LY24-5195-1
File No. S370-30

Program Product

VSE/VSAM Access Method Services Logic

Program Number 5746-AM2

Component 5745-SC-AMS

Release 2

IBM

Second Edition (December 1979)

This edition, LY24-5195-1, is a major revision of LY24-5195-0. It applies to Release 2 of IBM Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM) Program Product 5746-AM2, and to subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Amendments

For a list of changes, see page iii.

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

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

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

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

Summary of Amendments for VSE/VSAM Access Method Services Logic

Summary of Amendments for LY24-5195-1 Release 2

LY24-5195-1 contains information about the following items:

- Additional space classes
- CANCEL command
- Dedicated VSAM volume
- Default models
- Default volumes
- Dynamic files
- File disposition parameters
- JCL simplification
- Partition and processor independence
- Interface between Access Method Services and the VSE/VSAM Space Management for SAM Feature.

Additions, deletions, and corrections are included for the new items. Various editorial changes are also included to improve the usefulness of this book.

Preface

This book describes the internal logic of the routines of Access Method Services and provides diagnostic information. This information is directed to maintenance 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.

This volume is one of three logic manuals that describe the internal functioning of VSE/VSAM. The other two volumes are:

- *VSE/VSAM VSAM Logic, Volume 1: Catalog Management, Open/Close, DADSM, IIP, Control Block Manipulation*, LY24-5191.
- *VSE/VSAM VSAM Logic, Volume 2: Record Management*, LY24-5192

The interface between Access Method Services and the VSE/VSAM Space Management for SAM Feature is described in *VSE/VSAM Space Management for SAM Feature Logic*, LY24-5204.

You should be familiar with general programming techniques and VSE/VSAM concepts and use before reading this book. If you are unfamiliar with these concepts, read:

- *VSE/VSAM General Information*, GC24-5143.
- *Using VSE/VSAM Commands and Macros*, SC24-5144, which describes the general syntax of the Access Method Services language, the commands of this processor, VSAM macros, and how they are used.

Another book that may be helpful to you is:

- *VSE/Advanced Functions Serviceability Aids and Debugging Procedures*, SC33-6099, which describes how to analyze a main storage dump from VSE.

This book is divided into six chapters:

- "Chapter 1: Introduction" describes the design philosophy of this processor, and defines terms used later in the book.
- "Chapter 2: 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.
- "Chapter 3: Program Organization" shows how the processor is packaged into load modules. Relationships between the Access Method Services processor and the operating system are given.
- "Chapter 4: Microfiche Directory" relates the information in this book to the listings found on microfiche.
- "Chapter 5: Data Areas" describes the control blocks and other data areas that are internal to this processor.
- "Chapter 6: Diagnostic Aids" shows how to analyze a dump of the processor and find specific modules and data areas.

Contents

Chapter 1: Introduction 1-1
Requirements 1-1
The Access Method Services Processor 1-1
Naming Conventions 1-7
Character Code Dependencies 1-8

Chapter 2: Method of Operation 2-1

Chapter 3: Program Organization 3-1
Overall Organization 3-1
System Macros and Services Used by Access Method Services 3-3
Services Provided for Processor Modules 3-5
Processor Invocation 3-7
Processor Condition Codes 3-9
User I/O Routines 3-9
Overall Control Flow 3-9

Chapter 4: Microfiche Directory 4-1

Chapter 5: Data Areas 5-1
Block List - BLKLIST 5-2
Buffer Pool Control Block - BUFS 5-2
Buffer Pool Control Block Description 5-2
Command Descriptor 5-2
Verb Data Area 5-3
Positional Parameter Appendage 5-3
Default Parameter Appendage 5-4
Needed Parameters Appendage 5-4
Incompatible Parameters Appendage 5-4
Parameter Data Area 5-5
No Constant Appendage 5-5
Constant Appendage 5-6
Default Data Appendage 5-6
ID Appendage 5-6
Keyword Appendage 5-7
Conflicting Parameters Appendage 5-7
Necessary Parameters Appendage 5-7
Prompt Appendage 5-7
Subparameter Appendage 5-8
Command Descriptor Phase Table - IDCRILT 5-8
CRA Access Parameter List 5-8
Access Method Services/Catalog Communication Table (ACC) Description 5-8
CRA Access Translate Table (CTT) Description 5-9
CRA Volume Timestamp Table (VTT) Description 5-9
Dump List 5-9
Individual Field Entry 5-9
Array Header Entry 5-10
Dump List Terminator Entry 5-10
Dynamic Data LIST - DARGLIST 5-10
ERROR Conversion Table (ERCNVTAB) 5-11
Field Management Parameter List — FMPL 5-12
Field Management Parameter List Description 5-12
Field Management Field List (FMFL) Description 5-12
Format List - FMTLIST 5-13
Spacing 5-13
Insert Data 5-13
Default Text 5-14
Block Format 5-15
Replication 5-15
Static Text 5-15
Function Data Table - FDT 5-16
ALTER FDT 5-20

BLDINDEX FDT 5-21
CANCEL FDT 5-21
DEFINE FDT 5-22
DELETE FDT 5-31
EXPORT FDT 5-32
EXPORTRA FDT 5-33
IMPORT FDT 5-34
IMPORTRA FDT 5-35
LISTCAT FDT 5-36
LISTCRA FDT 5-36
PARM FDT 5-37
PRINT FDT 5-37
REPRO FDT 5-38
RESETCAT FDT 5-40
VERIFY FDT 5-40
Global Data Table - GDT 5-41
Global Data Table Description 5-41
Input Parameter Table - IPT 5-43
Input Parameter Table Description 5-43
I/O Adapter Historical Data Area - IODATA 5-44
I/O Adapter Historical Area Description 5-44
Input/Output Communications Structure - IOCSTR 5-44
Input/Output Communications Structure Description 5-44
IOCSTR Extension - IOCSEX 5-46
IOCSTR Extension Description 5-46
Inter-Module Trace Table 5-47
Inter-Module Trace Table Description 5-47
Intra-Module Trace Table 5-47
Intra-Module Trace Table Description 5-48
Modal Verb and Keyword Symbol Table - IDCRIKT 5-48
Modal Verb and Keyword Symbol Table Description 5-48
Open Argument List - OPNAGL 5-48
Open Argument List Description 5-49
Open Close Address Array - OCARRAY 5-50
Open Close Address Array Description 5-50
Phase Table 5-51
Phase Table Description 5-51
Positioning Argument List - OPRARG 5-51
Positioning Argument List Description 5-51
Print Control Argument List - PCARG 5-51
Print Control Argument List Description 5-51
Print Control Table - PCT 5-52
Print Control Table Description 5-52
Reader/Interpreter Communication Area - COMMAREA 5-54
Reader/Interpreter Communication Area Description 5-54
Reader/Interpreter Historical Area - HDAREA 5-55
Reader/Interpreter Historical Area Description 5-55
Scope Structure for UENQ - ENQSCOPE 5-55
Scope Structure for UENQ Description 5-55
System Adapter Historical Area - SAHIST 5-56
System Adapter Historical Area Description 5-56
TEST Option Data Area 5-56
TEST Option Data Area Description 5-56
Text Structure 5-57
Text Structure Description 5-58
Text Entry Description 5-58
UGPOOL Area 5-58
UGPOOL Area Description 5-58
UGSPACE Area 5-59
UGSPACE Area Description 5-59
UIOINFO - Option Byte and Return Area 5-59
UIOINFO Option Byte Description 5-59
UIOINFO Return Area Description 5-59
UREST Arguments 5-60
PCRST - Change Subtitle Lines 5-60
PCRLWS - Change Line Width 5-60
PCRPDS - Change Page Depth 5-60

PCRFTS - Change Footing Lines 5-61
PCRDSCS - Change Default Spacing Character 5-61
PCRPCS - Change Translate Table 5-61
PCRINP - Change Initial Page Number 5-61

Chapter 6: Diagnostic Aids 6-1
Trace Tables 6-1
Inter-Module Trace Table 6-1
Intra-Module Trace Table 6-1
Dump Points 6-2
Dumping Selected Areas of Virutal Storage 6-3
Test Option 6-3
TEST Keyword 6-3
How to Use the Test Option 6-4
Trace and Dump Points to Module Cross Reference 6-5
Module to Dump Points Cross Reference 6-26
ABORT Codes 6-36
Reading a Dump 6-37
How to Find Processor Phases 6-38
How to Find the Module and Registers at Time of the Dump 6-38
How to Find the GDT 6-39
How to Find Save Areas 6-39
How to Find the Trace Tables 6-44
How to Find the FDT 6-45
How to Find Automatic Storage Areas 6-45
How to Find Dynamic Storage Areas 6-46
UGPOOL ID List 6-48
Sample Dump 6-52
Debugging a Catalog Problem 6-52
Obtaining a Dump for a Catalog Problem 6-52
How to Find Catalog Management Argument Lists 6-58
Debugging a Formatting Problem 6-59
Obtaining a Dump for a Text Processor Problem 6-70
How to Find Text Processor Argument List 6-71
Debugging an I/O Problem 6-72
Obtaining a Dump for an I/O Problem 6-72
How to Find I/O Argument Lists 6-73
Open Argument Lists 6-73
UGET and UPUT Argument Lists 6-73
Messages 6-78

Appendix A: Portable Data Sets Created by the EXPORT Command A-1
Control Records A-2
Control Record Containing Timestamp Information A-2
Control Records Containing Dictionary Information A-3
Data Records A-4
Data Records Containing Catalog Work Area A-5
Data Records Containing Data Records From the Data Component A-5

Appendix B: Portable Data Sets Created by the EXPORTRA Command B-1
Control Records B-1
Control Record Containing the Logical Record Length B-1
Control Record Containing Timestamp Information B-1
Control Records Containing Dictionary Information B-3
Data Records B-5
Data Records Containing Catalog Work Area B-5
Data Records Containing Data Records From the Data Component B-6
Associated Objects for User Catalog Pointers, NonVSAMs, and GDGs B-6

Index I-1

Illustrations

Figures

Figure 1-1. The Structure of the Access Method Services Processor 1-2
Figure 1-3. Initialization of Access Method Services 1-3
Figure 1-4. Reading and Parsing a Command 1-4
Figure 1-6. Performing a Function 1-6
Figure 3-1. Argument List for Processor Invocation 3-8
Figure 3-2. Arguments Passed to and from User I/O Routine 3-10
Figure 3-3. Flow of Control Through Main Functions 3-11
Figure 3-4. Flow of Control Through Services 3-12
Figure 5-1. IMPORT FDT Mapping 5-19
Figure 6-1. Example of Test Option Output 6-5
Figure 6-2. Sample Dump 6-40
Figure 6-3. How to Find the GDT 6-45
Figure 6-4. Format of AUTOTBL 6-47
Figure 6-5. Example of an Automatic Storage Area 6-47
Figure 6-6. UGPOOL Area Chain 6-48
Figure 6-7. How to Find the CTGPL 6-59
Figure 6-8. Catalog Argument Lists in Storage Area of DEFINE FSR 6-60
Figure 6-9. Text Processor Format Structure Queue 6-71
Figure 6-10. Text Processor Print Buffer 6-72
Figure 6-11. IOCSTR Chain 6-74
Figure 6-12. I/O Control Blocks Before OPEN 6-75
Figure 6-13. Input to UPUT Macro 6-76
Figure 6-14. Output from UGET Macro 6-77
Figure A-1. Layout of Control Records and Data Records in the Portable Data Set A-1
Figure A-2. General Format of Control Records A-2
Figure A-3. Control Record Containing Timestamp Information A-2
Figure A-4. Control Record Containing Dictionary Information A-3
Figure A-5. Data Record Containing Catalog Work Area A-5
Figure A-6. Relationship of Dictionary and Catalog Work Area Information A-5
Figure A-7. Special Record at Beginning of Data Records from the Data Component A-5
Figure B-1. Layout of Control Records and Data Records in the Recovery Portable Data Set B-2
Figure B-2. General Format of Control Records B-2
Figure B-3. Control Record Containing the Logical Record Length B-2
Figure B-4. Control Record Containing Timestamp Information B-2
Figure B-5. Control Record Containing Dictionary Information B-3
Figure B-6. Data Record Containing Catalog Work Area B-5
Figure B-7. Relationship of Dictionary and Catalog Work Area Information B-6
Figure B-8. Special Record at Beginning of Data Records from the Data Component B-6

Diagrams

Access Method Services Visual Table of Contents 2-3
Access Method Services Overview 2-4

Initialization Visual Table of Contents 2-7
Diagram 1.0 Access Method Services Initialization Overview 2-8
Diagram 1.1 System Adapter Initialization 2-10
Diagram 1.2 I/O Adapter Initialization - UIOINIT Macro 2-12

Reader/Interpreter Visual Table of Contents 2-15
Diagram 2.0 Reader/Interpreter Overview 2-16
Diagram 2.1 Reader/Interpreter Initialization 2-18
Diagram 2.2 Reader/Interpreter Get Next Command 2-20
Diagram 2.2.1 Reader/Interpreter IF-THEN Modal Command 2-22
Diagram 2.2.2 Reader/Interpreter ELSE Modal Command 2-24
Diagram 2.2.3 Reader/Interpreter SET Modal Command 2-26
Diagram 2.2.4 Reader/Interpreter DO Modal Command 2-28
Diagram 2.2.5 Reader/Interpreter END Modal Command 2-30
Diagram 2.3 Reader/Interpreter Prepare To Scan Command 2-32
Diagram 2.4 Reader/Interpreter Scan Command 2-34

Diagram 2.4.1 Reader/Interpreter Syntax Check Parameter 2-36
Diagram 2.4.2 Reader/Interpreter Build FDT 2-38
Diagram 2.5 Reader/Interpreter Termination 2-40

Function Support Routine (FSR) Visual Table of Contents 2-43
Diagram 3.1 ALTER FSR 2-44
Diagram 3.2 DEFINE FSR 2-48
Diagram 3.2.1 DEFINE FSR - DEFINE MASTERCATALOG 2-50
Diagram 3.2.2 DEFINE FSR - DEFINE USERCATALOG 2-56
Diagram 3.2.3 DEFINE FSR - DEFINE NONVSAM 2-60
Diagram 3.2.4 DEFINE FSR - DEFINE SPACE 2-62
Diagram 3.2.5 DEFINE FSR - DEFINE CLUSTER 2-64
Diagram 3.2.6 DEFINE FSR - DEFINE ALTERNATEINDEX 2-68
Diagram 3.2.7 DEFINE FSR - DEFINE PATH 2-72
Diagram 3.3 DELETE FSR 2-74
Diagram 3.4 EXPORT FSR 2-76
Diagram 3.4.1 EXPORT FSR - CLUSTER or ALTERNATEINDEX 2-78
Diagram 3.5 IMPORT FSR 2-82
Diagram 3.5.1 IMPORT FSR - CLUSTER or ALTERNATEINDEX 2-84
Diagram 3.6 LISTCAT FSR 2-88
Diagram 3.6.1 LISTCAT FSR - Gets Information 2-92
Diagram 3.7 PARM FSR 2-94
Diagram 3.8 PRINT FSR 2-96
Diagram 3.9 REPRO FSR 2-98
Diagram 3.9.1 REPRO FSR - Catalog Reload 2-100
Diagram 3.10 VERIFY FSR 2-102
Diagram 3.11 BLDINDEX FSR 2-104
Diagram 3.11.1 BLDINDEX FSR - Get Information and Verify 2-108
Diagram 3.11.2 BLDINDEX FSR - Obtain Resources and Sort Initialization 2-110
Diagram 3.11.3 BLDINDEX FSR - Sort-Merge and Build Alternate Index 2-112
Diagram 3.12 LISTCRA FSR 2-116
Diagram 3.12.1 LISTCRA FSR - Process CRA 2-118
Diagram 3.13 EXPORTRA FSR 2-120
Diagram 3.13.1 EXPORTRA FSR - Field Management 2-122
Diagram 3.13.2 EXPORTRA FSR - EXPORTRA Driver 2-124
Diagram 3.13.2.1 EXPORTRA FSR - Export VSAM Data Set 2-126
Diagram 3.13.2.2 EXPORTRA FSR - Export NonVSAM 2-128
Diagram 3.14 IMPORTRA FSR 2-130
Diagram 3.14.1 IMPORTRA FSR - Cluster or Alternate Index 2-132
Diagram 3.14.2 IMPORTRA FSR - User Catalog 2-134
Diagram 3.14.3 IMPORTRA FSR - NonVSAM 2-136
Diagram 3.14.4 IMPORTRA FSR - GDG Base 2-138
Diagram 3.15 RESETCAT FSR 2-140
Diagram 3.15.1 RESETCAT FSR - Initialization 2-142
Diagram 3.15.2 RESETCAT FSR - Copy Catalog to Work File 2-144
Diagram 3.15.3 RESETCAT FSR - Merge CRAs to Work File 2-146
Diagram 3.15.3.1 RESETCAT FSR - Common VTOC Handler Functions 2-148
Diagram 3.15.4 RESETCAT FSR - Reassign CI numbers 2-150
Diagram 3.15.5 RESETCAT FSR - Check Associations 2-152
Diagram 3.15.6 RESETCAT FSR - Update the Catalog 2-154
Diagram 3.15.7 RESETCAT FSR - Update the CRA 2-156
Diagram 3.16 CANCEL FSR 2-158

Termination Visual Table of Contents 2-161
Diagram 4.1 Executive Controlled Termination 2-162
Diagram 4.2 Processor Termination 2-164
Diagram 4.2.1 I/O Adapter Termination - UIOTERM Macro 2-166

System Adapter Visual Table of Contents 2-169
Diagram 5.0 System Adapter Overview 2-170
Diagram 5.1.1 UCATLG Macro 2-172
Diagram 5.2.1 UABORT Macro 2-174
Diagram 5.2.2 USNAP Macro 2-176
Diagram 5.3.1 UCALL Macro 2-178
Diagram 5.3.2 ULOAD Macro 2-180
Diagram 5.3.3 UDELETE Macro 2-182
Diagram 5.4.1 UGSPACE Macro 2-184

Diagram 5.4.2 UFSPACE Macro 2-186
Diagram 5.4.3 UGPOOL Macro 2-188
Diagram 5.4.4 UFPOOL Macro 2-190
Diagram 5.4.5 PROLOG Macro 2-192
Diagram 5.4.6 UEPIL Macro 2-194
Diagram 5.5.1 UTIME Macro 2-196
Diagram 5.6.1 ULISTLN Macro 2-198
Diagram 5.6.2 USAVERC Macro 2-200
Diagram 5.7.1 UENQ Macro 2-202
Diagram 5.7.2 UDEQ Macro 2-204

I/O Adapter Visual Table of Contents 2-207
Diagram 6.0 I/O Adapter Overview 2-208
Diagram 6.1 UOPEN Macro 2-210
Diagram 6.1.1 UOPEN Macro - Build IOCSTR 2-212
Diagram 6.1.2 UOPEN Macro - Build Control Blocks 2-216
Diagram 6.1.3 UOPEN Macro - Check Open 2-218
Diagram 6.2 UCLOSE Macro 2-220
Diagram 6.3 UPOSIT Macro 2-222
Diagram 6.4 UGET Macro 2-224
Diagram 6.5 UPUT Macro 2-226
Diagram 6.6 UCOPY Macro 2-230
Diagram 6.7 UVERIFY Macro 2-232
Diagram 6.8 UIOINFO Macro 2-234

Text Processor Visual Table of Contents 2-237
Diagram 7.0 Text Processor Overview 2-238
Diagram 7.1 UESTS Macro 2-240
Diagram 7.2 UESTA Macro 2-242
Diagram 7.3 UREST Macro 2-244
Diagram 7.4 URESET Macro 2-246
Diagram 7.5 UPRINT Macro 2-248
Diagram 7.5.1 UPRINT Macro - CONVERT 2-252
Diagram 7.5.2 UPRINT Macro - PRINT 2-254
Diagram 7.6 UERROR Macro 2-256

Debugging Aids Visual Table of Contents 2-259
Diagram 8.0 Debugging Aids Overview 2-260
Diagram 8.1 UTRACE Macro 2-262
Diagram 8.2 UDUMP Macro 2-264
Diagram 8.2.1 UDUMP Macro - Dump Fields 2-266

Chapter 1: Introduction

Access Method Services is that part of the operating system that performs the utility-like functions required to establish and manage VSAM (Virtual Storage Access Method) data sets. (The terms "data set" and "file" are equivalent. We have used "data set" in this book.) Access Method Services allows you to define, print, delete, or copy VSAM data sets, build alternate indexes, recover data and catalog entries in the event of a catalog failure, convert ISAM or SAM data sets into VSAM data sets, alter or list the entries in a VSAM catalog, and create portable (or backup) copies. 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 phases. Consequently, this publication generally discusses modules. One 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, its concepts, or its data areas. For a discussion of VSAM, see *VSE/VSAM VSAM Logic, Volume 1,* and *VSE/VSAM VSAM Logic, Volume 2.*

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 *Using VSE/VSAM Commands and Macros.*

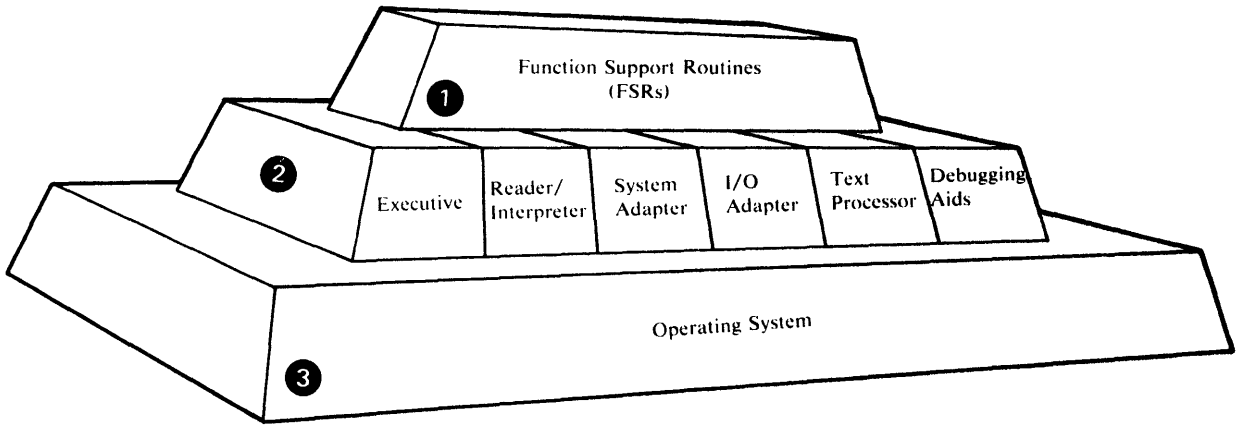
Requirements

This processor requires DOS/VSE as its operating system. The processor executes as a problem program. Virtual storage requirements for the processor are found in *DOS/VSE System Generation.*

The Access Method Services Processor

Figure 1-1 describes the structure of the processor. Figures 1-2 through 1-4 describe in general how the processor functions.

Figure 1-1 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."



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 Processor*, 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 Debugging Aids writes 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 receives 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 1-1. The Structure of the Access Method Services Processor

Following the flow of logic reveals more of the processor than the structure of executable modules. Figure 1-2 and the two which follow show the sequence in which modules execute, important internal tables, and how non-executable modules are used.

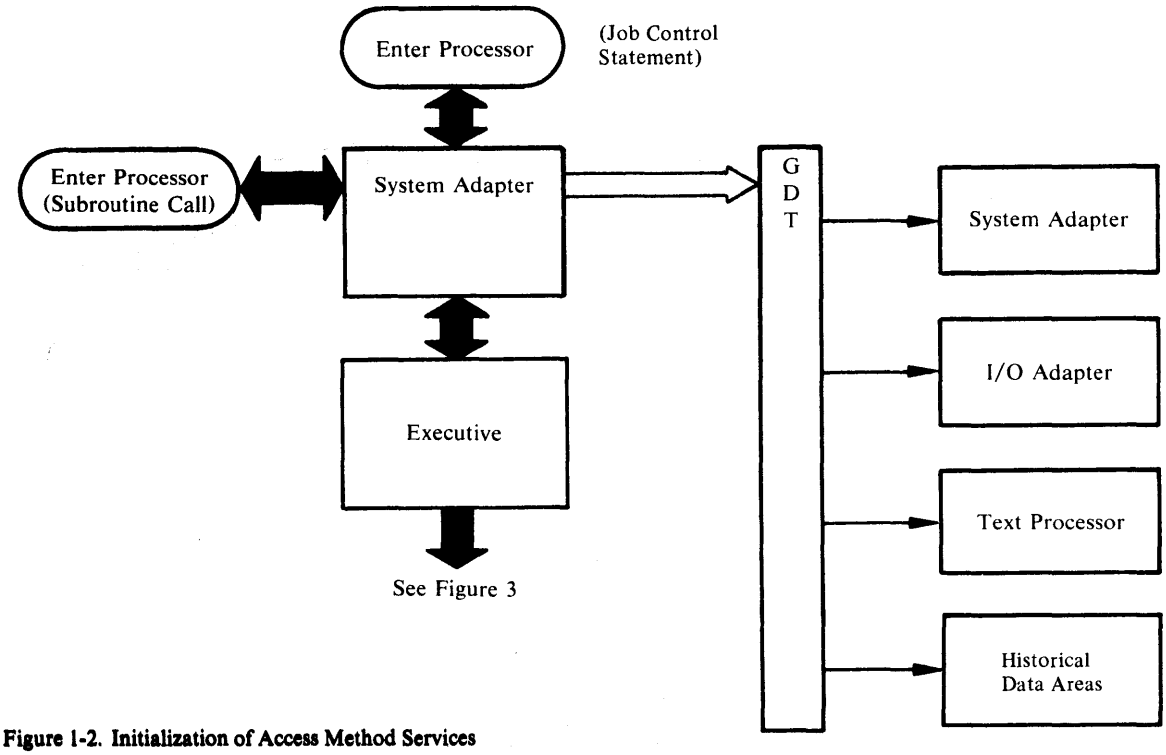


Figure 1-2. Initialization of Access Method Services

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 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 1-3 shows this transfer of control, and details the parsing operation of the processor.

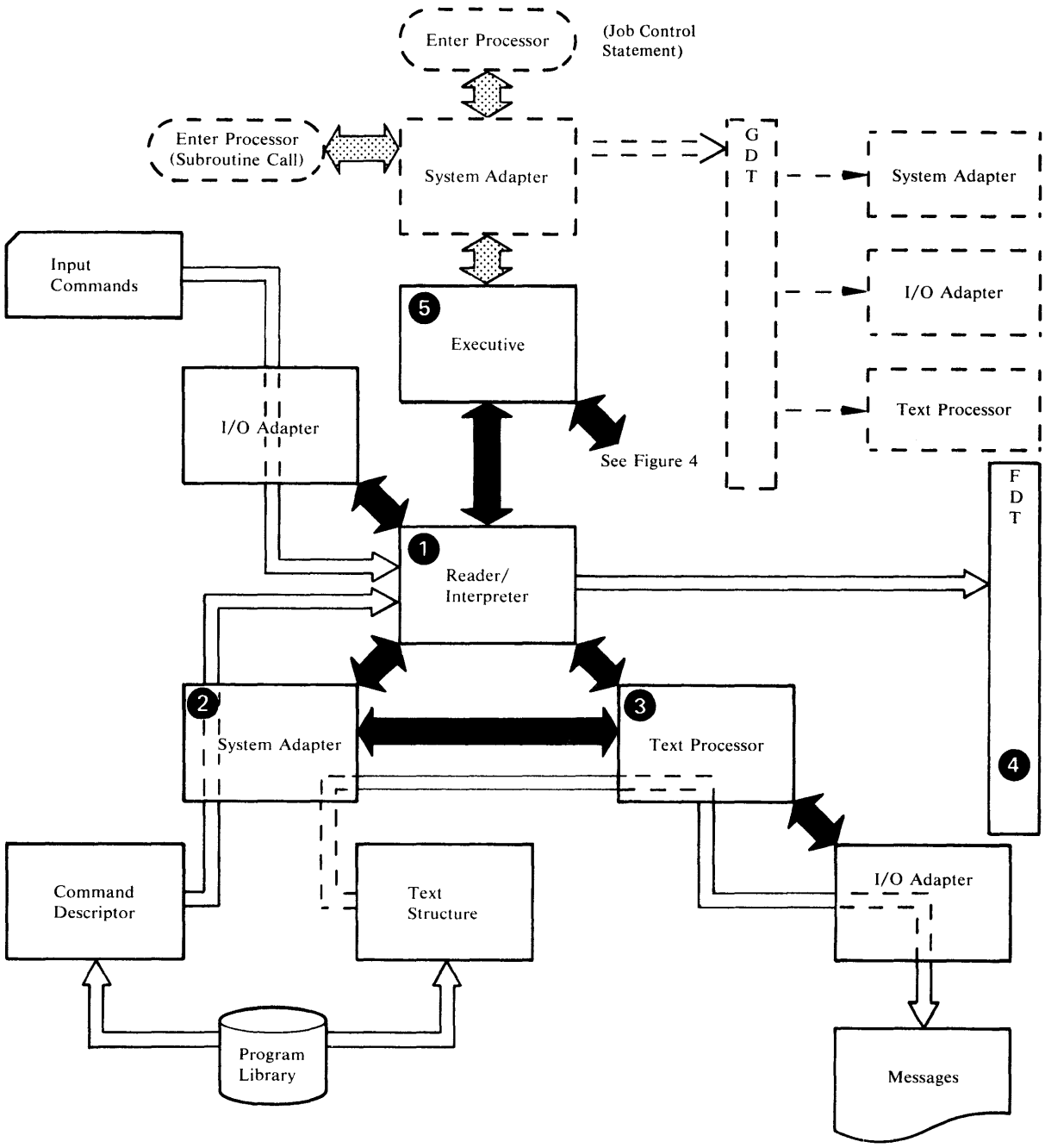
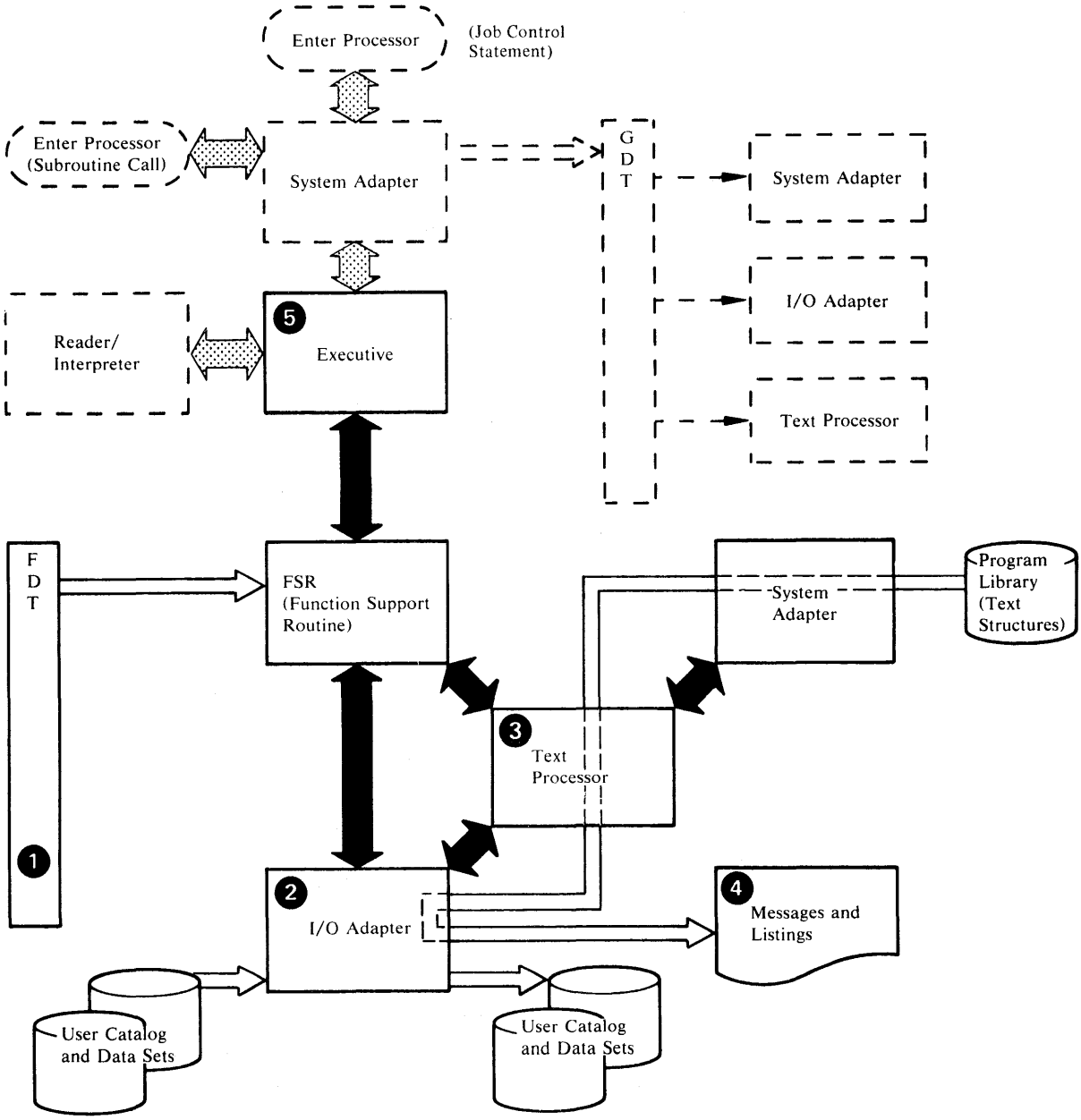


Figure 1-3. Reading and Parsing a Command (Part 1 of 2)

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 phase, and is loaded from the core image 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 the 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 phases (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.
5. Control returns to the Executive, along with the FDT and the name of the FSR needed to process this command. Figure 1-4 depicts the FSR in action.

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



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 1-4. 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 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
BI	BLDINDEX FSR	PM	PARM FSR
CD	Command Descriptor	PR	PRINT FSR
CL	CANCEL FSR	RC	EXPORTRA FSR
DB	Debugging Facility	RI	Reader/Interpreter
DE	DEFINE FSR	RM	IMPORTRA FSR
DI	NonVSAM Access	RP	REPRO FSR
	Method Macros	RS	RESETCAT FSR
DL	DELETE FSR	SA	System Adapter
EX	Executive	TP	Text Processor
IO	I/O Adapter	TS	Text Structure
LC	LISTCAT FSR	VY	VERIFY FSR
LR	LISTCRA FSR	XP	EXPORT FSR
MP	IMPORT FSR		

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

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 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.

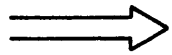
Most of the executable modules of the processor have no character dependencies. However, some modules of the Reader/Interpreter and the Text Processor have character dependencies. Such character dependencies are identified in the prologue of each module.

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.

Chapter 2: 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.

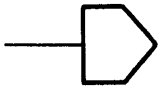


Data flow

Flow of control, entry and exit points

Data flow when existing data has been altered

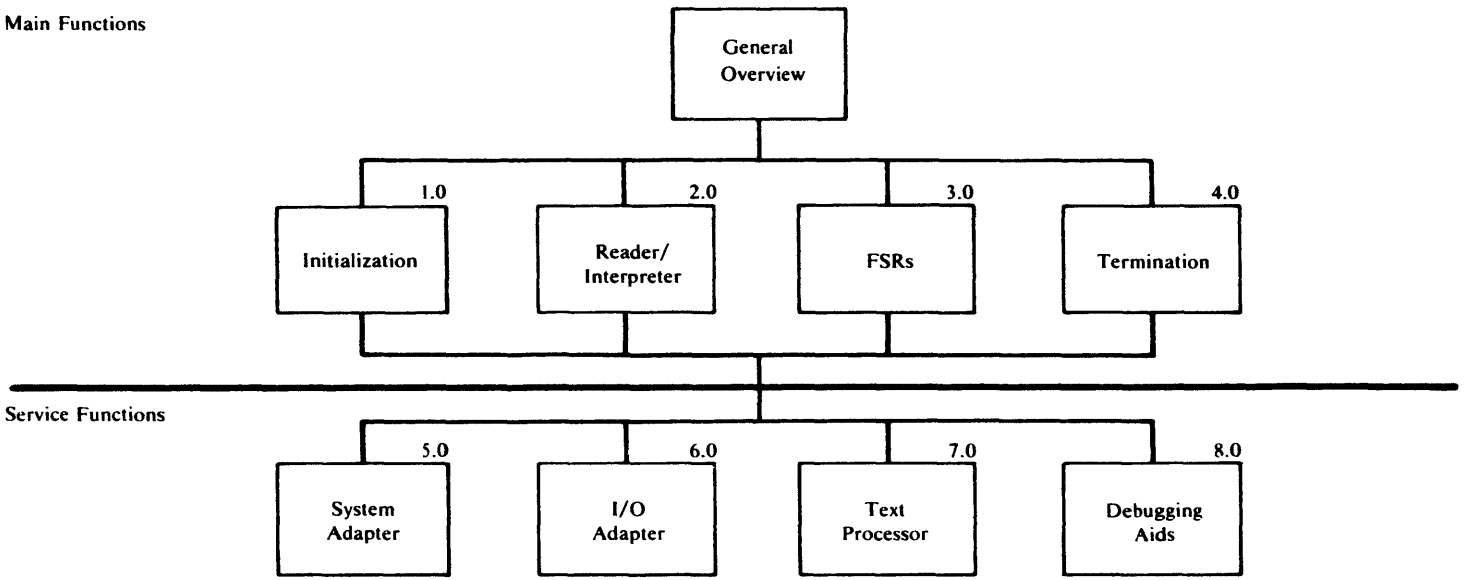
On page connector



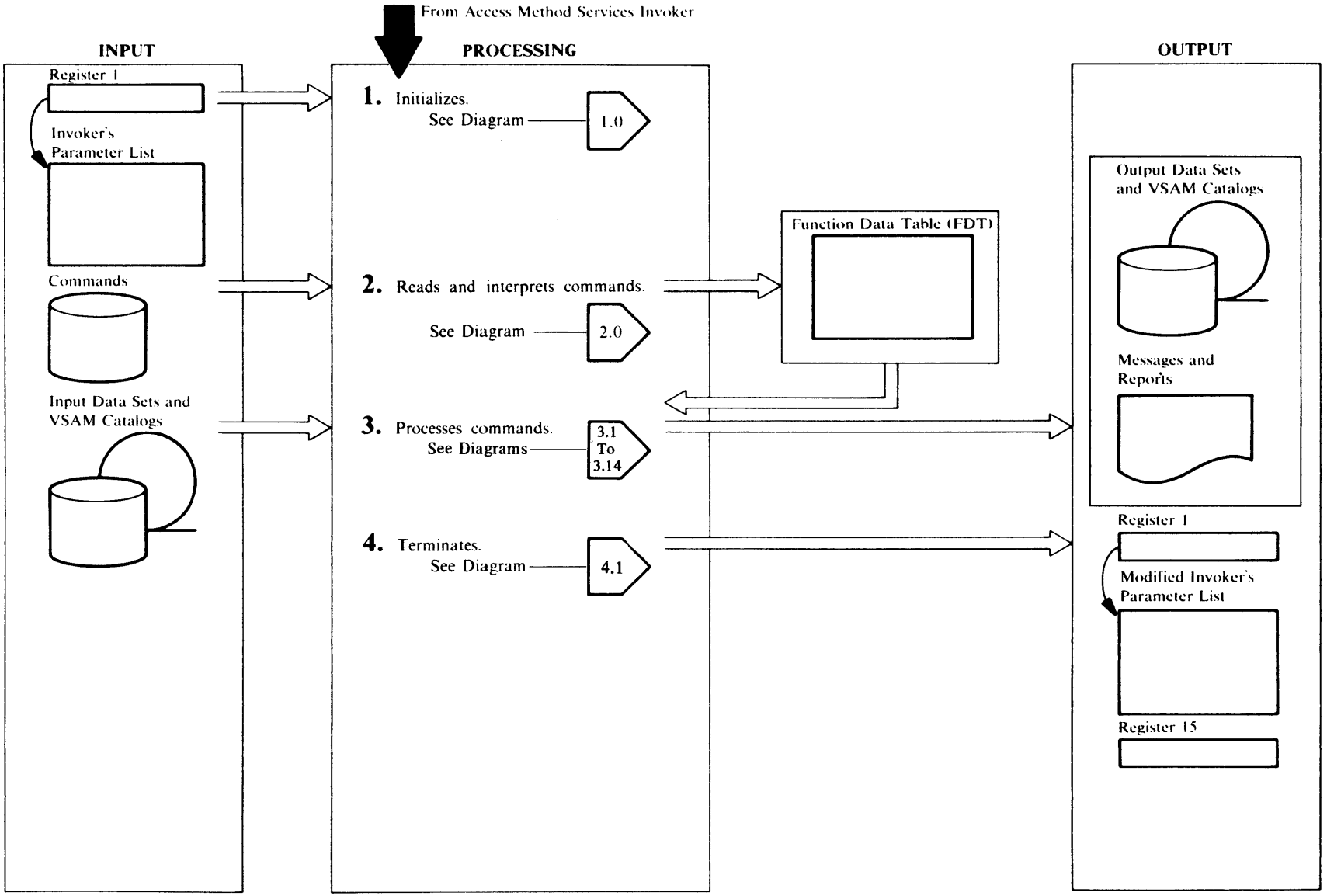
Off page connector

Pointer to more information

Access Method Services Visual Table of Contents



Access Method Services Overview



Extended Description for Access Method Services Overview

IDCEX01

1 **Procedure**: IDCEX01

Calls IDCEX02

2 **Procedure:** MAIN, CALLRI

If MAXCOND indicates termination, go to Step 4.
If MAXCOND does not indicate termination:

- Calls reader interpreter
- EOFCOND is set to R/I return code.

3 **Procedure:** MAIN, CALLFSR

If EOFCOND indicates end-of-file, go to Step 4.
If EOFCOND does not indicate end-of-file:

- FSR is called
- MAXCOND is set to maximum of MAXCOND and the return code for the FSR
- FDT is freed
- Text processor is reset
- Return to Step 2.

4 **Procedure:** IDCEX01

Calls IDCEX03.

Initialization Visual Table of Contents

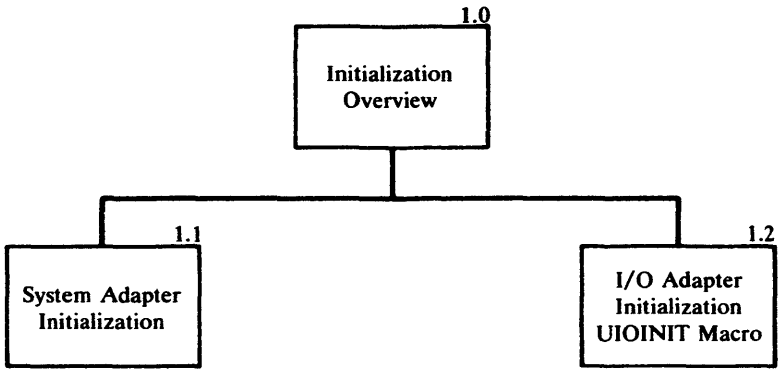
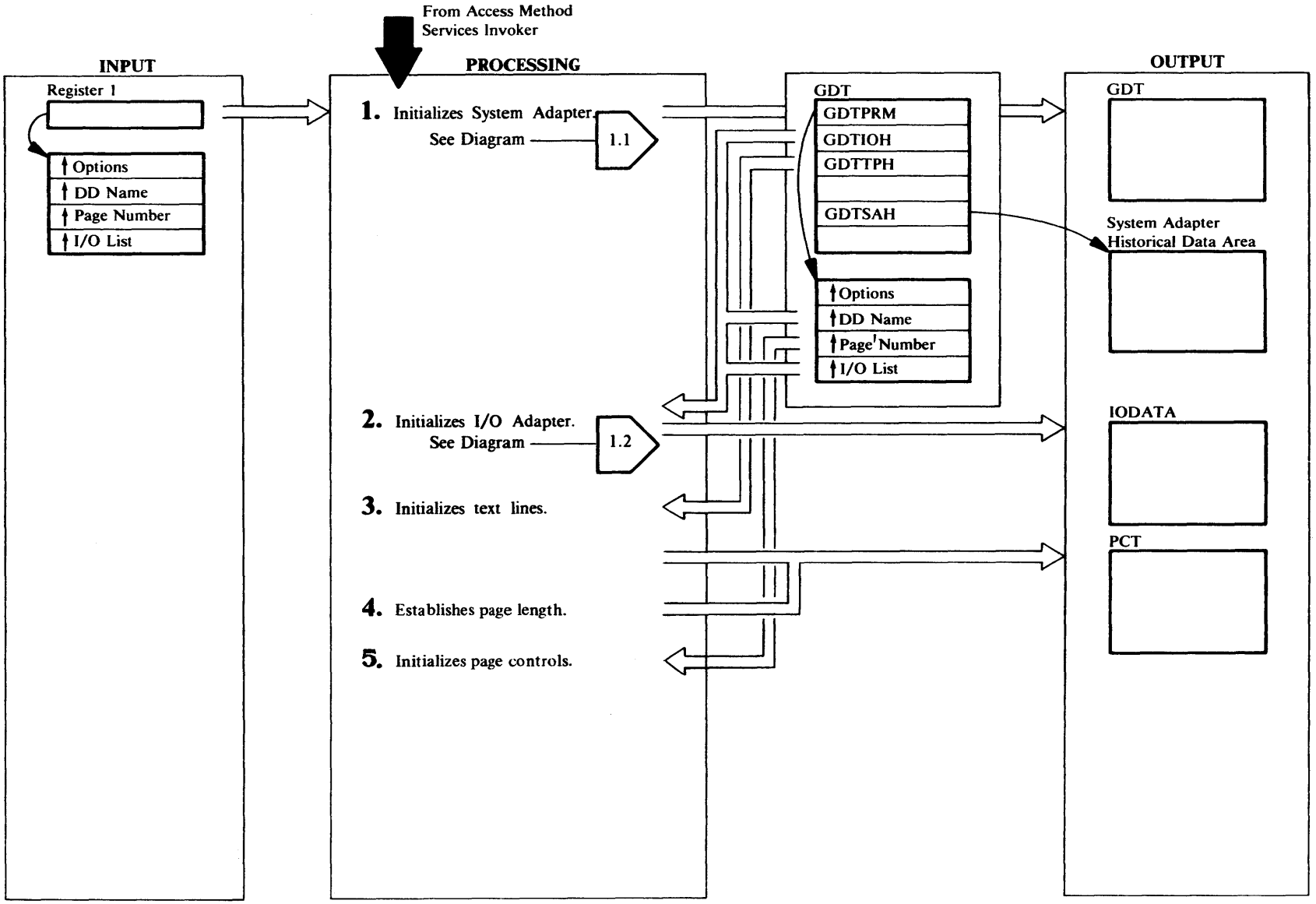


Diagram 1.0. Access Method Services Initialization Overview



Extended Description Diagram 1.0

IDCSA01

Procedure: IDCSA01

1 The System Adapter receives control from the invoker from either an EXEC statement or from a program. 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.

IDCEX02

Procedure: IDCEX02

2 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 alternate DD name list if supplied by the invoker. Diagram 1.2 shows I/O Adapter initialization in detail.

IDCEX02

Procedure: IDCEX02

3 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.

IDCEX02

Procedure: IDCEX02

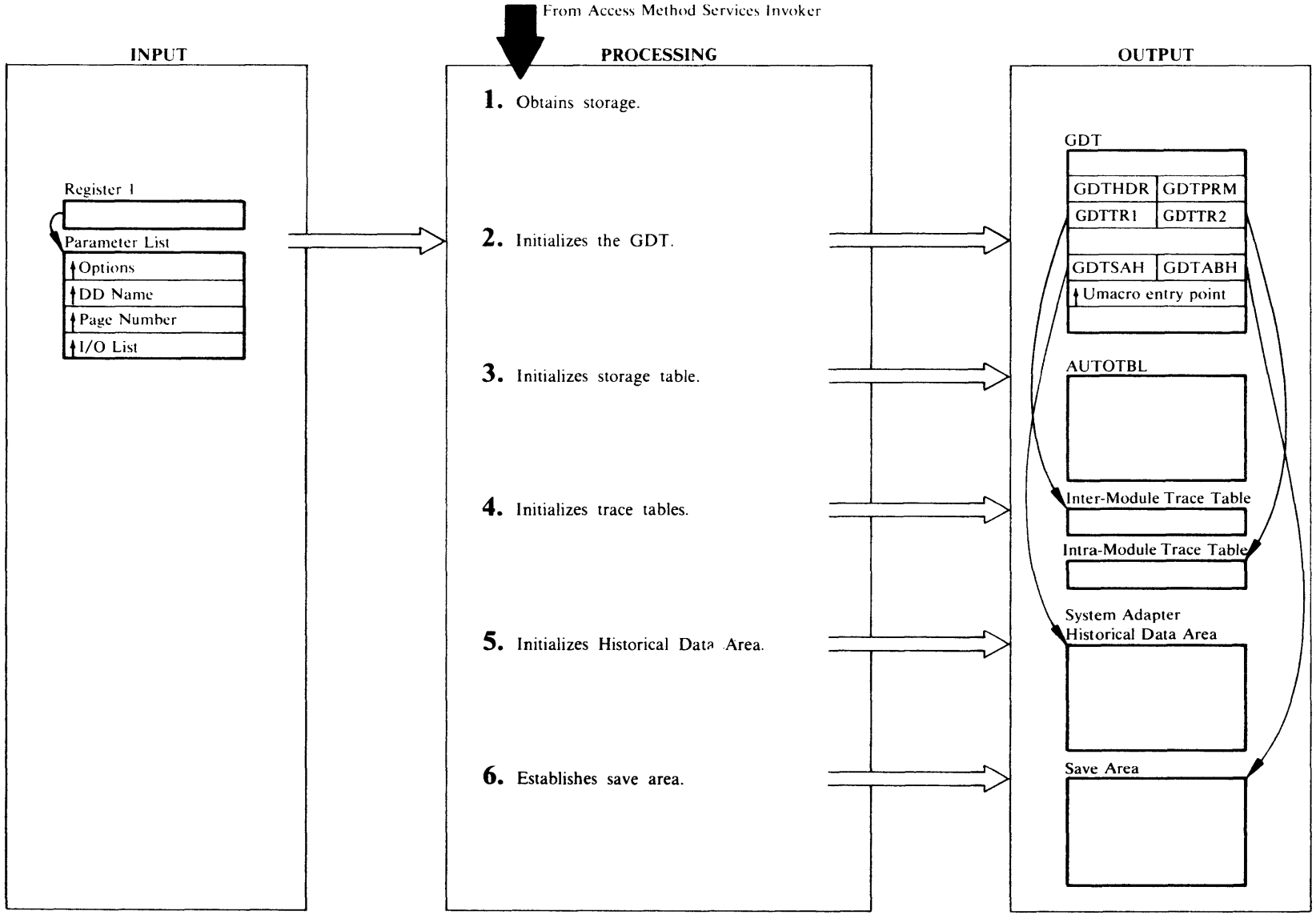
4 IDCEX02 issues a COMRG macro instruction to get the address of the partition communication region. It then extracts the value of "SYSLST lines per page" from displacement 78 and uses this value in a UREST macro instruction to establish the SYSLST page depth.

IDCEX02

Procedures: 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. Control is given to the R/I 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

IDCSA01

Procedure: IDCSA01

1 IDCSA01 issues a GETVIS 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

If the initial GETVIS fails, IDCSA01 issues an ABORT message via EXCP and returns to the invoker of Access Method Services.

IDCSA01

Procedure: IDCSA01

2 IDCSA01 puts the chatacters '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 U-macros, in the GDT. All remaining fields of the GDT contain zeros.

IDCSA01

Procedure: IDCSA01

3 Rather than obtaining new storage each time IDCSA02, IDCSA03, IDCTP01, or IDCIO01 is called, the System Adapter issues one GETVIS 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.

IDCSA01

Procedure: IDCSA01

4 IDCSA01 initializes the Inter- 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.

IDCSA01

Procedure: IDCSA01

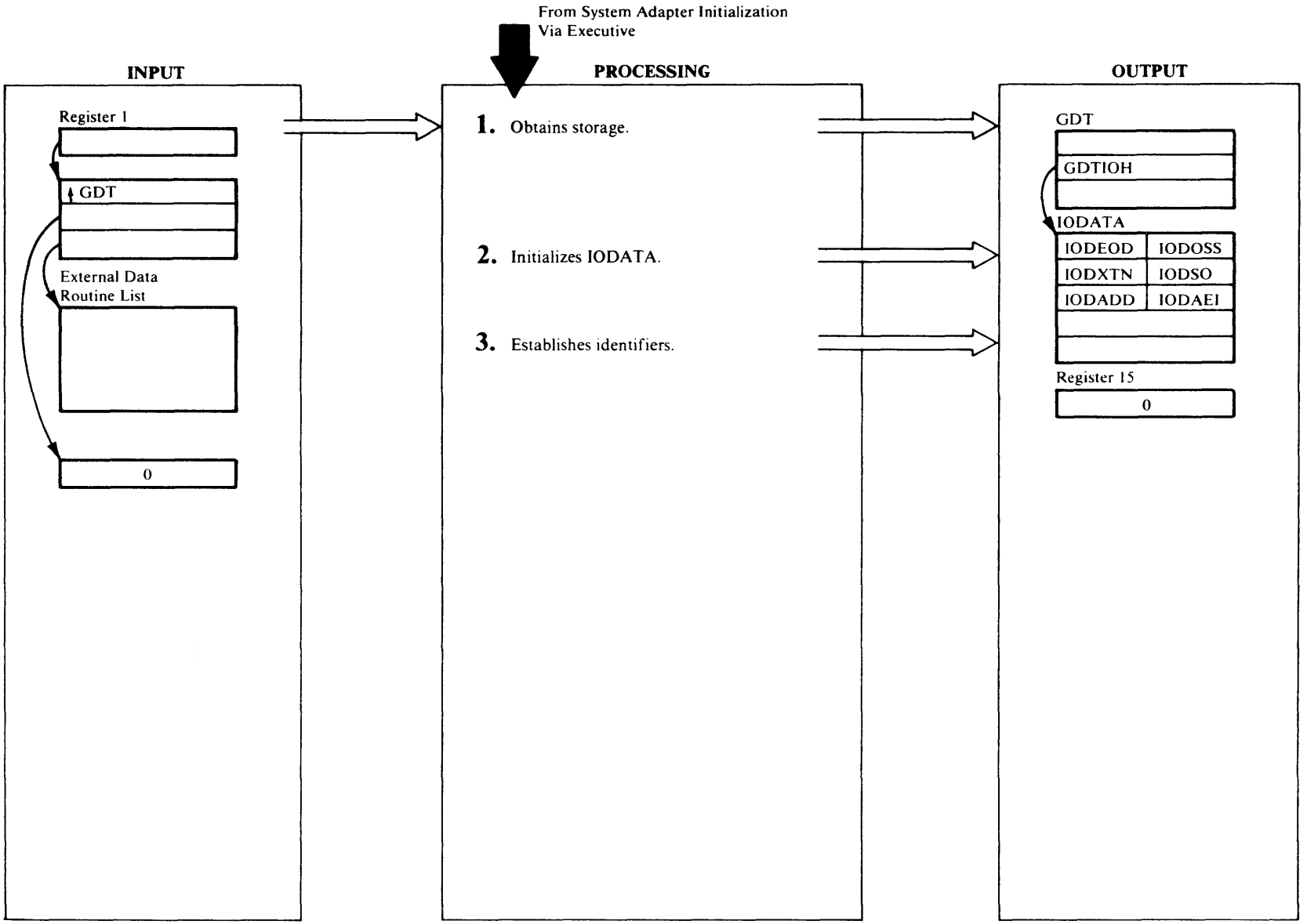
5 IDCSA01 sets the first UGPOOL storage area pointer in the System Adapter Historical Data Area to zero. It sets the last UGPOOL storage area pointer to the address of the first UGPOOL area pointer.

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. I/O Adapter Initialization – UIOINIT Macro



Extended Description for Diagram 1.2

IDCIO01

Procedure: IDCIOIT

1 The I/O Adapter issues a UGPOOL to obtain storage for its Historical Data Area—IODATA. IDCIOIT puts the IODATA address in the GDTIOH field in the GDT. If storage is not obtained from either UGPOOL, the I/O Adapter issues a UABORT to terminate the processor.

IDCIO01

Procedure: IDCIOIT

2 The I/O Adapter initializes IODATA. If the Access Method Services invoker supplied filenames for the system data sets, IDCIOIT puts the address of those filenames in the IODADD field of IODATA (this code is for compatibility with OS/VS; alternate filenames for system data sets cannot be used in VSE). If the invoker supplied the address a list of his own I/O programs, IDCIOIT puts that address in IODXTN. IDCIOIT puts the address of the Access Method Services End-of-Data routine in IODEOD. It puts the address for a synad routine for nonVSAM input data sets in IODOSS and the address for a synad routine for nonVSAM output data sets in IODSO. It also puts the address of the End-of-Data routine for VSAM data sets in IODAEI.

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. UOPEN gives the first data set the I/O Adapter is required to open the identification of IO01, the second 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 3.

Reader/Interpreter Visual Table of Contents

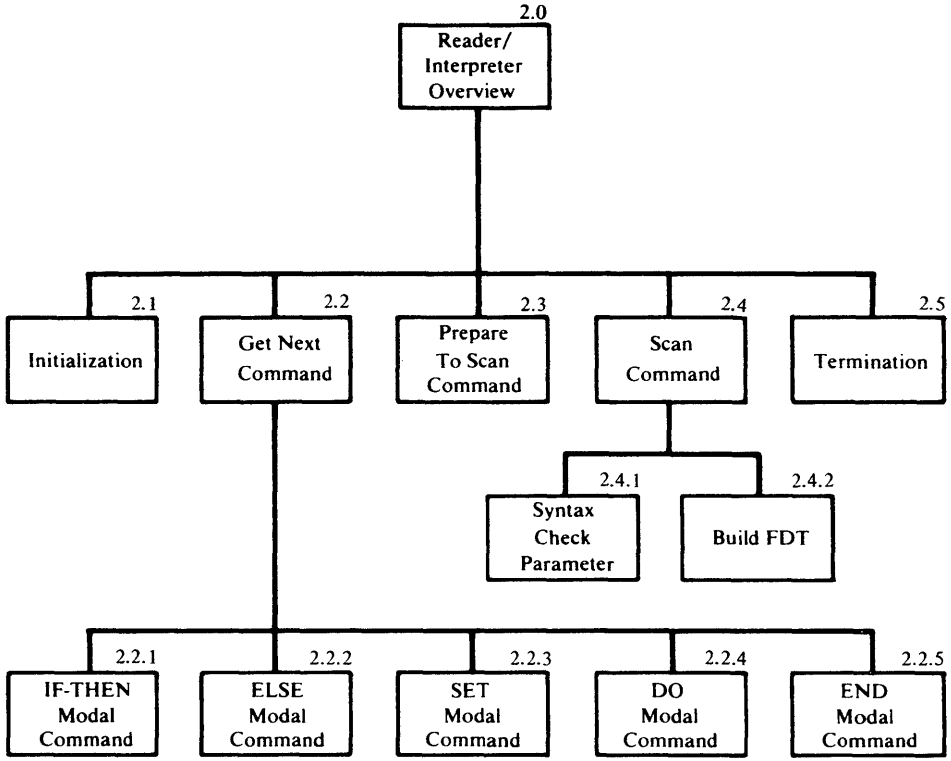
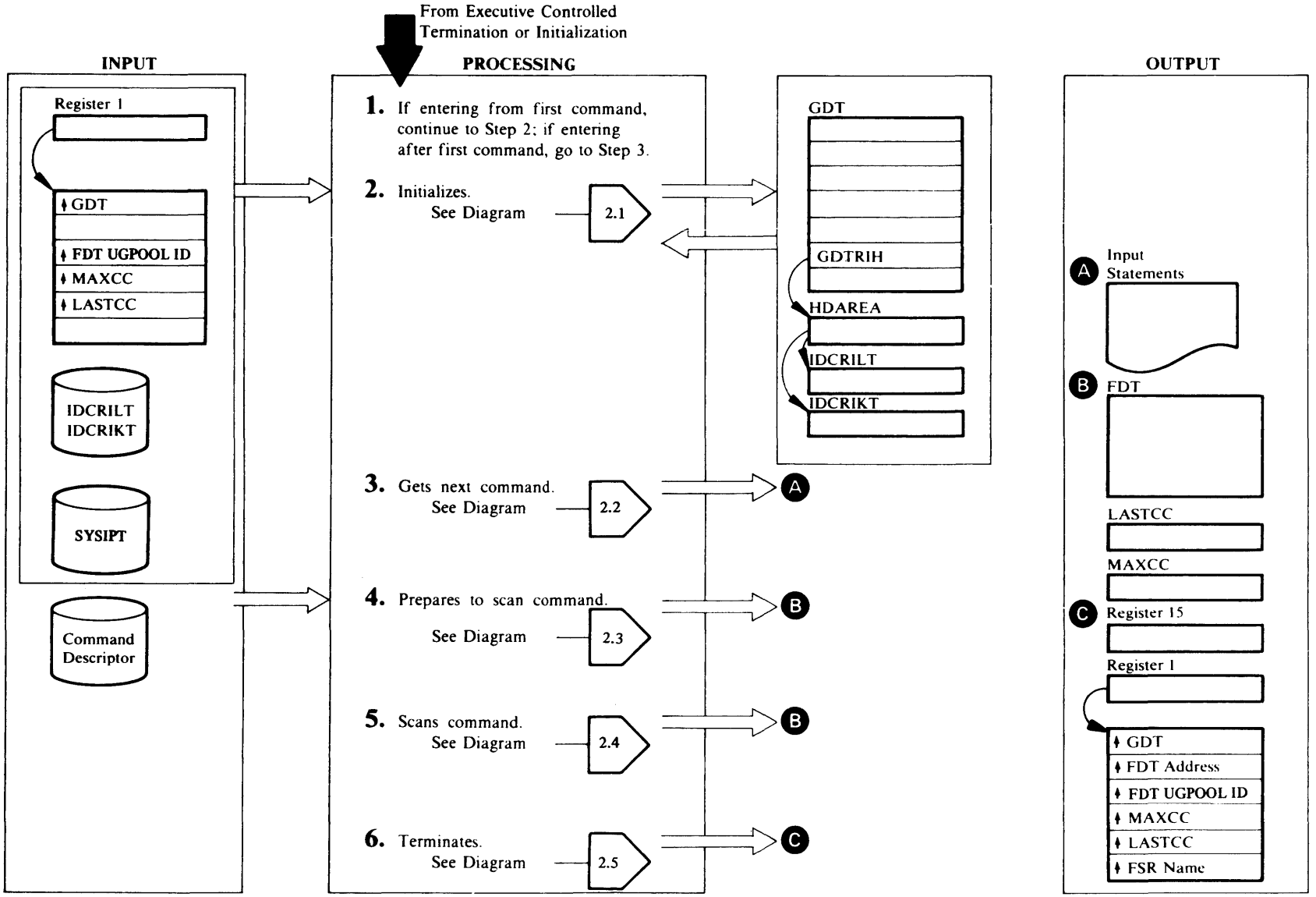


Diagram 2.0. Reader/Interpreter Overview



Extended Description for Diagram 2.0

IDCRI01

Procedure: RIINIT

1 If entrance is from Initialization, processing continues with step 2. If entrance is from Executive Controlled Termination, processing continues with step 3.

2 RIINIT initializes the Reader/Interpreter Historical Data Area, HDAREA. RIINIT loads the command descriptor name table, IDCRILT, and the modal command name table, IDCRIKT. RIINIT opens the input data set, SYSIPT, and RIINIT prepares the parameters from the EXEC statement for scanning, if they exist. Diagram 2.1 shows the initialization procedure in detail.

IDCRI01

Procedures: 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 2.2 shows obtaining a command in detail.

IDCRI02

Procedure: IDCRI02

4 IDCRI02 loads the command descriptor for the functional command to be scanned. IDCRI02 initializes the Function Data Table, FDT. Diagram 2.3 shows the preparation for command scanning in detail.

IDCRI01

Procedures: SCANCMD, KWDPARM, PCSPARM, INREPEAT, BUILDFDT, CONVERT, GETSPACE, DSIDCHK, ERROR1, ERROR2

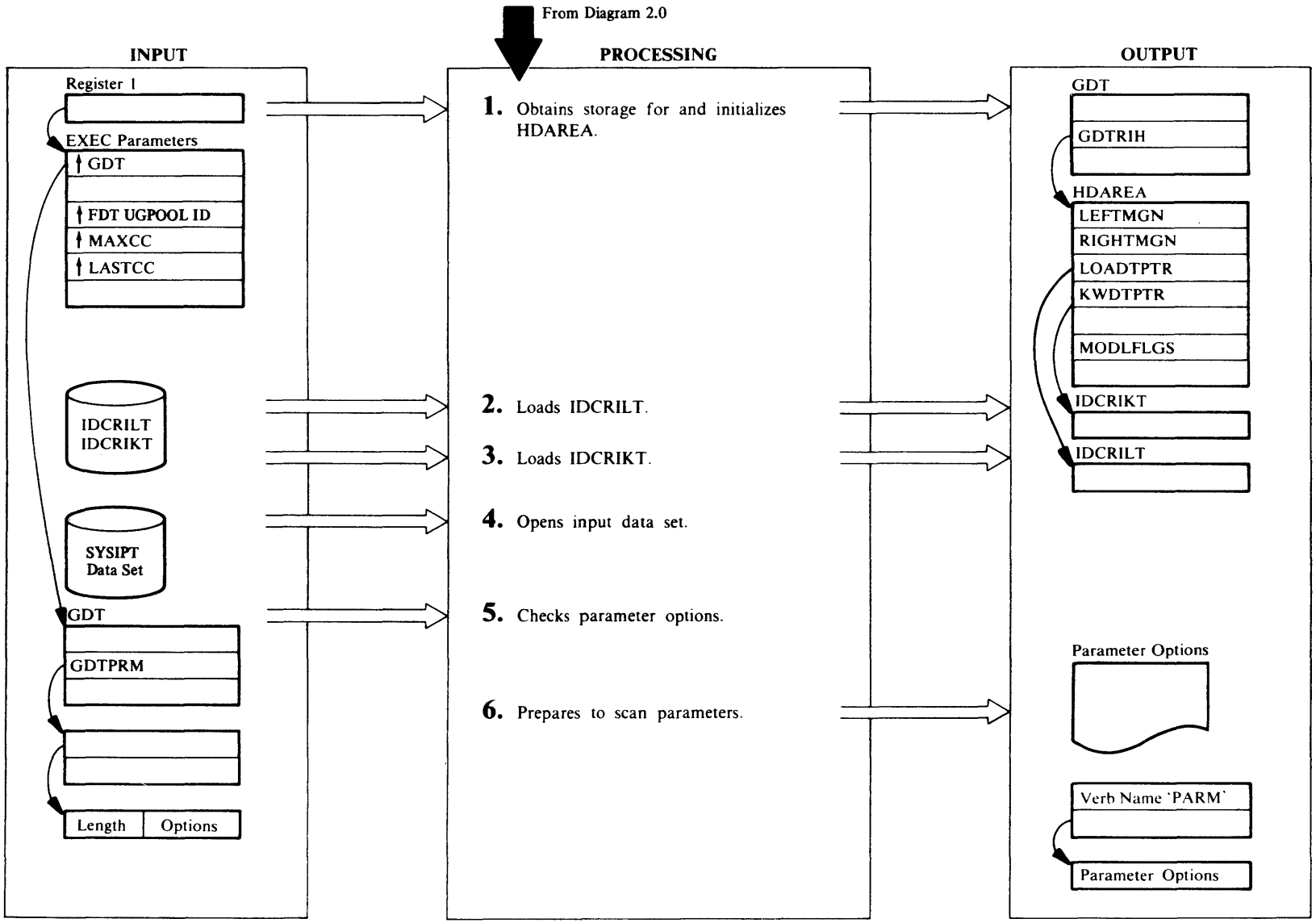
5 SCANCMD and BUILDFDT check the functional command for correctness. If the command is incorrect, ERROR1 or ERROR2 writes an error message. BUILDFDT and INREPEAT complete the FDT for correct commands. Diagram 2.4 shows the command scanning in detail.

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 which 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 4.1, with an indication that the processor cannot continue. Diagram 2.5 shows termination processing in detail.

Diagram 2.1 Reader/Interpreter Initialization



Extended Description for Diagram 2.1

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 2.5, with an indication that causes the processor to end.

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.

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.

IDCRI01

Procedure: RIINIT

4 RIINIT opens the input data set which has a default filename of SYSIPT. If SYSIPT cannot be opened, control is given to Reader/Interpreter termination, Diagram 2.5, with an indication that causes the processor to end.

IDCRI01

Procedure: RIINIT

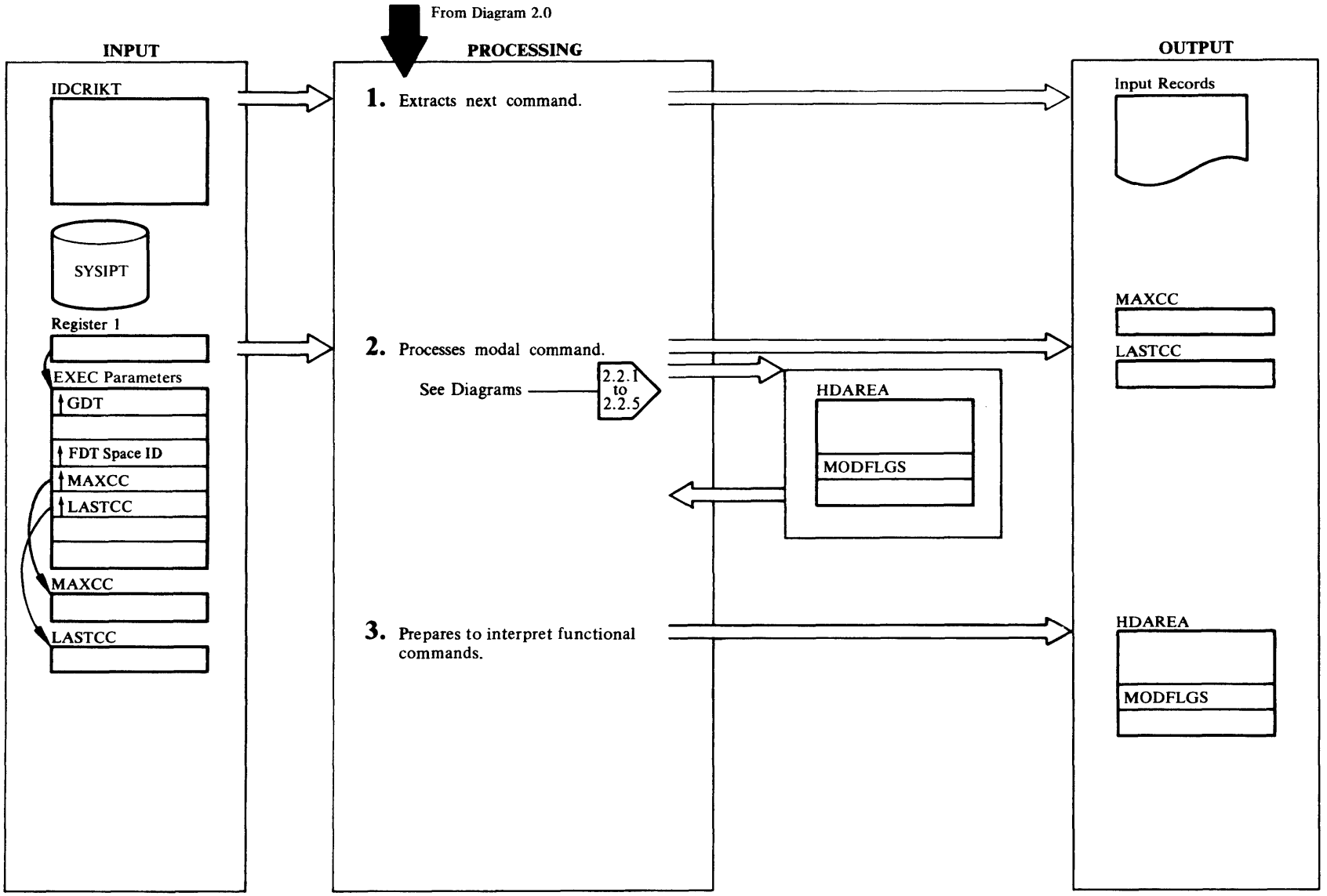
5 The Reader/Interpreter checks for parameters supplied before SYSIPT is read. The invoker may supply parameters by putting them in the EXEC job control statement. Parameters may also be supplied through the data the user provides to the processor at the time the user's program invokes Access Method 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 2 bytes of the parameters is the total length of the parameters. If no parameters are supplied, the length field is zero.

IDCRI01

Procedure: RIINIT

6 The parameters are printed on SYSLST 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 Reader/Interpreter just as though a PARM command had been encountered in SYSIPT. After the pseudo PARM command is executed by the PARM FSR, Executive Controlled Termination gives Reader/Interpreter control to read the first command. Control goes to Diagram 2.2 to get the first command.

Diagram 2.2. Reader/Interpreter Get Next Command



Extended Description for Diagram 2.2

IDCRI01

Procedures: GETNEXT, GETRECRD, NXTFIELD, NEXTCHAR

1 GETRECRD reads SYSIPT to get an input record and writes each input record on SYSLST. 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.

IDCRI01

Procedures: 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 2.2.1
ELSE, Diagram 2.2.2
SET, Diagram 2.2.3
DO, Diagram 2.2.4
END, Diagram 2.2.5

IDCRI01

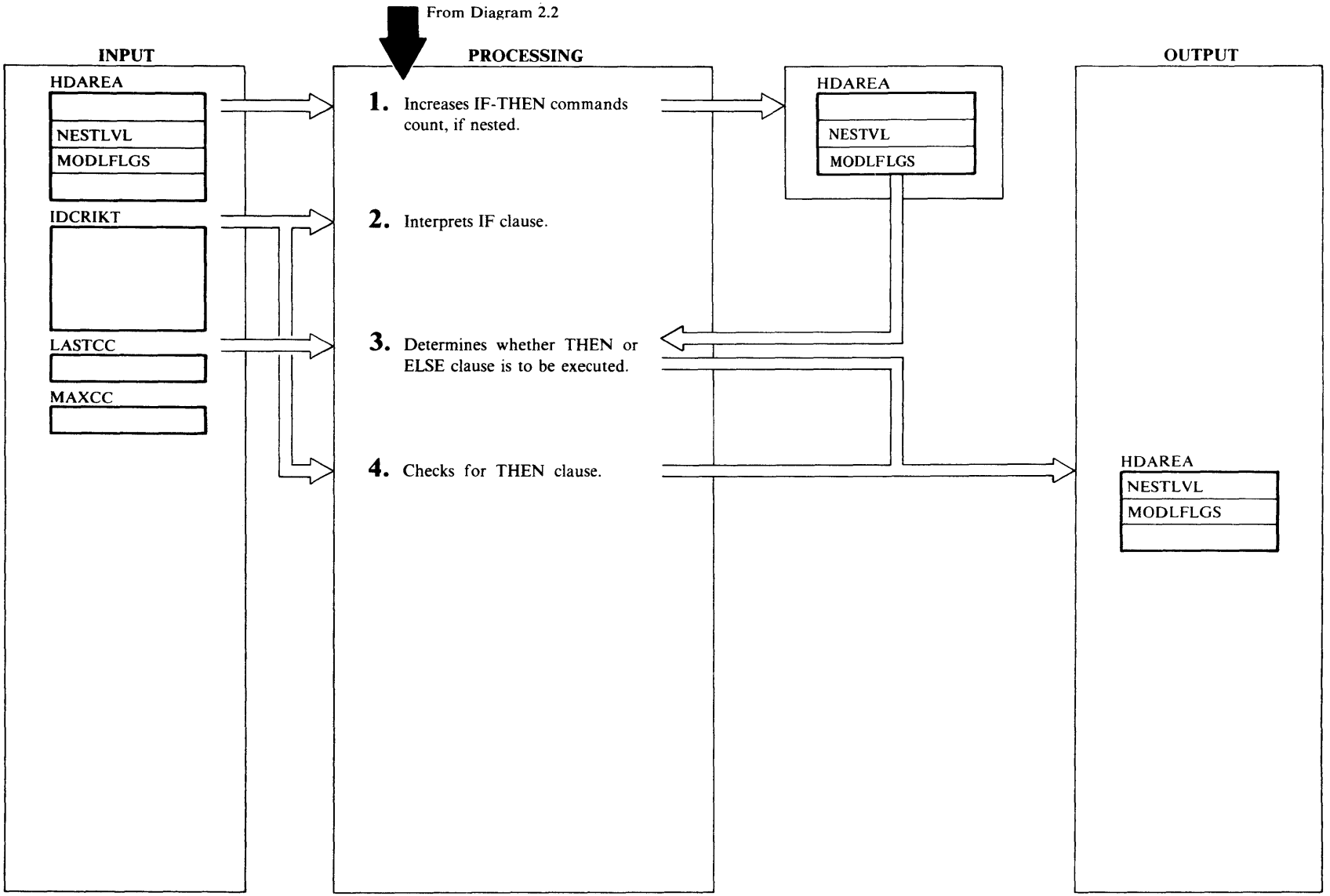
Procedure: GETNEXT

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
BLDINDEX, Diagram 3.11
CANCEL, Diagram 3.16
DEFINE, Diagram 3.2
DELETE, Diagram 3.3
EXPORT, Diagram 3.4
EXPORTRA, Diagram 3.13
IMPORT, Diagram 3.5
IMPORTRA, Diagram 3.14
LISTCAT, Diagram 3.6
LISTCRA, Diagram 3.12
PARM, Diagram 3.7
PRINT, Diagram 3.8
REPRO, Diagram 3.9
RESETCAT, Diagram 3.15
VERIFY, Diagram 3.10

Control goes to Diagram 2.4 to scan the command.

Diagram 2.2.1. Reader/Interpreter IF-THEN Modal Command



Extended Description for Diagram 2.2.1

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 2.5, to cause the Executive to terminate the processor.

IDCRI01

Procedures: 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.

IDCRI01

Procedure: MODALIF

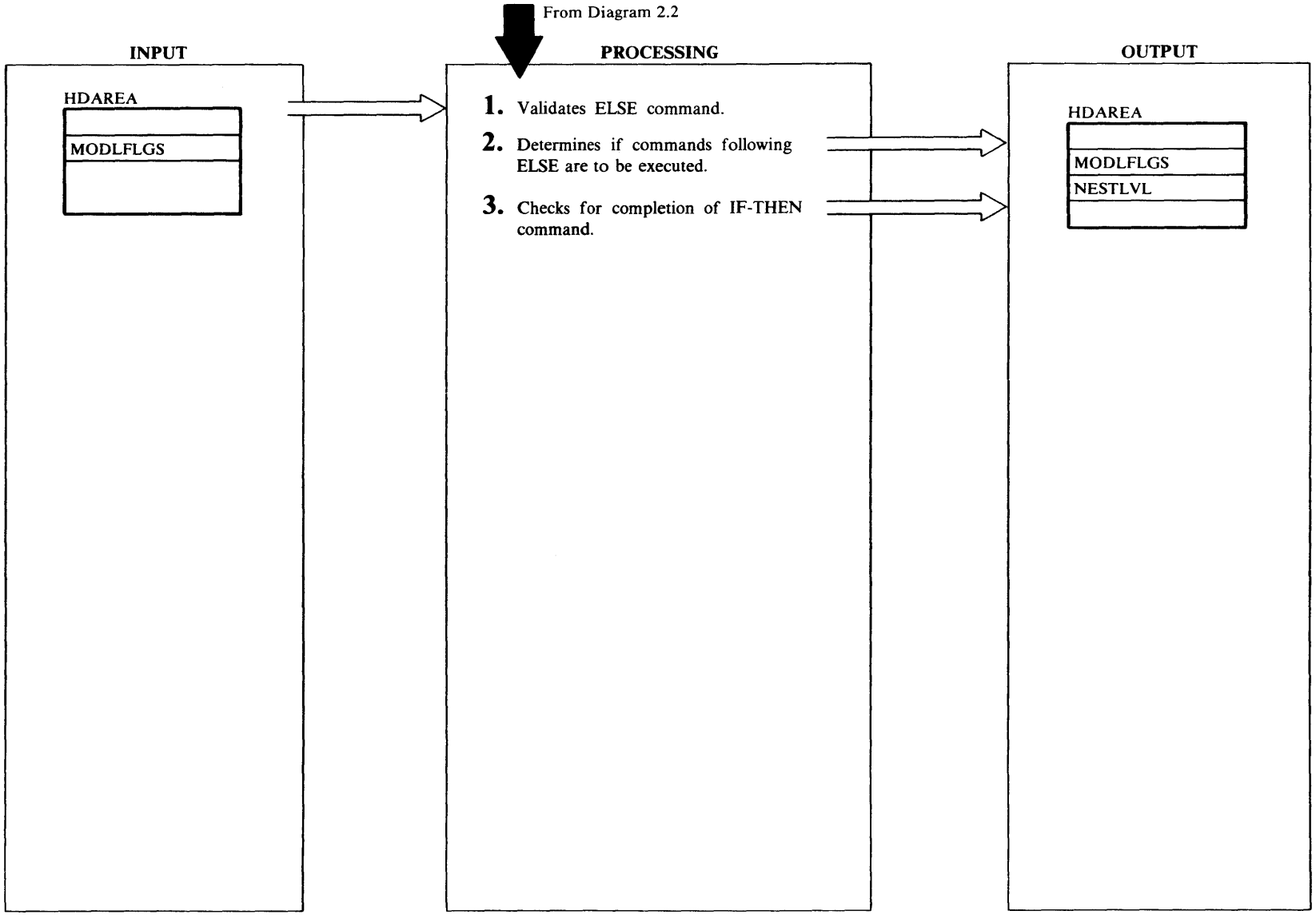
3 MODALIF sets the THENFLAG to 1 to indicate that the THEN clause of the IF—THEN command is being processed. MODALIF compares the value of LASTCC or MAXCC with the number in the IF –THEN command and evaluates it for true or false depending upon the operator. If the result is false, MODALIF sets the SKIPFLAG 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.

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 2.2 to obtain the next command.

Diagram 2.2.2. Reader/Interpreter ELSE Modal Command



Extended Diagram for Diagram 2.2.2

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 step 1, Diagram 2.2.1, is followed.

IDCRI01

Procedure: MODLELSE

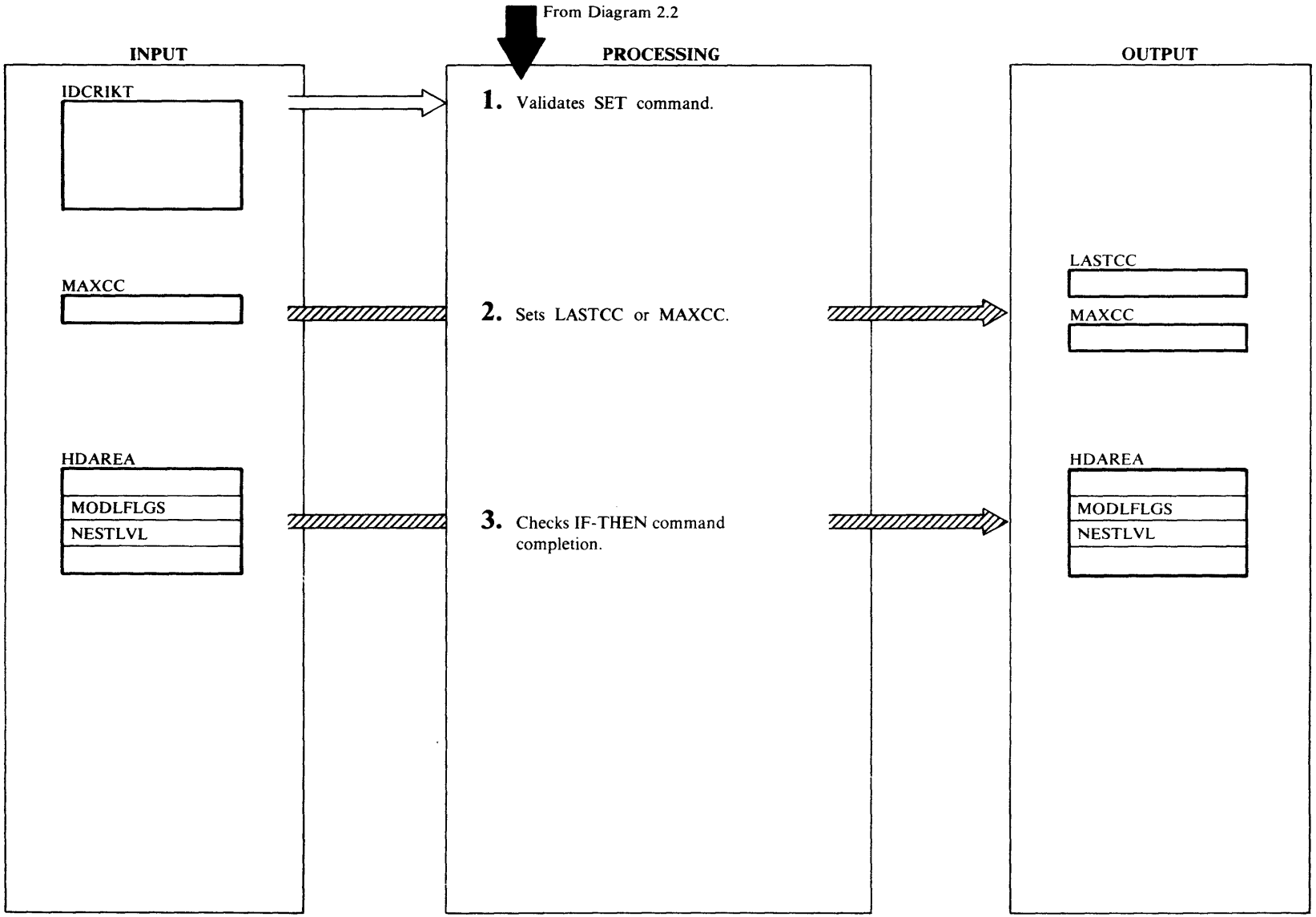
2 SKIPFLAG indicates whether the commands in the ELSE clause of the IF—THEN command should be executed or only checked for syntax errors. If SKIPFLAG is zero, the THEN clause of the IF—THEN command was executed; the ELSE clause should not be executed, and MODLELSE sets SKIPFLAG to 1. If SKIPFLAG is 1, 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 or the ELSE clause of the nested IF—THEN—ELSE command is executed.

IDCRI01

Procedures: 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 2.2 to obtain the next command whether or not a terminator follows the ELSE.

Diagram 2.2.3. Reader/Interpreter SET Modal Command



Extended Description for Diagram 2.2.3

IDCRI01

Procedures: 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 2.2.1, step 1 is followed.

IDCRI01

Procedure: MODALSET

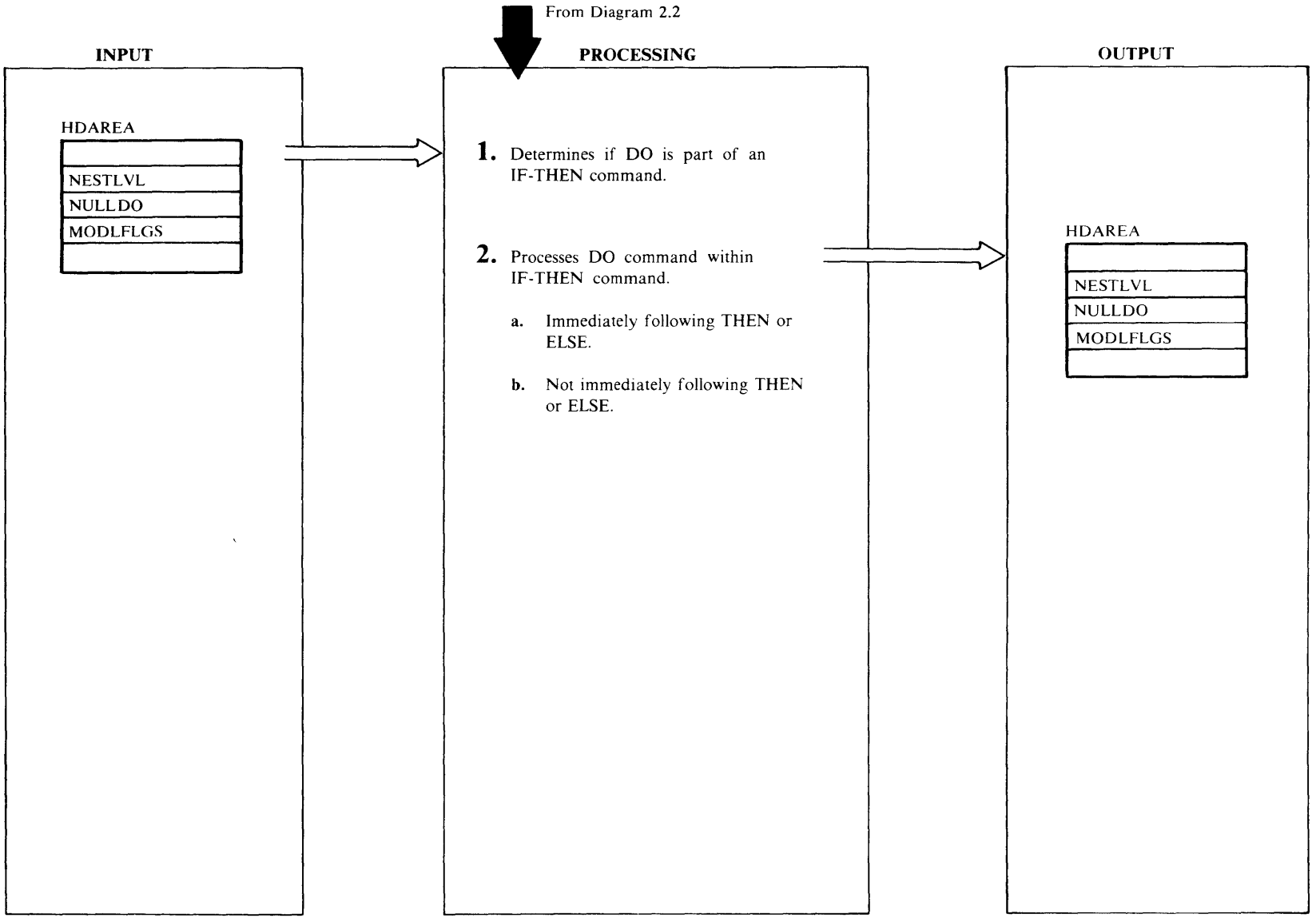
2 MODALSET obtains MAXCC or LASTCC and changes its value to the value specified in the SET command. If the command is SET LASTCC, MODALSET compares MAXCC and LASTCC, and the larger value is put into MAXCC. If the SET command is only being checked for syntax errors, neither MAXCC nor LASTCC is changed.

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 2.2 to obtain the next command.

Diagram 2.2.4. Reader/Interpreter DO Modal Command



Extended Description for Diagram 2.2.4

IDCRI01

Procedures: GETNEXT, NXTFIELD, NEXTCHAR

1 If a DO command is not part of an IF—THEN command, control returns to Diagram 2.2 to obtain the next command. If a DO command is part of an IF—THEN command, processing continues to step 2.

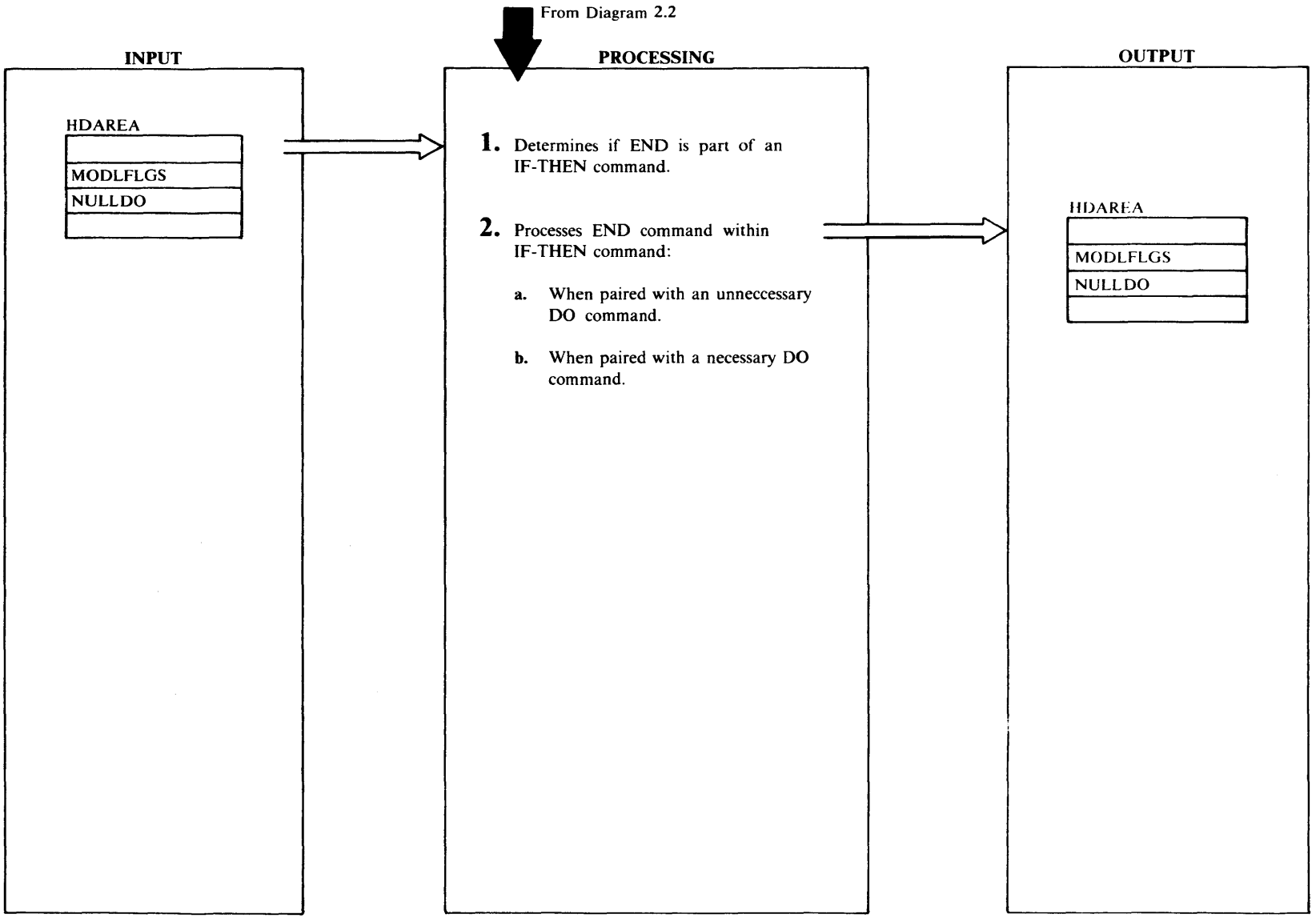
IDCRI01

Procedures: 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 2.2 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 2.2 for the first command of the unnecessary DO group.

Diagram 2.2.5. Reader/Interpreter END Modal Command



Extended Description for Diagram 2.2.5

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 2.2 to obtain the next command. If NESTLVL contains a value other than zero, processing continues with step **2**

IDCRI01

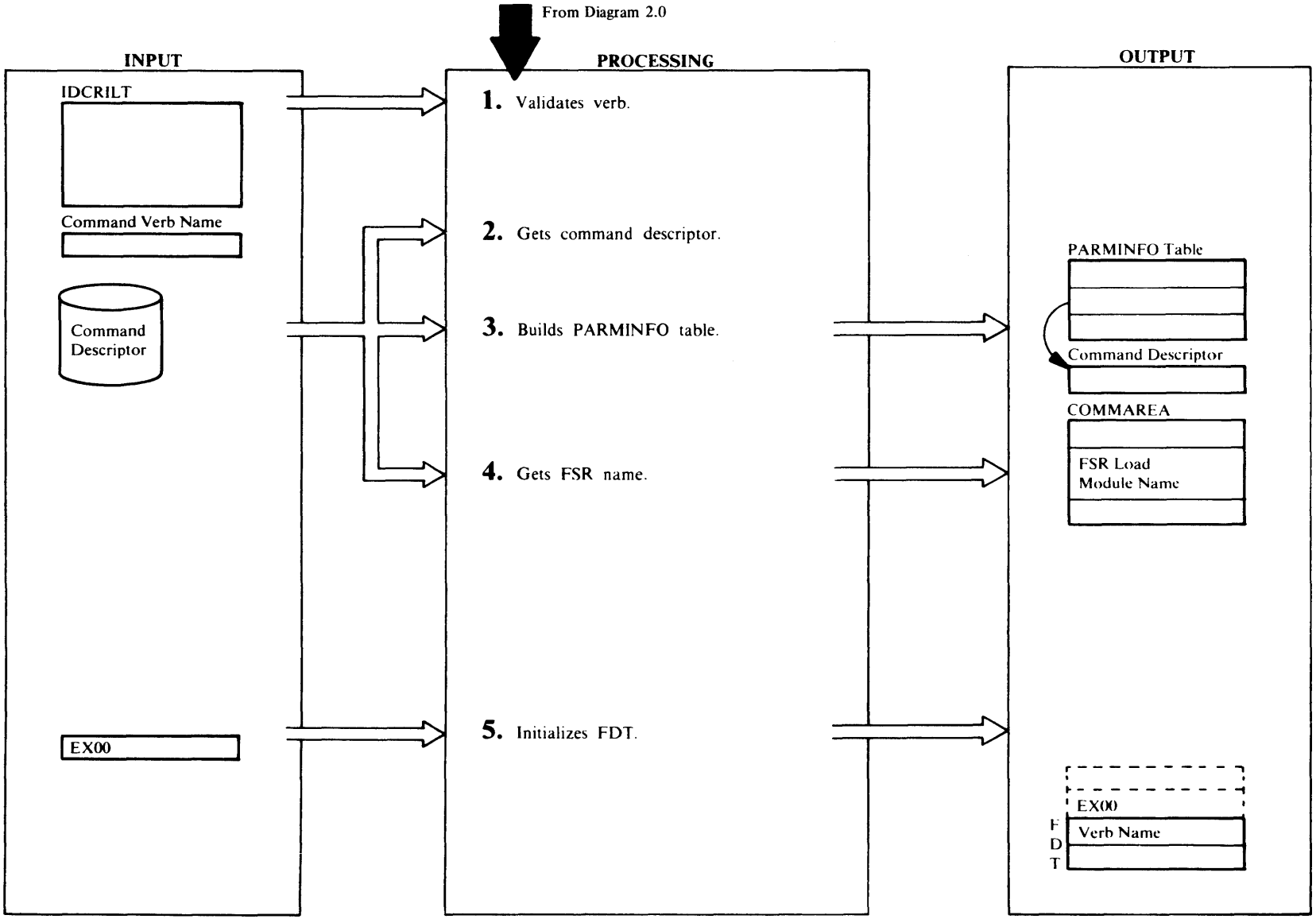
Procedure: GETNEXT

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 2.2 to obtain the next command.

Diagram 2.3 Reader/Interpreter Prepare to Scan Command



Extended Description for Diagram 2.3

IDCRI02
IDCRI01

Procedures: IDCRI02, ERROR2

1 Reader/Interpreter Initialization, Diagram 2.1, gives control to this section only if parameters were present before SYSIPT was read. Otherwise, control comes from Diagram 2.2. 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 or the load of the command descriptor fails due to phase not found, the verb is invalid, and ERROR2 prints a message on SYSLST. The remainder of the command is ignored, and control is given to Reader/Interpreter termination, Diagram 2.5

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 three parameters—KEYS, *length* with value 5, and *offset* with value 40. KEYS is a keyword while *length* and *offset* are each positional parameters. (*length*, *offset*) is not a list of constants because the second item, *offset*, is different from the first, *length*. *length* and *offset* are subparameters of KEYS.
- KEYRANGES ((5, 40), (50, 60), (70, 80)) is three 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.

IDCRI02
IDCRI01

Procedures: IDCRI02, SETFLAG

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 R/I uses to determine the relationship among parameters.

IDCRI02

Procedure: IDCRI02

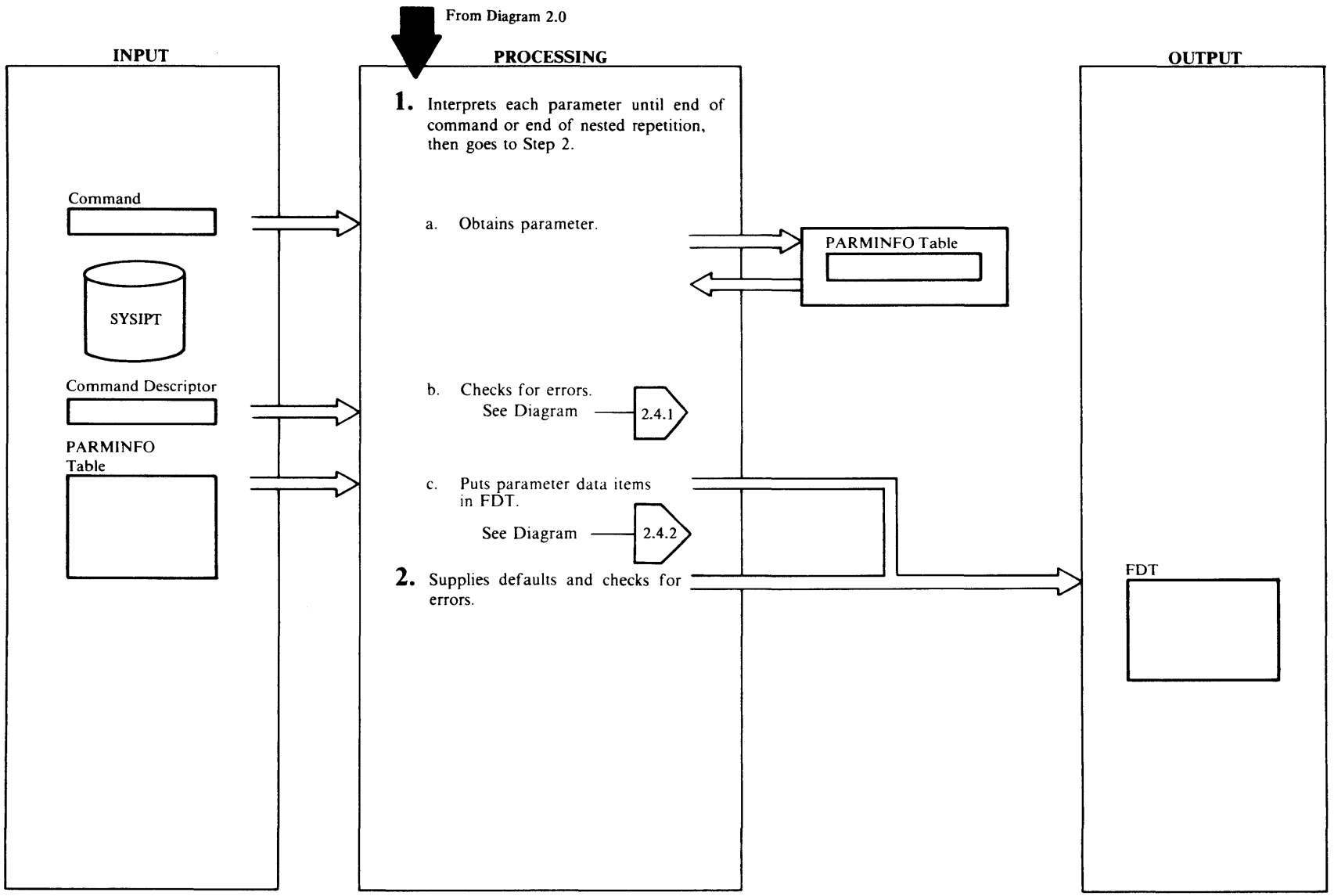
4 IDCRI02 obtains the FSR load module name from the command descriptor and places the name in the FSRLNAME field in COMMAREA. The Executive uses the FSR load module name to load the FSR that executes the command.

IDCRI02

Procedure: IDCRI02

5 IDCRI02 obtains storage for the Function Data Table, FDT. The verb uses 8 bytes of storage, and each parameter uses 4 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 8 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.

Diagram 2.4 Reader/Interpreter Scan Command



Extended Description for Diagram 2.4

IDCRI01

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

1 If the Reader/Interpreter is processing a specified parameter, processing continues with step 1a. If the Reader/Interpreter 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 is 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 SYSIPT 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 SYSLST and sets LASTCC to 12. The rest of the command is skipped, and control is passed to R/I termination.

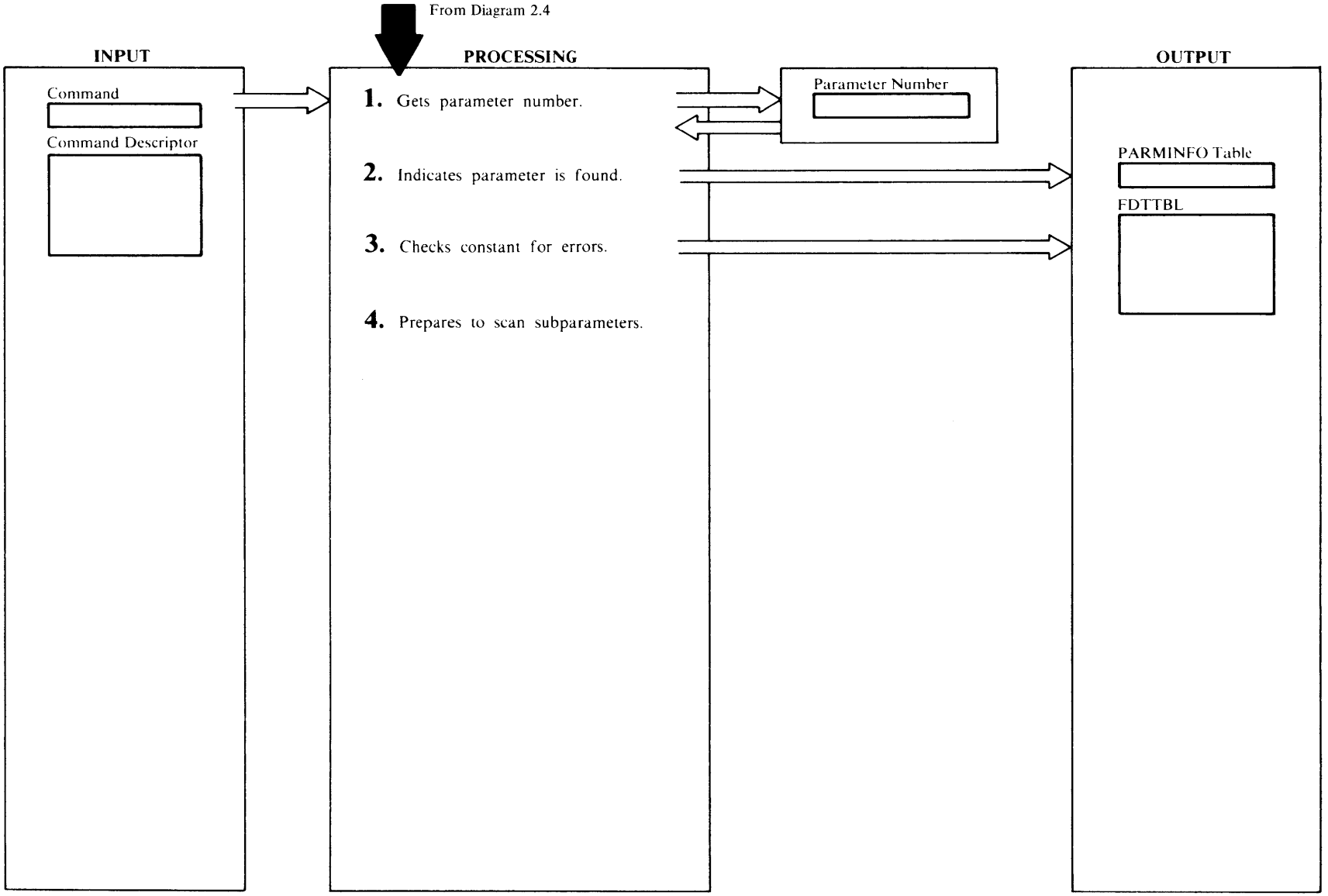
   c. As SCANCMD scans the command, BUILDFDT encodes the command into the FDT in order to describe the command to the FSR that will execute it. The data items are checked for additional errors (errors are processed as described in step 1.b). Parameter scanning continues one parameter at a time until the 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.

IDCRI01

Procedures: DEFAULTS, SETDELT, 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 2.4.1 Reader/Interpreter Syntax Check Parameter



Extended Description for Diagram 2.4.1

IDCRI01

Procedures: 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.

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.

IDCRI01

Procedures: 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 SYSLST 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.

IDCRI01

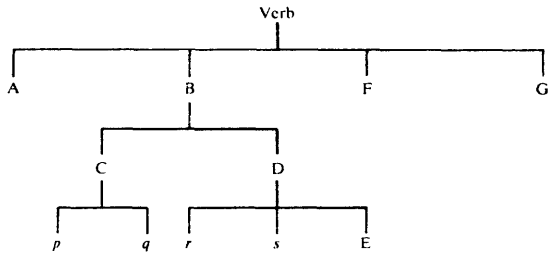
Procedure: SCANCMD

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*)

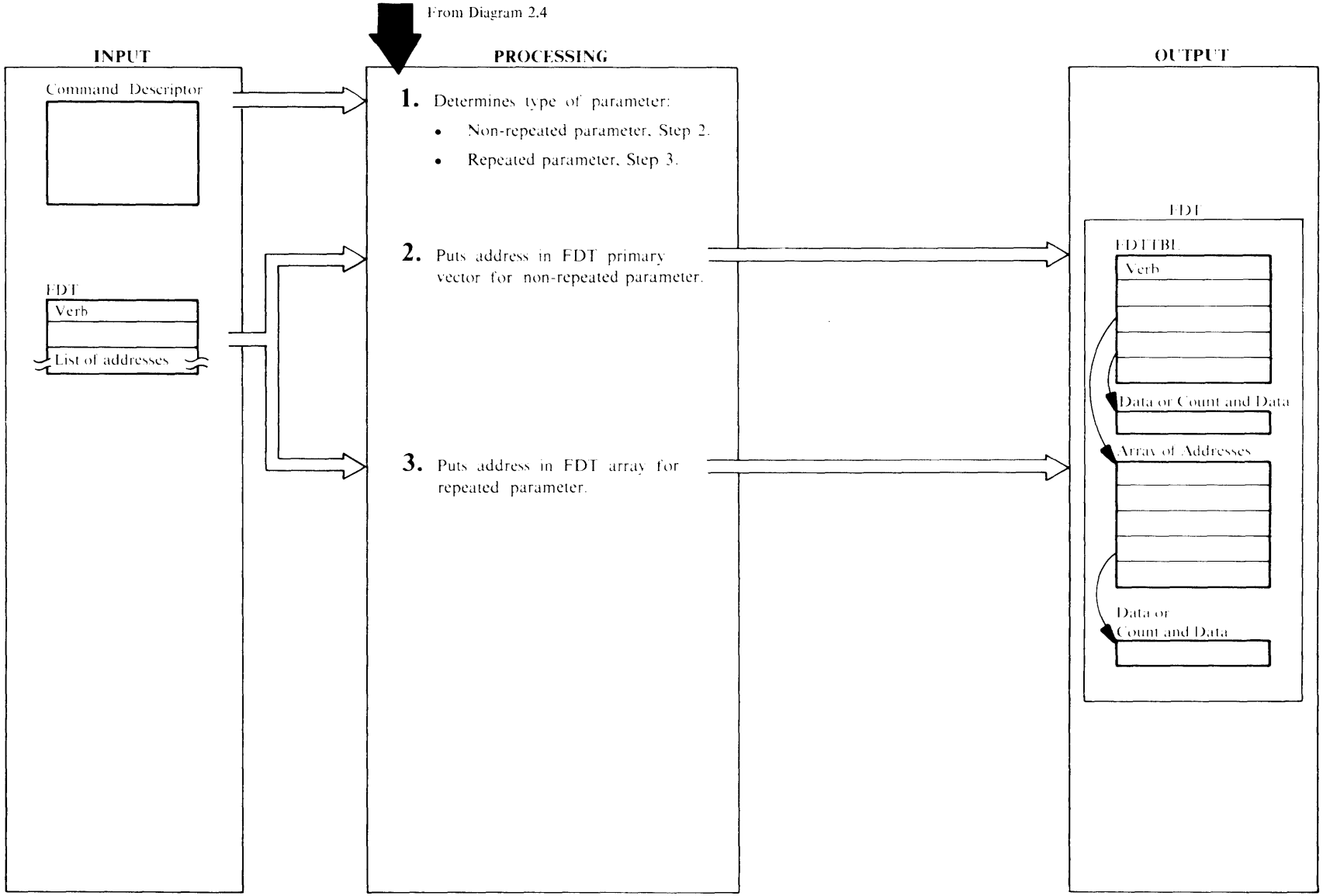
A, B, C, D, E, F, and G are keyword parameters. *p*, *q*, *r*, and *s* are positional parameters. *x* represents data.

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 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.

Diagram 2.4.2 Reader/Interpreter Build FDT



Extended Description for Diagram 2.4.2

IDCRI01

Procedures: PACKCVB, CONVERT, GETSPACE, MORESPACE

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 2.3 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 R/I listings, the word *scaler* is interchangable with the word *constant*. Control returns to Diagram 2.4 for the next parameter.

IDCRI01

Procedures: SCANMD, INREPEAT, DEFAULTS, NEEDNOTS

3 Each repeated parameter—positional 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:

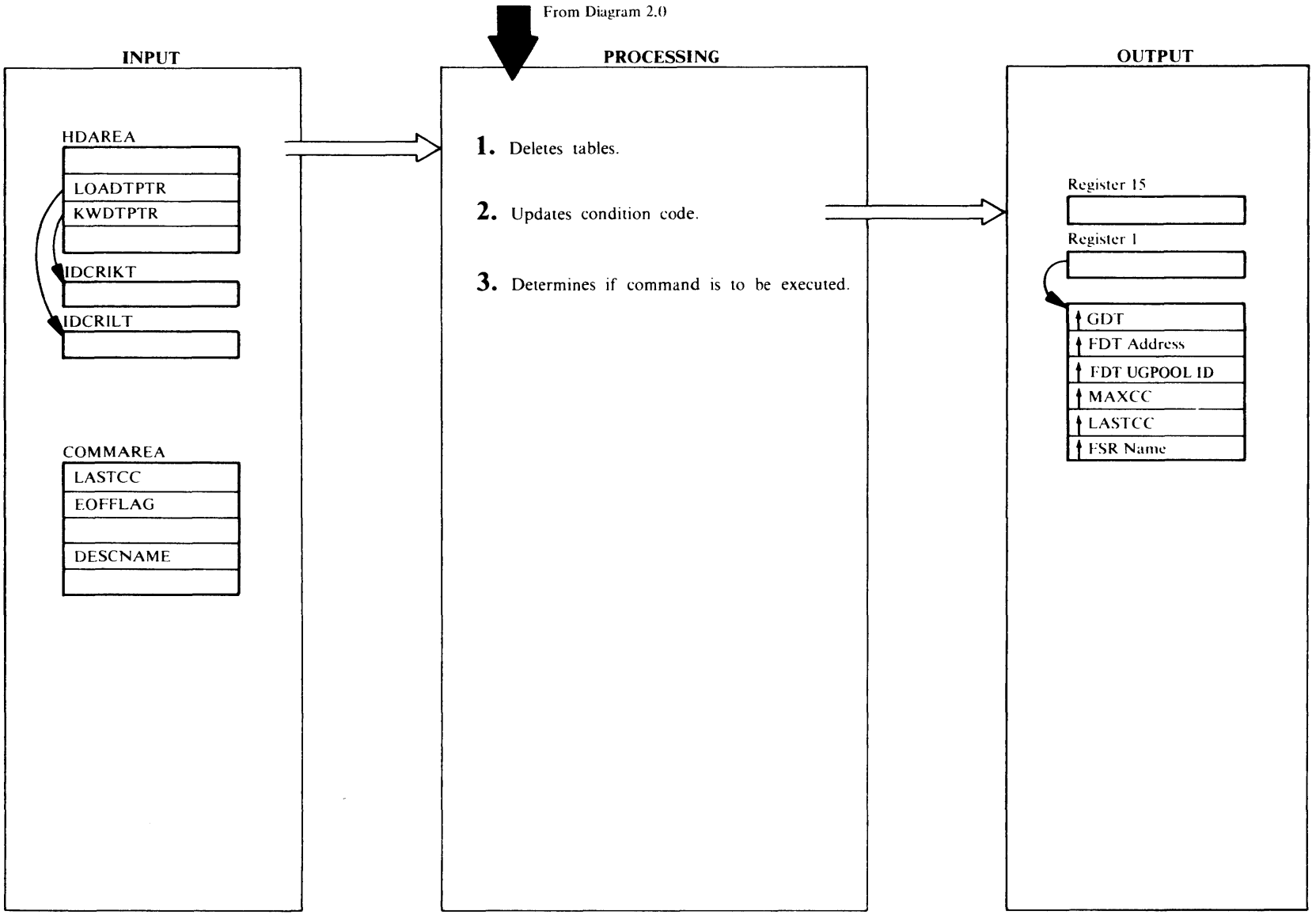
VERB A((B(C D((*x y*) ...)) E) ...) F

The type 1 parameters are B, C, D, and E because the entire parameter (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 2.4 for the next parameter.

Diagram 2.5 Reader/Interpreter Termination



Extended Description for Diagram 2.5

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.

IDCRI03

Procedure: IDCRI01, IDCRI03

2 If end-of-file is encountered on SYSIPT, IDCRI03 sets a flag in COMMAREA and IDCRI01 puts a nonzero value in register 15, indicating that the Executive is not to call the 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 SYSLST. IDCRI03 sets MAXCC to 16 which indicates that the Executive is not to call the R/I again.

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 2.2 to get the next command from SYSIPT. If the command is to be executed or severe errors were encountered, IDCRI01 gives control to Executive Controlled Termination Diagram 4.1.

Function Support Routine (FSR) Visual Table of Contents

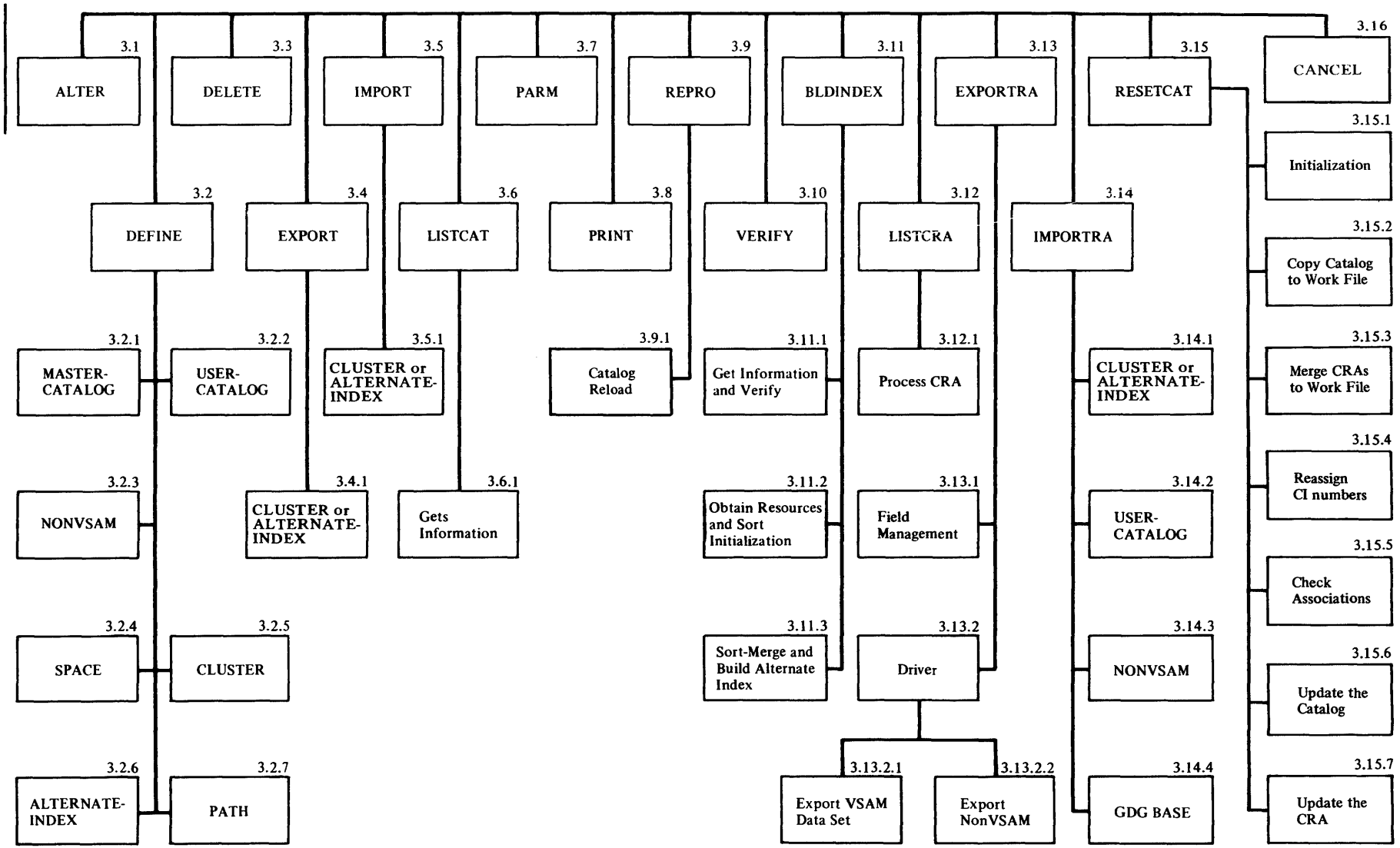
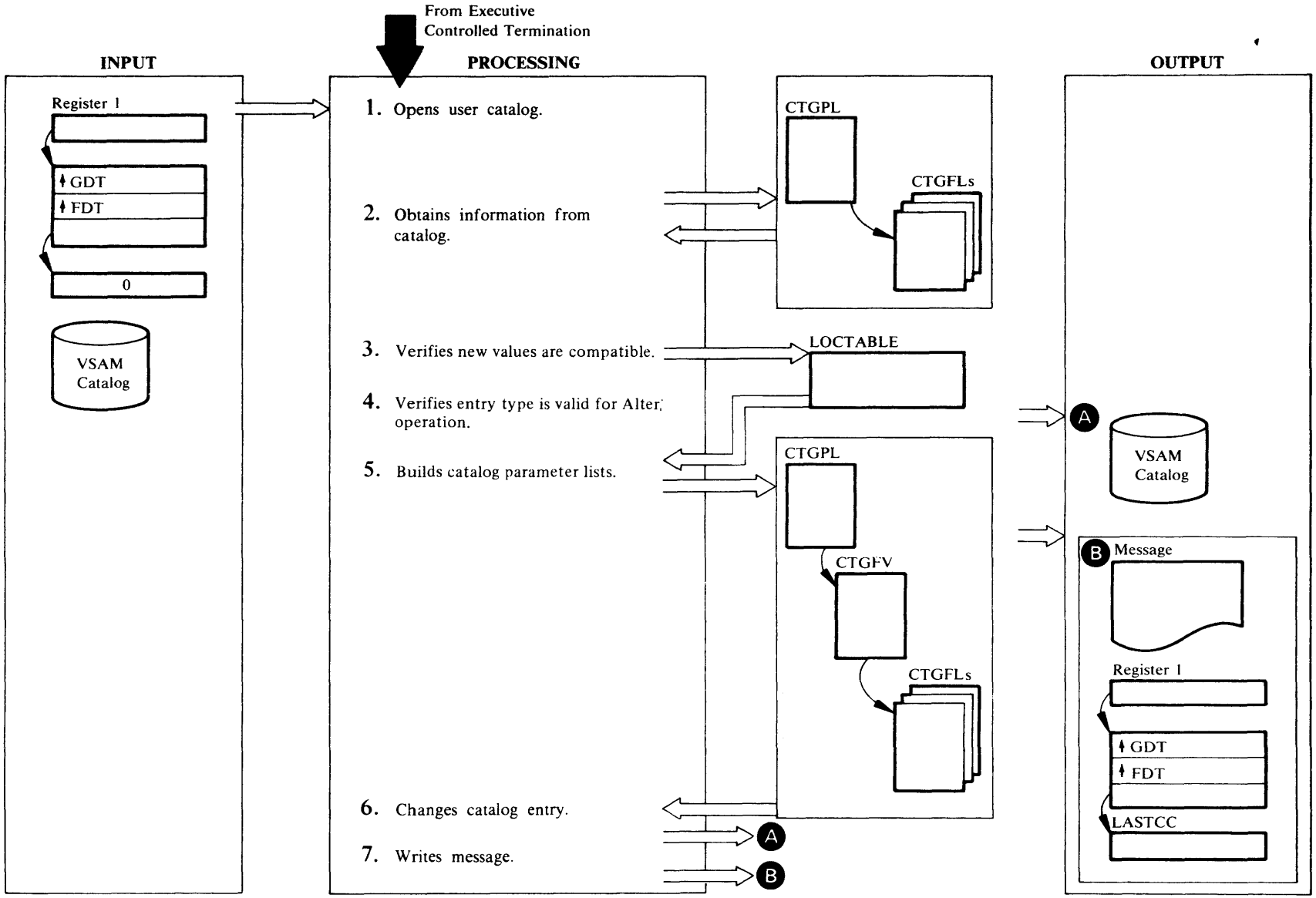


Diagram 3.1. ALTER FSR



Extended Description for Diagram 3.1

IDCAL01

Procedure: IDCAL01

1 First, IDCAL01 gets storage for the catalog parameter list. If a VSAM catalog is specified on the ALTER command, IDCAL01 builds an OPNAGL and issues a UOPEN to open the catalog. UOPEN returns the address of the catalog ACB. If the open is not successful, the ALTER command is terminated, and control goes to Step 7. If a catalog dname is passed, IDCAL01 compares the data set name returned from UOPEN (in IOCDSN) to that specified in the CATALOG parameter. If the compare is unequal, a message is written, the command is terminated and control goes to Step 7.

If an attempt is made to rename a reserved default model or rename an object to a reserved default model name, an error message is issued and the command is terminated. A reserved default model name is any name that begins with "DEFAULT.MODEL."

IDCAL01

Procedure: LOCATPRC

2 Due to the arrangement of 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, BUFFERSIZE, EXCEPTIONEXIT|NULLIFY EXCEPTIONEXIT, NOUPGRADE|UPDATE|NOUPDATE, or ADDVOLUMES|REMOVEVOLUMES is specified, control goes to Step 5. LOCATPRC builds a CTGPL and CTGFLs which reference the PASSWALL, DSATTR, AMDSBCAT, RGATTR, NAMEDS, HURBADS, ENTYPE and CATACB catalog fields. This initial locate performed in LOCATPRC is termed the *primary* locate.

A test is built to limit the number of associations returned for NAMEDS to a maximum of five. Refer to the list in Step 5 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 nonzero number, the ALTER command is terminated and control goes to Step 7.

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/or RECORDSIZE are not specified:

a. On the primary locate, if a password is supplied, reference it from the CPL. Set the verify master password bit.

b. If UPGRADE is specified, a secondary locate for the data HURBADS is required. If a password is supplied, reference it from the CPL. Turn off the verify master password bit. 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, reference it from the CPL. Turn off the verify master password bit. Password verification will be as in prior release (master password of catalog or entry being altered).

If KEYS and/or RECORDSIZE are specified:

a. On the primary locate, if a password is supplied, reference it from the CPL. Set the verify master password bit.

b. On the secondary locates, if a password is supplied, reference it from the CPL. Turn off the verify master password bit. Turn on the bypass verification bit. 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 will verify that the ENTYPE is a G (alternate index). If UPGRADE was specified, CHECKPRC will verify 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 will verify 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.

The major portion of the new CHECKPRC procedure will perform the validity checking required to alter the KEYS and/or RECORDSIZE values of an empty data set. This checking will require 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

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

After retrieval of the appropriate AMDSBCATs, the following check will be 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 will return to IDCAL01 with a terminating condition code.

If this is an alteration of an ESDS the index validity check will not be 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 must be made that requires retrieving the AMDSB of the base cluster's data component. The table below shows the locates that CHECKPRC will issue 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

Using the base cluster's data AMDSB, CHECKPRC will verify 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 will return to IDCAL01 with a terminating error.

Assuming no terminating errors have been found, CHECKPRC will now set 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 KEYS and/or RECORDSIZE values were default values	4	0
Previous KEYS and/or RECORDSIZE values were not default values	4	12

If the return code is 0, the alter will be performed. If the return code is 4, KEYS and RECORDSIZE will not be altered but alters will be performed for any other parameters specified. A return code of 12 is treated as a terminating condition code. If the verification of the new values fails, the return code is 12.

Control is returned to IDCAL01.

IDCAL01

Procedures: 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 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. An error is indicated if the ERASE, WRITECHECK, EXCEPTIONEXIT, or BUFFERSIZE option is specified for a SAM ESDS in NOCIFORMAT. If PARAMCHK detects an error, control goes to step 7, otherwise, control goes to step 5.

IDCAL01

Procedure: ALTERPRC

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. The following table lists the data areas that pass information to VSAM and the keywords whose data is passed.

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

If KEYS or RECORDSIZE was specified, CHECKPRC has saved the control interval number of the data component being altered. This number is moved to the CPL and is used instead of the data component name for faster access.

Prior to IDCAL01 issuing the UCATLG macro the CTGFVTYP field will be set to G if UPGRADE/NOUPGRADE is specified.

CTGFVTYP will be set to R if UPDATE/NOUPDATE is specified.

IDCAL01

Procedure: IDCAL01

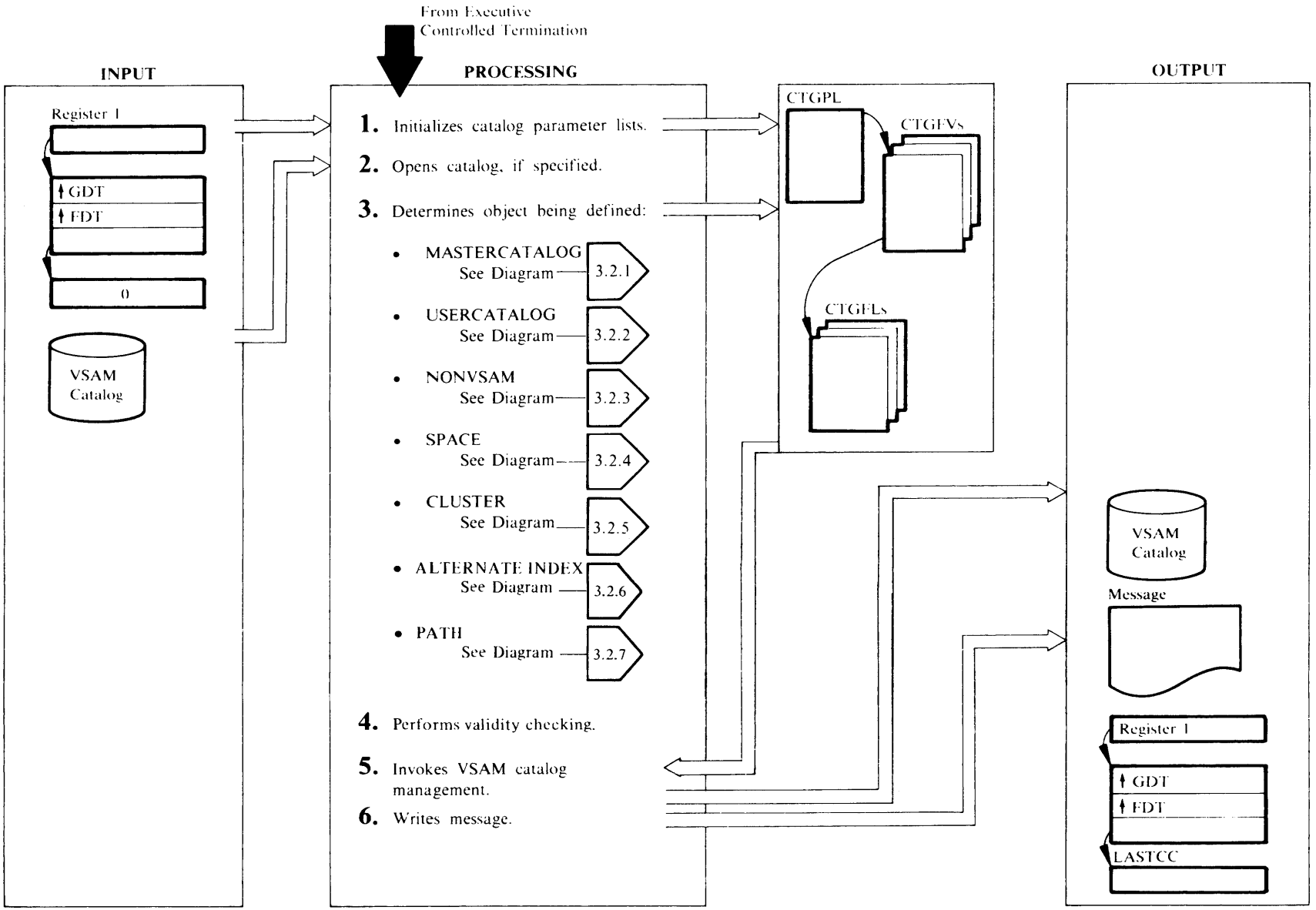
6 IDCAL01 issues a UCATLG macro to change the catalog entry. If the return code from UCATLG is nonzero, an error conversion table is built and a call is made to UERROR. UERROR will handle printing of the error message. If KEYS is specified for a KSDS or an alternate index, a second UCATLG macro is issued to change the catalog entry of the associated index object. If the return code is nonzero, it builds an error conversion table and calls UERROR. UERROR will handle the printing of the error message.

IDCAL01

Procedure: IDCAL01

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

Diagram 3.2. DEFINE FSR



Extended Description for Diagram 3.2

IDCDE01

Procedure: IDCDE01

1 IDCDE01 issues a UGPOOL macro to obtain core for a CTGPL, four CTGFVs and two work areas. One work area is used by catalog management during its processing. The second is used by catalog management to return the volume serial of the recovery volume for the object defined if the catalog is recoverable. 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 the VSE/VSAM LOGIC, Volume 1, 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 preceed the CTGPL. A call is made to IECDE02 to establish addressability for IDCDE02 to declarations common to all DEFINE modules. If a *catname* is supplied with a CATALOG parameter, IDCDE01 puts the address of the *catname* and the *password* in the CTGPL.

IDCDE01

Procedure: IDCDE01

2 If the CATALOG parameter specifies a *dname*, IDCDE01 opens the catalog with a UOPEN macro. If the return code from UOPEN is zero, IDCDE01 compares the data set name returned from UOPEN (in IOCDSN) to that specified in the CATALOG parameter. If the compare is unequal, a message is written and control goes to Step 6. The I/O Adapter returns the address of the ACB for the catalog in the IOCSTR. IDCDE01 puts the address of the ACB in the CTGPL. IDCDE01 puts the address of the catalog ACB in the same CTGPL field where the address of the *catname* was placed. The ACB is used instead of the name for faster catalog access by VSAM catalog management. If the return code from the UOPEN is nonzero, a message is written with a UPRINT macro and control goes to step 6. Otherwise, IDCDE01 calls IDCDE03 to format the catalog parameter list.

IDCDE03

Procedure: IDCDE03

3 IDCDE01 determines the type of DEFINE by testing for the following keywords: CLUSTER, MASTERCATALOG, USERCATALOG, NONVSAM, SPACE, ALTERNATEINDEX, PATH. The types of DEFINE are shown in detail in the following diagrams:

MASTERCATALOG see Diagram 3.2.1
USERCATALOG see Diagram 3.2.2
SPACE see Diagram 3.2.3
NONVSAM see Diagram 3.2.4
CLUSTER see Diagram 3.2.5
ALTERNATEINDEX see Diagram 3.2.6.
PATH see Diagram 3.2.7.

IDCDE01

Procedure: INTGCHK

4 INTGCHK performs validity checking to insure:

KSDS, ESDS, RRDS, and AIX

- Space parameters have been properly specified.
- User is warned when USECLASS has been ignored due to the absence of space parameters at the same level.
- Volumes have been specified in both DATA and INDEX FVTs.
- If KEYLENGTH and KEY POSITION (in Data AMDSB) have not been specified supply defaults: length=64, relative key position=0.
- 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 UNIQUE is specified insure CTGFVIND *(dname)* has been set and build null volume FVT.
- UNIQUE was not specified with a USECLASS other than zero.
- RECORDSIZE was omitted with RECORDFORMAT FIXUNB or FIXBLK
- If NOALLOCATION was specified for a KSDS/AIX, it should be specified in both the DATA and INDEX components.
- If a default model has been defined, it should have the NOALLOCATE attribute and should always have a volume list.
- If the DEFAULTVOLUMES parameter is ignored due to override by the VOLUMES parameter, a warning message is given.
- RECORDFORMAT was not specified with INDEXED, NUMBERED, SPANNED, or RECOVERY.
- NOCIFORMAT was not specified with WRITECHECK, ERASE, or EXECPTIONEXIT.
- A component with ORDERED attributes has a volume list.
- If an ESDS, KSDS or AIX has the REUSABLE attribute make sure it is not unique nor have KEYRANGES been specified.
- If AMDRRDS indicates an RRDS, insure that the average and maximum LRECL are equal.
- If the data AMDSB indicates an RRDS, insure that it does not also indicate spanned.
- If record size is greater than 32,761 (maximum CI size), insure that it has the spanned attribute.
- If KEYRANGES is specified, ensure key values do not exceed maximum key length.
- Because USECLASS is effective only when space parameters (for example, CYL) are specified, modeled, or propagated at the same level, INTGCHK performs the final audit and application of USECLASS to the SPACPARM CTGFLs.

SPACE

- Space parameters have been properly specified.
- Because USECLASS is effective only when space parameters (for example, CYL) are specified, modeled or propagated at the same level, INTGCHKperforms the final audit and application of USECLASS to the SPACPARM CTGFLs.

IDCDE01

Procedure: IDCDE01

5 IDCDE01 invokes VSAM catalog management by issuing a UCATLG macro. If a nonzero catalog management return code is received, and if it relates to volume allocation status, a UPRINT macro lists the volumes associated with the error conditions.

For allocation of space on a fixed block device, a UPRINT macro prints specific extents to indicate possible rounding of actual extents to conform to device characteristics.

If a list of names is returned, the list is written with a UPRINT macro. If the return code from UCATLG is nonzero, IDCDE01 builds an error conversion table and

invokes UERROR. UERROR will handle printing of the error message.

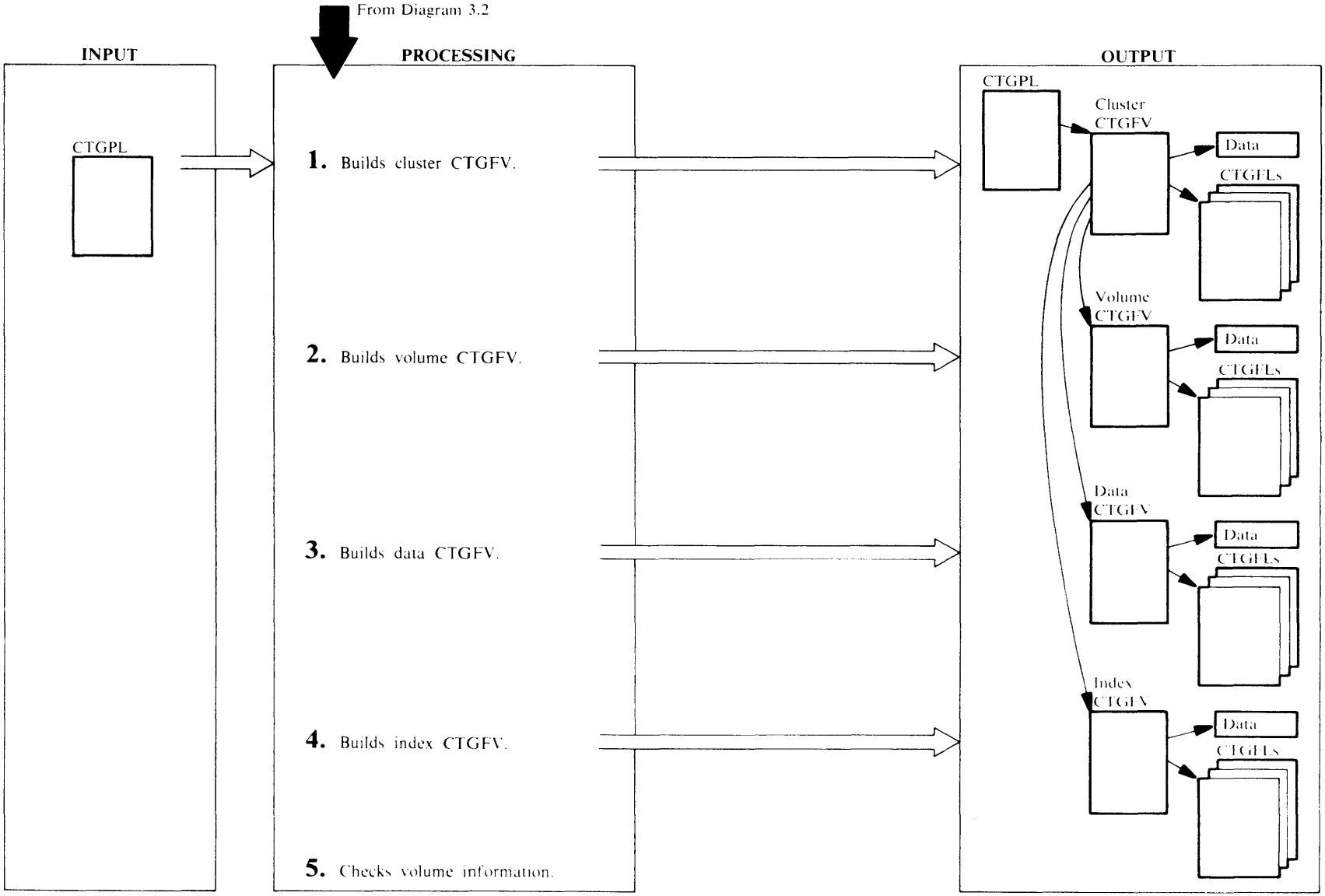
If a recovery volume serial is returned, it is printed with a UPRINT macro.

IDCDE01

Procedures: IDCDE01

6 If a catalog was opened in step 2, IDCDE01 closes the catalog with a UCLOSE macro. A message with LASTCC is written with a UPRINT macro. IDCDE01 calls FREESTG to free all automatic storage for CSECT IDCDE02. IDCDE01 issues a UFPOOL to free all the storage obtained for the DEFINE FSR. Control goes to Executive Controlled Termination.

Diagram 3.2.1. DEFINE FSR – DEFINE MASTERCATALOG

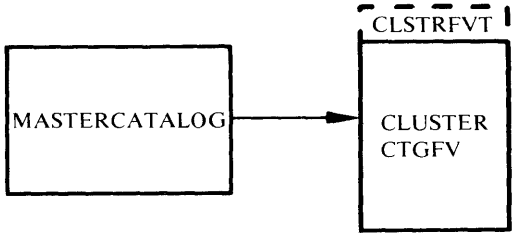


Extended Description for Diagram 3.2.1

IDCDE02, IDCDE03

Procedures: CTLGPROC, ALLCPROC, NAMEPROC, PROTPROC

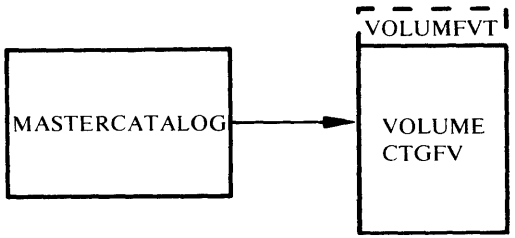
1 In the DEFINE MASTERCATALOG command, you specify information under three main keywords: MASTERCATALOG, DATA, and INDEX. The DEFINE FSR builds a CTGFV to describe the cluster, data and index components of the mastercatalog as well as building a volume CTGFV. Information specified under MASTERCATALOG 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 MASTERCATALOG completes the DATA or INDEX CTGFV. If information is duplicated under DATA or INDEX and under MASTERCATALOG—like WRITECHECK—information from DATA or INDEX overrides the information from MASTERCATALOG in the DATA or INDEX CTGFV. The exception is space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS. Space information is never copied from MASTERCATALOG to the DATA and INDEX CTGFVs. CTLGPROC sets the identification of CLSTRFVT in the 8 bytes before the CLUSTER CTGFV. An "M" is set in the CTGTYPE field in the CTGPL to indicate that a master catalog is being defined. CTLGPROC puts the address of the *objectname* from NAME in the CLUSTER CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS along with DEDICATE and CLASS indicators. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field of the cluster FVT. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. NAMEPROC also 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 information from OWNER.



IDCDE02, IDCDE03

Procedures: CTLGPROC, ALLCPROC

2 The DEFINE FSR builds a VOLUME CTGFV with information specified under MASTERCATALOG. CTLGPROC sets the identification of VOLUMFVT in the 8 bytes preceding the VOLUME CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS along with DEDICATE and CLASS indicators. ALLCPROC puts the address of *volser* from VOLUME and the address of *dname* if specified from FILE in the VOLUME CTGFV.



IDCDE02, IDCDE03

Procedures: CTLGPROC, NAMEPROC, KEYPROC, ALLCPROC

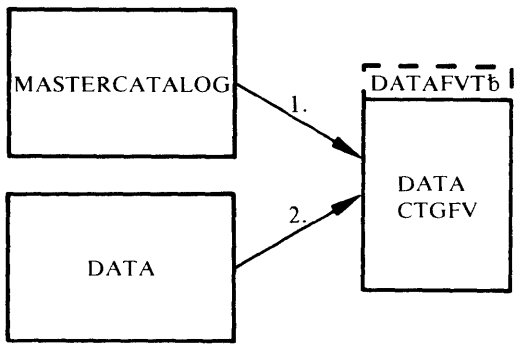
3 CTLGPROC sets the identification of DATAFVT in the 8 bytes preceding the DATA CTGFV. The DEFINE FSR builds the DATA CTGFV with information specified under MASTERCATALOG and under DATA. If information is duplicated under MASTERCATALOG and under DATA, the information in DATA overrides information from MASTERCATALOG. The DEFINE FSR first puts the information from MASTERCATALOG in the DATA CTGFV; second, information from DATA is put in the DATA CTGFV overriding anything already in the DATA CTGFV.

First, the information under MASTERCATALOG is put in the DATA CTGFV as follows:

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* if specified from VOLUME and the address of *dname* if specified from FILE in the DATA CTGFV. WRITECHECK|NOWRITECHECK is put in the AMDSBCAT CTGFL. ALLCPROC builds a BUFSIZE CTGFL with information from BUFFERSPACE. ALLCPROC builds a DSATTR CTGFL for data set attributes and, in addition, sets the Recoverable or Not Recoverable indicator in DSATTR. In the listings this is called the implicit pass.

Second, the information under DATA is put in the DATA CTGFV as follows:

ALLCPROC builds a SPACPARM CTGFL for primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS. The value specified for CLASS is also set in the SPACEPARM CTGFL. ALLCPROC initializes the Recoverable/Not Recoverable flag in the DSATTR CTGFL. IfWRITECHECK|NOWRITECHECK 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. In the listings this is called the explicit pass.



IDCDE02, IDCDE03

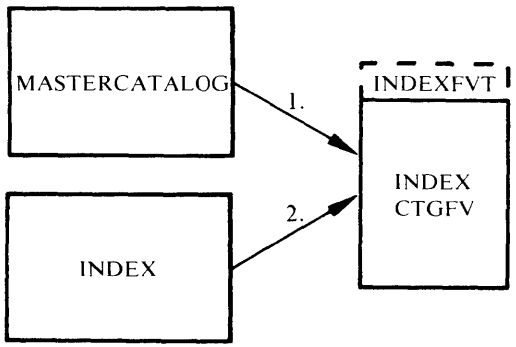
Procedures; CTLGPROC, NAMEPROC, KEYPROC, IXOPPROC, ALLCPROC

4 CTLGPROC sets the identification of INDEXFVT in the 8 bytes preceding the INDEX CTGFV. The DEFINE FSR builds the INDEX CTGFV with information specified under MASTERCATALOG and under INDEX. If information is duplicated under MASTERCATALOG and under INDEX, the information in INDEX overrides information from MASTERCATALOG. The DEFINE FSR first puts the information form MASTERCATALOG in the INDEX CTGFV; second, information from INDEX is put in the INDEX CTGFV overriding anything already in the INDEX CTGFV. First, the information under MASTERCATALOG is put in the INDEX CTGFV as follows:

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. In IXOPPROC, IMBED|NOIMBED is put into the AMDSB. ALLCPROC puts the address of the *volser* from VOLUME and the address of *dname* if specified from FILE in the INDEX CTGFV. WRITECHECK|NOWRITECHECK is put in the AMDSBCAT CTGFL. ALLCPROC builds a DSATTR CTGFL for data set attributes. In the listings this is called the implicit pass.

Second, the information under INDEX is put in the INDEX CTGFV as follows:

ALLCPROC builds a SPACPARM CTGFL for primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS. The value specified for CLASS is also set in the SPACPARM CTGFL. WRITECHECK|NOWRITECHECK is overridden in the AMDSBCAT CTGFL. ALLCPROC initializes the Recoverable/Not Recoverable flag in the DSATTR CTGFL. In the listings this is called the explicit pass.



IDCDE01

Procedure: INTGCHK

5 For MASTERCATALOG four CTGFV's have been built: one 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. The following table shows the possible CTGFVs where a SPACPARM CTGFL may have been built (in addition to the VOLUME CTGFV) and the action INTGCHK takes.

SPACPARM CTGFL

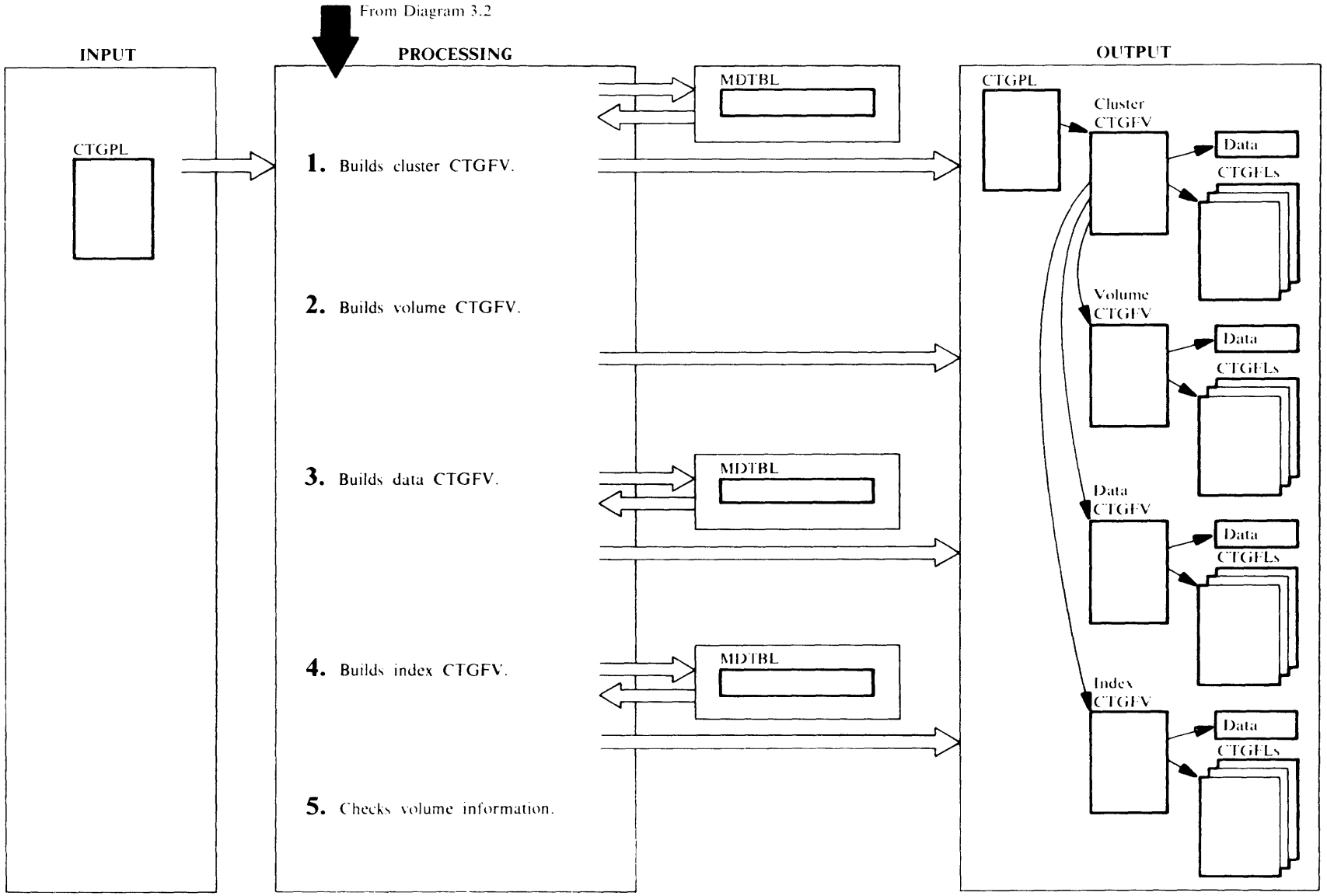
Cluster	Data	Index	Action
X	X	X	IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
X	X		IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
X		X	This is an error; IDCDE01 terminates the DEFINE.
X			OK; no action.
none	none	none	This is an error; IDCDE01 terminates the DEFINE.

INTGCHK insures that space parameters exist wherever CLASS|USECLASS has been specified (or internally generated). If space parameters do not exist, CLASS|USECLASS is dropped from the SPACPARM CTGFL.

Note that for DEFINE MASTERCATALOG, primary useclass is not specified explicitly; it is logically generated at the data and index levels to agree with the value established for CLASS. Secondary useclass is always the same as primary useclass.

The SPACPARM CTGFL is checked for a *dname* from FILE. Control goes to Diagram 3.2, step 4. If an error occurs, INTGCHK writes a message and control goes to step 6.

Diagram 3.2.2. DEFINE FSR – DEFINE USERCATALOG



Extended Description for Diagram 3.2.2

IDCDE02, IDCDE03

Procedures: CTLGPROC, NAMEPROC, MODELPRC, PROTPROC, ALLCPROC

1 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 usercatalog 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, BLOCKS, or RECORDS along with DEDICATE and CLASS indicators. 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

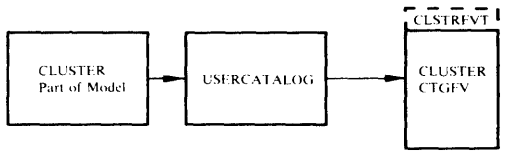
   DATA CTGFV

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

   INDEX CTGFV

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

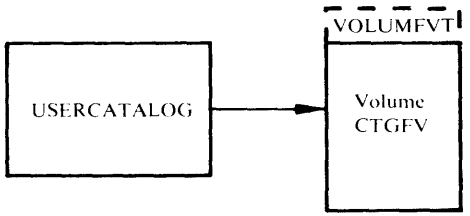
   CTLGPROC sets the identification of CLSTRFVT in the 8 bytes before the CLUSTER CTGFV. A U is put in the CTGTYPE field of the CTGPL to indicate that a user catalog is being defined. CTLGPROC puts the address of the *objectname* from NAME in the CLUSTER CTGFV. CTLGPROC checks for a MODEL keyword. If MODEL is specified, MODELPRC issues a UCATLG macro to retrieve information from the modeled catalog. The information from the cluster catalog entry of the modeled catalog is put in a table, MDLTABL, and the Control Interval number for the data and index entries of the modeled 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 a OWNERID CTGFL with *ownerid* from OWNER. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, and RECORDS along with DEDICATE and CLASS indicators. NAMEPROC issues a UTIME macro to get the creation date which is put in a DSETCRDT CTGFL. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field of the cluster FVT.



IDCDE02, IDCDE03

Procedures: 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 8 bytes preceding the VOLUME CTGFV. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS along with DEDICATE and CLASS indicators. ALLCPROC puts the address of *volser* from VOLUMES and the address of *dname* if specified from FILE in the VOLUME CTGFV.



IDCDE02, IDCDE03

Procedures: CTLGPPROC, NAMEPROC, KEYPROC, ALLCPROC, MODELPRC

3 CTLGPROC 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 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, the following occurs:

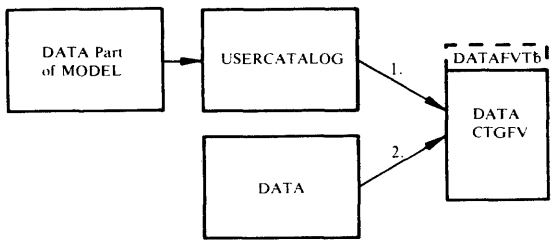
   If MODEL is not specified, the DATA CTGFV is built with information specified in the USERCATALOG parameters.

   If MODEL is specified, MODELPRC uses the saved Control Interval number for the data entry of the modeled catalog to get information from the dataentry. The information from the data entry of the modeled 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 a AMDSBCAT CTGFL, but no information is put in yet. ALLCPROC puts the address of the *volser* if specified from VOLUME and the address of *dname* if specified from FILE in the DATA CTGFV. WRITECHECK|NOWRITECHECK is put in the AMDSBCAT CTGFL. ALLCPROC builds a BUFSIZE CTGFL with information from BUFFERSPACE. ALLCPROC builds a DSATTR CTGFL for data set attributes and, in addition, sets the Recoverable/Not Recoverable flag of the field.

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.

If a DSETCRDT CTGFL does not exist, NAMEPROC builds one. Normally, a DSETCRDT CTGFL does exist. ALLCPROC builds a SPACPARM CTGFL for primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS. The value specified for CLASS is also set into the SPACPARM CTGFL. If WRITECHECK|NOWRITECHECK 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 flag in the DSATTR CTGFL.



IDCDE02, IDCDE03

Procedures: CTLGPROC, NAMEPROC, KEYPROC, IXOPPROC, ALLCPROC, MODELPRC

4 CTLGPROC 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 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, the following occurs:

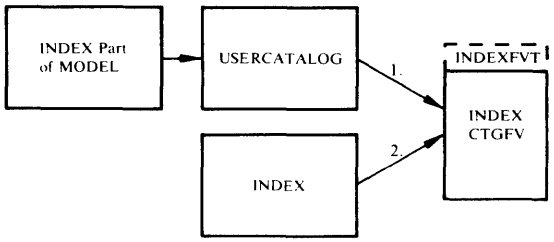
If MODEL is not specified, the INDEX CTGFV is built with information specified in USERCATALOG parameters.

If MODEL is specified, MODELPRC uses the saved Control Interval number for the index entry of the modeled catalog to get information from the index entry. The information from the index entry of the modeled 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. In IXOPPROC, IMBED|NOIMBED is put into the AMDSBCAT CTGFL. ALLCPROC puts the address of the *volser* from VOLUME and the address of *dname* if specified from FILE in the INDEX CTGFV. WRITECHECK|NOWRITECHECK is 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, BLOCKS, or RECORDS. The value specified for CLASS is also set into the SPACPARM CTGFL. WRITECHECK|NOWRITECHECK is overridden in the AMDSBCAT CTGFL.



IDCDE01

Procedure: INTGCHK

5 For USERCATALOG four CTGFVs have been built - one 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. 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:

SPACEPARM CTGFL

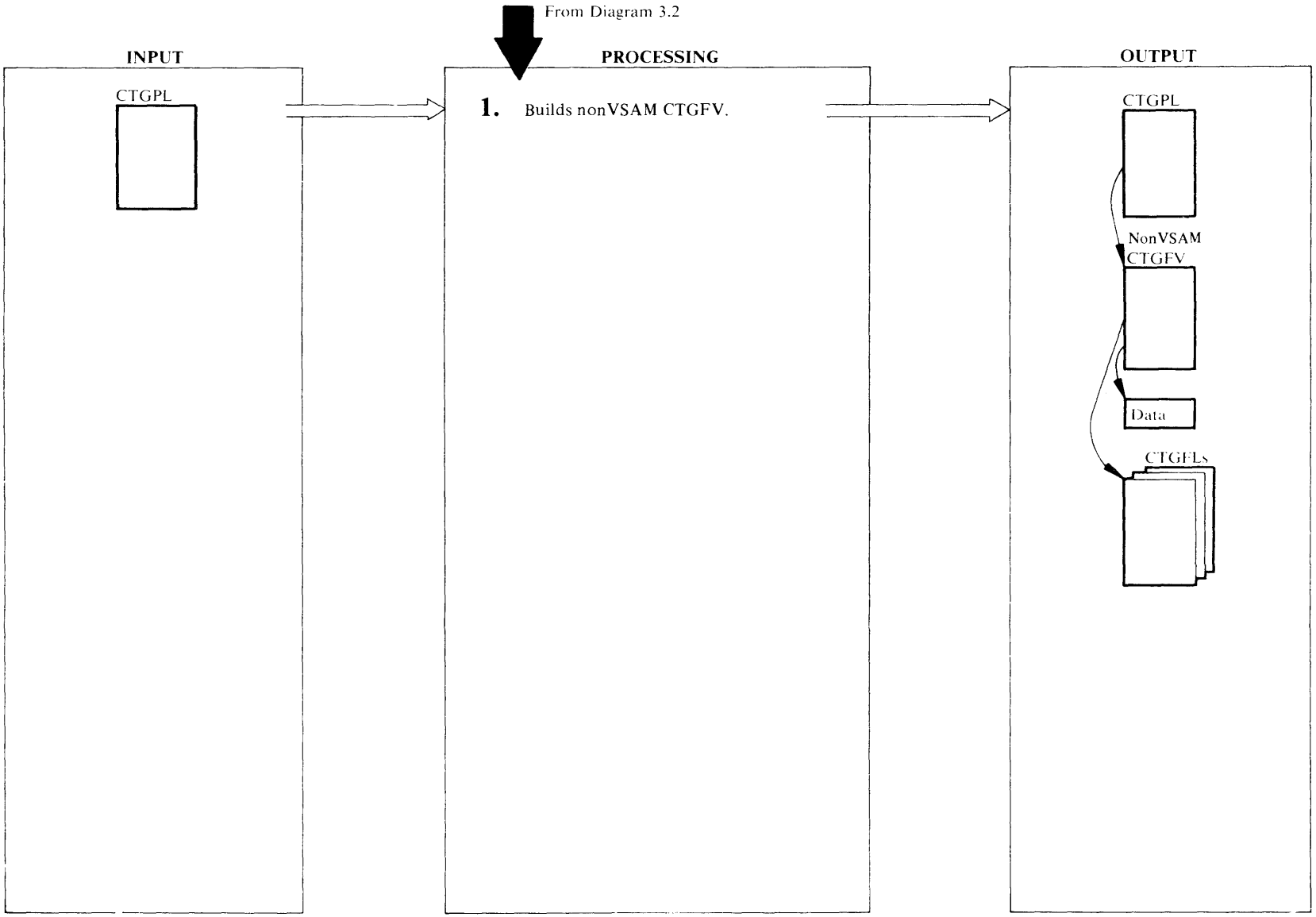
Cluster	Data	Index	Action
X	X	X	IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
X	X		IDCDE01 erases the SPACPARM CTGFL from the CLUSTER CTGFV.
X		X	This is an error; IDCDE01 terminates the DEFINE.
X			OK; no action.
none	none	none	This is an error; IDCDE01 terminates the DEFINE.

INTGCHK insures that space parameters exist wherever CLASS|USECLASS has been specified (or internally generated). If space parameters do not exist, CLASS|USECLASS is dropped from the SPACPARM CTGFL.

Note that for DEFINE USERCATALOG, primary useclass is not specified explicitly; it is logically generated at the data and index levels to agree with the value established for class. Secondary useclass is always the same as primary useclass.

The SPACPARM CTGFL is checked for a *dname* from FILE. Control goes to Diagram 3.2, step 4. If an error occurs, INTGCHK writes a message and control goes to Diagram 3.2, step 5.

Diagram 3.2.3. DEFINE FSR-DEFINE NONVSAM



Extended Description for Diagram 3.2.3

IDCDE02, IDCDE03

Procedures: NVSAMPRC, ALLCPROC, PROTPROC, NAMEPROC

1 NVSAMPRC sets the identification of NVSAMFVT in the 8 bytes preceding the area that is usually used for a CLUSTER CTGFV. 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. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field. Control goes to Diagram 3.2, step 4.

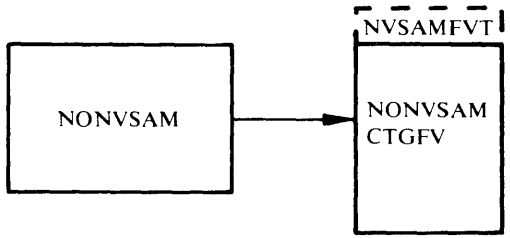
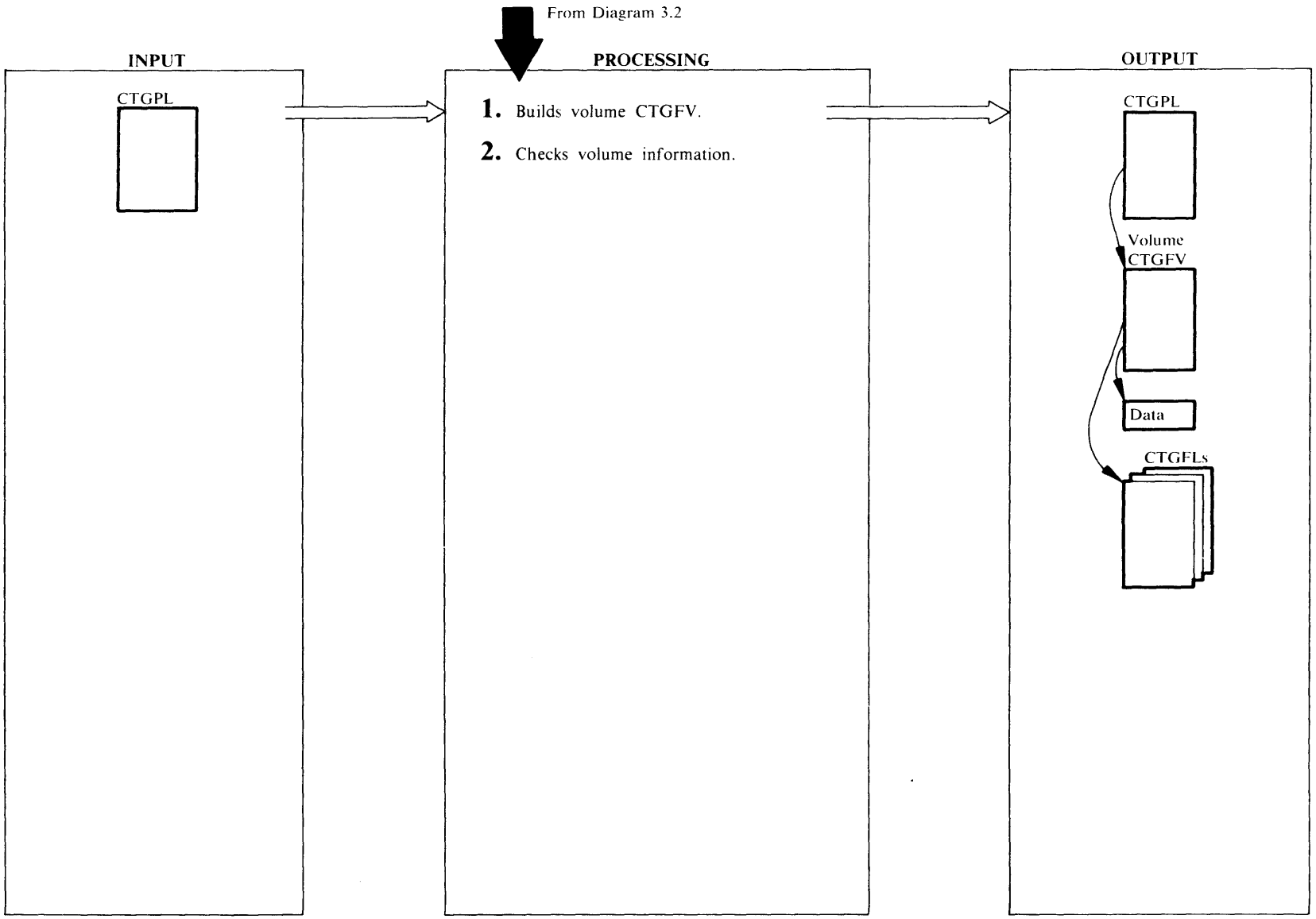


Diagram 3.2.4. DEFINE FSR – DEFINE SPACE



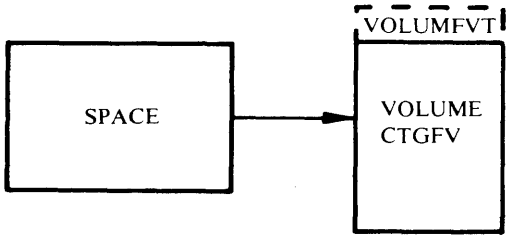
Extended Description for Diagram 3.2.4

IDCDE02, IDCDE03

Procedures: DSPACPRC, ALLCPROC

1 DSPACPRC sets the identification of VOLUMFVT in the 8 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* if specified from VOLUMES and the address of *dname* if specified from FILE in the VOLUME CTGFV. ALLCPROC builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS along with DEDICATE and CLASS indicators.

   If RECORDS is specified, ALLCPROC builds a LRECL CTGFL with information from RECORDSIZE. ALLCPROC sets the address of the recovery volume serial work area in the CTGFVWKA field of the volume FVT.

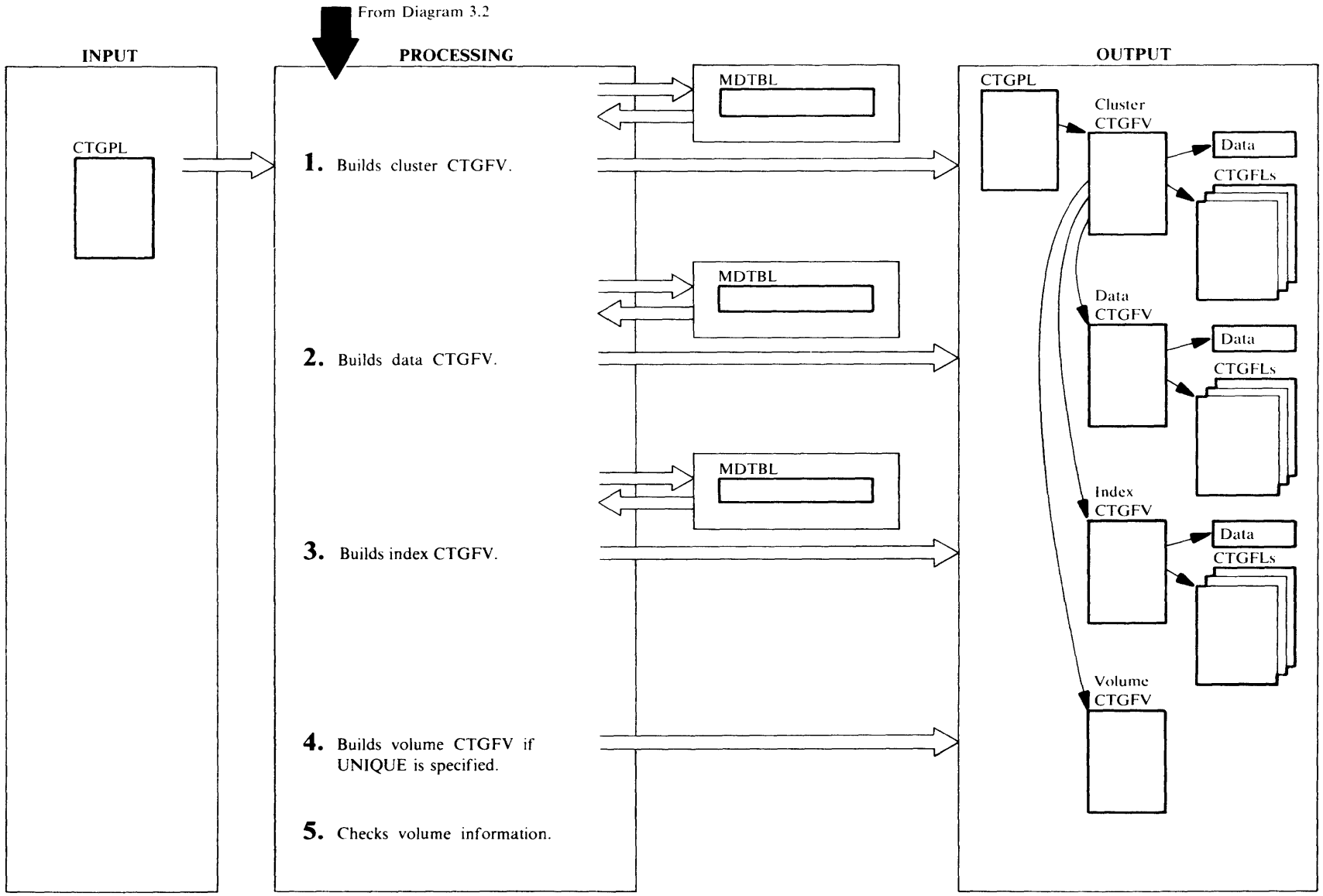


IDCDE01

Procedures: INTGCHK

2 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.

Diagram 3.2.5. DEFINE FSR – DEFINE CLUSTER



Extended Description for Diagram 3.2.5

IDCDE02, IDCDE03

Procedures: 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, BLOCKS, or RECORDS. This space information is never copied from CLUSTER.

Both explicit (MODEL parameter) and implicit (default) modeling are supported, but for any one component (CTGFVT), explicit and implicit modeling cannot be mixed, i.e., explicit models preclude implicit models.

If MODELs are applied, 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 for any particular parameter when modeling is shown below where 1 takes the highest precedence:

CLUSTER CTGFV

1. CLUSTER parameters
2. Cluster object of CLUSTER explicit or default model
3. System default

DATA CTGFV

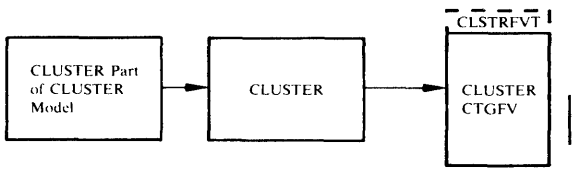
1. DATA parameters
2. DATA explicit model
3. CLUSTER parameters
4. Data object of CLUSTER explicit or default model
5. System default

INDEX CTGFV

1. INDEX parameters
2. INDEX explicit model
3. CLUSTER parameters
4. Index object of CLUSTER explicit or default model
5. System default

If MODEL is applied, MODELPRC issues a UCATLG to retreive 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 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 is then overlaid by information specified in the CLUSTER parameters.

DSETPROC sets the identification of CLSTRFVT in the 8 bytes before the CLUSTER CTGFV. DSETPROC also sets the address of the recovery volume serial work area in the CTGFVWKA field. 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. If a reserved name (default model name) prefix ("DEFAULT.MODEL.") is used, a check is made for additional valid qualifiers. PROTPROC builds a PASSWALL CTGFL with information from MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION. PROTPROC also builds a OWNERID CTGFL with information from OWNER. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS, along with USECLASS.



IDCDE02, IDCDE03

Procedures: DSETPROC, NAMEPROC, KEYPROC, MODELPRC, ALLCPROC, PROTPROC

2 DSETPROC sets the identification of DATA FVT in the 8 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, the following occurs:

If MODEL is not specified at the data level, the DATA CTGFV is built with information specified in the CLUSTER parameters.

If MODEL is applied under CLUSTER or a default model exists for the cluster type (KSDS, RRDS, VSAM ESDS, SAM ESDS) and MODEL is not specified under DATA, MODELPRC uses the saved Control Interval number for the data entry of the applicable 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. If the DEFAULTVOLUMES parameter is given at either the CLUSTER or the DATA level, nullify the volumes list pointer in the 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 also 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 *offset* from KEYS in the AMDSBCAT CTGFL. If no key values are specified, KEYPROC sets up default values. In addition, KEYPROC sets an indication in the AMDSB if SPANNED has been specified. KEYPROC also puts the address of *(lowkey highkey)...* from KEYRANGES in the DATA CTGFV. If NUMBERED has been specified, KEYPROC sets AMDRRDS in the AMDSB field. This FPL is being built by KEYPROC. ALLCPROC puts the address of *dname* from FILE and the address of *volser* from VOLUMES in the DATA CTGFV. Volumes are not taken from the default model. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS, along with USECLASS. ALLCPROC also builds a BUFSIZE CTGFL with

information from BUFFERSPACE. The following are inserted by ALLOCPROC and PROTPROC:

ORDERED|UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK|NOWRITECHECK
RECORDFORMAT
maximum from RECORDSIZE are put in the AMDSBCAT CTGFL

UNIQUE|SUBALLOCATION|NOALLOCATION and SPEED|RECOVERY are put in the DSATTR CTGFL. ERASE|NOERASE and DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS are put in the DSATTR CTGFL.

Protection information is obtained only from the explicit MODEL via MDLTABL in order to provide different protection at the CLUSTER and DATA. PROTPROC builds a PASSWALL CTGFL with protection information from the MODEL as well as an 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, the following occurs:

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.

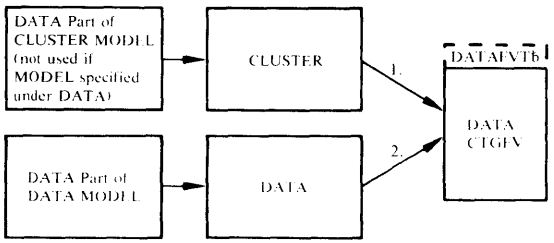
If MODEL is applied under DATA or a default model exists, MODELPRC issues a UCATLG to get information from the data catalog entry of the modeled data set. The information from the data entry of the modeled data set is put in MDLTABL. If the DEFAULTVOLUMES parameter is given at either the CLUSTER or DATA level, nulify the volume list pointer in the 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. If a reserved name was used at the CLUSTER level ("DEFAULT.MODEL." prefix), the DATA qualifier is added from the data component and this constructed name is forced. 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 in the AMDSB to indicate the SPANNED|NONSPANNED option.

KEYPROC puts *length* and *offset* from KEYS in the AMDSBCAT CTGFL. KEYPROC puts the address of *(lowkey highkey)*... range 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, BLOCKS, or RECORDS, along with USECLASS; 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 are inserted by ALLCPROC and PROTPROC:

ORDERED|UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK|NOWRITECHECK
RECORDFORMAT
maximum from RECORDSIZE or put in the AMDSBCAT CTGFL

UNIQUE|SUBALLOCATION|NOALLOCATION and SPEED|RECOVERY are put in the DSATTR CTGFL. ERASE|NOERASE and DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS are put in the DSATTR CTGFL.



IDCDE02, IDCDE03

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

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

INDEXED is specified
NONINDEXED or 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.

DSETPROC 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 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, the following occurs:

If MODEL is not specified at the data level, the INDEX CTGFV is built with information specified in CLUSTER parameters.

If MODEL is specified under CLUSTER or a default model exists for the CLUSTER type (KSDS, RRDS, VSAM ESDS, SAM ESDS) and MODEL is not specified under INDEX, MODELPRC uses the saved Control Interval number for the index entry of the applicable modeled data set to get information from the index entry. The information from the index entry of the modeled data set is put in MDLTABL. If the DEFAULTVOLUMES parameter is given at either the CLUSTER or INDEX level, nullify the volume list pointer in the 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. 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. Volumes are not taken from the default model.

ALLCPROC also builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS,along with USECLASS. The following is put in the AMDSBCAT CTGFL:

ORDERED|UNORDERED
WRITECHECK|NOWRITECHECK
size from CONTROLINTERVALSIZE

UNIQUE|SUBALLOCATION|NOALLOCATION 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 explict 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, the following occurs:

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.

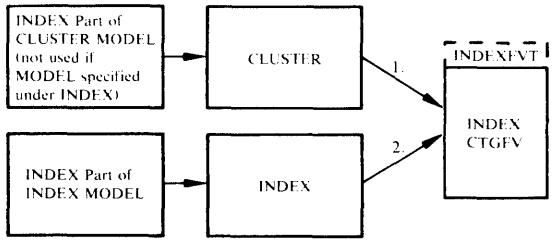
If MODEL is specified under INDEX or a default model exists, MODELPRC issues a UCATLG to get information from the index catalog entry of the modeled data set. The information from the index entry of the modeled data set is put in MDLTABL. If the DEFAULTVOLUMES parameter is given at either the CLUSTER or the INDEX level, nullify the volumes list pointer in the 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. If a reserved name was used at the CLUSTER level ("DEFAULT.MODEL." prefix), the INDEX qualifier is added for the INDEX component and this name is forced. 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, BLOCKS, or RECORDS, along with USECLASS. 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

The following is put in the DSATTR CTGFL:

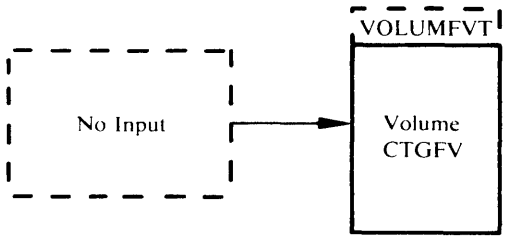
UNIQUE|SUBALLOCATION|NOALLOCATION
ERASE|NOERASE
DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS



IDCDE03

Procedures: DSETPROC, IDCDE01

4 If UNIQUE is specified, a null VOLUME CTGFV is built. DSETPROC puts the identification VOLUMFVT in the 8 bytes preceding the VOLUME CTGFV. The VOLUME CTGFV is not initialized because VSAM uses the VOLUME CTGFV for a work area.



IDCDE01

Procedure: INTGCHK

5 For a VSAM data set two or three CTGFVs have been built—one each for cluster, 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.

For an INDEXED data set:

SPACPARM CTGFL

Cluster	Data	Index	Action
X	X	X	If the data/index space parameter did not come from a model, 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; If index level space specification is taken from a model, nullify it.
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 an NONINDEXED data set:

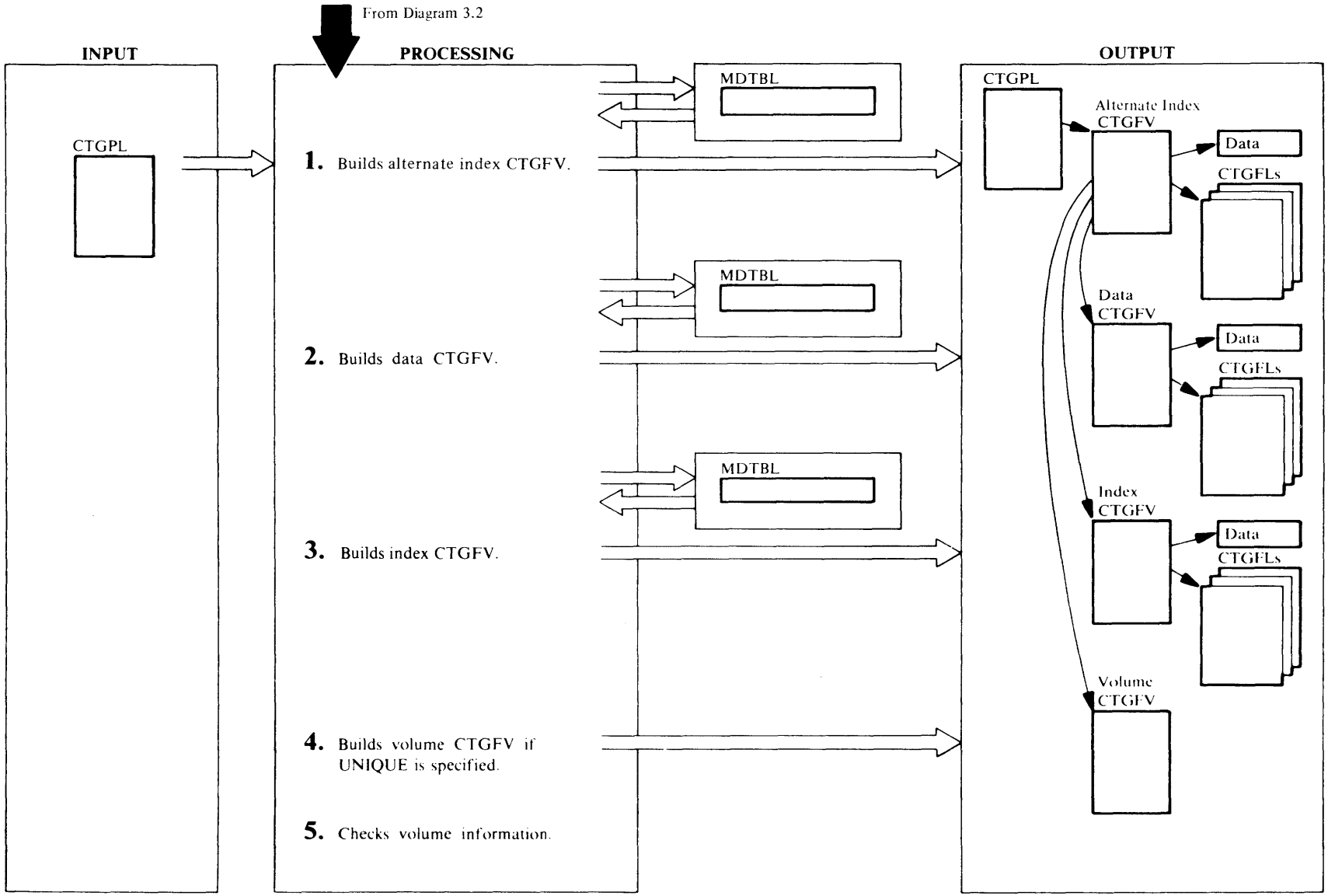
SPACEPARM CTGFL

Cluster	Data	Action
X	X	If the data level space parameters are from a model, this is an error; IDCDE01 terminates the DEFINE.
X		OK; no action.
	X	OK; no action.
none	none	This is an error; IDCDE01 terminates the DEFINE.

INTGCHK insures that space parameters exist wherever USECLASS has been specified, propagated, or modeled. If space parameters do not exist, USECLASS is dropped from the SPACPARM CTGFL.

INTGCHK checks the data CTGFV to be sure that Logical Record Length is specified with a LRECL CTGFL. If not, one is built with a default average recordsize. Control goes to Diagram 3.2, step 4.

Diagram 3.2.6. DEFINE FSR – DEFINE ALTERNATE INDEX



Extended Description for Diagram 3.2.6

IDCDE02, IDCDE03

Procedures: 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, BLOCKS, or RECORDS. This space information is never copied from ALTERNATEINDEX.

Both explicit (MODEL parameter) and implicit (default) modeling are supported, but for any one component (CTGFVT), explicit and implicit modeling cannot be mixed, i.e., explicit models preclude implicit models.

If MODELs are applied, 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 for any particular parameter when modeling is shown below where 1 takes the highest precedence:

ALTERNATEINDEX CTGFV

1. ALTERNATEINDEX parameters
2. Cluster object of ALTERNATEINDEX explicit or default model
3. System default

DATA CTGFV

1. DATA parameters
2. DATA explicit model
3. ALTERNATEINDEX parameters
4. Data object of ALTERNATEINDEX explicit or default model
5. System default

INDEX CTGFV

1. INDEX parameters
2. INDEX explicit model
3. ALTERNATEINDEX parameters
4. Index object of ALTERNATEINDEX explicit or default model.
5. System default

AIXPROC sets the identification of AIXFVT in the 8 bytes before the ALTERNATEINDEX CTGFV. If MODEL is applied, 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. If a reserved name (default model name) prefix ("DEFAULT.MODEL.") is used, a check is made for additional valid qualifiers. 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. ALLCPROC 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, BLOCKS, or RECORDS, along with USECLASS.

IDCDE02, IDCDE03

Procedures: 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, the following occurs:

If MODEL is not applied at the data level, the DATA CTGFV is built with the information specified in the ALTERNATEINDEX parameters.

If MODEL is specified under ALTERNATEINDEX or a default model exists 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. If the DEFAULTVOLUMES parameter is specified at either the ALTERNATEINDEX or the DATA level, nullify the volumes list pointer in the 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 an 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. Volumes are not taken from the default model. ALLCPROC builds a SPACPARM CTGFL with the primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS, along with USECLASS. ALLCPROC also builds a BUFSIZE CTGFL with information from BUFFERSPACE. The following are inserted by ALLCPROC and PROTPROC:

ORDERED|UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK|NOWRITECHECK
maximum from RECORDSIZE and put in the AMDSBCAT CTGFL

UNIQUE|SUBALLOCATION|NOALLOCATION and SPEED|RECOVERY are put in the DSATTR CTGFL. ERASE|NOERASE, REUSE|NOREUSE, and DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS are put in the DSATTR CTGFL.

Protection information is obtained only from the explicit 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, the following occurs:

If MODEL is not applied under DATA, the information specified in the DATA parameters overlays the information placed in the DATA CTGFV on the implicit pass.

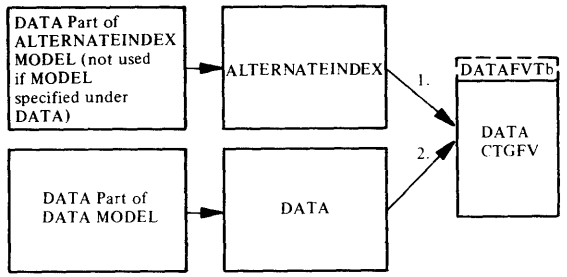
If MODEL is specified under DATA or a default model exists, MODELPRC issues a UCATLG to get information from the data catalog entry of the modeled alternate index. The information from the data entry of the modeled alternate index is put in MDLTABL. If the DEFAULTVOLUMES parameter is given at either the ALTERNATEINDEX or DATA level, nullify the volume list pointer in the 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. If a reserved name was used at the alternate level ("DEFAULT.MODEL." prefix), the DATA qualifier is added from the data component and this constructed name is forced. KEYPROC puts *length* and *offset* from the keys in the AMDSBCAT CTGFL. KEYPROC puts the address of *(lowkey highkey)...* from KEYRANGES in the DATA CTGFV. ALLCRPOC 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, BLOCKS, or RECORDS, along with USECLASS; 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 are inserted:

ORDERED|UNORDERED
cipercent and *capercent* from FREESPACE
size from CONTROLINTERVALSIZE
WRITECHECK|NOWRITECHECK
maximum from RECORDSIZE are put in the AMDSBCAT CTGFL

UNIQUE|SUBALLOCATION|NOALLOCATION and SPEED|RECOVERY are put in the DSATTR CTGFL. ERASE|NOERASE, REUSE|NOREUSE, and DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS are put in the DSATTR CTGFL.



IDCDE02, IDCDE03

Procedures: 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, the following occurs:

If MODEL is not specified at the index level, the INDEX CTGFV is built with the information specified in ALTERNATEINDEX parameters.

If MODEL is applied under CLUSTER and MODEL is not specified under INDEX, MODELPRC uses the saved control interval number for the index entry of the applicable 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. If the DEFAULTVOLUMES parameter is specified at either the ALTERNATEINDEX or INDEX level, nullify the volume list pointer in the 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. Volumes are not taken from default model. ALLCPROC also builds a SPACPARM CTGFL with primary and secondary space information from TRACKS, CYLINDERS, BLOCKS, or RECORDS, along with USECLASS. The following is put in the AMDSBCAT CTGFL:

ORDERED|UNORDERED
WRITECHECK|NOWRITECHECK
size from CONTROLINTERVALSIZE

UNIQUE|SUBALLOCATION|NOALLOCATION 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 explicit 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, the following occurs:

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.

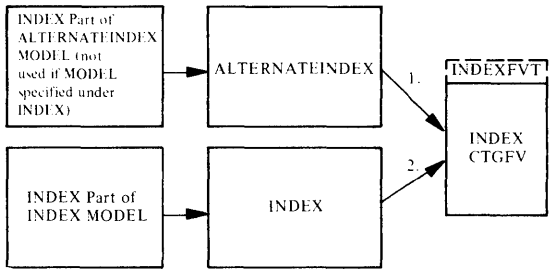
If MODEL is applied under INDEX or a default model exists, MODELPRC issues a UCATLG to get information from the index catalog entry of the modeled alternate index. The information from the index entry of the modeled alternate index is put in MDLTABL. If the DEFAULTVOLUMES parameter is specified at either the ALTERNATEINDEX or the INDEX level, nullify the volumes list pointer in the 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. If a reserved name was used at the ALTERNATEINDEX level ("DEFAULT.MODEL." prefix), the index qualifier is added for the INDEX component and the constructed name is forced. IXOPPROC puts REPLICATE|NOREPLICATE and IMBED|NOIMBED in the AMDSBCAT CTGFL. ALLCPROC puts the address of *dname* fromFILE and the address of *volser*from VOLUMES in the INDEX CTGFV. ALLCPROC also builds or modifies theSPACPARM CTGFL with primary and secondary space information from TRACKS,CYLINDERS, BLOCKS, or RECORDS, along with USECLASS. PROTPROC builds ormodifies 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

The following is put in the DSATTR CTGFL:

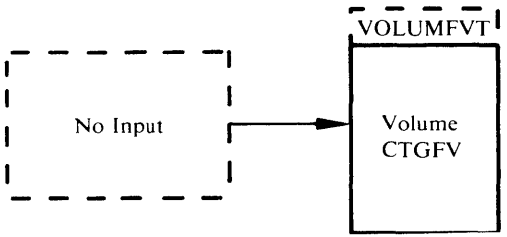
UNIQUE|SUBALLOCATION|NOALLOCATION
ERASE|NOERASE
REUSE|NOREUSE
DOS shareoptions and the reserved for OS shareoptions from SHAREOPTIONS



IDCDE03

Procedures: AIXPROC

4 If UNIQUE is specified, a null VOLUME CTGFV is built. AIXPROC puts the identification VOLUMFVT in the 8 bytes preceding the VOLUME CTGFV. The VOLUME CTGFV is not initialized because VSAM uses the VOLUME CTGFV for a work area.



IDCDE01

Procedure: INTGCHK

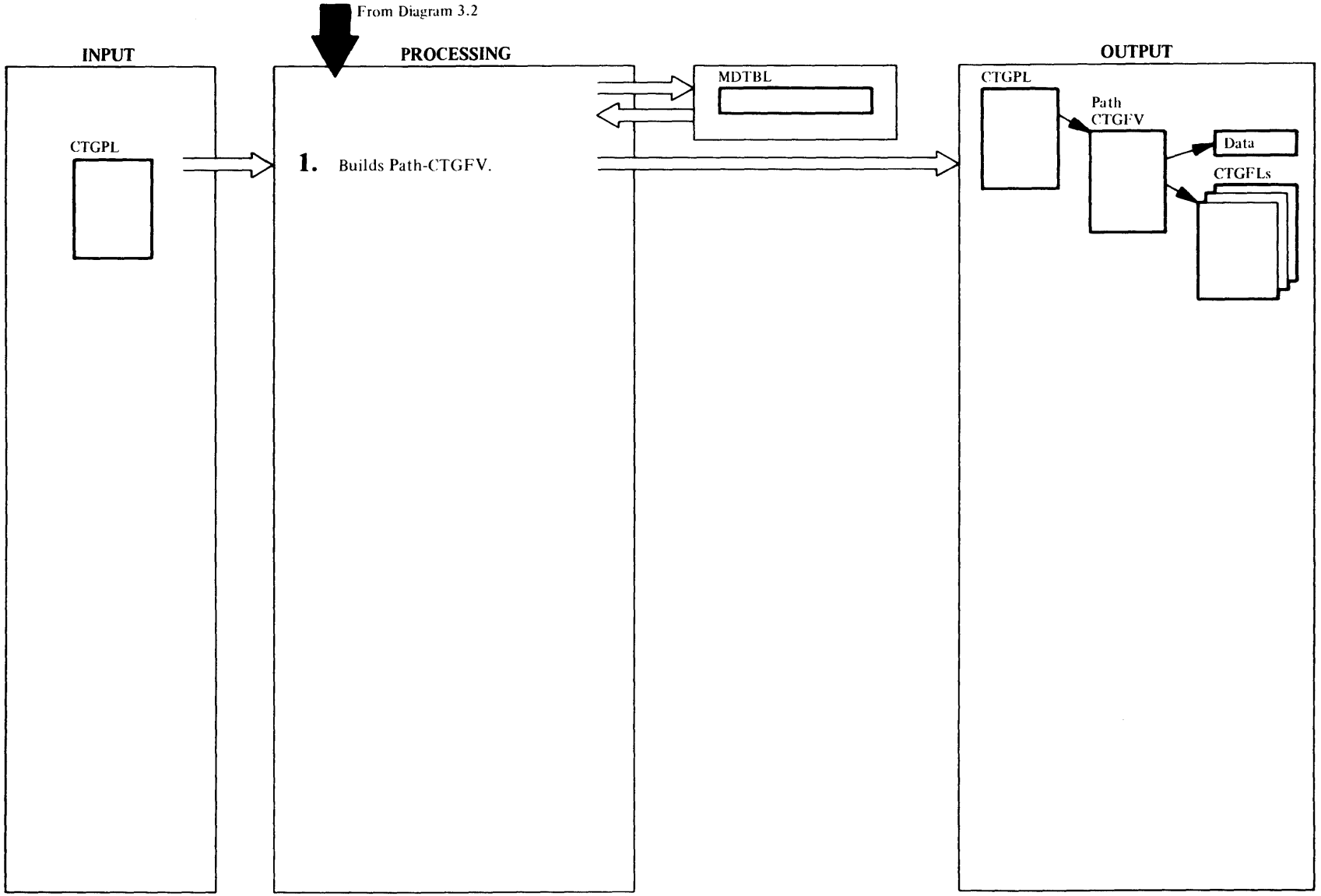
5 For an alternate index two or three CTGFVs have been built—one each for alternate index, data, and index information. If a VOLUME CTGFV has been built, it does not have any information in it because VSAM uses it for a work space. The following table shows the possible places where a SPACPARM CTGFL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

SPACPARM CTGFL

Alternate Index	Data	Index	Action
X	X	X	If the data/index space parameters did not come from a model, 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; If index level space specification is taken from a model, nullify it.
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.

INTGCHK checks the data CTGFV to be sure that logical record length is specified with a LRECL CTGFL. If not, an LRECL CTGFL is built with the default average recordsize. Control goes to Diagram 3.2, step 4.

Diagram 3.2.7. DEFINE FSR – DEFINE PATH



Extended Description for Diagram 3.2.7

IDCDE02, IDCDE03

Procedures: PATHPROC, NAMEPROC, MODELPRC PROTPROC, ALLCPROC

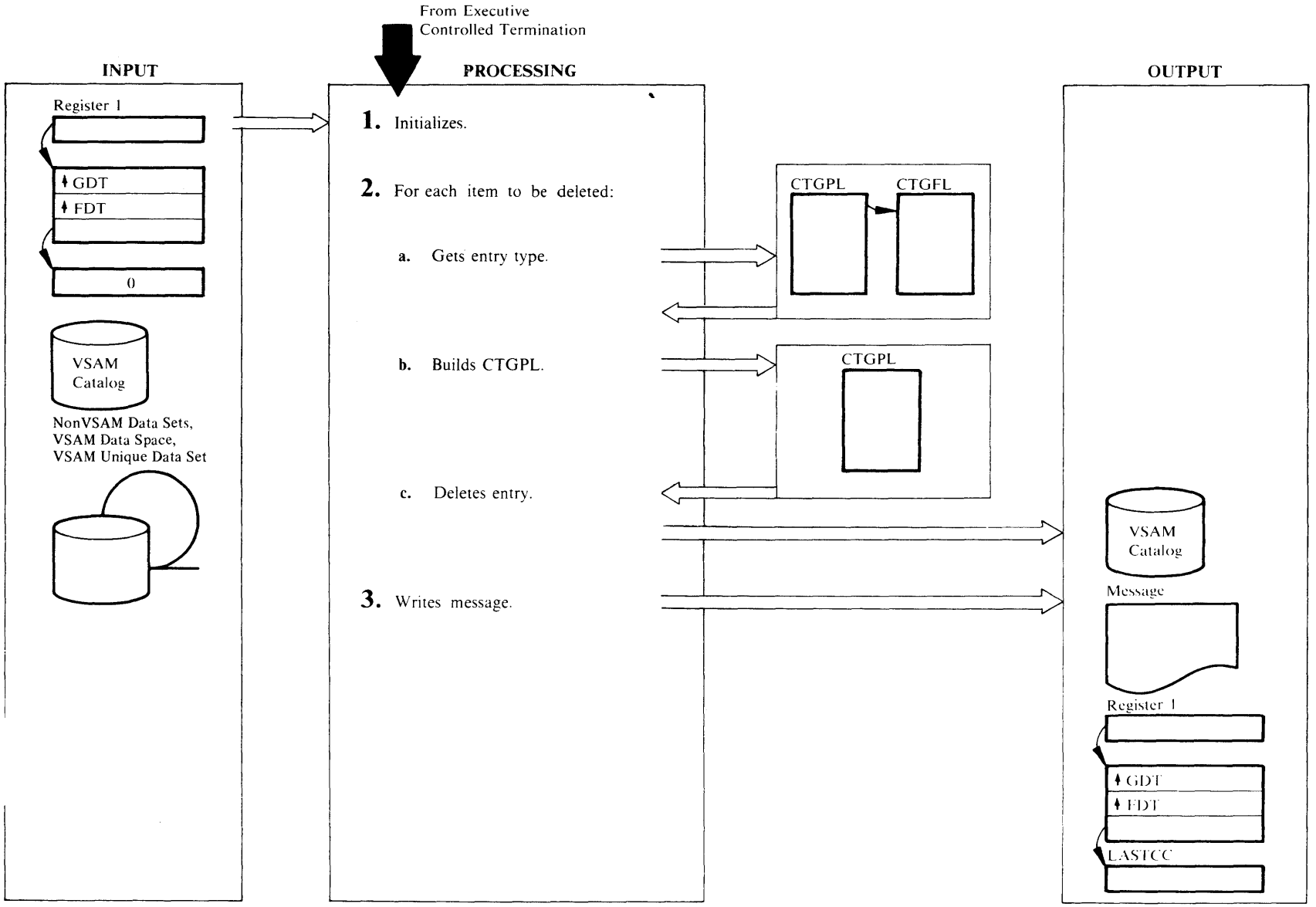
1 In the DEFINE PATH command, you specify information under one main keyword: PATH. The DEFINE FSR builds a CTGFV to describe the path. Information specified under PATH goes in the PATH CTGFV.

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 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 3.3. DELETE FSR



Extended Description For Diagram 3.3

IDCDL01

Procedure: CATOPEN

1 If a CATALOG is specified, CATOPEN builds an OPNAGL and issues a UOPEN to open the catalog. If the catalog does not open, CATOPEN prints an error message and the DELETE command is terminated. If the return code from UOPEN is zero, CATOPEN compares the data set name returned by UOPEN (in IOCDSN) to the name specified in the CATALOG parameter. If the compare is unequal, a message is written and the DELETE command is terminated.

IDCDL01

Procedures: FINDTYPE, BUILDCPL, CATCALL, MORESP, IDCDL01

2 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. If the entry type is not specified in the command, FINDTYPE builds a CTGPL and CTGFL in which VSAM returns the entry type. FINDTYPE initializes the CTGPL and CTGFL once for the entire DELETE command, and they are used over and over for each *entryname*. FINDTYPE issues a UCATLG macro to locate the entry type. If the return code is nonzero, FINDTYPE builds an error conversion table and invokes the UERROR macro to print a message, but the rest of the DELETE command is processed.

PARAMCHK checks for invalid or insufficient parameters which were not checked by the Reader/Interpreter. The Reader/Interpreter cannot do all the necessary parameter checking if the user has not specified the entry type or if the entry type is NONVSAM. If there is an invalid parameter, PARAMCHK writes an error message, but the rest of the DELETE command is processed.

b. 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 or ACB address if CATALOG is specified. BUILDCPL also puts the address of a work area needed by VSAM in the CTGPL. The work area passed to catalog management is set initially to a size large enough to contain twelve names. BUILDCPL puts the address of the entry name and the address of the entry password in the CTGPL. If the entry type is nonVSAM and neither SCRATCH or NOSCRATCH is specified, BUILDCPL sets SCRATCH in the CTGPL. If the entry was located from the catalog, BUILDCPL puts the entry type in the CTGPL.

c. CATCALL deletes the *entryname* by issuing a UCATLG macro with the CTGPL built by BUILDCPL. If the return code is zero, VSAM has returned a list of deleted objects. CATCALL writes the name of each deleted object in the entry with a UPRINT macro. Control is given to step 2. If the return code is 160, the entry type is SPACE and the space was deleted, but the volume entry in the catalog could not be removed because there are still some VSAM data sets on the volume. This is not a DELETE error so the condition code to the user is zero, but CATCALL writes an explanatory message.

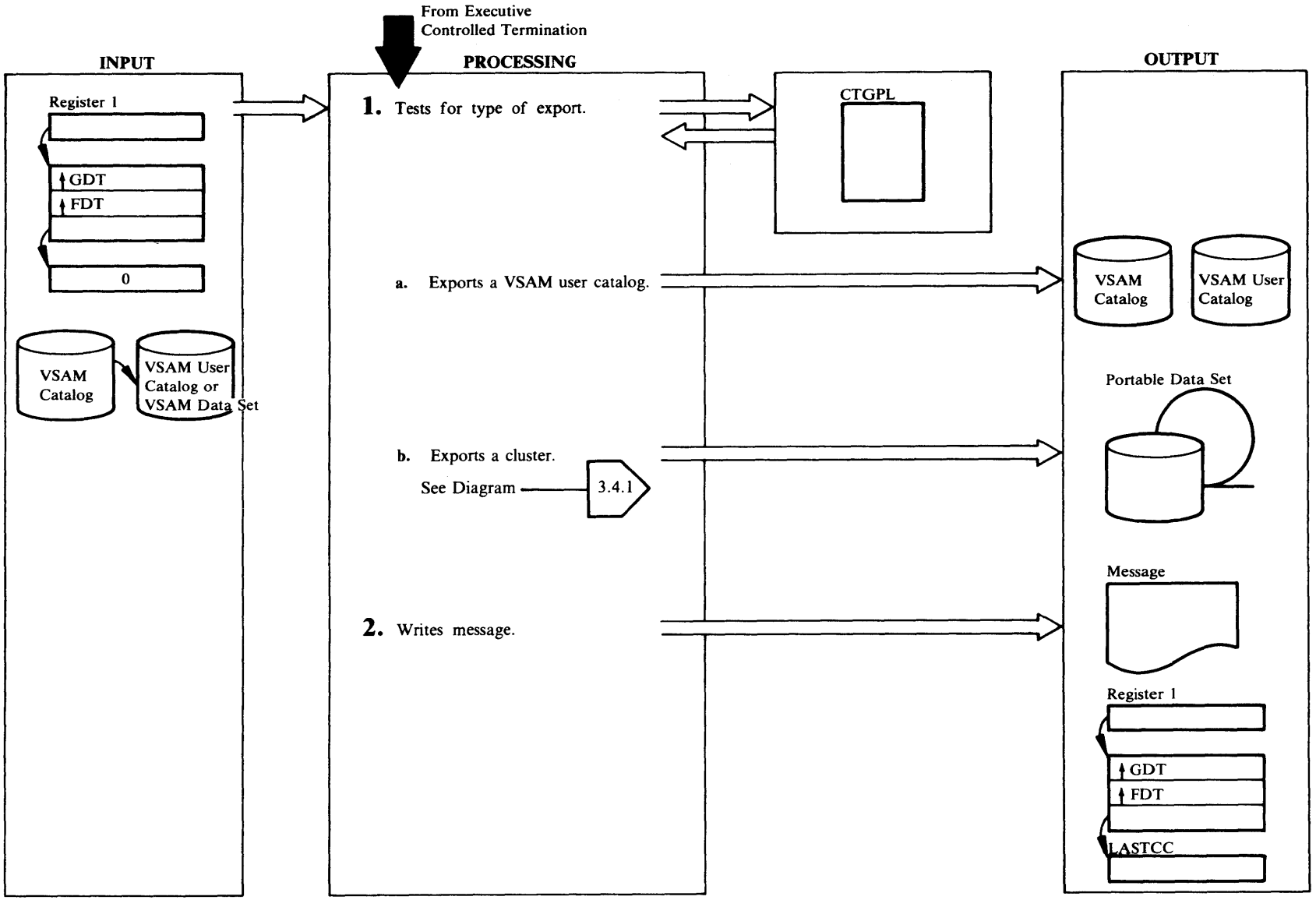
A return code of 40 indicates that insufficient space remains in the work area to contain the names associated with the next structure segment to be deleted (e.g. an alternate index with its associated data, index and path names). Catalog management services has placed in the work area the names of those objects successfully deleted thus far, plus a factor indicating the amount of space necessary for the next structure. Should catalog give a return code of 40, CATCALL calls MORESP. MORESP sets the CTGOVRID bit to 1 and the CTGERASE bit to 0 to prevent CMS from re-erasing the object being deleted. MORESP prints the names of those entries deleted thus far and calculates whether the current work area size can contain the next segment to be deleted. If enough space is available, the work area is reset to zero; otherwise the current work area is freed with a UGPOOL call (provided that it is not PL/S automatic storage) and a large enough work area obtained with a UGPOOL call. If the return from UGPOOL is nonzero, a message is written and control returns to Step 2 for the next entry. Otherwise, MORESP reissues the UCATLG macro with the same entry name. This process continues until the entire structure has been deleted or a terminating error occurs. If the return code from UCATLG is not 40 or 160 an error message is printed by building an error conversion table and invoking the UERROR macro.

IDCDL01

Procedures: CLEANUP, IDCDL01

3 If a catalog was opened by CATOPEN, CLEANUP closes the catalog with a UCLOSE macro. IDCDL01 prints a message with LASTCC. Control goes to Executive Controlled Termination, Diagram 4.1.

Diagram 3.4. EXPORT FSR



Extended Description for Diagram 3.4

IDCXP01

Procedures: IDCXP01, DELTPROC, LOCPROC, CTLGPROC, OPENPROC, PUTPROC, CLUSPROC

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 UFPOOL, 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 nonzero and not 40, or if the return code from the second UCATLG is nonzero, an error message is written by building an error conversion table and issuing the UERROR macro.

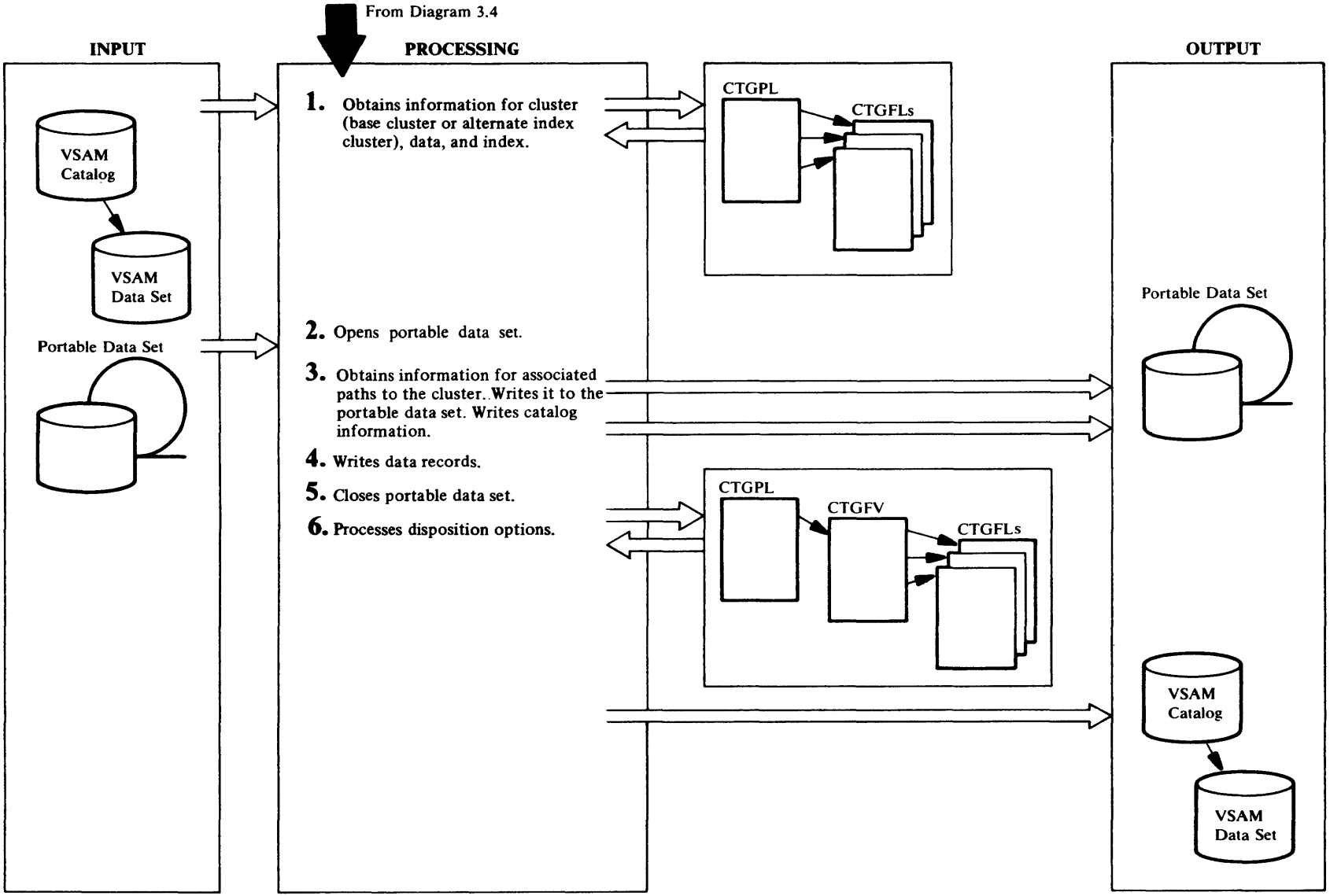
   b. LOCPROC gets catalog information about the cluster or alternate index, data, index, and path entries for the VSAM data set. OPENPROC opens the portable data set for output. PUTPROC writes catalog information and data records on the portable data set. CLUSPROC closes the portable data set and processes 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.

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 4.1.

Diagram 3.4.1. EXPORT FSR – CLUSTER or ALTERNATEINDEX



Extended Description for Diagram 3.4.1

IDCXP01

Procedures: LOCPROC, CTLGPROC, IDCXP01, CLUSPROC

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 infomation
Buffer size
Logical record length
Low key on volume
High key on volume
AMDSB control block
Exception exit
Alternate index and path attributes
Type and name of associated objects
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 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. CLUSPROC tests to be sure that the type of catalog entry is a cluster or an alternate index. If it is not, 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 or alternate index 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. After retrieval of the data entry information, CLUSPROC examines the data set attributes to determine if the object has been flagged as not usable. If so, an error message is written and the VSAM data set is not exported. The data component maximum recordsize (RECORDMODE) or control interval size (CIMODE) is extracted from the AMDSB for use as the maximum recordsize value for the portable data set. CLUSPROC examines the data component AMDSB for NOCIFORMAT SAM ESDS. If NOCIFORMAT SAM ESDS, an error message is written and the command is terminated. CLUSPROC tests for SAM ESDS and for the SAM ESDS feature. If SAM ESDS with the SAM ESDS feature is not installed, an error message is written and the command is terminated.

The processing in the above paragraph (except for the data component AMDSB processing) is repeated for the index entry.

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.

IDCXP01

Procedure: OPENPROC

2 OPENPROC builds an OPNAGL and issues a UOPEN to open the portable data set for output. User specified tape label and rewind options are placed in the OPNAGL for UOPEN processing. If the return code is nonzero, 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.

IDCXP01

Procedures: 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 preceeds the field in the CTGFL where the information would have been. CONTRBL always turns off the temporary export flag and the inhibit update flag in the exported DSATTR CTGFL. If INHIBITTARGET is specified, a flag is set in the portable data set timestamp record so IMPORT can process INHIBITTARGET. If export CIMODE is specified, a flag is set in the portable data set timestamp record so IMPORT can process CIMODE-format data. Flags are also set in the timestamp record when SAM ESDS and NOALLOCATION files are exported. PUTPROC writes the dictionary followed by the information from the CTGFLs. 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 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.

IDCXP01

Procedures: RECPROC, LOCPROC, OPENPROC

4 RECPROC calls OPENPROC to open 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 in export RECORDMODE, 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.

OPENPROC triggers CIMODE processing of data (by UCOPY) by setting the "export CIMODE" flag and the "CNV processing" flag in the OPNAGL of the input data set.

IDCXP01

Procedure: CLUSPROC

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

IDCXP01

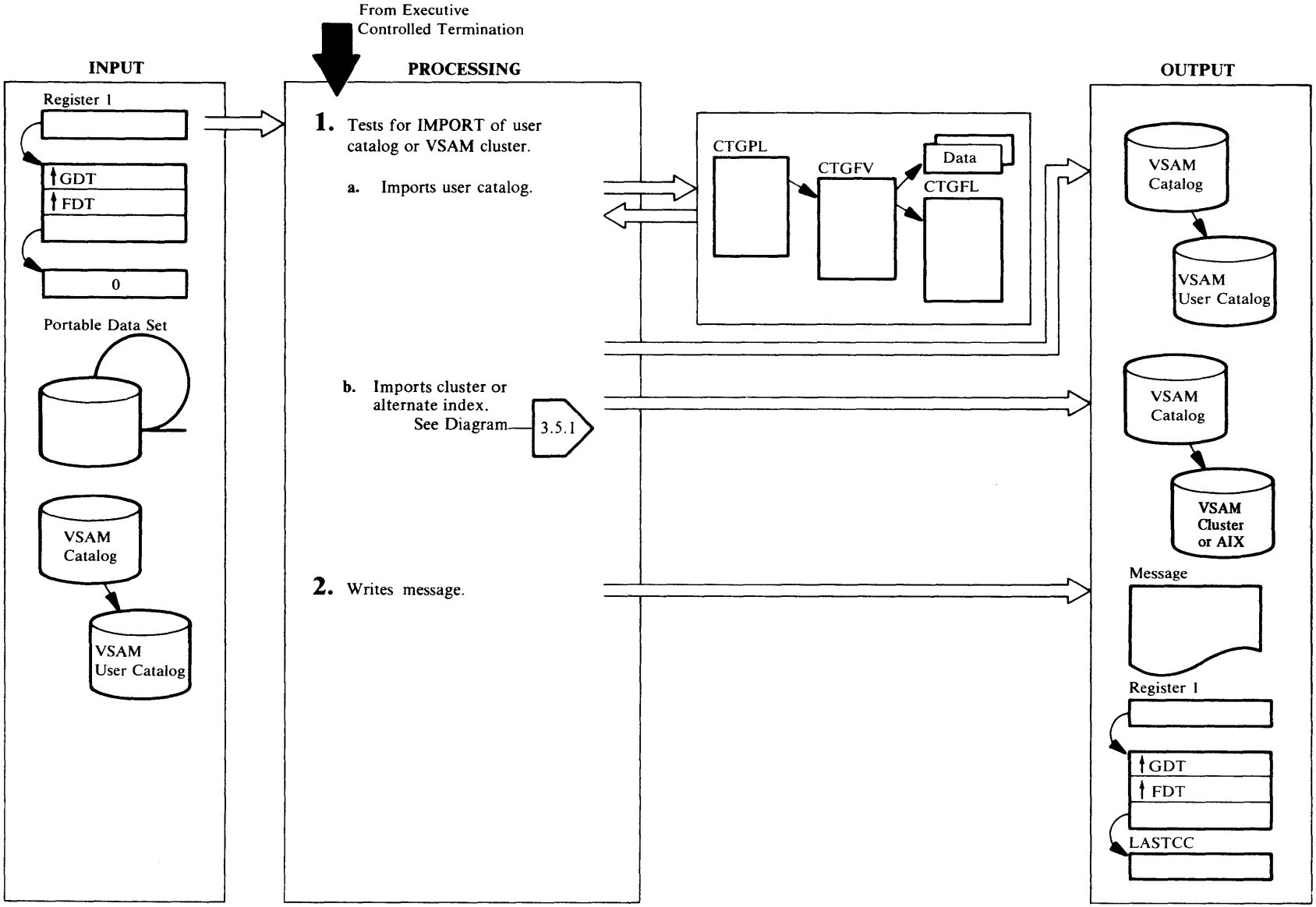
Procedures: DELTPROC, CLUSPROC, CTLGPROC, ALTRPROC, MORESP

6 If PERMANENT is specified, DELTPROC builds a CTGPL. If ERASE or PURGE is specified DELTPROC sets up the proper flags in the CTGFL. DELTPROC issues a UCATLG macro 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 and to finish deleting the object.

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 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. Control returns to Diagram 3.4, step 2.

Diagram 3.5. IMPORT FSR



Extended Description for Diagram 3.5

IDCMP01

Procedures: OPENPROC, IDCMP01, CLUSPROC, FVTPROC, CPLPROC, CNCTPROC, 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 CATALOG is specified, it is not opened because the catalog is assumed to be the job catalog or master catalog and the operating system has opened it. 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. (More than one user catalog can be imported with one IMPORT command.) CNCTPROC builds a CPL and an FVT for the connect operation. 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 the objectname contains the reserved default model prefix, an error message is written and control goes to step 2. 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 nonzero, an error message is written by building an error conversion table and issuing the UERROR macro. When all the catalogs have been connected, control goes to step 2.

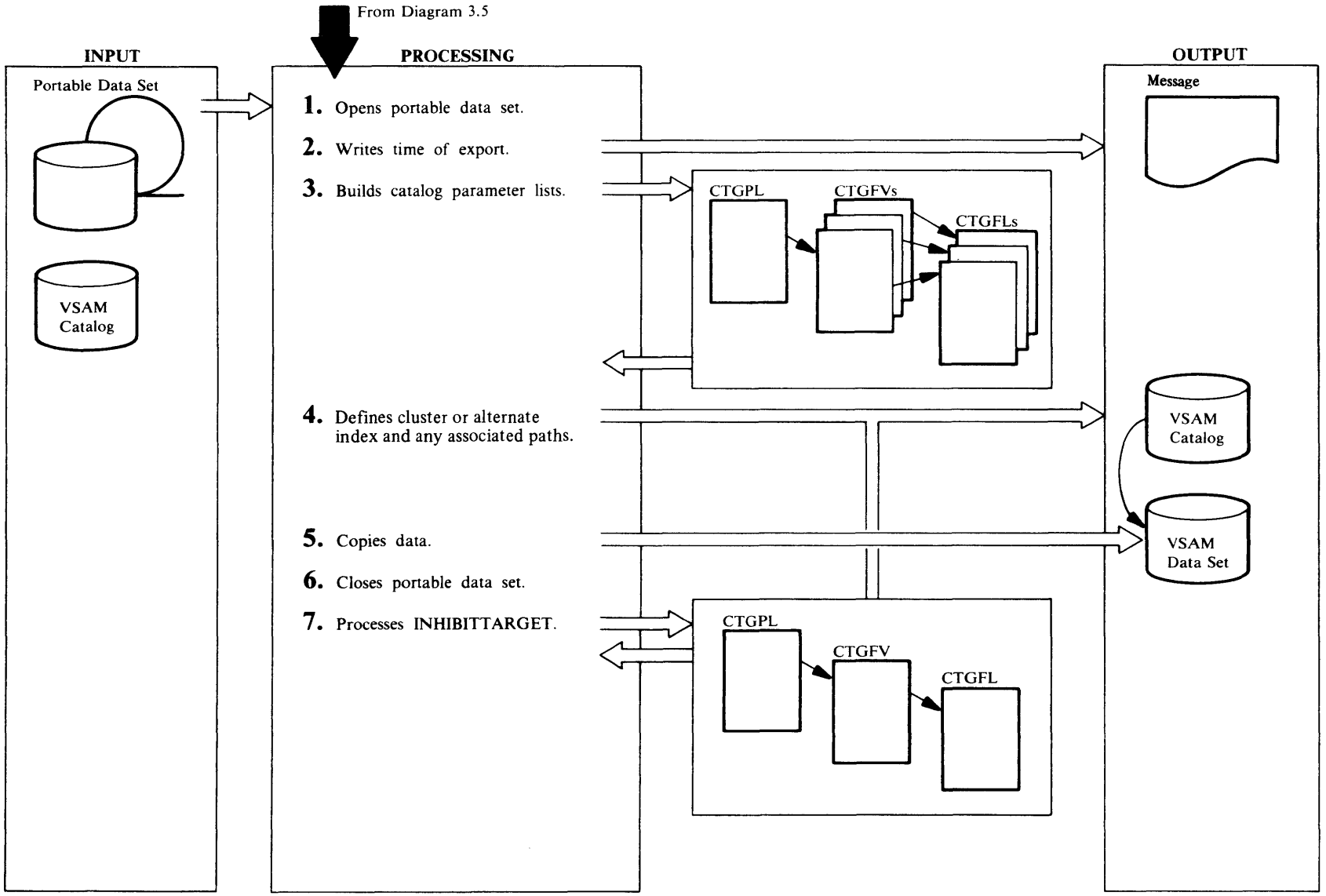
   b. OPENPROC opens the portable data set. CLUSPROC writes the time of export with a UPRINT macro. CLUSPROC uses the catalog information in the portable data set to "define" the VSAM data set. OPENPROC opens the VSAM data set and RECPROC copies the data records from the portable data set to the VSAM data 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.

IDCMP01

Procedure: IDCMP01

2 Based on the return code from CLUSPROC or CNCTPROC, IDCMP01 sets the value for LASTCC. If LASTCC is less than 12, a completion message (with LASTCC) is written; otherwise a termination message (with LASTCC) is written. Control goes to Executive Controlled Termination.

Diagram 3.5.1. IMPORT FSR – CLUSTER or ALTERNATEINDEX



Extended Description for Diagram 3.5.1

IDCMP01

Procedures: OPENPROC, IDCMP01

1 OPENPROC builds an OPNAGL and issues a UOPEN to open the portable data set. User specified tape label and rewind options are placed in the OPNAGL for UOPEN processing. 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 nonzero, 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.

IDCMP01

Procedures: CLUSPROC, MSGPROC

2 CLUSPROC gets the first record of the portable data set which contains the date and time the portable data set was created by the EXPORT FSR. (The record contains flags indicating whether EXPORT specified INHIBITTARGET and CIMODE or RECORDMODE.) MSGPROC writes the date and time with a UPRINT macro.

IDCMP01

Procedures: CLUSPROC, CPLPROC, FVTPROC, BFPLPROC, BPASPROC, IUNIQPRC, LVLRPROC, RANGPROC, DVOLPROC, DVOLCHK

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 entry and its associated data and index entries. FVTPROC obtains the data set maximum logical record size (RECORDMODE) or control interval size (CIMODE) from the data component AMDSBCAT CTGFL and passes it to the I/O adapter via a function of the UCLOSE macro that allows a larger work-area data buffer. The value obtained becomes the portable data set maximum logical record size. FVTPROC tests the AMDSB for SAM ESDS and for the SAM ESDS feature. If SAM ESDS with the SAM ESDS feature is not installed, an error message is written and the command is terminated. 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. CLUSPROC puts the address of the optional dname from OUTFILE on the IMPORT command in the cluster CTGFV. LVLRPROC puts the address of the *volser* ... list from VOLUMES in the CTGFV for the *objectname* in the OBJECTS parameter. Information about VOLUMES is available in the portable data set and is used unless superceded by the VOLUMES or DEFAULTVOLUMES subparameter.

c. If USECLASS (OBJECTS parameter) is specified for an objectname, CLUSPROC changes the SPACPARM CTGFL(s) for the objectname. If objectname is cluster or alternate index, data and index (if present) SPACPARM CTGFLs are changed. If objectname is a data or index component, only the component SPACPARM CTGFL is changed.

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. If the NEWNAME is a reserved default model name, an error message is issued and the command terminates.

If DEFAULTVOLUMES is specified for a particular objectname, DVOLPROC builds an empty volume list CTGFL attached to the object CTGFV, unless VOLUMES has already been specified for the object or at the cluster level. If VOLUMES or DEFAULTVOLUMES occurs at the cluster level, data and index volume list CTGFL pointers are nullified. After OBJECTS parameter processing completes, CLUSPROC propagates the cluster level volume list CTGFL to the data and index CTGFLs if they contain null volume list CTGFL pointers. DVOLCHK is called to ensure that no file with the unique attribute or object with the ordered attribute contains an empty (DEFAULTVOLUMES) volume list CTGFL and to determine if a DEFAULTVOLUMES specification was totally superceded by a VOLUMES specifications (warning condition).

Data from the IMPORT command overrides data from the portable data set.

IDCMP01

Procedures: 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, control goes to step 4b. Otherwise, control goes to step 4c.

b. 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 terminated. If the data set is empty, checks are made to insure that the data set organization, data length, and relative key position in the catalog entry are the same as those which were exported; that the maximum VSAM LRECL of the catalog catalog entry is greater than or equal to that of the export data set; that the RECORDFORMAT characteristics (AMDRCFRM) and SAM LRECL value (AMDBLREC) in the catalog entry are the same as those which were exported if the data sets are ESDS. If any of these conditions are not met, a message is written and the IMPORT is terminated. If the OBJECTS parameter was specified for the empty data set, a warning message is issued. Control then goes to step 4.c. 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. Then CTLGPROC reissues the UCATLG macro to define the VSAM data set. If the UCATLG return code is nonzero, CTLGPROC issues an error message by building an error conversion

table and invoking the UERROR macro, and the IMPORT is terminated.

c. If a recovery volume serial is returned for the define, a UPRINT macro is issued to print it. If the successful DEFINE was for a unique data set on a fixed block device, a message is printed for each volume indicating the actual blocks allocated for that volume.

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 subparameter of the OBJECTS parameter used for path objects is NEWNAME. If the NEWNAME is a reserved default model name, error messages are issued and the path is bypassed for import. If NEWNAME is omitted and any other subparameter is specified, an invalid OBJECTS parameter message is written, LASTCC is set to 8, 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 UERROR, and LASTCC is set to 8. However, the IMPORT is not terminated. CLUSPROC ensures that all OBJECTS parameter objectnames have corresponding component|path names. Warning messages are printed for any mismatches, and LASTCC is set to the current value of LASTCC or 4, whichever is greater.

IDCMP01

Procedures: OPENPROC, RECPROC

5 OPENPROC builds an OPNAGL and issues a UOPEN to open the newly defined VSAM data set. A flag is set in the OPNAGL to indicate RECORDMODE or CIMODE. If a password is specified via the OUTFILE or OUTPW parameter, this password is passed to UOPEN for use in building the ACB. Otherwise, the exported master password, if any, is used. If the OUTFILE parameter is omitted, the input file file-id and catalog name (if present) are passed to UOPEN for use in building the ACB. RECPROC issues a UCOPY to copy the data from the portable data set to the newly defined (or empty pre-defined) VSAM data set.

When importing a relative record data set in RECORDMODE, 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 if the OUTFILE parameter was specified 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 locate 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 terminated.

IDCMP01

Procedure: CLUSPROC

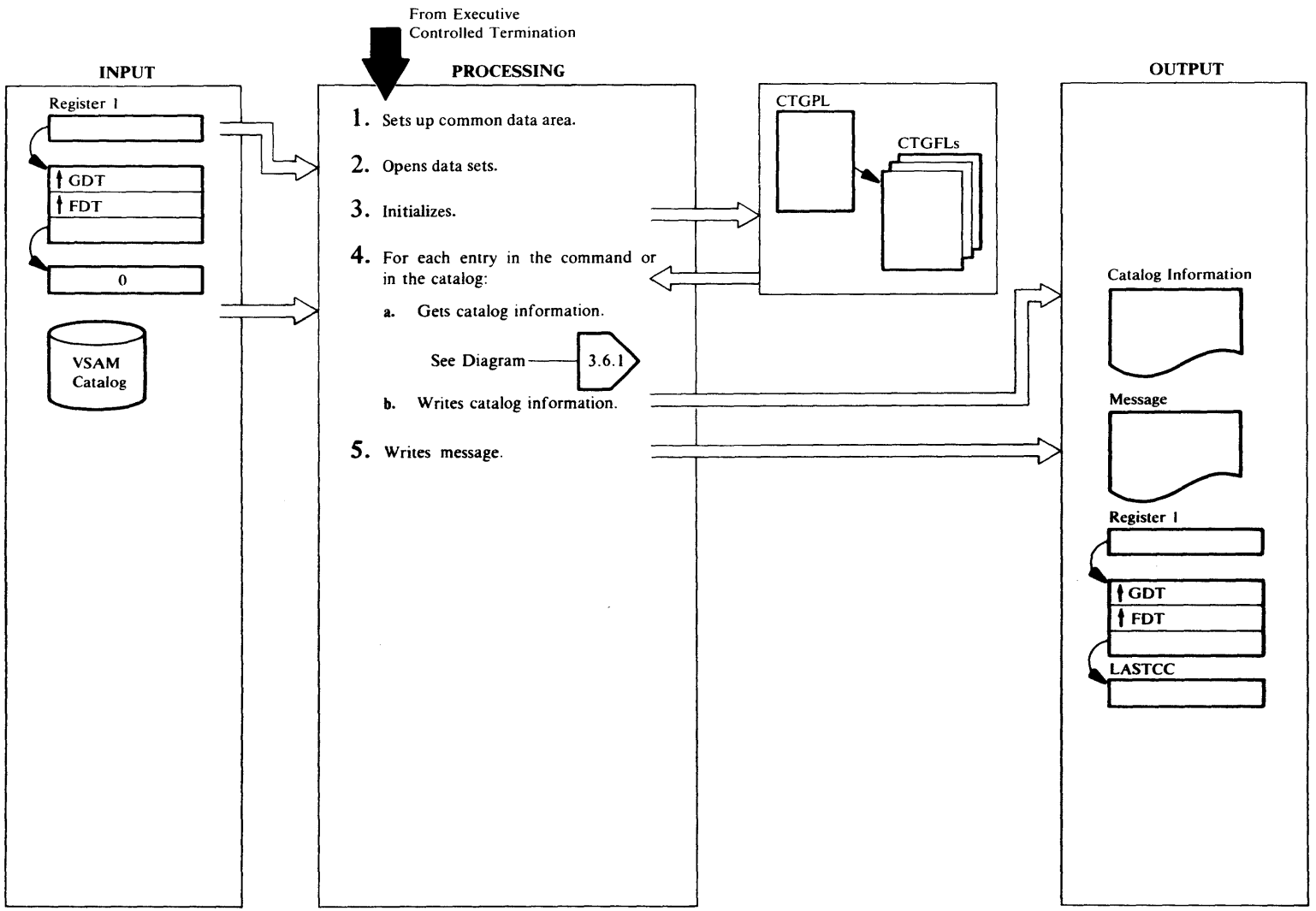
6 CLUSPROC issues a UCLOSE to close the portable data set.

IDCMP01

Procedures: 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 set. CTLGPROC issues a UCATLG macro to alter the VSAM data set to inhibit updating the VSAM data set. If the VSAM data set has an index component, the same steps are repeated to alter the index component to INHIBITTARGET. Control goes to Diagram 3.5, step 2.

Diagram 3.6. LISTCAT FSR



Extended Description for Diagram 3.6

IDCLC01, IDCLC02

Procedures: 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.

IDCLC01

Procedure: INITPROC

2 If OUTFILE is specified, INITPROC builds an OPNAGL and issues a UOPEN to open the alternate output data set. By opening the alternate file first, any LISTCAT error messages appear on the alternate file. If CATALOG is specified with *dname* as well as a *catname*, INITPROC builds an OPNAGL and issues a UOPEN for the *catname* and requests that the ACB be returned. INITPROC compares the catalog name returned by the UOPEN macro to the *catname* from the CATALOG parameter in the LISTCAT command. If the catalog names do not match, the LISTCAT command terminates and control goes to step 5. If a *dname* is not specified, but a *catname* is, 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.

IDCLC01

Procedure: INITPROC

3 INITPROC issues a ULOAD macro to load IKQDNT, the device name table. This table translates the hexadecimal UCB device type code to the external device name. (For example, the catalog UCB code X'3050200D' translates to the 3330-11.) 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, VOLUMES, ALLOCATION, or ALL. Catalog fields are obtained by control blocks named CTGFLs. The table following this description shows the CTGFLs that are used for each type of catalog entry.

If NAME is specified, INITPROC initializes CTGFLs 2 through 4. For VOLUMES, INITPROC initializes 2 through 10. For ALLOCATION, INITPROC initializes 2 through 14. For ALL, INITPROC initializes 2 through 28. INITPROC adds the DSATTR to the end of the NAME, VOLUME, and ALLOCATION list if NOTUSABLE is specified. If more than one entry type is being listed or if NOTUSABLE is specified, INITPROC adds the MULTITYP CTGFL to the beginning of the list of CTGFLs.

IDCLC01, IDCLC02

Procedures: ENTPROC, LOCPROC, RTEPROC, CDIPROC, AUPROC, VPROC, FPLPROC, ANSVPROC, DEVTCONV

4 If ENTRIES is specified, catalog information is found on each *entryname* in the command. If ENTRIES is not specifed, catalog 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. Diagram 3.6.1 shows getting catalog information in detail.

b. RTEPROC test the entry type of the catalog entry. If the type is PATH, ALTERNATEINDEX, CLUSTER, DATA, or INDEX, CDIPROC formats the information and writes it with a UPRINT macro. If the type is NONVSAM or USERCATALOG, AUPROC formats the information and writes it with a UPRINT macro. If the type is SPACE, VPROC formats the information and writes it with a UPRINT macro. DEVTCONV is involved to translate the hexadecimal UCB device type code to the external device name.

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 or alternate index, RTEPROC determines whether an association of the object—that is a data, index, or path entry—is to be listed. If it is, FPLPROC reinitializes the CTGFLs. ANSVPROC retrieves the information about the data, index, or path via the control interval rather than by name. Control returns to 4a to locate information about the data, index, or path. FPLPROC reinitializes the CTGFLs for the next catalog entry. If the type is not valid, RTEPROC writes a message. Control goes to step 4a for the next entry. Refer to *Using VSE/VSAM Commands and Macros* for a sample listing of LISTCAT output.

IDCLC01, IDCLC02

Procedure: IDCLC01, FREESTG

5 IDCLC01 writes a summary of the entries listed and suppressed due to incorrect passwords. If INITPROC opened a VSAM catalog, IDCLC01 issues a UCLOSE to close the VSAM catalog. If an alternate output file was opened by INITPROC, IDCLC01 issues a UCLOSE to close the file. Any storage obtained during the processing of the LISTCATALOG command is released with a UFPOOL macro. IDCLC01 then calls FREESTG (in IDCLC02) to free the automatic storage acquired by IDCLC02. IDCLC01 then writes a message containing LASTCC. Control goes to Executive Controlled Termination, Diagram 4.1.

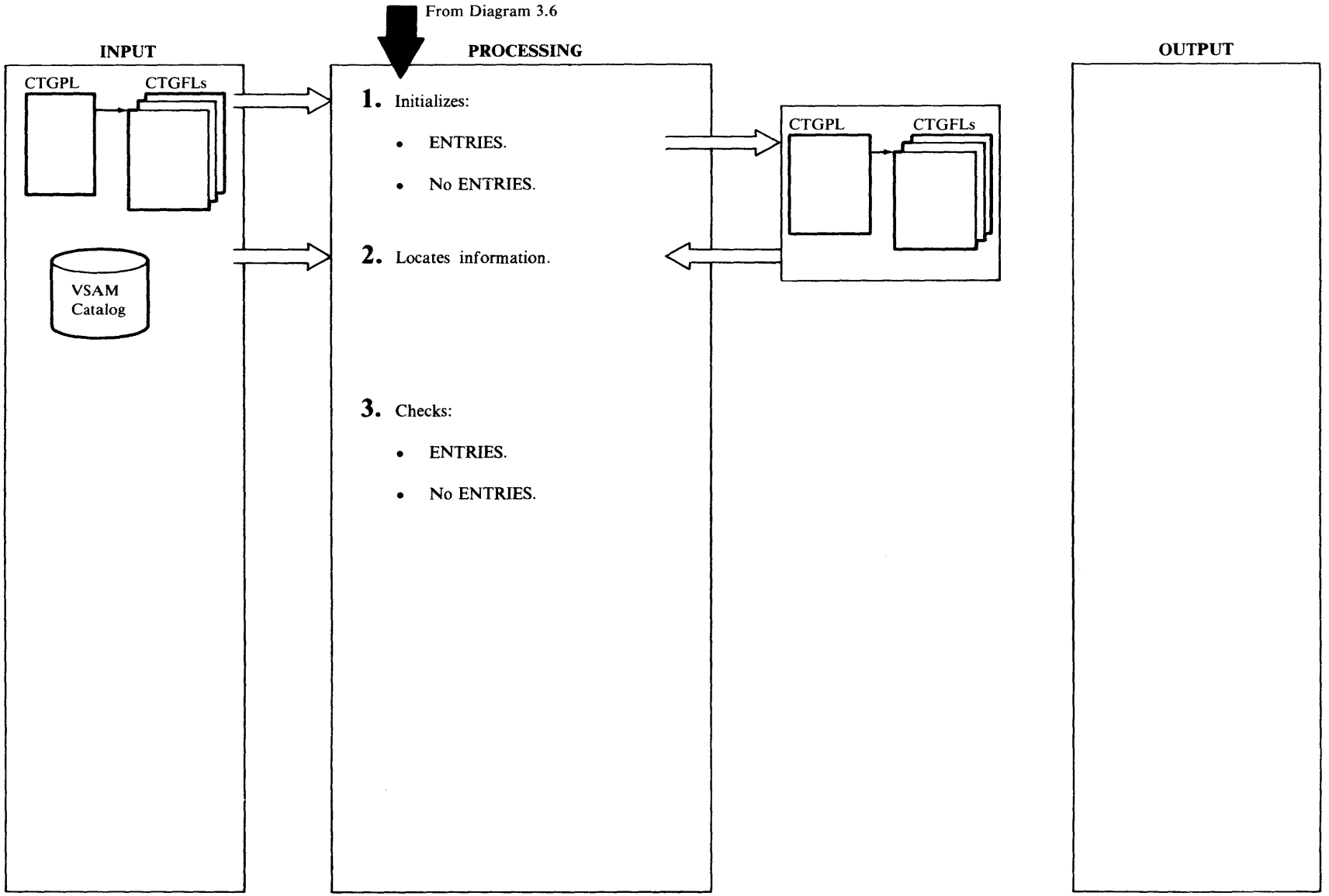
CTGFLs Used for Each Type of Catalog Entry

CTGFL Name	Entry Type CLUSTER	DATA	INDEX	NONVSAM	USER CATALOG	SPACE	ALTER NATE INDEX	PATH	Data in CTGFLs
1. MULTITYPE									Identifies multiple catalog types to be listed.
2. ENTYPE	X	X	X	X	X	X	X	X	Entry type.
3. ENTNAME	X	X	X	X	X	X	X	X	Entry name.
4. NAMEDS	X	X	X				X	X	CI number and entry type of each association.
5. DSETEXDT	X	X	X	X			X	X	Data set expiration date.
6. DSETCRDT	X	X	X	X			X	X	Data set creation date.
7. OWNERID	X	X	X	X			X	X	Data set owner.
8. RELCRA	X	X	X	X			X	X	VSAM release and catalog recovery information.
9. CATVOL		X	X	X	X				Volume information for data set.
10. VOLDVCHR						X			Volume device character.
11. FPACPARM		X	X						Primary and secondary allocation.
12. HURBADS		X	X						High used RBA.
13. HARBADS		X	X						High allocated RBA.
14. FNTVOL		X	X						Physical description of data set.
15. VOLTSTMP						X			Volume time stamp.
16. SYSEXTDS						X			System allowed extents.

CTGFLs Used for Each Type of Catalog Entry—continued

CTGFL Name	Entry Type CLUSTER	DATA	INDEX	NONVSAM	USER CATALOG	SPACE	ALTER NATE INDEX	PATH	Data in CTGFLs
17. NODSPACE						X			Number of data space on volume.
18. NODSET						X			Number of data sets on volume.
19. FPACEHDR						X			Characteristics and statistics of data space.
20. DSDIRECT						X			Data Set directory for a data space.
21. FSPDSCRP						X			Physical description of data space.
22. PASSWALL	X	X	X				X	X	Password (security) information.
23. AMDSBCAT		X	X						AMDSB control block.
24. DSATTR		X	X						Data set attributes.
25. BUFSIZE		X	X						Minimum buffer size.
26. LRECL		X	X						Logical record size.
27. RGATTR							X	X	AIX and PATH attributes.
28. EXCPEXIT		X	X						Exception exit module name.
29. CATACB									Catalog ACB address.

Diagram 3.6.1. LISTCAT FSR -- Gets Information



Extended Description for Diagram 3.6.1

IDCLC01

Procedures: ENTPROC, GNXTPROC

1 If ENTRIES is specified, control goes to 1a. If ENTRIES is not specified, control goes to 1b.

   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 3.6, step 5.

   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.

IDCLC02

Procedure: LOCPROC

2 LOCPROC issues a UCATLG macro with the CTGPL and CTGFLs to locate catalog information about the entry.

IDLCL01

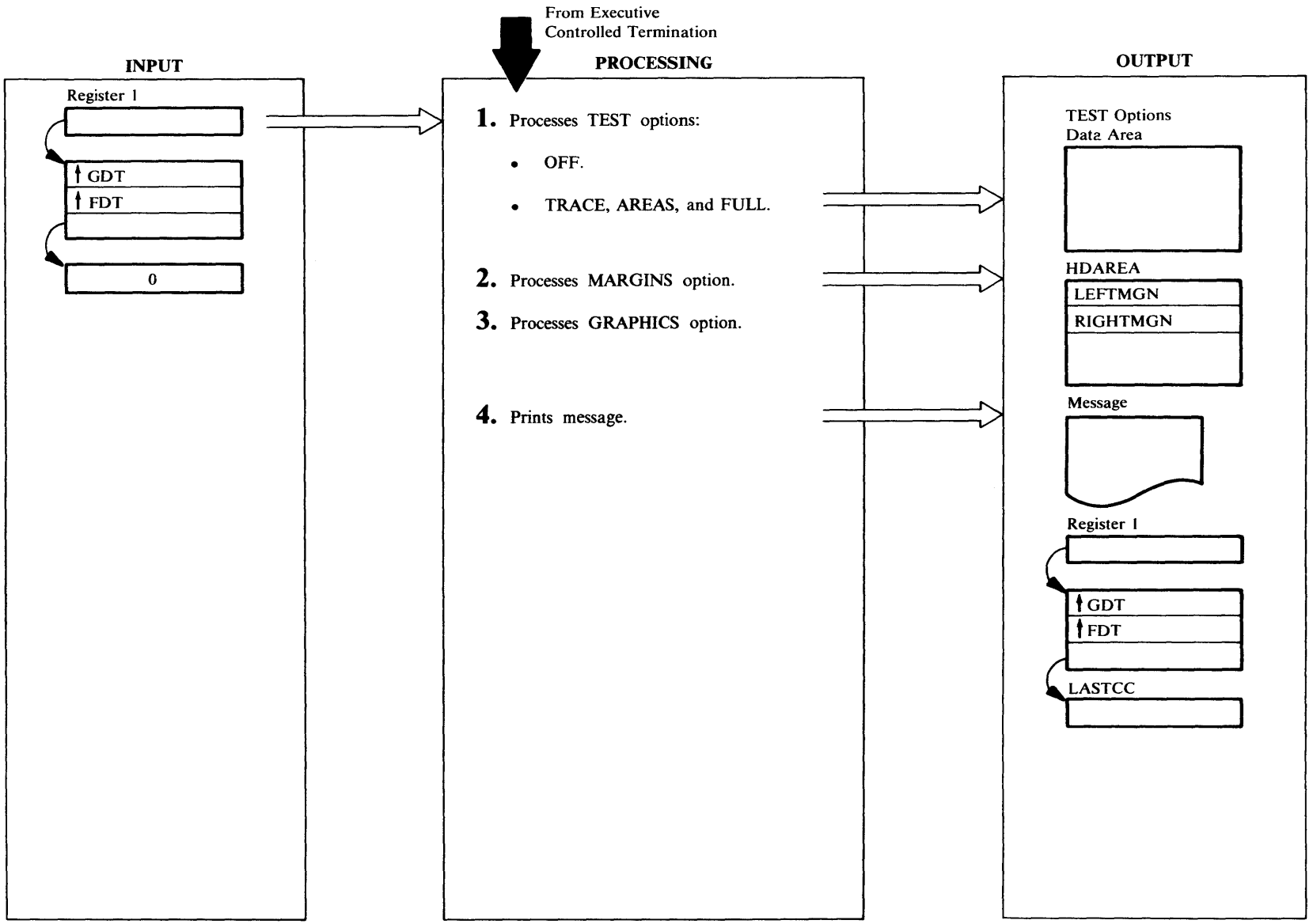
Procedures: ENTPROC, GNXTPROC

3 If ENTRIES is specified, control goes to 3a. If ENTRIES is not specified, control goes to 3b.

   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, or the entry is a cluster or an alternate index, control goes to Diagram 3.6, step 4b. If the entry type does not match the type requested in the command and the entry is not a cluster or an alternate index, 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 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, control goes to Diagram 3.6, step 4b; otherwise, control goes to Diagram 3.6, step 4a for the next *entryname* in the LISTCAT command. If the UCATLG return code is nonzero, ENTPROC also writes a message. Control goes to Diagram 3.6, step 4a 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 3.6, step 4b. If the return code from UCATLG indicates password verification failure or lack of workspace, GNXTPROC writes a message and control goes to Diagram 3.6, step 4a 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 3.6, step 5 to terminate the LISTCAT command.

Diagram 3.7. PARM FSR



Extended Description for Diagram 3.7

IDCPM01

Procedures: 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.

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, the Access Method Services default margins. MARGPARM puts the margin values in the first two halfwords of the Reader/Interpreter Historical Data Area.

IDCPM01

Procedure: GRPHPARM

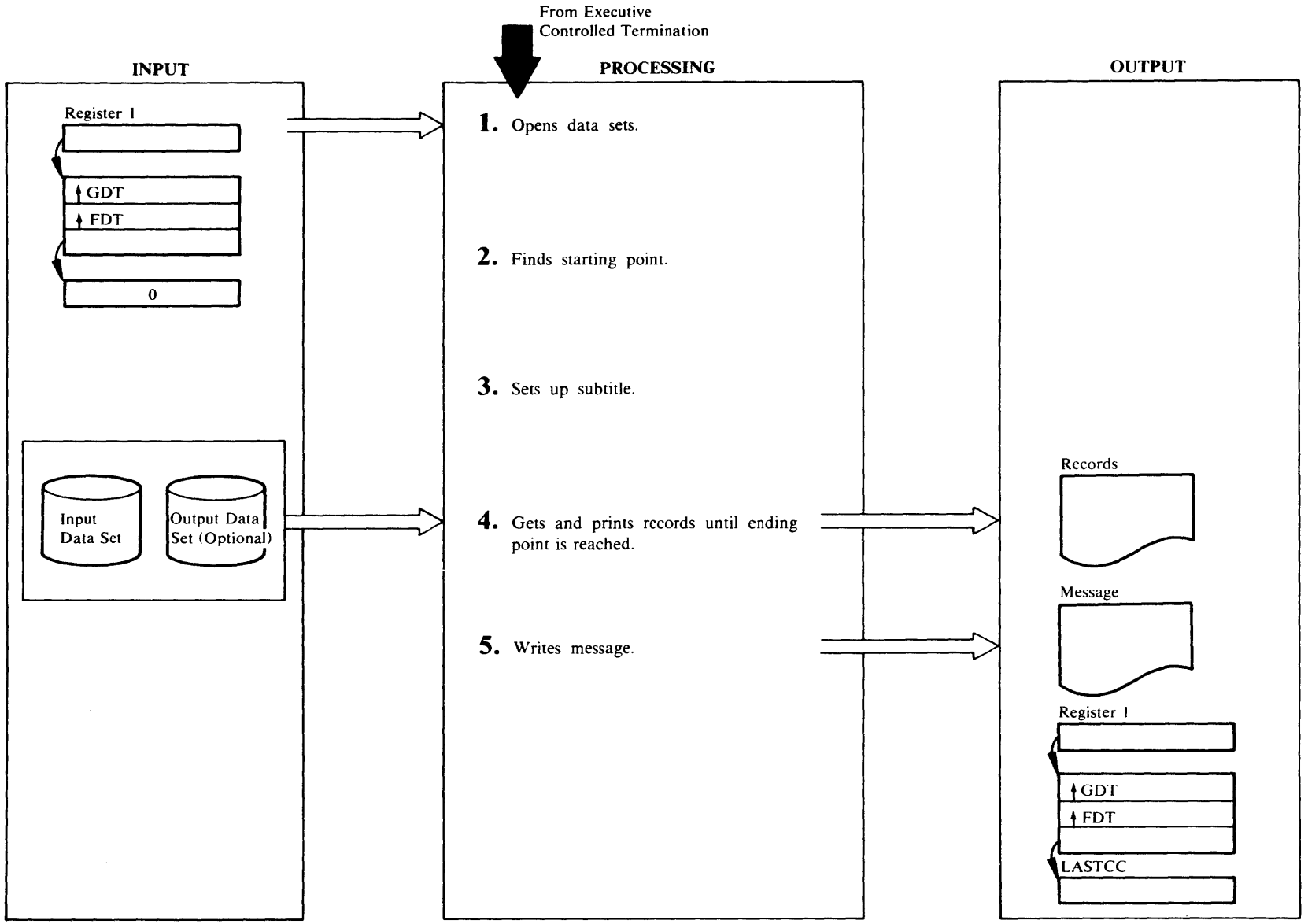
3 GRPHPARM puts the GRAPHICS parameter (CHAIN or TABLE) in a Text Processor Print Control Argument list. GRPHPARM issues a UREST macro for the Text Processor to use the new chain or table with Access Method Services output. The CHAIN parameter specifies one of several graphic character sets available.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

IDCPM01

Procedure: IDCPM01

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

Diagram 3.8. PRINT FSR



Extended Description for Diagram 3.8

IDCPR01

Procedure: IDCPR01

1 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 sequential record retrieval. IDCPR01 puts any ENVIRONMENT parameters in the OPNAGL. The input data set can be a VSAM catalog. 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 nonzero, IDCPR01 writes a message and terminates the PRINT command.

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

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.

IDCPR01

Procedure: TEXTPSET

3 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.

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. Retreiving 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.

   a. IDCPR01 issues a UGET to obtain a logical record. If the return code from the UGET macro is nonzero, 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.

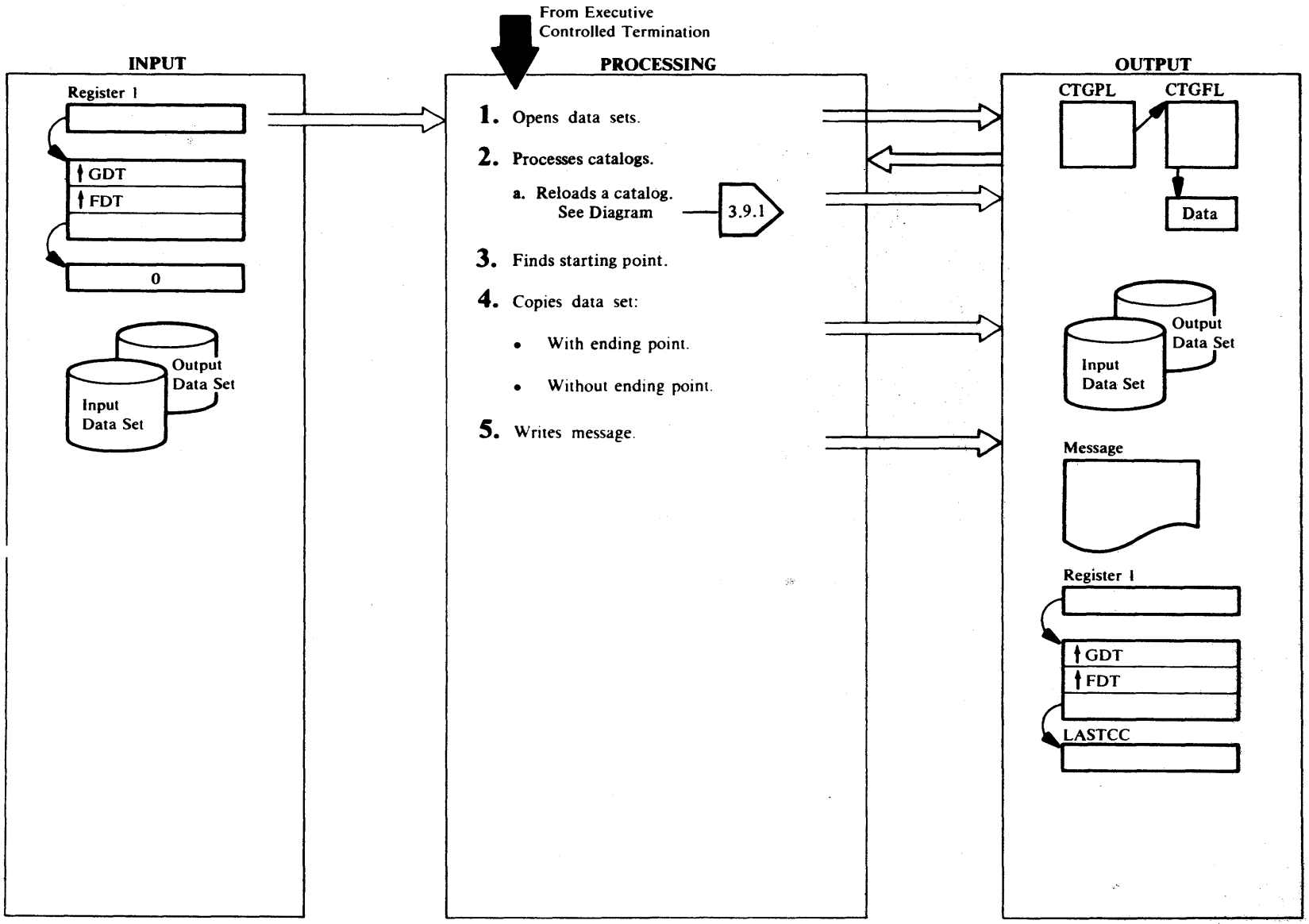
   b. IDCPR01 issues a UPRINT to print the logical record just obtained. A minimum of 3 lines is printed for each logical record from the input data set. The first line printed contains the record identification: key, address, sequence number (nonVSAM except ISAM) or relative record number. The relative 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.

IDCPR01

Procedure: IDCPR01

5 IDCPR01 writes a message with LASTCC to SYSLST. IDCPR01 issues a UCLOSE macro to close the input data set and any output data set other than SYSLST. SYSLST is not closed. Control returns to Executive Controlled Termination

Diagram 3.9. REPRO FSR



Extended Description for Diagram 3.9

IDCRP01

Procudures: IDCRP01

1 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, IDCRP01 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. IDCRP01 issues one UOPEN macro that opens both the input and output data sets. If the return code from the UOPEN macro is nonzero, IDCRP01 writes a message on SYSLST 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 SYSLST and the REPRO command is terminated.

IDCRP01

Procedures: VERIFYC, CATRELOD, TRUENAME, CATRANS, CNVRTCI, CATCOMP

2 If neither the input nor the output are VSAM data sets, processing continues with step 3. Each VSAM data set is checked and verified to see if it is a catalog. If the output data set is not a catalog, processing continues with step 3. If the output data set is a catalog, the catalog reload switch, CATRELSW, is set on. The REPRO command is checked to see if beginning or ending delimiters were specified. If any were specified, a message is issued, processing is set for termination, and control goes to step 5. If no delimiters were specified, a catalog reload function is assumed, a message is issued, and the reload function is initiated. See Diagram 3.9.1.

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.

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 nonzero, 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 4 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 nonzero, IDCRP01 checks the return code for a recoverable error. After a non-recoverable error or 4 recoverable errors, copying stops.

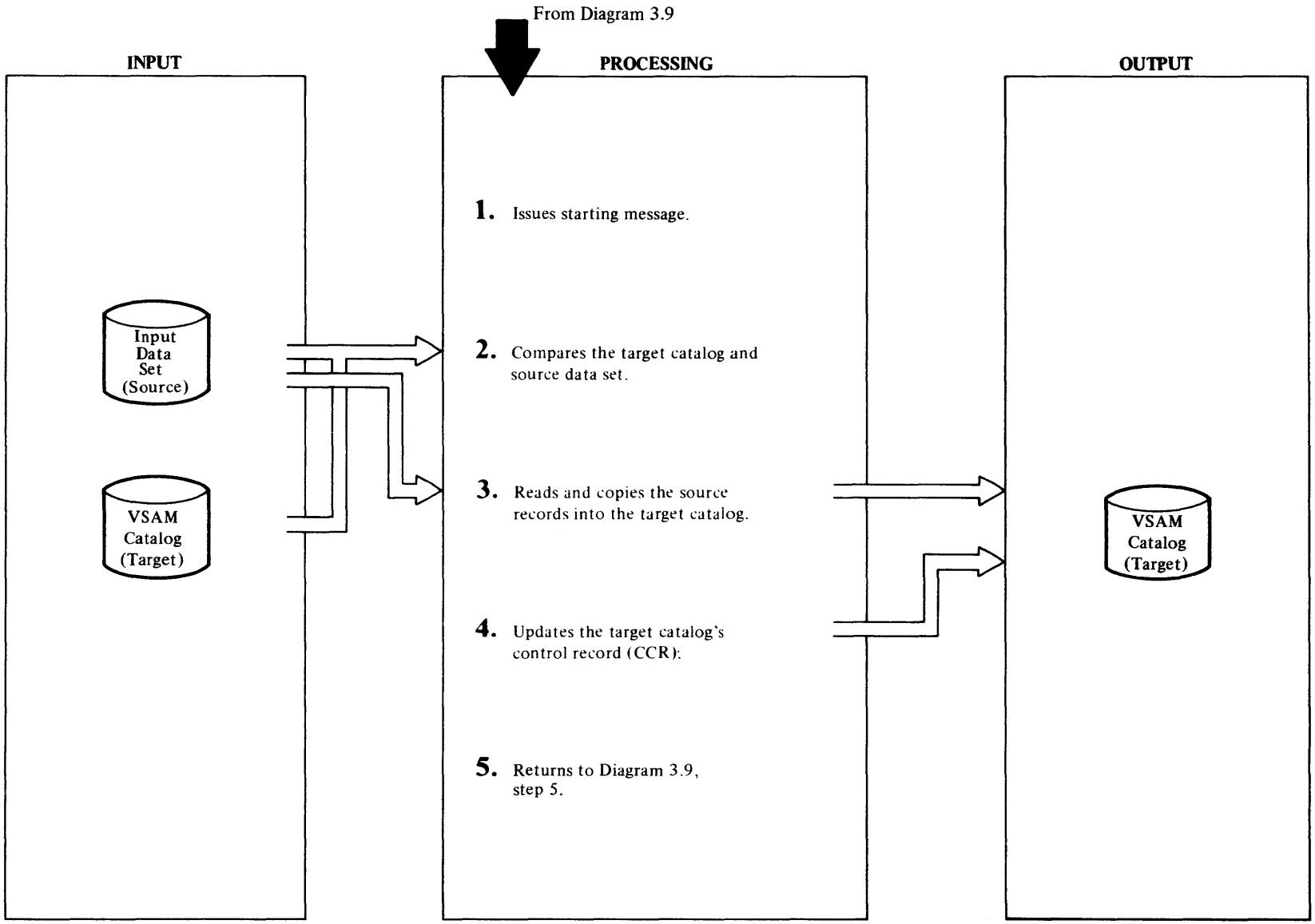
b. If no ending point is specified in the REPRO command, IDCRP01 issues a UCOPY macro to copy the input data set to the last record.

IDCRP01

Procedure: IDCRP01

5 IDCRP01 writes a message with LASTCC to SYSLST. It also closes the input and output data sets with one UCLOSE macro. Control returns to Executive Controlled Termination.

Diagram 3.9.1 REPRO FSR – Catalog Reload



Extended Description for Diagram 3.9.1

IDCRP01

Procedure: IDCRP01

1 The message says that catalog reload had begun.

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

Procedures: 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 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.

IDCRP01

Procedure: CATRELOD

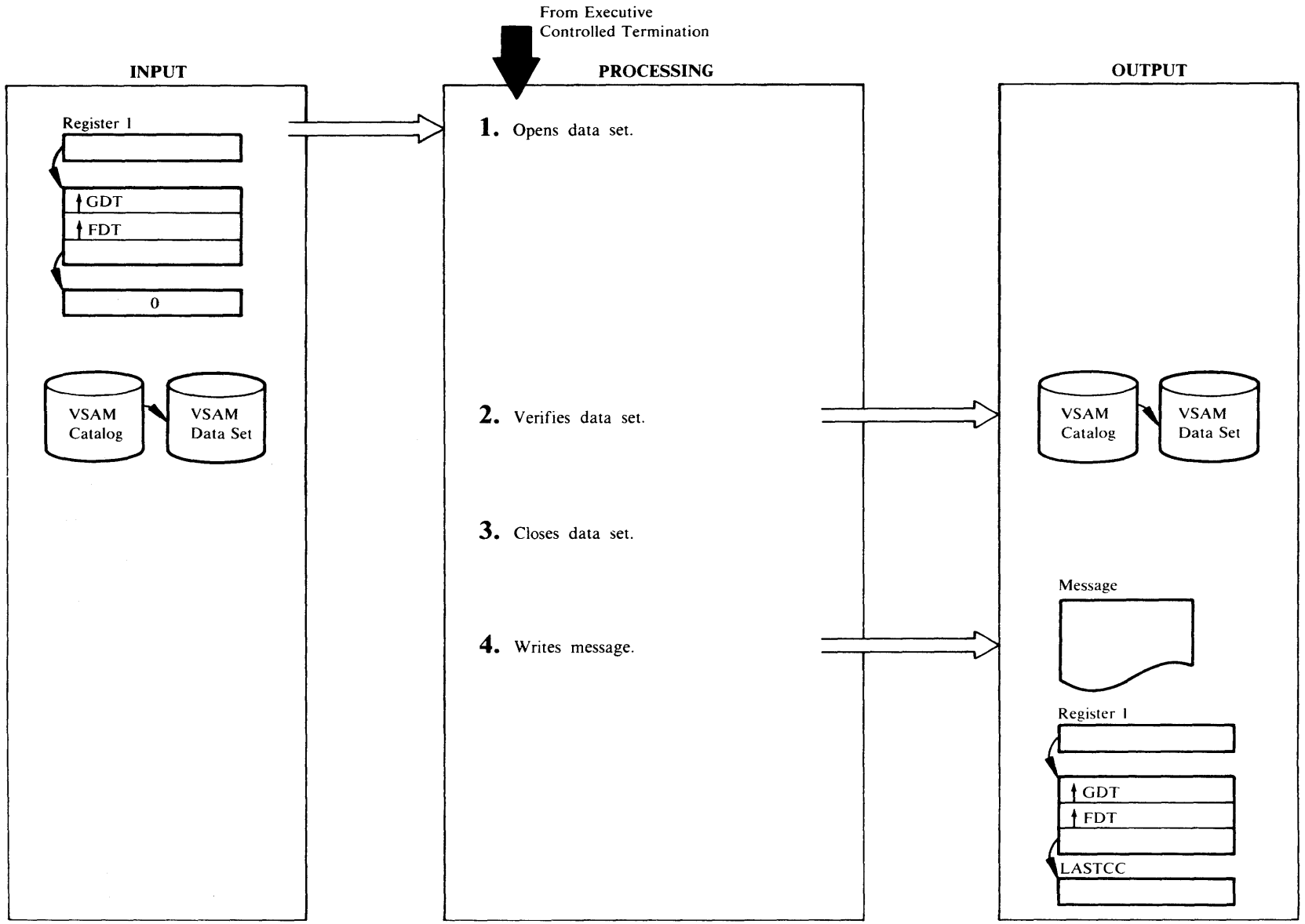
4 After both backup and target records have been processed sequentially by key to the end-of-file, one more record needs to be updated.

- 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.

5 Control passes to Step 5, Diagram 3.9, step 5, to print final messages.

Diagram 3.10. VERIFY FSR



Extended Description for Diagram 3.10

IDCVY01

Procedures: OPENPROC, IDCVY01

1 OPENPROC builds an OPNAGL to open the VSAM data set specified by the data set or FILE parameter 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 4.

IDCVY01

Procedure: IDCVY01

2 IDCVY01 issues a UVERIFY macro to verify the data set.

IDCVY01

Procedure: TERMPROC

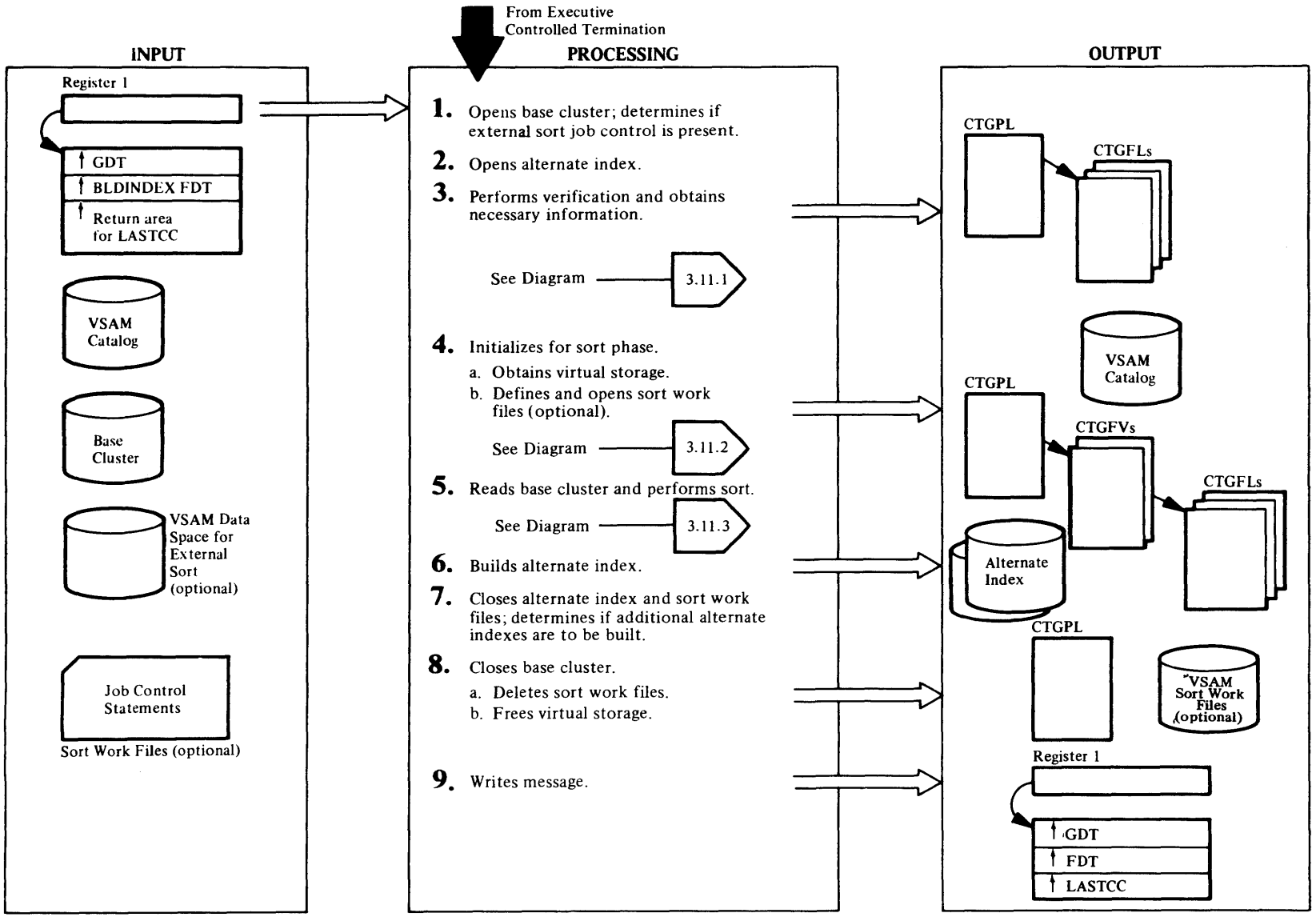
3 TERMPROC issues a UCLOSE macro to close the data set. If the close was not successful, LASTCC is 4.

IDCVY01

Procedure: IDCVY01

4 IDCVY01 prints a message containing LASTCC. Control goes to Executive Controlled Termination, Diagram 4.1.

Diagram 3.11. BLDINDEX FSR



Extended Description for Diagram 3.11

IDCBI01

Procedures: OPENPROC, JCPROC

1 IDCBI01 calls OPENPROC to build an OPNAGL and issue a UOPEN to open the base cluster for input. OPENPROC sets the INFILE dname or INDATASET entry name in the OPNAGL. OPENPROC also sets 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 data set identification , BLDINDEX will perform an external sort using two VSAM entry sequenced data sets. If you provide DLBL/EXTENT statements, you must also provide the following parameters:

Filename	-	As provided via the WORKFILES parameter, or defaulted to IDCUT1 and IDCUT2
File-ID	-	Required
Volume Serial Numbers	-	Required; must specify volume(s) containing VSAM data space accessable via a currently available catalog.
Access Method	-	'VSAM' 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.

If WORKVOLUMES is specified, two data set names are generated and catalog management is called by DEFPROC to define the two work data sets. If neither WORKFILES, WORKVOLUMES, nor default JCL is provided, DEFAULTVOLUMES is utilized through catalog management. Parameter lists for DEFPROC which do the DEFINE are built now.

IDCBI01

Procedures: MAINPROC, OPENPROC

2 Steps 2 through 7 are performed for each alternate index specified in the OUTFILE 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. OPENPROC indicates the OUTFILE dname or OUTDATASET entry name in the OPNAGL. 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.

IDCBI01

Procedures: MAINPROC, LOCPROC

3 In order to accomplish validity checking and obtain required information, MAINPROC calls LOCPROC to issue VSAM catalog locates. See Diagram 3.11.1.

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.

IDCBI01

Procedures: MAINPROC, 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 3.11.2.

IDCBI01

Procedures: MAINPROC, CNTLPROC

5 MAINPROC calls CNTLPROC to read the base cluster and control the sort-merge process. See Diagram 3.11.3.

IDCBI01

Procedures: 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 3.11.3 for BLDPROC and MERGPROC processing.

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 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.

IDCBI01

Procedures: 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.

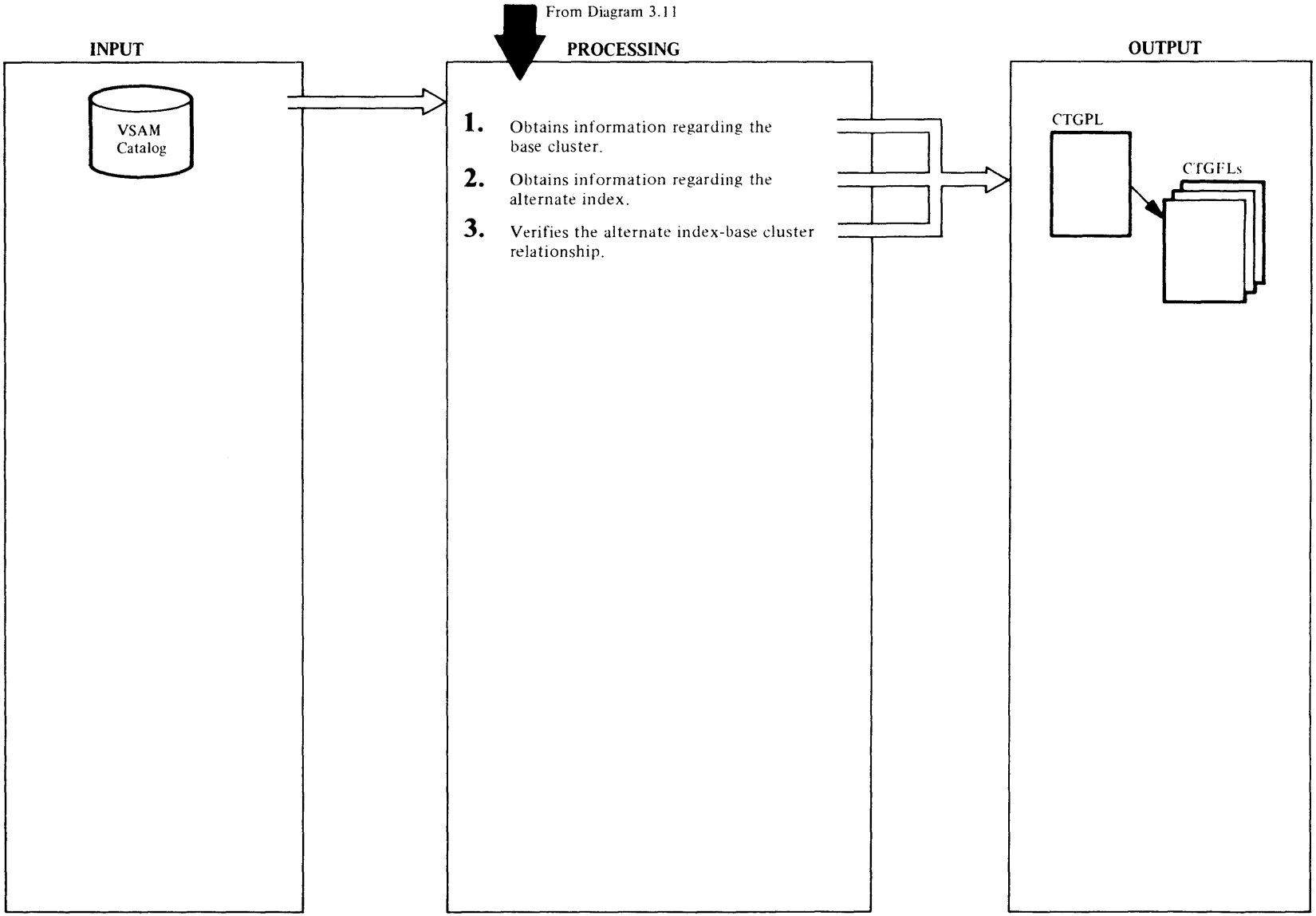
A UCATLG macro is issued by DELTPROC to delete each sort work file. TERMPROC issues a UFPOOL to free all remaining core obtained via UGPOOL.

IDCBI01

Procedure: TERMPROC

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

Diagram 3.11.1. BLDINDEX FSR – Get Information and Verify

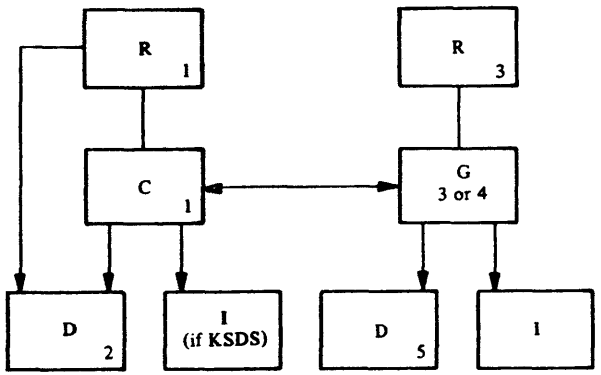


Extended Description for Diagram 3.11.1

IDCBI01

Procedures: 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, an additional locate to reach the G object will be required (Locate 4) is done.

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.

Locate 1

Locate 1 retrieves the associations of the name specified via INFILE. 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 file ID specified by the caller on the INDATASET parameter or in the job control pointed to by the INFILE 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 3.11). **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

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

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.

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 on the OUTDATASET parameter or via OUTFILE 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.

IDCBI01

Procedures: LOCPROC, CATPROC

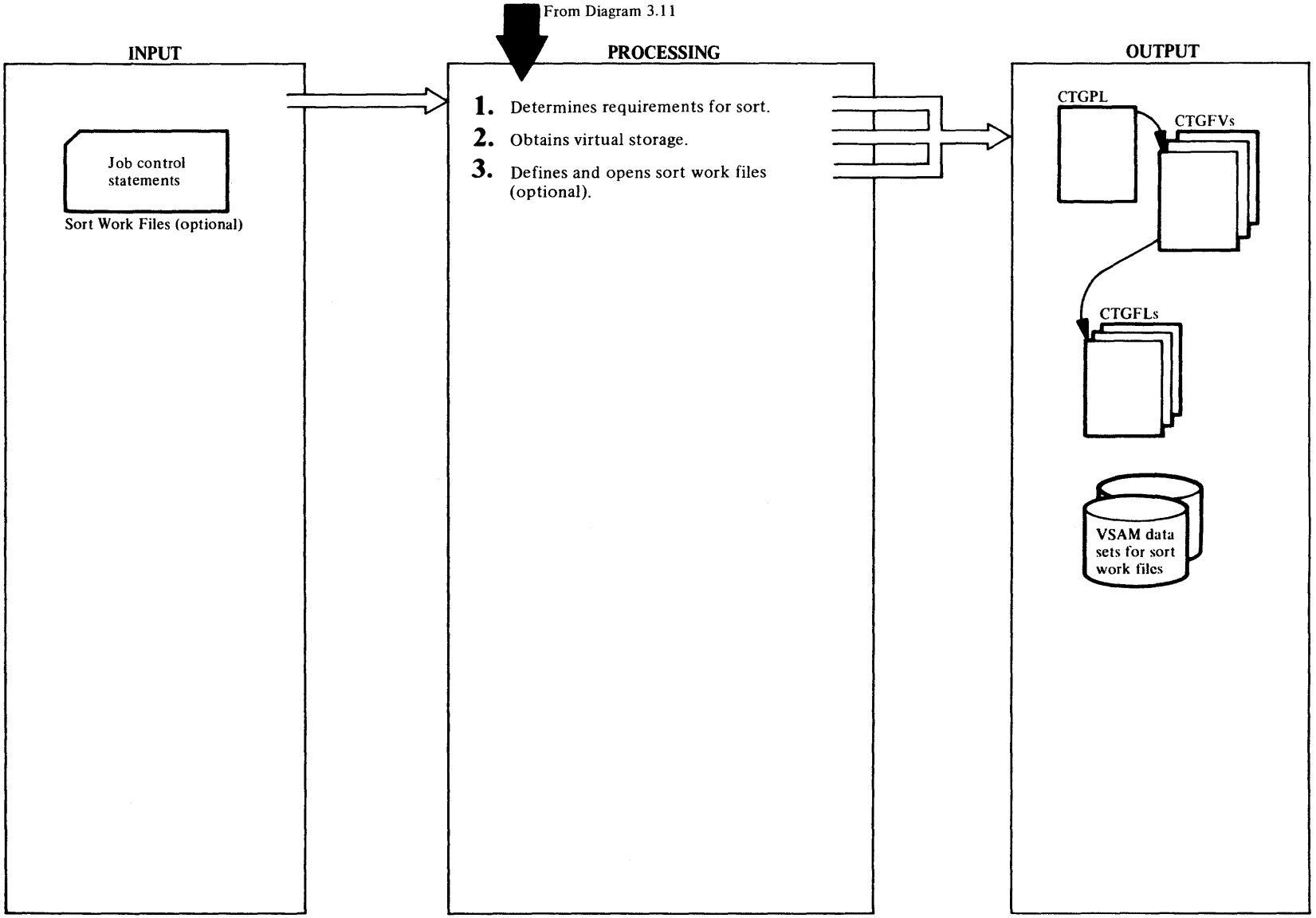
3 LOCPROC must now verify that the alternate index specified by the caller is in fact related to the base cluster specified. LOCPROC compares the control interval number of the base cluster saved from Locate 2 of the control interval number of the third association returned from Locate 3 or 4. This should 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 3.11 where control will be passed to Step 4 or Step 7 depending on the setting of LASTCC.

Diagram 3.11.2. BLDINDEX FSR – Obtain Resources and Sort Initialization



Extended Description for Diagram 3.11.2

IDCBI01

Procedures: 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:

$$\text{Sort Record Size} = \text{Alternate Index Key Length} + \text{Prime Key Length (KSDS) or 4 (ESDS)}$$

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

$$\text{Total number of 2K Blocks} = \left\{\frac{\text{\# of Records in Base Cluster}}{\text{\# of Records per Block}}\right\} + 10$$

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.

IDCBI01

Procedures: 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:

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

$$\text{Heap Size} = \text{RSA Capacity} * 4$$

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 message and returns control to IDCBI01, Step 7 (via MAINPROC).

IDCBI01

Procedures: 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.

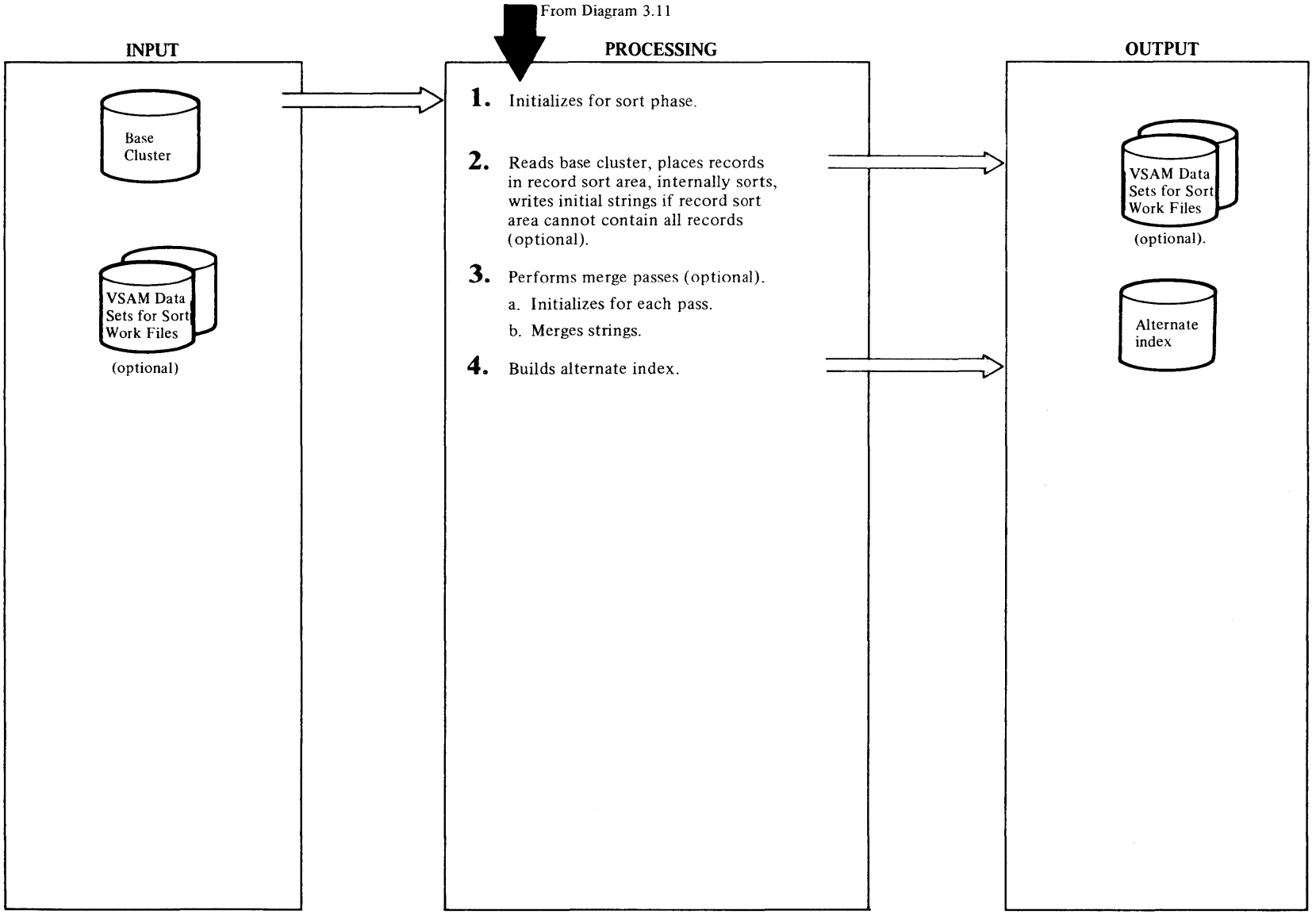
DEFPROC determines if large enough sort work files exist from a previous sort and, if so, bypasses the define process.

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 file-ID and the data FVT references the volume serial numbers obtained via (a) UIOINFO from the sort work job control statements, (b) built from the WORKVOLUMES parameter, or (c) built from the null volume list for the default-volume-define function of Catalog Management. Space allocation is in records: primary, the number of 2K blocks calculated by INITPROC; secondary, 10% of primary, plus 10. 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, sets 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 3.11 where control will be passed to Step 5 or Step 7 depending on the setting of LASTCC.

Diagram 3.11.3. BLDINDEX FSR – Sort-Merge and Build Alternate Index



Extended Description Diagram 3.11.3

IDCBI01

Procedure: CNTLPROC

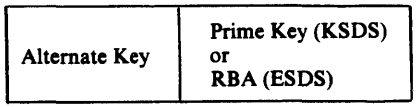
1 CNTLPROC initializes factors which will be used during the sort-merge including pointers to the record sort area (RSA), and the table of pointers which is used during the sort. CNTLPROC also initializes the output buffer with an RDF and CIDF in the event an external sort is performed (the sort work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

IDCBI01

Procedures: 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.

IDCBI01

Procedures: 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.

IDCBI01

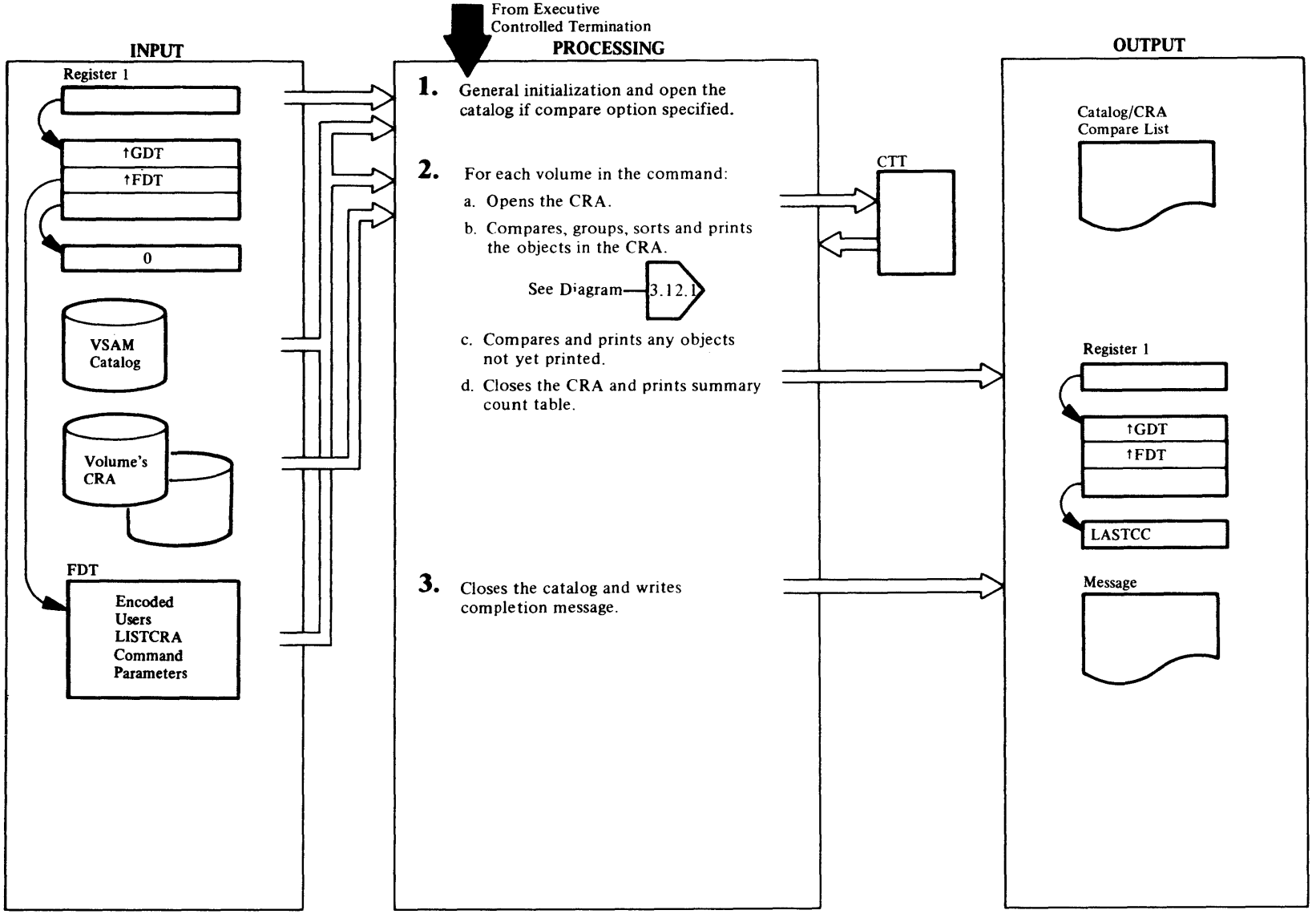
Procedures: BLDPROC

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 3.11, Step 7.

Diagram 3.12. LISTCRA FSR



Extended Description for Diagram 3.12

IDCLR01

Procedures: AATOPLR, INITLZE, CATOPEN, ERROR

1 Routine addresses and the UOPEN argument are initialized in the work area. If the COMPARE option was specified, a UOPEN is issued for the catalog identified by the CATALOG *dname* parameter or by the CATALOG *catname* parameter (*dname* parameter omitted). If the OPEN is successful, a UVERIFY is issued and the catalog name is obtained using Access Method Services field management (IDCRC04).

The volume serial is obtained via IDCRC04 and the catalog is locked to prevent it and its associated CRAs from being reset. If the COMPARE option was not specified on the OPEN of the catalog failed, the no compare indicator is set.

IDCLR01, IDCLR02, IDCRC04

Procedures: AATOPLR, CRAOPEN, PRTVOL, INTSORT, MEMSORT, DOVSAM, PRTVSAM, DOOTHR, PRTOTHR, PRTFIFO, GETPRT, PRTCMP, CLENCRA, SUMIT

2 For each of the CRAs identified by the INVOLUMES *volser* parameter or INFILE *dname* parameter, the following is repeated:

a. If the INFILE parameter was specified, a UIOINFO is issued to obtain the CRA volume serial. The UOPEN parameter list is set up with the *volser* and the catalog master password and the UOPEN and UVERIFY are issued for the CRA. If the COMPARE option was specified, the catalog and its CRAs are locked (UENQ) to prevent any concurrent updates. If they are successful and there is a match on the owning catalog name, 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.

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 3.12.1.

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.

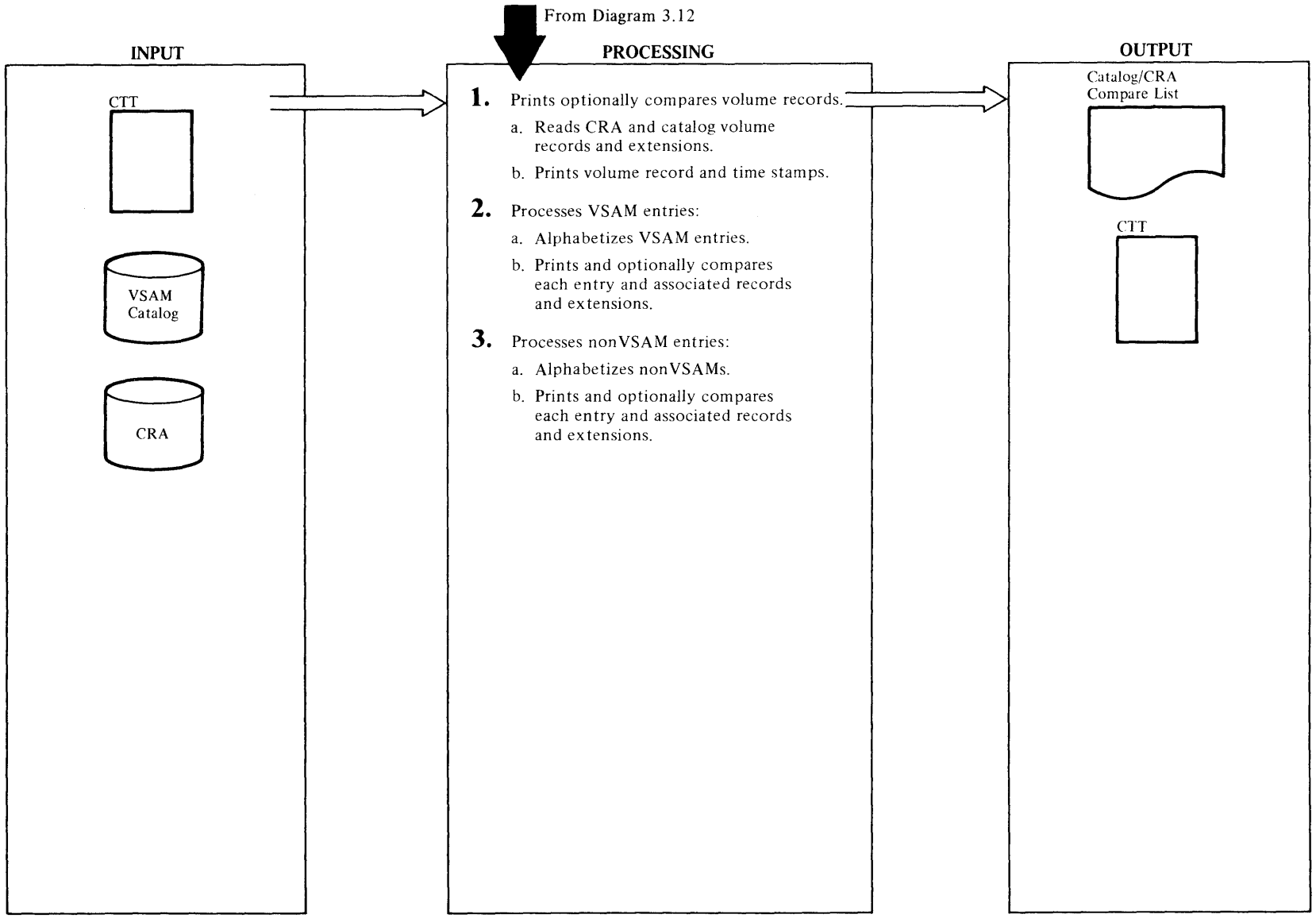
d. GETPRT is used to get the CRA copy of any other records, and the catalog record, if compare. These are printed and compared by PRTCMP. When all objects have been processed, the UDEQ macro is issued to release the update lockout for the catalog, the CRA is closed by CLENCRA, and a summary is printed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

IDCLR01

Procedures: AATOPLR, CLEANUP

3 The UCLOSE macro is issued to close the catalog data set and the UDEQ macro is issued to release the reset 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 3.12.1. LISTCRA FSR – Process CRA



Extended Description for Diagram 3.12.1

IDCLR01, IDCLR02, IDCRC04

Procedures: 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 timestamps from the CRA volume record and on the CRA volume and, if compare, in the catalog records are printed by PRTTIME.

IDCLR01, IDCLR02, IDCRC04

Procedures: 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.

IDCLR01, IDCLR02, IDCRC04

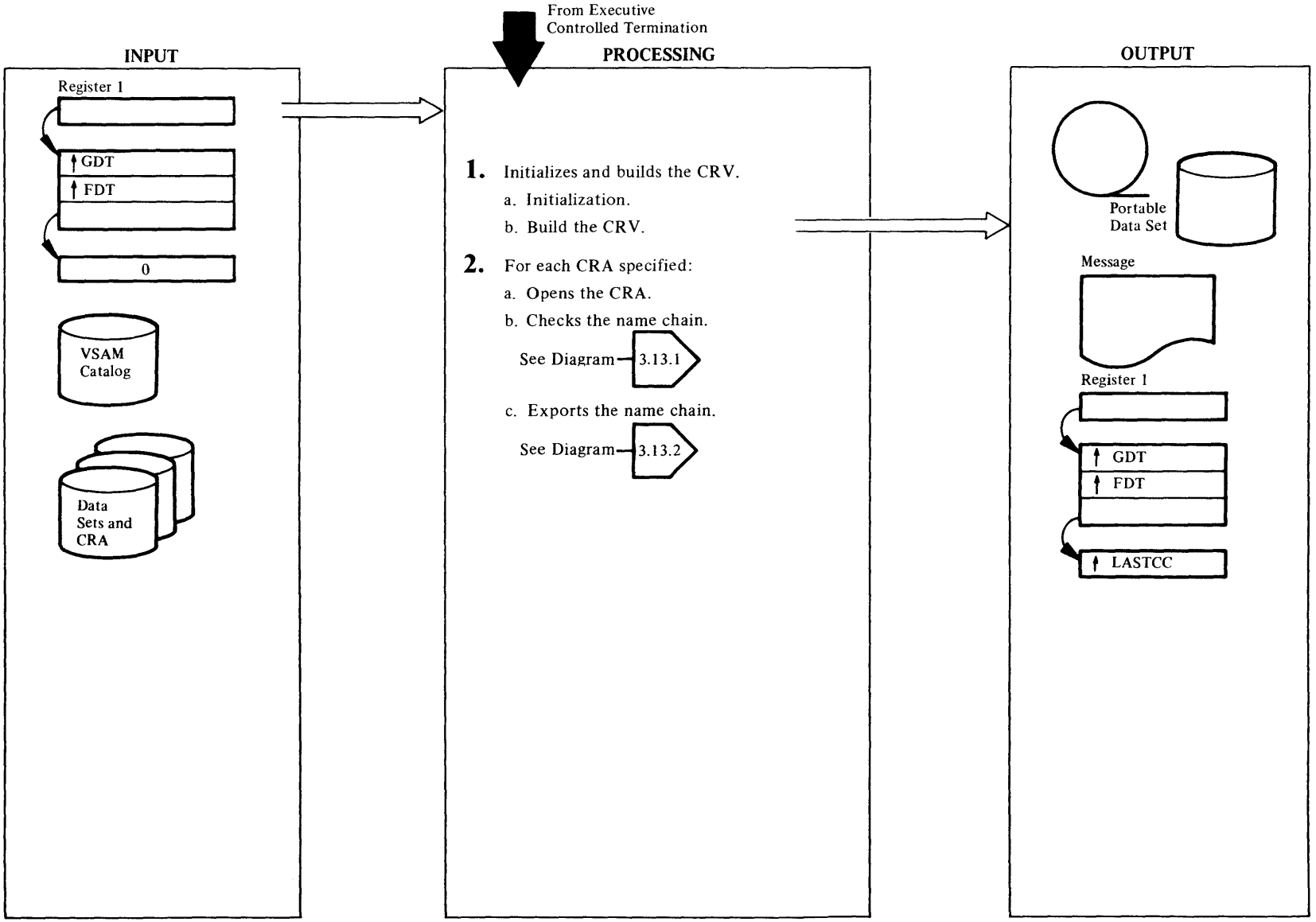
Procedures: INTSORT, MEMSORT, DOOTHR, PRTOTHR, GETPRT, VERTEXT, INTVEXT, TCICTCR, BLDVEXT, SUMIT, PRTMCWD, UPRINT, PRTOJAL, INTASOC

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.

For all of the steps above, GETPRT uses UGET to read the CRA record and the catalog record, if compare. IDCRC04 is used to extract all necessary fields from the records. 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 volume.

Diagram 3.13 EXPORTRA FSR



Extended Description for Diagram 3.13

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. If the CRA dname parameter form is specified, for each CRA volume, UIOINFO is used to obtain the volume serial number (for CRA dname1 option only), which is placed in the VTT. BUILDNAME 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.

IDCRC01, IDCRC02, IDCRC03, IDCRC04

Procedures: OPENCRA, OPEN, TIMESTMP, SCANCRA, NAMETABL, DIRECT, EXTRACT, ERRCK, MESSAGE, COMPNAME, CKCATNM, CKNAMES, DUPNAMCK, SYNCH, OBJVOLCK, CRAOPEN, EXPORTDR, OPENCRA, MESSAGE

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 calls TIMESTAMP, which issues the UIOINFO macro to get the CRA volume timestamp and place it into the VTT and to get the device characteristics and place them in the CRV. 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 the CRA record. ERRCK calls MESSAGE if an error occurred in this procedure. For IDCRC04 see Diagram 3.13.1.

b. CKNAMES is called to perform the following functions for each potentially exportable entry using EXTRACT:

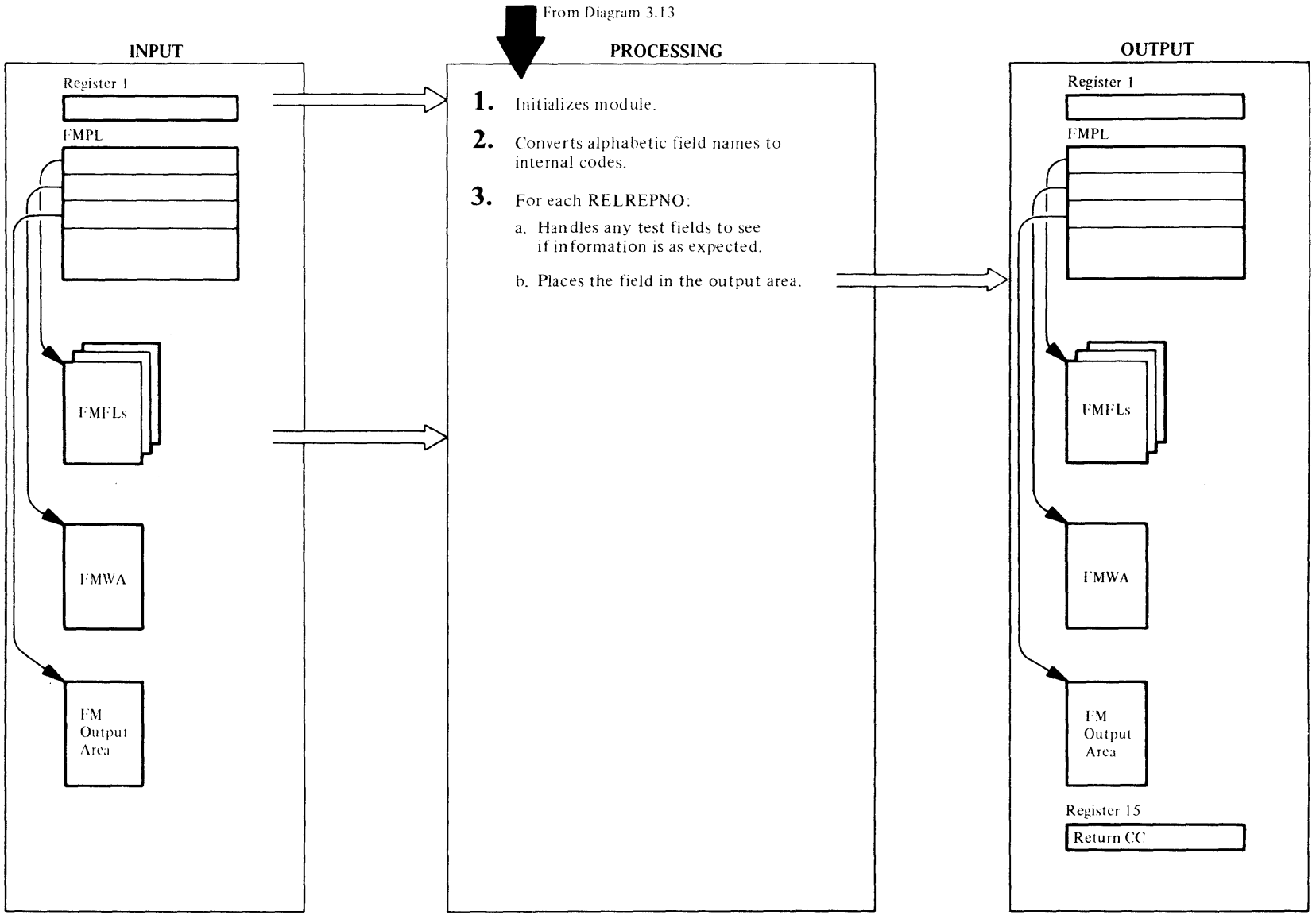
- Get the master password for VSAM entries.
- Locate and flag to bypass export any OS/VS2 paging data sets.
- Collect the data and index associated CI numbers for VSAM entries.
- Locate and flag to bypass data copy any VSAM entries that have no data (high-used RBA is zero).
- Locate and flag to bypass for export any NOCIFORMAT SAM ESDS entries.
- Locate and flag to bypass for export any SAM ESDS entries if the SAM ESDS feature is not installed.
- Collect the largest VSAM LRECL (RECORDMODE) or the largest data control internal size (CIMODE) for the nonempty files to be exported.
- Locate and flag to bypass export any OS/VS2 GDG bases.

For those entries bypassed for export but named in the ENTRIES parameter, an error message is printed and LASTCC is set to 8. For SAM ESDS entries not named in the ENTRIES parameter, a warning message is printed and LASTCC is set to 4.

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 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.

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 (however, if the name list is empty, the "nothing to export" message is issued instead). It checks the name chain for the CRA for null entries and nonmatches and marks them not exportable. It initializes the export table for each valid entry and calls IDCRC02 to export the entry. If the FDT parameter CIMODE was specified, a CIMODE flag is set in the export table. ENVIRONMENT parameters are obtained from the FDT and placed in the export table. See Diagram 3.13.2 for a description of IDCRC02. When the Export Driver (IDCRC02) returns, then the completion or error message is printed and processing continues with the next entry in the name chain for the CRA.

Diagram 3.13.1. EXPORTRA FSR – Field Management



Extended Description for Diagram 3.13.1

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).

IDCRC04

Procedures: 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.

IDCRC04, IDCLR02, IDCRC03

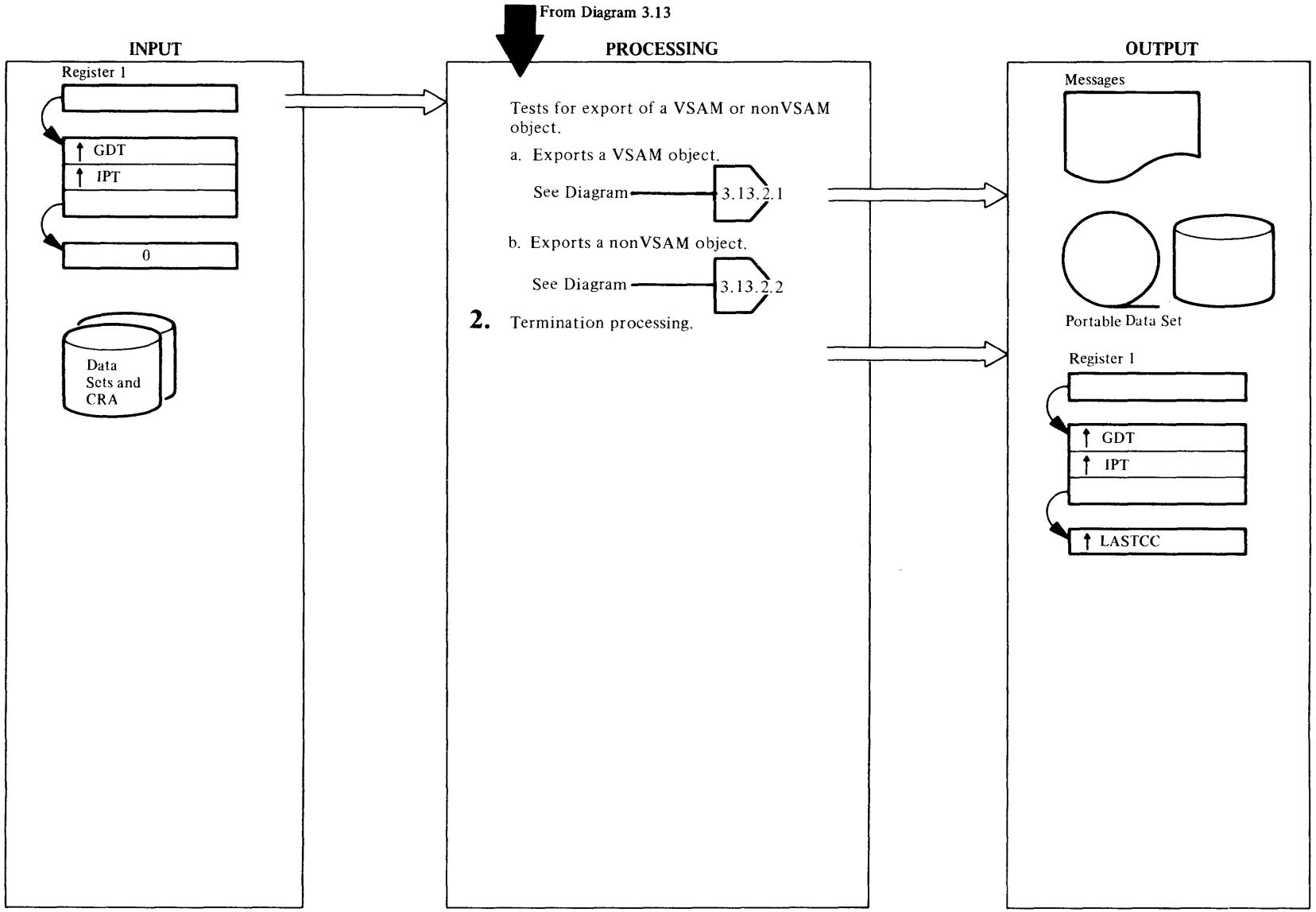
Procedures: PSCNF, PTSTS, PGVAL, PGREC, PCKLC, PEXPT, PLNRV, PTCMP, PLOC2, PGREP, PSHIN

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 3.13.2. EXPORTRA FSR – Driver



Extended Description for Diagram 3.13.2

IDCRC02

Procedures: OPENPROC, CLUSPROC, SAVEPROC, RECPROC, PUTPROC, NVSMPROC, ADSPROC, ALSPROC

1 IDCRC02 tests the input parameter list for export of a VSAM or nonVSAM object. OPENPROC opens the portable data set for output. ENVIRONMENT parameters from the export table are placed in the OPNAGL for UOPEN processing. If the object to be exported is a VSAM object then step 1.a is done; if it is a nonVSAM object, then step 1.b is done.

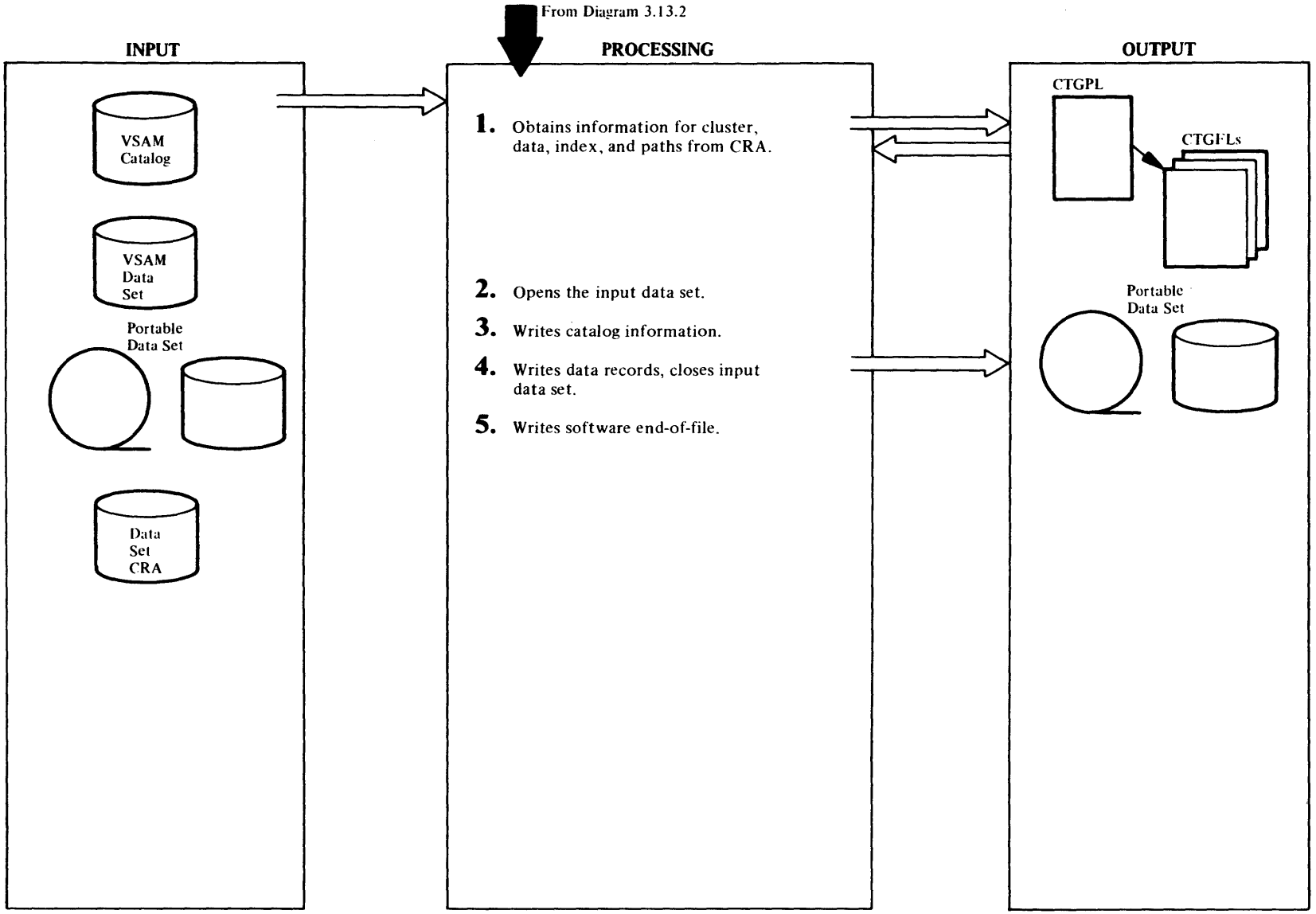
   a. CLUSPROC gets catalog information for the cluster, data, index and paths from the CRA. SAVEPROC holds the control records containing the catalog information until catalog processing is completed, then writes them to the portable data set. OPENPROC opens the cluster data for input. RECPROC copies the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

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 3.13.2.1. EXPORTRA FSR – Export VSAM Data Set



Extended Description for Diagram 3.13.2.1

IDCRC02

Procedures: 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, PASSATMP, USVRMDUL, USERAREC, LOKEYV, HIKEYV, VOLSER, AMDSBCAT, EXCPEXIT, RCATTR, 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.

A timestamp record is constructed as the first control record. Information is placed into it indicating the number of objects; whether the data set is KSDS, SAM ESDS, NOALLOCATE, or empty; and whether export CIMODE was specified.

IDCRC02

Procedure: OPENPROC

2 OPENPROC issues the UOPEN macro to open the VSAM data set for input and verifies the open. OPENPROC triggers CIMODE processing by setting the "export CIMODE" flag and the "CNV processing" flag in the OPNAGL of the input data set.

IDCRC02

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.

IDCRC02

Procedure: RECPROC

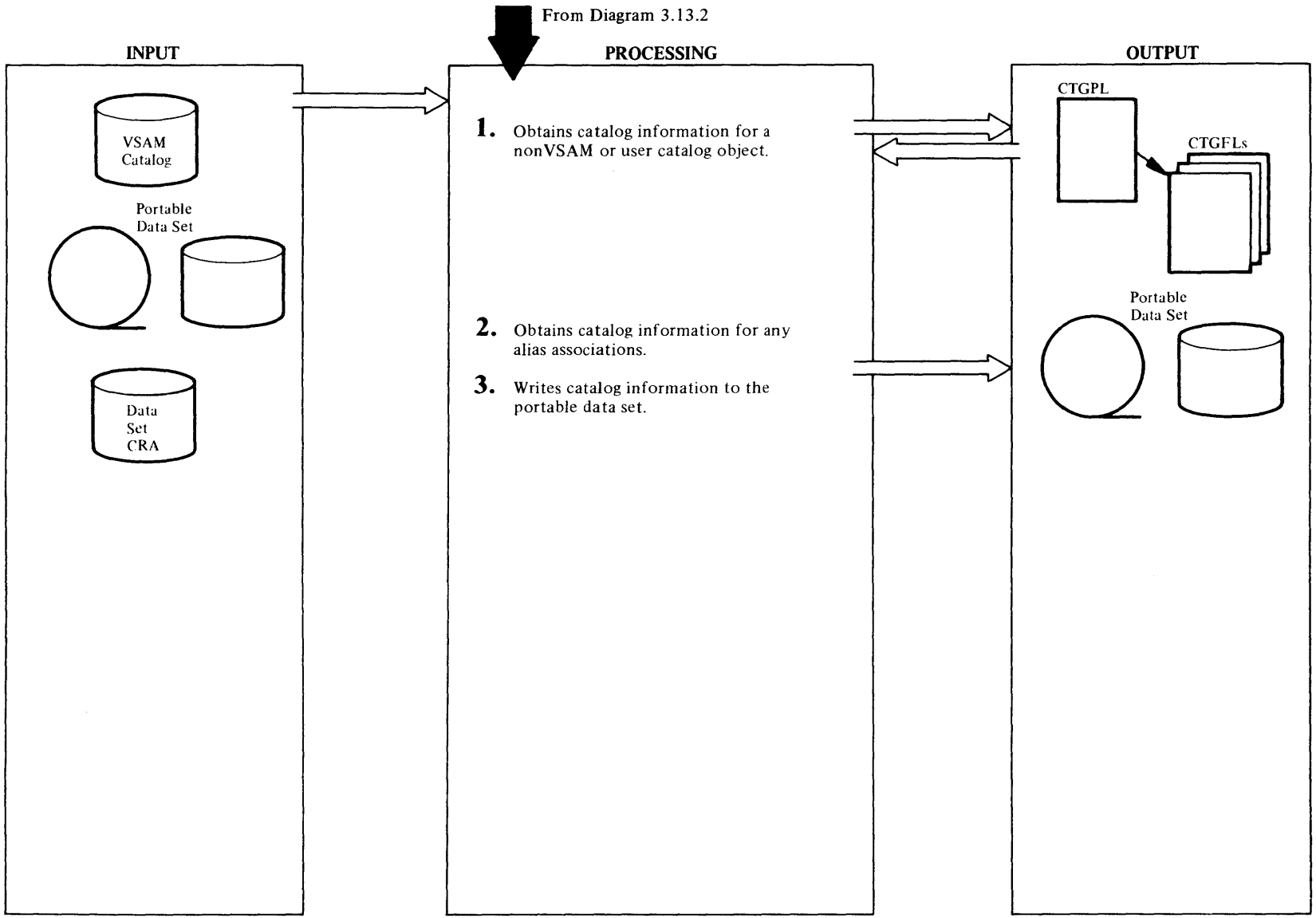
4 RECPROC copies the data to the portable data set and closes the input data set.

IDCC02

Procedure: CLUSPROC

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

Diagram 3.13.2.2. EXPORTRA FSR – Export NonVSAM



Extended Description for Diagram 3.13.2.2

IDCRC02

Procedures: LOCPROC, CTLGPROC

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

ENTYPE, ENTNAME, VOLSER, DEVTYP, NAMEDS, CATACB

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

IDCRC02

Procedures: LOCPROC, CTLGPROC

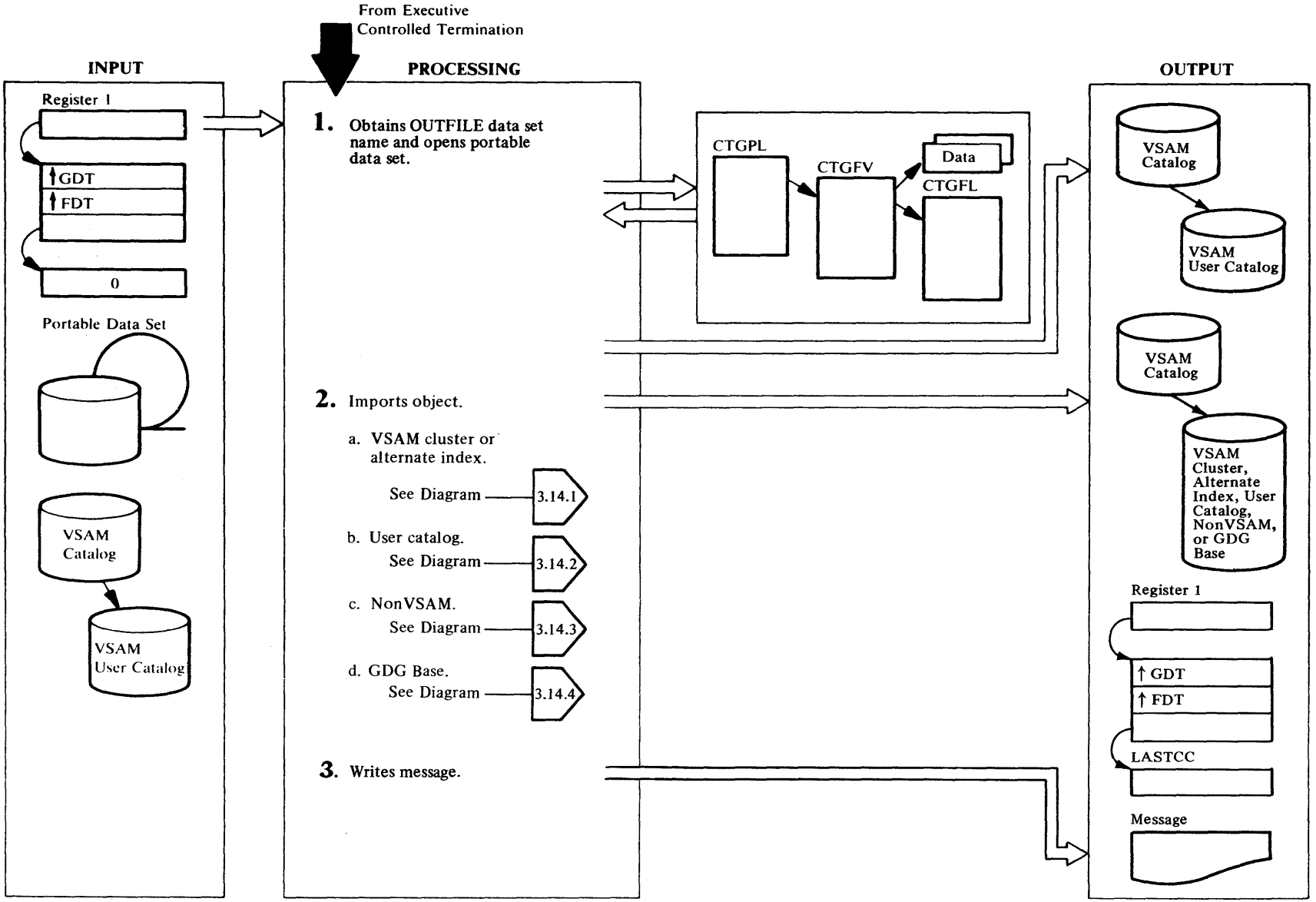
2 LOCPROC builds a CTGPL and multiple CTGFLs for any alias associations. A CTGFL is built for ENTYPE and ENTNAME catalog fields. CTLGPROC issues a UCATLG to obtain the catalog information.

IDCRC02

Procedures: NVSMPROC, ALSPROC

3 NVSMPROC and ALSPROC write control records containing the catalog information to the portable data set after catalog processing is completed. The first record written is the timestamp control record. It is flagged if export CIMODE processing has been specified.

Diagram 3.14. IMPORTRA FSR



Extended Description for Diagram 3.14

IDCRM01

Procedure: IDCRM01, OPENPROC

1 If the OUTFILE parameter is present, IDCRM01 issues a UIOINFO to obtain the data set name coded on the DLBL job control statement associated with the OUTFILE parameter (to be used later by ALTPROC). PENPROC builds an OPNAGL and issues a UOPEN to open the portable data set. User specified tape label and rewind options are placed in the OPNAGL for UOPEN processing. 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, a special UCLOSE is issued which reallocates sufficient space for the record size. An actual close of the portable data set is not done.

IDCRM01

Procedures: IDCRM01, CLUSPROC, UCATPROC, NVSMPROC, CLUSPROC, GDGPROC

2 For each item on the portable data set, IDCRM01 reads a timestamp record and prints a message indicating the time and date of the EXPORTRA operation. The timestamp record also indicates whether the portable data set is in CIMODE or RECORDMODE format and whether the file being imported is empty.

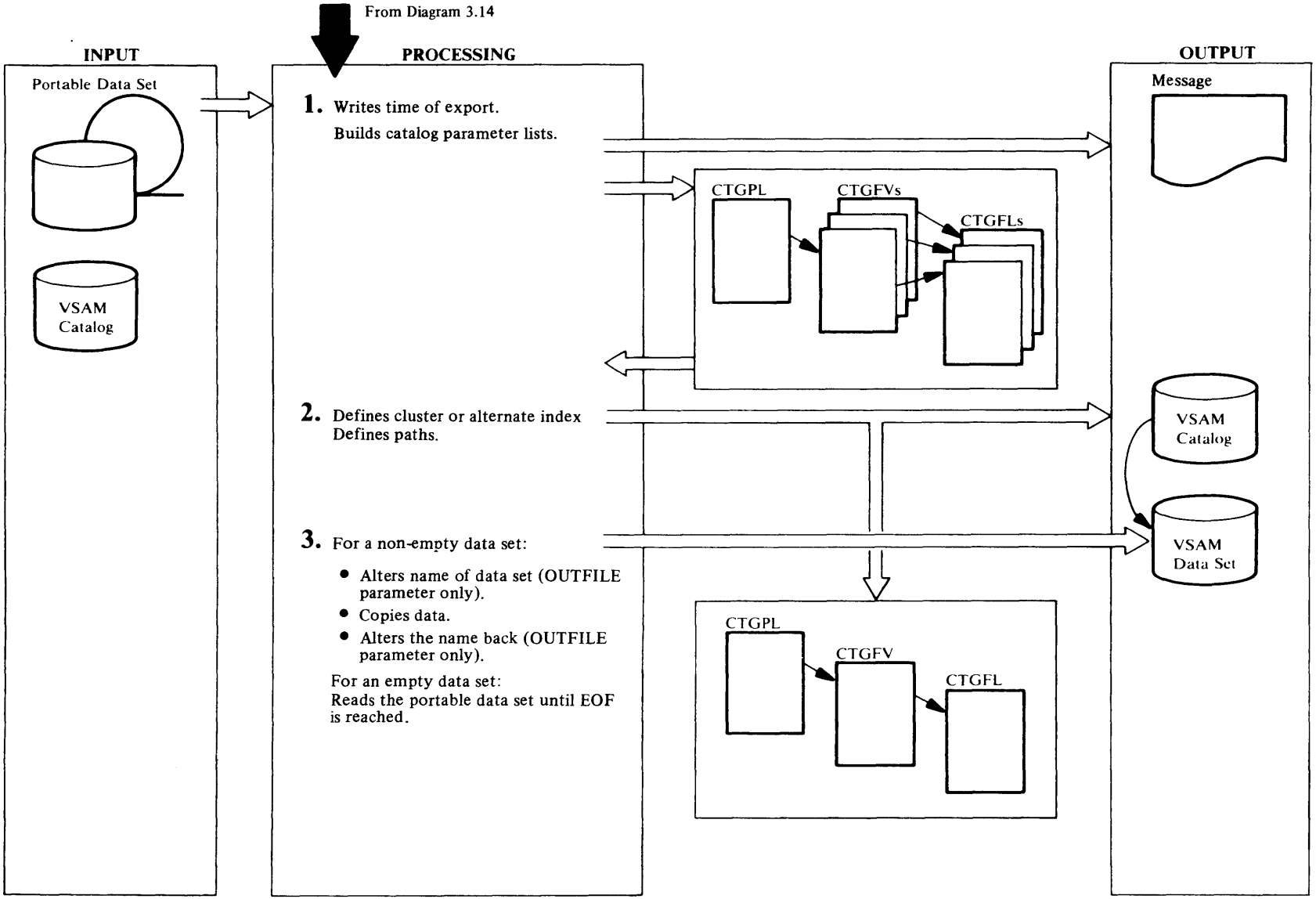
On the basis of the timestamp record, one of CLUSPROC, UCATPROC, NVSMPROC, or GDGPROC is called to actually import the object. If the read for a timestamp record should fail, IDCRM01 assumes that an end-of-file has been found on the portable data set and passes control to step 3.

IDCRM01

Procedure: IDCRM01

3 IDCRM01 writes a completion or termination message with LASTCC. Control goes to Executive Controlled Termination. If LASTCC is less than 12, a completion message (with LASTCC) is written; otherwise a termination message (with LASTCC) is written.

Diagram 3.14.1. IMPORTRA FSR – CLUSTER or ALTERNATE INDEX



Extended Description for Diagram 3.14.1

IDCRM01

Procedures: 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. FVTPROC tests the AMDSB for SAM ESDS and if the SAM ESDS feature is not installed an error message is written. A return code of 8 causes control to return to IDCRM01, which bypasses this entry and continues importing the next entry. BPASPROC builds an FPL for security information. If the data or index component was originally defined as unique, IUNIQPRC builds a null volume FVT for each unique component. The OBJECTS list is scanned for USECLASS, VOLUMES, and DEFAULTVOLUMES information about the object to be defined; if found, such information overrides that found on the portable data set. The OBJECTS list is also scanned for FILE information. If found, a pointer to the dname is passed in the component's FVT.

IDCRM01

Procedures: CTLGPROC, DELTPROC, CLUSPROC, CPLPROC, FVTPROC

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, and then CTLGPROC issues a UCATLG to define the object. Should any of these UCATLGs fail, or should the original define fail with a return code other than 8, an error conversion table is built for the UERROR function. UERROR is called to print the error message based on the catalog return code.

If any nonzero allocation condition codes are returned by catalog management, volume allocation status error message(s) are printed. Control is passed to IDCRM01 for the next object. If the define is successful, control returns to CLUSPROC.

If a recovery volume serial number is returned for the define, a UPRINT macro is issued to print it. If the define was for a unique data set on a fixed block device, UPRINT macro(s) are issued to print the actual blocks allocated on each volume.

If the cluster or alternate index has any associated paths, CLUSPROC builds catalog parameter lists for each path from control records on the portable data set. CPLPROC builds the CTGPL, and FVTPROC builds the FVTs and the FPLs. CLUSPROC calls CTLGPROC to issue the UCATLG macro to define the path. Then RECPROC is called to perform step 3.

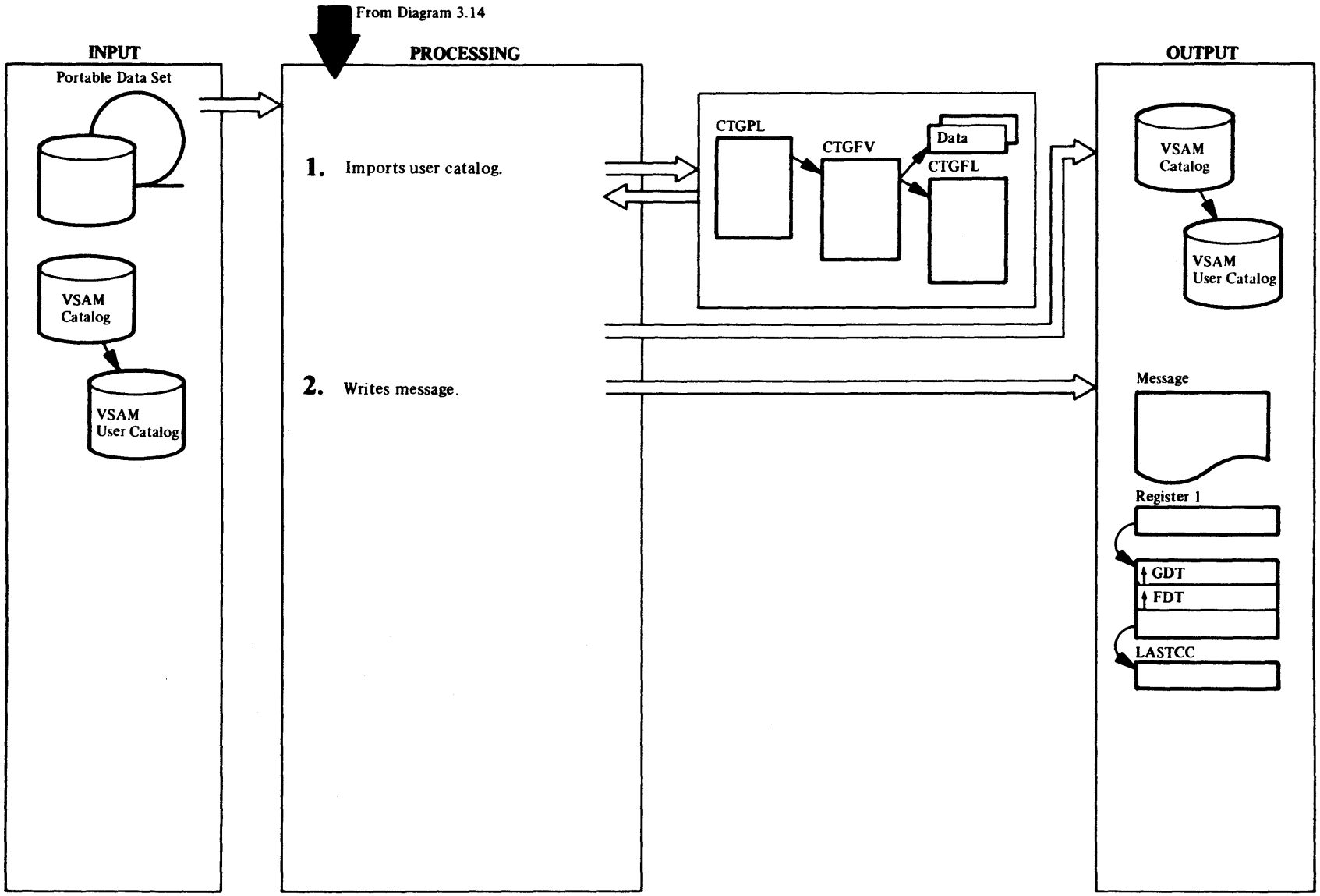
IDCRM01

Procedures: RECPROC, ALTRPROC, OPENPROC

3 If the data set is empty, GETs (UGET macro) are issued to the portable data set until an EOF is reached. If the OUTFILE parameter is present,RECPROC calls ALTRPROC to rename the VSAM object to be loaded to the dummy name returned by the UIOINFO. RECPROC calls OPENPROC to build an OPNAGL with a flag set to indicate RECORDMODE or CIMODE and to issue a UPOEN macro to open the newly-defined VSAM file. If the OUTFILE parameter is omitted, the newly defined file's file-id and catalog name (if present) from the CATALOG parameter are placed in the OPNAGL for UOPEN. RECPROC issues a UCOPY macro to copy data records from the portable data set to the VSAM object. UCLOSE closes the VSAM object. If the OUTFILE parameter is present, ALTRPROC is called to alter the name of the object just loaded back to that under which it was defined.

Processing returns to Diagram 3.14, step 2, for the next item on the portable data set.

Diagram 3.14.2. IMPORTRA FSR – USERCATALOG



Extended Description for Diagram 3.14.2

IDCRM01

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

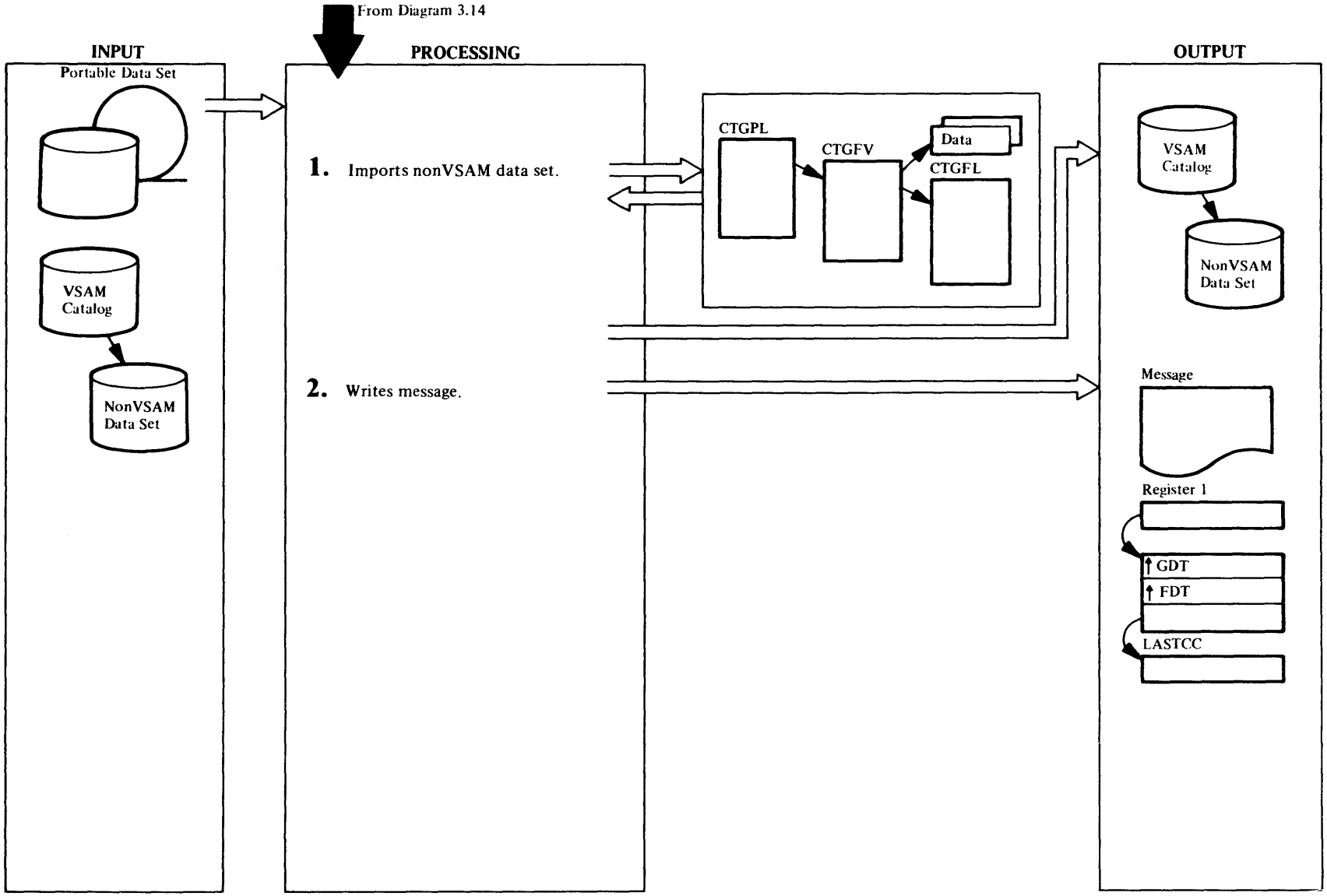
1 CPLPROC builds a CPL to be used to connect the user catalog pointer. UCATPROC then issues a UGET to get the catalog control record and calls GETPROC to obtain the catalog data record. LVLRPROC builds a DEVTYPE FPL and a volume serial list on the basis of information supplied on the portable data set 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.

IDCRM01

Procedure: ALISPROC

2 For each alias item on the portable data set, ALISPROC prints a message indicating that aliases are not processed in VSE. Control then returns to Diagram 3.14, step 2, for the next item on the portable data set.

Diagram 3.14.3. IMPORTRA FSR – NONVSAM



Extended Description for Diagram 3.14.3

IDCRM01

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

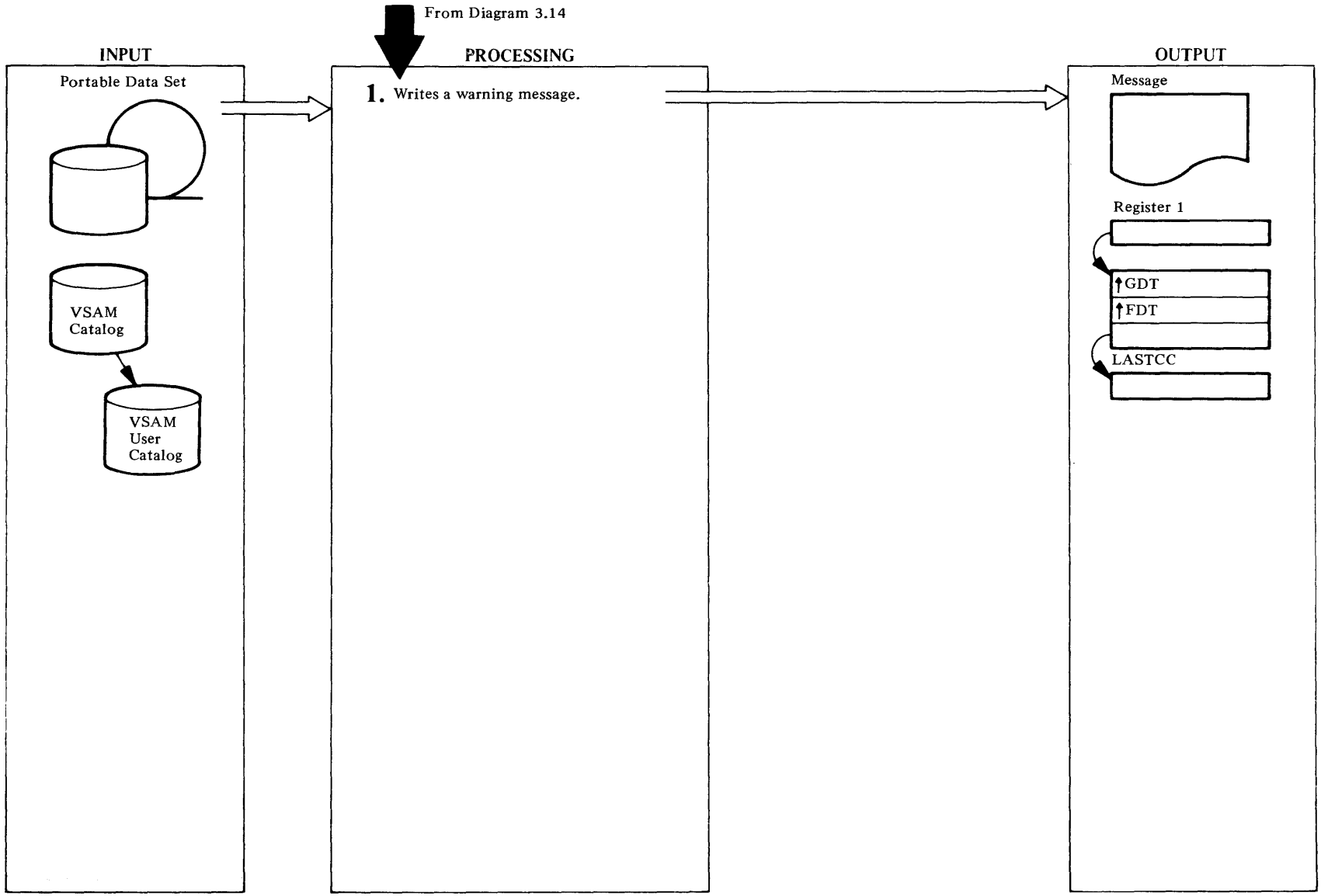
1 CPLPROC builds a CPL to be used to connect the user catalog pointer. NVSMPROC then issues a UGET to get the catalog control record and calls GETPROC to obtain the catalog data record. LVLRPROC builds a DEVTYPE FPL and a volume serial list on the basis of information supplied on the portable data set 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.

IDCRM01

Procedure: ALISPROC

2 For each alias item on the portable data set, ALISPROC prints a message indicating that aliases are not processed in VSE. Control then returns to Diagram 3.14, step 2, for the next item on the portable data set.

Diagram 3.14.4. IMPORTRA – GDG BASE



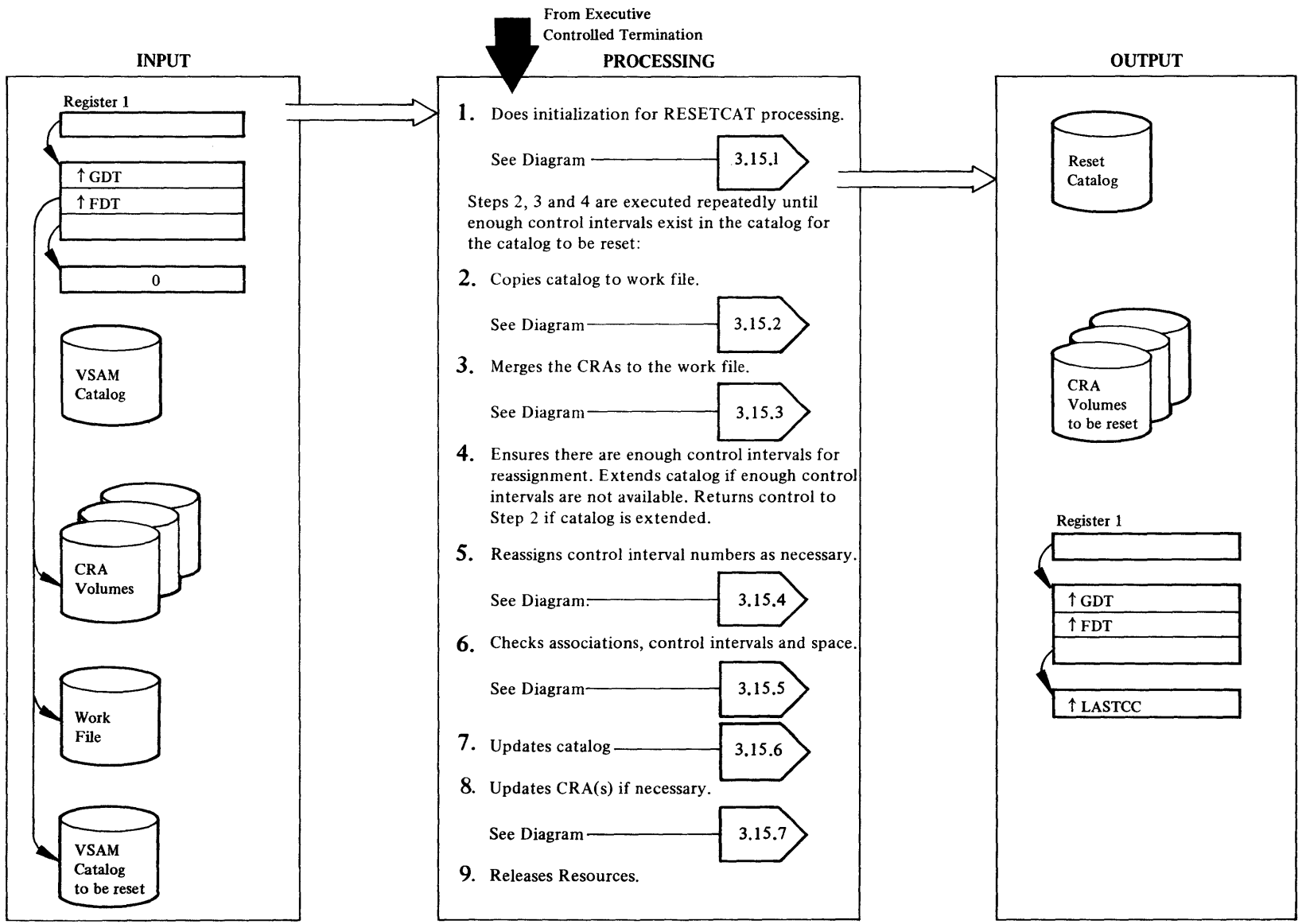
Extended Description for Diagram 3.14.4

IDCRM01

Procedures: GDGPROC

1 GDGPROC issues a warning message indicating that GDG bases cannot be defined in VSE. It then issues successive UGETs until an end-of-file indication is found.

Diagram 3.15 RESETCAT FSR



Extended Description for Diagram 3.15

IDCRS01, IDCRS06

Procedure: INIT, DSOPEN, CATINIT, WFDEF

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 UENQ macro. The catalog in which the work file will be defined is also opened and then the work file is defined and opened. An entry in the RESVOL table is created for each CRA volume identified by the CRAFILES parameter. 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 iterative loop. These three steps are executed repeatedly until the catalog to be reset has enough control intervals.

IDCRS01, IDCRS05, IDCRS06

Procedure: 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 interval 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 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.

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.

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 are 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).

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.

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.

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. The free record chain is rebuilt.

IDCRS01, IDCRS05, IDCRS06

Procedures: UPDCRA, SCNRLST, DSOPEN, DSCLOSE, CKERR

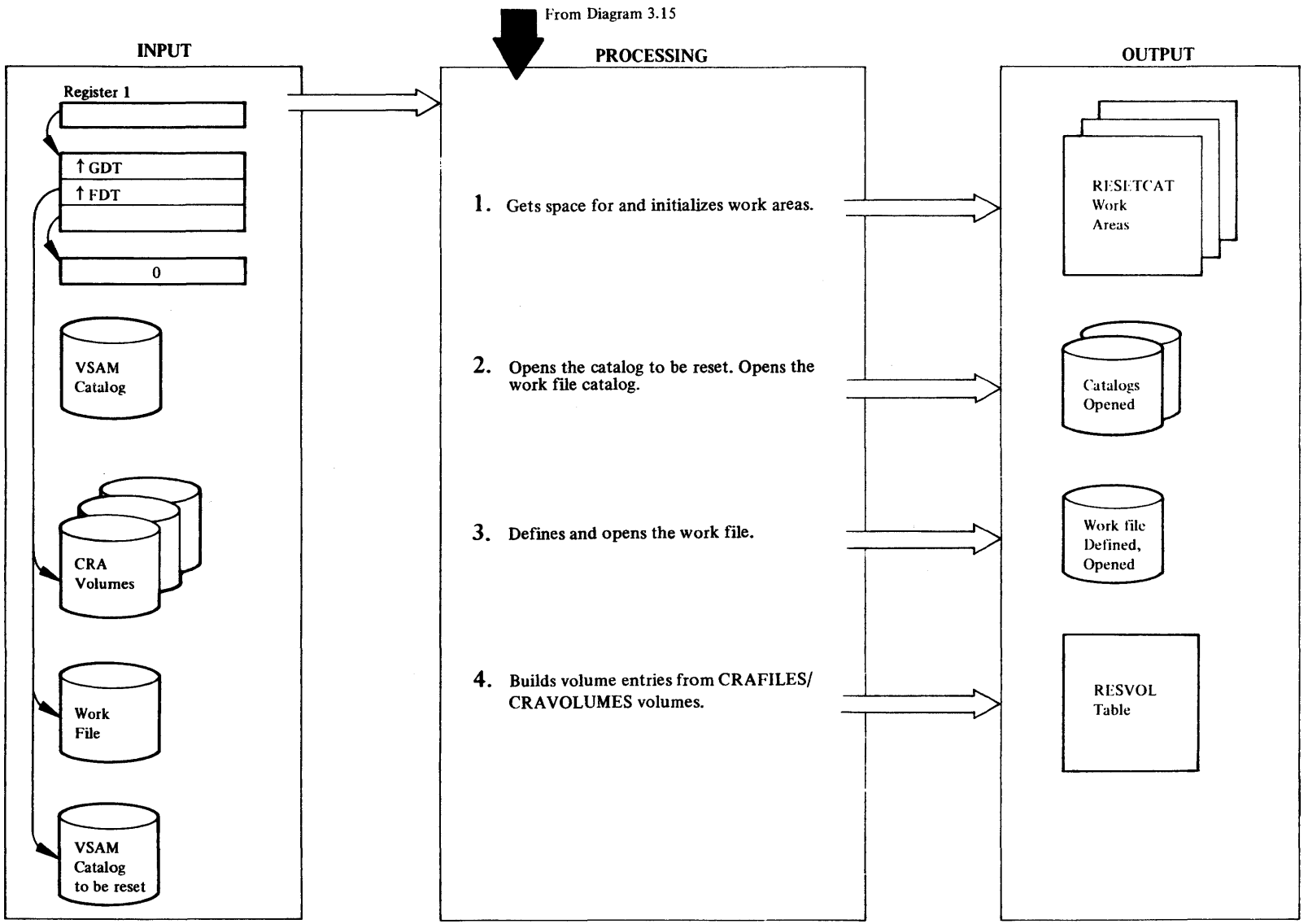
8 UPDCRA updates CRAs from the work file. Each entry in RESVOL (a table containing an entry for each volume whose CRA is required in the reset operation) is obtained. If there are any updates to be made in the CRA, it is opened, updated, and closed. If any free records are placed in the CRA, the CCR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

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 3.15.1 RESETCAT FSR – Initialization



Extended Description for Diagram 3.15.1

IDCRS01

Procedures: INIT

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

- CRA user buffer
- Record Management control blocks (GRAB, BUFFER)
- IJJHCPL CVH parameter list
- 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.

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. CKERR is called if these checks fail.

Exclusive use of the catalog is ensured by issuing the UENQ macro to obtain exclusive use of the ENQ name of the catalog (R*volser* RSC00). If it is determined that the catalog is in use by someone else, CKERR is called.

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.

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.

DSOPEN is called to open the catalog into which the work file is to be defined.

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.

IDCRS01, IDCRS05, IDCRS06

Procedures: INIT, CKERR, CATINIT

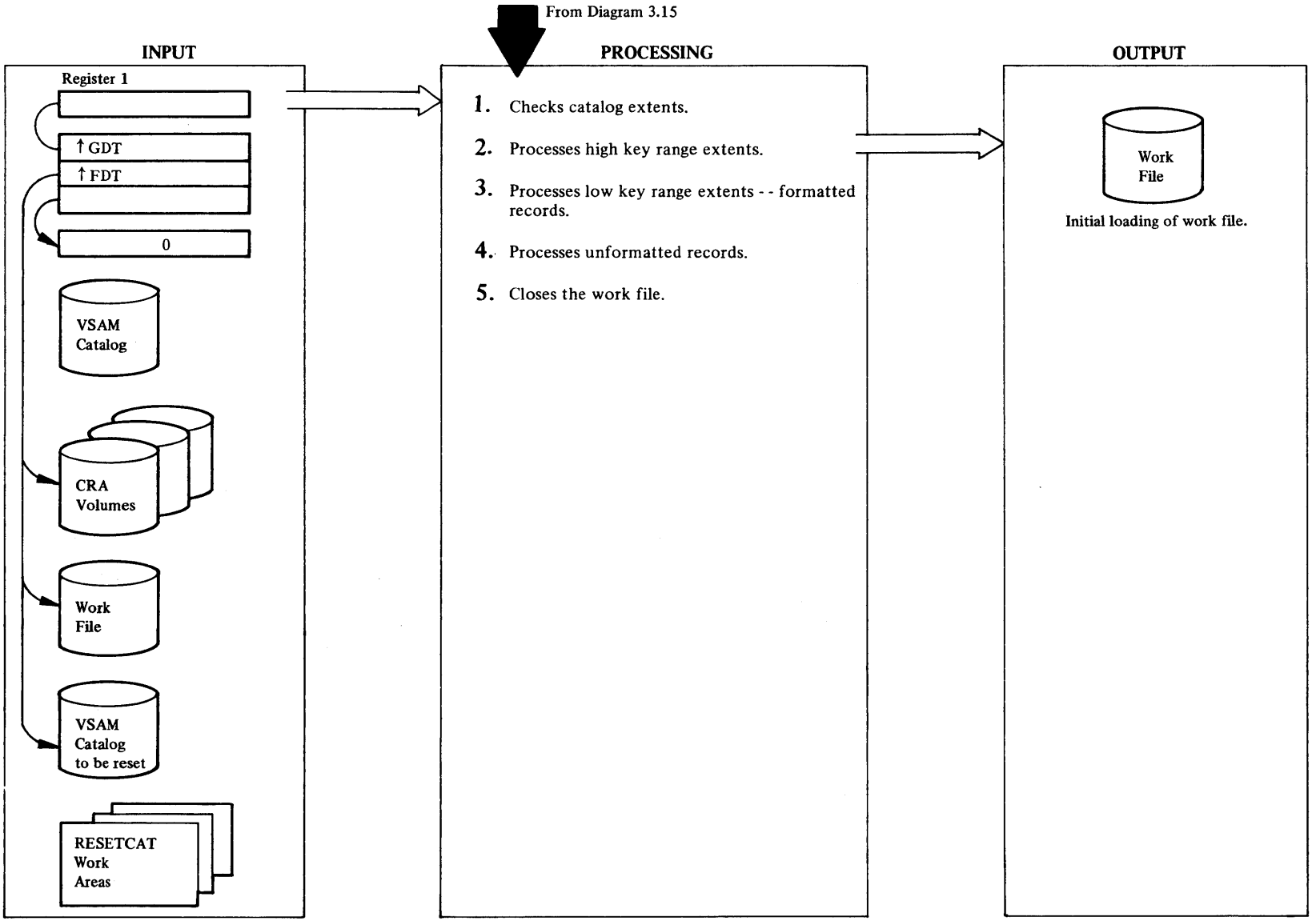
4 The RESVOL table is constructed consisting of an entry for each CRA volume supplied by the invoker of RESETCAT with the CRAFILES or CRAVOLUMES parameter. Each entry consists of fields for volume serial number, device type, system logical unit (CRAVOLUMES only), and the file name of the DLBL statement (CRAFILES only). 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.

If CRAFILES is specified, the volume serial number of the CRA is obtained via the UIOINFO macro. If CRAVOLUMES is specified, the volume serial number of the CRA is contained in the subparameters.The volume serial number of the CRA is inserted in 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 with the ALL subparameter, CKERR is called to flag an error condition.

CATINIT is 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 3.15.2. RESETCAT FSR – Copy Catalog to Work File



Extended Description for Diagram 3.15.2

IDCRS01

Procedures: 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 than 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:

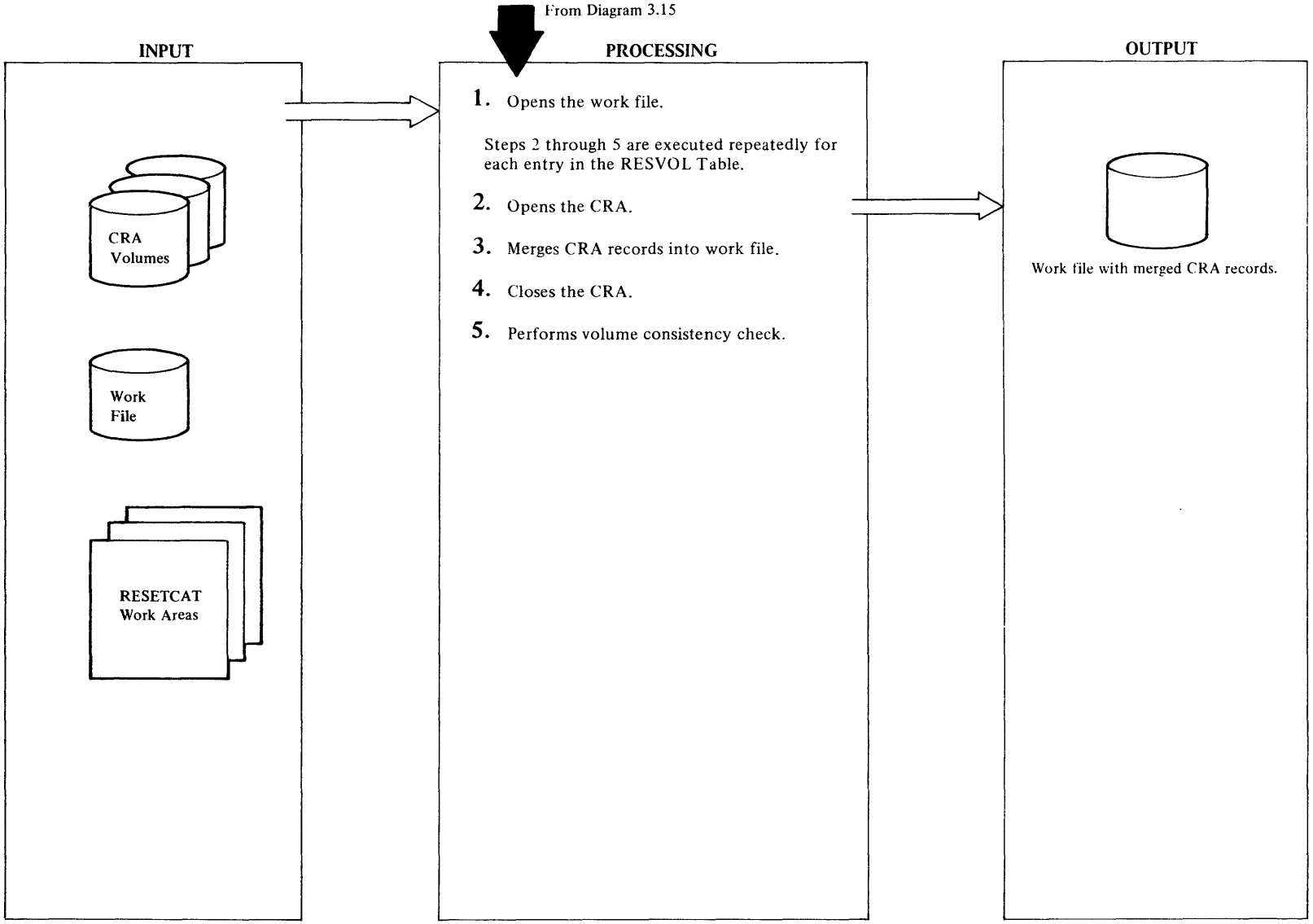
  - If the RRN is greater than the high formatted RRN, then ADDRCD is called to add the record to the work file.
  - If the RRN is not greater, UPDRCD is called to update the record in the work file.

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 3.15.3. RESETCAT FSR – Merge CRA(s) to the Work File



Extended Description of Diagram 3.15.3

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 inerative 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.

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.

IDCRS01, IDCRS06

Procedures: MERGECRA, DSCLOSE

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

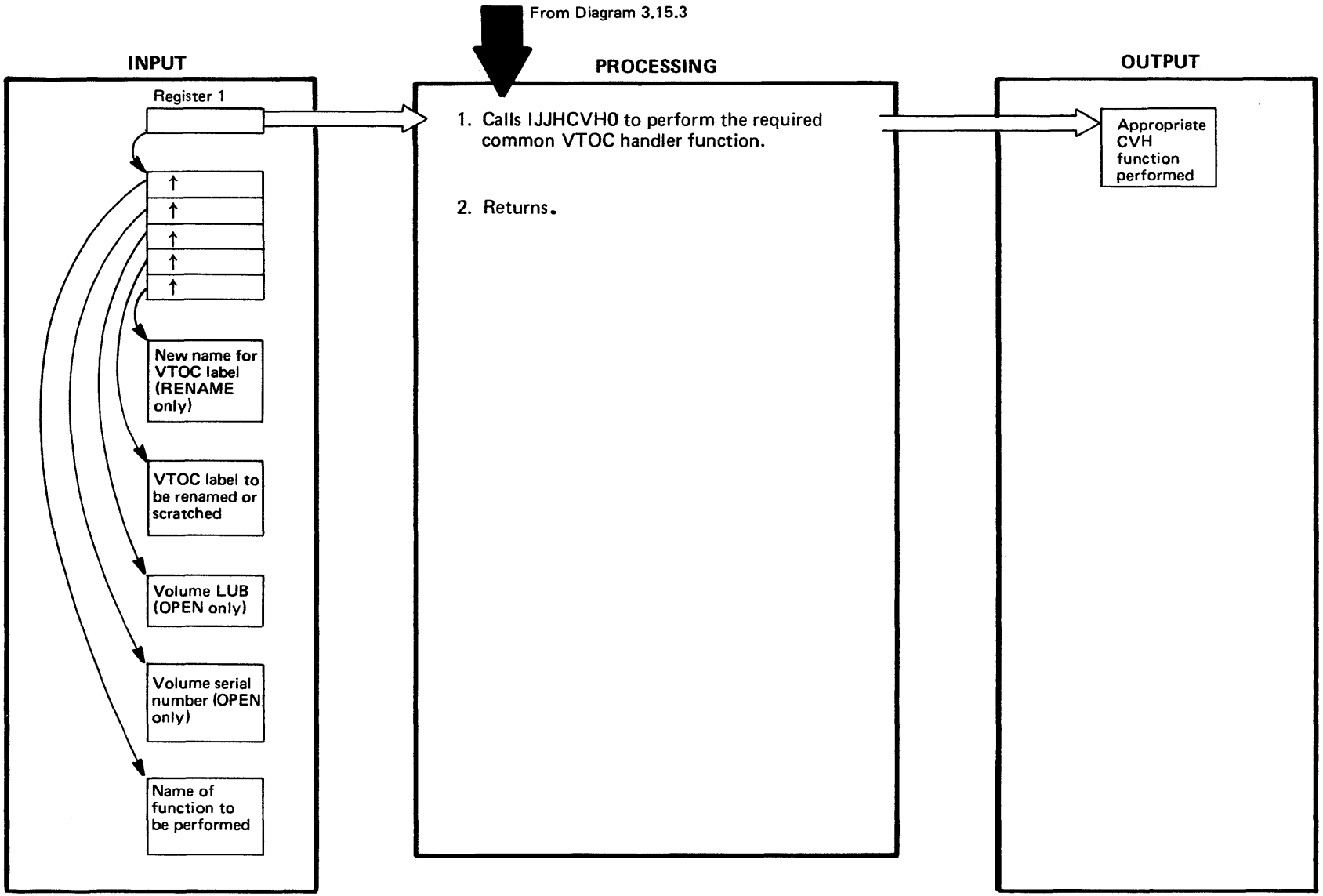
IDCRS01, IDCRS03

Procedures: MERGECRA, VOLCHK, HVTOC

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 label in the VTOC) and each space header in the volume record for that volume. This is done by calling the HVTOC procedure to read each label in the VTOC (through an interface with the common VTOC handler) and then comparing the VSAM-owned label with the corresponding volume record space header. If a format 1 label does not have a corresponding space header, the label is scratched by calling HVTOC. If a space header refers to a non-existent format 1 label, the space header is deleted. If the extents in a space header are not identical to the extents in the corresponding format 1 label, the extents in the space header are corrected.

Diagram 3.15.3.1 RESETCAT FSR - Common VTOC Handler Functions



Extended Description for Diagram 3.15.3.1

IDCRS07

Procedure: HVTOC

1 RESETCAT calls the HVTOC procedure to perform all common VTOC handler (CVH) functions. After examining the name of the function to be performed, HVTOC issues the appropriate CVH macro (OVTOC, CVTOC or various forms of PVTOC). This macro builds the CVH parameter list (IJJHCPL) and calls the topmost CVH module (IJJHCVH0).

Valid names of HVTOC functions to be performed are as follows:

CLOSE - close the VTOC

OPENb - open the VTOC

RADDR - read label from specified address

RENME - rename the label

RFMT4 - read the format-4 VTOC label

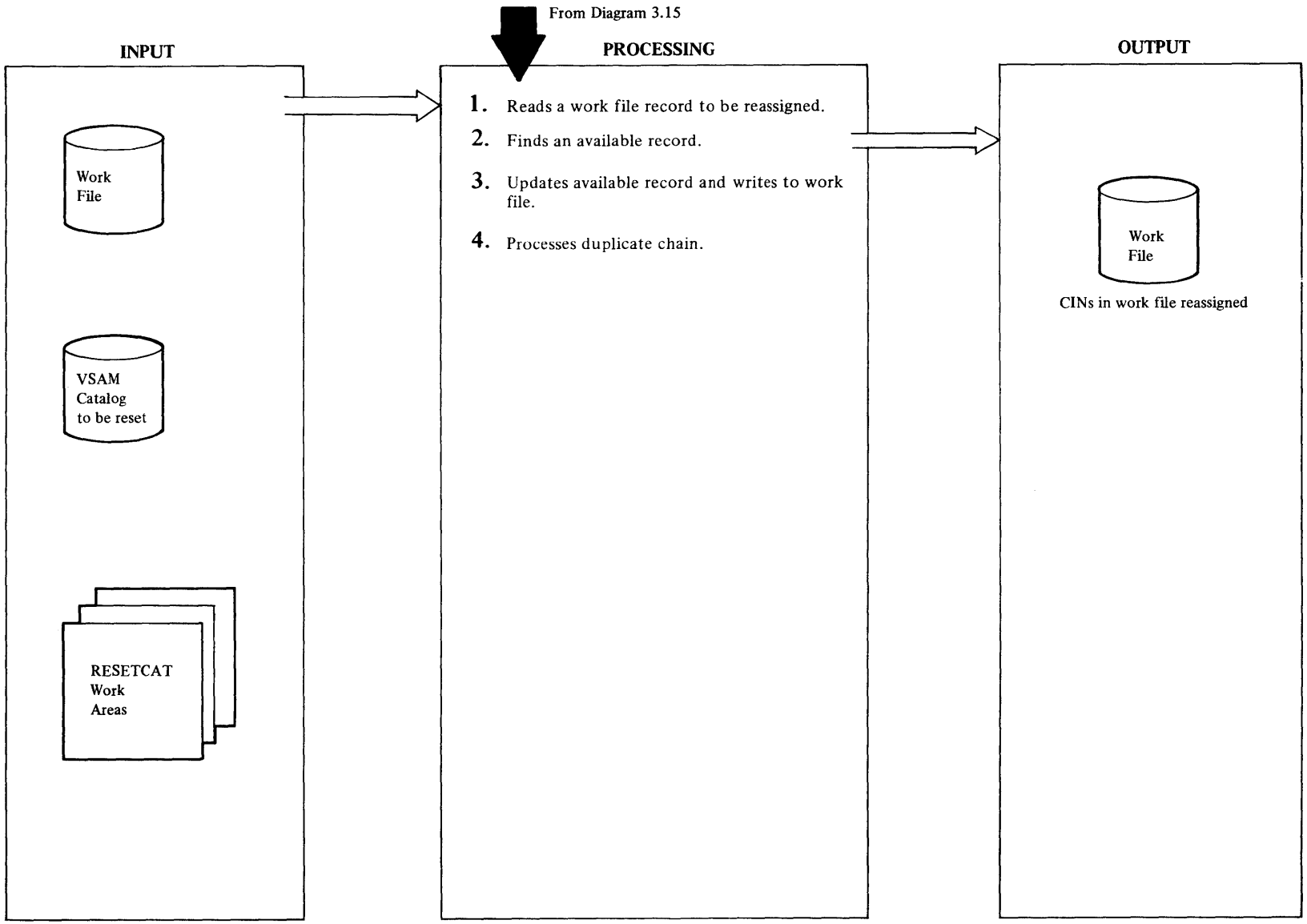
RNEXT - read the next VTOC label

SCRTH - scratch a label

WADDR - write label to specified address

For more information on the CVH parameter list and the VSE CVH routines that perform the above functions, see *DOS/VSE Fixed Block Architecture Logical IOCS.*

Diagram 3.15.4. RESETCAT FSR – Reassign CI Numbers



Extended Description of Diagram 3.15.4

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 returned 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 reassigning chain. The reassign chain pointer is saved.

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.

IDCRS01, 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.

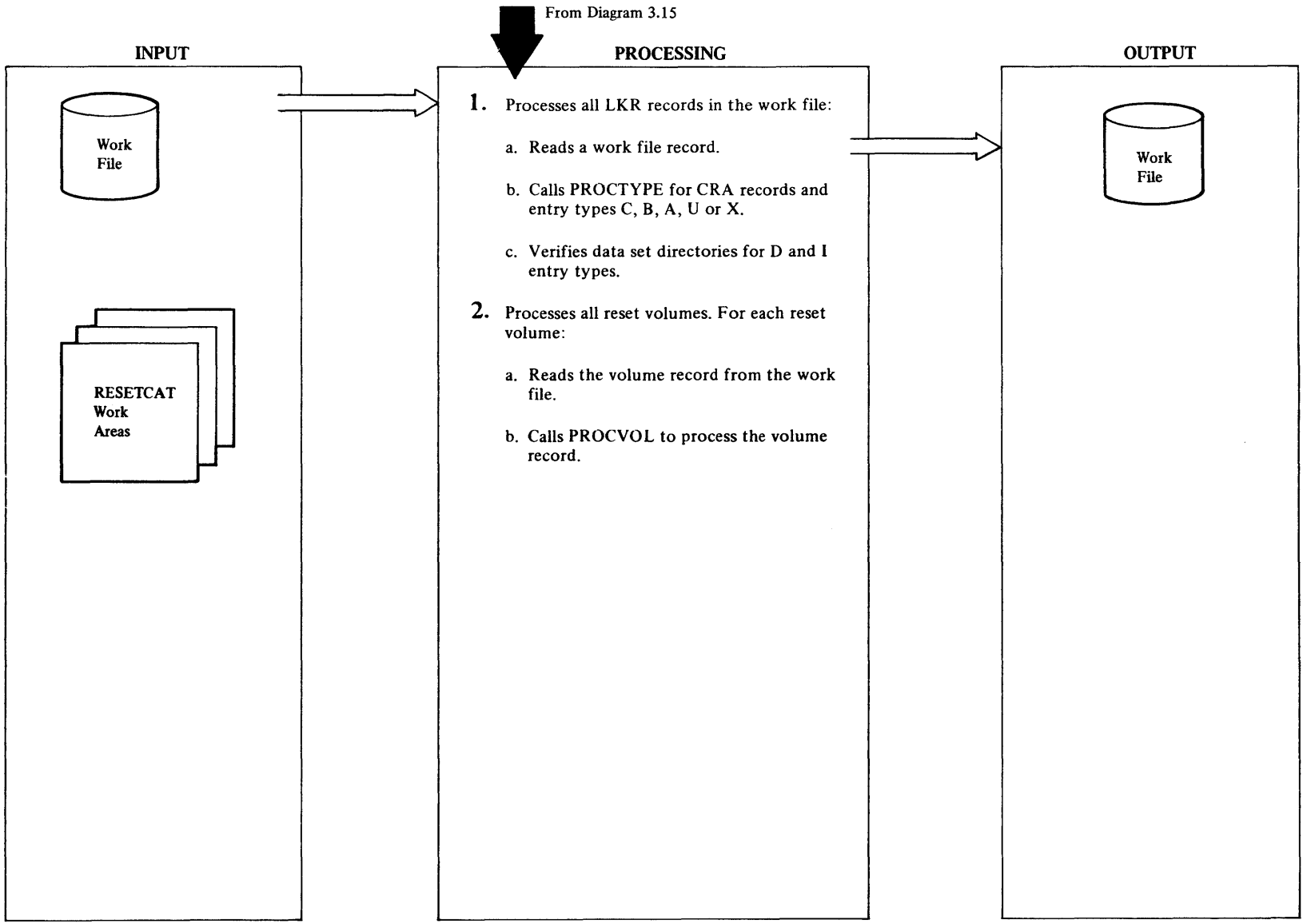
RECMGMT (with the UPDRCD option) is called to write the update available record to the work file.

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 3.15.5 RESETCAT FSR — Check Associations



Extended Description for Diagram 3.15.5

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 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.

IDCRS02, IDCRS06

Procedures: ASSOC, RECMGMT

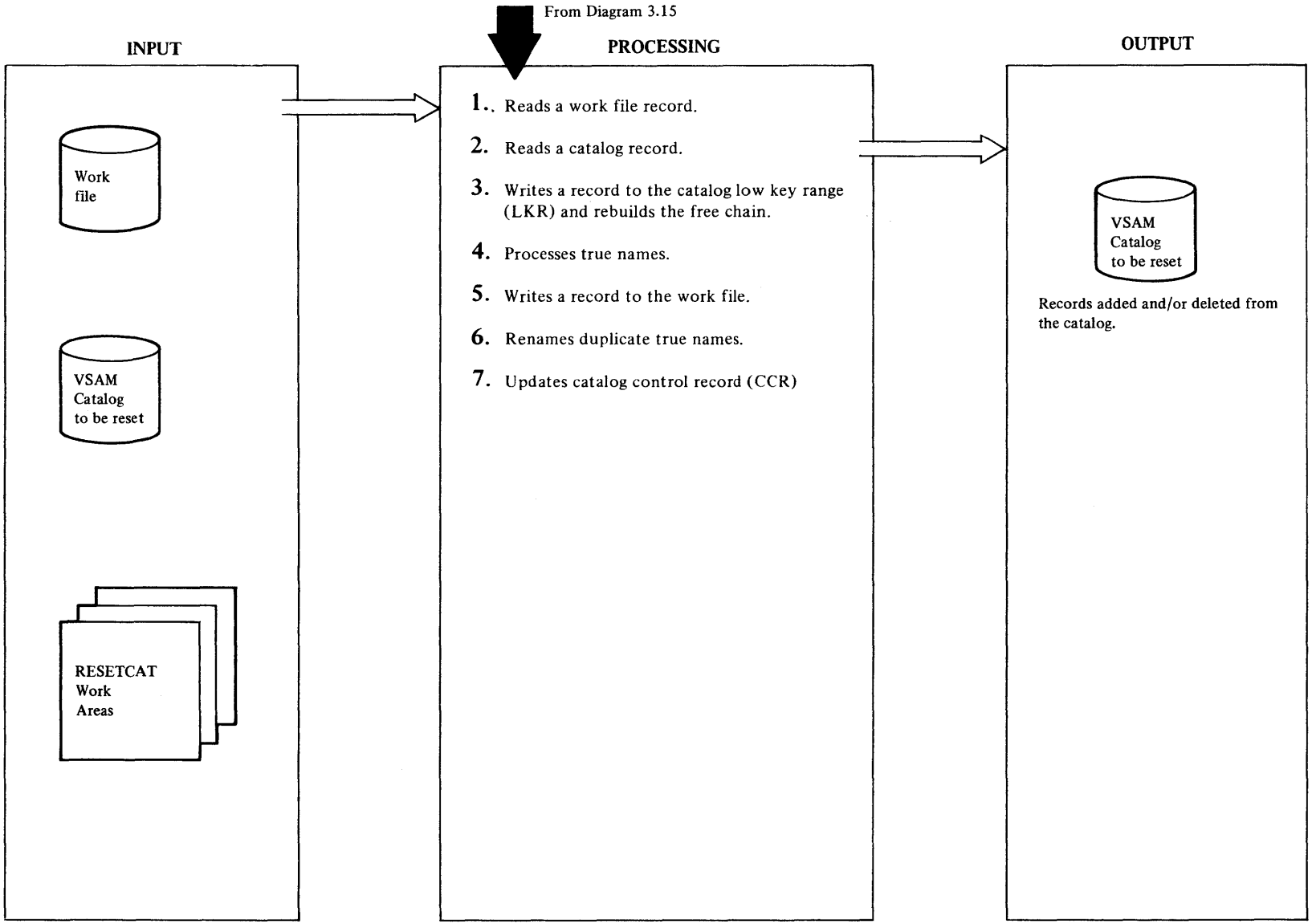
2 a. For each reset volume, the volume record is read from the work file via RECMGMT (GETRCD).

Procedures: ASSOC, RECMGMT, PROCTYPE, VERDSDIR

b. The PROCVOL procedure 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 3.15.6 RESETCAT FSR – Update the Catalog



Extended Description for Diagram 3.15.6

IDCRS05, IDCRS06, IDCRS07

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.

IDCRS06, IDCRS07

Procedures: UPDCAT, RECMGMT, ENTNMCK

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.

IDCRS06, IDCRS07

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.

IDCRS05, IDCRS06, IDCRS07

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.

IDCRS05, IDCRS06, IDCRS07

Procedures: UPDAT, SCNLST, 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. The "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.

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.

IDCRS06, IDCRS07

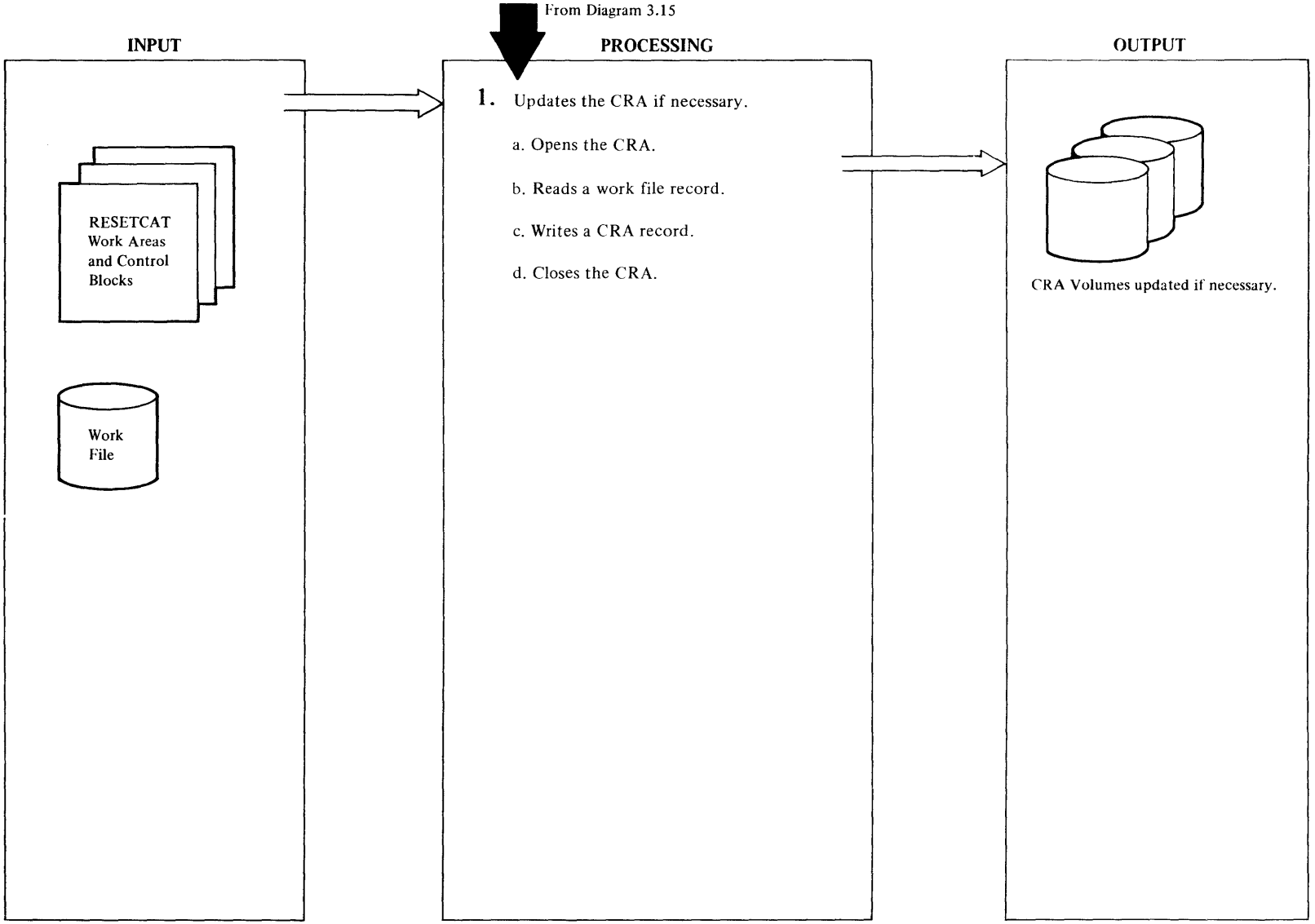
Procedures: UPDCAT, RECMGMT, UPDCCR

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 3.15.7 RESETCAT FSR – Updates the CRA



Extended Description for Diagram 3.15.7

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 open is successful, the "CRA open" flag is set, if not, the "termination" flag is set.

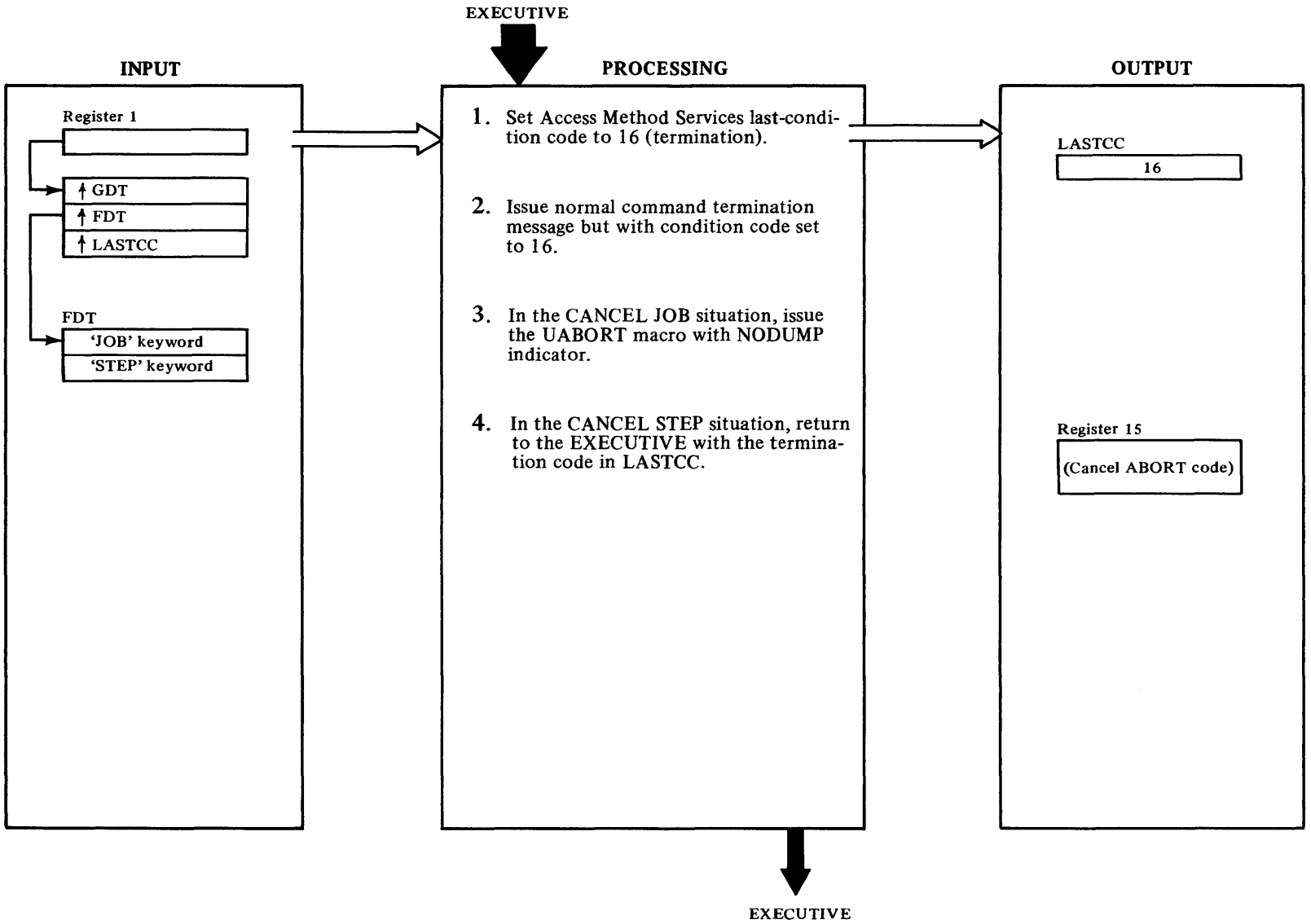
b. Each record in the CRA update chain is read from the work file RECMGMT (GETRCD). The control interval number of the next record in the chain is saved. If the record just read happens to be a free record, the CRA CCR record needs to be updated. If the CCR has not been read already, RECMGMT (GETRCD)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

Diagram 3.16. CANCEL FSR



Extended Description for Diagram 3.16

IDCCL01

1 Set the last-condition-code to 16.

IDCCL01

Procedure: IDCCL01

2 The message IDCC001I is issued. The condition code is set to 16.

IDCCL01

Procedure: IDCCL01

3 The UABORT code is 76. The value is negative to signal UABORT that no PDUMP is needed.

If Access Method Services was called as a subroutine, UABORT returns control to the caller of Access Method Services witha value of 16 in register 15.

If Access Method Services was not called as a subroutine, SVC06 is issued and the job stream is flushed to the next "/&" or "//JOB" card.

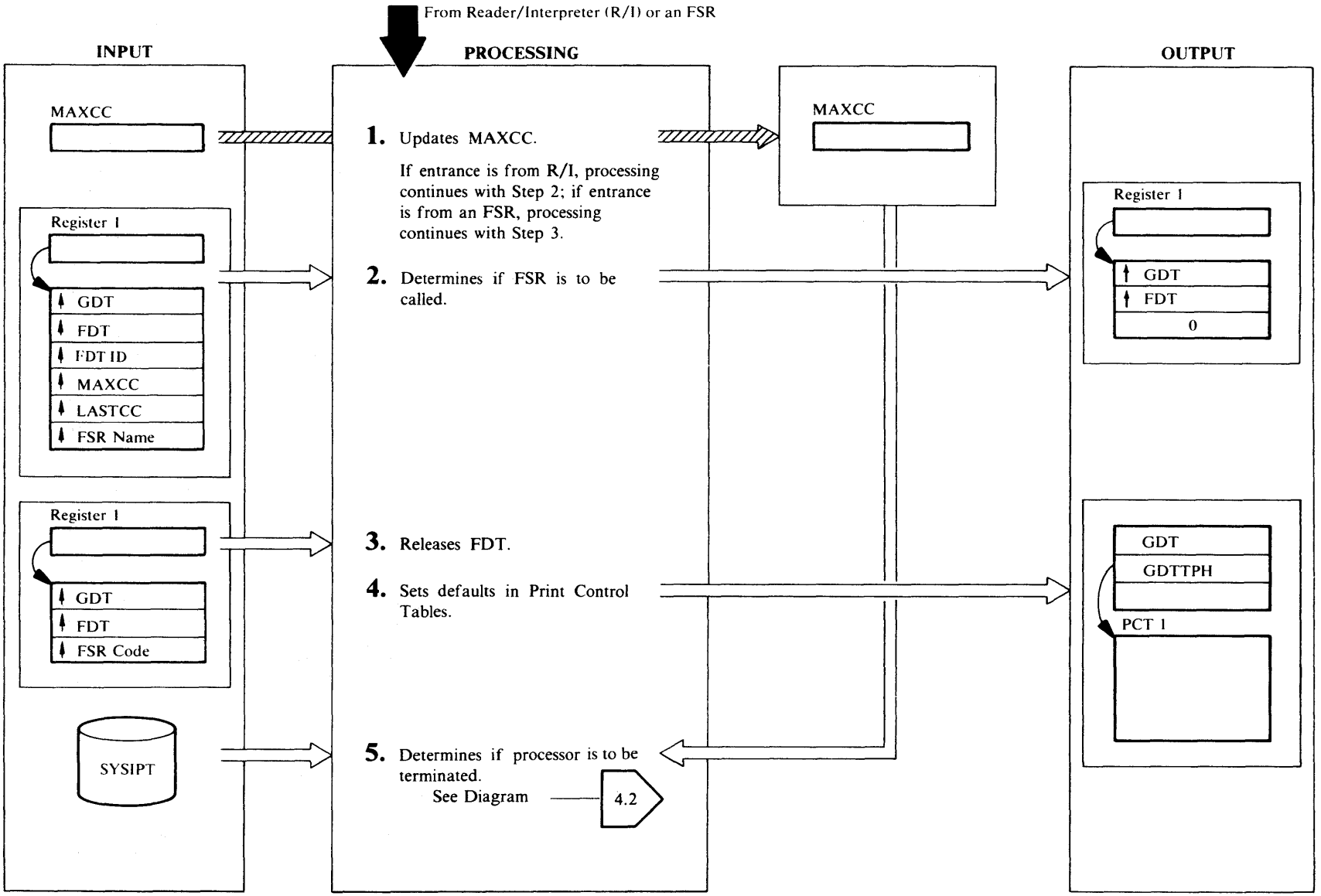
IDCCL01 Procedure: IDCCL01

4 If Access Method Services was called as a subroutine, UABORT returns control to the caller of Access Method Services with a value of 16 in register 15.

If Access Method Services was not called as a subroutine, the job stream is flushed to EOF by the Access Method Services Executive.

Termination Visual Table of Contents

Diagram 4.1. Executive Controlled Termination



Extended Description for Diagram 4.1

IDCEX01

Procedure: MAIN

1 IDCEX01 compares the LASTCC code returned by the FSR or the R/I with MAXCC and puts the greater number in MAXCC. If control is from the R/I, MAXCC has already been properly set by IDCRI01. If entrance is from the R/I, processing continues with step 2; if entrance is from an FSR, processing continues with step 3.

IDCEX01

Procedure: MAIN

2 If MAXCC is less than 16 or an end-of-file has not been reached on SYSIPT, IDCEX01 gives control to an FSR. The R/I passes the FSR name to IDCEX01. If MAXCC is greater than or equal to 16 or an end-of-file has been reached on SYSIPT, processing continues with step 5.

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.

IDCEX01

Procedure: CALLFSR

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

IDCEX01

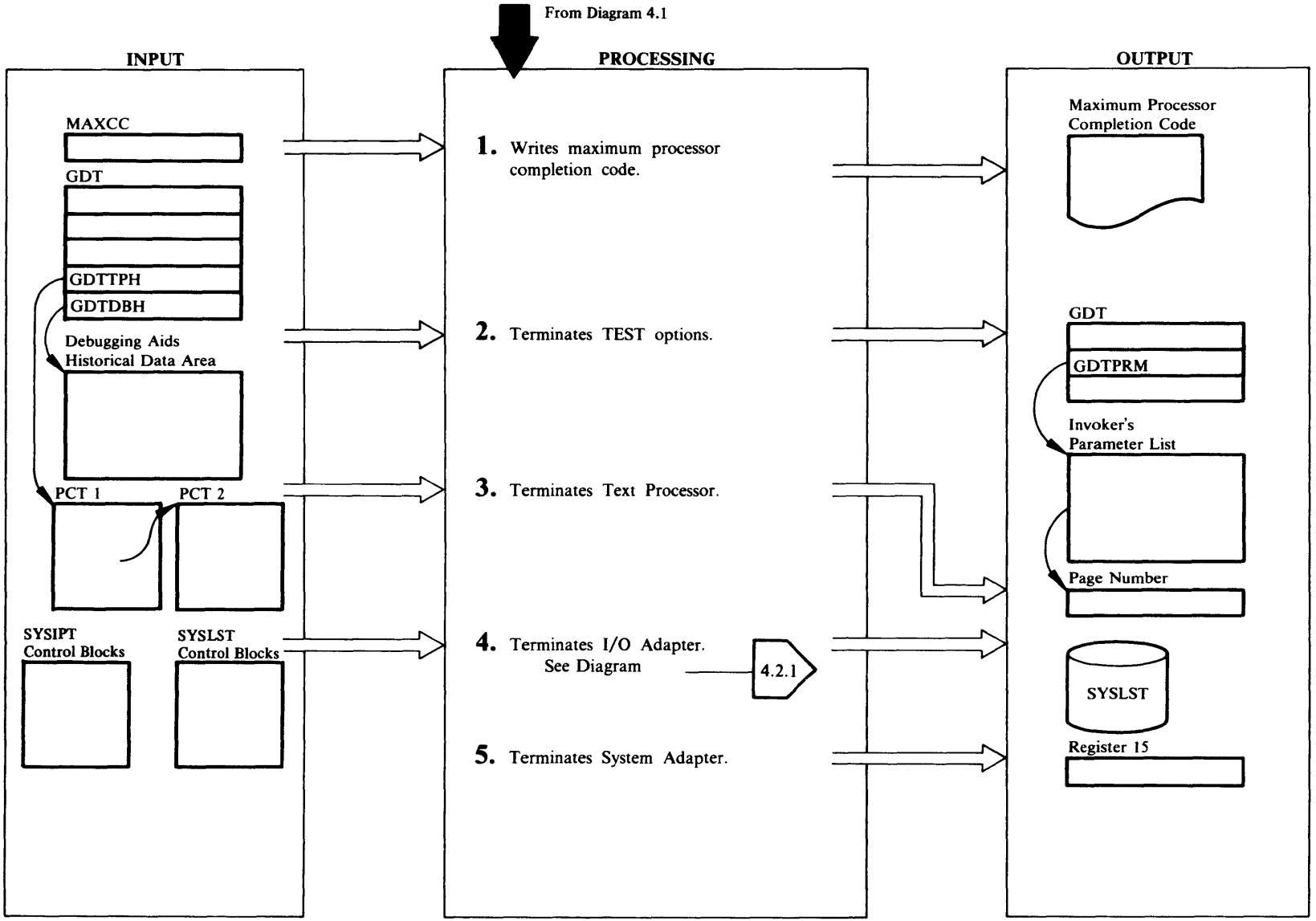
Procedure: MAIN

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

- The R/I has detected end-of-file on SYSIPT. In this case, the R/I puts a nonzero 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 R/I, Diagram 2.0, to obtain the next command.

Diagram 4.2. Processor Termination



Extended Description for Diagram 4.2

IDCEX03

Procedure: IDCEX03

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

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.

IDCEX03

Procedures: IDCEX03, SCANPARM

3 IDCEX03 terminates the Text 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.

IDCEX03

Procedure: IDCEX03

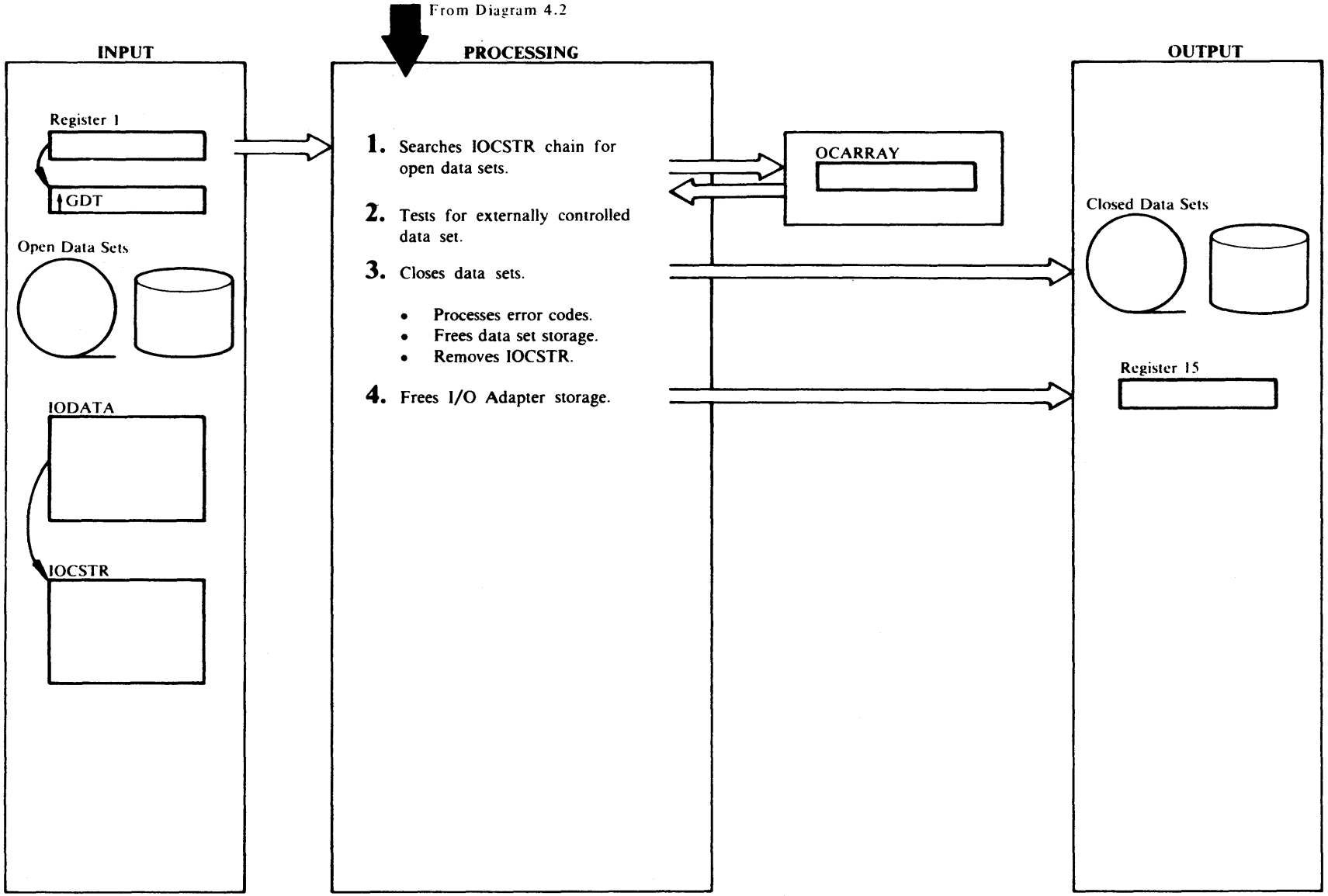
4 IDCEX03 terminates the I/O Adapter by issuing a UIOTERM macro. Diagram 4.2.1 shows I/O Adapter termination in detail.

IDCSA01

Procedure: IDCSA01

5 IDCSA01 terminates the System Adapter by freeing the storage for IDCSA02, IDCSA03, IDCTP01, and IDCIO01. The Storage Table, AUTOTBL, contains the storage addresses for IDCSA02, IDCTP01, and IDCIO01. 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. Control returns to the invoker.

Diagram 4.2.1. I/O Adapter Termination – UIOTERM Macro



Extended Description for Diagram 4.2.1

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 SYSIPT and SYSLST to be closed.

IDCIO02

Procedure: CLOSERTN

2 CLOSERTN examines the IOCSTR chain for the address of IOCSTRs to close. For a nonVSAM 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 externally controlled. If so, CLOSERTN passes arguments to the external routine. This check is made for up to the first four IOCSTRs in the IOCSTR chain. Normally, only the SYSIPT and SYSLST IOCSTRs are in the chain at termination.

IDCIO02

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 nonVSAM data set, the operating system close routine gives control to a CLOSE exit routine which sets a flag in IOCSTRN that will cause the I/O Adapter to print an error message. The message is written after control returns from the CLOSE SVC. 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 SYSLST is the first data set closed, BLDOCMSG issues a UABORT macro. No test is made for nonVSAM 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 nonzero, ENVFREE issues a FREEMAIN macro to release the control block. For open nonVSAM data sets, ENVFREE issues a FREEVIS to free any buffers obtained by the operating system open routines.
- CLOSERTN saves the address of the closed data set's IOCSTR 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.

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

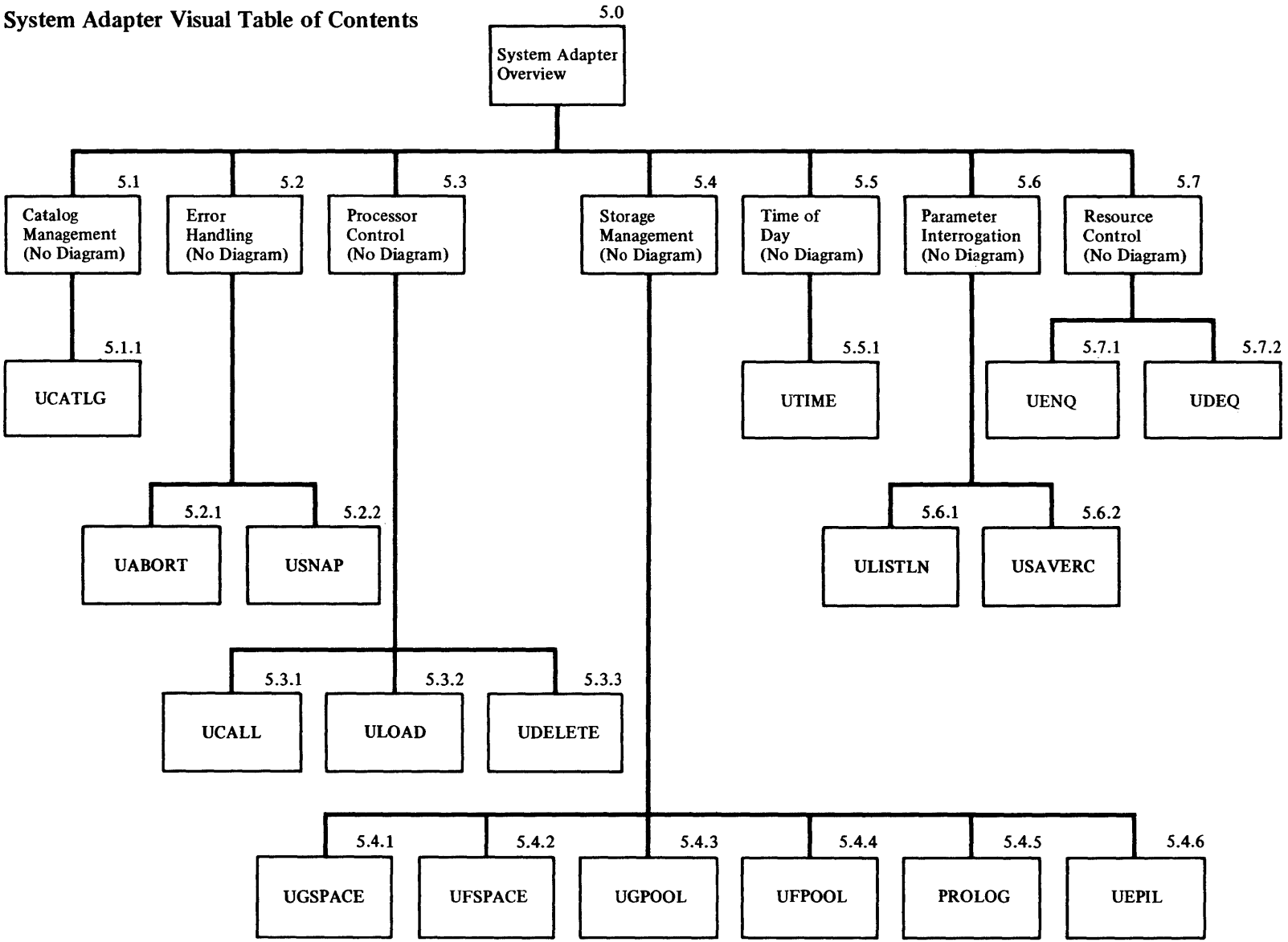
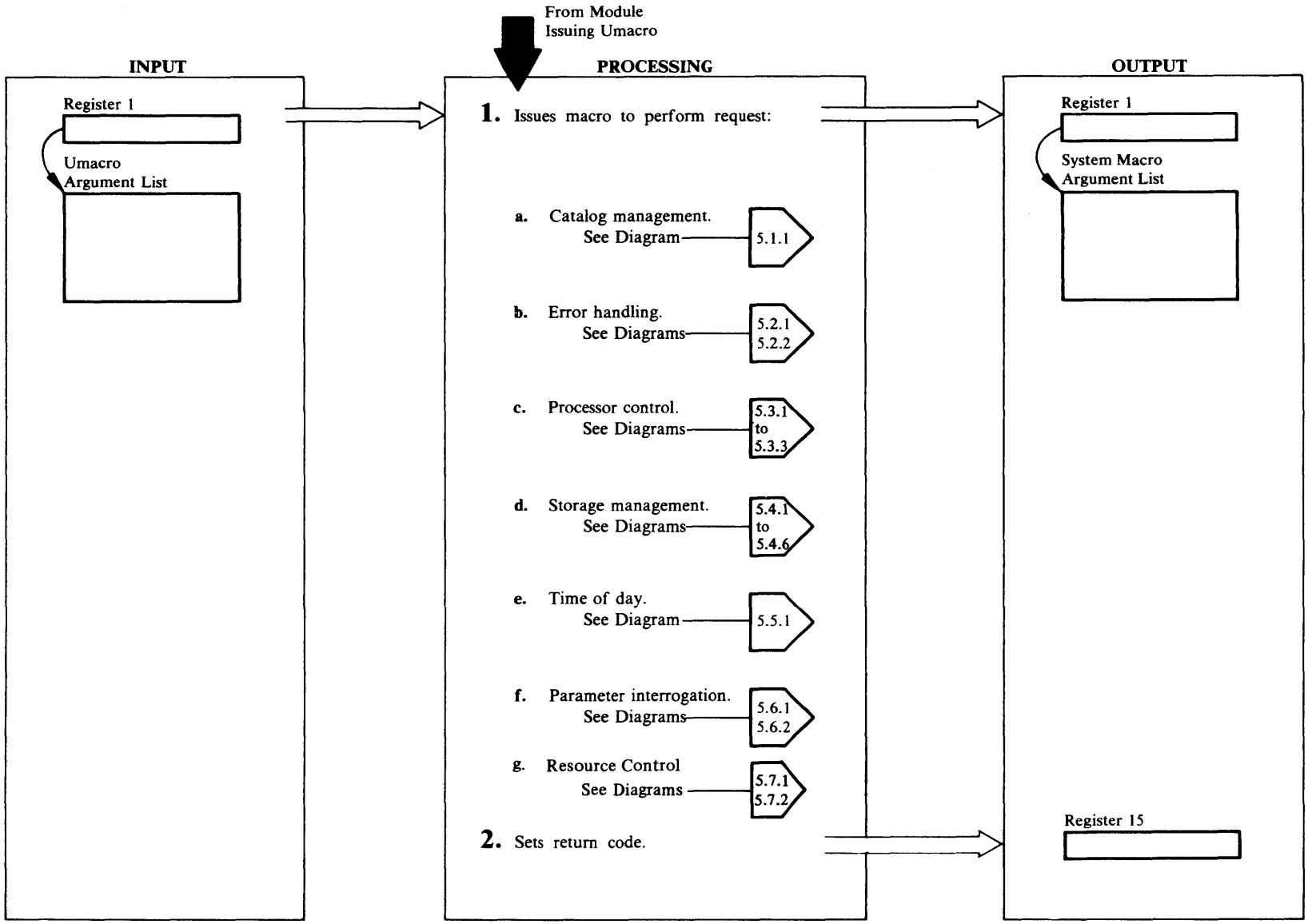


Diagram 5.0. System Adapter Overview



Extended Description for Diagram 5.0

IDCSA01, IDCSA02, IDCSA03, IDCSA05, IDCSA08

Procedures: IDCSA01, IDCSA02, IDCSA03, IDCSA05, IDCSA08

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 GETVIS, the processor calls the System Adapter with a Umacro. 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. 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, deleted, or retrieved from the VSAM catalog, a UCATLG macro is issued. Although the VSAM CATLG macro has the same parameters in OS/VS and VSE, the general code is different. The VSAM CATLG macro must be in a program that is assembled under the right operating system. Diagram 5.1.1 shows the UCATLG macro in detail.

   b. Error handling is accomplished with UABORT and USNAP. For errors, when processing cannot continue, a UABORT is issued to print an error message and a dump and return control to the operating system. If the error condition is due to no space available, only an error message is printed; no dump is printed. For debugging information, a USNAP is issued to print the partition 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.

   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. UDELETE does not take any action in DOS. Diagrams 5.3.1 through 5.3.3 show the UCALL, ULOAD, and UDELETE macros in detail.

   d. Storage management is performed with three types of 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 re-entrant enviroment, 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 that issued the UEPIL. The PL/S-generated code to free storage and to return control is never executed.

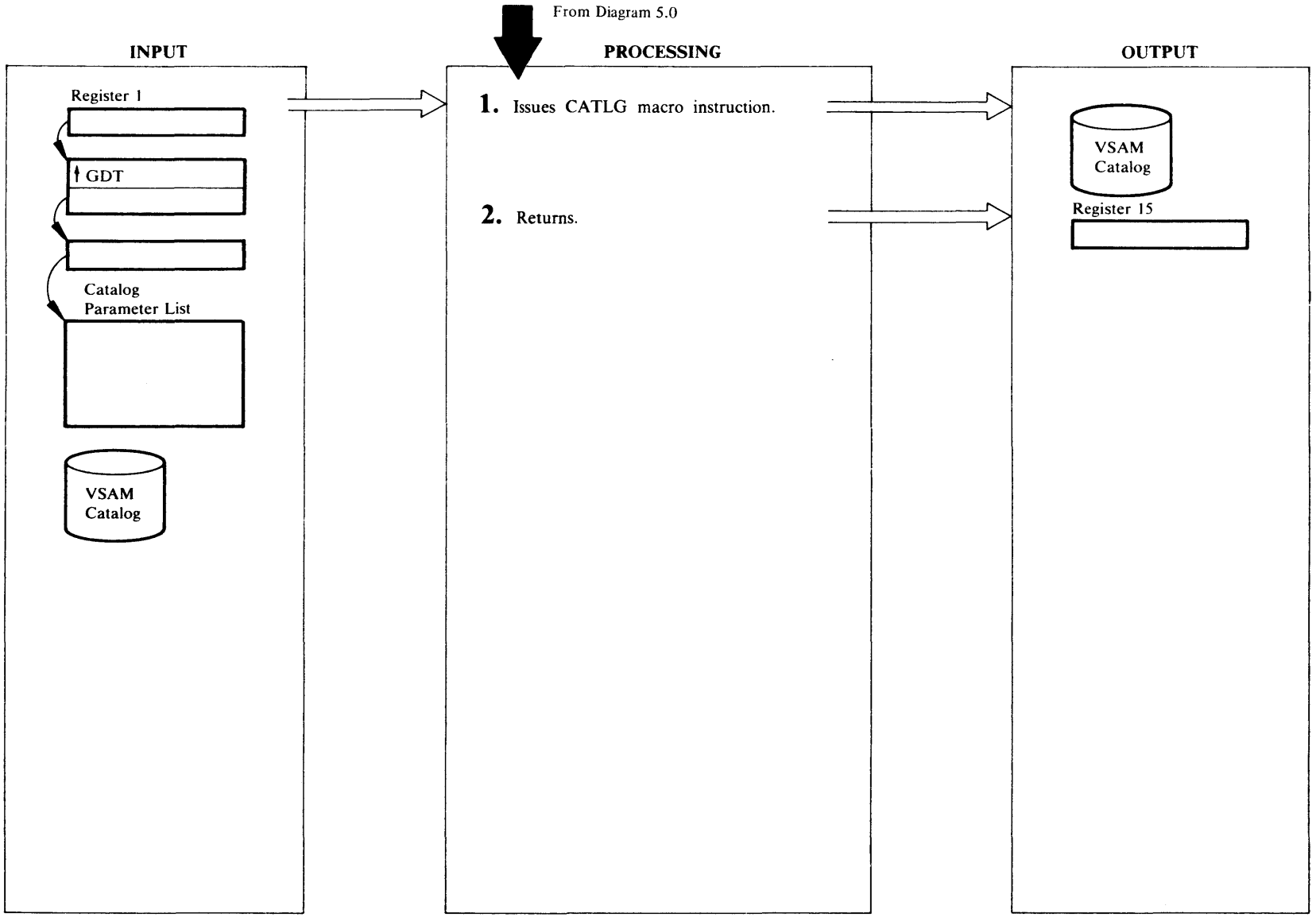
   e. The time of day is obtained with a UTIME macro, shown in Diagram 5.5.1. Several data formats for the time and date are allowed.

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

   g. Control of a resource is achieved with a UENQ macro. The resource may be released with a UDEQ macro. See Diagrams 5.7.1 and 5.7.2.

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, and UEPIL.

Diagram 5.1.1. UCATLG Macro



Extended Description for Diagram 5.1.1

IDCSA02

Procedure: IDCSA02

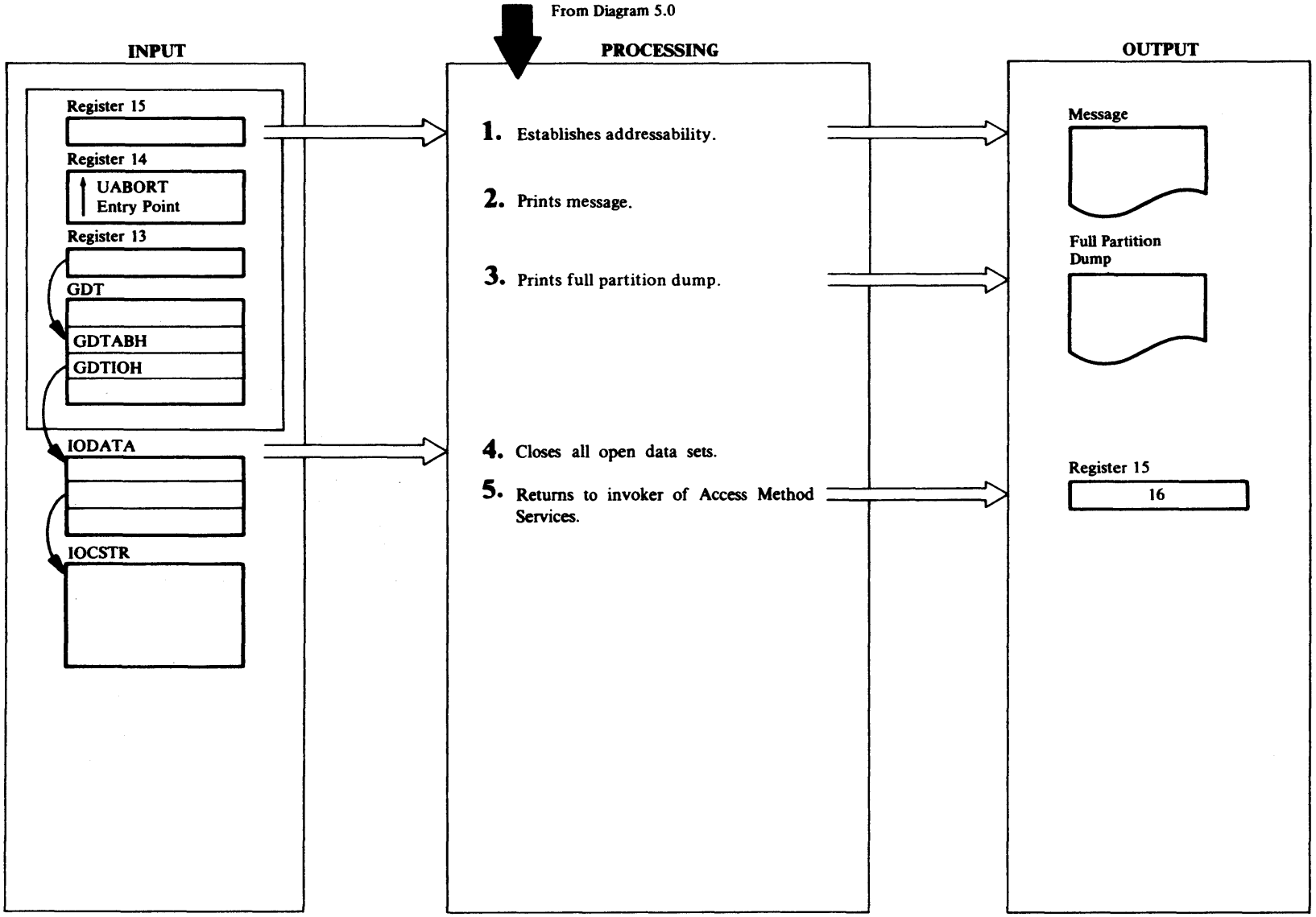
1 IDCSA02 passes the catalog parameter list to VSAM with a CATLG macro.

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 5.2.1. UABORT Macro



Extended Description for Diagram 5.2.1

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.

IDCSA01

Procedure: IDCSA01

2 UABORT issues an EXCP to write a message to the programmer.

IDCSA01

Procedure: IDCSA01

3 The UABORT routine issues the PDUMP macro and takes a full partition dump unless the UABORT code indicates a no-space-available condition, in which case no dump is issued. The partition beginning and ending addresses for the PDUMP are obtained by issuing the EXTRACT macro. The UABORT code is in register 15 in the dump.

If register 15 is negative, it is complemented and no PDUMP is done. The CANCEL Access Method Services Command requires this interface.

IDCSA01

Procedure: IDCSA01

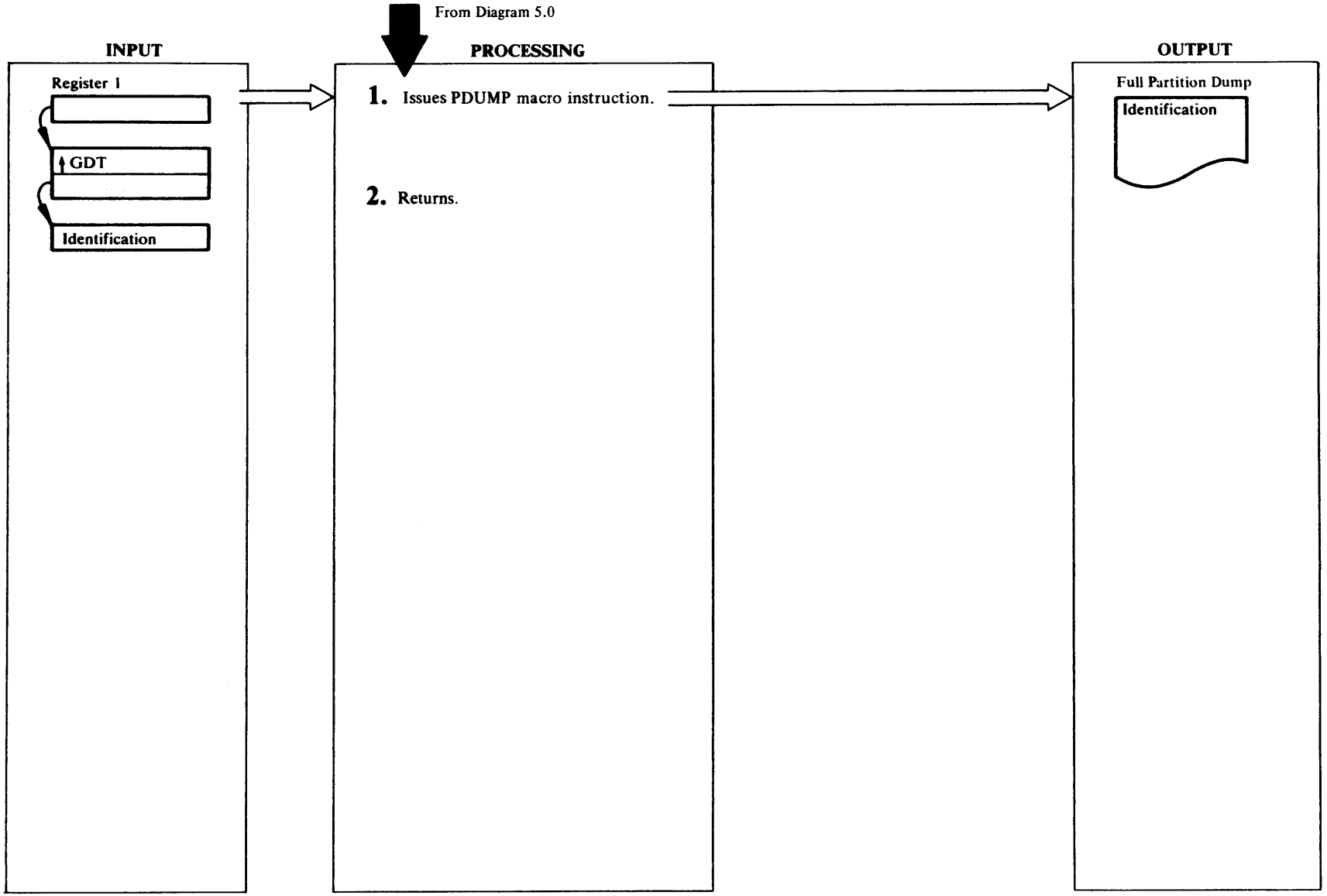
4 GDTIOH provides the address of the IODATA. The address of the IOCSTR chain is IODIOC. The UABORT routine goes through the chain of IOCSTRs and tests each one to determine if it is open. The DTF, for nonVSAM data sets, or the ACB, for VSAM data sets, is checked to determine if the data set is open or 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.

IDCSA01

Procedure: IDCSA01

5 If Access Method Services was invoked through job control, IDCSA01 issues a CANCEL macro to cancel the job. If Access Method Services was invoked through a subroutine call, IDCSA01 returns control to the invoker 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

IDCSA02

Procedure: IDCSA02

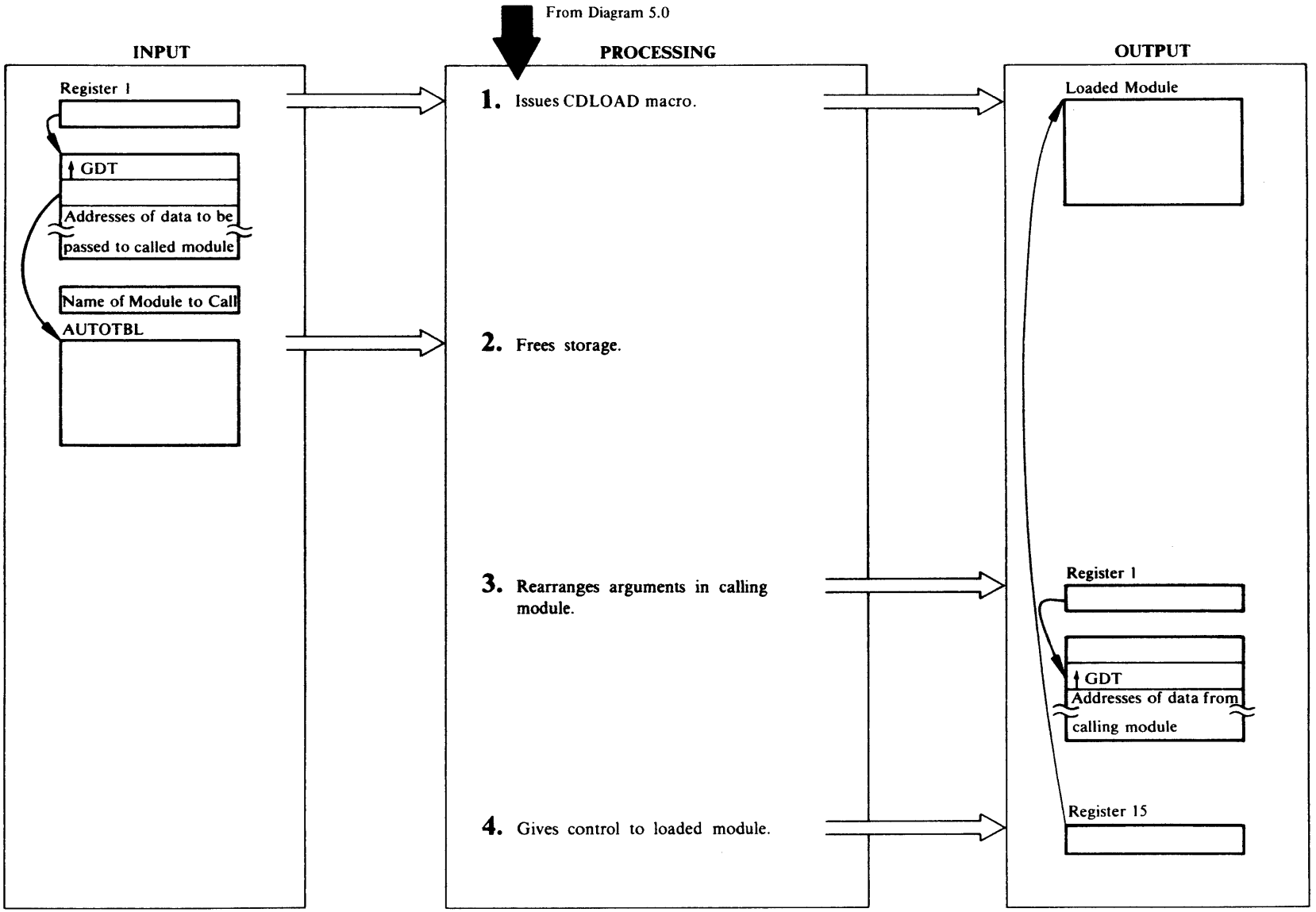
1 IDCSA02 issues an EXTRACT macro to determine the partition beginning and ending addresses for PDUMP. IDCSA01 then issues a PDUMP macro for a full partition dump.

IDCSA02

Procedure: IDCSA02

2 IDCSA02 returns control to the module that issued the USNAP macro.

Diagram 5.3.1. UCALL Macro



Extended Description for Diagram 5.3.1

IDCSA02

Procedure: AMSSACL

1 IDCSA02 loads the program named by the UCALL macro with a CDLOAD macro.

IDCSA02

Procedure: AMSSACL

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 addressed in the AUTOBL 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 FREEVIS and the number in STATUS is decreased by one. If the number in STATUS is one, a FREEVIS is not issued because the storage is saved for the next time IDCSA02 is given control. The status is reduced by one.

IDCSA02

Procedure: AMSSACL

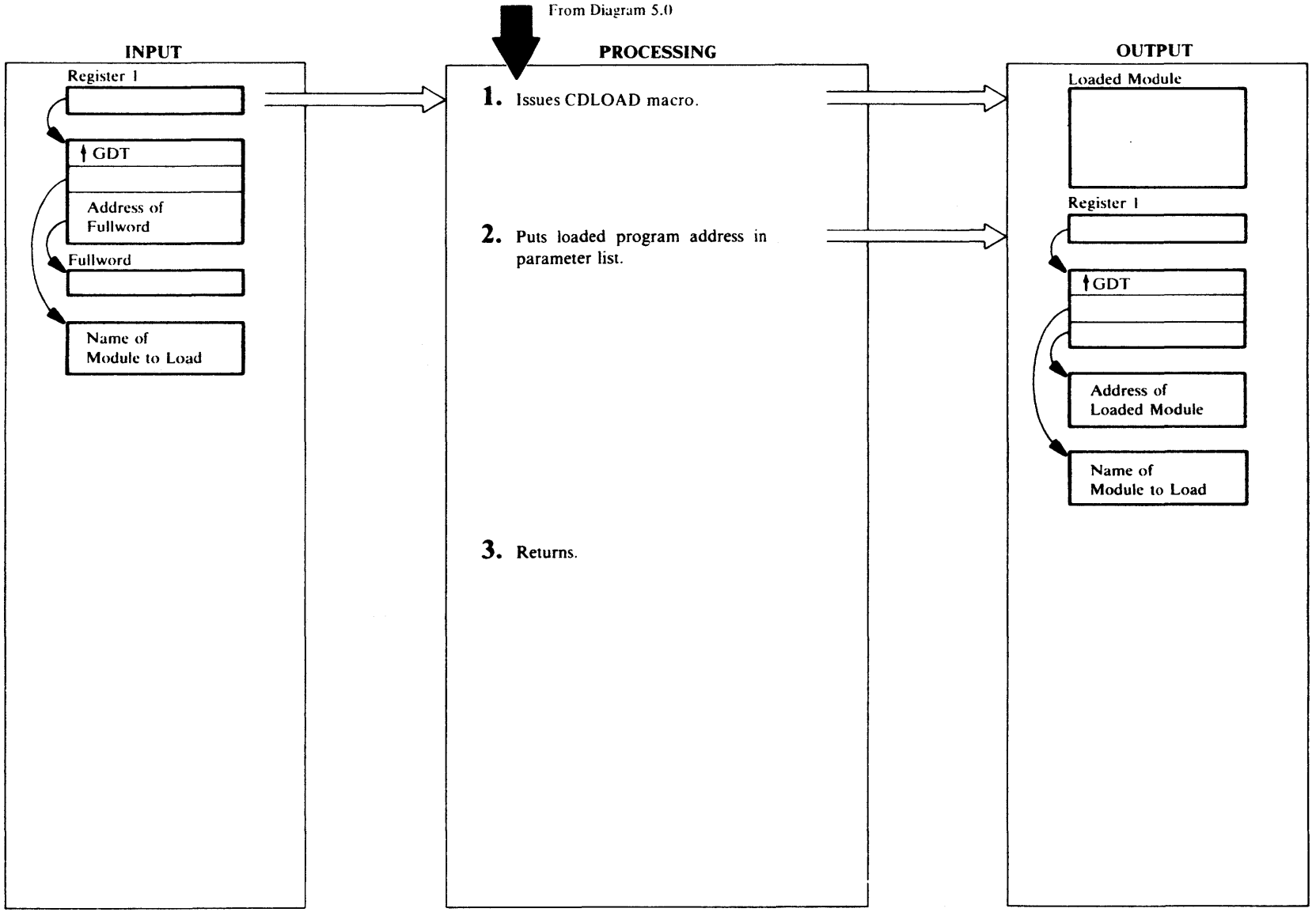
3 IDCSA02 copies the address of the GDT from the first parameter in the calling program to the second parameter in the calling program. IDCSA02 puts the address of the second parameter in the calling program, now the address of the GDT, in register 1. Register one now points to a contiguous list of parameters for the called program.

IDCSA02

Procedure: AMSSACL

4 IDCSA02 puts the address of the called program into register 15. IDCSA02 restores all registers, except 1 and 15, from the calling program's save area and gives control to the called program.

Diagram 5.3.2. ULOAD Macro



Extended Description for Diagram 5.3.2

IDCSA02

Procedure: AMSSALD

1 IDCSA02 issues a CDLOAD macro using the name of the program given to the ULOAD macro.

If the phase is not found, a UABORT is issued unless the caller has requested return of control.

If the anchor table (created by CDLOAD for all models loaded into this partition) is full:

- The phase table in IDCSA04 is searched for this phase name.
- If the phase name is not found, UABORT(52) is issued.
- If the phase is found and if the phase is already loaded, the normal exit is taken to the caller of ULOAD.
- If the phase is not already loaded, a GETVIS is issued for the amount of storage indicated in the phase table for this phase. A GETVIS failure is an ABORT condition.
- The phase is loaded into the GETVIS area and an exit is taken to the caller.

IDCSA02

Procedure: AMSSALD

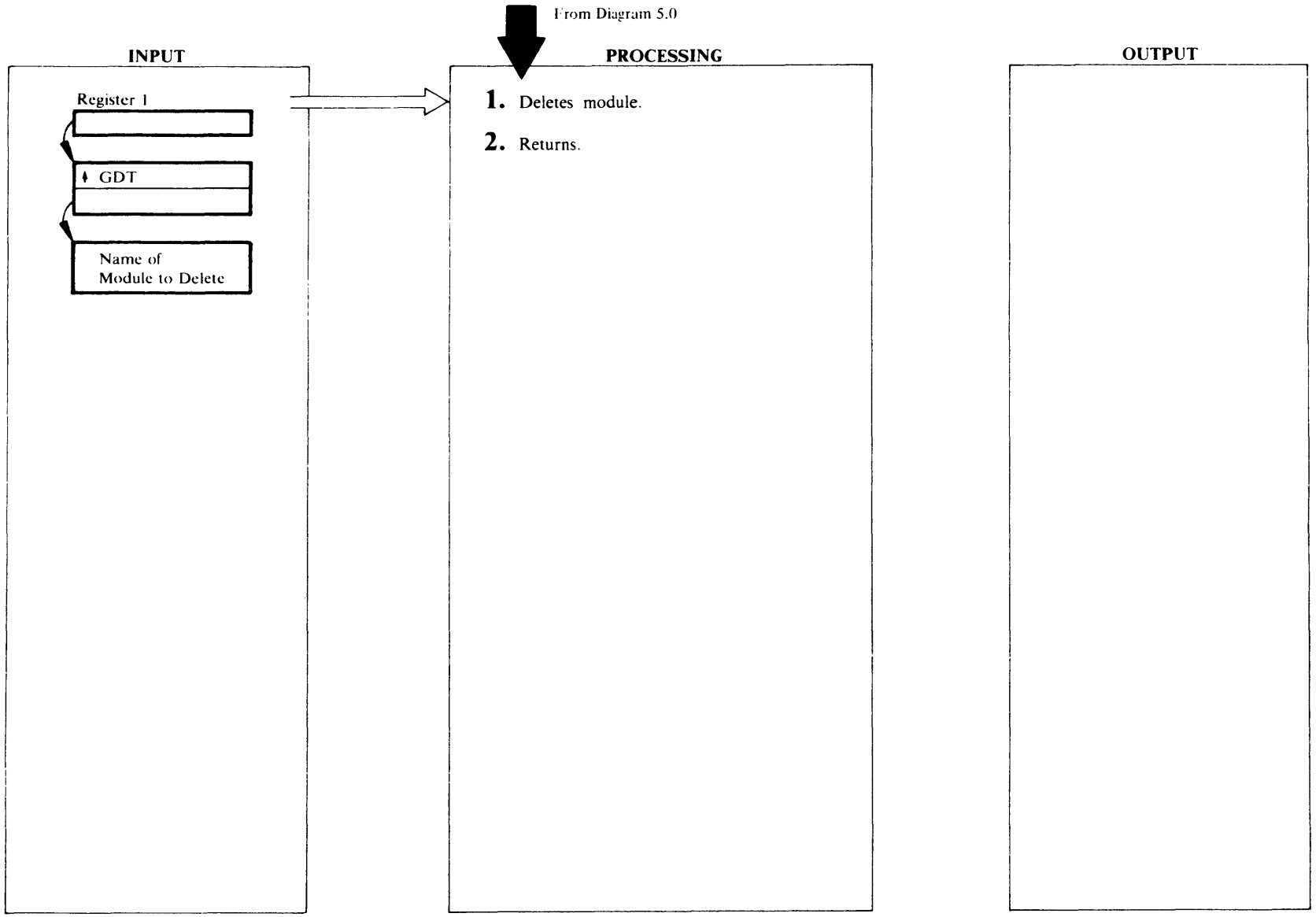
2 IDCSA02 puts the address of the loaded program in the calling program at the address specified with the third parameter.

IDCSA02

Procedure: AMSSALD

3 IDCSA02 returns control to the module that issued the ULOAD macro.

Diagram 5.3.3. UDELETE Macro



Extended Description for Diagram 5.3.3

IDCSA02

Procedure: IDCSA02

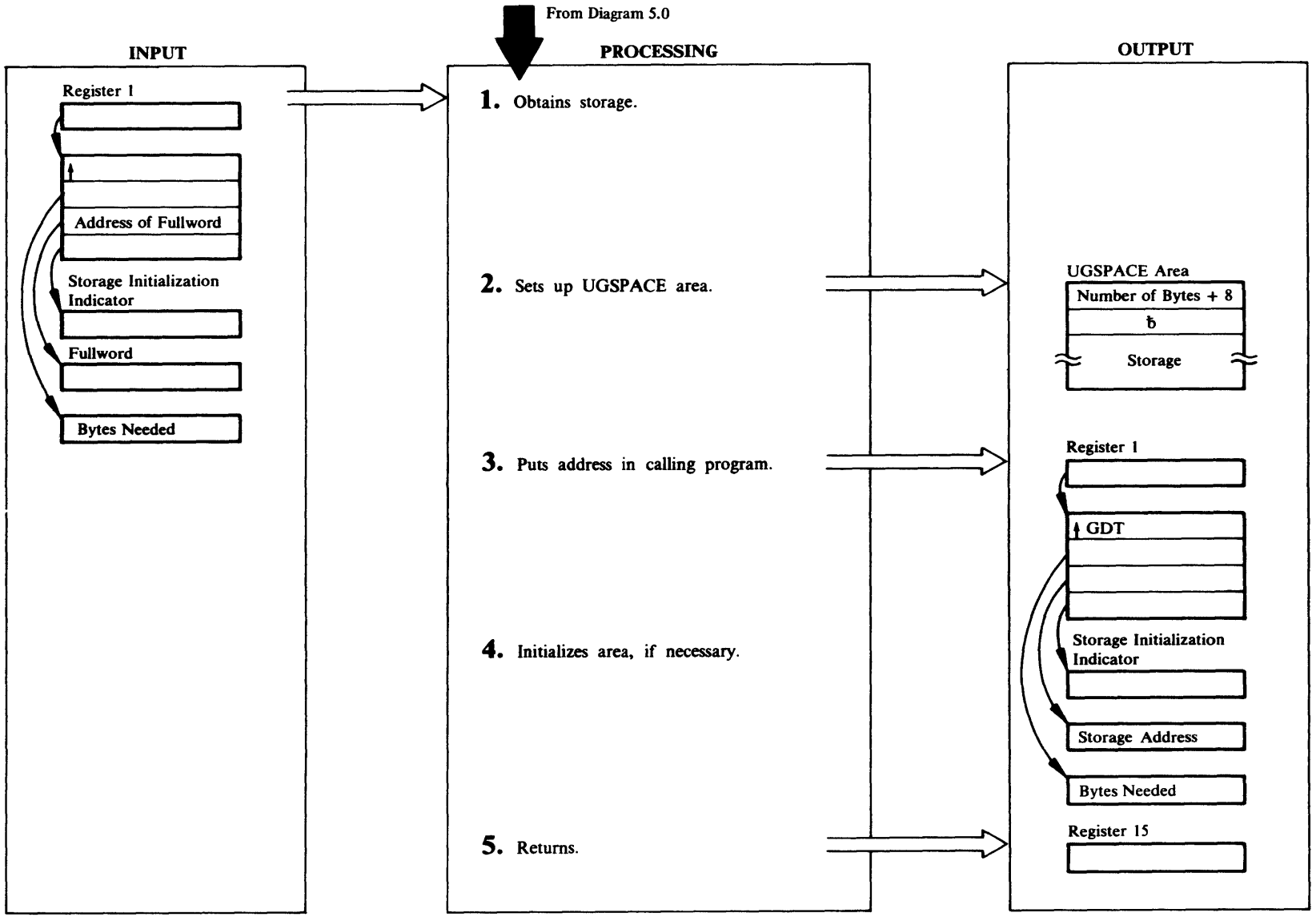
1 IDCSA02 does not delete the module but lets the system paging mechanism delete the module when necessary.

IDCSA02

Procedure: IDCSA02

2 IDCSA02 returns control to the module that issued the UDELETE macro.

Diagram 5.4.1. UGSPACE Macro



Extended Description for Diagram 5.4.1

IDCSA02

Procedure: IDCSA02

1 IDCSA02 issues a GETVIS for the number of bytes requested plus 8 for the UGSPACE area that proceeds each storage area. If the return code from the GETVIS is nonzero, the address of the storage area is set to zero and control is given to step 5. If the return code is zero, control is given to step 2.

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 second word blank to distinguish a UGSPACE area from a UGPOOL area.

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.

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, the storage area is not changed.

IDCSA02

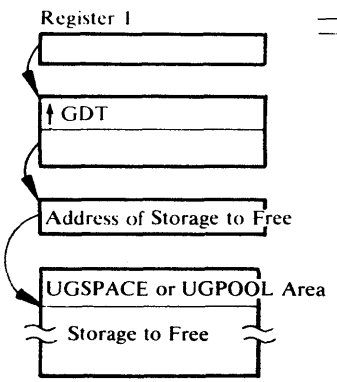
Procedure: IDCSA02

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 Macro

From Diagram 5.0

INPUT



PROCESSING

1. Checks for UGSPACE or UGPOOL area. If UGSPACE, continues to Step 2; if UGPOOL, goes to Step 3.
2. Calculates address.
3. Calculates address and removes area from chain.
4. Frees storage.
5. Returns.

OUTPUT

Extended Description for Diagram 5.4.2

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 fullword at the address minus 4 contains blanks, the area was obtained with a UGSPACE.

IDCSA02

Procedure: IDCSA02

2 If the storage area was obtained with UGSPACE, a UGSPACE area preceeds the area. The length of the area to free is at the first word in the UGSPACE area. The address of the area to free is calculated by subtracting 8 from the area address.

IDCSA02

Procedure: IDCSA02

3 If the storage area was obtained with a UGPOOL, a UGPOOL area preceeds 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.

IDCSA02

Procedure: IDCSA02

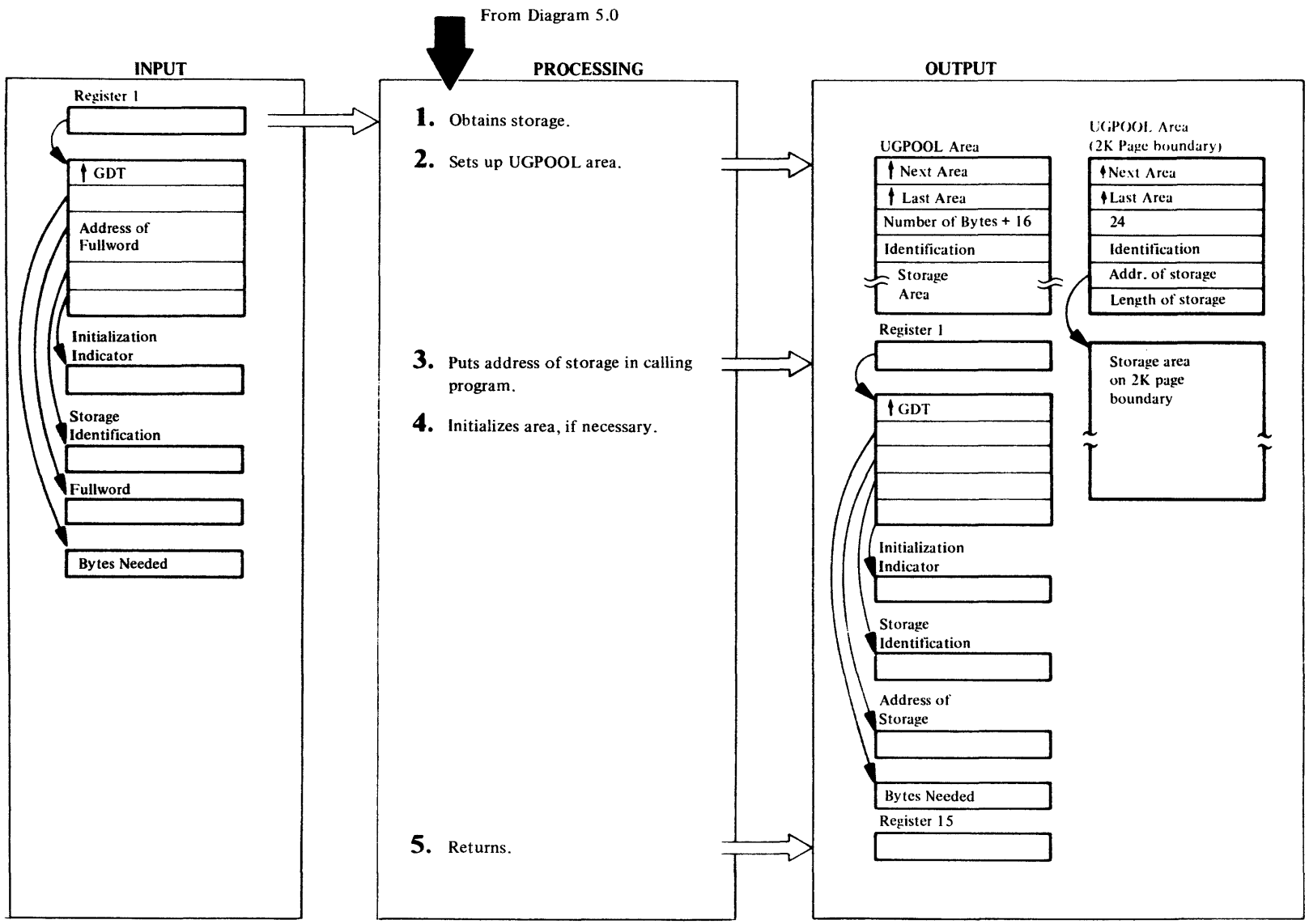
4 A FREEVIS macro is issued to release the storage plus its UGSPACE or UGPOOL area.

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

IDCSA02

Procedure: IDCSA02

1 If the UGPOOL storage identification specifies 'PG' as the third and fourth characters, IDCSA02 issues a GETVIS for the number of bytes requested starting on a 2K page boundary. The address and length is saved. A second GETVIS is issued by IDCSA02 for a 24-byte area. The address and length of the first area obtained are placed in the fifth and sixth words of the 24-byte area. Otherwise, a GETVIS is issued for the number of bytes requested plus 16 for the UGPOOL area. If the return code from the GETVIS is nonzero, the storage address in the calling program is set to zero and control is given to step 5, unless the GETVIS was for a 24-byte 'xxPG' storage area, in which case the space obtained on a 2K page boundary must be freed. A FREEVIS macro is issued to free the space and then the storage address in the calling program is set to zero and control is given to step 5. If the return code from the GETVIS is zero, control is given to step 2.

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 number of bytes in the storage area is the number of bytes requested plus 16 for the UGPOOL 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. The 24-byte area obtained for a 'xxPG' storage area is treated in the same manner as all other UGPOOL areas and chained into the UGPOOL storage area chain. The number of bytes is 24.

IDCSA02

Procedure: IDCSA02

3 IDCSA02 puts the address of the storage area, not the UGPOOL area, in the calling program at the address specified by the third parameter.

IDCSA02

Procedure: IDCSA02

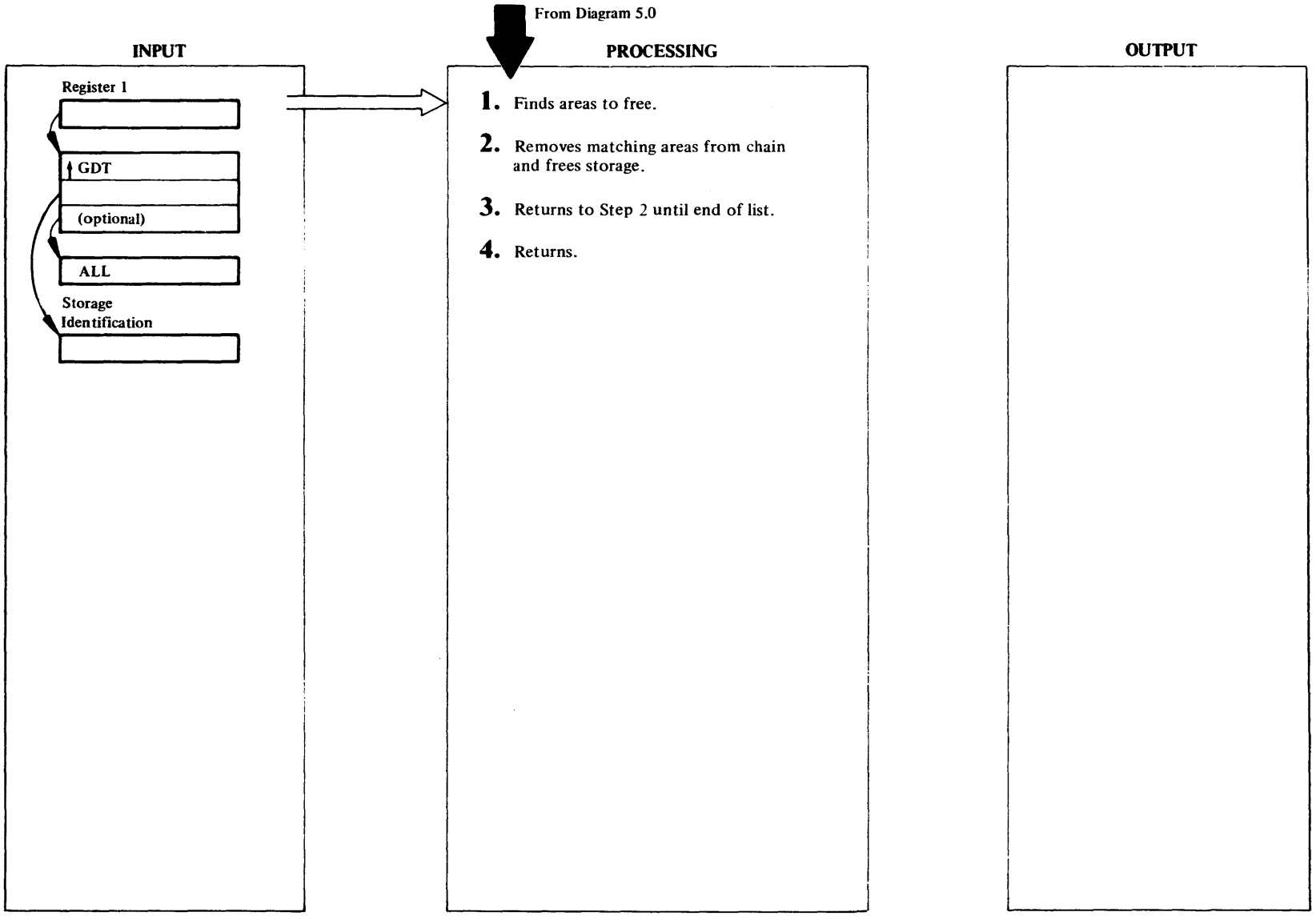
4 If SETZERO or SETBLANK was specified as the fifth parameter, IDCSA02 sets the storage area to zeros or blanks, respectively. If neither SETZERO or SETBLANK is specified, the storage is not changed.

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

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 storage areas to be released.

IDCSA02

Procedure: IDCSA02

2 If a match is found, IDCSA02 removes the UGPOOL area from the chain and releases the UGPOOL area with its storage area with a FREEVIS macro. If the storage identification is 'xxPG', the address and length of the area to be freed is in the fifth and sixth words of the area in the UGPOOL storage chain. IDCSA02 issues a FREEVIS for this area. The 24-byte area in the UGPOOL chain is then freed in the normal manner.

IDCSA02

Procedure: IDCSA02

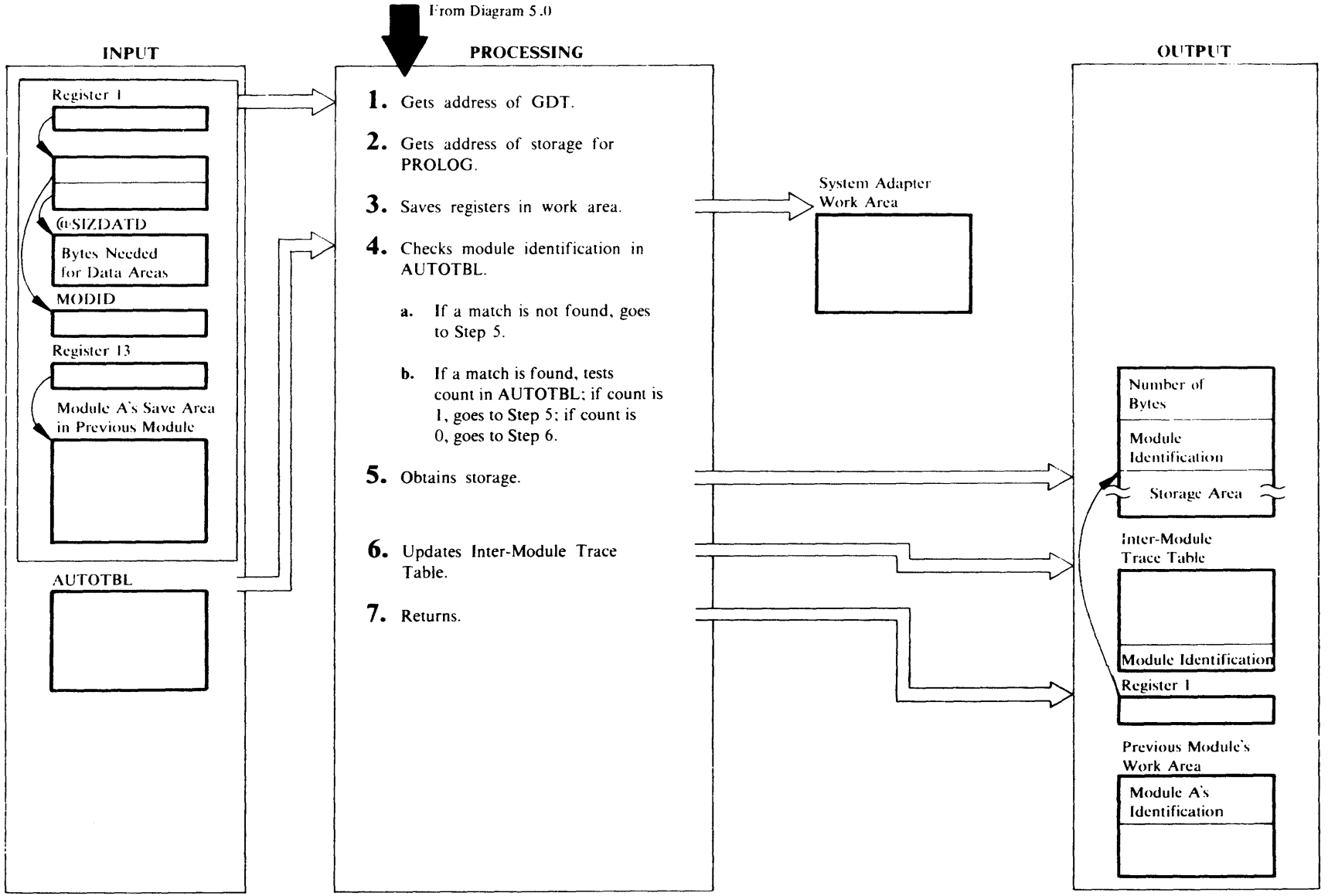
3 If the end of the chain has not been reached, IDCSA02 compares the next UGPOOL area. The entire list is searched for matching identifiers regardless of whether ALL is specified or not. IDCSA02 returns control to step 2 until the end of the chain is reached.

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

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. The first thing module B does is store registers in the save area in module A. The second thing module B does is obtain 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. A call to the PROLOG routine is substituted for the GETMAIN when module B is compiled on VS. So, instead of doing a GETMAIN, module B calls PROLOG to get storage for module B's data areas. At the time 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.

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.

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.

IDCSA03

Procedure: IDCSA03

4 IDCSA03 compares the module identifications in AUTOTBL with the 4 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, IDCI001, or IDCTP01, IDCSA01 may have already obtained storage for it. AUTOTBL contains the address of storage already obtained for IDCSA02, IDCTP01, and IDCIO01. 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 GETVIS and IDCSA03 gives to module B the storage from AUTOTBL for module B's 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 GETVIS. One is added to the number of times the module is called.

IDCSA03

Procedure: IDCSA03

5 If module B did not get storage from AUTOTBL, IDCSA03 issues a GETVIS. 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 GETVIS for the number of bytes in @SIZDATD plus 8 for header information. If the return code from GETVIS is nonzero, IDCSA03 issues a 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.

IDCSA03

Procedure: IDCSA03

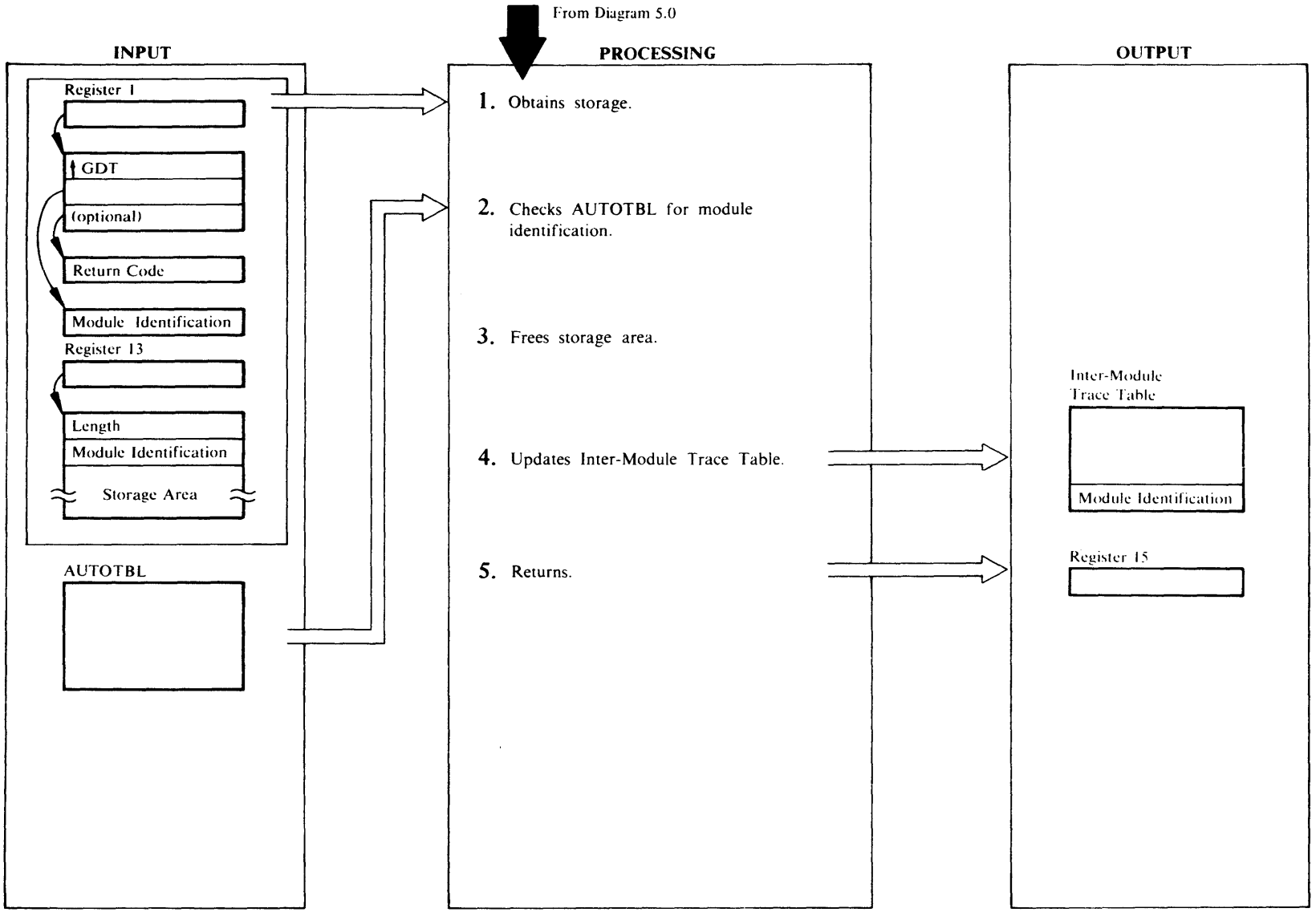
6 IDCSA03 adds module B's identification from MODID to the end of the Inter-Module-Trace table. The first, oldest entry in the table is removed.

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

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 which is 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.

IDCSA03

2 IDCSA03 compares module B's module identification against the module identifications 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 FREEVIS 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.

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 FREEVIS with the length of the storage area as specified in the first word of the header.

IDCSA03

Procedure: IDCSA03

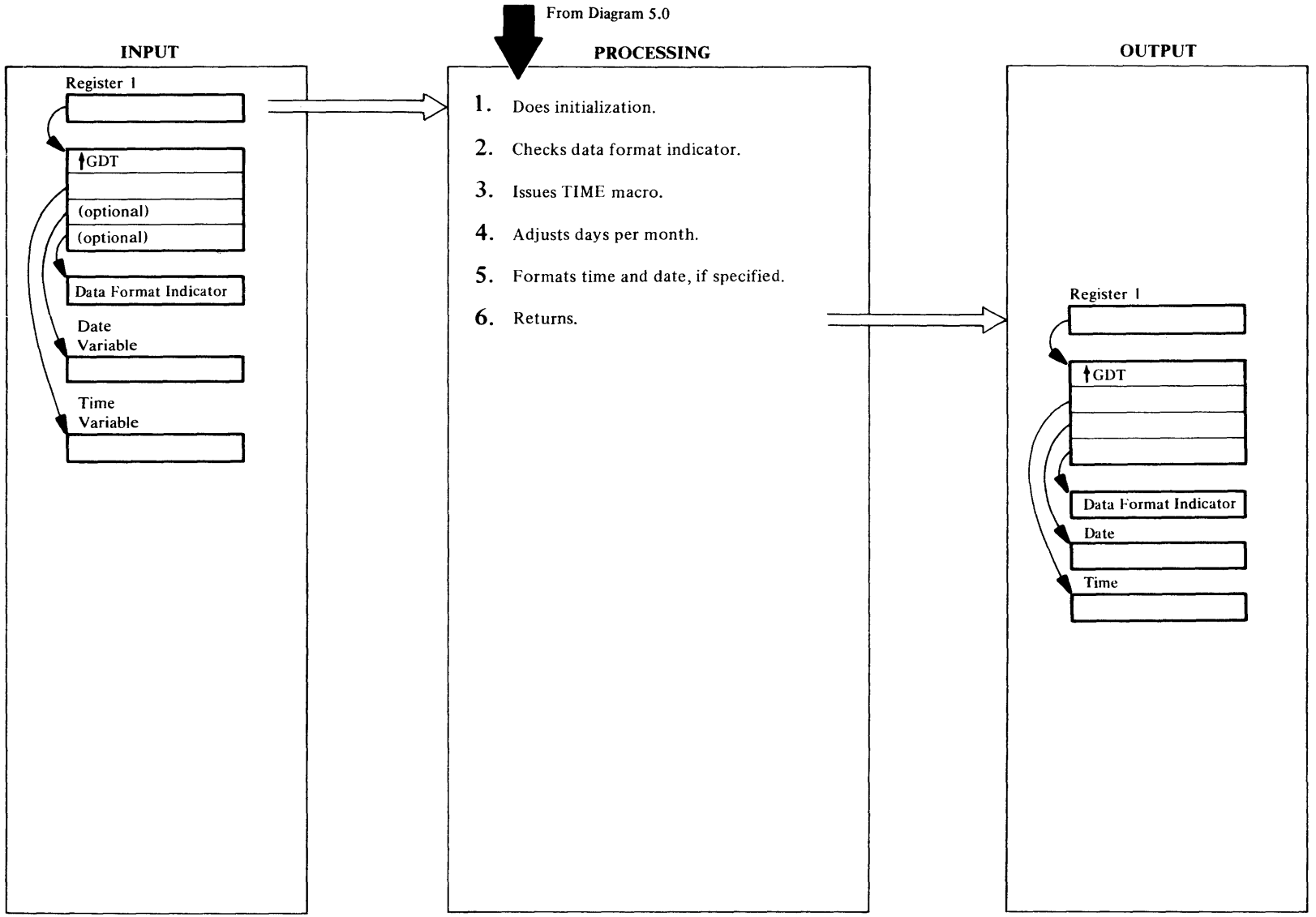
4 IDCSA03 puts the address of module A's save area in register 13. IDCSA03 removes the oldest module identification entry in the Inter-Module-Trace table. IDCSA03 adds module A's module identification to the end of the Inter-Module-Trace table. IDCSA03 obtains module A's module identification from the first word of the save area where module A saved registers when it was given control.

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

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.

IDCSA05

Procedure: IDSCA05

2 If the caller incorrectly specifies the data format indicator, IDCSA05 issues a UABORT macro.

IDCSA05

Procedure: IDSCA05

3 If the caller specifies FORMAT, IDCSA05 specifies a GETTIME macro. If CLOCK is specified, IDCSA05 issues a STCK instruction. If the caller does not indicate the data format, IDCSA05 issues a COMRG macro.

IDCSA05

Procedure: IDCSA05

4 IDCSA05 adjusts the number-of-days-per-month table for leap years. If the year returned by the GETTIME macro is divisible by four, IDCSA05 sets the number of days in February to 29.

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 data is in decimal digits.

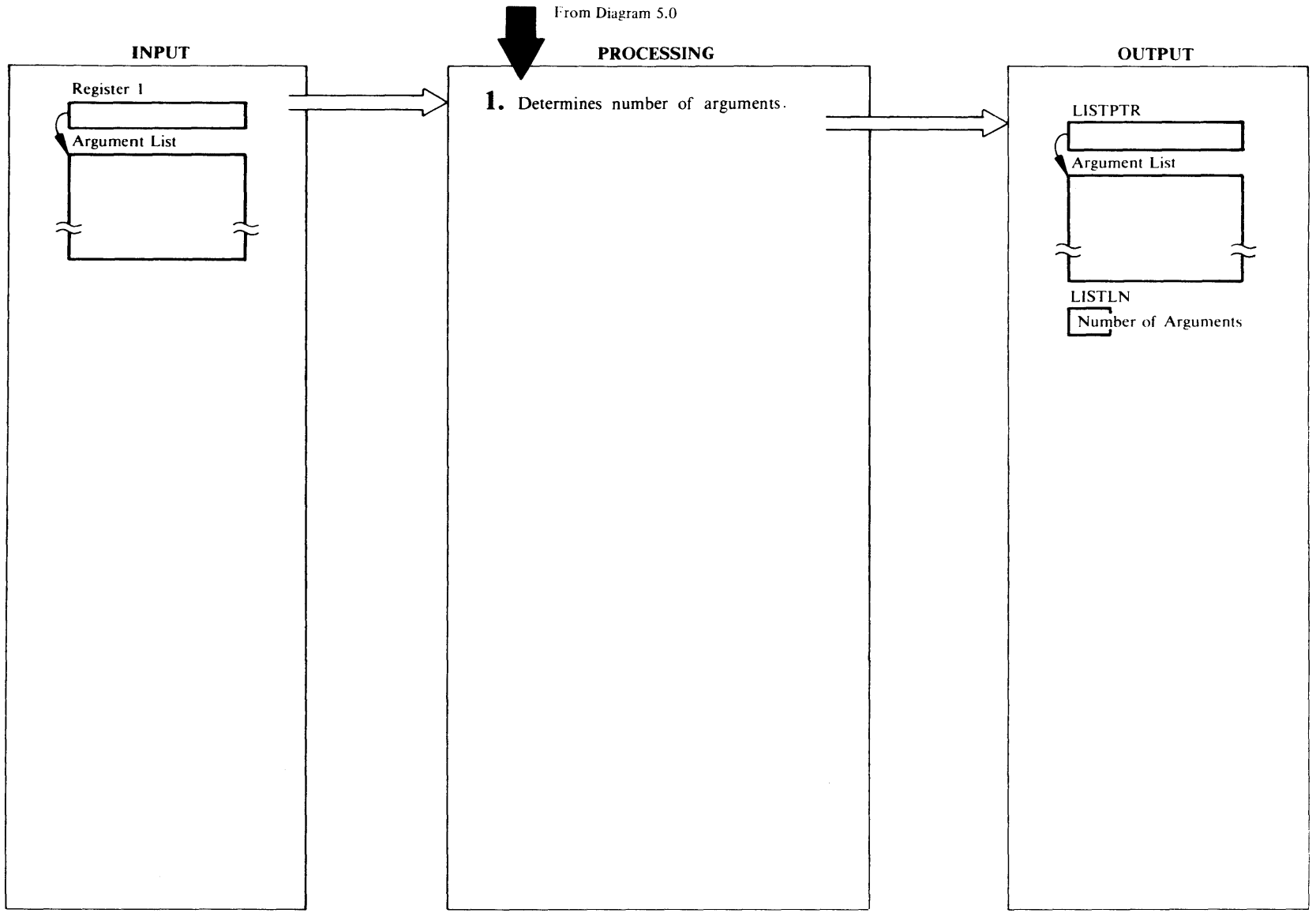
If CLOCK is specified, IDCSA05 returns the time from the time-of-day clock in microseconds. If the date is requested and no data format is indicated, or CLOCK is specified, IDCSA05 returns the date in packed-decimal format, 00YYDDDF, where YY is the year, DDD is the day, and F is the sign digit.

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.

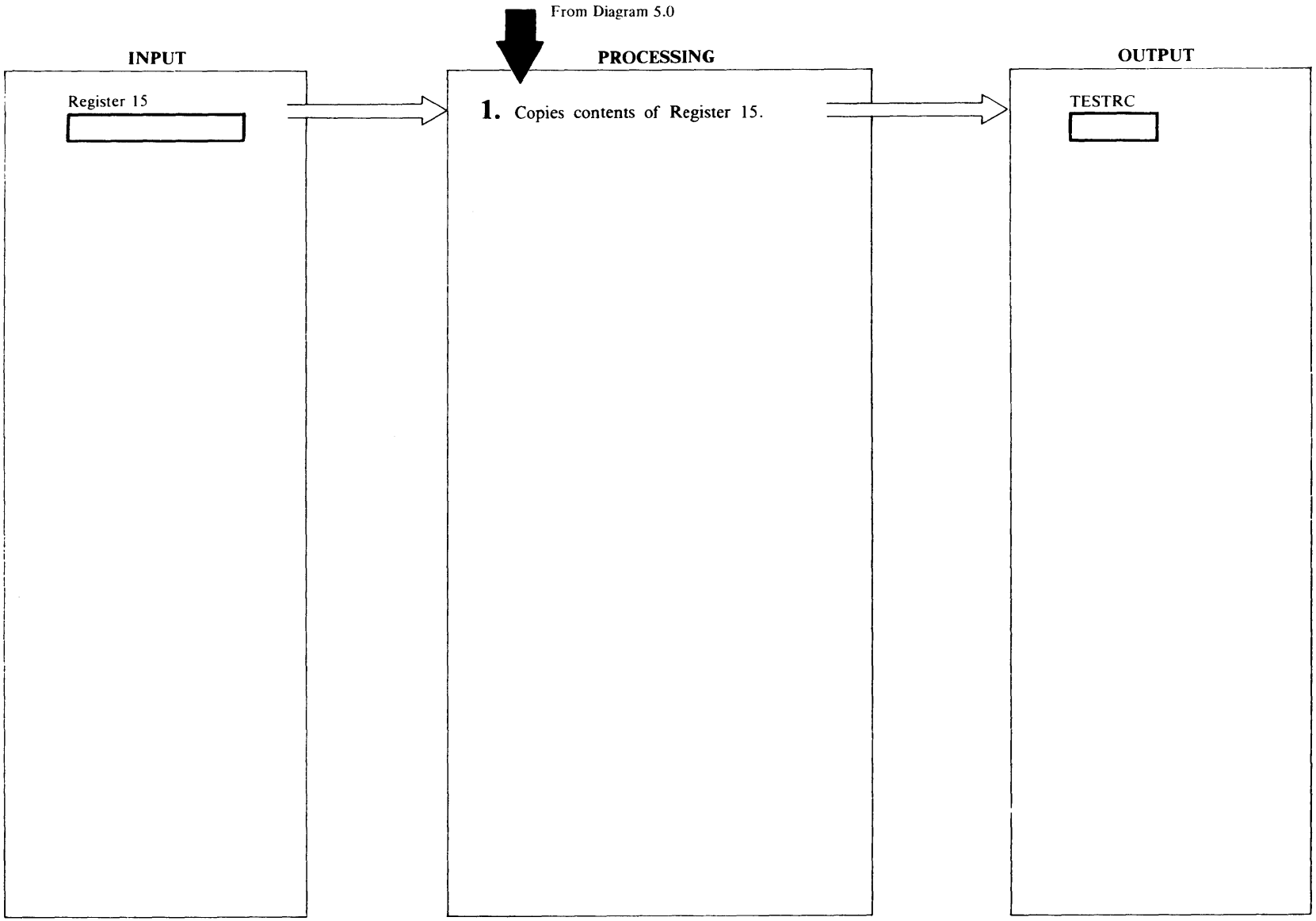
Diagram 5.6.1. ULISTLN Macro



Extended Description for Diagram 5.6.1

1 Unlike most Umacros ULISTLN generates in-line 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 seaches 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.

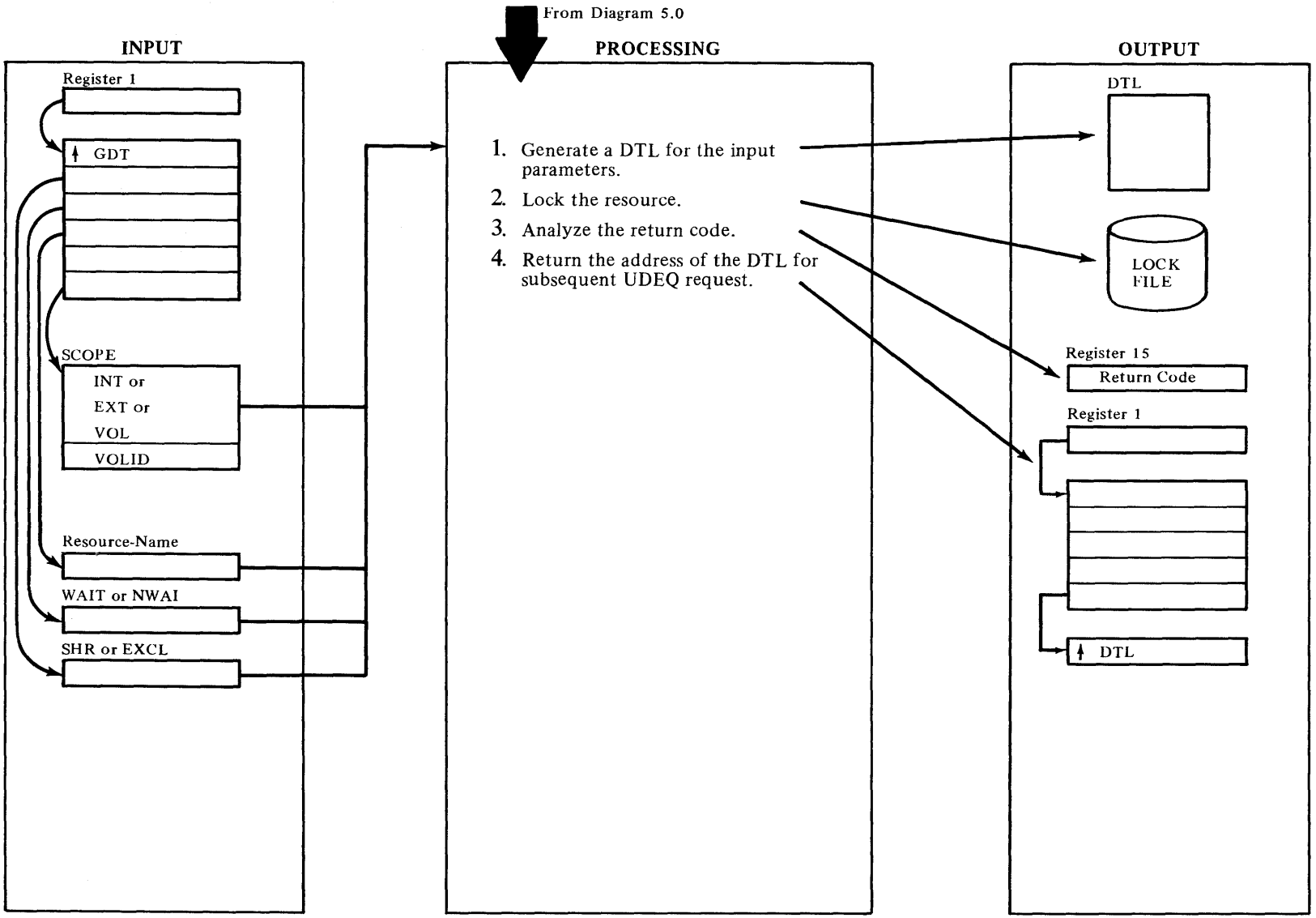
Diagram 5.6.2. USAVERC Macro



Extended Description for Diagram 5.6.2

1 Unlike most Umacros USAVERC generates in-line 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. UENQ Macro



Extended Description for Diagram 5.7.1

IDCSA08

Procedure: IDCSA08

1 A parameter list is built for the IKQDTL macro using the input parameters.

CONTROL is set to "E" (executive) or to "S" (shared).

SCOPE is set to "INT", "SHR", or X'00'. The value X'00' is used if VOLID is present.

VOLIDPTR is set to the address of a 6-byte volid or to 6 bytes of X'00'. The 6-bytes of X'00' are used if the SCOPE parameter is present. If this parameter is used, the supervisor determines if the scope is internal or external by the device address on which the volume is mounted.

GETVIS is specified as yes so that storage will be obtained for the DTL. (Must be freed by IKQUNLK).

If an error occurs, an out-of-storage message is issued (UV0-4) and control is returned to the calling routine with a value of 16 in register 15.

IDCSA08

Procedure: IDCSA08

2 The IKQLOCK macro is issued using the DTL from step 1. RETOPT is set to "WAITC" or "RETURN".

IDCSA08

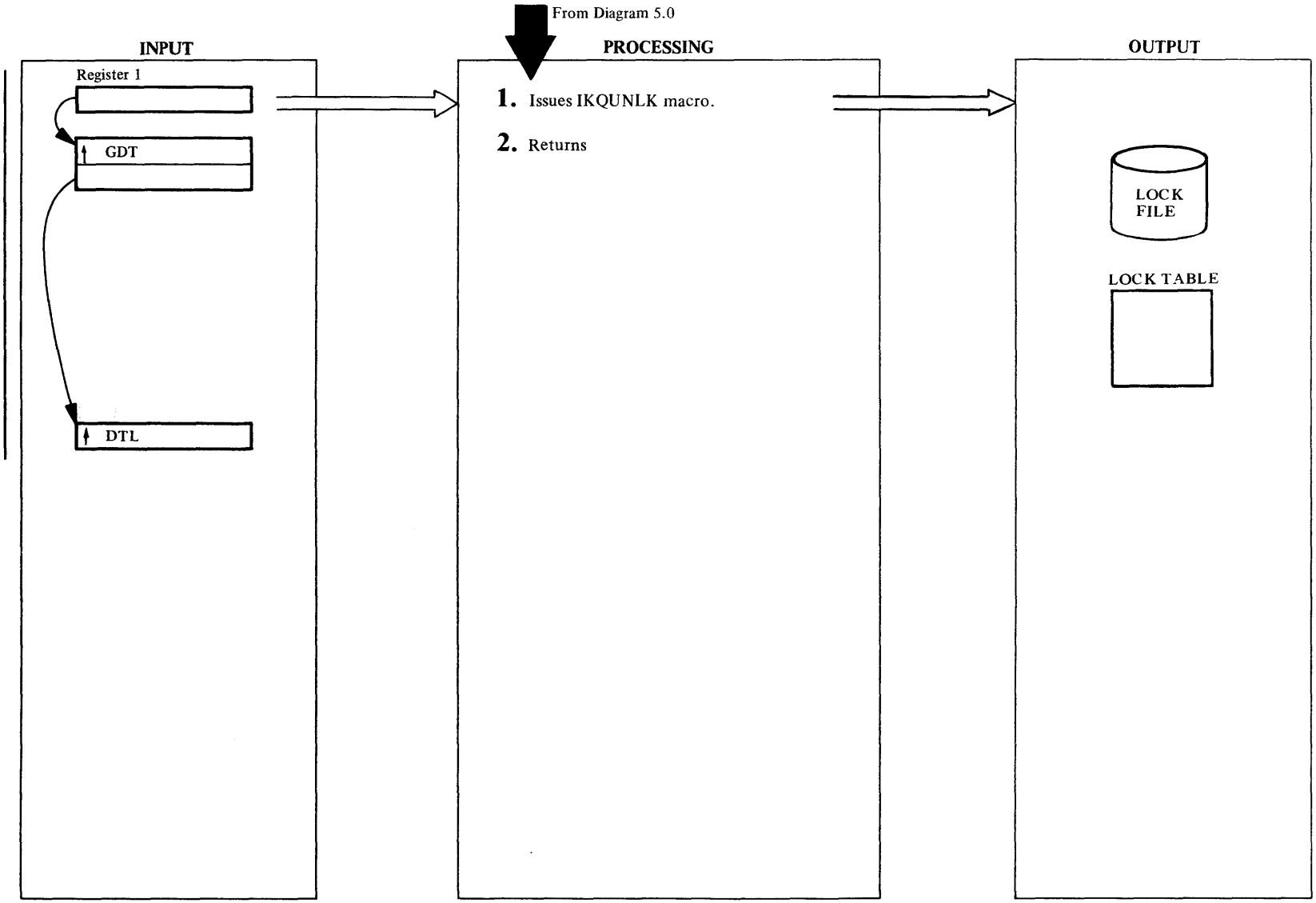
Procedure: IDCSA08

3 The return codes are translated as follows:

Reg 15 =0 Resource has been locked.
=4 (For CONDITION =NOWAIT) resource may become available at a future time (rc=4, 8, 28 from lock manager, that is, SUPVR).
=8 This task already owns this lock (rc=24 from lock manager).
=12 Definition error (rc=12, 16, 20, 32, 36 from lock manager).

For a lock manager return code of 4 or 24, no error message is issued. For a return code of 24, IKQUNLK is not called because the lock would be released in addition to freeing the DTL. For any other non-zero return code, a message is issued (UV0-8) and IKQUNLK is called to free the DTL.

Diagram 5.7.2. UDEQ Macro



Extended Description for Diagram 5.7.2

IDCSA08

Procedure: IDCSA08

1 IDCSA08 issues an IKQUNLK macro to release control of the resource. If the address of the DTL to be unlocked is zero, no unlock is necessary.

IDCSA08

Procedure: IDCSA08

2 IDCSA08 returns contol to the module that issued the UDEQ macro. The return code is always 0.

I/O Adapter Visual Table of Contents

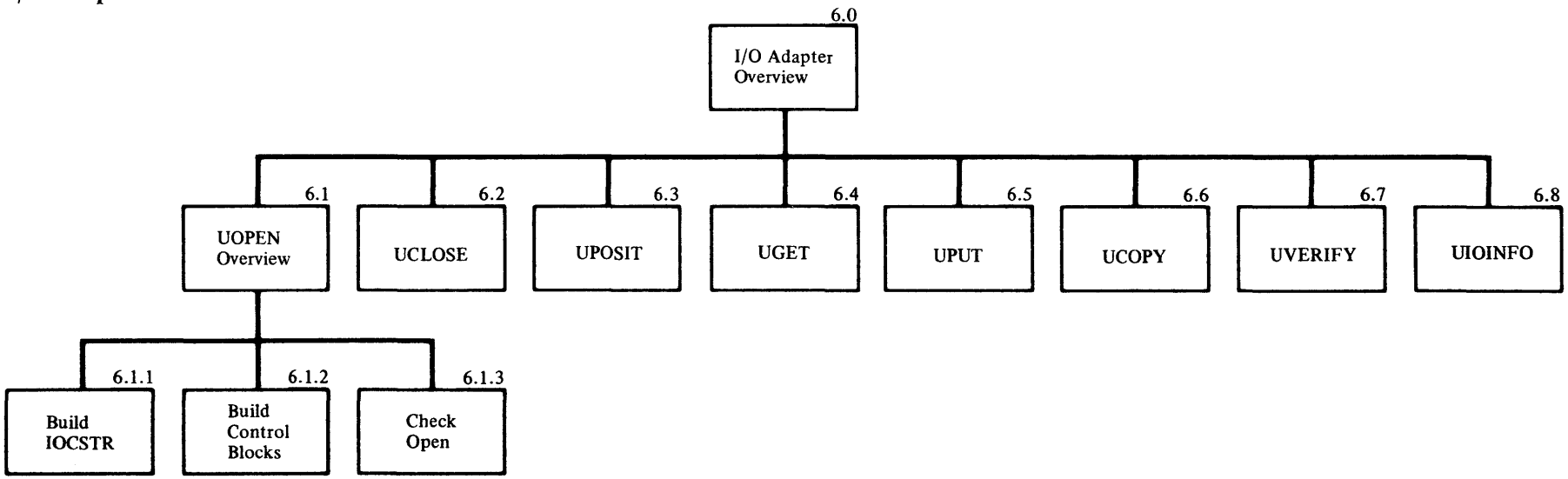
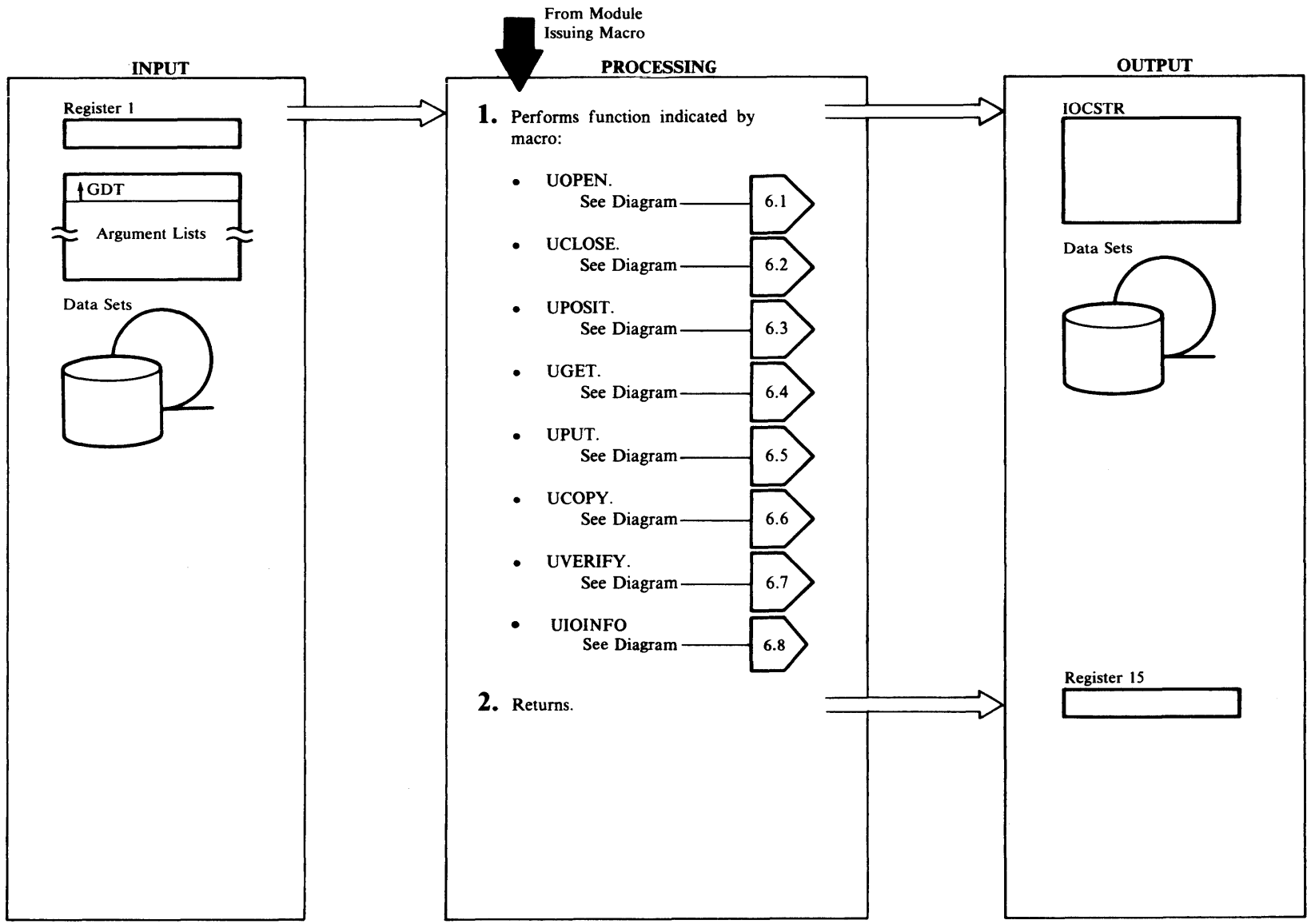


Diagram 6.0. I/O Adapter Overview



Extended Description for Diagram 6.0

IDCIO01

Procedure: IDCIO01

1 The type of I/O processing depends upon the Umacro issued:

- The UOPEN macro opens from one to four data sets.
- The UCLOSE macro closes from one to four data sets that were opened by the I/O Adapter. SYSIPT and SYSLST are not closed with this macro, but at processor termination with the UIOTERM macro. This is done to consolidate termination work.
- The UPOSIT macro is used to position to a record in a data set on a direct access device. The type of positioning depends upon the data set organization:

  For VSAM data sets, the positioning may be by key, relative byte address (RBA), or relative record number.

  For ISAM data sets, the positioning is by key only.
- The UGET macro is used to obtain a record from a data set opened with a UOPEN macro. If the data set is being processed with keys - ISAM or indexed VSAM - the key is returned with the record. If the data set is being processed with control intervals - VSAM with block processing - a control interval is returned. If a relative-record data set (RRDS) is being processed, a relative record number is returned. Only if the VSAM data set is opened for update processing may the record be modified in the buffer. Data sets opened for update processing must be processed with a UGET followed by a UPUT on the same record just obtained. This is true regardless of whether or not the record has been changed. A UPUT must be issued after each UGET, for UPDATE, even if it is the last UGET before the data set is closed. Update processing is used when the REPLACE option has been specified for the REPRO function.
- The UPUT macro is used to write 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 VSAM data set opened for update.
- 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:

  a. If the data set is VSAM with records in keyed sequential or relative record format, the input records are merged with the existing records.

  b. If the data set is VSAM with entry sequential record format, the input records are added after the existing records.

  c. If the data set is nonVSAM, the input records are written over the existing records. The existing records are lost. ISAM data sets cannot be used for output for UCOPY.
- The UVERIFY macro insures 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 FSR should ignore the return code form 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.
- The UIOINFO macro is used to obtain information concerning a data set. 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
  Timestamp

  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.)

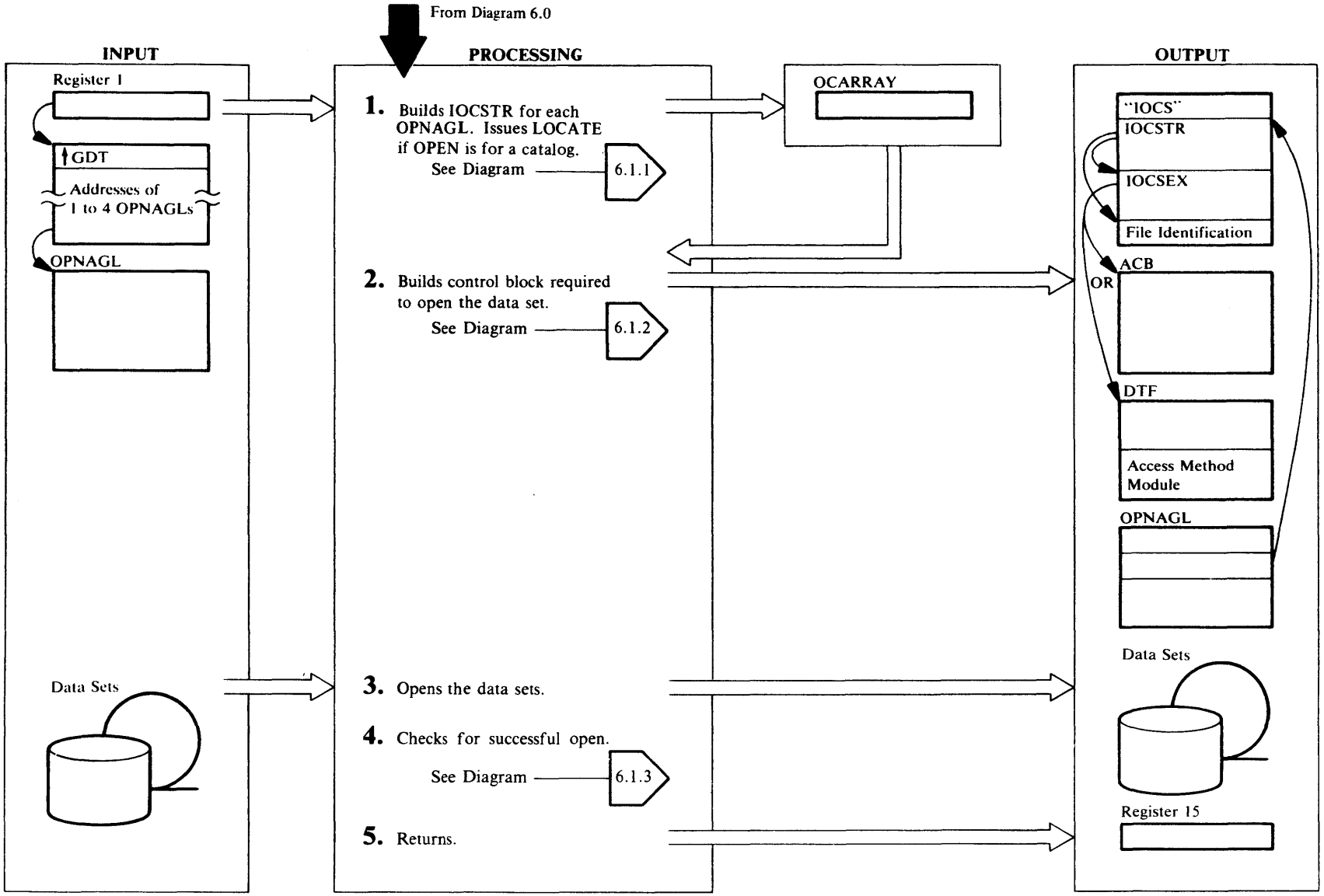
  The data requested is formatted into the return area and control is returned to the caller.

IDCIO01

Procedure: IDCIO01

2 A return code is put in register 15. If the return code is nonzero, error messages are written. Control returns to the module that issued the Umacro.

Diagram 6.1. UOPEN Macro



Extended Description for Diagram 6.1

IDCIO01, IDCIO02

Procedures: IDCIOOP, OPENRTN, DSDATA

1 IDCIOOP builds an internal array (OCARRAY) to describe the open to be performed. The rest of step 1 and all of step 2 are repeated for each open argument list (OPNAGL) that the calling module give to the UOPEN macro via register 1. 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 file identification save area. OPENRTN puts the IOCSTR into the chain of IOCSTRs addressed from IODIOC in the I/O Adapter Historical Data Area, IODATA.

DSDATA loads the VSAM IKQVLAB routine with a CDLOAD macro. The FILENAME and the address of a work area are passed as arguments.IKQVLAB reads the LABEL CYLINDER and returns information about the file in the work area. DSDATA saves the FILE ID and file organization.

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.

If the OPNAGL indicates that the open is for a catalog, OPENRTN issues a catalog Locate requesting the return of the catalog ACB address. Control is then passed to step 5.

If the open is not for a catalog, control is passed to Step 2.

IDCIO02

Procedures: BUILDACB, BUILDDBK

2 If the data set organization is VSAM, BUILDACB builds an EXLIST and an ACB control block. BUILDACB puts the addresses and length of the control blocks in the IOCSEX. If the data set organization is nonVSAM, BUILDDBK loads a module containing a DTF control block and the Access Method Module required to process the data set. BUILDDBK uses a table of module names and data set characteristics to find the right module to load. BUILDDBK updates the DTF with information from the OPNAGL. BUILDDBK uses a UGPOOL macro to obtain storage for subsequent GET/PUT operations. If the record format is spanned, one storage area is obtained, otherwise, two storage areas are obtained. The address of the ACB or DTF is put in IOCCBA in the IOCSEX.

IDCIO02

Procedure: OPENRTN

3 OPENRTN issues one OPEN macro for each ACB or DTF built in step 2. There are no exit routines. If OPEN detects an abend condition, OPEN abends.

IDCIO02

Procedures: OPENRTN, CKNONOP, BUILDRPL

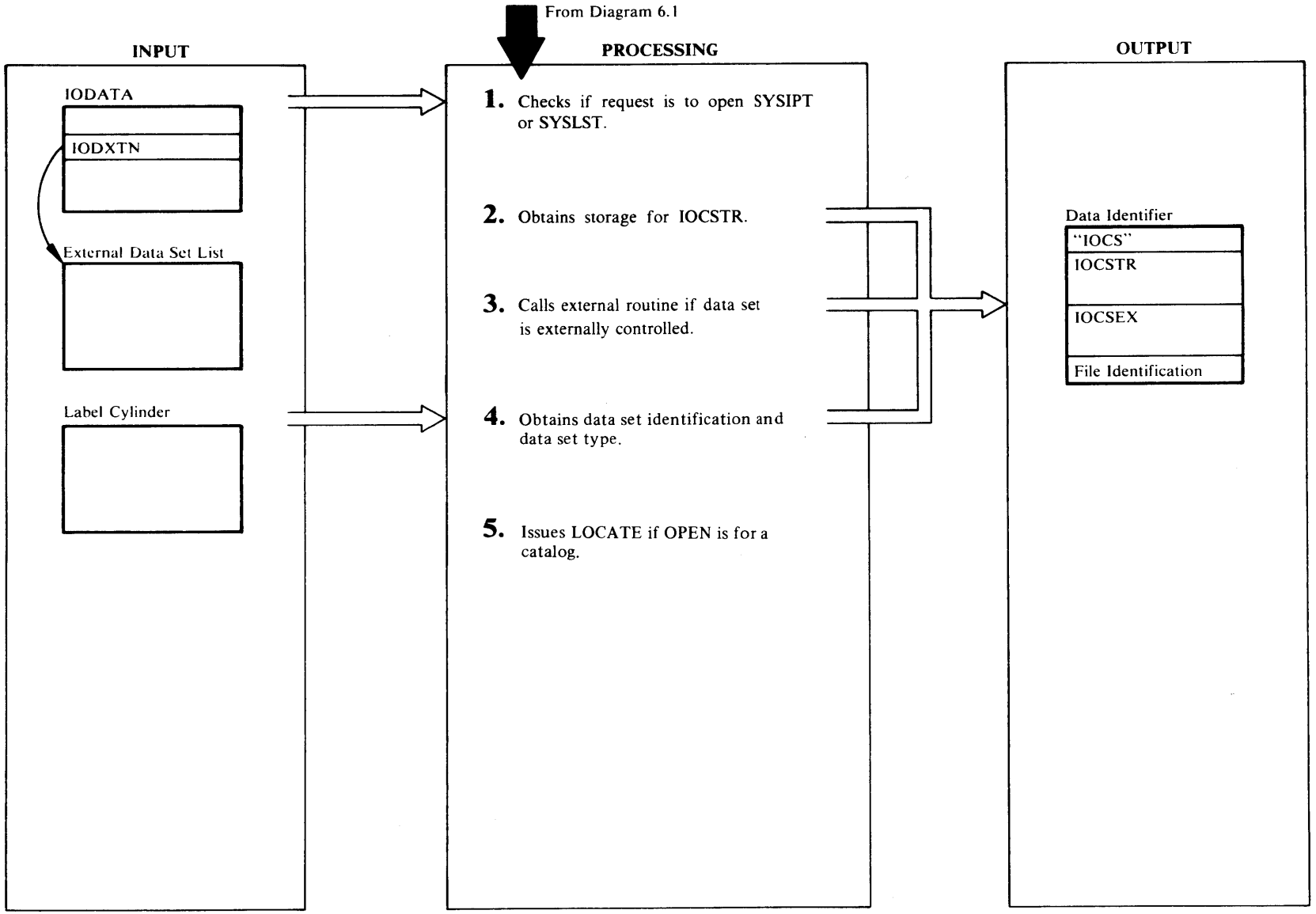
4 OPENRTN and CKNONOP test each data set for a successful open. If the data set is VSAM, OPENRTN tests the results of the OPEN. If the data set is sequential nonVSAM, CKNONOP checks the open flags in the DTF. No checking is done on ISAM or device independent data sets. If the data set opened successfully, OPENRTN and CKNONOP set IOCMSGOP in the IOCSTR and IOCFLGOP in the IOCSEX. If address or control interval processing is not specified in the OPNAGL for a VSAM data set, OPENRTN determines if the data set has an index. A second test is performed to determine if the data set is a Relative Record data set (RRDS). For all VSAM data set, OPENRTN obtains data set information and BUILDRPL builds a RPL to process the VSAM data set. For an ISAM data set, CKNONOP issues a SETL macro to position to the first record. CKNONOP obtains data set information from the ISAM DTF and saves it in the IOCSTR.

IDCIO02, IDCIO01

Procedures: OPENRTN, DSDATA, BUILDACB, BUILDRPL, CKNONOP, IDCIOOP

5 If any errors occurred, any of the procedures that check for error conditions sets a nonzero return code in register 15. IDCIOOP returns control to the module that issued the UOPEN macro.

Diagram 6.1.1. UOPEN Macro – Build IOCSTR



Extended Description for Diagram 6.1.1

IDCIO02

Procedure: OPENRTN

1 OPENRTN tests the OPNAGL for an open request for SYSIPT or SYSLST. SYSIPT is tested in two ways:

- SYSIPT is the Dname in the OPNAGL.
- OPNTYPSI flag in OPNAGL is on.

SYSLST is tested in two ways:

- SYSLST is the Dname in the OPNAGL.
- OPTYSO flag in OPNAGL is on.

If the file is SYSIPT, OPENRTN checks IODICS for an address of an IOCSTR already built for SYSIPT. If an IOCSTR is built, SYSIPT is already open (or an open was attempted), and OPENRTN returns the address of the IOCSTR for SYSIPT in the area addressed by OPNIOC in the OPNAGL. No further processing is done on SYSIPT. If the data set is SYSLST, OPENRTN checks IODOCS for an address of an IOCSTR already built for SYSLST. If an IOCSTR is built, SYSLST is already open and OPENRTN returns the address of the IOCSTR for SYSLST in the adrea addressed by OPNIOC in the OPNAGL. No further processing is done on SYSLST.

If the data set is not open, continue to Step 2.

IDCIO02

Procedures: OPENRTN, PRINTMSG

2 OPENRTN increments by 1 the file identifier in IODSID to form a unique identifier for the data set. OPENRTN issues a UGPOOL macro with the file identifier to obtain storage for the IOCSTR plus 4 bytes for the characters 'IOCS', the IOCSEX, and the *file id*. *file id* is the name of the data set. **Note:** the file identifier that the I/O Adapter creates is different from the *file id*. 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 SYSIPT or SYSLST, OPENRTN saves the address of the IOCSTR in the IODATA. OPENRTN checks the requested processing of the data set specified in OPNOPT in OPNAGL for input, update, or output, and copies it 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.

IDCIO02

Procedure: OPENRTN

3 If the invoker of Access Method Services supplied a list of TLBL/DLBL 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 Dname in OPNDDN in OPNAGL. If a match is found, OPENRTN puts the address of the external routine in IOCXAD. OPENRTN also builds a parameter list for the external routine and puts the address of the first parameter in the list in IOCXPM. OPENRTN then gives control to the external routine to do the open. For lack of any information about the external data set, OPENRTN sets the IOCSTR to indicate the data set is nonVSAM with variable length records and logical record length of 32,760. This does not restrict the type of data sets that can be externally controlled. It is just to make the data set appear as something to the FSR that requests the data set be used. If a data set is not externally controlled, control continues with step 4.

IDCIO02

Procedures: DSDATA, PRINTMSG

4 Information must be obtained from job control if: (a) the data set is not SYIPT or SYSLST, or (b) a DLBL name OPNDDN was passed as input to UOPEN. DSDATA issues a CDLOAD macro to load IKQVLAB, the VSAM Read Label Cylinder module. If the return code from CDLOAD is nonzero, DSDATA issues a UABORT macro. If the return code is 12 (indicating insufficient storage), DSDATA sets the UABORT code to 28, otherwise DSDATA sets the UABORT code to 64. DSDATA gives control to IKQVLAB. If the return code is nonzero, PRINTMSG writes a message and the UOPEN for the data set terminates. If the return code is zero, IKQVLAB placed information about the data set in a work area. Data set organization and *file id* are set in the IOCSTR and IOCSEX. For SYSIPT and SYSLST the *file id* is assumed to be the FILENAME and the data set organization is assumed to be physical sequential with record size of 80 for SYSIPT and 121 for SYSLST. If the OPNAGL specifies device type of 2400, the data set is assumed to be a tape and the information returned by IKQVLAB is from a TLBL statement. If the device type is not 2400, DSDATA checks the DLBL for ISAM or VSAM. If neither ISAM or VSAM is specified, the data set is assumed to be physical sequential nonVSAM.

For all data sets, DSDATA puts the *file id* in the file identification area addressed from the IOCSTR.

If the OPNAGL indicates that a catalog recovery area is being opened, DSDATA sets VSAM data set organization in the IOCSTR. If the OPNAGL indicates that a catalog recovery area is being opened, 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.

IDCIO02

Procedures: OPENRTN, PRINTMSG

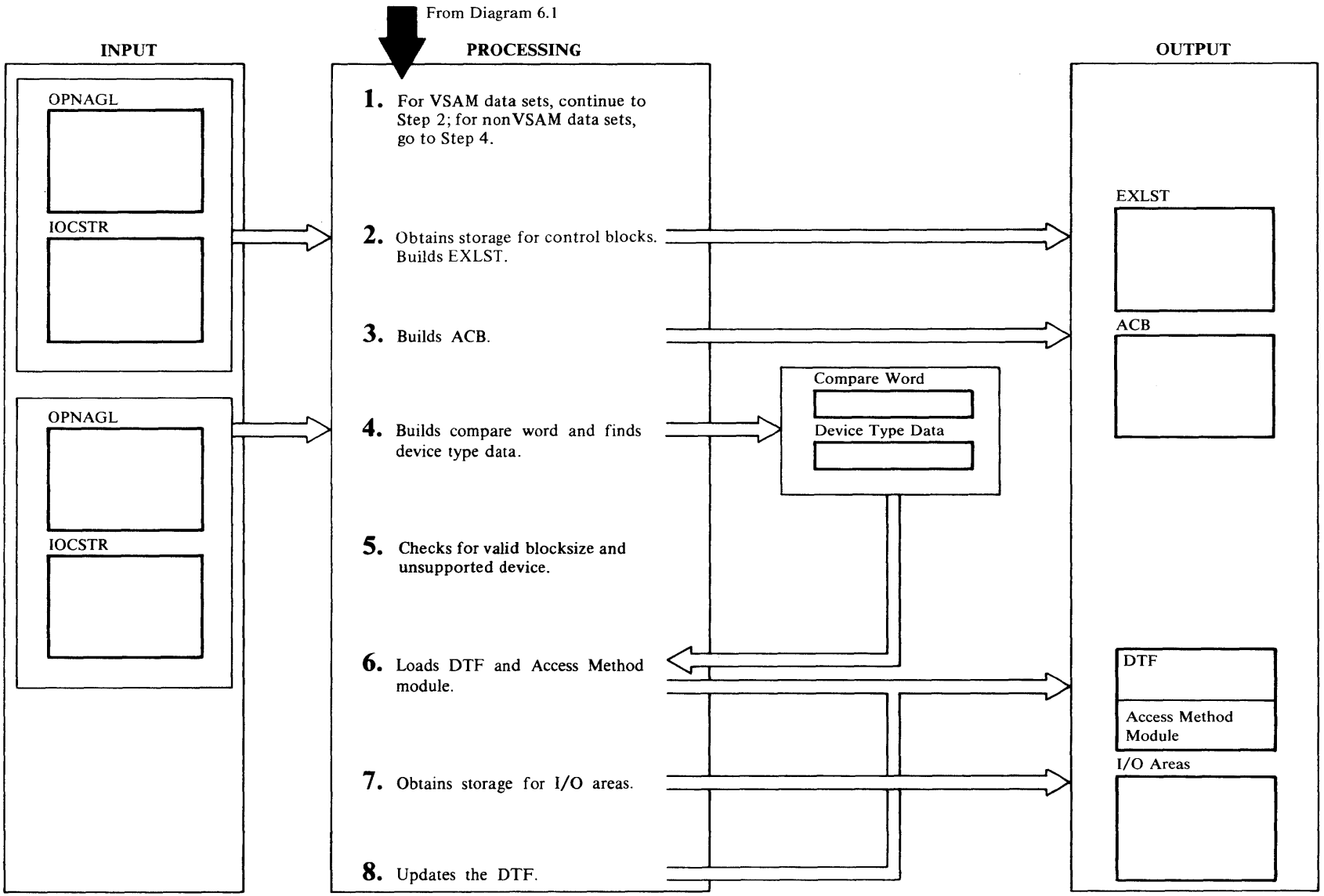
5 If the data set to be opened is a VSAM catalog, as indicated by IOCINFCT, a VSAM Locate is issued via the System Adapater UCATLG macro. OPENRTN builds a CTGPL and one CTGFL. The name used in the Locate (pointed to by CTGCAT and CTGENT) is the name as returned from IKQVLAB and contained in LABDSN. CTGPSWD is set equal to OPNPWA if a password has been specified via the OPNPWA field. The address of the catalog *dname* passed in OPNDDN is placed in CTGDDUC. The CTGFL requests the return of the catalog ACB address, CATACB. If the return code is nonzero, PRINTMSG writes a message. For all VSAM catalogs, control passes to the final phase of UOPEN for VSAM data sets.

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
OPNOPTCI	IOCMACCI = '1'	Export CIMODE
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

If OPNOPTBK or OPNOPTKS is not specified, IOCMACCR is set to '1'.

Diagram 6.1.2. UOPEN Macro – Build Control Blocks



Extended Description for Diagram 6.1.2

IDCIO02

Procedure: BUILDACB

1 For VSAM data sets continue to step 2; for nonVSAM data sets go to step 4.

IDCIO02

Procedure: BUILDACB

2 BUILDACB issues a UGPOOL to obtain storage for the three VSAM control blocks: EXLST, ACB, and RPL. If OPNSTRNO is 0, BUILDACB obtains storage for one RPL; otherwise the value of OPNSTRNO determines the number of RPLs required. If the return code from UGPOOL is nonzero, BUILDACB sets an error condition and terminates UOPEN processing.

BUILDACB first builds an EXLST control block issuing the EXLST macro. Only the EODAD exit will be taken if GETVSAM encounters an end-of-file. LERAD and SYNAD exits are specified, however, but they are set 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.

IDCIO02

Procedure: BUILDACB

3 BUILDACB builds an ACB control block by issuing the ACB macro. The ACB macro generates IN, SEQ, ADDR for the MACRF field. These attributes are overriden 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

In DOS, the CATALOG OPEN option is never specified since catalogs are opened as described in step 5, Diagram 6.1.1.

BUILDACB requests address processing if the data set organization (indexed or non-indexed) is not known. If the type of processing is set in the OPNAGL, BUILDACB uses it. The VSAM open routine will fill in the correct organization, if the specified organization is wrong. 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. 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 IOCRCVRA='1', BUILDACB specifies the CRA=UCRA option for opening a catalog recovery area. If a VOLID or SYSNO is passed as input, (i.e., not DNAME) SYSNO is set in the ACB. VOLID is translated to SYSNO by IKQASNMT.

Also, 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. The address and length of the ACB are put in IOCCBA and IOCCBL, respectively. If OPNMODRC in the OPNAGL is 1, BUILDACB puts the address of the ACB in IOCCBP.

If OPNTYPXM is on, the request is from EXPORT(RA)/IMPORT(RA), and the number of data buffers in the ACB (ACBBUFND) is changed from 2 to 5.

IDCIO02

Procedure: BUILDDBK

4 A nonVSAM data set cannot be opened as a catalog or opened for update. If either of these two conditions exist, BUILDDBK does not build control blocks for the data set. BUILDDBK builds a compare word, COMPWORD with data set organization, open options and record format. It saves the blocksize, record size, and the length of the required I/O areas. The information is in the OPNAGL, IOCSTR, and IOCSEX. The Access Method Module uses the I/O areas. The length of the I/O area is the blocksize plus 8.

IDCIO02

Procedure: BUILDDBK

5 BUILDDBK compares the device type specified in the OPNAGL against the table of allowable devices, DEVTABLE. When a match is found, the track length, constants used to determine the number of fixed length blocks per track, and the device code defined inthe DTF are saved. If a device type is not specified in the OPNAGL, '2314bbbb' is used as a default. The data set is not opened and an error message is written if the following conditions are found:

- Blocksize in OPNAGL is less than 1.
- Record format is fixed and blocksize is not a multiple of recordsize.
- A non-supported device is specified.

IDCIO02

Procedure: BUILDDBK

6 BUILDDBK compares COMPWORD against a table of allowable data set characteristics and corresponding load module names, DOSACC. When a match is found, the length of the load module is used to obtain storage for the load module with a UGSPACE macro. BUILDDBK loads the module with a LOAD macro that puts it in the storage just obtained. The load modules are named IDCDI*xx* where *xx* is 01 through 15 and contain one or two DTFs along with the Access Method Modules needed to processs the data set.

IDCIO02

Procedures: BUILDDBK, PRINTMSG

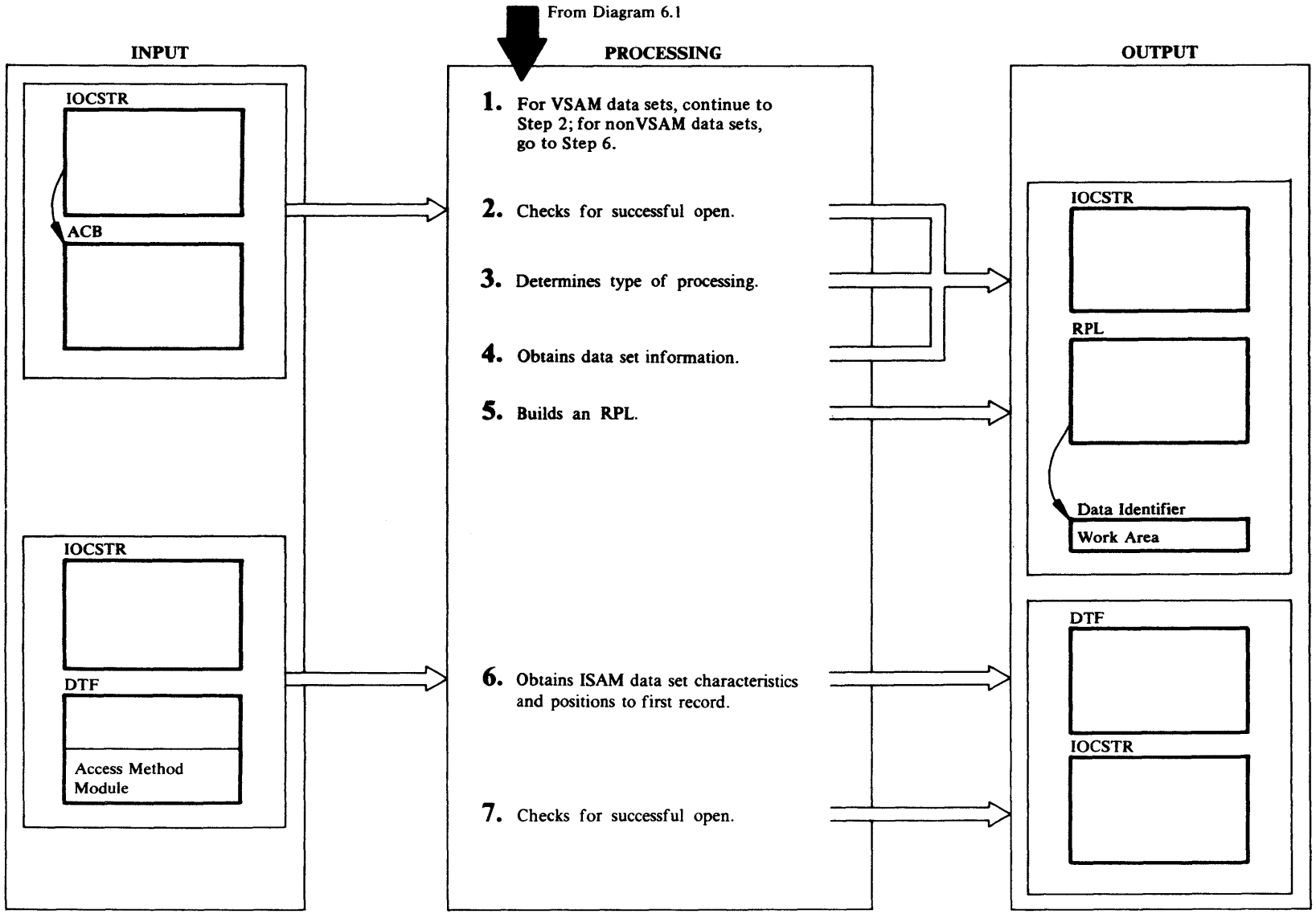
7 BUILDDBK issues a UGPOOL macro to obtain storage for the I/O areas. The Access Method Module uses the I/O areas as buffers. BUILDDBK puts the address of the storage in IOCWKA. If BUILDDBK finds no match in DOSACC or cannot obtain storage, the data set is not opened and PRINTMSG writes a message. If BUILDDBK cannot obtain storage for the load module, it issues a UABORT macro.

IDCIO02

Procedure: BUILDDBK

8 BUILDDBK updates the DTF with data set characteristics from the OPNAGL. Data set characteristics are record format, record size, blocksize, and device type. BUILDDBK updates the CCWs with the length of the data to get or put and the address of an I/O area.

Diagram 6.1.3. UOPEN Macro – Check Open



Extended Description for Diagram 6.1.3

IDCIO02

Procedure: OPENRTN

1 For VSAM data sets continue to step 2; for nonVSAM data sets go to step 6.

IDCIO02

Procedure: OPENRTN

2 OPENRTN checks the ACBOPEN flag 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.

IDCIO02

Procedures: 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 open object is not a path, the IOCSTR does not specify control interval or address processing, the type of processing is determined by checking the index portion of the file. 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. 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

IDCIO02

Procedures: OPENRTN, PRINTMSG

4 OPENRTN obtains the ACB error code, logical record length or control interval, high-used RBA, key length, and relative key position. If the data set did not open, only the error code, not the data, is obtained, and PRINTMSG writes a message. If the data set opened successfully, OPENRTN moves the ACB information to the IOCSTR.

IDCIO02

Procedures: BUILDRPL, PRINTMSG, OPENRTN

5 For any VSAM data set that is open, BUILDRPL builds a request parameter list (RPL) by issuing the RPL macro. Input work areas are required if the data set is opened for input or update processing. BUILDRPL issues a UGPOOL macro with the file 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 his 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'). This 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 the RPL with the address of the ACB, options, work area address, and maximum length of a data record. 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.

The RPL macro generates KEY, SEQ, NUP for the OPTCD field. These attributes are overridden with information indicated in IOCSTR/IOCSEX 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 IOCRPL. If ACBSTRNO is greater than 1, the first RPL is copied to each additional RPL area.

IDCIO02

Procedures: CKNONOP, PRINTMSG

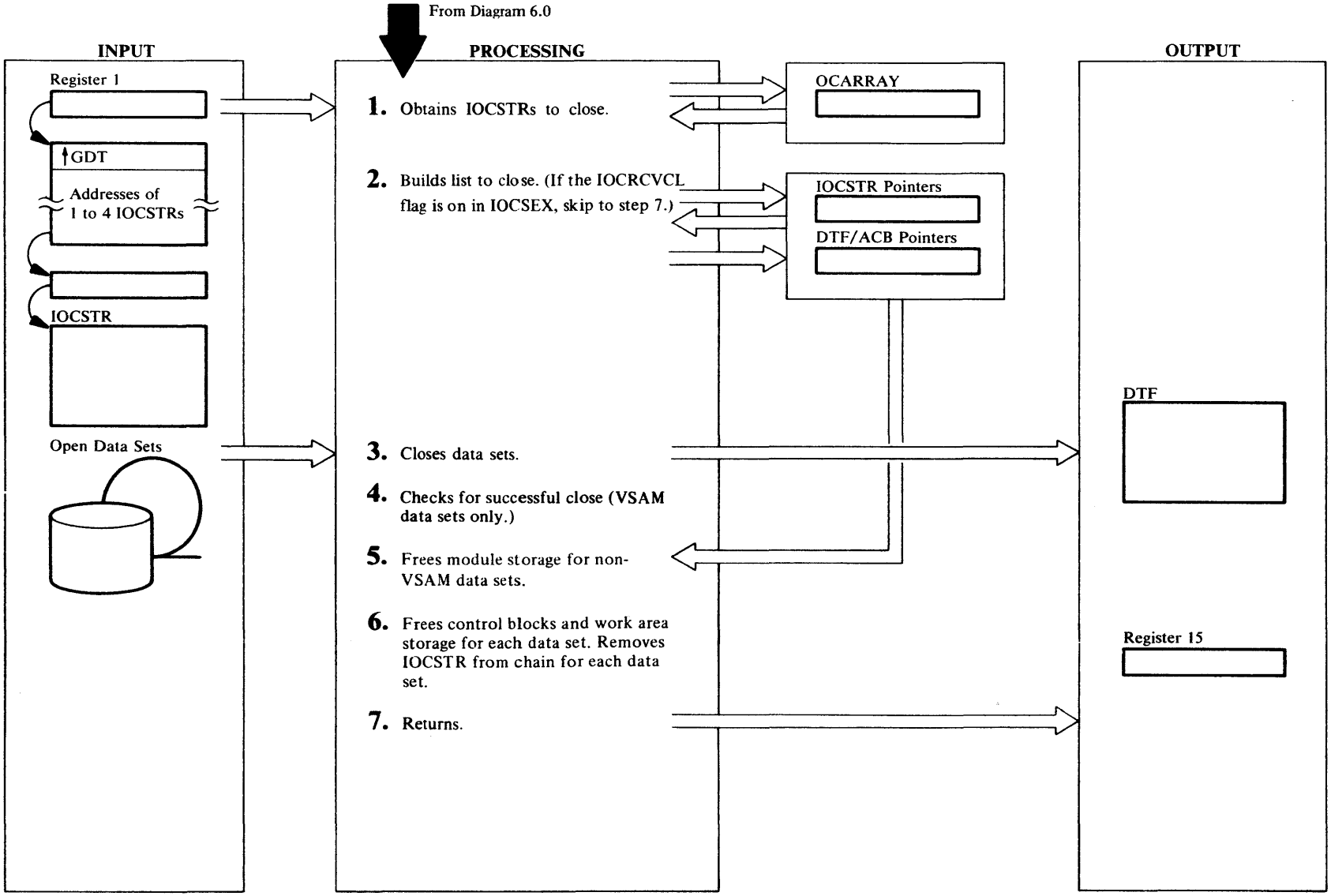
6 For ISAM data sets, CKNONOP obtains the true file block length, key length and relative key position from the DTF after the file is open. If the true block length is greater than the block length in the OPNAGL, PRINTMSG writes an error message, and CKNONOP turns off the open flag in IOCSTR. This is an error condition because ISAM open routines build their own CCW with the real data set characteristics obtained from the DSCB. If the I/O area for the data set is not large enough for a physical block, the block will overlay storage not belonging to the I/O Adapter. If the true block length is equal or less than the value in the DTF, CKNONOP puts the values from the DTF in the IOCSTR. CKNONOP issues a SETL macro to position to the first record in the data set.

IDCIO02

Procedure: CKNONOP

7 CKNONOP checks the DTF open flags for sequential data sets. There are no open flags for ISAM or device independent data sets like SYSIPT and SYSLST. If the open flags are set for a sequential data set or tape data set, CKNONOP sets flags in the IOCSTR and IOCSEX. CKNONOP always sets open flags for ISAM and device independent data sets. If the DTF open flag is not set for a sequential data set, PRINTMSG writes an error message, and CKNONOP sets an error return code.

Diagram 6.2. UCLOSE Macro



Extended Description for Diagram 6.2

IDCIO01

Procedure: IDCIOCL

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. IDCIOCL sets the type of operation to "Close" in OCATYP in OCARRAY.

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 SYSIPT and SYSLST. If the data set is SYSIPT or SYSLST, CLOSERTN does not close the data sets because they are needed until processor termination.

If a UCLOSE macro is issued and the IOCRCVCL bit is on in IOCSEX, the work area pointed to by IOCWKA is freed via UFSPACE. Next, a work area whose size is specified in IOCTRN is obtained via UGPOOL and the address is returned in IOCWKA. Control then passes to step 7 (a data set close is not done when the IOCRCVCL bit is on). This allows reallocation of the record work area after the file is opened. If IOCINFCT='1', indicating a close of a VSAM catalog, CLOSERTN merely frees up the control blocks associated with this catalog that were obtained by I/O Adapter. The issuer of the UCLOSE macro is given an RCOK return code. For any other nonzero IOCSTR, CLOSERTN saves the address. And, if the DTF or ACB is opened, CLOSERTN saves the address of the control block in preparation for closing. If the data set is not open, IOCFLGOP=0, CLOSERTN makes a check to determine if it is externally controlled. If it is externally controlled, CLOSERTN passes arguments to the external routine. CLOSERTN continues the above checking until:

- IDCIO01 specifies CLOSE ALL in OCARRAY and CLOSERTN has checked all IOCSTR addresses in OCARRAY. This happens during I/O termination.
- IDCIO01 does not specify CLOSE ALL in OCARRAY and CLOSERTN has checked all IOCSTR addresses in OCARRAY.

IDCIO02

Procedure: CLOSERTN

3 For up to four open DTFs or ACBs, CLOSERTN issues a CLOSE macro for each open DTF or ACB. The return code from the CLOSE macro is saved. If an abend occurs, no exits are taken; CLOSE abends.

For CRAs, CLOSERTN unassigns the logical unit number if IOCSYSNO is equal to the value obtained from the ASSGN macro by UOPEN.

IDCIO02

Procedures: CLOSERTN, PRINTMSG

4 For VSAM data sets, CLOSERTN checks the ACB error code. If the ACB error code is nonzero, PRINTMSG writes a message. No tests are made for nonVSAM data sets or user catalogs.

IDCIO02

Procedure: ENVFREE

5 For nonVSAM data sets, ENVFREE issues a FREEVIS macro to release the storage used for the IDCDI*xx* module where *xx* is from 01 to 15. For VSAM data sets the storage for the ACB, RPL, and exit list is freed in step 6 along with the IOCSTR and all other storage having the same IOCSID.

IDCIO02

Procedure: CLOSERTN

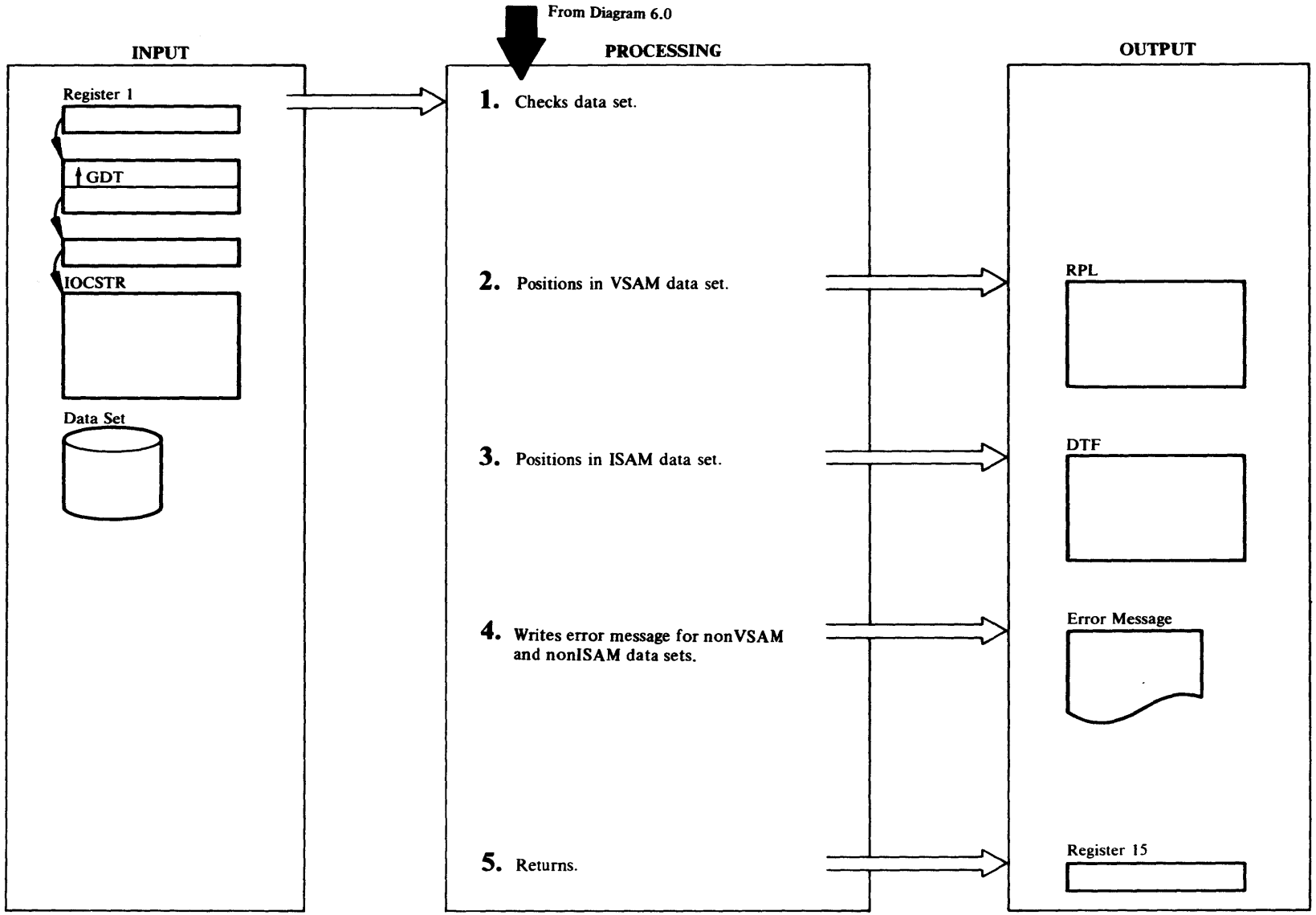
6 CLOSERTN saves the address of the IOCSTR that was closed and the address of the next IOCSTR in the chain after the IOSTR for the closed data set. CLOSERTN issues a UFPOOL to free all storage obtained for the data set that is closed. CLOSERTN passes the IOCSID field to UFPOOL which identifies all storage obtained for the data set. CLOSERTN seaches 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.

IDCIO01

Procedure: IDCIOCL

7 IDCIOCL puts a return code in register 15 and returns control to the module that issued the UCLOSE.

Diagram 6.3. UPOSIT Macro



Extended Description for Diagram 6.3

IDCIO03

Procedure: IDCIO03

1 If the IOCSTR address is zero or the data set is not open (IOCMSGOP=0), IDCIO03 issues a UABORT macro. If the data set is open for processing (IOCMSGOP=1), and the data set is externally controlled (IOCFLFEX=1), IDCIO03 returns control, with a return code of zero, to the module that issued the UPOSIT. No provision is made for positioning in externally controlled data sets.

IDCIO03

Procedures: PTAMDS, PRINTMSG

2 For VSAM data sets, PTAMDS inserts 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 the RPLARG field of the RPL. If the data set is RRDS (IOCMACRR='1'), the RPLARG field 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 RPLARG field 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 the RPLARG field. If the length of the key 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 inactivates the end-of-data routine in the EXLIST 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 re-activates 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 written. PRINTMSG prints the error message. If the return code from the POINT macro is 8, a logic error has occurred and PTAMDS checks the logical error. If the results indicate that no record was found or repositioning beyond end-of-file, PTAMDS sets a return code of "no record found." For all other logic errors, PRINTMSG writes a message containing the return code unless the suppress message flag, IOCMSGSM has been set by the caller.

IDCIO03

Procedure: PTISDS

3 For an ISAM data set, PTISDS does not position the record if the length of the key supplied is greater than the key length for the data set. For valid key lengths, PTISDS does the positioning. PTISDS expands the key to 256 bytes by padding on the right with binary zeros. PTISDS issues an ESETL macro because a SETL was issued when the data set was opened. PTISDS issues a SETL macro to position to the record with the key or next higher key. If the postioning is beyond the end of the data set, the SETL routine sets a flag in the DTF. If this flag is on, PTISDS returns a code of "no record found." If the flag is not on, positioning was successful and PTISDS returns a code of zero.

IDCIO03

Procedures: PRINTMSG, IDCIO03

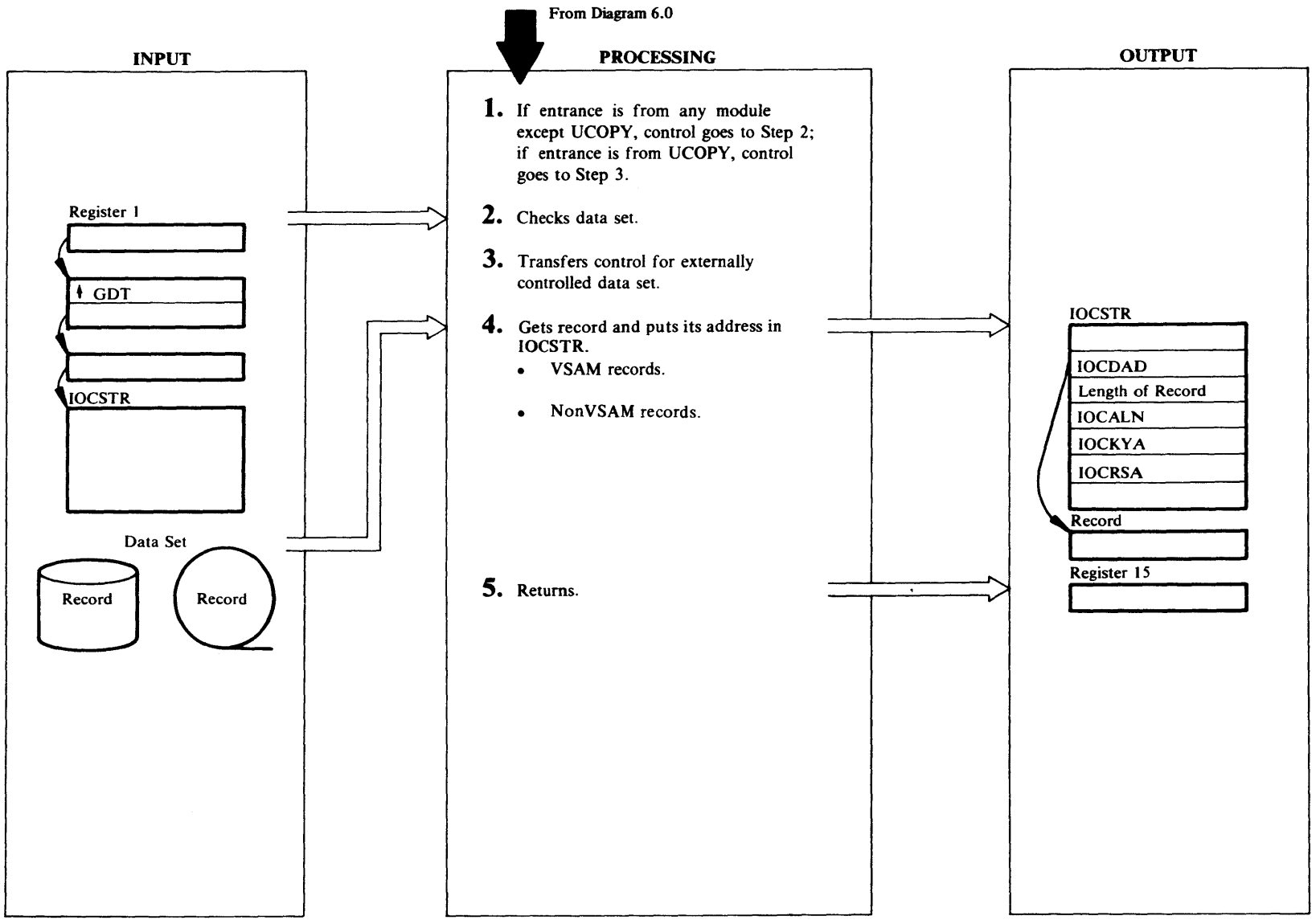
4 If the data set is nonVSAM and not ISAM, PRINTMSG writes an error message. If an error is detected, IDCIO03 turns off the open for processing idiciator, IOCMSGOP, so that no more I/O operations except close are permitted against the file.

IDCIO01

Procedure: IDCIOPO

5 IDCIOPO puts a return code in register 15 and returns control to the module that issued the UPOSIT.

Diagram 6.4. UGET Macro



Extended Description for Diagram 6.4

1 If entrance is from any module except UCOPY, control goes to step 2. If entrance is from UCOPY, control goes to step 3.

IDCIO01

Procedure: IDCIOGT

2 If the address of the IOCSTR is zero or the file is not open for processing, (IOCMSGOP=0), IDCIOGT issues a UABORT macro to terminate processing. If end-of-file has previously been encountered, (IOCFLGEF=1), on an input data set, IDCIOGT returns control to the module that issued the UGET. This check allows more than one module to issues UGETs on the same data set and both modules will get end-of-file indications by a return code.

IDCIO01

Procedure: GETEXT

3 If the data set is externally controlled, GETEXT passes an argument list to the external routine so the external routine can perform the I/O operation. GETEXT tests the return code from the external 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, GETEXT sets a return code of 4. IDCIOGT returns control to the module that issued the UGET.

IDCIO01

Procedures: GETVSAM, CHANGE, VSAMERR, PRINTMSG, GETNONVS, IROSEOD, IRSISYN, IRAMEOD

4 For VSAM data sets continue with 4.a, for nonVSAM data sets go to 4.b.

a. If any of the IOCSTR change processing flags are set, indicating a change in processing modes, the CHANGE procedure makes the appropriate change 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
IOCCHPSK	SKP
IOCCHPKS	KEY
IOCCHPCR	ADR
IOCCHPBK	CNV
IOCCHPKG	KGE
IOCCHPKE	KEQ
IOCCHPUP	UPD
IOCCHPNU	NUP

The CHANGE procedure 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. If IOCMACCI is set on (indicating export CIMODE processing), and the input data set is a KSDS/AIX, register 0 is set to X'30' before the GET is issued.

GETVSAM issues a GET macro in the move mode, specifing the address of the RPL built when the data set was opened. If end-of-file is encountered, the VSAM EODAD exit routine, IRAMEOD, 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 is nonzero, an error code has been placed 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 from the RPL to the IOCSTR. 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 nonzero, VSAMERR obtains the error code from the RPL and PRINTMSG writes the message. The call to VSAMERR by UGET to print logical error messages is bypassed if the suppress messages flag, IOCMSGSM, has been set by the UGET caller.

b. For nonVSAM data sets, GETNONVS issues a GET specifying the DTF address. For spanned records the address of the work area for the data set which was obtained when the data set was opened, is given the the GET macro. The GET routine puts the complete record in the work area. GETNONVS gets the length of variable length records from the Record Descriptor Word (RDW). 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 nonspanned records register 8 has been specified as the IOREG in the DTF. For undefined records the length is found in the RECLEN register defined in the DTF. The GET routine puts the address of the record in register 8.

For ISAM data sets with fixed unblocked records, the key is returned preceeding the data; however, register 8 has the address of the data. GETNONVS subtracts the key length from the data address to get the address of the key. If an error or end-of-file occurs attempting an ISAM GET, the GET routine sets flags in the DTF. GETNONVS tests the flags. If end-of-file has occurred, GETNONVS sets a return code. If an error has occurred, PRINTMSG writes a message and GETNONVS sets a return code. If no errors or no end-of-file has occurred, GETNONVS assumes the GET is successful and the record address and record length are set in IOCDAD and IOCDLN, respectively. GETNONVS puts the address of the key in IOCKYA.

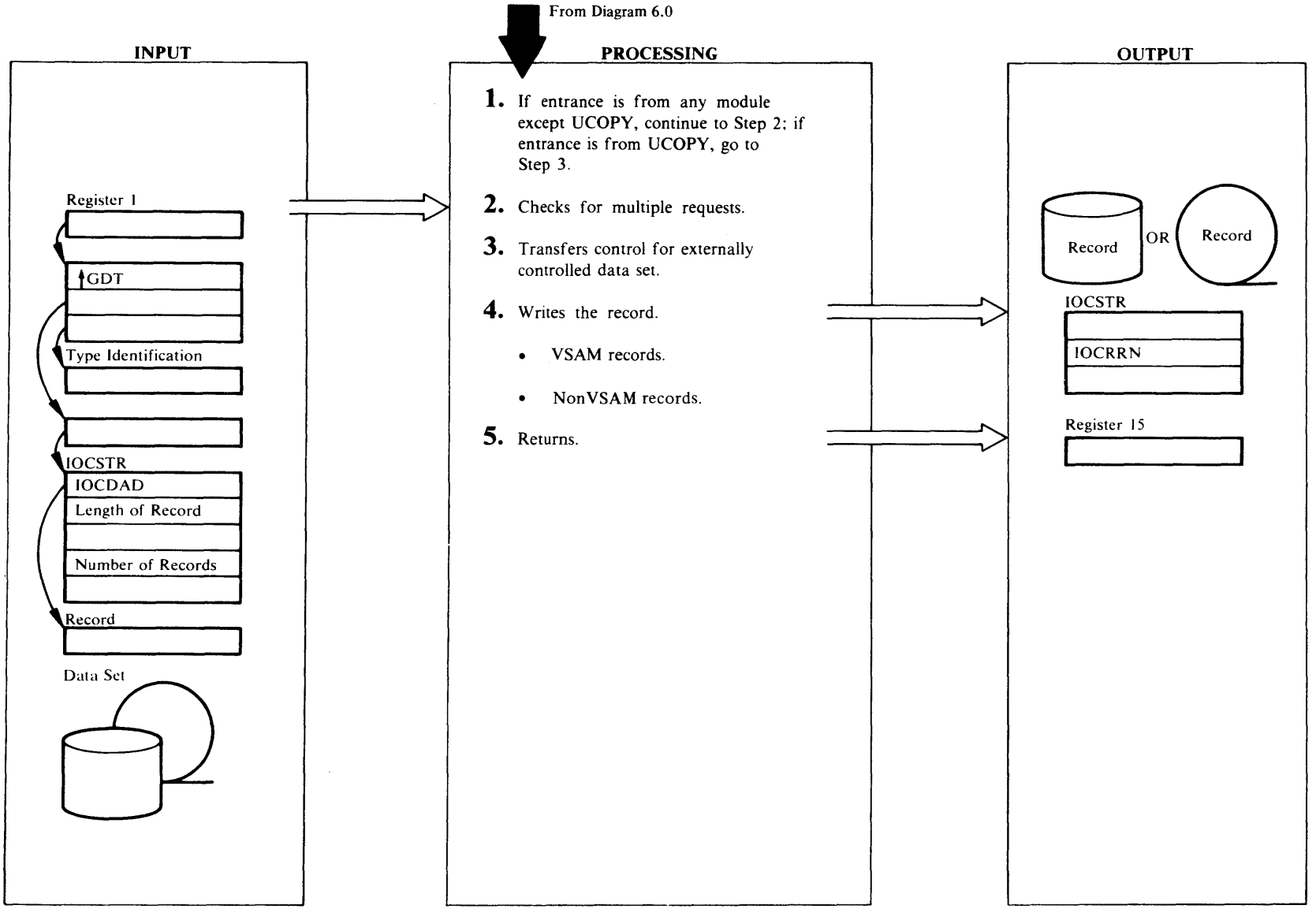
For non-ISAM data sets, if an error or end-of-file occurs, the EODAD exit routine, IROSEOD, or SYNAD exit routine, IRSISYN, gets control. If end-of-file occurs, IRSOEOD sets a return code. If an error has occurred, PRINTMSG writes a message and IRSISYN sets a return code. If no errors or no end-of-file has occurred, GETNONVS assumes the GET is successful and the record address and record length are set in IOCDAD and IOCDLN, respectively.

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. UPUT Macro



Extended Description for Diagram 6.5

1 If entrance is from any module except UCOPY, control goes to step 2. If entrance is from UCOPY, control goes to step 3.

IDCIO01

Procedure: IDCIOPT

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 nonzero 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 UPUT and the length of the record is assumed to be in IOCDLN. If IOCPNM is nonzero, one or more records are written with this UPUT. IOCDLN contains the total length of all the records, and each record is preceeded by a two byte length field for that record. IDCIOPT sets IOCPNM to one if it was initially zero. For multiple records, IDCIOPT puts the length of the first record in IOCDLN and IDCIOPT puts the address of the data for the first record in IOCDAD.

IDCIO01

Procedure: PUTEXT

3 If the data set is externally controlled, PUTEXT constructs an arguments list. PUTEXT gives control to the external routine addressed in IOCXDAD. If the return code from the external routine is zero, PUTEXT increments the number of successful UPUTs. 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.

IDCIO01

Procedures: PUTVSAM, CHANGE, VSAMERR, PRINTMSG, PUTNONVS, IRSOSYN, PUTREP

4 For VSAM data sets continue with 4.a, for nonVSAM data sets go to 4.b.

a. 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. In this case, a UGET for update must previously have been issued by the caller. If IOCMACEN is set by the UPUT caller, PUTVSAM issues the ENDREQ macro with a pointer to the RPL.

If any IOCSTR flag indicating a change in processing modes, has been set by the caller, CHANGE 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 nonzero, VSAMERR obtains 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 PRINTMSG writes a message. The call to VSAMERR by UPUT to print logical error messages is bypassed if the suppress messages flag, IOCMSGSM, has been set by the UPUT caller.

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, 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.

If the return code for an I/O error is greater than 4, VSAMERR turns off the open for processing flag (IOCMSGOP). PUTVSAM returns control to step 2 for the next record.

b. PUTNONVS checks the length of the record against the IOCTRN to be sure that the record can be written. If the length is too long, PRINTMSG writes an error message and control returns to step 2 for the next record. For the SYSLST data set, PUTNONVS compares the record length to the maximum and truncates the record if it is longer than the maximum. The record is processed according to the record format.

- For spanned records, PUTNONVS constructs a Record Descriptor Word (RDW) in the first four bytes of the work area. PUTNONVS moves the record to the work area making one spanned logical record. The address of the work area will be specified in the PUT macro.

  If the output IOCSEX indicates export of a catalog recovery area (IOCRCVM='1'), a 4-byte header must be prepended 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'.

- For variable blocked records, PUTNONVS checks to be sure the block will fit in the IO AREA being used as the buffer. If the block is too long, PUTNONVS issues the TRUNC macro to write the current buffer and to start processing in the other I/O area.

- For variable records, PUTNONVS constructs a RDW in the first four bytes of the area in the buffer and PUTNONVS moves the record following the RDW.

  PUTNONVS issues a PUT macro. The address of the next area is returned by the PUT macro—except for spanned records—and is saved. If the records are variable blocked, PUTNONVS saves the number of bytes remaining in the current area. If an I/O error is detected during the PUT

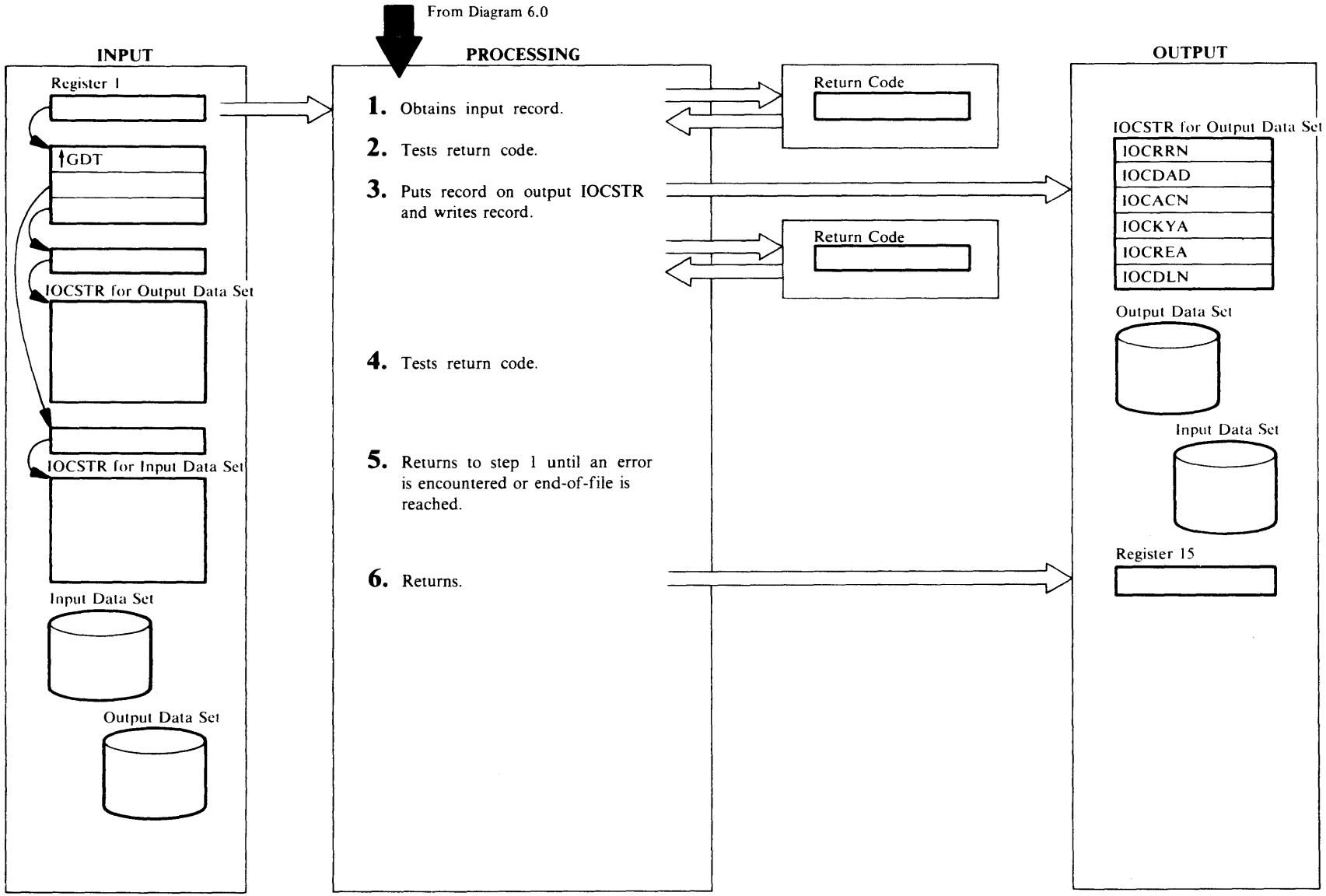
macro, IRSOSYN sets error data. PRINTMSG writes the message. IRSOSYN turns off the open for processing flag, IOCMSGOP. If there are no errors, PUTNONVS increments the count of successful UPUTs in IOCRRN. PUTNONVS can use device independent, magnetic tape, or sequential disk DTF processing. PUTNONVS returns control to step 2 for the next record.

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 macro.

Diagram 6.6. UCOPY Macro



Extended Description for Diagram 6.6

IDCIO01

Procedure: IDCIOCO

1 IDCIOCO obtains a record from the input data set by calling 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 just as though a UGET had been issued. If export CIMODE processing has been requested, the control interval is retrieved. For each input control interval that contains a segment of a spanned record, the segment is checked for a consistent level number; only a valid segment is written. If the segment is invalid, message IDC13291 is written and the remaining segments are ignored.

IDCIO01

Procedures: IDCIOCO, PRINTMSG

2 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, control goes to step 6. 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 4 or IOCMSGOP is off, IDCIOCO turns off IOCMSGOP and UCOPY quits.

IDCIO01

Procedure: IDCIOCO

3 • If the output IOCSTR does **not** indicate **export CIMODE** processing:
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.

• If the output IOCSTR does indicate **export CIMODE** processing:

For a spanned record, a GET is issued for each segment, and the spanned record is built in a work area. When all segments are retrieved, a PUT is issued for the record.

For a non-spanned record, the control interval is deblocked, and a PUT is issued for each record contained in it.

IDCIOCO writes the record by calling the same procedures used for the UPUT macro. IDCIOCO sets up the arguments to the procedures just as though a UPUT macro has been issued. The UPUT procedure returns control to this point in the UCOPY routine.

IDCIO01

Procedure: IDCIOCO

4 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.

IDCIO01

Procedures: PRINTMSG, IDCIOCO

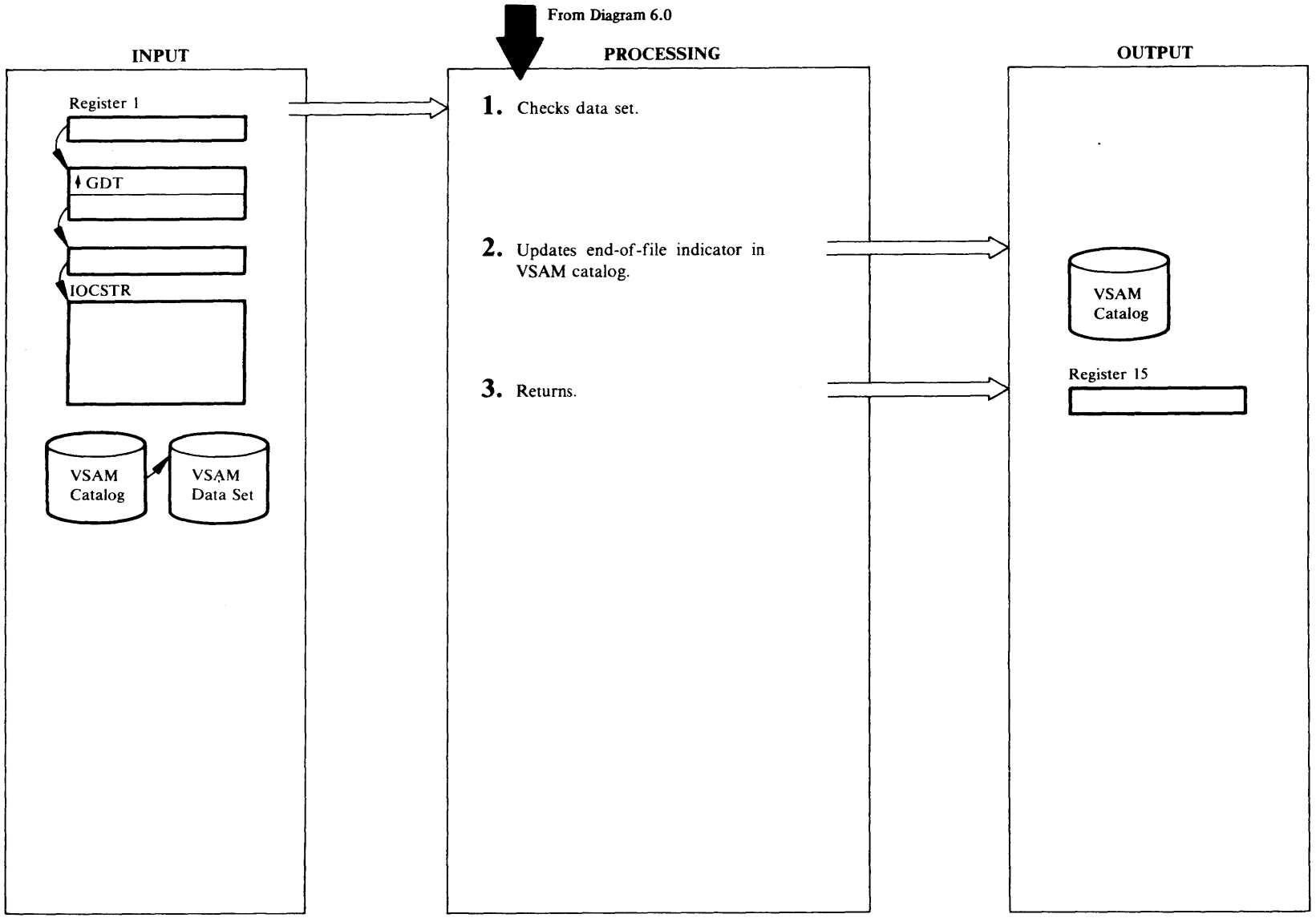
5 Control goes to step 1 for the next record. Processing continues if the number of errors is less than four, and IOCMSGOP is on. PRINTMSG writes a message if the message has been formatted. If the number of errors is 4, IDCIOCO truns off IOCMSGOP and UCOPY quits.

IDCIO01

Procedure: IDCIOCO

6 IDCIOCO puts a return code in register 15, and returns control to the module issuing the UCOPY.

Diagram 6.7. UVERIFY Macro



Extended Description for Diagram 6.7

IDCIO01

Procedure: IDCIOVY

1 The second argument is assumed to be a valid IOCSTR address. The UVERIFY does not continue if:

- The file is not VSAM.
- No RPL has been built for a VSAM file.
- The VSAM file is not open.

No error message is written for the last two conditions because message have been written at open.

IDCIO01

Procedure: IDCIOVY

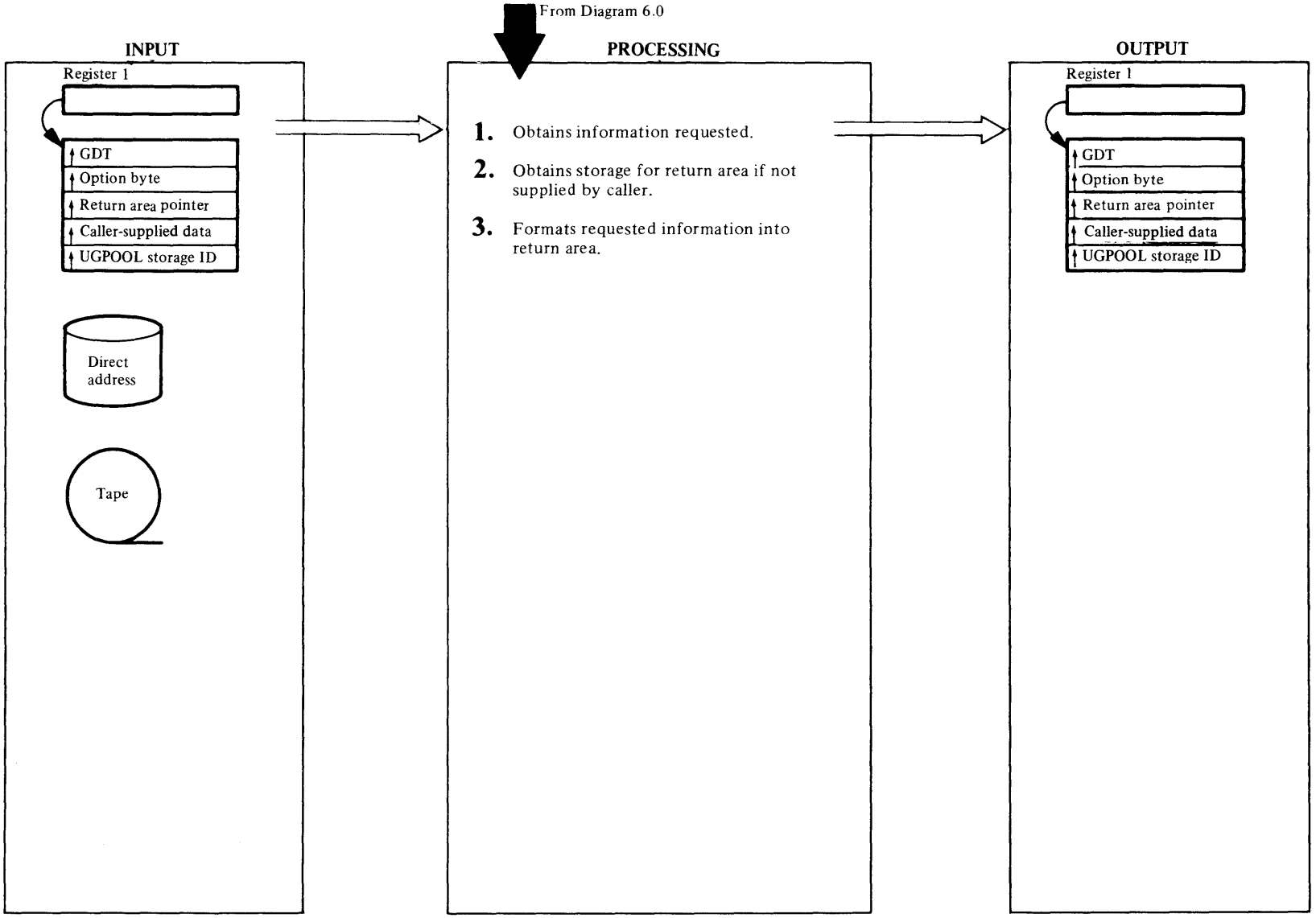
2 IDCIOVY issues a VERIFY macro.

IDCIO01

Procedures: VSAMERR, PRINTMSG, IDCIOVY

3 If the return code is not zero, VSAMERR 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, PRINTMSG writes an error message. If the error code returned in the RPL 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. UIOINFO Macro



Extended Description for Diagram 6.8

IDCIO03

Procedure: DSINFO

1 UIOINFO analyzes the option byte passed by the caller and determines what kind of information is required. Data set name, volume serial list and Logical Unit Blocks (LUB) require that UIOINFO obtain job control information. UIOINFO issues CDLOAD to load IKQVLAB, the VSAM read label cylinder module,and then gives control to IKQVLAB. The work area passed to IKQVLAB is that of the existing work area in IDCIO02's automatic storage. If the return code from IKQVLAB is nonzero, UIOINFO sets a return code and returns control to the calling procedure. If the return code from CDLOAD was nonzero, DSINFO issues a UABORT macro. If the return code is 12 (insufficient storage was available), DSINFO sets the UABORT code to 28; otherwise, DSINFO sets the UABORT code to 64.

If device type information is requested, UIOINFO issues a CDLOAD macro for IKQVDTPE and passes control to it providing a pointer to the label information that will be returned from IKQVLAB. Label information is not needed if the VOLID is already known. In that case, IKQVDTPE does a GETVCE using the VOLID. The reading of label information is needed only to find a VOLID. It is assumed that the volume is already assigned; if not, a job control error is returned.

If timestamp information is requested, UIOINFO issues an OVTOC macro to open the VTOC on the volume. It next issues a PVTOC macro with the read option to read the format-4 label of the VTOC. When processing is complete, a CVTOC macro is issued to close the VTOC.

IDCIO03

Procedure: DSINFO

2 All of the information that UIOINFO obtains in Step 1 is placed in IDCIO02'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 ID. If the caller passes a return area, UIOINFO determines if it is large enough (the length is contained in bytes 0 and 1 of the return area). If not, UIOINFO places the total size needed in bytes 2 and 3 of the return area, sets a return code, and passes control back to the caller.

If the caller has passed a UGPOOL ID, UIOINFO issues a UGPOOL macro for the required amount of storage with the storage identification passed by the caller. In this case the caller is responsible for freeing this storage.

IDCIO03

Procedure: DSINFO

3 UIOINFO formats the requested information into the return area and passes control back to the caller.

Text Processor Visual Table of Contents

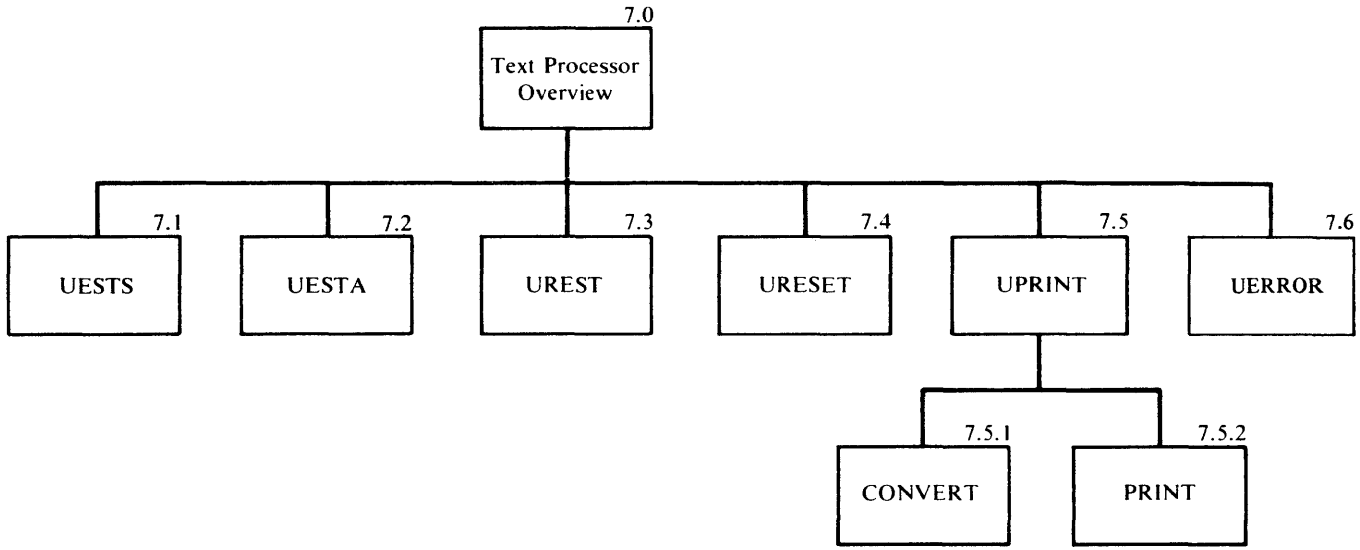
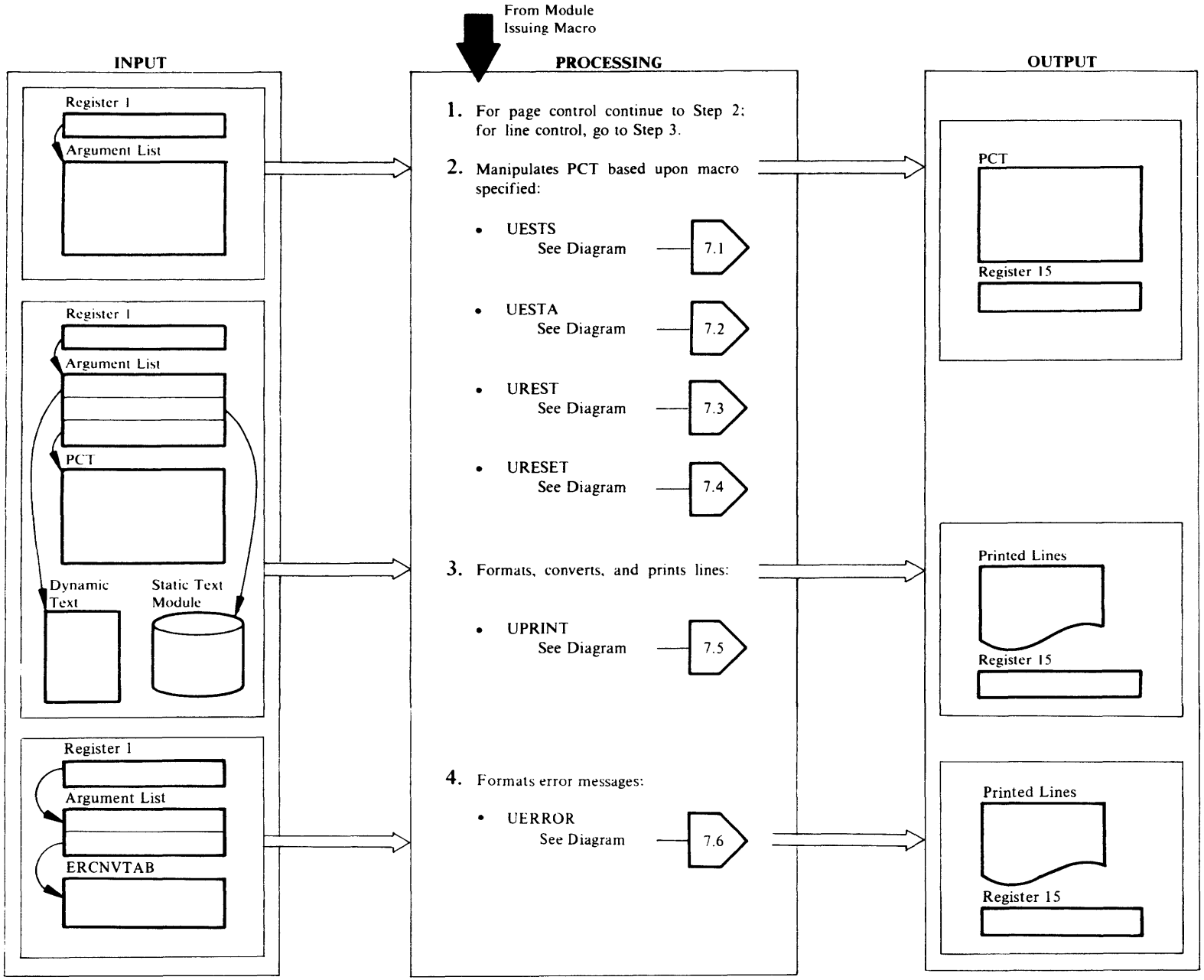


Diagram 7.0. Text Processor Overview



Extended Description for Diagram 7.0

IDCTP01

Procedure: IDCTP01

1 For page control continue with step 2; for line control go to step 3.

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 line
Data line

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 sections 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 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, 3, or vertical spaces
Translate table for print chain	standard tables	

3 The UPRINT macro formats data within a line, converts data to a printable form, and prints the line or lines. 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 is unchanging data and format structures that reside in a module referred to as 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

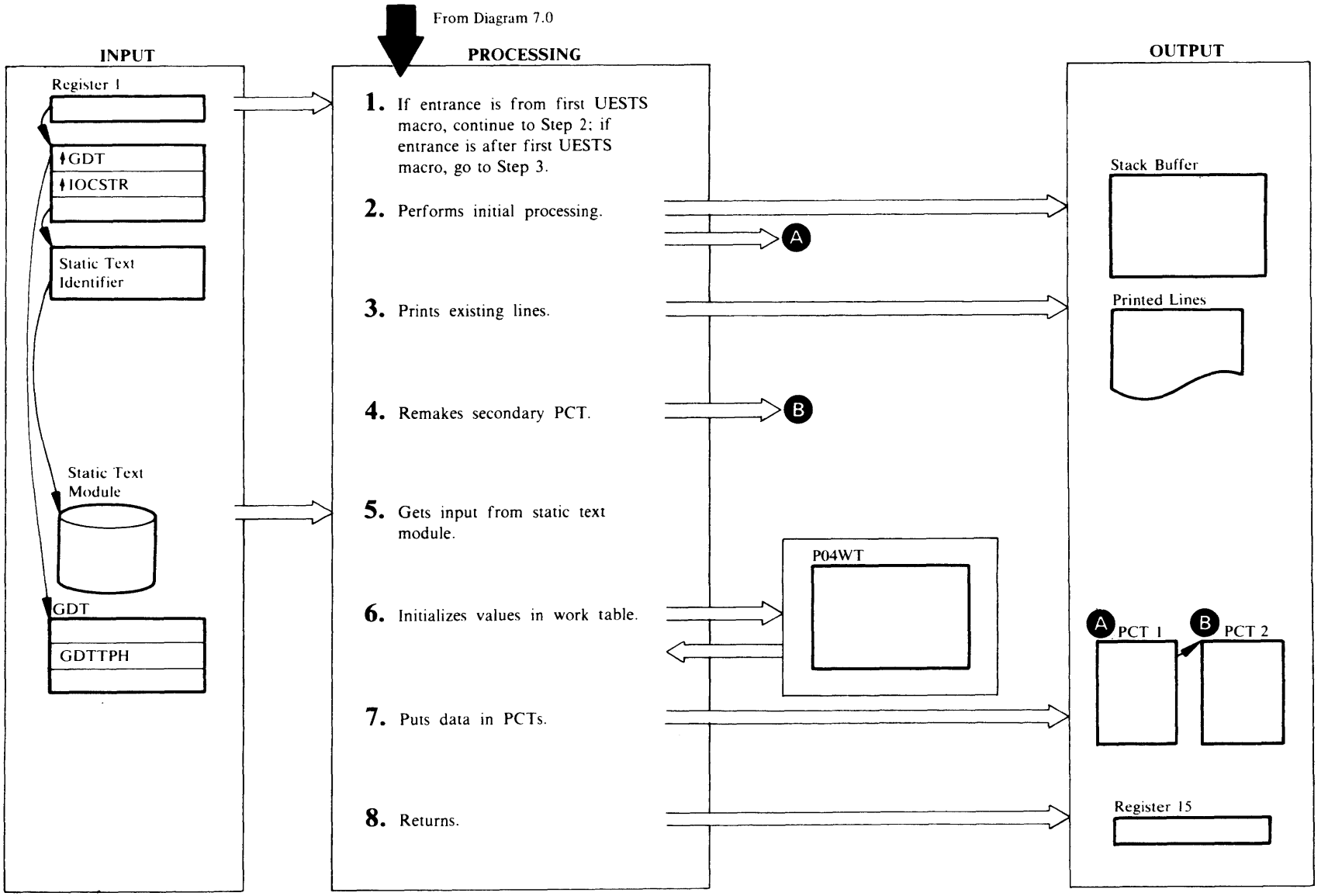
The line is printed at a number of vertical spaces counted from the last printed line. If there is not enough room on the page to print the line, the page is ejected, and the line is printed after the title section and header lines on the following page.

Eject

The line is printed after the title section and header lines on the following page.

4 The UERROR macro formulates prose messages for the return and reason codes caused by catalog errors. It instigates multilevel message requests to the UPRINT macro. Formatting and printing of the multilevel message is handled by the UPRINT macro.

Diagram 7.1. UESTS Macro



Extended Description for Diagram 7.1

IDCTP04, IDCTP01

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 passes control to INITPCT which 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.

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 controls change. STACKFL writes the stack buffer with a UPUT macro.

IDCTP04

Procedure: INITPCT

4 Prior to making any changes INITPCT gives control to STACKFL to flush the stack buffer. 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.

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 appropiate 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.

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.

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.

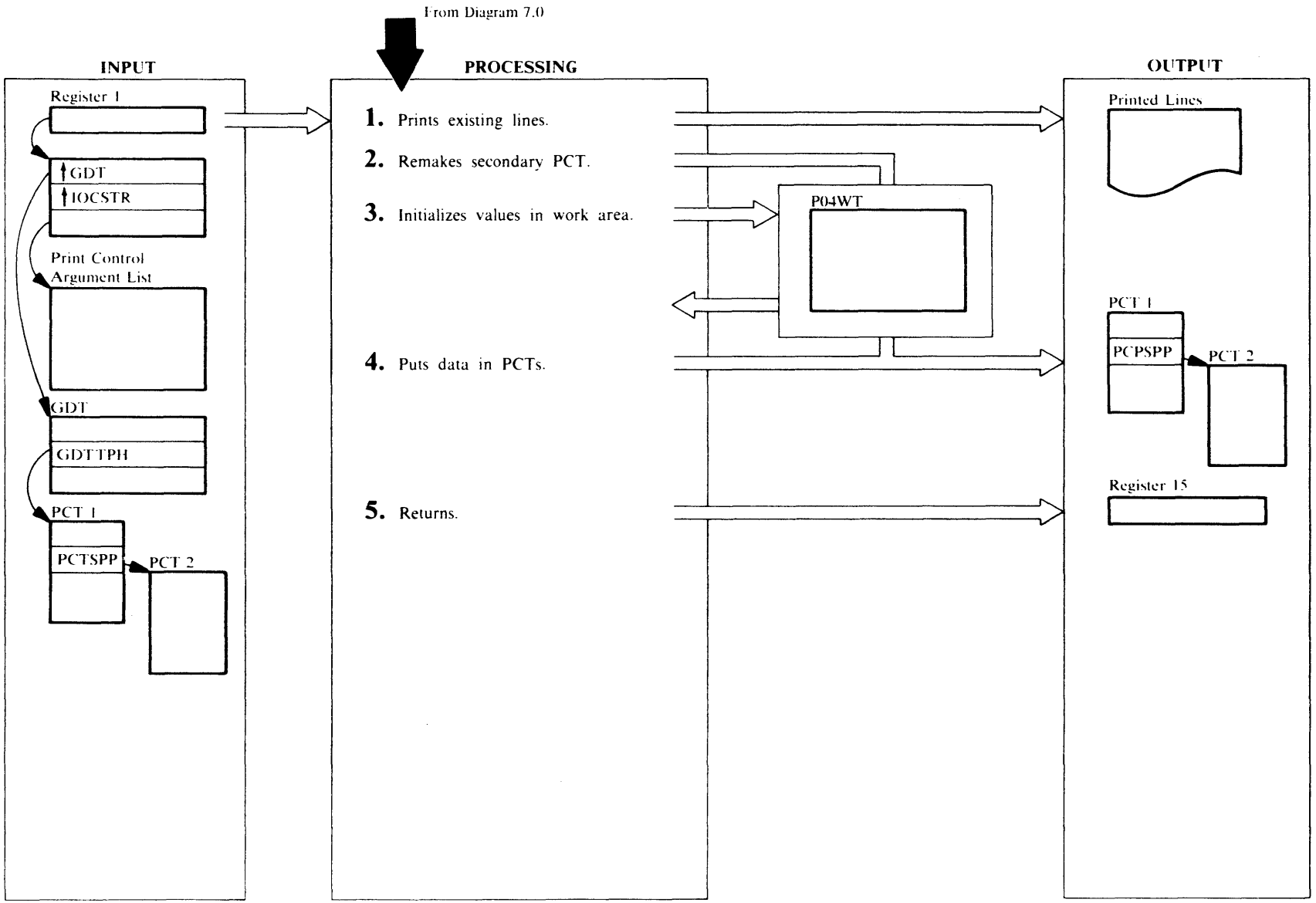
   For initializing either the primary PCT or the secondary PCT, PCTSETUP verifies the input data and puts it into the appropiate PCT.

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

IDCTP04

Procedures: ESTACONT, INITPCT

1 ESTACONT determines if a primary PCT exists. ESTACONT invokes INITPCT to get storage for the PCT. ESTACONT then invokes P04SETUP to build the work table; ESTACONT then invokes PCTSETUP which initializes the PCT. 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.

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.

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.

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

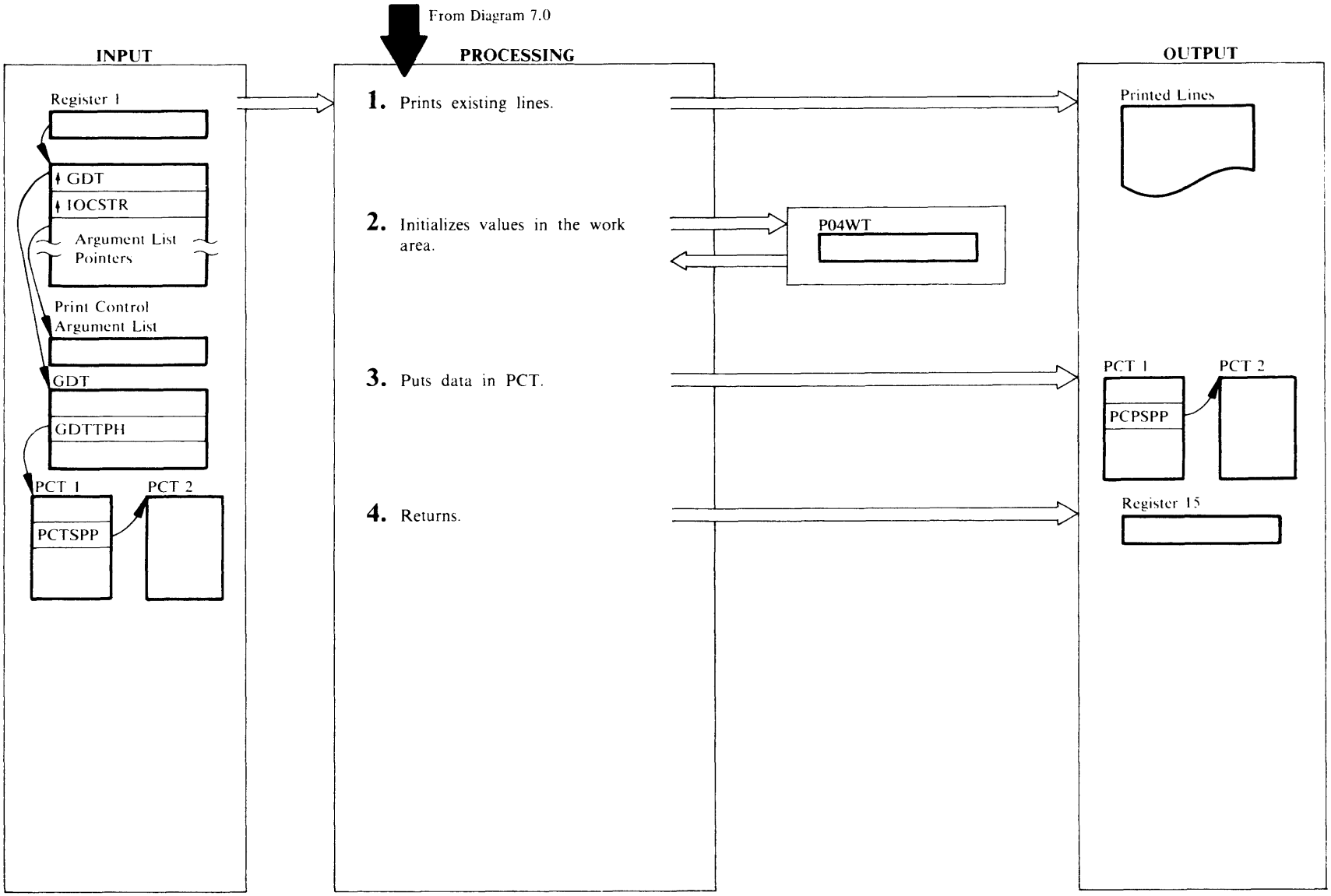
For initializing either the primary PCT or the secondary PCT, PCTSETUP verifies the data in the work table and puts it into the appropiate PCT.

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

IDCTP04

Procedures: RESTCONT, 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.

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.

IDCTP04

Procedures: 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

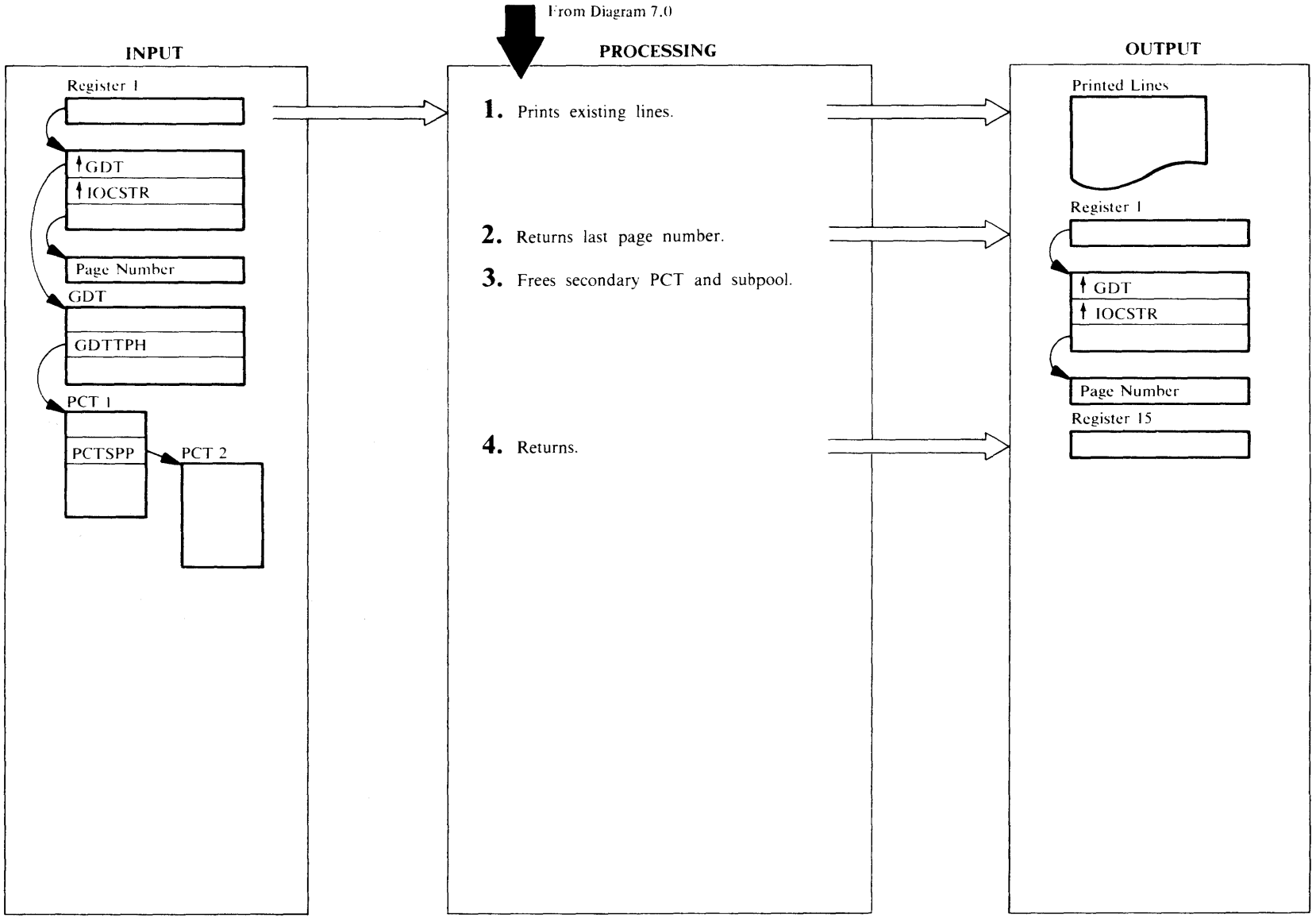
A value of zero in any of the parameter lists causes the item to be reset to the Access Method Services default. RESTCONT evaluates the input parameter list. If the secondary PCT exists, PCTSETUP modifies it. Otherwise, PCTSETUP modifies the primary PCT.

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

IDCTP04

Procedures: 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.

IECTP04

Procedure: RESETCON

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.

IDCTP04

Procedure: RESETCON

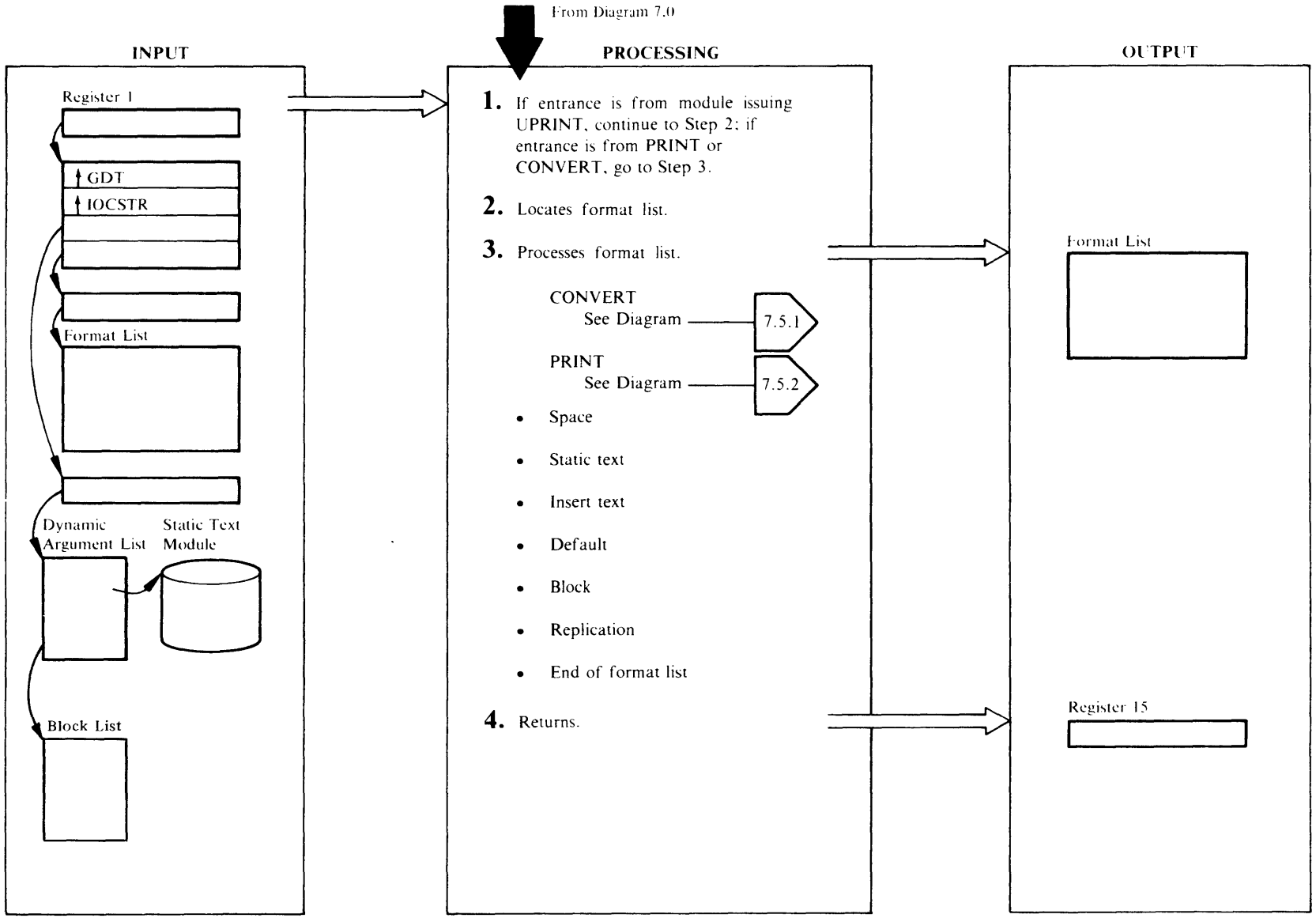
3 Before the secondary PCT is deleted, RESETCON copies some data into the primary PCT. One UFPOOL macro releases the secondary PCT, subtitle lines, footing lines, and any static text entries addressed from the secondary PCT in PCTSQP because everything was obtained with subpool identification 'TP01'. 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.

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 Macro



Extended Description for Diagram 7.5

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.

IDCTP01, IDCTP05

Procedures: IDCTPPR, IDCTP05

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, IDCTPRR uses that FMTLIST in the static text entry as input to UPRINT. If a match is not found, IDCTPRR 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 "Text Structure" in the chapter "Diagnostic Aids" for a discussion of static text entries.

IDCTP01

Procedures: IDCTPPR, SPACE, STATIC, INSERT, BLOCK, REDO

3 IDCTPPR takes action on the format list substructures in FMTLIST depending upon the structure type. The line buffer is a work area where each line is formatted. IDCTPPR processes substructures in order of their appearance in the FMTLIST. If the high order bit in FMTFLGS is on, this substructure is the last in the FMTLIST. If there is formatted data in the line buffer, IDCTPPR calls LINEPRT to write the line. (See diagram 7.5.2.) IDCTPPR sets a return code in register 15, and control returns to the module that issued the UPRINT macro.

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 2 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 IDCTPPR. (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 CVPSTRM: 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 2 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 2 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 IDCTPPR. (See diagram 7.5.1.)

- Block format

  BLOCK obtains input information from DARGDBP and DARGILP. If the DARGBPL flag is set on (more than one block is to be used for input data), then BLOCK adds the offset count in BLKLRIO to the address in BLKLPTR to get the address of the input data. BLOCK uses the input length specified in BLKLILP. The block number in the format list, FMTBLKNO, is used as an index into the BLKLIST.

  If the DARGBPL flag is not set, then 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 IDCTPPR. (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 2 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 prints a line because the output positions for each field are unchanging. (See diagram 7.5.2.) REDO saves the value in FMTRIO and adds to each address of block data in the substructures being repeated.

  If the DARGBPL flag is set on (more than one block is to be used for input data), then REDO calculates the redo input offset from the BLKLRIO field of the block list. The block number in the format list, FMTBLKNO, is used as an index into the BLKLIST.

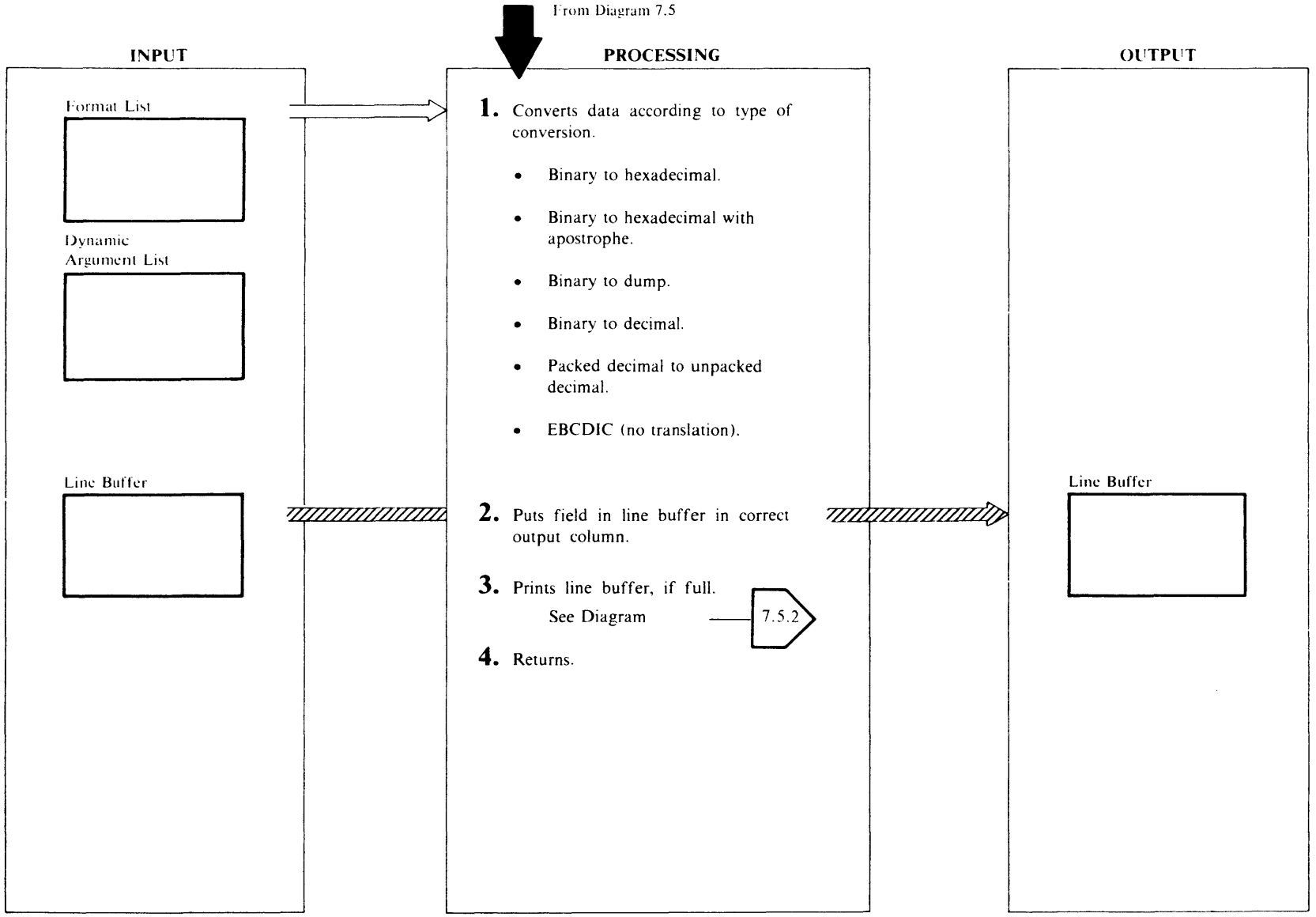
If the DARGBPL flag is not set, the redo input offset is obtained from the format list redo input offset field, FMTRIO.

IDCTP01

Procedure: IDCTPPR

4 IDCTPPR puts a return code in register 15 and returns control to module that issued the UPRINT macro.

Diagram 7.5.1. UPRINT Macro – CONVERT



Extended Description for Diagram 7.5.1

IDCTP01

Procedures: CONVERT, BHCONV, BHDCONV, BDCONV, PUPCONV, EBCDIC

1 CONVERT checks the conversion type from FMTCNVT and converts the field accordingly. Output fields can overlap. When a line of conversion is finished, LINEPRT prints the line. (See diagram 7.5.2.)

Control returns to the caller in diagram 7.5. (See diagram 7.5.) Types of conversion:

Binary to hexadecimal

BHCONV converts bytes of binary data to their equivalent 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 printable hexadecimal. BHCONV prints two characters for each byte. The output is preceded by a 'X' and followed by 'a'. 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 (See diagram 7.5.2.) 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 align right 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 nonblank 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.

IDCTP01

Procedure: CONVERT, BHCONV, BHICONV, BDCONV, PUPCONV, EBCDIC

2 The conversion routines put the converted data 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.

IDCTP01

Procedures: CONVERT, BHCONV, BHDCONV, PUPCONV, EBCDIC

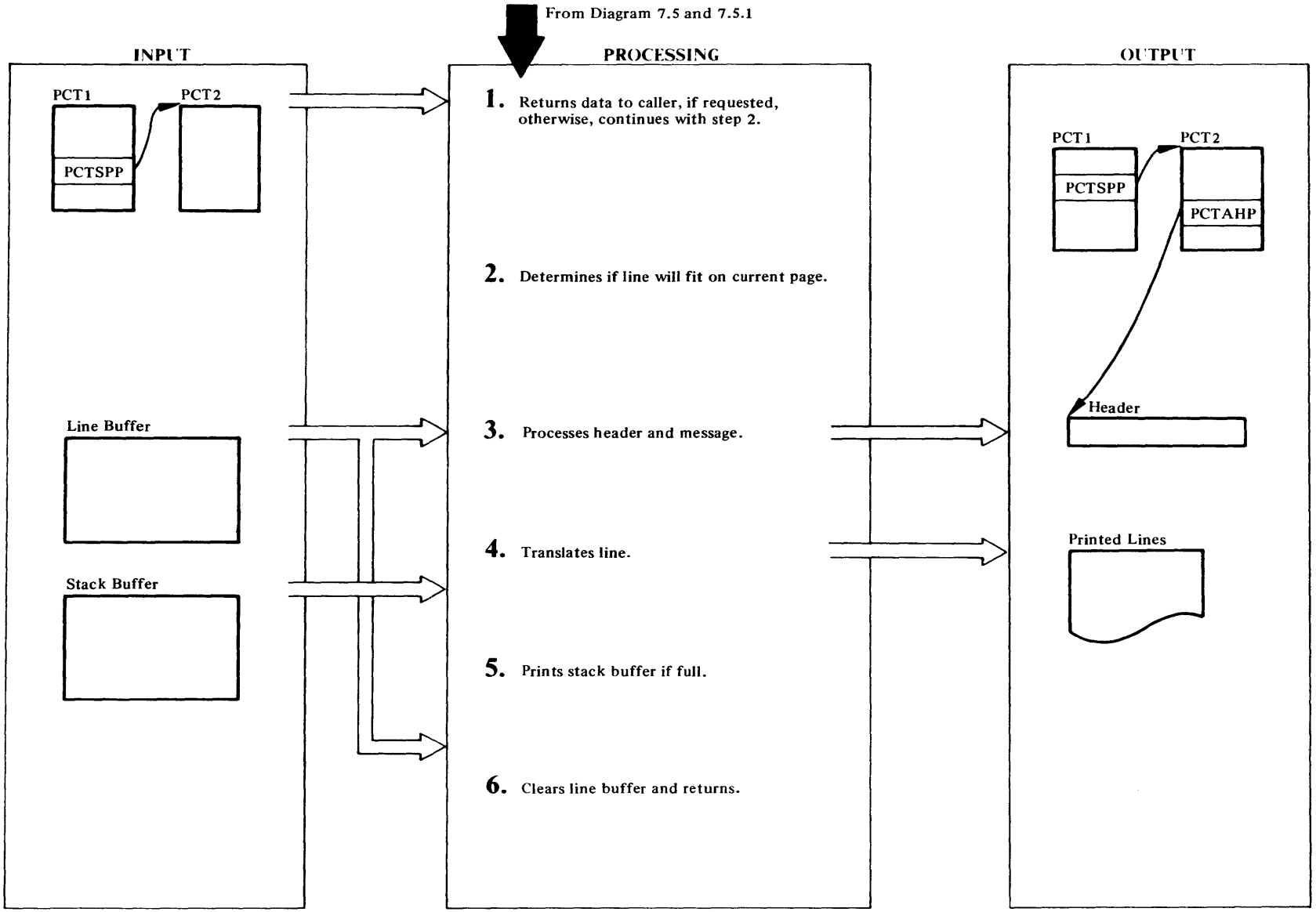
3 When the line buffer is full or a new line is to start, the conversion routines call LINEPRT to print the line. See Diagram 7.5.2.

IDCTP01

Procedures: 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.

Diagram 7.5.2. UPRINT Macro - PRINT



Extended Description for Diagram 7.5.2

IDCTP01

Procedures: LINEPRT, LINERET

1 LINEPRT tests the return area pointer in the argument list for zero. If it is not zero, procedure LINERET places the formatted line in the return area without checking for or setting page-related data such as carriage control, headings, etc. Only as many characters are returned as allowed by the return area length.

IDCTP01

Procedures: 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 alternate 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 the line number is within three lines of the line number where the line is to be written and the spacing character can handle spacing.

IDCTP01

Procedure: LINEPRT

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. If it is a message line, LINEPRT writes the stack buffer with a UPUT macro.

IDCTP01

Procedure: LINEPRT

4 LINEPRT translates the formatted line using the translate table supplied for the print chain and addressed from PCTTRP. The CHAIN or TABLE parameter of the PARM command determines the translate table. In Access Method Services translate tables, all non-printable bit combinations are changed to periods.

IDCTP01

Procedures: LINEPRT, STACKPUT

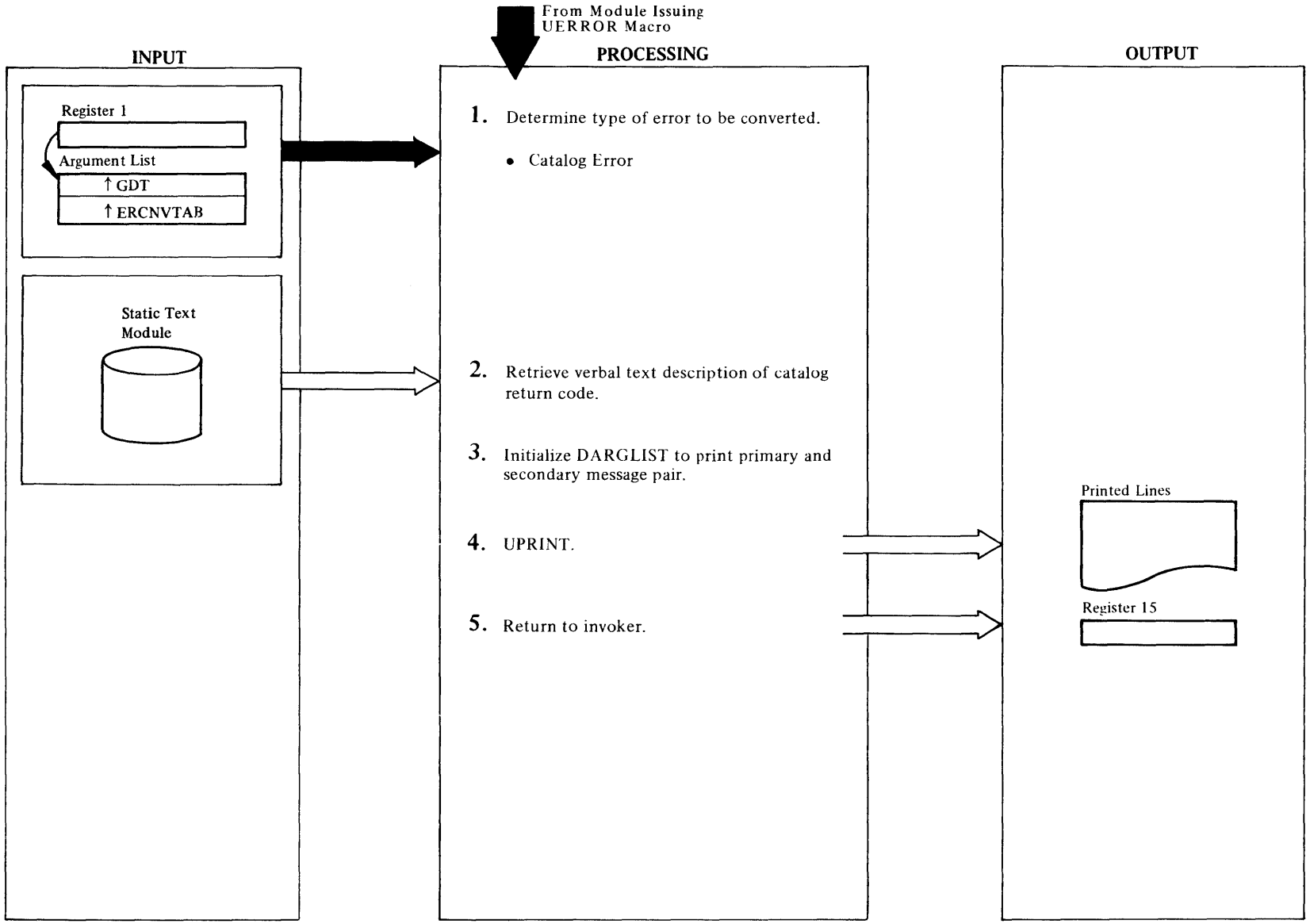
5 LINEPRT puts the translated line preceded by a two byte length field in the stack buffer. When the stack buffer is full, STACKPUT issues a UPUT against the entire buffer. Lines in the stack buffer are in variable format with as many trailing blanks removed as possible. The minimum line size is 10 bytes. If the line is a message, STACKPUT issues a UPUT against the message alone. This is done because all messages go to the standard SYSLST 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. Therefore, STACKPUT must issue a separate UPUT for each message. If an alternate 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.

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

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.

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.

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 SYSLST data set.

IDCTP06

Procedure: IDCTP06

4 Print the message pair via the Text Processor UPRINT macro.

IDCTP06

Procedure: IDCTP06

5 Control is returned to the issuer of the UERROR macro.

Debugging Aids Visual Table of Contents

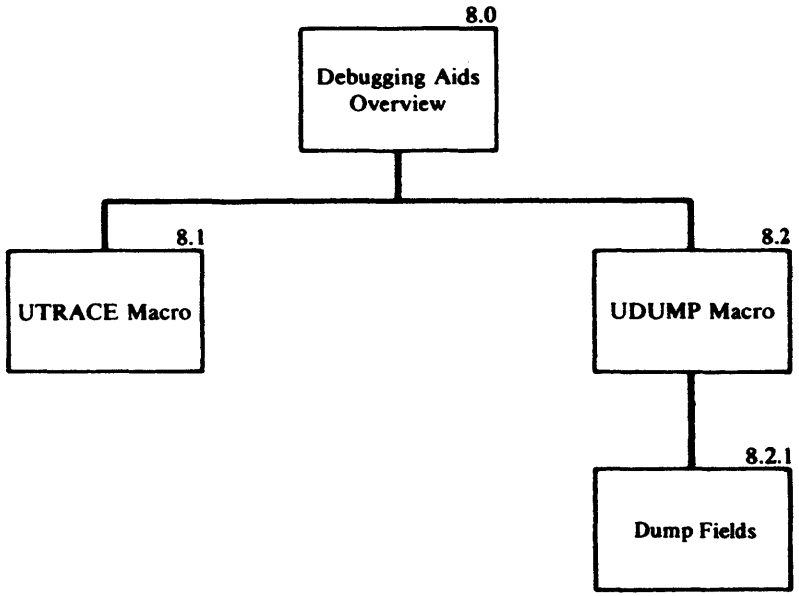
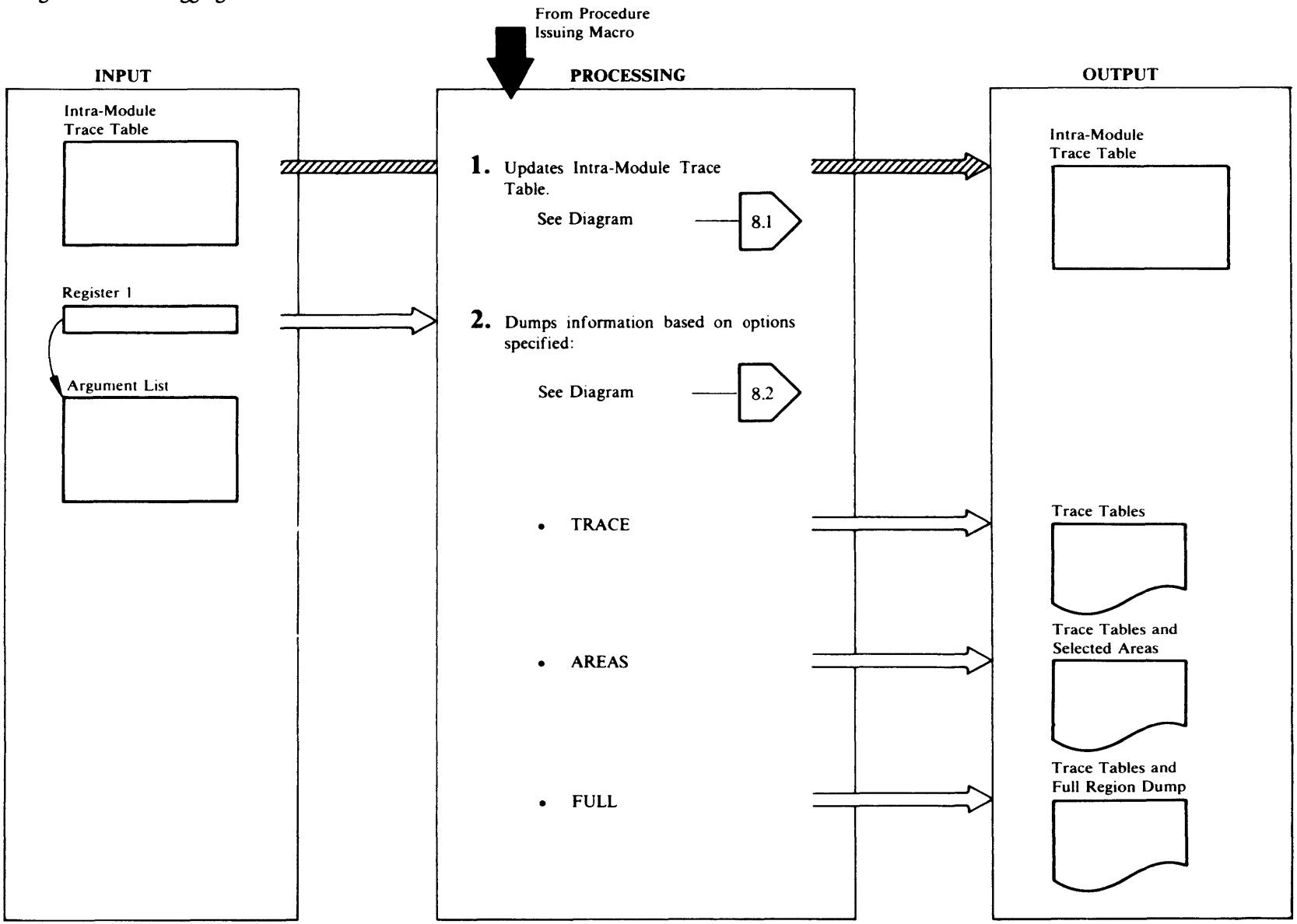


Diagram 8.0. Debugging Aids Overview



Extended Description for Diagram 8.0

IDCDB01

Procedure: IDCDB01

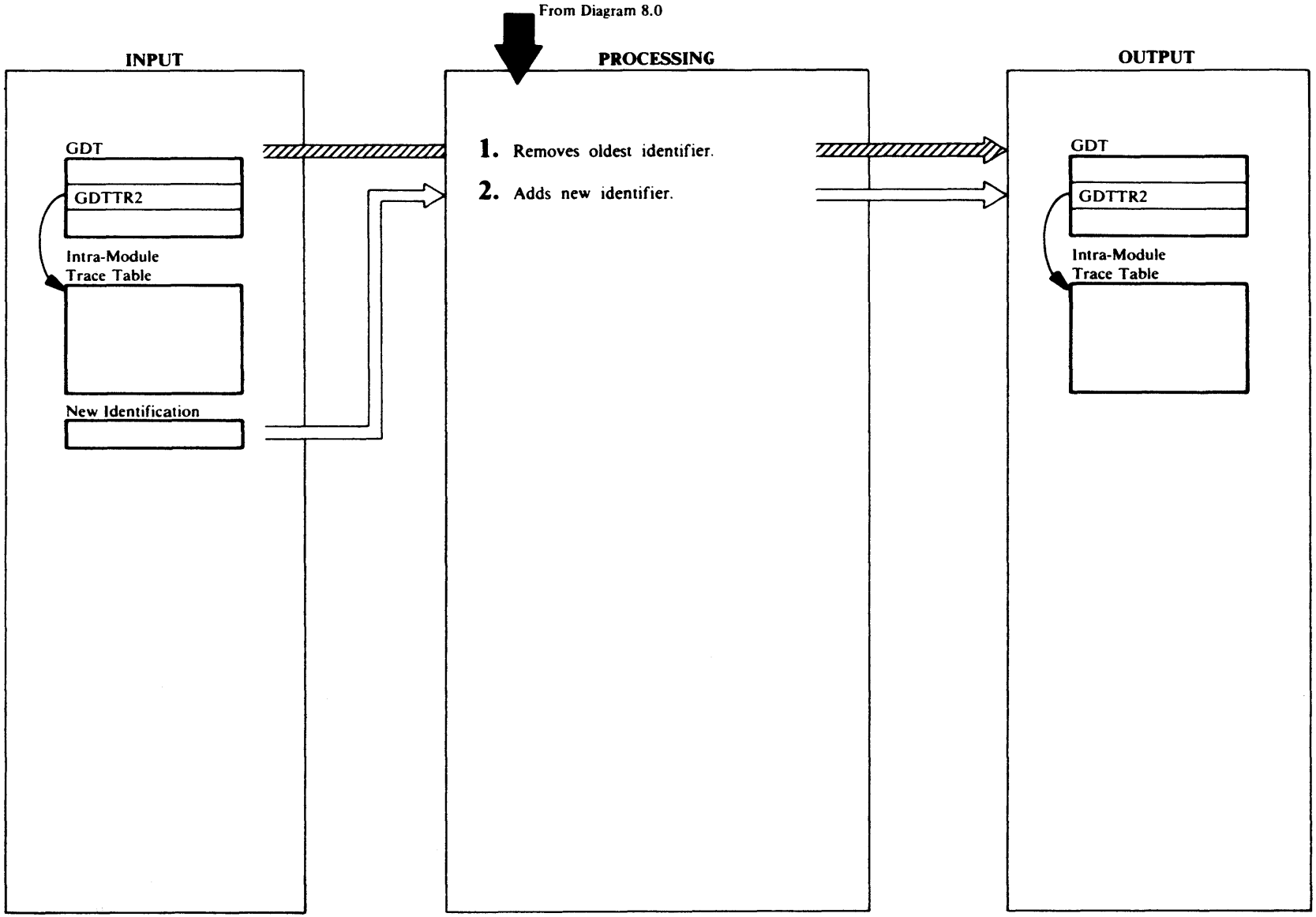
1 When a module issues a UTRACE macro instruction, the PL/S compiler generates inline code that updates the Intra-Module Trace Table. Diagram 8.1 shows the UTRACE macro instruction in detail. Processing continues with the statement following the UTRACE macro.

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 8.2 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

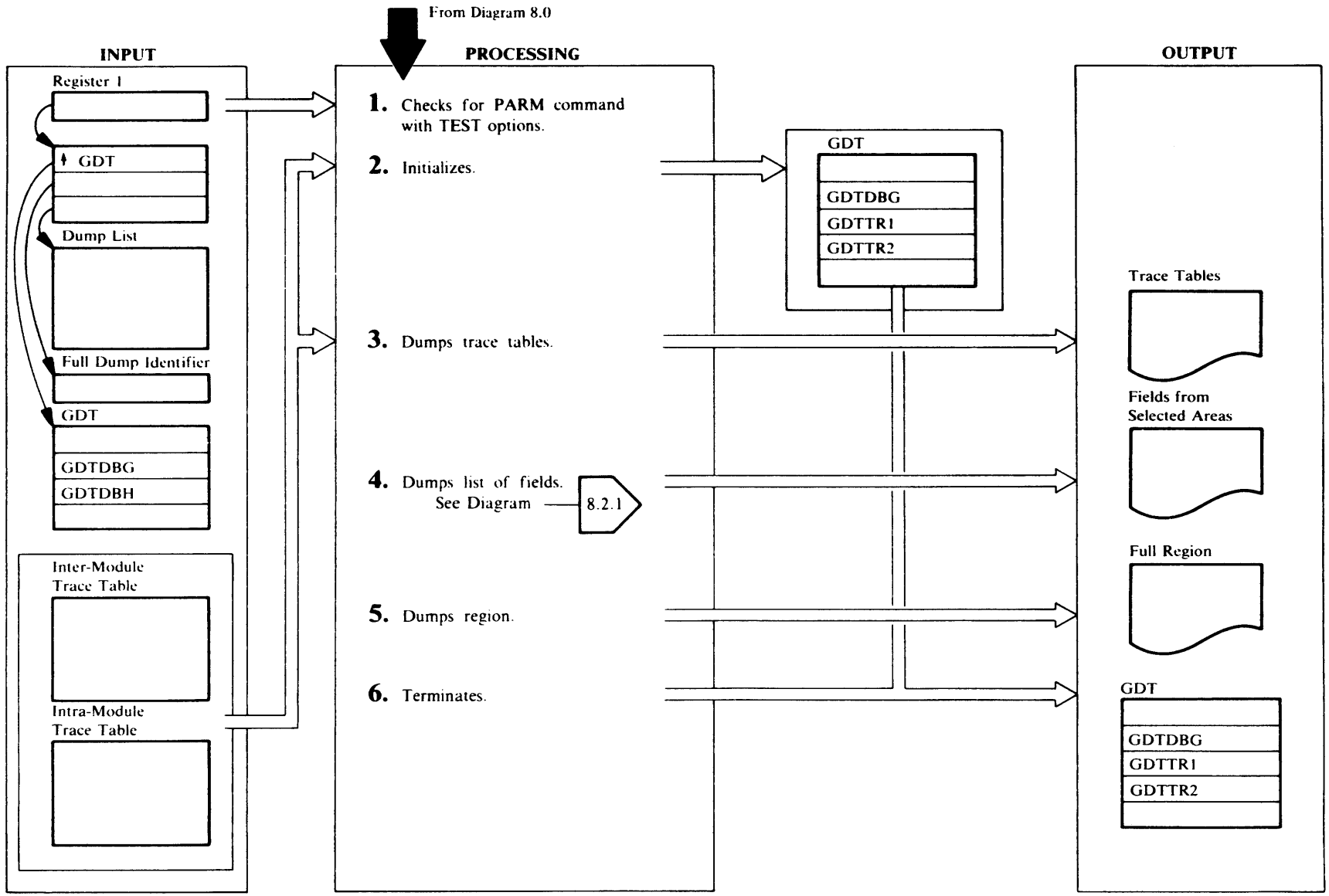
IDCDB01

Procedure: IDCDB01

1 The inline code generated by the UTRACE macro gets the address of the Intra-Module-Trace Table from the GDTTR2 field in the GDT. The inline 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

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 GDTDBG field in the GDT, if dumping is to take place. If GDTDBG is nonzero, control goes to Step 2. If GDTDBG is zero, the dump routine is not loaded and no dumping takes place; control returns to the module issuing the UDUMP macro.

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.

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 nonzero 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.

IDCDB01
IDCDB02

Procedures: 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.

IDCDB01

Procedure: IDCDB01

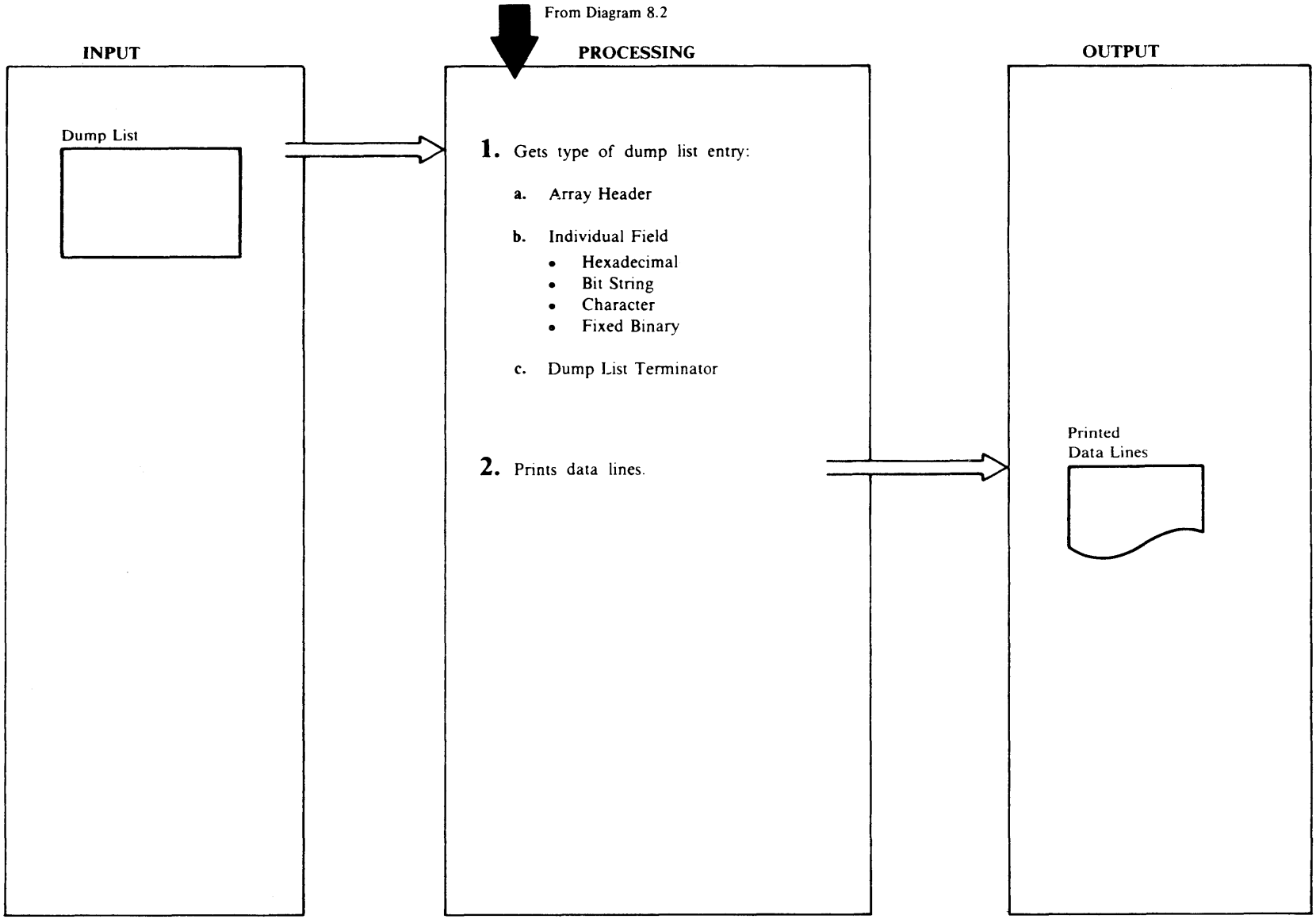
5 IDCDB01 compares the full dump identifier provided by the module issuing the UDUMP macro with full dump identifiers in FDUMPID. If no match is found, processing continues with step 6. IDCDB01 adds 1 to REALBEG and checks the number with FDUMPBEG to determine if the current pass is within the dumping 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).

IDCDB01

Procedure: IDCDB01

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 Fields



Extended Description for Diagram 8.2.1

IDCDB02

Procedures: 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 2 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.

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.

Chapter 3: 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 seven general areas, and non-executable modules (Command Descriptors and Text Structures). As described in the "Introduction," six 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, the Reader/Interpreter, and Debugging Aids. The seventh area consists of the Function Support Routines (FSRs), of which there are currently fifteen, one for each verb supported by the processor.

Several modules are link-edited together into one phase (named IDCAMS), which is loaded when the processor is invoked.

This phase is the *root phase* and consists of:

IDCEX01	Executive main routine
IDCIO01	I/O Adapter main routine
IDCSA01	System Adapter initialization/termination routine
IDCSA02	System Adapter services routine
IDCSA03	System Adapter prologue/epilogue routine
IDCTP01	Text Processor main routine
IDCSA08	System Adapter services routine

The following phases are loaded when required using CDLOAD and remain loaded until termination:

IDCEX02	Executive initialization, called by IDCEX01
IDCEX03	Executive termination, called by IDCEX01
IDCIO02	I/O Adapter Open/Close, called by IDCIO01
IDCIO03	I/O Adapter positioning and UIOINFO processing, called by IDCIO01
IDCSA05	System Adapter time routine, called by IDCSA02
IDCTP04	Text Processor page control, called by IDCTP01
IDCTP05	Text Processor Text Structure loading, called by IDCTP01 or IDCTP04
IDCTP06	Text Processor error message processor called by IDCTP01
IDCDB01	Dump routine, called by any routine
IDCDB02	Symbolic dump, called by IDCDB01

The following phases are loaded by the system when their services are required:

IDCDI01	SYSLST DTF and put phase
IDCDI02	SYSIPT DTF and get phase
IDCDI03	Fixed and fixed blocked sequential access method SDDTF and get phase
IDCDI04	Fixed and fixed blocked sequential access method SDDTF and put phase
IDCDI05	Variable and variable blocked sequential access method SDDTF and get phase
IDCDI06	Variable and variable blocked sequential access method SDDTF and put phase
IDCDI07	Undefined sequential access method SDDTF and get phase
IDCDI08	Undefined sequential access method SDDTF and put phase
IDCDI09	Spanned and spanned block sequential access method SDDTF and get phase
IDCDI10	Spanned and spanned block sequential access method SDDTF and put phase
IDCDI11	Fixed and fixed blocked sequential access method MTDTF and get/put phase
IDCDI12	Variable and variable blocked sequential access method MTDTF and get/put phase
IDCDI13	Spanned and spanned blocked sequential access method MTDTF and get/put phase
IDCDI14	Undefined sequential access method MTDTF and get/put phase
IDCDI15	Fixed and fixed blocked indexed sequential access method ISDTF and get phase

The FSRs and the Reader/Interpreter are alternately called by the Executive (IDCEX01) to perform their duties. The Reader/Interpreter is entered at IDCRI01 and loads IDCRILT and IDCRIKT when needed. The FSRs are named as follows:

IDCAL01	ALTER
IDCBI01	BLDINDEX
IDCCL01	CANCEL
IDCDE01	DEFINE
IDCDL01	DELETE
IDCXP01	EXPORT
IDCMP01	IMPORT
IDCLC01	LISTCAT
IDCLR01	LISTRCRA
IDCPM01	PARM
IDCPR01	PRINT
IDCRC01	EXPORTRA
IDCRM01	IMPORTRA
IDCRP01	REPRO
IDCRS01	RESETCAT
IDCVY01	VERIFY

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. The following 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 services in the following list to the rest of the processor. Non-system services are also provided by the adapters and by the Text Processor. Services are represented in the listings by a call to the appropriate service-module entry point.

System and I/O Macros Used by Access Method Services

Macro	Module	Label
CANCEL	IDCSA01	LCANCEL1, LCANCEL2
CATLG	IDCSA02	LCATLG1
CCB	IDCSA01	LCCB1, LCCB2
	IDCIO03	LCCB1
CDLOAD	IDCIO02, IDCIO03	LCDLOAD1
	IDCSA01	LCDLD1
	IDCSA02	LCDLD2, LCDLD3
	IDCRS01	LCDLOAD1
CLOSE	IDCIO02, IDCSA01	LCLOSE1
COMRG	IDCSA05	LCOMRG1, LCOMRG2
	IDCEX02	LCOMRG5
CVTOC	IDCIO03	LCVTOC1
	IDCRS07	LCVTOC1
DIMOD	IDCDI01, IDCDI02	
DTFDI	IDCDI01, IDCDI02	LDTFDI1
DTFIS	IDCDI15	LDTFIS1
DTFMT	IDCDI11	LDTFMT1, LDTFMT2
	IDCDI12	LDTFMT1, LDTFMT2
	IDCDI13	LDTFMT1, LDTFMT2
	IDCDI14	LDTFMT1, LDTFMT2
DTFSD	IDCDI03, IDCDI04	LDTFSD1
	IDCDI05, IDCDI06	LDTFSD1
	IDCDI07, IDCDI08	LDTFSD1
	IDCDI09, IDCDI10	LDTFSD1
ENDREQ	IDCRP01	
EOJ	IDCSA01	LEOJ1
ERASE	IDCRP01	
EXCP	IDCIO03	LEXCP1, LEXCP2
	IDCSA01	LEXCP, LEXCP2, LEXCP3
EXTRACT	IDCSA02	LEXTRCT1
	IDCSA01	LEXTRCT2
	IDCRI01	LEXTRCT1
FREEVIS	IDCIO02	LFREEV1, LFREEV2, LFREEV3, LFREEV4
	IDCSA03	LFREEV1, LFREEV2
	IDCSA01	LFREEV5, LFREEV6, LFREEV7, LFREEV8, LFREEV9
	IDCSA02	LFREEV11, LFREEV13, LFREEV14, LFREEV15
GET	IDCIO01	LGET1, LGET2, LGET3, LGET4, LGET5
GETIME	IDCSA05	LGETIME1, LGETIME2
GETVIS	IDCSA03	LGETV1
	IDCSA01	LGETV3, LGETV10
	IDCSA02	LGETV5, LGETV6

System and I/O Macros Used by Access Method Services

Macro	Module	Label
		LGETV7, LGETV8
	IDCIO02	LGETV1
ISMOD	IDCDI15	
LOAD	IDCSA02	LLDD2, LLDD3
	IDCIO02	LLOAD1
MTMOD	IDCDI11, IDCDI12 IDCDI13, IDCDI14	
OPEN	IDCIO02	LOPEN1
OVTOC	IDCIO03	LOVTOC1
	IDCRS07	LOVTOC1
PDUMP	IDCSA02	LPDUMP1
	IDCSA01	LPDUMP2
POINT	IDCIO03	LPOINT1
PUT	IDCIO01	LPUT1, LPUT2 LPUT3, LPUT4
PVTOC	IDCIO03	LPVTOC1, LPVTOC2,
	IDCRO7	LPVTOC3, LPVTOC4, LPVTOC5, LPVTOC6
SDMODFI	IDCDI03	
SDMODFO	IDCDI04	
SDMODUI	IDCDI07	
SDMODUO	IDCDI08	
SDMODVI	IDCDI05, IDCDI09	
SDMODVO	IDCDI06, IDCDI10	
SETL	IDCIO02, IDCIO03	LSETL1
TRUNC	IDCIO01	LTRUNC1
VERIFY	IDCIO01	LVRFY1
WAIT	IDCIO03	LWAIT1, LWAIT2
	IDCSA01	LWAIT1, LWAIT2, LWAIT3

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 points 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.

Services Provided for Processor Modules

The following is 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.

Internal Services Provided for Processor Modules

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)
UCALL	IDCSA02	IDCSACL	Load (if necessary) an executable module and and pass control to it.	GDT entry point name [list of arguments for called module]
UCATLG	IDCSA02	IDCSACA	Catalog request.	GDT ↑catalog parameter list
UCLOSE	IDCIO01 IDCIO02	IDCIOCL	Close one or more data sets.	GDT ↑IOCSTR[...]
UCOPY	IDCIO01	IDCIO01	Copy a data set.	GDT ↑input IOCSTR ↑output IOCSTR
UDELETE	IDCSA02	IDCSADE	No operation in DOS/VSE.	GDT module name
UDEQ	IDCSA08	IDCSADQ	Release control of a resource	GDT ↑DTL (from UENQ)
UDUMP	IDCDB01	IDCDB01	Print diagnostic output and storage dump.	GDT Dump Identifier [↑symbolic dump list]
UENQ	IDCSA08	IDCSANQ	Gain control of a resource	GDT 'SHR' | 'EXCL' 'NOWAIT' | 'WAIT' resource name reserved (not used) Scope (see "Scope Structure for UENQ - ENQSCOPE") ↑DTL (output)
UEPIL	IDCSA03	IDCSAEP	Free storage on exit from a routine.	GDT module identifier [return code]
UERROR	IDCTP06	IDCTPER	Verbalize catalog error messages.	GDT ERCNVTAB
UESTA	IDCTP01	IDCTPEA	Establish a PCT (print control table) from information in storage.	GDT alternate IOCSTR or zero for SYSPRINT PCARG
UESTS	IDCTP01	IDCTPES	Establish a PCT (print control table) from information in Text Structures.	GDT alternate IOCSTR or zero for SYSPRINT Text Structure identification
UFPOOL	IDCSA02	IDCSAFP	Release a named pool of storage.	GDT pool identification ["ALL"]
UFSPACE	IDCSA02	IDCSAFS	Release unnamed storage.	GDT address of storage to free
UGET	IDCIO01	IDCIOGT	Read a record.	GDT ↑IOCSTR

Internal Services Provided for Processor Modules

Service	Module	Entry Point	Description	Arguments
UGPOOL	IDCSA02	IDCSAGP	Allocate a named pool of storage and optionally initialize it.	GDT size of storage to obtain return storage address pool identification ["SETZERO" | "SETBLANK"]
UGSPACE	IDCSA02	IDCSAGS	Allocate unnamed storage, and optionally initialize it.	GDT size of storage to obtain return storage address ["SETZERO" | "SETBLANK"]
UIOINFO	IDCIO01 IDCIO03	IDCIOSI	Return file-ID, volume serial numbers, and/or device type information about a given filename.	GDT option flags ↑work area filename|volid logicalunitno,timestamp [pool identification]
UIOINIT	IDCIO01	IDCIOIT	Initialize the I/O Adapter.	GDT [↑zero] [↑external routine list]
UIOTERM	IDCIO01	IDCIOTM	Close all data sets that were opened with UOPEN and free all storage still used by the I/O Adapter.	GDT
ULISTLN	Inline	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. Maximum value is 255.	
ULOAD	IDCSA02	IDCSALD	Load (if necessary) a module; do not pass control to it.	GDT module name returned loaded module address [RETPNF=1] returns control on phase not found
UOPEN	IDCIO01 IDCIO02	IDCIOOP	Open one or more data sets.	GDT [{OPNAGL[...]|OPNAGL,CRAAPLIST}]
UPOSIT	IDCIO01 IDCIO03	IDCIOPO	Position to a logical record.	GDT ↑IOCSTR
UPRINT	IDCTP01	IDCTPPR	Format (and usually write) one or more lines.	GDT alternate IOCSTR or zero for SYSPRINT ↑DARGLIST [↑FMTLIST]
UPUT	IDCIO01	IDCIOPT	Write a record.	GDT ↑IOCSTR [ID code]
URESET	IDCTP01	IDCTPRE	Re-initialize PCT (print control table) for the next function.	GDT alternate IOCSTR or zero for SYSPRINT invoker's page number field
UREST	IDCTP01	IDCTPRS	Modify an existing PCT (print control table).	GDT alternate IOCSTR or zero for SYSPRINT arg^1 arg^2 arg_n
USAVERC	Inline code	None	Copies the low order half of register 15 into a halfword named TESTRC.	
USNAP	IDCSA02	IDCSASN	Call for a dump of the partition.	GDT SNAP dump-ID number
UTIME	IDCSA02	IDCSATI	Get date and time of day.	GDT field for returned time [field for returned date] ["FORM"|"KLOK"]
UTRACE	Inline	None code	Adds the current identification to the Inter-Module Trace Table.	
UVERIFY	IDCIO01	IDCIOVR	Issue VSAM VERIFY macro.	GDT ↑IOCSTR

Processor Invocation

Invocation of the Access Method Services processor is via standard DOS/VSE job control (// EXEC IDCAMS, SIZE=AUTO), or via a subroutine call. If tape or nonsequential nonVSAM files are to be processed by Access Method Services, use the LBLTYP statement to reserve storage for label information. Entry and exit to the Access Method Services processor occurs through IDCSA01, a module of the System Adapter. For a subroutine call, you must load phase IDCAMS which occupies 27,000 bytes and branch to the load address plus six. Standard linkage is used; that is, register 1 points to the argument list, register 13 points to a save area, register 14 contains the return address, and register 15 contains the entry point address. On return from the Access Method Services processor to a subroutine caller, all registers except register 15 are restored. Register 15 contains the value of MAXCC (see the section: "Processor Condition Codes" below.)

The argument list, as shown in Figure 3-1, can be a maximum of four full-word 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 can be indicated by either an address of zeros or a length of zero.

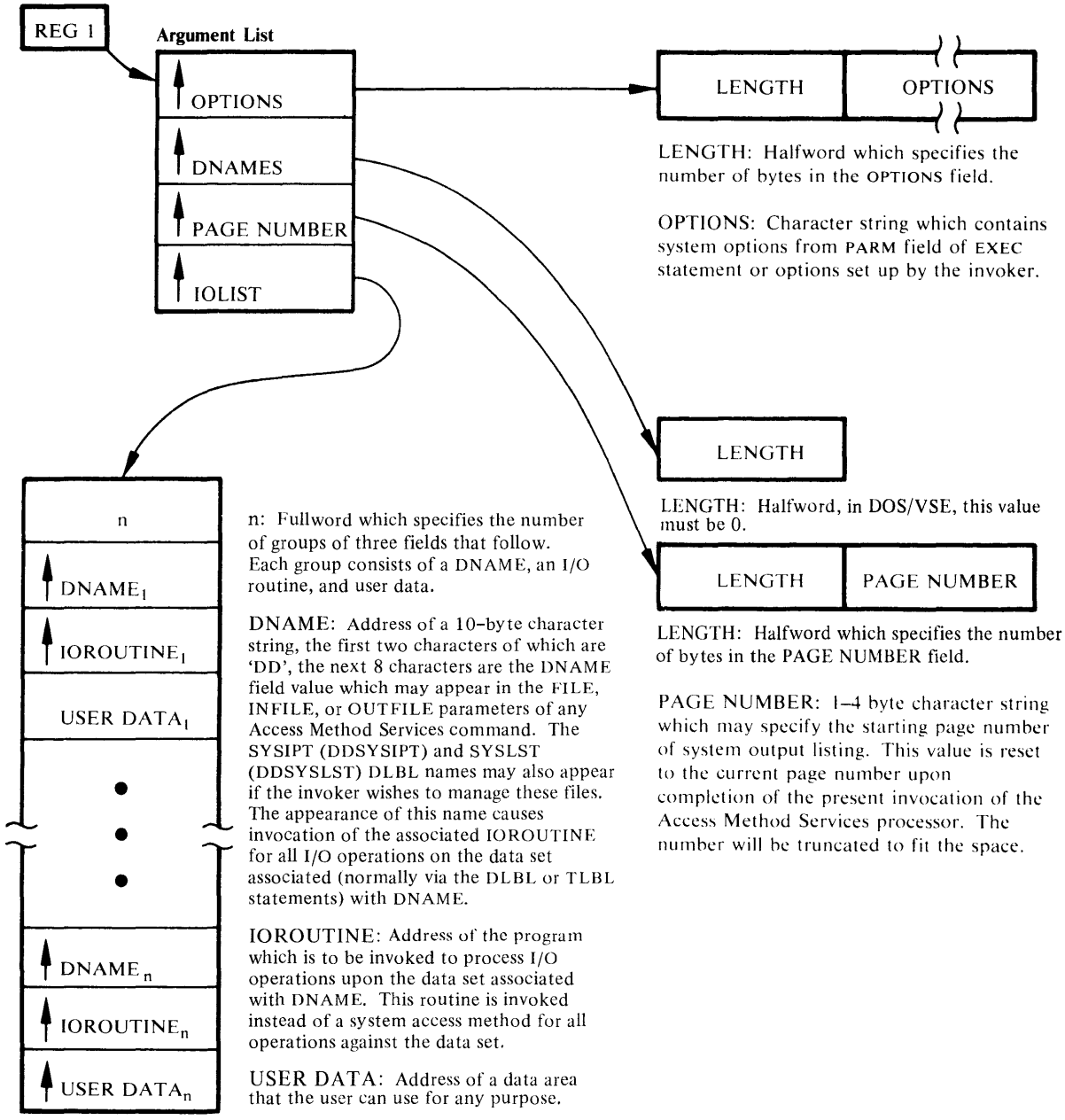


Figure 3-1. Argument List for Processor Invocation

Processor Condition Codes

The processor's condition code is LASTCC, which can be interrogated in the command stream with modal commands. The possible values, their meanings, and examples of causes 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 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 3-2 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 undefined records (maximum length—32,760 bytes) with a physical sequential organization. The processor does not test for a DLBL/TLBL 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 code are:

0 Operation successful.

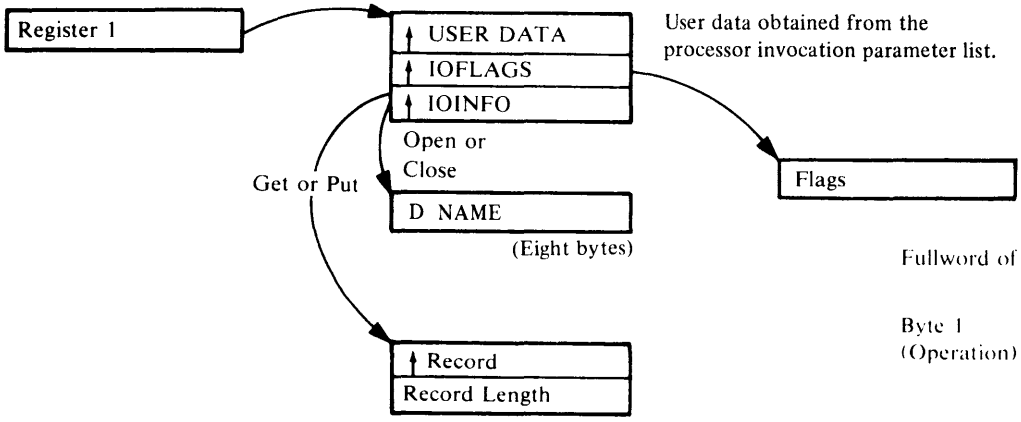
4 End of data for a GET operation.

8 Error occurred during a GET/PUT operation but continue processing.

12 Do not allow any further calls (except for CLOSE) to this routine.

Overall Control Flow

Figure 3-3 illustrates the overall control flow through the processor. 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 occurred, IDCEX01 gives control to IDCEX03. After IDCEX03 completes, IDCEX01 returns to IDCSA01.



For a GET this information is returned to the processor by the user's I/O routine in the 8-byte area passed to the routine.

For a PUT the processor gives this information to the user's I/O routine.

Fullword of flags:

	Value		Meaning
Byte 1 (Operation)	0	-	OPEN
	4	-	CLOSE
	8	-	GET
	12	-	PUT
Byte 2	32	-	Indicates IOINFO contains the address of a DD name on OPEN or CLOSE
	64	-	OPEN for output
	128	-	OPEN for input
Bytes 3,4 (Record type for PUT only)	0	-	Normal data record is to be written
	n	-	Message serial number converted to binary if IDC message is to be written

Figure 3-2. Arguments Passed to and from User I/O Routine

All modules in Figure 7 call the modules in Figure 8 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 8 also call each other for services.

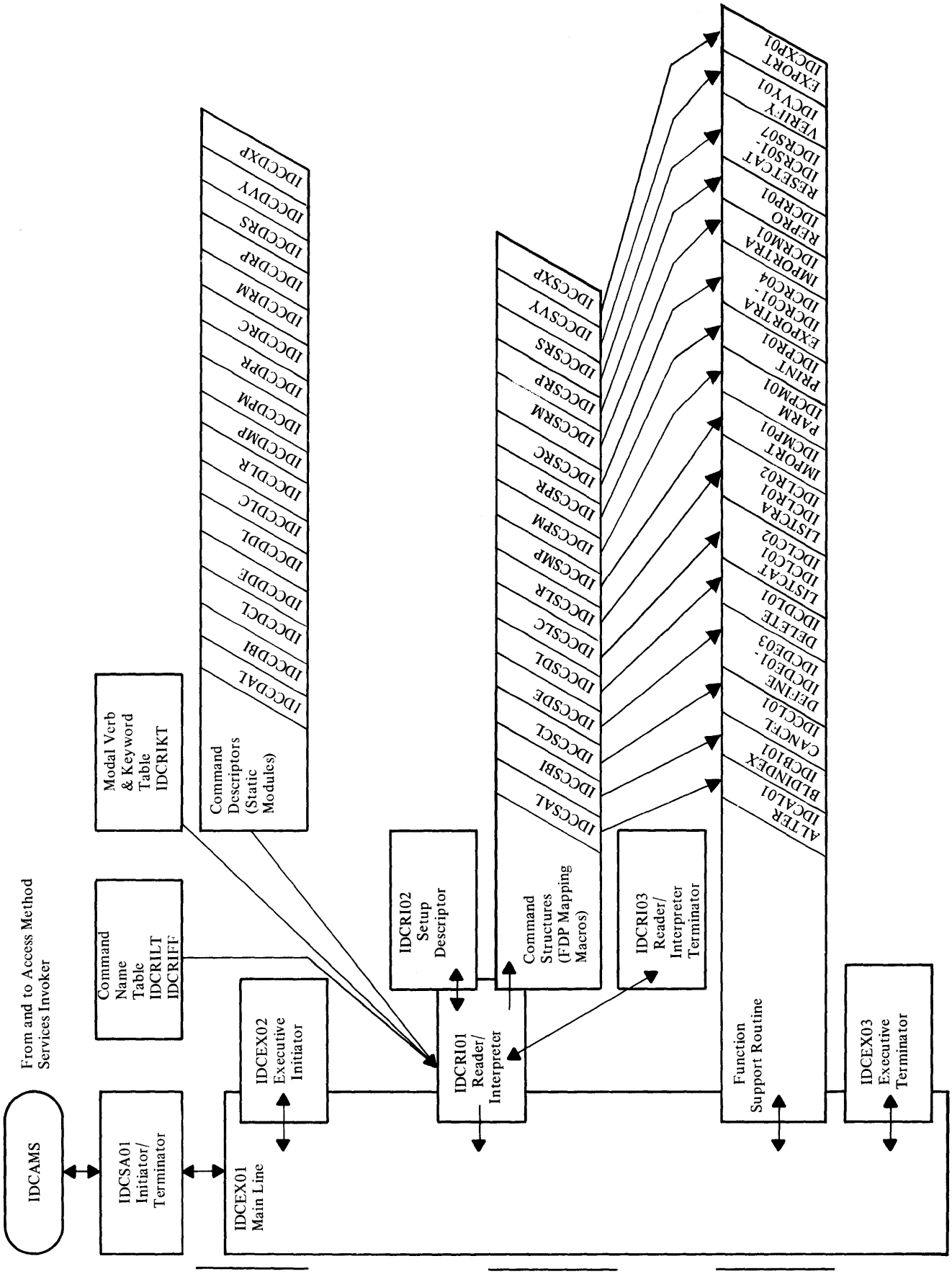


Figure 3-3. Flow of Control Through Main Functions

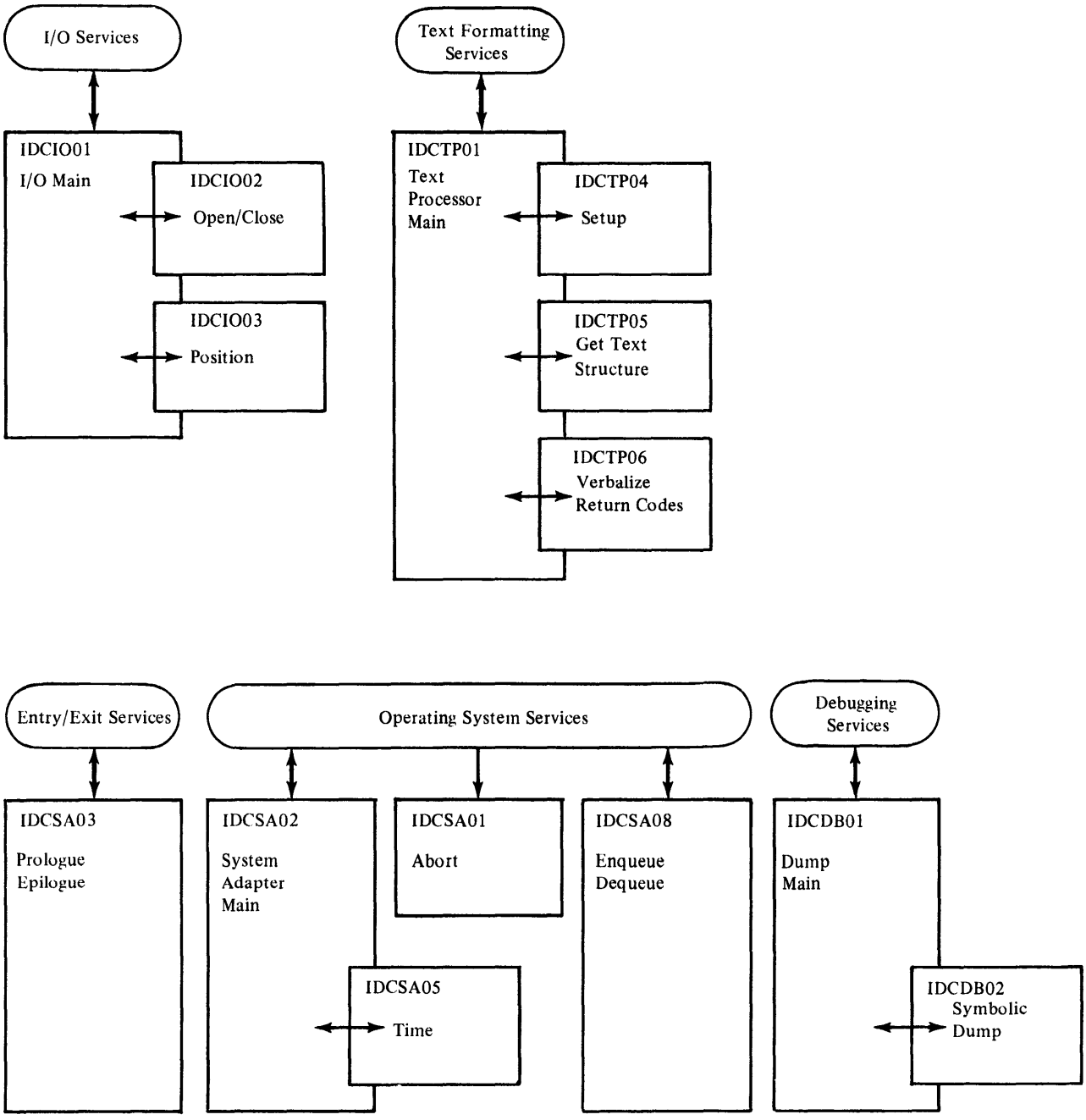


Figure 3-4. Flow of Control Through Services

Chapter 4: 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, a high-level, IBM proprietary system language. Listings that are produced for microfiche consist of the PL/S 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 and its listings.

Each module is designed with no explicit GOTOs or branches. All conditional phrases are contained within IF-THEN-ELSE clauses and DO-WHILE clauses of PL/S. 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 portion, which contains all commentary and is the most readable form of the program. All major 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 source statement numbers.

The listings are extensively commented. Each module begins with a prologue commentary that lists all standard information for that module. Throughout the listing, additional comments are boxed and structurally indented to make them easy to find. 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 *VSE/VSAM VSAM Logic, Volume 1* for a description of these 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	Description
IDCAL01			ALTER FSR; modify an existing catalog entry. Translate the encoded command parameters into the necessary catalog parameter lists and call IDCSACA for a catalog request (UCATLG macro).
	IDCAL01	EP	Only entry point to this module.
	LOCATPRC	PR	Locates catalog fields that must be altered in context. Procedure only locates those fields that contain multiple attributes. Thus, since the user may wish to change only one of several attributes, the original field must serve as the basis for alteration.
	ALTERPRC	PR	Builds the VSAM catalog management interface for the alter request.
	CHECKPRC	PR	Does validity checking on certain attributes to ensure compatibility between old values and new values.
	INDEXPROC	PR	If KEYS has been specified on the ALTER command, INDEXPRC builds the parameter list to alter the associated index object.
	PARAMCHK	PR	Verifies that parameters specified on the command are valid for the type of object to be altered.
IDCAMS		EP	Root phase for Access Method Services; consists of IDCSA01, IDCSA02, IDCSA03, IDCEX01, IDCIO01, and IDCTP01. See the directory for these modules for further description.
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.
	CATPROC	PR	Constructs CPL and FPLs 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 sort records; performs the entire internal sort or the initial sort phase of an external sort.

CSECT/Load Module Name	Label	Use	Description
	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.
IDCCDAL			Command Descriptor for ALTER verb.
IDCCDBI			Command Descriptor for BLDINDEX verb.
IDCCDCL			Command Description for CANCEL verb.
IDCCDDE			Command Descriptor for DEFINE verb.
IDCCDDL			Command Descriptor for DELETE verb.
IDCCDLC			Command Descriptor for LISTCAT verb.
IDCCDLR			Command Descriptor for LISTCRA verb.
IDCCDMP			Command Descriptor for IMPORT verb.
IDCCDPM			Command Descriptor for PARM verb.
IDCCDPR			Command Descriptor for PRINT verb.
IDCCDRC			Command Descriptor for EXPORTRA verb.
IDCCDRM			Command Descriptor for the IMPORTRA verb.
IDCCDRP			Command Descriptor for the REPRO verb.
IDCCDRS			Command Descriptor for the RESETCAT verb.
IDCCDVY			Command Descriptor for VERIFY verb.
IDCCDXP			Command Descriptor for EXPORT verb.
IDCCL01	IDCCL01	EP	CANCEL FSR; stops Access Method Services processing and optionally cancels the current job.
IDCDB01			Debug module (UDUMP macro).
	IDCDB01	EP	Only entry point to this module.
IDCDB02			Debug module (symbolic dump).
	IDCDB02	EP	Only entry point to this module.
	ARRAYHDR	PR	Processes any array header elements (TYPE="A") occurring in the dump list.
	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.
	HCONVERT	PR	Converts the value of the current dump item to hexadecimal representation.
	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.
IDCDE01			DEFINE FSR; define a new VSAM data set as a cataloged object.
	IDCDE01	EP	Only entry point to this module.
	INTGCHK	PR	Performs validity checking on completed catalog parameter list.
IDCDE02			Common processing routines for all define types.
	IDCDE02	EP	Initializes registers and obtains storage.
	NAMEPROC	EP	Initializes the data set creation and expiration dates in the CTGFL and the object name in the CTGFV.
	ALLCPROC	EP	Initializes several allocation and option related parameters in the CTGFL and CTGFV.
	KEYPROC	EP	Initializes the record management control block and the key range "pseudo-field" in the CTGFL.
	IXOPPROC	EP	Initializes index options.

CSECT/Load Module Name	Label	Use	Description
	PROTPROC	EP	Initializes the security combination and owner identification fields and the SHAREOPTIONS and ERASE|NOERASE flags in the CTGFL.
	MODELPRC	PR	Handles the retrieval of model objects to be used in defining components of VSAM user catalogs and data sets.
	FREESTG	EP	Frees automatic storage for IDCDE02 CSECT.
IDCDE03			Routes control to proper routine.
	IDCDE03	EP	Calls proper procedure to construct parameter list for the different object types.
	CTLGPROC	PR	Oversees the construction of the VSAM CTGPL, CTGFV, and CTGFL for defining a VSAM master or user catalog.
	DSETPROC	PR	Oversees the construction of VSAM key sequenced and entry sequenced data sets.
	AIXPROC	PR	Oversees the construction of the VSAM catalog interface for defining alternate index data sets.
	PATHPROC	PR	Oversees the construction of the VSAM catalog interface for defining paths.
	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 nonVSAM data set into a VSAM catalog.
IDCDI01			SYSLST DTF and put phase.
IDCDI02			SYSIPT DTF and get phase.
IDCDI03			Fixed and fixed blocked sequential access method SD DTF and get phase.
IDCDI04			Fixed and fixed blocked sequential access method SDDTF and put phase.
IDCDI05			Variable and variable blocked sequential access method SDDTF and get phase.
IDCDI06			Variable and variable blocked sequential access method SDDTF and put phase.
IDCDI07			Undefined seqeuntial access method SDDTF and get phase.
IDCDI08			Undefined sequential access method SDDTF and put phase.
IDCDI09			Spanned and spanned block sequential access method SDDTF and get phase.
IDCDI10			Spanned and spanned block sequential access method SDDTF and put phase.
IDCDI11			Fixed and fixed blocked sequential access method MTDTF and get/put phase.
IDCDI12			Variable and variable blocked sequential access method MTDTF and get/put phase.
IDCDI13			Spanned and spanned blocked sequential access method MTDTF and get/put phase.
IDCDI14			Undefined sequential access method MTDTF and get/put phase.
IDCDI15			Fixed and fixed blocked indexed sequential access method DTF and get phase.
IDCDL01			DELETE FSR; delete a catalog entry from the VSAM catalog.
	IDCDL01	EP	Only entry point to this module.
	CATOPEN	PR	Opens the user catalog if required.

CSECT/Load Module Name	Label	Use	Description
	FINDTYPE	PR	Locates the entry to be deleted in order to determine its type when type is not specified in command.
	PARAMCHK	PR	Checks for invalid type specification and other command parameter errors.
	BUILDCPL	PR	Constructs the CTGPL from parameters specified in the DELETE command and indicated in the FDT.
	CATCALL	PR	Calls VSAM catalog management to delete a single catalog entry.
	MORESP	PR	Obtains a larger catalog work area and reinvokes catalog management.
	CLEANUP	PR	Performs termination functions and closes the user catalog, if required.
IDCEX01			Main-line for Executive; routes control through processor.
	IDCEX01	EP	Only entry point to this module; entered from IDCSA01.
	MAIN	PR	Flip-flop control between Reader/Interpreter and FSR required for each command.
	CALLRI	PR	Invoke Reader/Interpreter to parse the next command.
	CALLFSR	PR	Invoke FSR named by the result of parse by Reader/Interpreter.
IDCEX02			Executive, initialize the processor.
	IDCEX02	EP	Only entry point to this module.
	SCANPARM	PR	Scan processor invocation parameter list.
IDCEX03			Executive, terminate processing.
	IDCEX03	EP	Only entry point to this module.
	SCANPARM	PR	Scan invoker's parameter list to return next available page number.
IDCIO01			Supply all I/O services to the remainder of the processor. At each of the following entry points, IDCIO01 converts the service request to the appropriate system macros and issues those macros.
	IDCIOIT	EP	First call to I/O Adapter: initialize the adapter for subsequent calls.
	IDCIOOP	EP	Open 1 to 4 data sets (UOPEN macro), by calling IDCIO02.
	IDCIOTM	EP	Close any data sets still open (UIOTERM macro).
	IDCIOCL	EP	Close 1 to 4 data sets (UCLOSE macro), by calling IDCIO02.
	IDCIOPO	EP	Position to a specific record in a data set (UPOSIT macro), by calling IDCIO03.
	IDCIOSI	EP	Obtain various pieces of information about data set.
	IDCIOGT	EP	Read a record (UGET macro).
	IDCIOPT	EP	Write a record (UPUT macro).
	IDCIOVY	EP	Verify data set (UVERIFY macro).
	IDCIOCO	EP	Copy a data set (UCOPY macro).
	CHANGE	PR	Handles change of processing modes for RPL.
	GETEXT	PR	Call an external routine to get a data record.
	GETVSAM	PR	Get a logical record or control interval from a VSAM data set.
	IRAMEOD	PR	End-of-data-set exit routine for VSAM data sets.
	GETNONVS	PR	Get a logical record from a nonVSAM data set.
	IROSEOD	PR	End-of-data-set exit routine for nonVSAM data sets.

CSECT/Load Module Name	Label	Use	Description
	PUTEXT	PR	Call a user-supplied routine for output.
	PUTVSAM	PR	Put a logical record to a VSAM data set.
	PUTNONVS	PR	Put a logical record to a nonVSAM data set.
	PUTREP	PR	Handle PUT (Replace) processing.
	VSAMERR	PR	Build VSAM error message argument list.
	BLDAMSG	PR	Prepare an error message.
	PRINTMSG	PR	Print a message.
	IDCIOS1	DE	Amount of storage IDCIO01 needs. Used by IDCSA01.
	IRSISYN	PR	Exit routine for I/O errors when attempting a GET on a nonVSAM, nonISAM data set.
	IRSOSYN	PR	Exit routine for I/O errors when attempting a PUT on a nonVSAM, nonISAM data set.
IDCIO02			Open/Close routine This routine can open or close 1 to 4 data sets with one call.
	IDCIO02	EP	Only entry point to this module.
	OPENRTN	PR	Open a data set.
	CKNONOP	PR	Check that a nonVSAM data set was opened successfully.
	CLOSERTN	PR	Close data sets that were opened by the I/O Open routine.
	ENVFREE	PR	Free storage used for a data set; system areas, buffers, control blocks, DTF, and access load module.
	DSDATA	PR	Issue CDLOAD and CALL for IKQVLAB, which returns the label information.
	BUILDRPL	PR	Build RPL for VSAM data set and get input workareas for buffers.
	BUILDACB	PR	Build ACB and EXLST for VSAM data set to be opened.
	BUILDDBK	PR	Load DTF and access module and modify DTF for a nonVSAM data set to be opened.
	BLDOCMSG	PR	Set up an error message.
	PRINTMSG	PR	Call Text Processor to print error message.
IDCIO03			Perform POINT, SETL and UIOINFO operations.
	IDCIO03	EP	Only entry point to this module.
	PTAMDS	PR	Point to VSAM logical record.
	PTISDS	PR	SETL to ISAM logical record.
	BLDAMSG	PR	Prepare error message.
	PRNTMSG	PR	Print message.
	DSINFO	PR	Find volume/data set information.
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.
	INITPROC	PR	Interrogates the FDT and initializes the catalog and DADSM parameter lists and workareas. Issues ULOAD for IKQDNT, the device name table.
	GNXTPROC	PR	Manages the request for all or a specified subset of the catalog entry types in alphameric sequence.
	ENTPROC	PR	Manages the request for specific entries from the catalog.
	RTEPROC	PR	Routes control to the appropriate formatting procedure. Then routes control for formatting the associated data sets in a cluster or alternate index grouping.

CSECT/Load Module Name	Label	Use	Description
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.
	FREESTG	EP	Issues a UEPIL umacro to free the automatic storage acquired by IDCLC02.
	FPLPROC	EP	Re-initializes the string of CTGFLs prior to each catalog locate request, by using SAVEAREA copy stored at the original CTGFL-build time.
	LISTPROC	EP	Issues the Text Processor macro UPRINT and zeros out the Dynamic Data Area Argument List upon exiting.
	AUPROC	EP	Repetitively builds the Text Processor Dynamic Data Argument List for formatting and listing the VSAM catalog fields for nonVSAM or user catalog entry. Repeatedly invokes LISTPROC to print the data.
	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 LISTC subtitle with the catalog name.
	CDIPROC	EP	Formats the VSAM catalog data for cluster, alternate index, data, index, and path associations. Issues the locate request to obtain associated data set names for listing the cluster-data set-index-path and alternate index-data set-index-path associations. Builds the Text Processor argument list and invokes LISTPROC to print the data.
	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 invokes 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 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 upon return to the caller.
	ANSVPROC	EP	Retrieves the list of associated C.I. numbers and types from the work area and creates a save area copy.
	DEVTCONV	PR	Translates the hexadecimal UCB device type code to the external device name.
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 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 and 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.

CSECT/Load Module Name	Label	Use	Description
	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 containing 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.
	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 timestamps 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	Loops through the VERTEXT and extensions and prints them.
IDCLR02		EP	Formats the buffer pool and reads CRA and catalog records.
IDCMP01			IMPORT FSR; reconstruct a VSAM cluster or alternate index from a portable copy that was created by IDCXP01. Any associated. paths are also recreated IDCSACA is called (UCATLG macro) to add the necessary entries to the VSAM

CSECT/Load Module Name	Label	Use	Description
			catalog, and a UCOPY macro is issued to copy the data set by logical records. When the input data set is a catalog, no copy is performed; instead the catalog is connected by a call to IDCSACA.
	IDCMP01	EP	Only entry point to this module.
	CLUSPROC	PR	Reads catalog and data records from the portable volume. Uses catalog information plus information from the command to perform a catalog define for the cluster or alternate index. Copies data into the object after successful definition in the catalog.
	CNCTPROC	PR	Connects one or more user catalogs.
	DUPNPROC	PR	This procedure is called when a duplicate entry name is found in the catalog when trying to define the data set to be imported. A locate will be performed. If the entry has the temporary export flag set in the attributes field, a delete is then performed so that the imported data set may be defined. If the entry is empty, checks are made for matching attributes so that import can be performed into a predefined empty data set.
	CPLPROC	PR	Constructs a CTGPL to be used for a catalog define, alter, delete, or locate operation.
	IUNIQPRC	PR	Checks the DSATTR field in the CTGFV to see if the cluster being defined is a unique data set. If so, a null space (volume) CTGFV must be supplied for catalog define.
	ALTRPROC	PR	Constructs a CTGPL and CTGFV to be employed by the catalog alter interface.
	LVLRPROC	PR	Constructs CTGFL for DEVTYPE lists and constructs list of volume serial numbers.
	CTLGPROC	PR	Invokes the VSAM catalog management to perform the operation indicated in the CTGPL.
	DELTPROC	PR	Deletes any temporarily exported data sets found by DUPNPROC.
	OPENPROC	PR	Performs all opens required for opening a VSAM object or user catalog for input or opening the portable volume for output.
	RANGPROC	PR	Processes all information dealing with key ranges.
	BFPLPROC	PR	Constructs a CTGFL from dictionary and workarea information.
	RECPROC	PR	Copies the data from the portable data set to the VSAM object being imported. The VSAM object is opened by OPENPROC. The UCOPY macro is employed to perform the copy. The UCLOSE macro is employed to close the object.
	MVDAPROC	PR	Moves data from one location in virtual storage to another as specified by input arguments.
	MSGPROC	PR	Uses the Text Processor interface to list messages.
	FVTPROC	PR	Constructs CTGFVs and CTGFLs from information in the dictionary. Obtains portable file LRECL and passes it to the I/O Adapter.
	BPASPROC	PR	Constructs PASSWALL CTGFL and moves information into PASSWALL.
	GETPROC	PR	Gets a data record and moves it into a buffer. Reconstructs the original record if it has been segmented.
|	DVOLPROC	PR	Constructs the special volumes CTGFLfrom the DEFAULTVOLUMES parameter.
|	DVOLCHK	PR	Performs diagnostics to assure that DEFAULTVOLUMES volumes CTGFLs were constructed only for components whose attributes

CSECT/Load Module Name	Label	Use	Description
			are compatible with DEFAULTVOLUMES. Also checks to warn if DEFAULTVOLUMES was specified but ignored.
IDCPM01			PARM FSR; establish or change the processor parameters. Processor parameters (TEST, MARGINS, and GRAPHICS) can be established through the PARM field of the EXEC card. This FSR provides an alternate way to set these options.
			The results of changing TEST appear in the area whose address is in GDTDBH. The results of changing MARGINS appear as the first two halfwords in the area whose address appears in GDTRIH, and GRAPHICS is recorded in the PCT.
	IDCPM01	EP	Only entry point to this module.
	TESTPARM	PR	Resets the previous test option if necessary. Processes new test option. Obtains and initializes the Test Option Data Area.
	TESTSAVE	PR	Extracts the specified test parameters from the FDT and places them in the Test Option Data Area to be used by the Access Method Services dump routine.
	MARGPARM	PR	Processes the input command source margins specified. The left and right margin values are placed into the Reader/Interpreter Historical Data Area to be used by the Reader/Interpreter when processing subsequent command input.
	GRPHPARM	PR	Determine graphics option chosen and issue UREST macro to establish the specified translate table.
IDCPR01			PRINT FSR; print the contents of a data set in EBCDIC, hexadecimal, or dump format. Page layout is established with a call to IDCTPEA (UESTA macro) and lines of data are printed by calling IDCTPPR (UPRINT macro).
	IDCPR01	EP	Only entry point to this module.
	TEXTPSET	PR	Communicates the page layout and record layout for the listing to the Text Processor.
	DELIMSET	PR	Establishes the boundaries for printing a subset of the input data set.
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, flags empty data sets, bypasses OS/VS-only 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.

CSECT/Load Module Name	Label	Use	Description
	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 timestamp 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 timestamp, 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.
	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 timestamp 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.
	IDCRC02	EP	Only entry point to this module.
	ALSPROC	PR	Bypasses portable file information for OS/VS2 alias associations of nonVSAM data sets.
	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.
	GDGPROC	PR	Bypasses portable file information for OS/VS2 GDG bases.
	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.

CSECT/Load Module Name	Label	Use	Description
	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.
	PTSTS	PR	Tests for existence of field and if there, places dictionary information into work area.
IDCRI01			Consists of CSECTs IDCRI01, IDCRI02, and IDCRI03. IDCRI01 is the Reader/Interpreter main-line routine. Its functions are: 1. On first entry only, load a table of Command Descriptor phase names and a table of modal command verbs, initialize the Reader/Interpreter Historical Data Area, and obtain PARM options input if it exists in the PARM field of the EXEC statement. 2. Scan the input stream for a command verb. 3. Handle modal commands (IF, ELSE, DO, END, and SET) to determine which command to process next. 4. Having found a function command verb, invoke IDCRI02 to find and load the appropriate Command Descriptor module and initialize the FDT. 5. Scan parameter set, using the Command Descriptor, to check syntax and semantics and to build FDT. 6. Invoke IDCRI03 for clean-up activity following each function command, and return to IDCEX01 if the function command is to be executed—that is, if it contains no syntax or semantic errors detectable by the Reader/Interpreter.
	IDCRI01	EP	Only entry point to this module.
	RIINIT	PR	Initialize Reader/Interpreter processing.

CSECT/Load Module Name	Label	Use	Description
	SCANCMD	PR	Control command scanning and FDT building.
	GETNEXT	PR	Get next function command verb name and pointer to its parameter set. Intepreter modal commands.
	MODALSET	PR	Process SET modal command.
	MODALIF	PR	Process IF modal command.
	MODLELSE	PR	Process ELSE modal command.
	BYPASTRM	PR	Prepare to obtain next verb name.
	KWDPARM	PR	Process a keyword after searching the Command Descriptor for its match.
	POSPARM	PR	Process a positional parameter.
	GETDATA	PR	Set up to extract constant or list of constants.
	GETSIMPL	PR	Extract an unquoted constant.
	GETQUOTD	PR	Extract a constant from within apostrophes.
	BUILDFDT	PR	Place constants into FDT (converting if needed).
	CONVERT	PR	Convert EBCDIC to binary, decimal, or hexadecimal.
	DSIDCHK	PR	Check data set name item for adherence to naming conventions.
	GETSPACE	PR	Allocate space for an FDT element.
	MORSPACE	PR	Allocate additional space for a list of constants in an FDT element.
	INREPEAT	PR	End of repetition of a subparameter list has occurred; prepare for another of the subparameter list repetitions.
	DEFAULTS	PR	Add defaults to parameters explicitly specified.
	ERRSETUP	PR	Make special preparations to print semantic error message.
	NEEDNOTS	PR	Check parameters to ensure that certain semantic requirements have not been violated. Check for mutually exclusive parameters, and required parameters.
	SKIPCMD	PR	Bypass remainder of current command.
	SETFLAG	PR	Flag that a particular parameter was found in the input or was implied by defaults.
	PACKCVB	PR	Convert EBCDIC string to fullword binary number.
	NXTFIELD	PR	Extract next field from the input stream.
	SCANSEP	PR	Scan past the next syntactic separator (comma, blanks, and/or comments).
	NEXTCHAR	PR	Extract the next character of the input stream.
	GETRECRD	PR	Read the next input record and print it.
	SCANENDS	PR	Find left and right scanning limits of command text in the input record just read.
	DSPLCALC	PR	Calculate offset into an array of pointers or counts.
	ERROR1	PR	Process an error whose message is static.
	ERROR2	PR	Process an error that requires variable data to be inserted into the message.
IDCRI02			Search the table of Command-Descriptor phases for the name of the phase that corresponds to the current command, and then load that phase. Initialize the FDT.
	IDCRI02	EP	Only entry point to this module.
IDCRI03			Reader/Interpreter function command termination. Free working space and delete unneeded phases.
	IDCRI03	EP	Only entry point to this module.

CSECT/Load Module Name	Label	Use	Description
IDCRIFF			Last entry indicator for Module Name Table for command descriptors used by the Reader/Interpreter.
IDCRIKT			Modal command verb and keyword table, used by the Reader/Interpreter.
IDCRILT			Load Module Name Table for command descriptors used by the Reader/Interpreter.
IDCRM01		EP	Only entry point to this module.
	ALISPROC	PR	Reads data records and checks for allowable type in the DOS system.
	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.
	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	If this procedure is called in DOS, it writes an error message.
	IUNIQPRC	PR	Checks the DSATTR field in the CTGFV to see if the cluster being defined is a unique data set. If so, a null space (volume) CGTFV must be supplied for catalog define.
	LVLRPROC	PR	Constructs the FPL from the DEVICETYPES parameter or LISTVOLS from the RANGES parameter.
	MSGPROC	PR	Uses the Text Processor to list messages.
	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.
	UCATPROC	PR	Reads catalog and data records from portable volume and performs a define of user catalog pointers and aliases.
	DVOLPROC	PR	Contructs the special volumes CTGFL from the DEFAULTVOLUMES parameter.
	DVOLCHK	PR	Performs diagnostics to assure that DEFAULTVOLUMES volumes CTGFLs were constructed only for components whose attributes

CSECT/Load Module Name	Label	Use	Description
			are compatible with DEFAULTVOLUMES. Also checks to warn if DEFAULTVOLUMES was specified but ignored.
IDCRP01			REPRO FSR; copy a SAM, ISAM, or VSAMdata 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 UPUTs 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 record.
	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.
	CATCOMP	PR	Indicates differences in truename entries between backup and target catalogs.
	VERIFYC	PR	Opens a data set for control interval processing in order to compare the end-of-data-set and end-of-key-range information stored in the VSAM catalog with the true data in the data set. Reopens the data set for normal keyed processing.
IDCRS01			RESETCAT FSR; synchronize a catalog with the CRA (s) of its owned volume.
	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.
	UPDCRA	PR	Update the CRAs from the workfile.

CSECT/Load Module Name	Label	Use	Description
	WRAPUP	PR	Handle clean-up operations after successful RESETCAT processing.
IDCRS02			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 users catalogs.
	VERX	PR	Verify the alias chain.
IDCRS03			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			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			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.

CSECT/Load Module Name	Label	Use	Description
	CKERR	PR	Print an error message.
	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			Handles I/O functions; defines and deletes the workfile.
	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			This module contains system dependent code designed specifically for RESETCAT functions.
	CATEOV	PR	Extend the catalog.
	CNVTCCHH	PR	Convert CCHH or BBBB 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.
	HVTOC	PR	Process all common VTOC handler functions.
	RENAMEP	PR	Rename duplicate true name entries.
	UPDCAT	PR	Update the catalog from the workfile.
	UPDCCR	PR	Update the catalog control record (CCR).
IDCSA01			Entry and exit module for the Access Method Services processor. Interface between the operating system and the processor. Create the GDT and call IDCEX01.
	IDCSA01	EP	Entry point for DOS Job Control invocation.
	IDCSAS1	EP	Entry point for subroutine call invocation. It is six bytes beyond IDCSA01.
	PRNTERR	PR	Print an error message using EXCP.
	GETCORE	PR	Issue GETVIS to allocate storage.
IDCSA02			Supply all system services to the remainder of the processor, except prologue and epilogue At each of the following entry points, IDCSA02 converts the service request to the appropriate system macros, and issues those macros.

CSECT/Load Module Name	Label	Use	Description
	IDCSACL	EP	Load an executable module and branch to it (UCALL macro).
	IDCSALD	EP	Load a module but do not branch to it (ULOAD macro).
	IDCSADE	EP	Not functional in DOS/VSE.
	IDCSAGS	EP	Get space, a request for non-pooled storage (UGSPACE macro).
	IDCSAFS	EP	Free space, release pooled or non-pooled storage (UFSPACE macro).
	IDCSAGP	EP	Get pool, a request for pooled storage (UGPOOL macro).
	IDCSAFP	EP	Free pool, release pooled storage (UFPOOL macro).
	IDCSATI	EP	Get date and time of day by calling IDCSA05 (UTIME macro).
	IDCSACA	EP	Issue the VSAM CATLG macro (UCATLG macro).
	IDCSASN	EP	Provide core dump (USNAP macro).
	COREINIT	PR	Initialize an area of storage to binary zeros or blanks.
	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.
	IDCSAPR	EP	Prologue entry point, acquire storage.
	IDCSAEP	EP	Epilogue entry point (UEPIL macro), release storage.
	GETCORE	PR	Get requested amount of storage.
	IDCSAS3	DE	Amount of storage IDCSA03 needs Used by IDCSA01.
IDCSA04			Phase table containing load status information of other phases. Used by the System Adapter.
IDCSA05			Get date and time of day (invoked by IDCSA02).
	IDCSA05	EP	Only entry point to this module.
IDCSA08			Acquire control of a resource. Release control of a resource.
IDCTP01			Text Processor: provide formatting for printed output. Each of the following entry points represents a service provided by the Text Processor. This module includes all conversion routines and controls the printing of each line of output text.
	IDCTPES	EP	Establish a PCT from static text (UESTS macro).
	IDCTPEA	EP	Establish a PCT from storage (UESTA macro).
	IDCTPER	EP	Establish linkage to error message processor (UERROR).
	IDCTPRS	EP	Modify an existing PCT (UREST macro).
	IDCTPRE	EP	Re-initialize Text Processor for the next function (URESET macro).
	IDCTPPR	EP	Print one or more lines (UPRINT macro).
	SPACE	PR	Set up line spacing.
	REDO	PR	Initiate replication.
	STATIC	PR	Set up static text.
	BLOCK	PR	Set up block data.
	INSERT	PR	Routine to insert data into predefined format, or use static text when an insert is missing and default data is called for.

CSECT/Load Module Name	Label	Use	Description
	CONVERT	PR	Converts data and sets it into the print line.
	BHCONV	PR	Convert binary data to hexadecimal characters or hex-apostrophe representation.
	BHDCONV	PR	Convert binary data to hex-dump format.
	EBCDIC	PR	Sets up transfer of EBCDIC characters to a print line.
	PUPCONV	PR	Convert packed-decimal data to unpacked-decimal characters.
	BDCONV	PR	Convert binary data to packed-decimal data, and call PUPCONV for conversion to unpacked-decimal characters.
	IDCTPS1	DE	Amount of storage IDCTP01 needs. Used by IDCSA01.
	ERROR	PR	Process error condition.
	STACKPUT	PR	Buffers data lines. Does a UPUT on the line when the stack is full, a message is to be printed, or the print file is changed.
	LINERET	PR	Returns formatted lines to the caller.
	LINEPRT	PR	Controls title lines, headings, spacing; translates data lines; and calls STACKPUT.
IDCTP04			Initialize and modify PCT; set up all page controls, define headings and footings, and define format of page.
	IDCTP04	EP	Only entry point to this module.
	ESTSCONT	PR	Get space for PCT and initialize it (UESTS macro).
	ESTACONT	PR	Get space for PCT and initialize it from storage parameters (UESTA macro).
	P04SETUP	PR	Set up working table for PCT initialization.
	RESTCONT	PR	Initialize working table for modifying existing PCT (UREST macro).
	PCTSETUP	PR	Verify and initialize elements of PCT.
	RESETCON	PR	Re-initialize Text Processor for next function, return page number, and clear PCT.
	INITPCT	PR	Get and initialize PCT.
	STACKFL	PR	Print lines in stack buffer.
IDCTP05			Read Text Structures into storage for use by either IDCTP01 or IDCTP04.
	IDCTP05	EP	Only entry point to this module.
IDCTP06			Formats error messages for any FSR.
	IDCTP06	EP	Only entry point to this module.
IDCTSAL0			Text Structure for ALTER messages.
IDCTSBI0			Text Structure for BLDINDEX message.
IDCTSDE0			Text Structure for DEFINE messages.
IDCTSDL0			Text Structure for DELETE messages.
IDCTSEX0			Text Structure for Executive routines messages.
IDCTSIO0			Text Structure for I/O Adapter routines messages.
IDCTSLC0			Text Structure for LISTCAT listing.
IDCTSLC1			Text Structure for LISTCAT messages.
IDCTSLR0			Text Structure for LISTCRA listing.
IDCTSLR1			Text Structure for LISTCRA messages.
IDCTSMP0			Text Structure for IMPORT and IMPORTRA messages.
IDCTSPR0			Text Structure for PRINT listings and PRINT/REPRO messages.
IDCTSRC0			Text Structure for EXPORTRA messages.

CSECT/Load Module Name	Label	Use	Description
IDCTSRI0			Text Structure for Reader/Interpreter routines messages.
IDCTSRS0			Text structure for RESETCAT messages.
IDCTSTP0			Text Structure for Text Processor routines; contains print chain definitions.
IDCTSTP1			Text Structure for Text Processor routines messages.
IDCTSTP6			Text Structure for VERROR messages.
IDCTSUV0			Text Structure for any routine (universal messages).
IDCTSXP0			Text Structure for EXPORT messages.
IDCVY01			VERIFY FSR; check a VSAM data set against its catalog entries and correct any discrepancies that may be found, by calling IDCIOVR (UVERIFY macro).
	IDCVY01	EP	Only entry point to this module.
	OPENPROC	PR	Opens the VSAM data set to be verified.
	TERMPROC	PR	Closes the VSAM data set that was verified.
IDCXP01			EXPORT FSR; create a portable copy of a VSAM cluster or alternate index. Copy is done by issuing a UCOPY macro. When the input data set is a catalog, no copy is performed. Instead, the catalog is disconnected by a call to IDCSACA.
	IDCXP01	EP	Only entry point to this module.
	CLUSPROC	PR	Gets catalog information and data for a cluster object and calls CONTRBL to write all the information to a portable volume. Processes the disposition options. If it is a permanent option, the cluster will be deleted. If it is a temporary option, the temporary export flag is turned on by issuing a catalog alter.
	DSCTPROC	PR	Disconnects a user catalog.
	LOCPROC	PR	Builds a CTGPL and multiple CTGFLs for use by catalog locate. CTGFLs used to locate catalog information to be exported.
	CTLGPROC	PR	Invokes the VSAM catalog management to perform the operation indicated in the CTGPL.
	OPENPROC	PR	Performs all opens required for opening a VSAM cluster for input or opening the portable volume for output.
	ALTRPROC	PR	Constructs the CTGPL and CTGFV for a catalog alter operation so that the data set attributes catalog field (DSATTR) can be modified.
	DELTPROC	PR	Constructs a CTGPL for a catalog delete operation so that a cluster or alternate index can be deleted or a user catalog disconnected. Invokes VSAM catalog management to delete clusters or alternate indexes.
	PUTPROC	PR	Writes a catalog record to the portable volume.
	RECPROC	PR	Copies the data from the VSAM cluster to be exported to the portable data set, record by record.
	MVDAPROC	PR	Copies data from one part of virtual storage to another or, optionally, zeros out part of virtual storage.
	CONTRBL	PR	Writes catalog information to a portable volume.
	MORESP	PR	Obtains a larger work area for VSAM catalog management and reinvokes catalog.

Chapter 5: 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.

Block List (BLKLIST)

The Block List contains addresses and offsets for each data block to be used by the text processor block data routine when one more than one data block is required.

Created by	Modified by	Used by	Size
Calling routine	IDCTP01	IDCTP01	Variable

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8×*n*	BLKLARY	The following fields are repeated **n** times, where **n** equals the number of data blocks being used. The FMTBLKNO field of a block data format list is used as the index into this array.
	2	BLKLRIO	Offset to add to all offsets contained in block-format sub-structures.
2 (2)	..2	BLKLILP	Length of block whose address is in BLKLPTR.
4 (4)	4	BLKLPTR	Address of a block of data.

Buffer Pool Control Block (BUFS)

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

Buffer Pool Control Block Description

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.

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. IDCRILT is loaded upon the first entry to IDCRI01.

The name of each load module and the corresponding verb, as supplied by IBM, is as follows:

IDCCDAL ALTER	IDCCDRC EXPORTRA	IDCCDPM PARM
IDCCDBI BLDINDEX	IDCCDMP IMPORT	IDCCDPR PRINT
IIDCCDCL CANCEL	IDCCDRM IMPORTRA	IDCCDRP REPRO
IDCCDDE DEFINE	IDCCDLC LISTCAT	IDCCDRS RESETCAT
IDCCDDL DELETE	IDCCDLR LISTCRA	IDCCDVY VERIFY
IDCCDXP EXPORT		

Each Command Descriptor consists of a series of variable-length entries. The first entry is always the verb-data entry, which names the FSR load module to use. Subsequent entries define default values, syntactic and semantic requirements, the structure of all possible parameters, and the structure of the FDT to be built from this command.

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCRI01	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	Not used in VSE.
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	LOAD NAME	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)	2x*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:
	2x*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:
	2x*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:
	2x*n*	NTGRPID*n*	List of ID numbers.

Parameter Data Area

The Parameter Data Area follows the Verb Data Area, and describes 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	Not used in VSE.
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	Not used in VSE.
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	Not used in VSE.
	.... 1...	VOLID	Indicates a volume serial number may replace a data set name.
	.... .1..	USERID	Not used in VSE.
	.... ..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	Not used in VSE.

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 tha 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)	2x*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.
	2x*n*	NGRPID*n*	List of parameter ID numbers for necessary parameters.

Prompt Appendage

This appendage, although it can be present in VSE, is not used. It 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.
Each set contains:			
	2	PDEPRMID	Not used.
	2	PDEPCLID	Not used.
	2	PDEOFST2	Not used.

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.

Command Descriptor Phase Table—IDCRILT

IDCRILT contains a table of all verbs accepted by the processor and the Command Descriptor phase names that are required to parse them.

Created by	Modified by	Used by	Size
IBM-Supplied	None	IDCRI02	258

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	2	LNAMECNT	Number of table entries.
2(2)	16×*n*		*n* table entries.
	8	TBIVERB	Verb character string.
	8	TBILNAME	Corresponding Command Descriptor phase name.
16×*n*	8	FFFF	End-of-table indicator (set to C'FF ').

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 label 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	IDCDB01	IDCDB02	Variable

Individual Field Entry

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	DMPIMNM	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	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 .	DMPARYTP	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	DMPTRM	End of dump list indicator—contains X'FF'.

Dynamic Data List—DARGLIST

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 the block of data, the address of the BLKLIST, or zero.
4 (4)	4	DARGRETP	Zero if printing is to occur; nonzero if no printing is to occur. If nonzero, contains the address of the area in which the formatted print lines are to be returned from the Text Processor (and not printed). Data will be returned to the specified location. The data is truncated to the length (DARGRETL) of the provided area if necessary. Spacing control characters are not returned.
8 (8)	4	DARGSTID	Zero if a format list is also passed as a parameter. If nonzero, 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).

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
19 (13)	. . . 1	DARGFLGS	DARGLIST flags:
	. 1	DARGBPL	DARGDBP contains the address of the BLKLIST, which contains addresses of multiple data blocks.
	. 0		DARGDBP contains the address of a single data block referred to by the format list.
	. . 1	DARGFUL	Output recordsize is greater than 32K.
	x . . x xxxx		Reserved.
20 (14)	8×*n*	DARGARY	Group array. The following fields are repeated *n* times, where n *n* = DARGCNT.
	2	DARGINS	Insert reference number.
		DARGREP	Replication reference number.
	. . 2	DARGINL	Input data length of the field pointed to by DARGDTM.
		DARGPCT	Replication count, number of times to replicate a series of format substructures (FMTLIST).
	4	DARGDTM	Dynamic data pointer, address of field to use for this insert. This field is not used for replication structures.

Error Conversion Table—ERCNVTAB

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. Not used in VSE.
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. Not used in VSE.
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	VSAM Catalog Management return code.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
20 (14)	4	ERCPLPT	Address of Catalog Parameter List (CTGPL) issued that resulted in error condition.
24 (18)	4		Reserved.
28 (1C)	4		Reserved.

Field Management Parameter List—FMPL

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 or 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) of Length/Data pointer pairs.
	4.	FMFLENTH	4-byte length of supplied data.
	.4	FMFLADDR	4-byte address of supplied data.

Format List—FMTLIST

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	IDCTP91	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 printing of the previously formatted data 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'.
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	FMTBLKNO	Block number (starting with 0). This value is used as the index into the BLKLIST array for more than one data block.
2 (2)	. . 2	FMTRFNO	Insert reference number: identification number for dynamic data insert that defines the input data to be used for formatting.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
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. (PCTAPC is the last non-blank in the previous field.)
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	Conversion flags (continued).
	1...	FMTZS	Suppress leading zeros by replacing with blanks.
	.1..	FMTAL	Aligned left; the high-order nonzero digit is put in first print column as specified by FMTCOL.
	..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 EDCDIC 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 the values are not equivalent, 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	FMTBLKNO	Block number (starting with 0). This value is used as the index into the BLKLIST array for more than one data block.
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 indicates 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 or column offset, 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.

Function Data Table—FDT

The Function Data Table is an encoded representation of a command. The Reader/Interpreter parses a command and constructs the FDT from information found in that 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 results. Figure 5-1 shows this vector and illustrates 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.

There are seven possible data formats for FDT entries. Each type is described below; the data format number corresponds to the number in the "Data Format Number" column in the descriptions of the various FDTs.

Data Format Number	Type	Sub-field Level	Subfield Description	Mode	Length (bytes)	Notes
1	pointer			pointer	4	May point to data, to itself (indicating that the parameter is specified), or may be binary zeros (parameter is not specified).
2	character					
		1	length	binary	1	
		1	value	character string	variable	
3	numeric value			binary	4	
4	character list					
		1	number of items in list	binary	2	
		1	for each item:			
		2	length	binary	1	
		2	value	character	variable	
5	binary word list					
		1	number of items in list	binary	2	
		1	for each item:			
		2	length	binary	1	
		2	value	binary	4	
6	name					
		1	password length	binary	1	
		1	password	characters	8	
		1	asterisk	binary	1	Not used.
		1	name flag	bit string	1	Bit 0 on = unqualified name; bit 0 off = qualified name.
		1	member name length	binary	1	Unused.
		1	member name	characters	8	Unused.
		1	name length	binary	1	
		1	name value	characters	44	
7	*dname/ password*					
		1	password length	binary	1	
		1	password	characters	8	
		1	dname length	binary	1	
		1	dname	characters	8	

Figure 5-1 shows the FDT mapping for IMPORT when the following parameters are specified:

```
IMPORT  INFILE(SOURCE -
   ENV (PDEV(2400) BLKSIZE(6000) REWIND)) -
   OUTPW(RECPW) -
   PURGE -
   OBJECTS( -
     (EXAMPLE.KSDS1 -
        USECLASS(7 P) -
        KEYRANGES( -
           (ABBOTT GESTNER) -
           (GESTRICH MERCY) -
           (MESHING ZUBRINSKI) -
                 ) -
      ) -
     (EXAMPLE.KSDS1.DATA -
        NEWNAME(EXAMNEW.KSDS1.DATA) -
        VOLUMES(VSER03,VSER04) -
        ORDERED -
        KEYRANGES( -
           (ABBOTT MERCY) -
           (MESHING ZUBRINSKI) -
                 ) -
        FILE(D1) -
        ) -
     (EXAMPLE.KSDS1.INDEX -
        VOLUMES(VSER03,VSER04,VSER05) -
        FILE(D2) -
        ) -
        )
```

The first five columns in the FDT descriptions are self-explanatory. The last three columns have the following meanings:

Points to data—information supplied by the specified parameter.
itself—address of the pointer itself if the parameter has been specified.
list—additional information is given in the "Notes" column.

Data Format Number corresponds to Data Format 1-7, described above.

Notes additional information and references to subparameters of the specified parameter.

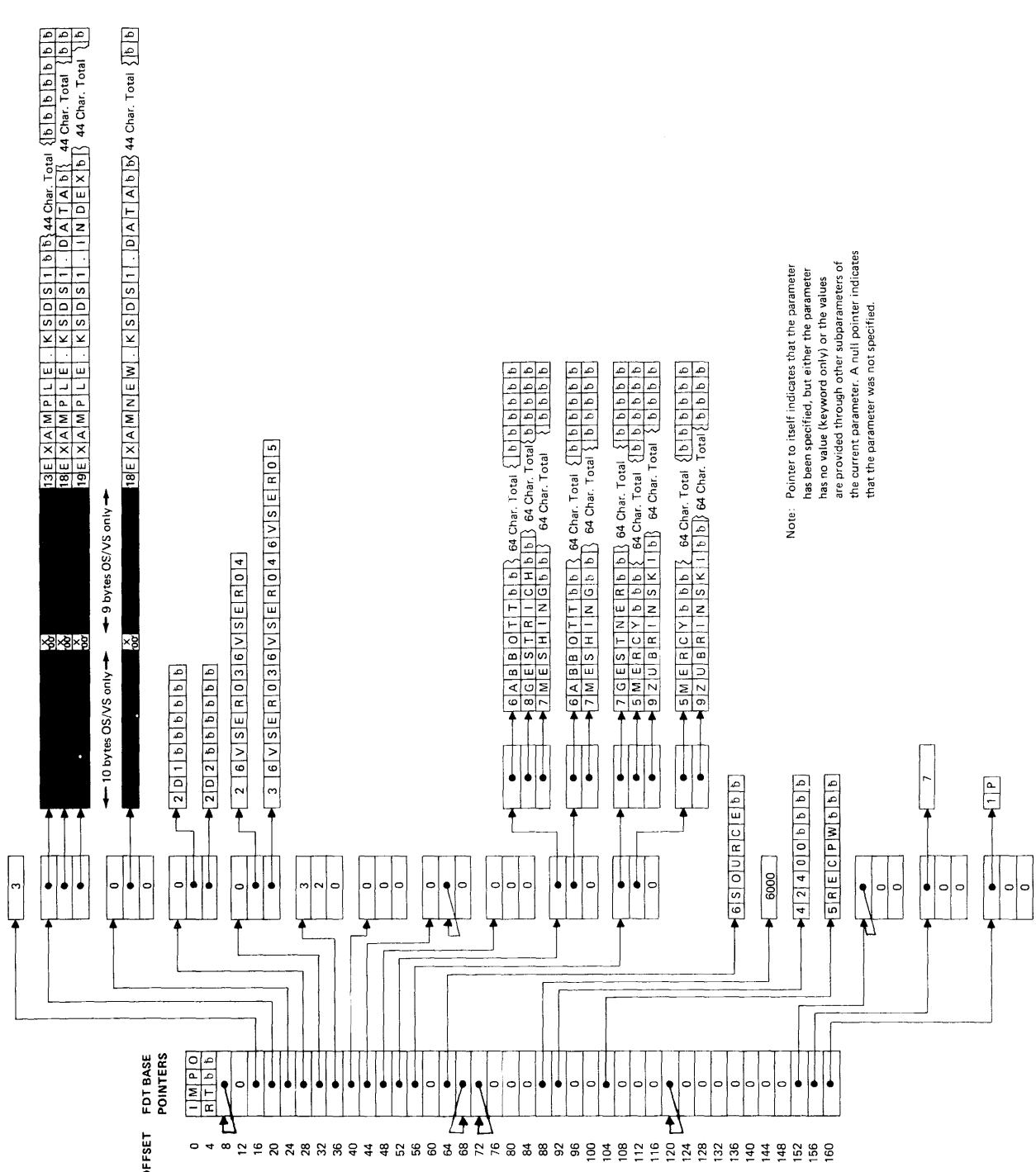


Figure 5-1. IMPORT FDT Mapping

ALTER FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			ALTERbbb
1	8 (8)	*entryname/ password*		ENTRY	data	6	
2	12 (C)	CATALOG		CAT	itself	1	See parms 3 and 4.
3	16 (10)	*catname/ password*	CATALOG	CATLG	data	6	
4	20 (14)	*dname*	CATALOG	CATDN	data	2	
5	24 (18)	NEWNAME		NEWNM	data	6	
6	28 (1C)	FILE		INDD	data	2	
7	32 (20)	unused - contains zeros					
8	36 (24)	MASTERPW		MASTR	data	2	
9	40 (28)	CONTROLPW		CNTVL	data	2	
10	44 (2C)	UPDATEPW		UPDAT	data	2	
11	48 (30)	READPW		READ	data	2	
12	52 (34)	CODE		CODNM	data	2	
13	56 (38)	ATTEMPTS		ATTP	data	3	
14	60 (3C)	AUTHORIZATION		AUTH	itself	1	See parms 15 and 16.
15	64 (40)	*entrypoint*	AUTHORIZATION	USVR	data	2	
16	68 (44)	*string*	AUTHORIZATION	USAR	data	2	
17	72 (48)	unused - contains zeros					
18	76 (4C)	TO		TO	data	3	
19	80 (50)	FOR		FOR	data	3	
20	84 (54)	OWNER		OWNER	data	2	
21	88 (58)	ERASE		ERASE	itself	1	
22	92 (5C)	NOERASE		NERAS	itself	1	
23	96 (60)	SHAREOPTIONS		SHARE	itself	1	See parms 48 and 49.
24	100 (64)	unused - contains zeros					
25	104 (68)	NULLIFY		NULLF	itself	1	See parms 26-29, 42-45, 55, and 68.
26	108 (6C)	MASTERPW	NULLIFY	NMSTR	itself	1	
27	112 (70)	CONTROLPW	NULLIFY	NCNTV	itself	1	
28	116 (74)	UPDATEPW	NULLIFY	NUPDT	itself	1	
29	120 (78)	READPW	NULLIFY	NREAD	itself	1	
30	124 (7C)	unused - contains zeros					
31	128 (80)	FREESPACE		FSPAC	itself	1	See parms 32 and 33.
32	132 (84)	*cipercent*	FREESPACE	FSPCI	data	3	
33	136 (88)	*capercent*	FREESPACE	FSPCA	data	3	
34	140 (8C)	WRITECHECK		WRTCK	itself	1	
35	144 (90)	NOWRITECHECK		NWTCK	itself	1	
36	148 (94)	BUFFERSPACE		BUFSZ	data	3	
37	152 (98)	ADDVOLUMES		ADDVL	list	4	For each item in the list, there is a list of volume serial numbers.
38	156 (9C)	REMOVEVOLUMES		REMVL	list	4	For each item in the list, there is a list of volume serial numbers.
39	160 (A0)	unused - contains zeros					
40	164 (A4)	INHIBIT		INHIB	itself	1	
41	168 (A8)	UNINHIBIT		UNHIB	itself	1	
42	172 (AC)	OWNER	NULLIFY	NOWNR	itself	1	
43	176 (B0)	CODE	NULLIFY	NCDNM	itself	1	
44	180 (B4)	RETENTION	NULLIFY	NRETN	itself	1	
45	184 (B8)	AUTHORIZATION	NULLIFY	NAUTH	itself	1	See parms 46 and 47.
46	188 (BC)	MODULE	NULLIFY, AUTHORIZATION	NMDNM	itself	1	
47	192 (C0)	STRING	NULLIFY, AUTHORIZATION	NSTRG	itself	1	
48	196 (C4)	*crosspartition/ value*	SHAREOPTIONS	SHAR1	data	3	
49	200 (C8)	reserved for OS	SHAREOPTIONS	SHAR2	data	3	
50	204 (CC)	unused - contains zeros					

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
51	208 (D0)	unused - contains zeros					
52	212 (D4)	unused - contains zeros					
53	216 (D8)	unused - contains zeros					
54	220 (DC)	unused - contains zeros					
55	224 (E0)	EXCEPTIONEXIT		EEXT	data	2	
56	228 (E4)	KEYS		KEY	itself	1	See parms 57 and 58.
57	232 (E8)	*length*	KEYS	KEYLN	data	3	
58	236 (EC)	*offset*	KEYS	KEYPS	data	3	
59	240 (F0)	RECORDSIZE		RECSZ	itself	1	See parms 60 and 61.
60	244 (F4)	*average*	RECORDSIZE	AREC	data	3	
61	248 (F8)	*maximum*	RECORDSIZE	MREC	data	3	
62	252 (FC)	UNIQUEKEY		UNQK	itself	1	
63	256 (100)	NONUNIQUEKEY		NUNQK	itself	1	
64	260 (104)	UPGRADE		UPG	itself	1	
65	264 (108)	NOUPGRADE		NUPG	itself	1	
66	268 (10C)	UPDATE		UPD	itself	1	
67	272 (110)	NOUPDATE		NUPD	itself	1	
68	276 (114)	EXCEPTIONEXIT	NULLIFY	NEEXT	itself	1	

BLDINDEX FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			BLDINDEX
1	8 (8)	INFILE		IFILE	data	7	
2	12 (C)	unused - contains zeros					
3	16 (10)	OUTFILE		OFILE	data	4/7	Count of number of dnames followed by the list of *dname/passwords* in data format 7.
4	20 (14)	unused - contains zeros					
5	24 (18)	*catname/ password*	CATALOG	CAT	data	6	
6	28 (1C)	WORKFILES		WFILE	itself	1	See parms 7 and 8.
7	32 (20)	*dname1*	WORKFILES	WFLE1	data	2	
8	36 (24)	*dname2*	WORKFILES	WFLE2	data	2	
9	40 (28)	EXTERNALSORT		ESORT	itself	1	
10	44 (2C)	INTERNALSORT		ISORT	itself	1	
11	48 (30)	INDATASET		IDS	data	6	base cluster data set name with optional password.
12	52 (34)	OUTDATASET		ODS	data	4/6	AIX data set names with optionalpasswords.
13	56 (38)	WORKVOLUMES		WVOL	data	4	list of VOLIDs for sort word volumes (CHAR(6)).

CANCEL FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			CANCEL
1	8 (8)	JOB		JOB	itself	1	
2	12 (C)	STEP		STEP	itself	1	

DEFINE FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			DEFINEbbb
1	8 (8)	CATALOG		CAT	itself	1	See parms 2 and 3.
2	12 (C)	*catname/ password*	CATALOG	CATLG	data	6	
3	16 (10)	*dname*	CATALOG	CATDN	data	2	
4	20 (14)	MASTERCATALOG		MCAT	itself	1	See parms 16, 39, 43, 47, 51, 55, 59, 63, 73, 75, 77, 108, 113, 139, 142, 145, 148, 168, 170, 186, 276, 279, 283, 284, 408, 433, and 469.
5	24 (18)	USERCATALOG		UCAT	itself	1	See parms 26, 149, 198-215, 218-220, 277, 280, 285, 286, 409, 434, and 470.
6	28 (1C)	CLUSTER		CLST	itself	1	See parms 17, 24, 25, 27, 40, 44, 48, 52, 56, 60, 64, 74, 76, 78, 81, 90, 91, 94, 100, 101, 104, 105, 109, 114, 119, 127, 128, 133, 136, 140, 143, 146, 161, 169, 171, 176, 177, 180, 187, 189, 192, 221, 258, 262, 265, 266, 269, 272, 274, 431, and 473.
7	32 (20)	unused - contains zeros					
8	36 (24)	DATA		DATA	itself	1	See parms 22, 31, 41, 45, 49, 53, 57, 61, 67, 79, 84, 92, 93, 97, 111, 117, 122, 129, 130, 134, 137, 150-152, 165, 172, 173, 178, 179, 183, 188, 190, 193, 222, 259, 263, 267, 270, 273, 275, 278, 281, 403, 404, 437, and 475.
9	40 (28)	INDEX		INDEX	itself	1	See parms 23, 35, 42, 46, 50, 54, 58, 62, 70, 80, 87, 102, 103, 106, 107, 112, 118, 131, 132, 135, 138, 155-157, 174, 175, 191, 194, 260, 264, 268, 271, 438, and 476.
10	44 (2C)	SPACE		SPACE	itself	1	See parms 110, 115, 141, 144, 147, 160, 162, 196, 407, 435, and 471.
11	48 (30)	NONVSAM		ALIEN	itself	1	See parms 19, 116, 125, 126 and 282.
12	52 (34)	unused - contains zeros					
13	56 (38)	unused - contains zeros					
14	60 (3C)	ALTERNATEINDEX		AIX	itself	1	See parms 195, 261, 338-402, 405, 406, 432, and 477.
15	64 (40)	PATH		PATH	itself	1	See parms 410-430.
16	68 (44)	NAME	MASTERCATALOG	METRY	data	6	
17	72 (48)	NAME	CLUSTER	CETRY	data	6	
18	76 (4C)	unused					
19	80 (50)	NAME	NONVSAM	AETRY	data	6	
20	84 (54)	unused					
21	88 (58)	unused					
22	92 (5C)	NAME	DATA	DETRY	data	6	
23	96 (60)	NAME	INDEX	IETRY	data	6	
24	100 (64)	INDEXED	CLUSTER	CINDX	itself	1	
25	104 (68)	NONINDEXED	CLUSTER	CNIDX	itself	1	
26	108 (6C)	MODEL	USERCATALOG	UMODL	itself	1	See parms 253-255.
27	112 (70)	MODEL	CLUSTER	CMODL	itself	1	See parms 28-30.
28	116 (74)	*entryname/ password*	CLUSTER, MODEL	CENAM	data	6	
29	120 (78)	*catname/ password*	CLUSTER, MODEL	CMDCT	data	6	
30	124 (7C)	*dname*	CLUSTER, MODEL	CMDNM	data	2	
31	128 (80)	MODEL	DATA	DMODL	itself	1	See parms 32-34.
32	132 (84)	*entryname/ password*	DATA, MODEL	DENAM	data	6	
33	136 (88)	*catname/ password*	DATA, MODEL	DMDCT	data	6	
34	140 (8C)	*dname*	DATA, MODEL	DMDNM	data	2	
35	144 (90)	MODEL	INDEX	IMODL	itself	1	See parms 36-38.
36	148 (94)	*entryname/ password*	INDEX, MODEL	IENAM	data	6	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
37	152 (98)	*catname/ password*	INDEX, MODEL	IMDCT	data	6	
38	156 (9C)	*dname*	INDEX, MODEL	IMDNM	data	2	
39	160 (A0)	MASTERPW	MASTERCATALOG	MMSTR	data	2	
40	164 (A4)	MASTERPW	CLUSTER	CMSTR	data	2	
41	168 (A8)	MASTERPW	DATA	DMSTR	data	2	
42	172 (AC)	MASTERPW	INDEX	IMSTR	data	2	
43	176 (B0)	CONTROLPW	MASTERCATALOG	MCINT	data	2	
44	180 (B4)	CONTROLPW	CLUSTER	CCINT	data	2	
45	184 (B8)	CONTROLPW	DATA	DCINT	data	2	
46	188 (BC)	CONTROLPW	INDEX	ICINT	data	2	
47	192 (C0)	UPDATEPW	MASTERCATALOG	MUPDT	data	2	
48	196 (C4)	UPDATEPW	CLUSTER	CUPDT	data	2	
49	200 (C8)	UPDATEPW	DATA	DUPDT	data	2	
50	204 (CC)	UPDATEPW	INDEX	IUPDT	data	2	
51	208 (D0)	READPW	MASTERCATALOG	MREAD	data	2	
52	212 (D4)	READPW	CLUSTER	CREAD	data	2	
53	216 (D8)	READPW	DATA	DREAD	data	2	
54	220 (DC)	READPW	INDEX	IREAD	data	2	
55	224 (E0)	CODE	MASTERCATALOG	MCODE	data	2	
56	228 (E4)	CODE	CLUSTER	CCODE	data	2	
57	232 (E8)	CODE	DATA	DCODE	data	2	
58	236 (EC)	CODE	INDEX	ICODE	data	2	
59	240 (F0)	ATTEMPTS	MASTERCATALOG	MATTP	data	3	
60	244 (F4)	ATTEMPTS	CLUSTER	CATTP	data	3	
61	248 (F8)	ATTEMPTS	DATA	DATTP	data	3	
62	252 (FC)	ATTEMPTS	INDEX	IATTP	data	3	
63	256 (100)	AUTHORIZATION	MASTERCATALOG	MAUTH	itself	1	See parms 65 and 66.
64	260 (104)	AUTHORIZATION	CLUSTER	CAUTH	itself	1	See parms 256 and 257.
65	264 (108)	*entrypoint*	MASTERCATALOG, AUTHORIZATION	MEPNM	data	2	
66	268 (10C)	*string*	MASTERCATALOG, AUTHORIZATION	MSTRG	data	2	
67	272 (110)	AUTHORIZATION	DATA	DAUTH	itself	1	See parms 68 and 69.
68	276 (114)	*entrypoint*	DATA, AUTHORIZATION	DEPNM	data	2	
69	280 (118)	*string*	DATA, AUTHORIZATION	DSTRG	data	2	
70	284 (11C)	AUTHORIZATION	INDEX	IAUTH	itself	1	See parms 71 and 72.
71	288 (120)	*entrypoint*	INDEX, AUTHORIZATION	IEPNM	data	2	
72	292 (124)	*string*	INDEX, AUTHORIZATION	ISTRG	data	2	
73	296 (128)	TO	MASTERCATALOG	MTO	data	3	
74	300 (12C)	TO	CLUSTER	CTO	data	3	
75	304 (130)	FOR	MASTERCATALOG	MFOR	data	3	
76	308 (134)	FOR	CLUSTER	CFOR	data	3	
77	312 (138)	OWNER	MASTERCATALOG	MOWNR	data	2	
78	316 (13C)	OWNER	CLUSTER	COWNR	data	2	
79	320 (140)	OWNER	DATA	DOWNR	data	2	
80	324 (144)	OWNER	INDEX	IOWNR	data	2	
81	328 (148)	SHAREOPTIONS	CLUSTER	CSHAR	itself	1	See parms 82 and 83.
82	332 (14C)	*crosspartition/ value*	CLUSTER, SHAREOPTIONS	CSHR1	data	3	
83	336 (150)	reserved for OS	CLUSTER, SHAREOPTIONS	CSHR2	data	3	
84	340 (154)	SHAREOPTIONS	DATA	DSHAR	itself	1	See parms 85 and 86.
85	344 (158)	*crosspartition/ value*	DATA, SHAREOPTIONS	DSHR1	data	3	
86	348 (15C)	reserved for OS	DATA, SHAREOPTIONS	DSHR2	data	3	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
87	352 (160)	SHAREOPTIONS	INDEX	ISHAR	itself	1	See parms 88 and 89.
88	356 (164)	*crosspartition/ value*	INDEX, SHAREOPTIONS	ISHR1	data	3	
89	360 (168)	reserved for OS	INDEX, SHAREOPTIONS	ISHR2	data	3	
90	364 (16C)	ERASE	CLUSTER	CERAS	itself	1	
91	368 (170)	NOERASE	CLUSTER	CNERS	itself	1	
92	372 (174)	ERASE	DATA	DERAS	itself	1	
93	376 (178)	NOERASE	DATA	DNERS	itself	1	
94	380 (17C)	KEYS	CLUSTER	CKEY	itself	1	See parms 95 and 96.
95	384 (180)	*length*	CLUSTER, KEYS	CKYLN	data	3	key length
96	388 (184)	*offset*	CLUSTER, KEYS	CKYPS	data	3	key offset
97	392 (188)	KEYS	DATA	DKEY	itself	1	See parms 98 and 99.
98	396 (18C)	*length*	DATA, KEYS	DKYLN	data	3	key length
99	400 (190)	*offset*	DATA, KEYS	DKYPS	data	3	key offset
100	404 (194)	REPLICATE	CLUSTER	CREPL	itself	1	
101	408 (198)	NOREPLICATE	CLUSTER	CNREP	itself	1	
102	412 (19C)	REPLICATE	INDEX	IREPL	itself	1	
103	416 (1A0)	NOREPLICATE	INDEX	INREP	itself	1	
104	420 (1A4)	IMBED	CLUSTER	CIMBD	itself	1	
105	424 (1A8)	NOIMBED	CLUSTER	CNIBD	itself	1	
106	428 (1AC)	IMBED	INDEX	IIMBD	itself	1	
107	432 (1B0)	NOIMBED	INDEX	INIBD	itself	1	
108	436 (1B4)	FILE	MASTERCATALOG	MINDD	data	2	*dname*
109	440 (1B8)	FILE	CLUSTER	CINDD	data	2	*dname*
110	444 (1BC)	FILE	SPACE	SINDD	data	2	*dname*
111	448 (1C0)	FILE	DATA	DINDD	data	2	*dname*
112	452 (1C4)	FILE	INDEX	IINDD	data	2	*dname*
113	456 (1C8)	VOLUMES	MASTERCATALOG	MVSER	data	4	A single serial number (character 6).
114	560 (1CC)	VOLUMES	CLUSTER	CVSER	data	4	A list of volume serial numbers (character 6).
115	464 (1D0)	VOLUMES	SPACE	SVSER	data	4	A list of volume serial numbers (character 6).
116	468 (1D4)	VOLUMES	NONVSAM	AVSER	data	4	A list of volume serial numbers (character 6).
117	472 (1D8)	VOLUMES	DATA	DVSER	data	4	A list of volume serial numbers (character 6).
118	476 (1DC)	VOLUMES	INDEX	IVSER	data	4	A list of volume serial numbers (character 6).
119	480 (1E0)	KEYRANGES	CLUSTER	CRANG	data	3	Count of sub-parms. See parms 120 and 121.
120	484 (1E4)	*lowkey*	CLUSTER, KEYRANGES	CRGLOPTR	list of pointers	2	Each pointer points to a low keyrange value.
121	488 (1E8)	*highkey*	CLUSTER, KEYRANGES	CRGHIPTR	list of pointers	2	Each pointer points to a high keyrange value.
122	492 (1EC)	KEYRANGES	DATA	DRANG	data	3	Count of sub-parms. See parms 123 and 124.
123	496 (1F0)	*lowkey*	DATA, KEYRANGES	DRGLOPTR	list of pointers	2	Each pointer points to a low keyrange value.
124	500 (1F4)	*highkey*	DATA, KEYRANGES	DRGHIPTR	list of pointers	2	Each pointer points to a high keyrange value.
125	504 (1F8)	DEVICETYPES	NONVSAM	ADEVT	data	4	A list of device types (character 8).
126	508 (1FC)	FILESEQUENCENO	NONVSAM	AFSNO	data	5	A list of file sequence numbers.
127	512 (200)	ORDERED	CLUSTER	CORDR	itself	1	
128	516 (204)	UNORDERED	CLUSTER	CUORD	itself	1	
129	520 (208)	ORDERED	DATA	DORDR	itself	1	
130	524 (20C)	UNORDERED	DATA	DUORD	itself	1	
131	528 (210)	ORDERED	INDEX	IORDR	itself	1	
132	532 (214)	UNORDERED	INDEX	IUORD	itself	1	
133	536 (218)	SUBALLOCATION	CLUSTER	CSUBA	itself	1	
134	540 (21C)	SUBALLOCATION	DATA	DSUBA	itself	1	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
135	544 (220)	SUBALLOCATION	INDEX	ISUBA	itself	1	
136	548 (224)	UNIQUE	CLUSTER	CUNIQ	itself	1	
137	552 (228)	UNIQUE	DATA	DUNIQ	itself	1	
138	556 (22C)	UNIQUE	INDEX	IUNIQ	itself	1	
139	560 (230)	TRACKS	MASTERCATALOG	MTRKS	itself	1	See parms 300 and 301.
140	564 (234)	TRACKS	CLUSTER	CTRKS	itself	1	See parms 302 and 303.
141	568 (238)	TRACKS	SPACE	STRKS	itself	1	See parms 304 and 305.
142	572 (23C)	CYLINDERS	MASTERCATALOG	MCYLD	itself	1	See parms 310 and 311.
143	576 (240)	CYLINDERS	CLUSTER	CCYLD	itself	1	See parms 312 and 313.
144	580 (244)	CYLINDERS	SPACE	SCYLD	itself	1	See parms 318 and 319.
145	584 (248)	RECORDS	MASTERCATALOG	MRCDS	itself	1	See parms 320 and 321.
146	588 (24C)	RECORDS	CLUSTER	CRCDS	itself	1	See parms 322 and 323.
147	592 (250)	RECORDS	SPACE	SRCDS	itself	1	See parms 324 and 325.
148	596 (254)	ORIGIN	MASTERCATALOG	MORIG	data	3	
149	600 (258)	ORIGIN	USERCATALOG	UORIG	data	3	
150	604 (25C)	TRACKS	DATA	DTRKS	itself	1	See parms 153 and 154.
151	608 (260)	CYLINDERS	DATA	DCYLD	itself	1	See parms 330 and 331.
152	612 (264)	RECORDS	DATA	DRCDS	itself	1	See parms 332 and 333.
153	616 (268)	*primary*	DATA, TRACKS	DTKPR	data	3	
154	620 (26C)	*secondary*	DATA, TRACKS	DTKSC	data	3	
155	624 (270)	TRACKS	INDEX	ITRKS	itself	1	See parms 158 and 159.
156	628 (274)	CYLINDERS	INDEX	ICYLD	itself	1	See parms 334 and 335.
157	632 (278)	RECORDS	INDEX	IRCDS	itself	1	See parms 336 and 337.
158	636 (27C)	*primary*	INDEX, TRACKS	ITKPR	data	3	
159	640 (280)	*secondary*	INDEX, TRACKS	ITKSC	data	3	
160	644 (284)	CANDIDATE	SPACE	SCAND	itself	1	
161	648 (288)	RECORDSIZE	CLUSTER	CRSIZ	itself	1	See parms 163 and 164.
162	652 (28C)	RECORDSIZE	SPACE	SRSIZ	itself	1	See parms 251 and 252.
163	656 (290)	*average*	CLUSTER, RECORDSIZE	CARSZ	data	3	
164	660 (294)	*maximum*	CLUSTER, RECORDSIZE	CMRSZ	data	3	
165	664 (298)	RECORDSIZE	DATA	DRSIZ	itself	1	See parms 166 and 167.
166	668 (29C)	*average*	DATA, RECORDSIZE	DARSZ	data	3	
167	672 (2A0)	*maximum*	DATA, RECORDSIZE	DMRSZ	data	3	
168	676 (2A4)	WRITECHECK	MASTERCATALOG	MWCK	itself	1	
169	680 (2A8)	WRITECHECK	CLUSTER	CWCK	itself	1	
170	684 (2AC)	NOWRITECHECK	MASTERCATALOG	MNWCK	itself	1	
171	688 (2B0)	NOWRITECHECK	CLUSTER	CNWCK	itself	1	
172	692 (2B4)	WRITECHECK	DATA	DWCK	itself	1	
173	696 (2B8)	NOWRITECHECK	DATA	DNWCK	itself	1	
174	700 (2BC)	WRITECHECK	INDEX	IWCK	itself	1	
175	704 (2C0)	NOWRITECHECK	INDEX	INWCK	itself	1	
176	708 (2C4)	SPEED	CLUSTER	CSPED	itself	1	
177	712 (2C8)	RECOVERY	CLUSTER	CRECV	itself	1	
178	716 (2CC)	SPEED	DATA	DSPED	itself	1	
179	720 (2D0)	RECOVERY	DATA	DRECV	itself	1	
180	724 (2D4)	FREESPACE	CLUSTER	CFSPC	itself	1	See parms 181 and 182.
181	728 (2D8)	*cipercent*	CLUSTER, FREESPACE	CCIFS	data	3	
182	732 (2DC)	*capercent*	CLUSTER, FREESPACE	CCAFS	data	3	
183	736 (2E0)	FREESPACE	DATA	DFSPC	itself	1	See parms 184 and 185.
184	740 (2E4)	*cipercent*	DATA, FREESPACE	DCIFS	data	3	
185	744 (2E8)	*capercent*	DATA, FREESPACE	DCAFS	data	3	
186	748 (2EC)	BUFFERSPACE	MASTERCATALOG	MBFSZ	data	3	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
187	752 (2F0)	BUFFERSPACE	CLUSTER	CBFSZ	data	3	
188	756 (2F4)	BUFFERSPACE	DATA	DBFSZ	data	3	
189	760 (2F8)	CONTROL-INTERVALSIZE	CLUSTER	CCINV	data	3	
190	764 (2FC)	CONTROL-INTERVALSIZE	DATA	DCINV	data	3	
191	768 (300)	CONTROL-INTERVALSIZE	INDEX	ICINV	data	3	
192	772 (304)	DEFAULTVOLUMES	CLUSTER	CDVCL	itself	1	
193	776 (308)	DEFAULTVOLUMES	DATA	DDVOL	itself	1	
194	780 (30C)	DEFAULTVOLUMES	INDEX	IDVOL	itself	1	
195	784 (310)	DEFAULTVOLUMES	ALTERNATEINDEX	GDVOL	itself	1	
196	788 (314)	ORIGIN	SPACE	SORIG	data	3	
197	792 (318)	unused - contains zeros					
198	796 (31C)	NAME	USERCATALOG	UETRY	data	6	
199	800 (320)	MASTERPW	USERCATALOG	UMSTR	data	2	
200	804 (324)	CONTROLPW	USERCATALOG	UCINT	data	2	
201	808 (328)	UPDATEPW	USERCATALOG	UUPDT	data	2	
202	812 (32C)	READPW	USERCATALOG	UREAD	data	2	
203	816 (330)	CODE	USERCATALOG	UCODE	data	2	
204	820 (334)	ATTEMPTS	USERCATALOG	UATTP	data	3	
205	824 (338)	AUTHORIZATION	USERCATALOG	UAUTH	itself	1	See parms 206 and 207.
206	828 (33C)	*entrypoint*	USERCATALOG, AUTHORIZATION	UEPNM	data	2	
207	832 (340)	*string*	USERCATALOG, AUTHORIZATION	USTRG	data	2	
208	836 (344)	TO	USERCATALOG	UTO	data	3	
209	840 (348)	FOR	USERCATALOG	UFOR	data	3	
210	844 (34C)	OWNER	USERCATALOG	UOWNR	data	2	
211	848 (350)	FILE	USERCATALOG	UINDD	data	2	*dname*
212	852 (354)	VOLUMES	USERCATALOG	UVSER	data	4	A single serial number (character 6).
213	856 (358)	TRACKS	USERCATALOG	UTRKS	itself	1	See parms 306 and 307.
214	860 (35C)	CYLINDERS	USERCATALOG	UCYLD	itself	1	See parms 314 and 315.
215	864 (360)	RECORDS	USERCATALOG	URCDS	itself	1	See parms 326 and 327.
216	868 (364)	unused - contains zeros					
217	872 (368)	unused - contains zeros					
218	876 (36C)	WRITECHECK	USERCATALOG	UWCK	itself	1	
219	880 (370)	NOWRITECHECK	USERCATALOG	UNWCK	itself	1	
220	884 (374)	BUFFERSPACE	USERCATALOG	UBFSZ	data	3	
221	888 (378)	RECORDFORMAT	CLUSTER	CRFMT	itself	1	See Parms 223-228
222	892 (37C)	RECORDFORMAT	DATA	DRFMT	itself	1	See Parms 229-234
223	896 (380)	UNDEF	RECORDFORMAT	CUNDF	itself	1	
224	900 (384)	FIXUNB	RECORDFORMAT	CFUNB	itself	1	
225	904 (388)	FIXBLK	RECORDFORMAT	CFBLK	data	3	
226	908 (38C)	VARUNB	RECORDFORMAT	CVUNB	itself	1	
227	912 (390)	VARBLK	RECORDFORMAT	CVBLK	itself	1	
228	916 (394)	NOCIFORMAT	RECORDFORMAT	CNCIF	itself	1	
229	920 (398)	UNDEF	RECORFORMAT	DUNDF	itself	1	
230	924 (39C)	FIXUNB	RECORDFORMAT	DFUNB	itself	1	
231	928 (3A0)	FIXBLK	RECORDFORMAT	DFBLK	data	3	
232	932 (3A4)	VARUNB	RECORDFORMAT	DVUNB	itself	1	
233	936 (3A8)	VARBLK	RECORDFORMAT	DVBLK	itself	1	
234	940 (3AC)	NOCIFORMAT	RECORDFORMAT	DNCIF	itself	1	
235	944 (3B0)	through					
250	1007 (3EF)	unused - contains zeros					
251	1008 (3F0)	*average*	SPACE, RECORDSIZE	SARSZ	data	3	
252	1012 (3F4)	*maximum*	SPACE, RECORDSIZE	SMRSZ	data	3	
253	1016 (3F8)	*entryname/*	USERCATALOG,	UENAM	data	6	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
		password	MODEL				
254	1020 (3FC)	*catname/ password*	USERCATALOG, MODEL	UMDCT	data	6	
255	1024 (400)	*dname*	USERCATALOG, MODEL	UMDNM	data	2	
256	1028 (404)	*entrypoint*	CLUSTER, AUTHORIZATION	CEPNM	data	2	
257	1032 (408)	*string*	CLUSTER, AUTHORIZATION	CSTRG	data	2	
258	1036 (40C)	NOALLOCATION	CLUSTER	CNOAL	itself	1	
259	1040 (410)	NOALLOCATION	DATA	DNOAL	itself	1	
260	1044 (414)	NOALLOCATION	INDEX	INOAL	itself	1	
261	1048 (418)	NOALLOCATION	ALTERNATEINDEX	GNOAL	itself	1	
262	1052 (41C)	EXCEPTIONEXIT	CLUSTER	CEEXT	data	2	
263	1056 (420)	EXCEPTIONEXIT	DATA	DEEXT	data	2	
264	1060 (424)	EXCEPTIONEXIT	INDEX	IEEXT	data	2	
265	1064 (428)	NUMBERED	CLUSTER	CNUMD	itself	1	
266	1068 (42C)	REUSE	CLUSTER	CRUS	itself	1	
267	1072 (430)	REUSE	DATA	DRUS	itself	1	
268	1076 (434)	REUSE	INDEX	IRUS	itself	1	
269	1080 (438)	NOREUSE	CLUSTER	CNRUS	itself	1	
270	1084 (43C)	NOREUSE	DATA	DNRUS	itself	1	
271	1088 (440)	NOREUSE	INDEX	INRUS	itself	1	
272	1092 (444)	SPANNED	CLUSTER	CSPND	itself	1	
273	1096 (448)	SPANNED	DATA	DSPND	itself	1	
274	1100 (44C)	NONSPANNED	CLUSTER	CNSPD	itself	1	
275	1104 (450)	NONSPANNED	DATA	DNSPD	itself	1	
276	1108 (454)	RECOVERABLE	MASTERCATALOG	MRVBL	itself	1	
277	1112 (458)	RECOVERABLE	USERCATALOG	URVBL	itself	1	
278	1116 (45C)	RECOVERABLE	DATA	DRVBL	itself	1	
279	1120 (460)	NOTRECOVERABLE	MASTERCATALOG	MNRVL	itself	1	
280	1124 (464)	NOTRECOVERABLE	USERCATALOG	UNRVL	itself	1	
281	1128 (468)	NOTRECOVERABLE	DATA	DNRVL	itself	1	
282	1132 (46C)	FILE	NONVSAM	AINDD	data	2	*dname*
283	1136 (470)	IMBED	MASTERCATALOG	MIMBD	itself	1	
284	1140 (474)	NOIMBED	MASTERCATALOG	MNIBD	itself	1	
285	1144 (478)	IMBED	USERCATALOG	UIMBD	itself	1	
286	1148 (47C)	NOIMBED	USERCATALOG	UNIBD	itself	1	
287 299	1152 (480) 1203 (4B3)	through unused - contains zeros					
300	1204 (4B4)	*primary*	MASTERCATALOG, TRACKS	MTKPR	data	3	
301	1208 (4B8)	*secondary*	MASTERCATALOG, TRACKS	MTKSC	data	3	
302	1212 (4BC)	*primary*	CLUSTER, TRACKS	CTKPR	data	3	
303	1216 (4C0)	*secondary*	CLUSTER, TRACKS	CTKSC	data	3	
304	1220 (4C4)	*primary*	SPACE, TRACKS	STKPR	data	3	
305	1224 (4C8)	*secondary*	SPACE, TRACKS	STKSC	data	3	
306	1228 (4CC)	*primary*	USERCATALOG, TRACKS	UTKPR	data	3	
307	1232 (4D0)	*secondary*	USERCATALOG, TRACKS	UTKSC	data	3	
308	1236 (4D4)	unused - contains zeros					
309	1240 (4D8)	unused - contains zeros					
310	1244 (4DC)	*primary*	MASTERCATALOG, CYLINDERS	MCLPR	data	3	
311	1248 (4E0)	*secondary*	MASTERCATALOG, CYLINDERS	MCLSC	data	3	
312	1252 (4E4)	*primary*	CLUSTER, CYLINDERS	CCLPR	data	3	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
313	1256 (4E8)	*secondary*	CLUSTER, CYLINDERS	CCLSC	data	3	
314	1260 (4EC)	*primary*	USERCATALOG, CYLINDERS	UCLPR	data	3	
315	1264 (4F0)	*secondary*	USERCATALOG, CYLINDERS	UCLSC	data	3	
316	1268 (4F4)	unused - contains zeros					
317	1272 (4F8)	unused - contains zeros					
318	1276 (4FC)	*primary*	SPACE, CYLINDERS	SCLPR	data	3	
319	1280 (500)	*secondary*	SPACE, CYLINDERS	SCLSC	data	3	
320	1284 (504)	*primary*	MASTERCATALOG, RECORDS	MRCPR	data	3	
321	1288 (508)	*secondary*	MASTERCATALOG, RECORDS	MRCSC	data	3	
322	1292 (50C)	*primary*	CLUSTER, RECORDS	CRCPR	data	3	
323	1296 (510)	*secondary*	CLUSTER, RECORDS	CRCSC	data	3	
324	1300 (514)	*primary*	SPACE, RECORDS	SRCPR	data	3	
325	1304 (518)	*secondary*	SPACE, RECORDS	SRCSC	data	3	
326	1308 (51C)	*primary*	USERCATALOG, RECORDS	URCPR	data	3	
327	1312 (520)	*secondary*	USERCATALOG, RECORDS	URCSC	data	3	
328	1316 (524)	unused - contains zeros					
329	1320 (528)	unused - contains zeros					
330	1324 (52C)	*primary*	DATA, CYLINDERS	DCLPR	data	3	
331	1328 (530)	*secondary*	DATA, CYLINDERS	DCLSC	data	3	
332	1332 (534)	*primary*	DATA, RECORDS	DRCPR	data	3	
333	1336 (538)	*secondary*	DATA, RECORDS	DRCSC	data	3	
334	1340 (53C)	*primary*	INDEX, CYLINDERS	ICLPR	data	3	
335	1344 (540)	*secondary*	INDEX, CYLINDERS	ICLSC	data	3	
336	1348 (544)	*primary*	INDEX, RECORDS	IRCPR	data	3	
337	1352 (548)	*secondary*	INDEX, RECORDS	IRCSC	data	3	
338	1356 (54C)	NAME	ALTERNATEINDEX	GETRY	data	6	
339	1360 (550)	MODEL	ALTERNATEINDEX	GMODL	itself	1	See parms 340-342.
340	1364 (554)	*entryname/ password*	ALTERNATEINDEX, MODEL	GENAM	data	6	
341	1368 (558)	*catname/ password*	ALTERNATEINDEX, MODEL	GMDCT	data	6	
342	1372 (55C)	*dname*	ALTERNATEINDEX, MODEL	GMDNM	data	2	
343	1376 (560)	MASTERPW	ALTERNATEINDEX	GMSTR	data	2	
344	1380 (564)	CONTROLPW	ALTERNATEINDEX	GCINT	data	2	
345	1384 (568)	UPDATEPW	ALTERNATEINDEX	GUPDT	data	2	
346	1388 (56C)	READPW	ALTERNATEINDEX	GREAD	data	2	
347	1392 (570)	CODE	ALTERNATEINDEX	GCODE	data	2	
348	1396 (574)	ATTEMPTS	ALTERNATEINDEX	GATTP	data	3	
349	1400 (578)	AUTHORIZATION	ALTERNATEINDEX	GAUTH	itself	1	See parms 350 and 351.
350	1404 (57C)	*entrypoint*	ALTERNATEINDEX, AUTHORIZATION	GEPNM	data	2	
351	1408 (580)	*string*	ALTERNATEINDEX, AUTHORIZATION	GSTRG	data	2	
352	1412 (584)	TO	ALTERNATEINDEX	GTO	data	3	
353	1416 (588)	FOR	ALTERNATEINDEX	GFOR	data	3	
354	1420 (58C)	OWNER	ALTERNATEINDEX	GOWNR	data	2	
355	1424 (590)	SHAREOPTIONS	ALTERNATEINDEX	GSHAR	itself	1	
356	1428 (594)	*crosspartition*	ALTERNATEINDEX	GSHR1	data	3	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
357	1432 (598)	*crosssystem*	ALTERNATEINDEX	GSHR2	data	3	
358	1436 (59C)	ERASE	ALTERNATEINDEX	GERAS	itself	1	
359	1440 (5A0)	NOERASE	ALTERNATEINDEX	GNERS	itself	1	
360	1444 (5A4)	KEYS	ALTERNATEINDEX	GKEY	itself	1	See parms 361 and 362.
361	1448 (5A8)	*length*	ALTERNATEINDEX, KEYS	GKYLN	data	3	key length
362	1452 (5AC)	*offset*	ALTERNATEINDEX, KEYS	GKYPS	data	3	key offset
363	1456 (5B0)	REPLICATE	ALTERNATEINDEX	GREPL	itself	1	
364	1460 (5B4)	NOREPLICATE	ALTERNATEINDEX	GNREP	itself	1	
365	1464 (5B8)	IMBED	ALTERNATEINDEX	GIMBD	itself	1	
366	1468 (5BC)	NOIMBED	ALTERNATEINDEX	GNIBD	itself	1	
367	1472 (5C0)	FILE	ALTERNATEINDEX	GINDD	data	2	*dname*
368	1476 (5C4)	VOLUMES	ALTERNATEINDEX	GVSER	data	4	A list of volume serial numbers (character 6).
369	1480 (5C8)	KEYRANGES	ALTERNATEINDEX	GRANG	data	3	Count of sub-parms. See parms 370 and 371.
370	1484 (5CC)	*lowkey*	ALTERNATEINDEX, KEYRANGES	GRGLOPTR	list of pointers	2	Each item points to a low keyrange value.
371	1488 (5D0)	*highkey*	ALTERNATEINDEX, KEYRANGES	GRGHIPTR	list of pointers	2	Each item points to a high keyrange value.
372	1492 (5D4)	ORDERED	ALTERNATEINDEX	GORDR	itself	1	
373	1496 (5D8)	UNORDERED	ALTERNATEINDEX	GUORD	itself	1	
374	1500 (5DC)	SUBALLOCATION	ALTERNATEINDEX	GSUBA	itself	1	
375	1504 (5E0)	UNIQUE	ALTERNATEINDEX	GUNIQ	itself	1	
376	1508 (5E4)	TRACKS	ALTERNATEINDEX	GTRKS	itself	1	See parms 377 and 378.
377	1512 (5E8)	*primary*	ALTERNATEINDEX, TRACKS	GTKPR	data	3	
378	1516 (5EC)	*secondary*	ALTERNATEINDEX, TRACKS	GTKSC	data	3	
379	1520 (5F0)	CYLINDERS	ALTERNATEINDEX	GCYLD	itself	1	See parms 380 and 381.
380	1524 (5F4)	*primary*	ALTERNATEINDEX, CYLINDERS	GCLPR	data	3	
381	1528 (5F8)	*secondary*	ALTERNATEINDEX, CYLINDERS	GCLSC	data	3	
382	1532 (5FC)	RECORDS	ALTERNATEINDEX	GRCDS	itself	1	See parms 383 and 384.
383	1536 (600)	*primary*	ALTERNATEINDEX, RECORDS	GRCPR	data	3	
384	1540 (604)	*secondary*	ALTERNATEINDEX, RECORDS	GRCSC	data	3	
385	1544 (608)	RECORDSIZE	ALTERNATEINDEX	GRSIZ	itself	1	See parms 386 and 387.
386	1548 (60C)	*average*	ALTERNATEINDEX, RECORDSIZE	GARSZ	data	3	
387	1552 (610)	*maximum*	ALTERNATEINDEX, RECORDSIZE	GMRSZ	data	3	
388	1556 (614)	WRITECHECK	ALTERNATEINDEX	GWCK	itself	1	
389	1560 (618)	NOWRITECHECK	ALTERNATEINDEX	GNWCK	itself	1	
390	1564 (61C)	SPEED	ALTERNATEINDEX	GSPED	itself	1	
391	1568 (620)	RECOVERY	ALTERNATEINDEX	GRECV	itself	1	
392	1572 (624)	FREESPACE	ALTERNATEINDEX	GFSPC	itself	1	See parms 393 and 394.
393	1576 (628)	*cipercent*	ALTERNATEINDEX, FREESPACE	GCIFS	data	3	
394	1580 (62C)	*capercent*	ALTERNATEINDEX, FREESPACE	GCAFS	data	3	
395	1584 (630)	BUFFERSPACE	ALTERNATEINDEX	GBFSZ	data	3	
396	1588 (634)	CONTROL-INTERVALSIZE	ALTERNATEINDEX	GCINV	data	3	
397	1592 (638)	RELATE	ALTERNATEINDEX	GREL	data	6	
398	1596 (63C)	EXCEPTIONEXIT	ALTERNATEINDEX	GEEXT	data	2	
399	1600 (640)	REUSE	ALTERNATEINDEX	GRUS	itself	1	
400	1604 (644)	NOREUSE	ALTERNATEINDEX	GNRUS	itself	1	
401	1608 (648)	UNIQUEKEY	ALTERNATEINDEX	GUNQK	itself	1	
402	1612 (64C)	NONUNIQUEKEY	ALTERNATEINDEX	GNUQK	itself	1	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
403	1616 (650)	UNIQUEKEY	DATA	DUNQK	itself	1	
404	1620 (654)	NONUNIQUEKEY	DATA	DNUQK	itself	1	
405	1624 (658)	UPGRADE	ALTERNATEINDEX	GUPG	itself	1	
406	1628 (65C)	NOUPGRADE	ALTERNATEINDEX	GNUPG	itself	1	
407	1632 (660)	DEDICATE	SPACE	SPED	itself	1	
408	1636 (664)	DEDICATE	MASTERCATALOG	MDED	itself	1	
409	1640 (668)	DEDICATE	USERCATALOG	UDED	itself	1	
410	1644 (66C)	NAME	PATH	RETRY	data	6	
411	1648 (670)	MODEL	PATH	RMODL	itself	1	See parms 412-414.
412	1652 (674)	*entryname/ password*	PATH, MODEL	RENAM	data	6	
413	1656 (678)	*catname/ password*	PATH, MODEL	RMDCT	data	6	
414	1660 (67C)	*dname*	PATH, MODEL	RMDNM	data	2	
415	1664 (680)	MASTERPW	PATH	RMSTR	data	2	
416	1668 (684)	CONTROLPW	PATH	RCINT	data	2	
417	1672 (688)	UPDATEPW	PATH	RUPDT	data	2	
418	1676 (68C)	READPW	PATH	RREAD	data	2	
419	1680 (690)	CODE	PATH	RCODE	data	2	
420	1684 (694)	ATTEMPTS	PATH	RATTP	data	3	
421	1688 (698)	AUTHORIZATION	PATH	RAUTH	itself	1	See parms 422 and 423.
422	1692 (69C)	*entrypoint*	PATH, AUTHORIZATION	REPNM	data	2	
523	1696 (6A0)	*string*	PATH, AUTHORIZATION	RSTRG	data	2	
424	1700 (6A4)	TO	PATH	RTO	data	3	
425	1704 (6A8)	FOR	PATH	RFOR	data	3	
426	1708 (6AC)	OWNER	PATH	ROWNR	data	2	
427	1712 (6B0)	FILE	PATH	RINDD	data	2	*dname*
428	1716 (6B4)	UPDATE	PATH	RUPD	itself	1	
429	1720 (6B8)	NOUPDATE	PATH	RNUPD	itself	1	
430	1724 (6BC)	PATHENTRY	PATH	RPENT	data	6	
431	1728 (6C0)	BLOCKS	CLUSTER	CBLKS	itself	1	See parms 439 and 447.
432	1732 (6C4)	BLOCKS	ALTERNATEINDEX	GBLKS	itself	1	See parms 440 and 448.
433	1736 (6C8)	BLOCKS	MASTERCATALOG	MBLKS	itself	1	See parms 441 and 449.
434	1740 (6CC)	BLOCKS	USERCATALOG	UBLKS	itself	1	See parms 442 and 450.
435	1744 (6D0)	BLOCKS	SPACE	SBLKS	itself	1	See parms 443 and 451.
436	1748 (6D4)	unused - contains zeros					
437	1752 (6D8)	BLOCKS	DATA	DBLKS	itself	1	See parms 445 and 453.
438	1756 (6DC)	BLOCKS	INDEX	IBLKS	itself	1	See parms 446 and 454.
439	1760 (6E0)	*primary*	CLUSTER, BLOCKS	CBLKR	data	3	
440	1764 (6E4)	*primary*	ALTERNATEINDEX, BLOCKS	GBLKR	data	3	
441	1768 (6E8)	*primary*	MASTERCATALOG, BLOCKS	MBLKR	data	3	
442	1772 (6EC)	*primary*	USERCATALOG, BLOCKS	UBLKR	data	3	
443	1776 (6F0)	*primary*	SPACE, BLOCKS	SBLKR	data	3	
444	1780 (6F4)	unused - contains zeros					
445	1784 (6F8)	*primary*	DATA, BLOCKS	DBLKR	data	3	
446	1788 (6FC)	*primary*	INDEX, BLOCKS	IBLKR	data	3	
447	1792 (700)	*secondary*	CLUSTER, BLOCKS	CBLKC	data	3	
448	1796 (704)	*secondary*	ALTERNATEINDEX, BLOCKS	GBLKC	data	3	
449	1800 (708)	*secondary*	MASTERCATALOG, BLOCKS	MBLKC	data	3	
450	1804 (70C)	*secondary*	USERCATALOG, BLOCKS	UBLKC	data	3	
451	1808 (710)	*secondary*	SPACE, BLOCKS	SBLKC	data	3	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
452	1812 (714)	unused - contains zeros					
453	1816 (718)	*secondary*	DATA, BLOCKS	DBLKC	data	3	
454	1820 (71C)	*secondary*	INDEX, BLOCKS	IBLKC	data	3	
455 468	1824 (720) 1879 (757)	through unused - contains zeros					
469	1880 (758)	CLASS	MASTERCATALOG	MCLAS	data	3	
470	1884 (75C)	CLASS	USERCATALOG	UCLAS	data	3	
471	1888 (760)	CLASS	SPACE	SCLAS	data	3	
472	1892 (764)	unused - contains zeros					
473	1896 (768)	USECLASS	CLUSTER	CUSCL	itself	1	See parms 482 and 483.
474	1900 (76C)	unused - contains zeros					
475	1904 (770)	USECLASS	DATA	DUSCL	itself	1	See parms 485 and 486.
476	1908 (774)	USECLASS	INDEX	IUSCL	itself	1	See parms 487 and 488.
477	1912 (778)	USECLASS	ALTERNATEINDEX	GUSCL	itself	1	
478	1916 (77C)	unused - contains zeros					
479	1920 (780)	unused - contains zeros					
480	1924 (784)	unused - contains zeros					
481	1928 (788)	unused - contains zeros					
482	1932 (78C)	*primary*	CLUSTER, USECLASS	CPUSC	data	3	
483	1936 (790)	*secondary*	CLUSTER, USECLASS	CSUSC	data	2	
484	1940 (794)	unused - contains zeros					
485	1944 (798)	*primary*	DATA, USECLASS	DPUSC	data	3	
486	1948 (79C)	*secondary*	DATA, USECLASS	DSUSC	data	2	
487	1952 (7A0)	*primary*	INDEX, USECLASS	IPUSC	data	3	
488	1956 (7A4)	*secondary*	INDEX, USECLASS	ISUSC	data	2	
489	1960 (7A8)	*primary*	ALTERNATEINDEX, USECLASS	GPUSC	data	3	
490	1964 (7AC)	*secondary*	ALTERNATEINDEX, USECLASS	GSUSC	data	2	

DELETE FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			DELETEbb
1	8 (8)	*entryname/ password*		NTRY	data	4/6	Count of repetitions followed by data format 6 data.
2	12 (C)	CATALOG		CATLG	itself	1	See parms 3 and 4.
3	16 (10)	*catname/ password*	CATALOG	CAT	data	6	
4	20 (14)	*dname*	CATALOG	CATDD	data	2	
5	24 (18)	FILE		INDD	data	2	
6	28 (1C)	PURGE		PURGE	itself	1	
7	32 (20)	NOPURGE		NOPUR	itself	1	
8	36 (24)	ERASE		ERASE	itself	1	
9	40 (28)	NOERASE		NOERA	itself	1	
10	44 (2C)	unused - contains zeros					
11	48 (30)	CLUSTER		CLUST	itself	1	
12	52 (34)	SPACE		SPACE	itself	1	
13	56 (38)	USERCATALOG		UCAT	itself	1	
14	60 (3C)	MASTERCATALOG		MCAT	itself	1	
15	64 (40)	NONVSAM		ALIEN	itself	1	
16	68 (44)	SCRATCH		SCR	itself	1	
17	72 (48)	NOSCRATCH		NSCR	itself	1	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
18	76 (4C)	unused - contains zeros					
19	80 (50)	unused - contains zeros					
20	84 (54)	unused - contains zeros					
21	88 (58)	ALTERNATEINDEX		AIX	itself	1	
22	92 (5C)	PATH		PATH	itself	1	
23	96 (60)	FORCE		FRC	itself	1	
24	100 (64)	NOFORCE		NFRC	itself	1	

EXPORT FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			EXPORTbb
1	8 (8)	*entryname/ password*		ENT	data	6	
2	12 (C)	INFILE		INDD	data	2	
3	16 (10)	OUTFILE		OUT	itself	1	See parms 4 and 5.
4	20 (14)	*dname*	OUTFILE	OUTDD	data	2	
5	24 (18)	ENVIRONMENT	OUTFILE	ENVIR	itself	1	See parms 20-26.
6	28 (1C)	TEMPORARY		TEMP	itself	1	
7	32 (20)	PERMANENT		PERM	itself	1	
8	36 (24)	INHIBITSOURCE		INHBS	itself	1	
9	40 (28)	INHIBITARGET		INHBT	itself	1	
10	44 (2C)	ERASE		ERASE	itself	1	
11	48 (30)	NOERASE		NOERS	itself	1	
12	52 (34)	PURGE		PURGE	itself	1	
13	56 (38)	NOPURG		NPRG	itself	1	
14	60 (3C)	DISCONNECT		DISCT	itself	1	
15	64 (40)	NOINHIBITSOURCE		NINHS	itself	1	
16	68 (44)	NOINHIBITTARGET		NINHT	itself	1	
17	72 (48)	CIMODE		CIM	itself	1	
18	76 (4C)	RECORDMODE		RECM	itself	1	
19	80 (50)	unused - contains zeros					
20	84 (54)	PRIMEDATADEVICE	OUTFILE, ENVIRONMENT	PDEV	data	2	
21	88 (58)	BLOCKSIZE	OUTFILE, ENVIRONMENT	BLKSZ	data	3	
22	92 (5C)	STDLABEL	OUTFILE, ENVIRONMENT	OSLBL	itself	1	
23	96 (60)	NOLABEL	OUTFILE, ENVIRONMENT	ONLBL	itself	1	
24	100 (64)	NOREWIND	OUTFILE, ENVIRONMENT	ONREW	itself	1	
25	104 (68)	REWIND	OUTFILE, ENVIRONMENT	OREW	itself	1	
26	108 (6C)	UNLOAD	OUTFILE, ENVIRONMENT	OUNLD	itself	1	

EXPORTRA FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			EXPORTRA
1	8 (8)	FORCE		FRC	itself	1	
2	12 (C)	NOFORCE		NFRC	itself	1	
3	16 (10)	OUTFILE		OUT	itself	1	See parms 11 and 13.
4	20 (14)	CRA		CRACNT	data	3	Count of the number of CRAs (*dname1*) provided. See parms 5-9, 14, and 15.
5	24 (18)	*dname1*	CRA	CRADDPTR	list of pointers*	2	Each points to the CRA *dname1* it relates to in the order that they appear in the EXPORTRA command.
6	28 (1C)	ALL	CRA	ALLNTPTR	list of pointers*	1	Each points to itself.
7	32 (20)	NONE	CRA	NONEPTR	list of pointers*	1	Each points to itself.
8	36 (24)	ENTRIES	CRA	ENTREPTR	list of counts	3	Each count indicates the number of entries specified for the related CRA. See parms 14 and 15.
9	40 (28)	INFILE	CRA	IFILEPTR	list of pointers*	2	Each pointer points to the *dname2* to be used for a CRA.
10	44 (2C)	*MASTERPW*		MRPW	data	2	
11	48 (30)	ENVIRONMENT	OUTFILE	ENVIR	itself	1	See parms 12 and 16-21.
12	52 (34)	PRIMEDATADEVICE	OUTFILE, ENVIRONMENT	PDEV	data	2	
13	56 (38)	*dname*	OUTFILE	OUTDD	data	2	
14	60 (3C)	*entryname*	CRA ENTRIES	ENTNMPTR	list of pointers*	6	Each pointer points to an *entryname* in the associated CRA.
15	64 (40)	*dname3*	CRA ENTRIES	ENTDNPTR	list of pointers*	2	Each pointer points to the *dname* used to export the associated *entryname* in the ENTNMPTR.
16	68 (44)	BLOCKSIZE	OUTFILE, ENVIRONMENT	BLKSZ	data	3	
17	72 (48)	STDLABEL	OUTFILE, ENVIRONMENT	OSLBL	itself	1	
18	76 (4C)	NOLABEL	OUTFILE, ENVIRONMENT	ONLBL	itself	1	
19	80 (50)	NOREWIND	OUTFILE, ENVIRONMENT	ONREW	itself	1	
20	84 (54)	REWIND	OUTFILE, ENVIRONMENT	OREW	itself	1	
21	88 (58)	UNLOAD	OUTFILE, ENVIRONMENT	OUNLD	itself	1	
22	92 (5C)	CIMODE		CIM	itselt	1	
23	96 (60)	RECORDMODE		RECM	itself	1	
24	100 (64)	CRAVOLUMES		CRAVLCNT	data	3	Count of the number of CRAVOLUMES *(volser)* provided. See Parms 25-29
25	104 (68)	*volser*	CRAVOLUMES	VOLIDPTR	list of pointers*	2	Each points to the CRAVOLUMES *volser* it relates to in the order that they appear in the EXPORTRA command.
26	108 (6C)	ALL	CRAVOLUMES	ALLVLPRT	list of pointers*	1	Each points to itself.
27	112 (70)	NONE	CRAVOLUMES	NONEVPTR	list of pointers*	1	Each points to itself.
28	116 (74)	ENTRIES	CRAVOLUMES	ENTVLPTR	list of counts	3	Each count indicates the number of entries specified for the related CRAVOLUMES *volser*. See Parm 29.
29	120 (78)	*entryname*	ENTRIES, CRAVOLUMES	ENTNVPTR	list of pointers*	6	Each pointer points to an *entryname* in the CRA identified by the *volser* parameter.

* One pointer per object.

IMPORT FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			IMPORTbb
1	8 (8)	INFILE		IN	itself	1	See parms 15 and 16.
2	12 (C)	OUTFILE		OUTDD	data	7	*dname/password*
3	16 (10)	OBJECTS		OBJTSCNT	data	3	Count of the number of names specified. See parms 4-11, 37-39.
4	20 (14)	*objectname*	OBJECTS	OBJNMPTR	list of pointers*	6	44-character entryname for each object.
5	24 (18)	NEWNAME	OBJECTS	NEWNMPTR	list of pointers*	6	44-character entryname for each object.
6	28 (1C)	FILE	OBJECTS	OBJFLPTR	list of pointers*	2	*Dname* for each object.
7	32 (20)	VOLUMES	OBJECTS	LISTVPTR	list of pointers*	4	List of volume serial numbers for each object.
8	36 (24)	KEYRANGES	OBJECTS	RANGEPTR	list of counts	3	Each count indicates the number of low/high keyrange pairs for the related *object name*. See parms 12 and 13.
9	40 (28)	DEVICETYPE	OBJECTS	DEVPTR	list of pointers*	5	Device type of each object.
10	44 (2C)	ORDERED	OBJECTS	ORDPTR	list of pointers*	1	Pointer to itself.
11	48 (30)	UNORDERED	OBJECTS	UNORDPTR	list of pointers*	1	Pointer to itself.
12	52 (34)	*lowkey*	OBJECTS, KEYRANGES	LOWKYPTR	list of pointers*	1/2	Each pointer points to another list of pointers (one per low keyrange value).
13	56 (38)	*highkey*	OBJECTS, KEYRANGES	HIKEYPTR	list of pointers*	1/2	Each pointer points to another list of pointers (one per high keyrange value).
14	60 (3C)	CONNECT		CON	itself	1	
15	64 (40)	*dname*	INFILE	INDD	data	2	*dname*
16	68 (44)	ENVIRONMENT	INFILE	ENV	itself	1	See parms 21-23 and 26-30.
17	72 (48)	PURGE		PRG	itself	1	
18	76 (4C)	NOPURGE		NPRG	itself	1	
19	80 (50)	ERASE		ERAS	itself	1	
20	84 (54)	NOERASE		NERAS	itself	1	
21	88 (58)	BLOCKSIZE	INFILE, ENVIRONMENT	BLKSZ	data	3	
22	92 (5C)	PRIMEDATADEVICE	INFILE, ENVIRONMENT	PDEV	data	2	Device type.
23	96 (60)	RECORDSIZE	INFILE, ENVIRONMENT	RECSZE	data	3	
24	100 (64)	unused - contains zeros					
25	104 (68)	OUTPW		OUTPW	data	2	*password*
26	108 (6C)	STDLABEL	INFILE, ENVIRONMENT	ISLBL	itself	1	
27	112 (70)	NOLABEL	INFILE, ENVIRONMENT	INLBL	itself	1	
28	116 (74)	NOREWIND	INFILE, ENVIRONMENT	INREW	itself	1	
29	120 (78)	REWIND	INFILE, ENVIRONMENT	IREW	itself	1	
30	124 (7C)	UNLOAD	INFILE, ENVIRONMENT	IUNLD	itself	1	
31	128 (80)	unused - contains zeros					
32	132 (84)	unused - contains zeros					
33	136 (88)	CATALOG		CAT	data	6	*catname/password*
34	140 (8C)	unused - contains zeros					
35	144 (90)	unused - contains zeros					
36	148 (94)	unused - contains zeros					
37	152 (98)	USECLASS	OBJECTS	USCLPTR	list of pointers*	1	Points to itself. See parms 38 and 39.

* One pointer per object.

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
38	156 (9C)	*primary*	OBJECTS,	PUSCPTR	data	3 USECLASS	
39	160 (A0)	*secondary*	OBJECTS,	SUSCPTR	data	2 USECLASS	

IMPORTRA FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			IMPORTRA
1	8 (8)	INFILE		IN	itself	1	See parms 15 and 16.
2	12 (C)	OUTFILE		OUTDD	data	2	*dname*
3	16 (10)	OBJECTS		OBJTS	data	3	Count of the number of object names specified. See parms 4-9, 37-39.
4	20 (14)	*objectname*	OBJECTS	OBJNMPTR	list of pointers*	6	Each points to the 44-character entryname for an object.
5	24 (18)	DEFAULTVOLUMES	OBJECTS	DVOLPTR	list of pointers*	1	Each points to itself.
6	28 (1C)	FILE	OBJECTS	OBJFLPTR	list of pointers*	2	Each points to the *dname* for an object.
7	32 (20)	VOLUMES	OBJECTS	LISTVPTR	list of pointers*	4	Each points to a list of volume serial numbers for each object.
8	36 (24)	unused - contains zeros					
9	40 (28)	DEVICETYPE	OBJECTS	DEVTPTR	list of pointers*	5	Each points to the device type of each object.
10	44 (2C)	unused - contains zeros					
11	48 (30)	unused - contains zeros					
12	52 (34)	unused - contains zeros					
13	56 (38)	unused - contains zeros					
14	60 (3C)	unused - contains zeros					
15	64 (40)	*dname*	INFILE	INDD	data	2	
16	68 (44)	ENVIRONMENT	INFILE	ENV	itself	1	See parms 21-22, 26-30.
17	72 (48)	unused - contains zeros					
18	76 (4C)	unused - contains zeros					
19	80 (50)	unused - contains zeros					
20	84 (54)	unused - contains zeros					
21	88 (58)	BLOCKSIZE	INFILE, ENVIRONMENT	BLKSZ	data	3	
22	92 (5C)	PRIMEDATADEVICE	INFILE, ENVIRONMENT	PDEV	data	2	Device type.
23	96 (60)	unused - contains zeros					
24	100 (64)	unused - contains zeros					
25	104 (68)	unused - contains zeros					
26	108 (6C)	STDLABEL	INFILE, ENVIRONMENT	ISLBL	itself	1	
27	112 (70)	NOLABEL	INFILE, ENVIRONMENT	INLBL	itself	1	
28	116 (74)	NOREWIND	INFILE, ENVIRONMENT	INREW	itself	1	
29	120 (78)	REWIND	INFILE, ENVIRONMENT	IREW	itself	1	
30	124 (7C)	UNLOAD	INFILE, ENVIRONMENT	IUNLD	itself	1	
31	128 (80)	unused - contains zeros					
32	132 (84)	unused - contains zeros					
33	136 (88)	CATALOG		CAT	data	6	*catname/password* (*dname*)
34	140 (8C)	unused - contains zeros					
35	144 (90)	unused - contains zeros					
36	148 (94)	unused - contains zeros					
37	152 (98)	USECLASS	OBJECTS	USCLPTR	list of	1	Points to itself.

* One pointer per object.

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
					pointers[a]		See parms 38 and 39.
38	156 (9C)	*primary*	OBJECTS, USECLASS	PUSCPTR	data	3	
39	160 (A0)	*secondary*	OBJECTS, USECLASS	SUSCPTR	data	2	

LISTCAT FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			LISTCAT[b]
1	8 (8)	CATALOG		CAT	itself	1	See parms 11 and 12.
2	12 (C)	OUTFILE		OUTDD	data	2	
3	16 (10)	ENTRIES		ENT	data	6	
4	20 (14)	unused - contains zeros					
5	24 (18)	CLUSTER		CLUST	itself	1	
6	28 (1C)	DATA		DATUM	itself	1	
7	32 (20)	INDEX		INDEX	itself	1	
8	36 (24)	SPACE		SPACE	itself	1	
9	40 (28)	NONVSAM		ALIEN	itself	1	
10	44 (2C)	USERCATALOG		UCAT	itself	1	
11	48 (30)	*catname/ password*	CATALOG	CATNM	data	6	
12	52 (34)	*dname*	CATALOG	CATDD	data	2	
13	56 (38)	unused - contains zeros					
14	60 (3C)	NAME		NAME	itself	1	
15	64 (40)	ALL		FALL	itself	1	
16	68 (44)	VOLUME		VOL	itself	1	
17	72 (48)	ALLOCATION		ALLOC	itself	1	
18	76 (4C)	unused - contains zeros					
19	80 (50)	unused - contains zeros					
20	84 (54)	unused - contains zeros					
21	88 (58)	unused - contains zeros					
22	92 (5C)	ALTERNATEINDEX		AIX	itself	1	
23	96 (60)	PATH		PATH	itself	1	
24	100 (64)	NOTUSABLE		NUSE	itself	1	

LISTCRA FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			LISTCRA[b]
1	8 (8)	INFILE		IFILE	list	4	Listof CRA *dnames.*
2	12 (C)	COMPARE		CMPR	itself	1	
3	16 (10)	NOCOMPARE		NCMPR	itself	1	
4	20 (14)	DUMP		DUMP	itself	1	
5	24 (18)	NAME		NAME	itself	1	
6	28 (1C)	CATALOG		CAT	itself	1	See parms 7 and 8.
7	32 (20)	*catname/ password*	CATALOG	CATNM	data	6	
8	36 (24)	*dname*	CATALOG	CATDN	data	2	
9	40 (28)	MASTERPW		MRPW	data	2	*password*
10	44 (2C)	SEQUENTIALDUMP		SDUMP	itself	1	
11	48 (30)	INVOLUMES		INVOL	list	4	List of CRA *volsers.*

PARM FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			PARMbbbb
1	8 (8)	TEST		TEST	itself	1	See parms 2-5.
2	12 (C)	OFF	TEST	TOFF	itself	1	
3	16 (10)	TRACE	TEST	TRACE	itself	1	
4	20 (14)	AREAS	TEST	AREA	list	2	
5	24 (18)	FULL	TEST	FULL	itself	1	See parms 6-8.
6	28 (1C)	*dumpid*	FULL,TEST	FIDPTR	data	2	
7	32 (20)	*count1*	FULL,TEST	BEGINPTR	data	3	Starting count for full dump.
8	36 (24)	*count2*	FULL,TEST	COUNTPTR	data	3	Number of full dumps desired.
9	40 (28)	GRAPHICS		GRAPH	itself	1	See parms 10 and 11.
10	44 (2C)	CHAIN	GRAPHICS	CHAIN	itself	1	See parms 15-21.
11	48 (30)	TABLE	GRAPHICS	TABLE	data	2	
12	52 (34)	MARGINS		MARG	itself	1	See parms 13 and 14.
13	56 (38)	*leftmargin*	MARGINS	LMARG	data	3	
14	60 (3C)	*rightmargin*	MARGINS	RMARG	data	3	
15	64 (40)	AN	CHAIN,GRAPHICS	CHNAN	itself	1	
16	68 (44)	HN	CHAIN,GRAPHICS	CHNHN	itself	1	
17	72 (48)	PN	CHAIN,GRAPHICS	CHNPN	itself	1	
18	76 (4C)	QN	CHAIN,GRAPHICS	CHNQN	itself	1	
19	80 (50)	RN	CHAIN,GRAPHICS	CHNRN	itself	1	
20	84 (54)	SN	CHAIN,GRAPHICS	CHNSN	itself	1	
21	88 (58)	TN	CHAIN,GRAPHICS	CHNTN	itself	1	

PRINT FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			PRINTbbb
1	8 (8)	INFILE		INDN	itself	1	See parms 9 and 16.
2	12 (C)	unused - contains zeros					
3	16 (10)	FROMKEY		FMKYC	data	2	
4	20 (14)	FROMADDRESS		FMRBA	data	3	
5	24 (18)	SKIP		SKIP	data	3	Number of records to skip.
6	28 (1C)	TOKEY		TOKYC	data	2	
7	32 (20)	TOADDRESS		TORBA	data	3	
8	36 (24)	COUNT		COUNT	data	3	Number of records to print.
9	40 (28)	*dname/ password*	INFILE	INPDD	data	7	
10	44 (2C)	unused - contains zeros					
11	48 (30)	unused - contains zeros					
12	52 (34)	HEX		FHEX	itself	1	
13	56 (38)	CHARACTER		FCHAR	itself	1	
14	60 (3C)	DUMP		FDUMP	itself	1	
15	64 (40)	unused - contains zeros					
16	68 (44)	ENVIRONMENT	INFILE	IENV	itself	1	See parms 17-29 and 32-36.
17	72 (48)	RECORDFORMAT	INFILE, ENVIRONMENT	IRFMT	itself	1	See parms 23-29.
18	76 (4C)	BLOCKSIZE	INFILE, ENVIRONMENT	IBKSZ	data	3	
19	80 (50)	RECORDSIZE	INFILE, ENVIRONMENT	IRCSZ	data	3	
20	84 (54)	unused - contains zeros					
21	88 (58)	HINDEXDEVICE	INFILE, ENVIRONMENT	IHDEV	data	2	
22	92 (5C)	PRIMEDATADEVICE	INFILE, ENVIRONMENT	IPDEV	data	2	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
23	96 (60)	FIXUNB	INFILE, ENVIRONMENT, RECORDFORMAT	IFUNB	itself	1	
24	100 (64)	FIXBLK	INFILE, ENVIRONMENT, RECORDFORMAT	IFBLK	itself	1	
25	104 (68)	VARUNB	INFILE, ENVIRONMENT, RECORDFORMAT	IVUNB	itself	1	
26	108 (6C)	VARBLK	INFILE, ENVIRONMENT, RECORDFORMAT	IVBLK	itself	1	
27	112 (70)	SPNUNB	INFILE, ENVIRONMENT, RECORDFORMAT	ISUNB	itself	1	
28	116 (74)	SPNBLK	INFILE, ENVIRONMENT, RECORDFORMAT	ISBLK	itself	1	
29	120 (78)	UNDEF	INFILE, ENVIRONMENT, RECORDFORMAT	IUNDF	itself	1	
30	124 (7C)	FROMNUMBER		FMNUM	data	3	
31	128 (80)	TONUMBER		TONUM	data	3	
32	132 (84)	STDLABEL	INFILE, ENVIRONMENT	ISLBL	itself	1	
33	136 (88)	NOLABEL	INFILE, ENVIRONMENT	INLBL	itself	1	
34	140 (8C)	NOREWIND	INFILE, ENVIRONMENT	INREW	itself	1	
35	144 (90)	REWIND	INFILE, ENVIRONMENT	IREW	itself	1	
36	148 (94)	UNLOAD	INFILE, ENVIRONMENT	IUNLD	itself	1	

REPRO FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			REPRObbb
1	8 (8)	INFILE		INDN	itself	1	See parms 9, 16-29, and 53-57.
2	12 (C)	OUTFILE		OUTDN	itself	1	See parms 10, 32-45, and 58-62.
3	16 (10)	FROMKEY		FMKYC	data	2	
4	20 (14)	FROMADDRESS		FMRBA	data	3	
5	24 (18)	SKIP		SKIP	data	3	Number of records to skip.
6	28 (1C)	TOKEY		TOKYC	data	2	
7	32 (20)	TOADDRESS		TORBA	data	3	
8	36 (24)	COUNT		COUNT	data	3	
9	40 (28)	*dname/ password*	INFILE	INPDD	data	7	
10	44 (2C)	*dname/ password*	OUTFILE	OUTDD	data	7	
11	48 (30)	unused - contains zeros					
12	52 (34)	unused - contains zeros					
13	56 (38)	FROMNUMBER		FMNUM	data	3	
14	60 (3C)	TONUMBER		TONUM	data	3	
15	64 (40)	unused - contains zeros					
16	68 (44)	ENVIRONMENT	INFILE	IENV	itself	1	See parms 17-29.
17	72 (48)	RECORDFORMAT	INFILE, ENVIRONMENT	IRFMT	itself	1	See parms 23-29 and 40-45.
18	76 (4C)	BLOCKSIZE	INFILE, ENVIRONMENT	IBKSZ	data	3	
19	80 (50)	RECORDSIZE	INFILE, ENVIRONMENT	IRCSZ	data	3	
20	84 (54)	unused - contains zeros					
21	88 (58)	HINDEXDEVICE	INFILE, ENVIRONMENT	IHDEV	data	2	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
22	92 (5C)	PRIMEDATADEVICE	INFILE, ENVIRONMENT	IPDEV	data	2	
23	96 (60)	FIXUNB	INFILE, ENVIRONMENT, RECORDFORMAT	IFUNB	itself	1	
24	100 (64)	FIXBLK	INFILE, ENVIRONMENT, RECORDFORMAT	IFBLK	itself	1	
25	104 (68)	VARUNB	INFILE, ENVIRONMENT, RECORDFORMAT	IVUNB	itself	1	
26	108 (6C)	VARBLK	INFILE, ENVIRONMENT, RECORDFORMAT	IVBLK	itself	1	
27	112 (70)	SPNUNB	INFILE, ENVIRONMENT, RECORDFORMAT	ISUNB	itself	1	
28	116 (74)	SPNBLK	INFILE, ENVIRONMENT, RECORDFORMAT	ISBLK	itself	1	
29	120 (78)	UNDEF	INFILE, ENVIRONMENT, RECORDFORMAT	IUNDF	itself	1	
30	124 (7C)	unused - contains zeros					
31	128 (80)	unused - contains zeros					
32	132 (84)	ENVIRONMENT	OUTFILE	OENV	itself	1	See parms 33-45.
33	136 (88)	RECORDFORMAT	OUTFILE, ENVIRONMENT	ORFMT	itself	1	
34	140 (8C)	BLOCKSIZE	OUTFILE, ENVIRONMENT	OBKSZ	data	3	
35	144 (90)	RECORDSIZE	OUTFILE, ENVIRONMENT	ORCSZ	data	3	
36	148 (94)	unused - contains zeros					
37	152 (98)	HINDEXDEVICE	OUTFILE, ENVIRONMENT	OHDEV	data	2	
38	156 (9C)	PRIMEDATADEVICE	OUTFILE, ENVIRONMENT	OPDEV	data	2	
39	160 (A0)	FIXUNB	OUTFILE, ENVIRONMENT, RECORDFORMAT	OFUNB	itself	1	
40	164 (A4)	FIXBLK	OUTFILE, ENVIRONMENT, RECORDFORMAT	OFBLK	itself	1	
41	168 (A8)	VARUNB	OUTFILE, ENVIRONMENT, RECORDFORMAT	OVUNB	itself	1	
42	172 (AC)	VARBLK	OUTFILE, ENVIRONMENT, RECORDFORMAT	OVBLK	itself	1	
43	176 (B0)	SPNUNB	OUTFILE, ENVIRONMENT, RECORDFORMAT	OSUNB	itself	1	
44	180 (B4)	SPNBLK	OUTFILE, ENVIRONMENT, RECORDFORMAT	OSBLK	itself	1	
45	184 (B8)	UNDEF	OUTFILE, ENVIRONMENT, RECORDFORMAT	OUNDF	itself	1	
46	188 (BC)	unused - contains zeros					
47	192 (C0)	unused - contains zeros					
48	196 (C4)	unused - contains zeros					
49	200 (C8)	REPLACE		REP	itself	1	
50	204 (CC)	NOREPLACE		NREP	itself	1	
51	208 (D0)	REUSE		RUS	itself	1	
52	212 (D4)	NOREUSE		NRUS	itself	1	
53	216 (D8)	STDLABEL	INFILE	ISLBL	itself	1	
54	220 (DC)	NOLABEL	INFILE	INLBL	itself	1	
55	224 (E0)	NOREWIND	INFILE	INREW	itself	1	
56	228 (E4)	REWIND	INFILE	IREW	itself	1	
57	232 (E8)	UNLOAD	INFILE	IUNLD	itself	1	

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
58	236 (EC)	STDLABEL	OUTFILE	OSLBL	itself	1	
59	240 (F0)	NOLABEL	OUTFILE	ONLBL	itself	1	
60	244 (F4)	NOREWIND	OUTFILE	ONREW	itself	1	
61	248 (F8)	REWIND	OUTFILE	OREW	itself	1	
62	252 (FC)	UNLOAD	OUTFILE	OUNLD	itself	1	

RESETCAT FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			RESETCAT
1	8 (8)	CATALOG		CAT	itself	1	See parms 2 and 3.
2	12 (C)	*catname/ password*	CATALOG	CATNM	data	6	
3	16 (10)	*dname*	CATALOG	CATDN	data	2	
4	20 (14)	*password*	MASTERPW	MRPW	data	2	
5	24 (18)	WORKFILE		WFDN	itself	1	See parm 16.
6	28 (1C)	WORKCAT		WCATP	itself	1	See parms 17 and 18.
7	32 (20)	IGNORE		IGN	itself	1	
8	36 (24)	NOIGNORE		NIGN	itself	1	
9	40 (28)	CRAFILES		CFILE	data	3	Count of the number of CRAs provided. See parms 10-12.
10	44 (2C)	*dname*	CRAFILES	CRADN	list of pointers*	2	Each pointer points to a *dname* for the CRA it relates to, in the order they appear in CRAFILES.
11	48 (30)	ALL	CRAFILES	ALLP	list of pointers*	1	Each pointer points to itself.
12	52 (34)	NONE	CRAFILES	NONE	list of pointers*	1	Each pointer points to itself.
13	56 (38)	unused - contains zeros					
14	60 (3C)	unused - contains zeros					
15	64 (40)	unused - contains zeros					
16	68 (44)	*dname/ password*	WORKFILE	WFILE	data	7	
17	72 (48)	*wcatname/ password*	WORKCAT	WCAT	data	6	
18	76 (4C)	*wdname*	WORKCAT	WCATD	data	2	
19	80 (50)	CRAVOLUMES		CRVOLCNT	data	3	Count to the number of CRAs provided. See parms 20, 21, 22.
20	84 (58)	*volser*	CRAVOLUMES	CRVSR	list of pointers*	4	Each pointer points to a *volser* for the CRA it relates to. The *volsers* are pointed to in the order that they appear in CRAVOLUMES.
21	88 (58)	ALL	CRAVOLUMES	ALLV	list of pointers*	1	Each pointer points to itself.
22	92 (5C)	none	CRAVOLUMES	NONEV	list of pointers*	1	Each pointer points to itself.
23	96 (60)	WORKVOLUMES		WVOL	data	4	List of workfile volume serial numbers, optional password specified with the first volume.

VERIFY FDT

Parm No.	Offset	Parm Name	Sub-Parm of	FDT Fieldname	Points to	Data Format Number	Notes
	0 (0)			FDTVERB			VERIFYbb
1	8 (8)	FILE		IN	data	7	*dname/password*
2	12 (C)	DATASET		DSET	data	6	*entryname/password*

* One pointer per object.

Global Data Table—GDT

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	188

Global Data Table Description

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. (See "Processor Invocation" in "Program Organization" for details.)
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. (See also "TEST Option data area.")
20 (14)	4	GDTSTH	Reserved.
24 (18)	4	GDTRIH	Address of Reader/Interpreter historical area.
28 (1C)	4	GDTTPH	Address of Text Processor historical area, the primary Print Control Table (PCT).
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.
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
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).

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
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, IDCSA-TI, (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	Reader/Interpreter name pointer.
120 (78)	4	GDTTOH	I/O Adapter data pointer.
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).
130 (84)	4	GDTGET	Address of entry point to get a logical record, IDCIOGT, (UGET macro).
134 (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.
164 (A4)	4	GDTSNP	Address of entry point to snap dump, IDCSASN, (USNAP macro).
168 (A8)	4	GDTSPR	Address of IDCSA03's storage.
172 (AC)	4	GDTVFY	Address of entry point to VERIFY data set, IDCIOVY (UVERIFY macro).
176 (B0)	4	GDTENQ	Address of entry point to UENQ macro.
180 (B4)	4	GDTDEQ	Address of entry point to DEQ macro.
184 (B8)	4	GDTIFO	Address of entry point to UIOINFO macro.
188 (BC)	4	GDTERR	Address of entry point to UERROR macro.

Input Parameter Table—IPT

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

Input Parameter Table Description

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 *dname.*
12 (C)	4		Address of the output *dname.*
16 (10)	4		Address of the prime data device (PDEV subparameter).
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 entrys.
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	RECORDMODE: Maximum record length of data components. CIMODE: Maximum control interval size of data components.
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.
	...1	CIMODE	This flag is on if output is to be in export control interval mode.
	 xxxx	*	Reserved.
17 (11)	1	PARMOPTS	Flags for parameter options.
	1...	STDLABEL	Standard label option.
	.1..	NOLABEL	No label option.
	..1.	NOREWIND	No rewind option.
	...1	UNLOAD	Unload option.
	.... 1...	NOINDNAME	No INPUT *dname.*
	xxx		Reserved.

The third pointer in the IPT points to an 8-byte input dname (INDDNM).

The fourth pointer in the IPT points to an 8-byte output dname (OUTDDNM).

The fifth pointer in the IPT points to an 8-byte field describing the prime data device (PDEV subparameter) (PDEVNM).

I/O Adapter Historical Area—IODATA

The I/O Adapter historical area is pointed to by GDTIOH. It is built on the first call to the I/O Adapter (UIOINIT macro), 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

I/O Adapter Historical Area Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	IODIOC	First IOCSTR in chain.
4 (4)	4	IODMSG	Reserved—contains zeros.
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:
	2..	IODMID	Module identifier.
	..2	IODINC	Pool identifier.
20 (14)	12	*	Reserved.
32 (20)	4	IODEOD	Address of end-of-data routine for nonVSAM data sets.
36 (24)	4	IODOSS	NonVSAM input SYNAD routine address.
40 (28)	4	IODOSO	NonVSAM output SYNAD routine address.
44 (2C)	4	IODICS	Address of Access Method Services system-input IOCSTR.
48 (30)	4	IODOCS	Address of the Access Method Services system-output IOCSTR.
52 (34)	4	*	Reserved.
56 (38)	4	*	Reserved.
60 (3C)	4	IODAEI	Address of VSAM EODAD routine.
64 (40)	4	*	Reserved.

Input/Output Communications Structure—IOCSTR

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

Input/Output Communications Structure Description

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.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
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	NonVSAM sequential data set.
	..1.	IOCDSOIS	Indexed sequential (ISAM) data set.
	...1	IOCDSOPO	Partitioned data set.
17 (11)	. 1	IOCRFM	NonVSAM 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.
	xxx	*	Reserved.
18 (12)	.. 1	IOCMAC	Macro form used:
	1...	IOCMACIN	Input processing.
	.1..	IOCMACOT	Output processing.
	..1.	IOCMACUP	Update processing.
	...0	IOCMACCR	Keyed sequence.
	...1		Entry sequence.
	.... 0...	IOCMACBK	Logical records.
	.... 1...		Blocks or control intervals.
	.... .0..	IOCMACDR	Sequential processing.
	.... .1..		Direct processing.
	.... ..x.	*	Reserved.
	0	IOCMACCI	Export RECORDMODE processing.
	1		Export CIMODE processing.
19 (13)	. . . 1	IOCMAC2	
	1...	IOCMACSK	Skip sequential processing.
	.1..	IOCMACAS	Asynchronous processing (OS/VS only).
	..1.	IOCMACRR	Relative record processing.
	...1	IOCMACCP	Change processing.
	.... 1...	IOCMACEN	PUT—ENDREQ processing.
	.... .1..	IOCMACPA	Reprocessing flag.
	.... ..1.	IOCMACER	PUT—ERASE processing
	x	*	Reserved.
20 (14)	1	IOCCHP	Change processing modes.
	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.
	1	IOCCHPNU	Change to no update.
21 (15)	. 1	IOCMSG	Message flags:
	1...	IOCHPKE	Change to key equal.
	.1..	IOCHPKG	Change to greater than or equal.
	..1.	IOCMSGOP	Data set is open.
	...1	IOCMSGOE	VSAM OPEN error.
	.... 1...	IOCMSGCE	VSAM CLOSE error.
	.... .1..	IOCMSGAE	VSAM action error.
	.... ..1.	IOCMSGSM	Suppress logical error messages.
	x	*	Reserved.
22 (16)	6	IOCVOLSR	Volume serial number of 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	Control block address.
40 (28)	4	IOCRBA	Record RBA (VSAM).
44 (2C)	4	IOCKYA	Address of key.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
48 (30)	2	IOCPTL	Length of key supplied for position request.
50 (32)	. . 2	IOCPNM	Number of stacked puts.
52 (34)	4	IOCRRN IOCSYSNO	Relative record number. For CRAs this field is passback from UOPEN and contains the two-byte field (in CCB format) that was passed back to UOPEN from IKQASNMT (ASSIGN macro). On UCLOSE, if this field is nonzero, IKQASNMT is called to unassign this logical unit.
56 (38)	4	IOCWORK	Address of input work area.
60 (3C)	4	IOCREL	Relative record number.
64 (40)	4	IOCEXT	IOCSTR extension address.

IOCSTR Extension—IOCSEX

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

IOCSTR Extension Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	4	IOCCBA	Address of ACB or DTF.
4 (4)	4	IOCRPL	Address of VSAM RPL.
8 (8)	2	IOCCBL	Length of ACB or DTF.
10 (A)	. . 2	IOCLRP	Length of RPL.
12 (C)	4	IOCWKA	Address of input work area.
At decimal displacements 16 and 20, one of the two following sets of fields appears:			
16 (10)	4	IOCXAD	External routine address.
16 (10)	4	IOCEXA	VSAM exit list address.
20 (14)	4	IOCXPM	External routine parameter address.
20 (14)	2	IOCEXL	VSAM exit list length.
22 (16)	. . 2		Reserved.
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	Externally controlled data set.
	.1..	IOCFLGDF	Data set is defined.
	..1.	IOCFLGEF	End-of-file on external data set.
	...1	IOCFLGIO	SYSLST or SYSIPT.
	.... 1...	IOCFLGOP	Data set is open.
	.... .1..	IOCFLGOE	Reserved.
	.... ..1.	IOCFLGSP	Access Method Services system-print data set.
33 (21)	. 1	IOCDEV	Device type flags:
	1...	IOCDEVDA	Direct access.
	.1..	IOCDEVMT	Magnetic tape.
	..1.	IOCDEVUR	Reserved.
34 (22)	. . 1	IOCINF	Information flags:
	1...	IOCINFPT	Reserved.
	.1..	IOCINFAE	Reserved.
	..x.	IOCINFND	Reserved.
	...x	IOCINFQX	Reserved.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
	.... 1...	IOCINFAC	ANSI control character.
	.... .1..	IOCINFDO	VSE data set.
	.... ..1.	IOCINFCT	Opened as a catalog.
	x	IOCINFR1	Reserved.
35 (23)	. . . 1	IOCMOD	Additional information flags:
	1...	IOCMODPD	Reserved—contains zero.
	.1..	IOCMODRR	Return RPL address.
	.... 1...	IOCMODUB	User buffering.
	.... .1..	IOCMODXM	Export/import.
	.... ..1.	IOCMODRP	Replace processing.
	1	IOCMODEX	Exclusive control.
36 (24)		IOCDLM	Address of VSE load module.
40 (28)		IOCDNM	Module length.
42 (2A)		IOCVLN	Length of VSE variable blocked remainder.
44 (2C)	1	IOCRCV	Flags for recovery.
	1...	IOCRCVXM	Recovery bit for VSAM.
	.1..	IOCRCVRA	Open CRA.
	..1.	IOCRCVCL	Skip close.
	...x xxxx	*	Reserved—contains zero.

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 SYSLST.

Created by	Modified by	Used by	Size
IDCSA01	UEPIL UPROL macros	IDCDB01	100

Inter-Module Trace Table Description

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.

Created by	Modified by	Used by	Size
IDCSA01	UTRACE macro	IDCDB01	100

Intra-Module Trace Table Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
-6 (-6)	6	*	Table identification 'INTRAb'.
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 'b'.

Modal Verb and Keyword Symbol Table—IDCRIKT

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

Modal Verb and Keyword Symbol Table Description

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.

Open Argument List—OPNAGL

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	80

Open Argument List Description

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	NonVSAM output record format Required.
	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 (SYSIPT) is to be opened. OPNIOC is the only other required field.
	.1..	OPNTYPSO	System output (SYSLST) is to be opened. OPNIOC is the only other required field.
	..1.	OPNTYPCI	Catalog to be opened.
	...1	OPNTYPXM	Export/import.
	.... 1...	OPNTYPRA	Catalog recovery area.
	.... .1..	OPNTYPEX	Exclusive control.
	.... ..1.	OPNTYPRV	VSAM recovery processing.
	1	OPNTYPSY	Reserved. Not used in VSE
3 (3)	. . . 1	OPNMOD	Open modifiers.
	1...	OPNMODPD	Reserved—contains zero.
	.1..	OPNMODAC	Reserved—contains zero.
	..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 the address of the IOCSTR built by the I/O Adapter.
8 (8)	4	OPNDDN	Address of eight-byte D name (not present when SYSIPT or SYSLST is being opened but required at all other times). The D name is the TLBL/DLBL name with one blank on the right.
12 (C)	4	OPNPWA	Address of an optional eight-byte password, used only with VSAM data sets.
16 (10)	4	OPNDSN	Pointer to 44-byte data-set name.
20 (14)	4	OPNCBP	Reserved—contains zeros.
24 (18)	4	OPNDEVDT	Address of device that nonVSAM data set resides on.
28 (1C)	4	OPNDEVIX	Address of device that ISAM index data set resides on.
32 (20)	4	OPNREC	Logical record length, optional.
36 (24)	4	OPNBLK	Block size, optional.
40 (28)	1	OPNKYL	Reserved.
41 (29)	. 1	OPNDSO	Data set organization.
	1...	OPNDSOAM	VSAM data set.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
	.1..	OPNDSOPS	NonVSAM data set.
	..1.	OPNDSOIS	ISAM data set.
	...1	OPNDSOPO	Partitioned data set.
	 xxxx		Reserved.
42 (2A)	. . 1	OPNOPT2	Second option byte.
	1...	OPNOPTAS	Asynchronous processing (OS/VS only).
	.1..	OPNOPTUM	PSNAME is in OPNAGL.
	..x.		Reserved.
	...0	OPNOPTRW	REWIND option.
	...1		NOREWIND option.
	.... 0...	OPNOPTUL	No unload option.
	.... 1...		UNLOAD option.
	.... .0..	OPNOPTSL	NOLABEL option.
	.... .1..		STDLABEL option.
	.... ..0.	OPNOPTCI	Export RECORDMODE option.
	.... ..1.		Export CIMODE option.
	x	*	Reserved.
43 (2B)	. . . 1	OPNSTRN0	Number of strings.
44 (2C)	4	OPNVOL	Pointer to volume serial number.
48 (3C)	1	OPNFRM2	Format related flags.
	1...	OPNFRMNB	Process records without SAM blocking. Set 'u' mode processing in AMDSB if data set is SAM ESDS.
	.xxx		Reserved.
49 (31)	1	OPNOPT3	OPTION related flags.
	1...	OPNOPTSN	Points to a 2-byte logical unit number and not to a data-set name.
	.xxx xxxx		Reserved.
50 (32)	2		Reserved.
52 (34)	4	OPNCAT	Pointer to a 44-byte catalog name to be used to open the given data set.
56 (38)	24		Reserved.

Open Close Address Array—OCARRAY

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
IDCI001	None	IDCI002	20

Open Close Address Array Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	OCATYP	Type of operation: 1 – open, 2 – close.
1 (1)	. 1	OCAOPT	Options:
	1...	OCAOPTCA	Close all open data sets.
2 (2)	. . 1	OCANUM	Number of data sets to open.
	. . . 1	*	Reserved.
4 (4)	4	OCADDR(1)	Address of OPNAGL for open or address of IOCSTR for close.
8 (8)	4	OCADDR(2)	Address of OPNAGL for open or address of IOCSTR for close.
12 (C)	4	OCADDR(3)	Address of OPNAGL for open or address of IOCSTR for close.
16 (10)	4	OCADDR(4)	Address of OPNAGL for open or address of IOCSTR for close.

Phase Table

The Phase Table is a phase (IDCSA04) loaded by IDCSA01 at initialization time. This phase contains an entry for each of the other phases within the Access Method Services system, excluding phase IDCAMS, IDCSA04, and the DTFs. Each entry contains phase status information that is needed for loading the particular phase during Access Method Services execution; only if the CDLOAD anchor table is full. One such entry is described below; the total size of all entries is 768.

Created by	Modified by	Used by	Size
IBM-Supplied	IDCSA02 IDCSA03	IDCSA02 IDCSA03	768

Phase Table Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	8	PLANAME	Name of phase this entry describes.
8 (8)	4	PLAADDR	Address of phase or 0 if not loaded via phase table.
12 (C)	1	PLAUSE	Number of requests to load this phase.
13 (E)	3	PLALN	Phase size in hex.

Positioning Argument List—OPRARG

OPRARG contains the address of the IOCSTR defining the data set to be positioned. It is used within the I/O Adapter.

Created by	Modified by	Used by	Size
IDCIO01	None	IDCIO03	12

Positioning Argument List Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	OPRTYP	Type of operation: 1 indicates POINT or SETL and 3 indicates UIOINFO.
1 (1)	1	OPRPNO	Total number of parameters passed to UIOINFO.
2 (2)	.2	*	Reserved.
4 (4)	4	OPRICS	Address of input IOCSTR (the data set to be positioned).
8 (8)	4	OPROCS	Address of output IOCSTR.

Print Control Argument List—PCARG

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

Print Control Argument List Description

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

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
			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.

Print Control Table—PCT

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	108

Print Control Table Description

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. This bit is set by IDCTP03.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
	...1	PCTLLM	Last line was a message.
	.... 1...	PCTAPF	Alternate 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 SYSLST.
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 alternate print data set.
32 (20)	2	PCTAPN	Current page number on alternate data set.
34 (22)	..2	PCTANL	Next absolute line number on the current page of alternate data set.
36 (24)	8	PCTSTM	Name of the Static Text module presently in virtual storage.
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 URE-SET.
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.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
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.

Reader/Interpreter Communication Area—COMMAREA

The COMMAREA is only used within the Reader/Interpreter to pass information between the phases of the Reader/Interpreter.

Created by	Modified by	Used by	Size
IDCRI01	IDCRI01 IDCRI02 IDCRI03	IDCRI01 IDCRI02 IDCRI03	55

Reader/Interpreter Communication Area Description

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 work area.
16 (10)	2	RISTATUS	Internal error code for the Reader/Interpreter; set to nonzero 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 phase 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.

Reader/Interpreter Historical Area—HDAREA

The Reader/Interpreter Historical Area is created and initialized on the first call to the Reader/Interpreter. 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

Reader/Interpreter Historical Area Description

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 Descriptor module 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.

Scope Structure for UENQ—ENQSCOPE

The scope parameter to ENQ is used for cross system sharing of DASD. The structure is created by the caller of UENQ and passed as the sixth parameter.

Created by	Modified by	Used by	Size
Caller of UENQ	N/A	IDCSANQ	7

Scope Structure for UENQ Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	1	ENQSFLAG	
	10..	ENQSEXT	Scope is external, that is, cross system.
	01..	ENQSVOL	Scope is to be determined by supervisor using a volume serial number (ENQSVID).
	00..		Int lock is required, that is, internal or intra-system.
	6	ENQSVID	VOLID that is used by the supervisor to find the correct DASD device. If the device is shareable, an EXT lock is set. If the device is not shareable, an INT lock (intra-system) is set.

System Adapter Historical Area—SAHIST

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	IDCSA02 IDCSA03	16

System Adapter Historical Area Description

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	PLAPTR	Address of phase table.

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

TEST Option Data Area Description

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 nonzero 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 of entries in areas identification table. One entry exists for each area identifier specified in the PARM command.
4 (4)	2xj	AREADUMP	Areas identifier table 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 table.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
FULLDATA:			
0 (0)	4	FULLINDX	Number of entries in Full Region Dump Table. One entry exists for each full dump.
4 (4)	12xk	FULLDUMP	Full Region Dump Table containing k entries.
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 18 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	IDCTSMP0	IMPORT/IMPORTRA
IDCTSBI0	BLDINDEX	IDCTSPR0	PRINT/REPRO
IDCTSDE0	DEFINE	IDCTSRC0	EXPORTRA
IDCTSDL0	DELETE	IDCTSRI0	Reader/Interpreter
IDCTSEX0	Executive	IDCTSRS0	RESETCAT
IDCTSIO0	I/O Adapter	IDCTSTP0	Text Processor (print chains)
IDCTSLC0	LISTCAT	IDCTSTP1	Text Processor (messages)
		IDCTSTP6	UERROR
IDCTSLC1	LISTCAT (messages)	IDCTSUV0	Universal (any module)
		IDCTSXP0	EXPORT
IDCTSLR0	LISTCRA		
IDCTSLR1	LISTCRA (messages)		

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

Text Structure Description

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 Description

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 $2 \times n + 2$ from the beginning of the Text Structure module.

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0 (0)	2	TXT^n	Length in bytes of the text entry that follows (not including these header fields).
2 (2)	2	FLG^{n}A	Flag byte:
	1...		Message entry.
	.1..		Header entry.
	..1.		Secondary message entry.
The following two fields only exist if this is a text entry for a header line:			
4 (4)	2	HDLI^n	The number of printable header lines.
6 (6)	2	HDSP^n	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 area is linked into a chain with other areas allocated by UGPOOL. Each such area is preceded by 16 bytes, as shown here.

Created by	Modified by	Used by	Size
IDCSA02	None	IDCSA02	16

UGPOOL Area Description

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.
12 (C)	4	GPID	Area identification code.
The storage area in the UGPOOL chain for an 'xxPG' storage identification has the following format:			
0 (0)	4	GPFORWRD	Address of next UGPOOL area.
4 (4)	4	GPBACK	Address of last UGPOOL area.
8 (8)	4	GPLEN	Length of this area = 24 (X'00000018')
12 (C)	4	GPID	Area identification code.
16 (10)	4	GPADRPG	Address of 'xxPG' storage area.
20 (14)	4	GPLENPG	Length of 'xxPG' storage area.

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

UGSPACE Area Description

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	bbbb for UGSPACE area.

UIOINFO—Option Byte and Return Area

The UIOINFO option byte is used by an FSR to indicate the type of data to be retrieved by the UIOINFO macro. The data retrieved is passed back by UIOINFO in the return area.

UIOINFO Option Byte Description

Offset	Bytes and Bit Pattern	Field Name	Description: Content, Meaning, Use
0(0)	1	IOINFOPT	
	1...	IOINFDVT	Retrieve 8-byte device type.
	.1..	IOINFVOL	Retrieve up to five volume serial numbers.
	..1.	IOINFDSN	Retrieve 44-byte data set name.
	...1	IOINFSUP	Suppress error message.
	.... 1...	IOINFTMS	Retrieve format-4 time stamp.
	.... .1..	IOINFOPT	Retrieve up to five Logical Unit Blocks.
	.... ..1.	IOINFVID	Parameter 4 passed to UIOINFO is a 6-byte *volid*, not a *dname*.

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 follows:

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		Timestamp. Bytes: 0-1 Identifier—X'0004'. 2-3 Length of data returned. 4-11 New timestamp. 12-19 Old timestamp.
n		Logical Unit (LUB) List (variable). Bytes: 0-1 Identifier—X'0005' 2-3 Length of data returned. 4-5 First LUB in same format as in a CCB. . . . . . . (n+1)-(n+2) Last LUB

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.

PCRST—Change Subtitle Lines

Created by	Modified by	Used by	Size
All routines	None	IDCTP01	Variable

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.

Chapter 6: 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 *DOS/VSE Serviceability Aids and Debugging Guide.* This manual and other publications that may be helpful are listed in the preface to this book.

Four major diagnostic aids are provided by the processor:

- Trace tables, which provide a trace of the flow of control between phases and CSECTs and within phases and CSECTs.
- 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.
- ABORT codes and full partition 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* phases and CSECTs, and the Intra-Module Trace Table, which records the flow of control *within* phases and CSECTs.

You can find the trace tables in any full partition dump, or you can print them using the Test option. The section "Reading a Dump" in this chapter explains how to find the tables in a dump; the section " Test Option" in this chapter explains how to print them.

Inter-Module Trace Table

The Inter-Module Trace Table begins with the characters INTER and contains the IDs of the last twenty phases and CSECTs that had control. The IDs are the last four characters of the phase or CSECT 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 phase or CSECT is entered, but also upon return from a phase or CSECT. 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 phases and CSECTs. Each phase and CSECT has trace points placed at key locations, for example, at the start of procedures.

The IDs of the trace points consist of four characters: the first two characters are the mnemonic identifier of the phase or CSECT being traced, and the last two characters identify a specific point within the phase or CSECT.

(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 phase or CSECT 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
```

by referring to this list, you would know that the last trace point encountered was at the start of the routine in CSECT IDCSA02 that processes a UGPOOL macro request.

For the period of time the Test option is set, 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 phase or CSECT can be found by examing 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

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 Text 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 EX1F and AL01 is encountered.

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 6-1 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.

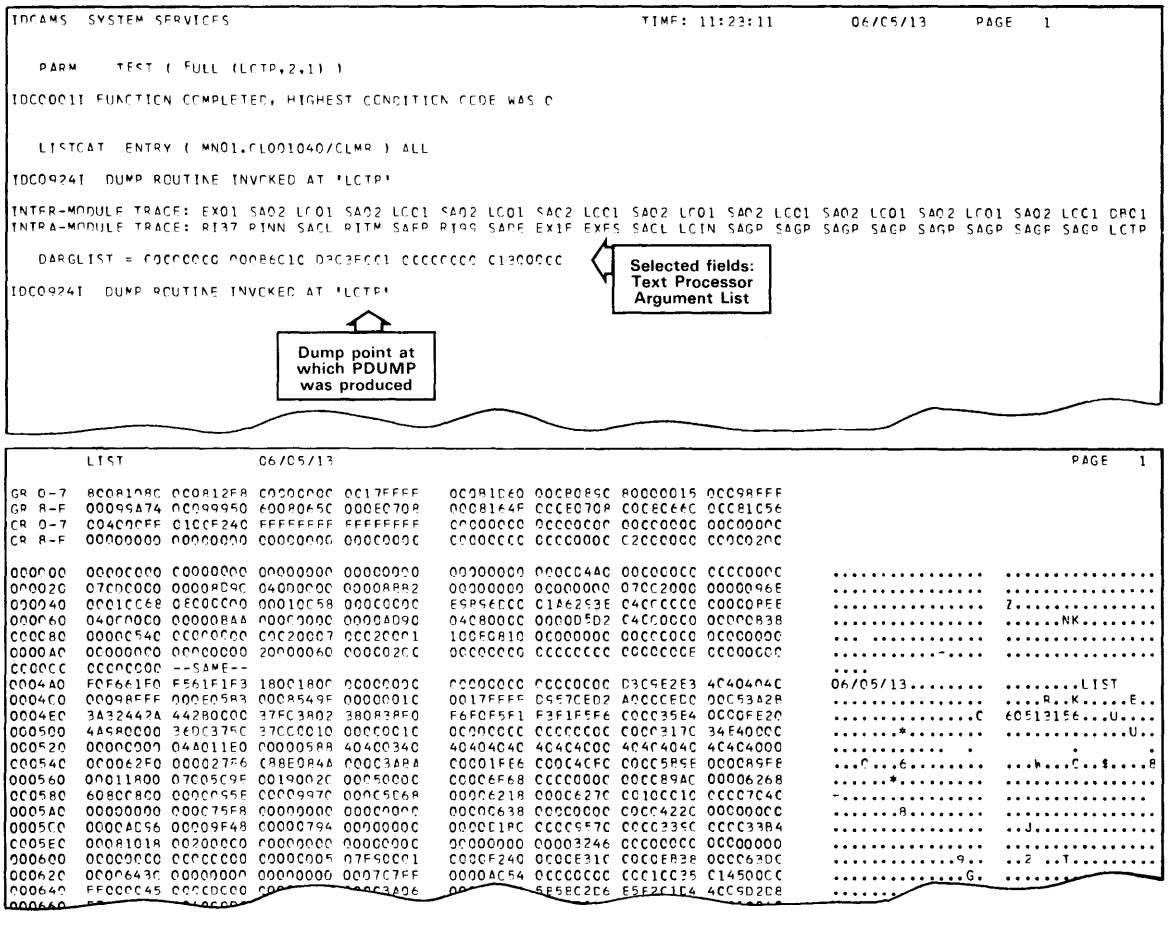


Figure 6-1. Example of Test Option Output

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 uou 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.

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 Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
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 builds the catalog parameter list.
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	dump	On entry to INDEXPRC.
AL81	IDCAL01	PARAMCHK	trace	On entry to PARAMCHK procedure.
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.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
CL01	IDCCL01	IDCCL01	trace	Start of CANCEL command.
CL02	IDCCL01	IDCCL01	trace	After printing of command - complete message.
CL03	IDCCL01	IDCCL01	trace	If CANCEL STEP. Just before return to executive.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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	IDCDE02	MODELPRC	dump	After calling the catalog to locate a model object.
DE04	IDCDE02	MODELPRC	dump	End of procedure that built the model table.
DE11	IDCDE01	IDCDE01	trace	Start of DEFINE FSR.
DE20	IDCDE03	IDCDE03	trace	On entry to IDCDE02 module.
DE21	IDCDE03	CTLGPROC	trace	Start of procedure that defines a master or user catalog.
DE22	IDCDE03	DSETPROC	trace	Start of procedure that defines a VSAM data set.
DE23	IDCDE03	DSPACPRC	trace	Start of procedure that defines a data space.
DE24	IDCDE03	NVSAMPRC	trace	Start of procedure that defines a nonVSAM data set.
DE25	IDCDE03	AIXPROC	trace	Start of procedure that defines an alternate index.
DE26	IDCDE03	PATHPROC	trace	Start of procedure that defines a path.
DE30	IDCDE02	IDCDE02	trace	Entry to IDCDE02.
DE31	IDCDE02	NAMEPROC	trace	Start of procedure that builds CTGFLs with name and date 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 information.
DE35	IDCDE02	IXOPPROC	trace	Start of procedure that initializes index fields in the AMDSBCAT.
DE36	IDCDE02	MODELPRC	trace	Start of procedure that locates the model object entry.
DE37	IDCDE02	FREESTG	dump	End of DEFINE FSR.
DLBC	IDCDL01	BUILDCPL	trace	Start of procedure that builds the CTGPL for the delete request.
DLBG	IDCDL01	IDCDL01	dump	Start of DELETE FSR.
DLCL	IDCDL01	CLEANUP	trace	Start of procedure that closes the user catalog.
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.
DLND	IDCDL01	IDCDL01	dump	End of DELETE FSR, before data sets are closed and the completion message is issued.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
DLOP	IDCDL01	CATOPEN	trace	Start of procedure that opens the user catalog.
DLPC	IDCDL01	PARAMCHK	trace	Start of procedure that checks for invalid parameters.
DLVL	IDCDL01	FINDTYPE	dump	Before and after calling the catalog to locate the entry type.
DLVS	IDCDL01	CATCALL	dump	Before and after calling the catalog to delete an entry.
DLVT	IDCDL01	MORESP	dump	Either side of UCATLG macro in MORESP.
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.
EX2X	IDCEX02	SCANPARM	trace	Before processing the caller's parameter list.
EX3S	IDCEX03	SCANPARM	trace	Before processing the caller's parameter list.
IOAC	IDCIO02	BUILDACB	dump	After ACB and EXLST have been built, at end of procedure.
			trace	Start of procedure that builds the ACB and EXLST.
IOCL	IDCIO01	IDCIOCL	trace	Start of routine that closes data set.
IOCP	IDCIO01	IDCIOCO	trace	Start of routine that copies a data set.
IODC	IDCIO02	BUILDDBK	trace	Start of procedure that builds a DTF.
IODS	IDCIO02	DSDATA	dump	After obtaining file information from the label cylinder.
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.
IOE2	IDCIO01	GETNONVS	trace	Start of end-of-file routine for a nonVSAM data set.
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.
IOIF	IDCIO03	DSINFO	trace	Entry to UIOINFO processing.
IOIT	IDCIO01	IDCIOIT	trace	Start of initialization routine.
IOII	IDCIO03	DSINFO	dump	After return from IKQVDTPE.
IOOP	IDCIO01	IDCIOOP	trace	Start of routine that opens data sets.
IOOT	IDCIO03	PTISDS	trace	Before SETL macro is issued.
IOPL	IDCIO01	PUTREP	trace	Entry to PUT (Replace) routine.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
IOPO	IDCIO01	IDCIOPO	trace	Start of routine that positions to a data record in an opened VSAM or ISAM data set.
	IDCIO03	IDCIO03	dump	After positioning is complete, before returning control to IDCIO-PO.
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.
IORP	IDCIO02	BUILDRPL	dump	After RPL is built, at end of procedure.
IOS2	IDCIO01	GETNONVS	trace	Start of SYNAD routine for nonVSAM read error.
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.
IOUO	IDCIO01	IDCIOSI	trace	Entry to UIOINFO entry process-ing.
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	Before the GET macro is issued for a VSAM data set.
IOVP	IDCIO01	PUTVSAM	dump	End of procedure that writes a VSAM record.
			trace	Before the PUT macro is issued for a VSAM data set.
IOVR	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.
IO02	IDCIO03	DSINFO	dump	After formatting work area.
IO1C	IDCIO02	CLOSERTN	dump	Before CLOSE macro is issued.
IO1O	IDCIO02	OPENRTN	dump	Before OPEN macro is issued.
IO2C	IDCIO02	CLOSERTN	dump	At completion of all UCLOSE processing.
IO2P	IDCIO01	PUTNONVS	dump	After writing a spanned record.
			trace	After writing a spanned record.
IO20	IDCIO02	OPENRTN	dump	After OPEN macro is issued.
IO21	IDCIO02	OPENRTN	dump	At completion of all UOPEN processing.
LCAL	IDCLC02	LOCPROC	dump	After calling the catalog to locate an entry.
LCAU	IDCLC02	AUPROC	trace	Start of procedure that formats catalog fields for a nonVSAM or user catalog entry.
LCBL	IDCLC02	LOCPROC	dump	Before calling the catalog to locate an entry.
LCCL	IDCLC02	CDIPROC	trace	Start of procedure that formats catalog fields for a cluster, data, or index entry.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
LCDC	IDCLC02	DEVTCONV	trace	Start of procedure that converts UCB code.
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.
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.
LCRT	IDCLC01	RTEPROC	trace	Start of procedure that directs the retrieved entry to the proper formatting procedure.
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.
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.
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 the completion message.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
LRDO	IDCLR01	DOOTHR	dump	Start of procedure that controls printing nonVSAM information.
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 an object's 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.
LRZZ	IDCLR01	ERROR	dump	After error that forced an ABORT of this execution.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
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.
MPGK	IDCMP01	DVOLCHK	trace	Entry to DVOLCHK
MPGL	IDCMP01	DVOLPROC	trace	Entry to DVOLPROC
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.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
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.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
RC33	IDCRC02	ALSPROC	trace	Start of procedure which processes catalog information for alias associations for nonVSAM objects.
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.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
RIDF	IDCRI01	DEFAULTS	dump	Start of procedure that adds default parameters to the FDT.
RIEX	IDCRI01	IDCRI01	dump	Start of Reader/Interpreter phase.
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.
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 phase 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 phase 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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
R104	IDCRI01	GETNEXT	trace	Modal command other than ELSE within an IF.
R105	IDCRI01	GETNEXT	trace	Found a functional command.
R109	IDCRI01	KWDPARM	trace	Found a keyword subparameter.
R111	IDCRI01	GETDATA	trace	Start of extracting a scalar value.
R112	IDCRI01	GETDATA	trace	Extract a character string.
R116	IDCRI02	IDCRI02	trace	Prior to loading the command descriptor.
R117	IDCRI02	IDCRI02	trace	Beginning of the code sequence to build the PARMINFO table.
R124	IDCRI01	CONVERT	trace	Start converting a binary number.
R127	IDCRI01	CONVERT	trace	Start converting a hexadecimal number.
R130	IDCRI01	CONVERT	trace	Change converted digits into a binary fullword.
R135	IDCRI01	INREPEAT	trace	Loop to reset parameter occurrence flags for possible parameters in the sublist.
R136	IDCRI01	INREPEAT	trace	End of last repeated sublist.
R144	IDCRI01	SETDFLT	trace	Found that default is allowable; ready to put in FDT.
R145	IDCRI01	SETDFLT	trace	Move a defaulted unquoted constant to FDT.
R149	IDCRI01	NXTFIELD	trace	Extract a filed from input (verb, keyword, or scalar).
R150	IDCRI01	NXTFIELD	trace	Extract a keyword field.
R151	IDCRI01	NXTFIELD	trace	Extract a quoted scalar.
R156	IDCRI01	NEXTCHAR	trace	End-of-file already found in input.
R157	IDCRI01	NEXTCHAR	trace	Extract first character of a new command.
R159	IDCRI01	NEXTCHAR	trace	End-of-file found while looking for next character.
R160	IDCRI01	SCANENDS	trace	Skip leading blanks and comments if preceding record indicated continuation.
R161	IDCRI01	SCANENDS	trace	Bypass a leading comment.
R162	IDCRI01	SCANENDS	trace	Bypass leading blanks.
R166	IDCRI01	DSPLCALC	trace	Calculate displacement into the FDT for a parameter in a first-level repeated parameter list.
R199	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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
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.
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.
RMEV	IDCRM01	CLUSPROC	trace	End of cluster or alternate index define sequence.
RMFN	IDCRM01	IDCRM01	dump	Termination of IDCRM01.
RMFV	IDCRM01	FVTPROC	trace	Entry to FVTPROC.
RMGD	IDCRM01	GDGPROC	trace	Entry to GDGPROC.
RMGK	IDCRM01	DVOLCHK	trace	Entry to DVOLCHK.
RMGL	IDCRM01	DVOLPROC	trace	Entry to DVOLPROC.
RMLV	IDCRM01	LVLRPROC	trace	Entry to LVLPROC.
RMMG	IDCRM01	MSGPROC	trace	Entry to MSGPROC.
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.
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.
RPDI	IDCRP01	CATRELOD	dump	At the end of all reload error checking before any updates have been done to the target catalog.
RPKS	IDCRP01	IDCRP01	trace	When a KSDS with shareoption 4 in non-load mode has been detected (after open). The file must be closed and reopened with KEYED in the ACB.
RPRO	IDCRP01	IDCRP01	trace	Failure to reopen a file.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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 maps 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.
RPIO	IDCRP01	DUMPIT	dump	After read or write to backup or target catalog.
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.
RSAD	IDCRS05	ADDUPCR	trace	Upon entry to routine which updates the CRA for a particular record.
RSAE	IDCRS01	AERROR	trace	On entry to routine that exists if enough storage is not available to establish automatic storage required for RESETCAT modules.
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 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 or BBBB to TTnn.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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 free.
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.
RSC3	IDCRS01	CLEANUP	dump	Before freeing the resources used by RESETCAT.
RSC4	IDCRS05	CKERR	dump	Before RESETCAT FSR is terminated due to an error.
RSC7	IDCRS07	CATEOV	dump	At conclusion of routine that extends the catalog.
RSDA	IDCRS07	HVTOC	trace	On entry to routine that processes all common VTOC handler functions.
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.
RSD4	IDCRS07	HVTOC	dump	At conclusion of routine that processes all common VTOC handler functions.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
RSE2	IDCRS01	ENSURECI	dump	A start of procedure prior to ensuring enough free CIs.
RSF1	IDCRS04	FIND	trace	On entry to routine that locates requested information from a set of catalog records.
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.
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 RESET-CAT.
RSI1	IDCRS01	INIR	dump	End of procedure that initializes data areas and obtains resource.
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	MERGECRA	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 reeset 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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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.
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	IDCPS07	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	IDCRS07	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	IDCRS03	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 that 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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase 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 labels 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	IDSCR01	IDCRS01	trace	Upon entry to main RESETCAT module.
SAAB	IDCSA01	SAABT	dump	In UABORT routine when a dump is not to be printed for a "no space available" condition.
SACA	IDCSA02	IDCSA02	trace	Start of routine that processes UCATLG macro.
SACL	IDCSA02	IDCSA02	trace	Start of routine that processes UCALL macro.
SADE	IDCSA02	IDCSA02	trace	Start of routine that processes UDELETE macro.
SADQ	IDCSA08	IDCSA08	trace	Start of routine that processes UDEQ macro.
SAEP	IDCSA01	PRNTERR	trace	Entry to routine which prints an error message via EXCP.
SAFP	IDCSA02	IDCSA02	trace	Start of routine that processes UFPOOL macro.
SAFS	IDCSA02	IDCSA02	trace	Start of routine that processes UFSPACE macro.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
SAGP	IDCSA02	IDCSA02	trace	Start of routine that processes UGPOOL macro.
SAGS	IDCSA02	IDCSA02	trace	Start of routine that processes UGSPACE macro.
SALD	IDCSA02	IDCSA02	trace	Start of routine that processes ULOAD macro.
SANQ	IDCSA08	IDCSA08	trace	Start of routine that processes UENQ macro.
SASN	IDCSA02	IDCSA02	trace	Start of routine that processes USNAP macro.
SATI	IDCSA02	IDCSA02	trace	Start of routine that processes UTIME macro.
SA05	IDCSA05	IDCSA05	trace	Before the TIME macro is issued.
TPCC	IDCTP01	IDCTPPR	trace	Before the call to the CONVERT routine is issued.
TPEA	IDCTP06	IDCTP06	dump	Start of UERROR procedure.
TPEB	IDCTP06	IDCTP06	dump	Before a converted UERROR message is printed.
TPER	IDCTP01	ERROR	dump	Start of procedure that prints a text processor error message.
TPE1	IDCTP06	IDCTP06	trace	Start of UERROR procedure.
TPE2	IDCTP06	CATERCNV	trace	Entry point to routine that converts catalog error messages to prose.
TPIN	IDCTP01	IDCTPPR	dump	At end of phase; the format structure for a UPRINT macro has been processed.
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.
TP4G	IDCTP04	INITPCT	trace	Get print control table after freeing any previous secondary PCT and before GETVIS for the new PCT.
TP4I	IDCTPO4	ESTSCONT	trace	Start of establishment of print control table.
TP4R	IDCTP04	RESTCONT	dump	End of procedure that processes UREST macro.
TP4S	IDCTP04	ESTSCONT	dump	End of procedure that processes UESTS macro.
TP5E	IDCTP05	IDCTP05	trace	Start of procedure that gets a static text module.
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.
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.

Trace and Dump Points to Phase or CSECT Cross Reference

Trace or Dump Point	Phase or CSECT	Procedure	Type	Situation at Dump or Trace Point
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 alternate index 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.
XPWC	IDCXP01	CLUSPROC	trace	Before writing the catalog information to the portable data set.
XPZX	IDCXP01	MORESP	dump	Just after the UCATLG macro.
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.
ZZCA	IDCSA02	IDCSA02	dump	Before and after CATLG macro is issued to invoke catalog management routines.

Module to Dump Points Cross Reference

The dump points, set up by UDUMP macros, have been placed at key locations in each phase and CSECT, for example, around calls to other processor and non-processor phases or CSECTs. Each dump point specifies the information that can be dumped at that point. Some dump points allow symbolic dumping of selected fields, for example, parameter lists or return codes; all dump points allow dumping of the full partition and printing of the trace tables.

The following list contains the dump points within each phase or CSECT 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.

Phase or CSECT to Dump Points Cross Reference

Phase 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.
	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.
	INDEXPRC	AL61	dump	On entry to INDEXPRC.
	LOCATPRC	AL03	dump	After calling the catalog to alter 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 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 that locates information about the base cluster and alternate index.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
		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.
IDCDE01	IDCDE01	DE01	dump	Before calling the catalog to define an object.
		DE02	dump	End of DEFINE FSR, before completion message is issued.
IDCDE02	MODELPRC	DE03	dump	After calling the catalog to locate a model object.
		DE04	dump	End of procedure that built the model table.
	FREESTG	DE37	dump	End of DEFINE FSR.
IDCDL01	CATCALL	DLVS	dump	Before and after calling the catalog to delete an entry.
	FINDTYPE	DLVL	dump	Before and 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.
	MORESP	DLVT	dump	Before and after the UCATLG macro in MORESP.
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.
	PUTREP	IOGR	dump	After the GET for update.
		IOPR	dump	After the PUT for update.
	PUTVSAM	IOVP	dump	End of procedure that writes a VSAM record.
	VSAMERR	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	dump	At end of all UCLOSE processing.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	DSDATA	IODS	dump	After obtaining file information from the label cylinder.
	OPENRTN	IO1O	dump	Before OPEN macro is issued.
		IO20	dump	After OPEN macro is issued.
		IO21	dump	At end of all UOPEN processing.
IDCIO03	DSINFO	IO11	dump	After return from IKQVDTPE.
		IO02	dump	After formatting the work area.
	IDCIO03	IOPO	dump	After positioning is complete, before returning control to IDCIO-PO.
IDCLC01	IDCLC01	LC99	dump	End of LISTCAT FSR, before freeing storage in IDCLC01.
	ERRPROC	LCMG	selected areas[1]	Before UPRINT macro is issued to print a message.
	FREESTG	LC98	dump	End of LISTCAT FSR, before freeing storage in IDCLC02.
IDCLC02	IDCLC02	LC02	dump	When IDCLC02 is called the first time to establish addressability.
	LISTPROC	LCTP	selected areas[2]	Before UPRINT macro is issued to print catalog data.
	LOCPROC	LCAL	selected areas[3]	After calling the catalog to locate an entry.
		LCBL	selected areas[4]	Before calling the catalog to locate an entry.
		LCWA	selected areas[5]	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 "home" 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 completion 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 controls printing nonVSAM information.
	DOVSAM	LRDV	dump	Start of procedure that controls printing VSAM information.
	ERROR	LRER	dump	Start of procedure that handles errors.
		LRZY	dump	After error message has been printed.
		LRZZ	dump	After error that forced an ABORT of this execution.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	GETPRT	LRGE	dump	Start of procedure that gets and prints records.
	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 and compares 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 an objects 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.
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.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	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.
	INIT	RC80	dump	Before initializing to begin processing.
	TERM	RC79	dump	Before special processing to terminate request (closing output data set.)
IDCRC02	CLUSPROC	RC06	dump	Start of procedure that processes VSAM objects.
	CTLGPROC	RC16	dump	Start of procedure that issues catalog locates.
	IDCRC02	RC02	dump	Start of main procedure.
	IDCRC02	RC04	dump	Before termination processing.
	NVSMPROC	RC28	dump	Start of procedure that 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

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
				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.
	MORSPACE	RIMC	dump	Start of procedure that scans an FDT space for a list of constants.
	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 full-word.
	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 P 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.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	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	RPC1	selected areas[6]	On exit from procedure that translates control interval numbers on the backup catalog.
	DUMPIT	RPIO	selected areas[7]	After read or write to backup or target catalog.
	TRUENAME	RPTU	selected areas[8]	On exit from procedure having built truenamerange table.
IDCRS01	CATINIT	RSC1	dump	End of procedure that builds CIN to RRN table.
	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.
	UPDCRA	RSU2	dump	End of procedure that updates the CRA from the workfile.
	IDSCR01	RS00	dump	End of RESETCAT FSR.
IDCRS02	ASSOC	RSA1	dump	End of procedure that ititiates association checking.
	PROCTYPE	RSP3	dump	After processing a set of records for associations.
	PROCCI	RSP4	dump	End of procedure that ensures that a contol 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	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.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
	VERB	RSV4	dump	Before freeing resources used by procedure which verifies GDG data sets.
IDCRS04	DELGO	RSD3	dump	End of procedure that deletes a group occurrence.
	FIND	RSF1	dump	End of routine that finds one or all group occurrences.
	MODGO	RSM2	dump	End of procedure that modifies a group occurrence.
IDCRS05	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	DSOPEN	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	CATEOV	RSC7	dump	At conclusion of routine that extends the catalog.
	HVTOC	RSD4	dump	At conclusion of routine that processes all common VTOC handler functions.
	RENAMEP	RSR4	dump	Before freeing resources used by the RENAMEP procedure.
	UPDCAT	RSU1	dump	End of procedure that updates the catalog from the workfile.
IDCSA01	SAABT	SAAB	dump	In UABORT routine, when a dump is not to be printed for a "no space available" condition.
IDCSA02	IDCSA02	ZZCA	dump	Before and after CATLG macro is issued to invoke catalog management routines.

Phase or CSECT to Dump Points Cross Reference

Phase or CSECT	Procedure	Dump Point	Type	Situation at Dump Point
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	TPSI	dump	After initialization of text processor parameters.
		TPIN	dump	At end of phase; the format structure for a UPRINT macro has been processed.
	LINPRT	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.
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.
		TPEB	dump	Before a converted UERROR message is printed.
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 UERROR procedure.
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.
	MORESP	XPZX	dump	Just after the UCATLG macro.

Selected Area Footnotes:

The following list describes the selected areas pointed at the specified dump points. On the printed output, the area title precedes each area dumped.

Dump Point	Area Title	Area Description
1. LCMG	ERRDARG	Text processor argument list (DARGLIST) used for printing messages
2. LCTP	DARGLIST	Text processor argument list (DARGLIST) used for printing the catalog area
3. LCAL	CATRC	VSAM catalog return code
	CTGENT	VSAM locate key (either the entryname or the CI number)
	CTGPSWD	User supplied password
	CTGPL	VSAM catalog parameter list

Dump Point	Area Title	Area Description
	CTGFL array	VSAM field parameter list.
	FPL (1)	Note: The number of FPLs
	.	(nn) varies with the amount
	.	of catalog information
	.	requested (i.e., NAME, HISTORY,
	FPL (nn)	VOL, etc.)
	MULTIFPL	VSAM field parameter list if a special function FPL is required
4. LCBL	Same as LCAL	
5. LCWA	CTGWKAPT	Workarea address of VSAM returned cataloged fields
	CTGWKA array	VSAM returned catalog fields
	WKA (1)	Note: This workarea is dumped as an array of 256 byte blocks and the last block less than 256 bytes is
	WKAEND	indicated as WKAEND.
6. 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#)
7. 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
	INSOTREC	A record which was inserted into the target catalog because it existed in the backup catalog but not in the target catalog
	UPOUTREC	A record which was 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
8. RPTU	SORSTABL	A table which maps the extents of the high key range of the backup catalog. Each entry maps one extent and contains: Word 1 - High RBA of the extent Word 2 - Number of CI's 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

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 message IDC4999I on SYSLST giving the ABORT code and, with the exception of code 28 and code 68, produces a full partition PDUMP with the ABORT code in register 15; the ABORT code indicates the type of error that occurred.

The following list identifies the ABORT codes set by the processor and the phase or CSECT and procedure that sets each ABORT code. The list also explains the situation that caused the ABORT condition.

ABORT Codes

ABORT Code	Phase 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	IDCTP04	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 avilable could not be issued because the SYSLST data set is not open.
		BUILDDBK	Storage was not available to load the phase that contains the DTF and access method routines (IDCDIxx).
	IDCSA01	GETCORE	Storage was not available for the automatic storage required for IDCSA02, IDCSA03, IDCIO01, or IDCTP01.
	IDCSA02	IDCSA02	The CDLOAD Anchor Table was full and storage was not available to load the phase requested by a UCALL or ULOAD macro.
	IDCSA02	IDCSA02	The CDLOAD Anchor Table was not full but storage was not available for CDLOAD to load the phase.
	IDCSA03	GETCORE	Storage was not available for the automatic storage required by a phase.
	IDCTP01	LINEPRT	Storage not available for new header line.
	IDCTP01	ERROR	Storage not available to save Conversion Table (CVPSTRU).
	IDCTP05	IDCTP05	Storage not available for static text entry.
	IDCTP04	ESTSCONT	Storage not available for Print Line Stack Buffer.
	IDCTP04	PCTSETUP	Storage not available for Print Chain Translate Table.
	IDCTP04	PCTSETUP	Storage not available for primary or secondary Print Control Table.
	IDCTP04	PCTSETUP	Storage not available for sub-title line or footing line change.
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.

ABORT Codes

ABORT Code	Phase or CSECT	Procedure	Situation that Caused ABORT
36(24)	IDCIO02	BLDOCMSG	The SYSLST data set could not be opened, or the SYSLST data set has already been closed and a message cannot be issued.
	IDCTP01	STACKPUT	An attempt to write to the output data set has failed.
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 three and less than 6.
	IDCSA02	IDCSA02	The argument list of a UGSPACE, UGPOOL, or UFPOOL macro is invalid.
	IDCSA05	IDCSA05	The argument list for the UTIME macro is invalid.
52(34)	IDCSA02	IDCSA02	The phase to be loaded (because the CDLOAD Anchor Table is full) was not found in the Phase Table.
64(40)	IDCSA01	IDCSA01	The CDLOAD macro failed loading phase IDCSA04 which contains the Phase Table.
	IDCSA02	IDCSA02	The CDLOAD macro failed loading a phase because the phase was not found.
68(44)	IDCSA01	IDCSA01	The initial GETVIS for IDCSA01's automatic storage failed.
72(48)	IDCRS05	CKERR	An internal RESETCAT error occurred. This situation should not occur in a working program.
76(4C)	IDCCL01	IDCCL01	CANCEL command was executed in the command stream, with CANCEL JOB option.

You can find UABORT macros by examining the microfiche listings. The expansion of a UABORT macro for an ABORT code of 60 looks like this:

```
RESPECIFY(REG13,REG14,REG15) RSTD;
REG15 = 60;
REG14 = GDTABT;
REG13 = GDTABH;
GEN(BR    REG14);
RESPECIFY(REG13,REG14,REG15)UNRSTD;
```

Reading a Dump

This section describes how to find phases and data areas belonging to the processor in a full partition dump, either a PDUMP or a system dump.

PDUMPs are produced by the processor on two different occasions. If the Test option is set and the FULL keyword is specified, the processor produces as many PDUMPs as requested, at the points requested. The processor prints a message following each such PDUMP to identify the point at which the dump was produced. If an ABORT condition occurs, the processor again produces a PDUMP except in the case of ABORT conditions 28 and 68. An ABORT PDUMP can be distinguished from a system dump because there is no system error message and the ABORT dump is preceded by message IDC4999I giving the ABORT condition code.

All executable phases, CSECTs, and certain data areas belonging to the processor are preceded by an EBCDIC character string to identify it. Phases and CSECTs are preceded by their full name, for example, IDCTP01b. (The

date of compilation, in character form, follows the 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 Global Data Table 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 Processor Phases

The System Adapter normally loads phases using the CDLOAD macro. Thus, you can use the Anchor Table to find where each phase has been loaded.

If, however, the Anchor Table is full, the System Adapter obtains storage for the phase to be loaded using the GETVIS macro and loads the phase into this area. You can find where these phases have been loaded from the Phase Table. The fourth word of the System Adapter historical area points to the Phase Table; however, the Phase Table normally follows the Global Data Table and the trace tables in a dump. The section "Data Areas" shows the format of the Phase Table.

Figure 6-2, Part 2, shows how the Phase Table appears in a dump. You can tell that no phases have been loaded using the Phase Table because all the phase addresses contain zeros.

How to Find the Module and Registers at Time of the Dump

The best way to determine which phase or CSECT caused the dump and to find the registers of that phase or CSECT varies according to the type of dump you have.

In a system dump, standard methods explained in your operating system's *Debugging Guide* should be used.

In a PDUMP caused by an ABORT condition, the last entry in the Inter-Module Trace Table identifies the phase or CSECT that issued the UABORT macro. Register 15 of the registers at the top of the dump contains the ABORT code set in the UABORT macro. Once you know the ABORT code and the phase or CSECT that issued the UABORT macro, you can use Figure 6-2 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 phase or CSECT 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 PDUMP produced at a dump point, the trace tables printed after the dump tell you at what point the dump occurred. The next to the last ID in the Inter-Module Trace Table identifies the phase or CSECT that called the dumping routine; 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 after 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. Register 13 at the top of the dump points to IDCDB01's save area. The first word of this save area contains the characters DB01; the word immediately preceding the previous

save area in the save area chain contains the ID of the phase or CSECT that issued the UDUMP macro.

Figure 6-2, Part 1, illustrates how to find the phase or CSECT that caused the dump and its registers in a PDUMP 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 6-2, Part 1) so you may be able to find it by scanning down the right side of the dump. The GDT follows right after the first phase (IDCAMS) of the processor and the Anchor Table. A more systematic way of finding the GDT depends upon the type of dump you have. Figure 6-3 shows the two methods of finding the GDT and is referred to in the following paragraphs.

In a PDUMP produced as the result of an ABORT condition, you must use Method 1 shown in Figure 6-3. The GDT is contained in the System Adapter's (IDCSA01) automatic storage area. Register 11 of the registers at entry to PDUMP points to the automatic storage area of IDCSA01. The GDT is at location GDTTBL in the storage area; you must examine the microfiche listing for IDCSA01 to find the offset of location GDTTBL. Add the offset of location GDTTBL to the contents of register 11 to obtain the address of the GDT.

In a system dump, if the dump 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 the top of the dump to the offset of GDTTBL, to find the GDT.

If the system dump occurred after IDCSA01 called IDCEX01, use Method 2 shown in Figure 6-3. 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. The first word of this save area contains EX01. 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 PDUMP 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 6-2, Part 1, shows the GDT as it appears in a dump.

How to Find Save Areas

The first word of the standard save area for processor phases and CSECTs contains the ID of the phase or CSECT that saved its caller's registers in that save area. (The ID is the last four characters of the phase or CSECT 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

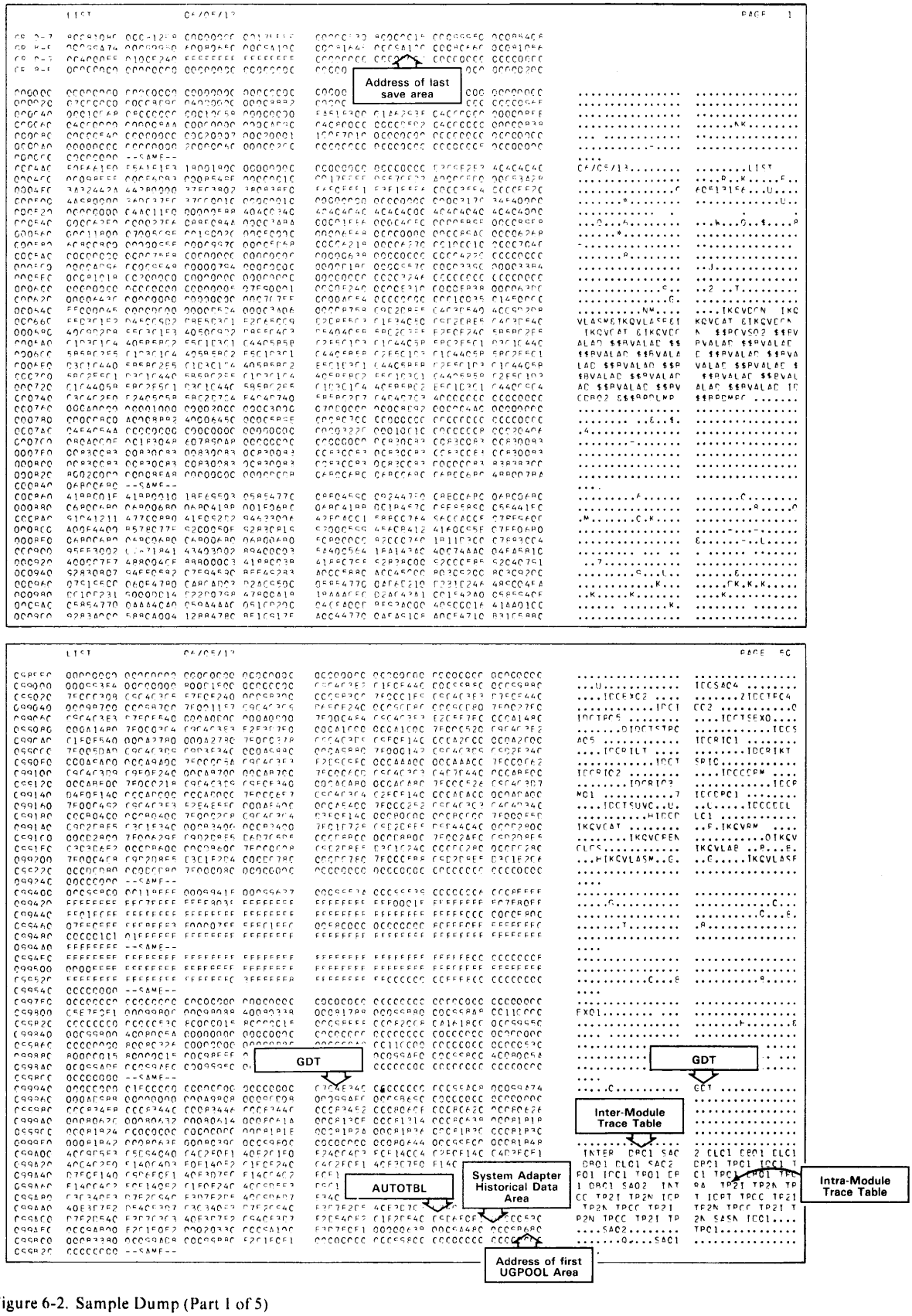
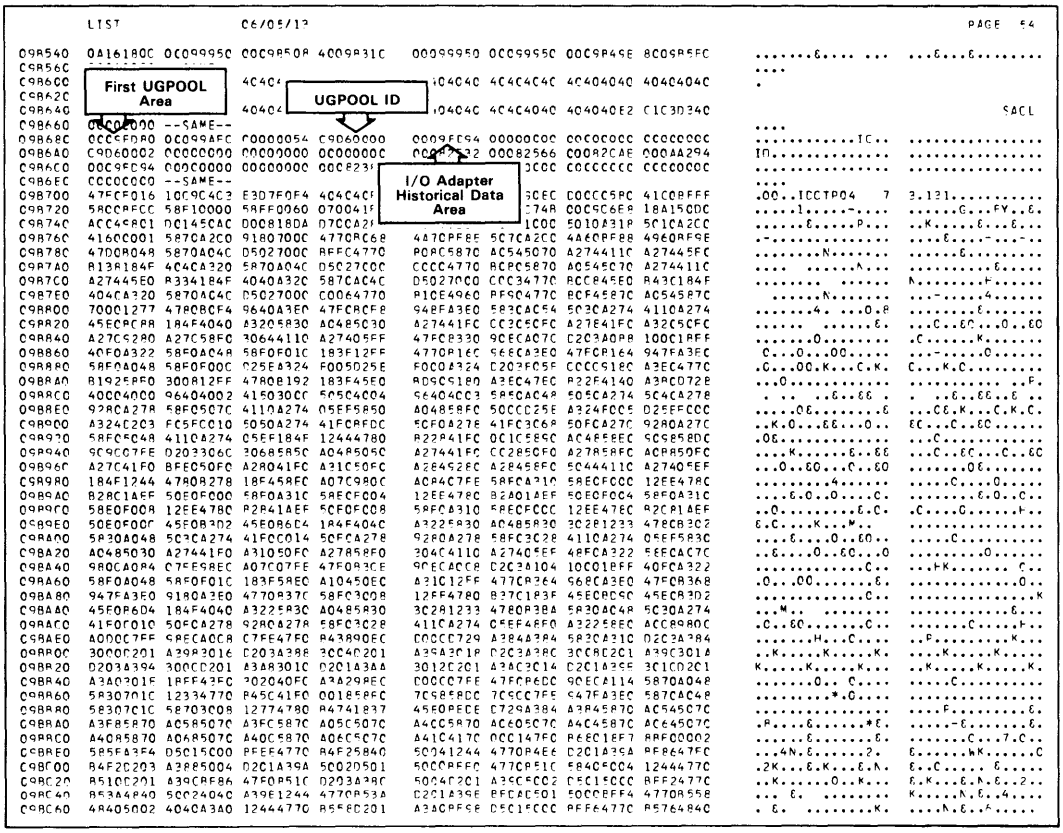


Figure 6-2. Sample Dump (Part 1 of 5)

Figure 6-2. Sample Dump (Part 2 of 5)



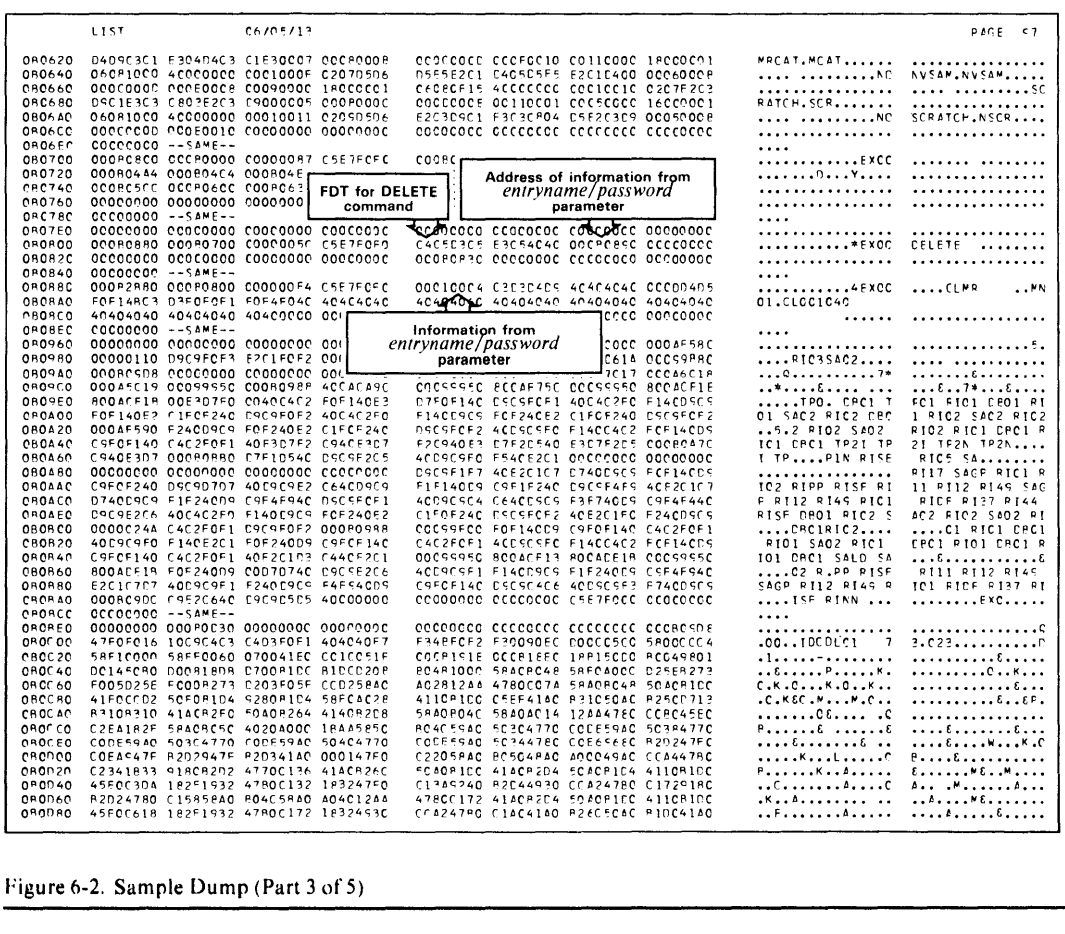


Figure 6-2. Sample Dump (Part 3 of 5)

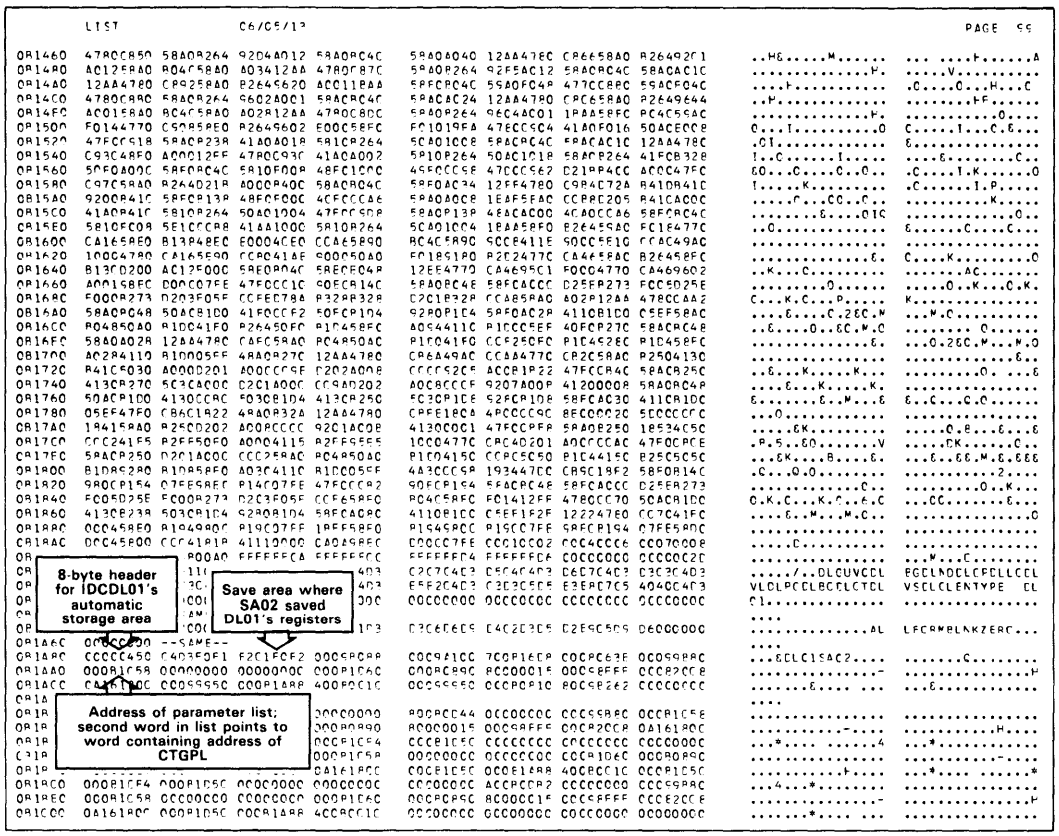


Figure 6-2. Sample Dump (Part 4 of 5)

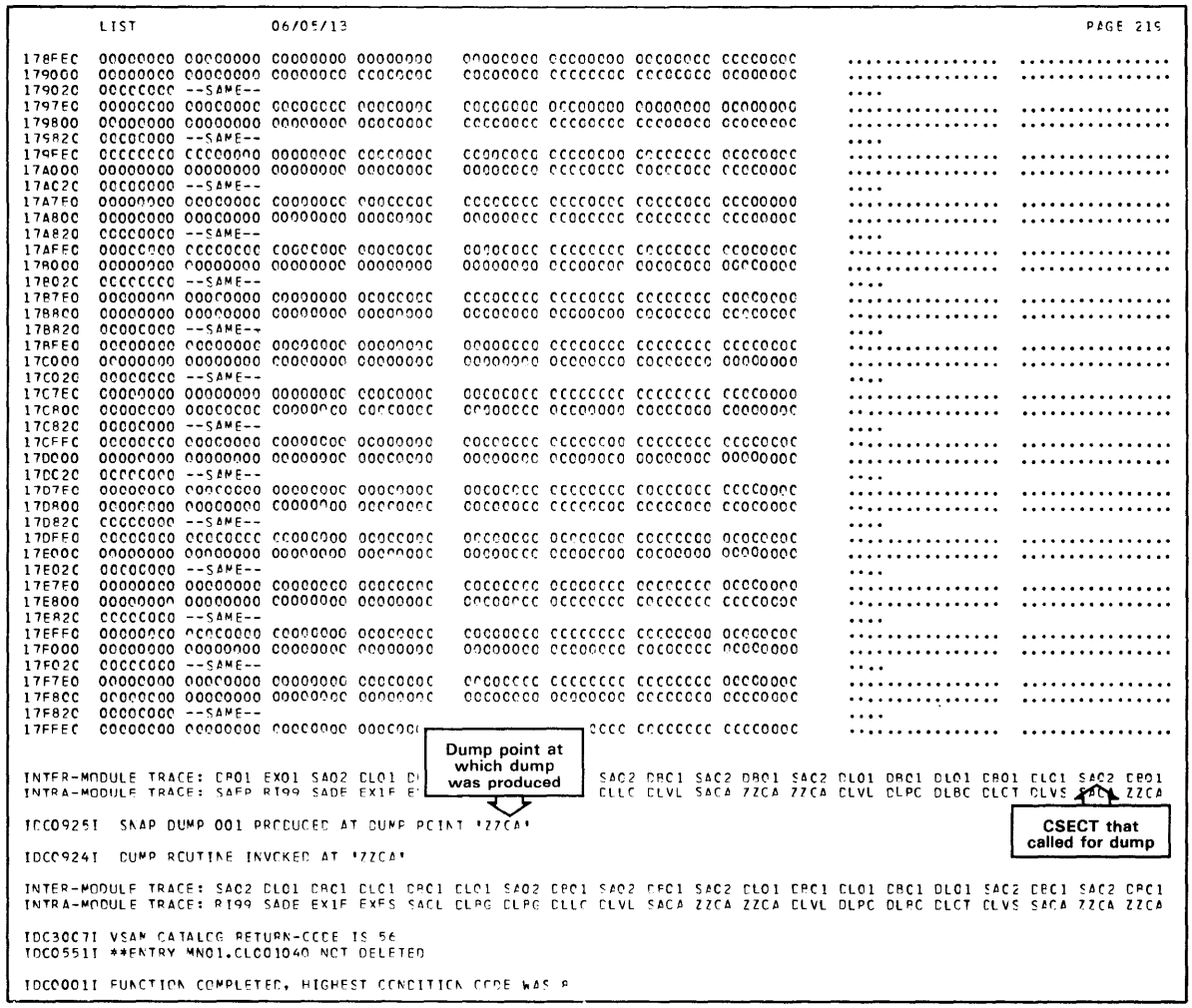


Figure 6-2. Sample Dump (Part 5 of 5)

remainder of the save area is set up following standard register saving conventions. Each module's save area is contained in the first 18 fullwords of the module's automatic storage area.

Figure 6-2, Part 2 shows a save area as it appears in a dump. The start of the save area chain is normally the psuedo save area built by the System Adapter. This is a three-word area which immediately follows the System Adapter Historical Data Area. The first word contains the identifier "SA01". The third word contains a pointer to the next save area. The forward chain is formed from the third word of each save area.

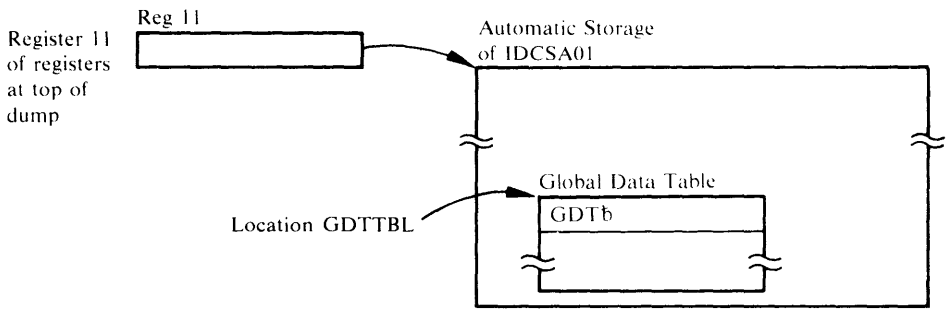
How to Find the Trace Tables

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 6-2, Part 1, shows how the trace tables appear in a dump. Note that the last (twentieth) trace point in the Intra-Module Trace Table is SASN.

Method 1.



Method 2.

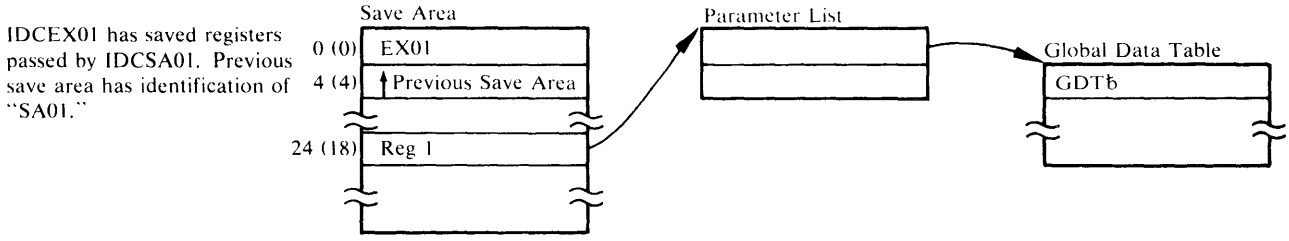


Figure 6-3. How to Find the GDT

IO01 is not part of the trace table. Also note that if, in the Inter-Module Trace Table, the sequence SA02 SA02 occurs, the second SA02 is really the ID for 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 ID of the FSR, for example, PR01 for the PRINT FSR. The previous save area in the save area chain 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 6-2, Part 3, shows how an FDT looks in a dump. Part 2 of Figure 6-2 shows the register belonging to IDCEX01 and saved by IDCDL01. Register 1 points to the parameter list. Part 4 of Figure 6-2 shows the parameter list and Part 3 shows the FDT.

How to Find Automatic Storage Areas

The automatic storage area for a phase or CSECT is that storage area obtained whenever the phase or CSECT is entered; dynamic storage areas, on the other hand, are those storage areas obtained by the phase or CSECT 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 phases and CSECTs 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 phase or CSECT ID. However, for commonly called CSECTs, namely, IDCIO01, IDCSA02, IDCSA03, and IDCTP01, no header precedes the storage area, unless the CSECT has been called recursively. On recursive calls (that is, the CSECT 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 phase or CSECT depends upon the phase or CSECT.

The address of the automatic storage area for CSECT IDCSA03 is kept in the GDT.

The addresses of the automatic storage areas for CSECTs IDCIO01, IDCSA02, and IDCTP01 are kept by the System Adapter in the AUTOTBL. Figure 6-4 shows the format of the AUTOTBL and how to find it. However, if one of these CSECTs has 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 CSECT's data register or save area register as explained in the next paragraphs.

Figure 6-2, Part 1, shows how the System Adapter Historical Area and AUTOTBL appear in a dump.

To find the automatic storage area for any phase or CSECT, you can examine the microfiche listings to find which register has been used by the compiler as the data register. This register points to the automatic storage area.

For all processor phases and CSECTs, 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 phase or CSECT. Alternatively, you can follow the save area chain as explained in the section "How to Find Save Areas"

Figure 6-5 shows the automatic storage area for IDCEX01. IDCEX01 has called IDCDL01; therefore, IDCDL01 has saved the registers belonging to IDCEX01 in the save area.

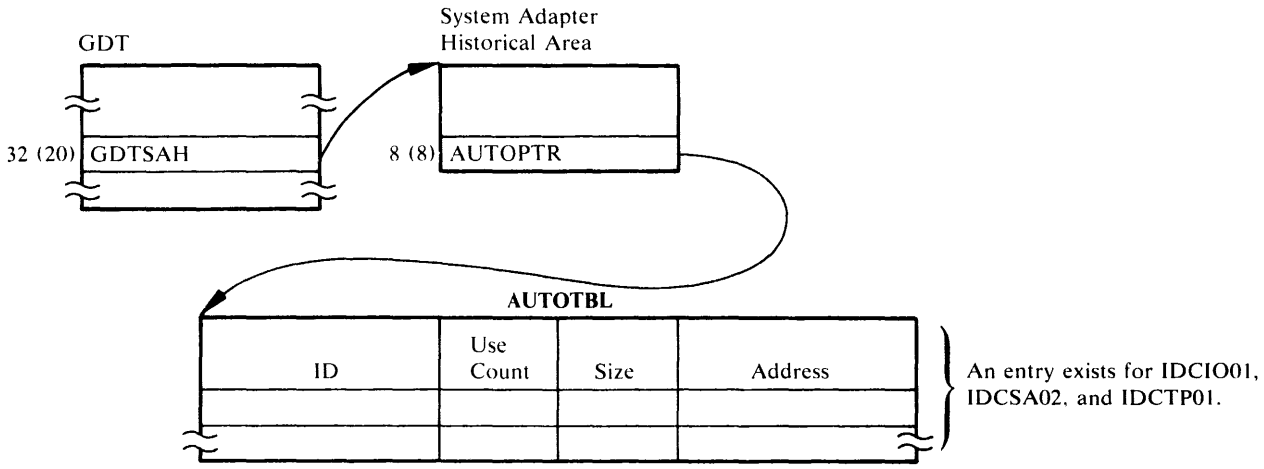
Figure 6-2, Part 4, shows an automatic storage area as seen in a dump.

How to Find Dynamic Storage Areas

A phase or CSECT 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 phase or CSECT has saved the address of that 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 6-6 shows how to find the chain of UGPOOL areas from the System Adapter's historical area.



Field	*Number of Bytes*	*Contents*
ID	4	CSECT ID
Use Count	2	Number of automatic storage areas obtained for the CSECT: 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 CSECT
Size	2	Number of bytes in automatic storage area
Address	4	Address of automatic storage area

Figure 6-4. Format of AUTOTBL

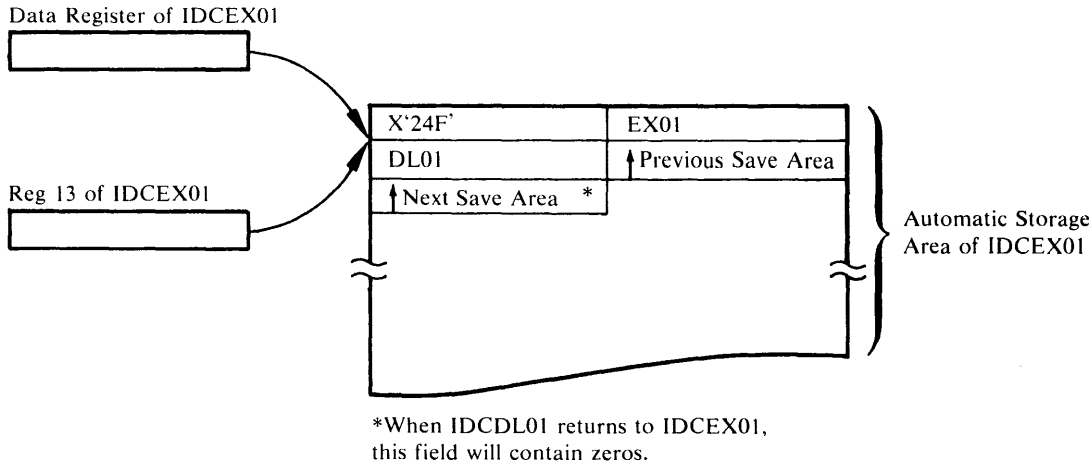


Figure 6-5. Example of an Automatic Storage Area

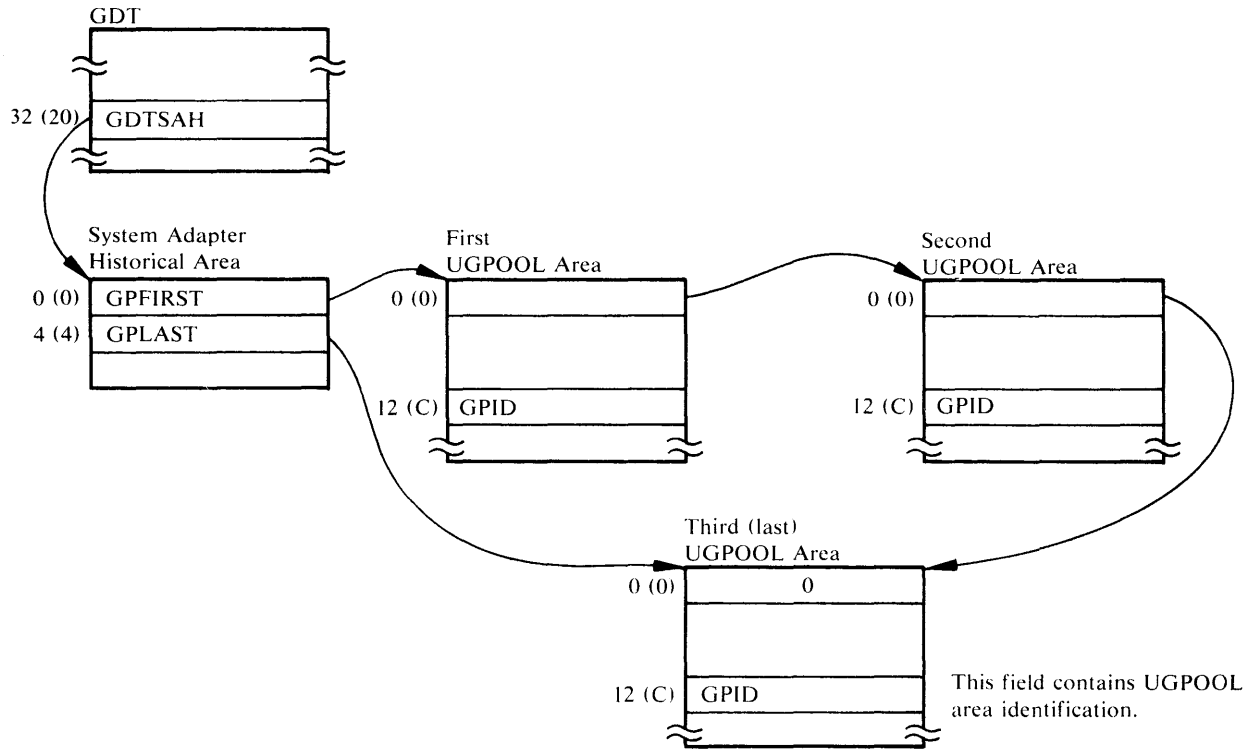


Figure 6-6. UGPOOL Area Chain

UGPOOL ID List

The following list contains the UGPOOL IDs used by different phases and CSECTs when they obtain storage. The list of UGPOOL areas also contains the name of the internal procedure that issues the UGPOOL macro, and the contents stored in the UGPOOL area.

Figure 6-2 shows the UGPOOL chain as it appears in a dump. Part 1 of Figure 6-2 shows the start of the chain in the GDT. Part 3 of Figure 6-2 shows a portion of the chain.

Contents of UGPOOL Areas

Phase or CSECT	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCAL01	AL00	ALTERPRC	One of the following: PASSWALL field or volume list.
		IDCAL01	CTGPL, CTGFV, and CTGFLs.
		LOCATPRC	Catalog work area for locate requests.
		INDEXPRC	CTGPL, CTGFV, and CTGFL to alter index KEY field.
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 CPL FVTs and FPLs, followed by alternate index record output buffer; area starts on page boundary.
	BIPG	INITPROC	Record sort area followed by table which controls the sort.
IDCDE01	DE00	IDCDE01	CTGPL and CTGFVs.
IDCDE02	DE00	ALLCPROC	One of the following: volume list, file sequence list, device type list, DSATTR, or CTGFLs.

Contents of UGPOOL Area

Phase or CSECT	UGPOOL ID	Procedure	Contents of UGPOOL Area
		KEYPROC	One of the following: MADSBCAT CTGFL and IDAAMDSB field, or key range list.
		MODELPROC	One of the following: CTGPL and CTGFLs used to locate a model object, or catalog locate work area.
		NAMEPROC	Creation and expiration date and EXCEPTION EXIT CTGFLs.
		PROTPROC	PASSWALL CTGFL, OWNERID CTGFL, PASSWALL field, RGATTR FPL, RGATTR, and User Authorization Record.
IDCDL01	DL01	MORESP	Larger VSAM catalog management services work area if necessary.
IDCIO01	IO00	IDCIOIT	I/O Adapter historical area.
		IDCIOC0	Work area where the copy routine builds spanned records that were exported in CI mode. The UGPOOL ID is the same as the ID for the associated IOCSTR.
	IOnn	PUTREP	Work area where VSAM moves records during GET. The UGPOOL ID is the same as the ID for the associated IOCSTR.
IDCIO02	IOnn	BUILDACB	ACB, RPL and EXLST for a VSAM data set. The UGPOOL ID is the same ID as the associated IOCSTR.
		BUILDDBK	IOAREA for nonVSAM files. 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 a nonVSAM spanned record. The UGPOOL ID is the same as the ID for the associated IOCSTR.
		CLOSERTN	If the UCLOSE is issued with the IOCRCVCL flag set, a new VSAM buffer is acquired. The UGPOOL ID is the same as the ID for the associated IOCSTR.
	IOnn	OPENRTN	IOCS prefix, IOCSTR, IOCSEX, and file ID. 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 this data set have the same UGPOOL IDs.
IDCIO03		DSINFO	Area in which data set name, volume serial numbers, device type, and/or format-4 timestamp 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 of cluster or alternate index.

Contents of UGPOOL Area

Phase or CSECT	UGPOOL ID	Procedure	Contents of UGPOOL Area
	LC05	INITPROC	String of control interval numbers and types of associated entries of a cluster or alternate index.
	LC06	INITPROC	Text processor argument list.
	LC07	INITPROC	Abbreviations used in catalog listing, loaded from static text phase.
	LC11	INITPROC	String of control interval numbers and types of associated entries of a data, index, or path.
IDCLC02	LC08	LOCPROC	Larger catalog work area. UGPOOL LC02 is released.
	LC09	ANSVPROC	Larger area for string of control interval numbers and types of associated entries. UGPOOL LC05 or LC11 is released.
IDCLR01	LR01	ADDASOC	Association table extension area.
		BLDVEXT	Vertical extension table extension area.
		INITLZE	Space for ASSOCTBL, ASSOCTB2 and VEXTTBL.
		INTASOC	Association table extension area.
	LR02	CTTBLD	CI translate table (CTT).
	LR03	INITLZE	Input/output buffers.
	LR04	INSORT	Sort table.
IDCMP01	MP01	BFPLPROC	Obtain one or two FPLs.
		BPASPROC	PASSWALL CTGFL.
		CLUSPROC	Buffer to read data records from the portable data set.
		CTLGPROC	Larger catalog work area.
		DELTPROC	Larger VSAM catalog management services work area if necessary.
		DVOLPROC	Volume serial list for DEFAULTVOLUMES.
		LVLRPROC	One of the following: volume list for define, or DEVTYPES CTGFL.
		RANGPROC	Range list.
		FVTPROC	FVT and pointers to FPLs.
IDCRC01	RC50	OPEN	Storage for OPNAGL.
	RC51	SUPSP	Name table storage.
	RC52	DIRECT	Buffer for directory record.
	RC54	SCANCRA	CRA translate table.
IDCRC02	RC02	IDCRC02	Control record output buffer.
		ALSPROC	Control record output buffer.
		ASOCPROC	Control record output buffer.
		CLUSPROC	Control record output buffer.
		CTLGPROC	Catalog work area.
		GDGPROC	Control record output buffer.
		LOCPROC	CPL, FPL, and work area for catalog.
		NVSMPROC	Control record output buffer.
		SAVEPROC	Input record save area.
IDCRI01	EX00	GETSPACE	FDT—data substructures.
		MORSPACE	FDT—data list substructures.
		SCANCMD	FDT—secondary pointer vectors.
	RInn	INREPEAT	FDT—temporary space for secondary pointer vectors. nn is the ID of the parameter associated with the secondary pointer vector.
IDCRI02	EX00	IDCRI02	Reader/Interpreter tables and FDT.

Contents of UGPOOL Area

Phase or CSECT	UGPOOL ID	Procedure	Contents of UGPOOL Area
	RInn	IDCRI02	FDT—temporary space for secondary pointer vectors. nn is the ID of the parameter associated with the secondary pointer vector.
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.
		DELTPROC	Larger catalog work area.
		DVOLPROC	Volume serial list for DEFAULTVOLUMES.
		FVTPROC	FVT and pointers to FPLs.
		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	RS01	IDCRS01	Automatic storage modules IDCRS01 - IDCRS07.
	RS01	INIT	Work area used for Umacro parameter lists, record access blocks, IJJHCPL parameter list and control interval translate table.
	RS03	INIT	Area obtained by UIOINFO for catalog data set information.
	RSPG	INIT	CRA user buffer.
IDCRS03	RS03	VOLCHK	UIOINFO return area and DSCB read in area.
	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	RS04	NINIT	Work area used for FIND processing.
	RS04	NXPND	Extension to FIND work area.
IDCRS05	RS01	BLDRLST	RESVOL table.
	RS02	BLDVLST	VOLSERTB table.
IDCRS06	RS03	WFDEF	Work area used for UCATLG parameter list to define the workfile, area obtained by UIOINFO for workfile data set information.
IDCRS07	RS03	RENAMEP	UIOINFO return area and work area.
		RENMBK	UIOINFO return area and work area.
	RS05	RENMSETV	CVH work area for RENAME.
IDCTP01	TP03	LINEPRT	Header line.
IDCTP04	TP01	INITPROC	Secondary Print Control Table.
		PCTSETUP	One of the following: Print Control Table, sub-title lines, or footing lines.

Contents of UGPOOL Area

Phase or CSECT	UGPOOL ID	Procedure	Contents of UGPOOL Area
IDCTP05	TP01	IDCTP05	Entry from a static text format structure.
IDCXP01	XP01	ALTRPROC	CTGFV and CTGFLs for catalog alter request.
		CONTRBL	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 in Figure 6-2 was obtained through the Test option at the ZZCA dump point. The commands that were specified are:

```
PARM TEST( FULL( ZZCA,3,1 ) )
DELETE   MN01.CL001040/CLMR
```

Various fields within the dump are marked; these fields are discussed more fully in this chapter.

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: a system error 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 section "Method of Operation" in this book and in *VSE/VSAM VSAM Logic, Volume 1*, which is listed in the preface to this book. The section "Method of Operation" explains what argument lists are passed to Catalog Management by each FSR; *VSE/VSAM VSAM Logic, Volume 1* 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 a system 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 list of Phase or CSECT to Dump Cross Reference 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 partition dump. Most FSRs that issue a UCATLG macro to call Catalog Management have dump points before and after the macro. In addition, the System Adapter routine that issues the CATLG macro has a dump point before and after the macro.

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 partition 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 use the following list or rerun the job with the AREAS option.

Instead of using the Sequence of Catalog Calls Made by FSRs, 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.

The following list summarizes the sequence of calls each FSR makes to Catalog Management. For example, assume that the LISTCAT FSR, IDCLC01, while listing all the information for a KSDS cluster entry, listed the cluster name under the index entry incorrectly. Referring to the list, you would know that the call to the catalog that retrieved that name was the seventh call the LISTCAT FSR made to Catalog Management.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
IDCAL01	1. A call to open the catalog if the dname subparameter of the CATALOG parameter was specified. 2. 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. 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 RECORDSIZE was supplied for the path object: 1. A call to locate the data object of the related alternate index or cluster.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
	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 entr's data object. If KEYS was supplied: 1. A call to alter the related index object's key values.
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.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
IDCDE01	1. A call to open the catalog if the dname subparameter of the CATALOG parameter was specified. 2. A call to define the entire entry.
IDCDE02	1. A call to open the catalog specified if the MODEL parameter was specified with the dname. subparameter. This call occurs prior to the first locate for cluster, data or index described in 3. 2. One or more calls to locate each object that is modeled, as follows: threecalls if the MODEL keyword is specified in the cluster parameter list for a KSDS cluster; 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	1. A call to open the catalog if the dname subparameter of the CATALOG parameter was specified. For each entry: 1. A call to locate the entry type, if the type was not specified on the command. 2. A call to delete the entire entry. 3. An iterative series of calls to delete any remaining parts of a structure as necessary.
IDCLC01	1. A call to open the catalog if the dname subparameter of the CATALOG parameter was specified. 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.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
	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 data entry: 1. A call to locate the data entry. 2. A call to locate the name of the cluster or alternate index entry associated with the data entry. For each index entry: 1. A call to locate the index entry. 2. A call to locate the name of the cluster or alternate index entry associated with the index entry. 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 nonVSAM entry: 1. A call to locate the nonVSAM entry. For each space entry: 1. A call to locate the space entry. 2. One or more calls to locate each file ID in a space entry, for example, three calls if three data sets are defined in the data space. For each user catalog entry: 1. A call to locate the user catalog entry.
IDCLR01	1. A call to open the catalog if the dname subparameter of the CATALOG parameter was specified.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
IDCMP01	1. A call to define the cluster or alternate index. 2. A call to locate the cluster entry, if the previous define failed bacause of a duplicate entry in the catalog. 3. A call to locate the data entry, only for a duplicate cluster entry. 4. A call to locate the index entry, only for a duplicate KSDS cluster entry or alternate index entry and if the temporary export flag is not set in the data entry. 5. A call to delete the entry, if there is a duplicate nonempty entry. 6. An iteractive series of calls to delete any remaining parts of a structure as necessary. 7. A call to define the cluster again, if there was a duplicate entry. 8. A call to 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 a structure as necessary. 10. A call to alter the data entry, if the INHIBITTARGET keyword was specified at export time. 11. A call to alter the index entry, if the INHIBITTARGET keyword was specified at export time for a KSDS cluster or an alternate index.
IDCRC01	1. A call to locate the cluster entry. 2. A call to locate the data entry. 3. A call to locate the index entry only for a KSDS cluster or an alternate index.
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 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 exported. 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	For VSAM data sets: 1. A call to identify the INFILE data set type. 2. A call to identify the OUTFILE data set type.
IDCRS01	1. A call to locate the catalog volume serial number. 2. A call to locate the catalog data set name. 3. A call to locate the catalog volume serial number and timestamp. 4. A call to locate the catalog ACB and data attributes. 5. A call to locate the ACB of the catalog in which the workfile was defined.
IDCRS06	1. A call to define the workfile. 2. A call to delete the workfile.

Sequence of Catalog Calls Made by FSRs

FSR	Sequence of calls to catalog management
IDCXP01	1. A call to locate the cluster or alternate index entry. 2. A call to locate the data entry. 3. A call to locate the index entry, only 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.

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: a system dump within Catalog Management, a PDUMP taken at a dump point within an FSR, or a PDUMP taken at the ZZCA dump point in the System Adapter.

In a system dump within Catalog Management, register 1 of the registers saved when Catalog Management was entered contains the address of the CTGPL.

In a PDUMP 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 PDUMP 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 6-7 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 you have: a PDUMP taken at a dump point within an FSR, or a PDUMP taken at dump point ZZCA.

In a PDUMP 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 nonzero return code is always printed by the FSR in a subsequent message.

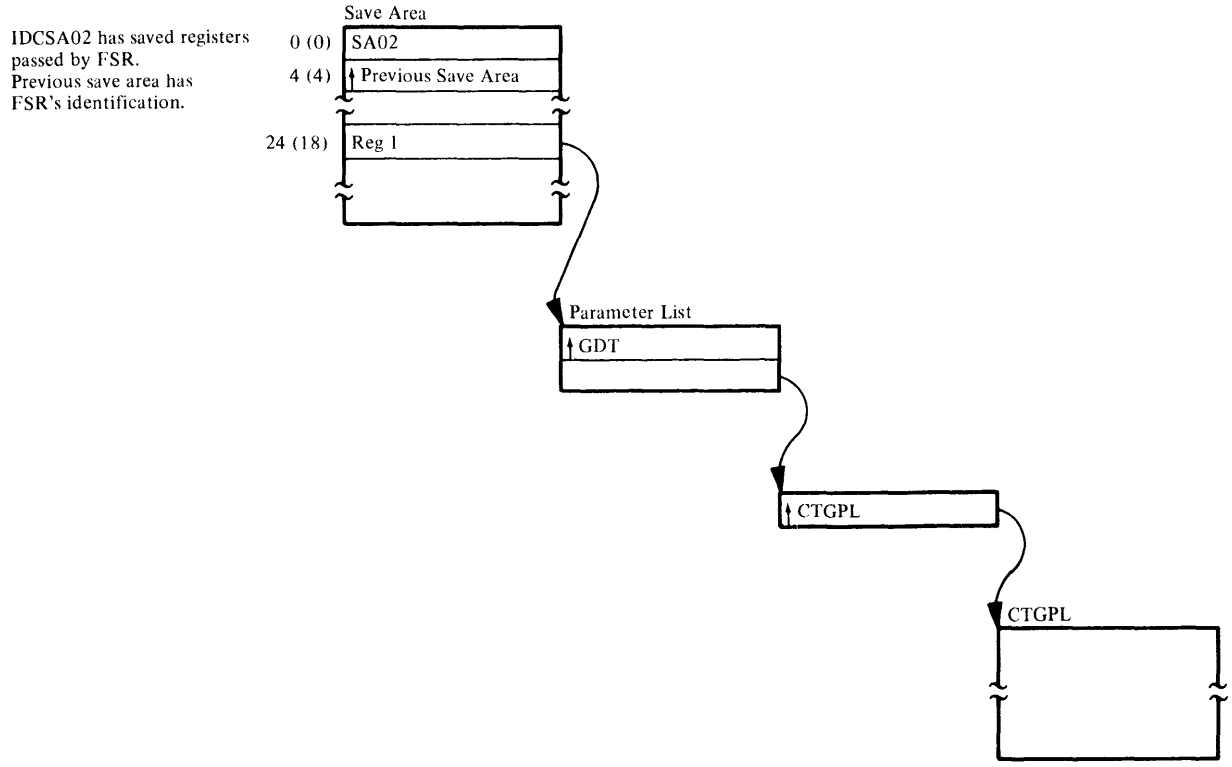


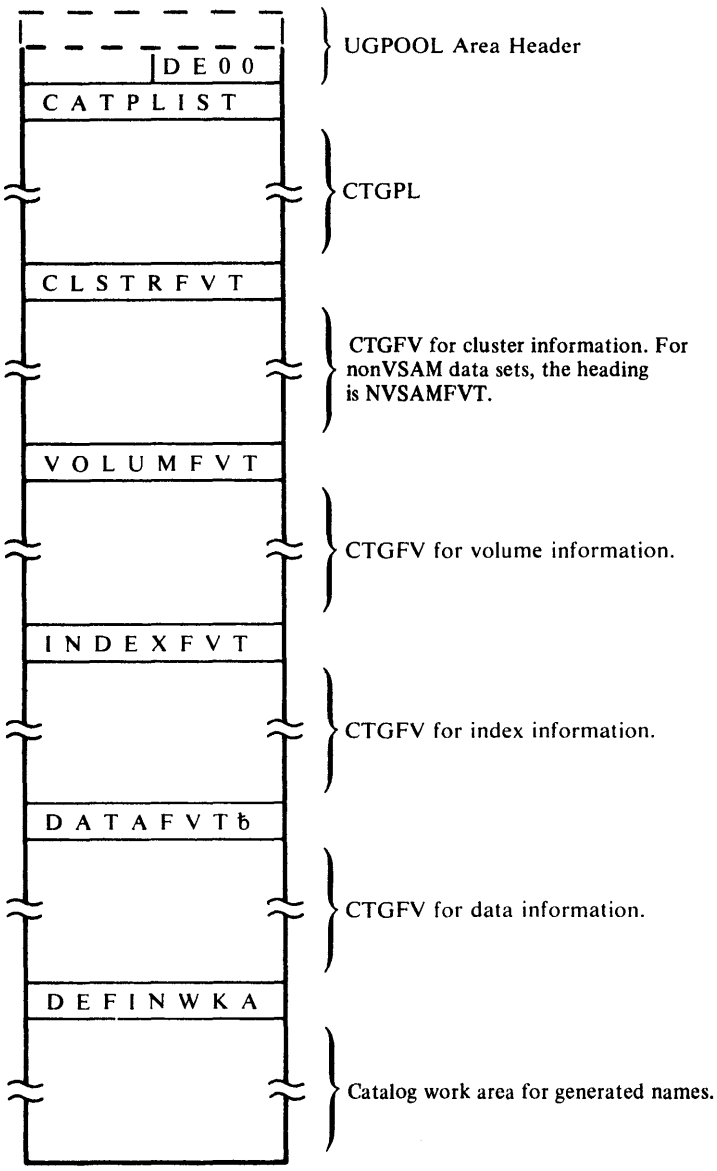
Figure 6-7. How to Find the CTGPL

In a PDUMP 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 6-8 shows the DEFINE FSR's storage area that contains the argument lists set up for a define request.

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.



If any of the above CTGFVs are not set up for a define request, the heading and CTGFV area contains zeros.

Figure 6-8. Catalog Argument Lists in Storage Area of DEFINE FSR

Example 1

A module wants to space one line then print data starting in column 10. 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:

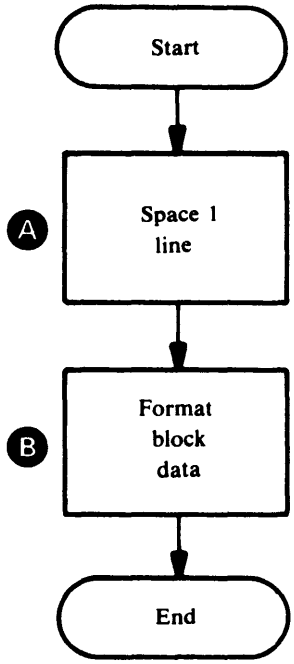
Offset	Name	Contents	Comments
0	DARGDBP	↑INFO	Address of the block of data to be printed.
4	DARGRETP	0	The line is to be printed rather than just formatted and returned to the module without printing.
8	DARGSTID	0	No static text module is used—the FMTLIST and data are in the module's storage.
12	DARGILP	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 FMTSPR 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.
	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 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. 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.

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 'RECORDbNUMBER'

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.

	Offset	Name	Contents	Comments
	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.
	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'RECORDbNUMBER'	Data for the second substructure.
	71	*any*	C'UNKNOWN'	Data for the default text substructure.

Discussion:

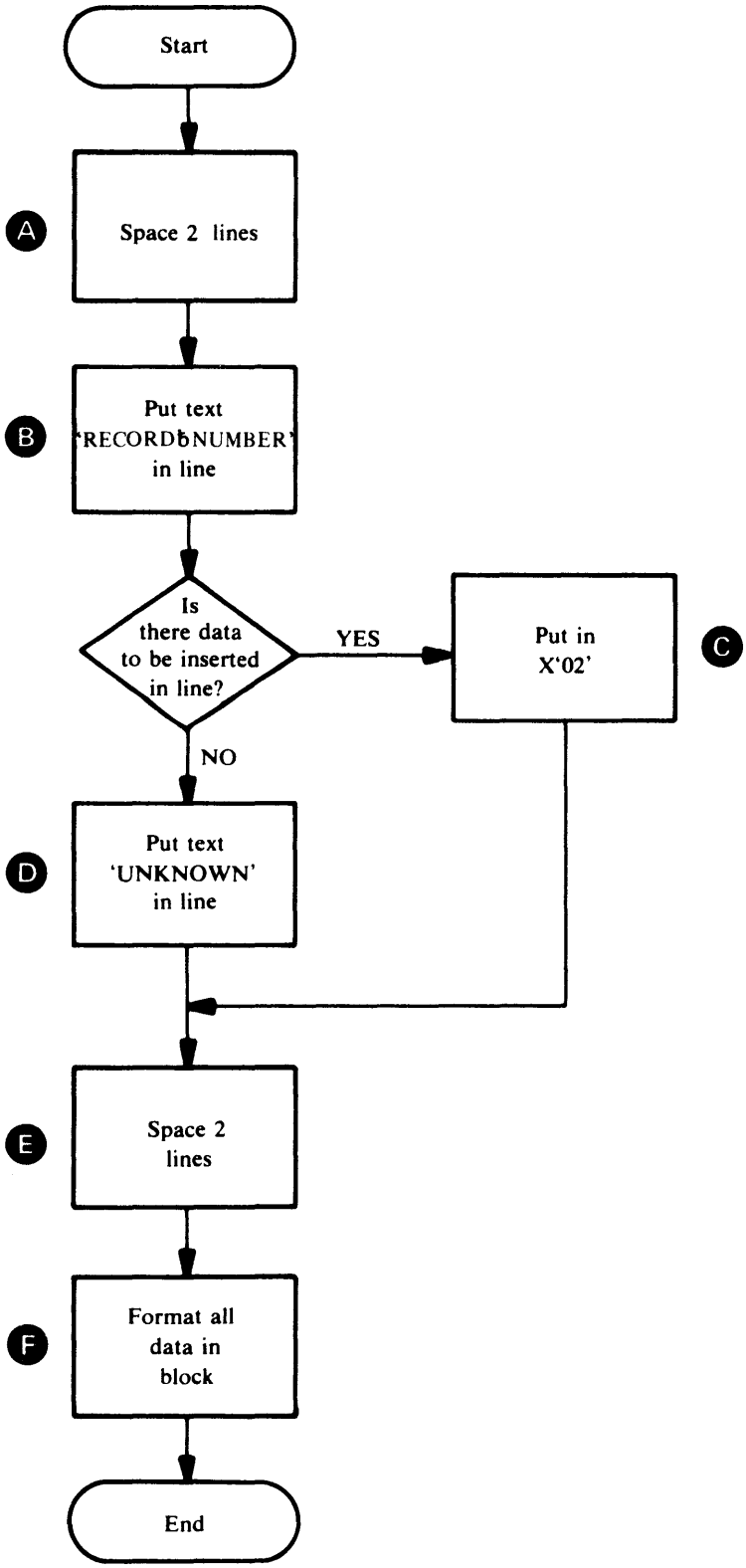
The first spacing substructure causes two lines to be spaced.

The static text 'RECORDbNUMBER' 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 C'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.


Start
A
Space 2 lines
B
Put text 'RECORDbNUMBER' in line
Is there data to be inserted in line?
YES
Put in X'02'
C
NO
D
Put text 'UNKNOWN' in line
E
Space 2 lines
F
Format all data in block
End

Example III

A module wants to space 3 lines then print repeating fields on different lines so the output would appear as:

```
(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.
19	*none*	0	Unused.
18	DARGIND	0	Printing starts in column indicated in FMTLIST.
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.
B	6	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure.

	Offset	Name	Contents	Comments
	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'*.
F	50	FMTFLGS	X'10'	Identifies these 12 bytes as a block data substructure for fields D1 and D2.
	51	*none*	0	Unused.

Offset	Name	Contents	Comments
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.
G 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.

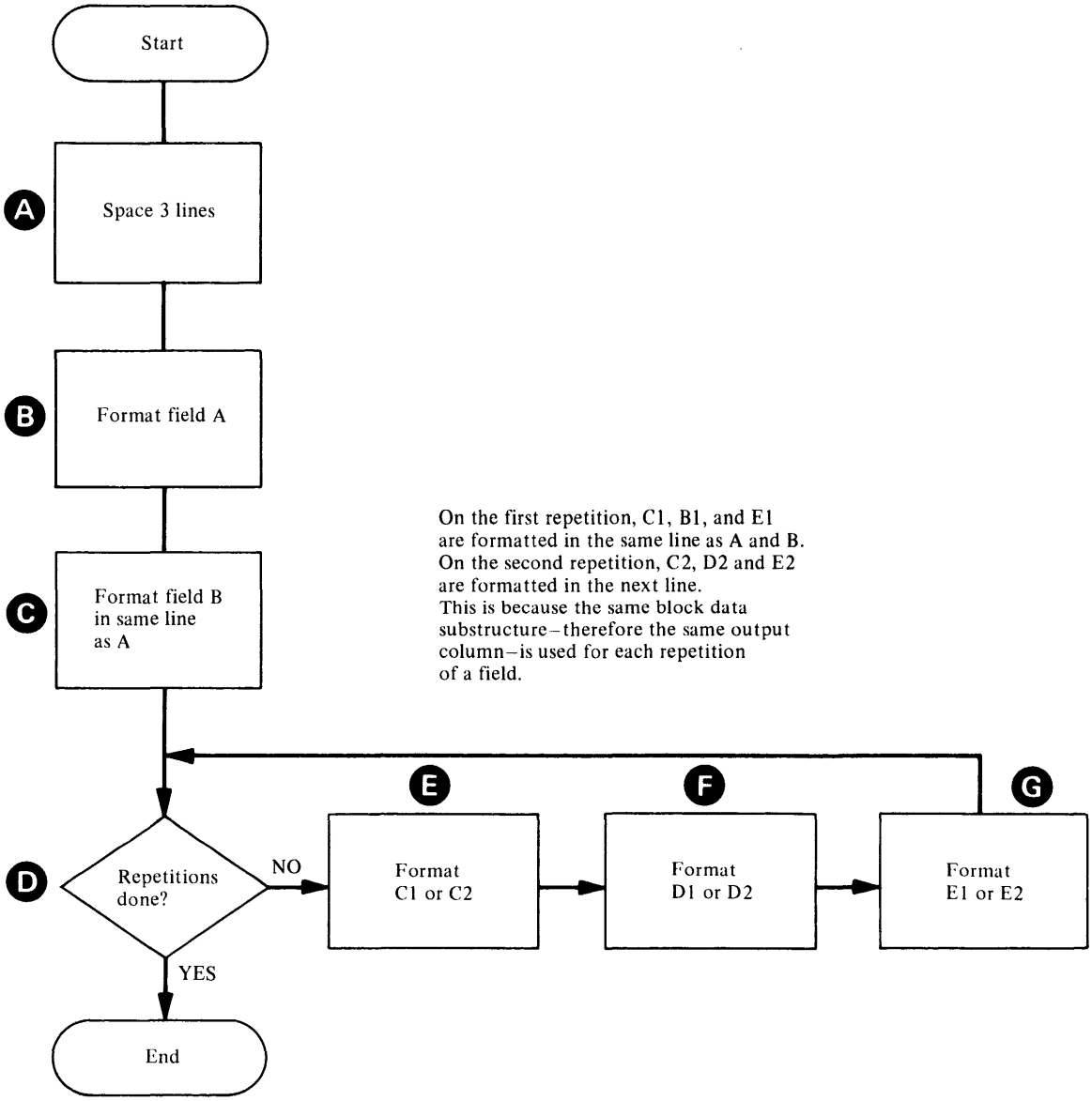
The replication substructure number is matched with an identification number in DARGREP. When a match is found, the DARGPCT immediately after DARGREP tells how many times to repeat the substructures. If the module wants to use the same FMTLIST and print another group of fields C, D, and E, only DARGPCT 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 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.



Obtaining a Dump For a Text Processor Problem

If you do not have an system 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 Phase or CSECT to Dump Points Cross Reference 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 partition 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 section "Data Areas" in this book. 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 print request (UPRINT macro).

Two other structures that you may find helpful to locate in a dump are the queue of format structures and the print buffer.

Figure 6-9 shows the queue of format structures maintained by the Text Processor. There is an entry in the queue for each format structure that has been used by the current function. Each entry in the queue contains the four-byte text structure ID specified in the DARGLIST. The first three bytes contain the last three characters of the text-structure phase name; the fourth byte contains the entry number of the format structure within the text-structure phase.

Figure 6-10 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.

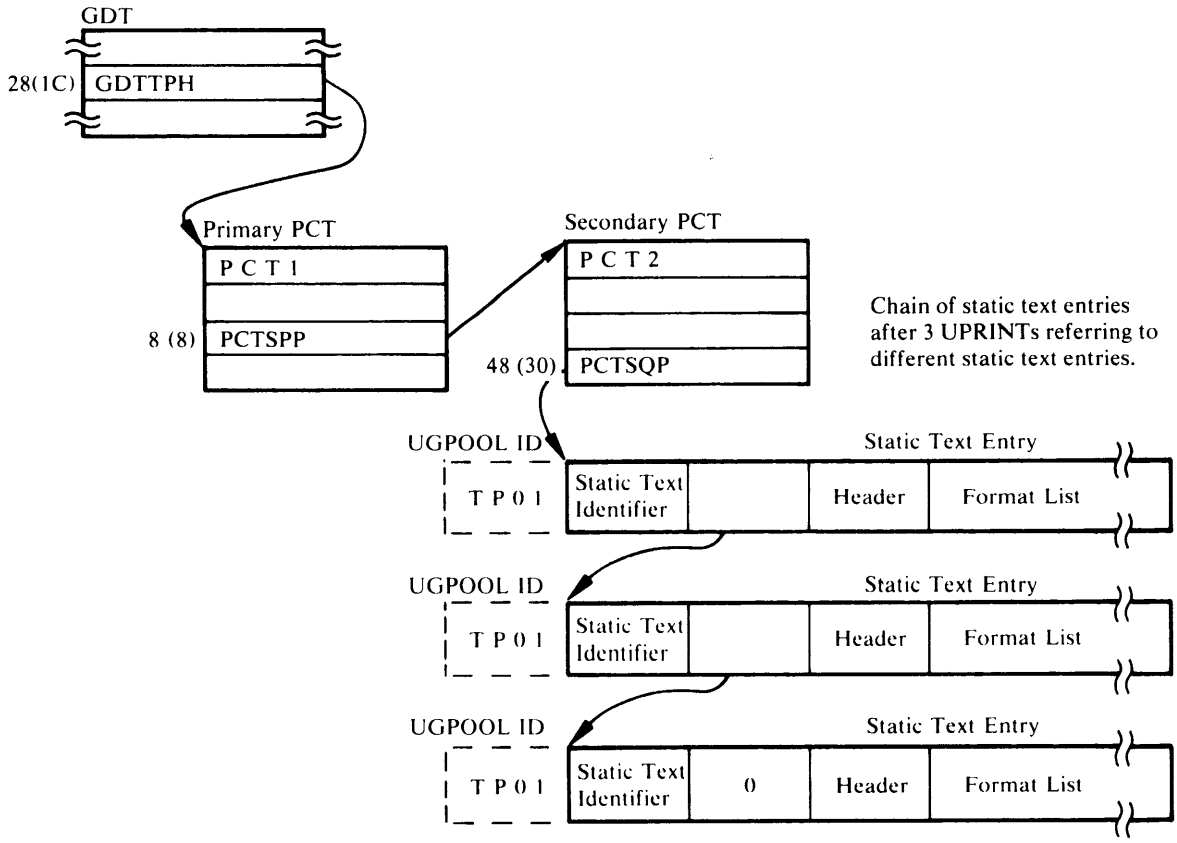


Figure 6-9. Text Processor Format Structure Queue

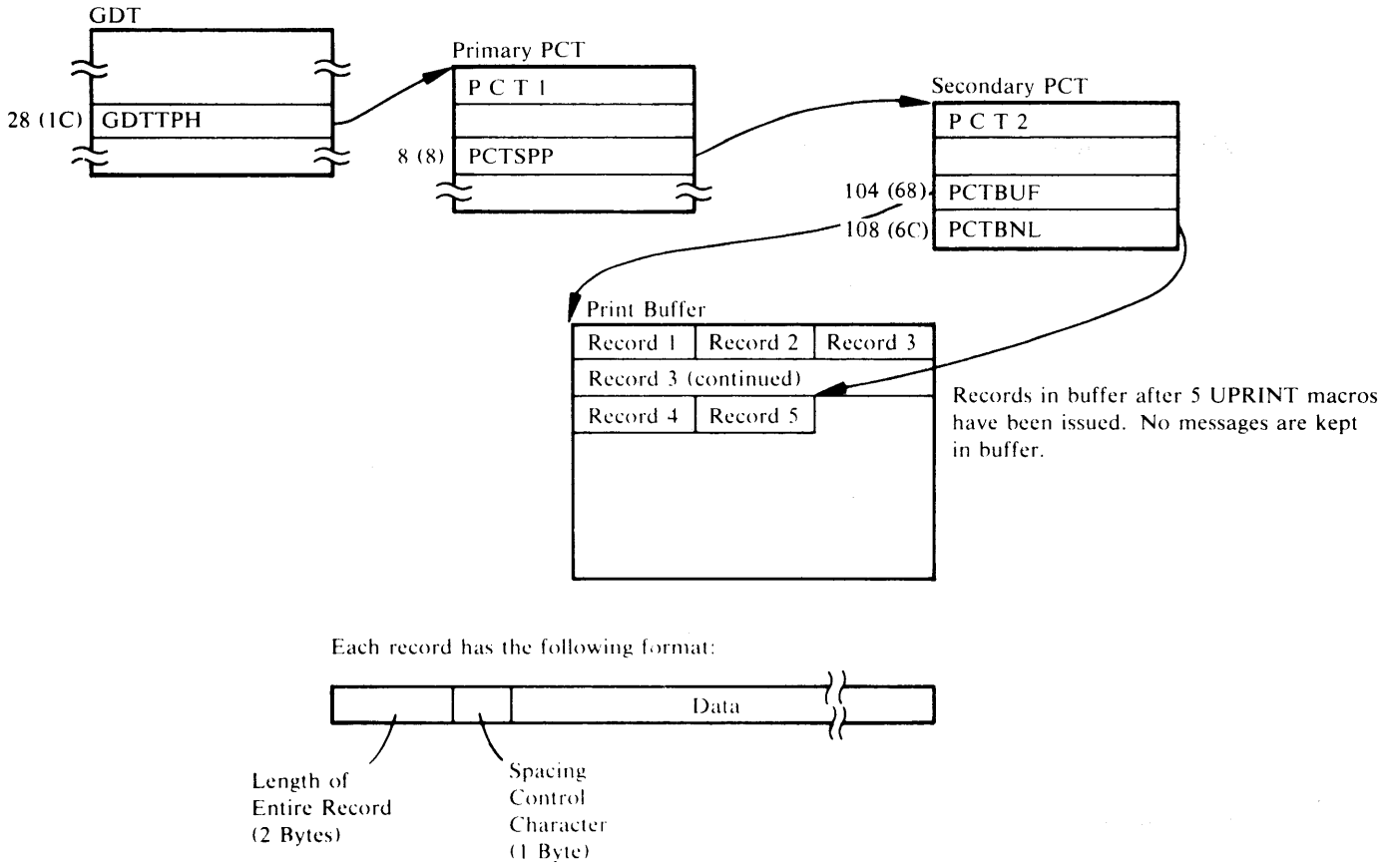


Figure 6-10. Text Processor Print Buffer

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 a system error 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 a system dump within system I/O routines or an ABORT PDUMP 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 Phase or CSECT to Dump Points Cross Reference 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 partition dump.

The I/O Adapter has dump points before and after it issues the OPEN macro (dump points IO1O and IO20) and before it issues the CLOSE macro

(dump point IO1C). You should use these dump points if there is an error opening or closing data sets. The I/O Adapter has a dump point (IOVR) after issuing a VSAM I/O request which returns a non-zero 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 section "Data Areas" in this book explains the format of the IOCSTR.

Figure 6-11 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 also contain addresses of additional IOCSTRs.

Open Argument Lists

Figure 6-12 shows how the I/O control blocks are connected before an OPEN macro is issued. The IOCSTR addresses can be found from the IOCSTR chain as shown in Figure 6-9. 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 DTFs 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 system open routine, register 0 points to a word that contains either the address of the ACB or the address of the DTF.

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 6-13 shows the IOCSTRs and records passed to the I/O Adapter via a UPUT macro.

Figure 6-14 shows the IOCSTRs and data returned by the I/O Adapter after a UGET macro is processed.

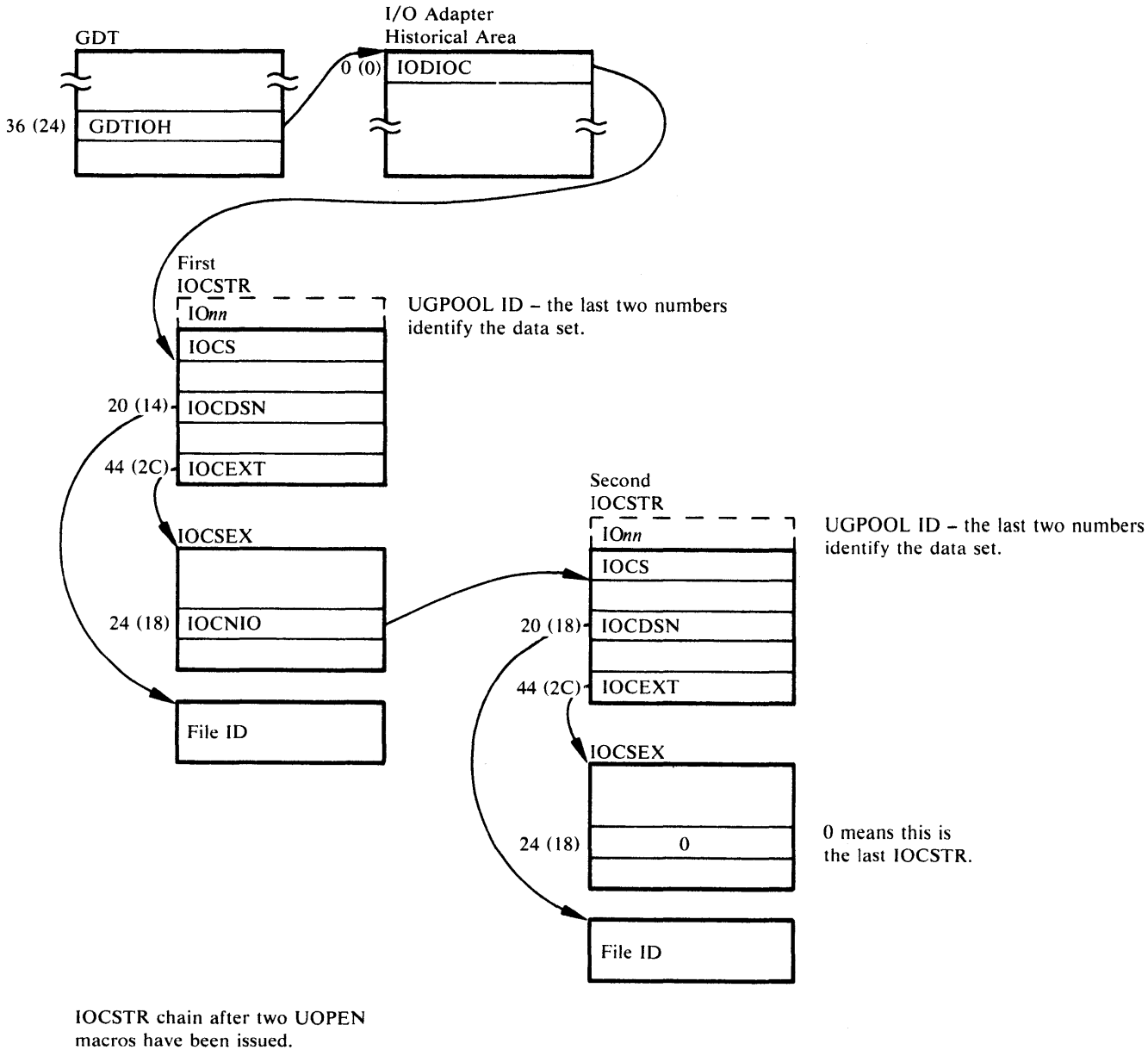
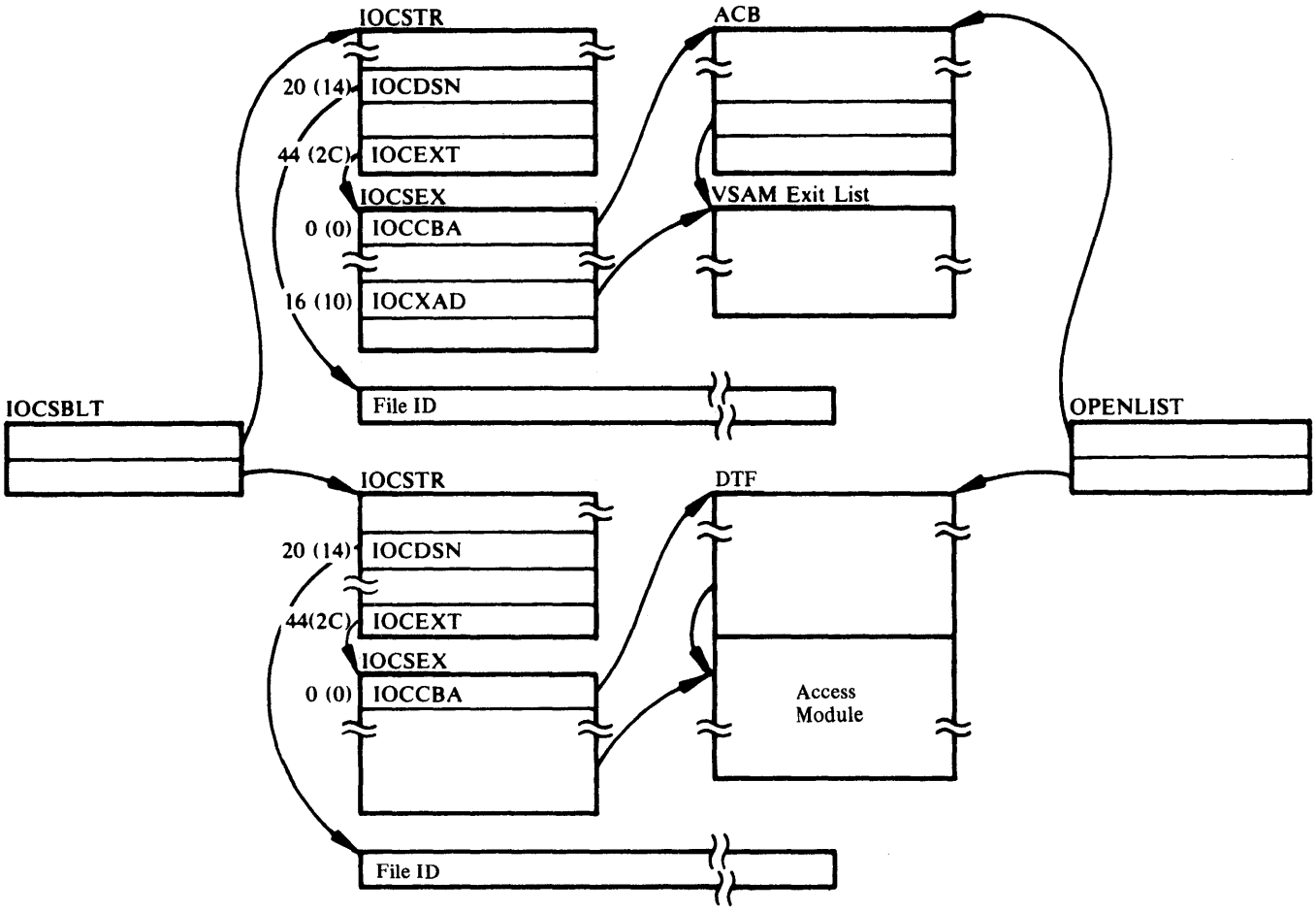


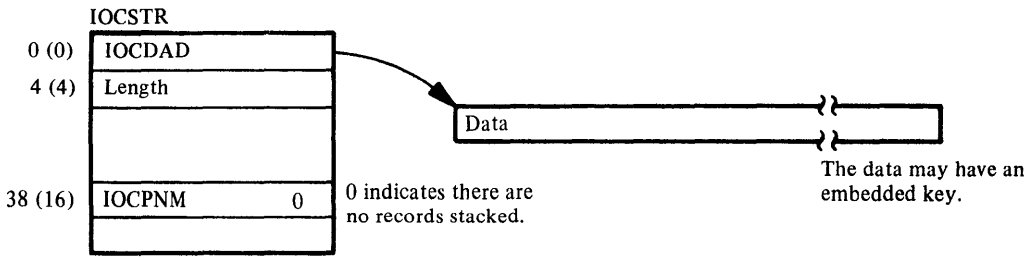
Figure 6-11. IOCSTR Chain



Two data sets are to be opened, one VSAM and one non-VSAM data set.

Figure 6-12. I/O Control Blocks Before OPEN

Example 1. VSAM or NonVSAM Data Set – Single Record Passed via UPUT



Example 2. VSAM or NonVSAM Data Set – Multiple Records Passed via UPUT

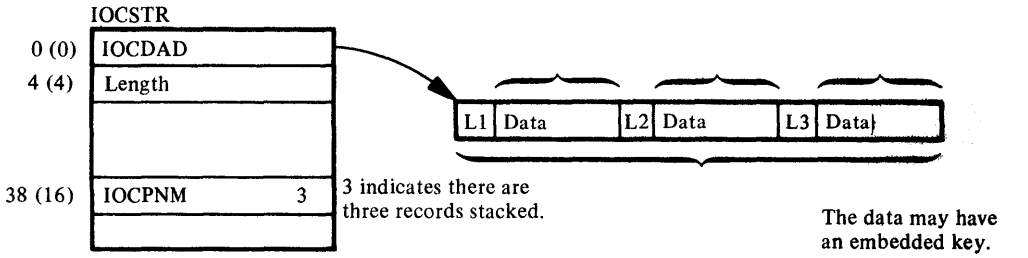


Figure 6-13. Input to UPUT Macro

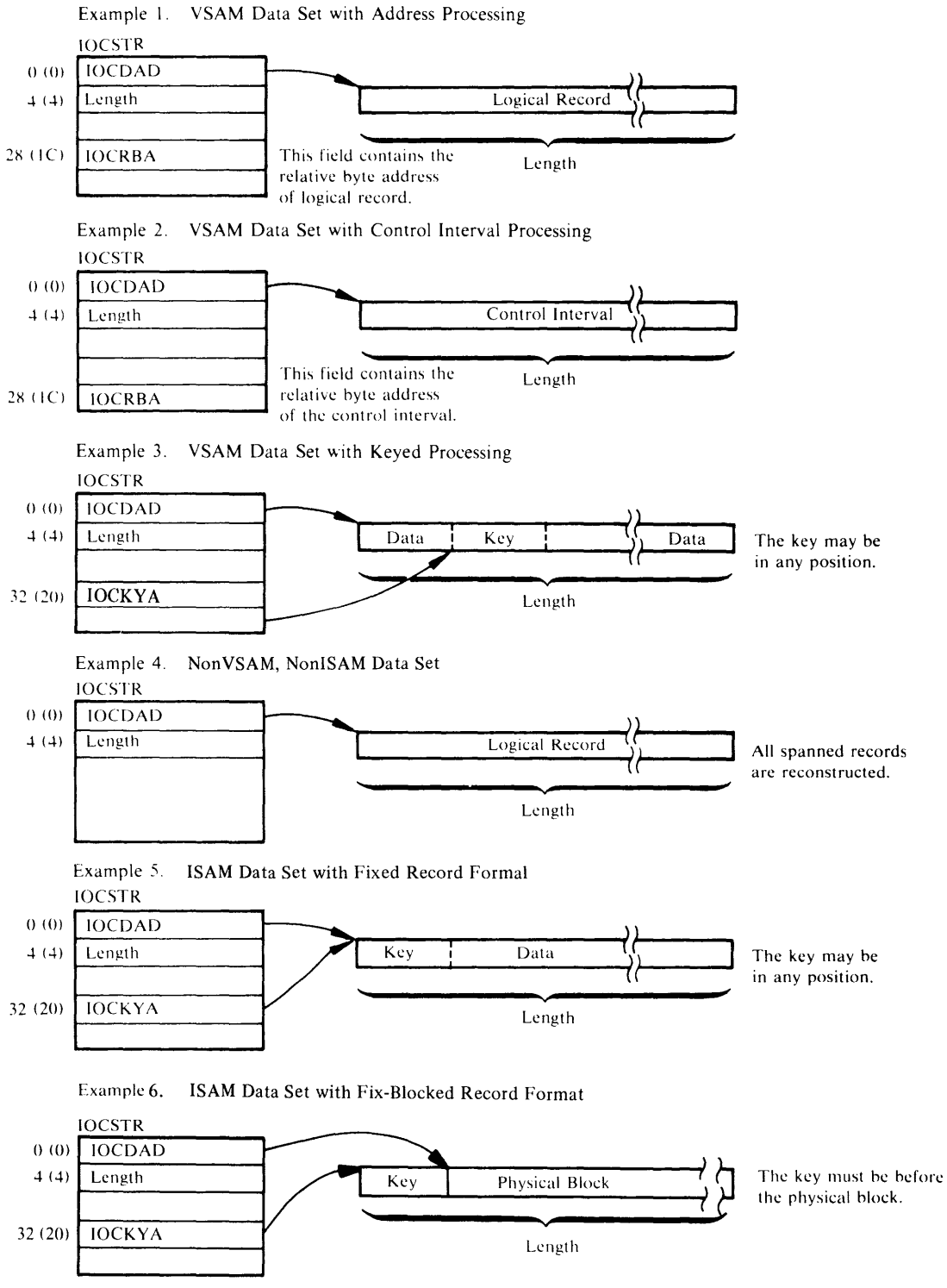


Figure 6-14. Output from UGET Macro

Messages

The following list shows all the messages printed by the processor. For each message, the following information is listed: the text-structure identifier used internally by the processor to identify the message; 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. After the text is the entry within the text structure.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC00011	UVO-1	IDCAL01	IDCAL01	Function was completed without a severe error.
		IDCBI01	TERMPROC	Function was completed without an error or without a severe error in processing the base cluster.
		IDCCL01	IDCCL01	Function will CANCEL Access Method Services as requested.
		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.
		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	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.
IDC00021	UV0-2	IDCEX03	IDCEX03	Access Method Services completed processing.
IDC00051	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
IDC01771	DE0-28	IDCMP01	CLUSPROC	IMPORT has successfully defined a unique data set on a fixed block device and is printing the actual blocks allocated on each volume.
		IDCRM01	CLUSPROC	IMPORTRA has successfully defined a unique data set on a fixed block device and is printing the actual blocks allocated on each volume.
		IDCDE01	IDCDE01	Possible rounding of fixed block extents. Actual extents are printed.
IDC02041	RI0-5	IDCRI03	IDCRI03	The preceding command was scanned for syntax-checking purposes only.
IDC02061	RI0-7	IDCRI01	SCANSEP	An extra comma was found between parameters.
IDC02221	RI0-23	IDCRI01	NXTFIELD	A semicolon was found within a quoted constant.
IDC02331	RI0-34	IDCRI01	SCANCMD	Too many closing parentheses were found at the end of a command or subparameter list.
IDC02341	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.
IDC02961	DEO-31	IDCDE01	IDCDE01	A default model had been successfully defined.
IDC05081	DE0-9	IDCDE01	IDCDE01	Define of the data set failed due to a space allocation error.
		IDCMP01	CTLGPROC	Define of the data set being imported failed due to a space allocation error.
		IDCRM01	CTLGPROC	Define of the data set being imported failed due to a space allocation error.
IDC05091	DE0-10	IDCDE01	IDCDE01	Define of the data set failed due to a space allocation error.
		IDCMP01	CTLGPROC	Define of the data set being imported failed due to a space allocation error.
		IDCRM01	CTLGPROC	Define of the data set being imported failed due to a space allocation error.
IDC05101	DE0-11	IDCDE01	IDCDE01	Define of the VSAM catalog failed due to a space allocation error.
IDC05111	DE0-12	IDCDE01	IDCDE01	Define of the data space failed due to a space allocation error.
IDC05121	DE0-13	IDCDE01	IDCDE01	Data and index name generation.
IDC05201	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.
IDC05261	AL0-1	IDCAL01	IDCAL01	Alter of the data object is completed.
IDC05501	DL0-1	IDCDL01	CATCALL MORESP	The catalog returned the name and type of a successfully deleted entry in the catalog work area.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
		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 MORESP	The object being exported was deleted successfully from the catalog.
IDC0551I	DL0-8	IDCDL01	IDCDL01	A catalog object was not deleted because of a catalog locate error, a command parameter error, or a catalog delete error.
		IDCXP01	DELTPROC MORESP	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 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 a group of associated objects on the portable data set contained the timestamp written at the time of the export.
IDC0622I	MP0-22	IDCRM01	UCATPROC	An existing duplicate catalog entry was deleted to allow a user catalog entry to be imported.
IDC0626I	MP0-26	IDCRM01	CLUSPROC UCATPROC NVSMPROC	The object named has been successfully imported.
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.
IDC0672I	RC0-17	IDCRC01	CKCATNM	Informational message stating the catalog name for which CRA's are being processed.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
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, or is a SAM ESDS which cannot be exported.
		IDCMP01	DVOLCHK	DEFAULTVOLUMES parameter was invalidly specified or ignored due to VOLUMES parameter override.
		IDCRM01	DVOLCHK	DEFAULTVOLUMES parameter was invalidly specified or ignored due to VOLUMES parameter override.
IDC0676I	RC0-5	IDCRC01	TERM	Informational message stating that the portability data set was created successfully.
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.
IDC0924I	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.
IDC1172I	DE0-25	IDCDE01	INTGCHK	USECLASS was specified at the data or index level, but was ignored because it was not accompanied by space parameters (CYLINDERS, for example) at the same level.
IDC1293I	UVO-13	IDCDE01	INTGCHK	DEFAULTVOLUMES parameter was explicitly specified but was overridden by explicit specification of the VOLUMES parameter at another component level.
		IDCDE02	ALLCPROC	ORIGIN was specified along with the DEDICATE parameter. ORIGIN was ignored.
		IDCMP01	DVOLCHK	DEFAULTVOLUMES parameter was specified but was overridden by specifications of the VOLUMES parameter at another component level.
		IDCRM01	DVOLCHK	DEFAULTVOLUMES parameter was specified but was overridden by specifications of the VOLUMES

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				parameter at another component level.
IDC1294I	RC0-10	IDCRC01	CKNAMES	SAM ESDS encountered in CRA and bypassed for export because the SAM ESDS feature is not installed or because NOCIFORMAT SAM ESDS cannot be exported.
IDC1329I	IO0-49	IDCIO01	IDCIOCO	Two or more segments of a spanned record are not at the same update level.
IDC1502I	DE0-5	IDCDE02	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 cluster associations.
			LOCPROC	The UGPOOL request for a larger catalog work area failed. A catalog entry required more space.
			CDIPROC	The UGPOOL request for a larger catalog work area failed. More space was required for block calculations.
			VPROC	The UGPOOL request for a larger catalog work area failed. More space was required to convert SPACEMAP to blocks.
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.
IDC1566I	LC1-8	IDCLC01	ENTPROC	Either (1) the correct password was not supplied for a cluster entry and so the data and index 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 entry, and, even though the user requested only entry names, the names of the data and index association were not returned by the catalog, or (2) the

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				correct password was not supplied for a data or index entry associated with a cluster 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 entry associated with a cluster entry was attempted, using the control interval number of the associated entry contained in the cluster entry. However, the entry could not be found in the catalog.
		IDCLC02	CDIPROC	Retrievel of a data or index entry associated with a cluster entry was attempted, using the control interval number of the associated entry contained in the cluster 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.
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.
IDC1614I	MP0-7	IDCMP01	CLUSPROC	The object name specified by the user does not match the object names found in the portable data set.
IDC1627I	MP0-25	IDCMP01	DUPNPROC	The OBJECTS parameter was specified for a component being imported into an empty data set.
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 BI0-8	IDCBI01	BLDPROC	Multiple occurrances 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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC16621	RC0-7	IDCRC01	EXPORTDR	Informational message stating that the data set was not exported and was out-of-synch.
IDC16631	RC0-8	IDCRC02	CLUSPROC	Catalog field could not be located for a path to a VSAM cluster.
			ALISPROC	Catalog field could not be located for an OS/VS2 alias for a nonVSAM file or OS/VS2 generation data group.
			ASOCPROC	Catalog field could not be located for an OS/VS2 alias for a nonVSAM file.
IDC16641	RC0-9	IDCRC02	NVSMPROC	An OS/VS catalog has been connected to a DOS system and contains nonsupported objects.
IDC16671	RC0-12	IDCRC01	OBJVOLCK	Volumes are out of synch because data set is not on both volumes.
IDC16781	RC0-2	IDCRC01	EXPORTDR	An error occurred while processing an association for an object being exported.
IDC16791	RC0-4	IDCRC01	EXPORTDR	The timestamps or CI of a multivolume data set were not equal.
IDC18701	LR1-1	IDCLR01	GETPRT	An I/O error occurred while reading the CRA.
		IDCLR02	IDCLR02	An I/O error occurred while reading the CRA.
IDC18711	LR1-2	IDCLR01	GETPRT	An I/O error occurred while reading the catalog.
		IDCLR02	IDCLR02	An I/O error occurred while reading the catalog.
IDC18751	LR1-15	IDCLR01	TCICTCR	The CI from the catalog record could not be found in the CTT table therefore it could not be translated.
		IDCLR02	IDCLR02	The CI from the catalog record could not be found in the CCT table therefore it could not be translated.
IDC18781	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 the owning catalog name 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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			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.
IDC18781	LR1-9	IDCLR01	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.
IDC18801	LR1-11	IDCLR01	PRTVOL	Timestamp for the format-4 record could not be read for the CRA volume.
IDC18851	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.
IDC18871	RC0-22	IDCRC01	SCANCRA	I/O error encountered on a CRA record.
			TIMESTAMP	Volume timestamp could not be obtained.
IDC19271	EX0-12	IDCPM01	MARGPARM	Margin values specified are invalid.
IDC20351	TP6-3	IDCTP06	IDCTP06	An error was detected in the information transmitted in the error conversion table when attempting to convert a numeric error code to a prose message.
IDC22851	RC0-24	IDCRC01	IDCRC01	The name list built for this CRA is empty. No other errors occurred. The CRA has nothing to export.
IDC25521	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.
IDC25531	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.
IDC25561	DL0-6, DL0-7	IDCDL01	MORESP	No storage is available for a larger catalog work area.
IDC25631	LC1-4	IDCLC02	AUPROC	The allocation request conflicts with a nonVSAM or user catalog entry specified in the entry list.
			VPROC	The allocation request conflicts with a space (volume) entry specified in the entry list.
		IDCLC01	INITPROC	Either the allocation request conflicts with the type specification of cluster, alternate index, path, space, nonVSAM, or user catalog, or the volume request conflicts with

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				the type specification of cluster, alternate index or path.
IDC2616I	MP0-16	IDCMP01	CLUSPROC	A path import operation failed.
		IDCRM01	CLUSPROC	A path import operation failed.
IDC2618I	MP0-18	IDCMP01	CLUSPROC	An invalid object's subparameter was found.
IDC2620I	MP0-20	IDCRM01	ALISPROC	A recovery portable data set being imported contains objects not definable in DOS/VSE.
			GDGPROC	A recovery portable data set being imported contains objects not definable in DOS/VSE.
			FVTPROC	A recovery portable data set being imported contains a SAM ESDS. The SAM ESDS feature is not installed and the data set cannot be imported.
		IDCMP01	FVTPROC	A recovery portable data set being imported contains a SAM ESDS. The SAM ESDS feature is not installed and the data set cannot be imported.
IDC2621I	MP0-21	IDCRM01	IDCRM01	The object named could not be imported.
IDC2640I	BI0-1	IDCBI01	LOCPROC	The file identified via OUTFILE 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.
IDC2647I	BI0-8	IDCBI01	INITPROC	Storage was not available to obtain buffers and work areas.
IDC2648I	BI0-9	IDCBI01	JCPROC FINPROC	DLBL 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 from an OS/VS volume and is of a type that is not supported in DOS.
		IDCRC02	CLUSPROC	The object named is from an OS/VS volume and contains associations not supported in DOS.
			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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC26711	RC0-16	IDCRC01	CKCATNM	The CRA has a different name than the others being processed.
IDC26731	RC0-19	IDCRC01	BUILDCRV	The volume serial number could not be obtained for CRA dname. The volume is not processed.
		IDCLR01	CRAOPEN	The volume serial number cannot be obtained from INFILE *dname*. The volume is not processed.
IDC26751	RC0-21	IDCRC01	DUPNAMCK	The same name was found in more than one CRA.
IDC26771	RC0-1	IDCRC01	EXPORTDR	The data set was not exported because of the error indicated in previous messages.
IDC28721	LR1-3	IDCLR01	CRAOPEN	The catalog specified in the input for compare was not the owning catalog found in the CRA.
IDC28731	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.
IDC28761	LR1-6	IDCLR01	CRAOPEN	A verify was issued after opening a CRA and it failed.
IDC28791	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 resetting of the catalog CRAs while they are being listed or the lock request failed.
			CRAOPEN	The lock request to prevent concurrent updates to the catalog and CRAs failed.
IDC28821	LR1-13	IDCLR01	CTTBLD	LISTCRA encountered an error reading the catalog control record.
IDC28841	LR1-7	IDCLR01	CATOPEN	A verify was issued after opening a catalog and it failed.
IDC28861	RC0-18	IDCRC01	ERRCK	CRA can not be opened or locked because of some errors encountered.
IDC29501	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.
IDC29511	TP1-2	IDCTP01	IDCTP01	The output column specified is not within the print line.
IDC29521	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.
IDC29531	TP1-4	IDCTP01	REDO	A REDO structure is nested.
IDC29541	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
IDC29551	TP1-7	IDCTP01	PUPCONV	An invalid packed decimal field was passed by the caller.
IDC30031	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 EXTERNAL-SORT parameter was specified but the job control for sort files was missing or in error.
		IDCDE01	IDCDE01	The VSAM catalog to contain the defined object could not be opened, or another severe error occurred.
		IDCDE02	MODELPRC	The VSAM catalog containing the model object could not be opened.
		IDCDL01	CATOPEN	The VSAM catalog could not be opened.
		IDCLC01	IDCLC01	A severe error occurred. Listing of the catalog was not attempted or terminated if begun.
		IDCLR01	ERROR	A severe error has occurred.
		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.
		IDCRS05	CKERR	A severe error occurred which prevented further processing.
		IDCVY01	IDCVY01	The VSAM data set to be verified could not be opener, or the verify was not successful.
		IDCXP01	IDCXP01	A severe error occurred.
IDC30041	UV0-4	IDCAL01	ALTERPRC	Storage was not available for one of the following: the volume list or the PASSWALL field.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			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.
		IDCDE01	IDCDE01	Storage was not available for the CTGPL and CTGFV.
		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.
			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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			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.
		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 EXPOR-TRA.
		IDCRC02	CLUSPROC	Storage was not available for the control record output buffer.
			CTLGPROC	Storage was not available for the catalog work area.
			IDCRC02	Storage was not available for the output buffer area.
			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.
			RIINIT	Storage was not available for the Reader/Interpreter 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 parameter list.
			DELTPROC	Storage was not available for the catalog work area.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			FVTPROC	Storage was not available for the FVT or FPLs.
			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 control record buffer.
			RANGPROC	Storage was not available for the range list.
			UCATPROC	Storage was not available for the data record.
		IDCRS01	IDCRS01	Storage was not available for automatic storage for modules IDCRS02 - IDCRS07.
			INIT	Storage was not available for any one of the following: the record access buffers (RAB), the CRA user buffer, record management and umacro work area, catalog management work area, IKQMDADS parameter list, the CIXLT table, the UIOINFO return area.
		IDCRS03	GETTAB	Storage was not available for the association work area.
		IDCRS03	PROCVOL	Storage was not available for the space bit map.
		IDCRS03	VERB	Storage was not available for the GDG level difference string work area.
		IDCRS04	NINIT	Storage was not available for the FIND work area.
		IDCRS04	NXPND	Storage was not available to expand the FIND work area.
		IDCRS05	BLDRLST	Storage was not available for the RESVOL table.
		IDCRS05	BLDVLST	Storage was not available for the VOLSERTB.
		IDCRS06	WFDEF	Storage was not available for the CPL, FPL, and DEFINE work area.
		IDCRS07	RENMSETV	Storage was not available for the RENAME volume list.
		IDCSA08	IDCSANQ	Storage was not available for DTL (UENQ request).
		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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
			MORESP	Storage was not available for the catalog work area.
IDC30061	UV0-6	IDCPR01	DELIMSET	Beginning positioning failed.
		IDCRP01	DELIMSET	Beginning positioning failed.
IDC30071	(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.
		IDCDE02	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 code of 160, however, because 160 indicates a normal condition.
			FINDTYPE	The catalog return code was nonzero for a locate request.
			MORESP	The catalog return code was nonzero for a delete 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	INIT	The catalog return code was non-zero for a locate request.
		IDCRS06	WFDEF	The catalog return code was non-zero when defining the work-file.
			WFDEL	The catalog return code was non-zero when deleting the work-file.
		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.
			MORESP	The catalog return code was nonzero for a delete request.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC30091	(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.
		IDCDE02	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.
			MORESP	The catalog return code was nonzero for a delete request.
			LOCPROC	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	INIT	The catalog return code was non-zero for a locate request.
		IDCRS06	WFDEF	The catalog return code was non-zero when defining the work-file.
			WFDEL	The catalog return code was non-zero when deleting the work-file.
		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.
			MORESP	The catalog return code was nonzero for a delete request.
IDC30101	UV0-11	IDCAL01	IDCAL01	The file identified in the DLBL statement does not match that given in the CATALOG parameter.
		IDCDE01	IDCDE01	The file identified in the DLBL

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
		IDCDE02	MODELPRC	statement does not match that given in the CATALOG parameter.
		IDCDL01	CATOPEN	The file identified in the DLBL statement does not match that given in the CATALOG parameter.
		IDCLC01	INITPROC	The file identified in the DLBL statement does not match that given in the CATALOG parameter.
		IDCMP01	RECPROC	The file identified in the OUTFILE parameter does not match the name given in the IMPORT command or any paths over it.
		IDCXP01	RECPROC	The file identified in the INFILE parameter does not match that given in the EXPORT command or any paths over it.
IDC30121	TP6-9	IDCTP06	CATERCNV	Verbalization of catalog return code 8. The entry name supplied by the user is not in the specified catalog.
IDC30131	TP6-10	IDCTP06	CATERCNV	Verbalization of catalog return code 8. The file name supplied by the user is already in the catalog.
IDC30141	TP6-11	IDCTP06	CATERCNV	An error occurred during a VSAM catalog operation.
IDC30161	TP6-12	IDCTP06	CATERCNV	Verbalization of catalog return code 4. An error occurred while a VSAM catalog was being opened or closed or the user catalog specified by the command cannot be found.
IDC30171	TP6-13	IDCTP06	CATERCNV	Verbalization of catalog return code 20. The catalog or the catalog recovery area (CRA) is full.
IDC30181	TP6-14	IDCTP06	CATERCNV	Verbalization of catalog return code 56. The maximum number of attempts to supply the correct password was exceeded by the operator, or the user-specified verification routine failed to authorize use of the file.
IDC30191	TP6-15	IDCTP06	CATERCNV	Verbalization of catalog return code 60. Invalid catalog action request for the entry named.
IDC30201	TP6-16	IDCTP06	CATERCNV	Verbalization of catalog return code 68. Either an attempt was made to extend a unique VSAM file, or a specified volume either cannot accommodate an initial allocation, or cannot be extended when required.
IDC30211	TP6-17	IDCTP06	CATERCNV	Verbalization of catalog return code 72. Either an illegal system symbolic unit was assigned or no system symbolic unit was assigned.
IDC30221	TP6-18	IDCTP06	CATERCNV	Verbalization of catalog return code 80. The object specified in the RELATE parameter of a DEFINE command does not exist, or is improper for the type of object being defined.
IDC30231	TP6-19	IDCTP06	CATERCNV	Verbalization of catalog return code 84. An attempt to delete an entry failed because its expiration

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				date has not been reached, and the PURGE option was not specified.
IDC30241	TP6-21	IDCTP06	CATERCNV	Verbalization of catalog return code 148. A volume owned by another catalog was specified.
IDC30251	TP6-22	IDCTP06	CATERCNV	Verbalization of catalog return code 156. A volume does not contain a data space with sufficient room for allocation of another VSAM file.
IDC30261	TP6-23	IDCTP06	CATERCNV	Verbalization of catalog return code 172. A DEFINE operation specified the name of a file with the UNIQUE attribute, but there is already a file on the volume with the same name.
IDC30271	TP6-24	IDCTP06	CATERCNV	Verbalization of catalog return code 176. During the definition of a data space, an attempt was made to perform a VSAM allocate function, but there was no space in the VTOC for an additional label.
IDC30281	TP6-25	IDCTP06	CATERCNV	Verbalization of catalog return code 184. The catalog is currently open and cannot be deleted.
IDC30291	TP6-26	IDCTP06	CATERCNV	Verbalization of catalog return code 192. The maximum logical record length specified is greater than 32,761 for a nonspanned file.
IDC30301	TP6-27	IDCTP06	CATERCNV	Verbalization of catalog return code 196, 200. The data component control interval size specified is greater than 32,767; or the index component control interval size is greater than the maximum block size of the index device.
IDC30311	TP6-28	IDCTP06	CATERCNV	Verbalization of catalog return code 204. The KEY specification extends beyond the end of the maximum logical record.
IDC30321	TP6-29	IDCTP06	CATERCNV	Verbalization of catalog return code 208. The buffersize specified during a DEFINE operation is too small to contain the minimum number of control intervals for the VSAM file being defined.
IDC30331	TP6-30	IDCTP06	CATERCNV	Verbalization of catalog return code 248. This condition arises when a function requires a volume that is not owned by the referenced VSAM catalog.
IDC30441	TP6-39	IDCTP06	CATERCNV	Verbalization of catalog return code 16. The CYLINDER parameter was specified in the DEFINE command but the extents found on the corresponding DLBL/EXTENT statements do not start or end on a cylinder boundary.
IDC30451	TP6-40	IDCTP06	CATERCNV	Verbalization of catalog return code 152. An attempt was made to delete a non-empty VSAM catalog.
IDC30461	TP6-41	IDCTP06	CATERCNV	Verbalization of catalog return code 100. An attempt was made to define a unique file on a volume

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				that does not contain a catalog recovery area (CRA).
IDC30471	TP6-42	IDCTP06	CATERCNV	Verbalization of catalog return code 216. A space allocation attempt failed because the new extent specified in a EXTENT statement overlapped the volume table of contents (VTOC), an existing file or other extents specified in the DLBL statement.
IDC30481	TP6-43	IDCTP06	CATERCNV	Verbalization of catalog return code 240. A DLBL or EXTENT statement is missing or in error or a system logical unit error was detected.
IDC31711	DE0-24	IDCDE02	ALLCPROC	Value specified for CLASS, **primary** USECLASS, or **secondary** USECLASS is invalid.
		IDCMP01	CLUSPROC	OBJECTS parameter USECLASS has an invalid **primary** or **secondary** value.
		IDCRM01	CLUSPROC	OBJECTS parameter USECLASS has an invalid **primary** or **secondary** value.
IDC3173	DE0-26	IDCDE01	INTGCHK	A nonzero USECLASS was specified for a CLUSTER/AIX or component that has the UNIQUE allocation attribute.
		IDCMP01	CLUSPROC	A nonzero USECLASS was specified for a CLUSTER/AIX or component that has the UNIQUE allocation attribute.
		IDCRM01	CLUSPROC	A nonzero USECLASS was specified for CLUSTER/AIX or component that has the UNIQUE allocation attribute.
IDC31901	AL0-24	IDCAL01	PARAMCHK	One of the parameters specified on the command is invalid for the entry type.
IDC32001	RI0-1	IDCRI01	SCANCMD	The number of positional parameters found (PPARMCNT) exceeds the number defined in the descriptor for the current subparameter list (SUBCOUNT).
IDC3201	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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC32021	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.
IDC32031	RI0-4	IDCRI01	DSIDCHK	A data set name does not have the correct syntax.
IDC32051	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.
IDC32071	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.
IDC32081	RI0-9	IDCRI01	KWDPARM	A keyword parameter, defined as having a subfield, does not have a left parentheses following the keyword.
IDC32091	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.
IDC32101	RI0-11	IDCRI01	INREPEAT	The next repetition of a repeated subparameter list does not begin with a left parenthesis.
IDC32111	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.
IDC32121	RI0-13	IDCRI01	POSPARM	A positional parameter that is not defined as a list begins with a left parenthesis.
IDC32131	RI0-14	IDCRI01	SETFLAG	An internal table (PARMFLAG) indicates that the keyword just found was found previously in this command.
IDC32141	RI0-15	IDCRI01	GETDATA	A numeric constant begins with a B or X, but an apostrophe does not follow directly after this character.
IDC32161	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.
IDC32171	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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC32181	RI0-19	IDCRIO1	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.
IDC32191	RI0-20	IDCRI01	IDCRI02	The input verb name does not match any name in IDCRILT or the command descriptor phase could not be found in the core image library.
IDC32201	RI0-21	IDCRI01	CONVERT	A numeric constant contains a invalid digit.
			PACKCVB	A numeric constant contains an invalid digit.
IDC32211	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.
IDC32231	RI0-24	IDCRI01	BUILDFDT	The number of constants found in a list (SCLRCNT) exceeds the number allowed (LISTMAX).
IDC32251	RI0-26	IDCRI01	NEEDNOTS	A parameter required for this command is missing, or parameter required when another parameter is coded is missing.
		IDCDE01	INTGCHK	A parameter required for this command is missing, or parameter required when another parameter is coded is missing.
IDC32261	RI0-27	IDCRI01	NEEDNOTS	An input parameter conflicts with some other input parameter.
IDC32871	IO0-16	IDCIO02	BUILDACB	During CRA OPEN, IKQASNMT couldn't mount the CRA volume because the operator cancelled the mount request. Any return code from IKQASNMT that is not 0 or 4 will also cause this message. Examples are: Lock Table full, GETVIS failure, or incorrect NEWPAC response.
IDC32881	IO0-11	IDCIO02	BUILDACB	During CRA OPEN, IKQASNMT couldn't successfully do an auto assign for the CRA.
IDC32891	UV0-8	IDCSA08	IDCSANQ	IKQLOCK passed back a supervisor lock manager error code. No message if lock is held by another task or is held by this task (i.e., recursion). See Appendix B, code 246.
IDC32911	MP0-15	IDCMP01	DVOLCHK	DEFAULTVOLUMES parameter was specified for an object which has the UNIQUE or ORDERED attribute.
		IDCRM01	DVOLCHK	DEFAULTVOLUMES parameter was specified for an object which has the UNIQUE or ORDERED attribute or has the default model reserved name ("DEFAULT.MODEL.").

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC32921	DE0-32	IDCDE01	INTGCHK	VOLUMES parameter is required with UNIQUE or ORDERED component and for default models.
IDC32951	XP0-8	IDCXP01	CLUSPROC	The requested file is NOCIFORMAT or CIFORMAT SAM.
		IDCRC01	CKNAMES	The requested file is NOCIFORMAT or CIFORMAT SAM.
IDC32971	DE0-30	IDCDE02	NAMEPROC	The name for a VSAM object was specified with the prefix "DEFAULT.MODEL." but not followed by the valid qualifiers for the file type given.
		IDCDE01	INTGCHK	The name for a VSAM object was specified with the prefix "DEFAULT.MODEL." but not followedby the valid qualifiers for the file type given.
		IDCMP01	CNCTPROC	The specified name for the user catalog (connect) contains the prefix "DEFAULT.MODEL."
IDC32981	AL0-5	IDCAL01	IDCAL01	An attempt was made to either rename a file to a default model name or rename a default model name.
		IDCMP01	CLUSPROC	An attempt was made to either rename a file to a default model name or rename a default model name.
IDC32991	UV-12	IDCDE01	INTGCHK	Inconsistent parameters specified, modeled, or defaulted.
IDC33001	IO0-1	IDCIO02	BLDOCMSG	An error occurred during open of a data set.
IDC33011	IO0-2	IDCIO02	BLDOCMSG	An error occurred during close of a data set.
IDC33021	IO0-3	IDCIO01	BLDAMSG	An error occurred while accessing a data set.
		IDCIO03	BLDAMSG	An error occurred while accessing a data set.
		IDCRS06	RECERR	A logical I/O error occurred while processing a CRA, catalog or the work file.
IDC33031	IO0-4	IDCIO02	BUILDDBK	The data set to be opened for update processing is not a VSAM data set.
IDC33041	IO0-5	IDCIO02	DSDATA	A Job Control statement specified for file to OPEN was not found.
IDC33051	IO0-6	IDCIO02	DSDATA	An attempt was made to open an ISAM data set for output.
IDC33061	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.
IDC33071	IO0-8	IDCIO02	BUILDDBK	The data set to be opened for keyed processing is not a VSAM or ISAM data set.
IDC33081	IO0-10	IDCIO01	VSAMERR	A record with the same key or relative record number as the input record already exists in the output data set.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC33091I	IO0-12	IDCIO01	PUTNONVS	The length for a record to be written is invalid.
			PUTVSAM	Length invalid for RRDS.
IDC33101I	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.
IDC33111I	IO0-14	IDCIO03	IDCIO03	The data set to be positioned is not a VSAM or ISAM data set.
IDC33121I	IO0-15	IDCIO02	CKNONOP	The DTF OPEN flag was not set by the system OPEN routines for magnetic tape or for a sequential disk file.
IDC33141I	IO0-17	IDCIO01	VSAMERR	The record to be written has a lower key than the last record in the data set.
IDC33161I	IO0-19	IDCIO02	BUILDDBK	The data set to be opened is not a VSAM catalog.
IDC33171I	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.
IDC33181I	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, (4) the blocksize is invalid for a variable length record format file, or (5) for SAM files, Fixed Unblocked, RECSZ is given but is not equal to BLKSZ.
			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.
IDC33201I	IO0-23	IDCIO02	BUILDDBK	1. Invalid device type specified for prime data. 2. Invalid device type specified for high level index of an ISAM file. 3. Tape device specified as the high level index of an ISAM file.
IDC33211I	IO0-24	IDCIO02	CKNONOP	An open ABEND error was detected.
IDC33221I	IO0-25	IDCIO01	IDCIOVY	The data set to be verified is not a VSAM data set.
IDC33231I	IO0-34	IDCIO02	OPENCAT	A user catalog open error occurred.
IDC33241I	IO0-36	IDCIO02	OPENCAT	A user catalog open error has occurred and problem determination information has been returned by catalog management.
IDC33251I	IO0-45	IDCIO01	GETNONVS	The blocksize specified for the portable data set is different than that of the portable data set.
IDC33261I	IO0-46	IDCIO02	OPENRTN	The REPLACE option has been specified for output through a path.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3327I	IO0-47	IDCIO01	VSAMERR	Duplicate record in the upgrade set.
IDC3328I	IO0-48	IDCIO02	BUILDDBK	ISAM processing was requested for an FBA device.
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	RECERR	A logical I/O error occurred while processing a CRA, catalog or the work file.
IDC3500I	DE0-3	IDCDE03	IDCDE03	The object parameter list (FDT) supplied is not a valid subcommand type.
IDC3501I	DE0-4	IDCDE02	MODELPRC	The entry type of an 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.
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	Space allocation was incorrectly specified for a VSAM catalog, data set, or data space.
IDC3507I	DE0-8	IDCDE01	IDCDE01	The record size was required but not specified for a VSAM data set or data space.
IDC3513I	DE0-14	IDCDE01	IDCDE01	A file name was not specified with the UNIQUE attribute.
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	Key length and position were not specified for a key sequenced data set.
IDC3517I	DE0-18	IDCDE02	ALLPROC	Unequal record sizes were specified for a relative record data set.
IDC3518I	DE0-19	IDCDE01	IDCDE01	REUSE cannot be specified with UNIQUE or KEYRANGES.
IDC3519I	DE0-20	IDCDE01	IDCDE01	A REUSE conflict exists between data and index.
IDC3521I	DE0-22	IDCDE01	IDCDE01	A RECORDSIZE greater than 32761 was specified for a nonspanned data set.
IDC3522I	DE0-23	IDCDE01	IDCDE01	SPANNED cannot be specified for a relative record data set.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC3524I	DEO-25	IDCDE01	INTGCHK	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	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.
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.
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 exists.
IDC3541I	AL0-16	IDCAL01	CHECKPRC	A conflict exists between the alternate index and the base cluster.
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.
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.
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.
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, or data component name, or LRECL when the object to be exported was located.
		IDCRC01	SYNCH	No data association could be found.
		IDCRC02	CLUSPROC	Either (1) the catalog did not return the entry type, data component name, or LRECL when the object to be exported 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 exported was located.
			NVSMPROC	The catalog did not return the entry type, or data component name when the object to be exported 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 timestamp record was not valid, or the special record preceding the data records was not valid.
		IDCRM01	IDCRM01	The portable data set's timestamp record was not valid.
			ALISPROC	A catalog control record for an alias entry was not read.
			CLUSPROC	The special record preceding the data records was not valid.
			GDGPROC	A catalog control record for an OS/VS generation data group was not valid.
			NVSMPROC	A catalog control record for a nonVSAM entry was not valid.
			UCATPROC	A catalog control record for a user catalog was not valid.
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.
IDC3610I	MP0-6	IDCMP01	CNCTPROC	The OBJECTS parameter, volumes list, or 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.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
		IDCRM01	IDCRM01	The open of the portable data set was not successful.
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 when attempting to rename a catalog entry.
IDC3624I	MP0-24	IDCRM01	IDCRM01	The UIOINFO issued to obtain the output data set name failed.
IDC3641I	BI0-2	IDCBI01	LOCPROC	The file identified in INFILE 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.
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.
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 via EXCP. See "ABORT Codes" section for ABORT codes.
IDC01002I	RS0-3	IDCRS01	INIT	Informational message indicating the catalog to be reset and the timestamp on the volume.
IDC01011I	RS0-12	IDCRS01	PROCCRA	Informational message indicating the CRA to be reset and the timestamp 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	RS0-48 RS0-22	IDCRS06 IDCRS02	PROCTYPE PROCTYPE	An object contains a dependency on a record that does not exist.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
IDC110231	RS0-24	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
	RS0-23	IDCRS02	VERC VERG	imcomplete 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.
IDC110291	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).
IDC110311	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.
IDC110331	RS0-35	IDCRS03	CHKUNQ VLNRESET	A unique file, on a volume not being reset has no corresponding DATA or INDEX component.
IDC110361	RS0-46	IDCRS03	CHKDSDIR	The file named may have invalid space information. The extents occupied by the named file are not in conflict with any other VSAM file or with the system; however, a self-checking field failed to check.
IDC110401	RS0-38	IDCRS03	VOLCHK	The VSAM Format 1 Label did not have a corresponding header in the volume record. Therefore, the catalog does not account for the space allocated to the file.
IDC110411	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 Label.
IDC110421	RS0-40	IDCRS03	VOLCHK	The space header for the data space referred to a nonexistent Format 1 Label.
IDC110431	RS0-41	IDCRS03	VOLCHK	The timestamp for the volume record did not match the timestamp in the VTOC.
IDC110441	RS0-42	IDCRS03	VOLCHK	The attempt to scratch the file for the reason stated in message IDC110401 failed.
IDC210091	RS0-10	IDCRS01 IDCRS03	INIT MARKUNUS	A multivolume file existed on a volume prior to reset.
IDC210201	RS0-21	IDCRS05 IDCRS07	ADDUPCR RENMSETV	A volume needed for the reset was not specified in a CRAFILES or CRAVOLUMES parameter.
IDC210241	RS0-25	IDCRS02	VERX	The alias chain for a USERCATALOG or NONVSAM entry is invalid.
IDC210251	RS0-26	IDCRS03	VERB	The records associating the GDG file with the GDG base are in error.

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
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 IDCRS03	VLNRESET VLRESET	The CRA extents or catalog extents have no matching extents in any data space.
IDC21030I	RS0-32	IDCRS03	MARKUNUS	The entry noted claims space on volume. That space is not allocated to that entry.
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.
IDC21034I	RS0-36	IDCRS03	VLNRESET VLRESET	The space map, which indicates 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 contained an object of the same name.
IDC31000I	RS0-1	IDCRS01	INIT	The catalog specified for reset is not a recoverable catalog.
IDC31004I	RS0-5	IDCRS06	WFDEF	DEFINE failed for the workfile.
IDC31005I	RS0-6	IDCRS01	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 while the catalog was being extended.
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 file 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 workfile.
IDC31016I	RS0-17	IDCRS01	INIT	The CRAFILES or CRAVOLUMES parameter specified no CRA with the ALL option; there-

Messages to Module Cross Reference

Message	STID	Module	Procedure	Situation That Caused Message
				fore, 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 (via *dnames*) or CRAVOLUMES parameter specified the same volume serial number more than once
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.
IDC31039I	RS0-50	IDCRS01 IDCRS06	INIT WFDEF	The DLBL job control statement named in a CATALOG, CRAFILES, WORKCAT, or WORKFILE parameter cannot be found.
IDC31048I	RS0-51	IDCRS03	VOLCHK	Error accessing the VTOC.

Note: The listed procedures call UERROR to issue the IDC3007I and IDC3009I messages. UERROR issues the messages as follows:

Message	STID	Module	Procedure
IDC3007I	TP6-1	IDCTP06	IDCTP06
IDC3009I	TP6-2	IDCTP06	IDCTP06

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.

Attributes of Portable Data Sets

Attribute	Value
LRECL	The larger of: (a) Maximum VSAM data set record size +4 (b) 264 (for nonRRDSs) or 268 (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 timestamp 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 A-1 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.

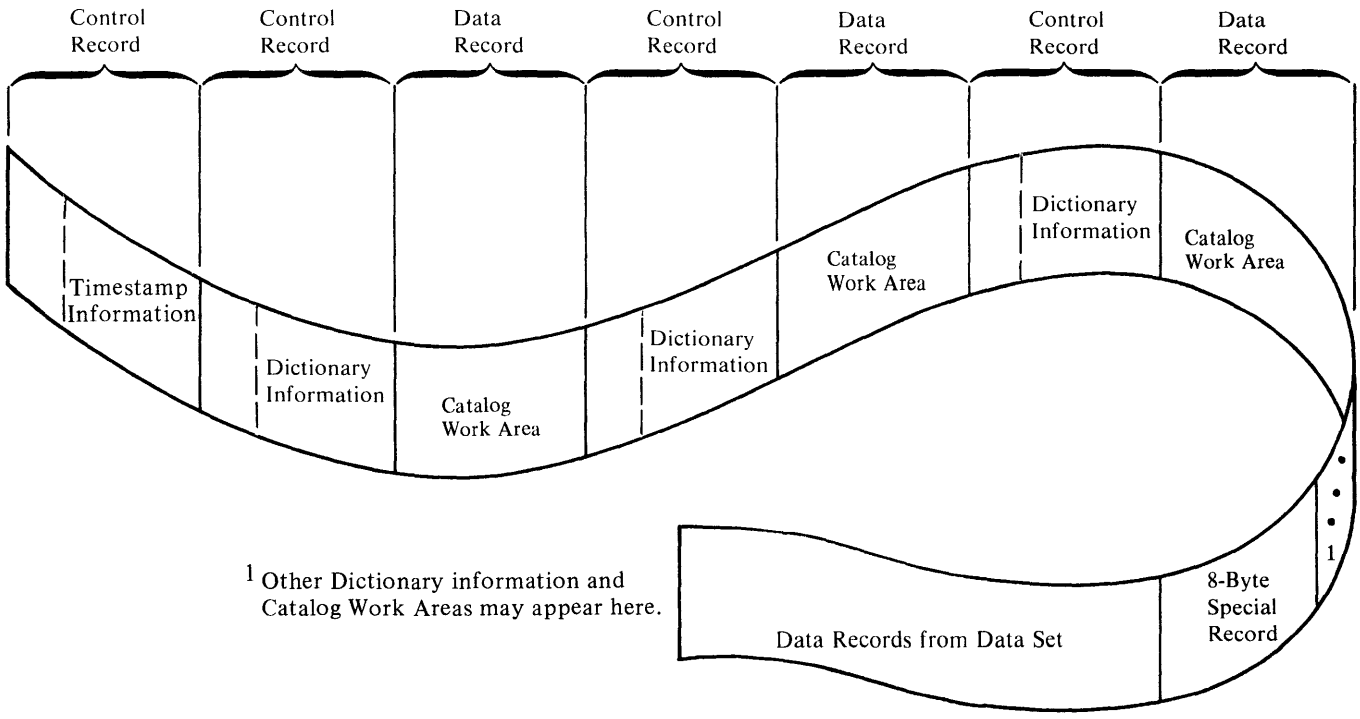


Figure A-1. Layout of Control Records and Data Records in the Portable Data Set

Control Records

Control records all have the same general format as shown in Figure A-2. 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 timestamp or dictionary information.

Control Record Containing Timestamp Information

The first record on every portable data set is a control record that contains timestamp information, as well as other fields. The format of this record is shown in Figure A-3.

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this control record contains timestamp information. There is no associated data, and those four bytes are reserved.

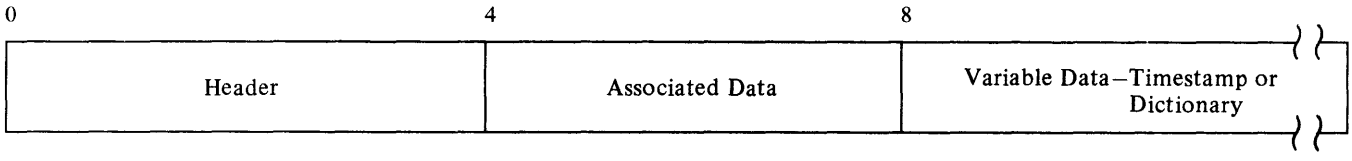


Figure A-2. General Format of Control Records

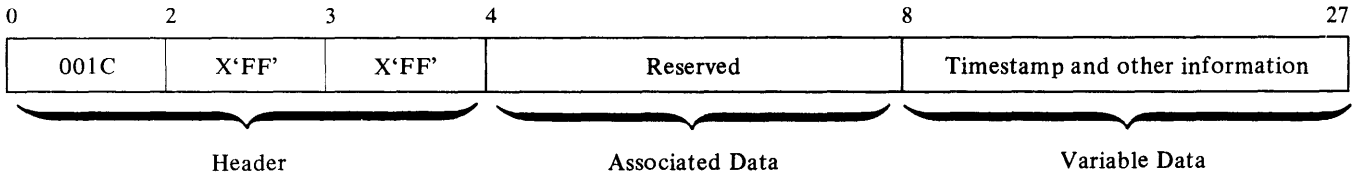


Figure A-3. Control Record Containing Timestamp Information

The format of the timestamp information is:

Displacement[1]	Description
8 (8)	Number of cluster components and paths being exported.
9 (9)	Flags: *Bit* — *Meaning When Set* 0 — Reserved (zero). 1 — Reserved. 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. 4-7 — Reserved.
10 (A)	Access Method Services release number in EBCDIC.
11 (B)	0 — 1 indicates export CIMODE. 0 indicates export RECORDMODE. 1 — 1 indicates a file with NOALLOCATE attribute. 0 indicates a file without NOALLOCATE attribute. 2 — 1 indicates a SAM ESDS file (in CI-format). 0 indicates a file that is not SAM ESDS. 3-7 — 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 A-4.

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).

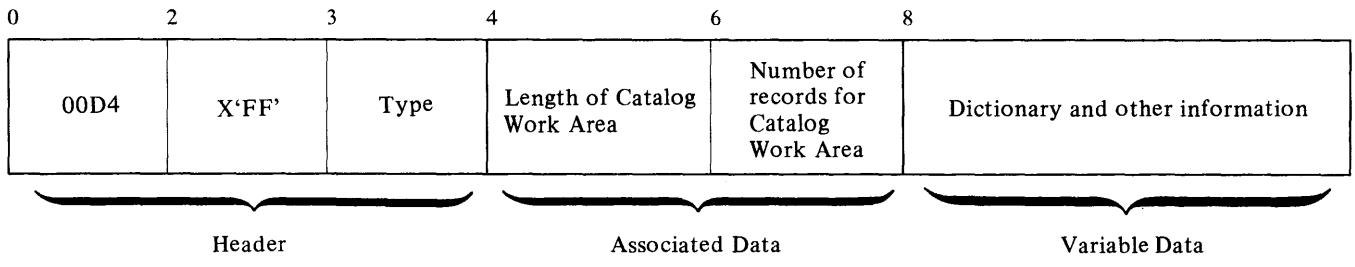


Figure A-4. Control Record Containing Dictionary Information

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 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 remaining fields are taken.
19	UNUSED	Reserved. Contains zeros.
20	UNUSED	Reserved. Contains zeros.
21	UNUSED	Reserved. Contains zeros.
22	UNUSED	Reserved. Contains zeros.
23	UNUSED	Reserved. Contains zeros.
24	UNUSED	Reserved. Contains zeros.
25	UNUSED	Reserved. Contains zeros.
26	UNUSED	Reserved. Contains zeros.
27	UNUSED	Reserved. Contains zeros.
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.

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 A-5.

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 A-6 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 A-7.

Following this special record are all of the data records from the data component being exported.

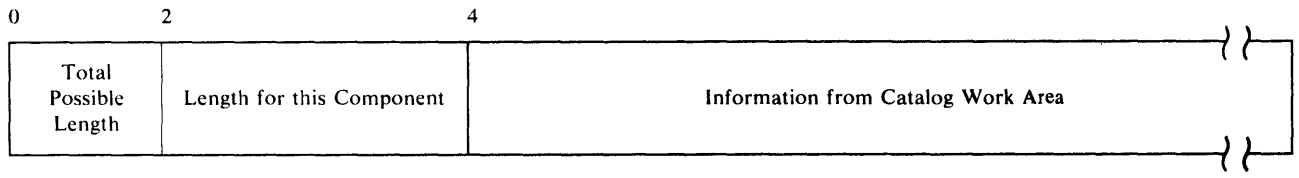


Figure A-5. Data Record Containing Catalog Work Area

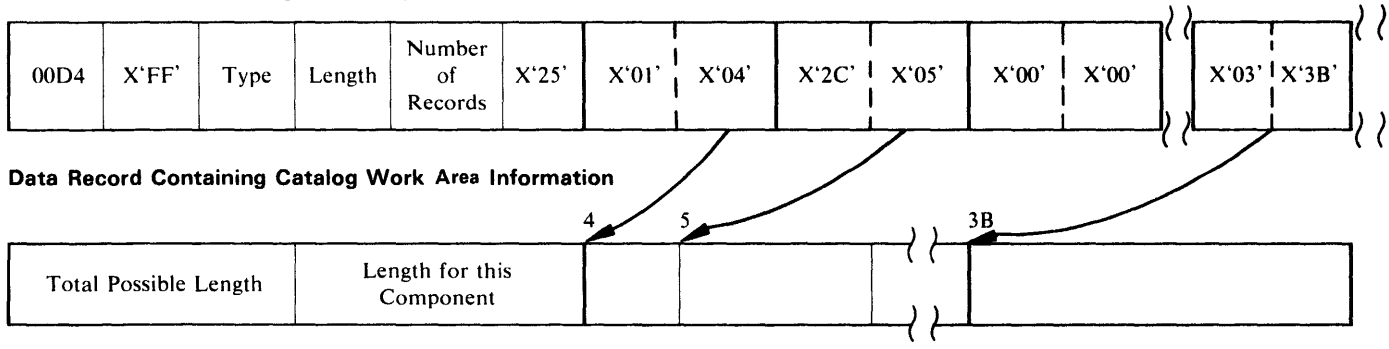


Figure A-6. Relationship of Dictionary and Catalog Work Area Information

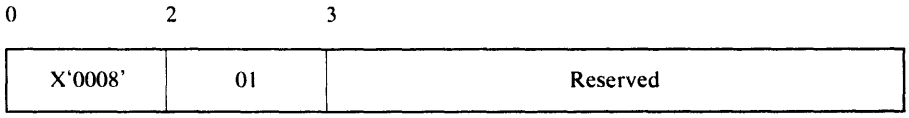


Figure A-7. Special Record at Beginning of Data Records from the Data Component

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 largest of: • Export RECORDMODE: Maximum VSAM data set record size + 16 • Export CIMODE: Maximum VSAM data set data component control interval size + 16, or • 268
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 B-1 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 timestamp 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 B-1 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.

Control Records

Control records all have the same general format as shown in Figure B-2. 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 timestamp, 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 B-3.

Control Record Containing Timestamp Information

The first record for each item on the recovery portable data set is a control record that contains timestamp information, as well as other fields. The format of this record is shown in Figure B-4.

The first two bytes of the header contain the length of this control record. The next two bytes indicate that this control record contains timestamp information. There is no associated data, and those four bytes are reserved.

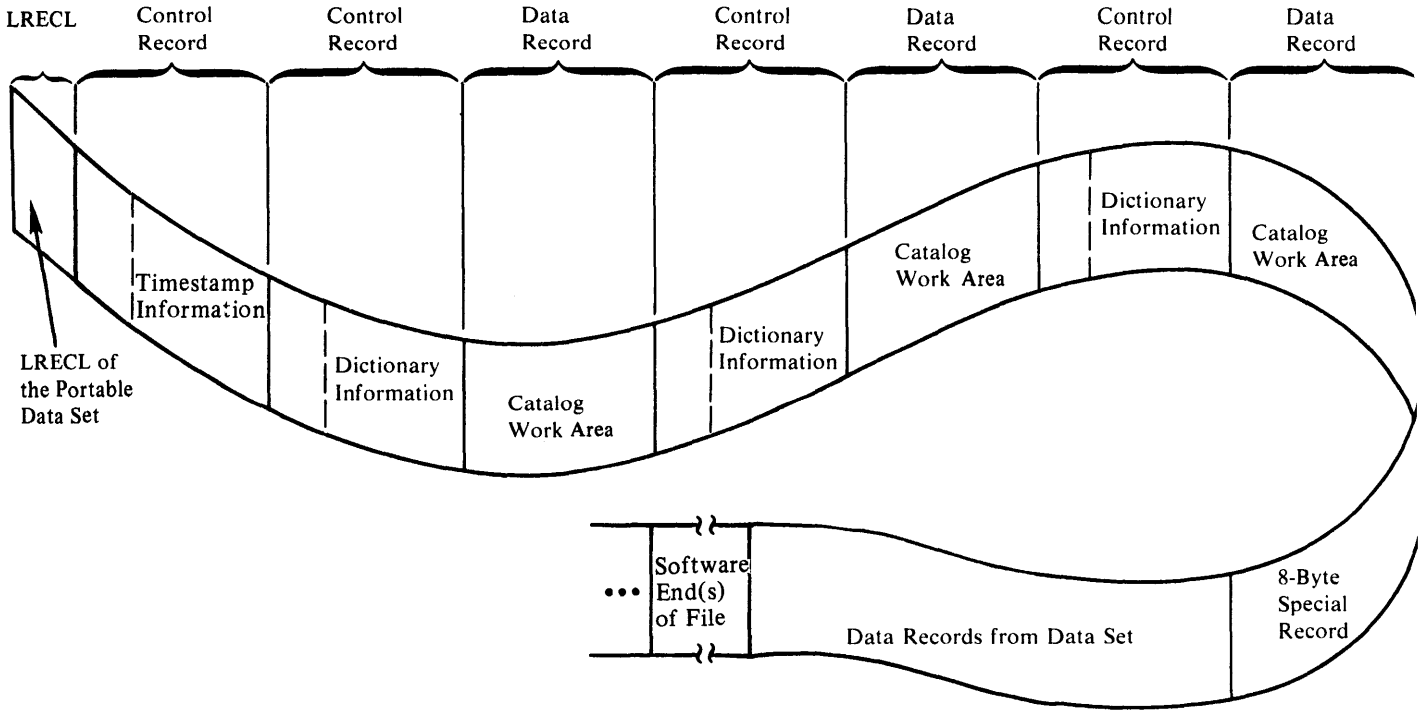


Figure B-1. Layout of Control Records and Data Records in the Recovery Portable Data Set

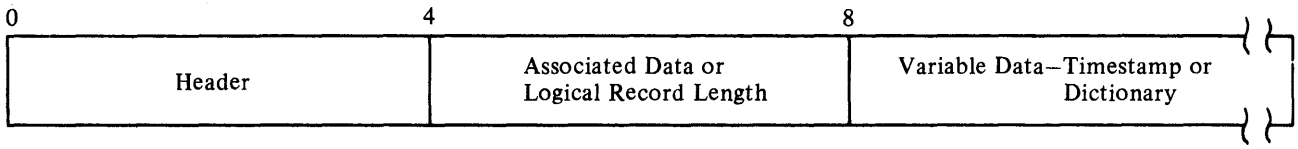


Figure B-2. General Format of Control Records

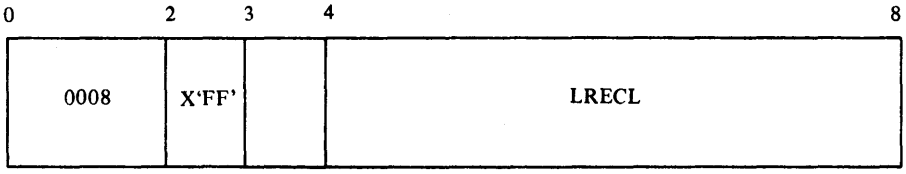


Figure B-3. Control Record Containing the Logical Record Length

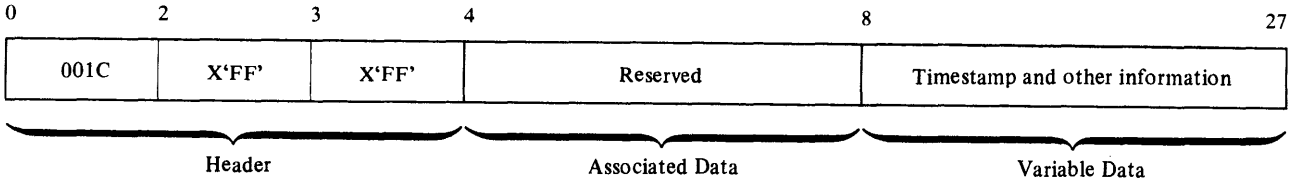


Figure B-4. Control Record Containing Timestamp Information

The format of the timestamp information is:

Displacement[1]	Description
8(8)	The maximum number of components associated with this item.
9(9)	Flags: *Bit* *Meaning When Set* 0 Reserved 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 pointer for an object other than a user catalog.
10(A)	Access Method Services release number in EBCDIC.
11(B)	0 1 indicates export CIMODE. 0 indicates export RECORDMODE. 1 1 indicates a file with NOALLOCATE attribute. 0 indicates a file without NOALLOCATE attribute. 2 1 indicates a SAM ESDS file (in CI-format). 0 indicates a file that is not SAM ESDS. 3 Reserved. 4 1 indicates that an empty file is being exported. 0 indicates that a non-empty file is being exported. 5-7 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 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 B-5.

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

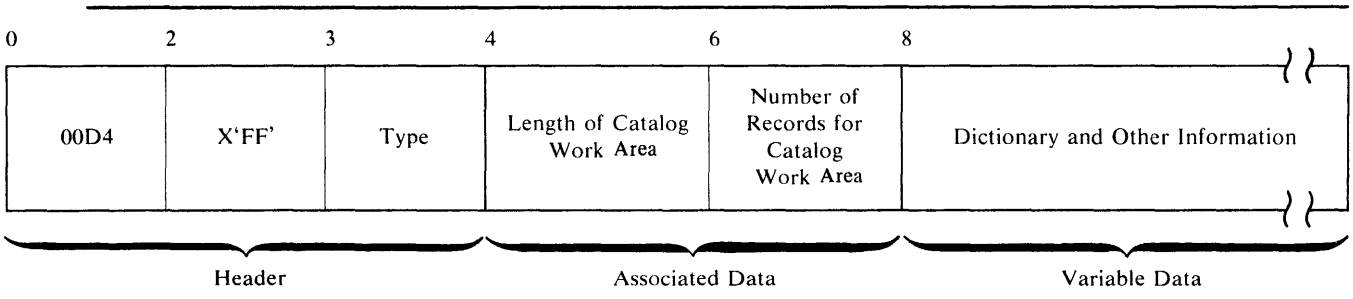


Figure B-5. Control Record Containing Dictionary Information

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

Cluster or Alternate Index

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 9 fields are taken.
19	UNUSED	Reserved. Contains zeros.
20	UNUSED	Reserved. Contains zeros.
21	UNUSED	Reserved. Contains zeros.
22	UNUSED	Reserved. Conatains zeros.
23	UNUSED	Reserved. Contains zeros.
24	UNUSED	Reserved. Contains zeros.
25	UNUSED	Reserved. Contains zeros.
26	UNUSED	Reserved. contains zeros.
27	UNUSED	Reserved. Contains zeros.
28	EXCPEXIT	Exception exit.
29	RGATTR	Alternate index or path attributes.
30	RELATE|	Alternate index related name or

Order of Associated Catalog Fields

Order	Associated Field in Catalog Work Area	Description	
		PATHENTRY	path entry name.
	31	PASSREL	Master password of path entry component.
	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.
	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 B-6.

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

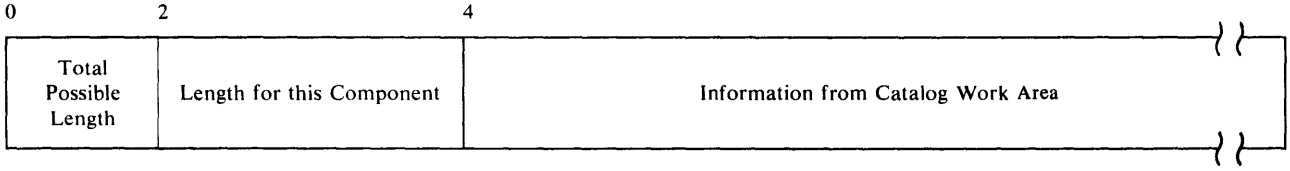


Figure B-6. Data Record Containing Catalog Work Area

the preceding topic. If there is no information for one of the fields, the field is completely omitted.

Figure B-7 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 B-8.

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.

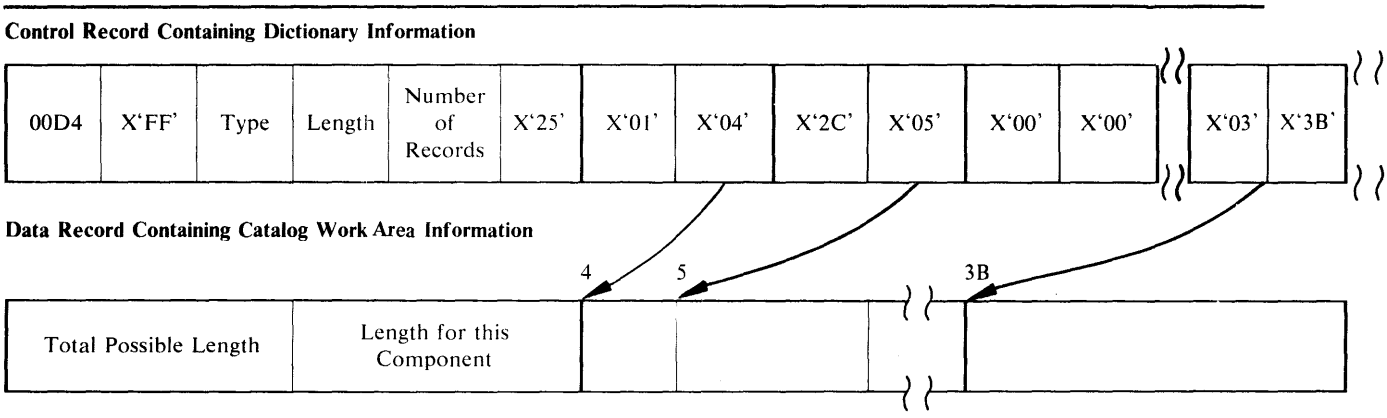


Figure B-7. Relationship of Dictionary and Catalog Work Area Information

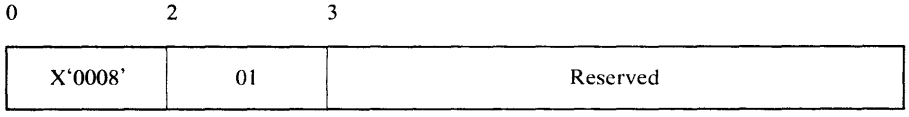


Figure B-8. Special Record at Beginning of Data Records from the Data Component

Index

ABORT codes 6-36
ACC 5-8
Access Method Services/Catalog Communication Table 5-8
Access Method Services
  functions 1-1
  initialization overview 2-8
  introduction 1-1
  logic features 1-1
  overview 2-4
  requirements 1-1
  structure 1-2
  visual table of contents 2-3
ALTER
  FDT 5-20
  IDCAL01 4-2
  method of operation 2-44
associated catalog fields A-3, B-4
associated objects B-6
attributes of portable data sets A-1
automatic storage areas, finding 6-45
AUTOTBL 6-46

BLDINDEX
  FDT 5-21
  IDCBI01 4-2
  method of operation 2-104
    get information and verify 2-108
    obtain resources and sort initialization 2-110
    sort-merge and build alternate index 2-112
BLKLIST 5-2
Block List 5-2
Buffer Pool Control Block (BUFS) 5-2
BUFS 5-2

CANCEL
  FDT 5-21
  IDCCL01 4-3
  macro 3-3
  method of operation 2-158
catalog management
  argument lists, finding 6-58
    sequence of calls made by FSRs 6-53
  debugging 6-52
  obtaining a dump 6-52
  order of associated catalog fields A-3, B-4
CATLG macro 3-3
CCB macro 3-3
CDLOAD macro 3-3
character code dependencies 1-8
CLOSE macro 3-3
Command Descriptor Phase Table (IDCRILT) 5-8
Command Descriptor 5-2
  areas
    Parameter Data Area 5-5
    Verb Data Area 5-2
  for ALTER (IDCCDAL) 4-3, 5-2
  for BLDINDEX (IDCCDBI) 4-3, 5-2
  for CANCEL (IDCCDCL) 4-3, 5-2
  for DEFINE (IDCCDDE) 4-3, 5-2
  for DELETE (IDCCDDL) 4-3, 5-2
  for EXPORT (IDCCDXP) 4-3, 5-2
  for EXPORTRA (IDCCDRC) 4-3, 5-2
  for IMPORT (IDCCDMP) 4-3, 5-2
  for IMPORTRA (IDCCDRM) 4-3, 5-2
  for LISTCAT (IDCCDLC) 4-3, 5-2
  for LISTCRA (IDCCDLR) 4-3, 5-2
  for PARM (IDCCDPM) 4-3, 5-2
  for PRINT (IDCCDPR) 4-3, 5-2
  for REPRO (IDCCDRP) 4-3, 5-2
  for RESETCAT (IDCCDRS) 4-3, 5-2
  for VERIFY (IDCCDVY) 4-3, 5-2
  format 5-2
  introduction 1-5
COMMAREA 5-54
COMRG macro 3-3
control flow 3-9
control records A-1, B-1
  containing directory information A-3 to A-5
  containing logical record length B-3
  containing timestamp information A-2, B-1
  general format A-2, B-1
control routing, IDCEX01 4-5
CRA Access Parameter List 5-8
CRA Access Translate Table 5-9
CRA Volume Timestamp Table 5-9
CTT 5-9
CVTOC macro 3-3

DARGLIST 5-10
data areas 5-1
  AUTOTBL 6-46
  Block List (BLKLIST) 5-2
  Buffer Pool Control Block (BUFS) 5-2
  Command Descriptor 5-3
  Command Descriptor Phase Table (IDCRILT) 5-8
  CRA Access Parameter List 5-8
  Dump List 5-9
  Dynamic Data List (DARGLIST) 5-10
  Error Conversion Table 5-11
  Field Management Parameter List (FMPL) 5-12
  Format List (FMTLIST) 5-13
  Function Data Table (FDT) 5-16
    ALTER FDT 5-20
    BLDINDEX FDT 5-21
    CANCEL FDT 5-21
    DEFINE FDT 5-22
    DELETE FDT 5-31
    EXPORT FDT 5-32
    EXPORTRA FDT 5-33
    IMPORT FDT 5-34
    IMPORTRA FDT 5-35
    LISTCAT FDT 5-36
    LISTCRA FDT 5-36
    PARM FDT 5-37
    PRINT FDT 5-37
    REPRO FDT 5-38
    RESETCAT FDT 5-40
    VERIFY FDT 5-40
  Global Data Table (GDT) 5-41
  I/O Adapter Historical Area (IODATA) 5-44
  I/O Communication Structure (IOCSTR) 5-44
  Input Parameter Table (IPT) 5-43
  Inter-Module Trace Table 5-47
  Intra-Module Trace Table 5-47
  IOCSTR Extension (IOCSEX) 5-46

Modal Verb and Keyword Symbol Table (IDCRIKT) 5-48
Open Argument List (OPNAGL) 5-48
Open Close Address Array (OCARRAY) 5-50
Phase Table 5-50
Positioning Argument List (OPRARG) 5-51
Print Control Argument List (PCARG) 5-51
Print Control Table (PCT) 5-52
Reader/Interpreter Communication Area (COMMAREA) 5-54
Reader/Interpreter Historical Area (HDAREA) 5-55
Scope Structure for UENQ (ENQSCOPE) 5-55
System Adapter Historical Area (SAHIST) 5-56
TEST Option 5-56
Test Structure 5-57
UGPOOL Area 5-58
UGSPACE Area 5-59
UIOINFO Area 5-59
UREST arguments 5-60
data records A-4, B-5
containing catalog work area A-5, B-5
containing data records A-1, B-1
relationship to control record A-1, B-1
debugging a formatting problem 6-59
debugging aids
introduction 1-1
method of operation
overview 2-260
UDUMP 2-264, 3-5
UDUMP - dump fields 2-266
UTRACE 2-264, 3-6
modules
IDCDB01 4-3
IDCDB02 4-3
visual table of contents 2-259
DEFINE
FDT 5-22
IDCDE01 4-3
method of operation 2-48
ALTERNATEINDEX 2-68
CLUSTER 2-64
MASTERCATALOG 2-50
NONVSAM 2-60
PATH 2-72
SPACE 2-62
USERCATALOG 2-56
DELETE
FDT 5-31
IDCDL01 4-4
method of operation 2-74
Diagnostic Aids 6-1
abort code 6-36
debugging a catalog problem 6-52
how to obtain a dump 6-52
debugging a formatting problem 6-59
how to obtain a dump 6-59
debugging a text processor problem 6-70
debugging an I/O problem 6-72
how to find I/O argument list 6-73
how to obtain a dump 6-72
OPEN argument lists 6-73
UGET and UPUT argument list 6-73
VSAM control block manipulation argument list 6-71
dump points 6-2, 6-26
dump, finding elements of
automatic storage areas 6-45
catalog management argument lists 6-58
dynamic storage areas 6-46
FDT 6-45
GDT 6-39
I/O argument lists 6-73
modules 6-38
phases 6-38
registers 6-38
save areas 6-39
trace tables 6-44
dump, sample 6-40
message to module cross reference 6-78
module to dump points cross reference 6-26
TEST option 6-3
trace and dump points to module cross reference 6-5
trace tables
inter-module 5-47, 6-1
intra-module 5-47, 6-1
DIMOD macro 3-3
DTFDI macro 3-3
DTFIS macro 3-3
DTFMT macro 3-3
DTFSD macro 3-3
Dump List 5-9
dump, reading 6-37
finding
automatic storage areas 6-45
catalog management argument lists 6-58
dynamic storage areas 6-46
FDT 6-45
GDT 6-39
I/O argument lists 6-73
modules 6-38
phases 6-38
registers 6-38
save areas 6-39
trace tables 6-44
points 6-2, 6-26
sample dump 6-40
Dynamic Data List (DARGLIST) 5-10
dynamic storage areas, finding 6-46

ENDREQ macro 3-3
ENQSCOPE 5-55
EOJ macro 3-3
ERASE macro 3-3
ERCNVTAB 5-11
Error Conversion Table (ERCNVTAB) 5-11
EXCP macro 3-3
executable load modules
IDCAL01 4-2
IDCAMS 4-2
IDCBI01 4-2
IDCCL01 4-3
IDCDB01 4-3
IDCDB02 4-3
IDCDE01 4-3
IDCDE02 4-3
IDCDE03 4-4
IDCDI01 4-4
IDCDI02 4-4
IDCDI03 4-4
IDCDI04 4-4
IDCDI05 4-4
IDCDI06 4-4
IDCDI07 4-4
IDCDI08 4-4

IDCDI09 4-4
IDCDI10 4-4
IDCDI11 4-4
IDCDI12 4-4
IDCDI13 4-4
IDCDI14 4-4
IDCDI15 4-4
IDCDL01 4-4
IDCEX01 4-5
IDCEX02 4-5
IDCEX03 4-5
IDCIO01 4-5
IDCIO02 4-6
IDCIO03 4-6
IDCLC01 4-6
IDCLC02 4-7
IDCLR01 4-7
IDCLR02 4-8
IDCMP01 4-8
IDCPM01 4-10
IDCPR01 4-10
IDCRC01 4-10
IDCRC02 4-11
IDCRC03 4-12
IDCRC04 4-12
IDCRIFF 4-14
IDCRIKT 4-14
IDCRILT 4-14
IDCRI01 4-12
IDCRI02 4-13
IDCRI03 4-13
IDCRM01 4-14
IDCRP01 4-15
IDCRS01 4-15
IDCRS02 4-16
IDCRS03 4-16
IDCRS04 4-16
IDCRS05 4-16
IDCRS06 4-17
IDCRS07 4-17
IDCSA01 4-17
IDCSA02 4-17
IDCSA03 4-18
IDCSA04 4-18
IDCSA05 4-18
IDCSA08 4-18
IDCTP01 4-18
IDCTP04 4-19
IDCTP05 4-19
IDCTP06 4-19
IDCVY01 4-20
IDCXP01 4-20
executive controlled termination 2-162
Executive
introduction 1-1
modules
IDCEX01 4-5
IDCEX02 4-5
IDCEX03 4-5
EXPORT
FDT 5-32
IDCXP01 4-20
method of operation 2-76, 2-120
ALTERNATEINDEX 2-78
CLUSTER 2-78
export nonVSAM 2-128
export VSAM data set 2-126
EXPORTRA driver 2-124
field management 2-122
portable data seta A-1
EXPORTRA
FDT 5-33
modules
IDCRC01 4-10
IDCRC02 4-11
IDCRC03 4-12
IDCRC04 4-12
portable data sets B-1
external entry point 3-7
external exit point 3-7
EXTRACT macro 3-3

FDT 5-16
finding the 6-45
introduction 1-5
Field Management Parameter List (FMPL) 5-12
finding
automatic storage areas 6-45
catalog management argument lists 6-58
dynamic storage areas 6-46
FDT 5-16, 6-45
GDT 5-41, 6-39
I/O argument lists 6-73
modules 6-38
phases 6-38
registers 6-38
save areas 6-39
trace tables 6-44
flow of control 3-9
FMPL 5-12
FMTLIST 5-13
Format List 5-13
format, debugging a problem 6-59
FREEVIS macro 3-3
FSRs
introduction 1-1
Function Data Table 5-16
Function Support Routines
(see desired routine
ALTER
BLDINDEX
CANCEL
DEFINE
DELETE
EXPORT
EXPORTRA
IMPORT
IMPORTRA
LISTCAT
LISTCRA
PARM
PRINT
REPRO
RESETCAT
VERIFY)
visual table of contents 2-43

GDT 5-41
finding the 6-39
introduction 1-1
GET macro 3-3
GETTIME macro 3-3
GETVIS macro 3-3

Global Data Table 5-41
finding the 6-39
introduction 1-1

HDAREA 5-55
hierarchy of modules 3-1

I/O Adapter Historical Area 5-44
I/O adapter initialization 2-12
I/O adapter termination 2-160
I/O Adapter
introduction 1-1
method of operation
overview 2-208
UCLOSE 2-220, 3-5
UCOPY 2-230, 3-5
UGET 2-224, 3-5
UIOINFO 2-234, 3-6
UIOINT 3-6
UIOTERM 3-6
UOPEN 2-210, 3-6
UOPEN - build control blocks 2-216
UOPEN - build IOCSTR 2-212
UOPEN - ckeck open 2-218
UPOSIT 2-222, 3-6
UPUT 2-226, 3-6
UVERIFY 2-232, 3-6
modules
IDCIO01 4-5
IDCIO02 4-6
IDCIO03 4-6
visual table of contents 2-207
I/O argument lists, finding 6-73
I/O Communication Structure 5-44
I/O macros 3-3
I/O problem, debugging 6-72
IDCAL01 4-2
IDCAMS 4-2
IDCBI01 4-2
IDCCDAL 4-3, 5-2
IDCCDBI 4-3, 5-2
IDCCDCL 4-3, 5-2
IDCCDDE 4-3, 5-2
IDCCDDL 4-3, 5-2
IDCCDLC 4-3, 5-2
IDCCDLR 4-3, 5-2
IDCCDMP 4-3, 5-2
IDCCDPM 4-3, 5-2
IDCCDPR 4-3, 5-2
IDCCDRC 4-3, 5-2
IDCCDRM 4-3, 5-2
IDCCDRP 4-3, 5-2
IDCCDRS 4-3, 5-2
IDCCDVY 4-3, 5-2
IDCCDXP 4-3, 5-2
IDCCL01 4-3
IDCDB01 4-3
IDCDB02 4-3
IDCDE01 4-3
IDCDE02 4-3
IDCDE03 4-4
IDCDI01 4-4
IDCDI02 4-4
IDCDI03 4-4
IDCDI04 4-4
IDCDI05 4-4
IDCDI06 4-4
IDCDI07 4-4
IDCDI08 4-4
IDCDI09 4-4
IDCDI10 4-4
IDCDI11 4-4
IDCDI12 4-4
IDCDI13 4-4
IDCDI14 4-4
IDCDI15 4-4
IDCDL01 4-4
IDCEX01 4-5
IDCEX02 4-5
IDCEX03 4-5
IDCIO01 4-5
IDCIO02 4-6
IDCIO03 4-6
IDCLC01 4-6
IDCLC02 4-7
IDCLR01 4-7
IDCLR02 4-8
IDCMP01 4-8
IDCPM01 4-10
IDCPR01 4-10
IDCRC01 4-10
IDCRC02 4-11
IDCRC03 4-12
IDCRC04 4-12
IDCRIFF 4-14
IDCRIKT 4-14, 5-48
IDCRILT 4-14
IDCRI01 4-12
IDCRI02 4-13
IDCRI03 4-13
IDCRM01 4-14
IDCRP01 4-15
IDCRS01 4-15
IDCRS02 4-16
IDCRS03 4-16
IDCRS04 4-16
IDCRS05 4-16
IDCRS06 4-17
IDCRS07 4-17
IDCSA01 4-17
IDCSA02 4-17
IDCSA03 4-18
IDCSA04 4-18
IDCSA05 4-18
IDCSA08 4-18
IDCTP01 4-18
IDCTP04 4-19
IDCTP05 4-19
IDCTP06 4-19
IDCTSAL0 4-19, 5-57
IDCTSBI0 4-19, 5-57
IDCTSDE0 4-19, 5-57
IDCTSDL0 4-19, 5-57
IDCTSEX0 4-19, 5-57
IDCTSIO0 4-19, 5-57
IDCTSLC0 4-19, 5-57
IDCTSLC1 4-19, 5-57
IDCTSLR0 4-19, 5-57
IDCTSLR1 4-19, 5-57
IDCTSMP0 4-19, 5-57
IDCTSPR0 4-19, 5-57
IDCTSRC0 4-20, 5-57
IDCTSRI0 4-20, 5-57

IDCTSRS0 4-20, 5-57
IDCTSTP0 4-20, 5-57
IDCTSTP1 4-20, 5-57
IDCTSTP6 4-20, 5-57
IDCTSUV0 4-20, 5-57
IDCTSXP0 4-20, 5-57
IDCVY01 4-20
IDCXP01 4-20
IMPORT
  FDT 5-34
  IDCMP01 4-8
  method of operation 2-84, 2-130
    ALTERNATEINDEX 2-84
    CLUSTER 2-84
    cluster or alternate index 2-132
    GDB base 2-138
    nonVSAM 2-136
    user catalog 2-134
IMPORTRA
  FDT 5-35
  IDCRM01 4-14
initialization
  I/O adapter 2-12
  IDCEX02 4-5
  system adapter 2-10
  visual table of contents 2-7
Input Parameter Table (IPT) 5-43
Inter-Module Trace Table 5-47, 6-1
internal services 3-5
Intra-Module Trace Table 5-47, 6-1
invoking user I/O routine 3-9
  arguments passed 3-9
IOCSEX 5-46
IOCSTR 5-44
IOCSTR Extensions 5-46
IODATA 5-44
IPT 5-43
ISMOD macro 3-4

job control 3-7

LASTCC 3-9
LISTCAT
  FDT 5-36
  IDCLC01 4-6
  IDCLC02 4-7
  method of operation 2-88
    gets information 2-92
LISTCRA
  FDT 5-36
  method of operation 2-116
    process CRA 2-118
  modules
    IDCLR01 4-7
    IDCLR02 4-8
LOAD macro 3-4

macros used, system and I/O 3-3
  CANCEL 3-3
  CATLG 3-3
  CCB 3-3
  CDLOAD 3-3
  CLOSE 3-3
  COMRG 3-3
  CVTOC 3-3
  DIMOD 3-3
  DTFDI 3-3
  DTFIS 3-3
  DTFMT 3-3
  DTFSD 3-3
  ENDREQ 3-3
  EOJ 3-3
  ERASE 3-3
  EXCP 3-3
  EXTRACT 3-3
  FREEVIS 3-3
  GET 3-3
  GETTIME 3-3
  GETVIS 3-3
  ISMOD 3-4
  LOAD 3-4
  MTMOD 3-4
  OPEN 3-4
  OVTOC 3-4
  PDUMP 3-4
  POINT 3-4
  PUT 3-4
  PVTOC 3-4
  SDMODFI 3-4
  SDMODFO 3-4
  SDMODUI 3-4
  SDMODUO 3-4
  SDMODVI 3-4
  SDMODVO 3-4
  SETL 3-4
  TRUNC 3-4
  VERIFY 3-4
  WAIT 3-4
MAXCC 3-7
message to module cross reference 6-78
messages 6-78
method of operation 2-1
  (see specific element desired)
  legend 1-7
microfiche directory 4-1
Modal Verb and Keyword Symbol Table (IDCRIKT) 5-48
module to dump points cross reference 6-26
modules, finding 6-38
MTMOD macro 3-4

naming conventions
  example 1-7
  for Command Descriptors 1-7
  for data areas 1-7
  for executable load modules 1-7
  for multiple entry-point modules 1-7
  for single entry-point modules 1-7
  for Text Structures 1-7
  mnemonic identifiers 1-7
nonexecutable load modules
  command descriptors 5-2
    IDCCDAL 4-3, 5-2
    IDCCDBI 4-3, 5-2
    IDCCDCL 4-3, 5-2
    IDCCDDE 4-3, 5-2
    IDCCDDL 4-3, 5-2
    IDCCDLC 4-3, 5-2
    IDCCDLR 4-3, 5-2
    IDCCDMP 4-3, 5-2
    IDCCDPM 4-3, 5-2
    IDCCDPR 4-3, 5-2

IDCCDRC 4-3, 5-2
IDCCDRM 4-3, 5-2
IDCCDRP 4-3, 5-2
IDCCDRS 4-3, 5-2
IDCCDVY 4-3, 5-2
IDCCDXP 4-3, 5-2
IDCRIFF 4-14
IDCRIKT 4-14, 5-48
IDCRILT 4-14, 5-8
text structures 5-57
IDCTSAL0 4-19, 5-57
IDCTSBI0 4-19, 5-57
IDCTSDE0 4-19, 5-57
IDCTSDL0 4-19, 5-57
IDCTSEX0 4-19, 5-57
IDCTSIO0 4-19, 5-57
IDCTSLC0 4-19, 5-57
IDCTSLC1 4-19, 5-57
IDCTSLR0 4-19, 5-57
IDCTSLR1 4-19, 5-57
IDCTSMP0 4-19, 5-57
IDCTSPR0 4-19, 5-57
IDCTSRC0 4-20, 5-57
IDCTSRI0 4-20, 5-57
IDCTSRS0 4-20, 5-57
IDCTSTP0 4-20
IDCTSTP1 4-20
IDCTSTP6 4-20
IDCTSUV0 4-20
IDCTSXP0 4-20

OCRRAY 5-50
Open Argument List (OPNAGL) 5-48
Open Close Address Array (OCRRAY) 5-50
OPEN macro 3-4
OPNAGL 5-48
OPRARG 5-51
Order of Associated Catalog Fields A-3, B-4
overview, Access Method Services 2-4
OVTOC macro 3-4

parameter data area 5-5
PARM
FDT 5-37
IDCPM01 4-10
method of operation 2-94
TEST option 6-3
parsing the command 1-5
PCARG 5-51
PCT 5-52
PDUMP macro 3-4
phase table 5-50
phases, finding 6-38
POINT macro 3-4
portable data set
(see also EXPORT, EXPORTRA, IMPORT)
attributes of A-1, B-1
major types of records A-1, B-1
control A-2, B-1
data A-1, B-5
special record A-5, B-6
types of control information A-1, B-1
dictionary A-3, B-3
timestamp A-2, B-1
types of data information A-1
catalog work area A-5, B-5
data record A-5, B-5
Position Argument List (OPRARG) 5-51
Print Control Argument List (PCARG) 5-51
Print Control Table (PCT) 5-52
PRINT
FDT 5-37
IDCPR01 4-10
method of operation 2-96
processor condition codes 3-7
LASTCC 3-9
MAXCC 3-7
processor invocation 3-7
arguments passed 3-8
processor termination 2-164
program organization
introduction 3-1
overall organization 3-1
root phase 3-1
PROLOG 3-5
PUT macro 3-4
PVTOC macro 3-4

Reader/Interpreter Communication Area (COMMAREA) 5-54
Reader/Interpreter Historical Area (HDAREA) 5-55
Reader/Interpreter
character code dependencies 1-8
introduction 1-1, 1-4
method of operation
build FDT 2-38
do modal command 2-28
else modal command 2-24
end modal command 2-30
get next command 2-20
if-then modal command 2-22
prepare to scan command 2-32
scan command 2-34
set modal command 2-26
syntex check parameter 2-36
termination 2-40
visual table of contents 2-15
modules
IDCRI01 4-12
IDCRI02 4-13
IDCRI03 4-13
reading a dump 6-37
register, finding 6-38
REPRO
FDT 5-38
IDCRP01 4-15
method of operation 2-98
catalog reload 2-100
requirements
storage 1-1
system 1-1
RESETCAT
FDT 5-40
method of operation 2-140
check associations 2-152
common VTOC handler functions 2-148
copy catalog to work file 2-144
initialization 2-142
merge CRAs to work file 2-146
reassign CI numbers 2-150
update the catalog 2-154
update the CRA 2-156

modules
IDCRS01 4-15
IDCRS02 4-16
IDCRS03 4-16
IDCRS04 4-16
IDCRS05 4-16
IDCRS06 4-17
IDCRS07 4-17
return codes 3-9
root phase 3-1

SAHIST 5-56
save areas, finding 6-39
scope sturcture for UENQ (ENQSCOPE) 5-55
SDMODFI macro 3-4
SDMODFO macro 3-4
SDMODUI macro 3-4
SDMODUO macro 3-4
SDMODVI macro 3-4
SDMODVO macro 3-4
SETL macro 3-4
storage requirements 1-1
substructure 1-1, 1-2
Executive 1-2
I/O Adapter 1-2
Reader/Interpreter 1-2, 1-4
System Adapter 1-2, 1-3
Text Processor 1-2
superstructure 1-1, 1-2
FSRs 1-2
System Adapter Historical Area (SAHIST) 5-56
System Adapter
initialization 2-10
introduction 1-2, 1-3
method of operation
overview 2-170
PROLOG 2-192, 3-5
UABORT 2-174, 3-5
UCALL 2-178, 3-5
UCATLG 2-172, 3-5
UDELETE 2-182, 3-5
UDEQ 2-204, 3-5
UENQ 2-202, 3-5
UEPIL 2-194, 3-5
UFPOOL 2-190, 3-5
UFSPACE 2-186, 3-5
UGPOOL 2-188, 3-6
UGSPACE 2-184, 3-6
ULISTLN 2-198, 3-6
ULOAD 2-180, 3-6
USAVERC 2-200, 3-6
USNAP 2-176, 3-6
UTIME 2-196, 3-6
modules
IDCSA01 4-17
IDCSA02 4-17
IDCSA03 4-18
IDCSA04 4-18
IDCSA05 4-18
IDCSA08 4-18
visual table of contents 2-169
system macros 3-3
system requirements 1-1

termination
IDCEX03 4-5
visual table of contents 2-161

TEST keyword 6-3
Test Option Data Area 5-56
TEST option 6-3
how to use 6-4
module to dump points cross reference 6-26
TEST keyword 6-3
trace dump points to module cross reference 6-5
Text Processor
character code dependencies 1-8
debugging 6-71
how to find argument list 6-71
introduction 1-1
method of operation
overview 2-238
UERROR 2-256, 3-5
UESTA 2-242, 3-5
UESTS 2-240, 3-5
UPRINT 2-248, 3-6
UPRINT - CONVERT 2-252
UPRINT - Print 2-154
URESET 2-246, 3-6
UREST 2-244, 3-6
modules
IDCTP01 4-18
IDCTP04 4-19
IDCTP05 4-19
IDCTP06 4-19
obtaining a dump 6-70
visual table of contents 2-237
Text Structures 5-57
for ALTER messages (IDCTSAL0) 4-19, 5-57
for BLDINDEX messages (IDCTSBI0) 4-19, 5-57
for DEFINE messages (IDCTSDE0) 4-19, 5-57
for DELETE messages (IDCTSDL0) 4-19, 5-57
for Executive messages (IDCTSEX0) 4-19, 5-57
for EXPORT messages (IDCTSXP0) 4-20, 5-57
for EXPORTRA messages (IDCTSRC0) 4-20, 5-57
for I/O Adapter messages (IDCTSIO0) 4-19, 5-57
for IMPORT/IMPORTRA messages (IDCTSMP0) 4-19, 5-57
for LISTCAT listing (IDCTSLC0) 4-19, 5-57
for LISTCAT messages (IDCTSLC1) 4-19, 5-57
for LISTCRA listing (IDCTSLR0) 4-19, 5-57
for LISTCRA messages (IDCTSLR1) 4-19, 5-57
for PRINT listings (IDCTSPR0) 4-19, 5-57
for Reader/Interpreter messages (IDCTSRI0) 4-20, 5-57
for RESETCAT messages (IDCTSRS0) 4-20, 5-57
for Text Processor (IDCTSTP0) 4-20, 5-57
for Text Processor messages (IDCTSTP1) 4-20, 5-57
for universal messages (IDCTSUV0) 4-20, 5-57
for VERROR messages (IDCTSTP6) 4-20, 5-57
format 5-57
timestamp information A-2, B-1
trace and dump points to module cross reference 6-5
trace tables
finding the 6-44
Inter-Module 5-47, 6-1
Intra-Module 5-47, 6-1
TRUNC macro 3-4

UABORT 3-5
UCALL 3-5
UCATLG 3-5
UCLOSE 3-5
UCOPY 3-5
UDELETE 3-5

UDEQ 3-5
UDUMP 3-5
UENQ 3-5
UEPIL 3-5
UERROR 3-5
UESTA 3-5
UESTS 3-5
UFPOOL 3-5
UFSPACE 3-5
UGET 3-5
UGPOOL 3-6
UGPOOL Area 5-58
UGPOOL Area contents 6-48
UGPOOL ID list 6-48
UGSPACE 3-6
UGSPACE area 5-59
UIOINFO 3-6
UIOINFO area 5-59
UIOINIT 3-6
  macro 2-12
UIOTERM 3-6
  macro 2-166
ULISTLN 3-6
ULOAD 3-6
UOPEN 3-6
UPOSIT 3-6
UPRINT 3-6
UPUT 3-6
URESET 3-6
UREST 3-6
UREST arguments 5-60
USAVERC 3-6
User I/O Routines 3-9
USNAP 3-6
UTIME 3-6
UTRACE 3-6
UVERIFY 3-6

VERIFY
  FDT 5-40
  IDCVY01 4-20
  macro 3-4
  method of operation 2-102
visual table of contents
  Access Method Services 2-3
  debugging aids 2-259
  function support routine 2-43
  I/O adapter 2-207
  initialization 2-7
  Reader/Interpreter 2-15
  system adapter 2-169
  text processor 2-237
VTT 5-9

WAIT macro 3-4
WAIT macro 3-4

VSE/VSAM Access Method Services Logic (File No. S370-30) Printed in U.S.A. LY24-5195-1

VSE/VSAM
Access Method Services Logic
LY24-5195-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

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

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

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

• What is your occupation? _____

• How do you use this publication:

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

Your comments:

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

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut or Fold Along Line

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

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

Fold | Fold

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

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

VSE/VSAM Access Method Services Logic (File No. S370-30) Printed in U.S.A. LY24-5195-1