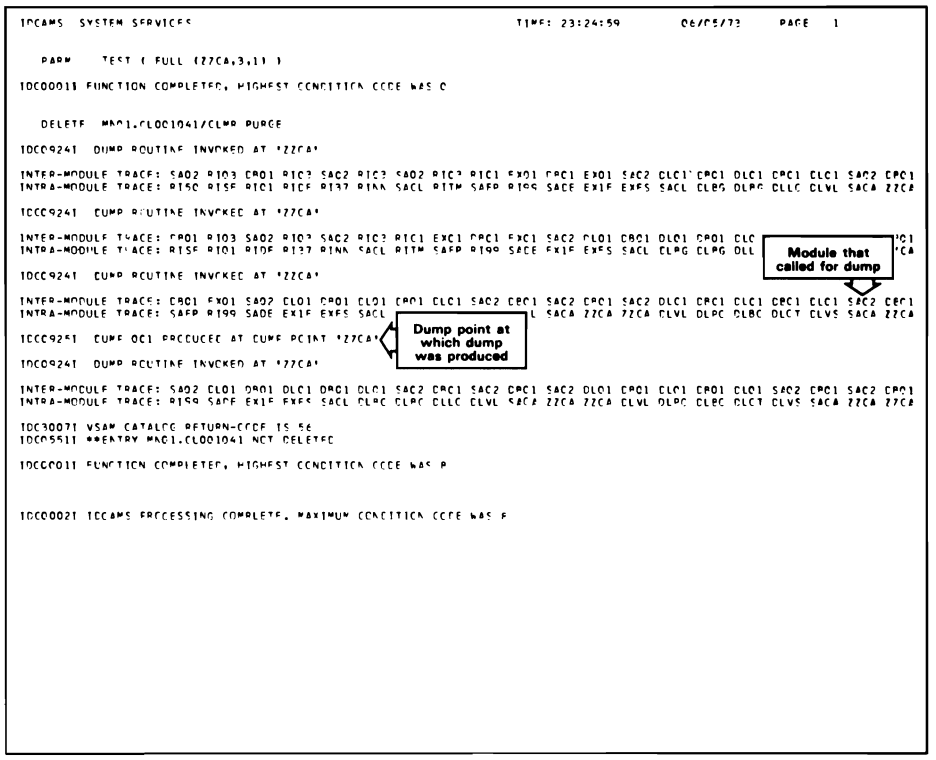
Module	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCXP01	XP01	ALTRPROC	CTGFV and CTGFLs for catalog alter request.
		CLUSPROC	Output buffer for control records.
		CTLGPROC	Larger catalog work area.
		DELTPROC	CTGPL for catalog delete request.
		LOCPROC	One of the following: CTGPL and CTGFLs for catalog locate request, or catalog work area for locate request.

Sample Dump

The dump displayed in Figure 19 was obtained through the Test option at the ZZCA dump point. The PARM command was specified as follows:

```
PARM TEST( FULL( ZZCA,3,1 ) )
```

Various fields within the dump are marked; these fields are discussed more fully in this chapter.



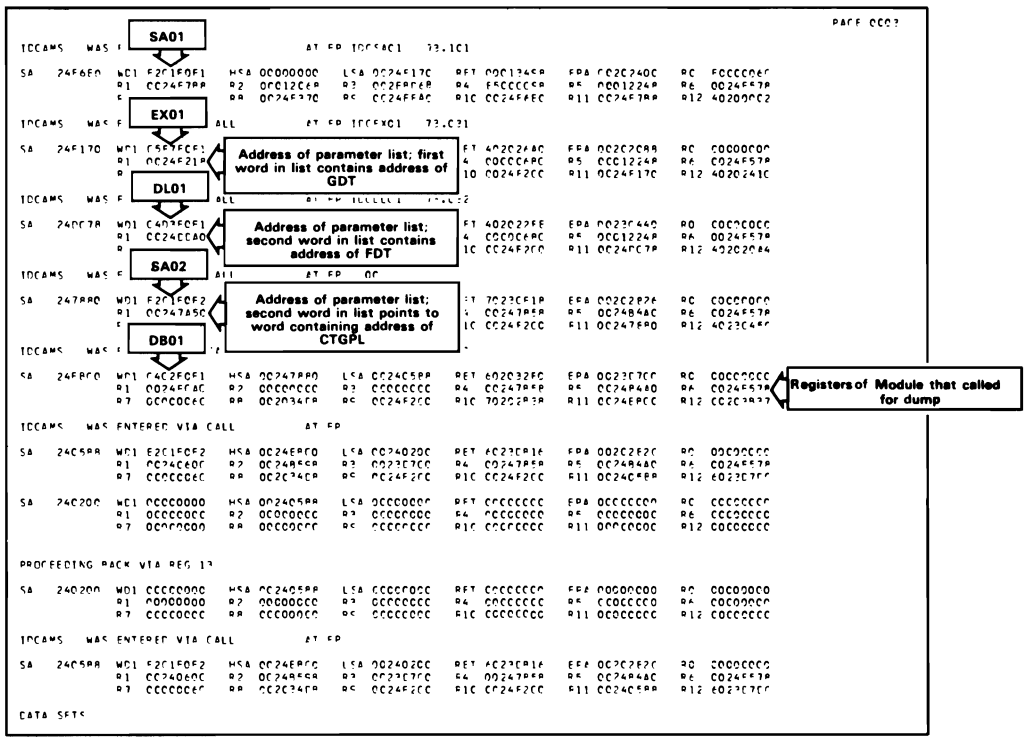
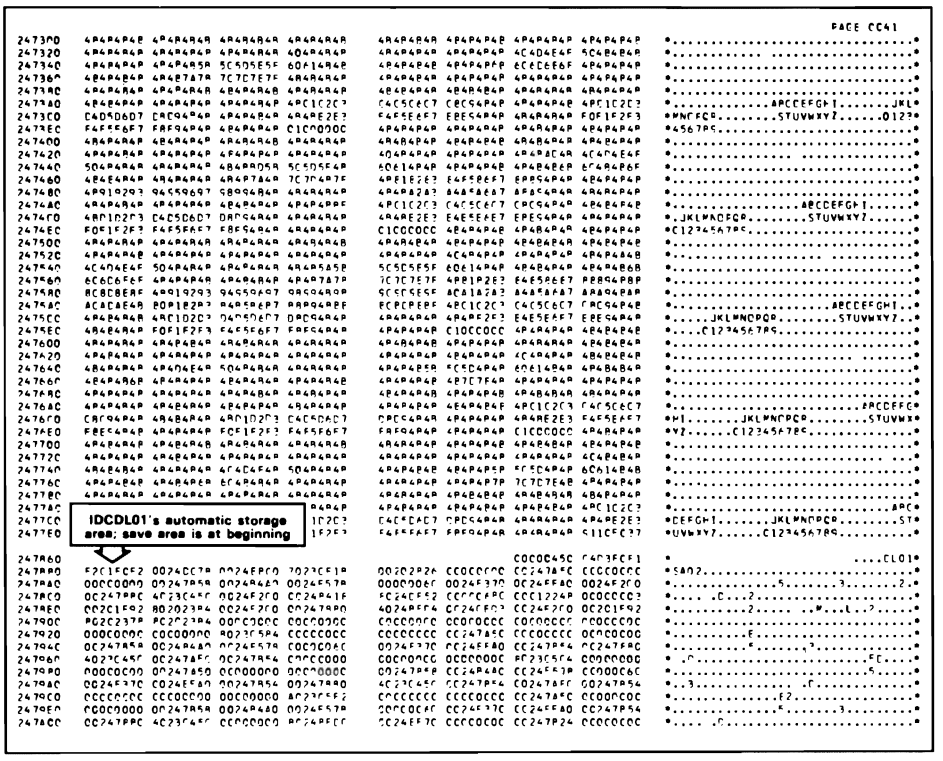


Figure 19 (Part 1 of 4). Sample Dump



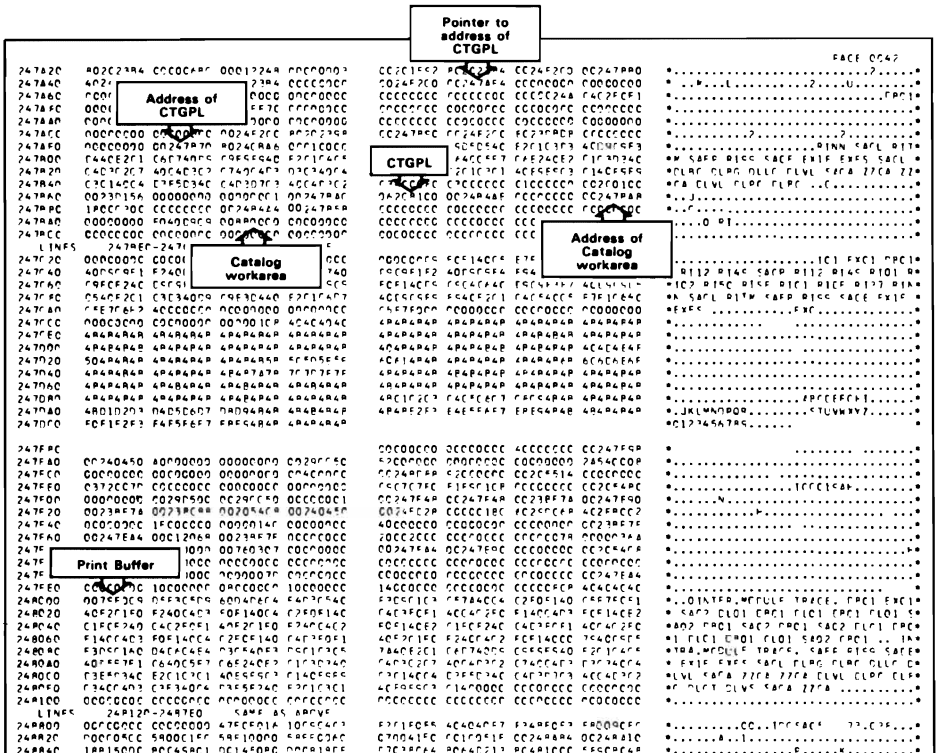
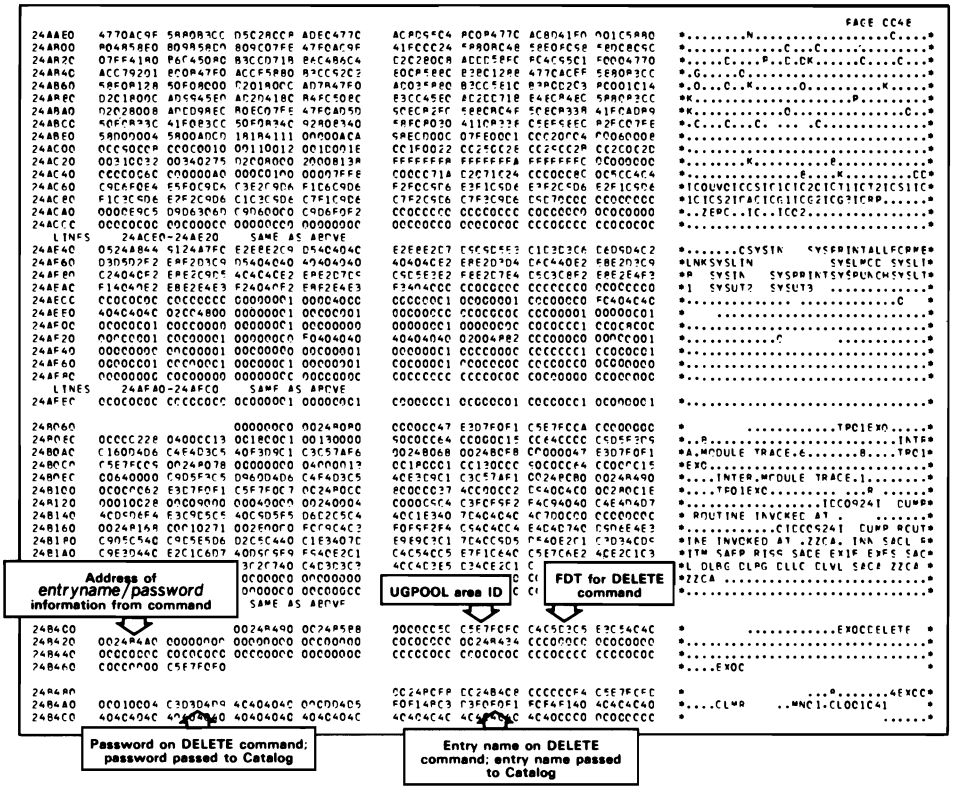


Figure 19 (Part 2 of 4). Sample Dump



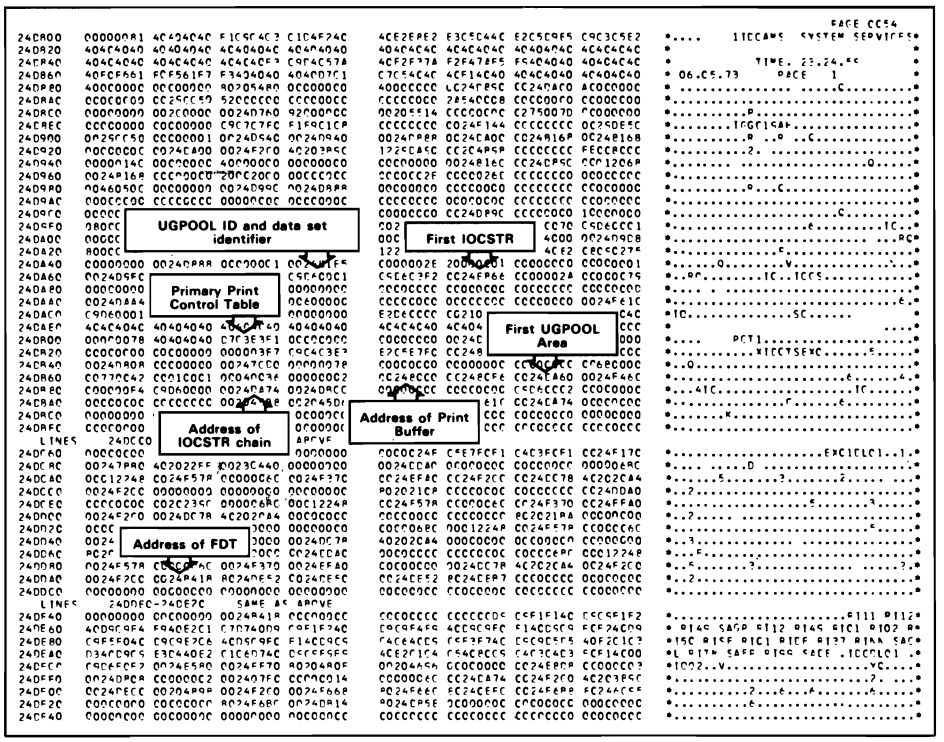


Figure 19 (Part 3 of 4). Sample Dump

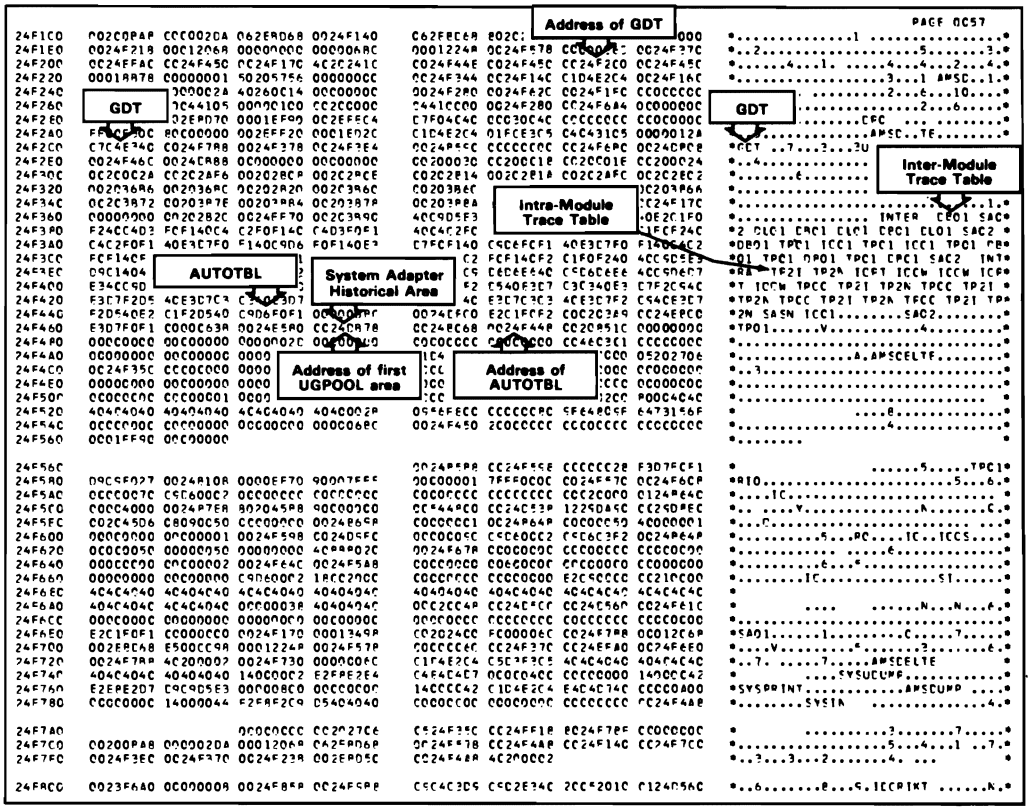


Figure 19 (Part 4 of 4). Sample Dump

Debugging a Catalog Problem

There may be a problem within Catalog Management routines or within Access Method Services routines that invoke Catalog Management if one of the following situations occurs: an ABEND occurs within Catalog Management routines, the return code from the catalog indicates a non-user error, or the printed output from the catalog is incorrect. To determine whether the problem exists in Access Method Services or in Catalog Management, you must examine the argument lists passed between the processor and Catalog Management.

This section explains how to obtain a dump that contains the Catalog Management argument lists and how to find the argument lists within the dump.

To determine whether the argument lists passed between the processor and Catalog Management are correct, see the chapter "Method of Operation" and the logic manual *OS/VS2 Independent Component: Virtual Storage Access Method (VSAM) Logic*, which is listed in the preface. The chapter "Method of Operation" explains what argument lists are passed to Catalog Management by each FSR; *OS/VS2 Independent Component: Virtual Storage Access Method (VSAM) Logic* explains the contents of the argument

lists and also explains the arguments that are returned by Catalog Management.

Obtaining a Dump for a Catalog Problem

If you do not have an ABEND dump within Catalog Management, you can use the Test option to obtain a dump within Access Method Services before and after the call to Catalog Management.

The "Module or CSECT to Dump Points Cross Reference" list contains all the dump points within the processor; you can specify these dump points on the FULL option of the TEST keyword to obtain a full region dump. Each FSR that issues a UCATLG macro to call Catalog Management has dump points before and after each macro. In addition, the System Adapter module that issues the Catalog Management SVC, SVC 26, has a dump point before and after the SVC.

Some FSRs have unique dump points around different types of calls to Catalog Management. For example, IDCDL01 has dump points DLVL around the call to locate the entry type and dump points DLVS around the call to delete the entry. Some FSRs have the same dump point around all calls to Catalog Management, for example, IDCMP01. Some FSRs have dump points at which you can obtain selected fields in addition to a full region dump, for example, dump points LCBL and LCAL in IDCLC01.

The System Adapter dump point ZZCA can always be used, for any FSR, to obtain dumps before and after a call to Catalog Management.

To determine at which iterations of a dump point you wish a full region dump, you must determine how many calls to Catalog Management have been made by the FSR before the call that caused the problem. You can either refer to "Sequence of Calls Made by FSR" or rerun the job with the AREAS option.

"Sequence of Calls Made by FSR" summarizes the sequence of calls each FSR makes to Catalog Management. Using this summary, assume that the LISTCAT FSR, IDCLC01, while listing all the information for an index cluster entry, listed the cluster name under the index entry incorrectly, you would know that the call to the catalog that retrieved that name was the seventh call the LISTCAT FSR made to Catalog Management.

Instead of using this summary, you can rerun the job with the AREAS option of the TEST keyword to determine which iteration of a dump point you need to use.

For example, if you wish to use dump point ZZCA to obtain a dump, rerun the job with the following Test option:

```
PARM TEST( AREAS( ZZ ) )
```

From the trace output you can see how many times dump point ZZCA was encountered before the problem occurred.

How to Find Catalog Management Argument Lists

The Catalog Parameter List (CTGPL) is the one argument list always passed between Access Method Services and Catalog Management. The CTGPL may point to a catalog work area, a CTGFV, or one or more CTGFLs. Thus, once you find the CTGPL, you can find all the Catalog Management argument lists.

The best way to find the CTGPL in a dump depends upon the type of dump you have: an ABEND dump within Catalog Management, a snap dump taken at a dump point within an FSR, or a snap dump taken at the ZZCA dump point in the System Adapter.

In an ABEND dump within Catalog Management, register 1 of the registers saved when SVC 26 was entered contains the address of the CTGPL.

In a snap dump taken at a dump point within an FSR, the address of the CTGPL is stored at location CTGPLPTR in the FSR's automatic storage area. You must examine the microfiche listings to determine the offset of location CTGPLPTR in the automatic storage area.

In a snap dump taken at dump point ZZCA within the System Adapter, the address of the CTGPL is again stored at location CTGPLPTR in the FSR's automatic storage area. However, the address of the CTGPL is also passed as an argument from the FSR to IDCSA02 when the UCATLG macro is issued.

Figure 20 shows how to find the address of the CTGPL using register 1 at entry to IDCSA02. Register 1 contains the address of a parameter list. The second word of the parameter list points to a full word that contains the address of the CTGPL.

In addition to the CTGPL, Catalog Management returns to the processor a code in register 15 that indicates the result of the catalog request. The best way to find the return code in a dump again depends upon the type of dump

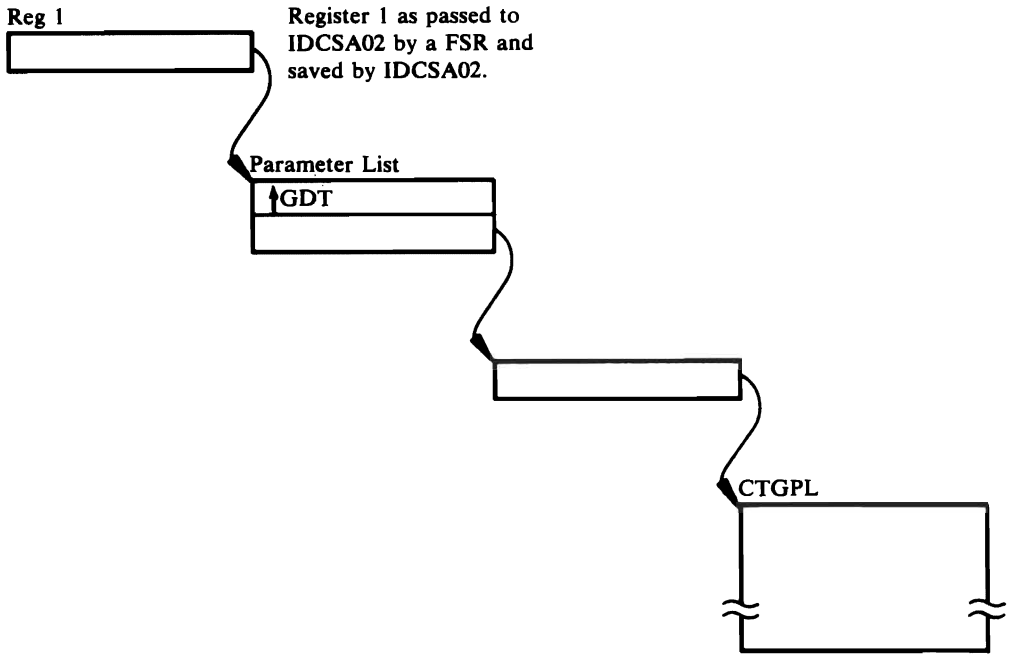


Figure 20. How to Find the CTGPL

you have: a snap dump taken at a dump point within an FSR, or a snap dump taken at dump point ZZCA.

In a snap dump taken at a dump point within an FSR, you must examine the microfiche listings to determine where the FSR has stored the return code. However, any non-zero return code is always printed by the FSR in a subsequent message.

In a snap dump taken at a dump point within the System Adapter, the catalog return code is stored at location TESTRC in IDCSA02's automatic storage area. You must examine the microfiche listings to determine the offset of TESTRC in the automatic storage area.

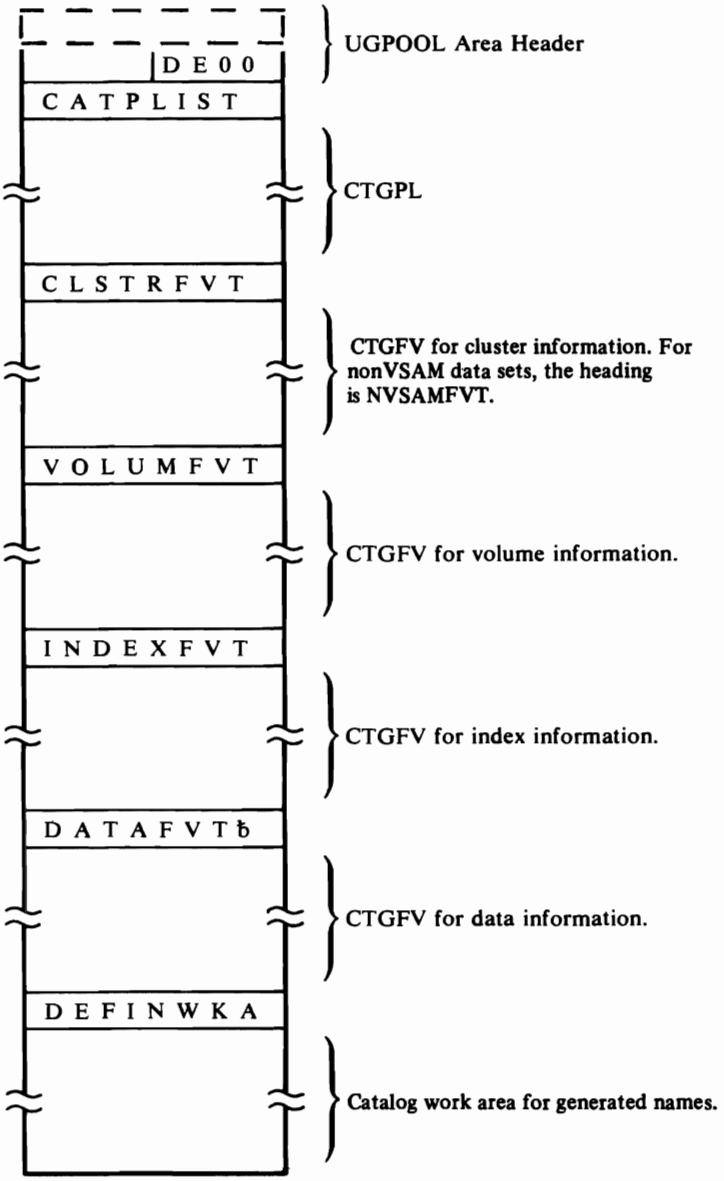
Some FSRs have headings before the storage areas that contain the Catalog Management argument lists. These headings may help you find the Catalog Management argument lists in a dump. Figure 21 shows the DEFINE FSR's storage area that contains the argument lists set up for a define request.

Sequence of Catalog Calls Made by FSR

The following table summarizes the sequence of calls each FSR makes to Catalog Management.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCAL01	1. A call to locate catalog fields if one of the following fields is being nullified or altered: MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, AUTHORIZATION, ERASE | NOERASE, SHAREOPTIONS, FREESPACE, WRITECHECK | NOWRITECHECK, UNINHIBIT | INHIBIT,UPGRADE, UNIQUEKEY, NONUNIQUEKEY, KEYS, or RECORDSIZE, SCRATCH | NOSCRATCH, EMPTY | NOEMPTY, ADDVOLUMES | REMOVEVOLUMES, STAGE | BIND | CYLINDERFAULT, DESTAGEWAIT | NO. If UPGRADE was supplied: 1. A call to locate the associated data component of the alternate index to verify that it is empty. 2. A call to alter the alternate index entry. If RECORDSIZE was supplied for the data object: 1. A call to locate the cluster or alternate index associated with the data object. 2. A call to locate the index associated with the cluster or alternate index related to the data object. 3. A call to alter the data entry. If RECORDSIZE was supplied for the cluster or alternate index object: 1. A call to locate the associated data object. 2. A call to locate the associated index object. 3. A call to alter the data entry.



If any of the above CTGFVs are not set up for a define request, the heading and CTGFV area contains zeros.

Figure 21. Catalog Argument Lists in Storage Area of DEFINE FSR

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCAL01 (cont'd)	If RECORDSIZE was supplied for the path object: 1. A call to locate the data object of the related alternate index or cluster. 2. A call to locate the index object of the related alternate index cluster, or cluster. 3. A call to alter the data entry. If KEYS was supplied for the data object: 1. A call to locate the cluster or alternate index associated with the data object. 2. A call to locate the index associated with the cluster or alternate index related to the data object. 3. A call to locate the alternate index's base cluster, if the data object is associated with an alternate index. 4. A call to locate the data object of the base cluster. 5. A call to alter the data entry. 6. A call to alter the related index object key values. If KEYS was supplied for the cluster object: 1. A call to locate the associated data object. 2. A call to locate the associated index object. 3. A call to alter the data entry. 4. A call to alter the related index object key values. If KEYS was supplied for the alternate index object: 1. A call to locate the associated data object. 2. A call to locate the associated index object. 3. A call to locate the base cluster object. 4. A call to locate the base cluster's data object. 5. A call to alter the data entry. 6. A call to alter the related index object key values. If KEYS was supplied for the path object: 1. A call to locate the data object of the related alternate index or cluster. 2. A call to locate the index object of the related alternate index or cluster. 3. A call to locate the base cluster's data object, if the path is related to an alternate index. 4. A call to alter the related entry's data object. 5. A call to alter the related index object's key values.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCBI01	1. A call to locate the catalog ACB, entry type and associations of the name specified for the base cluster—may be the base cluster itself or a path over the base cluster. 2. A call to locate the AMDSB of the base cluster's data component. 3. A call to locate the entry type and associations of the name specified for the alternate index—may be the alternate index itself or a path over the alternate index. 4. If locate 3 returned a path over the alternate index, a call to locate the entry type and associations of the alternate index. 5. A call to locate the AMDSB of the alternate index's data component. If an external sort is performed: 1. Two calls to define each sort work file. 2. Two calls to delete each sort work file.
IDCCC01	For each OS/VS catalog entry that is to be moved to the VSAM catalog a call to: 1. Define the data set or alias. 2. Locate information if the name of the OS/VS data set already exists in the VSAM catalog, and the OS/VS data set name does not begin with 'SYS1.'. 3. Alter the VSAM catalog entry, if necessary, to point to the OS/VS data set if the duplicate name in the VSAM catalog is not a VSAM data set, and the conditions under 2. are satisfied.
IDCDE01 (without **VS2.03.807**)	1. Locate each object that is modeled, as follows: three calls if the MODEL keyword is specified in the cluster parameter list for a KSDS cluster or in the user catalog parameter list; two calls if the MODEL keyword is specified in the cluster parameter list for an ESDS cluster or in both the data and index parameter lists; one call if the MODEL keyword is specified in a data parameter list or an index parameter list only. 2. Define the entire entry.
IDCDE01 (with **VS2.03.807**)	1. Define the entire entry.
IDCDE02 (with **VS2.03.807**)	2. Locate each object that is modeled, as follows: three calls if the MODEL keyword is specified in the cluster parameter list for a KSDS cluster or in the user catalog parameter list; two calls if the MODEL keyword is specified in the cluster parameter list for an ESDS cluster or in both the data and index parameter lists; one call if the MODEL keyword is specified in a data parameter list or an index parameter list only.
IDCDL01	For each entry: 1. Locate the entry type, if the type was not specified on the command. 2. Delete the entire entry. 3. An iterative series of calls to delete any remaining parts of a structure as necessary.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCLC01	For each alias entry: 1. Locate the alias entry. 2. Locate the true entry. For each cluster entry: 1. A call to locate the cluster entry. 2. A call to locate the name of the data entry associated with the cluster entry. 3. A call to locate the name of the index entry associated with the cluster entry, only for KSDS clusters. 4. Repetitive calls to locate the names of the alternate indexes and paths associated with the cluster entry (if any exist). 5. A call to locate the data entry. 6. A call to locate the name of the cluster entry associated with the data entry. 7. A call to locate the index entry, only for KSDS clusters. 8. A call to locate the name of the cluster entry associated with the index entry. 9. Repetitive calls to locate the path entries (if any exist). 10. Repetitive calls to locate the cluster, data, and index (for key-sequenced files) associated with the path entries. For each data entry: 1. Locate the data entry. 2. Locate the name of the cluster or alternate index entry associated with the data entry. For each generation data group entry: 1. Locate the generation data group entry. 2. Locate the nonVSAM generation index data set names. 3. Locate each generation index data set. 4. Locate the generation data group names associated with the nonVSAM entry. For each index entry: 1. Locate the index entry. 2. Locate the name of the cluster or alternate index entry associated with the index entry.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCLC01 (Cont.)	For each alternate index entry: 1. A call to locate the alternate index entry. 2. A call to locate the name of the data entry associated with the alternate index entry. 3. A call to locate the name of the index entry associated with the alternate index entry. 4. A call to locate the name of the cluster entry associated with the alternate index entry. 5. Repetitive calls to locate the names of the paths associated with the alternate index entry (if any exist). 6. A call to locate the data entry. 7. A call to locate the name of the alternate index entry associated with the data entry. 8. A call to locate the index entry. 9. A call to locate the name of the alternate index entry associated with the index entry. 10. Repetitive calls to locate the path entries (if any exist). 11. Repetitive calls to locate the alternate index, data and index (of alternate index), and data and index (of cluster) associated with the path entries. For each path entry: 1. A call to locate the path entry. 2. For a path over a cluster, a call to locate the name of the cluster, and data and index (of cluster) associated with the path entry. 3. For a path over an alternate index, a call to locate the name of the alternate index, data and index (of alternate index), and data and index (of cluster) associated with the path entry. For each non-VSAM entry: 1. Locate the non-VSAM entry. 2. Locate the name of any associations for each generation data group or alias. For each user catalog entry: 1. Locate the user catalog entry. 2. Locate the name of any associations for each alias. For each space entry: 1. Locate the space entry. 2. Locate each data set name in a space entry, for example, three calls if three data sets are defined in the data space.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCMP01	1. Define the cluster or alternate index. 2. Locate the cluster or alternate index entry if the previous define failed because of a duplicate entry in the catalog. 3. Locate the data entry, only for a duplicate cluster entry. 4. Locate the index entry, only for a duplicate KSDS cluster or alternate index entry and if the temporary export flag is not set in the data entry. 5. Delete the entry, if there is a duplicate nonempty entry. 6. An iterative series of calls to delete any remaining parts of the structure. 7. Define the cluster again, if there was a duplicate entry. 8. Delete the defined entry, if an error occurred copying data into the defined entry. 9. An iterative series of calls to delete any remaining parts of the structure. 10. Alter the data entry, if the INHIBITTARGET keyword was specified at export time. 11. Alter the index entry, if the INHIBITTARGET keyword was specified at export time for a KSDS cluster.
IDCRC01	For VSAM clusters: 1. A call to locate the cluster entry. 2. A call to locate the data entry. 3. A call to locate the index entry for a KSDS or AIX. 4. Repetitive call to locate path entries, if they exist for a VSAM cluster. For GDGs: 1. A call to locate the GDG entry. 2. A call to locate any nonVSAM association to the GDG. 3. Repetitive calls to locate alias entries for the nonVSAM association. For nonVSAM entries: 1. A call to locate the nonVSAM entry. 2. Repetitive calls to locate alias entries if present.
IDCRM01	1. A call to define the object. 2. A call to delete the object if a duplicate name is indicated following the first call to catalog. 3. A series of calls to catalog to delete the remainder of the structure. 4. A call to define the object if a duplicate name was found. 5. A call to alter the name of the object if it is a VSAM entry and OUTFILE was specified to the dummy name specified on the OUTFILE ddcard. 6. A call to alter the name of the object back to its original name if the previous call was executed. 7. A call to delete the object defined if import fails after the define. 8. A series of calls to catalog to delete the remainder of the structure.
IDCRP01	1. Locate the INFILE data-set type. 2. Locate the OUTFILE data-set type.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of Calls to Catalog Management
IDCRS01 **(VS2.03.808)**	1. A call to locate the catalog data set name.
	2. A call to locate the catalog volume serial number and time stamp.
	3. A call to locate the catalog ACB and data attributes.
	4. A call to locate the ACB of the catalog in which the work file is defined.
IDCRS06 **(VS2.03.808)**	1. A call to define the work file.
	2. A call to delete the work file.
IDCXP01	1. Locate the cluster or alternate index entry.
	2. Locate the data entry.
	3. Locate the index entry for a KSDS cluster or an alternate index.
	4. A call to locate the related base cluster name if the object being exported is an alternate index.
	5. A series of iterative calls to locate catalog information about the path objects associated with the object.
	6. A call to alter the data entry, if TEMPORARY, INHIBITSOURCE, or INHIBITTARGET was specified on the command.
	7. A call to alter the index entry, if TEMPORARY, INHIBITSOURCE, or INHIBITTARGET was specified on the command, and the object is a KSDS cluster or an alternate index.
	8. A call to delete the entry if PERMANENT was specified on the command.
	9. A series of iterative calls to the delete any remaining parts of the structure.

Debugging a Formatting Problem

If data is misformatted, the problem may be in the parameters given to the UPRINT macro. The UPRINT parameters are: (1) the address of the GDT; (2) the address of an alternate IOCSTR or zero; (3) the address of and a DARGLIST data area in storage; and (4) the address of a FMTLIST data area, if it is in storage. If the FMTLIST is in a static text module, the fourth parameter is zero and the DARGLIST contains information to find the FMTLIST. The DARGLIST and the FMTLIST control the formatting of the data. The DARGLIST in general contains information about the input data. within the FMTLIST. The FMTLIST controls the order of formatting by the placement of the substructures. Refer to the "Data Areas" chapter for a detailed description of the GDT, IOCSTR, DARGLIST, and FMTLIST.

Problems are most likely to occur between the DARGLIST and the FMTLIST. The examples show how the Text Processor uses the DARGLIST and FMTLIST to format the data. With each example is a flowchart with blocks keyed to the FMTLIST substructure.

Example I

A module wants to space one line then print data starting in column 10 as shown in Figure 22. The data is in the module's storage rather than in a static text module.

The output is:

70 characters of data starting in column 10

In the module's storage is:

- the data to be printed
- a DARGLIST
- a FMTLIST

The data is:

Offset	Name	Contents	Comments
0	any, INFO for example	70 characters of EBCDIC data	

The DARGLIST is:

Offset	Name	Contents	Comments
0	DARGDBP	↑INFO	Address of the block of data to be printed.
4	DARGRETP	0	The line is to be printed rather than just formatted and returned to the module without printing.
8	DARGSTID	0	No static text module is used—the FMTLIST and data are in the module's storage.
12	DARGILP	70	Number of characters to print.
14	DARGCNT	0	No insert or replication substructures occur in the FMTLIST.
16	DARGRETL	0	Since no data is returned, the length of the return area whose address is in DARGRETP.
18	DARGIND	0	Indicates printing is to start in the column indicated in FMTLIST. No DARGARY is defined because no insert or replication substructures are used in the FMTLIST.

The FMTLIST is:

	Offset	Name	Contents	Comments
A	0	FMTFLGS	X'40'	Identifies these 6 bytes as a spacing substructure.
	1	none	0	Unused.
	2	FMTSPF	1	Space one line.
	4	FMTSPT	C'R'	Space the number of lines in FMTSPF relative to the last line printed.
	5	none	0	Unused.
B	6	FMTFLGS	X'90'	Identifies these 12 bytes as a block substructure and the end of the FMTLIST.
	7	none	0	Unused.
	8	FMTILEN	70 or 0	If 70 is specified, it is used as the length of the data. If 0 is specified, the length of the converted data is used as the length to print. Since no conversion is being done in this example, the result is the same if 70 or 0 is specified.
	10	FMTIOFF	0	Get the data starting with the first byte.
	Offset	**Name**	**Contents**	**Comments**
	12	FMTOCOL	10	Place the data in output column 10.
	14	FMTOLEN	70	Number of bytes to print. 0 would give the same result since no conversion is being done.
	16	FMTCNVF	0	No conversion is being done on the data addressed by DARGDBP.

Discussion: The spacing substructure causes one line to be spaced.

The next substructure is identified as a block data substructure. The address of the block of data is in DARGDBP. No conversion is to be done on the data. The Text Processor moves the 70 bytes of data to the 10th byte in the next line.

Example II

A module wants to space 2 lines, print a header, space 2 more lines, and print all of a block of data no matter how many lines the block of data takes with single spacing between subsequent lines (see Figure 23). The header is in static text module IDCTSAL0 at entry X'03'. The block of data is in the module. Also, if there is no record number for the header, the module wants to print the word UNKNOWN.

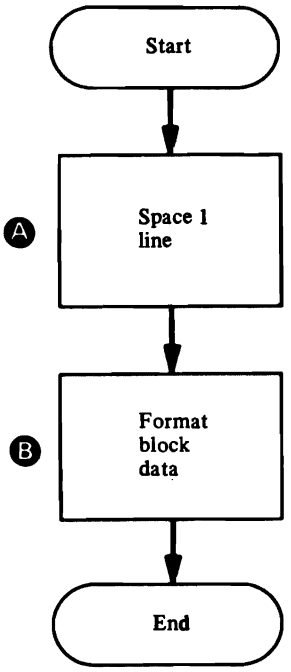


Figure 22. Formatting Example I

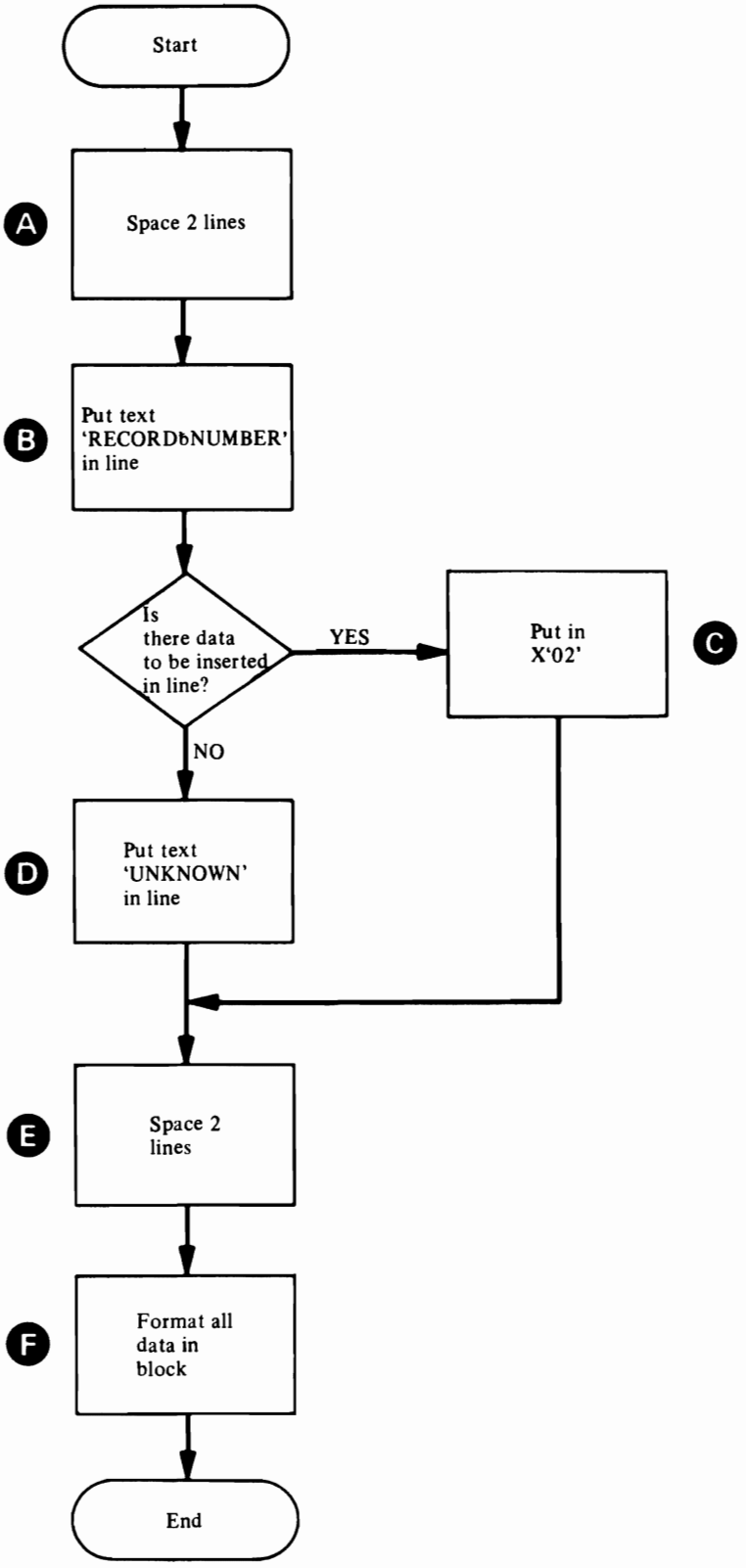


Figure 23. Formatting Example II

The output is:

```
(1 blank line)
RECORD NUMBER 002
(1 blank line)
xxxxxxx converted data for as many lines as necessary
```

The module has in its storage:

- the data for the record number in the header, in this example X'02'
- the block of data to convert and print
- a DARGLIST

Already existing in a static text module is:

- a FMTLIST
- text for the header, in this example the characters 'RECORD␣NUMBER'

The data is:

Offset	Name	Contents	Comments
0	any, RECNUM for example	one byte with the value X'02'	
1	any, DUMPIT for example	2000 bytes of binary data	The binary data will be converted to printable hexadecimal.

The DARGLIST is:

Offset	Name	Contents	Comments
0	DARGDBP	↑DUMPIT	Address of the block of data to convert.
4	DARGRETP	0	The lines are to be printed rather than just formatted and returned to the module without printing.
8	DARGSTID	C'AL0',X'03'	Static text identification to locate the FMTLIST—the FMTLIST IDCTSAL0 at entry 3.
12	DARGILP	2000	The length of DUMPIT.
14	DARGCNT	1	One insert data appears in DARGARY.
16	DARGRETL	0	The length of the converted data is used as the number of bytes to print.
18	DARGIND	0	Printing starts in the column indicated in FMTLIST.
19	none	0	Unused.
20	DARGARY	none	DARGARY is the name of the rest of DARGLIST.
20	DARGINS	4	This number is matched with a insert substructure in FMTLIST.
22	DARGINL	1	The number X'02' occupies one byte.
24	DARGDTM	↑RECNUM	Address of the number X'02' in the module.

At entry X'03' in static text module IDCTSAL0 is:

	Offset	Name	Contents	Comments
	0	TXT	71	Length of the FMTLIST and the data that follows the FMTLIST.
	2	FLG	0	This static text entry is for data not a message or header.
A	4	FMTFLGS	X'40'	Identifies these 6 bytes as a spacing substructure.
	5	none	0	Unused.
	6	FMTSPF	2	Space 2 lines.
	8	FMTSPT	C'R'	Space the lines relative to the last printed line.
	9	none	0	Unused.
B	10	FMTFLGS	X'04'	Identifies these 10 bytes as a static text substructure—the data is immediately after the FMTLIST.
	11	none	0	Unused.
	12	FMTSTL	13	Number of bytes in C'RECORDbNUMBER'.
	14	FMTSTO	54	Number of bytes the data C'RECORDbNUMBER' is from the first substructure in FMTLIST.
	16	FMTOCOL	1	The data C'RECORDbNUMBER' is to be printed in column 1.
	18	FMTOLEN	0	0 indicates the output length is the same as the input length for this data.
C	20	FMTFLG	X'20'	Identifies these 12 bytes as an insert substructure.
	21	none	0	Unused.
	22	FMTRFNO	4	This number is matched with the number in DARGINS in order to get the address of the data X'02'.
	24	none	0	Unused.
	26	FMTOCOL	15	The data X'02' is printed in column 15.
	28	FMTOLEN	3	The converted data is to take up 3 columns.
	30	FMTCNVF	X'1000'	The data X'02' is to be converted from byte to zoned decimal.
D	32	FMTFLGS	X'02'	Identifies these 8 bytes as a default text substructure.
	33	none	0	Unused.
	34	FMTILEN	7	Number of bytes in the data C'UNKNOWN'.
	36	FMTIOFF	67	Number of bytes the data C'UNKNOWN' is from the first substructure in FMTLIST.
	38	FMTOCOL	15	The data C'UNKNOWN' is printed in column 15.
E	40	FMTFLGS	X'40'	Identifies these 6 bytes as a spacing substructure.

Offset	Name	Contents	Comments
41	none	0	Unused.
42	FMTSPF	2	Space 2 lines.
44	FMTSPT	C'R'	The 2 lines are spaced relative to the last printed line.
45	none	0	Unused.
F 46	FMTFLGS	X'90'	Identifies these 12 bytes as a block data substructure and the last substructure in FMTLIST.
47	none	0	Unused.
48	FMTILEN	0	Zero means use the length of the block data in DARGILP.
50	FMTIOFF	0	Start at the first byte of the block data.
52	FMTOCOL	1	Start the block of data in output column 1.
54	FMTOLEN	0	Zero means print the block data until the input is exhausted no matter how many lines it takes.
56	FMTCNVF	X'8000'	Convert the block of data from binary to printable hexadecimal.
58	any	C'RECORD ŧNUMBER'	Data for the second substructure.
71	any	C'UNKNOWN'	Data for the default text substructure.

Discussion: The first spacing substructure causes 2 lines to be spaced.

The static text 'RECORDŧNUMBER' is put in the next line.

The insert number in the insert substructure is matched with the insert number in DARGLIST. The number X'02' from the module is converted to zoned decimal and placed in column 15.

The next spacing substructure causes 2 more lines to be spaced.

The block data substructure causes the data addressed by DARGDBP to be converted to printable hexadecimal until all the bytes in DARGILP have been converted and printed. If the module wants to print the same lines again but with a different record number and different block data, only DARGDBP, and DARGDTM need to be changed. If there had not been a reference number 4 in DARGLIST, the data 'UNKNOWN' will be printed instead of the record number '002'. This allows more freedom for the module to vary the output just by changing insert reference numbers in the DARGLIST.

Example III

A module wants to space 3 lines then print repeating fields on different lines so the output would appear as (see Figure 24):

```
(2 blank lines)
field A   field B   X'field C1'   field D1   field E1
                    X'field C2'   field D2   field E2
```

The module has in storage:

- all the data to be printed
- a DARGLIST

- a FMTLIST

The data is:

Offset	Name	Contents	Comments
0	A	four bytes of EBCDIC data	
4	B	four bytes of packed decimal data	
8	C1	two bytes of binary data	
10	D1	two bytes of binary data	
12	E1	one byte of EBCDIC data	
13	C2	two bytes of binary data	
15	D2	two bytes of binary data	
17	E2	one byte of EBCDIC data	

The DARGLIST is:

Offset	Name	Contents	Comments
0	DARGDBP	↑A	
4	DARGRETP	0	The lines are to be printed rather than just formatted and returned to the module.
8	DARGSTID	0	No static text module is used.
12	DARGILP	18	Number of bytes from field A through field E2.
14	DARGCNT	1	There is one repetition substructure in the FMTLIST.
16	DARGRETL	0	The length of the converted data is used as the number of bytes to print.
18	DARGIND	0	Printing starts in the column indicated in FMTLIST.
19	none	0	Unused.
20	DARGREP	7	Number that is matched with a repetition substructure in FMTLIST.
22	DARGPCT	2	The group of fields identified by repetition substructure 7 in FMTLIST is to be printed twice.

The FMTLIST is:

Offset	Name	Contents	Comments
(A) 0	FMTFLGS	X'40'	Identifies these 6 bytes as a spacing substructure.
1	none	0	Unused.
2	FMTSPF	3	Space 3 lines.
4	FMTSPT	C'R'	Space the lines relative to the last printed line.
5	none	0	Unused.

	Offset	Name	Contents	Comments
B	6	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure.
	7	none	0	Unused.
	8	FMTILEN	4	Number of bytes in field A.
	10	FMTIOFF	0	Field A begins zero bytes from the block of data whose address is in DARGDBP.
	12	FMTOCOL	1	Print field A starting in column 1.
	14	FMTOLEN	4	Number of bytes the converted field A occupies in the printed line.
	16	FMTCNVF	0	No conversion is done on field A.
C	18	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure.
	19	none	0	Unused.
	20	FMTILEN	4	Number of bytes of storage field B occupies.
	22	FMTIOFF	4	Field B starts 4 bytes from the block of data whose address is in DARGDBP.
	24	FMTOCOL	10	Print field B starting in column 10.
	26	FMTOLEN	10	Number of bytes the converted field B occupies in the printed line.
	28	FMTCNVF	X'0880'	Convert field B from packed decimal to unpacked decimal with zero suppression.
D	30	FMTFLGS	X'08'	Identifies these 8 bytes as a replication substructure.
	31	none	0	Unused.
	32	FMTRENO	7	Matched with a number in DARGLIST to find the number of iterations.
	34	FMTRBC	3	The data identified in the next 3 substructures is to be repeated.
	36	FMTRIO	5	The number of bytes from field C1 to field C2 in storage. This number is added to the address of the first field each time the field is repeated.
E	38	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure for fields C1 and C2.
	39	none	0	Unused.
	40	FMTILEN	2	Number of bytes fields C1 and C2 each occupy in storage.
	42	FMTIOFF	8	Number of bytes from field A to field C1.
	44	FMTOCOL	22	Print fields C1 and C2 starting in column 22.
	46	FMTOLEN	7	Number of bytes the converted fields C1 and C2 each occupy in the printed line.
	48	FMTCNVF	X'4000'	Convert fields C1 and C2 from binary to printable hexadecimal enclosed in X'data'.

Offset	Name	Contents	Comments
F 50	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure for fields D1 and D2.
51	none	0	Unused.
52	FMTILEN	2	Number of bytes fields D1 and D2 each occupy in storage.
54	FMTIOFF	10	Number of bytes from field A to field D1.
56	FMTOCOL	31	Print fields D1 and D2 starting in column 31.
58	FMTOLEN	6	Number of bytes the converted fields D1 and D2 each occupy in the printed line.
60	FMTCNVF	X'1000'	Convert fields D1 and D2 from binary to printable decimal.
62	FMTFLGS	X'90'	Identifies these 12 bytes as a block data substructure for fields E1 and E2 and the last substructure in the FMTLIST
63	none	0	Unused.
64	FMTILEN	1	Number of bytes fields E1 and E2 each occupy in storage.
66	FMTIOFF	12	Number of bytes from field A to field E1.
68	FMTOCOL	39	Print fields E1 and E2 each starting in column 39.
70	FMTOLEN	1	Number of bytes the converted fields E1 and E2 each occupy in the printed line.
72	FMTCNVF	X'0000'	No conversion is done on fields E1 and E2.

Discussion: The first spacing substructure causes 3 lines to be spaced.

The block data substructures for fields A and B describe the location of A and B within the block addressed in DARGDBP. Field A is not converted. Field B is converted from packed decimal to zoned decimal and leading zeros are replaced with blanks.

The replication substructure number is matched with an identification number in DARGREP. When a match is found, the DARGPCT immediately after DARGREP tells how many times to repeat the substructures. If the module wants to use the same FMTLIST and print another group of fields C, D, and E, only DARGPCT needs to be changed. The replication substructure tells how many substructures to repeat and an offset that is used to find the group of fields being repeated. On the first repetition the offset is not used, on the second it is added once; on the third repetition it is added twice.

The next substructure describe C1 and C2. On the first repetition the value in FMTIOFF is added to the value in DARGDBP to find field C1. To find field C2, FMTIOFF and FMTRIO in the repetition substructure are added to DARGDBP. Each time a group of substructures is repeated a new line is printed because the output columns for each substructure do not change. For example, in order to print both C1 and C2 in column 22, a new line must be printed. Both C1 and C2 are converted to printable hexadecimal preceded by X' and followed by a single quote.

Fields D1 and D2 are described by the next substructure. D1 and D2 are converted to printable decimal.

The substructure for fields E1 and E2 is also the end of FMTLIST. E1 and E2 are not converted.

After E1 is formatted, the three substructures following the repetition substructure are repeated. A new line is started because FMTOCOL keeps the output the columns the same each time a field is printed. Fields C2, D2, and E2 are put in the next line. The FMTLIST is finished after E2 is printed.

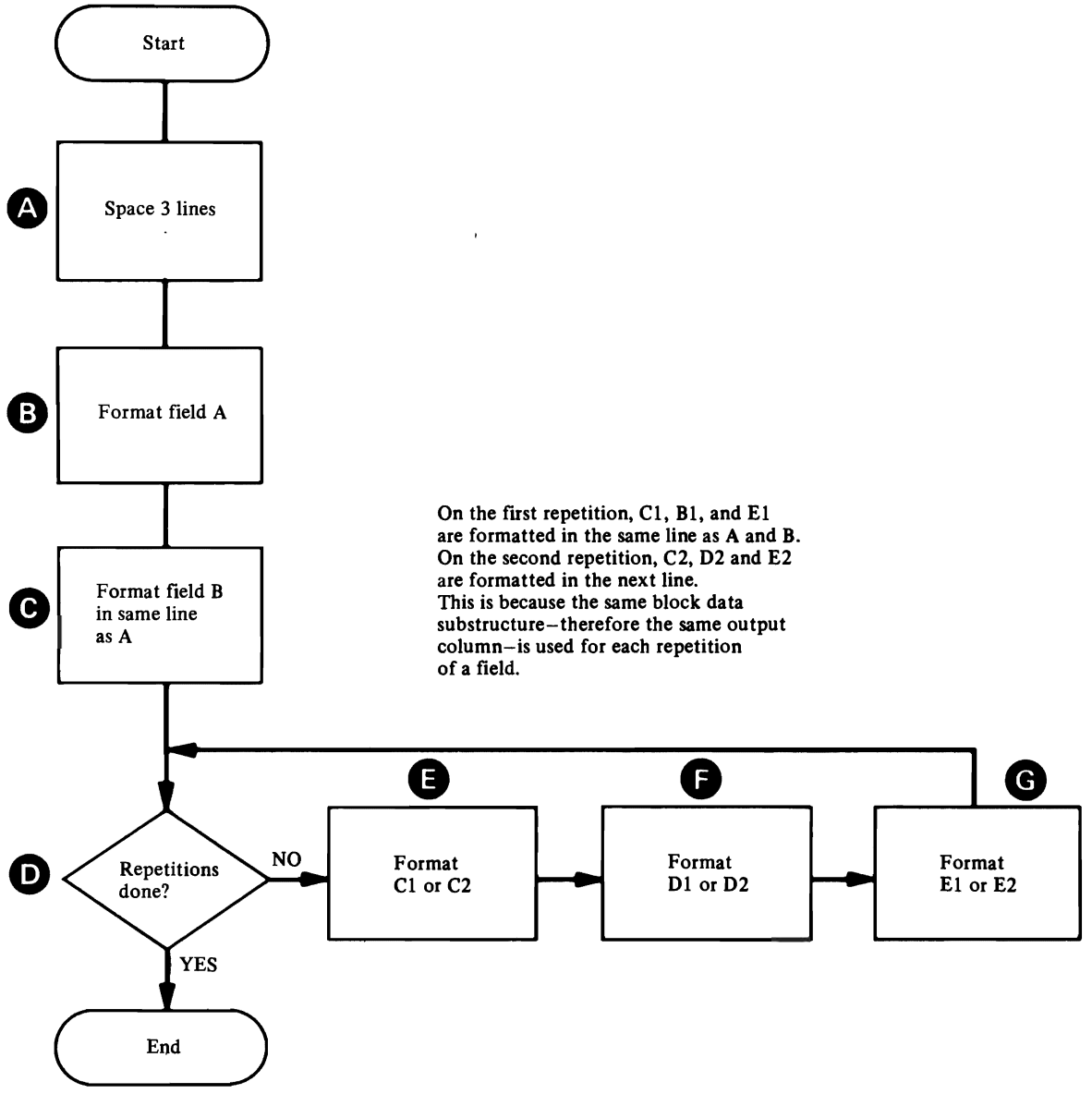


Figure 24. Formatting Example III

Obtaining a Dump for a Text Processor Problem

If you do not have an ABEND dump within the Text Processor routines or an ABORT snap dump within the Text Processor, you can use the Test option to obtain a dump. You may want to obtain a dump within the routine that invoked the Text Processor or within the Text Processor itself.

The Module or CSECT to Dump Points Cross Reference list contains all the dump points within the processor; you can specify these dump points on the FULL option of the TEST keyword to obtain a full region dump.

The Text Processor has dump points before and after it converts data to printable form. You should use these dump points if there is an error in converting the data.

How to Find Text Processor Argument Lists

If you suspect a problem within the Text Processor, the two structures you should locate in a dump are the Print Control Table (PCT) and the Dynamic Data Argument List (DARGLIST). The PCT and the DARGLIST are described in the chapter "Data Areas." The eighth word of the GDT contains the address of the PCT. The address of the DARGLIST is the third parameter passed to IDCTP01 for a UPRINT request.

Two other structures that you may find helpful to locate in a dump are the queue of format structures and the print buffer.

Note: in the listings the print buffer is called the stack buffer.

Figure 25 shows the queue of format structures maintained by the Text Processor. There is an entry in the queue for each text structure entry used for the current function. Each entry in the queue contains the four-byte static text identifier specified in the DARGLIST. The first three bytes contain the last three characters of the text-structure module name; the fourth byte contains the entry number of the format structure within the module.

Figure 26 shows the print buffer maintained by the Text Processor. It contains the records, other than messages, that have not been printed. The records to be printed are kept in the print buffer until the buffer becomes full or a message must be printed. The primary and secondary PCTs contain the address of the first record in the buffer and the address of the next empty space in the buffer. If both addresses are equal, the buffer is empty.

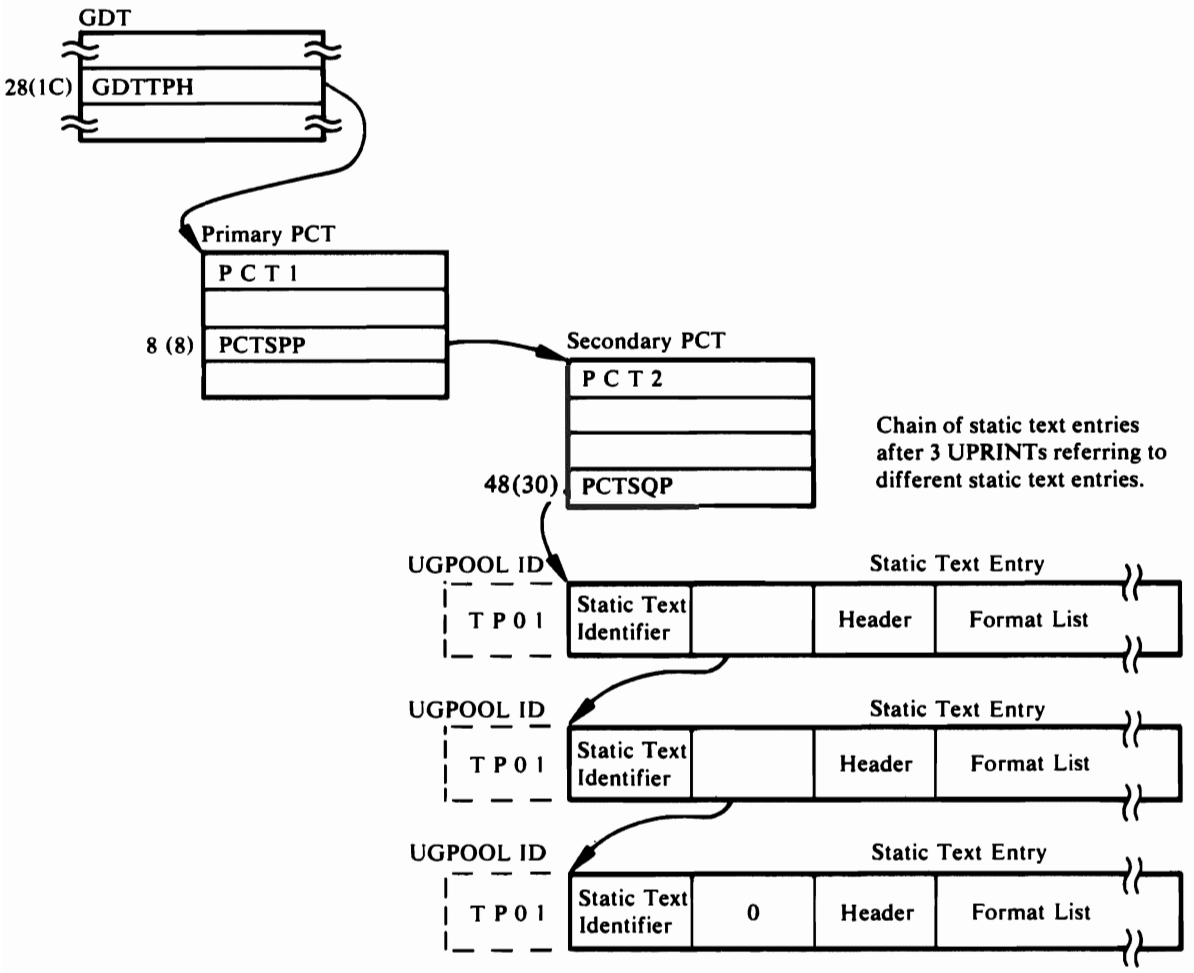


Figure 25. Text Processor Format Structure Queue

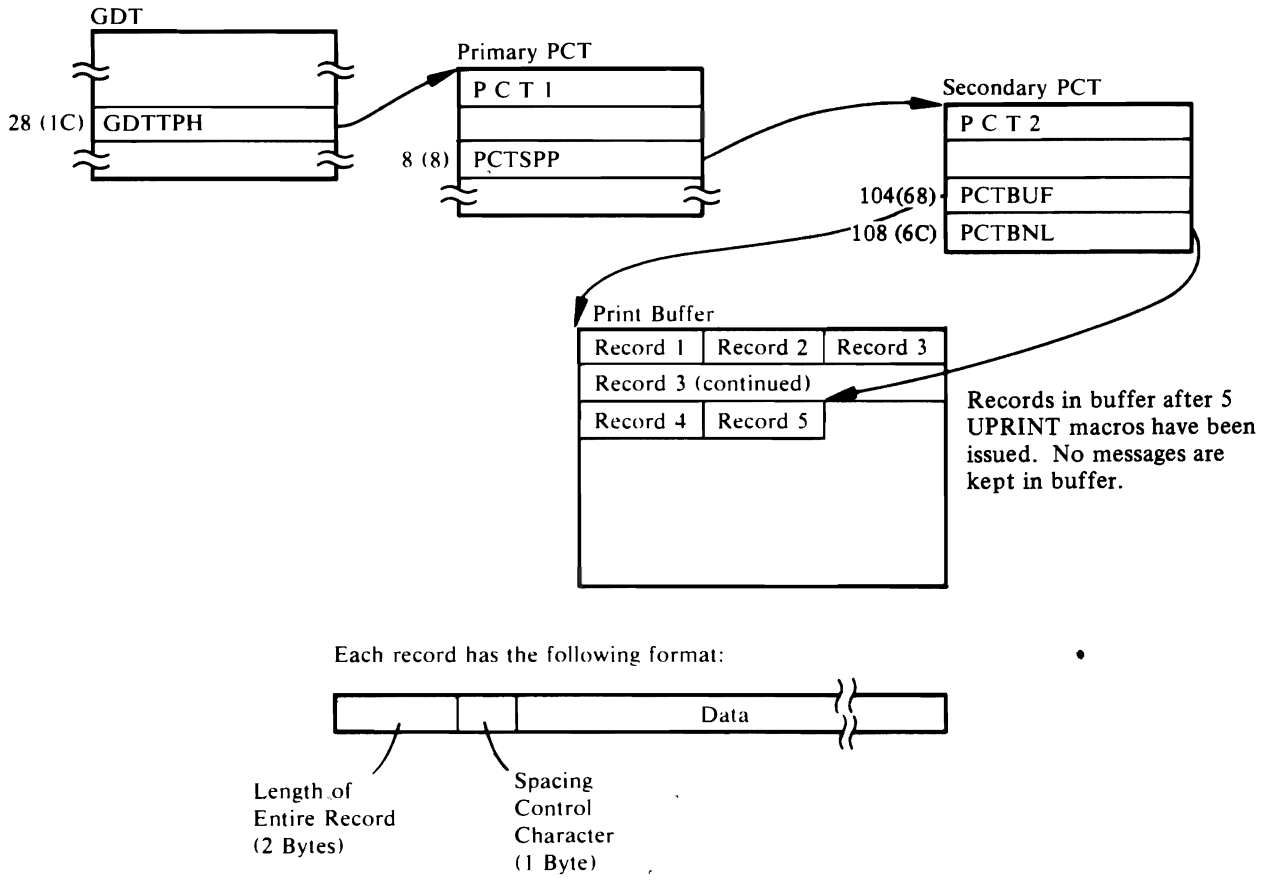


Figure 26. Text Processor Print Buffer

Debugging an I/O Problem

There may be an I/O problem within system I/O routines or within Access Method Services if an ABORT condition occurs in the I/O Adapter or if an ABEND occurs within the system I/O routines. To determine whether the problem exists in the routines that invoke the I/O Adapter, in the I/O Adapter itself, or in the system I/O routines, you must examine the argument lists passed between the I/O Adapter and the invoking routines, and the I/O Adapter and the system I/O routines.

This section explains how to obtain a dump that contains the I/O argument lists and how to find the argument lists in a dump.

Obtaining a Dump for an I/O Problem

If you do not have an ABEND dump within system I/O routines or an ABORT snap dump within the I/O Adapter, you can use the Test option to obtain a dump. You may want to obtain a dump within the routine that invoked the I/O Adapter or within the I/O Adapter itself.

The "Module or CSECT to Dump Points Cross Reference" list contains all the dump points within the processor; you can specify these dump points on the FULL option of the TEST keyword to obtain a full region dump.

The I/O Adapter has dump points before and after it issues the OPEN SVC (dump points IO1O and IO20) and before it issues the CLOSE SVC (dump

point IO1C). You should use these dump points if there is an error opening or closing data sets. For **VS2.03.808**, the I/O Adapter has a dump point (IOVR) after issuing a VSAM I/O request which returns a nonzero return code. You should use this dump point if you wish to obtain a dump in a VSAM I/O error situation.

How to Find I/O Argument Lists

The Input/Output Communications Structure (IOCSTR), which is constructed for each data set that has been opened, contains pointers to most of the control blocks used by the system I/O routines. The IOCSTR is also the argument list that is passed between the I/O Adapter and the routines that invoke the I/O Adapter, except for the initial open request. Thus, once you find the IOCSTR, you can find most of the other arguments passed between the I/O Adapter and other routines. The chapter "Data Areas" explains the format of the IOCSTR.

Figure 27 shows the chain of IOCSTRs constructed for all opened data sets; however, the data sets may not have been opened successfully. The I/O Adapter historical area contains a pointer to the start of the chain.

You can find the address of the IOCSTR for a particular I/O request by finding the parameter list passed to IDCIO01 by the invoking routine. Register 1 of the registers saved by IDCIO01 contains the address of a parameter list. The second word of the parameter list contains the address of the IOCSTR. The third, fourth, and fifth words may contain addresses of additional IOCSTRs.

Open Argument Lists

Figure 28 shows how the I/O control blocks are connected before the OPEN SVC is issued. The IOCSTR addresses can be found from the IOCSTR chain as shown in Figure 27. The IOCSBLT table, which contains pointers to the IOCSTRs for the data sets being opened, can be found at location IOCSBLT in IDCIO01's automatic storage area. The OPENLIST table, which contains pointers to the DCBs and ACBs for the data sets being opened, can be found at location OPENLIST in IDCIO01's automatic storage area. In a system dump within the OPEN SVC, register 1 of the registers saved at entry to the SVC contains the address of the OPENLIST table.

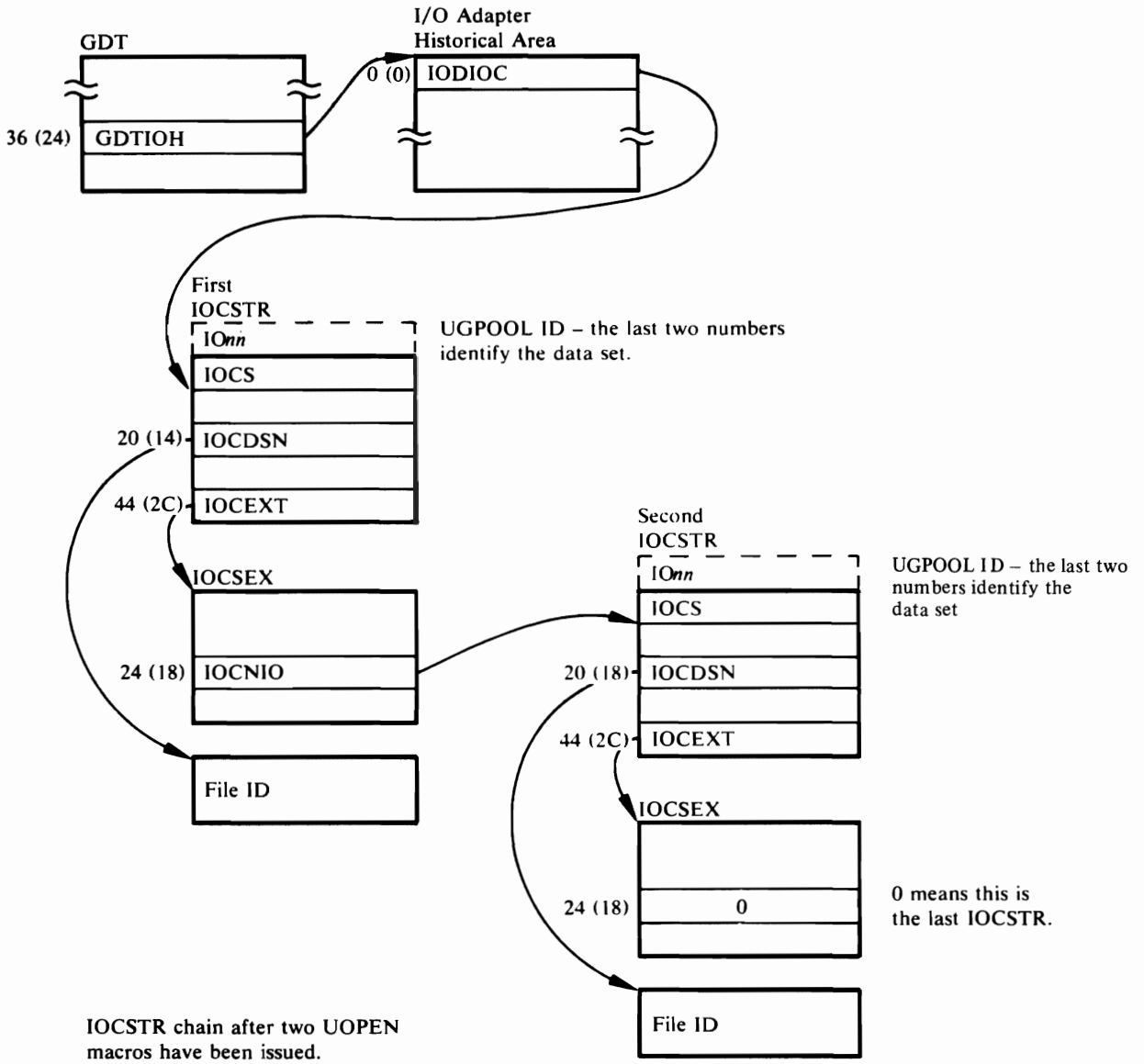
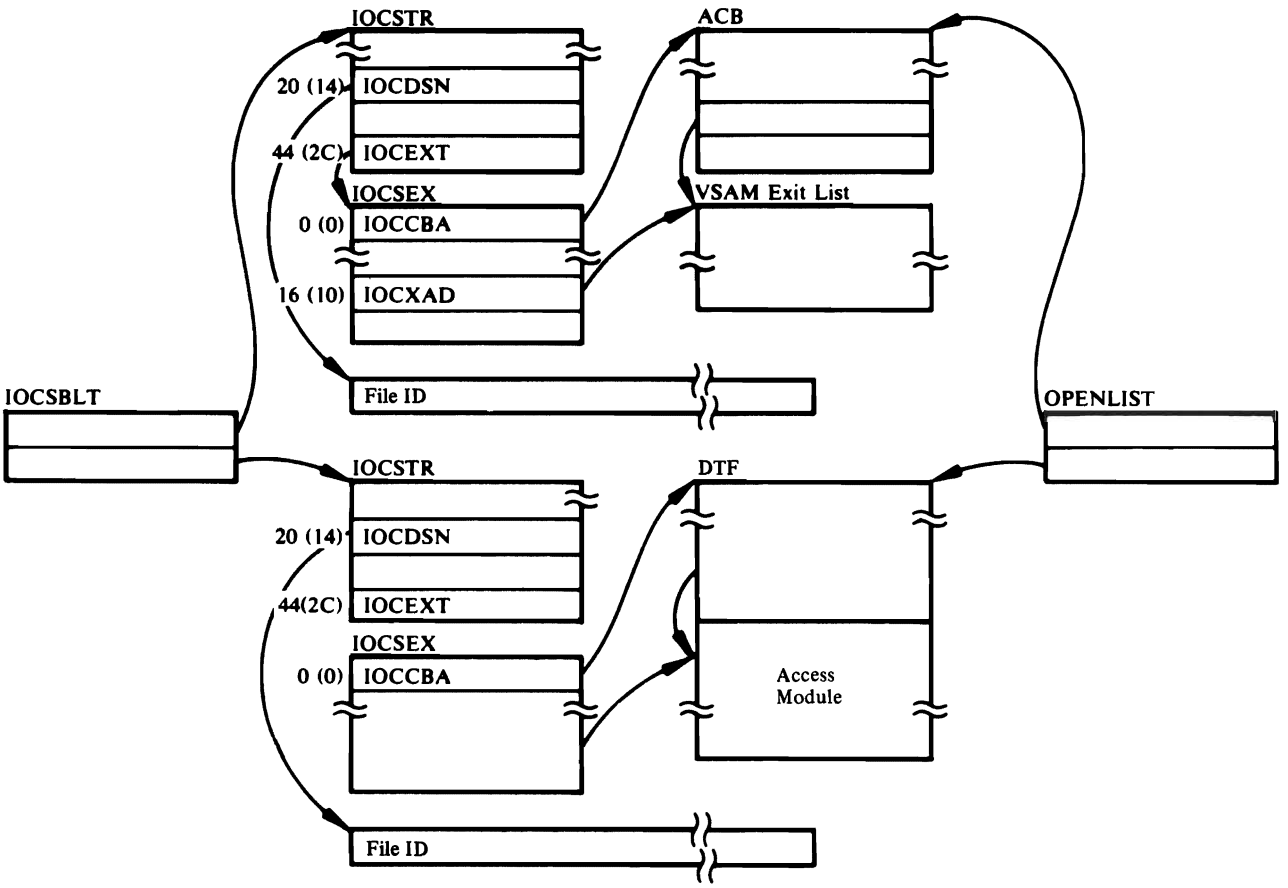


Figure 27. IOCSTR Chain



Two data sets are to be opened, one VSAM and one non-VSAM data set.

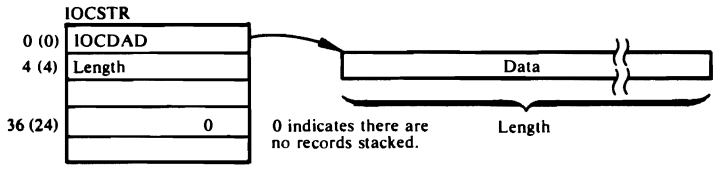
Figure 28. I/O Control Blocks Before OPEN

UGET and UPUT Argument Lists

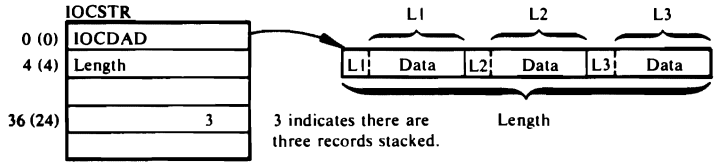
This section contains some examples of input and output from the UGET and UPUT macros. These examples may be helpful in determining whether the IOCSTR and records for a UPUT request have been passed correctly to the I/O Adapter, and whether the IOCSTR and records for a UGET request have been returned correctly by the I/O Adapter.

Figure 29 shows the IOCSTRs and records passed to the I/O Adapter via a UPUT macro. The I/O Adapter adds Record Descriptor Words (RDWs) for QSAM output before passing the records to QSAM.

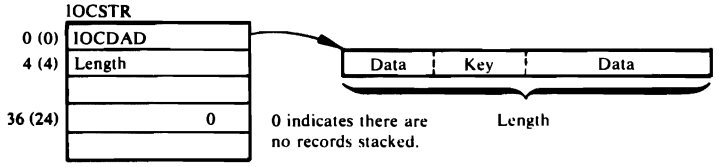
Example 1. VSAM or QSAM Data Set – Single Record with No Keys



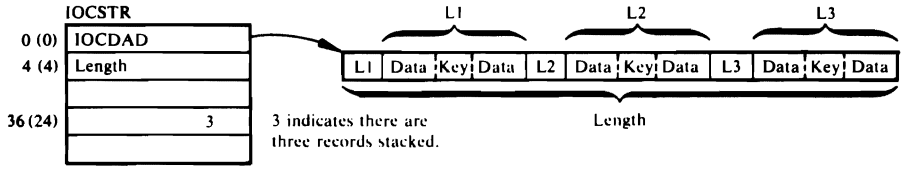
Example 2. VSAM or QSAM Data Set – Multiple Records with No Keys



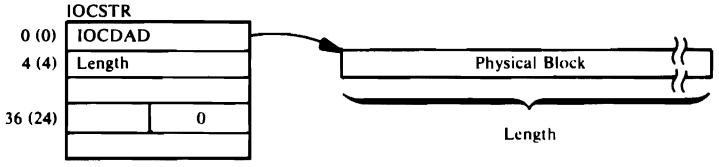
Example 3. VSAM Data Set – Single Record with Keys



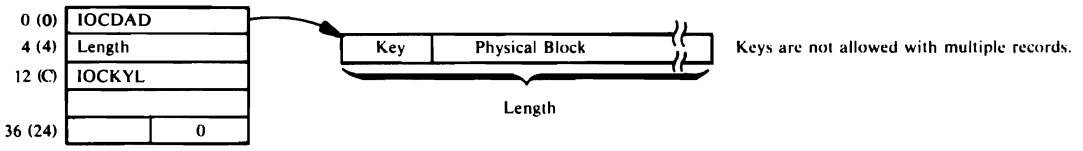
Example 4. VSAM Data Set – Multiple Records with Keys



Example 5. BSAM or BPAM Data Set — Single Block with No Keys



Example 6. BSAM or BPAM Data Set — Single Block with Keys



Example 7. BSAM or BPAM Data Set — Multiple Block with No Keys

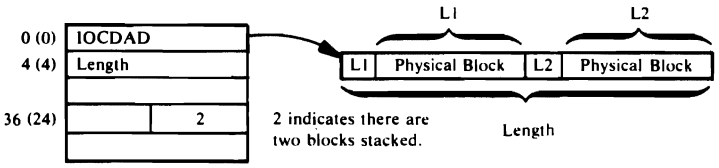
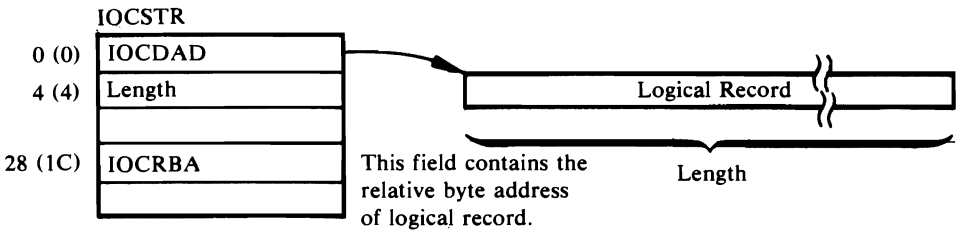


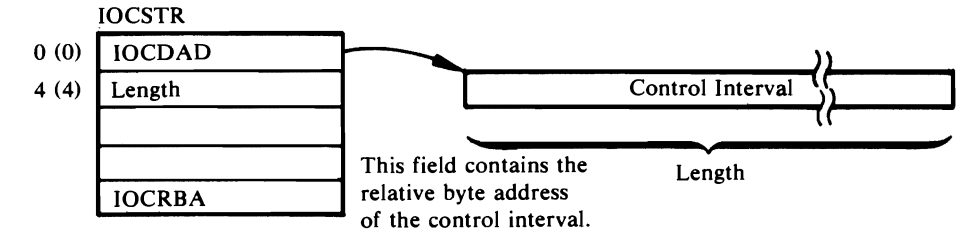
Figure 29. Input to UPUT Macro

Figure 30 shows the IOCSTRs and data returned by the I/O Adapter after a UGET macro is processed.

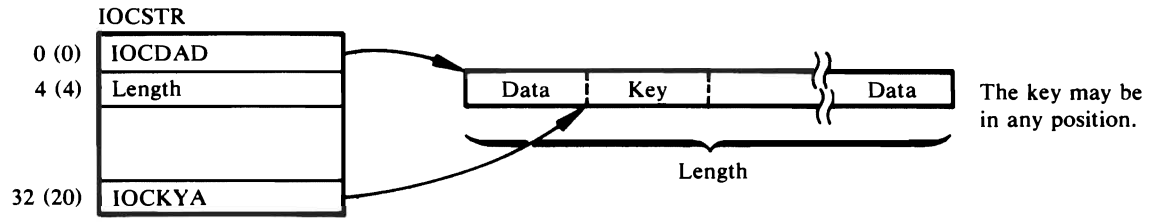
Example 1. VSAM Data Set with Address Processing



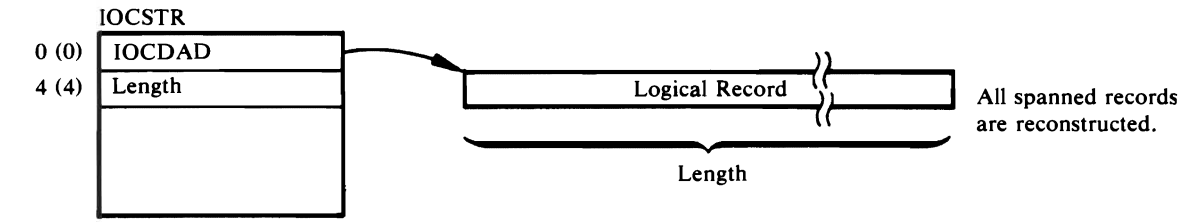
Example 2. VSAM Data Set with Control Interval Processing



Example 3. VSAM Data Set with Keyed Processing



Example 4. QSAM Data Set



Example 5. QISAM Data Set

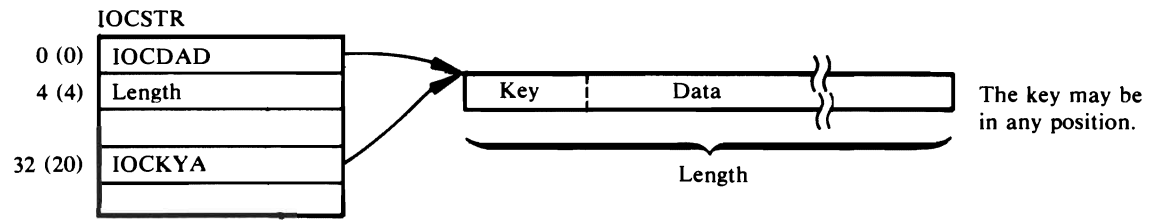


Figure 30. Output from UGET Macro

VSAM Control Block Manipulation Argument Lists (Deleted for VS2.03.808)

If the return code from a VSAM control block manipulation macro indicates a severe error, the I/O Adapter issues a UABORT macro and you will get a snap dump. (VSAM control block manipulation macros are: GENCB, MODCB, SHOWCB, and TESTCB.) The argument lists constructed by the I/O Adapter for these VSAM macros are stored at location AMAREA in the automatic storage area of the I/O Adapter module that issued the macro.

Figure 31 shows how the argument lists are connected. Register 1 at the time the macro is issued points to the parameter list at location AMAREA. See *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications* for the contents of the argument lists.

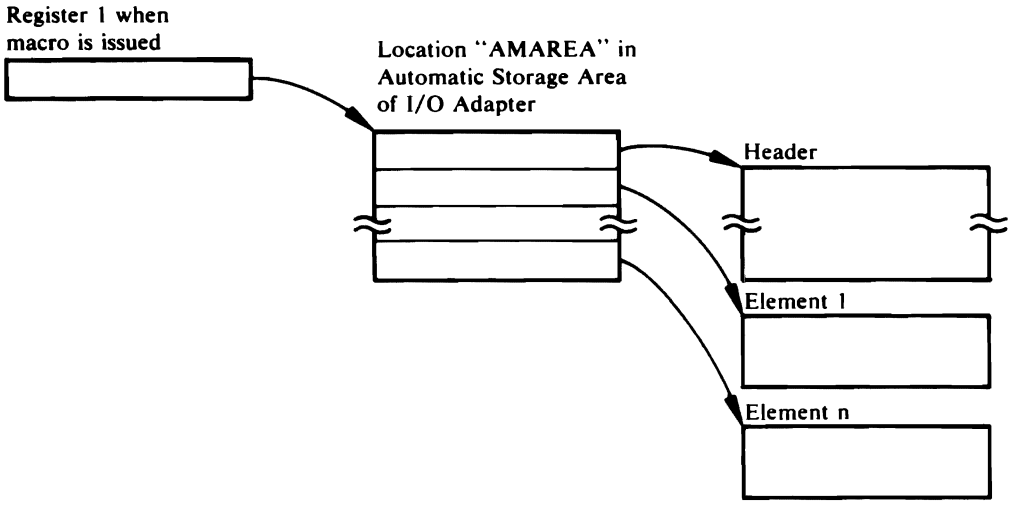


Figure 31. VSAM Control Block Manipulation Macro Argument Lists (Deleted for VS2.03.808)

Processor Messages

The following lists all the messages printed by the processor. For each message, the following information is listed: the last three characters of the text structure module that contains the message followed by the number of the message within the module; the module that causes the message to be printed; the procedure within that module that detects the situation that causes the message to be printed; and the situation that causes the message to be printed. For message IDC4999I there is no text structure module because the message is written when a UABORT macro is issued. The table lists the UABORT CODE instead of the text structure module for message IDC4999I.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC394I	SA6-14	IDCSA06	DEMNTVOL	IDCSA06 failed to demount the requested volume. (This is an informational message for the operator.)
		IDCSA10	RECOVERY	An attempt to demount a volume in STAE/ESTAE Exit routine failed. (This is an informational message for the operator.)
IDC0001I	UV0-1 UV0-9	IDCAL01	IDCAL01	Function was completed without a severe error.
		IDCCC01	IDCCC01	Function was completed without a severe error. All or part of the OS/VS catalog entries were converted to VSAM catalog entries.
		IDCBI01	TERMPROC	Function was completed without an error or without a severe error in processing the base cluster.
		IDCDE01	IDCDE01	Function was completed without a severe error.
		IDCDL01	IDCDL01	Function was completed without a severe error.
		IDCLC01	IDCLC01	Function was completed without a severe error. All or part of the desired catalog listing was generated.
		IDCLR01	CLEANUP	Function was completed without a severe error.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
		IDCMP01	IDCMP01	Function was completed without a severe error.
		IDCPM01	IDCPM01	Function was completed without a severe error.
		IDCPR01	IDCPR01	Function was completed without error, or (1) an end-of-file was reached in the input data set before the ending delimiter specified by the user, or (2) a recoverable I/O error occurred while retrieving or printing a record, or (3) an error occurred closing data sets.
		IDCRC01	EXITTHE	Function was completed without a severe error.
		IDCRM01	IDCRM01	Function was completed without a severe error.
		IDCRP01	IDCRP01	Function was completed without error, or (1) an end-of-file was reached in the input data set before the ending delimiter specified by the user, or (2) a recoverable I/O error occurred while copying a record, or (3) an error occurred closing data sets.
		IDCRS01 **(VS2.03.808)**	WRAPUP	Function was completed without a severe error.
		IDCVY01	IDCVY01	Function was completed without a severe error.
		IDCXP01	IDCXP01	Function was completed without a severe error.
IDC0002I	UV0-2	IDCEX03	IDCEX03	Access Method Services completed processing.
IDC0005I	UV0-5	IDCPR01	IDCPR01	Printing of records is completed.
		IDCRP01	IDCRP01	Copying of records is completed.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC0014I	UV0-15	IDCCC01	IDCCC01	In a TSO environment, a nonzero return code condition was encountered.
		IDCVY01	IDCVY01	In a TSO environment, a nonzero return code condition was encountered.
		IDCLC01	IDCLC01	In a TSO environment, a nonzero return code condition was encountered.
		IDCBI01	TERMPROC	In a TSO environment, a nonzero return code condition was encountered.
		IDCDL01	IDCDL01	In a TSO environment, a nonzero return code condition was encountered.
		IDCXP01	IDCXP01	In a TSO environment, a nonzero return code condition was encountered.
		IDCMP01	IDCMP01	In a TSO environment, a nonzero return code condition was encountered.
		IDCRS05 **(VS2.03.808)**	CKERR	In a TSO environment, a nonzero return code condition was encountered.
		IDCRM01	IDCRM01	In a TSO environment, a nonzero return code condition was encountered.
		IDCAL01	IDCAL01	In a TSO environment, a nonzero return code condition was encountered.
		IDCDE01	IDCDE01	In a TSO environment, a nonzero return code condition was encountered.
IDC0204I	RI0-5	IDCRI03	IDCRI03	The preceding command was scanned for syntax-checking purposes only.
IDC0206I	RI0-7	IDCRI01	SCANSEP	An extra comma was found between parameters.
IDC0222I	RI0-23	IDCRI01	NXTFIELD	A semicolon was found within a quoted constant.
IDC0233I	RI0-34	IDCRI01	SCANCMD	Too many closing parentheses were found at the end of a command or subparameter list.
IDC0234I	RI0-35	IDCRI01	INREPEAT	Too few parentheses were found at the end of a command.
			SCANCMD	Too few parentheses were found at the end of a command.
IDC0361I	SA7-9	IDCSA07	IDCSALC	The VSAM LOCATE request was unsuccessful. Refer to the message issued immediately prior to this one for the reason.
IDC0362I	SA7-10	IDCSA07	VSAMUCT	An error occurred during a VSAM DELETE request. Refer to a message issued prior to this one for the reason why the DELETE

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				failed. As a result of the error, the data set named in this message was not scratched.
IDC0363I	SA7-11	IDCSA07	VSAMUCT	An error occurred during a VSAM DELETE request. Refer to a message issued prior to this one for the reason why the DELETE failed. As a result of the error, the data set named in this message was not uncataloged.
IDC0396I	SA7-1	IDCSA07	GETENT	The data set named was not recataloged.
			UPDATENT	A previous message has the reason.
			TESTENT	A previous message has the reason.
IDC0397I	SA7-2	IDCSA07	UPDATENT	The data set identified was located but could not be recataloged. The return code from recataloging was 8. This situation should occur only when the volume was originally located in a VSAM catalog—the LOCATE macro supports a VSAM catalog, but the CATLG macro to recatalog does not. Since all data sets in the VSAM catalog that owns the volume have been recataloged, the data set must have been located in a VSAM catalog that did not own the volume.
IDC0398I	SA7-5	IDCSA07	TESTENT	The data set identified resides on more than 20 volumes and apparently has not been recataloged.
			SCANCMD	Too few parentheses were found at the end of a command.
IDC0508I	DE0-9	IDCDE01	IDCDE01	Define of the data set is completed.
		IDCMP01	CLUSPROC	Define of the data set being imported is completed.
		IDCRM01	CLUSPROC	Define of the data set is completed.
IDC0509I	DE0-10	IDCDE01	IDCDE01	Define of the data set is completed.
		IDCMP01	CLUSPROC	Define of the data set being imported is completed.
		IDCRM01	CLUSPROC	Define of the data set is completed.
IDC0510I	DE0-11	IDCDE01	IDCDE01	Define of the VSAM catalog is completed.
IDC0511I	DE0-12	IDCDE01	IDCDE01	Define of the data space is completed.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC0512I	DE0-13	IDCDE01	IDCDE01	Define of the data set is completed.
IDC0520I	DE0-21	IDCDE01	IDCDE01	The message identifies the recovery volume serial number.
		IDCMP01	CLUSPROC	The message identifies the recovery volume serial number.
		IDCRM01	CLUSPROC	The message identifies the recovery volume serial number.
IDC0526I	AL0-1	IDCAL01	IDCAL01	Alter of the volume is completed.
IDC0531I	AL0-7	IDCAL01	IDCAL01	All altered entry names are listed.
IDC0532I	AL0-8	IDCAL01	IDCAL01	All entry names not altered are listed.
IDC0534I	AL0-10	IDCAL01	MEMRENAM	A member name not renamed is listed.
IDC0535I	AL0-11	IDCAL01	MEMRENAM	The listed member name has been renamed.
IDC0548I	DL0-10	IDCDL01	MEMDELET	The listed member name has not been deleted.
IDC0549I	DL0-11	IDCDL01	MEMDELET	The listed member name has been deleted.
IDC0550I	DL0-1	IDCDL01	CATCALL	The catalog returned the name and type of a successfully deleted entry in the catalog work area.
		IDCMP01	DELTPROC	The object with the same name as the object being imported was deleted successfully from the catalog.
			DELTPROC	The object being imported was deleted successfully from the catalog after an error occurred copying data into the object.
		IDCRM01	DELTPROC	The object with the same name as the object being imported was deleted successfully from the catalog.
			DELTPROC	The object being imported was deleted successfully from the catlog after an error occurred copying data into the object.
		IDCXP01	DELTPROC	The object being exported was deleted successfully from the catalog.
IDC0551I	DL0-8	IDCDL01	DELTPROC	A catalog object was not deleted because of a catalog locate error, a command parameter error, or a catalog delete error.
		IDCXP01	DELTPROC	The object being exported could not be deleted from the catalog. The catalog return code indicates the reason.
IDC0555I	DL0-5	IDCDL01	CATCALL	The volume entry was not deleted although empty space on the

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				volume was deleted successfully. The catalog return code was 160.
IDC0571I	PR0-19	IDCRP01	IDCRP01	Reloading of a catalog was initiated.
IDC0594I	XP0-5	IDCXP01	CLUSPROC	The portable data set was created successfully.
IDC0603I	MP0-11	IDCMP01	CLUSPROC	The user catalog was connected successfully.
IDC0604I	MP0-12	IDCMP01	CLUSPROC	The first record of the portable data set contained the timestamp written at the time of export.
		IDCRM01	IDCRM01	The first record of each group of associated items on the portable data set contained the time and date of export.
IDC0611I	MP0-2	IDCMP01	CLUSPROC	The name of the object being imported already exists in the catalog. If NEWNAME is specified, an entry with that name already exists.
IDC0622I	MP0-22	IDCRM01	UCATPROC	A user catalog has been disconnected successfully.
IDC0626I	MP0-26	IDCRM01	CLUSPROC UCATPROC NVSMPROC GDGPROC	IMPORTRA succeeded for the object named in the message.
IDC0634I	CC0-7	IDCCC01	CONVERT	CONVERT prints the number of OS/VS catalog entries that were converted to VSAM catalog entries.
IDC0635I	CC0-9	IDCCC01	CATALOG	The OS/VS catalog entry was not converted to a VSAM catalog entry.
IDC0636I	CC0-13	IDCCC01	CONVERT	CONVERT prints the number of non-VSAM entries in the VSAM catalog whose volume information was updated.
IDC0637I	CC0-11	IDCCC01	CONVERT	CONVERT prints the name of a VSAM catalog entry that was not updated to match an OS/VS catalog entry.
IDC0652I	BI0-13	IDCBI01	FINPROC	The alternate was built with no errors.
IDC0665I	LR1-16	IDCLR01	CLENCRA	Informational message stating the number of entries that did not compare.
IDC0669I	RC0-14	IDCRC01	IDCRC01	Informational message stating the CRA from which the entries are processed.
IDC0670I	RC0-15	IDCRC01	EXPORTDR	Informational message stating that data set is on portability data set.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC0672I	RC0-17	IDCRC01	CKCATNM	Informational message stating the catalog name for which CRA's are being processed.
IDC0674I	RC0-20	IDCRC01	EXPORTDR	Secondary message containing the object name for which the export driver was called.
			SYNCH	Object named was invalid in the CRA in comparison with the data set.
			DUPNAMCK	Object name appeared twice in the CRA.
			CKNAMES	Object named was not of a type DOS supports.
IDC0676I	RC0-5	IDCRC01	TERM	Informational message stating that the portability data set was created successfully.
IDC0861I	CK0-6	IDCCK01	IDCCK01	No type 1 DSDR records for a tape data set were found.
IDC0862I	CK0-7	IDCCK01	BUILDTAB	The checkid identified was selected by the user more than once.
IDC0863I	CK0-8	IDCCK01	CHRPROC	A duplicate entry was found for a selected checkid.
IDC0874I	LR1-5	IDCLR01	INTSORT	Space could not be obtained for the sort table. The objects are printed first in, first out.
IDC0877I	LR1-8	IDCLR01	CLENCRA	Informational message stating the number of objects that did not compare.
IDC0888I	RC0-23	IDCRC01	EXPORTDR	Informational message stating that the exported entry contained no data.
IDC0922I	EX0-5	IDCDB02	ITEMDUMP	An invalid dump item was specified in the dump argument list.
IDC0923I	EX0-6	IDCDB02	ARRAYHDR	Invalid array header parameters were specified in the dump argument list.
IDCO924I	EX0-7	IDCDB01	IDCDB01	The dump routine was invoked through a UDUMP macro.
IDC0925I	EX0-8	IDCDB01	IDCDB01	A dump was requested through a UDUMP macro.
IDC0970I	VS0-22	IDCVS01	VTOCPUT	The number of tracks and the cylinder and head address could not be restored in the VTOC.
IDC1069I	CM0-57	IDCSA09	CHKCODE	High-order bit in reason code returned by SVC 126 was on, indicating an error updating inventory.
IDC1252I	RI1-16	IDCRI04	RISETUP	MAXID indicates no parameters are defined for this command, but ECTNOPD indicates that parameters were coded.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1502I	DE0-5	IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	Security information was suppressed when a model object was retrieved from the catalog.
IDC1543I	AL0-18	IDCAL01	CHECKPRC	New KEY/RECORDSIZE values equal to old default values.
IDC1544I	AL0-19	IDCAL01	CHECKPRC	New KEY/RECORDSIZE values equal to old non-default values.
IDC1561I	LC1-2	IDCLC02	ANSVPROC	The UGPOOL request for a larger catalog work area failed. More space was required to process the associations.
			LOCPROC	The UGPOOL request for a larger catalog work area failed. A catalog entry required more space.
IDC1562I	LC1-3	IDCLC01	ENTPROC	Only space entries were requested; however, an entry in the entry list is greater than six characters.
IDC1564I	LC1-5	IDCLC01	RTEPROC	An entry retrieved from the catalog is not a type that can be listed.
IDC1565I	LC1-6	IDCLC01	ENTPROC	An entry retrieved from the catalog and specified in the user's entry list is not one of the types requested by the user.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1566I	LC1-8	IDCLC01	ENTPROC	Either (1) the correct password was not supplied for a cluster or AIX entry and so the data and index and path association information could not be processed, or (2) the correct password was not supplied for an entry and the user requested more information than merely entry names, or (3) another type of catalog locate error occurred.
			GNXTPROC	Either the correct password was not supplied for an entry and the user requested more information than merely entry names, or another type of catalog locate error occurred.
			RTEPROC	Either (1) the correct password was not supplied for a cluster or AIX entry, and, even though the user requested only entry names, the names of the data and index or path association were not returned by the catalog, or (2) the correct password was not supplied for a data or index or path entry associated with a cluster or AIX entry, and field information other than entry names was not returned by the catalog, or (3) a non-supported entry type was returned from the catalog.
IDC1567I	LC1-9	IDCLC01	RTEPROC	Retrieval of a data or index or path entry associated with a cluster or AIX entry was attempted, using the control interval number of the associated entry contained in the cluster or AIX entry. However, the entry could not be found in the catalog.
		IDCLC02	CDIPROC	Retrieval of a data or index or path entry associated with a cluster of AIX entry was attempted, using the control interval number of the associated entry contained in the cluster or AIX entry. However, the entry could not be found in the catalog.
			VPROC	Retrieval of the data set names associated with a data space was attempted using the control interval number of the associated entry contained in the data space. However, the entry could not be found in the catalog.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1569I	LC1-12	IDCLC01	INITPROC	The EXPIRATION LISTCAT option was specified with entry type(s) that contain either no expiration date field or an expiration date field that is never initialized.
IDC1574I	PR0-22	IDCRP01	CATCOMP	More than 100 true name entries failed a comparison test during catalog reload. Processing continues but comparison does not.
IDC1575I	PR0-23	IDCRP01	CATCOMP	A true name record existed on a backup or target catalog without a corresponding record on the backup or target catalog.
IDC1595I	XP0-6	IDCXP01	CLUSPROC	Passwords were suppressed when the object to be exported was retrieved from the catalog.
IDC1631I	CC0-4	IDCCC01	CATALOG	VSAM catalog management gave a return code of 8 after trying to define the OS/VS catalog entry. The duplicate data set name started with 'SYS1.'.
IDC1632I	CC0-2	IDCCC01	CVPEPROC	The CVOLEQUATES parameter was missing or the volume serial number in the Control Volume Pointer Entry did not match a volume serial number in the CVOLEQUATES parameter.
IDC1638I	CC0-14	IDCCC01	GIPEPROC	A Generation Index Pointer Entry has been found and the "alias processing" flag is on.
IDC1644I	BI0-5 BI0-17	IDCBI01	SORTPROC	The base cluster record identified in the message was too short to contain the entire alternate key.
IDC1645I	BI0-6 BI0-18	IDCBI01	BLDPROC	Multiple occurrences of the same alternate key have been encountered in building an alternate index defined with the UNIQUEKEY attribute.
IDC1646I	BI0-7	IDCBI01	BLDPROC	The alternate index record identified in the message was too short to contain all the base cluster pointers.
IDC1653I	BI0-14	IDCBI01	FINPROC	The alternate index was built but nonterminating errors were encountered.
IDC1661I	RC0-6	IDCRC01	EXPORTDR	Informational message stating that the data set exported was out-of-synch.
IDC1662I	RC0-7	IDCRC01	EXPORTDR	Informational message stating that the data set was not exported and was out-of-synch.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1663I	RC0-8	IDCRC02	CLUSPROC	Catalog field could not be located for a path to a VSAM cluster or an alias association to a nonVSAM data set or a nonVSAM association to a GDG.
IDC1664I	RC0-9	IDCRC02	NVSMPROC	Invalid or no association to a GDG for a nonVSAM data set.
IDC1667I	RC0-12	IDCRC01	OBJVOLCK	Volumes are out of synch because data set is not on both volumes.
IDC1678I	RC0-2	IDCRC01	EXPORTDR	An error occurred while processing an association for an object being exported.
IDC1679I	RC0-4	IDCRC01		The timestamps or CI of a multivolume data set were not equal.
IDC18601	CK0-5	IDCCK01	IDCCK01	The checkid selected by the user wasn't found in the checkpoint data set.
IDC1864I	CK0-9	IDCCK01	IDCCK01	No CHR records were found in the checkpoint data set.
IDC1865I	CK0-10	IDCCK01	DSDRVOLS	A type 2 DSDR record was expected but not found.
IDC1866I	CK0-11	IDCCK01	GETNEXT	End-of-data occurred in the checkpoint data set while DSDR records were being processed.
IDC1867I	CK0-12	IDCCK01	DSDRPROC	Volume sequence number for a tape data set exceeded the volume of volumes.
IDC1870I	LR1-1	IDCLR01	GETPRT	An I/O error occurred while reading the CRA.
		IDCLR02	IDCLR02	An I/O error occurred while reading the CRA.
IDC1871I	LR1-2	IDCLR01	GETPRT	An I/O error occurred while reading the catalog.
		IDCLR02	IDCLR02	An I/O error occurred while reading the catalog.
IDC1875I	LR1-15	IDCLR01	TCICTCR	The CI from the catalog record could not be found in the CTT table therefore it could not be translated.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1878I	LR1-9	IDCLR01	CATOPEN	IDCRC04 encountered an error while searching for the catalog name in the cluster record of the catalog.
			CKEYRNG	IDCRC04 encountered an error while searching for the high key value in a given CRA record.
			CRAOPEN	IDCRC04 encountered an error while searching for either the owning catalog name or the volume serial in the CRA record.
			CTTBLD	IDCRC04 encountered an error while searching for the entry type of the catalog CI in the CRA record.
			GETPRT	IDCRC04 encountered an error while searching for the entry type or the entry name in the CRA record.
			INTASOC	IDCRC04 encountered an error while searching for the associated entry type or entry name fields in the CRA records.
			INTSORT	IDCRC04 encountered an error while searching for the name in a given CRA record.
			INTVEXT	IDCRC04 encountered an error while searching for the extension pointer in a given CRA record.
			PRTCMP	IDCRC04 encountered an error while searching for the used length field in a given CRA record.
			PRTDMP	IDCRC04 encountered an error while searching for the used length field in a given CRA record.
			PRTOJVL	IDCRC04 encountered an error while searching for the volume information or high key value in a given CRA record.
			PRTVOL	IDCRC04 encountered an error while searching for the volume timestamp information in a given catalog or CRA record.
IDC1880I	LR1-11	IDCLR01	PRTVOL	Timestamp for the format-4 record could not be read for the CRA volume.
IDC1881I	LR1-12	IDCLR01	INITLZE	Alternate output data set OPEN failed.
IDC1885I	LR1-17	IDCLR01	PRTMCWD	IDCRC04 encountered an error while searching for mismatched fields in a given CRA record. The CRA record had previously been read and had indicated that mismatches existed.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC1887I	RC0-22	IDCRC01	SCANCRA	I/O error encountered on a CRA record.
			TIMESTAMP	Volume timestamp could not be obtained.
IDC1890I **(VS2.03.807)**	DL0-12	IDCDL01	CATCALL	Catalog management indicated that the RACF profile of the data set being deleted could not itself be deleted because it is ineligible for deletion.
IDC1891I **(VS2.03.807)**	DL0-13	IDCDL01	CATCALL	When deleting a RACF indicated data set, the RACF profile could not be found.
IDC1927I	EX0-12	IDCPM01	MARGPARM	Margin values specified are invalid.
IDC1968I	VS0-11	IDCVS01	VTOCPUT	An error occurred during reading or updating of the VSAM time stamp.
	VS0-19	IDCVS01	VTOCPUT	An error occurred during reading or updating of the VSAM time stamp.
	VS0-20	IDCVS01	VTOCPUT	An error occurred during reading or updating of the VTOC VSAM time stamp.
ICD1969I	VS0-12	IDCVS01	VTOCPUT	An error occurred during reading or updating of the VTOC and prevented the alternate track information from being restored after the copy of data.
IDC2011I	UV0-12	IDCSA08	IDCSA08	A function was requested that required data sets to be scratched, but not enough virtual storage was available to build the parameter list for the SCRATCH macro.
		IDCSA09	IDCSA09	There was insufficient storage for the ECB and the message area that are required for SVC 126.
		IDCDL01	MORESP	Insufficient main storage was available for a work area for VSAM catalog management.
		IDCVS03	GETBLK	Insufficient storage for volume VTOC processing.
IDC2035I	TP6-3	IDCTP06	IDCTP06	Invalid ERCNVTAB passed to UERROR.
IDC2100I	SA7-3	IDCSA07	GETENT	The return code for an OS locate request was neither 0 nor 8.
IDC2101I	SA7-4	IDCSA07	UPDATENT	The return code from an OS recatalog request was neither 0 nor 8.
IDC2360I	SA7-8	IDCSA07	IDCSALC	The ULOCATE function, after locating a data set name in the catalog, determined that either the data set resides on a different device type than the catalog indicates or a duplicate data set name exists.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC2364I	SA7-12	IDCSA07	VSAMLOC	The ULOCATE function, after locating a data set name in the catalog, determined it was not a nonVSAM data set. The situation was probably caused by a duplicate data set name.
IDC2370I	IO5-1	IDCIO05	READJFCB	The READJFCB routine was unable to read the JFCB entry.
IDC2371I	IO5-2	IDCIO05	BLDBLK	The storage required to perform I/O processing was unavailable.
IDC2372I	IO5-3	IDCIO05	CKOPN	The OPEN SVC encountered an error that prevented the opening of the DCB and further I/O processing.
IDC2373I	IO5-6	IDCIO05	RETRY	Invalid password.
IDC2374I	IO5-7	IDCIO05	CLOSESTD	The CLOSE SVC encountered an error that prevented the closing of the DCB and further I/O processing.
IDC2375I	IO5-8	IDCIO05	SYNAD	A SYNAD error occurred during EXCP I/O processing; the message indicates the data set or volume label.
	IO5-9	IDCIO05	SYNAD	A SYNAD error occurred during EXCP I/O processing; the message indicates the data set or volume label.
IDC2376I	IO5-7	IDCIO05	CLOSESTD	CLOSE SVC encountered an error which resulted in the CLOSE ABEND exit being entered.
IDC2381I	SA6-1	IDCSA06	UCBPOST	The UCB could not be updated because updating would cause duplicate label in the system.
IDC2382I	SA6-2	IDCSA06	UCBCHECK	The volume was not mounted for exclusive use because it was mounted when dynamic allocation was invoked.
IDC2386I	SA6-6	IDCSA06	MOUNTCTL	The previous volume could not be demounted; therefore, the assigned unit could not be used to mount needed volumes.
IDC2387I	SA6-7	IDCSA06	MOUNTCTL	The specified volume could not be mounted.
IDC2388I	SA6-8	IDCSA06	DEMNTVOL	The specified volume could not be demounted.
IDC2389I	SA6-9	IDCSA06	DEMNTVOL	The specified volume was demounted but data could not be destaged.
IDC2390I	SA6-10	IDCSA06	ENQDEQ	The volume serial number is shared; therefore, the specified volume could not be enqueued for exclusive use.
IDC2391I	SA6-11	IDCSA10	SETSTAE	The ESTAE macro returned a nonzero return code that

Message	STID	Module	Procedure	Situation That Caused Message
				prevented recovery protection in case of an abnormal termination.
IDC2399I	SA7-7	IDCSA07	IDCSALC	The ULOCATE function, after locating a data set name in the catalog, determined that either the data set resides on a different volume than the catalog indicates or a duplicate data set name exists.
IDC2533I	AL0-9	IDCAL01	MEMRENAM	A member name is specified with a generic name.
IDC2552I	DL0-2	IDCDL01	PARAMCHK	The type of the entry to be deleted was retrieved from the catalog, but the type is not one the user is allowed to delete.
IDC2553I	DL0-3	IDCDL01	PARAMCHK	The type of the entry to be deleted was retrieved from the catalog, but the type conflicts with the erase option.
IDC2557I	DL0-7	IDCDL01	PARAMCHK	The scratch option was specified for an object of an invalid type.
IDC2559I	DL0-9	IDCDL01	MEMDLETE	A generic name was specified when attempting to delete a partitioned data set member.
IDC2563I	LC1-4	IDCLC01	INITPROC	Either the allocation request conflicts with the type specification of cluster, PGSPC, AIX, PATH, ALIAS, or GDG, space, nonVSAM, or user catalog, or the volume request conflicts with the type specification of cluster, PGSPC, AIX, PATH, ALIAS, or GDG.
		IDCLO02	AUPROC	The allocation request conflicts with a nonVSAM, ALIAS, or GDG or user catalog entry specified in the entry list or volume request conflicts with an alias entry specified in the entry list.
			VPROC	The allocation request conflicts with a space (volume) entry specified in the entry list.
IDC2616I	MP0-16	IDCMP01	CLUSPROC	A path import operation failed.
		IDCRM01	CLUSPROC	A path import operation failed.
IDC2618I	MP0-18	IDCMP01	CLUSPROC	An invalid OBJECTS subparameter was found.
IDC2621I	MP0-21	IDCRM01	CLUSPROC NVSAMPROC GDGPROC UCATPROC ALISPROC	The object named could not be imported.
IDC2630I	CC0-3	IDCCC01	CATALOG	VSAM catalog management gave a return code of 8 after trying to define the OS/VS catalog entry. A locate on the VSAM catalog

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				indicated that the duplicate data set is a VSAM data set.
IDC2640I	BI0-1	IDCBI01	LOCPROC	The file identified via OUTFILE or OUTDATASET is not an alternate index.
IDC2642I	BI0-3	IDCBI01	LOCPROC	The alternate index identified in the message is not related to the base cluster identified via INFILE or INDATASET.
IDC2647I	BI0-8	IDCBI01	INITPROC	Storage was not available to obtain buffers and work areas.
IDC2648I	BI0-9	IDCBI01	JCPROC FINPROC	DD statements for sort work files are either missing or in error.
IDC2649I	BI0-10	IDCBI01	DEFPROC	A sort work area was obtained smaller than that required and job control for sort work files was missing or in error.
IDC2650I	BI0-11	IDCBI01	DEFPROC	An internal sort could not be completed and job control for sort work files was missing or in error.
IDC2651I	BIO-12	IDCBI01	DEFPROC	Define of sort work files failed.
IDC2654I	BI0-15	IDCBI01	FINPROC	The alternate index was not built due to severe errors.
IDC2655I	BI0-16	IDCBI01	CATPROC	Catalog information was not returned for a locate request.
IDC2656I	BI0-19	IDCBI01	CATPROC	A VSAM catalog locate failed with a nonzero return code.
IDC2660I	RC0-3	IDCRC01	CKNAMES	The object named is not a VSAM cluster or valid nonVSAM data set.
		IDCRC02	CLUSPROC	The object named is not a VSAM cluster.
			GDGPROC	The object names is not a GDG or has invalid associations to a GDG.
			NVSMPROC	The object named was not a nonVSAM data set or a user catalog.
IDC2666I	RC0-11	IDCRC01	SYNCH	The selected entry was not found in the selected CRA.
IDC2668I	RC0-13	IDCRC01	OBJVOLCK	A required volume was not supplied in the CRA keyword.
IDC2671I	RC0-16	IDCRC01	CKCATNM	The CRA has a different name than the others being processed.
IDC2673I	RC0-19	IDCRC01	BUILDCRV	Required information about the volume could not be obtained.
IDC2675I	RC0-21	IDCRC01	CKNAMES	The same name was found in more than one CRA.
IDC2677I	RC0-1	IDCRC01	EXPORTDR	The data set was not exported because of the error indicated in previous messages.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC2872I	LR1-3	IDCLR01	CRAOPEN	The catalog specified in the input for compare was not the owning catalog found in the CRA.
IDC2873I	LR1-4	IDCLR01	CATOPEN	Catalog could not be opened, therefore the compare option was ignored.
			CRAOPEN	The CRA opened belongs to a catalog other that the one specified in the compare.
IDC2876I	LR1-6	IDCLR01	CRAOPEN	A verify was issued after opening a CRA and it failed.
IDC2879I	LR1-10	IDCLR01	CATOPEN	IDCRC04 could not find the catalog name from the cluster record or the volume serial of the catalog so it could not lock out all other usage of the CRA while it is being listed.
IDC2882I	LR1-13	IDCLR01	CTTBLD	LISTCRA encountered an error reading the catalog control record.
IDC2884I	LR1-7	IDCLR01	CATOPEN	A verify was issued after opening a catalog and it failed.
IDC2886I	RC0-18	IDCRC01	ERRCK	CRA can not be opened because of some errors encountered.
IDC2908I	IO0-36	IDCIO03	DSINFO	No data set or volume is allocated under that indicated DD name, or no DD statement corresponding to the given DD name could be found.
		IDCSA08	IDCSA08	No volume is allocated under the indicated DD name, or no DD statement corresponding to the given DD name could be found.
IDC2909I	IO0-37 (without (SU 5752-824) SA0-1 (with (SU 5752-824)	IDCSA08	IDCSA08	An error occurred while an attempt was being made to scratch the indicated data set.
IDC2910I	IO0-38 (without (SU 5752-824) SA0-2 (with (SU 5752-824)	IDCSA08	IDCSA08	None of the volumes specified for the data set is mounted.
IDC2912I	IO0-40 (without (SU 5752-824) SA0-4 (with (SU 5752-824)	IDCSA08	IDCSA08	The data set to be scratched is password-protected, and the operator didn't supply the proper password.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC2913I	IO0-41 (without (SU 5752-824) SA0-5 (with (SU 5752-824)	IDCSA08	IDCSA08	The OVERRIDE option wasn't specified, and the data set's retention period hasn't expired.
IDC2914I	IO0-42 (without (SU 5752-824) SA0-6 (with (SU 5752-824)	IDCSA08	IDCSA08	The VTOC couldn't be read because of an I/O error.
IDC2915I	IO0-43 (without (SU 5752-824) SA0-7 (with (SU 5752-824)	IDCSA08	IDCSA08	A required unit was not available for mounting.
IDC2916I	IO0-48 (without (SU 5752-824) SA0-8 (with (SU 5752-824)	IDCSA08	IDCSA08	The data set to be scratched was in use.
IDC2917I (SU 5752-824)	SA0-9	IDCSA08	IDCSA08	No RACF profile could be found for the specified data set.
IDC2918I (SU 5752-824)	SA0-10	IDCSA08	IDCSA08	The user does not have the proper RACF authorization for the specified data set.
IDC2919I (SU 5752-824)	SA0-11	IDCSA08	IDCSA08	The parameter list passed to URACHECK macro is invalid.
IDC2930I (SU 5752-824	SA0-12	IDCSA08	IDCSA08	The data set to be scratched is RACF-protected and the caller doesn't have the proper authorization.
IDC2950I	TP1-1	IDCTP01	IDCTP01	Either (1) no format list or static text identification was passed as input, or (2) no valid bits in FMTFLGS were turned on, or (3) the input or output length specified was less than 1.
IDC2951I	TP1-2	IDCTP01	IDCTP01	The output column specified is not within the print line.
IDC2952I	TP1-3	IDCTP01	BDCONV	For binary to decimal conversions, the input data length was more than 4 or the converted length was more than 16.
			PUPCONV	For packed to unpacked conversions, the converted length was more than 15, or the input data length was more than 8.
IDC2953I	TP1-4	IDCTP01	REDO	A REDO structure is nested.
IDC2954I	TP1-6	IDCTP05	IDCTP05	The requested static text entry was not in the specified module.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC2955I	TP1-7	IDCTP01	PUPCONV	An invalid packed decimal field was passed by the caller.
IDC2960I	VS0-2	IDCVS01	NVSAMCHK	A password-protected data set was found on the VTOC but no DD name with that data set name was specified in the PASSWORDFILE keyword.
IDC2961I	VS0-3	IDCVS01	FMT1CHK	The caller indicated that no nonVSAM data sets could reside on the volume. However, a Format-1 DSCB was found for a nonVSAM data set.
	VS0-15	IDCVS01	FMT1CHK	The caller indicated that no nonVSAM data sets could reside on the volume. However, a Format-1 DSCB was found for a nonVSAM data set.
	VS0-16	IDCVS01	FMT1CHK	The caller indicated that no nonVSAM data sets could reside on the volume. However, a Format-1 DSCB was found for a nonVSAM data set.
IDC2962I	VS0-4	IDCVS01	CATCHK	The volume entry could not be found in the VSAM catalog.
IDC2963I	VS0-5	IDCVS01	FMT4READ	The volume services module positioned to the first record in the VTOC, but it was not a Format-4 DSCB.
	VS0-17	IDCVS01	FMT4READ	The volume services module positioned to the first record in the VTOC, but it wasn't a Format-4 DSCB.
	VS0-18	IDCVS01	FMT4READ	The volume services module positioned to the first record in the VTOC, but it wasn't a Format-4 DSCB.
IDC2964I	VS0-6	IDCVS01	SCRATCH	An error occurred that prevented the scratching of data sets. The volume is still flagged as a VSAM volume.
IDC2965I	VS0-7	IDCVS01	SCRATCH	An error occurred reading or updating the VTOC that prevented the scratching of any more data sets.
IDC2966I	VS0-8	IDCVS01	RECATLOG	An I/O error occurred that prevented any recataloging of nonVSAM data sets.
IDC2967I	VS0-10	IDCVS01	RECATLOG	An I/O error occurred that terminated the recataloging of data sets.
IDC2971I	VS0-21	IDCVS01	OPENVTOC	A reserve for a volume with the HAVE option was unsuccessful.
IDC2972I	VS0-23	IDCVS03	USAGE	A nonzero return code was received from LSPACE.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3003I	UV0-3	IDCAL01	IDCAL01	The VSAM catalog could not be opened, or another severe error occurred.
		IDCBI01	TERMPROC	Either (1) a severe error was encountered in processing the base cluster, or (2) the EXTERNALSORT parameter was specified but the job control for sort files was missing or in error.
		IDCCC01	IDCCC01	A severe error occurred. Conversion of the OS catalog was not attempted or terminated if begun.
		IDCDE01	IDCDE01	The VSAM catalog to contain the defined object could not be opened, or another severe error occurred.
			MODELPRC	The VSAM catalog containing the model object could not be opened.
		IDCLC01	IDCLC01	A severe error occurred. Listing of the catalog was not attempted or terminated if begun.
		IDCMP01	IDCMP01	A severe error occurred.
		IDCPR01	IDCPR01	Either (1) an error occurred opening the input or alternate output data sets, or (2) a unrecoverable error occurred while retrieving or printing a record, or (3) more than three I/O errors occurred while retrieving records.
			TEXTPSET	The static text subtitle line could not be retrieved.
			DELIMSET	An incompatible use of delimiters was found during a data set print operation.
		IDCRC01	EXITTHE	Function was not completed because a severe error was encountered.
		IDCRM01	IDCRM01	A severe error occurred.
		IDCRP01	IDCRP01	Either (1) an error occurred opening the input or output data sets, or (2) a unrecoverable error occurred while copying the data set, (3) more than three I/O errors occurred while copying the data set, (4) an error occurred while attempting a catalog reload, or (5) a nonrelative record input data set did not have a non-empty relative record output data set.
			DELIMSET	An incompatible use of delimiters was found during a data set copy operation.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
		IDCRS05 (VS2.03.808)	CKERR	A severe error occurred which prevented further processing.
		IDCVY01	IDCVY01	The VSAM data set to be verified could not be opened, or the verify was not successful.
		IDCXP01	IDCXP01	A severe error occurred.
IDC3004I	UV0-4	IDCAL01	ALTERPRC	Storage was not available for one of the following: the volume list or the PASSWALL field.
			DALCPROC	Storage was not available for volume list.
			IDCAL01	Storage was not available for the CTGPL, CTGFV, and CTGFLs.
			INDEXPRC	Storage was not available for the index parameter list if KEYS was specified.
			LOCATPRC	Storage was not available for the catalog work area.
		IDCCC01	CNVTINIT	Storage was not available for: 22 levels of OS/VS catalog index blocks, VSAM parameter list, data addressed by the VSAM parameter list, or a VSAM catalog work area.
		IDCDE01	IDCDE01	Storage was not available for the CTGPL and CTGFVs.
			DALCPROC	Storage was not available for volume list.
		IDCDE02	ALLCPROC	Storage was not available for one of the following: CTGFLs, the volume list, the file sequence list, or the device type list.
			KEYPROC	Storage was not available for one of the following: the AMDSBCAT CTGFL and the AMDSBCAT field, or the key range list.
			MODELPRC	Storage was not available for the catalog parameter list or the catalog work area.
			NAMEPROC	Storage was not available for the CTGFLs.
			PROTPROC	Storage was not available for the CTGFLs needed to set up the protection attributes.
		IDCIO01	PUTREP	Storage was not available for the input work area.
		IDCIO02	BUILDACB	Storage was not available for the ACB or the EXLST.
			BUILDDBK	Storage was not available for the required I/O areas.
			BUILDRPL	Storage was not available for the input work area or the RPL.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			CKNONOP	No storage is available for the input work area required to process spanned, nonVSAM records.
			DSDATA	No space available to read the Label Cylinder.
			OPENRTN	Storage was not available for the IDCSTR.
		IDCLC01	INITPROC	Storage was not available for one of the following: catalog parameter lists, catalog work areas, or the static text used in the catalog listing.
		IDCLR01	ADDASOC	Storage was not available for the association table extension.
			BLDVEXT	Storage was not available for the VEXTTBL extension.
			CTTBLD	Storage was not available for the CI translate table.
			INITLZE	Storage was not available for the initial ASSOCTBL and VEXTTBL.
			INTASOC	Storage was not available for the association table extension.
		IDCMP01	FPLPROC	Storage was not available for CTGFLs.
			BPASPROC	Storage was not available for the PASSWALL field.
			CLUSPROC	Storage was not available for the catalog work area.
			CPLPROC	Storage was not available for the CTGPL.
			CTLPROC	Storage was not available for the catalog work area.
			DELTPROC	Storage was not available for the catalog work area.
			FVTPROC	Storage was not available for the CTGFV.
			LVLPROC	Storage was not available for one of the following: the catalog work area, CTGFLs, or volume serial lists.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3004I (continued)		IDCPM01	TESTPARM	Storage was not available for the Test Option Data Area.
		IDCRC01	IDCRC01	Storage was not available for one of the tables required by EXPORTRA.
		IDCRC02	CLUSPROC	Storage was not available for the control record output buffer.
			CTLGPROC	Storage was not available for the catalog work area.
			GDGPROC	Storage was not available for the control record output buffer.
			LOCPROC	Storage was not available for the CPL, FPL and the catalog work area.
			NVSMPROC	Storage was not available for the control record output buffer.
			SAVEPROC	Storage was not available for the input record save area.
		IDCRI01	GETSPACE	Storage was not available for The FDT.
			IDCRI02	Storage was not available for one of the following: work space or the FDT.
			INREPEAT	Storage was not available for the FDT.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3004I (continued)			RIINIT	Storage was not available for the Historical Data Area.
			SCANCMD	Storage was not available for the FDT.
		IDCRM01	ALISPROC	Storage was not available for the catalog data record buffer.
			BFPLPROC	Storage was not available for the FPLs.
			BPASPROC	Storage was not available for the PASSWALL information.
			CLUSPROC	Storage was not available for the buffer area or volume list.
			CPLPROC	Storage was not available for the catalog parameter list.
			CTLGPROC	Storage was not available for the catalog work area.
			DELTPROC	Storage was not available for the catalog work area.
			FVTPROC	Storage was not available for the FVT or FPLs.
			GDGPROC	Storage was not available for the catalog data record.
			LVLRPROC	Storage was not available for the volume serial list, the device types list, or the file sequence number list.
			NFVTPROC	Storage was not available for the FVT or FPLs.
			NVSMPROC	Storage was not available for the catalog data record.
			RANGPROC	Storage was not available for the range list.
			UCATPROC	Storage was not available for the catalog data record.
		IDCRS01 **(VS2.03.808)**	IDCRS01	Storage was not available for automatic storage for modules IDCRS01–IDCRS07.
			INIT	Storage was not available for any one of the following: record access blocks, umacro parameter lists, control interval translate table, and UIOINFO return area.
		IDCRS03 **(VS2.03.808)**	GETTAB	Storage was not available for tables needed for association checking.
			PROCVOL	Storage was not available for work areas for bit maps.
			VERB	Storage was not available for GDG association checking.
		IDCRS04 **(VS2.03.808)**	NINUT	Storage was not available for FIND processing.
			NXPND	Storage was not available for the extension to the FIND work area.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3004I (Continued)		IDCRS05 **(VS2.03.808)**	BLDRLST	Storage was not available for the RESVOL table.
			BLDVLST	Storage was not available for the VOLSERTB table.
		IDCRS06 **(VS2.03.808)**	WFDEF	Storage was not available for the work area used for the UCATLG parameter list to define the work file.
		IDCRS07 **(VS2.03.808)**	RENMSETV	Storage was not available for the CAMLST area for RENAME operations.
			SCRATCHP	Storage was not available for the volume list for the SCRATCH macro.
		IDCXP01	ALTRPROC	Storage was not available for the CTGFV.
			CLUSPROC	Storage was not available for the control record output buffer.
			CTLGPROC	Storage was not available for the second catalog work area obtained when the first work area was too small.
			DELTPROC	Storage was not available for the CTGPL or the catalog work area.
			LOCPROC	Storage was not available for the CTGPL or the catalog work area.
			MORESP	Storage was not available for the catalog work area.
IDC3006I	UV0-6	IDCPR01	DELIMSET	Beginning positioning failed.
		IDCRP01	DELIMSET	Beginning positioning failed.
IDC3007I	(See Note at end of list)	IDCAL01	IDCAL01	The catalog return code was nonzero for an alter request.
			CHECKPRC	The catalog return code was nonzero for a locate request.
			LOCATPRC	The catalog return code was nonzero for a locate request.
		IDCBI01	FINPROC	The catalog return code was nonzero for a locate request against the base cluster or alternate index, or for a define request for external sort work files.
		IDCDE01	IDCDE01	The catalog return code was nonzero for a define request.
		IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	The catalog return code was nonzero for a request to locate a model object.
		IDCDL01	CATCALL	The catalog return code was nonzero for a delete request. This message is not issued for a return

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				code of 160, however, because 160 indicates a normal condition.
			FINDTYPE	The catalog return code was nonzero for a locate request.
		IDCLC02	LOCPROC	The catalog return code was nonzero for a locate request.
		IDCMP01	CTLGPROC	The catalog return code was nonzero.
			DELTRPOC	The catalog return code was nonzero.
		IDCRC02	CTLGPROC	The catalog return code was nonzero for a locate request.
		IDCRM01	CTLGPROC	The catalog return code was nonzero for a define or alter request.
			DELTPROC	The catalog return code was nonzero for a delete request.
		IDCRS01 **(VS2.03.808)**	INIT	The catalog return code was nonzero for a locate request.
		IDCRS06 **(VS2.03.808)**	WFDEF	The catalog return code was nonzero for a define request.
			WFDEL	The catalog return code was nonzero for a delete request.
		IDCXP01	CTLGPROC	The catalog return code was nonzero for a delete, alter, or locate request.
			DELTPROC	The catalog return code was nonzero for a delete request.
IDC3008I	UV0-8	IDCSA02	IDCSA02	A UPROMPT or UQUAL macro was issued but Access Method Services was not invoked interactively with TSO.
IDC3009I	(See Note at end of list)	IDCAL01	IDCAL01	The catalog return code was nonzero for an alter request.
			CHECKPRC	The catalog return code was nonzero for a locate request.
			LOCATPRC	The catalog return code was nonzero for a locate request.
		IDCBI01	FINPROC	The catalog return code was nonzero for a locate request against the base cluster or alternate index, or for a define request for external sort work files.
		IDCCC01	CATALOG	The VSAM catalog management return code was nonzero for a define, locate, or alter request.
		IDCDE01	IDCDE01	The catalog return code was nonzero for a define request.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3009I (Continued)		IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	The catalog return code was nonzero for a request to locate a model object.
		IDCDL01	CATCALL	The catalog return was nonzero for a delete request. This message is not issued for a return code of 160, however, because 160 indicates a normal condition.
			FINDTYPE	The catalog return code was nonzero for a locate request.
		IDCLC02	LOCPROC	The catalog return code was nonzero for a locate request.
		IDCMP01	CTLGPROC	The catalog return code was nonzero.
			DELTPROC	The catalog return code was nonzero for a delete request.
		IDCRC02	CTLGPROC	The catalog return code was nonzero for a locate request.
		IDCRM01	CTLGPROC	The catalog return code was nonzero for a define or alter request.
			DELTPROC	The catalog return code was nonzero for a delete request.
		IDCRS01 **(VS2.03.808)**	INIT	The catalog return code was nonzero for a locate request.
		IDCRS06 **(VS2.03.808)**	WFDEF	The catalog return code was nonzero for a define request.
			WFDEL	The catalog return code was nonzero for a delete request.
		IDCSA02	IDCSA02	During a UCIR macro, the VSAM catalog return code was nonzero for a generic locate.
		IDCXP01	CTLGPROC	The catalog return code was nonzero for a delete, alter, or locate request.
			DELTPROC	The catalog return code was nonzero for a delete request.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3010I	UV0-11	IDCXP01	RECPROC	The file identified in the INFILE parameter does not match that given in the EXPORT command or any paths over it.
		IDCMP01	RECPROC	The file identified in the OUTFILE parameter does not match the name given in the IMPORT command or any paths over it.
		IDCLC02	ERRPROC	The locate function found that the name given for listing isn't in the specific catalog.
		IDCSA02	IDCSA02	The locate function found that the name given for a generic locate isn't in the specified catalog.
IDC3012I	TP6-9	IDCTP06	CATERCNV	Verbalization of catalog return code 8, Locate.
IDC3013I	TP6-10	IDCTP06	CATERCNV	Verbalization of catalog return code 8, Define.
IDC3014I	TP6-11	IDCTP06	CATERCNV	Error during catalog operation.
IDC3016I	TP6-12	IDCTP06	CATERCNV	Verbalization of catalog return code 4.
IDC3017I	TP6-13	IDCTP06	CATERCNV	Verbalization of catalog return code 20.
IDC3018I	TP6-14	IDCTP06	CATERCNV	Verbalization of catalog return code 56.
IDC3019I	TP6-15	IDCTP06	CATERCNV	Verbalization of catalog return code 60.
IDC3020I	TP6-16	IDCTP06	CATERCNV	Verbalization of catalog return code 68.
IDC3021I	TP6-17	IDCTP06	CATERCNV	Verbalization of catalog return code 72.
IDC3022I	TP6-18	IDCTP06	CATERCNV	Verbalization of catalog return code 80.
IDC3023I	TP6-19	IDCTP06	CATERCNV	Verbalization of catalog return code 84.
IDC3024I	TP6-21	IDCTP06	CATERCNV	Verbalization of catalog return code 148.
IDC3025I	TP6-22	IDCTP06	CATERCNV	Verbalization of catalog return code 156.
IDC3026I	TP6-23	IDCTP06	CATERCNV	Verbalization of catalog return code 172.
IDC3027I	TP6-24	IDCTP06	CATERCNV	Verbalization of catalog return code 176.
IDC3028I	TP6-25	IDCTP06	CATERCNV	Verbalization of catalog return code 184.
IDC3029I	TP6-26	IDCTP06	CATERCNV	Verbalization of catalog return code 192.
IDC3030I	TP6-27	IDCTP06	CATERCNV	Verbalization of catalog return code 196, 200.
IDC3031I	TP6-28	IDCTP06	CATERCNV	Verbalization of catalog return code 204.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3032I	TP6-29	IDCTP06	CATERCNV	Verbalization of catalog return code 208.
IDC3033I	TP6-30	IDCTP06	CATERCNV	Verbalization of catalog return code 248.
IDC3067I	CMO-41	IDCVS01	FMT4CHK	The caller indicated that the volume could not be owned by a VSAM catalog; however, the VSAM ownership bit was set.
	CMO-42	IDCVS01	FMT4CHK	The caller indicated that the volume couldn't be owned by a VSAM catalog; however, the VSAM ownership bit was set.
	CMO-43	IDCVS01	FMT4CHK	The caller indicated that the volume couldn't be owned by a VSAM catalog; however, the VSAM ownership bit was set.
IDC3070I	CMO-36	IDCVS02	PUTLAB	The volume serial number could not be updated on the volume label.
	CMO-37	IDCVS02	PUTLAB	The volume serial number couldn't be updated on the volume label.
	CMO-38	IDCVS02	PUTLAB	The owner name on the volume label could not be updated.
	CMO-39	IDCVS02	PUTLAB	The owner name on the volume label couldn't be updated.
	CMO-52	IDCVS02	PUTLAB	Neither the volume serial number nor the owner name could be updated.
	CMO-53	IDCVS02	PUTLAB	Neither the volume serial number nor the owner name could be updated.
IDC3190I	AL0-24	IDCAL01	PARAMCHK	One of the parameters specified on the command is invalid for the entry type.
IDC3200I	RI0-1	IDCRI01	SCANCMD	The number of positional parameters found (PPARMCNT) exceeds the number defined in the descriptor for the current subparameter list (SUBCOUNT).

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3201I	RI0-2	IDCRI01	BUILDFDT	The input constant length (UNITINDX) exceeds the maximum length defined by the descriptor.
			CONVERT	The input constant length (UNITINDX) exceeds the maximum length defined by the descriptor.
			NXTFIELD	The input constant length (UNITINDX) exceeds the maximum length that the Reader/Interpreter can handle (UNITMAX).
			PACKCVB	The input constant length (UNITINDX) exceeds the maximum length defined by the descriptor.
IDC3202I	RI0-3	IDCRI01	ERROR1	The remainder of a command was bypassed due to an error in it.
			ERROR2	The remainder of a command was bypassed due to an error in it.
IDC3203I	RI0-4	IDCRI01	DSIDCHK	A data set name does not have the correct syntax.
IDC3205I	RI0-6	IDCRI01	SCANCMD	The closing parentheses of a subparameter list was found before any parameters were found in the list or an opening parentheses was found before any keyword was found.
IDC3207I	RI0-8	IDCRI01	ERROR1	A severe error occurred. The condition code is set to 16, and the Reader/Interpreter will terminate processing.
			ERROR2	A severe error occurred. The condition code is set to 16, and the Reader/Interpreter will terminate processing.
IDC3208I	RI0-9	IDCRI01	KWDPARM	A keyword parameter, defined as having a subfield, does not have a left parentheses following the keyword.
IDC3209I	RI0-10	IDCRI01	KWDPARM	A keyword's subfield does not have a closing parenthesis following it.
			POSPARM	A list of constants is not delimited on the right by a closing parenthesis.
IDC3210I	RI0-11	IDCRI01	INREPEAT	The next repetition of a repeated subparameter list does not begin with a left parenthesis.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3211I	RI0-12	IDCRI01	KWDPARM	The descriptor does not define the input keyword as part of the current parameter list.
			NXTFIELD	An input keyword exceeds the maximum allowable length for a keyword.
IDC3212I	RI0-13	IDCRI01	POSPARM	A positional parameter that is not defined as a list begins with a left parenthesis.
IDC3213I	RI0-14	IDCRI01	SETFLAG	An internal table (PARMFLAG) indicates that the keyword just found was found previously in this command.
IDC3214I	RI0-15	IDCRI01	GETDATA	A numeric constant begins with a B or X, but an apostrophe does not follow directly after this character.
IDC3216I	RI0-17	IDCRI01	ERROR1	The remainder of a command, being scanned for syntax-checking purposes only, was bypassed due to an error in it.
			ERROR2	The remainder of a command, being scanned for syntax-checking purposes only, was bypassed due to an error in it.
IDC3217I	RI0-18	IDCRI01	GETQUOTD	A password-delimiting slash appears following a constant that does not allow a password.
			GETSIMPL	A password-delimiting slash appears following a constant that does not allow a password.
IDC3218I	RI0-19	IDCRI01	INREPEAT	The number of sublist repetitions (REPCOUNT) for the current repeated sublist exceeds the maximum repetitions allowed (REPMAX) for this parameter according to the descriptor.
IDC3219I	RI0-20	IDCRI01	IDCRI02	The input verb name does not match any name in IDCRILT.
IDC3220I	RI0-21	IDCRI01	CONVERT	A numeric constant contains a invalid digit.
			PACKCVB	A numeric constant contains an invalid digit.
IDC3221I	RI0-22	IDCRI01	CONVERT	A numeric constant has a value outside the value range specified in the descriptor for this parameter.
			PACKCVB	A numeric constant is too large to fit into a binary fullword.
IDC3223I	RI0-24	IDCRI01	BUILDFDT	The number of constants found in a list (SCLRCNT) exceeds the number allowed (LISTMAX).

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3225I	RI0-26	IDCRI01	NEEDNOTS	A parameter always required for this command is missing, or parameter required when another parameter is coded is missing.
IDC3226I	RI0-27	IDCRI01	NEEDNOTS	An input parameter conflicts with some other input parameter.
IDC3240I	RI1-1	IDCRI04	BUILDFDT	PDENVMBR is larger than HIVALUE or smaller than LOWVALUE.
IDC3241I	RI1-2	IDCRI04	BUILDFDT	The length of a data set name or a DD name is wrong.
IDC3242I	RI1-3	IDCRI04	BUILDFDT	The length of a constant is wrong.
IDC3243I	RI1-4	IDCRI04	GETSPACE	There are too many constants in a list.
			REPLIST	There are more subparameter sets in a list than the command descriptor allows.
IDC3244I	RI1-5,11	IDCRI04	NOTPARMS	Two conflicting parameters are coded.
IDC3246I	RI1-8	IDCRI04	RESOLVE	The TSO terminal user needs to choose between conflicting parameters.
IDC3247I	RI1-6	IDCRI04	NEEDPRMS	Parameters required by the command descriptor are missing.
IDC3248I	RI1-7	IDCRI04	GETSPACE	There are too many numbers in a list.
IDC3249I	RI1-9	IDCRI04	BUILDFDT	The TSO terminal user needs to reply with a correct constant.
IDC3250I	RI1-10	IDCRI04	TOOMANY	The TSO terminal user needs to choose whether or not to ignore excessive constants or subparameters.
IDC3251I	RI1-15	IDCRI04	IDCRI04	Parameter in error is a repeated subparameter.
IDC3253I	RI1-17	IDCRI04	ADDPARM	A UPROMPT to obtain a missing keyword has failed.
IDC3300I	IO0-1	IDCIO02	BLDOCMSG	An error occurred during open of a data set.
	IO05-4	IDCIO05	CHECKOPN	An error occurred during the Open of a data set or the VTOC.
IDC3301I	IO0-2	IDCIO02	BLDOCMSG	An error occurred during close of a data set.
IDC3302I	IO0-3	IDCIO01	BLDAMSG	An error occurred while accessing a data set.
		IDCIO03	BLDAMSG	An error occurred while accessing a data set.
		IDCR506 **(VS2.03.808)**	RECMGMT	A logical error occurred during an I/O operation to the work file.
IDC3303I	IO0-4	IDCIO02	BUILDDBK	The data set to be opened for update processing is not a VSAM data set.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3304I	IO0-5	IDCIO02	DSDATA	A Job Control statement specified for file to OPEN was not found.
IDC3305I	IO0-6	IDCIO02	DSDATA	An attempt was made to open an ISAM data set for output.
IDC3306I	IO0-7	IDCIO02	BUILDDBK	Cannot open an ISAM file for address processing.
			DSDATA	The data set to be opened for physical sequential processing is an ISAM data set.
IDC3307I	IO0-8	IDCIO02	BUILDDBK	The data set to be opened for keyed processing is not a VSAM or ISAM data set.
IDC3308I	IO0-10 IO0-35	IDCIO01	VSAMERR	A record with the same key or relative record number as the input record already exists in the output data set.
IDC3309I	IO0-12	IDCIO01	PUTNONVS	The length for a record to be written is invalid.
			PUTVSAM	For a relative record data set, the length of the record to be written is not equal to the record size of the data set.
IDC3310I	IO0-13	IDCIO03	PTAMDS	The key provided is longer than the key length of the data set.
			PTISDS	The key provided is longer than the key length of the data set.
IDC3311I	IO0-14	IDCIO03	IDCIO03	The data set to be positioned is not a VSAM or ISAM data set.
IDC3312I	IO0-15	IDCIO02	CKNONOP	The DCB OPEN flag was not set by the system OPEN routines for magnetic tape or for a sequential disk file.
IDC3313I	IO0-16	IDCIO01	GETNONVS	An I/O error occurred while reading a nonVSAM data set.
			PUTNONVS	An I/O error occurred while writing a nonVSAM data set.
	IO5-5	IDCIO05	SYNAD	There was a SYNAD error in EXCP I/O processing.
IDC3314I	IO0-17	IDCIO01	VSAMERR	The record to be written has a lower key than the last record in the data set.
IDC3315I	IO0-44	IDCIO02	BUILDDBK	The record length exceeds 32767.
IDC3316I	IO0-19	IDCIO02	BUILDDBK	The data set to be opened is not a VSAM catalog.
IDC3317I	IO0-20	IDCIO01	VSAMERR	Physical error detected in a VSAM file.
		IDCIO02	DSDATA	I/O attempting to read the Label Cylinder.
		IDCIO03	PTAMDS	Physical error detected by VSAM POINT routines.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3318I	IO0-21	IDCIO02	BUILDDBK	(1) Invalid environment or DLBL/TLBL parameters specified, (2) the blocksize is less than one, (3) the blocksize is invalid for a fixed length record format file, or (4) the blocksize is invalid for a variable length record format file.
			CKNONOP	The blocksize specified for an ISAM file is less than the file's true blocksize.
			DSDATA	Invalid parameters specified on the DLBL/TLBL statement.
IDC3321I	IO0-24	IDCIO02	CKNONOP	An open ABEND error was detected.
			ENVFREE	A close ABEND error was detected.
IDC3322I	IO0-25	IDCIO01	IDCIOVY	The data set to be verified is not a VSAM data set.
IDC3325I	IO0-45	IDCIO01	IRSISYN	The blocksize specified for the portable data set is different than that of the portable data set.
IDC3326I	IO0-46	IDCIO02	OPENRTN	The REPLACE option has been specified for output through a path.
IDC3327I	IO0-47	IDCIO01	VSAMERR	Duplicate record in the upgrade set.
IDC3330I	IO0-18	IDCIO03	STOWRTN	Name doesn't exist in the partitioned data set directory.
IDC3331I	IO0-22	IDCIO03	STOWRTN	Name already exists in the partitioned data set directory.
IDC3332I	IO0-26	IDCIO02	BUILDACB	Storage was not available for one of the following: The ACB, the EXLST or area for a modified JFCB.
			BUILDDBK	Storage was not available for one of the following: the DCB, an update DECB, an area for a modified JFCB and DCB exist list.
			BUILDRPL	Storage was not available for one of the following: the input work area or the RPL.
			CKNONOP	No storage was available for an input work area.
			OPENRTN	Storage was not available for the IOCSTR.
		IDCIO03	STOWRTN	No storage available for USTOW macro.
		IDCSA02	IDCSA02	VSAM generic locate requires more storage than the UCIR macro can obtain.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3333I	IO0-32	IDCIO01	COPYCAT	Output VSAM catalog is not empty.
IDC3334I	IO0-33	IDCIO01	COPYCAT	Invalid control interval number in VSAM catalog.
IDC3350I	IO0-11	IDCIO02	DSDATA	Cannot open ISAM data set for output.
		IDCIO03	PTAMDS	An I/O error occurred during a VSAM point operation.
		IDCIO01	VSAMERR	An I/O error occurred in the VSAM access method.
IDC3351I	IO0-9	IDCIO01	VSAMERR	An error was detected by a VSAM macro. The error was not a duplicate record or a record out of sequence.
		IDCIO02	CLOSERTN	The ACB was not closed successfully.
			OPENRTN	The ACB was not opened successfully.
		IDCIO03	PTAMDS	A logical error occurred during a VSAM point operation.
		IDCRS06 (**VS2.03.808**)	RECMGMT	A logical error occurred during an I/O operation to the work file.
IDC3380I	SA6-13	IDCSA06	UCBCHECK	The unit for the specified ddname is not assigned for exclusive control. Therefore, Open could not mount or demount volumes on this unit. Either the volume was already mounted for the job when this job was allocated, or the correct JCL parameters were not specified on the DD statement.
IDC3383I	SA6-3	IDCSA06	UCBCHECK	The UCB does not specify a virtual unit.
IDC3384I	SA6-4	IDCSA06	UCBCHECK	The UCB is not for a 3330 direct-access device.
IDC3385I	SA6-5	IDCSA06	GETEXCL	The UCB indicates that the unit is allocated to a permanently resident or reserved volume.
IDC3392I	SA6-12	IDCSA06	FINDEXCL	The test enqueue indicates the volume is allocated exclusively to the job step, but no unit is available for mounting the volume. All nonshareable units have open VSAM catalogs.
IDC3500I	DE0-3	IDCDE03	IDCDE03	The object parameter list supplied by the user is incorrect.
IDC3501I	DE0-4	IDCDE01 (without **VS2.03.807**) IDCDE02 (with **VS2.03.807**)	MODELPRC	The entry type of a model object is not the same as that of the object being defined, or the entry type of a model object conflicts with the specification of INDEXED, NONINDEXED or NUMBERED.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3503I	DE0-1	IDCDE02	ALLCPROC	The number of elements in the volume list does not match the number of elements in the file sequence list.
IDC3504I	DE0-2	IDCDE02	KEYPROC	The length of the key range list retrieved from a model exceeded the space allotted for the list by IDCDE01.
IDC3505I	DE0-6	IDCDE01	IDCDE01 (without **VS2.03.807**) INTGCHK (with **VS2.03.807**)	Space allocation was incorrectly specified for a VSAM catalog, data set, alternate index, or data space.
IDC3506I	DE0-7	IDCDE01	IDCDE01 (without **VS2.03.807**) INTGCHK (with **VS2.03.807**)	Volumes were not correctly specified for a VSAM data set or alternate index.
IDC3507I	DE0-8	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	The record size was required but not specified for a VSAM data space.
IDC3513I	DE0-14	IDCDE01	IDCDE01	Dynamic allocation error.
IDC3514I	DE0-15	IDCDE02	KEYPROC	The key ranges specified by the user overlap.
		IDCMP01	RANGPROC	The key ranges specified by the user overlap.
IDC3515I	DE0-16	IDCDE02	ALLCPROC	The average record size exceeds the maximum record size.
IDC3516I	DE0-17	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	Key length and position were not specified for a key sequenced data set.
IDC3517I	DE0-18	IDCDE02	ALLCPROC (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	Unequal record sizes were specified for a relative record data set.
IDC3518I	DE0-19	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	REUSE cannot be specified with UNIQUE or KEYRANGES.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3519I	DE0-20	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	A REUSE conflict exists between data and index.
IDC3521I	DE0-22	IDCDE01	IDCDE01 (without **VS2.03.807**) INTGCHK (with **VS2.03.807**)	A RECORDSIZE greater than 32761 was specified for a nonspanned data set.
IDC3522I	DE0-23	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK (with **VS2.03.807**)	SPANNED cannot be specified for a relative record data set.
IDC3523I	DE0-24	IDCDE02	NAMEPROC	Name specified for generation data group exceeds 35 characters.
IDC3524I	DE0-25	IDCDE01	IDCDE01 (without **VS2.03.807** INTGCHK **VS2.03.807** **(with VS2.03.807)**	Key range values are longer than key length.
		IDCDE02	KEYPROC	Key ranges are not in ascending order.
IDC3525I	AL0-23	IDCAL01	CHECKPRC	The password supplied is insufficient to alter key values.
IDC3527I	AL0-3	IDCAL01	LOCATPRC CHECKPRC	The entry retrieved from the catalog was an invalid type for alter requests, or required fields could not be located.
IDC3528I	AL0-4	IDCAL01	LOCATPRC	Passwords were suppressed when the object ot be altered was retrieved from the catalog.
IDC3529I	AL0-5	IDCAL01	IDCAL01	The data set name was longer than 44 characters after the * was replaced with a level name from a generic locate.
IDC3530I	AL0-6	IDCAL01	MEMRENAM	The entryname and the NEWNAME parameters both contain membername but each membername refers to a different partitioned data set.
IDC3536I	AL0-2	IDCAL01	IDCAL01	Either the entryname or the NEWNAME specified a generic name. Both must specify a generic name if one does.
IDC3537I	AL0-12	IDCAL01	CHECKPRC	UNIQUEKEY or UPGRADE was specified for a nonalternate index.
IDC3538I	AL0-13	IDCAL01	CHECKPRC	UNIQUEKEY or UPGRADE was specified for a nonempty alternate index.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3539I	AL0-14	IDCAL01	CHECKPRC	KEYS or RECORDSIZE was specified for a nonempty object.
IDC3540I	AL0-15	IDCAL01	CHECKPRC	A conflict between the control interval and KEYS or RECORDSIZE values exists.
IDC3541I	AL0-16	IDCAL01	CHECKPRC	A conflict exists between the alternate index key values and the base cluster record size.
IDC3542I	AL0-17	IDCAL01	CHECKPRC	Unequal record sizes were specified for a relative record data set.
IDC3545I	AL0-20	IDCAL01	CHECKPRC	Invalid values were specified for KEYS or RECORDSIZE.
IDC3546I	AL0-21	IDCAL01	CHECKPRC	Invalid value specified for KEYS.
IDC3547I	AL0-22	IDCAL01	CHECKPRC	KEYS or RECORDSIZE is invalid with entry type.
IDC3568I	LC1-11	IDCLC01	ENTPROC	Level name starts or ends with an '*'.
IDC3570I	PR0-18	IDCRP01	IDCRP01	Delimiters were specified for a catalog reload.
IDC3572I	PR0-20	IDCRP01	CATRELOD	Target catalog is too small to contain the backup catalog during catalog reload.
IDC3573I	PR0-21	IDCRP01	CATRELOD	Either the catalog name, the volume serial number, or the device type did not match during a catalog reload.
IDC3581I	PR0-16	IDCRP01	IDCRP01	IDCRP01 found beginning and/or ending delimiters when a VSAM catalog is to be copied.
IDC3582I	PR0-14	IDCRP01	IDCRP01	The organization of the input data set is incompatible with that of the output data set.
IDC3583I	PR0-17	IDCRP01	DELIMSET	Invalid delimiters were specified for a data set copy operation.
		IDCPR01	DELIMSET	Invalid delimiters were specified for a data set copy operation.
IDC3584I	PR0-24	IDCRP01	REOVCAT	An attempt was made to copy from or into a catalog with the recoverable attribute.
IDC3592I	XP0-3	IDCXP01	CLUSPROC	The object retrieved from the catalog for export is not a cluster or an alternate index.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3593I	XP0-4	IDCXP01	CLUSPROC	The catalog did not return the entry type, data component name, or LRECL when the object to be exported was located.
		IDCRC01	SYNCH	No data association could be found.
		IDCRC02	ASOCPROC	The catalog did not return the entry type when the object to be recovered was located.
			CLUSPROC	Either (1) the catalog did not return the entry type, data component name, or LRECL when the object to be recovered was located, or (2) the entry type was not a cluster or alternate index.
			CONTRBL	The catalog did not return the entry type, data component name or LRECL when the object to be recovered was located.
			GDGPROC	The catalog did not return the entry type when the object to be recovered was located.
			NVSMPROC	The catalog did not return the entry type when the object to be recovered was located.
IDC3596I	XP0-7	IDCXP01	CLUSPROC	The data set to be exported has been marked as not usable.
IDC3602I	MP0-9	IDCMP01	IDCMP01	Import of the data set failed after a successful define.
		IDCRM01	IDCRM01	Import of the data set failed after a successful define.
IDC3606I	MP0-1	IDCMP01	CLUSPROC	The portable data set's first record was not valid, or the record immediately prior to the data component's records was not valid.
		IDCRM01	IDCRM01	Opens of the portable data set failed.
			ALISPROC	A catalog control record for an alias entry was not read.
			CLUSPROC	There was no volume list from the input area.
			NVSMPROC	A catalog control record from the portable data set was not read.
			OPENPROC	Header flags were not set properly in the first record.
			UCATPROC	A catalog control record for a user catalog was not read.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3607I	MP0-13	IDCMP01	DUPNPROC	The temporary flag is not set in the catalog entry with the same name as the object being imported. If NEWNAME is specified, the temporary flag is not set in the entry with the new name.
IDC3608I	MP0-10	IDCMP01	CNCTPROC	The VSAM catalog could not connect the user catalog.
IDC3609I	MP0-5	IDCMP01	CLUSPROC	The VOLUMES parameter was not specified.
		IDCRM01	CLUSPROC	No volume information was available.
IDC3610I	MP0-6	IDCMP01	CNCTPROC	The device list was not specified for connect of a user catalog.
IDC3612I	MP0-8	IDCMP01	DUPNPROC	The catalog entry with the same name as the object being imported is not a cluster or alternate index.
IDC3613I	MP0-14	IDCMP01	CLUSPROC	The open of the portable data set was not successful.
		IDCRM01	IDCRM01	The open of the portable data set was not successful.
IDC3614I	MP0-7	IDCMP01	CLUSPROC	The object names specified by the user do not match the object names found in the portable data set.
IDC3615I	MP0-15	IDCMP01	RECPROC	The data set name on the OUTFILE JCL statement does not agree with the name found in the portable data set or, if NEWNAME is specified, the new name for the data set, or the name specified is not the name of path over the object to be imported.
IDC3617I	MP0-17	IDCMP01	DUPNPROC	The attributes of a predefined data set conflict with those of the data set to be imported.
IDC3619I	MP0-19	IDCRM01	ALTRPROC	The catalog return code was nonzero.
IDC3624I	MP0-24	IDCRM01	IDCRM01	The UIOINFO issued to obtain the output data set name failed.
IDC3625I **(VS2.03.807)**	MP0-25	IDCMP01	CLUSPROC	IDCMP01 found an empty data set with the same name but the INTOEMPTY keyword was not specified.
IDC3633I	CC0-5,6	IDCCC01	CONVERT	A valid catalog entry was not found while reading the OS/VS catalog.
			PUSHDOWN	An Index Control Entry was not found while reading the OS/VS catalog.
			VOLINDEX	A Volume Index Control Entry was not found while reading the OS/VS catalog.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3641I	BI0-2	IDCBI01	LOCPROC	The file identified in INFILE or INDATASET is not a base cluster.
IDC3643I	BI0-4	IDCBI01	OPENPROC	The base cluster is empty.
IDC3883I	LR1-14	IDCLR01	ERROR	More than 50 errors occurred while trying to complete the LISTCRA.
IDC3900I	RI1-12	IDCSA02	IDCSA02	The PUTGET macro failed in one of the following ways: no buffer space available, invalid parameters, TSO profile command did not allow prompting.
IDC3901I	RI1-13	IDCSA02	IDCSA02	The TSO service routine did not qualify a data set name because no qualifiers were found, invalid parameters, or failure to locate the data set name.
IDC3902I	RI1-14	IDCSA02	IDCSA02	The TSO default service routine did not qualify a data set name because no qualifiers were found, invalid parameters, or failure to locate the data set name.
IDC3903I **(Deleted for VS2.03.807)**	IO0-27	IDCSA02	IDCSA02	Dynamic allocation failed because storage not available or data set name not found.
IDC3904I **(Deleted for VS2.03.807)**	IO0-28	IDCSA02	IDCSA02	Data set deallocation failed because of a problem with overriding dispositions.
IDC3905I **(Deleted for VS2.03.807)**	IO0-29	IDCSA02	IDCSA02	Data set allocation or deallocation failed because storage not available, data set name not found, or problem with overriding dispositions.
IDC3906I **(Deleted for VS2.03.807)**	IO0-30	IDCSA02	IDCSA02	Volume dynamic allocation failed because storage not available or invalid unit number.
IDC3907I **(Deleted for VS2.03.807)**	IO0-31	IDCSA02	IDCSA02	Dynamic concatenation of volumes failed because storage not available or invalid parameter list.
IDC4227I	RI0-28	IDCRI01	GETNEXT	An ELSE command appears without a matching IF-THEN command (THENFLAG is not on with DOFLAG off).
IDC4228I	RI0-29	IDCRI01	GETNEXT	An END command appears without a matching DO command (DOFLAG is off).
IDC4229I	RI0-30	IDCRI01	MODAIIF	An IF command relational expression does not follow the required format.
IDC4230I	RI0-31	IDCRI01	MODALSET	A SET command assignment expression does not follow the required format.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC4232I	RI0-33	IDCRI01	MODALIF	A THEN keyword does not appear in an IF command.
IDC4236I	RI0-37	IDCRI01	IDCRI03	End-of-file occurred, but EOFOK flag is off, indicating that end-of-file occurred in the middle of a command.
IDC4237I	RI0-38	IDCRI01	MODALIF	The current IF command nesting level (NESTLVL) exceeds the maximum level allowed (IFNSTMAX).
IDC4999I		IDCSA01	PRNTERR	UABORT error message printed. See ABORT codes.
IDC01002I	RS0-3	IDCRS01	INIT	Informational message indicating the catalog to be reset and the time stamp on the volume.
IDC01011I	RS0-12	IDCRS01	PROCCRA	Informational message indicating the CRA to be reset and the time stamp on the volume.
IDC01037I	RS0-47	IDCRS01	UPDCAT	Informational message indicating that RESETCAT processing has been completed for the indicated catalog.
IDC11003I	RS0-4	IDCRS06	RECMGMT	IGNORE was specified and an I/O error was encountered.
IDC11015I	RS0-16	IDCRS06	RECMGMT	IGNORE was specified and an I/O error was encountered.
IDC11022I	RS0-22,48	IDCRS02	PROCTYPE	An object contains a dependency on a record that does not exist.
IDC11023I	RS0-23,24	IDCRS02	VERA VERC VERG VERR	An entry is chained to a record of a type different than anticipated or the object noted consists of an incomplete set of records. If the control interval number of the expected association is not given, then no association for that object exists in the base record; an association for that type is required for the entry name noted.
IDC11029I	RS0-31	IDCRS03	VLNRESET VLRESET	The suballocated data space has been corrected to reflect what is on the volume. This correction occurs if entries are deleted by RESETCAT or space stated as suballocated is not suballocated (that is, the space map is incorrect on entry to RESETCAT).
IDC11031I	RS0-33	IDCRS03	CHKUNQ	The unique data or index component has less space described than the data space. Informational message to indicate that space exists which is not in use.

Note: These messages IDC01002I through IDC31038I are for **VS2.03.808** only.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC11033I	RS0-35	IDCRS03	CHKUNQ VLNRESET	A unique data set on a volume not being reset has no corresponding Data or INDEX component.
IDC11036I	RS0-46	IDCRS03	CHKDSDIR	The data set named may have invalid space information. The extents occupied by the named data set are not in conflict with any other VSAM data set or with the system; however, a self-checking field failed to check.
IDC11040I	RS0-38	IDCRS03	VOLCHK	The VSAM Format 1 DSCB did not have a corresponding space header in the volume record. Therefore, the catalog does not account for the space allocated to the data set.
IDC11041I	RS0-39	IDCRS03	VOLCHK	The extents in the space header for the data space noted were not identical to the extents in the corresponding Format 1 DSCB.
IDC11042I	RS0-40	IDCRS03	VOLCHK	The space header for the data space referred to a nonexistent Format 1 DSCB.
IDC11043I	RS0-41	IDCRS03	VOLCHK	The timestamp for the volume record did not match the timestamp in the VTOC.
IDC11044I	RS0-42	IDCRS03	VOLCHK	The attempt to scratch the file for the reason stated in message IDC11040I failed.
IDC21009I	RS0-10	IDCRS01 IDCRS03	INIT MARKUNUS	A multivolume file existed on a volume prior to reset.
IDC21020I	RS0-21	IDCRS06	CRAMNT	A volume needed for the reset was not specified in a CRAFILES parameter.
IDC21024I	RS0-25	IDCRS02	VERX	The alias chain for a USERCATALOG or NONVSAM entry is invalid.
IDC21025I	RS0-26	IDCRS03	VERB	The records associating the GDG data set with the GDG base are in error.
IDC21026I	RS0-27	IDCRS02	SETCI	A previous message indicated an error which resulted in this entry being deleted from the catalog.
IDC21027I	RS0-28	IDCRS03	VLRESET	The CRA extents or catalog extents
	RS0-29	IDCRS03	VLRESET	have no matching extents in the data space.
IDC21030I	RS0-32	IDCRS03	MARKUNUS	The entry noted claims space on a volume. That space is not allocated to that entry.

Note: Messages on this page are for **VS2.03.808** only.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC21032I	RS0-34	IDCRS02 IDCRS03	VERCI VERB	An object of the type specified was defined over the entry named as *entryname*. However, the records describing the object could not be found. Therefore, an object of the type specified was deleted from the given *entryname's* description. No name for the deleted object is given because the record with its name cannot be found.
			VLRESET	what space is available for suballocation on a volume, is not the correct length in the catalog.
IDC21045I	RS0-43	IDCRS07	RENAMEP	An attempt was made to reset an object which bears the same name as some other object in the catalog.
IDC21046I	RS0-44	IDCRS07	RENAMEP	An attempt was made to reset a unique object into a catalog which contains an object of the same name.
IDC21047I	RS0-45	IDCRS07	RENAMEP	An attempt was made to reset a unique object into a catalog which contains an object of the same name.
IDC31000I	RS0-1	IDCRS02	INIT	The catalog specified for reset is not a recoverable catalog.
IDC31001I	RS0-2	IDCRS01	INIT	The master catalog was specified for reset.
IDC31004I	RS0-5	IDCRS06	WFDEF	DEFINE failed for the workfile.
IDC31005I	RS0-6	IDCRS02	INIT	The workfile was defined in the catalog to be reset.
IDC31006I	RS0-7	IDCRS07	CATEOV	A physical I/O error when accessing the catalog was encountered.
IDC31007I	RS0-8	IDCRS07	CATEOV	A logical I/O error was encountered while extending the catalog.
IDC31008I	RS0-9	IDCRS01	INIT	An error was encountered when trying to access the data set specified in the CATALOG parameter.
IDC31010I	RS0-11	IDCRS01	MERGECRA	The CRA was specified for reset. but it belongs to a catalog other than the catalog to be reset.
IDC31012I	RS0-13	IDCRS06	RECMGMT	The workfile relative record number limit has been exceeded.
IDC31013I	RS0-14	IDCRS01	MERGECRA	A preceding message indicates that either Open failed for the CRA, Close failed for the CRA, or the CRA does not belong to the catalog to be reset.
IDC31014I	RS0-15	IDCRS06	WFDEL	DELETE failed for the work file.

Note: Messages on this page are for **VS2.03.808** only.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC31016I	RS0-17	IDCRS01	INIT	The CRAFILES parameter specified no CRA with the ALL option; therefore, no volume was specified for reset.
IDC31017I	RS0-18	IDCRS01	INIT	Some other task is open to the catalog requested to be reset.
IDC31018I	RS0-19	IDCRS01	UPDCAT	RESETCAT required a volume that could not be allocated.
IDC31019I	RS0-20	IDCRS01	INIT	The CRAFILES parameter specified the same volume serial number via *dnames*.
IDC31035I	RS0-37	IDCRS01 IDCRS03	UPDCAT VLNRESET	In a CRA, either the volume record for the *volser* indicated does not exist or one of its secondary records does not exist.
IDC31038I	RS0-49	IDCRS01	UPDCRA	Either Open or Close failed for the CRA.

Note: Messages on this page are for **VS2.03.808** only.

APPENDIX A: PORTABLE DATA SETS CREATED BY THE EXPORT COMMAND

When a VSAM cluster or alternate index is exported via the Access Method Services EXPORT command, catalog information needed to define the VSAM data set plus all the records from the data component are written to a nonVSAM set called the portable data set. The following list shows the attributes of the portable data set.

Attribute of Portable Data Sets

Attribute	Value
LRECL	The larger of: (a) Maximum VSAM data set record size +4 (b) For portable data sets created on systems without **VS2.03.807:** 264 (for key sequenced and entry sequenced data sets) or 268 (for relative record data sets). For portable data sets created on systems with **VS2.03.807**: 272 (for nonRRDSs) or 276 (for RRDSs).
BLKSIZE	As specified by the user. The default is 2048.
RECFM	VBS
DSORG	PS
DEVTYPE	Tape or disk

The portable data set contains two *major* types of records: control records and data records. Control records contain one of two types of information: a time stamp or a dictionary. Data records also contain one of two types of information: a catalog work area or a data record from the data component of the cluster or alternate index exported. Figure 32 shows the general layout of control records and data records in the portable data set. The types of records and the types of information within those records are explained in this appendix.

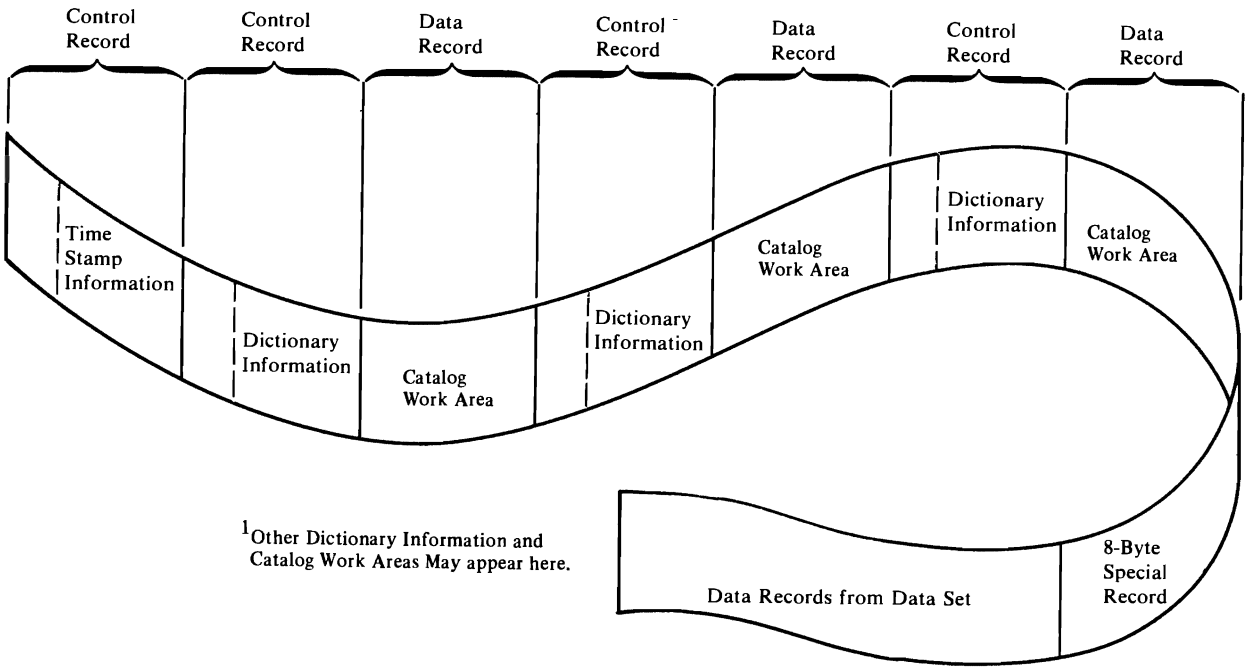


Figure 32. Layout of Control Records and Data Records in the Portable Data Set

Control Records

Control records all have the same general format as shown in Figure 33. The first four bytes of each control record contain header information. The next four bytes contain associated data. The remainder of the record contains the time stamp or dictionary information.

Control Record Containing Time Stamp Information

The first record on every portable data set is a control record that contains time stamp information, as well as other fields. The format of this record is shown in Figure 33.

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this control record contains time stamp information. There is no associated data, and those four bytes are reserved.

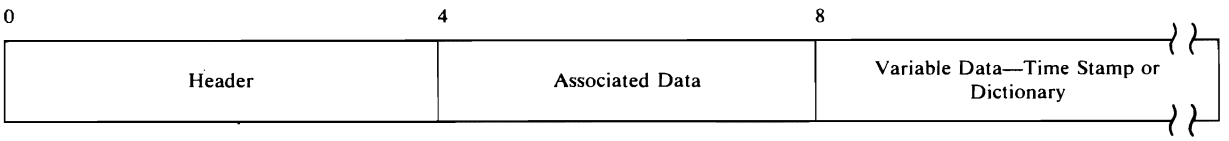


Figure 33. General Format of Control Records

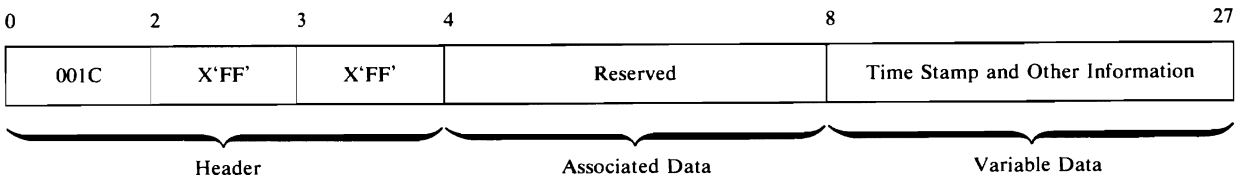


Figure 34. Control Record Containing Time Stamp Information

The format of the time stamp information is:

Displacement[1]	Description
8 (8)	Number of cluster or alternate index components and paths being exported.
9 (9)	Flags: Bit 0: 1 indicates a unique data set; 0 indicates a non-unique data set Bit 1: 1 indicates an inhibited target; 0 indicates a non-inhibited target Bit 2: 1 indicates path associations are present; 0 indicates no paths are present. Bit 3: If bit 2 is 1: 1 indicates that the base object has both data and index components; 0 indicates that the base object has only a data component.
10 (A)	Access Method Services release number in EBCDIC
11 (B)	Reserved
12 (C)	Time of EXPORT in EBCDIC, in the form hh.mm.ss, where hh is the number of hours, mm the number of minutes, and ss the number of seconds.
20 (14)	Date of EXPORT in EBCDIC, in the form mm/dd/yy, where mm is the month in digits, dd the day, and yy the year.

[1] The displacement is from the beginning of the control record.

Control Records Containing Dictionary Information

A control record containing dictionary information is written for the cluster or alternate index being exported and for each component within that cluster or alternate index. In addition, one control record is written for each path association of the object being exported. These records in essence describe the data record containing the catalog work area which follows. The format of control records containing dictionary information is shown in Figure 35.

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this record contains dictionary information and the type of component that the associated catalog work area information describes. The type of component is indicated by 'C' for cluster, 'D' for data, 'I' for index, 'G' for alternate index, or 'R' for path.

The associated data portion of the control record contains the length of the associated catalog work area (two bytes) and the number of records into which the associated catalog work area is broken (2 bytes).

The variable data portion of the control record contains the dictionary information. This portion of the control record begins with a four-byte field

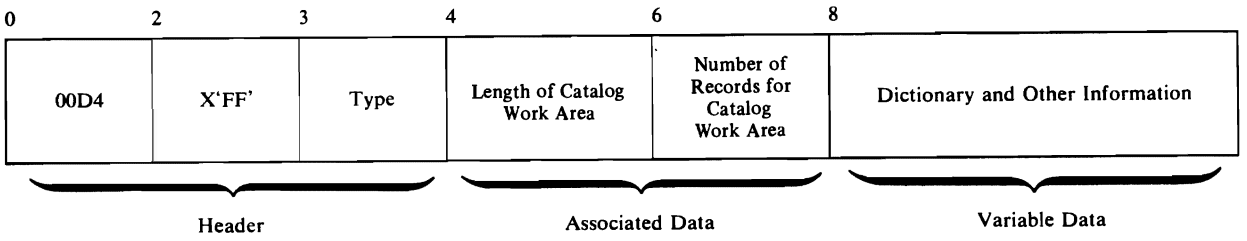


Figure 35. Control Record Containing Dictionary Information

that contains the number of entries in the dictionary. The entries themselves follow. Each entry consists of a pair of four-byte fields. The first four bytes contain the length of the associated catalog field in the catalog work area. (Remember, the catalog work area information is in a data record immediately following one of these control records.) The second four bytes contain the displacement of that field within the associated data record. If an associated catalog field contains no information, both four-byte fields in the dictionary entry contain zeros. The dictionary entries always point to the associated fields in the order shown in the following list.

Order of Associated Catalog Fields

Order	Associated Field in Catalog Work area	Description
1	ENTYPE	Component type.
2	ENTNAME	Component name.
3	DSATTR	Data set attributes.
4	OWNERID	Data set owner.
5	DSETCRDT	Data set creation date.
6	DSETEXDT	Data set expiration date.
7	BUFSIZE	Minimum buffer size.
8	LRECL	Logical record size.
9	SPACEPARM	Primary and secondary space.
10	PASSWORD	Four eight-character passwords.
11	PASSPRMT	Password prompting code name.
12	PASSATMP	Maximum number of attempts for password.
13	USVRMDUL	User security verification module.
14	USERAREC	User authorization record.
15	LOKEYV	Low key on volume.
16	HIKEYV	High key on volume.
17	VOLSER	Volume serial numbers.
18	AMDSBCAT	AMDSB, from which the next nine fields are taken.
19	AMDATTR	Attributes.
20	AMDRKP	Relative key position.
21	AMDKEYLN	Key length.
22	AMDCINV	Control interval size.
23	AMDLRECL	Maximum record size.
24	AMDPCTCA	Percent of free control intervals in control area.

Order of Associated Catalog Fields (continued)

Order	Associated Field in Catalog Work Area	Description
25	AMDPCTCI	Percent of free bytes in control intervals.
26	AMDATTR3	Attributes.
27	AMDAXRKP	Position of alternate key in base cluster record.
28	EXCPEXIT	Exception exit.
29	RGATTR	Alternate index or path attributes.
30	RELATE | PATHENTRY	Alternate index related name or pathentry name.
31	PASSREL	Master password of pathentry component.
32	SECFLAGS	RACF security field (**VS2.03.807**).

Data Records

Data records contain one of two types of information: the catalog work area or data records from the data component.

Data Records Containing Catalog Work Area

Following each control record that contains dictionary information there is a data record that contains the catalog work area for a given component. The format of these records is shown in Figure 36.

The first two bytes of each record contain the total possible length of the catalog work area. The next two bytes contain the length of the work area used for this component. Following these first four bytes are the fields from the catalog work area. The order of these fields is basically as described in the preceding topic. If there is no information for one of the fields, the field is completely omitted.

Figure 37 shows the relationship of the dictionary and catalog work area information.

Data Records Containing Data Records From the Data Component

Following all of the control records and data records that contain dictionary information is a special record which marks the beginning of the data records from the data component. This special record is eight bytes in length. The record always has the format shown in Figure 37.

Following this special record are all of the data records from the data component being exported.

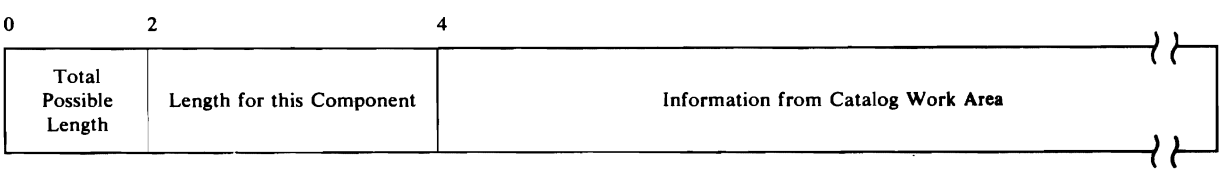


Figure 36. Data Record Containing Catalog Work Area

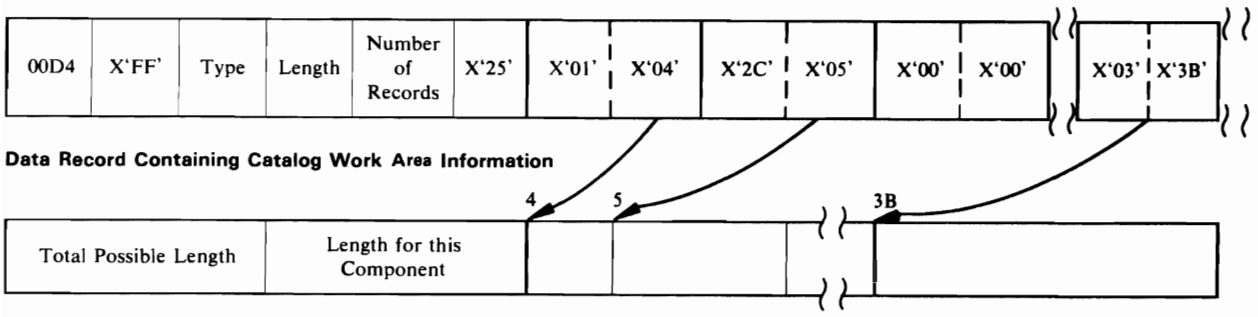


Figure 37. Relationship of Dictionary and Catalog Work Area Information

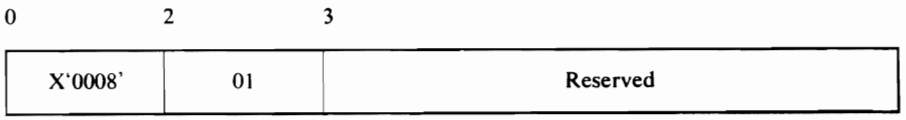


Figure 38. Special Record at Beginning of Data Records from the Data Component

APPENDIX B: PORTABLE DATA SETS CREATED BY THE EXPORTRA COMMAND

When the EXPORTRA command of Access Method Services executes, it produces a portable data set which contains catalog information obtained from a CRA (Catalog Recovery Area) and data records for VSAM clusters and alternate indexes, and also catalog information for user catalog pointers. In addition, portable data sets created by EXPORTRA (referred to as recovery portable data sets in this appendix) on OS/VS systems may contain catalog information for nonVSAM, alias, and generation data group (GDG) base objects. The following list shows the attributes of the portable data set.

Attribute	Value
LRECL	The larger of: (a) Maximum VSAM data set record size + 8 (b) For portable data sets created on systems without **VS2.03.807:** 268 (if the portable data set does not contain any VSAM relative record data sets) or 272 (if the portable data set contains any VSAM relative record data sets). For portable data sets created on systems with **VS2.03.807:** 276 (if the portable data set does not contain any VSAM relative record data sets) or 280 (if the portable data set contains any VSAM relative record data sets).
BLKSIZE	As specified by the user (the default is 2048)
RECFM	VBS
DSORG	PS
DEVTYPE	(Tape or disk)

Each record of the recovery portable data set has a special 4-byte header added that precedes the record itself. Information for unrelated objects on the recovery portable data set is separated by one or more software ends of file. These ends of file are special records that consist only of the 4-byte header. Only Figure 39 indicates that this particular type of header precedes each data record; the other figures do not show it.

The recovery portable data set contains two *major* types of records: control records and data records. Control records contain one of two types of information: a time stamp or a dictionary. Data records also contain one of two types of information: a catalog work area or a data record from the data component of the cluster exported. Figure 39 shows the general layout of control records and data records in the recovery portable data set. The types of records and the types of information within those records are explained in this appendix.

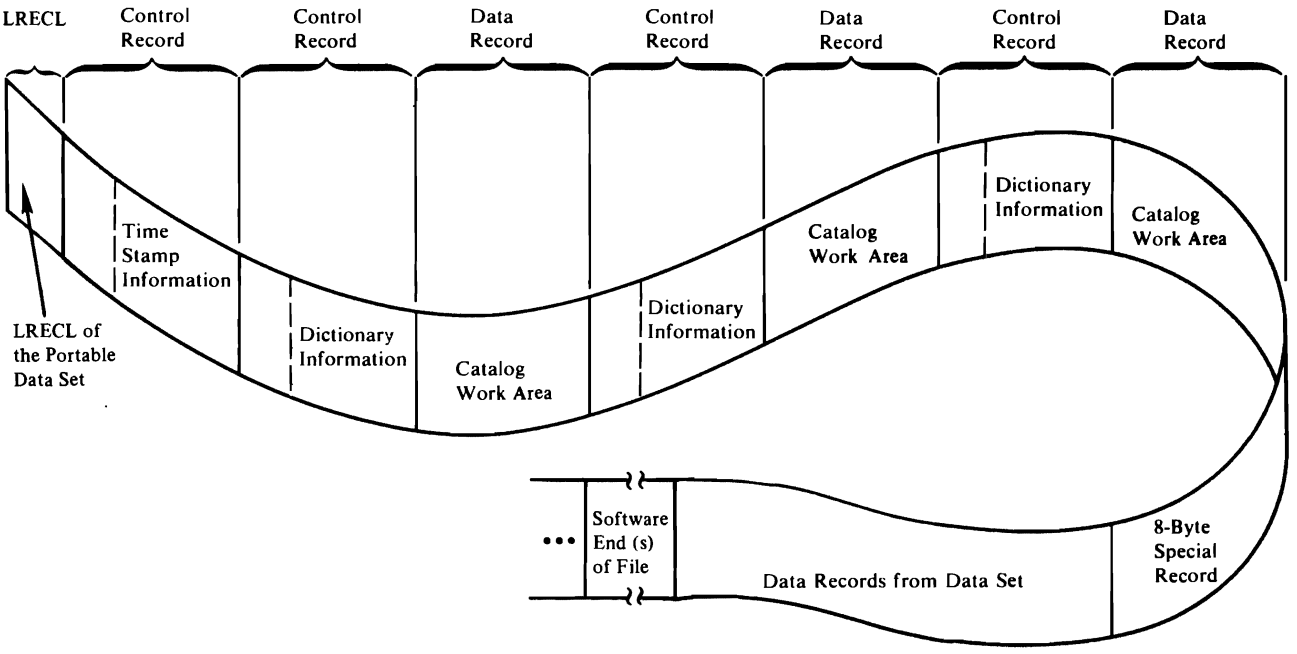


Figure 39. Layout of Control Records and Data Records in the Recovery Portable Data Set

Control Records

Control records all have the same general format as shown in Figure 40. The first four bytes of each control record contain header information. The next four bytes contain associated data. The remainder of the record contains the time stamp, dictionary information, or logical record length.

Control Record Containing the Logical Record Length

The first record of every recovery portable data set is a control record containing the logical record length of the portable data set itself. The format of this record is shown in Figure 41.

Control Record Containing Time Stamp Information

The first record for each item on the recovery portable data set is a control record that contains time stamp information, as well as other fields. The format of this record is shown in Figure 42.

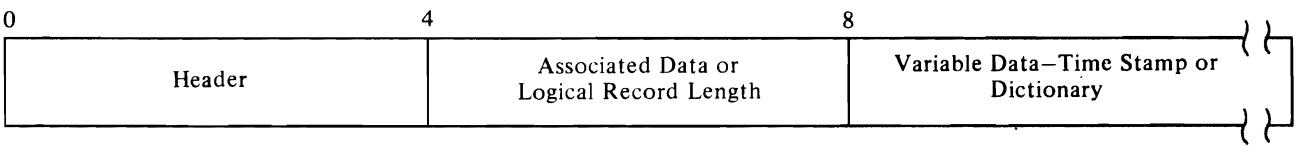


Figure 40. General Format of Control Records

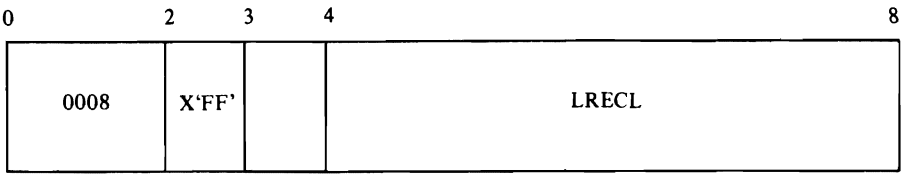


Figure 41. Control Record Containing the Logical Record Length

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this control record contains time stamp information. There is no associated data, and those four bytes are reserved.

The format of the time stamp information is:

Displacement[1]	Description
8(8)	The maximum number of components associated with this item.
9(9)	Flags:
	Bit — *Meaning When Set*
	0 — 1 indicates a unique data set 0 indicates a nonunique data set
	1 — 1 indicates an inhibited target 0 indicates a noninhibited target
	2 — 1 indicates path associations are present. 0 indicates no paths are present.
	3 — If bit 2 is 1: 1 indicates that the base object has both data and index components. 0 indicates that the base object has only a data component.
	4 — 1 always 1 for a recovery portable data set.
	5 — 1 indicates a nonVSAM object. 0 indicates an object other than a nonVSAM.
	6 — 1 indicates a GDG base object. 0 indicates an object other than a GDG base.
	7 — 1 indicates a user catalog pointer. 0 indicates a nonuser catalog pointer.
10(A)	Access Method Services release number in EBCDIC
11(B)	Reserved
12(C)	Time of export in EBCDIC, in the form hh.mm.ss, where hh is the number of hours, mm the number of minutes, and ss the number of seconds.
20(14)	Date of export in EBCDIC, in the form mm/dd/yy, where mm is the month in digits, dd the day, and yy the year.

[1] The displacement is from the beginning of the control record.

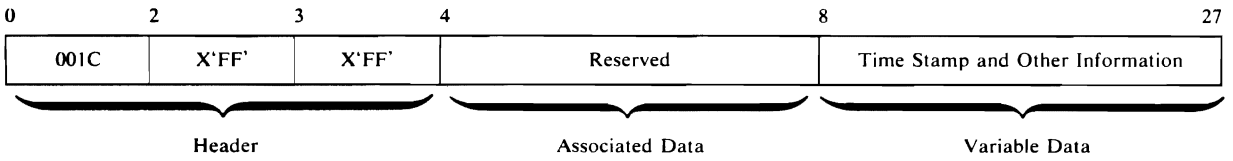


Figure 42. Control Record Containing Time Stamp Information

Control Records Containing Dictionary Information

A control record containing dictionary information is written for each object being exported and for each component associated with that object. These records in essence describe the data record containing the catalog work area which follows. The general format of control records containing dictionary information is shown in Figure 43.

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this record contains dictionary information and the type of component that the associated catalog work area information describes. The type of component is indicated by 'C' for cluster, 'D' for data, 'I' for index, 'G' for alternate index, 'R' for path, 'A' for nonVSAM, 'B' for GDG base, 'X' for alias, or 'U' for user catalog pointer.

The associated data portion of the control record contains the length of the associated catalog work area (2 bytes) and the number of records into which the associated catalog work area is broken (2 bytes).

The variable data portion of the control record contains the dictionary information. This portion of the control record begins with a four-byte field that contains the number of entries in the dictionary. The entries themselves follow. Each entry consists of a pair of four-byte fields. The first four bytes contain the length of the associated catalog field in the catalog work area. (Remember, the catalog work area information is in a data record immediately following one of these control records.) The second four bytes contain the displacement of that field within the associated data record. If an associated catalog field contains no information, both four-byte fields in the dictionary entry contain zeros.

The number of dictionary entries and their order depends upon the type of object being described. Dictionary formats are described for each possible kind of item in the following list.

Order of Associated Catalog Fields

Order VSAM Components	Associated Field in Catalog Work Area	Description
1	ENTYPE	Component type.
2	ENTNAME	Component name.
3	DSATTR	Data set attributes.
4	OWNERID	Data set owner.
5	DSETCRDT	Data set creation date.
6	DSETEXDT	Data set expiration date.
7	BUFSIZE	Minimum buffer size.

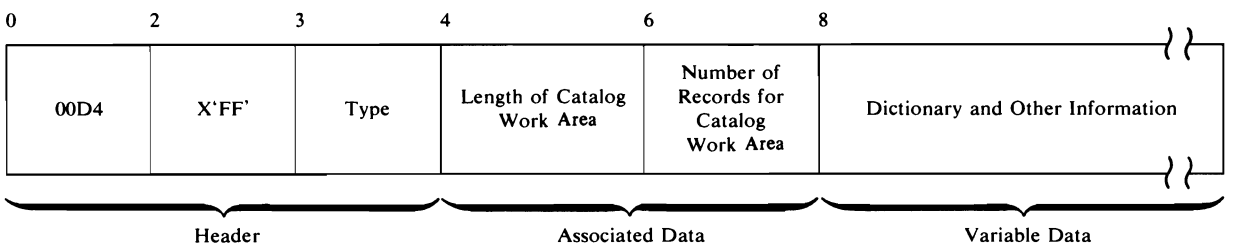


Figure 43. Control Record Containing Dictionary Information

Order of Associated Catalog Fields

Order VSAM Components	Associated Field in Catalog Work Area	Description
8	LRECL	Logical record size.
9	SPACEPARM	Primary and secondary space.
10	PASSWORD	Four eight-character passwords.
11	PASSPRMT	Password prompting code name.
12	PASSATMP	Maximum number of attempts for password.
13	USVRMDUL	User security verification module.
14	USERAREC	User authorization record.
15	LOKEYV	Low key on volume.
16	HIKEYV	High key on volume.
17	VOLSER	Volume serial numbers.
18	AMDSBCAT	AMDSB from which the next 9 fields are taken.
19	AMDATTR	Attributes.
20	AMDRKP	Relative key position.
21	AMDKEYLN	Key length.
22	AMDCINV	Control interval size.
23	AMDLRECL	Maximum record size.
24	AMDPCTCA	Percent of free control intervals in control area.
25	AMDPCTCI	Percent of free bytes in control intervals.
26	AMDATTR3	Attributes
27	AMDAXRKP	Position of alternate index key in base cluster record.
28	EXCPEXIT	Exception exit.
29	RGATTR	Alternate index or path attributes.
30	RELATE | PATHENTRY	Alternate index related name or path entry name.
31	PASSREL	Master password of path entry component.
32	SECFLAGS	RACF Security fields (VS2.03.807)
NonVSAM		
1	ENTYPE	Entry type.
2	ENTNAME	Entry name.
3	VOLSER	Volume serial numbers.
4	DEVTYP	Device types.
5	FILESEQ	File sequence numbers.
6	OWNERID	Data set owner.
7	DSETCRDT	Data set creation date.
8	DSETEXDT	Data set expiration date.
User Catalog Pointers		
1	ENTYPE	Entry type.
2	ENTNAME	Entry name.
3	VOLSER	Volume serial numbers.
4	DEVTYP	Device types.

Order of Associated Catalog Fields

Order VSAM Components	Associated Field in Catalog Work Area	Description
Aliases		
1	ENTYPE	Entry type.
2	ENTNAME	Entry name.
GDG Bases		
1	ENTYPE	Entry type.
2	ENTNAME	Entry name.
3	GDGLIMIT	GDG limit value.
4	GDGATTR	GDG attributes.
5	OWNERID	Data set owner.
6	DSETCRDT	Data set creation date.
7	DSETEXDT	Data set expiration date.

Data Records

Data records contain one of two types of information: the catalog work area or data records from the data component of a VSAM cluster.

Data Records Containing Catalog Work Area

Following each control record that contains dictionary information there is a data record that contains the catalog work area for a given component. The format of these records is shown in Figure 44.

The first two bytes of each record contain the total possible length of the catalog work area. The next two bytes contain the length of the work area used for this component. Following these first four bytes are the fields from the catalog work area. The order of these fields is basically as described in the preceding topic. If there is no information for one of the fields, the field is completely omitted.

Figure 45 shows the relationship of the dictionary and catalog work area information.

Data Records Containing Data Records from the Data Component

For a VSAM cluster or alternate index, following all of the control records and data records that contain dictionary information is a special record which marks the beginning of the data records from the data component. This special record is eight bytes in length. The record always has the format shown in Figure 46.

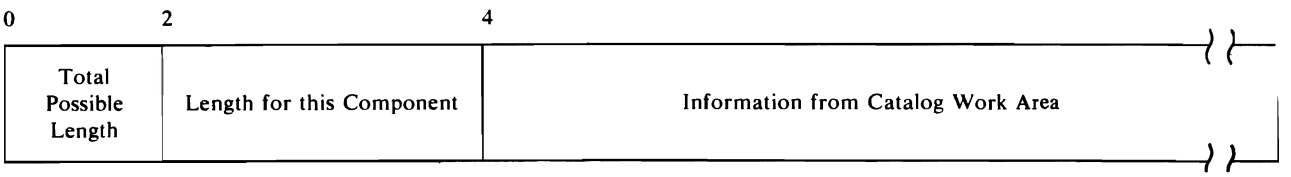


Figure 44. Data Record Containing Catalog Work Area

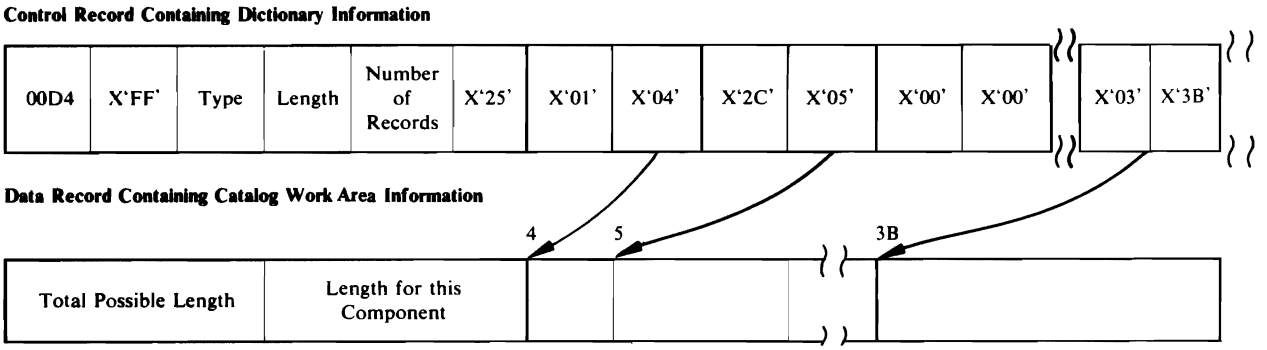


Figure 45. Relationship of Dictionary and Catalog Work Area Information

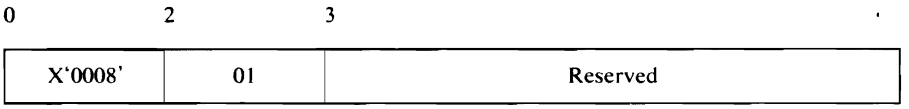


Figure 46. Special Record at Beginning of Data Records from the Data Component

Following this special record are all of the data records from the data component being exported.

Associated Objects for User Catalog Pointers, NonVSAMs, and GDGs

The aliases of a user catalog pointer or a nonVSAM are exported as associated objects. Similarly, the nonVSAMs that belong to a GDG base are exported as associated objects of the GDG; these nonVSAMs may, in turn, have aliases. An item and its associated objects are preceded by one time stamp control record and followed by one software end of file.

INDEX

For additional information about any subject listed in this index, refer to the publications that are listed under the same subject in the *OS/VS2 Master Index,* GC28-0693.

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

1,2,3

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2 Access Method
Services Logic
SY35-0010-2

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name and address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SY35-0010-3

Fold and Staple

First Class Permit
Number 6090
San Jose, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Staple

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2 Access Method
Services Logic
SY35-0010-2

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name and address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Fold and Staple

First Class Permit
Number 6090
San Jose, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Staple

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)