File No. S360-31
Order No. GC28-6719-2

OS

IBM Systems Reference Library

IBM System/360 Operating System:

Service Aids

OS Release 21

This publication explains when, why, and how to use IBM service aids to diagnose and fix failures in system or application programs. Each service aid is described in a separate chapter. The service aids are:

- IFCDIP00 -- Initializes the SYS1.LOGREC data set.
- IFCEREP0 -- Summarizes and prints records from the SYS1.LOGREC data set.
- GTF (Generalized Trace Facility) -- Traces selected system events such as SVC and I/O interruptions.
- IMCJQDMP -- Operates as a stand-alone program to format and print the system job queue.
- IMBLIST -- Formats and prints object modules, load modules, and CSECT identification records.
- IMBMDMAP -- Maps load modules.
- IMCOSJQD -- Operates as a problem program to format and print the system job queue.
- IMDPRDMP -- Formats and prints dumps, TSO swap data set, and GTF trace data.
- IMAPTFLE -- Generates JCL needed to apply to a PTF and/or applies the PTF.
- IMDSADMP -- Operates as a stand-alone program to produce a high-speed or low-speed dump of main storage.
- IMASPZAP -- Verifies and/or replaces instructions and/or data in a load module.

Information about how to write EDIT user programs is provided in a separate appendix.

Third Edition (March, 1972)

This is a major revision of, and obsoletes, GC28-6719-1 and Technical Newsletter GN28-2478. Text changes and illustration changes in chapters containing few changes are indicated by vertical lines to the left of the changes. Consult the Summary of Amendments following the Contents Directory for information about which chapters are new and which are changed.

This edition applies to release 21, of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM Systems, consult the latest IBM System/360 and System/370 SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

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 readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602. Comments become the property of IBM.

Preface

This publication is for system programmers and IBM programming systems representatives. It explains when, why, and how to use IBM service aids to diagnose and fix failures in system or application programs.

Each service aid is described in a separate chapter. The chapters are arranged so that the corresponding index tabs will appear in alphabetical order. The index tabs show the names of the programs minus the three-character component identifier (such as IMC). The form of the name shown on the index tab also appears in the index to help you locate the chapter you want.

Some information about service aids is not included in this publication, but is covered in the following publications:

IBM System/360 Operating System:

- Service Aids Logic PLM, GY28-6721 -- describes the internal logic of the service aid programs (how they work).
- Programmer's Guide to Debugging, GC28-6670 -- describes the dump-type output of the service aids.
- Messages and Codes, GC28-6631 -- describes the numbered messages issued by the service aids.

You should also be familiar with the following publications:

IBM System/360 Operating System:

- Utilities, GC28-6586 -- describes how to use utility programs to print certain types of service aid output.
- Operator's Reference, GC28-6691 -- describes how to perform certain basic operations, such as loading a stand-alone program.
- Job Control Language Reference, GC28-6704 -- describes how to use job control statements to override default parameters, use cataloged procedures, allocate space for data sets, etc.

CONTENTS DIRECTORY

Introduction → INTRO

Explains the service aid concept; guides selection of a service aid; summarizes the ways to retrieve service aids.

Chapter 1: IFCDIP00 → DIP00

Initializes the SYS1.LOGREC data set.

Chapter 2: IFCEREP0 → EREP0

Summarizes and prints records from the SYS1.LOGREC data set.

Chapter 3: GTF (Generalized Trace Facility) → GTF

Traces selected system events such as SVC and I/O interruptions.

Chapter 4: IMCJQDMP → JQDMP

Operates as a stand-alone program to format and print the system job queue.

Chapter 5: IMBLIST → LIST

Formats and prints object modules, load modules, and CSECT identification records.

Chapter 6: IMBMDMAP → MDMAP

Maps load modules.

Chapter 7: IMCOSJQD → OSJQD

Operates as a problem program to format and print the system job queue.

Chapter 8: IMDPRDMP → PRDMP

Formats and prints dumps, TSO swap data set, and GTF trace data.

Chapter 9: IMAPTFLE → PTFLE

Generates JCL needed to apply a PTF and/or applies the PTF.

Chapter 10: IMDSADMP → SADMP

Operates as a stand-alone program to produce a high-speed or low-speed dump of main storage.

Chapter 11: IMASPZAP → SPZAP

Verifies and/or replaces instructions and/or data in a load module.

Appendix: Writing EDIT User Programs → APNDX

Tells how to write and use EDIT user programs.

Each chapter has its own Table of Contents.

Summary of Amendments for GC28-6719-2 OS Release 21

GENERAL COMMENTS

- Program Design information has been moved to a new publication, the Service Aids Logic PLM, GY28-6721.
- Information relating to the Primary Control Program (PCP) has been deleted.
- Chapter order has been revised to accomodate several new chapters.

INTRODUCTION

References to services aids as a SYSGEN option have been deleted.

CHAPTER 1: IFCDIP00

IFCDIP00 has been moved from the Utilities SRL and rewritten. All information concerning the PARM parameter of IFCDIP00 has been deleted.

CHAPTER 2: IFCEREP0

IFCEREP0 has been moved from the Utilities SRL and rewritten. All information concerning SDR records has been deleted. Information for the Reliability Data Extractor (RDE) and MES has been added.

CHAPTER 3: GTF (THE GENERALIZED TRACE FACILITY)

GTF is a new feature of the operating system that executes as a problem program and is invoked by the START command. It traces selected system events, such as IO interruptions, SIO operations, program interruptions, etc. A special feature of GTF, the GTRACE macro instruction, allows you to record user data in GTF's output buffers. GTF output can be printed and formatted using the EDIT function of IMDPRDMP, which is described in Chapter 8.

CHAPTER 4: IMCJQDMP

This chapter is essentially unchanged.

CHAPTER 5: IMBLIST

IMBLIST is a new service aid that formats and prints object modules and CSECT Identification Records (IDRs) and maps load modules. It assumes the function of IMAPTFLS, a service aid which is no longer documented in this publication.

CHAPTER 6: IMBMDMAP

This chapter is essentially unchanged.

CHAPTER 7: IMCOSJQD

IMCOSJQD is a new service aid that dumps the system job queue data set (SYS1.SYSJOBQE), or formats and prints selected records from it. IMCOSJQD is identical in function to the IMCJQDMP service aid, but IMCOSJQD executes as a problem program whereas IMCJQDMP is stand-alone.

CHAPTER 8: IMDPRDMP

IMDPRDMP now includes the EDIT function, which formats and prints GTF output. The parameters of the EDIT control statement, which invokes the EDIT function, allow you to select records to be formatted; some of the parameters are: JOBNAME= , IO= , EXT.

The EDIT function also provides interfaces for user-written exit routines and format appendages. Exit routines examine every trace record to determine how it should be handled. Format appendages format and print specific types of user records. Information about how to write exit routines and format appendages is provided in the Appendix: Writing EDIT User Programs.

CHAPTER 9: IMAPTFLE

IMAPTFLE now allows you to include a Linkage Editor IDENTIFY control statement in the IMAPTFLE input stream; this is required for the application function and optional for the generate function. The purpose of the IDENTIFY statement is to flag specific CSECTs that are to be updated with PTFs.

CHAPTER 10: IMDSADMP

This chapter is essentially unchanged.

CHAPTER 11: IMASPZAP

IMASPZAP now provides a control statement, IDRDATA, that allows you to update the CSECT Identification Record of any module that is successfully updated with a REP operation.

APENDIX: Writing EDIT User Programs

This appendix provides all the information you need to write an exit routine or a format appendage for use with the EDIT function of IMDPRDMP and the GTRACE macro instruction. It describes the interfaces with EDIT, illustrates the use of the IMDMEDIT mapping macro instruction, shows samples of both exit routines and format appendages, discusses ways to avoid unrecoverable errors, and describes how to debug an exit routine.

Please note that change bars are not used in any chapter described as "new" in this summary of amendments.

Summary of Amendments for GC28-6719-1 as Updated by GN28-2478 OS Release 20.1

Item	Description	Areas Affected
IMAPTFLE Improvements	Permits IMAPTFLE to apply a PTF to OS/360 directly.	235-241,243-245
2305 and 3330 Support	Permits service aids to be used with these devices.	188,207,251
Multiprocessing Support	Permits IMDSADMP to dump the contents of both CPUs of the IBM System/360 Model 65 Multiprocessing System.	159,166,173-177, 179,180,184,185, 187-189,191,194,195
IMBMDMAP Improvement	Message improvement.	318
IMAPTFLE Improvement	Message improvement -- blocksize error.	243

Summary of Amendments for GC28-6719-1 OS Release 20

Item	Description	Pages Affected
System Generation	Permits six service aids to be added to the operating system during system generation.	2,16,17,183,280
SVC Dump	Permits IMDPRDMP to format and print system dumps.	25,26,29,32,33,37-39
System/370	Permits service aids to be used with the IBM System/370.	173-175,178-182,185, 187-191,194,195,251
TSO	Permits IMDPRDMP to format and print TSO dumps and swap data sets.	1,11,13,26,28,31,33,34, 39,40,42,46,47,77,83, 134-155
IMDSADMP Improvement	The address of the input dump device can be specified from the operator console.	180,185
Restriction	Release 20 IMAPTFLE will not process a Stage I output tape from a release before release 19.	241,243
IMDPRDMP Program Design	The "IMDPRDMP Program Design" section of the "IMDPRDMP" chapter has been rewritten.	25,56-71
PRDMP	The PRDMP PROCLIB procedure for calling IMDPRDMP has been documented with examples.	43,44
IMDPRDMP Examples	Examples on how to use the IMDPRDMP control statements and PRDMP PROCLIB procedure are included in the "IMDPRDMP" chapter.	44-47
MFT QCB Trace	IMDPRDMP formats and prints QCB traces for MFT users.	28
ONGO Clarification	When the ONGO verb of an IMDPRDMP control statement has no parameters specified, the original GO parameters are restored: QCBTRACE, LPAMAP, FORMAT, and PRINT ALL.	39
65MP Clarification	Occasionally only one prefix is shown on an IMDPRDMP listing. This occurs when the dump is initiated on one CPU, interrupted and then dispatched to the other CPU.	41
Messages	All messages have been altered, where necessary, to agree with the publication <u>IBM System/360 Messages and Codes</u>, GC28-6631.	48-55,192,193,223-227, 242,243,266-268,285, 286,315-318

(Part 1 of 2)

Item	Description	Pages Affected
Module Name	For dumps that are formatted and printed by IMDPRDMP, the name of the module that invoked the dump is printed in the header of the dump listing.	77-79,82,83,102,120, 135-137,156-158,160
Output Comments	Within the IMDPRDMP formatted dump, a number of output comments may be printed to assist in reading and interpreting the dump. These comments are explained.	77,83,161-167
IMAPTFLE Region Size	IMAPTFLE requires a 46K region or partition.	236
MFT LPA Maps	MFT link pack area maps do not include resident SVC routines (IMBMDMAP).	314
Control Blocks	Various changes have been made to system control blocks that are formatted and printed by IMDPRDMP.	84,85,87,88,106,107
Tables and Examples	Table and example numbers have been converted to figure numbers. All figures have been renumbered. See the figure list in each chapter.	All

(Part 2 of 2)

Introduction

Service aids are programs designed to help system programmers and IBM programming system representatives diagnose and fix failures in system or application programs. Service aids have three general functions:

Information Gathering

- To dump main storage, use the stand-alone program IMDSADMP. Its output can be formatted and printed using IMDPRDMP.
- To trace system evenets such as SVC and I/O interruptions, use GTF (the Generalized Trace Facility). Its output can be formatted and printed using the EDIT function of IMDPRDMP.

Formatting and Printing: Mapping

- To summarize and print records in the SYS1.LOGREC data set, use IFCEREP0.
- To format and print load module, use IMBMDMAP or IMBLIST.
- To format and print object modules and CSECT identification records, use IMBLIST.
- To format and print the system job queue, use IMCJQDMP (stnad-alone) or IMCOSJQD (problem program).
- For format and print IMDSADMP output, other system dumps, TSO swap data sets, and GTF trace output, use IMDPRDMP.

Generating and Applying Fixes

- To apply a PTF, use IMDPTFLE.
- To verify and/or replace instructions and/or data in a load module, use IMASPZAP.
- To initialize the SYS1/LOGREC data set, use IFCDIP00.

For more detailed information about choosing a service aid, refer to the table in figure INTRO-1.

SYMPTOM	INFORMATION GATHERING		MAPPING, FORMATTING, AND PRINTING					PATCHING		
	IMDSADMP	GTF	IMDPRDMP	IMBLIST	IFCEREPO	IMBMDMAP	IMCOSJQD IMCJQDMP	IMAPTFLE	IMASPZAP	IFCDIP00
Warm Start Failure	1	-	1c,d,e	-	-	-	1	-	2	-
Scheduler ABEND	-	1	4	1,2	-	1,2,3	1	-	2	-
Writer ABEND	-	1	4	-	-	-	3	-	2	-
Problem Program ABEND	-	3	4a	2	-	2	-	-	2	-
Recursive ABEND	1	1	1a,1c-d,4	2	-	1,3	2	-	2	-
Disabled Loop	1	1	1c-e,4	-	-	1,3	-	-	-	-
Problem Program Loop	-	3	4a	2	-	2	-	-	-	-
Large Loop with I/O	1	1	1a,1c-e, 4b-d	-	-	1,3	-	-	2	-
DAR Loop	1	1	1c,1e,4	2	1a	1,2,3	-	-	-	1
Hard Wait	1	1	1c-e,4	1,2	1a	1,2,3	-	-	-	1
Enabled Wait	1	1	1b,4	2	1a	1,2,3	-	-	-	1
Reader/Interpreter Failure	-	-	-	-	-	1	1	-	2	-
I/O Failure (e.g. console)	1	2	1a-e,4b-d	-	1b,2	-	-	-	2,4	1
Allocation Failure	1	-	1b-d	2	-	2	-	-	2	-
Enqueued Job Lost	-	-	-	-	-	-	3	-	-	-
Chain Scheduling Problem	1	2	1a,1c-e, 4b-d	-	-	-	-	-	-	-
Access Method Failure	-	2	4	-	2	-	-	-	4	-
Data Mgt. Program Check	-	1,3	4	2	-	2,3	-	-	-	-
Module Level Unknown	-	-	-	3	-	-	-	-	3	-
User Modification Unknown	-	-	-	4	-	-	-	-	3	-
Applying PTF	-	-	-	-	-	-	-	1	1	-
Applying Local Fix	-	-	-	-	-	-	-	-	1	-
APAR Documentation	1	1,3	1a,1c-e,4	3	-	1,2,3	1	-	3	-
Print SYS1.DUMP	-	-	1b-d,2,4	-	-	-	-	-	-	-
TSO Failure	1	1	2,3,4	-	-	-	-	-	2	-
Capturing System Before Re-IPL	1	-	1a-e,4	-	-	-	1	-	-	-

INFORMATION GATHERING

IMDSADMP

1. Dumps the contents of main storage to a tape, which can be formatted and printed using PRDMP. (Note that IMDSADMP output may also be directed to a printer.)

GTF (Generalized Trace Facility)

1. Traces all system events.
2. Traces selected events, such as I/O interruptions, SIO operations, etc.
3. Traces user programs with GTRACE macro instruction.

MAPPING, FORMATTING AND PRINTING

IMDPRDMP

1. Formats and prints the following from SADMP high-speed output:
   a. Link Pack Area.
   b. Queue Control Block Trace.
   c. Major Control Blocks.
   d. Selected Areas of Main Storage.
   e. Operating System Nucleus.
2. Formats and prints TSO control blocks and main storage from a SYS1.DUMP data set.
3. Formats and prints TSO Swap data set(s).
4. Formats and prints selected records from the GTF Trace data set or from trace buffers in a SYS1.DUMP or SADMP output data set. Records are selected by keywords such as:
   a. JOBNAME.
   b. I/O.
   c. SVC.
   d. SIO.

IMBLIST

1. Lists specific object modules or load modules in a data set.
2. Maps control sections and overlay structure and lists cross-references within a load module.
3. Lists CSECT identification Records for specific load modules.
4. Lists translation data, linkage editor modification data, or SPZAP modifications to control sections in a load module.

IFCEREPO

Selects, formats, and prints error records in the SYS1.LOGREC data set.

1. Selects records by record type, such as:
   a. Machine check and/or inboard.
   b. Outboard.
2. Selects records by device type or device address.

IFCEREPO can also select records by model number or time of creation.

IMBMDMAP

1. Maps the operating system nucleus.
2. Maps a failing module.
3. Maps the link pack area.

IMCJQDMP (Stand-Alone) and IMCOSJQD (Problem Program)

1. Dumps entire SYS1.SYSJOBQE data set.
2. Selects, formats, and prints job queue records associated with a specific job.
3. Selects, formats, and prints job queue records associated with a specific work queue.

PATCHING

IMAPTFLE

1. Generates control statements and JCL needed to apply PTFs; the application function also invokes the linkage editor.

IMASPZAP

Verifies or replaces instructions or data in a load module stored on a direct access storage device.

1. Modifies code or data in a load module.
2. Sets traps by inserting invalid instructions or user-written SVCs.
3. Dumps load modules by CSECT to allow examination of the text.
4. Dumps selected data to verify the count, key, and contents of the data.

IFCDIP00

1. Reinitializes the SYS1.LOGREC data set if destroyed. (Can also be used to allocate more space to SYS1.LOGREC.)

Figure INTRO-1. Service Aids Symptom Table

Retrieving Service Aids

All service aids except IMDSADMP and IMCJQDMP execute as problem programs under the operating system. They are automatically transferred to SYS1.LINKLIB during system generation. IMDSADMP and IMCJQDMP are stand-alone programs that must be retrieved from the distribution library before they can be used.

Retrieving IMCJQDMP

IMCJQDMP resides as an object module in distribution library SYS1.DN554A. Before you can load it into the system as a stand-alone program, you must retrieve it from the distribution library. To do this you can either transfer the module onto punch cards using the IEBPTPCH utiltiy, or copy it to magnetic tape using the IEBGENER utility.

Retrieving IMDSADMP

IMDSADMP resides as a macro definition in distribution library SYS1.MACLIB. The easiest way to retrieve IMDSADMP is to specify the MACLIB macro isntruction at system generation; IMDSADMP will automatically be transferred to the SYS1.MACLIB data set in the operating system.

If you choose not to create a SYS1.MACLIB data set at system generation, you can retrieve IMDSADMP by three other methods:

- If you want to retrieve IMDSADMP and execute it all in the same step, you can treat the distribution library as a private macro library. Figure INTRO-2 shows the job control statements needed to do this.
- You can copy IMDSADMP from the distribution library into a private library.
- You can punch IMDSADMP from the distrubtion library onto cards using the IEBPTPCH utility.

```
//ASMSAD      JOB      MSGLEVEL=(1,1)
//STEP        EXEC     ASMFC
//ASM.SYSLIB  DD       DSN=SYS1.MACLIB,DISP=OLD
//ASM.SYSIN   DD       *
    IMDSADMP
    END
/*
```

Figure INTRO-2. Sample JCL Statements Need to Assemble IMDSADMP Directly from the Distribution Library

Chapter 1: IFCDIP00

Initializes the SYS1.LOGREC data set.

Contents

INTRODUCTION 21

INPUT TO IFCDIP00 22
The SYS.LOGREC Data Set 22
Job Control Statements 22

RUNNING IFCDIP00 23
Reinitializing SYS1.LOGREC 23
Changing Space Allocation for SYS1.LOGREC 23

DIP00

Figures

Figure DIP00-1. Reinitializing the SYS1.LOGREC Data Set 23
Figure DIP00-2. Changing the Space Allocation for SYS1.LOGREC 24

Introduction

IFCDIPOO is a service aid that runs under the IBM System/360 Operating System. IFCDIPOO has three applications:

1. Initializing the SYS1.LOGREC data set during system generation. This application is discussed in the publication <u>IBM System/360 Operatinq: System Generation.</u>

2. Reinitializing the SYS1.LOGREC data set. During processing, some types of errors may destroy the SYS1.LOGREC header and make the data set unusable; IFCDIPOO can then be used to reinitialize the SYS1.LOGREC data set.

3. Modifying the space allocation for the SYS1.LOGREC data set. In some situations, the SYS1.LOGREC data set may be too large or too small for the system using it; IFCDIPOO can then be used to increase or decrease the space allocation for SYS1.LOGREC.

Input to IFCDIP00

The input to IFCDIP00 consists of the SYS1.LOGREC data set and job control statements.

The SYS1.LOGREC Data Set

The SYS1.LOGREC data set consists of a header record followed by environment records.

The header record is created by IFCDIP00; it keeps track of the number and location of the environment records.

The environment records are generated by the outboard recording routine (OBR), the miscellaneous data recorder (MDR), the recovery management routines MCH and CCH, and the reliability data extractor program RDE, and the environment recording routines SER0 and SER1. Each record reflects the condition that prevailed in the system when an error occurred.

Job Control Statements

IFCDIP00 is run and controlled by job control statements; no user or utility control statements are needed.

Running IFCDIP00

You run IFCDIP00 by providing job control language procedures to reinitialize and reallocate the SYS1.LOGREC data set. The following sections contain detailed examples of reinitializing and reallocating SYS1.LOGREC.

Reinitializing SYS1.LOGREC

Figure DIP00-1 is an example of the job control statements needed to reinitialize the SYS1.LOGREC data set using IFCDIP00.

```
//INSERLOG    JOB
//            EXEC      PGM=IFCDIP00
//SERERDS     DD        DSNAME=SYS1.LOGREC,UNIT=2311,DISP=(OLD,KEEP),
//                      VOL=SER=111111
```

Figure DIP00-1. Reinitializing the SYS1.LOGREC Data Set

Control Statements for Figure 1

The JOB statement initiates the job; the job name INSERLOG has no signifigance.

The EXEC statement specifies the program name (PGM=IFCDIP00).

The SERERDS DD statements specifies the output (SYS1.LOGREC) data set; the DSNAME must be SERERDS.

Changing Space Allocation for SYS1.LOGREC

IFCDIP00 may be used in conjunction with the IEHPROGM utility to increase or decrease the space allocated for the SYS1.LOGREC data set. First the SYS1.LOGREC data set is scratched and uncataloged, using IEHPROGM; then, using IFCDIP00, the data set is reallocated with increased or decreased space specifications; and, finally, the newly allocated data set is reinitialized.

If you use the preceding procedure and an error occurs after the SYS1.LOGREC data set has been scratched, but before it has been reallocated, the IFCDIP00 job will be terminated and the system will be marked ineligible for IPL procedures. To solve this problem, do one of the following:

- Use the IBCDMPRS utility to restore the system and thereby restore the SYS1.LOGREC data set. After the SYS1.LOGREC data set has been restored, you can reinitialize the system and reallocate SYS1.LOGREC.
- Execute the reallocate operation on another IBM System/360 Operating System, if one is available.

Figure DIP00-2 is an example of reallocating the SYS1.LOGREC data set.

```
//RELGREC      JOB
//SCR          EXEC      PGM=IEHPROGM
//DD1          DD        UNIT=2311,VOLUME=SER=111111,DISP=OLD
//SYSIN        DD        *
    SCRATCH              DSNAME=SYS1.LOGREC,VOL=2311=111111
    UNCATLG              DSNAME=SYS1.LOGREC
/*
//R            EXEC      PGM=IFCDIP00
//SERERDS      DD        DSNAME=SYS1.LOGREC,UNIT=2311,DISP=(NEW,CATLG),
//                       VOL=SER=111111,SPACE=(allocation,CONTIG)
```

Figure DIP00-2. Changing the Space Allocation for SYS1.LOGREC

Chapter 2: IFCEREP0

Summarizes and prints records from the SYS1.LOGREC data set. →

EREP0

Contents

INTRODUCTION 29
Editing and Writing Selected Records 29
Accumulating Selected Records 29
Summarizing Selected Records 30
Processing Records Produced on Different Machine Models 30

INPUT TO IFCEREP0 31
Environment Records 31

RUNNING AND CONTROLLING IFCEREP0 32
Job Control Statements 32
Keyword Parameters for IFCEREP0 33

IFCEREP0 EXAMPLES 36
Example 1: Printing Machine Check Records 36
Example 2: Writing Machine Check Records onto a 7-Track Magnetic Tape 37
Example 3: Printing and Accumulating Machine Check and Channel Inboard Records 38
Example 4: Printing and Accumulating Machine Check Records Contained in a History Data Set 39
Example 5: Printing Recently Generated Machine Check Records and Accumulated Machine Check Records 40

IFCEREP0 OUTPUT 41
Format of Edited Records 41

EREP0

Figures

Figure EREP0-1. Job Control Statements 32
Figure EREP0-2. Output Record Printout Structure 41
Figure EREP0-3. Sample Printout -- Outboard Data Editing and Printing Section 44
Figure EREP0-4. First Sample Printout -- TCAM Data Editing and Printing Section 45
Figure EREP0-5. Second Sample Printout -- TCAM Data Editing and Printing Section 45
Figure EREP0-6. Machine-Check Summary 46
Figure EREP0-7. Channel Inboard Summary 47
Figure EREP0-8. I/O Outboard Summary 48
Figure EREP0-9. TCAM I/O Outboard Summary 49

Introduction

IFCEREPO is a service aid that runs under the IBM System/360 Operating System. You can use IFCEREPO to:

- Select and format environment records from the SYS1.LOGREC data set and write them to an output device. The environment records on the SYS1.LOGREC data set are generated by the error environment recording programs OBR, SER0, SER1, MDR, by the recovery management programs CCH, and MCH, and by the reliability data extractor program RDE.

- Select environment records from the SYS1.LOGREC data set and accumulate them on a history data set.

- Write the records accumulated on the history data set to an output device.

- Summarize the information contained in the records on the SYS1.LOGREC data set or the history data set.

- Process (edit, write, accumulate, and summarize) records produced on different machine models.

Editing and Writing Selected Records

You can use IFCEREPO to retrieve selected environment records from the SYS1.LOGREC data set or a history data set, edit them, and write them to an output device. After the record is written to the output device, it is cleared to hexadecimal zeros on the SYS1.LOGREC data set unless you specify otherwise. If the input data set is the history data set, the records remain unchanged. The selection of records that IFCEREPO will process is based on the following factors:

- Record type: you can specify any type of environment record, or any combination of types.

- Model number: you can specify the model number of any computing system that is writing records on the SYS1.LOGREC data set; this specification is useful when several computing systems are writing records on the same SYS.LOGREC data set.

- Time period: you can specify that IFCEREPO only process records that were generated on certain dates.

- Devices: you can specify that IFCEREPO process records that are related to a specific device or device type.

Accumulating Selected Records

You can use IFCEREPO to move selected environment records from the SYS1.LOGREC data set to a history data set; this enables you to accumulate specific types of environment records on different volumes or on the same volume. When you move an environment record from the SYS1.LOGREC data set to the history data set, the environment record or the SYS1.LOGREC data set, is cleared to hexadecimal zeros unless you specify otherwise.

Summarizing Selected Records

You can use IFCEREPO to extract pertinent data from selected records and print the data in the form of a summary. The contents of the summary depend on the type of error you monitor.

Processing Records Produced on Different Machine Models

You can use IFCEREPO to edit, write, summarize, and accumulate environment records for any IBM System/360 or IBM System/370 model that supports the IBM System/360 Operating System. In addition, any SYS1.LOGREC data set or history data set generated on one system can be printed on another system.

Input to IFCEREPO

The input to IFCEREP0 consists of environment records located on the SYS1.LOGREC or history data sets.

Environment Records

You can use IFCEREP0 to process six types of environment records:

1. Machine check records - which are produced and stored in SYS1.LOGREC by the system environment recorders SER0 and SER1, and by the machine check handler (MCH). They record machine check interruptions caused by malfunctions in the central processing unit.

2. Inboard records - which are produced and stored by SER0, SER1, and by the channel check handler (CCH). They record input/output interruptions caused by specific channel failures.

3. Outboard records - which are produced and stored by the outboard recorder. They record permanent errors on input/output devices, and terminal statistics and errors for TCAM.

4. Miscellaneous data records - which are produced and stored by the miscellaneous data recorder (MDR). They record errors that are not reflected in any other record type.

5. System initialization (IPL) records - which are produced and stored in the SYS1.LOGREC data set by the reliability data extractor (RDE) programs. They record information related to each system initialization.

6. System termination (EOD) records - which are produced and stored in the SYS1.LOGREC data set by the reliability data extractor (RDE) programs. They record information related to each system termination. For a complete explanation of RDE see the publication, IBM System/360 Operating System: RDE Guide, GC28-6741.

Running and Controlling IFCEREP0

You run and control IFCEREP0 by job control statements and by specifying keyword parameters on the EXEC statement of your IFCEREP0 procedure; no user or utility control statements are needed.

Job Control Statements

IFCEREP0 Figure 1 shows the job control statements necessary for running IFCEREP0.

Statement	Usage
JOB Statement	This statement initiates the job.
EXEC Statement	This statement specifies the program name and keyword parameters necessary to control the function of the program.
SERLOG DD Statement	This statement defines the input data set as being the SYS1.LOGREC data set. Either a SERLOG DD statement or the ACCIN DD statement must be included for each application of the IFCEREP0 program.
ACCIN DD	This statement defines the input data set as being a history data set. Either an ACCIN DD statement or the SERLOG DD statement must be included for each application of the IFCEREP0 program.
EREPPT DD Statement	This statement defines the edited output data set. It must be included with each application of the program.
ACCDEV DD Statement	This statement defines an accumulated output data set. The accumulated data set can reside or magnetic tape or a direct access device. Space must be allocated for a new output data set that is to reside on a direct access volume. Space cannot be allocated for an existing output data set.
Notes:	The SERLOG, ACCIN, EREPPT, and ACCDEV DD statements define sequential data sets. If records produced on different machine models are to be processed, a JOBLIB DD statement is required to define the original system's link library.

Figure EREP0-1. Job Control Statements

Keyword Parameters for IFCEREPO

You can specify the following keyword parameters to control the functions of the IFCEREPO program.

```
PARM=  ⎧[TYPE [M][C][O][T][I][E],]                       ⎫
       ⎪[MOD=(nnn[,nnn...]),]                             ⎪
       ⎪      ⎧N⎫                                         ⎪
       ⎪ MES= ⎩Y⎭                                         ⎪
       ⎪[VOLID=(VOLID1,VOLID2,VOLID3,VOLID4)]             ⎪
       ⎪[CUA=(CUU[,CUU]),]                                ⎪
       ⎪[DEV=NNNN,]                                       ⎪
       ⎪[DATE=([YYDDD][,YYDDD]),]                         ⎪
       ⎪[      ⎧N⎫  ]                                     ⎪
       ⎪[ZERO= ⎩Y⎭, ]                                     ⎪
       ⎨[       ⎧PS⎫  ]                                   ⎬
       ⎪[PRINT= ⎨PT⎬, ]                                   ⎪
       ⎪[       ⎪SU⎪  ]                                   ⎪
       ⎪[       ⎩NO⎭  ]                                   ⎪
       ⎪[     ⎧Y⎫  ]                                      ⎪
       ⎪[ACC= ⎩N⎭, ]                                      ⎪
       ⎪[      ⎧N⎫  ]                                     ⎪
       ⎪[HIST= ⎩Y⎭, ]                                     ⎪
       ⎪[TERMN=1-8 chars,]                                ⎪
       ⎪[     ⎧1⎫  ]                                      ⎪
       ⎪[M67= ⎩2⎭, ]                                      ⎪
       ⎪[        ⎧N⎫  ]                                   ⎪
       ⎩[RDESUM= ⎩Y⎭, ]                                   ⎭
```

TYPE

specifies the type of records to be processed.

Code	Meaning
M	Machine-check records
C	Channel inboard records
O	I/O outboard records
T	T-type records
I	IPL Records
E	EOD Records

A combination of records can be specified. For example, PARM=(TYPE=MC,....). If no record type is specified, all record types are processed.

MOD

indicates that all records created on the model or models specified are to be processed. The operand is to be right justified and may be up to three digits in length.

MES

indicates that error statistics for specific volume/serials are to be summarized and printed. This parameter is valid only for the 3410 and 3420 tape subsystems, when "TYPE=O' is coded, or when no record type is specified.

VOLID

indicates specific volumes for error statistics (MES) processing. A maximum of four volumes can be specified. If this parameter is not coded and MES=Y is coded, all volumes will be processed.

If no model numbers are specified all models are accepted for processing.

CUA (maximum of two)

indicates that the selected record types that are related to the specific channel(s) and unit(s) are to be processed.

DEV (maximum of one)

indicates that selected record types that are related to a specific device type are to be processed.

if DEV is not specified, all selected records (as specified in the TYPE subparameter) are processed regardless of the device type.

if DEV =3410 or DEV =3420 is specified, both devices will be included in the report.

DATE (maximum of one set)

indicates that all of the selected record types generated within a specific period of calendar time are to be processed. The date is written yyddd yyddd where yyddd represents the year and the day (of the year) when the time period begins and yyddd represents the year and the day when the period ends.

If no date is specified, all selected records are processed regardless of when they were generated.

ZERO

indicates whether input records in the SYS1.LOGREC data set are to be cleared with hexadecimal zeros after they are processed. Records are not cleared to zeros in the history data set.

Note: It is possible to use the same operating system on several machines. Before moving the system packs to another machine, the operator must use the EREP program to copy the SYS1.LOGREC data set to tape so that the environmental data can later be related to the system that generated it.

PRINT

indicates how records are to be processed and written.

Code	Meaning
SU	Suppress full printing (print summary only).
PT	Suppress summary printing (print full record only).
NO	Suppress full printing and summary printing.
PS	Print full record and summary.

ACC

indicates whether selected records are to be accumulated in a history data set. If ACC=Y is coded; ZERO=Y must be coded if the input data set is SYS1.LOGREC.

HIST

indicates whether the input data set is a history data set. If HIST=Y is coded, the input data set must be defined with an ACCIN DD statement.

If HIST is not coded HIST=N is assumed and the input data set will be the SYS1.LOGREC data set.

TERMN

indicates the OBR and TCAM records are to be selected by terminal name. Up to eight characters may be specified.

If TERMN is not coded all terminal names are selected.

M67

indicates which Model 67 records are to be processed.

If M67 is not coded mod 1 Model 67 records are processed.

RDESUM

indicates that the IFCEREPO summary function for RDE records is to be run. The summary function produces an IPL report and a hardware error report. This parameter can be coded only if RDE has been selected during system generation. For a complete explanation of RDE see the publication IBM System/360 Operating System: RDE Guide, GC28-6741.

IFCEREPO Examples

The following examples show some of the typical uses of the IFCEREPO program.

Example 1: Printing Machine Check Records

In this example:

- Machine check records are printed in a full record format.
- The records on SYS1.LOGREC are zeroed.

```
//JOBA          JOB
//              EXEC PGM=IFCEREPO,PARM='TYPE=M,ZERO=Y,PRINT=PT,ACC,=N'
//SERLOG        DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT        DD   SYSOUT=A
```

Control Statements for Example 1

The EXEC statement specifies (1) that machine check records are to be processed, (2) the type of printout (full record), (3) no accumulation is to take place.

The SERLOG DD statement defines the input (SYS1.LOGREC) data set.

The EREPPT DD defines the edited output data set (printer assumed).

Example 2: Writing Machine Check Records onto a 7-Track Magnetic Tape

In this example:

- Date-dependent machine check records are written in full record and summary formats onto a 7-track magnetic tape at a density of 200 bits per inch.
- The SYS1.LOGREC data set is zeroed.

```
//JOB         JOBA
//            EXEC PGM=IFCEREP0,PARM='TYPE=M,DATE=(62110,62117),
                   ZERO=Y,PRINT=PS,ACC=N'
//SERLOG      DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT      DD   DSNAME=ERRDATA,UNIT=2400-2,LABEL=(,NL),
                   DCB=(DEN=0,TRTCH=C),DISP=(NEW,CATLG)
/*
```

Control Statements for Example 2

The EXEC statement specifies (1) that machine check records are to be processed, (2) the type of printout (full record and summary), (3) the applicable time period, and (4) that no accumulation is to take place.

The SERLOG DD statement defines the input (SYS1.LOGREC) data set.

The EREPPT DD statement defines the output data set.

Example 3: Printing and Accumulating Machine Check and Channel Inboard Records

In this example:

- Machine check and channel inboard records are printed in a full record and summary format.

- Machine check and channel inboard records are accumulated on a history data set.

- The records on SYS1.LOGREC are zeroed.

```
//JOB          JOBA
//             EXEC PGM=IFCEREP0,PARM='TYPE=MC,ACC,Y,PRINT=PS,ZERO=Y'
//SERLOG       DD   DSNAME=SYS1.LOGREC,DISP=(OLD,KEEP)
//EREPPT       DD   SYSOUT=A
//ACCDEV       DD   DSNAME=ACUMSET,UNIT=2311,DISP=(NEW,CATLG),
                    VOLUME=SER=111112,SPACE=(TRK,(40,10))
/*
```

Control Statements for Example 3

The EXEC statement specifies (1) that machine check and channel inboard records are to be processed, (2) the type of printout (full record and summary), and (3) accumulation on a history data set.

The SERLOG DD statement defines the input (SYS1.LOGREC) data set.

The EREPPT DD statement defines the output data set.

The ACCDEV DD statement defines the accumulated (history) output data set. The set is cataloged for ease of retrieval.

Example 4: Printing and Accumulating Machine Check Records Contained in a History Data Set

In this example:

- Machine check records in the history data set are printed in a full record format.
- Machine check records in the history data set are moved to a second (output) history data set.

```
//JOB     JOBA
//        EXEC      PGM=IFCEREP0,PARM='TYPE=M,HIST=Y,PRINT=PT,ACC=Y'
//ACCIN   DD        DSNAME=HISTRYIN,DISP=(OLD,CATLG)
//EREPPT  DD        SYSOUT=A
//ACCDEV  DD        DSNAME=EXISTACC,DISP=(MOD,CATLG)
/*
```

Control Statements for Example 4

The EXEC statement specifies (1) that machine check records are to be processed, (2) a history data set is the input data set, (3) the type of printout full record, and (4) accumulation.

The ACCIN DD statement defines the input (history) data set.

The EREPPT DD statement defines the output data set.

The ACCDEV DD statement defines the accumulated (history) output data set.

Example 5: Printing Recently Generated Machine Check Records and Accumulated Machine Check Records

This example is a two-step job. Together the job steps produce a printout of machine check records from the SYS1.LOGREC data set and machine check records from a history data set.

In the first job step (STEP1):

- Machine check records on SYS1.LOGREC are edited and printed in a full record format.
- Machine check records on SYS1.LOGREC are accumulated on a history data set.
- The records on SYS1.LOGREC are zeroed.

In the second job step (STEP2):

- Machine check records in the history data set, updated in STEP1, are printed in a full record format.

```
//JOBA   JOB
//STEP1  EXEC      PGM=IFCEREP0,PARM='TYPE=M,PRINT=PT,ACC=Y,ZERO=Y'
//SERLOG DD        DSNAME=SYS1.LOGREC,DISP=(OLD,CATLG)
//EREPPT DD        SYSOUT=A
//ACCDEV DD        DSNAME=HISTORY,DISP=(MOD,CATLG)
/*
//STEP2  EXEC      PGM=IFCEREP0,PARM='TYPE=M,PRINT=PT,HIST=Y,ACC=N'
//ACCIN  DD        DSNAME=HISTORY,DISP=(OLD,CATLG)
//EREPPT DD        SYSOUT=A
/*
```

Machine Records (for comparison)

Control Statements for Example 5

STEP1

The EXEC statement specifies (1) that machine check records are to be processed, (2) the type of printout, and (3) accumulation.

The SERLOG DD statement defines the input (SYS1.LOGREC) data set.

The EREPPT DD statement defines the output data set.

The ACCDEV DD statement defines the accumulation (history) data set.

STEP2

The EXEC statement specifies (1) that machine check records are to be processed, (2) a history data set is the input data set, (3) the type of printout (full record), and (4) no accumulation.

The ACCIN DD statement defines the input (history) data set.

The EREPPT DD statement defines the output data set.

IFCEREPO Output

You can use IFCEREP0 to write output to any output device supported by the basic sequential access method (BSAM). The output is written as 120-byte records with a control character as the first character of each record. After the records are written to the output device, they are normally cleared to hexadecimal zeros in the SYS1.LOGREC data set; the space occupied by the cleared records cannot be reused until the entire SYS1.LOGREC data set is cleared. You can, however, specify that the records remain uncleared in your procedure for running IFCEREP0.

Format of Edited Records

Figure EREP0-2 shows the printed format of an edited output record.

```
                                  Program heading

   Program section

   Model

   Source                         Record type

   Record data

   Additional data
```

Figure EREP0-2. Output Record Printout Structure

Program heading
identifies the IFCEREP0 program on the first page of the listing:

- ENVIRONMENT RECORD EDITING AND PRINTING PROGRAM

Program section
identifies the program section that is generating the printout. Valid program sections are:

- CPU (MC) DATA EDITING AND PRINTING SECTION
- INBOARD DATA EDITING AND PRINTING SECTION
- OUTBOARD DATA EDITING AND PRINTING SECTION
- MDR DATA EDITING AND PRINTING SECTION
- TCAM OUTBOARD DATA EDITING AND PRINTING SECTION

Model

identifies the IBM System/360 Model or System/370 for which the printout is applicable. Valid entries are:

- Model 40, 50, 65, 67, 75, 85, 91, 95, 135, 145, 155, 165, or 195 for machine-check records.
- Model 40, 50, 65, 75, 85, 91, 95, 135, 145, 155, 165, or 195 for channel inboard records. (Model 67 and 95 channel inboard records appear as Model 65 and 91 records, respectively.)
- Universal for I/O outboard printouts produced by Model 30, 40, 50, 65, 67, 75, 85, 91, 95, 135, 145, 155, 165, or 195.

Note: SER can produce channel inboard records on any of the SER supported models. CCH can produce channel inboard records on IBM Systems/360 Models 65, 75, 85, 91, and 195, and IBM System/370 Models 135, 145, 155 and 165. The channel recording facilities of some MCH programs can produce channel inboard records on these models when CCH is not in the system or when the channel error cannot be recorded by CCH.

Source

identifies the error environment or recovery management program that generated the record placed in the SYS1.LOGREC data set. valid sources are:

- RECORD ENTRY SOURCE - OBR
- RECORD ENTRY SOURCE - SER0
- RECORD ENTRY SOURCE - SER1
- REcORD ENTRY SOURCE - MCH
- RECORD ENTRY SOURCE - CCH
- RECORD ENTRY SOURCE - MDR

Record type

indicates the type of printout. Valid types are:

- TYPE - CPU
- TYPE - INBOARD
- TYPE - OUTBOARD
- TYPE - MDR

Record data

is a listing of the edited record from the input data set. This data, which constitutes the bulk of the printout, is the programming data and machine data collected at the time of the error.

Additional data

is a listing of records that were recorded in the SYS.LOGREC data set while the program was being executed.

The heading:

- THE FOLLOWING RECORDS WERE GENERATED WHILE EXECUTING EREP

is followed by a printout of the records.

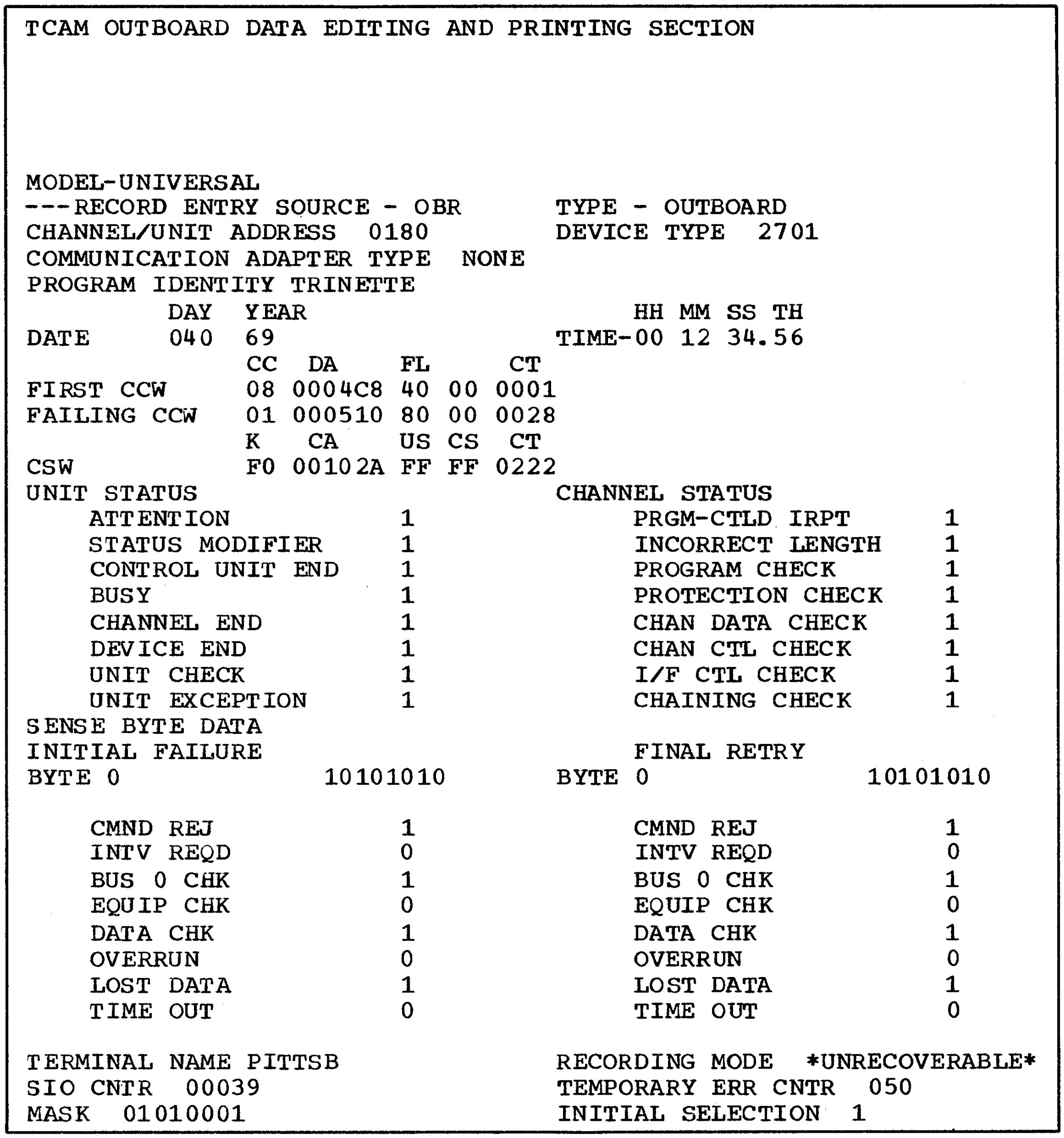
Figure EREP0-3 shows a sample outboard printout of an environment record that was processed by the outboard data editing and printing section of the utility program. The record was generated by the OBR program on an IBM System/360 Model 30, 40, 50, 65, 67, 75, 85, 91, 95, or 195 and on an IBM System/370 Model 135, 145, 155, or 165 (indicated by UNIVERSAL in the printout). The device falure occurred on a 2311 disk with a channel and unit address of 190.

Figures EREP0-4 and EREP0-5 show samples of a TCAM outboard printout of an environment record that was processed by the TCAM outboard data editing and printing section. The record was generated by the OBR program on an IBM System/360 Model 30, 40, 50, 65, 67, 85, 91, or 195, and on an IBM System/370 Model 135, 145, 155, or 165 (indicated by UNIVERSAL in the printout).

Note: The format for the MDR record is variable and requires special editing modules from the specific sub-types. Because of this variation, no sample printouts are shown for MDR record editing.

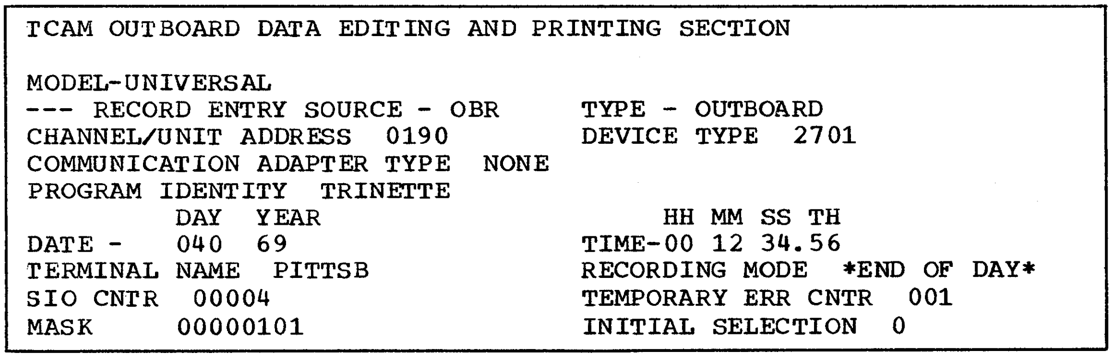
```
---RECORD ENTRY TYPE - UNIT CHECK      SOURCE- OUTBOARD               MODEL- xxx   SERIAL NO. xxxxxx
   OS  RELEASE xxx
                              DAY YEAR           HH MM SS.TH        JOB IDENTITY xxxxxxxx
                        DATE- xxx  xx      TIME- xx xx xx xx        xxxxxxxxxxxxxxxx

OBR RECORD CONVERTED TO THE STANDARD FORMAT                         MULTIPROCESSOR - CPU xx

 DEVICE TYPE                     xxxx
 PRIMARY   CHANNEL UNIT ADDRESS  xxxx
 ALTERNATE CHANNEL UNIT ADDRESS  xxxx
 PHYSICAL  DRIVE                 x
 PHYSICAL  CONTROL UNIT          x
 VOLUME LABEL                    xxxxxx

                      CC   CA   FL     CT                   K    CA   US CS  CT
    FAILING  CCW      xx xxxxxx xx xx xxxx             CSW  xx xxxxxx xx xx xxxx

                      M  B B  C C  H H  R
    LAST SEEK ADDRESS x  xxxx xxxx xxxx xx

    UNIT STATUS                 CHANNEL STATUS              STATISTICAL DATA            STATISTICAL DATA

  yyyyyy yyyy      x          yyyyyyyyy yyyyyy x          yyyy yyyy yyyyyy xxxx       yyyy yyyy        xxx
  yyyy yyyy yyyyyy x          yyyy yyyy yyy    x          yyyyyyy          xxxx       yyyyyy yyy yyy   xxx
  yyyyyy yyyy      x          yyyyy yyyy yyy   x          yyyy yyy yyyy    xxxx       yy yyy           xxx
  yyyy yyy         x          yyyyyyyyyyyyyyyy x          yyyyyy yyyyyyy   xxxx       yyyy yyyy yyy yy xxx
  yyyyyyyy yyyyyyy x          yyy yyy          x          yyy yyy          xxxx       yyyyy yyy        xxx
  yyyy yyy         x          yyyy yyy         x          yyyy yyy         xxxx       yyyy             xxx
  yyyyyy           x          yyyyyy yyy yyy   x          yyyyyyyyyy yyyyy xxxx       yyy yyy yyy yyy  xxx
  yyyyyyyy yyy     x          yyyyy            x          yyy  yyy         xxxx       yy yyyyyy        xxx

 SENSE BYTE DATA

  BYTE 0   xx    BYTE 1   xx    BYTE 2   xx    BYTE 3   xx     BYTE 4   xx   BYTE 5   xx

yyy yyyyyyy x  yyy yyy yyy x   xxxxxxxx      yyyyy yy yy x   yy yyy yyyy x  xxxxxxxx
yyyy yyy    x  yyyy yyy    x                             x   yyy yyy yyy x
yyyyy yyy   x  yyy yyy yyy x                 yy yyyy     x   yyy yy      x
yyyyyy yyy  x  yyyyyyy     x                 yyy yyy yyy x   yyyy yyy    x
yyyy yyy yy x  yyyyyyy yy  x                             x   yyyyyy yyy  x
yyy yyy yyy x  yyy  yyy    x                 yyy yyy     x   yyy yyy     x
yyyy yyy    x  yyy yyy yyy x                 yyyyy yyy   x               x
yyyyy yyy   x  yyyy yyyyyy x                 yyyyyy yyyy x   yyyy yyy    x
```

Figure EREP0-3. Sample Printout -- Outboard Data Editing and Printing Section



```
TCAM OUTBOARD DATA EDITING AND PRINTING SECTION




MODEL-UNIVERSAL
---RECORD ENTRY SOURCE - OBR      TYPE - OUTBOARD
CHANNEL/UNIT ADDRESS  0180        DEVICE TYPE  2701
COMMUNICATION ADAPTER TYPE  NONE
PROGRAM IDENTITY TRINETTE
         DAY  YEAR                      HH MM SS TH
DATE     040  69                  TIME-00 12 34.56
              CC  DA     FL     CT
FIRST CCW     08 0004C8 40 00 0001
FAILING CCW   01 000510 80 00 0028
              K   CA     US CS  CT
CSW           F0 00102A FF FF 0222
UNIT STATUS                       CHANNEL STATUS
    ATTENTION           1              PRGM-CTLD IRPT      1
    STATUS MODIFIER     1              INCORRECT LENGTH    1
    CONTROL UNIT END    1              PROGRAM CHECK       1
    BUSY                1              PROTECTION CHECK    1
    CHANNEL END         1              CHAN DATA CHECK     1
    DEVICE END          1              CHAN CTL CHECK      1
    UNIT CHECK          1              I/F CTL CHECK       1
    UNIT EXCEPTION      1              CHAINING CHECK      1
SENSE BYTE DATA
INITIAL FAILURE                        FINAL RETRY
BYTE 0              10101010      BYTE 0              10101010

    CMND REJ            1              CMND REJ            1
    INTV REQD           0              INTV REQD           0
    BUS 0 CHK           1              BUS 0 CHK           1
    EQUIP CHK           0              EQUIP CHK           0
    DATA CHK            1              DATA CHK            1
    OVERRUN             0              OVERRUN             0
    LOST DATA           1              LOST DATA           1
    TIME OUT            0              TIME OUT            0

TERMINAL NAME PITTSB              RECORDING MODE  *UNRECOVERABLE*
SIO CNTR  00039                   TEMPORARY ERR CNTR  050
MASK  01010001                    INITIAL SELECTION  1
```

Figure EREP0-4. First Sample Printout -- TCAM Data Editing and Printing Section



```
TCAM OUTBOARD DATA EDITING AND PRINTING SECTION

MODEL-UNIVERSAL
--- RECORD ENTRY SOURCE - OBR     TYPE - OUTBOARD
CHANNEL/UNIT ADDRESS  0190        DEVICE TYPE  2701
COMMUNICATION ADAPTER TYPE  NONE
PROGRAM IDENTITY  TRINETTE
         DAY  YEAR                     HH MM SS TH
DATE -   040  69                  TIME-00 12 34.56
TERMINAL NAME  PITTSB             RECORDING MODE  *END OF DAY*
SIO CNTR  00004                   TEMPORARY ERR CNTR  001
MASK     00000101                 INITIAL SELECTION  0
```

Figure EREP0-5. Second Sample Printout -- TCAM Data Editing and Printing Section

Machine-Check Summary: A machine-check summary can be generated on IBM Sytem/360 Models 40, 50, 65, 67, 85, 91, 95, and 195, and on IBM System/370 Model 135, 145, 155, 165. A summary consists of:

- Items that provide clues as to the type of machine malfunction.

- Parity information for registers in the diagnostic scan-out area (logout area), general purpose registers, and floating point registers.

- The status of binary triggers recorded in the logout area.

Notes: For the model 85, only the error triggers are summarized.

Figure EREP0-6 shows the format of a machine-check summary. Each summarized item is listed with its frequence of occurrence.

```
***MOD xx MACHINE-CHECK SUMMARY ***
NUMBER OF RECORDS EXAMINED = 10

     TITLE         TOTAL
ROBAR SUMMARY (UP TO FIRST 10)
     0AAAA         3
     1BBBB         4
     1CCCC         3

LOGOUT REG PARITY CHECK SUMMARY
     REG A         5
     REG B         2
     REG C         3

CHECKS AND INDICATORS SUMMARY
     ROAR CHECK    1
     LSAR PTY CHECK 3
     H DECODE CHECK 4
     D/Y8 CHECK    2
```

Figure EREP0-6. Machine-Check Summary

Channel Inboard Summary: A channel inboard summary can be generated on IBM System/360 models 40, 50, 65, 75, 85, 91, and 195 and IBM System/370 Model 165. (Model 67 and Model 95 channel inboard summaries are identified as Model 65 and 91 summaries, respectively.) Channel inboard records are summarized according to channel address. Each channel summary contains:

- The addresses of devices connected to the channel (a maximum of 10 devices).

- The status of hardware elements (pertaining to the channel) in the logout area.

- A summary of failing CCW command codes (a maximum of 24 entries). (The 24th CCW command code entry is a logical OR of the remainder of the failing command codes, if any.)

Figure EREP0-7 shows the format of channel inboard summary. Each summarized item is listed with its frequence of occurrence.

```
***MOD xx CHANNEL 1 SUMMARY ***
TOTAL NO. OF RECORDS FOR THE CHANNEL = 20

     TITLE           TOTAL
SUMMARY OF DEVICE ADDRESSES
     (MAX 10 ENTRIES)
     180             5
     190             6
     1F0             5
     UNDET.          4

SUMMARY OF CMND CODES
     (MAX 24 ENTRIES)
     CMND CODES      TOTAL
     '01'            7
     '02'            6
     '03'            3
     '14'            4

SUMMARY OF HARDWARE LOGOUT
     IF PARITY       8
     LWR WR          6
     IF TAG CHK      2
     WO PARITY CHK   4
```

Figure EREP0-7. Channel Inboard Summary

<u>I/O Outboard Summary</u>: An I/O outboard summary can be generated on IBM System/360 Models 30, 40, 50, 67, 75, 85, 91, 95, and 195, and IBM System/370 Models 135, 145, 155 and 165. I/O outboard summaries are organized according to device address; however, the order of appearance of the summaries is determined by the order in which device addresses are encountered in the OBR records selected for summarization. Where TCAM is used the summary will appear in CUA (channel unit address) and line (terminal name) sequence. Each I/O outboard summary contains:

- Volume labels (a maximum of 10 entries).
- A summary of failing CCW command codes (a maximum of 24 entries). (The 24th CCW command code entry is a logical OR of the remainder of the failing command codes, if any.)
- The sense bits (a maximum of 6 bytes)

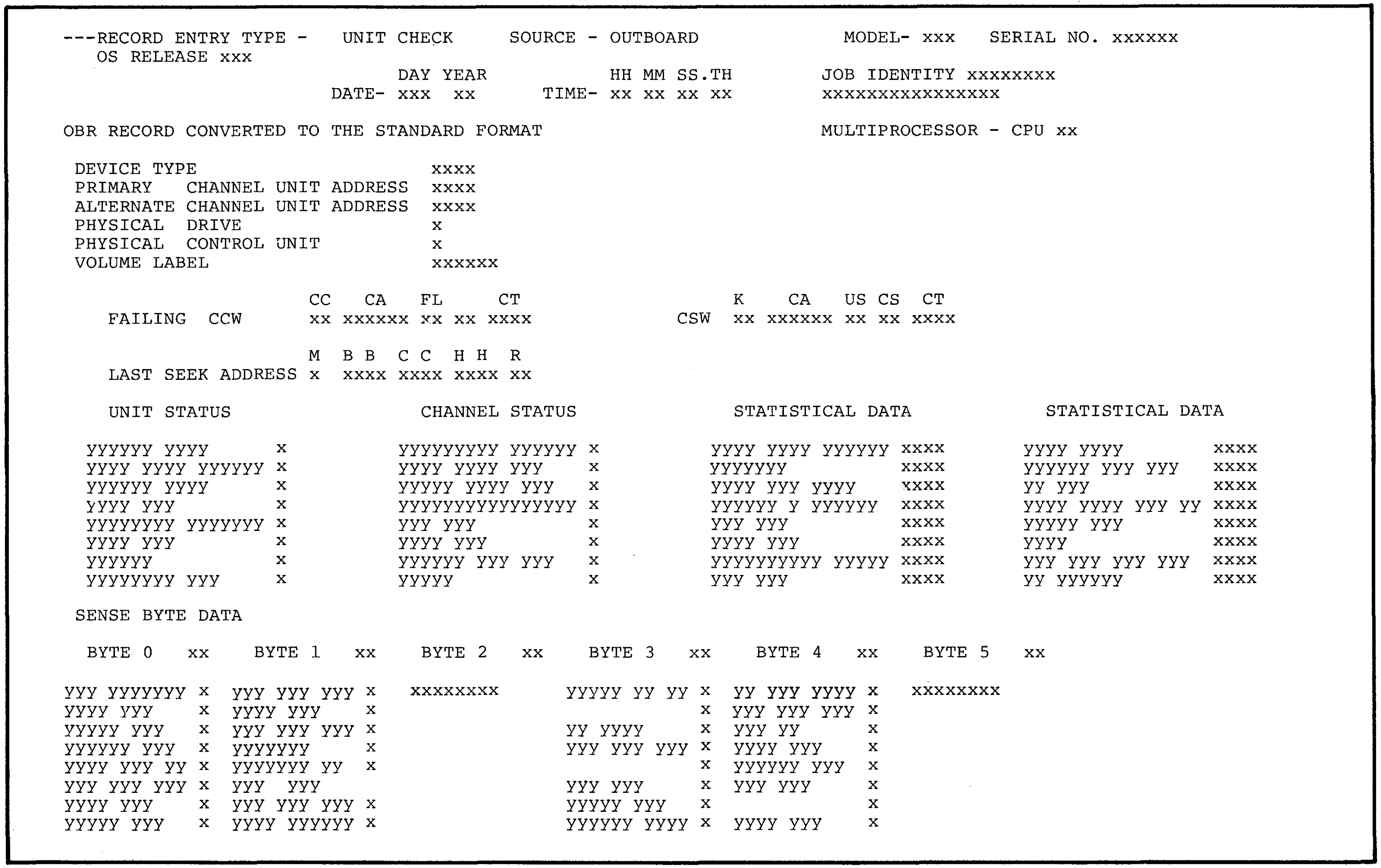
<u>Note</u>: Selected records can be edited and written, accumulated, and/or summarized in one execution of the program.

Figure EREP0-8 shows the format of an I/O outboard summary. Each summarized item is listed with its frequency of occurrence.

Figure EREP0-9 shows the format of the TCAM I/O outboard summary. All totals reference the CUA/line. All subtotals reference terminal names. Individual errors appear under their type of error for every terminal. Graphic errors always appear on the third line under their type of error.

```
     SUMMARY OF I/O OUTBOARD ENVIRONMENT RECORDS FOR DEVICE 031
     TOTAL NUMBER OF RECORDS 005                          DEVICE TYPE 2311

     VOLUME LABELS ENCOUNTERED (MAXIMUM OF 10 ENTRIES)
     VOL. LABEL  22222   001
     VOL. LABEL  22222   002
     VOL. LABEL  22224   002

     CCW COMMAND CODES ENCOUNTERED (MAXIMIM OF 24 ENTRIES)
     CMND       TOTAL
     02         005

     SENSE BYTE SUMMARY

BYTE 0        BYTE 1        BYTE 2        BYTE 3        BYTE 4   BYTE 5

CMND REJ   0  DATA CHK  0   UNSAFE     1  READY      0  BIT 0 0  COMMAND    0
INTV REQ   0  TRK OVERF 0   BIT 1      2  ON LINE    0  BIT 1 1  IN         0
BUS OUT    0  CYL END   1   SERIAL CH  3  UNSAFE     0  BIT 2 2  PROGRESS   0
EQUIP CHK  0  INV SEQ   2   TAG LINE   4  BIT 3      1  BIT 3 3  WHEN       0
DATA CHK   0  REC UNFND 0   ALU CHK    0  ON LINE    0  BIT 4 4  OVERFLOW   0
OVERRUN    0  FILE PROT 3   UNSEL STA  0  CYL END    0  BIT 5 5  INCMPLETE  0
TRK COND   0  MISG A MK 4   BIT 6      0  BIT 6      1  BIT 6 1  OCCURS     0
SEEK CHK   1  OVFL INC  5   BIT 7      0  SEEK INCP  1  BIT 7 1             1
```

Figure EREP0-8. I/O Outboard Summary



```
---RECORD ENTRY TYPE -   UNIT CHECK     SOURCE - OUTBOARD            MODEL- xxx   SERIAL NO. xxxxxx
   OS RELEASE xxx
                              DAY YEAR           HH MM SS.TH       JOB IDENTITY xxxxxxxx
                        DATE- xxx  xx      TIME- xx xx xx xx       xxxxxxxxxxxxxxxx

OBR RECORD CONVERTED TO THE STANDARD FORMAT                        MULTIPROCESSOR - CPU xx

 DEVICE TYPE                      xxxx
 PRIMARY   CHANNEL UNIT ADDRESS   xxxx
 ALTERNATE CHANNEL UNIT ADDRESS   xxxx
 PHYSICAL  DRIVE                  x
 PHYSICAL  CONTROL UNIT           x
 VOLUME LABEL                     xxxxxx

                      CC   CA    FL     CT                      K    CA   US CS  CT
    FAILING  CCW      xx xxxxxx xx xx xxxx               CSW  xx xxxxxx xx xx xxxx

                      M  B B  C C  H H  R
    LAST SEEK ADDRESS x  xxxx xxxx xxxx xx

    UNIT STATUS                 CHANNEL STATUS              STATISTICAL DATA            STATISTICAL DATA

  yyyyyy yyyy      x          yyyyyyyyy yyyyyy x          yyyy yyyy yyyyyy xxxx       yyyy yyyy        xxxx
  yyyy yyyy yyyyyy x          yyyy yyyy yyy    x          yyyyyyy          xxxx       yyyyyy yyy yyy   xxxx
  yyyyyy yyyy      x          yyyyy yyyy yyy   x          yyyy yyy yyyy    xxxx       yy yyy           xxxx
  yyyy yyy         x          yyyyyyyyyyyyyyyy x          yyyyyy y yyyyyy  xxxx       yyyy yyyy yyy yy xxxx
  yyyyyyyy yyyyyyy x          yyy yyy          x          yyy yyy          xxxx       yyyyy yyy        xxxx
  yyyy yyy         x          yyyy yyy         x          yyyy yyy         xxxx       yyyy             xxxx
  yyyyyy           x          yyyyyy yyy yyy   x          yyyyyyyyyy yyyyy xxxx       yyy yyy yyy yyy  xxxx
  yyyyyyyy yyy     x          yyyyy            x          yyy yyy          xxxx       yy yyyyyy        xxxx

 SENSE BYTE DATA

  BYTE 0   xx    BYTE 1   xx    BYTE 2   xx    BYTE 3   xx    BYTE 4   xx    BYTE 5   xx

yyy yyyyyyyy x  yyy yyy yyy x   xxxxxxxx      yyyyy yy yy x  yy yyy yyyy x   xxxxxxxx
yyyy yyy     x  yyyy yyy    x                             x  yyy yyy yyy x
yyyyy yyy    x  yyy yyy yyy x                 yy yyyy     x  yyy yy      x
yyyyyy yyy   x  yyyyyyy     x                 yyy yyy yyy x  yyyy yyy    x
yyyy yyy yy  x  yyyyyyy yy  x                             x  yyyyyy yyy  x
yyy yyy yyy  x  yyy  yyy                      yyy yyy     x  yyy yyy     x
yyyy yyy     x  yyy yyy yyy x                 yyyyy yyy   x              x
yyyyy yyy    x  yyyy yyyyyy x                 yyyyyy yyyy x  yyyy yyy    x
```

Figure EREP0-9. TCAM I/O Outboard Summary

Chapter 3: GTF (Generalized Trace Facility)

Traces selected system events such as SVC and I/O interruptions.

GTF

Contents

INTRODUCTION 55

FEATURES 56

STARTING GTF 57
Using the START Command 57
Using the GTF Cataloged Procedures 59
Specifying GTF Trace Options 60
Prompting 62
Storing Trace Options in SYS1.PARMLIB 63

CALCULATING STORAGE REQUIREMENTS 65

RECORDING USER DATA 67
Printing User Data 67
Coding the GTRACE Macro 67

GTF ERROR RECOVERY HANDLING 69

GTF OUTPUT 70
Trace Records 70
Control Records 76

Figures

Figure GTF-1. General Format of the START Command 57
Figure GTF-2. GTF Cataloged Procedure 59
Figure GTF-3. GTF Messages and Operator Replies While Starting GTF . 63
Figure GTF-4. Adding Trace Options to SYS1.PARMLIB Using IEBUPDTE . 64
Figure GTF-5. Main Storage Requirements for GTF Options, by Module . 66
Figure GTF-6. General Format of the GTRACE Macro, Standard Form . . 67
Figure GTF-7. An Example of the GTRACE Macro 68
Figure GTF-8. Fields in a Trace Record 70
Figure GTF-9. Format of Comprehensive Trace Records for DSP, IO (Including PCI, SIO, PI, EXT, and SVC (MFT and MVT 72
Figure GTF-10. Format of Comprehensive Trace Records for DSP, IO (including PCI),SIO, PI (including SSM) and EXT (Model 65 Multiprocessing) 73
Figure GTF-11. Format of Minimal Trace Records for DSP, IO (including PCI), SIO, PI, EXT, and SVC (MFT and MVT 74
Figure GTF-12. Format of Minimal Trace Records for DSP, IO (including PCI), SIO, PI (including SSM), EXT, and SVC (Model 65 Multiprocessing) 75
Figure GTF-13. Hexadecimal Format Record 75
Figure GTF-14. General format of a timestemp Control Record 76
Figure GTF-15. General Format of a Lost Event Control Record 77

Introduction

The Generalized Trace Facility (GTF) is a feature of OS/360 that allows you to trace selected system events. It also allows you to create your own user trace records and include them in the trace output. The trace output, when formatted and printed by the EDIT function of IMDPRDMP, is useful in determining and diagnosing problems that may arise while using the operating system.

Features

GTF operates as a system task under the operating system; it is compatible with all configurations of the operating system. If the TRACE option has been selected at system generation, the OS Trace facility will function normally except during GTF processing, when OS Trace processing will be suspended.

GTF can trace any or all of the following system events:

- Input/output interruptions (IO)
- START I/O operations (SIO)
- Supervisor Call interruptions (SVC)
- Program interruptions (PI) (including SSM)
- External interruptions (EXT)
- Dispatcher task-switch operations (DSP)

If you choose IO or SIO, you can supply specific device names in response to a prompting message; GTF will then filter out all IO or SIO events that are not associated with the devices you specified. Similarly, you can supply specific SVC numbers when you choose SVC tracing, and specific program interrupt codes when you choose PI tracing.

GTF will ordinarily ignore traceable events that are associated with its own task, but you can request that such events be included as part of the trace output (TRC). You can also request that a timestamp be included in each trace record (TIME=YES).

GTF trace output can be maintained in main storage (MODE=INT) or directed to a data set on an external storage device (MODE=EXT). The output device may be any magnetic tape or direct access device supported by the operating system.

If data is maintained internally or written to a direct access output device, it is "wrapped". That is, when the buffers or available tracks become full, GTF will overlay previously stored or written information beginning at the first buffer or block.

Any abnormally terminating user who has requested ABEND processing will be supplied with formatted trace data as part of the ABEND dump if GTF was active with MODE=INT when ABEND was given control. Similarly, trace data will be provided for SNAP dumps if the user has included the SDATA=TRT parameter in the SNAP macro.

Starting GTF

Use the START command to initiate GTF processing. By specifying certain optional parameters, you can choose whether the trace records should be recorded internally or externally, whether or not they should be time-stamped, and whether or not GTF should terminate if it encounters errors while gathering trace information. You can also select trace options, either by entering them directly through the console or by retrieving them from SYS1.PARMLIB where you have stored them.

Using the START Command

Figure GTF-1 shows the general format of the START command as it is used to start GTF.

```
START   procname[.identifier],[devaddr],[volser],[parmvalue]
         [,keyword=option][...,keyword=option][,REG=size]
```

Figure GTF-1. General Format of the Start Command for GTF

The following discussion describes the parameters of the START command as they are used for GTF.

procname

defines one of the two cataloged procedures (GTF and GTFSNP) described in the next section.

devaddr

indicates the address of the device to which trace output is to be written, if you have specified MODE=EXT. If you have specified MODE=INT, omit this field.

volser

defines the volume serial number of the direct access storage pack to which trace output is to be written, if you have specified MODE=EXT. If you specified MODE=INT, omit this field.

parmvalue

overrides the value specified in the PARM= parameter of the EXEC statement in the cataloged procedure GTF or GTFSNP. This field may contain any combination of the following parameters:

```
MODE=(INT     )
     {EXT     }
     ((INT,S) )
```

defines where the trace data is to be maintained. If you omit this parameter, GTF will assume the default specified in the cataloged procedure (MODE=EXT) and write the trace data on the SYS1.TRACE data set. When MODE=EXT is in effect, you will be prompted to supply trace options unless you have specified a member of SYS1.PARMLIB where trace options are stored.

When MODE=INT is in effect, the trace data is maintained in main storage, and GTF will not prompt you to supply trace options. It will gather basic data (similar to that contained in the OS trace table) for the following events:

- Dispatcher entries
- External interrupts
- I/O interrupts, including program-controlled interrupts.
- Program interrupts
- SIO operations
- SVC interrupts.

When any task in the system terminates abnormally and the ABEND routine is invoked, GTF will suspend tracing until the ABDUMP program can format the trace data as part of the dump output. Trace events missed during ABEND processing will be counted in a special control record that will be included in the trace buffers. If ABEND is not invoked, tracing will continue unaffected. If you specified MODE=(INT,S), GTF will not pause for ABEND or SNAP processing, and the trace buffers will not be formatted.

TIME=$\left\{\begin{matrix}\text{YES}\\\text{NO}\end{matrix}\right\}$

TIME=YES requests that every logical trace record be timestamped (in addition to the block time stamp associated with every block of data). This record timestamp will be four bytes of timer units for systems without Time-of-Day Clock support; for systems with Time-of-Day Clock support, the record timestamp will be the clock value at the time the record was constructed. Note that if no timer option is present in the system, this parameter will be ignored and a warning message will be issued.

If you code TIME=NO, or if you omit this parameter, GTF will not timestamp individual records.

DEBUG=$\left\{\begin{matrix}\text{YES}\\\text{NO}\end{matrix}\right\}$

GTF may encounter errors while attempting to create a trace record. If you specify DEBUG=YES, most errors of this kind will cause GTF to issue an error message and then terminate, so that the contents of the GTF buffers immediately prior to the error will be unchanged. If you have named the GTFSNP procedure in the START command, a SNAP dump will be produced if GTF terminates abnormally.

If you specify DEBUG=NO, or if you omit this parameter, GTF will not terminate immediately, but instead will initiate error recovery procedures. For more information about error recovery procedures, refer to the section "GTF Error Recovery Handling" later in this chapter.

keyword=option

You may use this parameter to override specific parameters in the IEFRDER DD statement in the cataloged procedure. For example:

- To specify a different name for the trace data set, code DSNAME=newname.

- To prevent the system from sending mount messages to the operator's console when specifying MODE=INT, code DSN=NULLFILE.

- To request more than two output buffers, code DCB=(BUFNO=number).

- To modify the GTF buffer size code, DCB=(BLKSIZE=number). The block size cannot be less than 350 bytes.

- To specify an existing data set as the output data set, code DISP=OLD. (Note: If you specify DISP=MOD, GTF will change the data set disposition to OLD.)

Do not use this parameter to request DCB=OPTCD=C; GTF does not support chain-scheduling.

REG=size

supplies a region size for GTF. This will override the value specified in the REGION= parameter of the EXEC statement in the cataloged procedure.

Using the GTF Cataloged Procedures

The START command for GTF names one of two cataloged procedures supplied in SYS1.PROCLIB. The first, GTF, contains job control statements as shown in Figure GTF-2. The second, GTFSNP, is identical to cataloged procedure GTF except that the SNAPDUMP DD statement, shown as optional in Figure GTF-2, is supplied, and the default region size is 30K.

```
//GTF          PROC      REG=26
//IEFPROC      EXEC      PGM=IHLGTF,REGION=&REG.K,
//        PARM='MODE=EXT,DEBUG=NO,TIME=NO'
//IEFRDER      DD        DSNAME=SYS1.TRACE,UNIT=SYSDA,
//        SPACE=(3500,20),DISP=(NEW,KEEP)
//SYSPRINT     DD        SYSOUT=A,SPACE=(TRK,(1,1))
[//SYSLIB      DD        DSN=SYS1.PARMLIB (membername),]
[//       DISP=SHR]
[//SNAPDUMP    DD        SYSOUT=A]
```

Figure GTF-2. The GTF Cataloged Procedure

PROC Statement

defines the default region size for the symbolic REGION= parameter in the EXEC statement. This default value is used if you do not specify a region size in the START command.

EXEC Statement

calls for the execution of GTF. The REGION parameter is specified as a symbolic parameter so that you can vary it according to need.

IEFRDER DD Statement

defines the trace output data set. If you do not override this statement in the START command, the trace output data set will have the name SYS1.TRACE; it will be directed to a direct access device with sufficient allocation to allow the data set to contain twenty 3500-byte physical blocks.

SYSPRINT DD Statement

defines the GTF message data set.

SNAPDUMP DD Statement (Optional in the cataloged procedure GTF, supplied in GTFSNP.)

causes GTF to issue the SNAP macro to dump the nucleus and the GTF region if an error condition causes GTF to terminate. This statement increases GTF's region size requirements by 4K.

SYSLIB DD Statement (Optional)

defines a member in the SYS1.PARMLIB data set that contains GTF options. If such a member exists, GTF will not prompt you to supply options, but will use the options in the member.

Specifying GTF Trace Options

When you start GTF with MODE=EXT, you will receive the following message:

```
IHL100A   SPECIFY TRACE OPTIONS.
```

Use the following format to specify the events to be recorded during GTF execution:

```
TRACE=option1[,option2]...[,optionx]
```

You can specify any of the following trace option values:

$\left\{\begin{array}{l}\text{SYS}\\ \text{SYSM}\\ \text{SYSP}\end{array}\right\}$

SYS requests that comprehensive trace data be recorded for the following system events:

- I/O interrupts
- SVC interrupts
- Program interrupts
- External interrupts
- Start I/O operations

Note: Dispatcher task switching must be requested separately through the DSP keyword. Similarly, program-controlled interrupt must be requested separately through the PCI keyword.

SYSM requests that minimal trace data be recorded for all system events listed above. SYSP requests further prompting for IO, SIO, SVC, and PI; that is, if you specify SYSP, GTF will prompt you to supply specific device addresses, SVC numbers, or program interrupt codes. Comprehensive trace data will be recorded for events associated with the devices or interrupts that you specify; all other events will be filtered out and ignored. If SYS and SYSM, or SYS and SYSP, are both specified, SYS will be ignored. Similarly, if SYSP and SYSM are both specified, SYSP will be ignored.

{SIO }
{SIOP}

SIO requests comprehensive recording for system SIO operations on all devices. SIOP requests further prompting for specific devices for which trace data should be recorded.

This keyword will be ignored if SYS, SYSM, or SYSP has also been specified.

{IO }
{IOP}

IO requests comprehensive recording for all I/O interrupts except program-controlled interrupts, which must be requested separately through the PCI keyword. IOP requests further prompting for specific devices for which I/O interrupts should be recorded.

This keyword will be ignored if SYS, SYSM, or SYSP has also been specified.

{SVC }
{SVCP}

SVC requests comprehensive recording for all SVC interrupts. SVCP requests further prompting for specific SVC numbers for which trace data should be recorded.

This keyword will be ignored if SYS, SYSM, or SYSP has also been specified.

{PI }
{PIP}

PI requests comprehensive recording for all program interrupts. PIP requests further prompting for specific interrupt codes for which trace data should be recorded.

This keyword will be ignored if SYS, SYSM, or SYSP has also been specified.

EXT

requests comprehensive recording for all external interrupts. This keyword will be ignored if SYS, SYSM, or SYSP has also been specified.

DSP

requests that a trace record be created whenever the dispatcher is entered for task switching. The trace data collected will be comprehensive unless you have requested SYSM.

USR

requests that all data passed to GTF via the GTRACE macro be recorded with the system data in the trace data set.

PCI

requests that all program-controlled I/O interrupts be recorded. This keyword will be ignored unless IO, IOP, SYS, SYSM, or SYSP is also specified. If you have specified IOP or SYSP, program-controlled I/O interrupts will be recorded only for those devices that you supplied in response to a prompting message.

TRC

requests tracing of trace events associated with the trace task while operating under GTF's task control block. Such events will be traced according to the GTF trace options selected while starting GTF. If this keyword is not specified, GTF task events will be filtered out and not recorded.

SSM

requests all program interrupts caused by SSM instructions to be recorded. This keyword is effective only in a multiprocessing environment, and only when PI, PIP, SYS, SYSM or SYSP is also specified.

Prompting

When you specify SYSP, IOP, SIOP, SVCP, or PIP as trace options, GTF will prompt you to supply specific values. These values are:

SIO=(devaddr1[,devaddr2][...,devaddr50])

specifies up to 50 device addresses for which you want SIO operations traced. All other SIO operations will be filtered out. If you have specified SIOP or SYSP, and do not specify SIO= in response to the prompting message, no SIO filtering will take place.

IO=(devaddr1[,devaddr2][...,devaddr50])

specifies up to 50 device addresses for which you want I/O interruptions traced. All other IO interruptions will be filtered out. If you have specified IOP or SYSP, and do not specify IO= in response to the prompting messages, no IO interruption filtering will take place.

SVC=(svcnum1[,svcnum2][...,svcnum50])

specifies up to 50 SVC numbers that you want traced. All other SVC numbers will be filtered out. If you have specified SVCP or SYSP, and do not specify SVC= in response to the prompting message, no SVC filtering will take place.

PI=(code1[,code2][...,code15])

specifies up to 15 program interrupt codes that you want traced. All other program interruptions will be filtered out. If you have specified PIP or SYSP, and do not specify PI= in response to this prompting message, no program interruption filtering will take place.

IO=SIO=(devaddr1[,devaddr2][...,devaddr50])

specified after requesting SYSP or both IOP and SIOP, names up to 50 device addresses for which you want GTF to trace both IO and SIO events. All other IO and SIO events, except those requested specifically by IO= or SIO=, will be filtered out.

Note that in each case GTF imposes a limit on the number of specific values you can supply through prompting. If you exceed this limit, GTF will issue a message and you must respecify all values.

Figure GTF-3 shows an example of an exchange between GTF and the operator when GTF is being started.

```
    START GTF,,,(MODE=EXT),REG=34
          .
          .
          .
 00 IHL100A SPECIFY TRACE OPTIONS
r00,'TRACE=SYSP,USR'
          .
          .
          .
 01 IHL101A SPECIFY TRACE EVENT KEYWORDS--SVC=,IO=,SIO=,PI=
r01,'SVC=(1,2,3,4,10),IO=(191,192)'
          .
          .
          .
 02 IHL102A CONTINUE TRACE DEFINITION OR REPLY END
r02,'SIO=282,END'
IHL103I TRACE OPTIONS SELECTED--SYSP,USR
IHL103I SVC=(1,2,3,4,10),IO=(191,192),SIO=(282)
 03 IHL125A RESPECIFY TRACE OPTIONS OR REPLY U
r03,'U'
```

Figure GTF-3. GTF messages and operator replies while starting GTF.

Storing Trace Options in SYS1.PARMLIB

You can save time in starting GTF by storing one or more set combinations of trace options as members in SYS1.PARMLIB. GTF will not prompt you to supply trace options, but will will look in SYS1.PARMLIB if you include a SYSLIB DD statement in the GTF or GTFSNP cataloged procedures.

Figure GTF-4 shows the job control statements and utility control statements needed to add trace options to SYS1.PARMLIB using IEBUPDTE. For full descriptions of the statements, refer to the publications IBM System/360 Operating System: Utilities, GC28-6586, and Job Control Language Reference, GC28-6703.

```
//GTFPARM      JOB       MSGLEVEL=(1,1)
//             EXEC      PGM=IEBUPDTE,PARM=NEW
//SYSPRINT     DD        SYSOUT=A
//SYSUT2       DD        DSNAME=SYS1.PARMLIB,DISP=SHR
//SYSIN        DD        DATA
./        ADD       NAME=GTFA,LIST=ALL,SOURCE=0
TRACE=SYSP,USR
SVC=(1,2,3,4,10),IO=(191,192),SIO=282,PI=15
./        ADD       NAME=GTFB,LIST=ALL,SOURCE=0
TRACE=IO,SIO,TRC
./        ADD       NAME=GTFC,LIST=ALL,SOURCE=0
TRACE=SYS,PCI,SSM
/*
```

Figure GTF-4. Adding Trace Options to SYS1.PARMLIB Using IEBUPDTE.

A sample //SYSLIB DD statement to be included in the GTF or GTFSNP cataloged procedure might look like this:

```
    //SYSLIB  DD   DSN=SYS1.PARMLIB(GTFA),DISP=SHR
```

Calculating Storage Requirements

GTF's region requirements vary according to the GTF options that you specify.

If you have requested MODE=INT, you must specify a minimum region size of 16K bytes of main storage. This minimum will provide you with four 1024-byte buffers. If you need more buffers, you must specify 1K of additional storage for each buffer. If you use the GTFSNP cataloged procedure, or if you use an installation-defined procedure that contains a SNAPDUMP DD statement, you must add 4K to the minimum region size.

If you have requested MODE=EXT, you must specify a minimum region size of 26K. For larger regions, use the following formula to compute your region requirements. Note that all intermediate values must be rounded up to the nearest 2K multiple. The final region size that you calculate must also be rounded up to the nearest 2K multiple.

region = 16K + n(b+8) + 88(n) + m + a

Where:

16K

minimum main storage required for minimal trace.

n

number of trace buffers, ordinarily two unless you have specified more in the START command.

b

the size of the trace buffers, ordinarily 3500 bytes unless you have specified a different value in the START command. (Note: When trace output is directed to a direct access device, the buffer size should equal the track size. This is necessary to prevent too much previously stored data from being lost when the trace data is "wrapped". The 8 additional bytes are needed for the GTF buffer prefix.

88

the size of the input/output block (IOB); one IOB is required for each buffer.

m

total main storage required to process GTF options requested. In some cases, several GTF options are contained within one module. Even if you request two or more GTF functions that are contained in the same module, you only need to provide enough space for one copy of the module. Refer to Figure GTF-5 for a summary of GTF options, the modules that contain them, and the amount of main storage required for each module.

To calculate m, add together the storage requirements for each module that you will need, and add 1K to the total if you have requested filtering for any option. For example, if you specify EXT, SVCP, and USR:

$$m = 2K + 8K + 1K + 0.5K$$

$$m = 11.5K$$

a

The amount of main storage required for ABEND or SNAP processing. If you have requested either ABEND or SNAP, or both, when starting GTF, this value is 4K. If you have not requested ABEND or SNAP, this value is zero.

GTF OPTIONS SELECTED	MODULES REQUIRED	MAIN STORAGE REQUIRED
SYSM[,DSP][,PCI]	IHLSYSV or IHLSYSP	1K
DSP EXT PI PI= SSM	IHLTPED	2K
IO IO= SIO SIO= PCI	IHLTSIO	1K
SVC SVC=	IHLTSVC	8K
SYS[,DSP][,PCI] [,SSM]	IHLTPED, IHLTSIO, and IHLTSVC	11K
USR	IHLTUSR	0.5K
IOP SIOP SVCP PIP	IHLTFIL	1K

Figure GTF-5. Main Storage Requirements for GTF Options, By Module. Note that TRC can be considered to require 0 (zero K) bytes of main storage.

Recording User Data

If you want your own trace data to be recorded in the GTF trace buffers, you can specify that data in the GTRACE macro instruction. In one invocation of GTRACE, an application program can record up to 256 bytes of data in a GTF trace buffer. Secure data should not be recorded using the GTRACE macro since security protection cannot be guaranteed. Note, however, that GTRACE can record only data that has the same protect key as the GTRACE user.

GTRACE will be effective only when GTF is active, when it is directing its output to an external data set, and when it is accepting user data -- that is, when GTF has been started with MODE=EXT and TRACE=USR specifications.

Printing User Data

Like other trace data, information recorded by the GTRACE macro can be printed by the EDIT function of IMDPRDMP. Usually user data will be printed in hexadecimal, since EDIT cannot format records not created by GTF. However, you can write format appendages to format specific types of user data records.

Every time you issue GTRACE to create a user record, you specify which format appendage should process it; you do this by including the optional FID (format identifier) parameter in the GTRACE invocation. The FID corresponds to the last two hexadecimal characters in the name of the format appendage, IMDUSRxx.

Coding the GTRACE Macro

Figure GTF-6 shows the general format of th e GTRACE macro, standard form.

```
[symbol] GTRACE    DATA=address,LNG=number,ID=number[,FID=value]
```

Figure GTF-6. The General Format of the GTRACE Macro, Standard Form

The parameters in the macro are described below.

DATA=address

gives the main storage address of the data to be recorded.

LNG=number

specifies the number of bytes (1 to 256) to be recorded from the address specified in the DATA= parameter. The number may be specified in decimal or in hexadecimal (as X'number').

ID=value

is the identifier to be associated with the record. ID values are assigned as follows:

0 to 1023 -- user events

1024 to 4095 -- reserved

The value may be specified in decimal or in hexadecimal (as X'value').

FID=value

indicates the format appendage that is to format this record when the trace output is processed by the EDIT function of IMDPRDMP. FID values are assigned as follows:

0 (or FID= parameter omitted) -- record to be dumped in hexadecimal

1 to 80 -- user format identifiers

81 to 255 -- reserved

The value may be specified in decimal or in hexadecimal (as X'value').

Figure GTF-7 shows how the GTRACE macro might be coded to record 200 bytes of data, beginning at the address of AREA, with an event identifier of 37 and to be formatted by the format appendage with the name IMDUSR64.

```
GTRACE    DATA=AREA,LNG=200,ID=37,FID=100
```

Figure GTF-7. An Example of the GTRACE Macro.

For more details about the GTRACE macro instruction, consult the publication IBM System/360 Operating System: Supervisor Services and Macro Instructions, GC28-6646.

GTF Error Recovery Handling

GTF recognizes all errors that occur while building a trace record as potentially recoverable. Whether recovery takes place or not depends on what you code in the START command.

If you specify DEBUG=YES, GTF will not attempt error recovery. It will issue an error message and then terminate, so that the contents of the GTF buffers immediately prior to the error will be preserved.

If you specify DEBUG=NO, GTF will initiate the following error procedures:

> For minor errors in the routine that builds the trace record (the build routine), GTF flags the field that led to the error and continues processing. It does not issue a message to the operator's console or disable the function that caused the error; instead, it proceeds as if no error had occurred. All errors that occur while building an SVC record fall into this category.
>
> For severe errors in the build routine, GTF flags the entire record that was being built, issues a message to the console, and continues processing with the function that caused the error suppressed.
>
> For errors in the routine that filters trace events, GTF suppresses filtering for future events of the same type, issues a message to the console, and continues processing.

Errors that occur outside the build and filter routines are not recoverable; they result in immediate abnormal termination of GTF.

Note that the termination of GTF will never cause termination of a user's task.

GTF Output

GTF creates two kinds of records: trace records and control records.

Trace Records

GTF creates trace records for each system event you select. The records have the general format shown in Figure GTF-8.

```
length  00  AID  FID  Timestamp   EID  DATA
  |      |    |    |      |        |    |
  |      |    |    |      |        |    └── Trace data (up to 256 bytes)
  |      |    |    |      |        |
  |      |    |    |      |        └── Event identifier (2 bytes)
  |      |    |    |      |
  |      |    |    |      └── Timestamp (optional; 8 bytes)
  |      |    |    |
  |      |    |    └────── Format Identifier (1 byte)
  |      |    |
  |      |    └────── Application Identifier (1 byte)
  |      |
  |      └Always zero (2 bytes)
  |
  └─Number of bytes in trace record (2 bytes)
```

Figure GTF-8. Fields in a trace record.

The fields in the record are described as follows:

length

indicates the total length of the record.

00

always zero.

AID

defines whether the data record is a trace record or a GTF control record.

X'FF' -- trace record

X'00' -- GTF control record

X'01' to X'FE' -- reserved

FID

is the format identifier, a one-byte hexadecimal number that identifies the program that will format the trace record during EDIT execution. (For information on specifying the FID in the GTRACE macro, refer to the section "Coding the GTRACE Macro" in this chapter.)

If this field is zero, the trace record will not be formatted, but will be dumped in hexadecimal.

timestamp

If TIME=YES was specified in the START command and a timer option is in effect in the system, a time stamp will be included in this eight-byte field. If GTF is executing on a system without the Time-Of-Day Clock, the time stamp will be four bytes of timer units, right justified. On a system with Time-Of-Day clock support, the value in the record will be the clock value at the time the record was constructed.

EID

defines the event that caused the trace record to be created. It is not present in GTF control records. You can determine the EID of a trace record by issuing the IMDMEDIT mapping macro, which is described in the Appendix: Writing EDIT User Programs.

data

This field contains the trace data gathered for the requested event. The length of this field varies according to the event being traced.

Figures GTF-9 through GTF-13 are examples of trace output as processed by the EDIT function of IMDPRDMP. In all the examples, fields flagged with hhhhhhhh are hexadecimal representations, and fields flagged with cccccccc are alphameric characters. N/A signifies that the field label does not apply to this paricular record. For explanation of the fields in the records, refer to the Programmer's Guide to Debugging, GC28-6670.

```
DSP RES PSW hhhhhhhh hhhhhhhh   JOBN {cccccccc}   MODN {WAITTCB } NUTCB hhhhhhhh   PRTY hh
                                     {  N/A   }        {SVC-cccc}
                                                       {SVC-RES }
                                                       {**IRB***}
                                                       {cccccccc}
                                                       {Icccccccc}

{IO }  cuu   OLD PSW hhhhhhhh hhhhhhhh   JOBN {********}  DDNM {********}  OLTCB hhhhhhhh
{PCI}                                         {cccccccc}       {cccccccc}
                                              {  N/A   }       {  N/A   }

               CSW hhhhhhhh hhhhhhhh   RQE {******** ******** ********}  RQE TCB {********}  SENS {hhhhhhhh}
                                           {hhhhhhhh hhhhhhhh hhhhhhhh}          {hhhhhhhh}       {  N/A   }
                                           {          N/A             }          {  N/A   }

SIO   cuu   CC hh   CAW hhhhhhhh JOBN {cccccccc} OLTCB hhhhhhhh CSW hhhhhhhh hhhhhhhh   RQE hhhhhhhh hhhhhhhh hhhhhhhh   RQE TCB hhhhhhhh
                                      {  N/A   }

PGM ccc   OLD PSW hhhhhhhh hhhhhhhh   JOBN {cccccccc}   MODN {WAITTCB }   OLTCB hhhhhhhh
                                           {  N/A   }        {SVC-cccc}
                                                             {SVC-RES }
                                                             {**IRB***}
                                                             {cccccccc}
                                                             {Icccccccc}

        R0  hhhhhhhh R1  hhhhhhhh R2  hhhhhhhh R3  hhhhhhhh R4  hhhhhhhh R5  hhhhhhhh R6  hhhhhhhh R7  hhhhhhhh
        R8  hhhhhhhh R9  hhhhhhhh R10 hhhhhhhh R11 hhhhhhhh R12 hhhhhhhh R13 hhhhhhhh R14 hhhhhhhh R15 hhhhhhhh

EXT   OLD PSW hhhhhhhh hhhhhhhh   JOBN {cccccccc}   MODN {WAITTCB }   OLTCB hhhhhhhh   TQEFLG/TCB {********}  EXIT {********}
                                       {  N/A   }        {SVC-cccc}                               {hhhhhhhh}       {hhhhhhhh}
                                                         {SVC-RES }                               {  N/A   }       {  N/A   }
                                                         {**IRB***}
                                                         {cccccccc}
                                                         {Icccccccc}

SVC nnn   OLD PSW hhhhhhhh hhhhhhhh   JOBN cccccccc   MODN cccccccc   OLTCB hhhhhhhh   R15/R0 hhhhhhhh hhhhhhhh   R1 hhhhhhhh
```

Figure GTF-9. Format of Comprehensive Trace Records for DSP, IO (including PCI), SIO, PI, EXT, and SVC (MFT and MVT)

```
{A}  DSP RES PSW hhhhhhhh hhhhhhhh  JOBN {cccccccc}  MODN (WAITTCB  ) NUA hhhhhhhh NUB hhhhhhhh  PRTY hh
{B}                                      {  N/A   }       (SVC-cccc )
                                                          (SVC-RES  )
                                                          (**IRB*** )
                                                          (cccccccc )
                                                          (Icccccccc)


{A} {IO }  cuu   OLD PSW hhhhhhhh hhhhhhhh  JOBN {********} DDNM {********} OLA hhhhhhhh  OLB hhhhhhhh
{B} {PCI}                                        {cccccccc}      {cccccccc}
                                                 {  N/A   }      {  N/A   }

            CSW hhhhhhhh hhhhhhhh  RQE{******** ******** ********}  RQE TCB {********} SENS{hhhhhhhh}
                                      {hhhhhhhh hhhhhhhh hhhhhhhh}          {hhhhhhhh}     {  N/A   }
                                      {           N/A            }          {  N/A   }


{A}  SIO cuu  CC hh  CAW hhhhhhhh  JOBN {cccccccc}  OLA hhhhhhhh OLB hhhhhhhh  CSW hhhhhhhh hhhhhhhh
{B}                                     {  N/A   }

            RQE hhhhhhhh hhhhhhhh hhhhhhhh  RQE TCB hhhhhhhh


{A}  PGM ccc  OLD PSW hhhhhhhh hhhhhhhh  JOBN  {cccccccc}  MODN (WAITTCB  )  OLA hhhhhhhh OLB hhhhhhhh
{B}                                            {  N/A   }       (SVC-cccc )
                                                                (SVC-RES  )
                                                                (**IRB*** )
                                                                (cccccccc )
                                                                (Icccccccc)

            R0  hhhhhhhh R1  hhhhhhhh R2  hhhhhhhh R3  hhhhhhhh R4  hhhhhhhh R5  hhhhhhhh R6  hhhhhhhh R7  hhhhhhhh
            R8  hhhhhhhh R9  hhhhhhhh R10 hhhhhhhh R11 hhhhhhhh R12 hhhhhhhh R13 hhhhhhhh R14 hhhhhhhh R15 hhhhhhhh


{A}  SSM OLD PSW hhhhhhhh hhhhhhhh JOBN {cccccccc}  MODN (WAITTCB  )  OLA hhhhhhhh  OLB hhhhhhhh  LKID hh
{B}                                     {  N/A   }       (SVC-cccc )
                                                         (SVC-RES  )
                                                         (**IRB*** )
                                                         (cccccccc )
                                                         (Icccccccc)


{A}  EXT  OLD PSW hhhhhhhh hhhhhhhh  JOBN  cccccccc  MODN (WAITTCB  )  OLA hhhhhhhh OLB hhhhhhhh  STMSK hhhhhhhh
{B}                                                       (SVC-cccc )
                                                          (SVC-RES  )
                                                          (**IRB*** )
                                                          (cccccccc )
                                                          (Icccccccc)

        TQEFLG/TCB {********}  EXIT {********}
                   {hhhhhhhh}       {hhhhhhhh}
                   {  N/A   }       {  N/A   }


{A}  SVC nnn  OLD PSW hhhhhhhh hhhhhhhh  JOBN {********}  MODN (**IRB***)  OLA hhhhhhhh OLB hhhhhhhh
{B}                                           {cccccccc}       (SVC-RES )
                                              {  N/A   }       (SVC-cccc)
                                                               (  N/A   )
                                                               (********)
        R15/R0 hhhhhhhh hhhhhhhh      R1 hhhhhhhh
```

Figure GTF-10. Format of Comprehensive Trace Records for DSP, IO (including PCI), SIO, PI (including SSM) EXT, and SVC (Model 65 Multiprocessing)

```
DSP   NEW PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  NEW TCB hhhhhhhh

{IO }  OLD PSW hhhhhhhh hhhhhhhh  CSW hhhhhhhh hhhhhhhh  RQE TCB {********} OLD TCB hhhhhhhh
{PCI}                                                            {hhhhhhhh}
                                                                 {  N/A   }

SIO   CC/DEV/CAW hhhhhhhh hhhhhhhh  CSW hhhhhhhh hhhhhhhh  RQE TCB {********} OLD TCB hhhhhhhh
                                                                   {hhhhhhhh}
                                                                   {  N/A   }

PGM   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLD TCB hhhhhhhh

EXT   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  TQE TCB {********}
                                                                                {hhhhhhhh}
                                                                                {  N/A   }

SVC   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLD TCB hhhhhhhh
```

Figure GTF-11. Format of Minimal Trace Records for DSP, IO (including PCI), SIO, PI, EXT, and SVC (MFT and MVT)

```
{A}  DSP   RES PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  NUA hhhhhhhh NUB hhhhhhhh
{B}

{A} {IO }  OLD PSW hhhhhhhh hhhhhhhh  CSW hhhhhhhh hhhhhhhh  RQE TCB {********} OLA hhhhhhhh OLB hhhhhhhh
{B} {PCI}                                                           {hhhhhhhh}
                                                                    {  N/A   }

{A}  SIO   CC/DEV/CAW hhhhhhhh hhhhhhhh  CSW hhhhhhhh hhhhhhhh  RQE TCB {********} OLA hhhhhhhh OLB hhhhhhhh
{B}                                                                    {hhhhhhhh}
                                                                       {  N/A   }

{A}  PGM   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLA hhhhhhhh OLB hhhhhhhh
{B}

{A}  SSM   LK hh  OPSW hhhhhhhh hhhhhhhh  R15/R0  hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLA hhhhhhhh  OLB hhhhhhhh
{B}

{A}  EXT   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLA hhhhhhhh OLB hhhhhhhh
{B}

{A}  SVC   OLD PSW hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh  OLA hhhhhhhh OLB hhhhhhhh
{B}
```

Figure GTF-12. Format of Minimal Trace Records for DSP, IO (including PCI), SIO, PI (including SSM), EXT, and SVC (Model 65 Multiprocessing)

```
{HEXFORMAT}  AID hh  FID hh  EID hh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
{USER     }                          hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
{SYSTEM   }                          hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
{SUBSYS   }                          hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                                     hhhhhhhh hhhh...
```

Figure GTF-13. Hexadecimal Format Records

Control Records

GTF produces two types of control records: timestamp records and lost data records. The first record in every block of trace output is a timestamp record. A lost data record appears to signal trace events that were not recorded because the GTF buffers were full or because GTF has temporarily suspended operations during ABEND or SNAP processing. Figure GTF-14 shows the general format of a timestamp record.

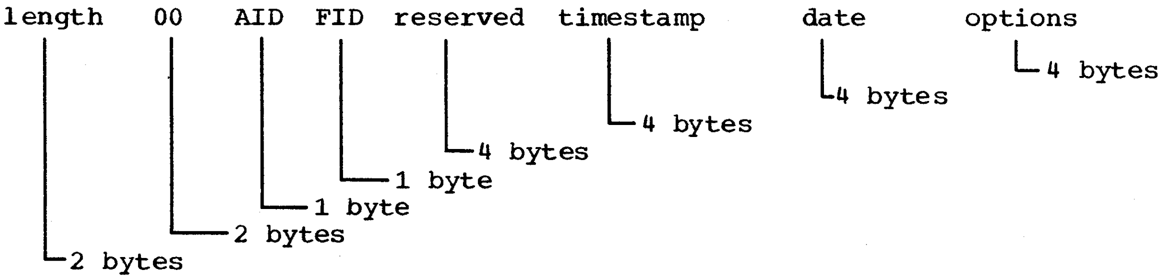


Figure GTF-14. General Format of a Timestamp Control Record.

The fields in the record contain the following information:

length

total length of the record in bytes.

00

always zero.

AID

For control records this field is always zero.

FID

For timestamp control records, this field is always X'04'.

reserved

reserved for future use.

timestamp

timer units (in hexadecimal) representing the time when the control record was constructed. If GTF is running on an MFT system with no timer option, this field is zero.

date

year and julian day, in hexadecimal. The format is X'00 yy dd dc', where c is the packed decimal sign.

options

GTF options in effect. For detailed information about this field, see Figure APNDX-2 in the Appendix.

Figure GTF-15 shows the general format of a lost event record.

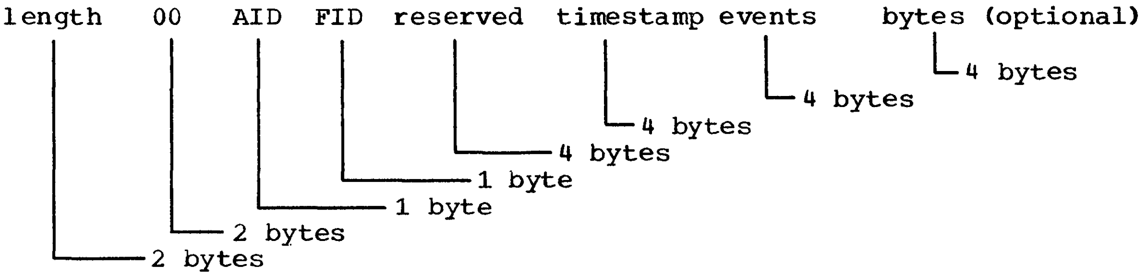


Figure GTF-15. General Format of a Lost Event Record.

The fields in the record contain the following information.

length

total record length in bytes.

00

always 00

AID

always 00 in control records

FIC

format identifier. Valid values are:

X'05' -- events lost because buffers full.

X'06' -- events lost because GTF disabled temporarily.

reserved

reserved for future use.

timestamp

timer units (in hexadecimal) representing the time when the control record was constructed. If GTF is running on an MFT system with no timer option, this field is zero.

events

number of traceable events lost (in hexadecimal).

bytes (present only in records created under buffer-full condition)

number of bytes of data lost (in hexadecimal). This field is not formatted by the EDIT function of IMDPRDMP.

Chapter 4: IMCJQDMP

Operates as a stand-alone program to format and print the system job queue. ——→

JQDMP

Contents

INTRODUCTION 83

FUNCTION OF IMCJQDMP 84

RETRIEVING IMCJQDMP 86

JOB QUEUE FORMAT 88

USING IMCJQDMP 91
Device Identification Command 91
  Output Address Parameter 91
  Input Address Parameter 92
  Selective Dumping Parameters 92
    QCR= Parameter 92
    JOBNAME= Parameter 93
    Combining QCR= and JOBNAME= Parameters 93
Completion Message 93
Tape Output Processing 94
  Standard Label Processing 94
    Standard Labeled Output 95
    Non-Labeled Output 95
Abnormal Termination of IMCJQDMP 95

IMCJQDMP OUTPUT 97
Record Identification 99
Zero Records in the Dump 100
Contents of the Dump Listing .101

OPERATIONAL CONSIDERATIONS 102

Figures

Figure JQDMP-1. Sample JCL Statements Needed to Punch IMCJQDMP from Component Library SYS1.DN554A 86
Figure JQDMP-2. Flow of Processing from IMCJQDMP 87
Figure JQDMP-3. Sample Job Queue (SYS1.SYSJOBQE) Format After Initialization 88
Figure JQDMP-4. Logical Track Header (LTH) Record Format 89
Figure JQDMP-5. Example of Minor Job Queue Control Record 90
Figure JQDMP-6. Master Job Queue Control Record Format 90
Figure JQDMP-7. Sample JCL Needed to Print 9-Track IMCJQDMP Tape Output 94
Figure JQDMP-8. Sample of IMCJQDMP Output Listing 98
Figure JQDMP-9. Queue Record Type Identification 100
Figure JQDMP-10. Queue Control Record Identification 100
Figure JQDMP-11. IMCJQDMP Execution Time Per 100 Tracks of Input, As a Function of the Output Device 102

Introduction

IMCJQDMP is a service aid program that produces a formatted copy of the contents of the IBM System/360 operating system's job queue data set (SYS1.SYSJOBQE). The program operates in stand-alone mode; that is, it is independent of any operating system.

It may be said that system control is centered in the job queue. Its component tables and blocks store the dynamic environmental descriptions that regulate the processing of all jobs submitted to the operating system. Detailed descriptions and layouts of the record types which may be encountered in the job queue data set may be found in the following publications: IBM System/360 Operating System: MVT Job Management, GY28-6603, and Control Program with MFT, GY27-7128.

IMCJQDMP may be used to dump the entire job queue, or the user may optionally specify selected portions of it for printing.

Function of IMCJQDMP

In determining the cause of a job or system failure, it is often desirable to know precisely what was contained in the job queue, or in specific portions of it, at the time of such failure.

For example, the user may attempt to initiate a warm start, and fail. A warm start failure tends to be a critical problem, as it is dependent upon job queue structure for its proper functioning. A dump of the job queue would be an invaluable aid in tracing the cause of such a failure.

There are also the instances in which the Scheduler ABEND 0B0 occur, indicating an I/O error on the job queue data set. This ABEND is often caused by an invalid TTR-address being used to access the job queue. A job queue dump provides precise information as to the address of each record, and, in addition, allows access to certain queue records which are chained together by a TTR-address contained in a primary record. Such information is vital in determining the cause of the I/O failure.

In many other situations, it may be necessary to interpret and examine the main storage chains reflected in the control blocks contained in the job queue.

Optimally, this information should be made available to the user:

- Without disturbing the prevailing status of the job queue;
- whether or not the system is operational;
- without prior knowledge of the exact location of the job queue data set on its assigned direct access volume;
- on a record-by-record basis, according to direct access volume address; and
- conveniently formatted for ready access and interpretation.

The IMCJQDMP program is designed to supply specialized job queue dumps incorporating all these features.

The program functions in stand-alone fashion, a circumstance which is particularly beneficial in instances where the system is involved in the failure. Since it does not function under the operating system, it is not enqueued upon the job queue data set and, therefore, does not alter the existing status of the records that are to be dumped. The printed queue records reflect precisely what they contained at the time of malfunction. Nor is it required that the user know the explicit address of SYS1.SYSJOBQE. Only the address assigned to the direct access device on which the volume containing the job queue is mounted need be supplied to the dump program. The program determines the address of the job queue data set by reading the queue volume's VTOC (volume table of contents). The VTOC contains data set control blocks (DSCBs) corresponding to each data set and to contiguous blocks of unassigned tracks on the volume.

When the queue has been found by IMCJQDMP, records are read and, according to the user's exercise of the available options, are either serially or selectively identified by type and address, formatted, and written to the chosen output device. This may be either a 1403 printer or an unlabeled 9- or 7-track magnetic tape volume. Printing the tape output of IMCJQDMP is discussed under "Tape Output Processing."

Retrieving IMCJQDMP

The Job Queue Dump program is supplied in object module form, together with an absolute loader. The program resides on the OS/360 Distribution Library packs as a member (IMCJQDMP) of component library SYS1.DN554A. In preparing the program for use, the module IMCJQDMP must be punched from the component library or copied to a nonlabeled magnetic tape. The card deck or tape may then be used to load the program for execution.

The JCL statements for punching the program from the component library are shown in Figure JQDMP-1. This example assumes that the distribution libraries are cataloged; if they are not, add the UNIT and VOL=SER parameters to the SYSUT1 data definition statement.

```
//QDUMP       JOB       MSGLEVEL=(1,1)
//STEP        EXEC      PGM=IEBPTPCH
//SYSPRINT    DD        SYSOUT=A
//SYSUT1      DD        DSN=SYS1.DN554A(IMCJQDMP),DISP=OLD,
//                 DCB=(BLKSIZE=3600,LRECL=80,RECFM=FB)
//SYSUT2      DD        UNIT=2540-2
//SYSIN       DD        *
    PUNCH
/*
```

Figure JQDMP-1. Sample JCL Statements Needed to Punch IMCJQDMP from Component Library SYS1.DN554A

IMCJQDMP may be used with any S/360 or S/370 CPU, and requires about 18K bytes of main storage for execution. I/O device requisites are a card reader (or, optionally, a 2400 tape drive) for initial program loading (IPL); one of the following consoles: 1052, 3066, 3210, 3215, or 5450; and one of the following DASD devices -- 2311, 2312, 2313, 2314, 2318, 2319, 2301, 2303, 2305, or 3330 -- for input, and either a 1403 printer or a 2400 tape drive for output. Figure JQDMP-2 describes the flow of processing when IMCJQDMP is used.

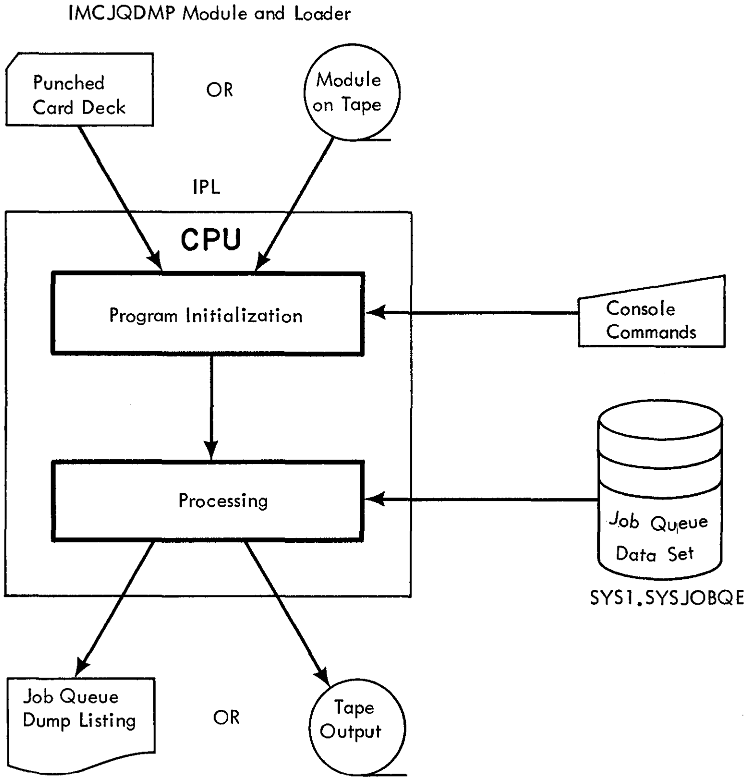
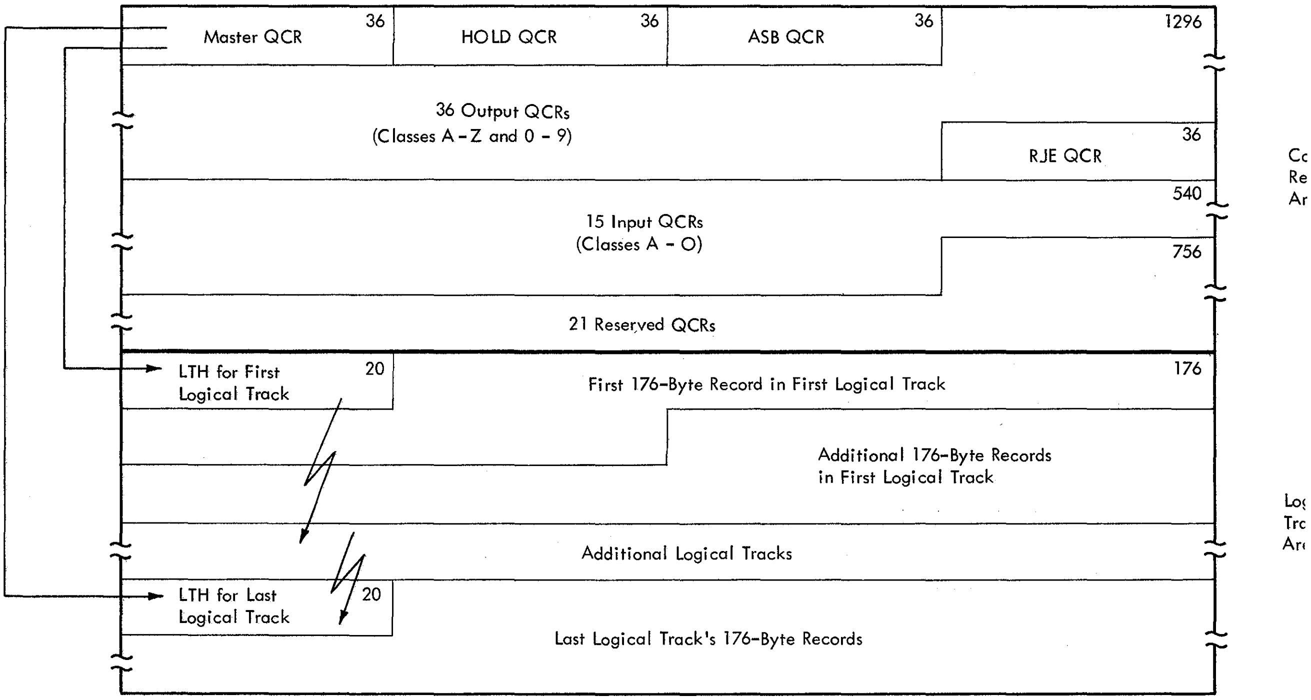


Figure JQDMP-2. Flow of Processing for IMCJQDMP

Job Queue Format

Input to IMCJQDMP is the system's job queue data set (SYS1.SYSJOBQE), which is maintained on a permanently resident direct access volume. The job queue is composed of control records and work queues, created and updated by diverse system components. Figure JQDMP-3 shows the format of a job queue.



Master QCR
36
HOLD QCR
36
ASB QCR
36
1296
36 Output QCRs
(Classes A-Z and 0 - 9)
36
RJE QCR
540
15 Input QCRs
(Classes A - O)
756
21 Reserved QCRs
Cc
Re
Ar
LTH for First Logical Track
20
First 176-Byte Record in First Logical Track
176
Additional 176-Byte Records in First Logical Track
Additional Logical Tracks
LTH for Last Logical Track
20
Last Logical Track's 176-Byte Records
Lo
Trc
Ar

Figure JQDMP-3. Sample Job Queue (SYS1.SYSJOBQE) Format After Initialization

The job queue data set consists of 76 work queues:

- 15 input queues, one for each job class.
- 36 output queues, one for each output class.
- 1 free-track queue, from which work queue space is assigned as needed. Immediately after job queue initialization, the entire data set consists of free tracks.
- 1 automatic SYSIN batching (ASB) queue.
- 1 TSO Background Reader queue.
- 1 remote job entry (RJE) queue.
- 1 HOLD queue for temporarily dequeued jobs.
- 21 reserved queues.

These work queues consist of assigned logical tracks. A logical track may be defined as an area of contiguous space in the data set large enough to contain a 20-byte logical track header (LTH) record, followed by a predetermined number of 176-byte data records. Figure JQDMP-4 describes the format of a logical track header record.

<table>
<tr><th colspan="2">Offset</th><th colspan="4"></th></tr>
<tr><th>Hex</th><th>Dec</th><th colspan="4"></th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Reserved 4</td></tr>
<tr><td>4</td><td>4</td><td colspan="4">Reserved 4</td></tr>
<tr><td>8</td><td>8</td><td>Reserved 1</td><td colspan="2">First Logical Track of the Job 2</td><td>Reserved 1</td></tr>
<tr><td>C</td><td>12</td><td colspan="2">Next Logical Track of the Job 2</td><td>Number of Tracks Assigned 1</td><td>Type 1</td></tr>
<tr><td>10</td><td>16</td><td>Reserved 1</td><td>Jobclass of the Job 1</td><td colspan="2">Last Logical Track of the Next Job 2</td></tr>
</table>

Figure JQDMP-4. Logical Track Header (LTH) Record Format

In Figure JQDMP-4 and subsequent figures, where applicable, byte size of a field is shown in the upper right corner; offset from the beginning of the record, in hexadecimal and decimal notation, is given along the left margin.

Content of the type field in an LTH record indicates the type of queue to which the logical track has been assigned:

Field Content	Queue Type
1	HOLD Queue
2	ASB Queue
3-38	Output class queues
39	RJE Queue
40-54	Input class Queues

To keep track of individual work queues, a control area in the job queue data set maintains a series of 36-byte minor queue control records (QCRs) -- one QCR for each work queue arrayed upon the job queue (see Figure JQDMP-5), plus a master QCR (see Figure JQDMP-6).

Offset Hex	Offset Dec				
0	0	Address of Last LTH of Highest Priority Entry on Queue	2	14	2
4	4	13	2	12	2
8	8	11	2	10	2
C	12	9	2	8	2
10	16	7	2	6	2
14	20	5	2	4	2
18	24	3	2	2	2
1C	28	1	2	0	2
20	32	Hold Queue ¦ Highest Priority	1	Address of ECB for First Task Requesting Work	3

Addresses of last LTH of l
entry having indicated pr

Figure JQDMP-5. Example of Minor Job Queue Control Record

Offset Hex	Offset Dec					
0	0	8-byte Disk Address of the Master QCR MBBCCHHR				
8	8	Reserved	1	Displacement of First Track of the Free Queue	2	Reserved
C	12	Number of Logical Tracks in the Job Queue Data Set	2	Number of Logical Tracks in the Free-track Queue		
10	16	Number of Tracks Reserved for Canceling of Job Steps When Queue Is Full	2	Number of Tracks Reserved for Any Initiator		
14	20	Displacement of Last Available Logical Track	2	Displacement of First Track Containing Only Job Queue Records		
18	24	Number of QCRs per Physical Track	2	Number of Job Queue Records per Physical Track		
1C	28	Number of Records per Logical Track	2	Number of Logical Tracks for Each Problem Program Partition		
20	32	Number of QCRs on the Mixed Track	2	Address of First Record on First Track Containing Only Job Queue Records		

Figure JQDMP-6. Master Job Queue Control Record Format

Using IMCJQDMP

To use IMCJQDMP, initial program load (IPL) the program from the card reader or from the tape unit on which a tape-copy of the deck is mounted. This is done by setting the LOAD UNIT dials on the console control panel to the unit address of the card reader or the tape drive, and depressing the LOAD key on the control panel. When loading has been accomplished, the program enters a wait state, indicated by the lighting of the WAIT light on the console. Pressing the console request or enter key at this point results in the console message:

```
IMC000A ENTER O=XXXD,Q=YYY(,S) OR PRESS
        INTERRUPT KEY FOR O=00E,Q=191
```

Message IMC000A is a request for parameters giving specifications for the desired dump. If the operator responds by depressing the external interrupt key without entering a device identification command through the console, the dump output will be written to the 1403 printer assigned device address 00E; input will be read from the direct access volume mounted on the disk drive assigned device address 191.

Device Identification Command

If the device identification command is entered, its format is:

```
O=xxxd,Q=yyy {,SELECT}
             {,S     }
```

where

O=xxxd

is the output address parameter;

Q=yyy

is the input address parameter;

SELECT (or S)

indicates that selective rather than full printing of the job queue is desired.

Output Address Parameter

The output address parameter may be omitted entirely. If it is, the output address will default to the 1403 printer at device address 00E. If the parameter is entered, it must precede the input address parameter. In making the entry,

xxx

is replaced with the address of the desired output device. Valid choices are the 1403 printer and the 2400 tape drive.

d

indicates the output device type. The character T is entered if a 2400 tape drive address has been specified in the xxx field. Example: O=182T.

The d field is omitted if output is to go to the 1403 printer. Example: O=00E.

Input Address Parameter

In the input address parameter,

yyy

is replaced with the address of the direct access device upon which the volume containing the SYS1.SYSJOBQE data set has been mounted.

Selective Dumping Parameters

If an entire job queue data set is to be dumped, the SELECT (or S) parameter is omitted from the device identification command.

If the SELECT (or S) parameter is included in the command, the program will issue the console message:

```
IMC001A SPECIFY SELECT PARAMETERS
```

and wait for a reply. The two valid parameters, QCR= and JOBNAME=, and their possible values are discussed separately.

QCR= Parameter

The QCR= parameter specifies that a particular work queue within the job queue data set is to be dumped. When this parameter is specified, the dump output listing will contain the data set's master queue control record and the queue records associated with the named work queue. The possible values for the QCR= parameter are:

$$\text{QCR=}\left\{\begin{array}{l}\text{ASB}\\ \text{CLASS=y}\\ \text{FREE}\\ \text{HOLD}\\ \text{RJE}\\ \text{SYSOUT=x}\\ \text{SUBMT}\end{array}\right\}$$

where:

y is replaced with one of the 15 input job class indicators, A through O, and

x is replaced with one of the 36 output class indicators, A through Z and 0 through 9.

If FREE is the value used, the master QCR and all logical tracks enqueued upon the free-track queue are dumped. The output listing for any of the other values will include the associated minor QCRs as well.

The values CLASS= and SYSOUT= must be completed with the system-assigned symbol of the particular input or output class desired. Examples:

```
QCR=SYSOUT=C
```

will result in a dump of the job queue's master QCR, the C-class output work queue's minor QCR, and the logical tracks assigned to the C-class output work queue.

```
QCR=RJE
```

will produce output consisting of the job queue's master QCR, the RJE work queue's minor QCR, and its assigned logical tracks.

JOBNAME= Parameter

The JOBNAME= parameter signifies to IMCJQDMP that the fifteen input work queues are to be searched for logical track areas assigned to the named job or jobs. Associated system message blocks and data set blocks will also be dumped. From one to four jobnames, enclosed in parentheses, may be specified in the value field of the parameter. Example:

```
JOBNAME=(TAX,NUMBER)
```

will produce a dump listing containing the assigned logical track areas, the system message blocks and the data set blocks, if any, associated with jobs named TAX and NUMBER, respectively.

Combining QCR= and JOBNAME= Parameters

The time required to search out the records associated with a particular job may be considerably reduced if the input class is known to the dump program. This passing of class information to IMCJQDMP may be effected by using the QCR= and JOBNAME= parameters in combination. For instance:

```
QCR=CLASS=G,JOBNAME=(LIST)
```

will cause only the class G input work queue to be searched for records concerned with the job named LIST.

Completion Message

After the selective dump parameters have been accepted, IMCJQDMP performs the requested task. When the operation has been completed, message IMC001A is reissued. Additional selective dump parameters may be entered if more information is desired. When all user requests have been fulfilled:

```
END
```

is entered through the console. The message:

```
IMC004I DUMP COMPLETED
```

is then issued. Note that when no selective dump parameters are entered, the program ends automatically after dumping the full job queue, issuing message IMC004I at completion of the operation.

Tape Output Processing

For magnetic tape output, IMCJQDMP creates 121-byte records, one record to a block. Each record contains a machine control character in its first byte.

Figure JQDMP-7 gives a sample of the job control statements needed to print IMCJQDMP 9-track tape output with the IEBPTPCH utility program.

```
//PRINT       JOB       MSGLEVEL=(1,1)
//STEP        EXEC      PGM=IEBPTPCH
//SYSPRINT    DD        SYSOUT=A
//SYSUT1      DD        UNIT=2400,LABEL=(,NL),VOL=SER=QDUMPT,
//                 DISP=(OLD,KEEP),DCB=(RECFM=F,BLKSIZE=121,LRECL=121)
//SYSUT2      DD        SYSOUT=A
//SYSIN       DD        *
         PRINT     PREFORM=M
/*
```

Figure JQDMP-7. Sample JCL Needed to Print 9-Track JQDMP Tape Output

For 7-track tape printing, there is an additional consideration. Initial program loading of IMCJQDMP generates a system reset which, on a 7-track control unit, has the following effect:

1. Mode is set to 800 bits per inch.

2. If the data conversion feature is present in the control unit, the data converter is turned on.

3. The translator is turned off.

4. Odd parity is established.

When the dump output has been written to 7-track tape, therefore, the following additional DCB parameters should be coded on the SYSUT1 DD statement for the IEBPTPCH utility if the data conversion feature exists:

DEN=2,TRTCH=C

If the data conversion feature is not included in the system, the TRTCH keyword must be omitted.

Standard Label Processing

For output to magnetic tape, IMCJQDMP automatically performs standard label processing; the user has no option to bypass this function. The extent of the label processing is confined to protecting security protected data sets and tapes with unexpired expiration dates; and, if multiple-volume output is produced, to maintaining standard labeled tapes, provided the first volume of IMCJQDMP output has standard labels.

When verifying that a mounted tape has standard labels, IMCJQDMP will read the labels (if present) in the density set for the tape drive; therefore, the user must be careful to ensure that the labels on the tape were recorded in the same density as the recording density set for the tape drive on which the tape is mounted. If the recording density for the drive is different from that of the tape, IMCJQDMP will assume that the tape has no labels, and will create non-labeled tape output.

Standard Labeled Output

If the user desires standard labeled output, he must mount a standard labeled tape. IMCJQDMP checks for an IBM standard volume label (VOL1) and the standard data set header label one (HDR1). Any user labels will be ignored and destroyed if the tape is eventually used for IMCJQDMP output. If the mounted tape contains a security protected data set, IMCJQDMP will request a new tape. If the expiration date in the HDR1 label has not occurred, IMCJQDMP will request permission to use the tape; if the operator's reply is negative (M), a new tape is requested. Otherwise, the tape will be used, and will contain standard labels, with the VOL1 label remaining the same as it was when the tape was mounted. The header and trailer labels will be created to be compatible with OS/360 standard labels, with a data set name of "JQDUMP."

Non-Labeled Output

If the user desires non-labeled tape output, the first volume mounted must be non-labeled. A non-labeled tape, to the IMCJQDMP program, is a tape that does not have a first record of 80 characters whose first four characters are equal to "VOL1." If the first record on the first volume is a standard volume label, processing as outlined in "Standard Labeled Output" will occur.

Abnormal Termination of JQDMP

It is conceivable that a condition can arise that will prevent IMCJQDMP from running to normal completion. Indeed, it may be the same error condition that caused the system to malfunction; that is, I/O error on the queue device, or invalid chaining of queue records. Under unrecoverable error conditions, the program comes to a halt in a wait state. The type of error encountered by the program may be determined by examination of the contents of the program status word (PSW) which was current at the time of the malfunction.

The PSW is a doubleword, having the following format:

Program Status Word

System Mask	Key	AMWP	Interruption Code
0 7	8 11	12 15	16 31

ILC	CC	Program Mask	Instruction Address
32 33	34 35	36 39	40 63

The publication IBM System/360 Principles of Operation, GA22-6821, gives a comprehensive description of each of the fields in the PSW. For the purpose of locating the cause of trouble in running IMCJQDMP, the user would be concerned mainly with the contents of the instruction address field, bits 40 through 63, in the event of a program check error, or with the interruption code, bits 16 through 31, if there has been an unrecoverable I/O error.

By displaying the contents of the instruction address register (IAR) on the system maintenance panel of the console, the address in main storage of the pertinent PSW can be obtained. The two low-order bytes of the IAR will be set according to the pattern:

0Dnn

where nn will contain the hexadecimal value of the location where the PSW was stored at the time the error condition was discovered.

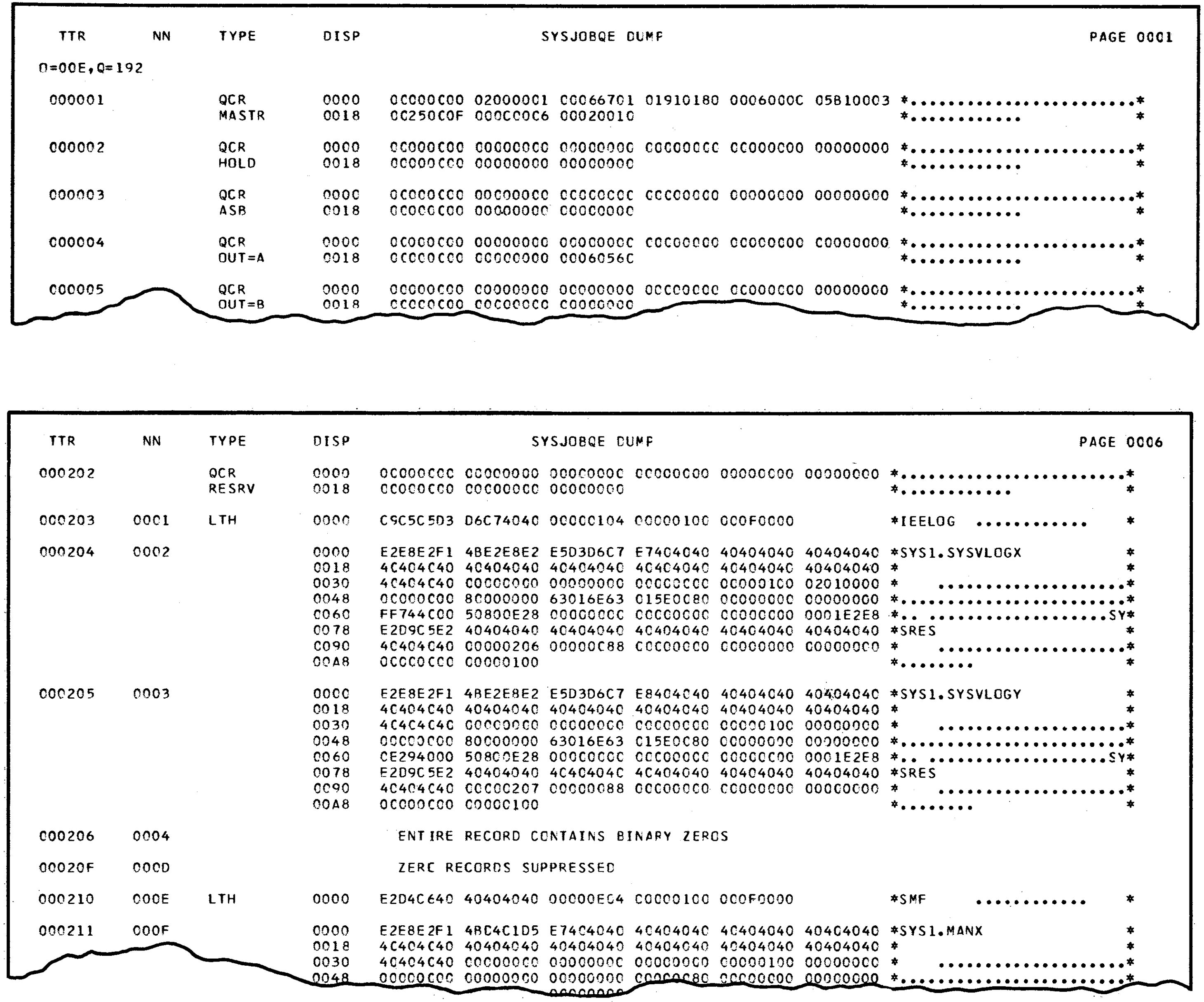
For example, should a program check occur, the IAR will be set to 0D28, indicating that the double word at location hex 28 will contain the Program Interrupt old PSW. (A note for users of doubleword fetch machines, such as M65 or M75: The IAR is updated by 8 after an interrupt, and this must be subtracted from the IAR setting to obtain the true location to be checked. In this example, for instance, the reading would be 0D30, and subtracting hex 8 would give the true 0D28 location.)

If the IAR display indicates 0D20, inspecting the interruption code in the PSW stored at hexadecimal location 20 will indicate the nature of the I/O error:

IC Content	Error Cause
x'00'	Channel end, device end, and unit check bits are all off in a stored channel status word (CSW).
x'02'	Invalid track-per-cylinder count in the format 4 DSCB (data set control block) of the queue volume.
x'03'	I/O error during write operation to output device or system console. The number of retries for recoverable tape I/O errors is set at 20.
x'20'	I/O error during read operation from SYS1/SYSJOBQE data set. The number of retries for recoverable DASD I/O errors is set at 16.
x'26'	I/O error during read operation from system console.

IMCJQDMP Output

IMCJQDMP dumps the contents of job queue records in hexadecimal representation, with six 4-byte words appearing in a line of printed output. In addition, translatable EBCDIC characters are printed in a one-character-per-byte format at the end of the printline. EBCDIC characters which cannot be interpreted in print are represented by periods. Record identification is shown on the sample listing page depicted in Figure JQDMP-8.



```
  TTR        NN      TYPE       DISP                  SYSJOBQE CUMP                                            PAGE 0001

 0=00E,Q=192

  000001             QCR        0000    0C000C00 02000001 C0066701 01910180 0006000C 05B10003 *........................*
                     MASTR      0018    0C250C0F 000CC0C6 00020010                            *............            *

  000002             QCR        0000    0C000C00 0000000C 00000000 C000000C 0C000C00 00000000 *........................*
                     HOLD       0018    0C000CC0 00000000 00000000                            *............            *

  000003             QCR        000C    0C000CC0 00C0000C CC0C0C0C CCC00000 00000000 00000000 *........................*
                     ASB        C018    0C0C0C00 00000000 C000000C                            *............            *

  000004             QCR        000C    0C000CC0 00000000 000C000C C0C00000 0C000C00 C0000000 *........................*
                     OUT=A      0018    0CC0C0CC CC000000 0006056C                            *............            *

  000005             QCR        0000    00000C00 C0000000 00000000 0CC0C0CC 0C000C00 00000000 *........................*
                     OUT=B      0018    CCC0C0C0 C0C0C0CC C0000000                            *............            *
```

```
  TTR        NN      TYPE       DISP                  SYSJOBQE CUMP                                            PAGE 0006

 000202             QCR        0000    0C0000CC C0000000 000C000C CC000C00 00000C00 00000000 *........................*
                    RESRV      0018    0C0C0C00 C0C0000C 00000000                            *............            *

 000203     0001    LTH        0000    C9C5C5D3 D6C74040 0000C104 00000100 000F0000          *IEELOG  ............    *

 000204     0002               0000    E2E8E2F1 4BE2E8E2 E5D3D6C7 E7404040 40404040 40404040 *SYS1.SYSVLOGX           *
                               0018    4C404C40 40404040 40404040 40404040 4040404C 40404040 *                        *
                               0030    4C404C40 C0000000 00000000 0000C0CC 0C000100 02010000 *    ....................*
                               0048    0C000C00 80000000 63016E63 015E0C80 0C00000C 00000000 *........................*
                               C060    FF744C00 50800E28 0000000C C0000000 C0000C00 0001E2E8 *.. ....................SY*
                               0078    E2D9C5E2 40404040 40404040 40404040 40404040 40404040 *SRES                    *
                               C090    4C404C40 00000206 00000C88 C0C000C0 C0000000 0000000C *    ....................*
                               00A8    0CCC0CCC 00000100                                     *........                *

 000205     0003               000C    E2E8E2F1 4BE2E8E2 E5D3D6C7 E8404040 40404040 40404040 *SYS1.SYSVLOGY           *
                               0018    4C404C40 40404040 40404040 40404040 40404040 40404040 *                        *
                               0030    4C4C4C4C 00C0000C 0000000C C0C0C0C0 C000010C 00000000 *    ....................*
                               0048    00C00C00 80000000 63016E63 C15E0C80 C0000000 00000000 *........................*
                               0060    CE294000 508C0E28 000C0CCC CCC0000C C0000C00 0001E2E8 *.. ....................SY*
                               0078    E2D9C5E2 40404040 4C40404C 4C404040 40404040 40404040 *SRES                    *
                               0C90    4C404C40 CCC00207 00000088 0CC000C0 C0000000 00000000 *    ....................*
                               00A8    0C000C00 C0000100                                     *........                *

 000206     0004                          ENTIRE RECORD CONTAINS BINARY ZEROS

 00020F     000D                          ZERC RECORDS SUPPRESSED

 000210     000E    LTH        0000    E2D4C640 40404040 00000EC4 C0000100 000F0000          *SMF     ............    *

 000211     000F               0000    E2E8E2F1 4BC4C1D5 E74C4040 4C40404C 4C404040 40404040 *SYS1.MANX               *
                               0018    4C404C40 40404040 40404040 40404040 40404040 40404040 *                        *
                               0030    40404C40 C0C0000C 0000000C 00000000 C0000100 0000000C *    ....................*
                               0048    00000C00 00000000 00000000 C0000C80 C0000000 00000000 *........................*
                                                        00000000
```

Figure JQDMP-8. Sample of IMCJQDMP Output Listing

Record Identification

Record identification on the listing includes:

TTR

The direct access address, relative to the beginning of SYS1.SYSJOBQE, is supplied for both QCR and logical track records.

NN

Supplied for logical track records only, this address is a binary number assigned relative to the beginning of the specific work queue in which the printed record resides. Starting with an assignment of 1 for the first logical track header allotted to the queue, the NN address increases by 1 for each additional record in the work queue.

TYPE

Figure JQDMP-9 lists the types of queue records dumped by IMCJQDMP, and the listing identification given to each recognizable type. QCRs and LTHs are identified through their position in the job queue's structure. Identification for records from the logical track area is obtained from the ID field, hexadecimal offset 03 (byte 4) of each record. Recognizable ID values are listed in the figure. Unidentifiable nonzero records -- the job file control block (JFCB), job file control block extension (JFCBX), and system output class directory (SCD) -- are printed without type labeling.

QCR ID

Each queue control record is further labeled with the name of the unique work queue with which the QCR is associated. Figure JQDMP-10 lists the identification given by IMCJQDMP to each work queue type.

DISP

Indicates the displacement, or position, within a queue record of the next hexadecimal word to be printed on the listing. The first word of the first printed line for a given record is at displacement 0000; the first word of the second printed line, if one exists, is displacement 0018 hex (24 decimal).

Hex ID Value	Output Type ID	Job Queue Record
01	ACT	Account Control Table
15	DSB	Data Set Block
0F	DSENQ	Data Set Enqueue Table
07	DSNT	Data Set Name Table
00	JCT	Job Control Table
		Job File Control Block (JFCB)
		Job File Control Block Extension (JFCBX)
	LTH	Logical Track Header
0A	POT	Procedure Override Table
	QCR	Queue Control Record
		System Output Class Directory (SCD)
02	SCT	Step Control Table
0C	SCTX	Step Control Table Extension
03	SIOT	Step Input Output Table
05	SMB	System Message Block
06	VOLT	Volume Table

Figure JQDMP-9. Queue Record Type Identification

Output QCR ID	Corresponding Work Queue
ASB	Automatic SYSIN Batching Queue.
CLS=y	System Input Job Class Queues; y is replaced with the appropriate class, A-O.
HOLD	Hold queue.
MASTR	Master QCR.
OUT=x	System Output Job Class Queues; x is replaced with the appropriate class, A-Z or 0-9.
RESRV	Reserved QCRs.
RJE	Remote Job Entry Queue.
SUBMT	TSO Background Reader Queue.

Figure JQDMP-10. Queue Control Record Identification

Zero Records in the Dump

Records in each logical track are read and dumped sequentially. When a record in the logical track area contains only binary zeroes, its TTR and NN positions are given, but the record is not dumped. The notation:

```
ENTIRE RECORD CONTAINS BINARY ZEROES
```

is printed on the listing. A second contiguous zero-filled record would be similarly treated. But when three or more contiguous zero-filled records are encountered, only the first is treated as outlined above. Subsequent records are bypassed until a nonzero record or a logical track header, whichever occurs first, is encountered. Then the TTR and NN of the last zero-filled record and the listing message:

```
ZERO RECORDS SUPPRESSED
```

are printed. The number of suppressed records may be computed by subtracting the NN of the first such record from that of the last.

Contents of the Dump Listing

If an entire job queue is being dumped, the output listing is produced in two sections. The first contains all queue control records; the second, the logical track area records.

If selective dumping of a job queue data set is stipulated, the program prints the specified parameters on the top of an output page, then follows with the appropriate QCRs and logical track area records. When particular job names are given as the selective dump parameters (see "Using The Job Queue Dump Program"), the records associated with each job are collected and printed under the given name. Each data set block (DSB) is printed immediately following the related step input/output table (SIOT) and labeled as such. The system message block (SMB) chain is printed as the last records for a given job.

Operational Considerations

- The time required to produce a full job queue dump is dependent upon space allocated to the SYS1.SYSJOBQE data set. The time required for this stand-alone operation may be reduced by using the tape output option of the program. In this way, the operating system may be more quickly brought back into service and the queue dump tape printed with a system utility program such as IEBPTPCH. Figure JQDMP-11 shows the execution time difference between tape and printer output for various queue devices.

	Output Device	
Queue Device	Printer (1403)	Tape (2400)
2311	11.3 minutes	4.0 minutes
2314	19.5 minutes	6.9 minutes
2301	49.5 minutes	17.4 minutes

Figure JQDMP-11. IMCJQDMP Execution Time per 100 Tracks of Input, As a Function of the Output Device

Chapter 5: IMBLIST

Formats and prints object modules, load modules, and CSECT identification records.

LIST

Contents

INTRODUCTION 107

FEATURES 108

EXECUTING LIST 109
Listing a Load Module 109
Listing an Object Module 110
Listing CSECT Identification Records 111

OUTPUT 112

EXAMPLES 119
Example 1: Listing Several Object Modules 119
Example 2: Using the LISTLOAD Control Statement 120
Example 3: Using the LISTIDR Control Statements 121
Example 4: Verifying an Object Deck 122
Example 5: Combining LISTOBJ, LISTLOAD, and LISTIDR 123

Figures

Figure LIST-1. Sample Module Summary for LISTLOAD 112
Figure LIST-2. Sample LISTLOAD Output - Load Module Map (Part 1 of 2) 113
Figure LIST-2. Sample LISTLOAD Output - Load Module Map (Part 2 of 2) 114
Figure LIST-3. Sample LISTLOAD Output - Cross-Reference Listing (Part 1 of 2) 115
Figure LIST-3. Sample LISTLOAD Output - Cross-Reference Listing (Part 2 of 2) 116
Figure LIST-4. Sample LISTOBJ Output 117
Figure LIST-5. Sample LISTIDR Output 118

Introduction

IMBLIST is a service aid that operates as a problem program under the IBM System/360 Operating System. It produces the following kinds of output that can help you debug complex programs:

- A formatted listing of an object module.
- A formatted listing of a load module.
- A load module cross reference listing.
- A formatted listing of all information in a load module's CSECT identification records (IDRs).
- A listing of all program modifications for a load module or library.

LIST

Features

IMBLIST can help you solve programming problems in several ways.

If you want to verify an object module, you can use IMBLIST to obtain a formatted listing of it. The listing contains SYM records produced by TESTRAN (if there are any), the external symbol dictionary (ESD), the relocation dictionary (RLD), the text of the program containing instructions and data, and the END record.

If you are interested in the relationships of control sections in a load module, you can use IMBLIST to get a listing of the load module along with its module map and cross-reference listing. You can then examine the control sections in the load module, the overlay structure, and the cross-references for each control section.

If you want to trace modifications to the executable code in a control section, you can use IMBLIST to produce a formatted listing of all information in the load module's CSECT identification records (IDRs). An IDR provides the following information:

- It identifies the version and modification level of the language translator and the date that each control section was translated. (Translation data is available only for control sections that were produced by a translator that supports IDR generation.)

- It identifies the version and modification level of the linkage editor that built the load module and gives the date the load module was created.

- It identifies by date modifications to the load module performed by IMASPZAP.

An IDR also may contain optional user-supplied data associated with the executable code of the control sections.

Executing IMBLIST

You control IMBLIST processing by supplying control statements in the input stream. You must code the control statements according to the following rules:

- Leave column 1 blank, unless you want to supply an optional symbolic name. A symbolic name must be terminated by one or more blanks.
- If a complete control statement will not fit on a single card, end the first card with a comma and continue on the next card. Begin all continuation cards in columns 2 - 16. You must not split parameters between two cards; the only exception is the MEMBER parameters, which may be split at any internal comma.

Listing a Load Module

Use the LISTLOAD control statement to get a formatted listing of a load module. The format of this statement is:

```
LISTLOAD  [OUTPUT={MODLIST}] [,TITLE=('title',position)]
          [       { XREF  }]
          [       { BOTH  }]
          [,DDN=ddname] [,MEMBER={(list,...) }]
                        [        { membername}]
```

The parameters of the LISTLOAD control statement are as follows:

OUTPUT=type

specifies the type of load module listing to be produced. OUTPUT=MODLIST requests a formatted listing of the control and text records of a load module, including its External Symbol Dictionary and Relocation Dictionary Records. OUTPUT=XREF requests a module map and cross-reference listing for the load module. OUTPUT=BOTH requests both a formatted listing of the load module and its map and cross-references. If this parameter is omitted, OUTPUT=BOTH will be assumed.

TITLE=('title',position)

specifies a title, from one to forty characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position subparameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title will be printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. Note: Do not punctuate your title with commas, since IMBLIST recognizes these as delimiters. Anything that follows an embedded comma in a title will be ignored.

DDN=ddname

identifies the DD statement that defines the data set containing the input module. If the DDN= parameter is omitted, IMBLIST will assume SYSLIB as the default ddname.

```
MEMBER= {(member1,...membern)}
        {member              }
```

identifies the input load module(s) by membername or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, IMBLIST will print all modules in the data set.

Listing an Object Module

Use the LISTOBJ control statement to obtain a listing of an object module. The format of this control statement is:

```
LISTOBJ    [TITLE=('title',position)]
           [,DDN=ddname] [,MEMBER= {(member1,...membern)}]
                         [         {  member             }]
```

TITLE=('title',position)

specifies a title, from one to forty characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position parameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title will be printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. Note: Do not punctuate your title with commas, since IMBLIST recognizes these as delimiters. Anything that follows an embedded comma in a title will be ignored.

DDN=ddname

identifies the DD statement that defines the data set containing the input module. If the DDN= parameter is omitted, IMBLIST will assume SYSLIB as the default ddname.

```
MEMBER= {(member1,...membern)}
        {member              }
```

identifies the input object module(s) by membername or alias name. To specify more than one object module, enclose the list of names in parentheses and separate the names with commas. CAUTION: You must include the MEMBER= parameter if the input object modules exist as members in a partitioned data set. If you do not include the MEMBER= parameter, IMBLIST will assume that the input data set is organized sequentially, and that it contains a single, continuous object module.

Listing CSECT Identification Records

Use the LISTIDR control statement to get a formatted listing of a module's CSECT identification record (IDR). The format is:

```
LISTIDR  [OUTPUT={IDENT}][,TITLE=('title',position)]
         [       { ALL }]
          [,DDN=ddname] [,MEMBER={(member1,...membern)}]
                        [        {  member            }]
```

OUTPUT= type

specifies whether IMBLIST should print all CSECT identification records or only those containing IMASPZAP data and user data. If you specifiy OUTPUT=ALL, all IDRs associated with the module will be printed. If you specify OUTPUT=IDENT, IMBLIST will print only those IDRs that contain IMASPZAP data or user-supplied data. If you omit this parameter, IMBLIST will assume a default of OUTPUT=ALL.

TITLE=('title',position)

specifies a title, from one to forty characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position parameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title will be printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. Note: Do not punctuate your title with commas, since IMBLIST recognizes these as delimiters. Anything that follows an embedded comma in a title will be ignored.

LIST

DDN=ddname

identifies the DD statement that defines the data set containing the input module. If you omit the DDN= parameter, IMBLIST will assume SYSLIB as the default ddname.

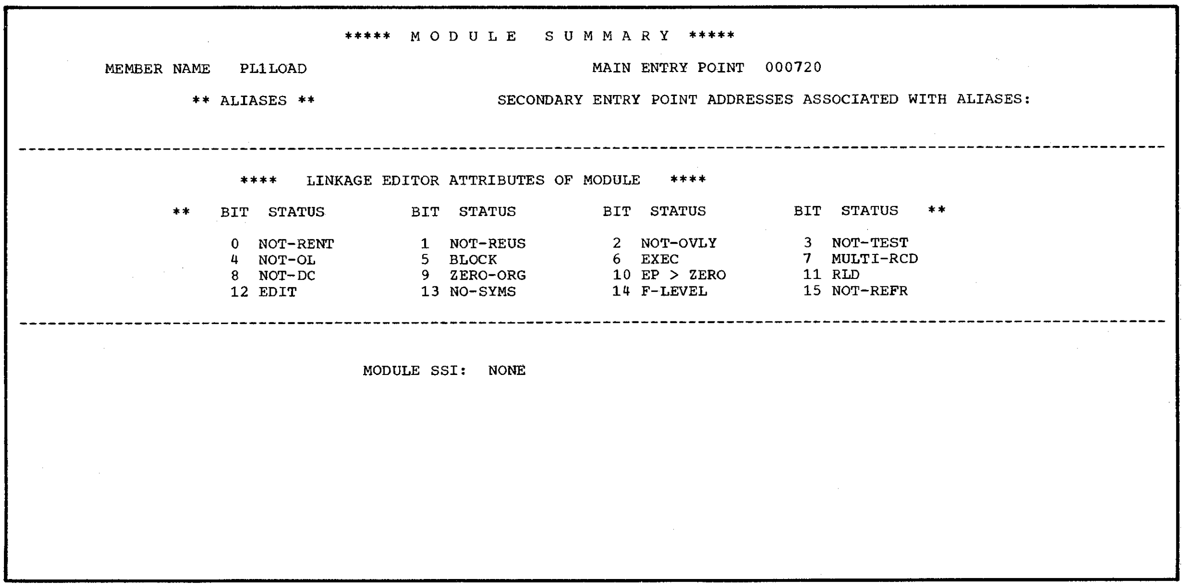
```
MEMBER={(member1,...membern)}
       {member              }
```

identifies the input load module(s) by membername or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, IMBLIST will print all modules in the data set.

Output

IMBLIST produces a separate listing for each control statement that you specify. The first page of each listing always shows the control statement as you entered it. The second page of the listing is a module summary, unless you requested LISTOBJ; in that case, no module summary will be produced, and the second page of the listing will be the beginning of the formatted output.

The module summary gives the member name (with aliases), the entry point, the linkage editor attributes, and system status index information (SSI) for the module being formatted. Figure LIST-1 shows a typical module summary.



```
                         *****  M O D U L E   S U M M A R Y  *****

     MEMBER NAME   PL1LOAD                            MAIN ENTRY POINT  000720

           ** ALIASES **                  SECONDARY ENTRY POINT ADDRESSES ASSOCIATED WITH ALIASES:

------------------------------------------------------------------------------------------------------

                  ****   LINKAGE EDITOR ATTRIBUTES OF MODULE   ****

          **   BIT  STATUS         BIT  STATUS         BIT  STATUS         BIT  STATUS    **

                0  NOT-RENT         1  NOT-REUS         2  NOT-OVLY         3  NOT-TEST
                4  NOT-OL           5  BLOCK            6  EXEC             7  MULTI-RCD
                8  NOT-DC           9  ZERO-ORG        10  EP > ZERO       11  RLD
               12  EDIT            13  NO-SYMS         14  F-LEVEL         15  NOT-REFR
------------------------------------------------------------------------------------------------------

                        MODULE SSI:   NONE
```

Figure LIST-1. Sample Module Summary for LISTLOAD

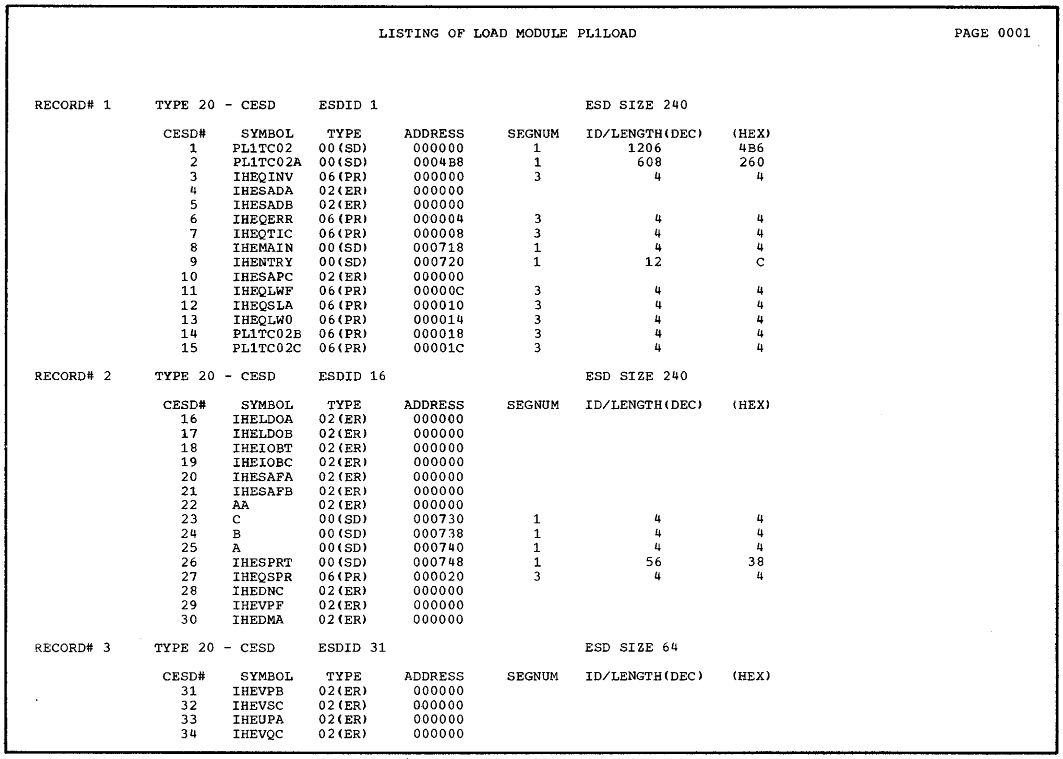
The third page of the listing (or, for LISTOBJ, the second page) is the beginning of the formatted output itself.

For LISTLOAD, this consists of the load module and/or the module map and cross-reference listing. Figure LIST-2 shows an example of LISTLOAD module map output. Figure LIST-3 shows an example of the cross-reference listing for the same module.

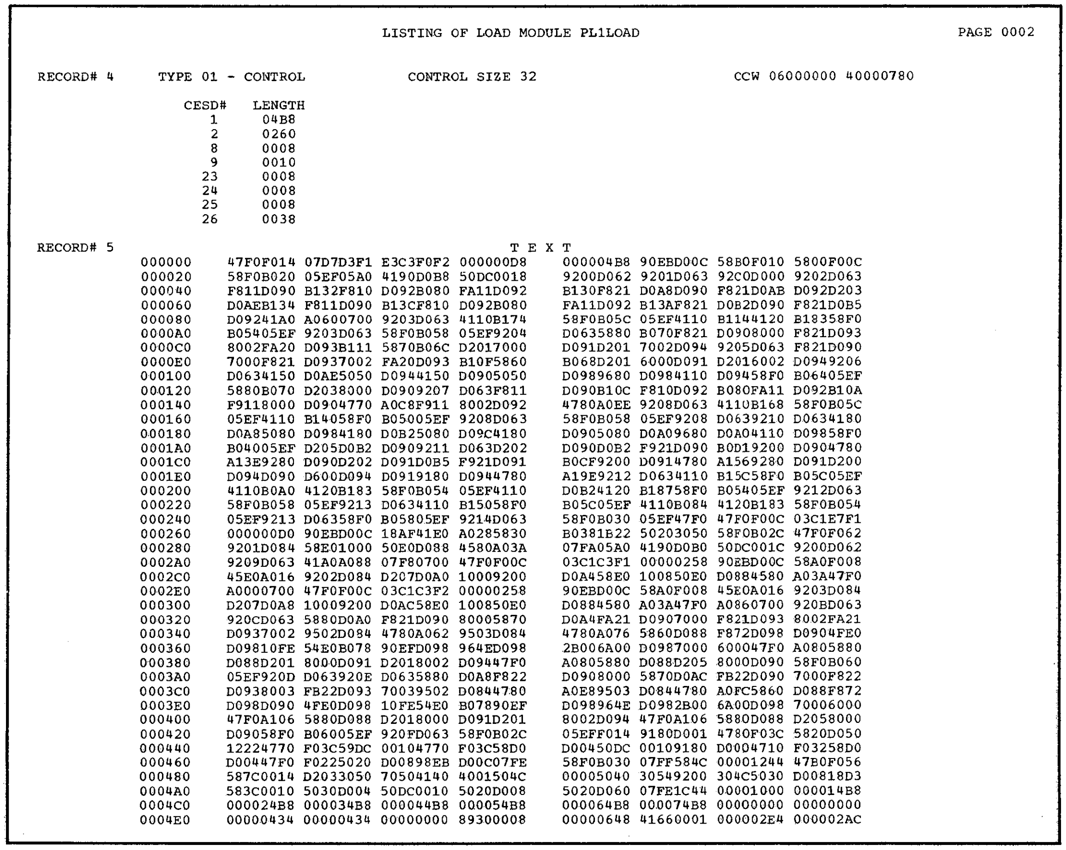
For LISTOBJ, the body of the listing consists of the object module listing, the module's external symbol dictionary, and its relocation dictionary. Figure LIST-4 shows an example of LISTOBJ output.

For LISTIDR, the third page of the listing begins a complete list of all CSECT identification records for the module. Figure LIST-5 shows an example of LISTIDR output.

Complete descriptions of the fields in the formatted output listings can be found in the publication IBM System/360 Operating System: Linkage Editor (E) Program Logic Manual, GY28-6610, and Linkage Editor (F) Program Logic Manual, GY28-6667.

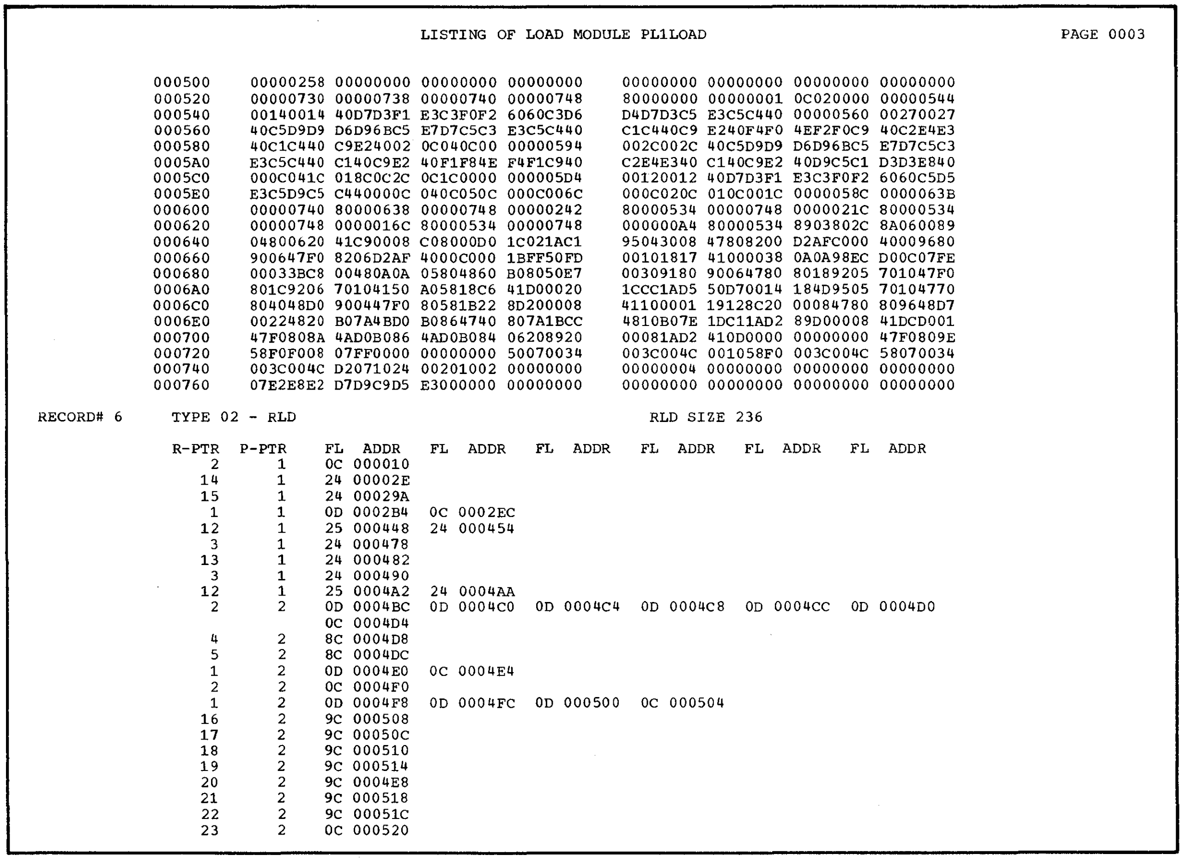


```
                                  LISTING OF LOAD MODULE PL1LOAD                                    PAGE 0001



RECORD# 1     TYPE 20 - CESD     ESDID 1                           ESD SIZE 240

               CESD#     SYMBOL    TYPE      ADDRESS      SEGNUM   ID/LENGTH(DEC)    (HEX)
                  1      PL1TC02   00(SD)     000000         1          1206          4B6
                  2      PL1TC02A  00(SD)     0004B8         1           608          260
                  3      IHEQINV   06(PR)     000000         3             4            4
                  4      IHESADA   02(ER)     000000
                  5      IHESADB   02(ER)     000000
                  6      IHEQERR   06(PR)     000004         3             4            4
                  7      IHEQTIC   06(PR)     000008         3             4            4
                  8      IHEMAIN   00(SD)     000718         1             4            4
                  9      IHENTRY   00(SD)     000720         1            12            C
                 10      IHESAPC   02(ER)     000000
                 11      IHEQLWF   06(PR)     00000C         3             4            4
                 12      IHEQSLA   06(PR)     000010         3             4            4
                 13      IHEQLW0   06(PR)     000014         3             4            4
                 14      PL1TC02B  06(PR)     000018         3             4            4
                 15      PL1TC02C  06(PR)     00001C         3             4            4

RECORD# 2     TYPE 20 - CESD     ESDID 16                          ESD SIZE 240

               CESD#     SYMBOL    TYPE      ADDRESS      SEGNUM   ID/LENGTH(DEC)    (HEX)
                 16      IHELDOA   02(ER)     000000
                 17      IHELDOB   02(ER)     000000
                 18      IHEIOBT   02(ER)     000000
                 19      IHEIOBC   02(ER)     000000
                 20      IHESAFA   02(ER)     000000
                 21      IHESAFB   02(ER)     000000
                 22      AA        02(ER)     000000
                 23      C         00(SD)     000730         1             4            4
                 24      B         00(SD)     000738         1             4            4
                 25      A         00(SD)     000740         1             4            4
                 26      IHESPRT   00(SD)     000748         1            56           38
                 27      IHEQSPR   06(PR)     000020         3             4            4
                 28      IHEDNC    02(ER)     000000
                 29      IHEVPF    02(ER)     000000
                 30      IHEDMA    02(ER)     000000

RECORD# 3     TYPE 20 - CESD     ESDID 31                          ESD SIZE 64

               CESD#     SYMBOL    TYPE      ADDRESS      SEGNUM   ID/LENGTH(DEC)    (HEX)
                 31      IHEVPB    02(ER)     000000
                 32      IHEVSC    02(ER)     000000
                 33      IHEUPA    02(ER)     000000
                 34      IHEVQC    02(ER)     000000
```

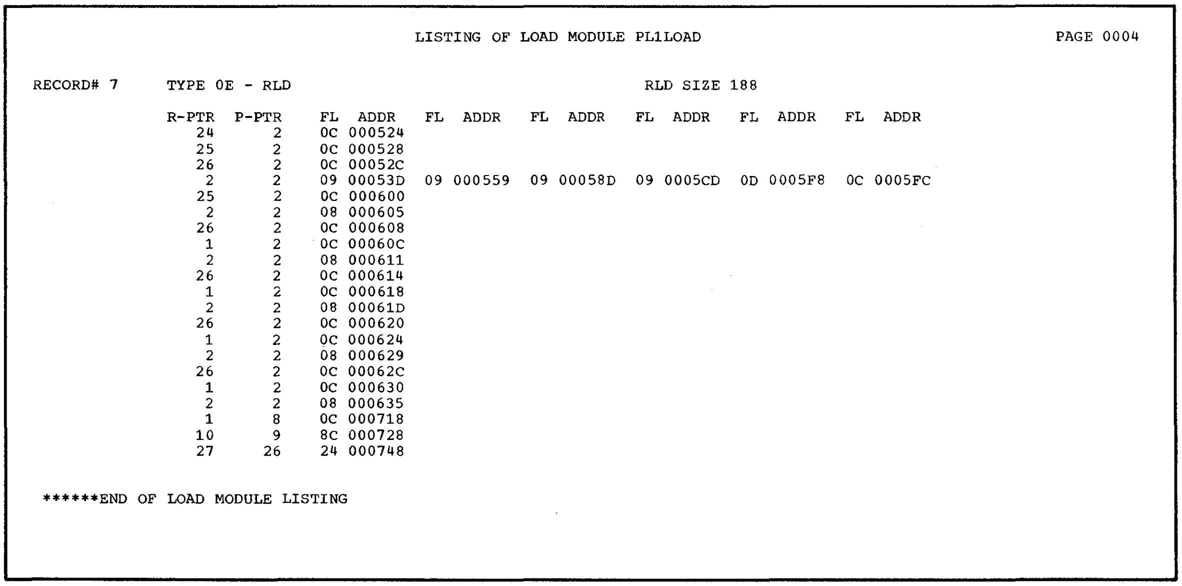


```
                                  LISTING OF LOAD MODULE PL1LOAD                                    PAGE 0002


RECORD# 4     TYPE 01 - CONTROL            CONTROL SIZE 32                          CCW 06000000 40000780

                  CESD#   LENGTH
                     1     04B8
                     2     0260
                     8     0008
                     9     0010
                    23     0008
                    24     0008
                    25     0008
                    26     0038

RECORD# 5                                               T E X T
             000000     47F0F014 07D7D3F1 E3C3F0F2 000000D8    000004B8 90EBD00C 58B0F010 5800F00C
             000020     58F0B020 05EF05A0 4190D0B8 50DC0018    9200D062 9201D063 92C0D000 9202D063
             000040     F811D090 B132F810 D092B080 FA11D092    B130F821 D0A8D090 F821D0AB D092D203
             000060     D0AEB134 F811D090 B13CF810 D092B080    FA11D092 B13AF821 D0B2D090 F821D0B5
             000080     D09241A0 A0600700 9203D063 4110B174    58F0B05C 05EF4110 B1144120 B18358F0
             0000A0     B05405EF 9203D063 58F0B058 05EF9204    D0635880 B070F821 D0908000 F821D093
             0000C0     8002FA20 D093B111 5870B06C D2017000    D091D201 7002D094 9205D063 F821D090
             0000E0     7000F821 D0937002 FA20D093 B10F5860    B068D201 6000D091 D2016002 D0949206
             000100     D0634150 D0AE5050 D0944150 D0905050    D0989680 D0984110 D09458F0 B06405EF
             000120     5880B070 D2038000 D0909207 D063F811    D090B10C F810D092 B080FA11 D092B10A
             000140     F9118000 D0904770 A0C8F911 8002D092    4780A0EE 9208D063 4110B168 58F0B05C
             000160     05EF4110 B14058F0 B05005EF 9208D063    58F0B058 05EF9208 D0639210 D0634180
             000180     D0A85080 D0984180 D0B25080 D09C4180    D0905080 D0A09680 D0A04110 D09858F0
             0001A0     B04005EF D205D0B2 D0909211 D063D202    D090D0B2 F921D090 B0D19200 D0904780
             0001C0     A13E9280 D090D202 D091D0B5 F921D091    B0CF9200 D0914780 A1569280 D091D200
             0001E0     D094D090 D600D094 D0919180 D0944780    A19E9212 D0634110 B15C58F0 B05C05EF
             000200     4110B0A0 4120B183 58F0B054 05EF4110    D0B24120 B18758F0 B05405EF 9212D063
             000220     58F0B058 05EF9213 D0634110 B15058F0    B05C05EF 4110B084 4120B183 58F0B054
             000240     05EF9213 D06358F0 B05805EF 9214D063    58F0B030 05EF47F0 47F0F00C 03C1E7F1
             000260     000000D0 90EBD00C 18AF41E0 A0285830    B0381B22 50203050 58F0B02C 47F0F062
             000280     9201D084 58E01000 50E0D088 4580A03A    07FA05A0 4190D0B0 50DC001C 9200D062
             0002A0     9209D063 41A0A088 07F80700 47F0F00C    03C1C3F1 00000258 90EBD00C 58A0F008
             0002C0     45E0A016 9202D084 D207D0A0 10009200    D0A458E0 100850E0 D0884580 A03A47F0
             0002E0     A0000700 47F0F00C 03C1C3F2 00000258    90EBD00C 58A0F008 45E0A016 9203D084
             000300     D207D0A8 10009200 D0AC58E0 100850E0    D0884580 A03A47F0 A0860700 920BD063
             000320     920CD063 5880D0A0 F821D090 80005870    D0A4FA21 D0907000 F821D093 8002FA21
             000340     D0937002 9502D084 4780A062 9503D084    4780A076 5860D088 F872D098 D0904FE0
             000360     D09810FE 54E0B078 90EFD098 964ED098    2B006A00 D0987000 600047F0 A0805880
             000380     D088D201 8000D091 D2018002 D09447F0    A0805880 D088D205 8000D090 58F0B060
             0003A0     05EF920D D063920E D0635880 D0A8F822    D0908000 5870D0AC FB22D090 7000F822
             0003C0     D0938003 FB22D093 70039502 D0844780    A0E89503 D0844780 A0FC5860 D088F872
             0003E0     D098D090 4FE0D098 10FE54E0 B07890EF    D098964E D0982B00 6A00D098 70006000
             000400     47F0A106 5880D088 D2018000 D091D201    8002D094 47F0A106 5880D088 D2058000
             000420     D09058F0 B06005EF 920FD063 58F0B02C    05EFF014 9180D001 4780F03C 5820D050
             000440     12224770 F03C59DC 00104770 F03C58D0    D00450DC 00109180 D0004710 F03258D0
             000460     D00447F0 F0225020 D00898EB D00C07FE    58F0B030 07FF584C 00001244 47B0F056
             000480     587C0014 D2033050 70504140 4001504C    00005040 30549200 304C5030 D00818D3
             0004A0     583C0010 5030D004 50DC0010 5020D008    5020D060 07FE1C44 00001000 000014B8
             0004C0     000024B8 000034B8 000044B8 000054B8    000064B8 000074B8 00000000 00000000
             0004E0     00000434 00000434 00000000 89300008    00000648 41660001 000002E4 000002AC
```

Figure LIST-2. Sample LISTLOAD Output - Load Module Map (Part 1 of 2)



```
                                         LISTING OF LOAD MODULE PL1LOAD                                      PAGE 0003

            000500    00000258 00000000 00000000 00000000    00000000 00000000 00000000 00000000
            000520    00000730 00000738 00000740 00000748    80000000 00000001 0C020000 00000544
            000540    00140014 40D7D3F1 E3C3F0F2 6060C3D6    D4D7D3C5 E3C5C440 00000560 00270027
            000560    40C5D9D9 D6D96BC5 E7D7C5C3 E3C5C440    C1C440C9 E240F4F0 4EF2F0C9 40C2E4E3
            000580    40C1C440 C9E24002 0C040C00 00000594    002C002C 40C5D9D9 D6D96BC5 E7D7C5C3
            0005A0    E3C5C440 C140C9E2 40F1F84E F4F1C940    C2E4E340 C140C9E2 40D9C5C1 D3D3E840
            0005C0    000C041C 018C0C2C 0C1C0000 000005D4    00120012 40D7D3F1 E3C3F0F2 6060C5D5
            0005E0    E3C5D9C5 C440000C 040C050C 000C006C    000C020C 010C001C 0000058C 0000063B
            000600    00000740 80000638 00000748 00000242    80000534 00000748 0000021C 80000534
            000620    00000748 0000016C 80000534 00000748    000000A4 80000534 8903802C 8A060089
            000640    04800620 41C90008 C08000D0 1C021AC1    95043008 47808200 D2AFC000 40009680
            000660    900647F0 8206D2AF 4000C000 1BFF50FD    00101817 41000038 0A0A98EC D00C07FE
            000680    00033BC8 00480A0A 05804860 B08050E7    00309180 90064780 80189205 701047F0
            0006A0    801C9206 70104150 A05818C6 41D00020    1CCC1AD5 50D70014 184D9505 70104770
            0006C0    804048D0 900447F0 80581B22 8D200008    41100001 19128C20 00084780 809648D7
            0006E0    00224820 B07A4BD0 B0864740 807A1BCC    4810B07E 1DC11AD2 89D00008 41DCD001
            000700    47F0808A 4AD0B086 4AD0B084 06208920    00081AD2 410D0000 00000000 47F0809E
            000720    58F0F008 07FF0000 00000000 50070034    003C004C 001058F0 003C004C 58070034
            000740    003C004C D2071024 00201002 00000000    00000004 00000000 00000000 00000000
            000760    07E2E8E2 D7D9C9D5 E3000000 00000000    00000000 00000000 00000000 00000000

RECORD# 6     TYPE 02 - RLD                                    RLD SIZE 236

              R-PTR  P-PTR    FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR
                  2      1    0C 000010
                 14      1    24 00002E
                 15      1    24 00029A
                  1      1    0D 0002B4  0C 0002EC
                 12      1    25 000448  24 000454
                  3      1    24 000478
                 13      1    24 000482
                  3      1    24 000490
                 12      1    25 0004A2  24 0004AA
                  2      2    0D 0004BC  0D 0004C0  0D 0004C4  0D 0004C8  0D 0004CC  0D 0004D0
                              0C 0004D4
                  4      2    8C 0004D8
                  5      2    8C 0004DC
                  1      2    0D 0004E0  0C 0004E4
                  2      2    0C 0004F0
                  1      2    0D 0004F8  0D 0004FC  0D 000500  0C 000504
                 16      2    9C 000508
                 17      2    9C 00050C
                 18      2    9C 000510
                 19      2    9C 000514
                 20      2    9C 0004E8
                 21      2    9C 000518
                 22      2    9C 00051C
                 23      2    0C 000520
```



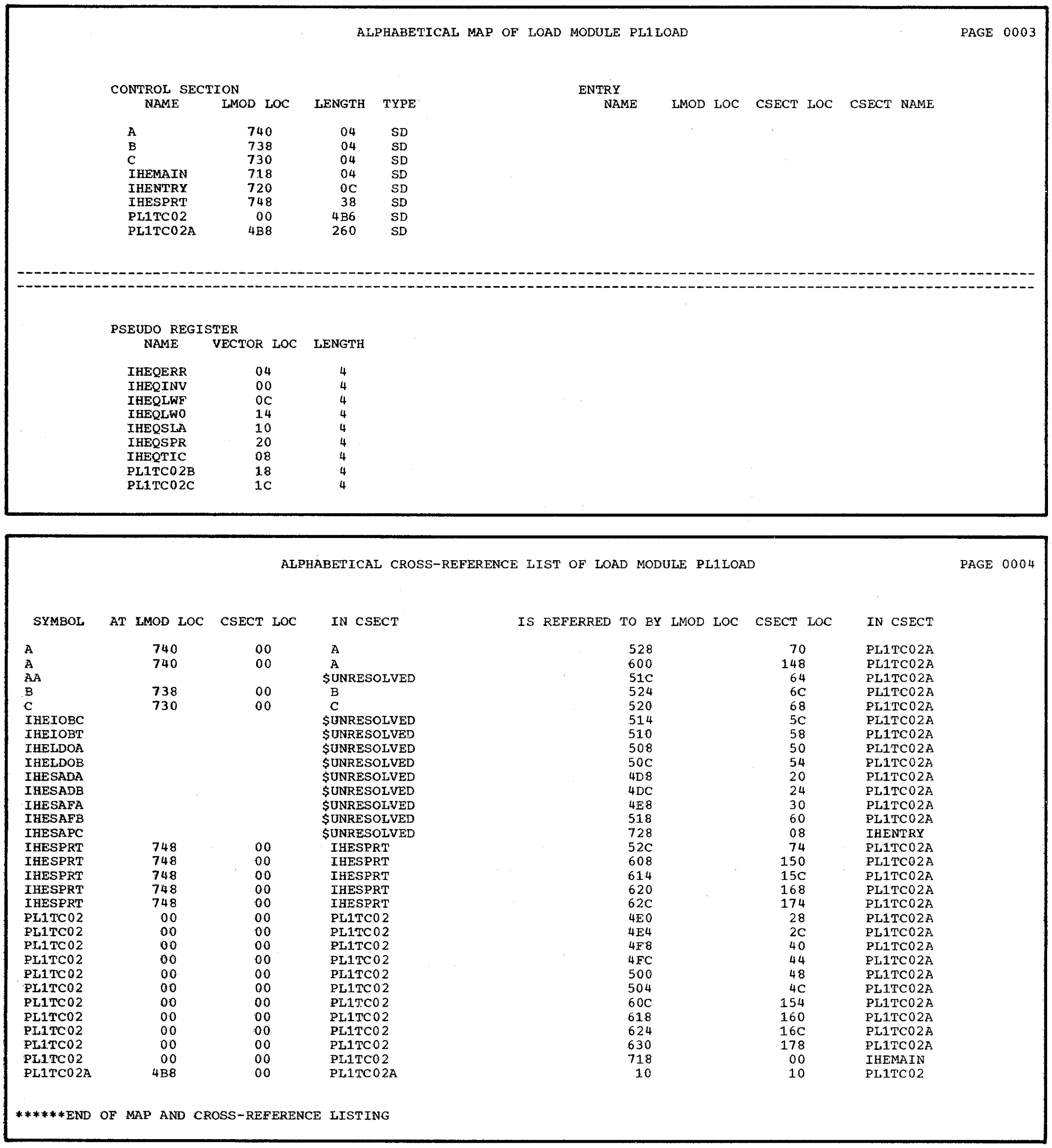
```
                                         LISTING OF LOAD MODULE PL1LOAD                                      PAGE 0004

RECORD# 7     TYPE 0E - RLD                                    RLD SIZE 188

              R-PTR  P-PTR    FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR   FL  ADDR
                 24      2    0C 000524
                 25      2    0C 000528
                 26      2    0C 00052C
                  2      2    09 00053D  09 000559  09 00058D  09 0005CD  0D 0005F8  0C 0005FC
                 25      2    0C 000600
                  2      2    08 000605
                 26      2    0C 000608
                  1      2    0C 00060C
                  2      2    08 000611
                 26      2    0C 000614
                  1      2    0C 000618
                  2      2    08 00061D
                 26      2    0C 000620
                  1      2    0C 000624
                  2      2    08 000629
                 26      2    0C 00062C
                  1      2    0C 000630
                  2      2    08 000635
                  1      8    0C 000718
                 10      9    8C 000728
                 27     26    24 000748


 ******END OF LOAD MODULE LISTING
```

Figure LIST-2. Sample LISTLOAD Output - Load Module Map (Part 2 of 2)

```
                        NUMERICAL MAP AND CROSS-REFERENCE LIST OF LOAD MODULE PL1LOAD                    PAGE 0001


                CONTROL SECTION                                           ENTRY
                 LMOD LOC     NAME      LENGTH  TYPE                       LMOD LOC  CSECT LOC      NAME

                       00   PL1TC02       4B6    SD
                      4B8   PL1TC02A      260    SD
                      718   IHEMAIN        04    SD
                      720   IHENTRY        0C    SD
                      730   C              04    SD
                      738   B              04    SD
                      740   A              04    SD
                      748   IHESPRT        38    SD

--------------------------------------------------------------------------------------------------------------------


     LMOD LOC  CSECT LOC    IN CSECT              REFERS TO SYMBOL  AT LMOD LOC  CSECT LOC    IN CSECT

          10        10    PL1TC02                   PL1TC02A          4B8         00       PL1TC02A
         4D8        20    PL1TC02A                  IHESADA                                $UNRESOLVED
         4DC        24    PL1TC02A                  IHESADB                                $UNRESOLVED
         4E0        28    PL1TC02A                  PL1TC02            00         00       PL1TC02
         4E4        2C    PL1TC02A                  PL1TC02            00         00       PL1TC02
         4E8        30    PL1TC02A                  IHESAFA                                $UNRESOLVED
         4F8        40    PL1TC02A                  PL1TC02            00         00       PL1TC02
         4FC        44    PL1TC02A                  PL1TC02            00         00       PL1TC02
         500        48    PL1TC02A                  PL1TC02            00         00       PL1TC02
         504        4C    PL1TC02A                  PL1TC02            00         00       PL1TC02
         508        50    PL1TC02A                  IHELDOA                                $UNRESOLVED
         50C        54    PL1TC02A                  IHELDOB                                $UNRESOLVED
         510        58    PL1TC02A                  IHEIOBT                                $UNRESOLVED
         514        5C    PL1TC02A                  IHEIOBC                                $UNRESOLVED
         518        60    PL1TC02A                  IHESAFB                                $UNRESOLVED
         51C        64    PL1TC02A                  AA                                     $UNRESOLVED
         520        68    PL1TC02A                  C                 730         00       C
         524        6C    PL1TC02A                  B                 738         00       B
         528        70    PL1TC02A                  A                 740         00       A
         52C        74    PL1TC02A                  IHESPRT           748         00       IHESPRT
         600       148    PL1TC02A                  A                 740         00       A
         608       150    PL1TC02A                  IHESPRT           748         00       IHESPRT
         60C       154    PL1TC02A                  PL1TC02            00         00       PL1TC02
         614       15C    PL1TC02A                  IHESPRT           748         00       IHESPRT
         618       160    PL1TC02A                  PL1TC02            00         00       PL1TC02
         620       168    PL1TC02A                  IHESPRT           748         00       IHESPRT
         624       16C    PL1TC02A                  PL1TC02            00         00       PL1TC02
         62C       174    PL1TC02A                  IHESPRT           748         00       IHESPRT
         630       178    PL1TC02A                  PL1TC02            00         00       PL1TC02
         718        00    IHEMAIN                   PL1TC02            00         00       PL1TC02
         728        08    IHENTRY                   IHESAPC                                $UNRESOLVED

LENGTH OF LOAD MODULE      780
```

```
                        NUMERICAL MAP AND CROSS-REFERENCE LIST OF LOAD MODULE PL1LOAD                    PAGE 0002


--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------


                PSEUDO REGISTER
                 VECTOR LOC   NAME      LENGTH

                       00   IHEQINV        4
                       04   IHEQERR        4
                       08   IHEQTIC        4
                       0C   IHEQLWF        4
                       10   IHEQSLA        4
                       14   IHEQLWO        4
                       18   PL1TC02B       4
                       1C   PL1TC02C       4
                       20   IHEQSPR        4

LENGTH OF PSEUDO REGISTERS      24
```

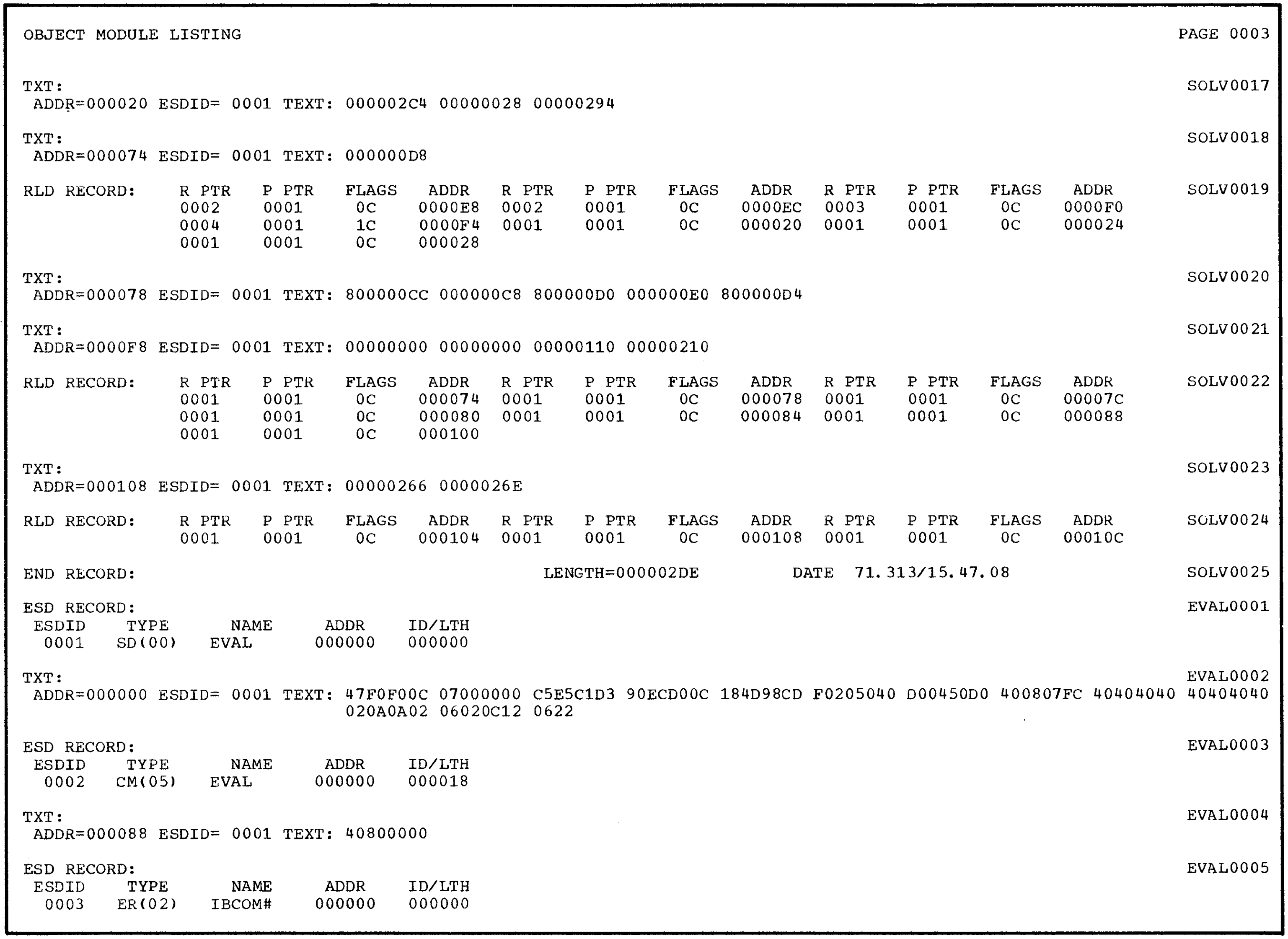
Figure LIST-3. Sample LISTLOAD Output - Cross Reference Listing (Part 1 of 2)



```
                                  ALPHABETICAL MAP OF LOAD MODULE PL1LOAD                           PAGE 0003


          CONTROL SECTION                                               ENTRY
              NAME      LMOD LOC    LENGTH   TYPE                          NAME     LMOD LOC   CSECT LOC   CSECT NAME

            A             740         04      SD
            B             738         04      SD
            C             730         04      SD
            IHEMAIN       718         04      SD
            IHENTRY       720         0C      SD
            IHESPRT       748         38      SD
            PL1TC02        00        4B6      SD
            PL1TC02A      4B8        260      SD

-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------


          PSEUDO REGISTER
              NAME     VECTOR LOC   LENGTH

            IHEQERR       04          4
            IHEQINV       00          4
            IHEQLWF       0C          4
            IHEQLWO       14          4
            IHEQSLA       10          4
            IHEQSPR       20          4
            IHEQTIC       08          4
            PL1TC02B      18          4
            PL1TC02C      1C          4
```

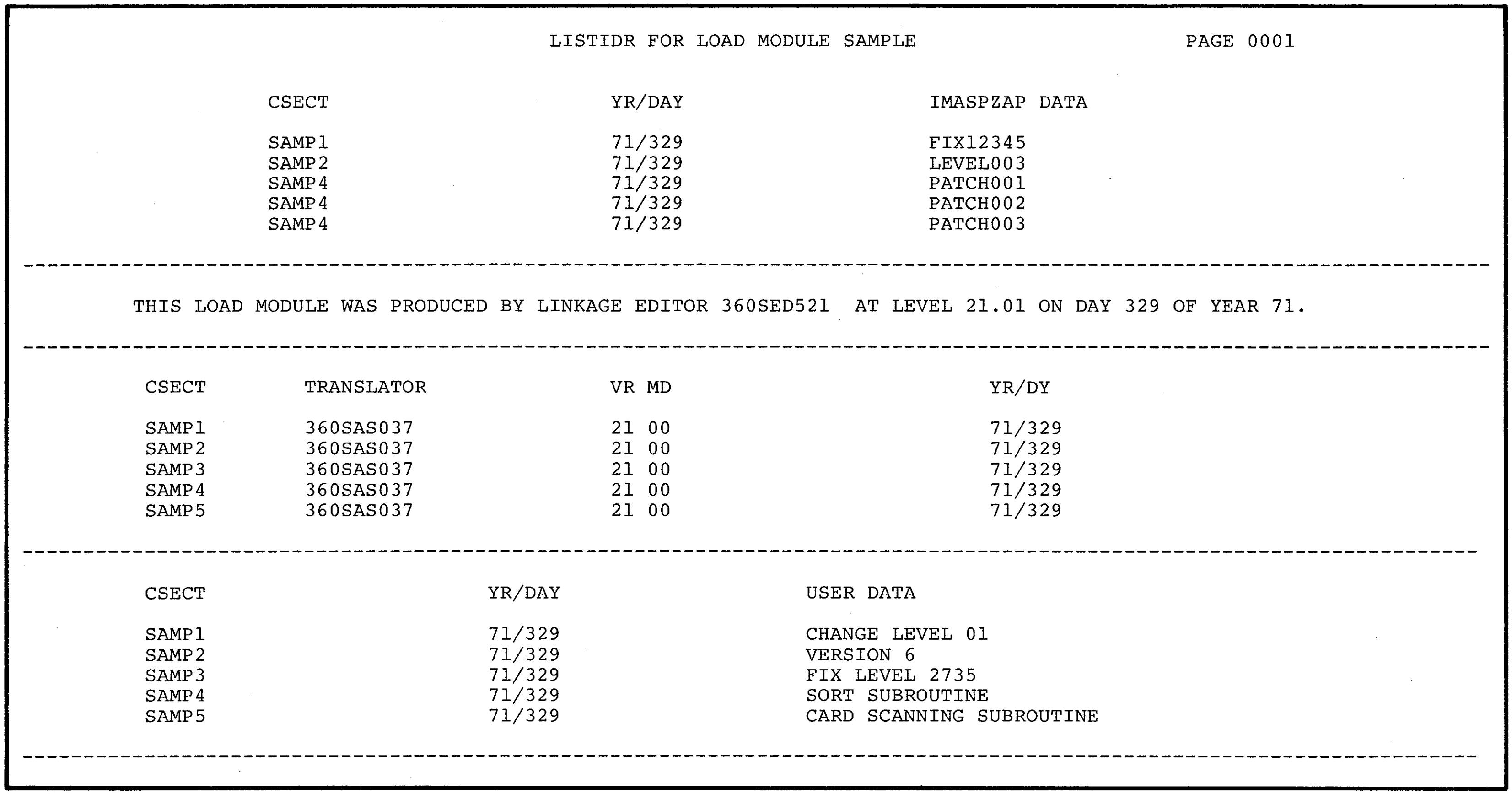
```
                          ALPHABETICAL CROSS-REFERENCE LIST OF LOAD MODULE PL1LOAD                          PAGE 0004


  SYMBOL   AT LMOD LOC   CSECT LOC    IN CSECT                IS REFERRED TO BY LMOD LOC   CSECT LOC      IN CSECT

 A              740          00        A                                         528           70        PL1TC02A
 A              740          00        A                                         600          148        PL1TC02A
 AA                                   $UNRESOLVED                                51C           64        PL1TC02A
 B              738          00        B                                         524           6C        PL1TC02A
 C              730          00        C                                         520           68        PL1TC02A
 IHEIOBC                              $UNRESOLVED                                514           5C        PL1TC02A
 IHEIOBT                              $UNRESOLVED                                510           58        PL1TC02A
 IHELDOA                              $UNRESOLVED                                508           50        PL1TC02A
 IHELDOB                              $UNRESOLVED                                50C           54        PL1TC02A
 IHESADA                              $UNRESOLVED                                4D8           20        PL1TC02A
 IHESADB                              $UNRESOLVED                                4DC           24        PL1TC02A
 IHESAFA                              $UNRESOLVED                                4E8           30        PL1TC02A
 IHESAFB                              $UNRESOLVED                                518           60        PL1TC02A
 IHESAPC                              $UNRESOLVED                                728           08        IHENTRY
 IHESPRT        748          00        IHESPRT                                   52C           74        PL1TC02A
 IHESPRT        748          00        IHESPRT                                   608          150        PL1TC02A
 IHESPRT        748          00        IHESPRT                                   614          15C        PL1TC02A
 IHESPRT        748          00        IHESPRT                                   620          168        PL1TC02A
 IHESPRT        748          00        IHESPRT                                   62C          174        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   4E0           28        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   4E4           2C        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   4F8           40        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   4FC           44        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   500           48        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   504           4C        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   60C          154        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   618          160        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   624          16C        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   630          178        PL1TC02A
 PL1TC02         00          00        PL1TC02                                   718           00        IHEMAIN
 PL1TC02A       4B8          00        PL1TC02A                                   10           10        PL1TC02

******END OF MAP AND CROSS-REFERENCE LISTING
```

Figure LIST-3. Sample LISTLOAD Output - Cross Reference Listing (Part 2 of 2)



```
OBJECT MODULE LISTING                                                                                          PAGE 0003

TXT:                                                                                                            SOLV0017
 ADDR=000020 ESDID= 0001 TEXT: 000002C4 00000028 00000294

TXT:                                                                                                            SOLV0018
 ADDR=000074 ESDID= 0001 TEXT: 000000D8

RLD RECORD:    R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR       SOLV0019
               0002    0001     0C    0000E8  0002    0001     0C    0000EC  0003    0001     0C    0000F0
               0004    0001     1C    0000F4  0001    0001     0C    000020  0001    0001     0C    000024
               0001    0001     0C    000028

TXT:                                                                                                            SOLV0020
 ADDR=000078 ESDID= 0001 TEXT: 800000CC 000000C8 800000D0 000000E0 800000D4

TXT:                                                                                                            SOLV0021
 ADDR=0000F8 ESDID= 0001 TEXT: 00000000 00000000 00000110 00000210

RLD RECORD:    R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR       SOLV0022
               0001    0001     0C    000074  0001    0001     0C    000078  0001    0001     0C    00007C
               0001    0001     0C    000080  0001    0001     0C    000084  0001    0001     0C    000088
               0001    0001     0C    000100

TXT:                                                                                                            SOLV0023
 ADDR=000108 ESDID= 0001 TEXT: 00000266 0000026E

RLD RECORD:    R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR       SOLV0024
               0001    0001     0C    000104  0001    0001     0C    000108  0001    0001     0C    00010C

END RECORD:                                       LENGTH=000002DE         DATE  71.313/15.47.08             SOLV0025

ESD RECORD:                                                                                                     EVAL0001
 ESDID    TYPE      NAME     ADDR    ID/LTH
  0001   SD(00)    EVAL     000000   000000

TXT:                                                                                                            EVAL0002
 ADDR=000000 ESDID= 0001 TEXT: 47F0F00C 07000000 C5E5C1D3 90ECD00C 184D98CD F0205040 D00450D0 400807FC 40404040 40404040
                               020A0A02 06020C12 0622

ESD RECORD:                                                                                                     EVAL0003
 ESDID    TYPE      NAME     ADDR    ID/LTH
  0002   CM(05)    EVAL     000000   000018

TXT:                                                                                                            EVAL0004
 ADDR=000088 ESDID= 0001 TEXT: 40800000

ESD RECORD:                                                                                                     EVAL0005
 ESDID    TYPE      NAME     ADDR    ID/LTH
  0003   ER(02)    IBCOM#   000000   000000
```

Figure LIST-4. Sample LISTOBJ Output



```
                                           LISTIDR FOR LOAD MODULE SAMPLE                    PAGE 0001

                    CSECT                       YR/DAY                    IMASPZAP DATA

                    SAMP1                       71/329                    FIX12345
                    SAMP2                       71/329                    LEVEL003
                    SAMP4                       71/329                    PATCH001
                    SAMP4                       71/329                    PATCH002
                    SAMP4                       71/329                    PATCH003

------------------------------------------------------------------------------------------------------------------------

         THIS LOAD MODULE WAS PRODUCED BY LINKAGE EDITOR 360SED521  AT LEVEL 21.01 ON DAY 329 OF YEAR 71.

------------------------------------------------------------------------------------------------------------------------

          CSECT        TRANSLATOR                VR MD                          YR/DY

          SAMP1        360SAS037                 21 00                          71/329
          SAMP2        360SAS037                 21 00                          71/329
          SAMP3        360SAS037                 21 00                          71/329
          SAMP4        360SAS037                 21 00                          71/329
          SAMP5        360SAS037                 21 00                          71/329

-----------------------------------------------------------------------------------------------------------------------

          CSECT                    YR/DAY                    USER DATA

          SAMP1                    71/329                    CHANGE LEVEL 01
          SAMP2                    71/329                    VERSION 6
          SAMP3                    71/329                    FIX LEVEL 2735
          SAMP4                    71/329                    SORT SUBROUTINE
          SAMP5                    71/329                    CARD SCANNING SUBROUTINE

-----------------------------------------------------------------------------------------------------------------------
```

Figure LIST-5. Sample LISTIDR Output

Examples

Example 1: Listing Several Object Modules

In this example, IMBLIST is used to list all object modules contained in the data set named OBJMODS, three specific object modules from another data set called OBJMOD, and finally all object modules in OBJMOD.

```
//OBJLIST      JOB       MSGLEVEL=(1,1)
//LISTSTEP     EXEC      PGM=IMBLIST
//SYSPRINT     DD        SYSOUT=A
//OBJLIB       DD        DSN=OBJMODS,DISP=OLD
//OBJSDS       DD        DSN=OBJMOD=DISP=OLD
//SYSIN        DD        *
    LISTOBJ         DDN=OBJSDS,
         TITLE=('OBJECT MODULE LISTING OF OBJSDS',20)
    LISTOBJ         DDN=OBJLIB,MEMBER=(OBJ1,OBJ2,OBJ3),
         TITLE=('OBJECT MODULE LISTING OF OBJ1 OBJ2 OBJ3),20)
    LISTOBJ         DDN=OBJLIB,
         TITLE=('OBJ MOD LISTING OF ALL MODS IN OBJLIB',20)
/*
```

SYSPRINT DD Statement

defines the message data set. This statement must be included; if it is omitted, IMBLIST will produce no output.

OBJLIB and OBJSDS DD Statements

define input data sets that contain object modules.

SYSIN DD Statement

defines the data set in the input stream containing IMBLIST control statements.

LISTOBJ Control Statement #1

instructs IMBLIST to format the data set defined by the OBJSDS DD statement, treating them as a single continuous object module. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTOBJ Control Statement #2

instructs IMBLIST to format three members of the partitioned data set defined by the OBJLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTOBJ Control Statement #3

instructs IMBLIST to format the entire data set defined by the OBJLIB DD statement, treating it as a sequential data set. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example 2: Using the LISTLOAD Control Statement

In this example, IMBLIST is used to produced formatted listings of several load modules.

```
//LOADLIST    JOB       MSGLEVEL=(1,1)
//LISTSTEP    EXEC      PGM=IMBLIST
//SYSPRINT    DD        SYSOUT=A
//SYSLIB      DD        DSNAME=SYS1.LINKLIB,DISP=OLD
//LOADLIB     DD        DSNAME=LOADMOD,DISP=OLD
//SYSIN       DD        *
    LISTLOAD       OUTPUT=MODLIST,DDN=LOADLIB,
         MEMBER=TESTMOD,
         TITLE=('LOAD MODULE LISTING OF TESTMOD',20)
    LISTLOAD       OUTPUT=XREF,DDN=LOADLIB,
         MEMBER=(MOD1,MOD2,MOD3),
         TITLE=('XREF LISTINGS OF MOD1 MOD2 AND MOD3',20)
    LISTLOAD  TITLE=('XREF & LD MOD LSTNG - ALL MOD IN LINKLIB',20)
/*
```

In this example:

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines an input data set, SYS1.LINKLIB, that contains load modules to be formatted.

LOADLIB DD Statement

defines a second input data set.

SYSIN DD Statement

defines the data set (in the input stream) containing the IMBLIST control statements.

LISTLOAD Control Statement #1

instructs IMBLIST to format the control and text records, including the external symbol dictionary and relocation dictionary records, of the load module TESTMOD in the data set defined by the LOADLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #2

instructs IMBLIST to produce a module map and cross=reference listing of the load modules MOD1, MOD2, and MOD3 in the data set defined by the LOADLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #3

instructs IMBLIST to produce a formatted listing of the load module and its map and cross-reference listing. Because no DDN= parameter is included, the input data set is assumed to be the one defined by the SYSLIB DD statement. Because no MEMBER= parameter is specified, all load modules in the data set will be processed. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example 3: Using the LISTIDR Control Statement

In this example, IMBLIST is used to list the CSECT identification records in several load modules.

```
//IDRLIST     JOB       MSGLEVEL=(1,1)
//LISTSTEP    EXEC      PGM=IMBLIST
//SYSPRINT    DD        SYSOUT=A
//SYSLIB      DD        DSN=SYS1.LINKLIB,DISP=OLD
//LOADLIB     DD        DSN=LOADMODS,DISP=OLD
//SYSIN       DD        *
    LISTIDR   TITLE=('IDR LISTINGS OF ALL MODS IN LINKLIB',20)
    LISTIDR   OUTPUT=IDENT,DDN=LOADLIB,MEMBER=TESTMOD
              TITLE=('LISTING OF MODIFICATIONS TO TESTMOD',20)
    LISTIDR   OUTPUT=ALL,DDN=LOADLIB,MEMBER=(MOD1,MOD2,MOD3),
              TITLE=('IDR LISTINGS OF MOD1 MOD2 MOD3',20)
/*
```

In this example:

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the input data set SYS1.LINKLIB, which contains load modules to be processed.

LOADLIB DD Statement

defines a second input data set.

SYSIN DD Statement

defines the data set (in the input stream) containing the IMBLIST control statements.

LISTIDR Control Statement #1

instructs IMBLIST to list all CSECT identification records for all modules in SYS1.LINKLIB (this is the default data set since no DDN= parameter was included). It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTIDR Control Statement #2

instructs IMBLIST to list CSECT identification records that contain IMASPZAP or user-supplied data for load module TESTMOD. TESTMOD is a member of the data set defined by the LOADLIB DD statement. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTIDR Control Statment #3

instructs IMBLIST to list all CSECT identification records for load modules MOD1,MOD2, and MOD3. These are members in the data set defined by the LOADLIB DD statement. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example 4: Verifying an Object Deck

In this example, IMBLIST is used to format and list an object module included in the input stream.

```
//LSTOBJDK     JOB       MSGLEVEL=(1,1)
//             EXEC      PGM=IMBLIST
//SYSPRINT     DD        SYSOUT=A
//OBJDECK      DD        *
    object deck
//SYSIN        DD        *
    LISTOBJ         DDN=OBJDECK,
         TITLE=('OBJECT DECK LISTING FOR MYJOB',25)
/*
```

SYSPRINT DD Statement

defines the message data set.

OBJDECK DD Statement

defines the input data set, which follows immediately. In this case the input data set is an object deck.

SYSIN DD Statement

defines the data set containing IMBLIST control statements, which follows immediately.

LISTOBJ Control Statement

instructs IMBLIST to format the data set defined by the IBJDECK DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example 5: Combining LISTOBJ, LISTLOAD, and LISTIDR

An unsuccessful attempt has been made to link edit an object module with two load modules to produce one large load module. This example shows how to use IMBLIST to verify all three modules.

```
//LSTLDOBJ     JOB       MSGLEVEL=(1,1)
//             EXEC      PGM=IMBLIST
//SYSPRINT     DD        SYSOUT=A
//OBJMOD       DD        DSN=MYMOD,DISP=OLD
//LOADMOD1     DD        DSN=YOURMOD,DISP=OLD
//LOADMOD2     DD        DSN=HISMOD,DISP=OLD
//SYSIN        DD        *
    LISTOBJ         DDN=OBJMOD,
         TITLE=('OBJECT LISTING FOR MYMOD',20)
    LISTLOAD        DDN=LOADMOD1,OUTPUT=BOTH,
         TITLE=('LISTING FOR YOURMOD',25)
    LISTIDR         DDN=LOADMOD1,OUTPUT=ALL,
         TITLE=('IDRS FOR YOURMOD',25)
    LISTLOAD        DDN=LOADMOD2,OUTPUT=BOTH,
         TITLE=('LISTING FOR HISMOD',25)
    LISTIDR         DDN=LOADMOD2,OUTPUT=ALL,
         TITLE=('IDRS FOR HISMOD',25)
/*
```

SYSPRINT DD Statement

defines the message data set.

OBJMOD DD Statement

defines an input load module data set.

LOADMOD1 and LOADMOD2 DD Statements

define input load module data sets.

SYSIN DD statement

defines the data set containing IMBLIST control statements, which follows immediately.

LISTOBJ Control Statement

instructs IMBLIST to format the data set defined by the OBJMOD DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #1

instructs IMBLIST to format all records associated with the data set defined by the LOADMOD1 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTIDR Control Statement #1

instructs IMBLIST to list all CSECT identification records associated with the data set defined by the LOADMOD1 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTLOAD Control Statement #2

instructs IMBLIST to format all records associated with the data set defined by the LOADMOD2 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

Chapter 6: IMBMDMAP

Maps load modules.

MDMAP

Contents

INTRODUCTION 129

CHARACTERISTICS OF THE LOAD MODULE MAP 130
Load Module and Nucleus Maps 130
Link Pack Area Maps 130
Specialized Maps 130

INPUT TO MDMAP 131
Load Modules 131
The Modular Concept 131
Control Sections 131
Object Modules 131
External Symbol Dictionaries .132
Relocatable Load Dictionaries .133
Text 133
Linkage Editor Output 133
Load Module Attributes 135
Programmer-Assigned Attributes 135
Linkage Editor-Assigned Attributes 136
Load Modules with Overlay Characteristics 136
MVT Link Pack Area 137
MFT Resident Reenterable Load Module Area 137
Nucleus 138

EXECUTING MDMAP 140
JCL Statements 140
EXEC Statement Parameters 140

OUTPUT FORMATS 143
Numerical ESD Listing 143
Numerical RLD Listing 144
Alphabetic ESD Listing 144
Alphabetic RLD Listing 145

MDMAP EXAMPLES 146
Example 1: Mapping an MVT Link Pack Area and Nucleus 146
Example 2: Mapping the ESDs of a Load Module, Using the Relocation Option 148
Example 3: Mapping a Load Module with the DEBUG and Relocation Options 150

OPERATIONAL CONSIDERATIONS 152

Figures

Figure MDMAP-1. Structure of an Object Module 132
Figure MDMAP-2. Creation of a Load Module by the Linkage Editor . .134
Figure MDMAP-3. Structure of a Load Module With Overlay 136
Figure MDMAP-4. Upper Main Storage After IPL of an MVT System . . .137
Figure MDMAP-5. Lower Main Storage Organization in MFT 138
Figure MDMAP-6. A Nucleus After IPL 139
Figure MDMAP-7. IMBMDMAP Execution JCL 140
Figure MDMAP-8. Snap Dumps Taken When the DEBUG Parmaeter Is Used .141
Figure MDMAP-9. Excerpts from the Map Resulting From Example 1 . .147
Figure MDMAP-10. Excerpts from the Map Resulting From Example 2 . .149
Figure MDMAP-11. Excerpts from the Map Resulting From Example 3 . .151

Introduction

IMBMDMAP, the Load Module Mapping service aid program, operates under the control of IBM Operating System/360, and provides the facility for mapping:

- A system's nucleus;
- The load modules included in an MVT link pack area or an MFT resident reenterable load module area; or
- Load modules previously link edited into a partitioned data set.

In determining the cause of problems in the execution of system component programs or complex user problem programs, the load module maps produced by IMBMDMAP, used in conjunction with main storage dumps, constitute powerful debugging aids. They enable the user to readily locate and identify individual control sections and their entry points, and to verify load module attributes and aliases.

Characteristics of the Load Module Map

A load module map contains edited information regarding the control sections, entry points, aliases, external references, attributes, type codes, overlay segments and hierarchy designations for each load module for which a map is requested.

Load Module and Nucleus Maps

A map of load modules from a partitioned data set (PDS) or a map of a nucleus consists of external symbol dictionary (ESD) and relocatable load dictionary (RLD) items, sorted first to numeric order by location, and then to alphabetic order by name. ESD and RLD items are discussed more fully under their respective headings in this chapter.

Link Pack Area Maps

A map of an MVT link pack area (LPA) contains contents directory entries (CDEs); that of an MFT resident reenterable load module area (analogous to the MVT link pack area) contains loaded program request block (LPRB) entries. The nature of these entries is discussed under the headings "MVT Link Pack Area" and "MFT Resident Reenterable Load Module Area" in this chapter.

In a map of either area type, the entries are sorted numerically, then alphabetically; and the length, entry points, and relative addresses of each module in the area are listed.

Specialized Maps

The user can request a map containing only ESD items in numeric sequence when executing IMBMDMAP. Or, an address relocation value may be specified to the program; that value will be assigned as the map's base address, and the result will be the printing of an absolute main storage location for each record, providing an added storage dump debugging aid. Or, the user may request a map that includes a series of "snapshot" dumps, taken at strategic points in time during IMBMDMAP's execution, and useful in determining the cause of certain load module structural problems -- including those which might arise during execution of IMBMDMAP.

Input to MDMAP

Input to IMBMDMAP may be a load module, a link pack area (MVT), a resident reenterable load module area (MFT), or any OS/360 nucleus. The following sections describe the contents and function of each type of input.

Load Modules

A load module is composed of all the edited modules (object, load, or an intermix of both types) that are input to the linkage editor for a given linkage. In addition to text items, a load module contains composite ESD and RLD entries. Any load module is both relocatble and executable.

The Modular Concept

Every program is designed to fulfull a particular purpose. In achieving that purpose, a program can be divided into logical functional units. Each of these units, defined as a section of coding that performs a specific task or several related functions, can be termed a module.

Control Sections

A module contains one or more control sections. A control section, or CSECT, is a unit of instructions and data that, within itself, is an entity. All elements of a control section (CSECT) are loaded and executed in a constant relationship to one another. A CSECT is, therefore, the smallest separately relocatable unit of a program.

Object Modules

Each module within a program can be separately assembled or compiled by a language translator. During this processing, references between the module's component control sections are unresolved. Object modules, the output of the language translator, consist of control dictionaries and text. Control dictionaries contain the information necessary to resolve cross-references between control sections and modules. A module's text area contains its instructions and data. Figure MDMAP-1 illustrates the structure of an object module. An object module is relocatable, but not executable.

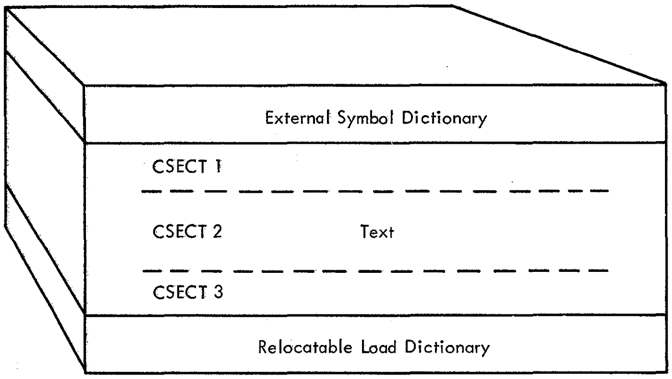


Figure MDMAP-1. Structure of an Object Module

External Symbol Dictionaries

An external symbol dictionary (ESD) entry identifies and defines the position of the external symbols contained, or referred to, in a module. Each entry is classified as either an external name or an external reference.

External Names

An external name is a defined value within the module, bearing a name that can be referred to by any control section or by any separately assembled or compiled module. There are four types of external names:

a. Control Section Name: The symbolic name of a control section. The ESD entry specifies the name, the assembled origin, and the length of a control section. The defined value of the symbol is the address of the first byte of the control section.

b. Entry Name: A name within a control section defining a point in the coding unit where processing may begin, or "enter." The ESD entry specifies the assembled address of the name and identifies the control section to which it belongs.

c. Blank or Named Common Area: A control section used to reserve a main storage area (containing no data or instructions) for CSECTs supplied by other modules, or as a center for communication between modules within a program. The ESD entry specifies the name and length of a named common area. The name field of a blank common area contains blanks.

d. Private Code: An unnamed control section. The ESD entry specifies the assembled address and assigned length of the area. The name field contains blanks. Since it has no name, a private code area cannot be referred to by any other control section.

External References

An external reference is a symbol referred to in a given module, but defined as an external name in another module. The ESD dictionary for the current module specifies the name only.

Relocatable Load Dictionaries

Relocatable load dictionaries (RLDs) contain information about address constants within the module. Each RLD entry identifies an address constant by:

- Indicating its location within the module, and

- Identifying the ESD symbol whose contents are used in determining the value of the address constant.

For a detailed discussion of ESD and RLD items, see the publication, IBM System/360 Operating System: Linkage Editor and Loader, GC28-6538.

Text

A text item includes the addresses of the instructions and data in a module, and indicates the ESD entry defining the CSECT in which the subject text is contained.

Linkage Editor Output

The linkage editor's output, a completed load module, is placed in a partitioned data set (SYSLMOD library) as a named member. In addition to its member name, the load module may carry as many as sixteen other names, or aliases. Under MFT it can contain up to 524,288 bytes; MVT allows larger modules. Figure MDMAP-2 illustrates the relationship of input to output of the linkage process.

In linking the input modules, the linkage editor resolves all references between control sections, just as if they had been assembled as a single module. The output load module contains the information necessary to load and relocate the module in main storage, and to compute the relocated value of location-dependent address constants. When it places the load module in the output module library, the linkage editor stores the module's member name, aliases, and attribute control information in the library's PDS directory.

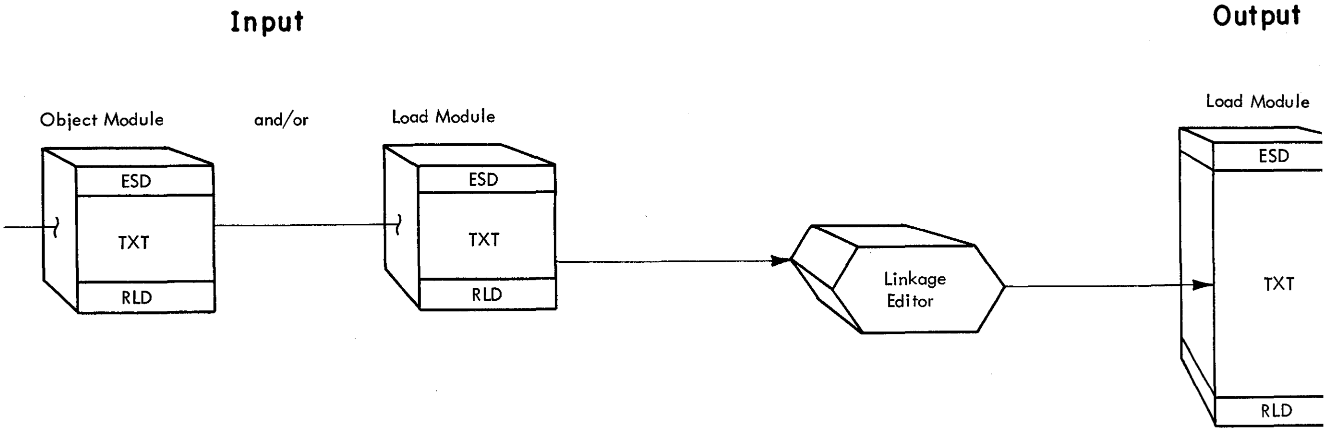


Figure MDMAP-2. Creation of a Load Module by the Linkage Editor

Load Module Attributes

Each load module has specific characteristics, or attributes, which are used by the control program when the module is loaded for execution. Some of these attributes are programmer-specified; others are assigned by the linkage editor as a result of information obtained during its processing of the module.

Programmer-Assigned Attributes

Attributes that can be assigned to a load module by the programmer, and the characteristics assumed by the load module under each assignment, are:

ASSIGNED ATTRIBUTE	LOAD MODULE CHARACTERISTICS
Reenterable	Executable by more than one task at a time; cannot be modified by any other load module during execution; cannot modify itself unless disabling techniques (such as the ENQ macro instruction, the test and set (TS) instruction, etc.) are used to prevent another routine from using this load module.
Serially Reusable	Executable by only one task at a time; will initialize itself and/or will restore any altered instructions or data before a new task takes control.
Refreshable	Cannot be modified by itself or by any other load module during execution, since it must be capable of being replaced by a new copy during execution without changing the results of proecessing.
Scatter Format	Is suitable for either block loading (placement in main storage in one contiguous block of space); or scatter loading (possible placement in main storage, by control section, in non-contiguous areas), thus taking better advantage of available storage space.
Hierarchy Format	Suitable for either block or scatter loading into either hierarchy 0 or hierarchy 1, as specified to the linkage editor when hierarchy support is included in the system.
Not Editable	Cannot be reprocessed by the linkage editor; that is, cannot be link edited again into a larger load module.
Only Loadable	Can be brought into main storage only by use of the LOAD macro instruction.
Downward Compatible	Can be reprocessed by either level E or level F of the linkage editor.
Overlay	Structured as directed by linkage editor OVERLAY statements.
Test	Applies only to Assembler Language programs that are to be tested; causes inclusion of the test symbol dictionary.

Linkage Editor-Assigned Attributes

Linkage editor-assigned attributes, and the load module characteristics that result, are:

ASSIGNED ATTRIBUTE	LOAD MODULE CHARACTERISTICS
Block Format	Suitable for block loading only.
Not Executable	Assigned when errors that would prevent successful execution of the load module are detected during linkage editing.

Load Modules with Overlay Characteristics

When a load module contains overlay characteristics, the linkage editor structures the module somewhat differently, incorporating segment and entry tables (SEGTABs and ENTABs) into the text.

The single segment table created by the linkage editor fo: an overlay program structure is used to keep track of:

- the relationship of the segments in the program;
- which segments are in main storage, or in the process of being loaded; and
- other control information.

Entry tables are linkage editor-generated for control program use in determining the segment to be loaded in response to a branch instruction or one of the macro instructions used to transfer control between overlay segments. Figure MDMAP-3 illustrates the structure of a load module containing overlay characteristics.

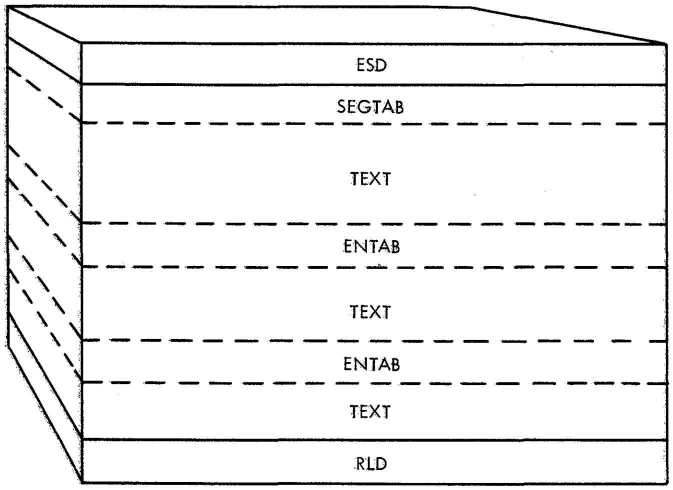


Figure MDMAP-3. Structure of a Load Module With Overlay

MVT Link Pack Area

The link pack area in MVT is a required feature of the system. It resides in upper main storage and contains reenterable routines from the linkage and supervisor call libraries (SYS1.LINKLIB and SYS1.SVCLIB).

MVT link pack area routines are available to all tasks requiring them, and thus need not be separately loaded into the various regions of main storage. Figure MDMAP-4 shows the arrangement of the library routines in an MVT link pack area. The types 3 and 4 SVC routines operate in the supervisor state; the others generally operate in the same state as the calling routine.

In MVT systems, the link pack area control queue (LPACQ) is composed of contents directory entries (CDEs), which are linked together. Each CDE on the LPACQ describes a routine resident in the LPA, giving the name, entry point and other attributes.

For a more detailed explanation of the link pack area control queue and associated control blocks, consult the publications, MVT Supervisor, GY28-6659, and System Control Blocks, GC28-6628.

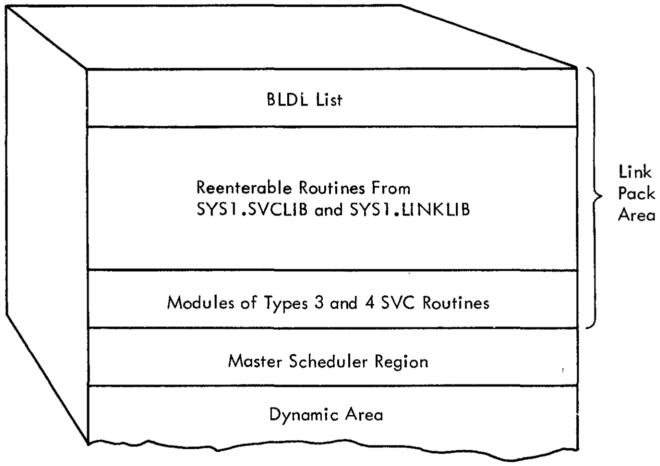

BLDL List
Reenterable Routines From SYS1.SVCLIB and SYS1.LINKLIB
Modules of Types 3 and 4 SVC Routines
Master Scheduler Region
Dynamic Area
Link Pack Area

Figure MDMAP-4. Upper Main Storage After IPL of an MVT System

MFT Resident Reenterable Load Module Area

Under MFT, the resident reenterable load module area is a SYSGEN option. If the option is selected, the access method routines from SYS1.SVCLIB and the routines from SYS1.LINKLIB which are to be made resident are loaded during system initialization. Figure MDMAP-5 shows the relative location of these routines in main storage.

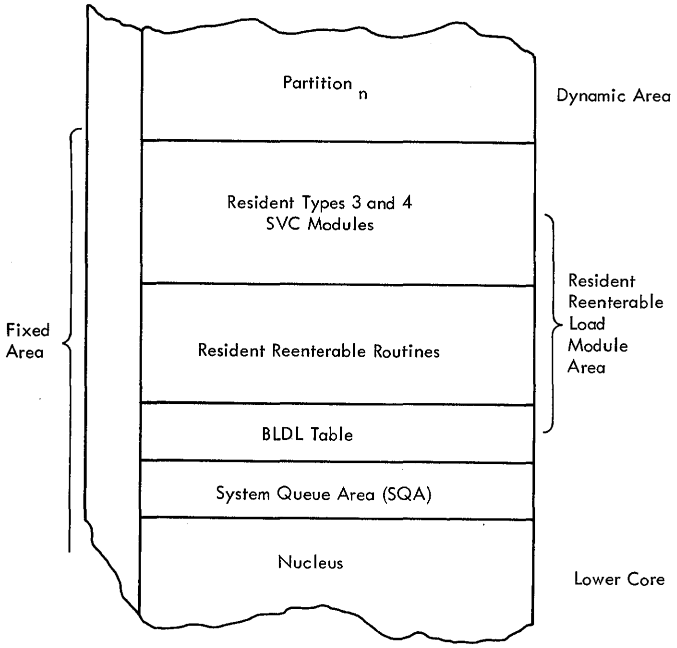

Partition n
Dynamic Area
Resident Types 3 and 4 SVC Modules
Resident Reenterable Load Module Area
Fixed Area
Resident Reenterable Routines
BLDL Table
System Queue Area (SQA)
Nucleus
Lower Core

Figure MDMAP-5. Lower Main Storage Organization in MFT

In MFT systems with the resident routine option selected, a queue of request blocks (RBs), called the reenterable load module queue, is maintained. Each request block describes a resident routine and can be either a loaded program request block (LPRB) or a loaded request block (LRB). For a more detailed description of the resident reenterable load module area and the associated control blocks, consult the publications Control Program with MFT, GY27-7128, and System Control Blocks, 3C28-6628.

For ease of reference, the term "link pack area" will herafter be used to denote either the MVT link pack area or the MFT resident reenterable load module area.

Nucleus

The nucleus, a member of the partitioned data set SYS1.NUCLEUS, is the resident portion of a control program. It is loaded into the fixed area of main storage at IPL time. The nucleus contains:

- all task supervision routines, except the nonresident types 3 and 4 SVC routines;
- the data management I/O supervisor and BLDL routine;
- the resident recovery management routines; and
- small transient areas into which certain nonresident SVC routines and I/O error handling routines, all resident in SYS1.SVCLIB, are loaded as needed.

Figure MDMAP-6 shows the layout of a nucleus after IPL. All control programs assign the nucleus to lower main storage.

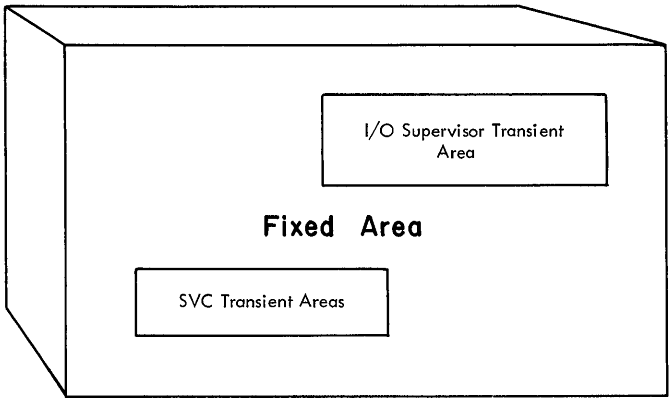


Figure MDMAP-6. A Nucleus after IPL

Executing MDMAP

IMBMDMAP runs in the problem program mode under any of the OS/360 control programs.

IMBMDMAP can be executed by use of the job control statements described in the next section. Main storage requirements for executing IMBMDMAP are variable, depending upon the number of ESD and RLD items present in the module being mapped. The average execution requires about 35K; an extremely complex module might require 70K.

JCL Statements

The statements required for executing IMBMDMAP are shown in Figure MDMAP-7.

Statement	Usage
JOB Statement	This statement initiates the JOB.
EXEC Statement	This statement specifies the program name: PGM=IMBMDMAP The statement may, in addition, contain from one to four parameters, randomly coded in any combination, to designate particular specialization of a map's format. These parameters are: [,PARM='LINKPACK,BASIC,DEBUG,hhhhhh'] These will be discussed individually under "EXEC Statement Parameters."
SYSPRINT DD Statement	This statement defines a sequential message data set, such as SYSOUT. The device defined for SYSPRINT may be a system output class, system output device, magnetic tape volume, or direct access volume.
//ddname DD Statement	This statement defines the load module to be mapped. One such statement must be supplied for each load module for which a map is to be produced. The statement's format is: //ddname DD DSN=YYY(ZZZ),DISP=SHR in which //ddname is any unique ddname. YYY is the dsname of the partitioned data set in which the load module to be mapped resides; e.g., SYS1.NUCLEUS. ZZZ is the name or alias of the member load module -- resident in the YYY data set -- to be mapped. For example, DSN=SYS1.NUCLEUS (IEANUC01) would accomplish mapping of the nucleus. IEANUC01 is the name of a load module contained in the data set called SYS1.NUCLEUS.
//SNAPDUMP DD Statement	This statement defines a sequential output data set, to be used as output for the SNAP dumps taken as a result of the DEBUG parameter (discussed under "EXEC Statement Parameters" in this section). This DD statement is required only if the DEBUG parameter is specified on the EXEC statement. The device specified for SNAPDUMP may be a system output class or any system output device.
//SYSABEND DD Statement or //SYSUDUMP DD Statement	These statements define a sequential output data set, to be used as output for the ABEND dump issued by IMBMDMAP as a result of the DEBUG parameter (discussed under "EXEC Statement Parameters" in this section). One of these statements is required only if the DEBUG parameter is specified on the EXEC statement. The device specified for either SYSABEND or SYSUDUMP may be a system output class or any system output device.

Figure MDMAP-7. IMBMDMAP Execution JCL

EXEC Statements Parameters

The parameters associated with IMBMDMAP's EXEC statement are:

PARM='LINKPACK'

indicates that a map of the link pack area of main storage under an MVT or MFT environment is to be produced. To obtain a complete map of all LPA modules (that is, to pick up the resident SVC routines), the nucleus currently resident in main storage must also be mapped. The user must, therefore, include in the jobstream a DD statement for the nucleus currently in core when requesting a link pack area map.

PARM='BASIC'

specifies that the resultant map is to contain only numerically-ordered external symbol dictionary items. Neither the alphabetic ESD nor either of the RLD listings is produced. The LINKPACK BASIC map will contain only numeric CDE or LPRB items.

PARM='hhhhhh'

where hhhhhh is a hexadecimal address of from one to six characters, and represents a relocation, or base, address. This parameter causes the program to add this value to the relative address of each mapped item, thus providing an absolute main storage address for the output listing. This does not apply to mapping a nucleus, which already has relocated addresses.

PARM='DEBUG'

provides for up to seven "snapshots" of main storage, taken at strategic intervals during execution of IMBMDMAP. These dumps are useful in the debugging of module construction problems, including any which may arise during the running of IMBMDMAP. The content of each dump, and the interval at which it is taken, are described in Figure MDMAP-8.

DUMP NUMBER	CONTENT	EXECUTION INTERVAL WHEN TAKEN
1	ESD entries	After read
	RLD entries	After read and first sorting pass
2	TRANSLATE table	After read (nucleus only)
	SCATTER table	After read (nucleus only)
3	IPLTABLE	Simulation of IPL conditions to ensure accuracy of map (nucleus only)
4	TRANSLATE Table	After read
	SCATTER Table	After read or after IPL relocation (nucleus only)
5	ESD entries	After relocation (scatter loading
	RLD entries	After second sorting pass
6	ESD entries	After EXEC parameter relocation if specified
	RLD entries	After EXEC parameter relocation if specified
7	CDE Table or	If MVT link pack area is being mapped
	LPRB Table	If MFT link pack area is being mapped

Figure MDMAP-8. Snap Dumps Taken When the DEBUG Parameter Is Used

Of course, the production of certain of these dumps depends upon the nature of the area being mapped. For example, maps 2, 3, and 4 would not be provided if the modules in the nucleus were not being mapped.

The jobstream must include a SNAPDUMP DD statement, defining a sequential message data set, for output of these dumps.

Additionally, the DEBUG parameter produces:

- a hexadecimal dump of each mapped module, the text portion of which may be truncated;

- a dump of each involved PDS directory; and:

- an ABEND dump, if a SYSABEND or SYSUDUMP DD statement has been provided, since IMBMDMAP terminates with a user 100 ABEND code. The SYSABEND statement provides the user with a more complete main storage dump than the SYSUDUMP statement.

Output Formats

The printed output of IMBMDMAP is a formatted listing, giving the user detailed information about the CSECTs contained in each mapped load module. The "regulation" map -- that is, one not limited or expanded through the use of parameters -- provides this information in four sections.

Numerical ESD Listing

A map, ordered by main storage location, of:

- all attributes assigned to a given load module,
- all aliases assigned to the load module,
- the module's primary entry point,
- all CSECTs and entry points within the load module, and
- all external references within the load module.

CSECTs are listed to the left of a page; entry points, in two sets of columns, in the center; and external references to the right. Each CSECT is identified, in addition to name, address, and length, by one of six type codes:

- CM (Common) indicates that the name defines a common area, named or unnamed. A constant, $BLK COM, is assigned to the name field if the area is unnamed. For example,

```
                           CSECT

      NAME          ADDRESS        LENGTH         TYPE

    $BLK COM         080090        000008          CM
```

- LR (Label Reference) indicates that the name defines a label, or symbol, within a control section.
- PC (Private Code) indicates that the name defines the beginning of an unnamed control section. A constant, $PRIVATE, is assigned to the name field of such CSECTs on the listing.
- PD (Private Code Marked Delete) indicates that this is an ENTAB or a SEGTAB. The code is used with modules having the overlay attribute.
- PR (Pseudo Register) defines an area external to the load module, but referred to within it, for which storage is allocated at the time the load module is executed.
- SD (Section Definition) indicates that the CSECT name defines the beginning of a named control section.

Two additional mutually exclusive CSECT definitions exist. When applicable, they appear immediately to the right of the type column on the listing. They are:

- SEG (Segment), a column heading under which appears the overlay segment in which a CSECT is contained. This is used with modules that have the overlay attribute. An example of this identification is:

```
                                  CSECT

    NAME       ADDRESS           LENGTH          TYPE       SEG

  $SEGTAB      010A20            00004C          PD         01
  IEKAA01      010A70            000114          SD         01
     .            .                 .             .          .
     .            .                 .             .          .
     .            .                 .             .          .
  IEKXRS       017B78            0000E8          SD         02
     .            .                 .             .          .
     .            .                 .             .          .
     .            .                 .             .          .
  IEKVFP       0225C8            000A60          SD         0D
  IEKP25       023028            000244          CM         0D
```

- HIERARCHY1, a constant printed beside the TYPE when a load module has been link edited with hierarchy designation, and the CSECT has been marked for loading into Hierarchy 1. Hierarchy 0 loading is indicated by the absence of such a notation. An example of the use of this identification is:

```
                                  CSECT

    NAME       ADDRESS           LENGTH          TYPE

IMBTST05       000000            00000C          SD
IMBTST06       000010            00000C          SD
IMBTST07       000020            00000C          SD   HIERARCHY1
IMBTST08       000030            00000C          SD   HIERARCHY1
```

Numerical RLD Listing

A map, in order by main storage location, of the RLD items within a load module. Column headings on the listing are:

```
         1          2         3          4          5

    LOCATION     REL ADR  IN CSECT  REFERS TO   IN CSECT
```

This line of column headings may be interpreted as:

"In this location (column 1) . . . at this relative address (column 2) . . . in this control section (column 3) . . . there is a reference to the area identified by this name (column 4) . . . which resides in this control section (column 5).

Alphabetic ESD Listing

The same as the numerical ESD listing but sorted to order by ESD name rather than by location.

Alphabetic RLD Listing

The same as the numerical RLD listing but sorted to order by reference name (column 4) rather than by location.

A map of a link pack area is formatted to give the following information, appearing in two sets of columns on a listing page:

```
    LOCATION  LENGTH    NAME      EP ADR    EP REL ADR
```

where EP ADR is entry point address, and EP REL ADR is entry point relative address.

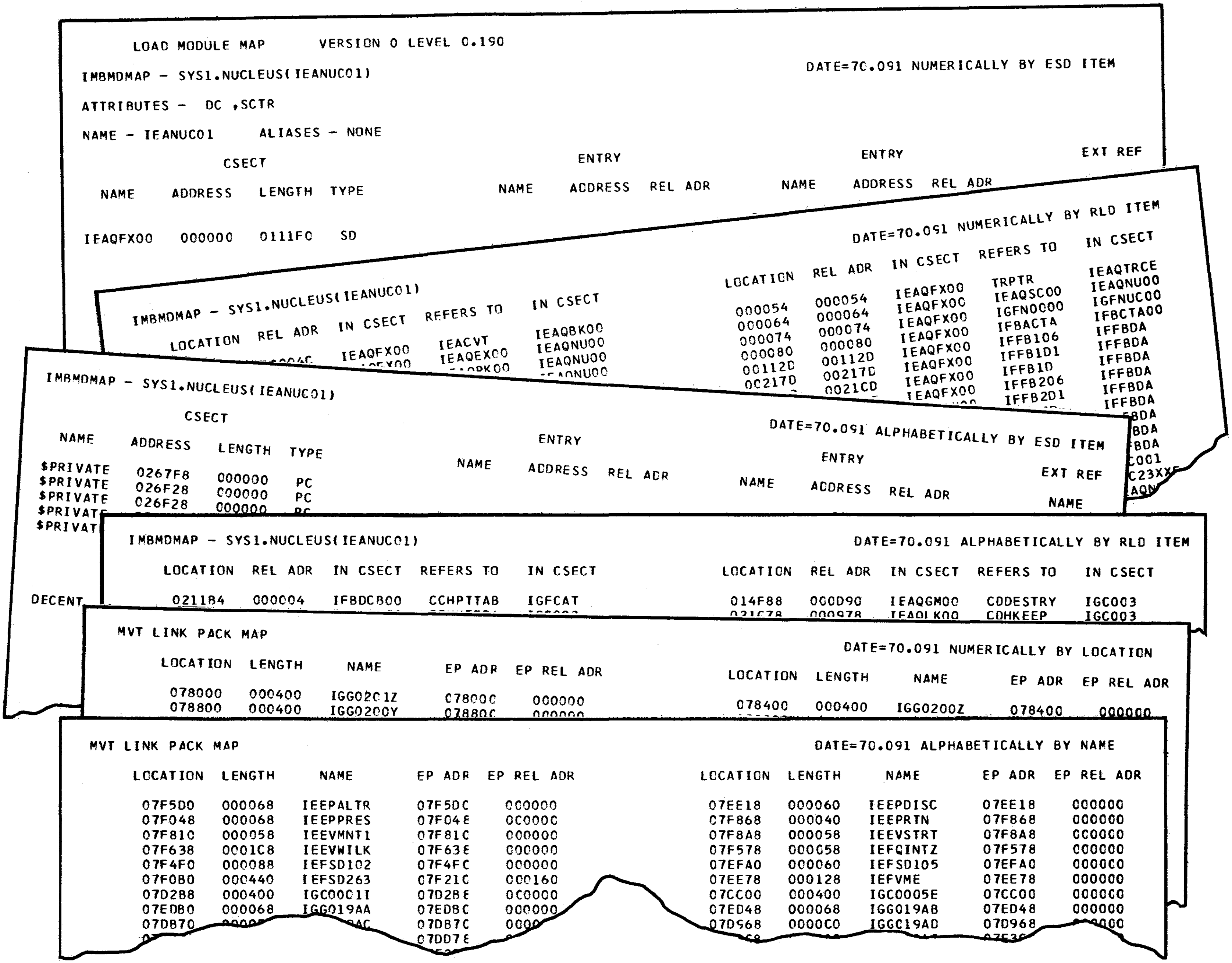
MDMAP Examples

The following examples and figures illustrate the job control language statements needed to produce sample configurations of IMBMDMAP maps; excerpts from the resulting maps show the headings and data for the edited portions of the information.

Example 1: Mapping an MVT Link Pack Area and Nucleus

This example shows the statements for a map of an MVT nucleus and link pack area. Figure MDMAP-9 shows the resulting map.

```
//JOB1         JOB       MSGLEVEL=(1,1)
//STEP1        EXEC      PGM=IMBMDMAP,PARM='LINKPACK'
//DD1          DD        DSN=SYS1.NUCLEUS(IEANUC01),DISP=SHR
//SYSPRINT     DD        SYSOUT=A
/*
```



```
     LOAD MODULE MAP       VERSION 0 LEVEL 0.190

IMBMDMAP - SYS1.NUCLEUS(IEANUC01)                                          DATE=70.091 NUMERICALLY BY ESD ITEM

ATTRIBUTES -   DC ,SCTR

NAME - IEANUC01      ALIASES - NONE
                 CSECT                                       ENTRY                            ENTRY                       EXT REF

  NAME      ADDRESS    LENGTH   TYPE                  NAME     ADDRESS   REL ADR        NAME     ADDRESS   REL ADR

IEAQFX00    000000     0111FC    SD
```

```
IMBMDMAP - SYS1.NUCLEUS(IEANUC01)                                            DATE=70.091 NUMERICALLY BY RLD ITEM

    LOCATION  REL ADR   IN CSECT  REFERS TO   IN CSECT        LOCATION  REL ADR   IN CSECT  REFERS TO   IN CSECT

                        IEAQFX00  IEACVT      IEAQBK00                                                   IEAQTRCE
                                  IEAQEXC0    IEAQNU00                                       TRPTR       IEAQNU00
                                              IEAQNU00        000054    000054    IEAQFX00  IEAQSC00    IGFNUC00
                                                              000064    000064    IEAQFXOC  IGFN0000    IFBCTA00
                                                              000074    000074    IEAQFX00  IFBACTA     IFFBDA
                                                              000080    000080    IEAQFX00  IFFB106     IFFBDA
                                                              00112D    00112D    IEAQFXC0  IFFB1D1     IFFBDA
                                                              00217D    00217D    IEAQFX00  IFFB1D      IFFBDA
                                                                        0021CD    IEAQFX00  IFFB206     IFFBDA
                                                                                  IEAQFX00  IFFB2D1     IFFBDA
                                                                                                        BDA
                                                                                                        BDA
                                                                                                        BDA
                                                                                                        C001
                                                                                                        C23XX
                                                                                                        AQN
```

```
IMBMDMAP - SYS1.NUCLEUS(IEANUC01)
                CSECT                                                    DATE=70.091 ALPHABETICALLY BY ESD ITEM
  NAME      ADDRESS    LENGTH   TYPE                      ENTRY                            ENTRY
                                                  NAME     ADDRESS   REL ADR                                     EXT REF
$PRIVATE    0267F8     000000    PC                                                NAME    ADDRESS   REL ADR
$PRIVATE    026F28     C00000    PC                                                                              NAME
$PRIVATE    026F28     000000
$PRIVAT
$PRIVAT

DECENT
```

```
IMBMDMAP - SYS1.NUCLEUS(IEANUC01)                                          DATE=70.091 ALPHABETICALLY BY RLD ITEM

    LOCATION   REL ADR   IN CSECT   REFERS TO   IN CSECT       LOCATION   REL ADR   IN CSECT   REFERS TO   IN CSECT

     021184    000004    IFBDCB00   CCHPTTAB    IGFCAT          014F88    000D90    IEAQGM00   CDDESTRY    IGC003
                                                                                    IEAQLK00   CDHKEEP     IGC003
```

```
MVT LINK PACK MAP
                                                                         DATE=70.091 NUMERICALLY BY LOCATION
     LOCATION  LENGTH     NAME        EP ADR  EP REL ADR
                                                                  LOCATION  LENGTH     NAME        EP ADR  EP REL ADR
      078000   000400    IGG02C1Z     C7800C    000000
      078800   000400    IGG0200Y                                  078400   000400    IGG0200Z     078400    000000
```

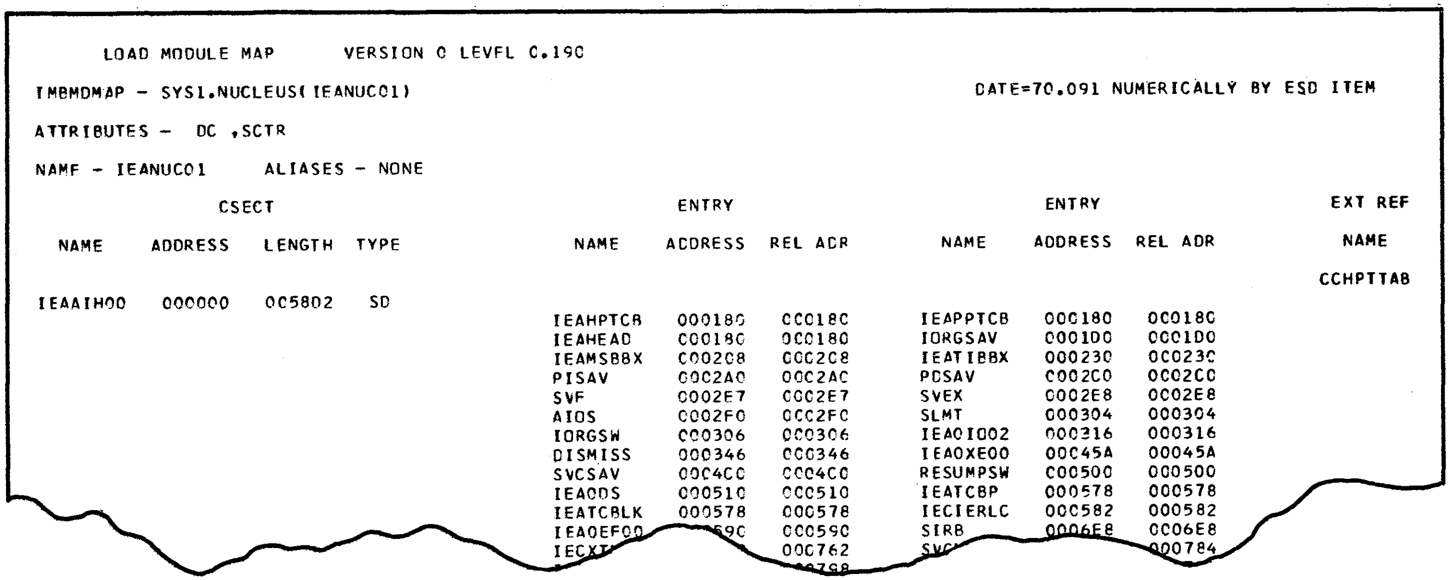
```
MVT LINK PACK MAP                                                      DATE=70.091 ALPHABETICALLY BY NAME

    LOCATION  LENGTH     NAME       EP ADR  EP REL ADR             LOCATION  LENGTH     NAME       EP ADR  EP REL ADR

     07F5D0   000068   IEEPALTR     07F5DC    C00000                07EE18   000060   IEEPDISC     07EE18    000000
     07F048   000068   IEEPPRES     07F04E    0C000C                C7F868   000040   IEEPRTN      07F868    000000
     07F81C   000058   IEEVMNT1     07F81C    00000C                07F8A8   000058   IEEVSTRT     07F8A8    CC000C
     07F638   0001C8   IEEVWILK     07F63E    C00000                07F578   000058   IEFQINTZ     07F578    C00000
     07F4F0   000088   IEFSD102     07F4FC    000000                07EFA0   000060   IEFSD105     07EFAC    0000C0
     07F0B0   000440   IEFSD263     C7F21C    000160                07EE78   000128   IEFVME       07EE78    000000
     07D288   000400   IGC0001I     07D2BE    0C0000                07CC00   000400   IGC0005E     07CC00    00000C
     07EDB0   000068   IGG019AA     07EDBC    000000                07ED48   000068   IGG019AB     07ED48    000000
     07DB70                         07DB7C    0000                  07D968   0000C0   IGGC19AD     07D968
                                    07DD7E
```

Figure MDMAP-9. Excerpts From the Map Resulting From Example 1

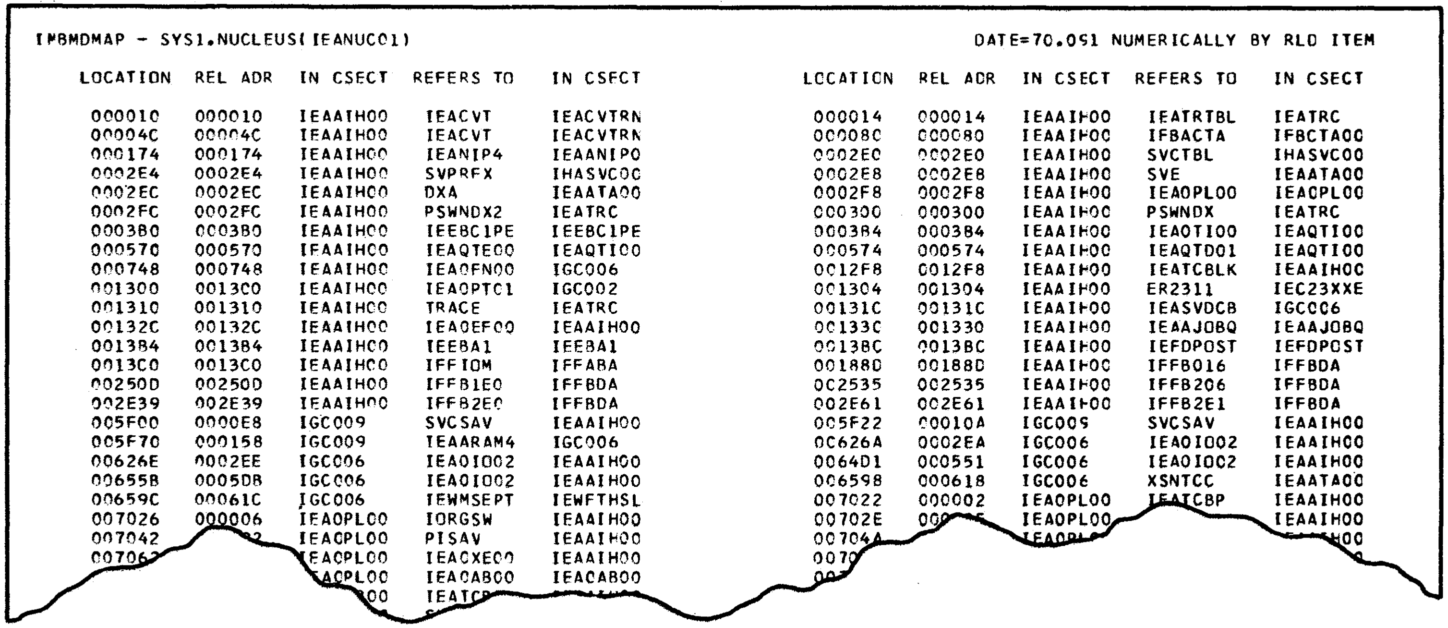
Example 2: Mapping the ESDs of Load Module, Using the Relocation Option

The statements in this example will produce a basic map of a load module in SYS1.LINKLIB with relocation of the relative addresses to base address 10A20. Only the numerically arranged ESD entries are produced. See Figure MDMAP-10 also.

```
//JOB3         JOB        MSGLEVEL=(1,1)
//MAPMOD       EXEC       PGM=IMBMDMAP,PARM='BASIC,10A20'
//DD3          DD         DSN=SYS1.LINKLIB(IEFAS061),DISP=SHR
//SYSPRINT     DD         SYSOUT=A
/*
```



```
     LOAD MODULE MAP      VERSION 0 LEVFL 0.190

IMBMDMAP - SYS1.NUCLEUS(IEANUC01)                                                    DATE=70.091 NUMERICALLY BY ESD ITEM

ATTRIBUTES -  DC ,SCTR

NAMF - IEANUC01     ALIASES - NONE

               CSECT                                    ENTRY                            ENTRY                      EXT REF

  NAME     ADDRESS   LENGTH  TYPE             NAME     ADDRESS  REL ADR         NAME     ADDRESS  REL ADR            NAME

                                                                                                                   CCHPTTAB

IEAAIH00   000000    005802   SD
                                           IEAHPTCB   000180   00018C       IEAPPTCB   000180   00018C
                                           IEAHEAD    000180   000180       IORGSAV    0001D0   0001D0
                                           IEAMSBBX   000208   000208       IEATIBBX   000230   000230
                                           PISAV      0002A0   0002AC       PDSAV      0002C0   0002C0
                                           SVF        0002E7   0002E7       SVEX       0002E8   0002E8
                                           AIOS       0002F0   0002FC       SLMT       000304   000304
                                           IORGSW     000306   000306       IEAOI002   000316   000316
                                           DISMISS    000346   000346       IEAOXE00   00045A   00045A
                                           SVCSAV     0004C0   0004C0       RESUMPSW   000500   000500
                                           IEAODS     000510   000510       IEATCBP    000578   000578
                                           IEATCBLK   000578   000578       IECIERLC   000582   000582
                                           IEAOEF00            000590       SIRB       0006E8   0006E8
                                           IECX                000762                           000784
```



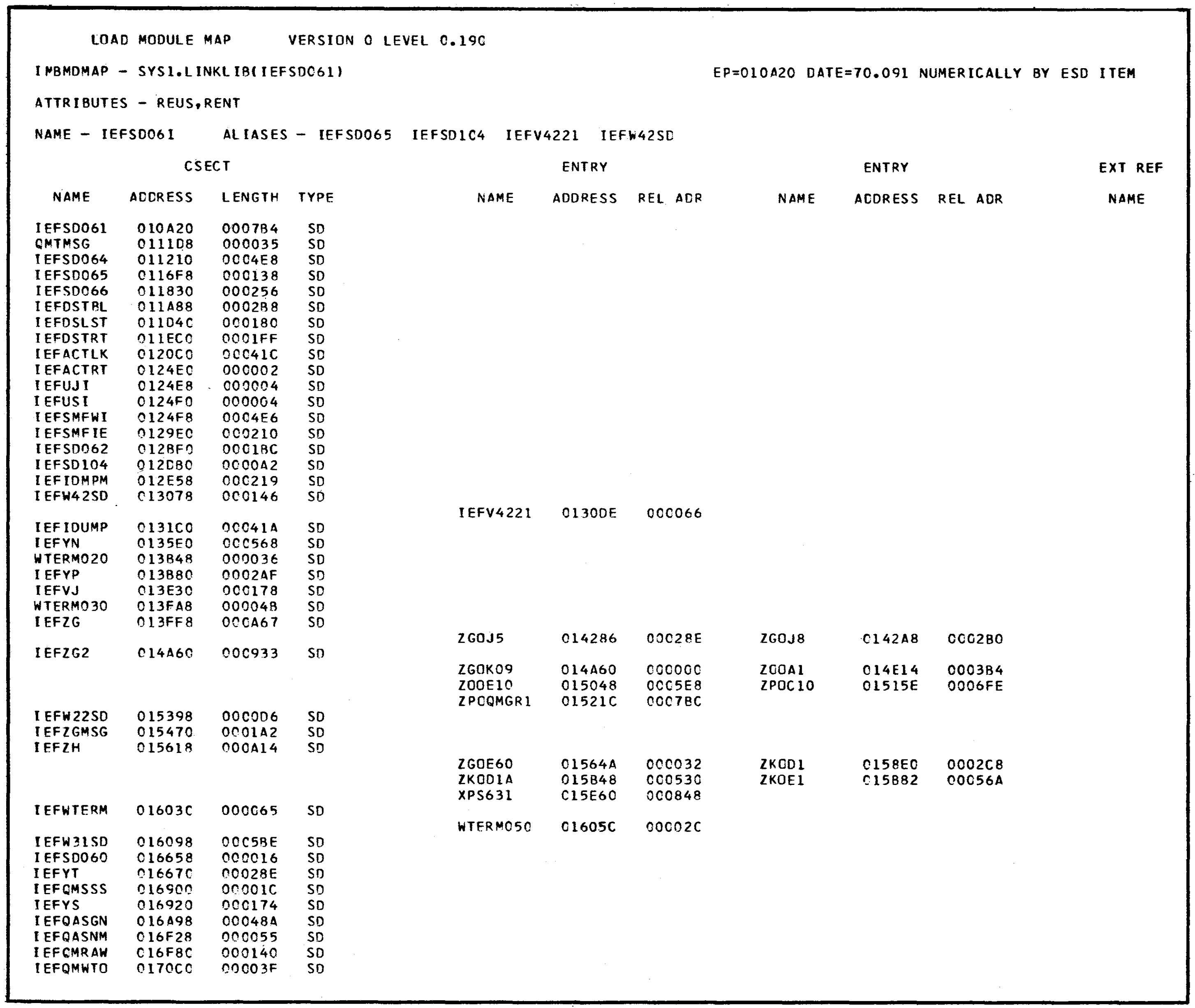
```
IMBMDMAP - SYS1.NUCLEUS(IEANUC01)                                                    DATE=70.091 NUMERICALLY BY RLD ITEM

    LOCATION  REL ADR   IN CSECT  REFERS TO   IN CSECT               LOCATION  REL ADR   IN CSECT  REFERS TO   IN CSECT

     000010   000010    IEAAIH00   IEACVT      IEACVTRN                000014   000014    IEAAIH00   IEATRTBL    IEATRC
     00004C   00004C    IEAAIH00   IEACVT      IEACVTRN                00008C   000080    IEAAIH00   IFBACTA     IFBCTA00
     000174   000174    IEAAIH00   IEANIP4     IEAANIP0                0002E0   0002E0    IEAAIH00   SVCTBL      IHASVC00
     0002E4   0002E4    IEAAIH00   SVPRFX      IHASVC00                0002E8   0002E8    IEAAIH00   SVE         IEAATA00
     0002EC   0002EC    IEAAIH00   DXA         IEAATA00                0002F8   0002F8    IEAAIH00   IEAOPL00    IEAOPL00
     0002FC   0002FC    IEAAIH00   PSWNDX2     IEATRC                  000300   000300    IEAAIH00   PSWNDX      IEATRC
     0003B0   0003B0    IEAAIH00   IEEBC1PE    IEEBC1PE                0003B4   0003B4    IEAAIH00   IEAOTI00    IEAQTI00
     000570   000570    IEAAIH00   IEAQTE00    IEAQTI00                000574   000574    IEAAIH00   IEAQTD01    IEAQTI00
     000748   000748    IEAAIH00   IEAOFN00    IGC006                  0012F8   0012F8    IEAAIH00   IEATCBLK    IEAAIH00
     001300   0013C0    IEAAIH00   IEAOPTC1    IGC002                  001304   001304    IEAAIH00   ER2311      IEC23XXE
     001310   001310    IEAAIH00   TRACE       IEATRC                  00131C   00131C    IEAAIH00   IEASVDCB    IGC006
     00132C   00132C    IEAAIH00   IEAOEF00    IEAAIH00                00133C   001330    IEAAIH00   IEAAJOBQ    IEAAJOBQ
     0013B4   0013B4    IEAAIH00   IEEBA1      IEEBA1                  0013BC   0013BC    IEAAIH00   IEFDPOST    IEFDPOST
     0013C0   0013C0    IEAAIH00   IFFIOM      IFFABA                  00188D   00188D    IEAAIH00   IFFB016     IFFBDA
     00250D   00250D    IEAAIH00   IFFB1E0     IFFBDA                  002535   002535    IEAAIH00   IFFB206     IFFBDA
     002E39   002E39    IEAAIH00   IFFB2E0     IFFBDA                  002E61   002E61    IEAAIH00   IFFB2E1     IFFBDA
     005F00   0000E8    IGC009     SVCSAV      IEAAIH00                005F22   00010A    IGC009     SVCSAV      IEAAIH00
     005F70   000158    IGC009     IEAARAM4    IGC006                  00626A   0002EA    IGC006     IEAOI002    IEAAIH00
     00626E   0002EE    IGC006     IEAOI002    IEAAIH00                0064D1   000551    IGC006     IEAOI002    IEAAIH00
     00655B   0005DB    IGC006     IEAOI002    IEAAIH00                006598   000618    IGC006     XSNTCC      IEAATA00
     00659C   00061C    IGC006     IEWMSEPT    IEWFTHSL                007022   000002    IEAOPL00   IEATCBP     IEAAIH00
     007026   000006    IEAOPL00   IORGSW      IEAAIH00                00702E             IEAOPL00               IEAAIH00
     007042             IEAOPL00   PISAV       IEAAIH00                00704A
     00706              IEAOPL00   IEAOXE00    IEAAIH00                0070
                        IEAOPL00   IEAOAB00    IEAOAB00
                          00       IEATC
```

Figure MDMAP-10. Excerpts From the Map Resulting From Example 2

Example 3: Mapping a Load Module with the DEBUG and Relocation Options

This example includes the DEBUG and relocation options. Because of the nature of the module being mapped, IMBMDMAP produced (in addition to the regulation map items) hexadecimal dumps of the PDS directory and of the load module, and snapdumps 1, 5, and 6 of the series described under the DEBUG parameter discussion. Figure MDMAP-11 shows excerpts from the listing.

```
//JOB4         JOB      MSGLEVEL=(1,1)
//DEBUGDMP     EXEC     PGM=IMBMDMAP,PARM='DEBUG,7FFFF'
//DD4          DD       DSN=SYS1.LINKLIB(IEFW21SD),DISP=SHR
//SYSPRINT     DD       SYSOUT=A
//SNAPDUMP     DD       SYSOUT=A
/*
```



```
      LOAD MODULE MAP      VERSION 0 LEVEL 0.190

IMBMDMAP - SYS1.LINKLIB(IEFSD061)                                         EP=010A20 DATE=70.091 NUMERICALLY BY ESD ITEM

ATTRIBUTES - REUS,RENT

NAME - IEFSD061     ALIASES - IEFSD065  IEFSD104  IEFV4221  IEFW42SD

                 CSECT                                      ENTRY                               ENTRY                         EXT REF

  NAME    ADDRESS  LENGTH   TYPE                   NAME     ADDRESS  REL ADR         NAME     ADDRESS  REL ADR               NAME

IEFSD061   010A20   0007B4   SD
QMTMSG     0111D8   000035   SD
IEFSD064   011210   0004E8   SD
IEFSD065   0116F8   000138   SD
IEFSD066   011830   000256   SD
IEFDSTBL   011A88   0002B8   SD
IEFDSLST   011D40   000180   SD
IEFDSTRT   011EC0   0001FF   SD
IEFACTLK   0120C0   00041C   SD
IEFACTRT   0124E0   000002   SD
IEFUJI     0124E8   000004   SD
IEFUSI     0124F0   000004   SD
IEFSMFWI   0124F8   0004E6   SD
IEFSMFIE   0129E0   000210   SD
IEFSD062   012BF0   0001BC   SD
IEFSD104   012DB0   0000A2   SD
IEFIDMPM   012E58   000219   SD
IEFW42SD   013078   000146   SD
                                                IEFV4221   0130DE   000066
IEFIDUMP   0131C0   00041A   SD
IEFYN      0135E0   000568   SD
WTERM020   013B48   000036   SD
IEFYP      013B80   0002AF   SD
IEFVJ      013E30   000178   SD
WTERM030   013FA8   00004B   SD
IEFZG      013FF8   000A67   SD
                                                ZGOJ5      014286   00028E     ZGOJ8      0142A8   0002B0
IEFZG2     014A60   000933   SD
                                                ZGOK09     014A60   000000     ZOOA1      014E14   0003B4
                                                ZOOE10     015048   0005E8     ZPOC10     01515E   0006FE
                                                ZPOQMGR1   01521C   0007BC
IEFW22SD   015398   0000D6   SD
IEFZGMSG   015470   0001A2   SD
IEFZH      015618   000A14   SD
                                                ZGOE60     01564A   000032     ZKOD1      0158E0   0002C8
                                                ZKOD1A     015B48   000530     ZKOE1      015B82   00056A
                                                XPS631     015E60   000848
IEFWTERM   01603C   000065   SD
                                                WTERM050   01605C   00002C
IEFW31SD   016098   0005BE   SD
IEFSD060   016658   000016   SD
IEFYT      016670   00028E   SD
IEFQMSSS   016900   00001C   SD
IEFYS      016920   000174   SD
IEFQASGN   016A98   00048A   SD
IEFQASNM   016F28   000055   SD
IEFQMRAW   016F8C   000140   SD
IEFQMWTO   0170C0   00003F   SD
```

Figure MDMAP-11. Excerpts From the Map Resulting From Example 3

Operational Considerations

You should pay careful attention to the following points when using IMBMDMAP:

- A maximum of sixteen aliases will be printed, since the linkage editor assigns no more than sixteen.
- Link pack area maps for MFT do not include resident SVC routines.
- If the DEBUG parameter is used, a SNAPDUMP DD statement must be included in the jobstream.
- A DD statement is required for each load module to be mapped.
- To map a nucleus load module, the nucleus must be a member of a partitioned data set named SYSn.NUCLEUS, or incorrect will result.
- To obtain a main storage dump in the event of abnormal termination when using the DEBUG parameter, a SYSABEND or SYSUDUMP DD statement must be included in the jobstream.

Chapter 7: IMCOSJQD

Operates as a problem program to format and print the system job queue.

OSJQD

Contents

INTRODUCTION 157

STARTING OSJQD 158
Restarting the System 158
Invoking OSJQD by JCL 159
Invoking OSJQD from the System Console 159

CONTROLLING OSJQD 162
Defining the Input Data Set .162
Using the Control Statements 162

OSJQD OUTPUT 164
Record Identification Headings 166
Output Comments 167
Error Recovery Procedures 167

JCL AND CONTROL STATEMENT EXAMPLES 169
Example 1: Dumping the Input Job Queues 169
Example 2: Searching the Input Job Queues for a Specific Job . . .169
Example 3: Dumping the Entire Job Queue 169

Figures

Figure OSJQD-1. Sample Job Control Statements Used to Invoke OSJQD 159
Figure OSJQD-2. An Example of a User-Written Cataloged Procedure to Call OSJQD from the System Console . .160
Figure OSJQD-3. A Sample Exchange Between the Operator and OSJQD .161
Figure OSJQD-4. OSJQD Execution Time as a Function of Output and Input Devices 164
Figure OSJQD-5. Sample Job Control Statements and Control Statements Used to Print a 9-Track Tape Containing OSJQD Output 164
Figure OSJQD-6. Sample OSJQD Output, Showing Output Comments . . .165

Introduction

IMCOSJQD is a service aid that formats and prints the contents of the system job queue data set (SYS1.SYSJOBQE). IMCOSJQD is similar in function to the standalone service aid IMCJQDMP; however, IMCOSJQD operates as a problem program under the operating system, using standard access methods. IMCOSJQD can therefore be used without disrupting normal operating system processing; this is a great advantage in a large installation where stopping and restarting the operating system can take a long time.

To save even more time, you can specify that IMCOSJQD output should be stored temporarily on tape rather than printed immediately. The tape can be printed later, at your convenience.

You can use IMCOSJQD to dump the entire job queue, or you can select specific queues within the job queue and their associated logical tracks.

Starting IMCOSJQD

IMCOSJQD resides in the linkage library (SYS1.LINKLIB data set). You can invoke it either through job control statements in the input stream or through the system console.

In almost every case you will run IMCOSJQD to produce a listing that will help you diagnose a problem connected with the job queue. If the problem is relatively minor, and the system can continue processing, you can schedule IMCOSJQD immediately. For more severe problems, when the operating system cannot continue processing, you must restart the system before running IMCOSJQD.

Restarting the System

If the system goes down, first try a system restart (warm start); that is, IPL without reformatting the job queue. If the restart fails, take action as suggested below:

If your installation has a volume containing an alternate SYS1.SYSJOBQE data set, restart the system, requesting that that volume be formatted as the new job queue data set. Then run IMCOSJQD, specifying the original job queue data set as input.

If your installation has more than one operating system, and time is not critical, mount the volume containing the job queue on another system. Then run IMCOSJQD on that system, specifying the transferred data set as input.

If you cannot use an alternate volume, or if the volume containing the job queue data set cannot be moved, dump the job queue data set to another direct access volume with a different volume serial number, as follows:

1. Execute the IBCDMPRS utility to dump the SYS1.SYSJOBQE data set to a direct access device. Use IBCDMPRS control statements like those shown in the following example:

   ```
   DUMP JOB DUMP 2314 ONTO 2314
        DUMP FROMDEV=2314,FROMADDR=230,
             TODEV=2314,TOADDR=232,
             VOLID=ALTQUE
        END
   ```

   For more information about the IBCDMPRS utility program, refer to the publication IBM System/360 Operating System, Utilities, GC28-6586.

2. Restart the operating system, specifying that the job queue should be reformatted. This will establish a fresh job queue.

4. Run IMCOSJQD, specifying the new direct access data set as input.

Invoking OSJQD by JCL

Figure OSJQD-1 shows an example of job control statements used to invoke IMCOSJQD. The statements are described below.

```
//DUMP              JOB        MSGLEVEL=(1,1)
//                  EXEC       PGM=IMCOSJQD
//OSJQDIN           DD         DSNAME=SYS1.SYSJOBQE,
//              UNIT=2314,VOL=SER=111111,DISP=SHR
//OSJQDOUT          DD         UNIT=2400,DISP=(NEW,KEEP),
//              DSNAME=QUEUEOUT,LABEL=(,NL)
//SYSPRINT          DD         SYSOUT=A
[//SYSIN            DD         *]
         .
         .
         .
/*
```

Figure OSJQD-1. An Example of Job Control Statements Used to Invoke IMCOSJQD

EXEC Statement

calls for the execution of IMCOSJQD.

OSJQDIN DD Statement

defines the job queue to be processed. Note that the DD statement that defines the input data set must be named OSJQDIN.

OSJQDOUT DD Statement

defines the output data set. In this case the output data set, named QUEUEOUT, resides on a tape device. Note that the DD statement that defines the output data set must be named OSJQDOUT.

SYSPRINT DD Statement

defines the IMCOSJQD message data set.

OSJQD

SYSIN DD Statement (optional)

defines the data set that contains IMCOSJQD options. In this case, the options follow the job control statements in the input stream. If this statement is omitted, the operator will be prompted to supply options.

Invoking OSJQD from the System Console

If you wish, you can include the job control statements shown in Figure OSJQD-1 as a cataloged procedure in the procedure library (SYS1.PROCLIB data set); this allows the operator to initiate IMCOSJQD processing from the console.

Use the IEBUPDTE Utility to include your IMCOSJQD cataloged procedure in SYS1.PROCLIB. The name you specify in the ADD control statement for IEBUPDTE is the name of the procedure that you must specify in the START command. For information on using IEBUPDTE, refer to the publication IBM System/360 Operating System: Utilities, GC28-6586.

Figure OSJQD-2 shows an example of a cataloged procedure that calls IMCOSJQD.

```
//OSJBQDMP          PROC      REG=20,D='SYS1.SYSJOBQE',U=2314,VS=111111,
//       DSP=SHR,UN=2400,DISP=(NEW,KEEP),DSN=QUEUEOUT
//                  EXEC      PGM=IMCOSJQD,REGION=&REG.K
//OSJQDIN           DD        DSNAME=&D,UNIT=&U,VOL=SER=&VS,DISP=&DSP
//OSJQDOUT          DD        UNIT=&UN,DISP=&DISP,DSNAME=&DSN
//SYSPRINT          DD        SYSOUT=A
/*
```

Figure OSJQD-2. An Example of a User-Written Cataloged Procedure to Call IMCOSJQD from the System Console

PROC Statement

defines the name of the cataloged procedure and default values for any symbolic parameters included in the remaining statements in the procedure. In this case, the defaults are as follows: the input data set is SYS1.SYSJOBQE, the output data set is QUEUEOUT, and the region size is 20K. Note that you can specify any name for the procedure on the PROC statement.

EXEC Statement

calls for the execution of IMCOSJQD, and specifies the region size by a symbolic parameter. (The default region size specified in the PROC statement is 20K; this is the minimum region size required for IMCOSJQD processing.)

OSJQDIN DD Statement

defines the input data set. In this case, symbolic parameters permit the operator to specify an input data set or accept the defaults specified in the PROC statement.

OSJQDOUT DD Statement

defines the output data set. In this case, symbolic parameters permit the operator to specify an output data set or accept the defaults specified in the PROC statement.

SYSPRINT DD Statement

defines the message data set.

Note that the SYSIN DD statement has been omitted from this cataloged procedure; as a result the operator will be prompted to supply options when he starts IMCOSJQD.

Figure OSJQD-3 shows an example of an exchange between the operator and IMCOSJQD while starting IMCOSJQD. Note that in this example the operator made an error the first time he selected dump parameters, and IMCOSJQD prompted him to correct his error.

```
start osjbqdmp,,,reg=24
    .
    .
    .
 00 IMC001A SPECIFY SELECT PARAMETERS OR END
r00,'qcr=cls=c'
 01 IMC002A COMMAND ERROR - ENTER QDUMP PARAMETERS
r01,'qcr=class=c'
 00 IMC001A SPECIFY SELECT PARAMETERS OR END
r00,'qcr=class=g'
    .
    .
    .
IMC005I SPECIFIED QUEUE IS EMPTY
 02 IMC001A SPECIFY SELECT PARAMETERS OR END
r02,'qcr=class=a,jobname=(myjob,youjob,hisjob)'
    .
    .
    .
IMC006I THESE JOBS NOT FOUND
HISJOB
 03 IMC001A SPECIFY SELECT PARAMETERS OR END
r03,'qcr=class=a,jobname=(myjob,herjob)'
    .
    .
    .
 04 IMC001A SPECIFY SELECT PARAMETERS OR END
r04,'end'
    IMC004I QDUMP COMPLETE
```

Figure OSJQD-3. A sample exchange between operator and IMCOSJQD.

Controlling OSJQD

You control IMCOSJQD processing by defining the input data set and by supplying control statements.

Defining the Input Data Set

In most cases, the input to IMCOSJQD will be the system job queue, SYS1.SYSJOBQE. However, IMCOSJQD will accept as input any data set on a direct access device that has the format of the system job queue. This feature is useful when you have transferred the contents of the SYS1.SYSJOBQE data set to another volume, as described earlier in "Preparing to Use IMCOSJQD".

Using the Control Statements

Several control statements allow you to specify how much of the job queue you want IMCOSJQD to format and print. You can enter these control statements in two ways:

- If you invoke IMCOSJQD with JCL and include a SYSIN DD *, you can include control statements as cards in the input stream. If you want more than one dump operation, you must supply a separate card for each dump. IMCOSJQD will process the cards sequentially and produce a separate output listing for each one. (Blank cards will be ignored.) IMCOSJQD will terminate when it reaches end-of-file.

- If you start IMCOSJQD from the console, or if you omit the SYSIN DD * statement from the JCL, IMCOSJQD will prompt you to supply dump options. In reply you should define one dump operation fully. IMCOSJQD will prompt you again when it has finished processing the first dump, and you can then define a new dump operation. If you want to terminate IMCOSJQD processing, you must wait for a prompting message and reply END. (See Figure OSJQD-3.)

There are four IMCOSJQD control statements: QCR= , JOBNAME= , ALL, and END.

$$\text{QCR=}\left\{\begin{array}{l}\text{ASB}\\ \text{CLASS=y}\\ \text{FREE}\\ \text{HOLD}\\ \text{RJE}\\ \text{SYSOUT=x}\\ \text{SUBMIT}\end{array}\right\}$$

specifies that the job queue data set's master queue control record and the queue records associated with the named work queue should be formatted and printed. The parameters are mutually exclusive; if you want more than one specific work queue, you must request separate dump operations for each.

For each QCR= option, IMCOSJQD dumps the master queue control record, the requested minor queue control record, and the logical tracks associated with that minor queue. The QCR= options and the minor queue control records they request are as follows:

ASB	- Automatic SYSIN Batching Queue
CLASS=y	- An input job queue (A through O)
FREE	- Free Track Queue
HOLD	- Hold Queue
RJE	- Remote Job Entry Work Queue
SYSOUT=x	- An output job queue (A through Z and 0 through 9)
SUBMIT	- TSO Background Reader Queue

JOBNAME=(jobname1[...,jobname4])

requests IMCOSJQD to search all fifteen input work queues for logical track areas assigned to the specified jobname(s). These will be dumped along with associated system message blocks and data set blocks.

Note that searching all the input work queues for a job is a time-consuming operation. To reduce this time, use the QCR=CLASS=x control statment in combination with the JOBNAME= control statement to specify the input class of the requested job(s). For this purpose both control statements may be coded on a single card or entered as a single reply to a prompting message. An example of such an entry is:

```
QCR=CLASS=B,JOBNAME=(NEWJOB)
```

ALL

requests a dump of the entire job queue. This is the default option; it will take effect if the operator replies to the message prompting him for dump options by entering r xx,'U'.

IMCOSJQD Output

IMCOSJQD output can be directed either to a printer device or to a scratch tape, from which it can be printed later. Immediate printing can take a long time, so in most cases you should direct IMCOSJQD's output to a tape. Figure IMCOSJQD-4 shows the differences in execution time per 100 tracks between tape and printer output for various devices on which the job queue can reside.

	Output Device	
Queue Device	Printer (1403)	Tape (2400)
2311	11.3 minutes	4.0 minutes
2314	19.5 minutes	6.9 minutes
2301	49.5 minutes	17.4 minutes

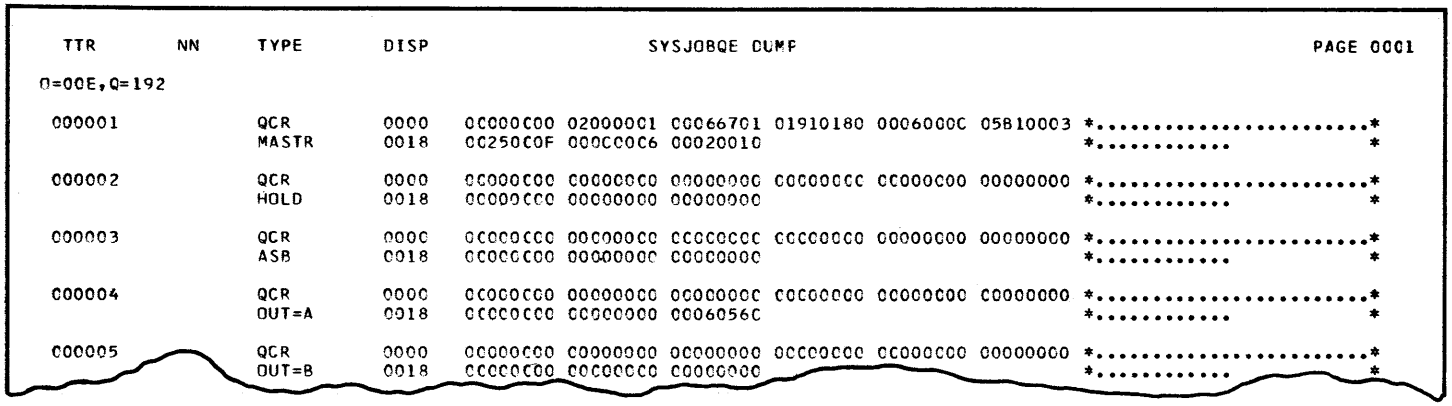
Figure OSJQD-4. IMCOSJQD Execution Time per 100 Tracks of Input as a Function of Output and Input Devices

Once IMCOSJQD's output is on a scratch tape, you can print it at any time using IEBPTPCH. Figure OSJQD-5 shows an example of the job control statements needed for this operation. For more information, refer to the publication IBM System/360 Operating System, Utilities, GC28-6586.

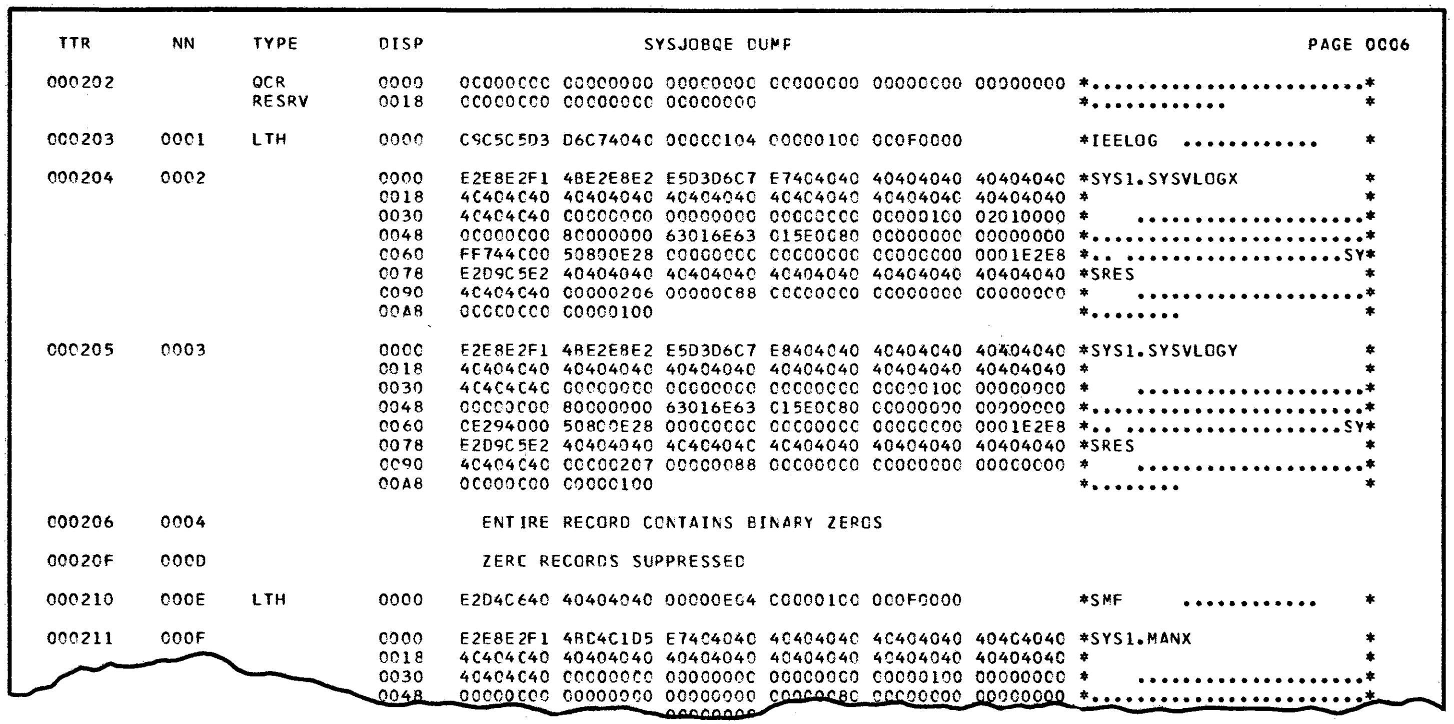
```
//PRINT        JOB       MSGLEVEL=(1,1)
//             EXEC      PGM=IEBPTPCH
//SYSPRINT     DD        SYSOUT=A
//SYSUT1       DD        UNIT=2400,LABEL=(,NL),VOL=SER=QDUMPT,
//        DISP=(OLD,KEEP),DCB=(RECFM=F,BLKSIZE=121,LRECL=121)
//SYSUT2       DD        SYSOUT=A
//SYSIN        DD        *
          PRINT      PREFORM=M
/*
```

Figure OSJQD-5. Sample JCL and Control Statements Used to Print a 9-Track Tape Containing IMCOSJQD Output

Figure OSJQD-6 shows a sample listing of a job queue as produced by IMCOSJQD.



```
  TTR        NN      TYPE        DISP                     SYSJOBQE CUMF                                         PAGE 0001
0=00E,Q=192

 000001              QCR         0000    0C000C00 02000001 C0066701 01910180 0006000C 05B10003 *........................*
                     MASTR       0018    0C250C0F 000C00C6 00020010                            *...........            *

 000002              QCR         0000    0C000C00 C000000C0 0000000C C000000CC CC000C00 00000000 *........................*
                     HOLD        0018    0C000CC0 00000000 00000000                            *...........            *

 000003              QCR         000C    0C000CC0 00C0000C0 CC0C0C0C C0C000C0 00000000 00000000 *........................*
                     ASB         0018    0C000C00 00000000 C0000000                            *...........            *

 000004              QCR         000C    0C0C0C00 0000000C0 0000000C C0C00000 0C000C00 C0000000 *........................*
                     OUT=A       0018    0CC0C0C0 C0C00000 0006056C                            *...........            *

 000005              QCR         0000    00000C00 C0000000 0C000000 0CC0C0C0 CC000C00 00000000 *........................*
                     OUT=B       0018    CCCC0C00 C0C0C0C0 C0000000                            *...........            *
```



```
 TTR        NN      TYPE        DISP                     SYSJOBQE CUMF                                         PAGE 0006

000202              QCR         0000    0C0000CC 00000000 000C000C CC0000C00 00000C00 00000000 *........................*
                    RESRV       0018    CC0C0CC0 C0C0000C 0C000000                            *...........            *

000203      0001    LTH         0000    C9C5C5D3 D6C7404C 0000C104 00000100 0C0F0000           *IEELOG  ...........    *

000204      0002                0000    E2E8E2F1 4BE2E8E2 E5D3D6C7 E7404040 40404040 4040404C *SYS1.SYSVLOGX           *
                                0018    4C404C40 40404040 40404040 4C4C4040 4040404C 40404040 *                        *
                                0030    4C404C40 C0000000 00000000 0000C0CC 0C000100 02010000 *     ...................*
                                0048    0C000C00 80000000 63016E63 C15E0C80 0C00000C 00000000 *........................*
                                C060    FF744C00 50800E28 C000C0CC C0C0C0C0C CC00C000 0001E2E8 *.. ...................SY*
                                0078    E2D9C5E2 40404040 40404040 40404040 40404040 40404040 *SRES                    *
                                C090    4C404C40 C0000206 00000C88 CCC0C0C0 CC000000 C0000000 *     ...................*
                                00A8    0CCC0CC0 C0000100                                     *.......                 *

000205      0003                000C    E2E8E2F1 4BE2E8E2 E5D3D6C7 E8404040 40404040 40404040 *SYS1.SYSVLOGY           *
                                0018    4C404C40 40404040 40404040 40404040 40404040 40404040 *                        *
                                0030    4C4C4C4C 00000000 0C0000C0 C0C0C0C0 C0000100 00000000 *     ...................*
                                0048    0C0C0C00 80000000 63016E63 C15E0C80 C0000000 00000000 *........................*
                                0060    CE294000 508C0E28 000C0C0C CCC0000C 00000C00 0001E2E8 *.. ...................SY*
                                0078    E2D9C5E2 40404040 4C4C404C 4C404040 40404040 40404040 *SRES                    *
                                CC90    4C404C40 C0C0C207 C0000088 0CC000C0 C0000000 00000C00 *     ...................*
                                00A8    0C000C00 C0000100                                     *.......                 *

000206      0004                          ENTIRE RECORD CONTAINS BINARY ZEROS

00020F      000D                          ZERC RECORDS SUPPRESSED

000210      000E    LTH         0000    E2D4C640 40404040 00000E04 C000010C 000F0000           *SMF     ...........    *

000211      000F                0000    E2E8E2F1 4BC4C1D5 E74C4040 4C40404C 4C404040 4040404C *SYS1.MANX               *
                                0018    4C404C40 40404040 40404040 40404040 40404040 4040404C *                        *
                                0030    4C404C40 C0C0000C0 0000000C 00000000 C0000100 000000C0 *     ...................*
                                0048    00000C00 00000000 00000000 C000C08C CCC00C00 00000000 *........................*
                                                          00000000
```

Sample OSJQD-6. Sample OSJQD Output, Showing Output Comments

Record Identification Headings

For each record, IMCOSJQD supplies information under the following headings:

TTR

Direct access address, relative to the beginning of the data set, for QCR and logical track records.

NN (not supplied for queue control records)

gives the sequence number of the logical track record within the specific work queue. This is a hexadecimal number assigned to each new record as it is added to the queue. The first logical track header record in the queue is always 1; for each new record added to the queue, the value of nn is increased by 1.

TYPE

identifies the record type. IMCOSJQD recognizes the record type in two ways: queue control records and logical track header records are identified through their position in the structure of the job queue. Records from the logical track area are identified by the value in the ID field of each record (byte 4, at offset X'03').

The following table shows the type labels and their significance. Where applicable, the ID field value is also shown.

TYPE	ID	RECORD
QCR	-	Queue Control Record.
LTH	-	Logical Track Header Record
ACT	01	Account Control Table
DSB	15	Data Set Block
DSENQ	0F	Data Set Enqueue Table
DSNT	07	Data Set Name Table
JCT	00	Job Control Table
POT	0A	Procedure Override Table
SCT	02	Step Control Table
SCTX	0C	Step Control Table Extension
SIOT	03	Step Input Output Table
SMB	05	System Message Block
VOLT	06	Volume Table

If no TYPE identifier is shown in the listing, the record is either a job file control block (JFCB), job file control block extension (JFCBX), or system output class directory (SCD), etc.

The column headed TYPE in the listing also identifies the name of the specific work queue associated with a queue control record. The following table shows the work queue identifiers and their significance.

ASB	Automatic SYSIN Batching Queue
CLS=y	System Input Job Class Queues; y is the class identifier (A through O).
HOLD	Hold queue
MASTR	Master queue control record.
OUT=x	SYSTEM Output Class Queues; x is the class identifier (A through Z and 0 through 9).
RESRV	Reserved queue control records.
RJE	Remote Job Entry Queue.
SUBMT	Background Reader Queue

DISP

gives the displacement within a record of the next hexadecimal word to be printed on the listing. The first word of the first printed line for a given record has a displacement of X'0000'; the first word of the second printed line, if one exists, has a displacement of X'0018'.

Output Comments

IMCOSJQD does not dump records that consist entirely of binary zeroes. Instead, when it comes to an all-zero record, it prints

ENTIRE RECORD CONTAINS BINARY ZEROES

and supplies TTR and NN information as described in the previous section. If IMCOSJQD comes to subsequent all-zero records, it will stop printing records until it comes to the next non-zero record or the next logical track header record. To indicate that all-zero records are not being printed, IMCOSJQD prints

ZERO RECORDS SUPPRESSED

See Figure OSJQD-6 for an example of an output listing showing these comments.

Error Recovery Procedures

IMCOSJQD error recovery depends on what kind of dump is being produced, what record was being read when the error occurred, and how many times the error has already occurred.

If you have requested a full dump (by specifying ALL when starting IMCOSJQD), IMCOSJQD will attempt to recover from all errors except those that occur while reading the master queue control record. To recover, IMCOSJQD prints an output error indicator, attempts to print the record associated with the error, and proceeds by reading the next record. If IMCOSJQD could not read the record associated with the error, it prints an appropriate output error indicator on the output listing, and then continues processing with the next queue record.

IMCOSJQD will permit up to 20 consecutive errors to occur before abandoning its attempts to recover. After the twentieth consecutive error, however, it will issue message IMC016I (PERMANENT I/O ERROR ON OSJQDIN), print the contents of the SYNAD buffer, and obtain the next dump option.

If you have requested a selective dump, or if an error occurs while reading the master queue control record, IMCOSJQD does not attempt to recover from any errors. It prints the record associated with the error or an output error indicator, issues message IMC016I, prints the contents of the SYNAD buffer, and obtains the next dump option. It does this by searching the SYSIN data set, if control statements were entered from the input stream, or by prompting the operator to supply dump options, if control statements were entered from the console. It will not terminate processing unless it encounters an END control statement or an end-of-file condition.

The error messages and their meanings are as follows:

badttr - INVALID TTR

IMCOSJQD will print this line in place of the record it could not find, followed by the contents of the SYNAD buffer.

UNABLE TO READ RECORD

An input/output error occurred while IMCOSJQD was trying to read a queue record. IMCOSJQD prints the TTR and NN values associated with the record, and substitutes this message for the contents of the record itself. The message is followed by the contents of the SYNAD buffer.

I/O ERROR READING FOLLOWING RECORD

An input/output error occurred while IMCOSJQD was trying to read a queue record; the error did not prevent IMCOSJQD from reading the record. IMCOSJQD prints this message to indicate that the record contains an error, and follows it with the record itself and the contents of the SYNAD buffer. It also prints the TTR and NN values assocaited with the record.

INVALID LENGTH RECORD

IMCOSJQD has encountered a record which is not a standard length (for a normal queue record, standard length is 176 bytes; for logical track header records, 20 bytes; for queue control records, 36 bytes). IMCOSJQD prints this message, followed by the record and its associated TTR and NN values. No SYNAD information is included.

JCL and Control Statement Examples

The following examples illustrate some of the functions that IMCOSJQD can perform.

Example 1: Dumping the Input Job Queues

This example shows how to format and print three input job queues, the automatic SYSIN batching queue, and two output job queues. Note that the only JCL statement shown is the SYSIN DD statement; for an example of the other JCL statements required to invoke IMCOSJQD, see Figure OSJQD-1.

```
//SYSIN       DD        *
QCR=CLASS=A
QCR=CLASS=B
    QCR=CLASS=C
         QCR=ASB
  QCR=SYSOUT=A
QCR=SYSOUT=B
/*
```

Note that each control statement requests a separate queue, and that the control statements are entered in free form.

Example 2: Searching the Input Queues for a Specific Job

This example shows how to combine the QCR= and JOBNAME= control statements to search a limited number of queues for specific jobs. Note that the only JCL statement shown is the SYSIN DD statement; for an example of the other JCL statements required to invoke IMCOSJQD, see Figure OSJQD-1.

```
//SYSIN  DD   *
    QCR=CLASS=A,JOBNAME=(MYJOB,YOURJOB,HISJOB,HERJOB)
/*
```

Note that the maximum of four jobnames are specified in the JOBNAME= control statement.

Example 3: Dumping the Entire Job Queue

This example shows how to dump the entire job queue. Note that the only JCL statement shown is the SYSIN DD statement; for an example of the other JCL statements required to invoke IMCOSJQD, see Figure OSJQD-1.

```
//SYSIN  DD   *
    ALL
/*
```

Coding the ALL control statement has the same effect as replying r xx,'U' to message IMC001A.

Chapter 8: IMDPRDMP

Formats and prints dumps, TSO swap data set, and GTF trace data. →

Contents

INTRODUCTION 175

FUNCTIONS 177
Formatting Control Blocks 177
Editing GTF Trace Data 177
Dumping the TSO Swap Data Set 177
Clearing SYS1.DUMP 177
Selective Printing 177
  Mapping Resident System Modules 178
  Tracing Queue Control Records 178

JOB CONTROL LANGUAGE STATEMENTS 179
Input DD Statements 180
Output DD Statements 181

USER CONTROL STATEMENTS 183
Function Control Statements 183
Format Control Statements .185
EDIT Control Statement 187
  EDIT Parameters 187
  EDIT Parameter Defaults and Priorities 191
Combining Control Statements 192

ALLOCATING SPACE FOR THE OUTPUT DATA SET 193
Specifying the Maximum Output Block Size 193
Increasing the Space Allocated to SYSOUT 193
Calculating Space Requirements by Block Size 194
Calculating Space Requirements for EDIT Output 194
  Editing Internal Trace Data 194
  Editing an External Trace Data Set 195

CATALOGED PROCEDURE 196

PRDMP OUTPUT 197

JCL AND CONTROL STATEMENT EXAMPLES 208
Example 1: Using the Cataloged Procedure 208
Example 2: Transferring a Dump Data Set 209
Example 3: Processing Multiple Data Sets 210
Example 4: Processing a TSO Dump 212
Example 5: Recording the TSO Swap Data Set 213
Example 6: Editing GTF Trace Data From a Dump 215
Example 7: Editing a GTF Trace Data Set 216

PRDMP

Figures

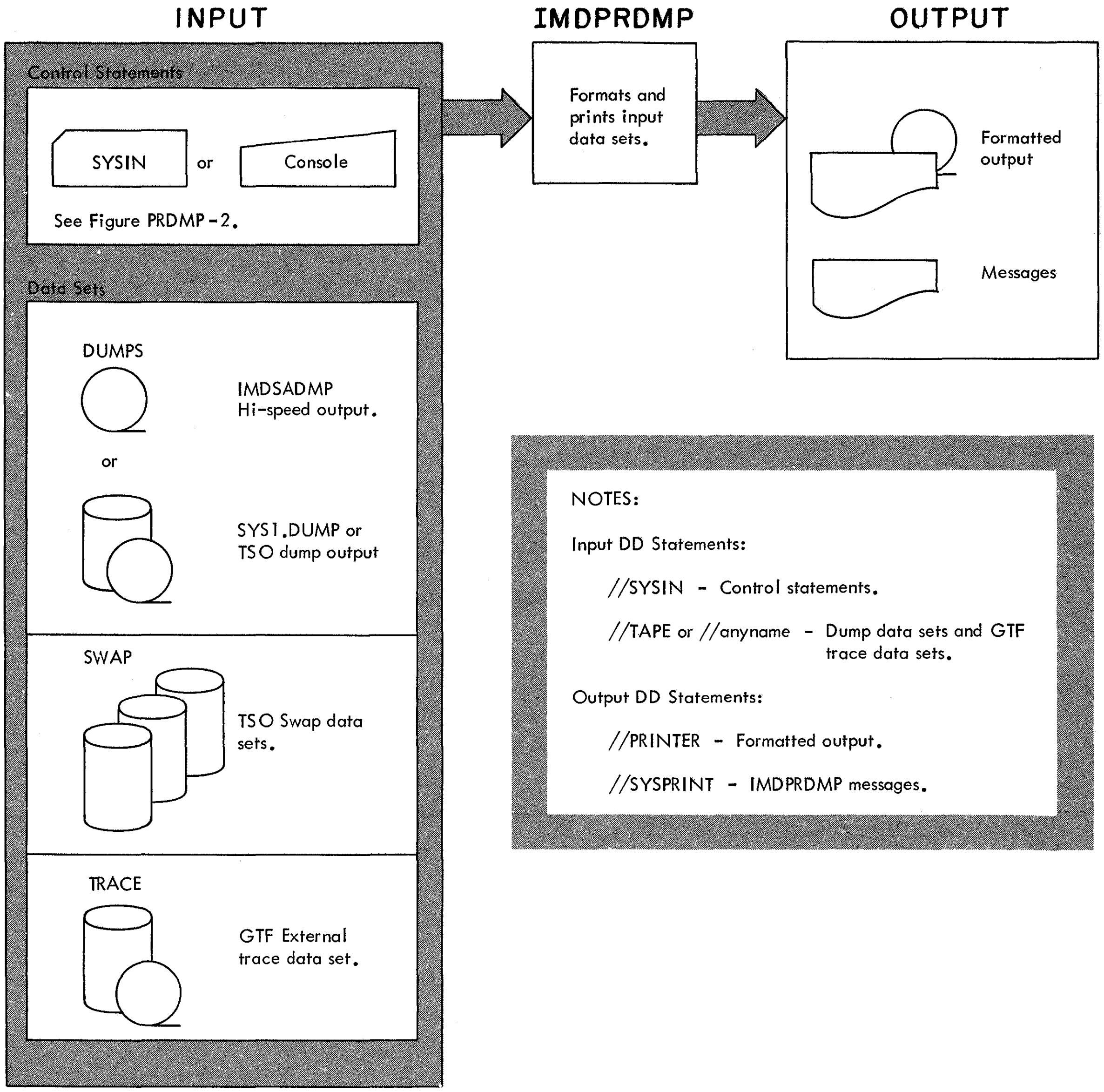
Figure PRDMP-1. IMDPRMDP Input and Output 176
Figure PRDMP-2. IMDPRDMP Function and Format Control Statements, Standard and Abbreviated Forms 183
Figure PRDMP-3. Format of the EDIT Control Statement, Showing All Valid Keywords 188
Figure PRDMP-4. Priorities and Effects of EDIT Keywords Used to Select Records by Trace Event Type 192
Figure PRDMP-5. Number of Lines of EDIT Output Per Buffer As a Function of Maximum Buffer Size and Trace Type 195
Figure PRDMP-6. The Cataloged Procdure PRDMP 196
Figure PRDMP-7. Sample Queue Control Block Trace 197
Figure PRDMP-8. Sample MVT Link Pack Area Map 198
Figure PRDMP-9. Sample MVT Major Control Block Format 199
Figure PRDMP-10. Sample MFT Major Control Block Format 200
Figure PRDMP-11. Sample TSO Control Block Format (Part 1 of 3) . . .201
Figure PRDMP-11. Sample TSO Control Block Format (Part 2 of 3) . . .202
Figure PRDMP-11. Sample TSO Control Block Format (Part 3 of 3) . . .203
Figure PRDMP-12. Sample TCB Summary for MVT or MFT With Subtasking .204
Figure PRDMP-13. Sample TCB Summary for MFT Without Subtasking . . .205
Figure PRDMP-14. Sample Dump - General Format 206
Figure PRDMP-15. Sample EDIT for Trace Data Set 207

IMDPRDMP is a service aid that prints system dump and trace information. Its principal function is to save you time; it does this by producing formatted output that you can scan quickly and easily. Within certain limits, it even allows you to suppress formatting and printing of information that does not interest you.

IMDPRDMP can process the following kinds of input:

- Dump data sets. These include:
    - IMDSADMP high-speed dump data set.
    - SYS1.DUMP data set.
    - TSO dump data set.
- TSO swap data sets.
- GTF trace data. This may exist as:
    - GTF external trace data set (usually called SYS1.TRACE).
    - GTF trace data in buffers within a main storage dump.

Figure PRDMP-1 shows the general characteristics of these types of input and how they relate to IMDPRDMP processing.



NOTES:

Input DD Statements:

//SYSIN - Control statements.

//TAPE or //anyname - Dump data sets and GTF trace data sets.

Output DD Statements:

//PRINTER - Formatted output.

//SYSPRINT - IMDPRDMP messages.

Figure PRDMP-1. IMDPRMDP Input and Output

Functions

You vary the formatting and printing of a dump by supplying IMDPRDMP control statements. You can enter these either as replies to prompting messages issued to the console, or as cards in the input stream.

The control statements provide the following functions:

Formatting Control Blocks

You can specify one control statement (FORMAT) that will cause IMDPRDMP to format all major system control blocks for each task in the system. When printed, the formatted output will look like a SYSABEND dump. Note: IMDSADMP low-speed dump tapes can be printed using IMDPRDMP, but they will not be formatted.

Editing GTF Trace Data

IMDPRDMP can format GTF trace data either as records in the trace data set or as buffers contained in a dump data set. You can edit trace data by specifying special keywords in the EDIT control statements. You can also write exit programs to inspect the data before IMDPRDMP formats it. Suggestions on how to write a user exit program will be provided in the Appendix: Writing EDIT User Programs.

Dumping the TSO Swap Data Set

If a failure occurs in the TSO subsystem or in the operating system, it is important to capture the TSO SWAP data set quickly so that TSO can be restarted without undue delay. You can do this by executing IMDPRDMP against a SWAP data set and a dump data set, and directing its output to tape. The tape may be printed later, at your convenience.

Clearing SYS1.DUMP

You can use IMDPRDMP to transfer the contents of the SYS1.DUMP data set to another data set for later formatting and printing at a more convenient time. This allows you to clear the SYS1.DUMP data set and resume processing without pausing to print the contents.

Selective Printing

In a single control statement called PRINT, you can specify precisely what areas of main storage you want IMDPRDMP to print. IMDPRDMP will format and print control blocks that are associated with specified areas of main storage, unless you specify only PRINT NUCLEUS or PRINT STORAGE.

PRINT allows you to specify printing of main storage areas that are associated with:

- A certain jobname.
- The current task.
- The task terminated by the damage assessment routine (DAR), where applicable.

You can also choose printing of the nucleus, system queue area, and/or all of allocated main storage.

Other control statements provide the following functions:

Resident System Module Mapping

IMDPRDMP can generate a link pack area map (MVT) or a resident reenterable load module area map (MFT). These maps describe resident system modules that were loaded into main storage by the nucleus initialization program (NIP). If you request a map, it will be printed on a separate page or pages of the IMDPRDMP formatted dump listing. These maps are useful in diagnosing system failures that occurred in program modules residing outside the user's region or partition.

Queue Control Block Trace

IMDPRDMP can provide a separate listing of the formatted queue control blocks for all task control blocks in the system. This listing, known as a QCB trace, may be used to resolve problems arising from task contention or system interlock.

Job Control Language Statements

Job control statements are important in determining what functions IMDPRDMP is to perform. This section describes the JCL statements that have special significance in executing IMDPRDMP. For more complete information about using JCL statements, refer to the publication IBM System/360 Operating System: Job Control Language Reference, GC28-6704.

JOB Statement

initiates the job, and provides the opportunity to override the default region size. IMDPRDMP requires a minimum region size of 64K. In most cases it executes more efficiently if its region size is larger than the minimum.

EXEC Statement

calls for the execution of IMDPRDMP and specifies certain actions that IMDPRDMP should take. The operands are:

PGM=IMDPRDMP

identifies IMDPRDMP to the system. This is the only required operand.

PARM='[n][,T][,FREEnnn][,LINECNT=nn][,S][,ER=x]'

n should be used only when the input is a dump data set. It specifies what IMDPRDMP should do if it detects a permanent I/O error or format error while processing a dump.

0 -- print the nucleus (and the system queue area in MVT)

1 (or n not specified) -- print the entire input data set.

2 -- read the next control card from the SYSIN data set, or request control statements from the operator.

T specifies that the operator should be prompted to supply a title for the listing. If T is not specified, no prompting will occur.

FREEnnn specifies the size of the work space within IMDPRDMP's region or partition, excluding the size of the root module, control module, service modules, and input buffer area. nnn is the number of K-bytes in the work space. The default is 8K. This value is usually adequate; however, if the input data set is very large or complex, use the FREEnnn parameter to specify a larger work space. Also, if you need additional storage for a work area in an EDIT user program, use the FREEnnn parameter to reserve it.

LINECNT=nn specifies the number of lines per page to be printed on the output listing. The value specified for nn may be any decimal integer greater than 10. If this parameter is omitted, LINECNT=58 is assumed.

S instructs IMDPRDMP to issue a message which the operator may reply to at any time during processing. In his reply, the operator may stop IMDPRDMP from processing the current input data set and start a new phase of IMDPRDMP execution.

ER=x specifies what action the EDIT portion of IMDPRDMP should take if it detects an error in an exit or format routine while editing trace data from a dump or trace data set. The valid values of x and their meanings are:

0 -- EDIT will display in hexadecimal the record associated with the error and ignore the faulty routine in subsequent processing. If the error was in a format routine, all subsequent records that require processing by the same format routine will be ignored. If the error was in an exit routine, record formatting will continue.

1 -- EDIT will display in hexadecimal the record associated with the error and ignore the faulty routine in subsequent processing. If the error was in a format routine, all subsequent records that require processing by the same format routine will be dumped in hexadecimal. If the error was in an exit routine, record formatting will continue.

2 -- EDIT will display in hexadecimal the record associated with the error; EDIT will then terminate, and the next IMDPRDMP verb will be executed.

3 -- EDIT will allow ABEND to get control if a program check occurs in an exit or format routine. (If ER=3 is not specified, EDIT will issue the SPIE macro before entering the exit routine or format appendage and thus bypass ABEND processing.) If the recognized error is not a program check, the associated record will be dumped in hexadecimal; then EDIT will terminate and the next IMDPRDMP verb will be executed.

If this value is not included in the PARM= parameter list, a value of ER=2 will be assumed. Note that ER=1 and ER=2 are the same for exit programs.

Input DD Statements

$\left\{\begin{array}{l}\text{TAPE}\\ \text{anyname}\end{array}\right\}$ DD Statement

defines an input dump or trace data set, which may reside on direct access storage or on tape. If the input data set is a dump, you can specify any ddname. Remember, however, that for ddnames other than TAPE, you must use a NEWDUMP control statement to identify the input data set. You can define any number of input data sets, as long as each is identified by a different ddname, and each ddname except TAPE is specified in a separate NEWDUMP control statement.

If the input is a GTF trace data set, the ddname must be the same as the one specified in the DDNAME parameter of the EDIT control statement. You can define any number of trace data sets, provided that you identify each data set with a unique ddname and a separate EDIT control statement.

Here are some of the parameters that you may use to describe each input data set; note that you may also need other parameters to describe certain types of input data set. For more information about DD statement parameters, refer to the publication Job Control Language Reference, GC28-6704.

```
* DSNAME=name   (for direct access only)
VOL=SER=volser
UNIT=ddd
* LABEL={(,NL)}   (for tape only)
        {(,SL)}
DISP=OLD
DCB=(BUFNO=number,BLKSIZE=size)  (for trace data sets only)
```

* If the input is a trace data set on a standard label tape, you must include the DSNAME= parameter and code the LABEL= parameter as LABEL=(,SL).

Use the DCB parameter to specify a greater blocksize or more input buffers, or both, if you think the default values will be inadequate. The default blocksize is 3500 bytes; the default number of buffers is 2.

Do not specify a file sequence number in the LABEL= parameter if you intend to use the NEWTAPE or NEWDump FILESEQ=x control statement.

If you omit the TAPE DD statement, IMDPRDMP assumes that the input data is in the SYSUT1 data set, and has the correct format.

SYSWAPmn DD Statement

defines the TSO swap data set(s). With one possible exception, the operands should be identical to those used in the TSO procedure; the exception is that if the TSO procedure is coded DISP=(NEW,KEEP), the IMDPRDMP SYSWAPmn DD statment should be coded DISP=(OLD,KEEP). For an explanation of the values for m and n, refer to the TSO Guide.

SYSIN DD Statement

defines the data set that contains the IMDPRDMP control statements. (If you want to enter control statements from the console, omit this statement.)

Output DD Statements

PRINTER DD Statement

defines the IMDPRDMP output data set.

SYSPRINT DD Statement

defines the IMDPRDMP message data set.

SYSUT1 DD Statement (optional if input is a dump data set on tape, not used if input is an external trace data set)

defines a direct access work data set in which IMDPRDMP can collect input data. Performance improves when a SYSUT1 DD statement is included, because IMDPRDMP can reference dump information directly rather than searching for records in a sequential data set.

Required parameters are:

```
UNIT=ddd

SPACE=(2052,(n,10))
```

n is calculated as (K/2048)+1, where K is the number of bytes of input data.

SYSUT2 DD Statement

identifies a data set into which IMDPRDMP may transfer the contents of the SYS1.DUMP data set when time will not permit immediate formatting and printing of the SYS1.DUMP data set. For more information about this function, refer to the section "Transferring a Dump Data Set" later in this chapter.

User control statements allow you to select specific dump formatting options and control basic operation of the IMDPRDMP program.

IMDPRDMP will prompt you to supply control statements if no SYSIN data set exists, or if the supply of control statements in the SYSIN data set is exhausted before IMDPRDMP finds an END control statement.

There are two kinds of user control statements: function control statements and format control statements. All the control statements are fully described below. Figure PRDMP-2 shows the complete format of the function control statements.

Function Control Statements

Standard Form	Abbreviated Form
CVT={hhhhhh { P	C={hhhhhh { P
NEWDUMP [DDNAME={TAPE }] [,FILESEQ=nn] [,DUMPSEQ=nn] {anyname}	ND [DD={TAPE }] [,F=nn] [,D=nn] {anyname}
NEWTAPE	N
GO	G

ONGO [QCBTRACE] [,LPAMAP] [,FORMAT] [,CVT=parm]
{[,PRINT parm]}
{[,TSO parm] }
{[,EDIT parm] }
	O [Q] [,L] [,F] [,C=parm]
	{[,P parm] }
	{[,TSO parm]}
	{[,E parm] }

TITLE text	T text
END	EN

Format Control Statements

Standard Form	Abbreviated Form
QCBTRACE	Q
LPAMAP	L
FORMAT	F
PRINT [ALL] [,CURRENT] [,NUCLEUS] [,STORAGE=(parm)] [,JOBNAME=(parm)] [,F03]	P [A] [,C] [,N] [,S=(parm)] (,J=(parm)] [,F]

TSO [SYSTEM={YES }] [,USER={PRINT }]
{USER} {STORAGE}
{NO } {FORMAT }
{NO }
	TSO [S={YES }] [,U={PRINT }]
	{USER} {STORAGE}
	{NO } {FORMAT }
	{NO }

EDIT parm	E parm

Figure PRDMP-2. IMDPRDMP Function and Format Control Statements, Standard and Abbreviated Forms

Function Control Statement

The function control statements allow you to control certain operations of the IMDPRDMP program, such as input tape handling, dump listing titles, job termination, etc.

```
CVT={hhhhhh}
    {P     }
```

allows you to specify the address of the communications vector table (CVT) in the main storage dump information. Use this if you think that the CVT pointer, in main storage location X'4C' of the system that was dumped, has been destroyed. If you omit this control statement, and IMDPRDMP cannot locate the CVT at location X'4C', it will scan the dump data set for unique identifiers associated with the CVT. If IMDPRDMP cannot locate the CVT by this scanning process, it will not format the input but will instead take action as specified by "n" in the parameter list supplied in the PARM= operand of the EXEC statement. Once the CVT has been located, it remains in effect until a NEWDUMP or NEWTAPE control statement is encountered.

hhhhhh

is a hexadecimal address specifying the location of the CVT in the input dump information.

P

specifies that the location found at X'4C' in the system on which IMDPRDMP is being executed can be used as a valid pointer to the CVT of the dumped system.

```
NEWDUMP  DDNAME={TAPE   }[,FILESEQ=n][,DUMPSEQ=n]
                {anyname}
```

defines an input data set. If you want to process more than one input data set in a single execution of IMDPRDMP you must supply a separate NEWDUMP control statement for each. If there is only one input data set, NEWDUMP is not needed.

NEWDUMP has three keyword parameters:

DDNAME=

gives the ddname of the input dump data set. This parameter is not required if the TAPE DD statement describes the input data set.

FILESEQ=

identifies the sequence number of an input data set that is one of several data sets on a single magnetic tape volume. If this parameter is omitted, IMDPRDMP assumes a default value of FILESEQ=1.

DUMPSEQ=

specifies the sequence number of a TSO dump that is one of several TSO dumps in a single data set. If this parameter is omitted, IMDRPDMP assumes a default value of DUMPSEQ=1.

NEWTAPE

has the same function as the NEWDUMP statement with parameters specified as DDNAME=TAPE, FILESEQ=1, and DUMPSEQ=1. Use it when the TAPE DD statement defines a single tape device on which are to be mounted multiple volumes, each containing one dump data set.

GO

specifies a predefined set of format control statements. They are: QCBTRACE, LPAMAP, FORMAT, EDIT, and PRINT ALL. The effects of the GO control statement may be overridden by the ONGO control statement, which is described next.

```
ONGO [QCBTRACE][,LPAMAP][,CVT=parm][,FORMAT][,PRINT parm]
     [,EDIT parm][,TSO parm]
```

overrides the predefined set of format control statements requested by the GO control statement. The new set of format control statements will remain in effect for all subsequent uses of the GO control statement, until IMDRPDMP ends or a new ONGO control statement is entered. An ONGO control statement with no parameters restores the original GO functions: QCBTRACE, LPAMAP, FORMAT, EDIT, and PRINT ALL.

NOTE: The ONGO-GO combination is not required for IMDPRDMP execution. You need not specify GO unless you want to use a predefined set of IMDPRDMP options; you need not use ONGO unless you want to change that predefined set. Each IMDPRDMP control statement may be specified directly at any time.

I'ITLE text

specifies a title to be printed at the top of each page in the output listing. Use this statement if you do not expect IMDPRMDP to prompt you to supply title information; that is, if you did not specify T in the PARM= field of the EXEC statement or if you are not entering control statement from the console. You can specify any title up to 62 characters in length.

END

signals IMDPRMDMP to stop processing, close all data sets, and return control to the system control program. (If END is the only control statement specified, IMDPRDMP will load the data set defined by the SYSUT2 DD statement. See Example 1.)

Format Control Statements

Format control statements allow you to choose particular parts of the input to be formatted and printed.

QCBTRACE

requests a trace of the queue control blocks (QCBs) in the input data set.

LPAMAP

causes IMDPRDMP to format and list the contents of the link pack area (MVT) or the resident reenterable load module area (MFT) in the input data set. If the input data set does not contain these areas, LPAMAP wlll be ignored.

FORMAT

causes IMDPRDMP to format and print the contents of the major system control blocks in the input data set.

PRINT [ALL][,CURRENT][,NUCLEUS][,STORAGE=(addresses)]
[,JOBNAME=(jobnames)][,F03]

indicates which parts of the input data set IMDPRDMP should print, according to several parameters.

ALL

instructs IMDPRDMP to print the nucleus, the system queue area, and all allocated regions of main storage in the input data set. This parameter also requests printing of the dumped system's registers.

CURRENT

instructs IMDPRDMP to print only the area of main storage that was associated with the current task when the input data set was created. This parameter also requests printing of the dumped system's registers.

NUCLEUS

instructs IMDPRDMP to print the nucleus portion of the input data set. If the input data set was taken from a system that was executing under MVT, the system queue area will also be printed. For the IBM System/360 Model 65 Multiprocessor, both the high and the low prefixes will be shown on the dump listing.

STORAGE=(startaddr1,endaddr1,...[,startaddrn,endaddrn])

allows you to supply beginning and ending addresses of areas in the input data set that you want printed. You may specify any number of pairs of hexadecimal addresses, so long as the beginning address in each pair is lower than the ending address. If you specify a beginning address and no ending address, IMDPRDMP prints the entire contents of main storage starting at the address you specify. If you do not specify any addresses, IMDPRDMP will print the entire contents of main storage, whether allocated or not. If you specify this parameter at all, IMDPRDMP will also print the dumped system's registers.

JOBNAME=(jobname1,jobname2...,jobname10)

allows you to limit the scope of the output listing to areas in main storage that are associated with specific jobs. You can specify up to ten jobnames. IMDPRDMP will print the areas associated with each job name in the order specified in the JOBNAME= parameter.

F03

instructs IMDPRDMP to print areas of main storage that were associated with a task terminated by the damage assessment routine (DAR).

```
TSO ┌SYSTEM=(YES )┐┌,USER=(PRINT  )┐
    │       {USER}││       {STORAGE}│
    │       (NO  )││       {FORMAT )│
    └             ┘└       (NO     )┘
```

instructs IMDPRDMP to process the TSO dump data set and the TSO swap data sets. IMDPRDMP will not format the swap data sets unless you have defined them in SYSWAPmn DD statements.

Two parameters allow you to limit the amount of formatting that IMDPRDMP will do. If you omit a parameter, IMDPRDMP will give you maximum formatting.

SYSTEM=

defines the extent of formatting for TSO system control blocks. The default value is SYSTEM=YES; it causes IMDPRDMP to format the following control blocks:

TCB family for TSC
TSCVT
RCBs for each TS region
Active TJBs
SWAP CBs for each swap device
Active TSBs
User Main Storage Map.

If you specify SYSTEM=USER, IMDPRDMP will format only active TJBs, active TSBs, and the User Main Storage Map. If you specify SYSTEM=NO, IMDPRDMP will not format any TSO system control blocks.

USER=

defines the extent of formatting for the TSO user region and the TSO user control blocks. The default is USER=PRINT, which causes IMDRPDMP to format both the region and the control blocks. USER=STORAGE requests only the region, USER=FORMAT requests only the control blocks. USER=NO requests no formatting of the user region or control blocks.

EDIT Control Statement

The EDIT control statement causes IMDPRDMP to obtain and process trace data created by the Generalized Trace Facility (GTF). Like other control statements, it may be specified either from the operator's console or through cards in the input stream.

Edit Keyword Parameters

The keywords associated with the EDIT control statement are shown in Figure PRDMP-3; they are described on the next page. All EDIT keyword parameters are optional.

```
EDIT      [EXIT=pgmname]

          ┌ ┌DDNAME┐         ┐
          [,{        }=ddname]
          └ └  DD  ┘         ┘

          [,START=(ddd,hh.mm.ss)]

          [,STOP=(ddd,hh.mm.ss)]

          [,{JOBNAME}=(jobname1[,jobname2]...[,jobname5])]
          [ {J      }                                     ]

          [,TCB=(address1[,address2]...[,address5])]

          [,SYS]

          ┌ ┌IO    ┐[=(cuu1[,cuu2]...[,cuu50])]┐
          │,│IO=SIO│                           │
          │ │SIO=IO│                           │
          └ └SIO   ┘                           ┘

          [,{SVC                                     }]
          [ {SVC=(svcnum1[,svcnum2]...[,svcnum256])}]

          [,{PI                                    }]
          [ {PI=(code1[,code2]...[,code15][,SSM])}]

          [,EXT]

          [,DSP]

          ┌,USR=┌  ALL                                                          ┐┐
          │     │( ┌symbol1 ┐ ┌,┌symbol2 ┐┐...┌,┌symbol20 ┐┐ )│
          │     │( │idvalue1│ │ │idvalue2│ │   │ │idvalue20│ │ )│
          └     └( └idrange1┘ └ └idrange2┘┘   └ └idrange20┘┘ )┘┘
```

Figure PRDMP-3. Format of the EDIT Control Statement, Showing All Valid Keywords

EXIT=pgmname

defines the program name of a user-written exit routine that will inspect all trace records when IMDPRDMP gives it control. If the routine does not exist or cannot be loaded successfully, EDIT execution will terminate and the next IMDPRDMP control statement will be read.

DDNAME=ddname

specifies the name of the DD statement that defines the input trace data set. If you omit this keyword, IMDPRDMP assumes that trace data exists in buffers in a dump of main storage, and therefore will not accept any other EDIT keywords except EXIT. You must include this parameter if you want to selectively edit data management trace records.

START=(ddd,hh.mm.ss)

STOP=(ddd,hh.mm.ss)

These optional keywords specify that IMDPRDMP is to edit all trace records produced during the time of day indicated. If no START= time is specified, EDIT processing will begin at the beginning of the trace data set. If no STOP= time is specified, EDIT processing will continue to the end of the data set. If the trace data was recorded on an MFT system with no timer option, IMDPRDMP will ignore these keywords.

JOBNAME=(jobname1],[,jobname2]...[,jobname5])

allows you to specify up to five 8-character jobnames for which EDIT will process trace data. If all the jobnames to be specified cannot fit on one line, close the first line with a right parenthesis followed by a comma; on the next line respecify the JOBNAME keyword with the additional jobnames.

This keyword is not valid if SYSM data is to be edited.

TCB=(address1[,address2]...[,address5])

allows you to specify addresses of up to five task control blocks for which EDIT should process trace data. The addresses must be specified as 1- to 6-digit hexadecimal addresses. If all addresses cannot fit on one line, close the first line with a right parenthesis followed by a comma; on the next line respecify the TCB keyword with the additional addresses.

This keyword is not valid if SYSM data is to be edited.

SYS

This optional keyword requests EDIT to process all system event trace records -- that is, SVC, SIO, IO, PI, EXT, and DSP. If no EDIT keyword except DDNAME, EXIT, START, STOP, JOBNAME, and/or TCB is specified, EDIT will assume SYS as the default.

```
{IO     }  [=(cuu1,[,cuu2]...[,cuu50])]
{SIO    }
{IO=SIO }
{SIO=IO }
```

defines up to fifty different devices for which IO trace records, SIO trace records, or both should be formatted. If no specific devices are requested, all IO and/or SIO trace records will be formatted. If any specific devices are specified, only trace records associated with those devices will be formatted and all others will be ignored.

Devices should be specified as 3-digit device addresses. If all devices to be specified cannot fit on one line, close the first line with a right parenthesis followed by a comma; on the next line respecify the keyword with the remaining addresses.

PRDMP

SVC

SVC=(svcnum1[,svcnum2]...,[,svcnum256]

defines up to 256 SVC trace records that EDIT is to format. svcnum is a 1- to 3-digit decimal SVC number.

If no svcnum parameters are specified or if both SVC and SVC= are specified, all SVC trace records will be formatted. If any SVC numbers are specified, only trace records associated with those SVC numbers will be formatted; all others will be ignored.

If all SVC numbers cannot fit on one line, close the first line with a right parenthesis followed by a comma; on the next line respecify the keyword with the remaining SVC numbers.

PI

PI=(code[,code2]...[,code15][,SSM]

requests EDIT to format trace records associated with up to fifteen specified program interrupt codes. If no program interrupt codes are specified or if both PI and PI= are specified, all program interrupt trace records will be formatted. If any program interrupt codes are specified, only those program interrupt trace records will be formatted; all others will be ignored. If SSM is specified, EDIT will format SSM interrupt trace records for data recorded on a Model 65 Multiprocessing System.

If all codes to be specified cannot fit on one line, close the first line with a right parenthesis followed by a comma; on the next line respecify the keyword with the remaining codes.

EXT

requests that EDIT format all external interrupt trace records.

DSP

requests that EDIT format all dispatcher task-switch trace records.

```
USR=⎧  ALL                                                   ⎫
    ⎪⎧symbol1 ⎫ ⎡⎧,symbol2 ⎫⎤ ⎡⎧...,symbol20 ⎫⎤⎪
    ⎨⎨idvalue1⎬ ⎢⎨,idvalue2⎬⎥ ⎢⎨...,idvalue20⎬⎥⎬
    ⎪⎩idrange1⎭ ⎣⎩,idrange2⎭⎦ ⎣⎩...,idrange20⎭⎦⎪
    ⎩                                                        ⎭
```

specifies which user/subsystem trace records should be formatted; (user or subsystem trace records are created by the GTF GTRACE macro.) You can specify up to 20 ID values, ranges or symbols representing single components or subsystems. Idvalue is a 3-digit hexadecimal ID specified in the GTRACE macro when the records to be formatted were created. Idrange is a pair of idvalues defining a range of records to be formatted, for example, USR=(010-040,BFD-BFF). If you want to edit data management trace records, specify USR=DMA1.

If ALL is specified alone or in combination with other parameters, all user or subsystem trace entries will be formatted. (See Figure PRDMP-4.)

If all parameters cannot fit on one line, close the first line with a right parenthesis followed by a comma, making sure that any idrange specified is complete; on the next line respecify the USR= keyword and continue with the remaining parameters.

EDIT Parameter Defaults and Priorities

All EDIT defaults depend on the presence or absence of the DDNAME= parameter.

- If it is present, the input is an external trace data set. All parameters are valid. If none except DDNAME= are specified, EDIT assumes a default of SYS.

- If it is absent, the input is a main storage dump containing trace buffers. No parameters except EXIT= are valid, since EDIT cannot select records from a dump. All records, both system and user, will be processed. If you attempt to select specific records, EDIT will prompt you to supply the missing DDNAME= parameter or terminate EDIT processing.

Figure PRDMP-4 summarizes the priority and effect of those EDIT parameters that select records by trace event type. Any keyword shown in the table can be considered to include as subsets all the parameters shown indented below it; for example, SVC=svcnum is a subset of SVC, and SVC is a subset of SYS. Any parameter can override another parameter in the same set that has a lower priority.

You should not combine any parameter with another parameter that can override it; for example, do not combine SIO with SIO=ddd. You can, however, combine parameters that are part of separate sets; for example, you can combine SIO=ddd with IO and SVC, or SYS with USR=ALL. You can also combine any parameters that have the same priority; for example, you can combine SIO=aaa with SIO=IO=bbb. In this case the effect will be IO=bbb and SIO=(aaa,bbb).

Note: START=, STOP=, JOBNAME=, and TCB= have no effect on trace event selection. They merely exercise further selectivity over records already chosen by default or by by parameters that select system trace events.

EDIT Parameter Priorities				Trace Events Selected
1	2	3	4	
SYS				All SIO, IO, SVC, PI, DSP, and EXT
	SIO=IO			All SIO and IO
		SIO		All SIO
			SIO=ddd	SIO for device(s) ddd
			SIO=IO=ddd	SIO and IO for device(s) ddd
		IO		All IO
			IO=ddd	IO for device(s) ddd
			IO=SIO=ddd	IO and SIO for device(s) ddd
	SVC			All SVCs
		SVC=num		Specified SVCs
	PI			All PIs
		PI=code		Specified PI code(s)
	DSP			All DSP
	EXT			All EXT
USR=ALL				All USR
	USR=notall			Specified USR

Figure PRDMP-4. Priorities and Effects of EDIT Parameters Used to Select Records by Trace Event Type

Combining Control Statements

The following control statements may be combined freely with each other on a single card or in a single reply to a prompting message. They may be specified in any order.

```
CVT=parm
NEWTAPE
QCBTRACE
LPAMAP
FORMAT
EDIT (coded with no parameters)
```

All other control statements are restricted; that is, no more than one may be specified on a single card or in a single reply to a prompting message. If a control statement from this group is combined with any of the control statements listed above, the restricted control statement must come last.

Here are some examples of control statements combined correctly:

```
LPAMAP,EDIT,P N

F,QCBTRACE,EDIT DDNAME=TRACE,SVC,SIO=IO=ALL,PI

F,P F

Q,L,F,E,TSO S=YES,U=NO
```

Allocating Space for the Output Data Set

IMDPRDMP output is usually directed to a SYSOUT device; therefore in most cases its output is stored temporarily on a direct access storage device from which it is later written to the printer. This temporary storage allows the user to specify space allocation and blocking factors that will enhance IMDPRDMP's performance.

(Note that if time is not critical and the output data set is very large, the output data set may be allocated directly to a printer. Do this by specifying the UNIT parameter in the PRINTER DD statement, for example UNIT=00E.)

Specifying the Maximum Output Block Size

Since IMDPRDMP uses QSAM as the access method for the SYSOUT data set, you can improve performance by specifying the largest possible block size for the data set. The maximum block size within the limits of the track capacity of the output device can be calculated by the following method: Divide the maximum track capacity in bytes by the output record length, 121 bytes, and ignore any remainder. The quotient is the number of records per block. Multiply this number by 121 to find the maximum block size.

To illustrate: A 2311 disk storage unit has a track capacity of 3625 bytes. The IMDPRDMP output record length is 121 bytes. Thus the number of records per block is 29. This value multiplied by the output record length (121) gives the maximum block size, 3509 bytes. Code this value in the DCB= parameter of the PRINTER DD statement as follows:

```
    DCB=(BLKSIZE=3509)
```

Increasing the Space Allocated to SYSOUT

Depending on the number of lines to be printed, the amount of space normally allocated to a SYSOUT data set may not be enough to contain the entire formatted dump or trace listing. To eliminate this potential problem, allocate extra direct access storage space for the SYSOUT data set via the SPACE= operand in the PRINTER DD statement that represents the data set. This extra space may be expressed in terms of bytes, tracks, or cylinders.

PRDMP

Use the table below to determine the approximate number of lines that will be printed in a dump listing. (The table does not include figures for the EDIT function of IMDPRDMP.)

STORAGE SIZE	PRINTED LINES
16K	500
32K	1000
64K	2000
128K	4000
256K	8000
512K	16000
1024K	32000

Calculating Space Requirements by Block Size

Each printed line is represented by a 121-byte record; the space requirement can therefore be expressed in bytes as the record length multipled by the number of records. As an example, the SPACE= operand for a 512K dump SYSOUT data set might be expressed as: SPACE=(121,(16000,100)).

If a blocking factor was specified for this SYSOUT data set (as discussed above), the space allocation can be expressed in terms of block size. For example, if the block size has been calculated as 3509 bytes (or a blocking factor of 29 records per block), the same 512K dump listing would require 552 blocks to contain all of the listing information. This block figure was calculated as follows:

    16000 Output records / 29 Records per block   = 552 Blocks

The PRINTER DD statement might then be expressed as:

```
//PRINTER DD SYSOUT=x,
//           SPACE=(3509,(552,10)),
//           UNIT=2311,DCB=(BLKSIZE=3509)
```

Calculating Space Requirements for EDIT Output

When GTF trace data is edited using the EDIT function of IMDPRDMP, the number of lines of output can be estimated provided the maximum GTF trace buffer size and the number of blocks to be edited are known. Figure PRDMP-5 shows the number of lines of EDIT output as a function of maximum buffer size (block size) and the type of trace.

Editing Internal Trace Data

To estimate the number of lines to be printed when GTF buffers are edited from a dump data set, use the following formula to determine the number of buffers:

    (GTF Region Size-11K) / Buffer Size = Number of Buffers

Then multiply the number of buffers by the number of lines per buffer as shown in Figure PRDMP-5. (Note that the size of the region in which GTF was running must be known.)

Maximum Trace Buffer Size	SYSM Trace	SYSM With User Time Stamp	Comprehensive Trace	Comprehensive Trace With User Time Stamp
1024	25	50	30	60
2048	50	100	60	120
3500	65	130	110	220
4096	100	200	120	240

Figure PRDMP-5. Number of Lines of EDIT Output per Buffer as a Function of Maximum Buffer Size and Trace Type

To illustrate: if a GTF internal (SYSM) trace is to be edited from a stand-alone dump taken by IMDSADMP, and GTF had been running in a 20K region, then the buffer size is 1024 bytes (implied by the specification MODE=INT); thus

Number of buffers = (20K-11K)/1K

Number of buffers = 9

Figure PRDMP-5 indicates that for a SYSM trace the number of lines per buffer is 25; thus 9 (25) or 225 is the expected number of printed lines. The PRINTER DD statement in this case might be expressed as

```
//PRINTER DD SYSOUT=A,SPACE=(121,(225,10))
```

Editing an External Trace Data SET

To estimate the number of lines to be printed when GTF data is edited from the trace data set on a direct access device, determine the number of blocks per track and multiply that value by the allocated number of tracks; the resulting value is the number of blocks per data set. Multiply that value by the number of lines per block as indicated in Figure PRDMP-5.

PRDMP

For example: A comprehensive trace with user time stamps is to be edited from a data set that occupies 50 tracks of a device whose track capacity is 7200 bytes. The maximum blocksize for the trace (established by the IEFRDER DD statement in the GTF start procedure) is 3500 bytes. Thus the number of blocks per track (in round figures) is 2, and the number of blocks in the data set is 2(50) or 100. Figure PRDMP-5 indicates that for a comprehensive trace with user time stamps the number of lines per block is 220; thus the expected number of printed lines is 100(220) or 22000.

In this case the PRINTER DD statement might be expressed as:

```
//PRINTER DD SYSOUT=A,SPACE=(121,(22000,100))
```

If the trace data set is on a tape volume, you can estimate the maximum number of lines to be printed by calculating the number of blocks per foot of tape and multiplying by the length of the tape.

Cataloged Procedure

Figure PRDMP-6 shows the cataloged procedure, PRDMP, that IBM supplies for executing IMDPRDMP.

```
//PRDMP        PROC
//DMP          EXEC PGM=IMDPRDMP
//SYSPRINT     DD   SYSOUT=A
//TAPE         DD   DSNAME=SYS1.DUMP,DISP=OLD
//PRINTER      DD   SYSOUT=A
//SYSUT1       DD   UNIT=SYSDA,SPACE=(2052,(257,64))
```

Figure PRDMP-6. The cataloged procedure PRDMP.

The statements are explained below.

EXEC Statement

calls for the execution of IMDPRDMP.

SYSPRINT DD Statement

defines the IMDPRDMP message data set.

TAPE DD Statement

defines the input data set. Unless overridden with other data set names, this statement defines SYS1.DUMP as the input data set.

PRINTER DD Statement

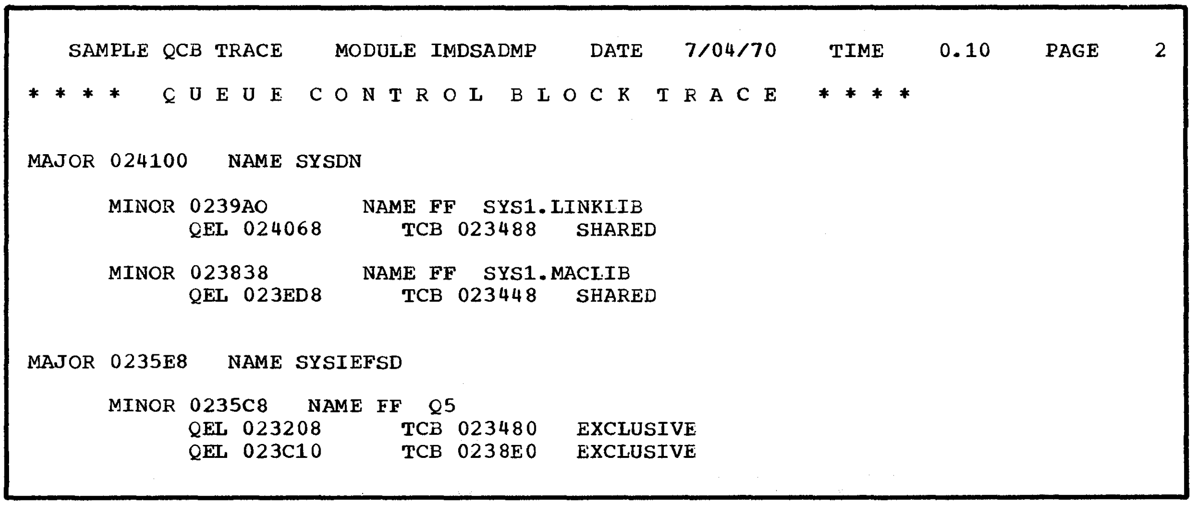
defines the output data set.

SYSUT1 DD Statement

defines the work data set.

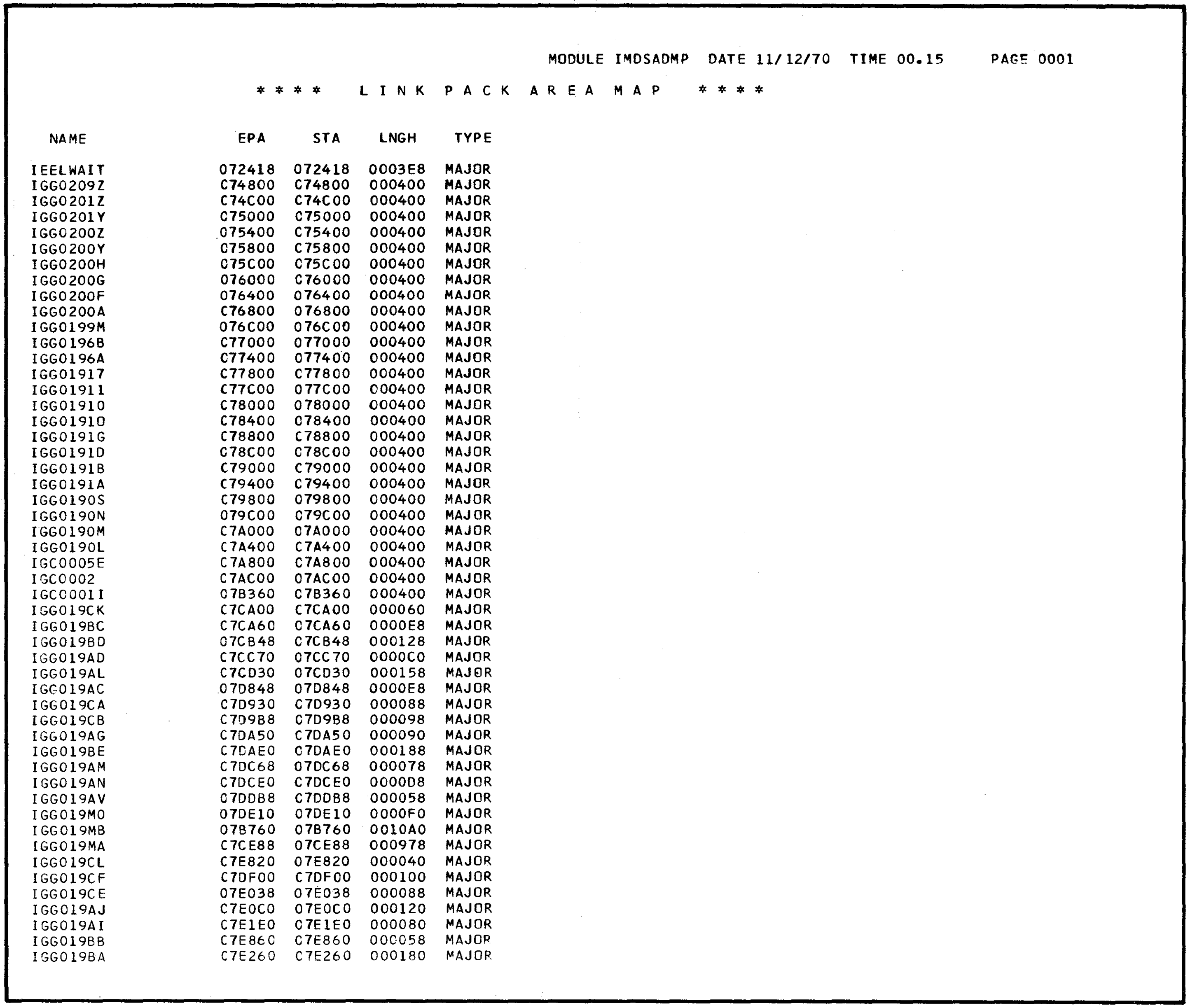
Note that the SYSIN DD statement has been omitted. Unless this statement is supplied, IMDPRDMP will prompt the operator to enter control statements through the console.

Figures PRDMP-7 through PRDMP-15 are samples of IMDPRDMP output. The formats are explained in detail in the Programmer's Guide to Debugging, GC28-6670.



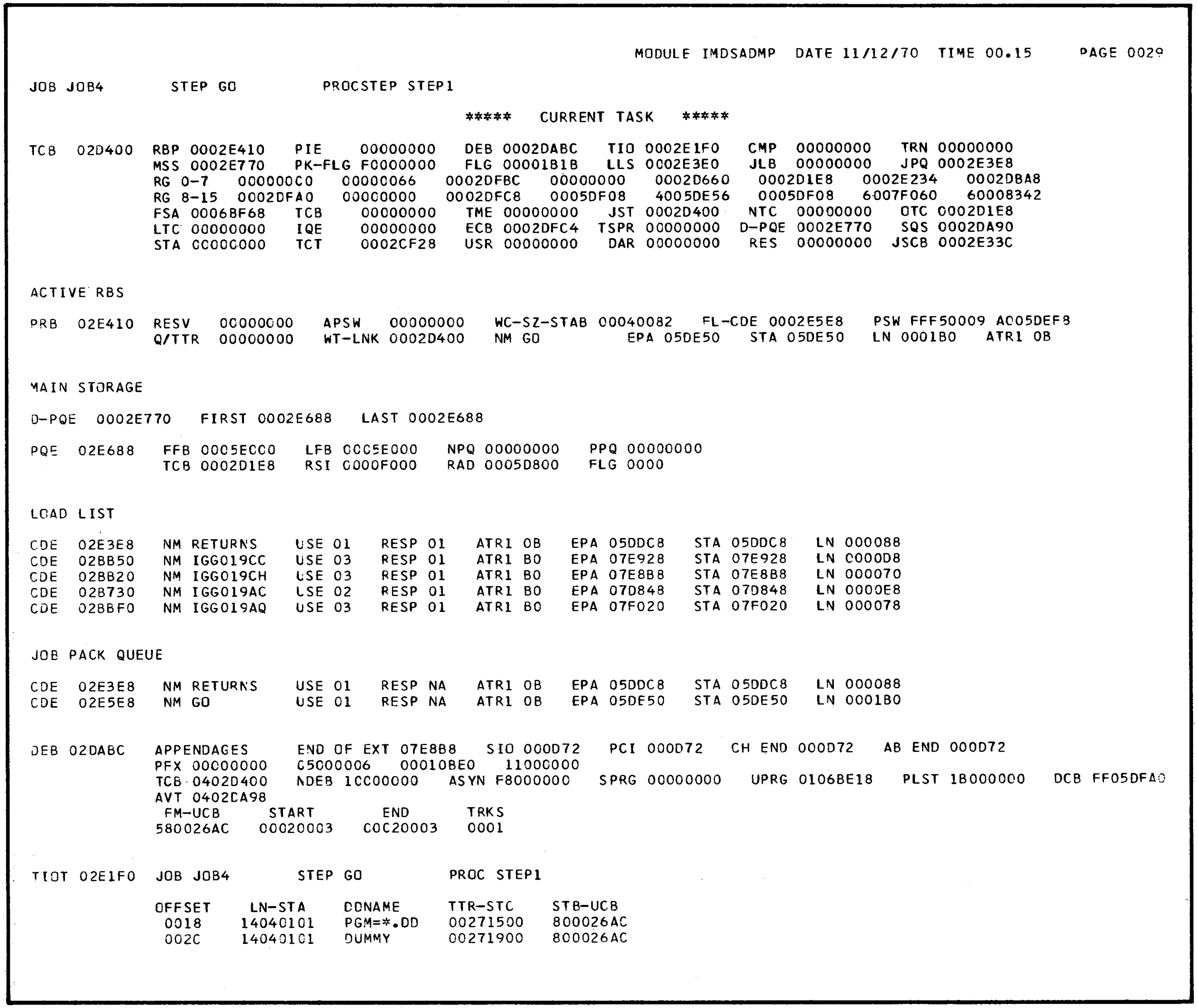
```
   SAMPLE QCB TRACE    MODULE IMDSADMP    DATE   7/04/70    TIME    0.10    PAGE    2

* * * *   Q U E U E   C O N T R O L   B L O C K   T R A C E   * * * *


MAJOR 024100    NAME SYSDN

      MINOR 0239A0       NAME FF  SYS1.LINKLIB
            QEL 024068      TCB 023488   SHARED

      MINOR 023838       NAME FF  SYS1.MACLIB
            QEL 023ED8      TCB 023448   SHARED


MAJOR 0235E8    NAME SYSIEFSD

      MINOR 0235C8   NAME FF  Q5
            QEL 023208      TCB 023480   EXCLUSIVE
            QEL 023C10      TCB 0238E0   EXCLUSIVE
```

Figure PRDMP-7. Queue Control Block Trace Sample



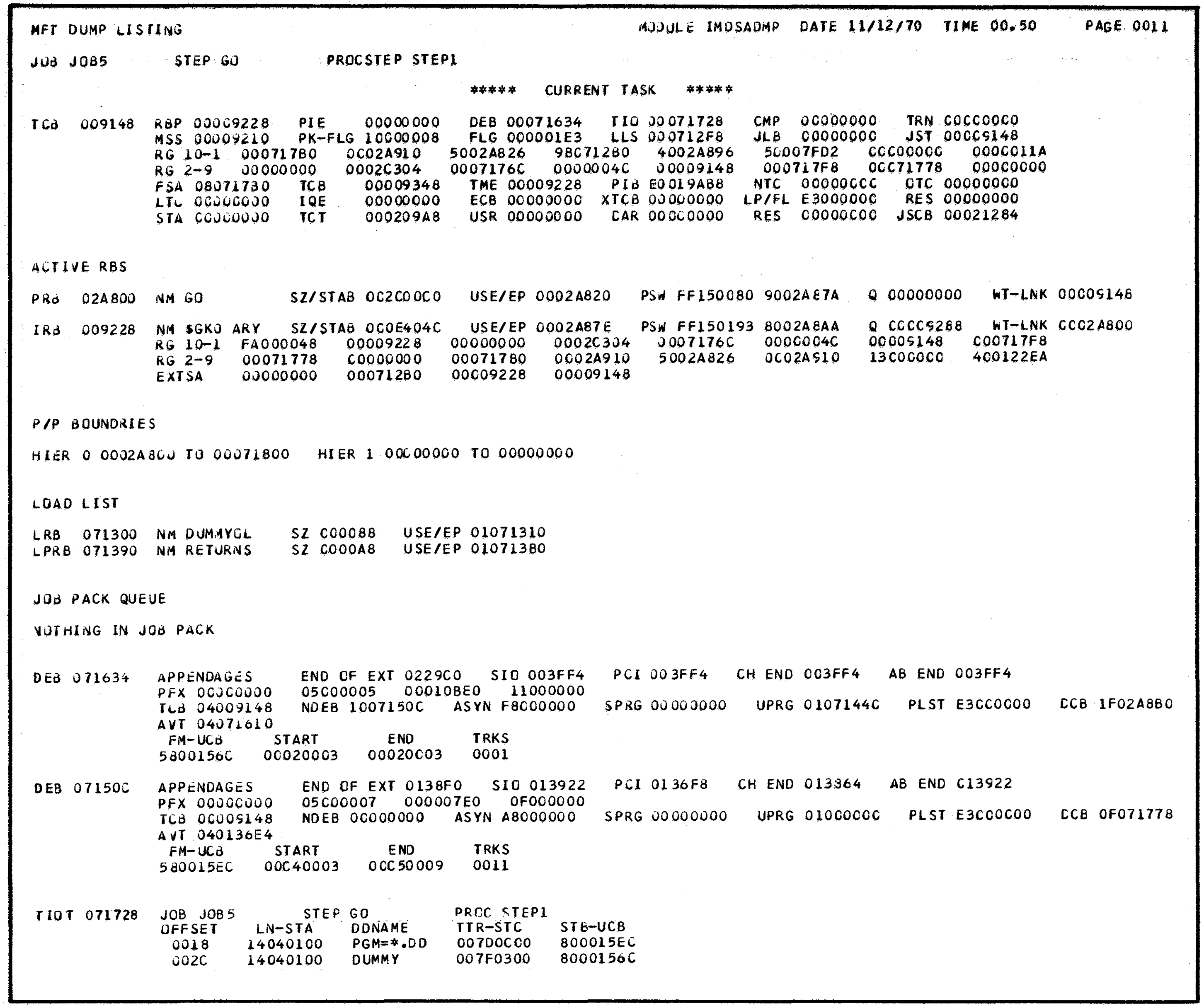
```
                                                          MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0001
                          * * * *   L I N K  P A C K  A R E A  M A P    * * * *

  NAME                EPA     STA     LNGH    TYPE

 IEELWAIT            072418  072418  0003E8  MAJOR
 IGG0209Z            C74800  C74800  000400  MAJOR
 IGG0201Z            C74C00  C74C00  000400  MAJOR
 IGG0201Y            C75000  C75000  000400  MAJOR
 IGG0200Z            075400  C75400  000400  MAJOR
 IGG0200Y            075800  C75800  000400  MAJOR
 IGG0200H            075C00  075C00  000400  MAJOR
 IGG0200G            076000  C76000  000400  MAJOR
 IGG0200F            076400  076400  000400  MAJOR
 IGG0200A            C76800  076800  000400  MAJOR
 IGG0199M            076C00  076C00  000400  MAJOR
 IGG0196B            C77000  077000  000400  MAJOR
 IGG0196A            C77400  077400  000400  MAJOR
 IGG01917            C77800  C77800  000400  MAJOR
 IGG01911            C77C00  077C00  000400  MAJOR
 IGG01910            C78000  078000  000400  MAJOR
 IGG0191O            C78400  078400  000400  MAJOR
 IGG0191G            C78800  C78800  000400  MAJOR
 IGG0191D            C78C00  078C00  000400  MAJOR
 IGG0191B            C79000  C79000  000400  MAJOR
 IGG0191A            C79400  C79400  000400  MAJOR
 IGG0190S            C79800  079800  000400  MAJOR
 IGG0190N            079C00  079C00  000400  MAJOR
 IGG0190M            C7A000  07A000  000400  MAJOR
 IGG0190L            C7A400  C7A400  000400  MAJOR
 IGC0005E            C7A800  C7A800  000400  MAJOR
 IGC0002             C7AC00  07AC00  000400  MAJOR
 IGC0001I            07B360  07B360  000400  MAJOR
 IGG019CK            C7CA00  C7CA00  000060  MAJOR
 IGG019BC            C7CA60  07CA60  0000E8  MAJOR
 IGG019BD            07CB48  C7CB48  000128  MAJOR
 IGG019AD            C7CC70  07CC70  0000C0  MAJOR
 IGG019AL            C7CD30  07CD30  000158  MAJOR
 IGG019AC            07D848  07D848  0000E8  MAJOR
 IGG019CA            C7D930  C7D930  000088  MAJOR
 IGG019CB            C7D9B8  C7D9B8  000098  MAJOR
 IGG019AG            C7DA50  C7DA50  000090  MAJOR
 IGG019BE            C7DAE0  07DAE0  000188  MAJOR
 IGG019AM            C7DC68  07DC68  000078  MAJOR
 IGG019AN            C7DCE0  C7DCE0  0000D8  MAJOR
 IGG019AV            07DDB8  C7DDB8  000058  MAJOR
 IGG019MO            07DE10  07DE10  0000F0  MAJOR
 IGG019MB            07B760  07B760  0010A0  MAJOR
 IGG019MA            C7CE88  07CE88  000978  MAJOR
 IGG019CL            C7E820  07E820  000040  MAJOR
 IGG019CF            C7DF00  C7DF00  000100  MAJOR
 IGG019CE            07E038  07E038  000088  MAJOR
 IGG019AJ            C7E0C0  07E0C0  000120  MAJOR
 IGG019AI            C7E1E0  07E1E0  000080  MAJOR
 IGG019BB            C7E860  C7E860  000058  MAJOR
 IGG019BA            C7E260  C7E260  000180  MAJOR
```

Figure PRDMP-8. Sample MVT Link Pack Area Map



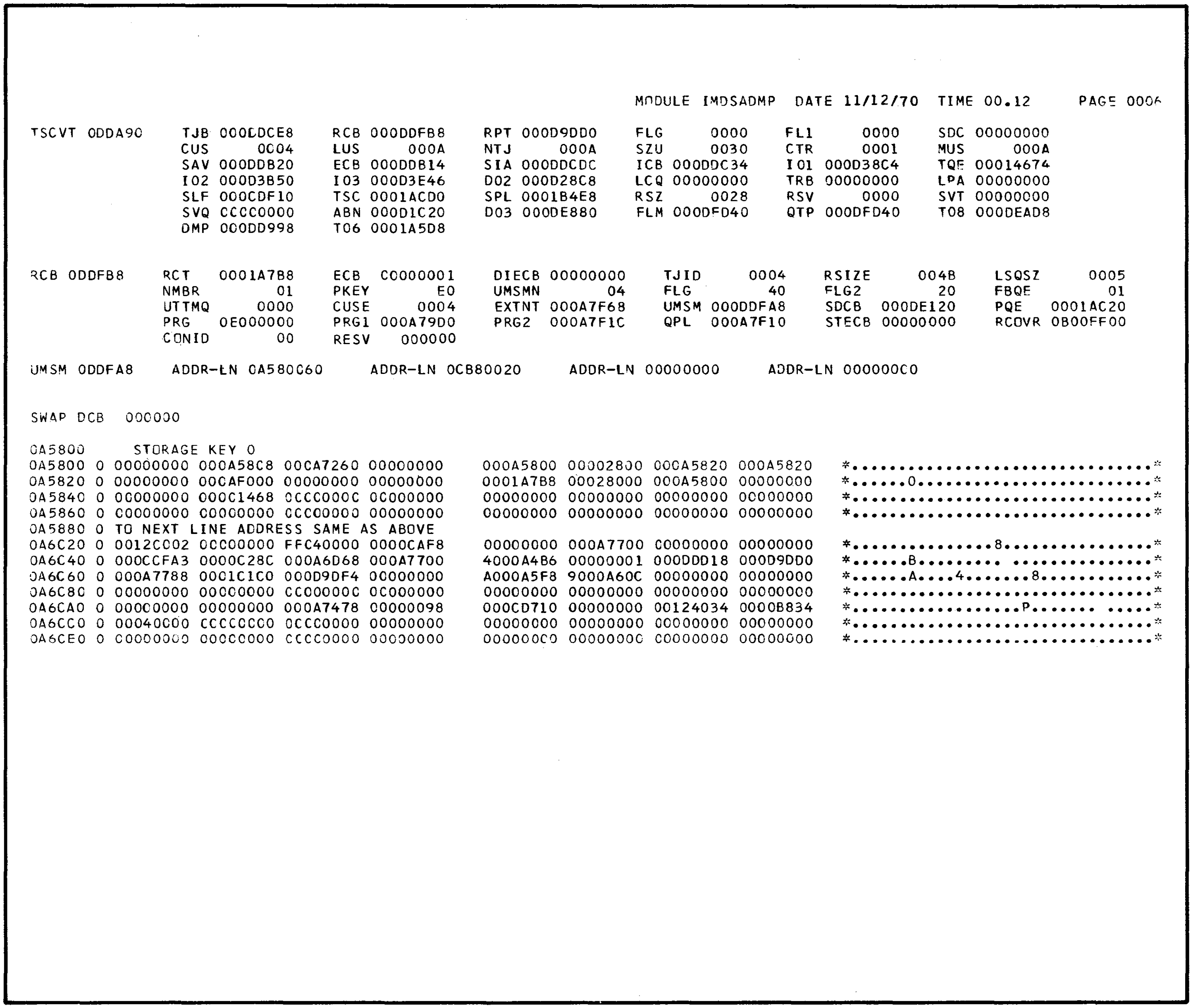
```
                                                            MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0029

JOB JOB4       STEP GO         PROCSTEP STEP1

                                                  *****   CURRENT TASK   *****

TCB  02D400  RBP 0002E410   PIE      00000000   DEB 0002DABC   TIO 0002E1F0   CMP  00000000   TRN 00000000
             MSS 0002E770   PK-FLG F0000000   FLG 00001B1B   LLS 0002E3E0   JLB  00000000   JPQ 0002E3E8
             RG 0-7   000000C0   00000066   0002DFBC   00000000   0002D660   0002D1E8   0002E234   0002DBA8
             RG 8-15  0002DFA0   00000000   0002DFC8   0005DF08   4005DE56   0005DF08   6007F060   60008342
             FSA 0006BF68   TCB      00000000   TME 00000000   JST 0002D400   NTC  00000000   OTC 0002D1E8
             LTC 00000000   IQE      00000000   ECB 0002DFC4  TSPR 00000000  D-PQE 0002E770   SQS 0002DA90
             STA CC00C000   TCT      0002CF28   USR 00000000   DAR 00000000   RES  00000000  JSCB 0002E33C


ACTIVE RBS

PRB  02E410  RESV    0C000000   APSW   00000000   WC-SZ-STAB 00040082   FL-CDE 0002E5E8   PSW FFF50009 A005DEF8
             Q/TTR   00000000   WT-LNK 0002D400   NM GO          EPA 05DE50   STA 05DE50   LN 0001B0   ATR1 0B


MAIN STORAGE

D-PQE  0002E770   FIRST 0002E688   LAST 0002E688

PQE  02E688   FFB 0005EC00   LFB 0C05E000   NPQ 00000000   PPQ 00000000
              TCB 0002D1E8   RSI C000F000   RAD 0005D800   FLG 0000


LOAD LIST

CDE  02E3E8   NM RETURNS    USE 01   RESP 01   ATR1 0B   EPA 05DDC8   STA 05DDC8   LN 000088
CDE  02BB50   NM IGG019CC   USE 03   RESP 01   ATR1 B0   EPA 07E928   STA 07E928   LN C000D8
CDE  02BB20   NM IGG019CH   USE 03   RESP 01   ATR1 B0   EPA 07E8B8   STA 07E8B8   LN 000070
CDE  02B730   NM IGG019AC   USE 02   RESP 01   ATR1 B0   EPA 07D848   STA 07D848   LN 0000E8
CDE  02BBF0   NM IGG019AQ   USE 03   RESP 01   ATR1 B0   EPA 07F020   STA 07F020   LN 000078


JOB PACK QUEUE

CDE  02E3E8   NM RETURNS    USE 01   RESP NA   ATR1 0B   EPA 05DDC8   STA 05DDC8   LN 000088
CDE  02E5E8   NM GO         USE 01   RESP NA   ATR1 0B   EPA 05DE50   STA 05DE50   LN 0001B0


DEB 02DABC    APPENDAGES      END OF EXT 07E8B8   SIO 000D72   PCI 000D72   CH END 000D72   AB END 000D72
              PFX 00000000   C5000006   00010BE0   11000000
              TCB 0402D400   NDEB 1CC00000   ASYN F8000000   SPRG 00000000   UPRG 0106BE18   PLST 1B000000   DCB FF05DFA0
              AVT 0402CA98
               FM-UCB     START       END      TRKS
              580026AC   00020003   C0C20003   0001


TIOT 02E1F0   JOB JOB4        STEP GO         PROC STEP1

              OFFSET    LN-STA     DDNAME      TTR-STC    STB-UCB
               0018    14040101   PGM=*.DD    00271500   800026AC
               002C    14040101   DUMMY       00271900   800026AC
```

Figure PRDMP-9. Sample MVT Major Control Block Format



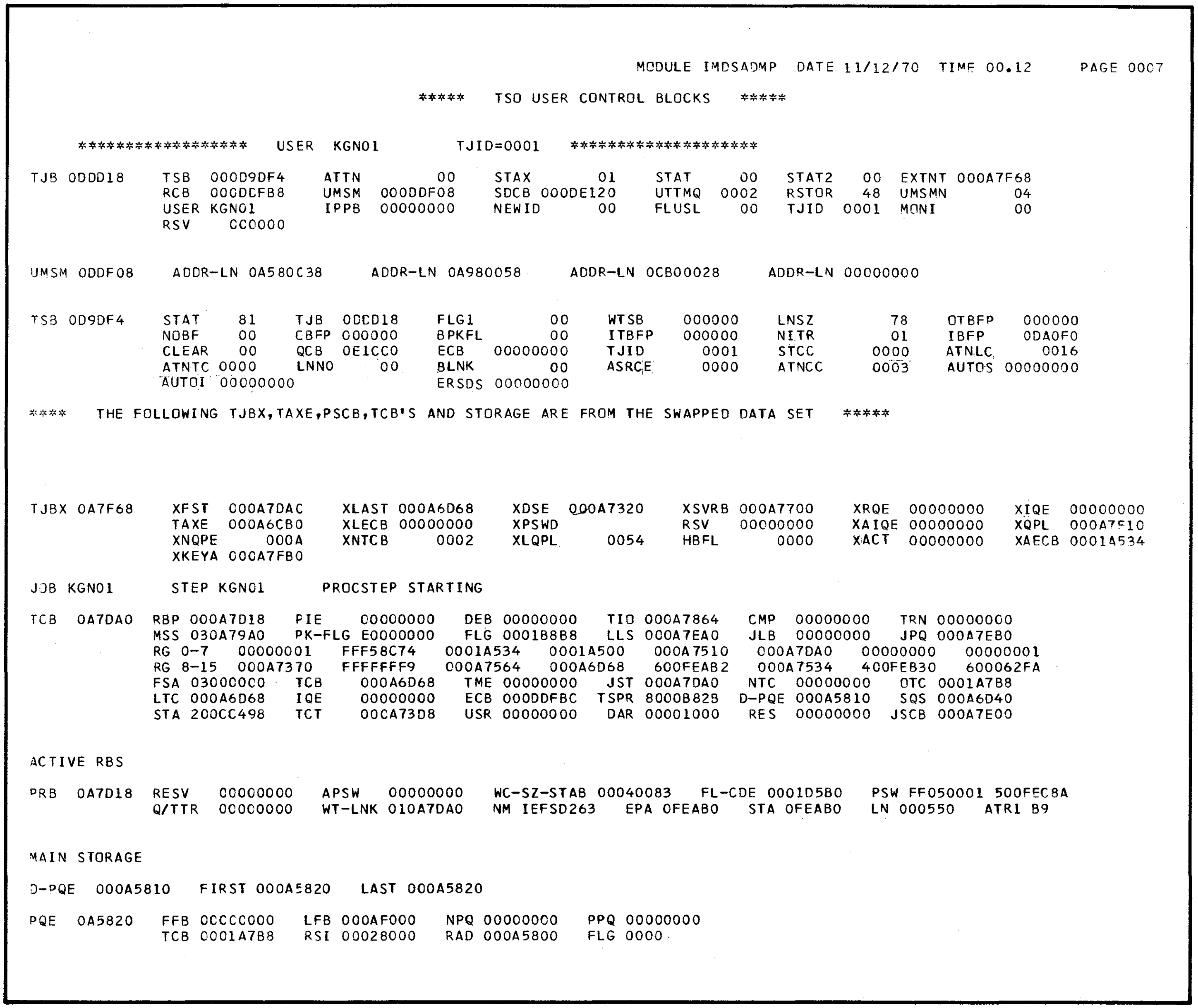
```
MFT DUMP LISTING                                                 MODULE IMDSADMP  DATE 11/12/70  TIME 00.50     PAGE 0011

JOB JOB5        STEP GO          PROCSTEP STEP1

                                                      *****   CURRENT TASK   *****

TCB   009148   RBP 00009228   PIE      00000000   DEB 00071634   TIO 00071728   CMP  00000000   TRN 00000000
               MSS 00009210   PK-FLG 10000008   FLG 000001E3   LLS 000712F8   JLB  0000000C   JST 00009148
               RG 10-1  000717B0   0C02A910   5002A826   9BC712B0   4002A896   50007FD2   CCC0000C   0000011A
               RG 2-9   00000000   0002C304   0007176C   0000004C   00009148   000717F8   00071778   00000000
               FSA 08071730   TCB      00009348   TME 00009228   PIB E0019A88   NTC  00000CCC   OTC 00000000
               LTC 00000000   IQE      00000000   ECB 00000000  XTCB 00000000  LP/FL E300000C   RES 00000000
               STA 00000000   TCT      000209A8   USR 00000000   CAR 00000000   RES  00000C00  JSCB 00021284


ACTIVE RBS

PRB   02A800   NM GO          SZ/STAB 0C2C00C0   USE/EP 0002A820   PSW FF150080 9002A87A   Q 00000000   WT-LNK 00009148

IRB   009228   NM $GKO ARY    SZ/STAB 0C0E404C   USE/EP 0002A87E   PSW FF150193 8002A8AA   Q 00009288   WT-LNK 0002A800
               RG 10-1  FA000048   00009228   00000000   0002C304   0007176C   0000004C   00009148   C00717F8
               RG 2-9   00071778   C0000000   000717B0   0002A910   5002A826   0C02A910   13C000C0   400122EA
               EXTSA    00000000   000712B0   00009228   00009148


P/P BOUNDRIES

HIER 0 0002A800 TO 00071800   HIER 1 00000000 TO 00000000


LOAD LIST

LRB  071300   NM DUMMYGL     SZ C00088   USE/EP 01071310
LPRB 071390   NM RETURNS     SZ C000A8   USE/EP 010713B0


JOB PACK QUEUE

NOTHING IN JOB PACK


DEB 071634    APPENDAGES      END OF EXT 0229C0   SIO 003FF4   PCI 003FF4   CH END 003FF4   AB END 003FF4
              PFX 00000000    05C00005    00010BE0    11000000
              TCB 04009148    NDEB 1007150C   ASYN F8C00000   SPRG 00000000   UPRG 0107144C   PLST E3CC0000   DCB 1F02A8B0
              AVT 04071610
               FM-UCB      START       END       TRKS
              5800156C    00020003    00020C03   0001

DEB 07150C    APPENDAGES      END OF EXT 0138F0   SIO 013922   PCI 0136F8   CH END 013364   AB END C13922
              PFX 00000000    05C00007    000007E0    0F000000
              TCB 00009148    NDEB 00000000   ASYN A8000000   SPRG 00000000   UPRG 0100000C   PLST E3C00C00   DCB 0F071778
              AVT 040136E4
               FM-UCB      START       END       TRKS
              580015EC    00C40003    00C50009   0011


TIOT 071728   JOB  JOB5          STEP GO          PROC STEP1
              OFFSET    LN-STA      DDNAME       TTR-STC    STB-UCB
               0018     14040100    PGM=*.DD     007D0C00   800015EC
               002C     14040100    DUMMY        007F0300   8000156C
```

Figure PRDMP-10. Sample MFT Major Control Block Format



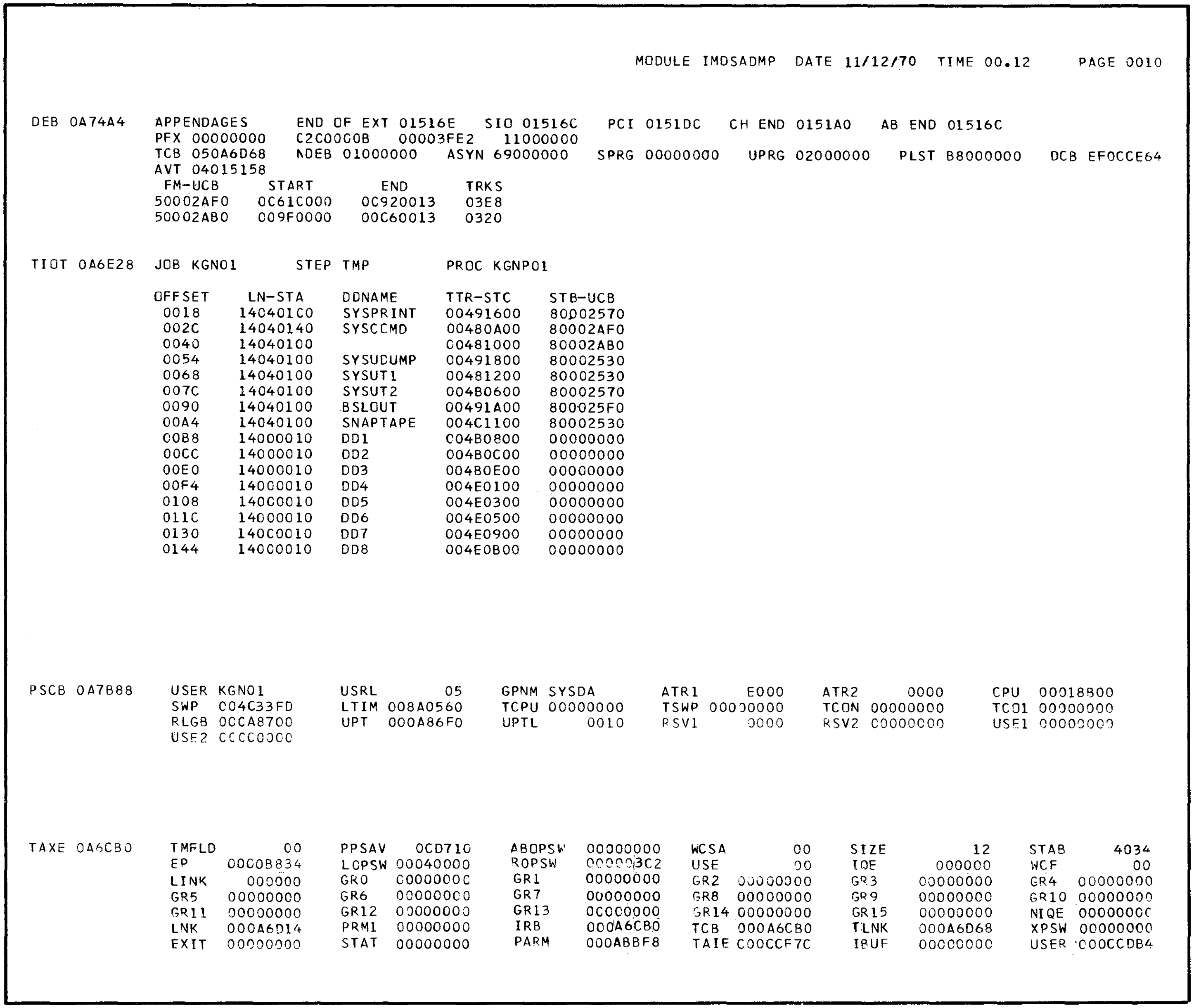
```
                                                                                    MODULE IMDSADMP  DATE 11/12/70  TIME 00.12      PAGE 0006

TSCVT 0DDA90      TJB 000DDCE8     RCB 000DDFB8     RPT 000D9DD0     FLG      0000     FL1      0000     SDC 00000000
                  CUS     0004     LUS     000A     NTJ     000A     SZU      0030     CTR      0001     MUS     000A
                  SAV 000DDB20     ECB 000DDB14     SIA 000DDCDC     ICB 000DDC34     I01 000D38C4     TQE 00014674
                  I02 000D3B50     I03 000D3E46     D02 000D28C8     LCQ 00000000     TRB 00000000     LPA 00000000
                  SLF 000CDF10     TSC 0001ACD0     SPL 0001B4E8     RSZ      0028     RSV      0000     SVT 00000000
                  SVQ CCCC0000     ABN 000D1C20     D03 000DE880     FLM 000DFD40     QTP 000DFD40     T08 000DEAD8
                  DMP 000DD998     T06 0001A5D8


RCB 0DDFB8      RCT     0001A7B8   ECB   C0000001    DIECB 00000000    TJID       0004    RSIZE      004B    LSQSZ       0005
                NMBR          01   PKEY        E0    UMSMN       04    FLG          40    FLG2         20    FBQE          01
                UTTMQ       0000   CUSE      0004    EXTNT 000A7F68    UMSM  000DDFA8    SDCB   000DE120   PQE     0001AC20
                PRG     0E000000   PRG1  000A79D0    PRG2  000A7F1C    QPL   000A7F10    STECB  00000000   RCOVR   0B00FF00
                CONID         00   RESV    000000

UMSM 0DDFA8     ADDR-LN 0A580060     ADDR-LN 0CB80020     ADDR-LN 00000000     ADDR-LN 000000C0


SWAP DCB  000000

0A5800     STORAGE KEY 0
0A5800 0 00000000 000A58C8 00CA7260 00000000    000A5800 00002800 000A5820 000A5820   *................................*
0A5820 0 00000000 000AF000 00000000 00000000    0001A7B8 00028000 000A5800 00000000   *......0.........................*
0A5840 0 00000000 000C1468 CCCC000C 0C000000    00000000 00000000 00000000 00000000   *................................*
0A5860 0 C0000000 C0000000 CCC00000 00000000    00000000 00000000 00000000 00000000   *................................*
0A5880 0 TO NEXT LINE ADDRESS SAME AS ABOVE
0A6C20 0 0012C002 0CC00000 FFC40000 0000CAF8    00000000 000A7700 C0000000 00000000   *...............8................*
0A6C40 0 000CCFA3 0000C28C 000A6D68 000A7700    4000A4B6 00000001 000DDD18 000D9DD0   *......B........ ................*
0A6C60 0 000A7788 0001C1C0 000D9DF4 00000000    A000A5F8 9000A60C 00000000 00000000   *......A....4......8.............*
0A6C80 0 00000000 00000000 CC00000C 0C000000    00000000 00000000 00000000 00000000   *................................*
0A6CA0 0 00000000 00000000 000A7478 00000098    000CD710 00000000 00124034 0000B834   *..................P...... ......*
0A6CC0 0 00040000 CCCCCCC0 0CCC0000 00000000    00000000 00000000 00000000 00000000   *................................*
0A6CE0 0 C0000000 00000000 CCCC0000 00000000    000000C0 0000000C C0000000 00000000   *................................*
```

Figure PRDMP-11. Sample TSO Control Block Format (Part 1 of 3)



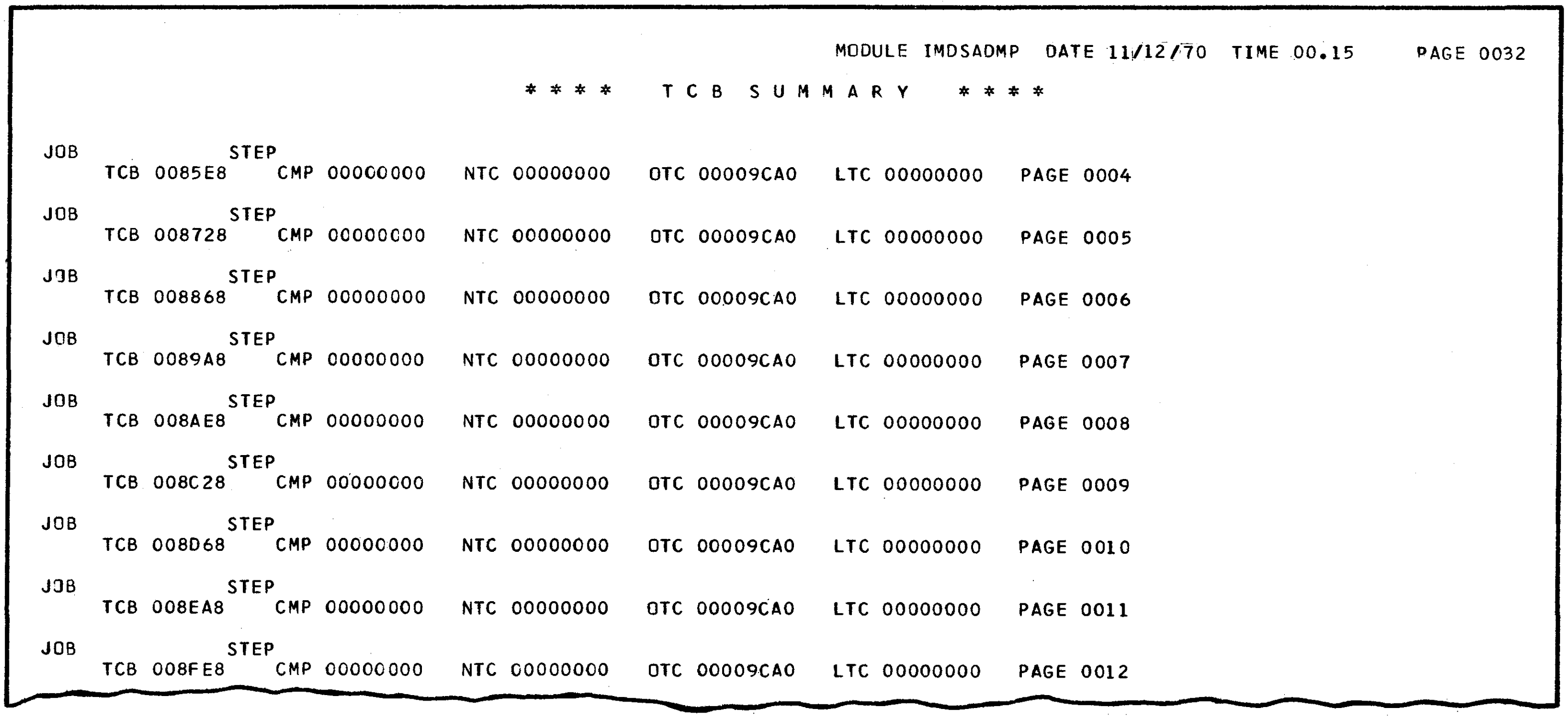
```
                                                                       MODULE IMDSADMP  DATE 11/12/70  TIME 00.12     PAGE 0007
                                                    *****   TSO USER CONTROL BLOCKS   *****

    *****************   USER  KGN01       TJID=0001   *******************

TJB 0DDD18     TSB  000D9DF4    ATTN          00    STAX        01    STAT      00    STAT2    00  EXTNT 000A7F68
               RCB  000DCFB8    UMSM  000DDF08    SDCB 000DE120    UTTMQ  0002    RSTOR    48  UMSMN          04
               USER KGN01       IPPB  00000000    NEWID       00    FLUSL     00    TJID  0001  MONI           00
               RSV    CC0000


UMSM 0DDF08     ADDR-LN 0A580C38      ADDR-LN 0A980058      ADDR-LN 0CB00028      ADDR-LN 00000000


TSB 0D9DF4     STAT     81     TJB   0DCD18     FLG1          00     WTSB      000000     LNSZ          78     OTBFP    000000
               NOBF     00     CBFP 000000      BPKFL         00     ITBFP     000000     NITR          01     IBFP     0DA0F0
               CLEAR    00     QCB   0E1CC0     ECB     00000000     TJID        0001     STCC        0000     ATNLC      0016
               ATNTC 0000      LNNO      00     BLNK          00     ASRCE       0000     ATNCC       0003     AUTOS 00000000
               AUTOI  00000000                  ERSDS 00000000

****   THE FOLLOWING TJBX,TAXE,PSCB,TCB'S AND STORAGE ARE FROM THE SWAPPED DATA SET   *****



TJBX 0A7F68     XFST   C00A7DAC     XLAST 000A6D68     XDSE   000A7320     XSVRB 000A7700     XRQE   00000000     XIQE   00000000
                TAXE   000A6CB0     XLECB 00000000     XPSWD                RSV   00000000     XAIQE 00000000      XQPL   000A7F10
                XNQPE      000A     XNTCB     0002     XLQPL      0054      HBFL      0000     XACT   00000000     XAECB 0001A534
                XKEYA C00A7FB0

JOB KGN01       STEP KGN01      PROCSTEP STARTING

TCB   0A7DA0  RBP 000A7D18    PIE      00000000    DEB 00000000    TIO 000A7864    CMP   00000000    TRN 00000000
              MSS 030A79A0    PK-FLG E0000000    FLG 000188B8    LLS 000A7EA0    JLB   00000000    JPQ 000A7EB0
              RG 0-7    00000001    FFF58C74    0001A534    0001A500    000A7510    000A7DA0    00000000    00000001
              RG 8-15   000A7370    FFFFFFF9    000A7564    000A6D68    600FEAB2    000A7534    400FEB30    600062FA
              FSA 030000C0    TCB      000A6D68    TME 00000000    JST 000A7DA0    NTC   00000000    OTC 0001A7B8
              LTC 000A6D68    IQE      00000000    ECB 000DDFBC   TSPR 8000B82B  D-PQE 000A5810    SQS 000A6D40
              STA 200CC498    TCT      00CA73D8    USR 00000000    DAR 00001000    RES   00000000   JSCB 000A7E00


ACTIVE RBS

PRB   0A7D18  RESV     00000000    APSW     00000000    WC-SZ-STAB 00040083    FL-CDE 0001D5B0    PSW FF050001 500FEC8A
              Q/TTR    00000000    WT-LNK 010A7DA0    NM IEFSD263    EPA 0FEAB0    STA 0FEAB0    LN 000550    ATR1 B9


MAIN STORAGE

D-PQE   000A5810    FIRST 000A5820    LAST 000A5820

PQE   0A5820    FFB 0CCCC000    LFB 000AF000    NPQ 00000000    PPQ 00000000
                TCB C001A7B8    RSI 00028000    RAD 000A5800    FLG 0000
```

Figure PRDMP-11. Sample TSO Control Block Format (Part 2 of 3)

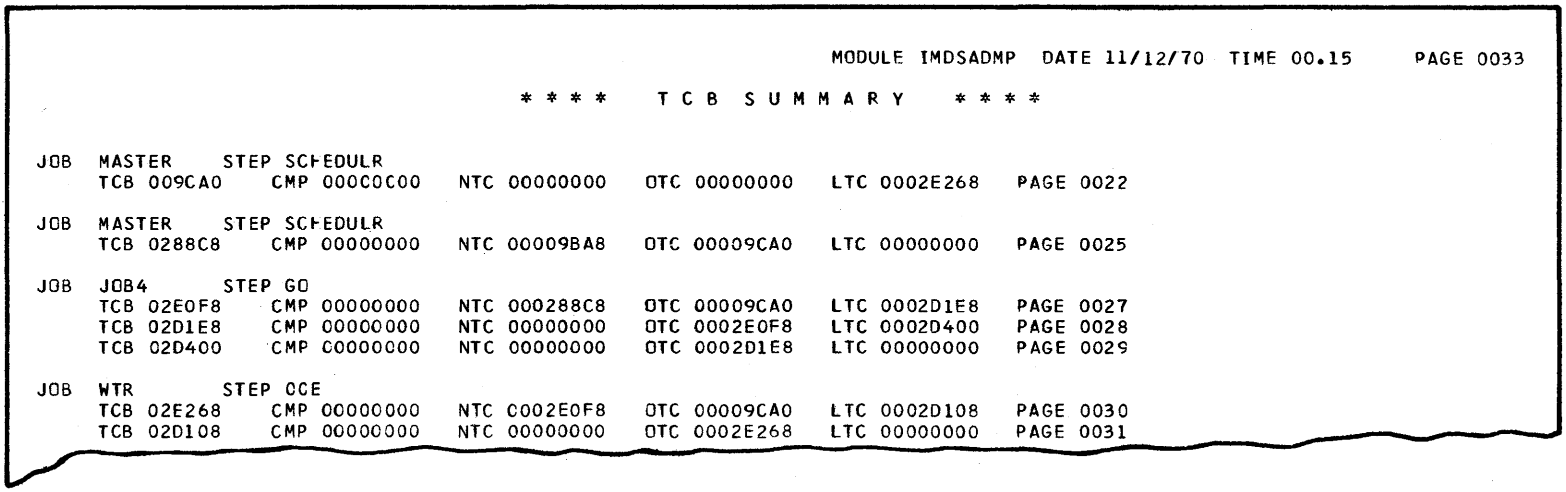


```
                                                  MODULE IMDSADMP  DATE 11/12/70  TIME 00.12     PAGE 0010


DEB 0A74A4    APPENDAGES      END OF EXT 01516E   SIO 01516C    PCI 0151DC    CH END 0151A0    AB END 01516C
              PFX 00000000    C2C000OB    00003FE2    11000000
              TCB 050A6D68    NDEB 01000000    ASYN 69000000    SPRG 00000000    UPRG 02000000    PLST B8000000    DCB EF0CCE64
              AVT 04015158
               FM-UCB       START        END        TRKS
              50002AF0     0C61C000     0C920013     03E8
              50002AB0     009F0000     00C60013     0320


TIOT 0A6E28   JOB KGN01        STEP TMP          PROC KGNP01

              OFFSET     LN-STA      DDNAME      TTR-STC     STB-UCB
               0018     140401C0    SYSPRINT    00491600    80002570
               002C     14040140    SYSCCMD     00480A00    80002AF0
               0040     14040100                C0481000    80002AB0
               0054     14040100    SYSUDUMP    00491800    80002530
               0068     14040100    SYSUT1      00481200    80002530
               007C     14040100    SYSUT2      004B0600    80002570
               0090     14040100    BSLOUT      00491A00    800025F0
               00A4     14040100    SNAPTAPE    004C1100    80002530
               00B8     14000010    DD1         C04B0800    00000000
               00CC     14000010    DD2         004B0C00    00000000
               00E0     14000010    DD3         004B0E00    00000000
               00F4     14000010    DD4         004E0100    00000000
               0108     14000010    DD5         004E0300    00000000
               011C     14000010    DD6         004E0500    00000000
               0130     140C0010    DD7         004E0900    00000000
               0144     14000010    DD8         004E0B00    00000000




PSCB 0A7B88     USER KGN01          USRL         05    GPNM SYSDA        ATR1      E000    ATR2       0000      CPU  00018B00
                SWP  C04C33FD       LTIM 008A0560      TCPU 00000000     TSWP 00000000     TCON 00000000     TC01 00000000
                RLGB 0CCA8700       UPT  000A86F0      UPTL      0010    RSV1      0000    RSV2 C0000000     USE1 00000000
                USE2 CCCC0000



TAXE 0A6CB0     TMFLD        00    PPSAV    0CD710    ABOPSW   00000000    WCSA        00    SIZE          12    STAB       4034
                EP     0000B834    LOPSW 00040000     ROPSW    000003C2    USE         00    IQE       000000    WCF          00
                LINK     000000    GR0   C000000C     GR1      00000000    GR2   00000000    GR3     00000000    GR4    00000000
                GR5    00000000    GR6   00000000     GR7      00000000    GR8   00000000    GR9     00000000    GR10   00000000
                GR11   00000000    GR12  00000000     GR13     0C000000    GR14  00000000    GR15    00000000    NIQE   0000000C
                LNK    000A6D14    PRM1  00000000     IRB      000A6CB0    TCB   000A6CB0    TLNK    000A6D68    XPSW   00000000
                EXIT   00000000    STAT  00000000     PARM     000ABBF8    TAIE  C00CCF7C    IBUF    00000000    USER C00CCDB4
```

Figure PRDMP-11. Sample TSO Control Block Format (Part 3 of 3)

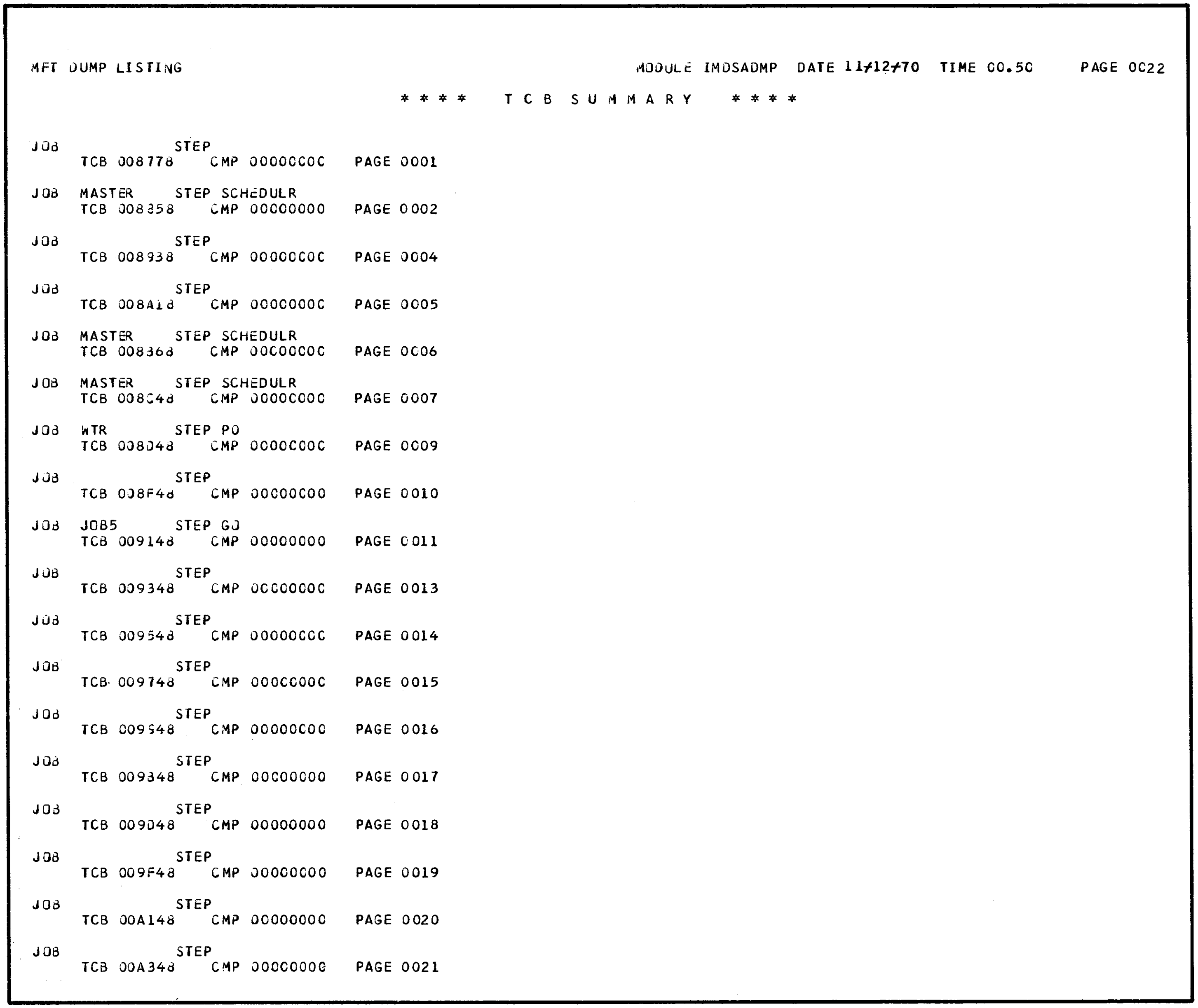


```
                                                        MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0032
                                       * * * *    T C B   S U M M A R Y    * * * *

JOB             STEP
     TCB 0085E8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0004

JOB             STEP
     TCB 008728     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0005

JOB             STEP
     TCB 008868     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0006

JOB             STEP
     TCB 0089A8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0007

JOB             STEP
     TCB 008AE8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0008

JOB             STEP
     TCB 008C28     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0009

JOB             STEP
     TCB 008D68     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0010

JOB             STEP
     TCB 008EA8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0011

JOB             STEP
     TCB 008FE8     CMP 00000000    NTC C0000000    OTC 00009CA0    LTC 00000000    PAGE 0012
```



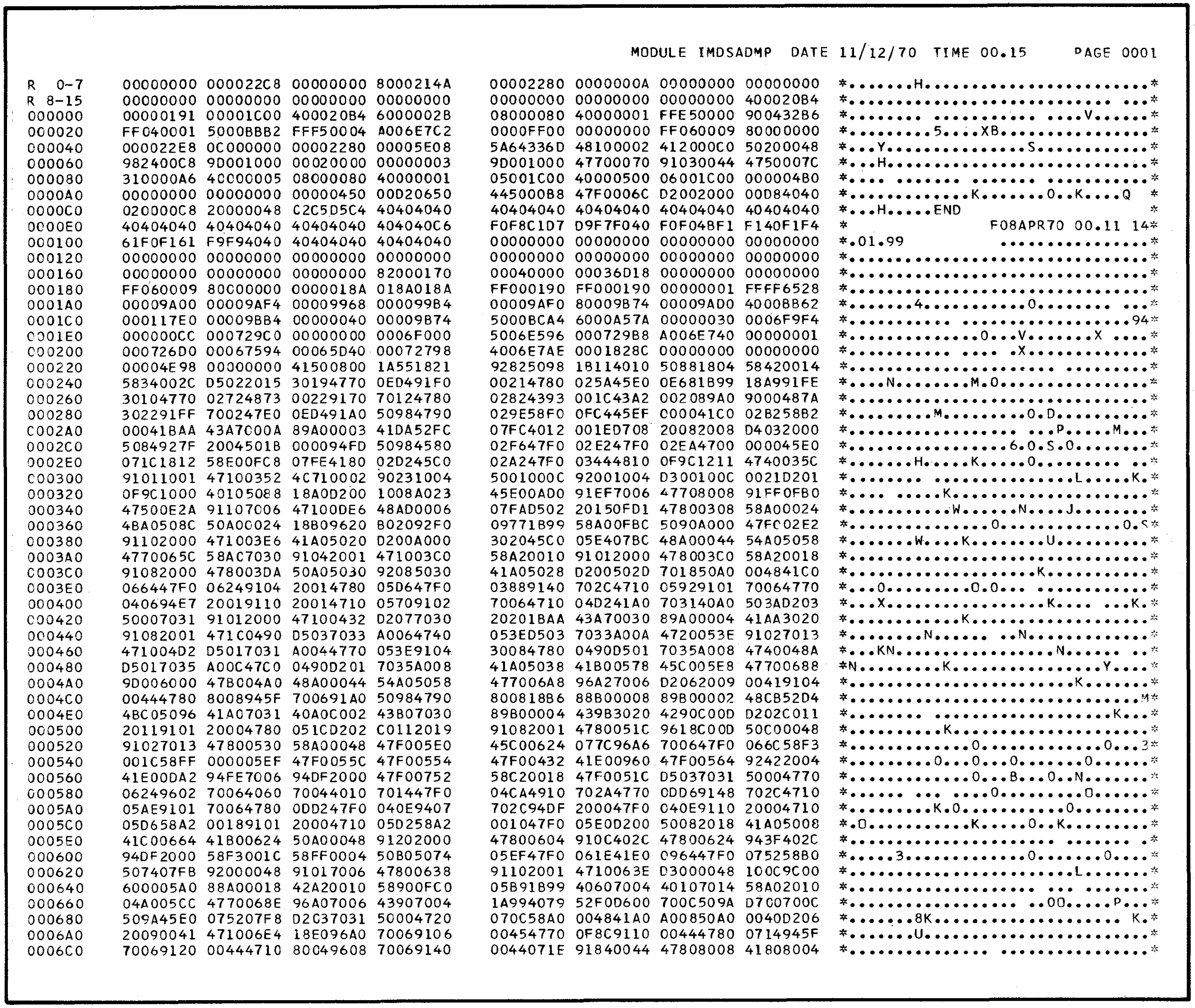
```
                                                        MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0033
                                       * * * *    T C B   S U M M A R Y    * * * *

JOB  MASTER     STEP SCHEDULR
     TCB 009CA0     CMP 000C0C00    NTC 00000000    OTC 00000000    LTC 0002E268    PAGE 0022

JOB  MASTER     STEP SCHEDULR
     TCB 0288C8     CMP 00000000    NTC 00009BA8    OTC 00009CA0    LTC 00000000    PAGE 0025

JOB  JOB4       STEP GO
     TCB 02E0F8     CMP 00000000    NTC 000288C8    OTC 00009CA0    LTC 0002D1E8    PAGE 0027
     TCB 02D1E8     CMP 00000000    NTC 00000000    OTC 0002E0F8    LTC 0002D400    PAGE 0028
     TCB 02D400     CMP C0000000    NTC 00000000    OTC 0002D1E8    LTC 00000000    PAGE 0029

JOB  WTR        STEP OCE
     TCB 02E268     CMP 00000000    NTC C002E0F8    OTC 00009CA0    LTC 0002D108    PAGE 0030
     TCB 02D108     CMP 00000000    NTC 00000000    OTC 0002E268    LTC 00000000    PAGE 0031
```

Firgure PRDMP-12. Sample TCB Summary for MVT or MFT With Subtasking



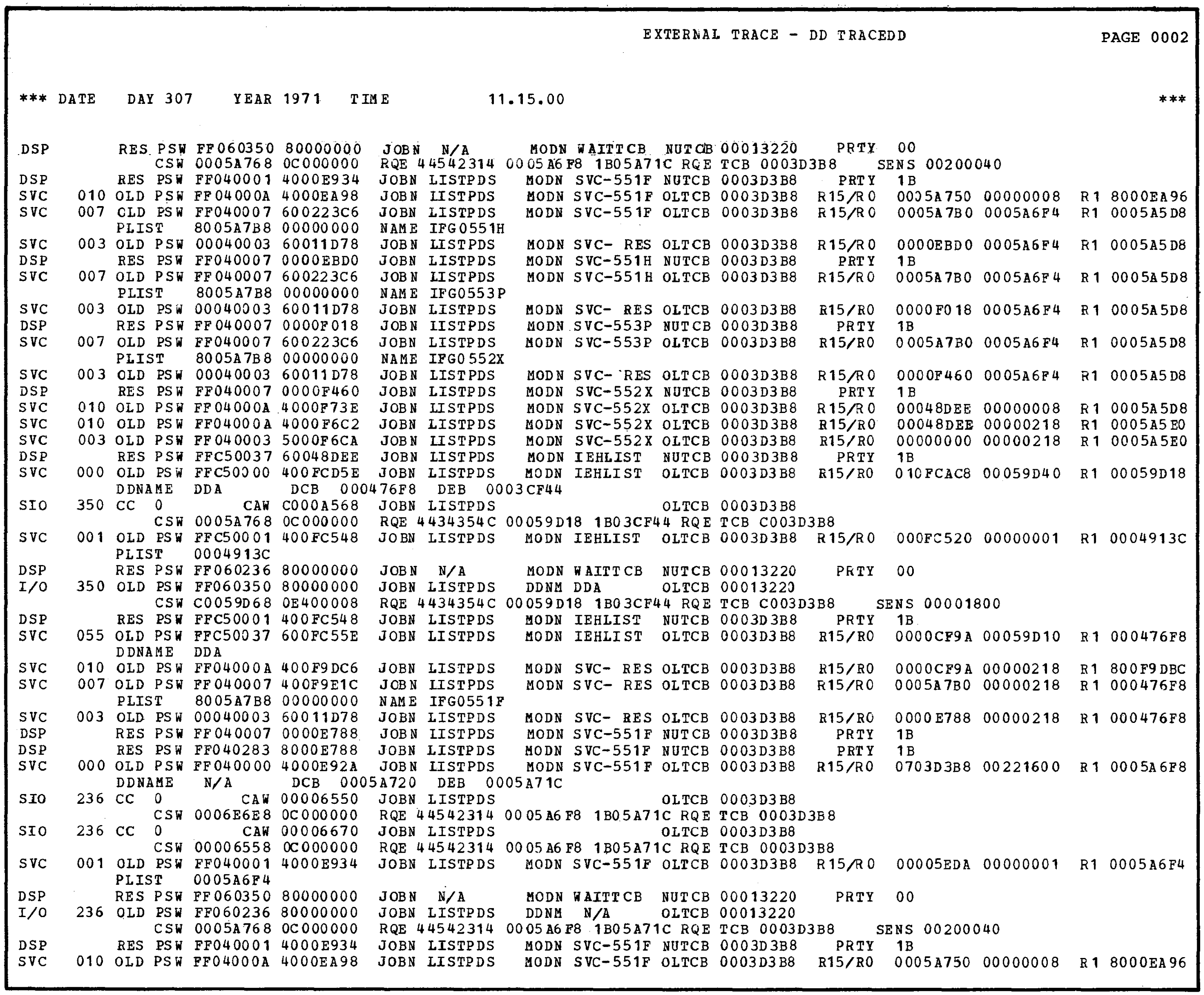
```
MFT DUMP LISTING                                                MODULE IMDSADMP  DATE 11/12/70  TIME 00.50    PAGE 0022
                                        * * * *   T C B  S U M M A R Y   * * * *

JOB            STEP
     TCB 008778    CMP 00000C0C   PAGE 0001

JOB  MASTER    STEP SCHEDULR
     TCB 008358    CMP 00C00000   PAGE 0002

JOB            STEP
     TCB 008938    CMP 00000C0C   PAGE 0004

JOB            STEP
     TCB 008A18    CMP 0000000C   PAGE 0005

JOB  MASTER    STEP SCHEDULR
     TCB 008368    CMP 00C0000C   PAGE 0C06

JOB  MASTER    STEP SCHEDULR
     TCB 008C48    CMP 0000C000   PAGE 0007

JOB  WTR       STEP PO
     TCB 008D48    CMP 0000C00C   PAGE 0009

JOB            STEP
     TCB 008F48    CMP 00000C00   PAGE 0010

JOB  JOB5      STEP GJ
     TCB 009148    CMP 00000000   PAGE C011

JOB            STEP
     TCB 009348    CMP 0C00000C   PAGE 0013

JOB            STEP
     TCB 009548    CMP 00000CCC   PAGE 0014

JOB            STEP
     TCB 009748    CMP 000CC00C   PAGE 0015

JOB            STEP
     TCB 009548    CMP 00000C00   PAGE 0016

JOB            STEP
     TCB 009348    CMP 00C00000   PAGE 0017

JOB            STEP
     TCB 009D48    CMP 00000000   PAGE 0018

JOB            STEP
     TCB 009F48    CMP 00000C00   PAGE 0019

JOB            STEP
     TCB 00A148    CMP 0000000C   PAGE 0020

JOB            STEP
     TCB 00A348    CMP 000C0000   PAGE 0021
```

Figure PRDMP-13. Sample TCB Summary for MFT Without Subtasking



```
                                                             MODULE IMDSADMP  DATE 11/12/70  TIME 00.15      PAGE 0001

R  0-7     00000000 000022C8 00000000 8000214A   00002280 0000000A 00000000 00000000   *.......H........................*
R  8-15    00000000 00000000 00000000 00000000   00000000 00000000 00000000 400020B4   *............................ ...*
000000     00000191 00001C00 400020B4 6000002B   08000080 40000001 FFE50000 900432B6   *........ ........... ....V......*
000020     FF040001 5000BBB2 FFF50004 A006E7C2   0000FF00 00000000 FF060009 80000000   *.........5....XB................*
000040     000022E8 0C000000 00002280 00005E08   5A64336D 48100002 412000C0 50200048   *...Y...............S............*
000060     982400C8 9D001000 00020000 00000003   9D001000 47700070 91030044 4750007C   *...H............................*
000080     310000A6 40000005 08000080 40000001   05001C00 40000500 06001C00 000004B0   *.... ....... ....... ...........*
0000A0     00000000 00000000 00000450 00D20650   445000B8 47F0006C D2002000 00D84040   *.............K.......0..K....Q  *
0000C0     020000C8 20000048 C2C5D5C4 40404040   40404040 40404040 40404040 40404040   *...H.....END                    *
0000E0     40404040 40404040 40404040 404040C6   F0F8C1D7 D9F7F040 F0F04BF1 F140F1F4   *               F08APR70 00.11 14*
000100     61F0F161 F9F94040 40404040 40404040   00000000 00000000 00000000 00000000   *.01.99          ................*
000120     00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   *................................*
000160     00000000 00000000 00000000 82000170   00040000 00036D18 00000000 00000000   *................................*
000180     FF060009 80000000 0000018A 018A018A   FF000190 FF000190 00000001 FFFF6528   *................................*
0001A0     00009A00 00009AF4 00009968 000099B4   00009AF0 80009B74 00009AD0 4000BB62   *.......4...........0........ ...*
0001C0     000117E0 00009BB4 00000040 00009B74   5000BCA4 6000A57A 00000030 0006F9F4   *........... ..................94*
C001E0     000000CC 000729C0 00000000 0006F000   5006E596 000729B8 A006E740 00000001   *..............0...V.......X ....*
C00200     000726D0 00067594 00065D40 00072798   4006E7AE 0001828C 00000000 00000000   *........... .... .X.............*
000220     00004E98 00000000 41500800 1A551821   92825098 1B114010 50881804 58420014   *...................... .........*
000240     5834002C D5022015 30194770 0ED491F0   00214780 025A45E0 0E681B99 18A991FE   *....N........M.0................*
000260     30104770 02724873 00229170 70124780   02824393 001C43A2 002089A0 9000487A   *................................*
000280     302291FF 700247E0 0ED491A0 50984790   029E58F0 0FC445EF C00041C0 02B258B2   *.........M.........0.D..........*
C002A0     00041BAA 43A7C00A 89A00003 41DA52FC   07FC4012 001ED708 20082008 D4032000   *................... ..P.....M...*
0002C0     5084927F 2004501B 000094FD 50984580   02F647F0 02E247F0 02EA4700 000045E0   *.................6.0.S.0........*
0002E0     071C1812 58E00FC8 07FE4180 02D245C0   02A247F0 03444810 0F9C1211 4740035C   *.......H.....K.....0......... ..*
C00300     91011001 47100352 4C710002 90231004   5001000C 92001004 D300100C 0021D201   *........ ...............L.....K.*
000320     0F9C1000 40105088 18A0D200 1008A023   45E00AD0 91EF7006 47708008 91FF0FB0   *.... .....K.....................*
000340     47500E2A 91107006 47100DE6 48AD0006   07FAD502 20150FD1 47800308 58A00024   *...........W......N....J........*
000360     4BA0508C 50A00024 18B09620 B02092F0   09771B99 58A00FBC 5090A000 47FC02E2   *...............0.............0.S*
000380     91102000 471003E6 41A05020 D200A000   302045C0 05E407BC 48A00044 54A05058   *.......W....K........U..........*
0003A0     4770065C 58AC7030 91042001 471003C0   58A20010 91012000 478003C0 58A20018   *................................*
C003C0     91082000 478003DA 50A05030 92085030   41A05028 D200502D 701850A0 004841C0   *....................K...........*
0003E0     066447F0 06249104 20014780 05D647F0   03889140 702C4710 05929101 70064770   *...0.........O.0... ............*
000400     040694E7 20019110 20014710 05709102   70064710 04D241A0 703140A0 503AD203   *...X.................K.... ...K.*
C00420     50007031 91012000 47100432 D2077030   20201BAA 43A70030 89A00004 41AA3020   *............K...................*
000440     91082001 471C0490 D5037033 A0064740   053ED503 7033A00A 4720053E 91027013   *........N...... ..N.............*
000460     471004D2 D5017031 A0044770 053E9104   30084780 0490D501 7035A008 4740048A   *...KN.................N...... ..*
000480     D5017035 A00C47C0 0490D201 7035A008   41A05038 41B00578 45C005E8 47700688   *N.........K................Y....*
0004A0     9D006000 47B004A0 48A00044 54A05058   477006A8 96A27006 D2062009 00419104   *........................K.......*
0004C0     00444780 8008945F 700691A0 50984790   800818B6 88B00008 89B00002 48CB52D4   *...............................M*
0004E0     4BC05096 41A07031 40A0C002 43B07030   89B00004 439B3020 4290C00D D202C011   *........ ...................K...*
000500     20119101 20004780 051CD202 C0112019   91082001 4780051C 9618C00D 50C00048   *..........K.....................*
000520     91027013 47800530 58A00048 47F005E0   45C00624 077C96A6 700647F0 066C58F3   *.............0.............0...3*
000540     001C58FF 000005EF 47F0055C 47F00554   47F00432 41E00960 47F00564 92422004   *.........0...0...0.......0......*
000560     41E00DA2 94FE7006 94DF2000 47F00752   58C20018 47F0051C D5037031 50004770   *.............0...B...0..N.......*
000580     06249602 70064060 70044010 701447F0   04CA4910 702A4770 0DD69148 702C4710   *...... ... ....0.........O......*
0005A0     05AE9101 70064780 0DD247F0 040E9407   702C94DF 200047F0 040E9110 20004710   *.........K.0...........0........*
0005C0     05D658A2 00189101 20004710 05D258A2   001047F0 05E0D200 50082018 41A05008   *.O...........K.....0..K.........*
0005E0     41C00664 41B00624 50A00048 91202000   47800604 910C402C 47800624 943F402C   *...................... ....... .*
000600     94DF2000 58F3001C 58FF0004 50B05074   05EF47F0 061E41E0 096447F0 075258B0   *.....3.............0.......0....*
000620     507407FB 92000048 91017006 47800638   91102001 4710063E D3000048 100C9C00   *........................L.......*
000640     600005A0 88A00018 42A20010 58900FC0   05B91B99 40607004 40107014 58A02010   *.................... ... .......*
000660     04A005CC 4770068E 96A07006 43907004   1A994079 52F0D600 700C509A D7C0700C   *.................. ..0O.....P...*
000680     509A45E0 075207F8 D2C37031 50004720   070C58A0 004841A0 A00850A0 0040D206   *.......8K.................... K.*
0006A0     20090041 471006E4 18E096A0 70069106   00454770 0F8C9110 00444780 0714945F   *.......U........................*
0006C0     70069120 00444710 80049608 70069140   0044071E 91840044 47808008 41808004   *............... ................*
```

Figure PRDMP-14. Sample Dump - General Format



```
                                                        EXTERNAL TRACE - DD TRACEDD                    PAGE 0002


*** DATE   DAY 307    YEAR 1971   TIME          11.15.00                                                         ***


DSP          RES PSW FF060350 80000000  JOBN  N/A       MODN WAITTCB  NUTCB 00013220    PRTY  00
                 CSW 0005A768 0C000000  RQE 44542314 0005A6F8 1B05A71C RQE TCB 0003D3B8    SENS 00200040
DSP          RES PSW FF040001 4000E934  JOBN LISTPDS    MODN SVC-551F NUTCB 0003D3B8    PRTY  1B
SVC     010 OLD PSW FF04000A 4000EA98  JOBN LISTPDS    MODN SVC-551F OLTCB 0003D3B8  R15/R0  0005A750 00000008  R1 8000EA96
SVC     007 CLD PSW FF040007 600223C6  JOBN LISTPDS    MODN SVC-551F OLTCB 0003D3B8  R15/R0  0005A7B0 0005A6F4  R1 0005A5D8
            PLIST    8005A7B8 00000000  NAME IFG0551H
SVC     003 OLD PSW 00040003 60011D78  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0000EBD0 0005A6F4  R1 0005A5D8
DSP          RES PSW FF040007 0000EBD0  JOBN LISTPDS    MODN SVC-551H NUTCB 0003D3B8    PRTY  1B
SVC     007 OLD PSW FF040007 600223C6  JOBN LISTPDS    MODN SVC-551H OLTCB 0003D3B8  R15/R0  0005A7B0 0005A6F4  R1 0005A5D8
            PLIST    8005A7B8 00000000  NAME IFG0553P
SVC     003 OLD PSW 00040003 60011D78  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0000F018 0005A6F4  R1 0005A5D8
DSP          RES PSW FF040007 0000F018  JOBN LISTPDS    MODN SVC-553P NUTCB 0003D3B8    PRTY  1B
SVC     007 OLD PSW FF040007 600223C6  JOBN LISTPDS    MODN SVC-553P OLTCB 0003D3B8  R15/R0  0005A7B0 0005A6F4  R1 0005A5D8
            PLIST    8005A7B8 00000000  NAME IFG0552X
SVC     003 OLD PSW 00040003 60011D78  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0000F460 0005A6F4  R1 0005A5D8
DSP          RES PSW FF040007 0000F460  JOBN LISTPDS    MODN SVC-552X NUTCB 0003D3B8    PRTY  1B
SVC     010 OLD PSW FF04000A 4000F73E  JOBN LISTPDS    MODN SVC-552X OLTCB 0003D3B8  R15/R0  00048DEE 00000008  R1 0005A5D8
SVC     010 OLD PSW FF04000A 4000F6C2  JOBN LISTPDS    MODN SVC-552X OLTCB 0003D3B8  R15/R0  00048DEE 00000218  R1 0005A5E0
SVC     003 OLD PSW FF040003 5000F6CA  JOBN LISTPDS    MODN SVC-552X OLTCB 0003D3B8  R15/R0  00000000 00000218  R1 0005A5E0
DSP          RES PSW FFC50037 60048DEE  JOBN LISTPDS    MODN IEHLIST  NUTCB 0003D3B8    PRTY  1B
SVC     000 OLD PSW FFC50000 400FCD5E  JOBN LISTPDS    MODN IEHLIST  OLTCB 0003D3B8  R15/R0  010FCAC8 00059D40  R1 00059D18
            DDNAME  DDA        DCB  000476F8  DEB  0003CF44
SIO     350 CC  0        CAW C000A568  JOBN LISTPDS                  OLTCB 0003D3B8
                 CSW 0005A768 0C000000  RQE 4434354C 00059D18 1B03CF44 RQE TCB C003D3B8
SVC     001 OLD PSW FFC50001 400FC548  JOBN LISTPDS    MODN IEHLIST  OLTCB 0003D3B8  R15/R0  000FC520 00000001  R1 0004913C
            PLIST    0004913C
DSP          RES PSW FF060236 80000000  JOBN  N/A       MODN WAITTCB  NUTCB 00013220    PRTY  00
I/O     350 OLD PSW FF060350 80000000  JOBN LISTPDS    DDNM DDA      OLTCB 00013220
                 CSW C0059D68 0E400008  RQE 4434354C 00059D18 1B03CF44 RQE TCB C003D3B8    SENS 00001800
DSP          RES PSW FFC50001 400FC548  JOBN LISTPDS    MODN IEHLIST  NUTCB 0003D3B8    PRTY  1B
SVC     055 OLD PSW FFC50037 600FC55E  JOBN LISTPDS    MODN IEHLIST  OLTCB 0003D3B8  R15/R0  0000CF9A 00059D10  R1 000476F8
            DDNAME  DDA
SVC     010 OLD PSW FF04000A 400F9DC6  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0000CF9A 00000218  R1 800F9DBC
SVC     007 OLD PSW FF040007 400F9E1C  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0005A7B0 00000218  R1 000476F8
            PLIST    8005A7B8 00000000  NAME IFG0551F
SVC     003 OLD PSW 00040003 60011D78  JOBN LISTPDS    MODN SVC- RES OLTCB 0003D3B8  R15/R0  0000E788 00000218  R1 000476F8
DSP          RES PSW FF040007 0000E788  JOBN LISTPDS    MODN SVC-551F NUTCB 0003D3B8    PRTY  1B
DSP          RES PSW FF040283 8000E788  JOBN LISTPDS    MODN SVC-551F NUTCB 0003D3B8    PRTY  1B
SVC     000 OLD PSW FF040000 4000E92A  JOBN LISTPDS    MODN SVC-551F OLTCB 0003D3B8  R15/R0  0703D3B8 00221600  R1 0005A6F8
            DDNAME   N/A       DCB  0005A720  DEB  0005A71C
SIO     236 CC  0        CAW 00006550  JOBN LISTPDS                  OLTCB 0003D3B8
                 CSW 0006E6E8 0C000000  RQE 44542314 0005A6F8 1B05A71C RQE TCB 0003D3B8
SIO     236 CC  0        CAW 00006670  JOBN LISTPDS                  OLTCB 0003D3B8
                 CSW 00006558 0C000000  RQE 44542314 0005A6F8 1B05A71C RQE TCB 0003D3B8
SVC     001 OLD PSW FF040001 4000E934  JOBN LISTPDS    MODN SVC-551F OLTCB 0003D3B8  R15/R0  00005EDA 00000001  R1 0005A6F4
            PLIST    0005A6F4
DSP          RES PSW FF060350 80000000  JOBN  N/A       MODN WAITTCB  NUTCB 00013220    PRTY  00
I/O     236 OLD PSW FF060236 80000000  JOBN LISTPDS    DDNM  N/A     OLTCB 00013220
                 CSW 0005A768 0C000000  RQE 44542314 0005A6F8 1B05A71C RQE TCB 0003D3B8    SENS 00200040
DSP          RES PSW FF040001 4000E934  JOBN LISTPDS    MODN SVC-551F NUTCB 0003D3B8    PRTY  1B
SVC     010 OLD PSW FF04000A 4000EA98  JOBN LISTPDS    MODN SVC-551F OLTCB 0003D3B8  R15/R0  0005A750 00000008  R1 8000EA96
```

Figure PRDMP-15. Sample EDIT for Trace Data Set

JCL and Control Statement Examples

The following examples illustrate some of the functions that IMDPRDMP can perform.

Example 1: Using the Cataloged Procedure

IBM supplies a cataloged procedure, called PRDMP, that defines the input and output data sets and a work data set for IMDPRDMP. This example shows how to use the cataloged procedure.

```
//PROCDMP      JOB        MSGLEVEL=(1,1)
//             EXEC       PROC=PRDMP,PARM=DMP=T
//DMP.SYSIN    DD         *
    GO
    END
/*
```

In this example:

EXEC Statement

calls the cataloged procedure, and requests prompting for a dump title.

DMP.SYSIN DD Statement

defines the data set that contains the IMDRPDMP control statements. The data set follows immediately.

GO Control Statement

requests formatting and printing according the the QCBTRACE, LPAMAP, FORMAT, EDIT, and PRINT ALL control statements.

END Control Statement

terminates IMDPRDMP processing.

Example 2: Tranferring a Dump Data Set

If you need to clear the SYS1.DUMP data set quickly to make room for more dump information, you can use IMDPRDMP to transfer its contents to another data set. This new data set is not formatted or printed during this execution of IMDPRDMP, but it can be used as input later.

This example shows how to transfer the SYS1.DUMP data set, which ordinarily is a cataloged data set on direct access storage, to a tape volume described by the SYSUT2 DD statement.

```
//CLEAR       JOB     MSGLEVEL=(1,1)
//            EXEC    PGM=IMDPRDMP
//SYSPRINT    DD      SYSOUT=A
//PRINTER     DD      SYSOUT=A
//TAPE        DD      DSNAME=SYS1.DUMP,DISP=OLD
//SYSUT2      DD      UNIT=2400,VOL=SER=DUMP,LABEL=(,NL),
//       DISP=NEW
//SYSIN       DD      *
         END
/*
```

In this example:

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the data set to which IMDPRDMP ordinarily directs its output. This statement must be included, even though its function is not used in this application.

TAPE DD Statement

defines the input data set, SYS1.DUMP.

SYSUT2 DD Statement

defines the data set to which the contents of SYS1.DUMP will be transferred

SYSIN DD Statement

defines the data set that contains the IMDPRDMP control statements. The data set follows immediately.

END Control Statement

terminates IMDPRDMP processing. Note that this is the only IMDPRDMP control statement needed.

Example 3: Processing Multiple Data Sets

IMDPRDMP can process any number of input data sets in a single execution, provided that each data set is properly defined by both DD statements and control statements. This example shows how to process three data sets in the same execution, two of which are on the same tape volume.

```
//NOLINK      JOB      MSGLEVEL=(1,1)
//            EXEC     PGM=IMDPRDMP,PARM=T
//SYSPRINT    DD       SYSOUT=A
//PRINTER     DD       SYSOUT=A,SPACE=(121,(1600,100))
//TAPE        DD       UNIT=2400,VOL=SER=DPTAPE,
//       LABEL=(,NL),DISP=OLD
//TODAYDMP    DD       UNIT=SYSDA,VOL=SER=DPDADMP,
//       DSNAME=DMPDS,DISP=OLD
//SYSUT1      DD       UNIT=SYSDA,DISP=(NEW,DELETE),
//       SPACE=(2052,(257,10))
//SYSIN       DD       *
    ONGO           Q,F,P A
    GO
    NEWDUMP        FILESEQ=2
    GO
    NEWDUMP        DDNAME=TODAYDMP
    ONGO
    GO
    END
/*
```

In this example:

EXEC Statement

invokes IMDPRDMP and requests that the operator be prompted for a dump title.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the output data set.

TAPE DD Statement

defines two input data sets on the same tape volume.

TODAYDMP DD Statement

identifies an input data set on a direct access volume.

SYSUT1 DD Statement

defines the IMDPRDMP work data set; it is required in this example because one of the input data sets is on a direct access volume.

SYSIN DD Statement

defines the data set containing the control statements. The data set follows immediately.

ONGO Control Statement with Q, F, and P A parameters

alters the default parameters for all subsequent GO statements by deleting the LPAMAP and EDIT parameters.

GO Control Statement

instructs IMDPRDMP to process the first data set on the volume described by the TAPE DD statement.

NEWDUMP Control Statement with FILESEQ=2

identifies the second data set to be processed. Since no DDNAME= parameter is specified, IMDPRDMP assumes that the data set resides on the volume described by the TAPE DD statement. FILESEQ=2 specifies that the second data set on the volume should be processed.

GO Control Statement

instructs IMDPRDMP to process the data set described by the NEWDUMP control statement.

NEWDUMP Control Statement with DDNAME=TODAYDMP

identifies the third data set to be processed. DDNAME=TODAYDMP specifies that the data set is the one described by the TODAYDMP DD statement.

ONGO Control Statement with No Parameters

restores the original default parameters for the GO control statement.

GO Control Statement

instructs IMDPRDMP to process the data set described by the last NEWDUMP control statement. The original default parameters will be used.

END Statement

terminates IMDPRDMP processing.

Example 4: Processing a TSO Dump

IMDPRDMP can produce a complete dump of a TSO system by merging the system dump data set with the TSO swap data sets and formatting and printing the resulting data set. This example shows how to request a TSO dump.

```
//TSODUMP     JOB       MSGLEVEL=(1,1)
//            EXEC      PGM=IMDPRDMP
//SYSPRINT    DD        SYSOUT=A
//PRINTER     DD        SYSOUT=A,SPACE=(121,(32000,100))
//TAPE        DD        UNIT=2400,VOL=SER=DUMP,
//       LABEL=(,NL),DISP=OLD
//SYSUT1      DD        UNIT=SYSDA,DISP=(NEW,DELETE),
//       SPACE=(2052,(513,10))
//SYSWAP00    DD        DSNAME=SYS1.SWAP.D1100,UNIT=2311,
//       VOL=SER=SWAP00,DISP=OLD
//SYSIN       DD        *
    LPAMAP
    FORMAT
    PRINT ALL
    TSO
    END
/*
```

In this example:

PRINTER DD Statement

defines a very large output data set. If you prefer not to allocate so much space to SYSOUT=A, you can direct IMDPRDMP's output directly to a printer by coding this statement as:

```
 //PRINTER DD UNIT=printeraddress
```

CAUTION: In a multiprogramming environment conflicts with the system writers may arise if the output data set is allocated directly to a printer.

TAPE DD Statement

defines the input dump data set.

SYSUT1 DD Statement

defines the IMDPRDMP work data set. Although it is not required in this example, it has been included to reduce IMDPRDMP processing time.

SYSWAP00 DD Statement

defines the swap data set for this particular TSO system. If this system had more than one swap data set, each one would have to be defined on a separate SYSWAPnn DD statement.

SYSIN DD Statement

defines the data set containing the IMDPRDMP control statements. The data set follows immediately.

LPAMAP Control Statement

requests a map of the link pack area of the dumped system.

FORMAT Control Statement

requests that the major control blocks of the dumped system be formatted and printed.

PRINT Control Statement with the ALL Parameter

requests printing of the nucleus, system queue area, and all allocated regions of main storage in the dumped system.

TSO Control Statement with No Parameters

requests formatting and printing of all TSO system and user control blocks and TSO user regions.

END Control Statement

terminates processing.

Note that the GO control statement is not used in this example.

Example 5: Recording the TSO Swap Data Set

If the TSO subsystem fails and must be restarted, or if the operating system fails while TSO is being used, the TSO swap data sets must be recorded so that the failure may be diagnosed. The fastest way to do this is to restart the operating system, if necessary, and use IMDPRDMP to store the swap data set on tape before restarting TSO. Later, if the failure cannot be diagnosed solely by analyzing the main storage dump that was produce when the failure occurred, the swap data set that was stored on tape may be printed using IEBPTPCH.

This example shows how to use IMDPRDMP to store the swap data set and how to use IEBPTPCH to print it later.

```
//SWAPDUMP         JOB       MSGLEVEL=(1,1)
//                 EXEC      PGM=IMDPRDMP,REGION=200K
//SYSPRINT         DD        SYSOUT=A
//PRINTER          DD        UNIT=2400,VOL=SER=SCRTCH,
//         DISP=(NEW,KEEP),LABEL=(,NL),
//         DCB=(BLKSIZE=1210,LRECL=121,RECFM=FB,BUFNO=100,OPTCD=C)
//TAPE             DD        UNIT=2400,VOL=SER=DUMP,
//         DISP=(OLD,KEEP),LABEL=(,NL)
//SYSUT1           DD        UNIT=SYSDA,DISP=(NEW,DELETE),
//         SPACE=(2052,(513,10))
//SYSWAP00         DD        DSN=SYS1.SWAP00,VOL=SER=SWAP00,
//         DISP=(OLD,KEEP),UNIT=2314
//SYSWAP01         DD        DSN=SYS1.SWAP01,VOL=SER=SWAP01,
//         DISP=(OLD,KEEP),UNIT=2314
//SYSIN            DD        *
           TSO
           END
/*
```

```
************************* RESTART TSO ***************************

//PRNTSWAP         JOB       MSGLEVEL=(1,1)
//                 EXEC      PGM=IEBPTPCH
//SYSPRINT         DD        SYSOUT=A
//SYSUT1           DD        UNIT=2400,VOL=SER=SWPDMP,
//       DISP=(OLD,KEEP),LABEL=(,NL),
//       DCB=(BLKSIZE=1210,LRECL=121,RECFM=FB)
//SYSUT2           DD        UNIT=1403
//SYSIN            DD        *
         PRINT     PREFORM=M
/*
```

This example is actually composed of two job steps. In the IMDPRDMP step:

EXEC Statement

invokes IMDPRDMP and overrides the default region size with a value of 200K. This large figure is necessary to accomodate the large number of output buffers requested in the PRINTER DD statement.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the output data set. The output is directed to magnetic tape to make IMDPRDMP execution time as brief as possible; speed is further increased by the blocked records, large number of output buffers, and chain scheduling requested in the DCB operand.

TAPE DD Statement

defines an input dump data set.

SYSUT1 DD Statement

defines the IMDPRDMP work data set. Although it is not required in this example because the input data set is on tape, it is included to reduce IMDPRDMP processing time.

SYSWAP00 and SYSWAP01 DD Statements

define the TSO swap data sets. These statements are identical to those used in the cataloged procedure for starting TSO.

SYSIN DD Statement

defines the data set containing the IMDPRDMP control statements. The data set follows immediately.

TSO Control Statement

requests formatting and printing of TSO system and user control blocks and TSO user regions.

END Statement

terminates IMDPRDMP processing.

In the IEBPTPCH step:

EXEC Statement

invokes IEBPTPCH.

SYSPRINT DD Statement

defines the IEBPTPCH message data set.

SYSUT1 DD Statement

defines the input data set, which in the IMDPRDMP step was the output data set defined by the PRINTER DD statement.

SYSUT2 DD Statement

defines the IEBPTPCH output data set, which in this case is allocated directly to a printer.

SYSIN DD Statement

defines the data set containing the IEBPTPCH control statements. The data set follows immediately.

PRINT control statement with PREFORM=M

tells IEBPTPCH that each record begins with a machine control character.

Example 6: Editing GTF Trace Data from a Dump

```
//EDIT              JOB        MSGLEVEL=(1,1)
//                  EXEC       PGM=IMDPRDMP
//SYSPRINT          DD         SYSOUT=A
//PRINTER           DD         SYSOUT=A
//TAPE              DD         UNIT=2400,VOL=SER=DUMP,LABEL=(,NL),
//         DISP=OLD
//SYSUT1            DD         UNIT=SYSDA,SPACE=(2052,(257,10))
//SYSIN             DD         *
         EDIT
         END
/*
```

In this example:

EXEC Statement

invokes IMDPRDMP.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the output data set.

TAPE DD Statement

defines the input data set.

SYSUT1 DD Statement

defines the IMDPRDMP work data set. Although it is not required unless the input data set is on direct access, it should be included to reduce IMDPRDMP processing time. When it is included, it must specify enough space to contain the entire dump.

SYSIN DD Statement

defines the data set containing the IMDPRDMP control statements. The data set follows immediately.

EDIT Control Statement with No Parameters

instructs IMDPRDMP to format and print GTF trace buffers in the input data set, according to the default options SYS and USR=ALL.

END Control Statement

terminates IMDPRDMP processing.

Example 7: Editing a GTF Trace Data Set

When GTF trace data is recorded in an external data set, you can specify editing of only selected records. This example shows how to edit trace records associated with two specific jobs.

```
//EDIT              JOB       MSGLEVEL=(1,1)
//                  EXEC      PGM=IMDPRDMP,PARM='ER=0'
//SYSPRINT          DD        SYSOUT=A
//PRINTER           DD        SYSOUT=A
//TRACE             DD        UNIT=2400,LABEL=(,NL),VOL=SER=TRACE,
//        DISP=OLD,DCB=(BLKSIZE=2048,BUFNO=10)
//SYSIN             DD        *
          EDIT            DDNAME=TRACE,JOBNAME=X57A
          EDIT            DDNAME=TRACE,JOBNAME=X56B,
                    SIO=IO=(190,191)
          END
/*
```

In this example:

EXEC Statement

invokes IMDPRDMP and specifies the action that IMDPRDMP should take if a program interruption occurs in a user program.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the output data set.

TRACE DD Statement

defines the input trace data set. Since this data set resides on a non-labeled tape, subparameters of the DCB parameter are used to specify the same trace block size as was specified when creating the trace record, and to request that ten input buffers be used to process the trace data.

SYSIN DD Statement

defines the data set containing the IMDPRDMP control statements. The data set follows immediately.

EDIT Control Statement

instructs IMDPRDMP to edit trace records in the data set defined by the TRACE DD statement. The JOBNAME=X57A parameter requests editing for only those records associated with job X57A.

EDIT Control Statement

instructs IMDPRDMP to edit trace records from the data set defined by the TRACE DD statement; that is, the same data set referred to in the first EDIT statement. This time, however, only records associated with job X56B are to be processed; of those, only SIO and I/O interrupt traces for devices 190 and 191 are edited.

END Control Statement

terminates IMDPRDMP processing.

Chapter 9: IMAPTFLE

Generates JCL needed to apply a PTF and/or applies the PTF.

Contents

INTRODUCTION 223
Generate Function 224
Application Function 225

EXECUTING PTFLE 227
Application Function 227
Generate Function 228

CONTROL STATEMENTS 230
IMAPTFLE Control Statements 230
IDENTIFY Control Statement 231

OUTPUT 233
Application Function 233
Generate Function 233

EXAMPLES 236
Example 1: Generate Function .236
Example 2: Application Function 237

OPERATIONAL CONSIDERATIONS 238
General Considerations 238
Generate Function Considerations 239
Application Function Considerations 239

Figures

Figure PTFLE-1. The Generate Function of IMAPTFLE 224
Figure PTFLE-2. The Application Function of IMAPTFLE 225
Figure PTFLE-3. Minimum Main Storage Required for IMAPTFLE When Using the Application Function 227
Figure PTFLE-4. PTFLE Cataloged Procedure. 227
Figure PTFLE-5. Sample JCL Needed to Execute the Generate Function of IMAPTFLE 229
Figure PTFLE-6. Sample Linkage Editor (IEWL) Output from IMAPTFLE Generate Function (Sample #1). 234
Figure PTFLE-7. Sample Linkage Editor (IEWL) Output from IMAPTFLE Generate Function (Sample #2) 235
Figure PTFLE-8. Sample IEBCOPY Output from IMAPTFLE Generate Function 235
Figure PTFLE-9. Sample IEHIOSUP Output from IMAPTFLE Generate Function 235

PTFLE

Introduction

The IMAPTFLE service aid is a problem program that is used to apply program temporary fixes (PTFs) to the IBM System/360 Operating System. You can use IMAPTFLE to;

- Generate the JCL and execution control statements needed to add PTF to an operating system in a later step, or

- Apply PTFs to an operating system by dynamically invoking the linkage editor.

Either the generate function or the application function of IMAPTFLE can be used to add PTFs to an operating system. The method is determined by the PARM operand of the EXEC statement in the execution JCL.

Both functions of IMAPTFLE require the Stage I output from sysgen as input. A brief explanation of the system generation process will clarify this requirement.

An operating system is generated in two stages. During Stage I, user-supplied macro instructions that describe both the installation's machine configuration and the desired programming options are analyzed and used to generate a job stream. The Stage I output contains the JCL that makes up this job stream. In Stage II, the job stream is processed to generate the libraries that form the user's operating system. Each member of these libraries has a certain set of attributes. When a member (load module) is to be modified by a PTF, these attributes must be maintained.

The attributes of the load module being modified by the PTF are contained in the JCL and control statements for the linkage editor and IEBCOPY utility generated during Stage I of system generation (SYSGEN). To ensure that the PTF will be correctly applied, IMAPTFLE uses the Stage I output to determine the attributes of the load module being replaced with the PTF module.

Generate Function

When using the generate function, two steps are required to apply PTFs. In the first step, IMAPTFLE generates the JCL and control statements for the linkage editor and IEBCOPY utility that are needed to apply the PTFs. In the second step, these JCL and control statements are executed to apply the PTFs to the operating system. Figure PTFLE-1 shows the generate function; the shaded area is performed after IMAPTFLE completes processing.

One control statement is provided for each module that comprises the PTF. Each control statement contains the module name and system status index (SSI) for the PTF module. (Alias names of modules that were copied by the IEBCOPY utility during system generation must be provided in additional control statements following the control statements that contain the associated module name. These additional control statements should contain only one alias each. They may not be used to add new alias names.) IMAPTFLE searches the Stage I output for the module names contained in the control statements. From this search, IMAPTFLE produces the necessary JCL and control statements needed to apply the PTFs to the operating system.

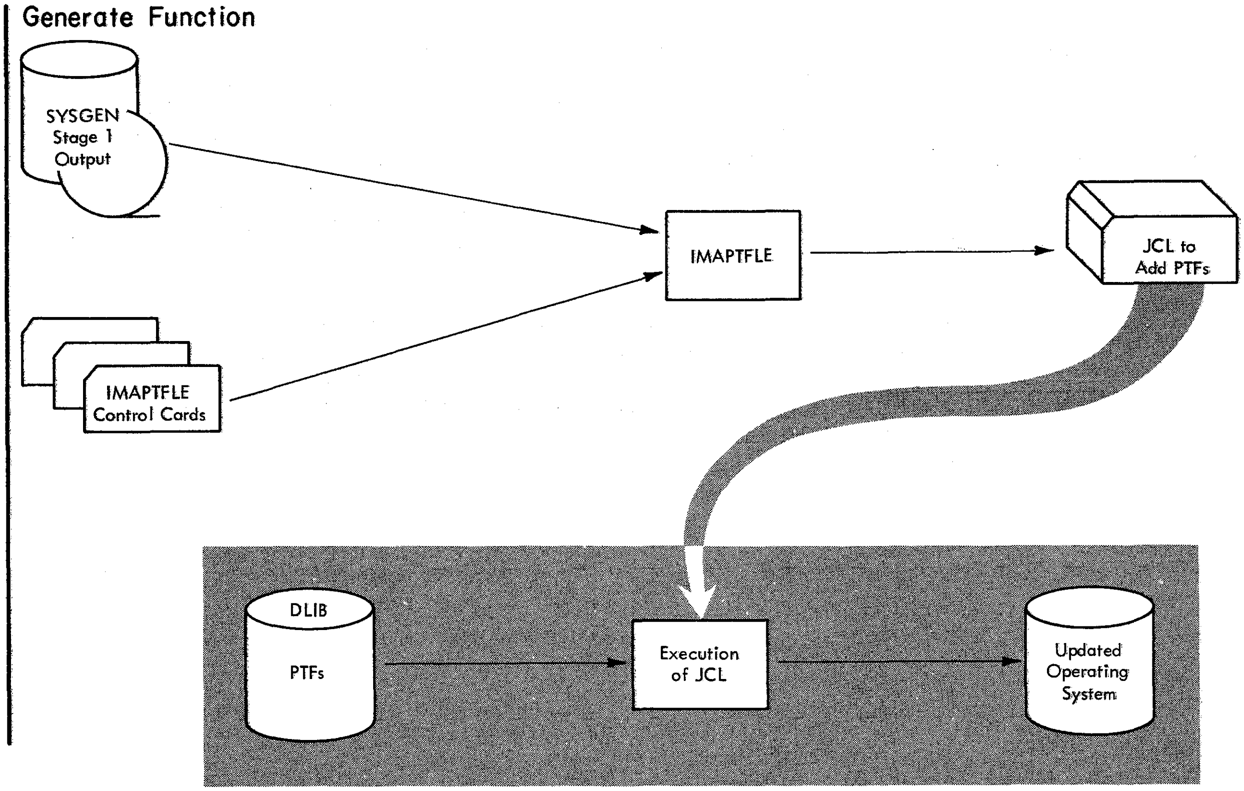


Figure PTFLE-1. The Generate Function of IMAPTFLE

Application Function

When using the application function, only one step is required to apply PTFs. One control statement is used for each module that comprises the PTF. Each control statement contains the module name and system status index (SSI) for the PTF module. (Alias names of modules that were copied by the IEBCOPY utility during system generation must be provided in additional control statements following the control statements that contain the associated module name. These additional control statements should contain only one alias each. They may not be used to add new alias names.) When preparing the input, the PTF object modules are placed immediately behind their corresponding control statement(s), as shown in Figure PTFLE-2.

IMAPTFLE reads all of the control statements and object modules into a work data set, creates a table of PTF module names, and then searches the Stage I output from the generated system being updated. When a module name from the Stage I output matches a PTF module name in the table, IMAPTFLE internally produces the information necessary to apply the PTF, and then invokes the linkage editor to update the operating system. IMAPTFLE then repeats the operation until all PTFs have been applied or the Stage I output reaches end-of-file.

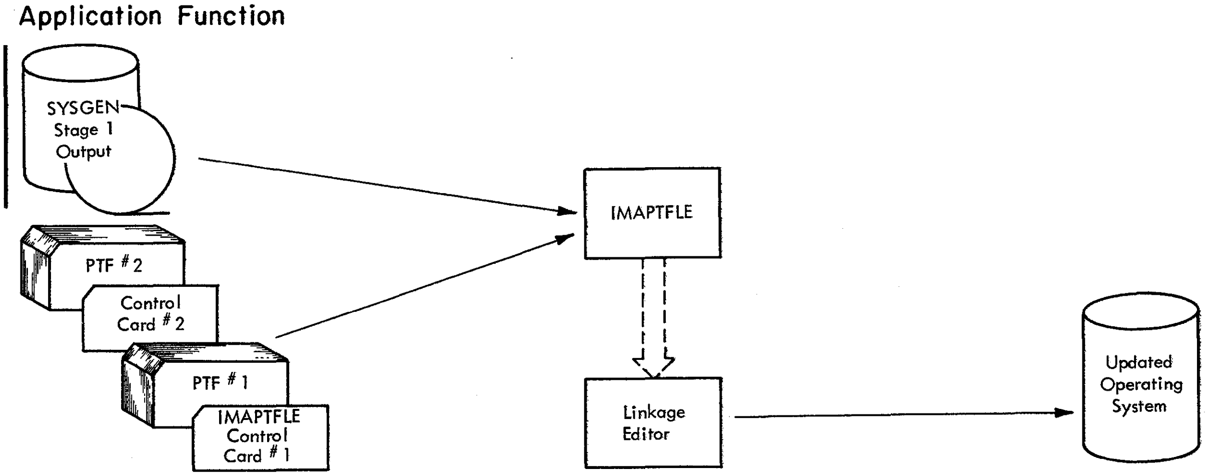


Figure PTFLE-2. The Application Function of IMAPTFLE

Executing IMAPTFLE

The requirements for executing the IMAPTFLE service aid vary according to the desired function: generate or application.

Application Function

For execution of the application function, the main storage space is dependent on both the linkage editor and operating system, as shown in Figure PTFLE-3. Input to IMAPTFLE consists of the Stage I output from the generated system to be updated, IMAPTFLE control statements identifying the CSECTs being replaced, and the object module PTF CSECT replacements. The control statements are discussed under "IMAPTFLE Control Statement."

Design Level	Minimum Main	
of	Storage Requirements	
Linkage Editor	MVT	MFT
44K(F)	68K	58K
88K(F)	109K	103K
128K(F)	149K	144K

Figure PTFLE-3. Minimum Main Storage Required for IMAPTFLE When Using The Application Function.

Figure PTFLE-4 shows the cataloged procedure that IBM supplies for executing the application function of IMAPTFLE. This procedure, called PTFLE, resides in the SYS1.PROCLIB data set.

```
//             PROC      USE='IEWL',LIB1=LINKLIB,REG=68K
//PTF          EXEC      PGM=IMAPTFLE,PARM=&USE,REGION=&REG
//PRINT        DD        SYSOUT=A
//PCHF         DD        UNIT=SYSQ,LABEL=(,NL),DISP=OLD,
//        VOL=SER=STAGE1,DCB=(BLKSIZE=80)
//OUTF         DD        UNIT=SYSDA, SPACE=(TRK,(20,20))
//SYSUT1       DD        UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSUT2       DD        UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSPRINT     DD        SYSOUT=A
//SYSLMOD      DD        DSNAME=SYS1.&LIB1,DISP=OLD
```

Figure PTFLE-4. PTFLE Cataloged Procedure.

The statements in the cataloged procedures and their meanings are:

PTFLE

PROC Statement

defines values for the symbolic parameters in the PTFLE cataloged procedure. The default values are designated by USE, LIB, and REG in the parameter field of this statement.

EXEC Statement

specifies the program to be executed, in this case IMAPTFLE. The PARM= field contains the symbolic parameter &USE that will be assigned the default value of 'IEWL' in the PROC statement; if IEWL is not the linkage editor to be used, override &USE with the name of another linkage editor.

The default value for the symbolic region size (&REG) is 68K; this value assumes that MVT is being used with the 44K linkage editor. If these assumptions do not apply, replace the PROC statement with one that contains the appropriate region size.

PRINT DD Statement

defines the message data set for IMAPTFLE.

PCHF DD Statement

defines the Stage I output from the generated system to be updated. This data set is input to IMAPTFLE. If the data set resides on an unlabeled tape, add a DCB parameter specifying the logical record length (80 bytes) and the blocksize.

OUTF DD Statement

defines a temporary sequential data set used by IMAPTFLE and the linkage editor. This data set may reside on magnetic tape or a direct access device. Do not specify the blocksize.

SYSUT1 DD Statement

defines a work data set for the linkage editor. This data set must reside on a direct access device.

SYSUT2 DD Statement

defines a work data set for IMAPTFLE. This data set must reside on a direct access device. Do not specify the blocksize.

SYSPRINT DD Statement

defines the message data set for the linkage editor.

SYSLMOD DD Statement

defines the output module library for the PTF being added to the system the DSNAME keyword contains the symbolic parameter &LIB1. The &LIB1 parameter is assigned the value LINKLIB from the PROC statement when the procedure is invoked. Before overriding the LINKLIB data set name, see the publication IBM System/360 Operating System: Job Control Language Reference, GC28-6704.

Generate Function

For execution of the generate function, IMAPTFLE requires at least a 46K region or partition. Input to IMAPTFLE must consist of the Stage I output from SYSGEN and control statements identifying the modules for which JCL output is to be produced.

Figure PTFLE-5 illustrates the JCL needed to execute the generate function.

```
//JOB          JOB      MSGLEVEL=(1,1),REGION=46K
//STEP         EXEC     PGM=IMAPTFLE
//PRINT        DD       SYSOUT=A
//OUTF         DD       UNIT=2400,LABEL=(,NL),
//        DISP=(,KEEP),VOL=SER=OUTPUT
//PCHF         DD       UNIT=2400,LABEL=(,NL),
//        DISP=OLD,VOL=SER=SYSGEN,DCB=(BLKSIZE=80)
//MODF         DD       *
    control statements
/*
```

Figure PTFLE-5. Sample JCL Needed to Execute the Generate Function of IMAPTFLE

JOB Statement

initiates the job, and specifies a region size of 46K.

EXEC Statement

invokes IMAPTFLE. Do not specify any other parameters on this statement.

PRINT DD Statement

defines the IMAPTFLE message data set.

OUTF DD Statement

defines a sequential data set to which IMAPTFLE will direct its output. This data set may reside on a direct access device or a magnetic tape, or it may be directed to a SYSOUT data set. Do not specify a block size.

PCHF DD Statement

defines the Stage I output from SYSGEN to be used as input to IMAPTFLE. If an unlabeled tape is used, the DCB parameter specifying logical record length (80 bytes) and blocksize must be specified.

MODF DD Statement

defines the input stream that contains the IMAPTFLE control statements.

Control Statements

Two types of control statement are valid in IMAPTFLE: the IMAPTFLE control statement and the linkage editor IDENTIFY statement. When using the application function, each IMAPTFLE control statement must be followed by the PTF object module named in the control statement, which in turn must be followed by the corresponding IDENTIFY control statement. When using the generate function, the IDENTIFY control statement is optional; if used it must follow the corresponding IMAPTFLE control statement.

The following sections describe the IMAPTFLE control statement and the IDENTIFY control statement.

IMAPTFLE Control Statement

The IMAPTFLE control statement has the following general format:

```
module name     SSI number     comments
```

module name

identifies the name of the module for which JCL is to be created. The length of this name can vary, but it must not exceed eight characters. If an input module can be specified by either of two names (component library name or system library name), the component library name must be used. Statements containing duplicate module names will be ignored by IMAPTFLE. JCL will be produced for the module the first time the name is encountered.

SSI number

reflects the bit settings that are to be placed in the library directory entry for a load module after the PTF has been applied. The SSI information consists of indicators that reflect the status of the load module. The SSI must be updated to show that a module has been modified. The number must begin in column 10 and be exactly eight characters long. To determine the exact bit settings of the SSI before the PTF is applied, the utility program IEHLIST may be used to obtain the current SSI information for all the members of a library.

comments

any user data.

The coding specifications for this statement are:

- Each control statement must contain only one module name and its 8-character System Status Index (SSI) number. (As mentioned, when a user applies a PTF to a module, he is responsible for making sure that the SSI is updated to reflect these changes. For information on the SSI see the discussion "Updating System Status Information" in the IMASPZAP chapter of this publication, and the publication IBM System/360 Operating System: Maintenance Program, GC27-6918.

- The module name must begin in column 1 of the control card. If the module name is less than eight characters, leave blanks between the end of the module name and column 9.

- The SSI number must begin in column 10 of the control card.

- Comments are permitted through and including columns 19 and 80 of the control card.

- Columns 9 and 18 may contain delimiting blanks or commas.

- When using the application function, each control statement must be followed by the PTF object module named in the control statement.

Directory entries for existing alias names of modules that were copied by the IEBCOPY utility during system generation will be updated properly only if such alias names are provided in control statements that follow the control statements for associated module. These additional control statements need not contain SSI information. (Note: The alias names in additional control statements must be only those that appear in the same copy step as the true name of the module in the Stage I output from system generation.)

IMAPTFLE control statements are included in the input stream following the MODF DD statement, as previously described. A /* record denotes the end of input for the execution of IMAPTFLE.

Multiple control statements can be used in any execution of IMAPTFLE, but the total number of control statements must not exceed 150. After the limit has been reached, error message IMA001I will be issued.

The IMAPTFLE control statements may be entered in any order. Any module named in a control statement must exist on the Stage I output tape. Any module names that cannot be found on this tape will be listed by an error message. Duplicate module names detected will also be flagged by the message.

IDENTIFY Control Statement

An IDENTIFY statement for use by the Linkage Editor may also be included in the input defined by the MODF DD statement.

The IDENTIFY statement is not a control statement for IMAPTFLE, but for the linkage editor. IMAPTFLE will copy it (exactly as it appears in the MODF input stream) into the SYSLIN input stream that it creates for the linkage editor.

The IDENTIFY statement is required for the application function and optional for the generate function. For the application function each PTF object module must be followed immediately by an IDENTIFY statement; if the IDENTIFY statement is absent, IMAPTFLE will terminate processing and issue message IMA010I. For the generate function the IDENTIFY statement must follow the IMAPTFLE module name control statement that it is associated with. Only 150 IDENTIFY statements, including continuation statements, are permitted in a job step. If this limit is exceeded, IMAPTFLE will terminate processing with a return code of 16 and issue message IMA011I.

The format of the statement must be identical to that of the Linkage Editor IDENTIFY control statement, as follows:

```
     IDENTIFY  (csectname('data')...,csectname('data'))
csectname('data')
```

csectname

is the symbolic name of the control section that is to be identified. If the CSECT name is changed at system generation by a CHANGE statement, the resulting name should be used.

data

is the identifying information (maximum of 40 characters) that is used to identify the CSECT. This must be enclosed in quotes.

Column one of the statement must be blank. The outer parentheses may be deleted if only one control section is identified in the operand field.

Output

IMAPTFLE produces two different types of output, as described below.

Application Function

The final result of running the IMAPTFLE application function is the updated load module. Because the application function is a self-contained operation, it produces no physical printed output.

Generate Function

The final result of running the IMAPTFLE generate function is a data set that consists of the job control language statements, linkage editor control statements, and the IEBCOPY control statements needed to add the PTFs to the generated operating system in a later run. Three types of JCL statements are produced:

- Linkage Editor (IEWL) JCL: This type of JCL is produced if the load module requested for processing was originally link edited into the system during system generation.
- IEBCOPY JCL: This type of JCL is produced if the member was originally copied into the system.
- IEHIOSUP JCL: This type of JCL is produced in addition to LINK EDIT and/or IEBCOPY JCL. The IEHIIOSUP statements are used to execute the IEHIOSUP utility. This program updates any TTR entries in the transfer control tables of the supervisor call library (SVC library) that may require a change as a result of applying a PTF.

Figures PTFLE-6, 7, 8, and 9 show sample output from the generate function of IMAPTFLE. All of these samples were derived by using the IMAPTFLE JCL and control statements illustrated in Figure PTFLE-5. For a more detailed explanation of the JCL statements and their parameters, refer to the publication IBM System/360 Operating System: Job Control User's Guide, GC28-6703.

Note: The generate function IMAPTFLE will produce a JOB statement to precede any other JCL produced.

```
//SG43         EXEC      PGM=IEWL,COND=(8,LT),
//       PARM='NCAL,LIST,XREF,OVLY,LET,DC'
//SYSUT1       DD        DISP=OLD,VOLUME=(,RETAIN),DSNAME=SYS1.UT3
//SYSPRINT     DD        SPACE=(121,(500,100),RLSE),DCB=(RECFM=FB,
//       LRECL=121,BLKSIZE=121),SYSOUT=A
//SYSLMOD      DD        DISP=OLD,UNIT=2311,VOLUME=SER=111111,
//       DSNAME=SYS1.LINKLIB
//UT506        DD        DISP=OLD,VOLUME=(,RETAIN),DSNAME=SYS1.UT506
//SYSPUNCH     DD        DISP=OLD,VOLUME=(,RETAIN),
//       DCB=(,RECFM=F,BLKSIZE=80),DSNAME=SYS1.OBJECT
//SYSLIN       DD        *

         INCLUDE UT506(IEBGEN03)
         ENTRY IEBGENER
         INCLUDE SYSLMOD(IEBGENER)
         OVERLAY1
         INSERT IEBCCS02
         INSERT IEBGSCAN
         OVERLAY1
         INSERT IEBGENR3
         INSERT IEBCONP2
         INSERT IEBCONH2
         INSERT IEBCONZ2
         INSERT IEBEDIT2
         INSERT IEBLENP2
         INSERT IEBMOVE2
         OVERLAY2
         INSERT IEBGENS3
         OVERLAY2
         INSERT IEBGEN03
         SETSSI 05199133
         NAME IEBGENER(R)
    /*
```

Figure PTFLE-6. Sample Linkage Editor (IEWL) Output from IMAPTFLE Generate Function (Sample #1)

```
//SG63         EXEC      PGM=IEWL,COND=(8,LT),
//       PARM='NCAL,LIST,XREF,DC'
//SYSUT1       DD        DISP=OLD,VOLUME=(,RETAIN),DSNAME=SYS1.UT3
//SYSPRINT     DD        SPACE=(121,(500,100),RLSE),DCB=(RECFM=FB,
//       LRECL=121,BLKSIZE=121),SYSOUT=A
//SYSLMOD      DD        DISP=OLD,UNIT=2311,VOLUME=SER=111111,
//       DSNAME=SYS1.LINKLIB
//AL531        DD        DISP=OLD,VOLUME=(,RETAIN),DSNAME=SYS1.AL531
//SYSPUNCH     DD        DISP=OLD,VOLUME=(,RETAIN),
//       DCB=(,RECFM=F,BLKSIZE=80),DSNAME=SYS1.OBJECT
//SYSLIN       DD        *
         INCLUDE AL531(IEX51)
         ENTRY IEX51000
         ALIAS IEX51000,IEX51002,IEX51ER1,IEX51ER2
         INCLUDE SYSLMOD(IEX51)
         IDENTIFY IEX51000('PTF20191')
         SETSSI 02150191
         NAME IEX51(R)
    /*
```

Figure PTFLE-7. Sample Linkage Editor (IEWL) Output from IMAPTFLE Generate Function (Sample #2)

```
//SG44        EXEC      PGM=IEBCOPY,COND=(8,LT)
//SYSUT3      DD        DISP=SHR,DSNAME=SYS1.UT3
//SYSPRINT    DD        SPACE=(121,(500,1000),RLSE),
//                 DCB=(RECFM=FB,LRECL=121,BLKSIZE=121),
//                 SYSOUT=A
//CI505       DD        DISP=SHR,VOLUME=(,RETAIN),DSNAME=SYS1.CI505
//SVCLIB      DD        DSNAME=SYS1.SVCLIB,VOLUME=(,RETAIN,SER=SYSRES),
//                 UNIT=2314,DISP=OLD
//SYSIN       DD        *
    COPY OUTDD=SVCLIB,INDD=CI505
    SELECT    MEMBER=((IGE0000A,,R))
    SELECT    MEMBER=((IGE0000D,,R))
    SELECT    MEMBER=((IGE0000G,,R))
/*
```

Figure PTFLE-8. Sample IEBCOPY Output from IMAPTFLE Generate Function

```
//SG79        EXEC      PGM=IEHIOSUP
//SYSPRINT    DD        SPACE=(121,(500,1000),RLSE,DCB=RECFM=FB,
//                 LRECL=121,BLKSIZE=121),SYSOUT=A
//SYSUT1      DD        DSNAME=SYS1.SVCLIB,DISP=(OLD,PASS),
//                 VOLUME=(,RETAIN,SER=111111),UNIT=2311
```

Figure PTFLE-9. Sample IEHIOSUP Output from IMAPTFLE Generate Function

PTFLE

Examples

Example 1: Generate Function

This example shows the JCL and control statements needed to execute the generate function of IMAPTFLE. In this case, the input data set from sysgen resides on a magnetic tape.

```
//JOB         JOB       MSGLEVEL=(1,1)
//STEP        EXEC      PGM=IMAPTFLE
//PRINT       DD        SYSOUT=A
//OUTF        DD        UNIT=SYSDA,VOL=SER=OUTPUT,DISP=(,KEEP),
//                 DSNAME=DAOUTPUT,SPACE=(TRK,(20,10))
//PCHF        DD        UNIT=2400,LABEL=(,NL),DISP=OLD,
//                 VOL=SER=SYSGEN,DCB=(BLKSIZE=80)
//MODF        DD        *
IEBGEN03 05199133
IEX51    02150191
 IDENTIFY IEX51000('PTF20191')
IGE0000A 03144004
IGE0000D 02155123
IGE0000G 05194025
/*
```

In this example:

JOB Statement

initiates the job.

EXEC Statment

invokes IMAPTFLE.

PRINT DD Statement

defines the message data set.

OUTF DD Statement

defines the output data set, in this case residing on a direct access volume.

PCHF DD Statement

defines the input data set containing the Stage I SYSGEN output.

MODF DD Statement

defines the input stream that contains the IMAPTFLE control statements.

Example 2: Application Function

This example illustrates the JCL needed to execute the Application function of IMAPTFLE using the cataloged procedure PTFLE.

```
//PTFPROC           JOB       MSGLEVEL=(1,1)
//STEP              EXEC      PTFLE
//PTF.MODF          DD        *
IEFSD082 01117251
    Insert PTF Object Deck
    Insert Linkage Editor IDENTIFY Statement
IEFSD085 01117251
    Insert PTF Object Deck
    Insert Linkage Editor IDENTIFY Statement
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes the PTFLE cataloged procedure, which executes the application function of IMAPTFLE. When PTFLE is invoked, these statements merge with the JCL statements in the cataloged procedure.

PTF.MODF DD Statement

defines the input stream, which contains the IMAPTFLE control statements.

IMAPTFLE Control Statements

identify the module to be updated with the PTF, and supplies the SSI information to be placed in the library directory entry for the module once the PTF has been successfully applied.

IDENTIFY Control Statements

identify the CSECT within the module identified by the IMAPTFLE control statement that is to be updated with a PTF, and supplies information needed to identify that CSECT once the PTF application is successful.

Operational Considerations

Before attempting to use IMAPTFLE, the following considerations should be examined.

General Considerations

- IMAPTFLE will not accept more than 150 module names as input. If the number of names exceeds this limit, the job must be divided into more than one job of no more than 150 module names each. If control statements are provided for alias names of modules that were copied during system generation, these additional names must be counted toward the total of 150 when the generate function is being invoked.

- The Stage I output must be from the generated system of the operating system being updated with the PTFs.

- If Stage I output is an unlabeled tape, the DCB parameter containing the logical record length and blocksize must be added to the PCHF DD statement.

- The Stage I output must not contain control characters (i.e., printer or punch).

- If an input module name can be specified by either of two names (component library name or system library name), the component library name must be used. For example, IEAATM02 is a component library name; its system name is IGC0201C. If JCL were required for this module, IEAATM02 would have to be specified as the input module name.

- It is the user's responsibility to ensure that the SSI is correctly updated when the module is applied to the system. The user, therefore, should make sure that the correct SSI information is placed on each control card. (The correct SSI data appears on the cover letter for the PTF.) Absence of the SSI on the control card will cause the SSI in the module's directory entry to be set to zeros.

- If an input load module was created from multiple load modules in the distribution library, the user should make sure that a linkage editor ENTRY statement exists for that module in the Stage I output from system generation. If no such statement is present, IMAPTFLE should not be used, since it may cause the module to be updated with an incorrect entry point.

- IMAPTFLE should not be used to apply a PTF to a module if the module name in the distribution library is different from the CSECT name in the module, and if the module's overlay structure was defined during system generation by INCLUDE statements rather than by INSERT statements. An example of such a module is the FORTRAN H compiler.

- IMAPTFLE should not be used to apply a PTF to a module that is a member of a library copied totally from the distribution library at system generation. Libraries containing modules to be processed by IMAPTFLE should have been copied selectively by the IEBCOPY utility during system generation (that is, the SELECT statement must have been used.)

Generate Function Considerations

- IMAPTFLE will not produce JCL for either an IMASPZAP PTF (discussed in Section III of this publication) or a PTF that requires some degree of system generation for its application.

- IMAPTFLE requires Stage I output from a system generation of Release 19 or later. Output from earlier system generations cause the IMAPTFLE program to be terminated with an error message.

- The system library being updated by a PTF must not be used as a driver to run the JCL job stream created by IMAPTFLE. It is recommended that the STARTER SYSTEM be used instead.

- The user should verify that both the component libraries and the four utility data sets are cataloged on the driver system before the PTF is applied. (For more complete information on utility data sets, refer to the publication IBM System/360 Operating System: Utilities, GC28-6670. System data sets are cataloged, and successful application of the PTF therefore depends on their being cataloged as described.

- IMAPTFLE does not produce JCL to apply PTFs to the Distribution Libraries (DLIBs). The JCL produced by IMAPTFLE is designed to be used in updating the system by using the DLIBs. Therefore, before running the JCL produced by IMAPTFLE, the user must apply the PTFs to the DLIBs to ensure a successful update of the system when the JCL stream is run.

- The IMAPTFLE generate function will not accept more than 150 IDENTIFY cards, including continuation cards. If the number of cards exceeds this limit, the job should be divided into more than one step of no more than 150 IDENTIFY cards and continuation cards each.

Application Function Considerations

- PTFs containing multiple CSECTs can only be applied to load modules residing on the same system library.

- TTR entries in the transfer control tables of the supervisor call library (SVCLIB) are updated for PTFs applied to SYS1.SVCLIB. It is not necessary to run the IEHIOSUP utility.

Chapter 10: IMDSADMP

Operates as a stand-alone program to produce a high-speed or low-speed dump of main storage.

SADMP

Contents

INTRODUCTION 245

SIZE OF SADMP 246

THE HIGH-SPEED DUMP PROGRAM 248
Loading the High-Speed Dump Program 248
Output of the High-Speed Dump Program 248

THE LOW-SPEED DUMP PROGRAM 250
Loading the Low-Speed Dump Program 250
Output of the Low-Speed Dump Program 250

SPECIFYING THE DUMP 253

RETRIEVING AND CREATING THE DUMP PROGRAM 258
The Specification Step 258
The Initialization Step 262
    Tape Initialization (High-Speed Only) 262
    Direct Access Initialization 263

EXECUTING THE DUMP PROGRAM 265

OPERATIONAL CONSIDERATIONS 267

ERROR CONDITIONS 270
Error Handling 270
    Errors During Initialization of Direct Access Resident Programs . 270
    Errors During Dump Execution 270
Macro Expansion Messages 272

Figures

Figure SADMP-1. Size of the IMDSADMP Program in Bytes 246
Figure SADMP-2. Format of Cylinder 0, Track 0 for Disk Resident IMDSADMP 247
Figure SADMP-3. Output Tape Formats for the High Speed Version of the Dump Program 249
Figure SADMP-4. IMDSADMP Low-Speed Dump Output Sample 251
Figure SADMP-5. IMDSADMP Low-Speed Dump Output Sample of a Model 65 Multiprocessing System 252
Figure SADMP-6. IMDSADMP Parameter-Dependent Program Generation . . 253
Figure SADMP-7. The IMDSADMP Macro Instruction Statement 254
Figure SADMP-8. Availability of IMDSADMP Macro Definition Statements 259
Figure SADMP-9. An Example of IMDSADMP JCL Statements for for Designating SYSLIB 260
Figure SADMP-10. Example of the IMDSADMP Specification Step 261
Figure SADMP-11. IMDSADMP Initialization Procedures 262
Figure SADMP-12. An Example of IEBGENER JCL Statements for Dump Tape 263
Figure SADMP-13. Maximum Number of User Labels Depending on Device and Options Selected 263

Introduction

When a system goes into a disabled wait state or an unending loop, a stand-alone dump program is needed to dump the contents of main storage so that the condition can be analyzed. Optimally, this dump program should be high-speed so that the system is inoperative for as short a period of time as possible. IBM provides IMDSADMP for this purpose. IMDSADMP is a macro instruction that allows a user to generate a stand-alone dump program specifically tailored to his installation's needs.

IMDSADMP can generate two types of dump program: a high-speed version that can quickly write the contents of main storage to a tape volume in large blocks, and a low-speed version in which the contents of main storage are written to either a printer or a tape volume in unblocked, printable format.

The high-speed version of the dump program may reside on either a tape or direct access volume; the low-speed version may reside only on a direct access volume. See the IMDPRDMP service aid for instructions on processing the high-speed output of IMDSADMP.

Creation and usage of the dump program is simple. The user employs the IMDSADMP macro instruction to define the type of dump program he wants (see the topic "Specifying the Dump"). The dump creation process includes a specification step and an initialization step. In the specification step, the macro instruction is assembled with the IBM-provided IMDSADMP macro definition. This specification step produces:

- IPL text necessary to make the dump program loadable for execution.

- Code that allows the IPL text and the dump program module to be stored on a selected tape or direct access volume.

- The dump program itself.

In the initialization step, the IPL text and the dump program module are placed on the specified device. To execute the dump program, the user loads it into main storage from the device by means of standard IPL procedure. The main storage dump information is written to either a tape or printer device based upon user-specified operands of the IMDSADMP macro instruction. During execution of the direct access resident version of the dump program, the operator can override the device address which was specified as a result of the expansion of the macro instruction.

The two steps required to create an executable dump program and a discussion of dump program execution follow the detailed descriptions of the high and low-speed versions.

Multiprocessing: In multiprocessing systems, IMDSADMP can dump the contents of the registers in both CPUs when the direct control feature is operational, and can dump all of addressable main storage. This is accomplished by an optional parameter of the IMDSADMP macro instruction.

Size of SADMP

The size of the assembled IMDSADMP program depends on the output option selected and whether or not IMDSADMP will be on a multiprocessing system; see Figure SADMP-1. The size of IMDSADMP is the same both in main storage and on the resident volume; see Figure SADMP-2.

Output Option	Without Multiprocessing	With Multiprocessing
High-Speed	1024	1088
Low-Speed Printer	1088	1344
Low-Speed Tape	1280	1472

Figure SADMP 1. Size of the IMDSADMP Program in Bytes

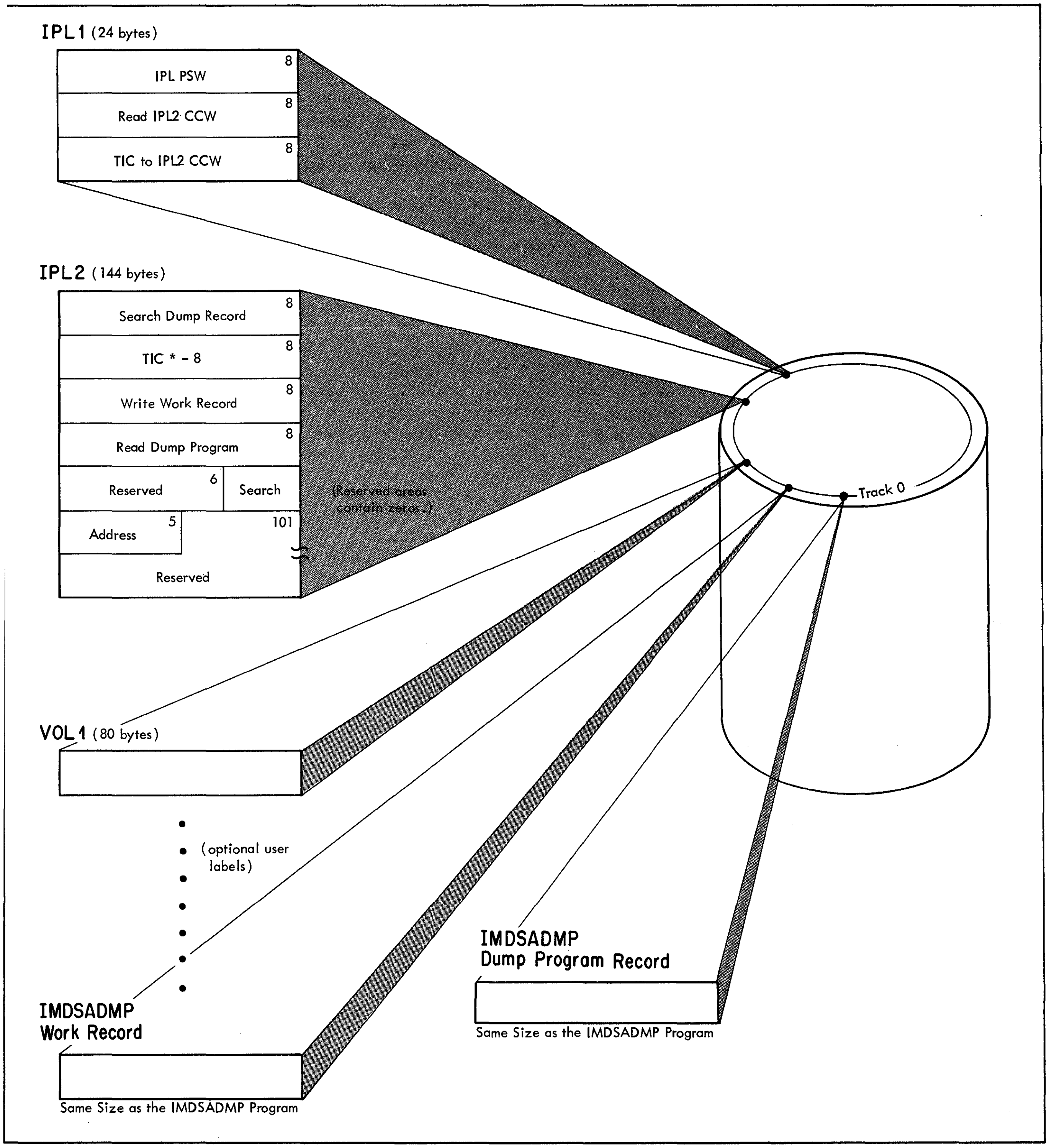


Figure SADMP 2. Format of Cylinder 0, Track 0 for Disk Resident IMDSADMP

The High-Speed Dump Program

This version of the IMDSADMP generated dump program (hereinafter referred to as the dump program) dumps the contents of main storage to a tape volume. Each dump record is 2052 bytes long. To further expedite the dump and conserve program storage requirements, the main storage information is written to a nonlabeled tape volume in an untranslated, hexadecimal form. Formatting, converting and printing of the information is performed by the IMDPRDMP service aid.

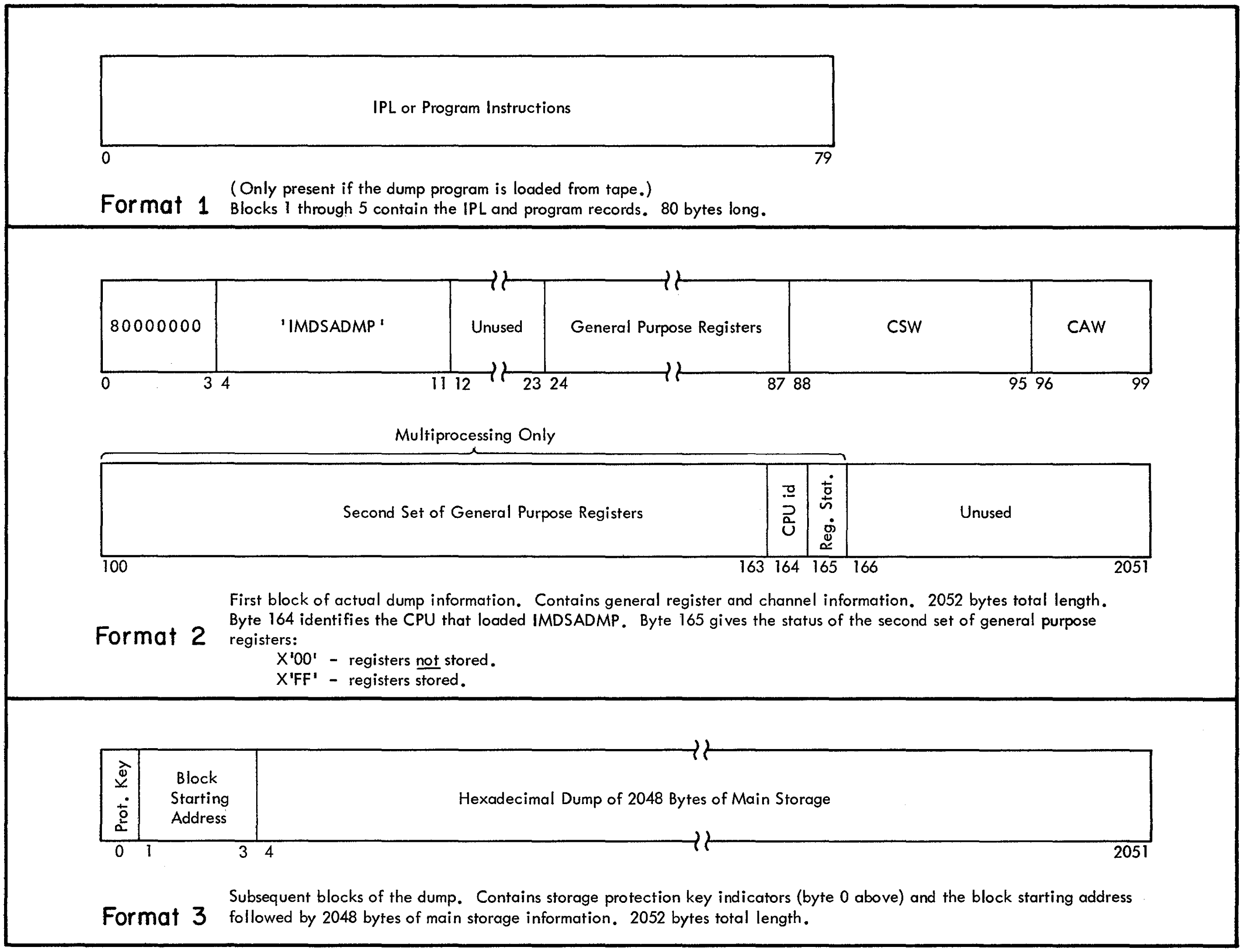
Loading the High-Speed Dump Program

The high-speed dump program may reside on either a tape or direct access volume. In either case, the user loads the program from the device into main storage by means of the IPL procedure. The high-speed dump program is loaded into the CPU Log Out Area or into a storage specified by the user through an operand of the IMDSADMP macro instruction. If IMDSADMP is loaded into the CPU logout area, IMDSADMP destroys the contents of the logout area. In case of hardware errors, or when requested by the system, it may be necessary to display the contents of the CPU Log Out Area before invoking the dump program. This can be done by executing the System Environment Recording, Edit and Print routine, SEREP, which is discussed in the publication IBM System/360 Operating System: Operator's Reference, GC28-6691.

Output of the High-Speed Dump Program

If the user selects the high-speed version of the dump program during the specification step, he must select a tape device as the output medium, even though the dump program itself may reside on either tape or disk. The input device type selected has an effect on output retrieval.

If the dump program resides on a tape volume, the dump information is written to the nonlabeled tape that contains the program. The information, in untranslated, hexadecimal form, follows the IPL text and the dump program module records (see Figure SADMP-3, format 1). Each dump information record is 2052 bytes long (see Figure SADMP-3, formats 2 and 3).



(Only present if the dump program is loaded from tape.)
Blocks 1 through 5 contain the IPL and program records. 80 bytes long.

First block of actual dump information. Contains general register and channel information. 2052 bytes total length. Byte 164 identifies the CPU that loaded IMDSADMP. Byte 165 gives the status of the second set of general purpose registers:

X'00' - registers not stored.
X'FF' - registers stored.

Subsequent blocks of the dump. Contains storage protection key indicators (byte 0 above) and the block starting address followed by 2048 bytes of main storage information. 2052 bytes total length.

Figure SADMP 3. Output Tape Formats for the High Speed Version of the Dump Program

If the dump program resides on a direct access device, the 2052-byte dump information records are written to the nonlabeled output tape volume (see Figure SADMP-3, formats 2 and 3). The IPL text and dump program records and work record are contained on cylinder 0, track 0 of the volume on which the dump program resides (see Figure SADMP-2). The work record is used to temporarily record the main storage information from the area into which the dump program is to be loaded.

The Low-Speed Dump Program

The low-speed version of the dump program writes the contents of main storage to either a printer or a tape device. If output is to tape, the information may be subsequently printed by a program such as the IEBGENER utility program, as discussed in the publication IBM System/360 Operating System: Utilities, GC28-6586, or by IMDPRDMP.

Loading the Low-Speed Dump Program

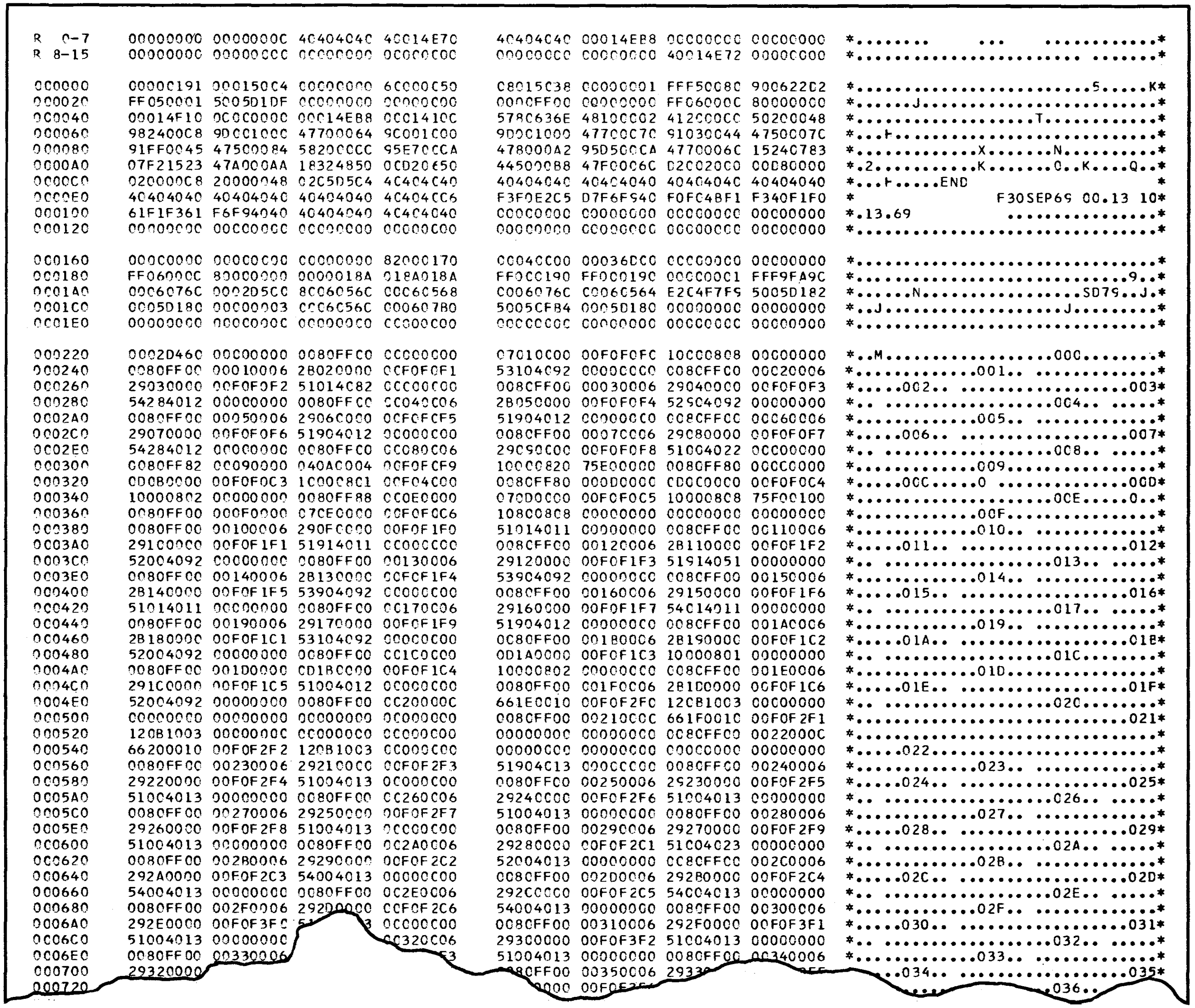
The low-speed dump program must reside on disk. To execute the program, the user performs the IPL procedure to load the dump program from its resident device. The IPL statements and dump program reside on cylinder 0, track 0 (see Figure SADMP-2).

During the specification step, the user may either select an address at which to begin loading, or use the default value. If the user selects his own starting address, the value he specifies must be at least 128 decimal or 80 hexadecimal.

Output of the Low-Speed Dump Program

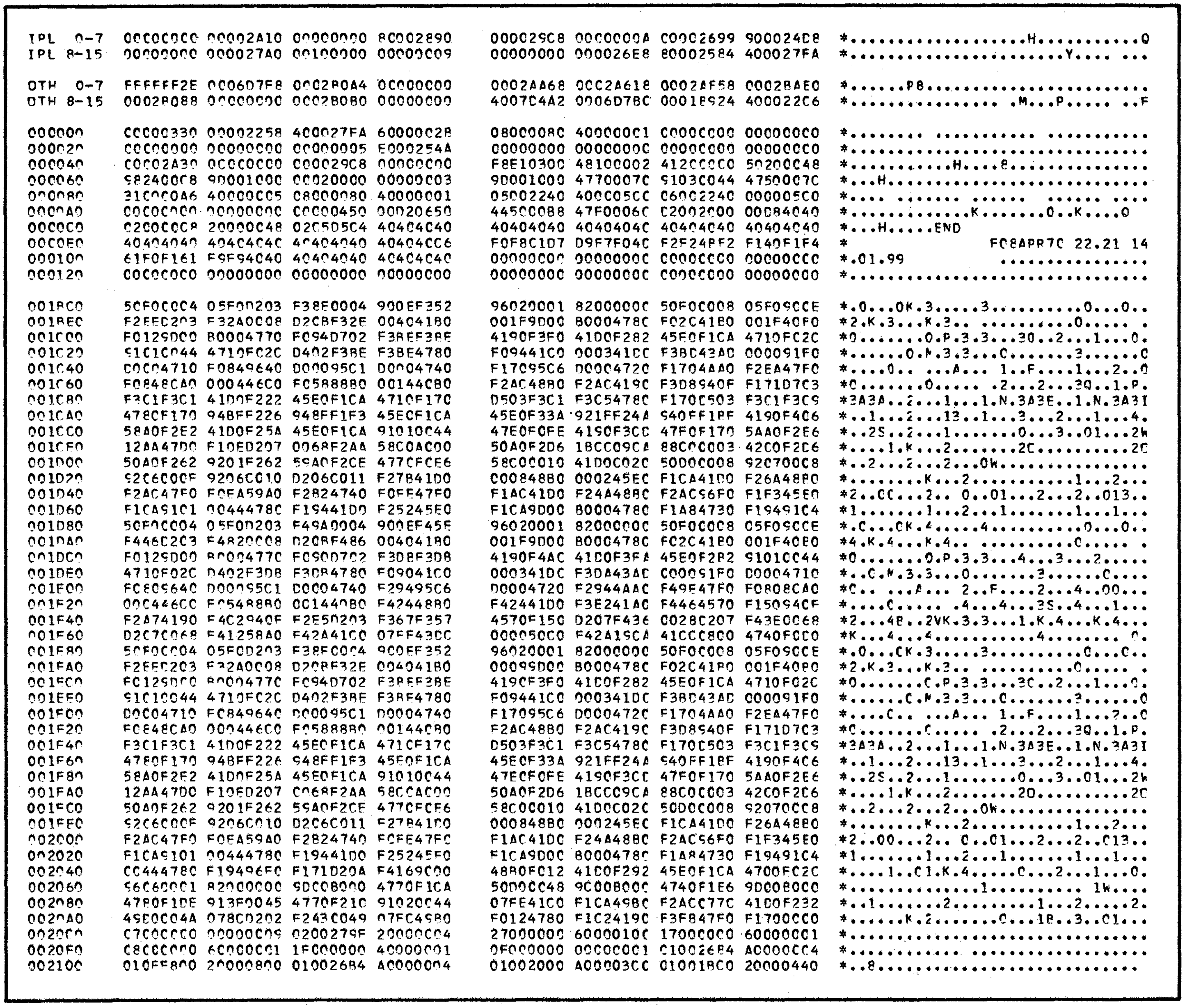
The low speed version of the dump program writes dump information to either a tape volume or a printer. The format of the main storage information is the same, regardless of the output device type to which it is being written. Each dump record contains 120 characters of formatted dump information. An output sample is shown in Figure SADMP-4. The contents of the general purpose registers are printed first, followed by the remainder of main storage. (Note that for low-speed dumps of a Model 65 Multiprocessing System, IMDSADMP shows both sets of general purpose registers; see Figure SADMP-5.) A storage location field containing the address of the first byte is printed to the left of each line. A character translation field, showing the EBCDIC translation of the hexadecimal contents, is displayed to the right of each line. Only alphabetic or numeric representations of hexadecimal information are given in the character translation field; all other bytes are represented by a period. If a line duplicates the contents of the previous line, it is not printed; instead the duplicate line is left blank.

If the output of the dump program is directed to a tape volume, each dump information record is preceded by a one-byte ASA character that is used by the subsequent printing program to control printer spacing. This results in a total record length of 121 bytes. This tape volume may be printed by using the IEBGENER utility program or IMDPRDMP.



```
R  0-7    00000000 0000000C 40404040 40014E70    40404040 00014EB8 00000000 00000000   *........     ...    ............*
R 8-15    00000000 00000000 00000000 00000000    00000000 00000000 40014E72 00000000   *........................ .......*

000000    00000191 000150C4 00000000 6C000C50    C8015C38 00000001 FFF50080 900622D2   *.........................5.....K*
000020    FF050001 5005D1DF 00000000 00000C00    0000FF00 00000000 FF06000C 80000000   *......J.........................*
000040    00014F10 0C0C000C 00014EB8 0C01410C    578C636E 48100C02 412C00CC 50200048   *...................T............*
000060    982400C8 9DCC100C 47700064 9C001C00    9D0C1000 4770C07C 91030044 4750007C   *...H............................*
000080    91FF0045 47500084 58200CCC 95E70CCA    478000A2 95D500CA 4770006C 15240783   *.............X.......N..........*
0000A0    07F21523 47A000AA 18324850 0CD20650    445000B8 47F0006C D2002000 00D80000   *.2...........K.......0..K....Q..*
0000C0    020000C8 20000048 02C5D5C4 40404040    40404040 40404040 40404040 40404040   *...H.....END                    *
0000E0    40404040 40404040 40404040 404040C6    F3F0E2C5 D7F6F940 F0F04BF1 F340F1F0   *               F30SEP69 00.13 10*
000100    61F1F361 F6F94040 40404040 40404040    00000000 00000000 00000000 00000000   *.13.69          ................*
000120    00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*

000160    00000000 00000C00 00000000 82000170    00040C00 00036D00 00000000 00000000   *................................*
000180    FF06000C 80000000 0000018A 018A018A    FF000190 FF000190 000000C1 FFF9FA9C   *.............................9..*
0001A0    0006076C 0002D5C0 8C06056C 00060568    C006076C C0060564 E2C4F7F9 5005D182   *......N.................SD79..J.*
0001C0    0005D180 00000003 C006C56C 000607B0    5005CFB4 0005D180 00000000 00000000   *..J...................J.........*
0001E0    00000000 0000000C 000000C0 CC000C00    0000000C 00000000 0000000C 00000000   *................................*

000220    0002D460 00000000 0080FFC0 00000000    07010C00 00F0F0F0 10C00808 00000000   *..M..................000........*
000240    0080FF00 00010006 2B020000 00F0F0F1    53104092 00000000 008CFFC0 00020006   *.............001.. .............*
000260    29030000 00F0F0F2 51014082 00000000    008CFF00 00030006 29040000 00F0F0F3   *.....002.. ..................003*
000280    54284012 00000000 0080FFC0 00040006    2B050000 00F0F0F4 52904092 00000000   *.. ..................004.. .....*
0002A0    0080FF00 00050006 29060000 00F0F0F5    51904012 00000000 008CFFC0 00060006   *.............005.. .............*
0002C0    29070000 00F0F0F6 51904012 00000C00    008CFF00 0007C006 29080000 00F0F0F7   *.....006.. ..................007*
0002E0    54284012 00000000 0080FFC0 00080006    29090000 00F0F0F8 51004022 00000000   *.. ..................008.. .....*
000300    0080FF82 00090000 040A0004 00F0F0F9    10000820 75E00000 0080FF80 000C0000   *.............009................*
000320    0D0B0000 00F0F0C3 1C00C8C1 00F04000    008CFF80 000D0000 CD0C0000 00F0F0C4   *.....00C.....0 ..............00D*
000340    10000802 00000000 0080FF88 000E0000    070D0000 00F0F0C5 10000808 75F00100   *.....................00E.....0..*
000360    0080FF00 000F0000 07CE0000 00F0F0C6    10800808 00000000 00000000 00000000   *.............00F................*
000380    0080FF00 00100006 290F0000 00F0F1F0    51014011 00000000 008CFF00 00110006   *.............010.. .............*
0003A0    291C0000 00F0F1F1 51914011 00000000    008CFF00 00120006 2B110000 00F0F1F2   *.....011.. ..................012*
0003C0    52004092 00000000 0080FF00 00130006    29120000 00F0F1F3 51914051 00000000   *.. ..................013.. .....*
0003E0    0080FF00 00140006 2B130000 00F0F1F4    53904092 00000000 008CFF00 00150006   *.............014.. .............*
000400    2B140000 00F0F1F5 53904092 00000C00    0080FF00 00160006 29150000 00F0F1F6   *.....015.. ..................016*
000420    51014011 00000000 0080FF00 00170006    29160000 00F0F1F7 54014011 00000000   *.. ..................017.. .....*
000440    0080FF00 00190006 29170000 00F0F1F9    51904012 00000000 008CFF00 001A0006   *.............019.. .............*
000460    2B180000 00F0F1C1 53104092 00000C00    0C80FF00 001B0006 2B190000 00F0F1C2   *.....01A.. ..................01B*
000480    52004092 00000000 0080FF00 001C0000    0D1A0000 00F0F1C3 10000801 00000000   *.. ..................01C........*
0004A0    0080FF00 001D0000 CD1B0000 00F0F1C4    10000802 00000000 008CFF00 001E0006   *.............01D................*
0004C0    291C0000 00F0F1C5 51004012 00000C00    0080FF00 001F0006 2B1D0000 00F0F1C6   *.....01E.. ..................01F*
0004E0    52004092 00000000 0080FF00 0020000C    661E0010 00F0F2F0 12CB1003 00000000   *.. ..................020........*
000500    00000000 00000000 00000000 00000000    008CFF00 0021000C 661F0010 00F0F2F1   *.............................021*
000520    120B1003 0000000C 00000000 00000C00    00000000 00000000 008CFF00 0022000C   *................................*
000540    66200010 00F0F2F2 120B1003 00000C00    00000000 00000000 00000000 00000000   *.....022........................*
000560    0080FF00 00230006 29210000 00F0F2F3    51904C13 00000000 0080FF00 00240006   *.............023.. .............*
000580    29220000 00F0F2F4 51004013 00000C00    0080FF00 00250006 29230000 00F0F2F5   *.....024.. ..................025*
0005A0    51004013 00000000 0080FF00 00260006    29240000 00F0F2F6 51004013 00000000   *.. ..................026.. .....*
0005C0    0080FF00 00270006 29250000 00F0F2F7    51004013 00000000 0080FF00 00280006   *.............027.. .............*
0005E0    29260000 00F0F2F8 51004013 00000C00    0080FF00 00290006 29270000 00F0F2F9   *.....028.. ..................029*
000600    51004013 00000000 0080FF00 002A0006    29280000 00F0F2C1 51004023 00000000   *.. ..................02A.. .....*
000620    0080FF00 002B0006 29290000 00F0F2C2    52004013 00000000 0C80FF00 002C0006   *.............02B.. .............*
000640    292A0000 00F0F2C3 54004013 00000C00    008CFF00 002D0006 292B0000 00F0F2C4   *.....02C.. ..................02D*
000660    54004013 00000000 0080FF00 002E0006    292C0000 00F0F2C5 54004013 00000000   *.. ..................02E.. .....*
000680    0080FF00 002F0006 292D0000 00F0F2C6    54004013 00000000 0080FF00 00300006   *.............02F.. .............*
0006A0    292E0000 00F0F3F0 5      3 00000C00    0080FF00 00310006 292F0000 00F0F3F1   *.....030.. ..................031*
0006C0    51004013 00000000          00320006    29300000 00F0F3F2 51004013 00000000   *.. ..................032.. .....*
0006E0    0080FF00 00330006                F3    51004013 00000000 0080FF00 00340006   *.............033.. .............*
000700    29320000                                 80FF00 00350006 2933                    ..034.. ..................035*
000720                                                0000 00F0F                                        .036..
```

Figure SADMP-4. IMDSADMP Low-Speed Dump Output Sample



```
IPL  0-7  00C0C0C0 00002A10 00000000 80002890    000029C8 0000000A C0002699 900024D8  *....................H...........Q
IPL 8-15  00000000 000027A0 00100000 00000C09    00000000 000026E8 80002584 400027FA  *......................Y.... ...

OTH  0-7  FFFFFF2E 0006D7F8 0002B0A4 0C000000    0002AA68 0CC2A618 0002AF58 0002BAE0  *......P8.........................
OTH 8-15  0002B088 00000000 0002B0B0 00000000    4007C4A2 0006D7BC 0001E924 400022C6  *................ .M...P..... ..F

000000    CC000330 00002258 400027FA 6000002B    0800008C 400000C1 C000C000 000000C0  *........ .......... ..........
000020    C0C00000 00000000 00000005 E000254A    00000000 0000000C 0000C000 000000C0  *................................
000040    C0C02A30 0C0C0C00 C00029C8 00000000    F8E10300 48100002 412CC0C0 50200C48  *..........H....8................
000060    98240008 9D001000 CC020000 00000C03    9D001000 4770007C 9103C044 4750007C  *...H............................
000080    31C0C0A6 40000CC5 08000080 40000001    05002240 400C05CC 06002240 000005C0  *.... ....... ......  ...... ....
0000A0    00C0C000 0000000C C0C00450 00D20650    445C00B8 47F0006C D2002C00 00D84040  *..............K........0..K....0
0000C0    C200C0C8 20000C48 02C5D5C4 40404040    40404040 4040404C 40404040 40404040  *...H.....END
0000E0    40404040 40404C4C 40404040 40404CC6    F0F8C1D7 D9F7F04C F2F24BF2 F140F1F4  *                   F08APR70 22.21 14
000100    61F0F161 F9F94040 40404040 4C4C4C40    00000C00 0000000C C000CCC0 00000CCC  *.01.99            ...............
000120    00C0C0C0 00000000 00000000 00000000    00000000 0000000C C000C000 00000000  *................................

001BC0    5CF0C004 05F0D203 F38E0004 900EF352    96020001 8200000C 50F0C008 05F09CCE  *.0...0K.3.....3..........0...0..
001BE0    F2EED203 F32A0C08 D2CBF32E 004041B0    001F9D00 B000478C F02C41B0 001F40F0  *2.K.3...K.3.. ..........0..... .
001C00    F0129D00 B0004770 F094D702 F3BEF3BE    4190F3F0 41D0F282 45E0F1CA 4710FC2C  *0........0.P.3.3...30..2...1...0.
001C20    91C1C044 4710FC2C D402F3BE F3BE4780    F09441C0 000341DC F3BD43AD 000091F0  *.......0.M.3.3...C........3......C
001C40    D0C04710 F0849640 D00095C1 D0004740    F17095C6 D0004720 F1704AA0 F2EA47F0  *....0.. ...A... 1..F....1...2..0
001C60    F0848CA0 000446C0 F05888B0 00144CB0    F2AC48B0 F2AC419C F3D8940F F171D7C3  *0........0..... .2...2...3Q..1.P.
001C80    F3C1F3C1 41D0F222 45E0F1CA 4710F170    D503F3C1 F3C5478C F170D503 F3C1F3C9  *3A3A..2...1...1.N.3A3E..1.N.3A3I
001CA0    478CF170 94BFF226 948FF1F3 45E0F1CA    45E0F33A 921FF24A 940FF1BF 4190F406  *..1...2...13..1...3...2...1...4.
001CC0    58A0F2E2 41D0F25A 45E0F1CA 91010C44    47E0F0FE 4190F3CC 47F0F170 5AA0F2E6  *..2S..2...1.......0...3..01...2W
001CE0    12AA47D0 F10ED207 0068F2AA 58C0AC00    50A0F2D6 1BCC09CA 88C0C003 42C0F2D6  *....1.K...2.......2C..........2C
001D00    50A0F262 9201F262 59A0F2CE 477CFCE6    58C0C010 41D0C02C 50D0C008 920700C8  *..2...2...2...0W................
001D20    92C6C00F 9206CC10 D206C011 F27B41D0    C00848B0 000245EC F1CA41D0 F26A48B0  *.........K...2...........1...2...
001D40    F2AC47F0 F0EA59A0 F2824740 F0FE47F0    F1AC41D0 F24A488C F2AC96F0 F1F345E0  *2..CC...2.. 0..01...2...2..013..
001D60    F1CA91C1 0044478C F19441D0 F25245E0    F1CA9D00 B000478C F1A84730 F19491C4  *1........1...2...1........1...1...
001D80    50F0C004 05F0D203 F49A0004 900EF45E    96020001 8200C00C 50F0C008 C5F09CCE  *.C...CK.4.....4..........0...0..
001DA0    F446C2C3 F4820C08 D20BF486 004041B0    001F9D00 B000478C FC2C41B0 001F40B0  *4.K.4...K.4.. ..........C..... .
001DC0    F0129D00 B0004770 F090D702 F3DBF3DB    4190F4AC 41D0F3FA 45E0F2B2 91010C44  *0........0.P.3.3...4...3...2.....
001DE0    4710F02C D402F3DB F3DB4780 F09041C0    000341DC F3DA43AC C00091F0 D0004710  *..C.M.3.3...0........3......C....
001E00    FC8C9640 D00095C1 D0004740 F29495C6    D0004720 F2944AAC F49E47F0 F0808CA0  *C.. ...A... 2..F....2...4..00...
001E20    00C446CC F05488B0 0C1440B0 F42448B0    F42441D0 F3E241AC F4464570 F15094CF  *....C...... .4...4...3S..4...1...
001E40    F2A74190 F4C2940F F2E5D203 F367F357    4570F150 D207F436 0028C207 F43E0068  *2...4B..2VK.3.3...1.K.4...K.4...
001E60    D2C7C068 F41258A0 F42A41C0 07FF43CC    000050C0 F42A19CA 41CCC800 4740FCD0  *K...4...4..............4........ 0.
001E80    50F0CC04 05F0D203 F38FC0C4 9C0EF352    96020001 8200000C 50F0C008 05F09CCE  *.0...CK.3.....3..........C...C..
001EA0    F2EED203 F32A0C08 D20BF32E 004041B0    00099D00 B000478C F02C41B0 001F40B0  *2.K.3...K.3.. ..........C..... .
001EC0    FC129DC0 B0004770 FC94D702 F3BEF3BE    419CF3F0 41D0F282 45E0F1CA 4710F02C  *0........C.P.3.3...3C..2...1...0.
001EE0    91C10044 4710FC2C D402F3BE F3BE4780    F09441C0 000341DC F3BD43AC 000091F0  *.......C.M.3.3...0........3......0
001F00    D0C04710 FC849640 D00095C1 D0004740    F17095C6 D000472C F1704AA0 F2EA47F0  *....C.. ...A... 1..F....1...2..C
001F20    FC848CA0 000446C0 F05888B0 00144CB0    F2AC48B0 F2AC419C F3D8940F F171D7C3  *C........C..... .2...2...3Q..1.P.
001F40    F3C1F3C1 41D0F222 45E0F1CA 471CF17C    D503F3C1 F3C5478C F170D503 F3C1F3C9  *3A3A..2...1...1.N.3A3E..1.N.3A3I
001F60    4780F170 94BFF226 948FF1F3 45E0F1CA    45E0F33A 921FF24A 940FF1BF 4190F4C6  *..1...2...13..1...3...2...1...4.
001F80    58A0F2E2 41D0F25A 45E0F1CA 91010C44    47E0F0FE 4190F3CC 47F0F170 5AA0F2E6  *..2S..2...1.......0...3..01...2W
001FA0    12AA47D0 F10ED207 C068F2AA 58CCAC00    50A0F2D6 1BCC09CA 88C0C003 42C0F2D6  *....1.K...2.......2D..........2C
001FC0    50A0F262 9201F262 59A0F2CE 477CFCE6    58C0C010 41D0C02C 50D0C008 92070CC8  *..2...2...2...0W................
001FE0    92C6C00F 9206C010 D2C6C011 F27B41D0    000848B0 000245EC F1CA41D0 F26A48B0  *.........K...2...........1...2...
002000    F2AC47F0 F0EA59A0 F2B24740 FCFE47FC    F1AC41DC F24A488C F2AC96F0 F1F345E0  *2..00...2.. C..C1...2...2..C13..
002020    F1CA9101 0044478C F19441D0 F25245F0    F1CA9D0C B000478C F1A84730 F19491C4  *1........1...2...1........1...1...
002040    CC44478C F19496F0 F171D20A F4169C00    48B0F012 41D0F292 45E0F1CA 4700FC2C  *....1..C1.K.4.....C...2...1...0.
002060    96C600C1 82000000 9DC0B000 4770F1CA    50D0CC48 9C00B00C 4740F1E6 9D00B0C0  *...............1.......... 1W....
002080    47B0F1DE 913F0045 4770F210 91020C44    07FE41C0 F1CA49BC F2ACC77C 41D0F232  *..1........2..........1...2.....2.
0020A0    49E0C04A 078CD202 F243C049 07FC49B0    F0124780 F1C2419C F3F847F0 F1700CC0  *......K.2........C...1B..3..C1...
0020C0    C7C0CCC0 00000C09 0200279F 20000C04    27000009 6000010C 1700C0C0 60000CC1  *................................
0020E0    C8C0C000 6C000C01 1F000000 40000001    0F000000 00C0C0C1 C10026B4 A0000CC4  *............. ..................
002100    010FF800 20000800 010026B4 A0000004    01002000 A00003CC 01001BC0 20000440  *..8..............................
```

Figure SADMP 5. IMDSADMP Low-Speed Dump Output Sample of a Model 65 Multiprocessing System

The particular version of the IMDSADMP dump program to be generated is specified by the operands entered in the IMDSADMP macro instruction. Depending upon the operands coded, the program will be generated as shown in Figure SADMP-6. The IMDSADMP macro instruction statement is coded as shown in Figure SADMP-7.

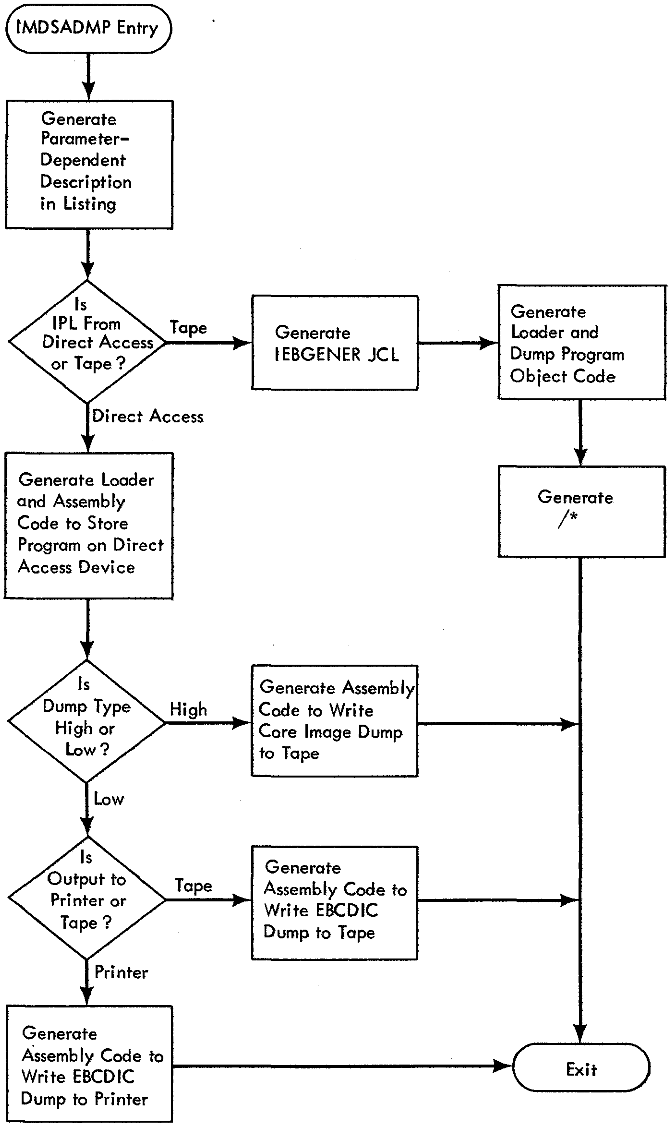
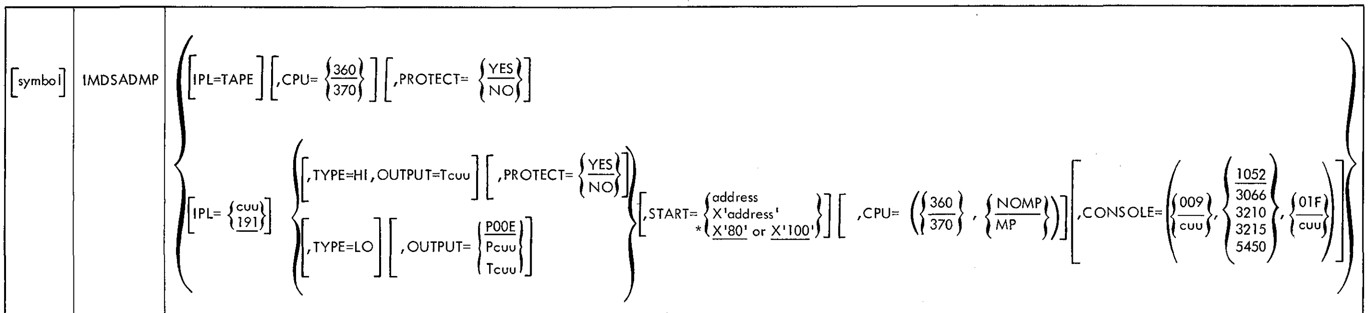

IMDSADMP Entry
Generate Parameter-Dependent Description in Listing
Is IPL From Direct Access or Tape ?
Tape
Generate IEBGENER JCL
Generate Loader and Dump Program Object Code
Generate /*
Direct Access
Generate Loader and Assembly Code to Store Program on Direct Access Device
Is Dump Type High or Low ?
High
Generate Assembly Code to Write Core Image Dump to Tape
Low
Is Output to Printer or Tape ?
Tape
Generate Assembly Code to Write EBCDIC Dump to Tape
Printer
Generate Assembly Code to Write EBCDIC Dump to Printer
Exit

Figure SADMP-6. IMDSADMP Parameter-Dependent Program Generation


[symbol]
IMDSADMP
[IPL=TAPE]
[,CPU= 360 370]
[,PROTECT= YES NO]
[IPL= cuu 191]
[,TYPE=HI,OUTPUT=Tcuu]
[,PROTECT= YES NO]
[,TYPE=LO]
[,OUTPUT= P00E Pcuu Tcuu]
[,START= address X'address' *X'80' or X'100']
[,CPU= ((360 370), (NOMP MP))]
[,CONSOLE= ((009 cuu), (1052 3066 3210 3215 5450), (01F cuu))]

*Note: The default value of the START= operand is dependent upon the value specified for the CPU= operand.

X'80' is the default if CPU=360 is specified or if CPU= is omitted.

X'100' is the default if CPU=370 is specified.

Figure SADMP-7. The IMDSADMP Macro Instruction Statement

symbol

any symbol may be associated with the IMDSADMP macro instruction. However, this symbol should not be referenced by any other assembler input statement, such as the END statement.

IMDSADMP

The name of the macro instruction is coded as shown.

IPL=

describes the device upon which the dump program resides. As such, it dictates the operation of the initialization step of the dump creation procedure. The allowable options for the IPL= operand and their meanings are:

TAPE

specifies that the dump module is stored on a tape device. If this option is coded, all keywords except CPU= and PROTECT= are ignored. TYPE=HI is assumed. When the dump is executed, output is written to the same tape device on which the dump program resides, immediately following the IPL and dump program records.

cuu

specifies a direct access device address where "c" indicates the channel address, and "uu" indicates the device address. The direct access volume that is to contain the dump program must be mounted on this device during the initialization phase. After initialization, the volume may be moved to any other direct access device.

If the IPL= operand is not specified, a default value of direct access device 191 is assumed.

TYPE=

specifies the version of the dump program to be generated for direct access residence. The allowable options for this operand and their meanings are:

HI

specifies the high-speed version of the dump program that will write unformatted core-image output to a tape volume in 2052-byte blocks. Note: The resultant output tape must be used as input to the Print Dump service aid (IMDPRDMP) to format and print the dump information.) If this option is coded and the dump program resides on a direct access device, OUTPUT=Tcuu is required. (See the options for the OUTPUT=keyword operand.)

LO

specifies the low-speed version of the dump program that will produce formatted EBCDIC output to either a tape device or the printer. If no options are entered for the TYPE= operand, TYPE=LO is assumed.

OUTPUT=

specifies the output device type. It also specifies the default output device address to which the dump is to be written if the operator chooses to use the default address rather than specify an address through a console reply in response to a message. The allowable options for this operand and their meanings are:

Tcuu

specifies the channel and unit address of a tape output device where "T" indicates tape, "c" indicates the channel address, and "uu" indicates the unit address. This is the only valid option for this operand when TYPE=HI is specified. If TYPE=HI is specified and this option is not specified or the entire OUTPUT= operand is omitted, TYPE=HI will be changed to TYPE=LO and the default value of P00E will be used as the OUTPUT= operand.

Pcuu

specifies the channel and unit address of a printer where "P" indicates printer output, "c" indicates the channel address, and "uu" indicates the unit address.

If the OUTPUT= operand is not specified and TYPE=LO is coded, a default value of printer 00E (P00E) is assumed.

PROTECT=

applicable only if TYPE=HI is selected. This operand specifies whether or not the storage protection feature is available on the CPU. The allowable options for this operand and their meanings are:

YES

This value must not be coded if the storage protection feature is not available on the CPU on which the dump program is intended to be executed, as the dump program will not work. If it is coded or assumed, it specifies that the feature is implemented. The storage protection key field in the output of the high-speed version of the dump program will contain the storage protection key associated with the block of storage being dumped (see Figure SADMP-3, format 3).

NO

If the storage protection feature is not available on the CPU, or if it is not to be used, the NO value must be coded for the PROTECT= operand. If NO is coded, the storage protection key field in the output of the high-speed version of the dump program will contain zero (see Figure SADMP-3, format 3).

If this operand is not coded, PROTECT=YES is assumed.

START=

specifies the storage location into which the CCW's for loading the direct access resident dump program will be read. 43 bytes of storage are required for the load CCW's and, with the 24 bytes of storage starting at location 0 that are required for the IPL procedure, represent the only storage destroyed by execution of the dump program.

The START parameter is valid for both high and low speed options of the direct access resident dump program. Allowable values for this operand and their meanings are:

address

specifies the starting address of the CCW loading area expressed as a decimal number. The storage address must be greater than or equal to 128 and be aligned on a doubleword boundary. If the value specified is less than 128, it is ignored and 128 is used; if the value is not a multiple of eight, the next higher multiple of eight is used. The maximum allowable address must be at least 48 bytes less than the maximum main storage address of the CPU on which the dump program is to be executed.

X'address'

specifies the starting address of the CCW loading area expressed as a hexadecimal number. The address specified in this operand must be X'80' or greater, and be aligned on a doubleword boundary. If the value specified is less than X'80', it is ignored and X'80' is used; if the value is not a multiple of eight, the next higher multiple of eight is used. The maximum allowable address must be at least X'30' bytes less than the maximum main storage address of the CPU on which the dump program is to be executed.

The default value for the START parameter is dependent on the values of the CPU parameter. If CPU=360 is specified, or if the CPU parameter is omitted, the default value used for the START parameter will be X'80'. If CPU=370 is specified, a default of X'100' will be used for the START parameter. Adjusting the START value in this way is done to ensure that the storage overlayed by the dump program will be contained in the log-out area of the CPU on which the dump program is to be executed.

CPU=

defines the IBM computer system that IMDSADMP will dump. There are two possible subparameters:

1. The system subparameter -- 360 or 370 for the IBM System/360 and IBM System/370 respectively.

2. The multiprocessing subparameter -- NOMP for non-multiprocessing systems and MP for multiprocessing systems.

Implicit in the system subparameter is the location of the log-out area (sometimes called the diagnostic scan-out area). For the IBM System/360, the log-out area is located at X'80'; for System/370, it it located at X'100'.

When IMDSADMP is loaded from magnetic tape (IPL=TAPE), the IPL procedure overlays the first 24 bytes of main storage and the entire 256 bytes of log-out area.

When IMDSADMP is loaded from a direct access device (IPL=cuu), the log-out area is used as the default value for the START parameter (refer to START).

One version of IMDSADMP is used for a non-multiprocessing system, and another version is used for a multiprocessing system; CPU=(,NOMP) and CPU=(,MP) specify the different systems. When applied to a multiprocessing system, IMDSADMP must be resident on a direct access device; in that case, therefore, define IPL=cuu. At the present time, the only multiprocessing system that IMDSADMP can be used with is the IBM System/360 Model 65 Multiprocessing system; for this system define CPU=(360,MP).

CONSOLE=

specifies the address and type of the console through which commands will be entered. Valid values and their meanings are:

$\begin{Bmatrix}\text{009}\\ \text{cuu}\end{Bmatrix}$

The console address. If you omit the CONSOLE= parameter, IMDSADMP assumes a default address of 009.

$\begin{Bmatrix}\text{1052}\\ \text{3066}\\ \text{3210}\\ \text{3215}\\ \text{5450}\end{Bmatrix}$

The console device type. If you omit the CONSOLE=parameter, assumes 1052 as the default device type. (Model 65 Multiprocessing only).

$\begin{Bmatrix}\text{01F}\\ \text{cuu}\end{Bmatrix}$

The address of the second console in a multiprocessing system. This value is not valid for non-multiprocessing systems. If this value is omitted, IMDSADMP assumes a default address of 01F for the second console.

Retrieving and Creating the Dump Program

The dump program is created in two steps: a specification step and an initialization step. The specification step involves the creation of a dump initialization deck that will be used as input to the initialization step. These two steps are discussed below:

The Specification Step

Before commencing operation on the specification step, the user must have made two decisions. First, he must have decided which version and options of the dump program he wishes to be in effect, as detailed in the previous discussion. Second, he must also have decided whether he wants the macro definition to be in a library (and, if so, which one) or in card image form.

Before the IMDSADMP macro definition can be assembled into a stand-alone program, the macro definition statements must be available on a media from which they can be assembled. Figure SADMP-8 shows five media from which IMDSADMP can be assembled.

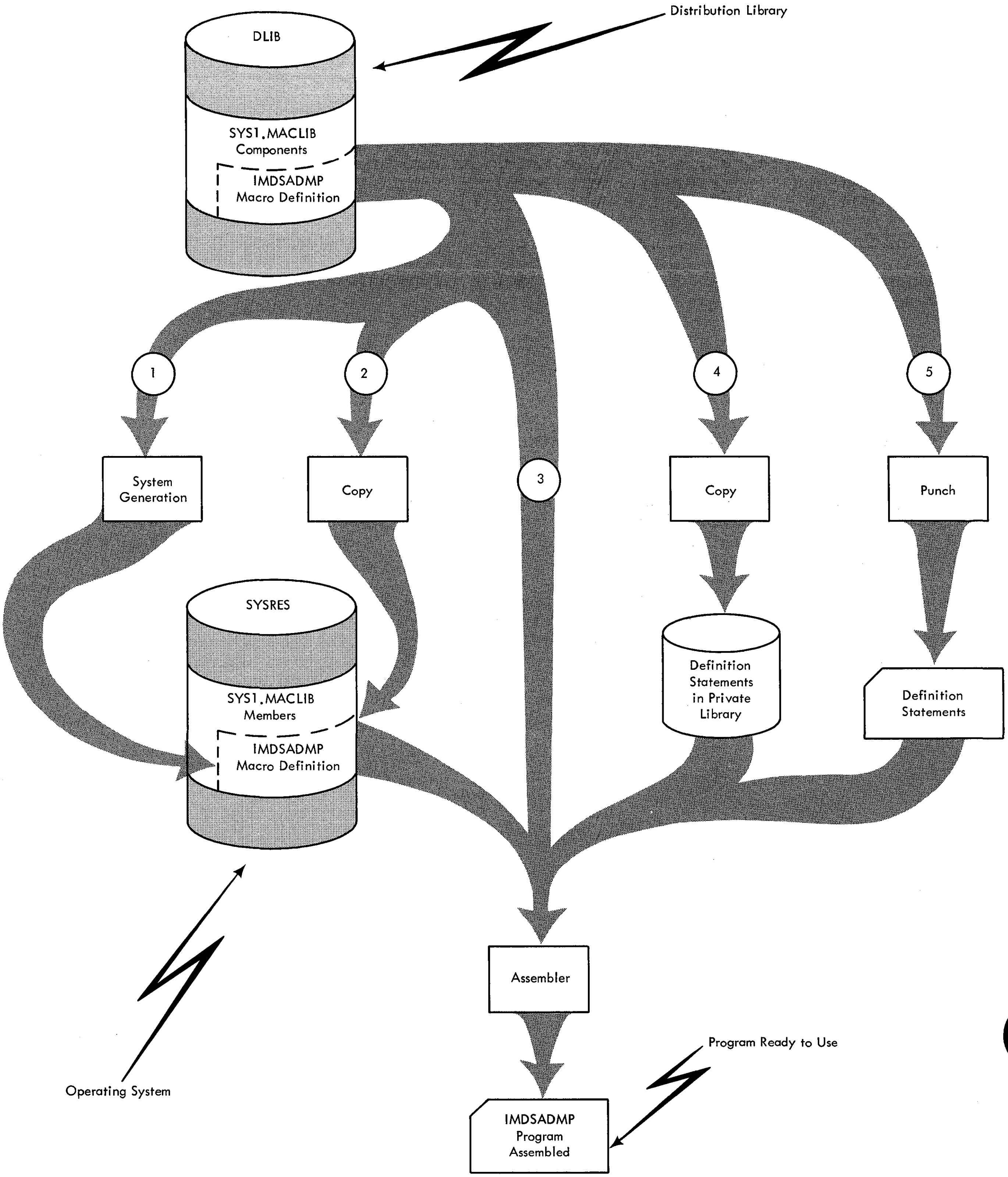


Figure SADMP-8. Availability of IMDSADMP Macro Definition Statements

If the MACLIB macro instruction was specified During system generation, the macro definition for IMDSADMP is transferred from the SYS1.MACLIB component data set in the distribution library to the SYS1.MACLIB system data set. The IMDSADMP stand-alone program can then be assembled in the same manner as any other program in macro definition form. If MACLIB was not specified, use one of the following techniques to obtain the IMDSADMP macro definition:

Distribution Library as a Private Library: The distribution library can be used as a private library for the assembling of the IMDSADMP stand-alone program, see Figure SADMP-9. This example assumes that the distribution libraries are cataloged; if not, add the UNIT and VOL=SER operands to the ASM.SYSLIB data definition statement.

```
//ASMSAD       JOB       MSGLEVEL=(1,1)
//             EXEC      ASMFC
//ASM.SYSLIB   DD        DSN=SYS1.MACLIB,DISP=OLD
//ASM.SYSIN    DD        *
               .
               .    (include the IMDSADMP macro instruction here)
               .
               END
/*
```

Figure SADMP-9. An Example of IMDSADMP JCL Statements for Designating SYSLIB

Copying to a Private Library: The IMDSADMP member of the SYS1.MACLIB component of the distribution library can be copied to a user defined library. The IMDSADMP stand-alone program can then be assembled from the user defined library.

Punching the Definition Statements: The IMDSADMP member of the SYS1.MACLIB component of the distribution library can be punched into cards using a utility program. With the macro definition statements on cards, the IMDSADMP stand-alone program can be assembled using these cards as input. Figure SADMP-10 shows the specification step when the macro definition statements are in punched card form.

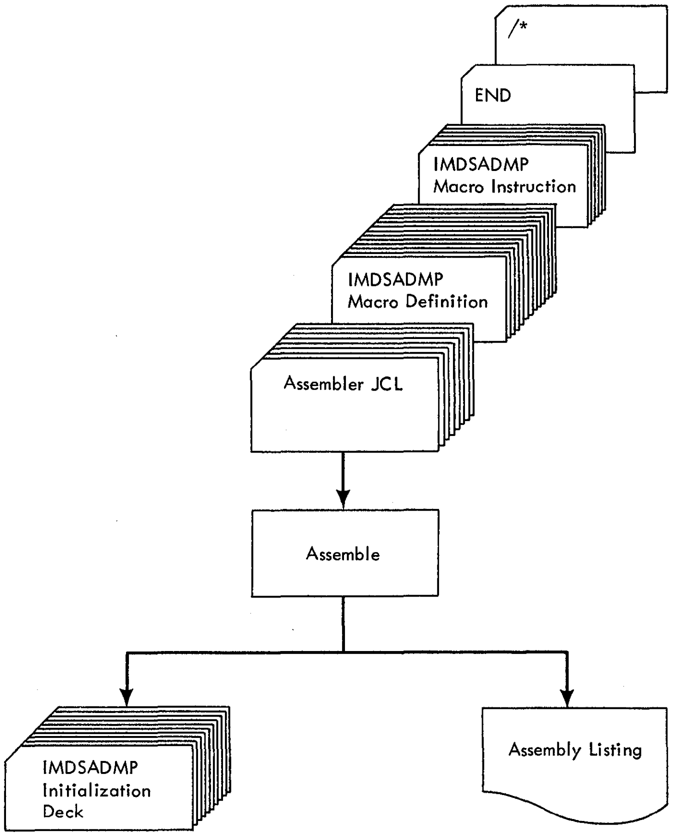

/*
END
IMDSADMP Macro Instruction
IMDSADMP Macro Definition
Assembler JCL
Assemble
IMDSADMP Initialization Deck
Assembly Listing

Figure SADMP-10. Example of the IMDSADMP Specification Step

Prior to executing the specification step, the user should ensure that he has all the required elements:

- The Assembler job control cards.

- The IMDSADMP macro definition, in either card image form or in a library as discussed above.

- The IMDSADMP macro instruction containing the operands that defines the version of the dump program that the user wishes to generate, The macro instruction may be included only once per assembly.

The specification step, then, is an assembly that creates a dump initialization deck, to be used as input to the initialization step. This dump initialization deck consists of:

- Code that allows the remainder of the dump initialization deck to be stored on the selected tape or direct access device.

- IPL text necessary to make the dump program loadable for execution.

The Initialization Step

The specification step provides input to the initialization step. The output of the initialization step is an executable dump program, stored on an I/O device from which it is loaded by the IPL procedure into main storage for execution. Initialization may be performed in one of two ways, depending upon the device type specified in the IMDSADMP macro instruction IPL= operand. The different initialization step procedures are illustrated by Figure SADMP-11.

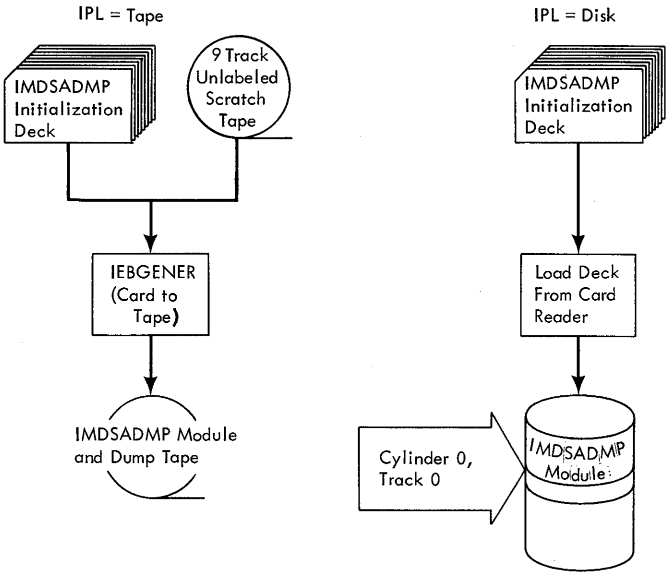

IPL = Tape
IMDSADMP Initialization Deck
9 Track Unlabeled Scratch Tape
IEBGENER (Card to Tape)
IMDSADMP Module and Dump Tape
IPL = Disk
IMDSADMP Initialization Deck
Load Deck From Card Reader
Cylinder 0, Track 0
IMDSADMP Module

Figure SADMP-11. IMDSADMP Initialization Procedures

Tape Initialization (High-Speed Only)

If the user has specified that the high-speed version of the dump program is to reside on tape, the specification step will have provided three types of statements to the initialization step: job control statements, IPL text and the dump program object module. The JCL statements invoke and control the operation of the IEBGENER utility program (as discussed in the publication IBM System/360 Operating System; Utilities, GC28-6586) that copies the remainder of the dump initialization deck to the specified tape volume. The IPL text allows the dump program to be loaded from the tape volume into main storage for execution. The dump program object module consists of the actual machine instructions that perform the desired dump function. The initialization process for a dump program that is loadable from a tape volume is under control of the operating system, and is performed in the same manner as for any other job. During execution of the job, the operator will receive message IEF233A from the job scheduler, asking for tape serial number "DUMP". At this time, a non-labeled scratch tape is mounted to be initialized.

Note: The output of IEBGENER describes the tape to be initialized as is shown in Figure SADMP-12. If a 7-track unit is to be used to initialize a dump tape, the UNIT= parameter must be changed to describe a 7-track tape device; that is, UNIT=2400-2. If the data conversion feature is not present on the 7-track unit, the TRTCH=C parameter should be removed.

```
//SYSUT2       DD         VOL=SER=DUMP,LABEL=(,NL),DISP=(NEW,KEEP),
//        UNIT=2400,DCB=(BLKSIZE=80,LRECL=80,RECFM=F,DEN=2,TRTCH=C)
```

Figure SADMP-12. An Example of IEBGENER DD Statements for Dump Tape Initialization

The JCL statements that head the dump initialization deck invoke the IEBGENER utility program, which in turn copies the remainder of the dump initialization deck onto the selected tape volume. When a user wishes to obtain a high-speed dump of main storage, the necessary program and storage space are available to him on the volume he has initialized.

Direct Access Initialization

A direct access device must be used to store the low-speed dump program, whereas the high-speed version may be stored on either direct access or tape devices. When the user specifies a direct access device, the specification step will have passed a loader and initialization program, IPL text, and the dump program object module to the initialization step. The initialization program transfers the IPL and program statements to cylinder 0, track 0, of the volume on the specified direct access device.

The volume on which the assembled IMDSADMP service aid resides must have a standard 80-character label located at cylinder 0, track 0. There may be up to seven 80-character labels on that track. The number of labels depends upon the IMDSADMP options selected and the track capacity of the device; see Figure SADMP-13.

Dump Residence Device	IMDSADMP Options	Maximum Number of User Labels
2301	All Options	7
2303	Multiprocessing Low-Speed Tape Output	6
	All Other Options	7
2305	All Options	7
2311	High-Speed Tape Output	7
	Low-Speed Tape Output	5
	Low-Speed Printer Output	6
	Multiprocessing High-Speed Tape Output	6
	Multiprocessing Low-Speed Tape Output	0
	Multiprocessing Low-Speed Printer Output	2
2314 2319	All Options	7
3330	All Options	7

Figure SADMP-13. Maximum Number of User Labels Depending on Device and Options Selected

The IPL text is then used to load the dump program from the direct access device into main storage for execution. The dump program object module consists of actual machine instructions that perform the specified dump function. Initialization of a dump program that is loadable from a direct access volume is a stand-alone process and proceeds as follows:

- Ready the desired direct access volume on the device specified by the IPL= operand of the IMDSADMP macro instruction.

- Place the dump initialization deck in the card reader.

- Set the Load Unit dials to the address of the card reader.

- Press the Load key on the operator's console.

When the initialization program has successfully transferred the IPL text and the dump program module to the direct access volume, a completion code of X'01' appears in the instruction address register (IAR). After the initialization step has been completed, the direct access volume containing the dump program may be moved to any device. The direct access volume may be repeatedly dumped and restored without reinitialization of the dump program. If the user keeps the dump program direct access volume permanently mounted, the dump program is immediately available when the user wishes to obtain a stand-alone dump of main storage.

If the direct access initialization process is not successful, an error code is set in the IAR. This code indicates the cause of the initialization failure:

X'04'

The VTOC of the volume being initialized begins on cylinder 0, track 0; hence the record containing the dump program cannot be written on this track. Such a direct access volume cannot contain the IMDSADMP program.

X'08'

The unused space on cylinder 0, track 0 is not sufficient to hold the dump program. Only standard IPL records, the 80-character volume label, and one to seven user labels can reside on cylinder 0, track 0.

X'0F0F0F'

A permanent I/O error (the condition persisted after 16 retries) occurred on the direct access device being initialized. This condition is usually caused by cylinder 0, track 0 being defective. A direct access volume with a defective cylinder 0, track 0 is not suitable for use as an IPL volume. The volume should be analyzed, using either utility program IEHDASDR or IBCDASDI, and the initialization process repeated.

Executing the Dump Program

The operating procedures for the tape resident version of the dump program vary slightly from those of the direct access resident version. Console operation procedures for the execution of the tape resident version of the dump program are as follows:

- Ensure that the initialized tape volume containing the dump program has the write ring in place.
- Mount the initialized dump tape volume (discussed under Creating the Dump) on an appropriate tape device.
- Set the Load Unit dials to the address of the tape device containing the initialized dump tape volume.
- Press the Load key on the operator's console.

The contents of main storage are written to the same tape volume that contains the dump program. The dump information is written to the tape volume immediately behind the dump program records (see Figure SADMP-3). Successful completion of the dump is indicated by the appearance of X'01' in the instruction address register. At this point, the user must perform the OS/360 IPL procedure in order to restart the operating system. The tape containing the dump information must then be used as input to the Print Dump service aid to format and print the information. After the information contained on the tape volume has been printed, the same initialized volume may be used to perform another dump. The IPL text and the program module heading the initialized tape volume are not destroyed in the dump process.

A direct access resident dump program is executed as follows:

- Mount the initialized direct access volume (discussed under "Creating the Dump") on any suitable direct access device and bring the device to ready status. (Usually, the dump program would be stored on a permanently mounted direct access volume, so that it would always be available.)
- Set the Load Unit dials to the address of the direct access device containing the initialized volume.
- Press the Load key on the operator's console.
- Message IMD001A will be issued to the console at the address specified by the CONSOLE= parameter. This message asks for the address of the device to which the dump output is to be written. When message IMD001A is issued, the operator should ready the desired output device and enter the address of that device or signal end-of-block if the default output device is to be used. If the operator responds with end-of-block, or if an error occurs during an I/O console operation, the output device specified by the OUTPUT= parameter when the IMDSADMP macro was assembled will be used. If the OUTPUT= parameter had not been specified, the default value of P00E will be used.

  The device address specified in response to message IMD001A must be that of a device whose type agrees with the device type specified by message IMD001A. If Tuu was specified for the OUTPUT= parameter as the device type, message IMD001A TAPE= will be issued,

SADMP

indicating that a tape device is desired. If Puu was specified for the OUTPUT= parameter as the device type, or if the OUTPUT= parameter was allowed to default, the message IMD001A PTR= will be issued, indicating that a printer device is desired. When output is to a tape device, the volume mounted on the specified device is checked for standard labels before the dump is written. Standard labels are checked by comparing the first four bytes of the first record for VOL1. (The VOL1 identifier is checked against both EBCDIC and ASCII encodings.) If such a label is found, or if an I/O error occurs during the label checking procedure, the volume is unloaded and the message IMD002I LBL ERR is issued. Message IMD001A is reissued and the operator must ready and specify the output device again. The operator can mount a non-labeled scratch tape and enter the device address again, or he can enter the address of a different device on which a non-labeled tape has been previously mounted.

The contents of main storage are written to the specified output device. Successful completion of the dump is indicated by the appearance of message IMD005I. At this point, the user must perform the IPL procedure in order to restart the operating system. If the dump information is written to tape, it must be printed by a subsequent program. In the case of the low-speed version of the dump program, the tape output may be printed by the IEBGENER utility program, as discussed in IBM System/360 Operating System: Utilities, GC28-6586, or by IMDPRDMP. Tape output produced by the high-speed version of the dump program must be formatted and printed by IMDPRDMP.

Note 1: If the printer runs out of paper during the execution of the dump program, insert more paper and start the printer. IMDSADMP will continue normally.

Note 2: Neither version of the dump program issues a mode set command to the tape output device. If output is to a 7-track tape, additional JCL parameters are required on the input DD statement for programs which read the dump tape. When the dump has been written to a 7-track tape, the following must be coded as subparameters of the DCB parameter: DEN=2,TRTCH=C. If the data conversion feature is not included on the 7-track device, the TRTCH= keyword myust be omitted.

Operational Considerations

Following are points to which careful consideration should be given when using the stand-alone dump service aid (IMDSADMP):

- If IMDSADMP output is to tape, the tape volume mounted must be non-labeled. If the output volume has standard labels, or if an I/O error occurs during this checking procedure, the tape volume is unloaded and message IMD002I is issued. A non-labeled scratch tape (e.g., one with a tapemark as the first record) must then be mounted and IMDSADMP reloaded.

- Non-labeled scratch tapes on 7-track devices may not be accepted by IMDSADMP. The volume on a 7-track unit will be unloaded unless it is one of the following types:

  1. A scratch tape with a tapemark as the first record, or

  2. A non-labeled tape with data recorded in the mode: 800 BPI, odd parity, translator off. For example, a dump tape previously produced by IMDSADMP.

- If the user specifies the disk resident version of IMDSADMP he must consider the direct access space requirements. The IPL text, dump program records, and work record are contained on cylinder 0, track 0 of the volume on which the dump program resides (see Figure SADMP-2). This direct access volume must have the standard 80-character volume label, and may have one to seven 80-character user labels, on cylinder 0, track 0. The number of user labels possible is dependent upon the dump program output option specified by the user.

Option Specified	Number of User Labels Possible
High-Speed	1 to 7
Low-Speed to Printer	1 to 6
Low-Speed to Tape	1 to 5

- Depending on the track capacity of the IMDSADMP resident direct access device, the user may need to limit the number of user labels written on that track; see Figure SADMP-10.1.

- When specifying the IMDSADMP macro instruction operands, PROTECT=YES must not be coded if the storage protect feature is not implemented, as the dump program will not execute.

- If the dump program resides on a direct access volume, the IPL text and dump program records are contained on cylinder 0, track 0, of that volume. The resident volume must have a standard 80-character label on cylinder 0, track 0. With the IBM System/360 Disk Operating System, the volume table of contents for that volume must begin at some location other than cylinder 0, track 0.

- If IPL=cuu or IPL=191 is specified or implied, the direct access volume that contains the dump program must be mounted on the specified direct access device during the initialization step. After initialization the volume may be moved to any other applicable device.

- If the IMDSADMP macro definition resides in either the component library or a private library, the user should not attempt to concatenate either library to SYS1.MACLIB unless the attributes and device type are identical.

- Neither version of the dump program issues a mode set command. Therefore, output to a 7-track tape may produce a volume that cannot be read by other programs. If output is to a 7-track tape, additional JCL parameters are required on the input DD statement for programs which read the dump tape. When the dump has been written to a 7-track tape, the following must be coded as subparameters of the DCB parameter: DEN=2,TRTCH=C. If the data conversion feature is not included on the 7-track device, the TRTCH= keyword must be omitted.

- If the user specifies the START= parameter for the disk resident version of IDSADMP, the address he specifies must be equal to or greater than 128 or X'80'. The address specified must also be at least 48 bytes (X'30') less than the maximum main storage address of the CPU on which the dump program is to be executed.

- The low-speed version of the dump program must reside on a direct access device. The high-speed version may reside on either a tape or direct access device.

- Initialization of a disk-resident dump program must be performed on a System/360 model 40, or higher.

- The output tape produced by the high-speed version of the dump program must be printed by IMDPRDMP.

- The output tape produced by the low-speed version of the dump program may be printed by the IEBGENER utility program or IMDPRDMP.

- Error recovery during dump execution: If output is to tape, a failing I/O operation is retried indefinitely. Before the operation is retried, the tape volume is backspaced and a record gap is erased.

- Occurrence of a Unit Check or Unit Exception condition on the printer as the result of an I/O operation will cause the WRITE Operation to be retried until the condition is cleared. If the Unit Check condition exists when the I/O operation is initiated, the program will enter a two instruction loop. When the Unit Check condition is cleared (that is, when the device is made ready), the dump operation will continue.

- Occurrence of the Unit Check condition on the first I/O operation to the console causes the dump to be written to the device specified by OUTPUT=.

- IMDSADMP supports only the following devices:

  1. Printer - 1403, 3211

  2. TAPE - 2400 series, 3400 series

  3. DASD - 2311,2312,2313,2314,2318,

     2319,2301,2303,2305,3330

  4. Card reader - 2540

  5. Console - 1052,3066,3210,3215,5450

Note: IMDSADMP uses data chaining when writing a high-speed dump. Therefore, when running IMDSADMP on a System/360 Model 30, do not direct output to a tape device with a high data transfer rate.

- Location X'10' is used by the system to locate the CVT. The IPL procedure used to load the dump program when using IMDSADMP destroys this location. Therefore, if there is reason to believe that this location has been overlaid during processing by the system, its value must be manually displayed and recorded prior to taking the stand-alone dump.

When using IMDSADMP on a multiprocessing (MP) system, the following additional points should be considered:

- IMDSADMP should be permanently resident on a shared volume to permit IMDSADMP to be loaded by either CPU.

- IMDSADMP must be loaded by the CPU whose prefix switch is set to disable. If the CPU that is not prefixing has had a hardware malfunction, set the prefix switch on the other CPU to disable and load IMDSADMP from that CPU.

- For IMDSADMP to dump the registers of both CPUs, both CPUs must be in multiprocessing mode when IMDSADMP is executed.

Error Conditions

This section describes various error conditions which can occur during execution of IMDSADMP. During such execution, it is imperative that the data in core to be dumped remain unaltered. Error recovery is consequently limited to providing attempted retries of I/O operations and presenting an indication of the error. If an error occurs, the system operator should note the error indication (IAR content, wait state, loop, load light on, or incomplete output), and execute the program again. If the problem recurs, call IBM for programming support.

Error Handling

All operations of IMDSADMP are executed with machine check disable. A machine check during IMDSADMP execution will remain pending so that the dumping function can continue to completion. When dump execution is complete, a wait PSW is loaded by IMDSADMP to enable machine checks. Any pending machine check interrupts will be presented at this time.

All I/O operations in IMDSADMP are done with the system mask disabled for I/O interrupts. I/O status is received by IMDSADMP through use of the TIO instruction.

Errors During Initialization of Direct Access Resident Version

1. Loading of Initialization Program

   a. If, during IPL, an I/O error occurs on the card reader, the CPU will enter a wait state with the console load light on.

   b. Loading of the initialization program is done by IMDSADMP, executing within the CPU. Each I/O operation to the card reader is checked for unit check, unit exception or any condition indicated in the second status byte in the CSW. If any of these conditions is present, a one-instruction loop is entered.

2. Once the initialization program is loaded, I/O errors can occur only on the direct access device being initialized. I/O errors on this device are indicated by light settings in the IAR. Possible indications and remedial actions are described in the initialization discussion in this section.

3. A program interrupt during initialization will result in the program entering a WAIT condition with X'03' set in the IAR.

Errors During Dump Execution

If an I/O error should occur on the load device during loading of the dump program from tape or direct access, the CPU will enter a wait state with console load light on.

The program check new PSW is modified after the storage location containing that PSW is written. A program interrupt before this PSW has been initialized cannot be indicated, since to do so would overlay the data to be dumped. Therefore, the result of a program check at this time is unpredictable. If a program check occurs after this PSW has been modified, the dump will be terminated normally with message IMD005I, but output will be incomplete.

1. Direct Access Resident Dump Program

IMDSADMP tests console availability by issuing a TIO instruction to the console device. If the resulting condition code is zero, the console is assumed to be operational. All other condition codes indicate to IMDSADMP that the console is not operational; therefore, the dump is written to the output device specified in the OUTPUT parameter of the macro instruction.

Before each I/O operation, a TIO instruction is issued to the device to be used. If the device is not available the TIO instruction is repeated until the device is ready. When an I/O operation completes, the CSW is checked for the following conditions:

- Channel program check.
- Protection violation.
- Channel data check.
- Interface control check.
- Chaining check.

If any of these conditions occurs, it is indicated by message IMD003I CHAN ERR and execution is terminated. If unit check is indicated in the CSW and the I/O operation was not being performed on the dump output device, the operation is retried until the unit check condition is cleared. Unit exception conditions on devices other than the output device are ignored.

a. Printer Output

Condition code zero is the only status accepted on the SIO instruction. The SIO is repeated until the condition code becomes zero (that is, when the device is made ready). If an I/O operation completes with either unit check or unit exception, then the write operation (not the spacing command) is retried until the condition is cleared. The dump then continues.

b. Tape Output

Condition Code 1 following the SIO instruction is interpreted as an error condition and the error recovery procedure for that device is entered. If unit exception is indicated when an operation to tape completes, message IMD004I EOR is issued and execution is terminated. A unit check condition will initiate a recovery channel program which will be retried repeatedly until the unit check condition is cleared.

2. Tape Resident Dump Program

I/O operation are issued only to the tape device from which the dump program was loaded. If condition codes 2 or 3 occur as a result of the SIO instruction, the SIO instruction is repeated until the condition is cleared. A unit exception condition is ignored. If a unit check condition occurs, a recovery channel program will be initiated and repeated until the condition is cleared. A condition code of 1 occurring as a result of an instruction also causes the recovery channel program to be executed.

Macro Expansion Messages

During the expansion of the IMDSADMP macro definition, the operands of the IMDSADMP macro instruction statement are examined for validity. If an invalid operand value is detected, a diagnostic error message is issued, indicating the error and showing what corrective action was taken, or what assumption has been made. The message texts, their severity codes, and their meanings are shown in the table below:

ALTERNATE CONSOLE AVAILABLE ONLY FOR CPU=(360,MP). ALTERNATE IGNORED

Explanation: The user specified two consoles in the CONSOLE= parameter, but did not specify CPU=(,MP). A second console can be specified only for a multiprocessing system (MP). The assembly continues, and the alternate console definition is ignored.

Severity Code: 4.

CONSOLE DEVICE TYPE XXXX NOT SUPPORTED, 1052 CONSOLE ASSUMED

Explanation: IMDSADMP does not support the device specified in the CONSOLE= parameter. A 1052 console is assumed.

Severity code: 4

CPU VALUE ERROR, S/360 ASSUMED

Explanation: The value specified by the user for the CPU= parameter was not one of the valid values, 360 or 370. The default value of 360 is assumed.

Severity Code: 4.

CPU=xxx INVALID, CPU=(yyy,NOMP)IS ASSUMED

Explanation: The second subparameter of the CPU parameter is invalid. The first parameter has already been tested. The second subprarameter must be either MP or NOMP. xxx is the entry made by the user; yyy is the CPU type, either 360 or 370, entered as the first parameter of CPU=. The assembly continues, and NOMP is assumed.

Severity Code: 4.

CPU=xxx INVALID, CPU=(360,MP) IS ASSUMED

Explanation: The user has specified CPU=(360,MP), which is invalid. The MP option of IMDSADMP is available only for the M65MP system, for which the CPU= parameter must specify CPU=(360,MP). IMDSADMP assembly continues with CPU=(360,MP) assumed.

Severity Code: 4.

HIGHSPEED MEMORY DUMP REQUIRES TAPE OUTPUT, TYPE = HI IGNORED

Explanation: The user has attempted to generate the high-speed version of the dump program with the output directed to a printer. The output must be assigned to a tape device. The TYPE=HI operand has been ignored and output has been assigned to the printer. Check the specifications of the OUTPUT= operand.

Severity Code: 4.

IMDSADMP MACRO ALLOWED ONLY ONCE PER ASSEMBLY

Explanation: The use has attempted to issue the IMDSADMP macro instruction more than once within this assembly.

Severity Code: 8.

INVALID CHARACTER IN DECIMAL PARAMETER, START= xxx INVALID, X '80' USED

Explanation: The value specified for the START= operatind was not decimal. The value was coded as xxx. Review the description of the START= operand. The two parts of the message in this discussion may be issued independently.

Severity Code: 4.

MP OPTION NOT AVAILABLE FOR IPL=TAPE. IPL=TAPE IS ASSUMED

Explanation: Parameter conflict. The user specified both the MP and IPL=TAPE parameters. MP requires the direct access resident option. The assembly continues, and the MP option is ignored.

Severity Code: 4.

OUTPUT = xyyy IS INVALID. OUTPUT = P00E USED

Explanation: The channel and unit address (yyy) specified for the output device is invalid. (x indicates the device type, 'P' for a printer and 'T' for a tape device.) The printer (P00E) will be used for output if possible.

Severity Code: 4.

x DENOTES INVALID OUTPUT DEVICE, A PRINTER IS ASSUMED

Explanation: The device type indicator of the OUTPUT= operand was specified as other than 'P' (for a printer) or 'T' (for a tape device); x is the character that was coded. P00E is assumed.

Severity Code: 4.

Chapter 11: IMASPZAP

Verifies and/or replaces instructions and/or data in a load module.

SPZAP

Contents

INTRODUCTION 279
Capabilities of SPZAP 279
Monitoring the Use of SPZAP 279

DATA MODIFICATION AND INSPECTION 281
Inspecting and Modifying a Load Module 281
Accessing a Load Module 281
Inspecting and Modifying a Data Record 283
Accessing a Data Record 283

DUMPING DATA 284

UPDATING SYSTEM STATUS INFORMATION 285

OPERATIONAL CONSIDERATIONS 287

EXECUTING SPZAP 288
JCL Statements 288
IMASPZAP Control Statements 288

IMASPZAP OUTPUT 296
The Formatted Hexadecimal Dump 296
The Translated Dump 298

IMASPZAP EXAMPLES 300
Example 1: Inspecting and Modifying a Load Module Containing a Single CSECT 300
Example 2: Inspecting and Modifying a CSECT in a Multiple-CSECT Load Module 301
Example 3: Inspecting and Modifying Two CSECTs in the Same Load Module 303
Example 4: Inpsecting and Modifying a Data Record 305
Example 5: Entering SPZAP Control Statements Through the Console 306
Example 6: Using the BASE Control Statement 307

Figures

Figure SPZAP-1. Sample Assembly Listing Showing Multiple Control Sections 282
Figure SPZAP-2. SSI Bytes in a Load Module Directory Entry 285
Figure SPZAP-3. Flag Buts in the System Status Index Field 286
Figure SPZAP-4. Sample Formatted Hexadecimal Dump 297
Figure SPZAP-5. Sample Translated Dump 299

Introduction

IMASPZAP is a service aid program that operates under control of the System/360 Operating System. This program is designed to enable authorized personnel to:

- Inspect and modify instructions and data in any load module that exists as a member of a partitioned data set.
- Inspect and modify data in a specific data record that exists in a direct access data set.
- Dump an entire data set, a specific member of a partitioned data set, or any portion of a data set residing on a direct access device.
- Update the System Status Index (SSI) in the directory entry for any load module.

Capabilities of SPZAP

The functions of IMASPZAP provide the user with many capabilities. Three of these are suggested below.

- By using the inspect and modify functions of IMASPZAP, programming errors that require only the replacement of instructions in a load module can be fixed on the spot, thus eliminating the need for immediate recompilation of the program.
- In another instance, the user may want to obtain a storage dump for the purpose of diagnosing a problem. The modify function of IMASPZAP could be used to alter an instruction in the problem program and cause the execution of the job to terminate at a precise location. A dump of storage would automatically be given at the forced termination of the program.
- Since IMASPZAP can replace data directly on a direct access device, it could also be used to reconstruct VTOCs or data records that may have been destroyed as the result of a device I/O error or a programming error.

Monitoring the Use of SPZAP

Because IMASPZAP provides the ability to modify data on a direct access storage device, misuse of this program could result in serious damage to both user and system load modules or data sets. To protect against the occurrence of such damage by IMASPZAP, two means of controlling its use are suggested below:

- One means of exercising control is provided by IBM under MFT II and MVT. The System Management Facility (SMF) provides a system interface with user exit routines for the purpose of monitoring the job stream. Essentially, this facility, when incorporated into the system, affords an internal means of checking to see whether a particular user is authorized to execute the program specified on the EXEC job control language statement. (For further information on the SMF facility, refer to the publication Data Management for System Programmers, GC28-6550.)

- A second means of protecting against unauthorized use of IMASPZAP is to store IMASPZAP in a "password protected" private library. If IMASPZAP is located in such a library, any person trying to execute this program would be required to include in his JCL statements a JOBLIB DD statement defining the library, and at initiation time he would be required to give the password associated with the library. Only personnel knowing the password would then be able to execute IMASPZAP. Password protected libraries are discussed in the publication Data Management for System Programmers, GC28-6550.

Data Modification and Inspection

IMASPZAP can be used to inspect and modify data in either a specific record of a direct access data set or a load module that is part of a partitioned data set. The specific functions performed are governed by the use of control statements.

The modification of data is implemented through the REP control statement. The REP operation allows the user to replace instructions or data at a specific location in a load module or physical record.

The inspection of data is implemented through the VERIFY statement. This operation is provided to protect the user against erroneous replacement of data and to allow him to conditionally modify data. The VERIFY function should be used to check the contents of a specific location in a load module or physical record prior to replacing it. If the contents at the specified location do not agree with the contents as specified in the VERIFY statement, subsequent REP operations will not be performed.

Note: Although it is not required that the VERIFY function be employed prior to the REP function, it is strongly recommended that this control function be utilized to avoid possible errors in the replacement of data.

Inspecting and Modifying a Load Module

To reference data in a load module, the user must supply IMASPZAP with the member name of the load module through the use of a NAME control statement. The load module must be a member of the partitioned data set identified by the SYSLIB DD statement included in the execution JCL.

If the load module being inspected or modified contains more than one control section (CSECT), the user must also supply IMASPZAP with the name of the CSECT that is to be involved in the operations of the program. If no CSECT name is given in the NAME statement, IMASPZAP will assume that the control section to be referenced is the first one encountered in searching the load module.

IMASPZAP will place descriptive maintenance data in the IMASPZAP CSECT Identification Record (IDR) of the load module whenever a REP operation associated with a NAME statement is performed on a control section contained in that module. This function will be performed automatically after all REP statements associated with the NAME statement have been processed; any optional user data that has to be placed in the IDR will come from the IDRDATA statement (See "IMASPZAP Control Statements" for an explanation of the IDRDATA statement).

Accessing a Load Module

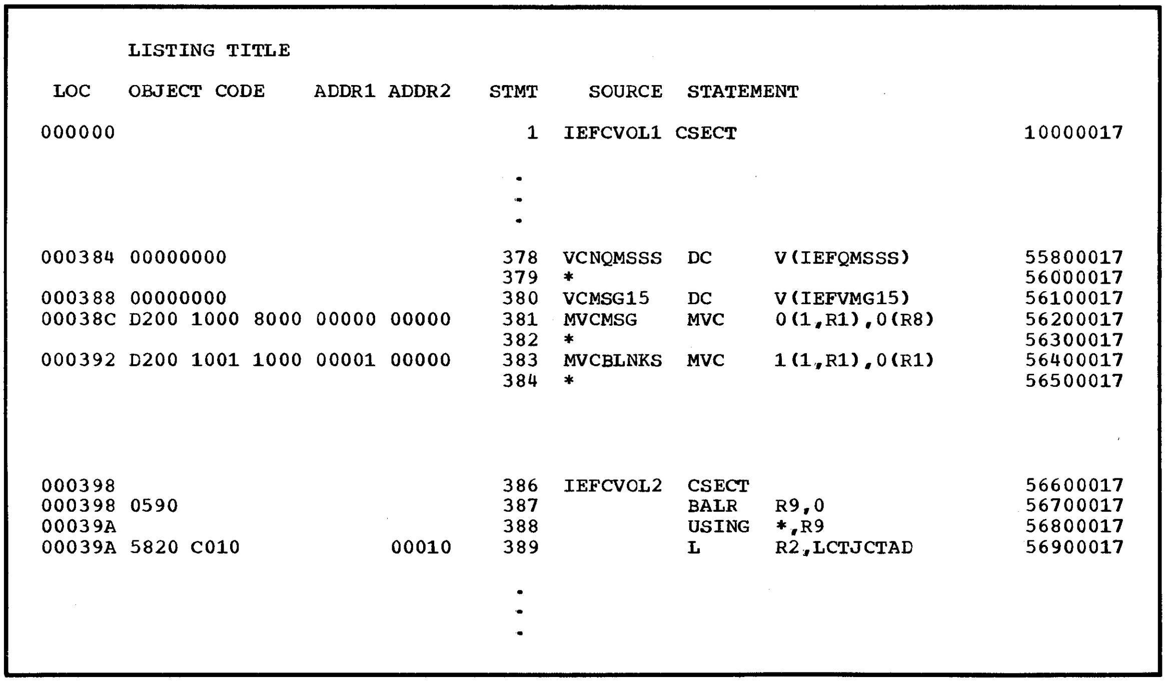
Once the CSECT has been found, IMASPZAP must locate the data that is to be verified and replaced. This is accomplished through the use of offset parameters in the VERIFY and REP statements. These parameters are specified in hexadecimal notation, and define the displacement of the data relative to the beginning of the CSECT. For example, if a hexadecimal offset of X'40' is specified in a VERIFY statement, IMASPZAP will find the location that is 64 bytes beyond the beginning of the CSECT identified by the NAME statement, and begin verifying the data from that point.

Normally, the assembly listing address associated with the instruction to be inspected or modified can be used as the offset value in the VERIFY or REP statement. However, if a CSECT has been assembled with other CSECTs so that its origin is not at assembly location zero, then the locations in the assembly listing do not reflect the correct displacements of data in the CSECT. The proper displacements must be computed by subtracting the assembly listing address delimiting the start of the CSECT from the assembly listing address of the data to be referenced.

To eliminate the need for such calculations and allow the user to use the assembly listing locations, IMASPZAP provides a means of adjusting the offset values on VERIFY and REP statements. This is achieved through the use of the BASE control statement. This statement should be included in the input to IMASPZAP immediately following the NAME statement that identifies the CSECT. The parameter in the BASE statement must be the assembly listing address (in hexadecimal) at which the CSECT beings. IMASPZAP will then subtract this value from the offset specified on any VERIFY or REP statement that follows the BASE statement, and use the difference as the displacement of the data.

The usage of the control statements mentioned in the above discussion is explained in detail in the section entitled "IMASPZAP Control Statements."

Figure SPZAP-1 exemplifies an assembly listing showing more than one control section. If a user were to reference the second CSECT (IEFCVOL2), he could include in the input to IMASPZAP a BASE statement with a location of 0398. Then, to refer to the subsequent LOAD instruction (L R2,LCTJCTAD), he could use an offset of 039A in the VERIFY or REP statements that follow in the IMASPZAP input stream.



```
        LISTING TITLE

  LOC   OBJECT CODE    ADDR1 ADDR2   STMT    SOURCE  STATEMENT

000000                                  1  IEFCVOL1 CSECT                           10000017

                                        .
                                        .
                                        .

000384 00000000                       378  VCNQMSSS  DC      V(IEFQMSSS)            55800017
                                      379  *                                        56000017
000388 00000000                       380  VCMSG15   DC      V(IEFVMG15)            56100017
00038C D200 1000 8000 00000 00000     381  MVCMSG    MVC     0(1,R1),0(R8)          56200017
                                      382  *                                        56300017
000392 D200 1001 1000 00001 00000     383  MVCBLNKS  MVC     1(1,R1),0(R1)          56400017
                                      384  *                                        56500017



000398                                386  IEFCVOL2  CSECT                          56600017
000398 0590                           387            BALR    R9,0                   56700017
00039A                                388            USING   *,R9                   56800017
00039A 5820 C010            00010     389            L       R2,LCTJCTAD            56900017

                                        .
                                        .
                                        .
```

Figure SPZAP-1. Sample Assembly Listing Showing Multiple Control Sections

Inspecting and Modifying a Data Record

To reference a specific data record, the user must specify the actual cylinder, track and record numbers that comprise the direct access address associated with it. The CCHHR control statement used by IMASPZAP relates this information to the program. This CCHHR address must be within the limits of the direct access data set defined in the SYSLIB DD control statement.

To provide a record of modifications to potentially sensitive data records, a REP operation associated with a CCHHR statement will cause IMASPZAP to write message IMA121I to the operator.

Accessing a Data Record

When this type of reference is used, IMASPZAP is able to read directly the physical record the user wants to inspect or modify. The offset parameters specified in subsequent VERIFY and REP statements are then used to locate the data that is to be verified or replaced within the record. These offsets must be specified in hexadecimal notation and define the displacement of data relative to the beginning of the record. Also, the user must include the length of any key data field in the calculation of his offset values. This is because IMASPZAP considers the key associated with a direct access record to be part of it.

Dumping Data

The dumping options provided by IMASPZAP constitute a very necessary facility for the user. By providing a visual picture of the load module or data record that has been changed, the dump feature allows the user to double check the modifications he has made.

There are two formats in which the data may be dumped. In the first (formatted hexadecimal dump) the data requested for the dump is printed in hexadecimal. The second format (translated dump) includes the hexadecimal data, a translation of all printable characters, and, where applicable, an indication of mnemonic operation code equivalents. (Refer to "IMASPZAP Output" for figures showing these two kinds of dumps.)

The DUMP and ABSDUMP statements are the control statements used to specify the options described above. A user may also indicate the portion of the data he wants IMASPZAP to dump. The operation code in the DUMP and ABSDUMP statements indicates the kind of dump wanted; the parameters identify the portion of the data to be dumped. (Use of the DUMP and ABSDUMP statements is discussed in detail under the topic "IMASPZAP Control Statements.")

Updating System Status Information

The system status index (SSI) is a 4-byte field created (upon request) in the directory entry of a load module at linkage editor processing time. Its primary function is to retain information pertaining to the status of the load module. This index is useful for keeping track of any modifications that are performed on a load module. The IMASPZAP program will, as part of its normal function, update the system status index (when there is one) to reflect local modification when a replacement of data in a module is effected. The user can also, by means of the SETSSI control statement, insert his own 4-byte information field into the SSI, overlaying the information originally stored there. However, for purposes of maintaining an accurate record of the status of a load module, the SETSSI statement should be used with caution.

To ensure proper use of the SETSSI statement, an explanation of the location, significance, and format of the system status index is provided here. For more detailed information regarding the SSI, refer to the publication IBM System/360 Operating System: Maintenance Program, GC27-6918.

The System Status Index (if it exists) is located in the last four bytes of the user data field in the directory entry for a load module. Figure SPZAP-2 shows the position of the SSI in load module directory entries.

Member Name	TTR	C	User Data Field	SSI
1 8	9 11	12	13 to 70 maximum	variable

Figure SPZAP-2. SSI Bytes in a Load Module Directory Entry

Figure SPZAP-3 gives a breakdown of the System Status Index field and the flag bits used to indicate the types of changes made to the corresponding load module program. A detailed explanation of this field and its applicability to the IMASPZAP program follows.

As shown in Figure SPZAP-3, the first byte of SSI information contains the member's change level. When a load module is initially released by IBM, its change level is set at one. Thereafter, the change level is incremented by one for each release that includes a new version of that program. If a user makes a change to the SSI for any of the IBM-released programs, he should take care not to destroy this maintenance level indicator unless he purposely means to do so. To keep the change level byte at its original value, he should find out what information is contained in the SSI before using the SETSSI function of IMASPZAP.

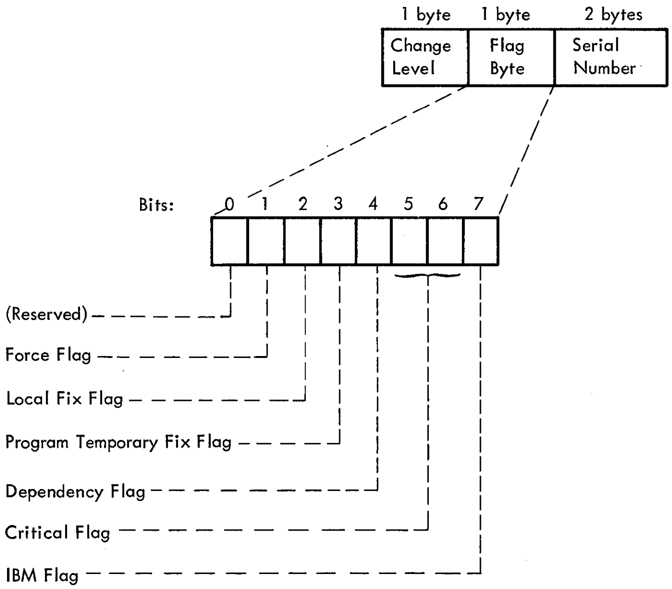


Figure SPZAP-3. Flag Bits in the System Status Index Field

Note: IMBLIST can be used to determine the SSI setting prior to making any modification to this status indicator.

The second byte of the SSI is termed the _flag byte_. Bits within the flag byte contain information reflecting the member's maintenance status. Using IMASPZAP, a user need only be concerned with two of the eight bits:

- The local fix flag contained in the third bit (bit 2) is used to indicate that the user has modified a particular member. (This is opposed to IBM PTF changes.) IMASPZAP sets this local fix flag bit on after successful completion of a modify operation to a load module.

- The program temporary fix flag (relative bit 3) is set on when an IBM-authorized program temporary fix (PTF) is applied to a system library to correct an erroneous IBM module.

All other bits in the flag byte should be retained in the SSI as they appeared before the SETSSI operation was enacted, so as not to interfere with the normal system maintenance procedures.

The third and fourth bytes of the system status index are used to store a serial number that identifies the first digit and the last three digits of a PTF number. These bytes are not altered by IMASPZAP unless the user deliberately changes them with a SETSSI statement.

Operational Considerations

Technical considerations for the use of the IMASPZAP service aid program are listed below:

- IMASPZAP utilizes system OPEN, and therefore cannot modify "read-only" or inspect "write-only" password protected data sets unless the correct password is provided at OPEN.

- Unexpired data sets such as system libraries cannot be modified unless the operator replies r 00,'U' to the expiration message that occurs during OPEN.

- If IMASPZAP is used to modify an operating system module that is loaded only at IPL time, an additional IPL is required to invoke the new version of the altered module.

- The SYSLIB DD statement cannot define a concatenated data set.

- IMASPZAP supports only the following direct access devices: 2311, 2312, 2313, 2314, 2318, 2319, 2301, 2302, 2303, 2305, 2321, and 3330. One of these devices must be specified in the unit parameter of the SYSLIB DD statement.

- IMASPZAP is a non-reusable module.

- When modifying a system data set, such as SYS1.LINKLIB, DISP=OLD should be specified on the SYSLIB DD statement.

Executing IMASPZAP

Both JCL and control statements are requred to execute IMASPZAP. The following sections describe the required statements.

JCL Statements

SPZAP can be executed using the following job control statements. The minimum region for execution is 13K plus the larger of 3K or the blocksize in bytes for the data set specified on the SYSLIB DD statement.

JOB Statement

marks the beginning of the job.

EXEC Statement

invokes the program IMASPZAP.

SYSPRINT DD Statement

defines a sequential output message data set, that can be written on a system printer, a magnetic tape volume, or a direct access volume. This statement is required for each execution of IMASPZAP.

SYSLIB DD Statement (required for each execution)

defines the direct access data set that will be accessed by IMASPZAP when performing the operations specified on the control statements. The DSNAME parameter and DISP=OLD or DISP=SHR must always be defined. The VOLUME and UNIT parameters are necessary only of the data set is not cataloged. When this data set is the VTOC, DSNAME=FORMAT4.DSCB must be specified. This statement cannot define a concatenated data set.

SYSABEND DD Statement

defines a sequential output data set to be used in case IMASPZAP terminates abnormally. This data set can be written to a printer, a magnetic tape volume, or a direct access volume.

SYSIN DD Statement

defines the input stream data set that contains IMASPZAP control statements.

IMASPZAP Control Statements

The IMASPZAP control statements (entered either through the user's input stream or through the system console) are used to define the processing functions to be enacted during a particular execution of IMASPZAP. The statements may be grouped into three categories depending upon the program's usage of them: those that are used to reference load modules (NAME, DUMP, DUMPT, IDRDATA, SETSSI, BASE), those that refer to specific records within a data set (CCHHR, ABSDUMP, ABSDUMPT), and those that can be used to specify processing control for either type of input mentioned in the first two categories (VERIFY, REP, CONSOLE).

IMASPZAP control statements must be coded according to the following rules:

- IMASPZAP control statements may begin in any column, but the operation name must precede the parameters.
- There must be at least one blank between the specified operation name and the first parameter.
- All parameters must also be separated by at least one blank space.
- Data field parameters may be formatted with commas for easier visual check, but imbedded blanks within data fields are not permitted.
- Data and offset parameter values must be specified as a multiple of two hexadecimal digits.
- The size of an IMASPZAP control statement is 80 bytes.
- Following the last required parameter and its blank delimiter, the rest of the control statement space can be used for comments. Exceptions to this are the NAME and DUMP control statements. If the CSECT parameter is omitted from either of these statements, the space following the load module parameter should not be used for comments.
- A record beginning with an asterisk and a blank is considered to be a comment statement.

The control statements are the following:

NAME member [csect]

used to identify a CSECT in a load module that is to be the object of subsequent VERIFY, REP, or SETSSI operations. The parameters are:

member

the member name of the load module that contains the control section in which the data to be inspected and/or modified is resident. The load module must be a member of the partitioned data set defined by the SYSLIB DD statement.

csect

the name of the particular control section that contains the data to be verified or replaced. When this parameter is omitted, it is assumed that the first CSECT contained in the load module (if there is more than one) is the one to be referenced. If there is only one CSECT in the load module, this parameter is not necessary.

Note: More than one NAME statement can be defined in the input to IMASPZAP. However, the VERIFY, REP and SETSSI statements associated with each NAME statement must immediately follow the NAME statement to which they apply.

CCHHR record address

used to identify a physical record on a direct access device that is to be modified or verified. The record must be in the data set defined by the SYSLIB DD statement. Any immediately following REP or VERIFY statements will reference the data in the specified record. The parameter is:

record address

the actual direct access device address of the record containing the data to be replaced or verified. It must be specified as a 10-position hexadecimal number in the form cccchhhhrr. For all direct access devices other than the 2321 data cell, cccc is the cylinder, hhhh is the track, and rr is the record number. For example, 0001000A01 addresses record 1 of cylinder 1, track 10.

In the case of the 2321 data cell, cccc indicates the subcell and strip; hhhh indicates the cylinder and track; rr indicates the record number. The bin number to which the CCHHR applies is determined by the UNIT parameter in the SYSLIB DD statement. For example, if the SYSLIB DD specifies UNIT=2321/2 and the CCHHR statement specifies 0102000103, then record 3 of subcell 1, strip 2, cylinder 0, track 1 in bin 2 will be retrieved.

In both cases a zero record number is invalid and will default to 1.

<u>Note</u>: More than one CCHHR statement can be defined in the input to IMASPZAP. However, the VERIFY, REP and SETSSI statements associated with each CCHHR statement must immediately follow the specific CCHHR statement to which they apply.

$\left\{\begin{array}{c}\text{VERIFY}\\ \text{VER}\end{array}\right\}$ offset expected content

causes the contents at a specified location within a control section or physical record to be compared with the data the user supplies in the statement. If the two fields being compared are not in agreement, no succeeding REP or SETSSI operations, pertinent to the NAME or CCHHR statement in effect, will be performed. The parameters are:

offset

the hexadecimal displacement of the data to be inspected in a CSECT or record. This displacement does not have to be aligned on a fullword boundary, but it must be specified as a multiple of two hexadecimal digits (0D, 021C, 014682, etc.). If this offset value is outside the limits of the CSECT or data record defined by the preceding NAME or CCHHR statement, the VERIFY statement will be rejected. When inspecting a record with a key, the length of the key should be considered in the calculation of the displacement; i.e., offset zero is the first byte of the key.

expected content

defines the bytes of data that are expected at the specified location. As with the offset parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameters may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, the data may look like this:

5840C032 (without commas),

or like this:

5840,C032 (with commas)

If all the data will not fit into one VERIFY statement (80-byte logical record), then another VERIFY statement must be defined.

A formatted dump of the CSECT or data record is automatically provided following each rejected VERIFY, so that the cause of the rejection can be determined. Subsequent REP (replacement) or SETSSI operations will not be performed if a verification is rejected, but other VERIFY operations will be performed, permitting complete verification in one execution of IMASPZAP. The error condition caused by the VERIFY reject will be in effect only until another NAME or CCHHR statement is encountered. Any subsequent VERIFY or REP statements can then be processed.

REP offset data

used to modify data at a specified location in a CSECT or physical record that has been previously defined by a NAME or CCHHR statement. The data specified on the REP statement will replace the data at the record or CSECT location stipulated in the offset parameter field. This replacement is on a "one for one" basis; that is, one byte of data defined in the statement replaces one byte of data at the specified location. The parameters are:

offset

is the hexadecimal displacement of the data to be replaced in a CSECT or data record. This displacement need not address a fullword boundary, but it must be specified as a multiple of two hexadecimal digits (0D, 02C8, 001C52). If this offset value is outside the limits of the data record (physical block) or CSECT being modified, the replacement operation will not be performed. When replacing data in a record with a key, the length of the key should be considered in the calculation of the displacement; i.e., offset zero is the first byte of the key.

data

defines the bytes of data that are to be inserted at the specified location. As with the offset parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameter may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, a REP data parameter may look like this:

4160B820 (without commas)

or like this:

4160,B820 (with commas).

If all the data to be modified will not fit into one REP statement (80- byte logical record), then another REP statement must be defined.

NOTE: ALTHOUGH IMASPZAP DOES NOT REQUIRE THE USER TO VERIFY A LOCATION BEFORE PERFORMING A REP OPERATION, IT IS ADVISABLE TO CHECK THE CONTENTS TO MAKE SURE THAT THE DATA BEING CHANGED IS, IN FACT, WHAT THE USER EXPECTS IT TO BE.

The user should also keep in mind the fact that IMASPZAP, as a part of its normal function, updates the system status index (SSI) for the specified module upon successful completion of the last REP operation performed on a control section of that particular module.

For a more complete explanation of the value of the SSI to the maintenance of a load module, refer to "Updating System Status Information" in this chapter.

Two programming notes that are pertinent to this discussion of the REP statement are listed below:

- If multiple VERIFY and REP operations are to be performed on a CSECT, then all the VERIFY statements should precede all the REP statements. This procedure will ensure that all the REP operations are ignored if a VERIFY reject occurs.

- When a record in the VTOC (i.e., a DSCB) is accessed for modification, message IMA117D is written to the console. No message is issued, however, when an ABSDUMPT operation is performed on the VTOC.

IDRDATA xxxxxxxx

causes IMASPZAP to place up to eight bytes of user data into the IMASPZAP CSECT Identification Record of the load module; this is only done if a REP operation associated with a NAME statement is performed and the load module has been processed by the Linkage Editor to include CSECT Identification Records. The parameter is:

xxxxxxxx

is the eight (or less) bytes of user data (with no imbedded blanks) that is to be placed in user data field of the IMASPZAP IDR of the load module. If more than eight characters are in the parameter field only the first eight characters will be used.

The IDRDATA statement is valid only when used in conjunction with the NAME statement. It must follow its associated NAME statement and precede any DUMP or ABSDUMP statement. IDRDATA statements associated with CCHHR statements will be ignored.

SETSSI xxyynnnn

places user-supplied system status information in the PDS (partitioned data set) directory entry for the library member specified in the preceding NAME statement. The SSI, however, must have been created when the load module was link edited. The parameter is:

xxyynnnn

represents the 4 bytes of system status information the user wishes to place in the SSI field for this member. Each byte is supplied as two hexadecimal digits signifying the following:

xx - change level

yy - flag byte

nnnn - modification serial number

If an error has been detected in any previous VERIFY or REP operation, the SETSSI function will not be performed.

Note: Since all bits in the SSI entry are set (or reset) by the SETSSI statement, extreme care should be exercised in its use to avoid altering information vital to the depiction of the maintenance status of the program being changed. (See the discussion in this chapter entitled "Updating System Status Information.")

```
{DUMP  }  member  csect
{DUMPT }          ALL
```

used to dump a specific control section or all control sections in a load module. The format of the output of this dump is hexadecimal (see the discussion in this chapter entitled "IMASPZAP Output"). The DUMPT statement differs from the DUMP statement in that it also gives the user an EBCDIC and instruction mnemonic translation of the hexadecimal data. The parameters are:

member

the member name of the load module that contains the control section(s) to be dumped. (Note: This load module must be a member of a partitioned data set that is defined by the SYSLIB DD statement.)

csect

defines the name of the particular control section that is to be dumped. To dump all the CSECTs of a load module, code "ALL" instead of the CSECT name; if the CSECT parameter is omitted entirely, it is assumed that the user means to dump only the first control section contained in the load module.

```
{ABSDUMP  }   {startaddr stopaddr}
{ABSDUMPT }   {membername        }
              {ALL               }
```

These statements are used to dump a group of data records, a member of a partitioned data set, or an entire data set, as defined in the SYSLIB DD statement. If the key associated with each record is to be formatted, DCB=(KEYLEN=nn), where "nn" is the length of the record key, must also be specified by the SYSLIB DD statement. Note that when dumping a VTOC, DCB=(KEYLEN=44) should be specified; when dumping a PDS directory, DCB=(KEYLEN=8) should be specified. ABSDUMP produces a hexadecimal printout only, while ABSDUMPT prints the hexadecimal data, the EBCDIC translation, and the mnemonic equivalent of the data (see "IMASPZAP Output"). The parameters are:

startaddr

is the absolute direct access device address of the first record to be dumped. This address must be specified in hexadecimal in the form ccccHhhhrr (cylinder, track and record numbers).

stopaddr

is the absolute direct access device address of the last record to be dumped, and it must be in the same format as the start address.

Note: Both addresses must be specified when this method of dumping records is used, and both addresses must be within the limits of the data set defined by the SYSLIB DD statement. The record number specified in the start address must be a valid record number. The record number specified as the stop address need not be a valid record number, but if it is not, the dump will continue until the last record on the track specified in the stop address has been dumped.

membername

is the name of a member of a partitioned data set. The member can be a group of data records or a load module. In either case, the entire member is dumped when this parameter is specified.

ALL

specifies that the entire data set defined by the SYSLIB DD statement is to be dumped.

How much of the space allocated to the data set is dumped depends on how the data set is organized:

For sequential data set, IMASPZAP dumps until it reaches end of file.

For indexed sequential and direct access data sets, IMASPZAP dumps all extents.

For partitioned data sets, IMASPZAP dumps all extents, including all linkage editor control records, if any exist.

BASE xxxxxx

used by IMASPZAP to adjust offset values that are to be specified in any subsequent VERIFY and REP statements. This statement should be used when the offsets given in the VERIFY and REP statements for a CSECT are to be obtained from an assembly listing in which the starting address of the CSECT is not location zero.

For example, assume that CSECT ABC begins at assembly listing location X'000400', and that the data to be replaced in this CSECT is at location X'000408'. The actual displacement of the data in the CSECT is X'08'. However, an offset of X'0408' (obtained from the assembly listing location X'000408') can be specified in the REP statement if a BASE statement specifying X'000400' is included prior to the REP statement in the IMASPZAP input stream. When IMASPZAP processes the REP statement, the base value X'000400' will be subtracted from the offset X'0408' to determine the proper displacement of data within the CSECT. The parameter is:

xxxxxxxx

is a 6-character hexadecimal offset that is to be used as a base for subsequent VERIFY and REP operations. This value should reflect the starting assembly listing address of the CSECT being inspected or modified.

The BASE statement should be included in the IMASPZAP input stream immediately following the NAME statement that identifies the control section that is to be involved in the IMASPZAP operations. The specified base value remains in effect until all VERIFY, REP, and SETSSI operations for the CSECT have been processed.

CONSOLE

indicates that IMASPZAP control statements are to be entered through the system console.

When this statement is encountered in the input stream, the following message is written to the operator:

```
IMA116A ENTER IMASPZAP CONTROL STATEMENT OR END
```

The operator may then key in any valid IMASPZAP control statement conforming to the specifications described at the beginning of this control statement discussion. After each operator entry through the console is read, validated, and processed, the message is reissued, and additional input is accepted from the console until "END" is replied. IMASPZAP will then continue processing control statements from the input stream until an end-of-file condition is detected.

Note: The control statements can be entered through the console in either upper or lower case letters.

* (Comment)

can be used to annotate the IMASPZAP input stream and output listing. Any number of comment statements can be included in the input stream. When such a statement is encountered, IMASPZAP writes the entire statement to the data set specified for SYSPRINT.

The asterisk (*) can be specified in any position of the statement, but it must be followed by at least one blank space as a delimiter.

IMASPZAP Output

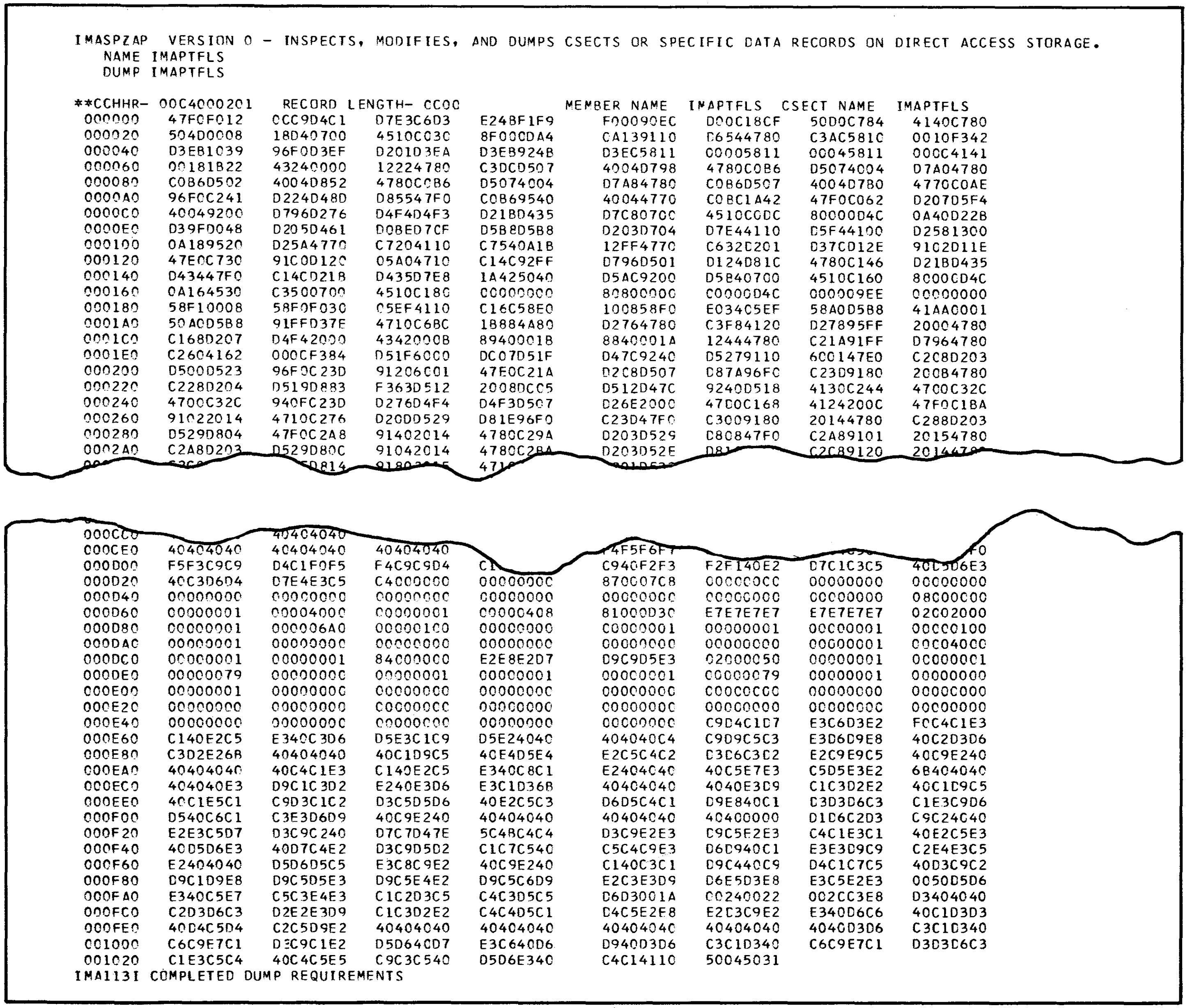
IMASPZAP provides two different dump formats for the purpose of checking the data that has been verified and/or replaced. These dumps (written to the SYSPRINT data set specified by the user) may be of the formatted hexadecimal type or the translated form. Both formats are discussed below in detail with examples showing how each type will look.

The Formatted Hexadecimal Dump

When DUMP or ABSDUMP is the control statement used, the resulting printout will be a hexadecimal representation of the requested data. Figure SPZAP-4 gives a sample of the formatted hexadecimal dump. A heading line is printed at the beginning of each block. This heading consists of the hexadecimal direct access address of the block, a two-byte record length field, and the names of the member and the control section that contain the data being printed (if the dump is for a specific CSECT or load module). Each printed line thereafter has a three-byte displacement address at the left, followed by eight groups of four data bytes each. The following message:

```
    IMA113I COMPLETED DUMP REQUIREMENTS
```

is printed directly under the last line of the dump printout.

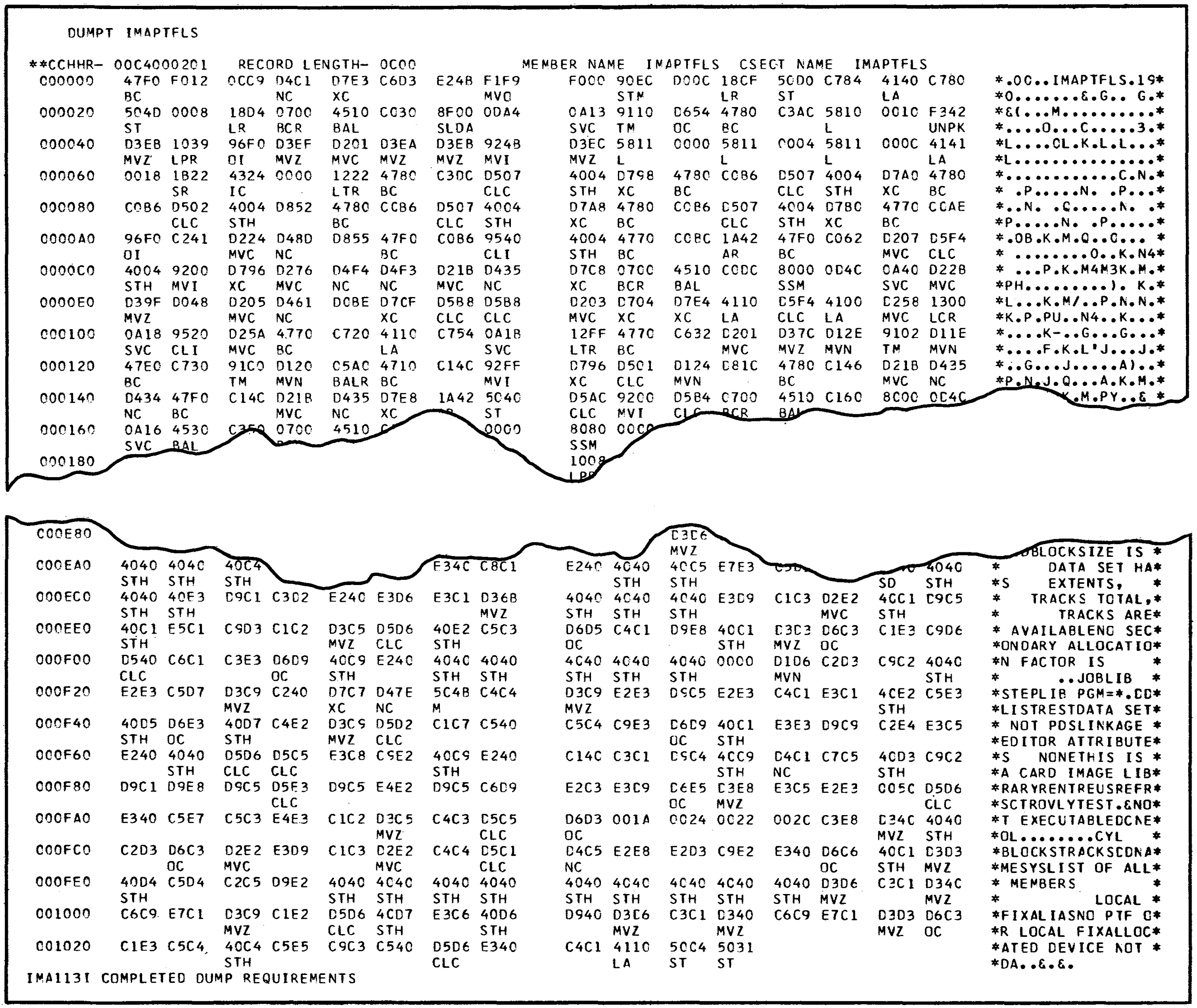


```
IMASPZAP  VERSION 0 - INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
   NAME IMAPTFLS
   DUMP IMAPTFLS

**CCHHR- 00C4000201   RECORD LENGTH- CC00              MEMBER NAME  IMAPTFLS  CSECT NAME  IMAPTFLS
 000000   47F0F012   CCC9D4C1   D7E3C6D3   E24BF1F9     F00090EC   D00C18CF   50D0C784   4140C780
 000020   504D0008   18D40700   4510C03C   8F000DA4     CA139110   D6544780   C3AC581C   0010F342
 000040   D3EB1039   96F0D3EF   D201D3EA   D3EB924B     D3EC5811   00005811   00045811   000C4141
 000060   00181B22   43240000   12224780   C3DCD507     4004D798   4780C0B6   D5074004   D7A04780
 000080   C0B6D502   4004D852   4780C0B6   D5074004     D7A84780   C0B6D507   4004D780   4770C0AE
 0000A0   96F0C241   D224D48D   D85547F0   C0B69540     40044770   C0BC1A42   47F0C062   D207D5F4
 0000C0   40049200   D796D276   D4F4D4F3   D21BD435     D7C8070C   4510C0DC   80000D4C   0A40D22B
 0000E0   D39FD048   D205D461   D0BED7CF   D5B8D5B8     D203D704   D7E44110   D5F44100   D2581300
 000100   0A189520   D25A4770   C7204110   C7540A1B     12FF4770   C632D201   D37CD12E   9102D11E
 000120   47E0C730   91C0D120   05A04710   C14C92FF     D796D501   D124D81C   4780C146   D21BD435
 000140   D43447F0   C14CD21B   D435D7E8   1A425040     D5AC9200   D5B40700   4510C160   8000CD4C
 000160   0A164530   C3500700   4510C18C   00000000     80800000   C0000D4C   000009EE   00000000
 000180   58F10008   58F0F030   05EF4110   C16C58E0     100858F0   E0340 5EF   58A0D5B8   41AA0001
 0001A0   50A0D5B8   91FFD37E   4710C6BC   1B884A80     D2764780   C3F84120   D27895FF   20004780
 0001C0   C168D207   D4F42000   4342000B   8940001B     8840001A   12444780   C21A91FF   D7964780
 0001E0   C2604162   000CF384   D51F6000   DC07D51F     D47C9240   D5279110   60C147E0   C2C8D203
 000200   D500D523   96F0C23D   91206C01   47E0C21A     D2C8D507   D87A96FC   C23D9180   200B4780
 000220   C228D204   D519D883   F363D512   2008DCC5     D512D47C   9240D518   4130C244   4700C32C
 000240   4700C32C   940FC23D   D276D4F4   D4F3D507     D26E2000   47D0C168   4124200C   47F0C1BA
 000260   91022014   4710C276   D20DD529   D81E96F0     C23D47F0   C3009180   20144780   C288D203
 000280   D529D804   47F0C2A8   91402014   4780C29A     D203D529   D80847F0   C2A89101   20154780
 0002A0   C2A8D203   D529D80C   91042014   4780C2BA     D203D52E   D81        C2C89120   20144

 000CC0              40404040
 000CE0   40404040   40404040   40404040                F4F5F6F7                          F0
 000D00   F5F3C9C9   D4C1F0F5   F4C9C9D4   C1           C940F2F3   F2F140E2   D7C1C3C5   40C5D6E3
 000D20   40C3D6D4   D7E4E3C5   C4000000   0000000C     870007C8   000000CC   00000000   00000000
 000D40   00000000   00000000   00000000   00000000     00000000   00000000   00000000   08000000
 000D60   00000001   00004000   00000001   00000408     81000D30   E7E7E7E7   E7E7E7E7   02002000
 000D80   00000001   000006A0   00000100   00000000     00000001   00000001   00000001   00000100
 000DAC   00000001   00000000   00000000   00000000     00000000   00000000   00000001   00C04000
 000DC0   00000001   00000001   84000000   E2E8E2D7     D9C9D5E3   02000050   00000001   0C000001
 000DE0   00000079   0000000C   00000001   00000001     00000001   00000079   00000001   00000000
 000E00   00000001   0000000C   00000000   0000000C     0000000C   C000C000   00000000   0000C000
 000E20   00000000   00000000   C000000C   00000000     0000000C   00000000   0000000C   00000000
 000E40   00000000   0000000C   00000000   00000000     0000000C   C9D4C1D7   E3C6D3E2   FCC4C1E3
 000E60   C140E2C5   E340C3D6   D5E3C1C9   D5E2404C     404040C4   C9D9C5C3   E3D6D9E8   40C2D3D6
 000E80   C3D2E26B   40404040   40C1D9C5   40E4D5E4     E2C5C4C2   D3D6C3D2   E2C9E9C5   40C9E240
 000EA0   40404040   40C4C1E3   C140E2C5   E340C8C1     E2404040   40C5E7E3   C5D5E3E2   6B404040
 000EC0   404040E3   D9C1C3D2   E240E3D6   E3C1D36B     40404040   4040E3D9   C1C3D2E2   40C1D9C5
 000EE0   40C1E5C1   C9D3C1C2   D3C5D5D6   40E2C5C3     D6D5C4C1   D9E840C1   C3D3D6C3   C1E3C9D6
 000F00   D540C6C1   C3E3D6D9   40C9E240   40404040     40404040   40400000   D1D6C2D3   C9C24040
 000F20   E2E3C5D7   D3C9C240   D7C7D47E   5C4BC4C4     D3C9E2E3   C9C5E2E3   C4C1E3C1   40E2C5E3
 000F40   40D5D6E3   40D7C4E2   D3C9D5D2   C1C7C540     C5C4C9E3   D6D940C1   E3E3D9C9   C2E4E3C5
 000F60   E2404040   D5D6D5C5   E3C8C9E2   40C9E240     C140C3C1   D9C440C9   D4C1C7C5   40D3C9C2
 000F80   D9C1D9E8   D9C5D5E3   D9C5E4E2   D9C5C6D9     E2C3E3D9   D6E5D3E8   E3C5E2E3   0050D5D6
 000FA0   E340C5E7   C5C3E4E3   C1C2D3C5   C4C3D5C5     D6D3001A   00240022   002CC3E8   D3404040
 000FC0   C2D3D6C3   D2E2E3D9   C1C3D2E2   C4C4D5C1     D4C5E2E8   E2D3C9E2   E340D6C6   40C1D3D3
 000FE0   40D4C5D4   C2C5D9E2   40404040   40404040     4040404C   40404040   4040D3D6   C3C1D340
 001000   C6C9E7C1   D3C9C1E2   D5D640D7   E3C640D6     D940D3D6   C3C1D340   C6C9E7C1   D3D3D6C3
 001020   C1E3C5C4   40C4C5E5   C9C3C540   D5D6E340     C4C14110   50045031
IMA113I COMPLETED DUMP REQUIREMENTS
```

Figure SPZAP-4. Sample Formatted Hexadecimal Dump

The Translated Dump

The control statements DUMPT and ABSDUMPT also provide an operation code translation and an EBCDIC representation of the data contained in the dump. Figure SPZAP-5 shows the format of the translated dump. The first byte of each halfword of data is translated into its mnemonic operation code equivalent, provided such a translation is possible. If there is no equivalent mnemonic representational value to be given, the space is left blank. This translated line of codes and blanks is printed directly under the corresponding hexadecimal line. An EBCDIC representation of each byte of data is printed on two lines to the right of the corresponding line of text with periods (.) substituted for those bytes that do not translate to valid printable characters.



```
    DUMPT IMAPTFLS

**CCHHR- 00C4000201   RECORD LENGTH- 0C00              MEMBER NAME  IMAPTFLS  CSECT NAME  IMAPTFLS
 000000    47F0 F012  0CC9 D4C1  D7E3 C6D3  E24B F1F9    F000 90EC  D00C 18CF  50D0 C784  4140 C780   *.0C..IMAPTFLS.19*
           BC              NC    XC              MVO          STM        LR    ST         LA          *0.......&.G.. G.*
 000020    504D 0008  18D4 0700  4510 C030  8F00 0DA4    0A13 9110  D654 4780  C3AC 5810  0010 F342   *&(...M..........*
           ST         LR   BCR   BAL        SLDA         SVC  TM    OC   BC         L          UNPK   *....O...C.....3.*
 000040    D3EB 1039  96F0 D3EF  D201 D3EA  D3EB 924B    D3EC 5811  0000 5811  0004 5811  000C 4141   *L....OL.K.L.L...*
           MVZ  LPR   OI   MVZ   MVC  MVZ   MVZ  MVI     MVZ  L          L          L          LA     *L...............*
 000060    0018 1B22  4324 0000  1222 4780  C3DC D507    4004 D798  4780 CCB6  D507 4004  D7A0 4780   *............C.N.*
                SR    IC         LTR  BC         CLC     STH  XC    BC         CLC  STH   XC   BC     * .P.....N. .P...*
 000080    C0B6 D502  4004 D852  4780 CCB6  D507 4004    D7A8 4780  CCB6 D507  4004 D7B0  4770 CCAE   *..N. .C.....N. .*
                CLC   STH        BC         CLC  STH     XC   BC         CLC   STH  XC    BC          *P.....N. .P.....*
 0000A0    96F0 C241  D224 D48D  D855 47F0  C0B6 9540    4004 4770  C0BC 1A42  47F0 C062  D207 D5F4   *.0B.K.M.Q..C... *
           OI         MVC  NC         BC         CLI     STH  BC         AR    BC         MVC  CLC    * ........O..K.N4*
 0000C0    4004 9200  D796 D276  D4F4 D4F3  D21B D435    D7C8 0700  4510 C0DC  8000 0D4C  0A40 D22B   * ...P.K.M4M3K.M.*
           STH  MVI   XC   MVC   NC   NC    MVC  NC      XC   BCR   BAL        SSM        SVC  MVC    *PH.........). K.*
 0000E0    D39F D048  D205 D461  D0BE D7CF  D5B8 D5B8    D203 D704  D7E4 4110  D5F4 4100  D258 1300   *L...K.M/..P.N.N.*
           MVZ        MVC  NC         XC    CLC  CLC     MVC  XC    XC   LA    CLC  LA    MVC  LCR    *K.P.PU..N4..K...*
 000100    0A18 9520  D25A 4770  C720 4110  C754 0A1B    12FF 4770  C632 D201  D37C D12E  9102 D11E   *....K-..G...G...*
           SVC  CLI   MVC  BC         LA         SVC     LTR  BC         MVC   MVZ  MVN   TM   MVN    *....F.K.L'J...J.*
 000120    47E0 C730  91C0 D120  C5AC 4710  C14C 92FF    D796 D501  D124 D81C  4780 C146  D21B D435   *..G...J.....A)..*
           BC         TM   MVN   BALR BC         MVI     XC   CLC   MVN        BC         MVC  NC     *P.N.J.Q...A.K.M.*
 000140    D434 47F0  C14C D21B  D435 D7E8  1A42 5040    D5AC 9200  D5B4 C700  4510 C160  8000 0C4C          K.M.PY..& *
           NC   BC         MVC   NC   XC         ST      CLC  MVI   CL   BCR   BA
 000160    0A16 4530  C350 0700  4510 C          0000    8080 00C
           SVC  BAL                                      SSM
 000180                                                  1008
                                                         LPR

 000E80                                                       D3D6
                                                              MVZ                                        BLOCKSIZE IS *
 000EA0    4040 4040  40C4                  E34C C8C1    E240 4040  40C5 E7E3                  4040   *     DATA SET HA*
           STH  STH   STH                                     STH   STH                   SD   STH    *S    EXTENTS,   *
 000EC0    4040 40E3  D9C1 C3D2  E240 E3D6  E3C1 D36B    4040 4040  4040 E3D9  C1C3 D2E2  40C1 D9C5   *   TRACKS TOTAL,*
           STH  STH                              MVZ     STH  STH   STH             MVC   STH         *      TRACKS ARE*
 000EE0    40C1 E5C1  C9D3 C1C2  D3C5 D5D6  40E2 C5C3    D6D5 C4C1  D9E8 40C1  D3D3 D6C3  C1E3 C9D6   * AVAILABLENO SEC*
           STH                   MVZ  CLC   STH          OC                STH   MVZ  OC                *ONDARY ALLOCATIO*
 000F00    D540 C6C1  C3E3 D6D9  40C9 E240  4040 4040    4040 4040  4040 0000  D1D6 C2D3  C9C2 4040   *N FACTOR IS     *
           CLC             OC    STH        STH  STH     STH  STH   STH        MVN                  STH    *       ..JOBLIB *
 000F20    E2E3 C5D7  D3C9 C240  D7C7 D47E  5C4B C4C4    D3C9 E2E3  D9C5 E2E3  C4C1 E3C1  40E2 C5E3   *STEPLIB PGM=*.DD*
                      MVZ        XC   NC    M            MVZ                              STH         *LISTRESTDATA SET*
 000F40    40D5 D6E3  40D7 C4E2  D3C9 D5D2  C1C7 C540    C5C4 C9E3  D6D9 40C1  E3E3 D9C9  C2E4 E3C5   * NOT PDSLINKAGE *
           STH  OC    STH        MVZ  CLC                           OC   STH                          *EDITOR ATTRIBUTE*
 000F60    E240 4040  D5D6 D5C5  E3C8 C9E2  40C9 E240    C14C C3C1  D9C4 40C9  D4C1 C7C5  40D3 C9C2   *S   NONETHIS IS *
                STH   CLC  CLC              STH                          STH   NC         STH         *A CARD IMAGE LIB*
 000F80    D9C1 D9E8  D9C5 D5E3  D9C5 E4E2  D9C5 C6D9    E2C3 E3D9  D6E5 D3E8  E3C5 E2E3  005C D5D6   *RARYRENTREUSREFR*
                           CLC                                      OC   MVZ                   CLC    *SCTROVLYTEST.&NO*
 000FA0    E340 C5E7  C5C3 E4E3  C1C2 D3C5  C4C3 D5C5    D6D3 001A  0024 0022  002C C3E8  D34C 4040   *T EXECUTABLEDCNE*
                                      MVZ        CLC     OC                               MVZ  STH    *OL........CYL   *
 000FC0    C2D3 D6C3  D2E2 E3D9  C1C3 D2E2  C4C4 D5C1    D4C5 E2E8  E2D3 C9E2  E340 D6C6  40C1 D3D3   *BLOCKSTRACKSDDNA*
                OC    MVC             MVC        CLC     NC                          OC   STH  MVZ    *MESYSLIST OF ALL*
 000FE0    40D4 C5D4  C2C5 D9E2  4040 4040  4040 4040    4040 4040  4040 4040  4040 D3D6  C3C1 D34C   * MEMBERS        *
           STH                   STH  STH   STH  STH     STH  STH   STH  STH   STH  MVZ        MVZ    *          LOCAL *
 001000    C6C9 E7C1  D3C9 C1E2  D5D6 40D7  E3C6 40D6    D940 D3D6  C3C1 D340  C6C9 E7C1  D3D3 D6C3   *FIXALIASNO PTF O*
                      MVZ        CLC  STH        STH          MVZ        MVZ              MVZ  OC     *R LOCAL FIXALLOC*
 001020    C1E3 C5C4  40C4 C5E5  C9C3 C540  D5D6 E340    C4C1 4110  50C4 5031                         *ATED DEVICE NOT *
                      STH                   CLC               LA    ST   ST                           *DA..&.&.
 IMA113I COMPLETED DUMP REQUIREMENTS
```

Figure SPZAP-5. Sample Translated Dump

IMASPZAP Examples

Example 1: Inspecting and Modifying a Load Module Containing a Single CSECT

This example shows how to inspect an modify a load module containing a single CSECT.

```
//ZAPCSECT         JOB        MSGLEVEL=(1,1)
//STEP             EXEC       PGM=IMASPZAP
//SYSPRINT         DD         SYSOUT=A
//SYSLIB           DD         DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN            DD         *
    NAME       IEEVLNKT
    VERIFY     0018        C9C8,D2D9,D1C2,C7D5
    REP        0018        E5C6,D3D6,E6F0,4040
    SETSSI     01211234
    IDRDATA    71144
    DUMP       IEEVLNKT
/*
```

In this example:

JOB Statement

initiates the job

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the system library SYS1.LINKLIB containing the module IEEVLNKT that SPZAP is to process.

SYSIN DD Statement

defines the input stream.

NAME Control Statement

instructs IMASPZAP that the operations defined by the control statements that follow are to be performed on the module IEEVLNKT.

VERIFY Control Statement

requests that IMAPSPZAP check the hexadecimal data at the location that is offset X'0018' from the start of the module IEEVLNKT to make sure that it is the same as the hexadecimal data specified in this statement. If the data is the same, IMASPZAP continues processing the subsequent statements sequentially. If the data is not identical, IMASPZAP dumps a hexadecimal image of the module IEEVLNKT to the SYSPRINT data set. As a result of this "VERIFY REJECT", IMASPZAP will not perform the REP and SETSSI operations requested for the module. It will, however, perform the DUMP operation requested before discontinuing the processing.

REP Control Statement

causes IMASPZAP to replace the data at hexadecimal offset 0018 from the start of module IEEVLNKT with the data given in this control statement, provided the VERIFY statement was successful.

SETSSI Control Statement

instructs IMASPZAP that it is to replace the system status information in the directory entry for module IEEVLNKT with the SSI data given in the statement, provided the VERIFY statement was successful. The new SSI is to contain:

1. A change level of 01,
2. A flag byte of 21,
3. A serial number of 1234.

IDRDATA Control Statement

causes IMASPZAP to update the IDR in module IEEVLNKT with the data 71144, if the REP operation is successful.

DUMP Control Statement

requests that a hexadecimal image of module IEEVLNKT be dumped to the SYSPRINT data set. Since the DUMP statement follows the REP statement, the image will reflect the changes made by IMASPZAP (provided the control statements were successfully verified).

Example 2: Inspecting and Modifying a CSECT in a Multiple-CSECT Load Module

This example show how to apply an IBM-supplied PTF in the form of an IMASPZAP fix, rather than a module replacement PTF.

```
//PTF40228         JOB       MSGLEVEL=(1,1)
//STEP             EXEC      PGM=IMASPZAP
//SYSPRINT         DD        SYSOUT=A
//SYSLIB           DD        DSNAME=SYS1.NUCLEUS,DISP=OLD
//SYSIN            DD        *
    NAME       IEANUC01  IEWFETCH
    IDRDATA    LOCFIX01
    VERIFY     01F0 47F0C018
    VERIFY     0210 5830C8F4
    REP        01F0 4780C072
    REP        0210 4130C8F4
    SETSSI     02114228
    DUMPT      IEANUC01  IEWFETCH
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the library (SYS1.NUCLEUS) that contains input module IEANUC01.

SYSIN DD Statmenet

defines the input stream that contains the SPZAP control statements.

NAME Control Statement

instructs IMASZAP that the operations defined by the control statements that immediately follow this statement are to be performed on the CSECT IEWFETCH contained in the load module IEANUC01.

IDRDATA Control Statement

causes IMASPZAP to update the IDR in module IEANUC01 for CSECT IEWFETCH with the data LOCIX01, if either of the REP operations is successful.

VERIFY Control Statements

request that IMASPZAP compare the contents of the locations X'01F0' and X'0210' in the control section IEWFETCH with the data given in the VERIFY control statements. If the comparisons are equal, IMASPZAP will continue processing subsequent control statements in the order in which they are encountered. However, if the data at the locations does not compare identically to the data given in the VERIFY control statements, IMASPZAP will dump a hexadecimal image of CSECT IEWFETCH to the SYSPRINT data set; the subsequent REP and SETSSI statements will be ignored. The DUMPT function specified will be performed before IMASPZAP terminates processing.

REP Control Statements

cause IMASPZAP to replace the data at hexadecimal offsets X'01F0' and X'0210' from the start of CSECT IEWFETCH with the hexadecimal data specified on the corresponding REP statements.

SETSSI Control Statement

requests that IMASPZAP replace the system status information in the directory for module IEANUC01 with the SSI data given in the SETSSI statement after the replacement operations have been effected. The new SSI will contain:

1. A change level of 02,

2. A flag byte of 11,

3. A serial number of 4228.

DUMPT Control Statement

causes IMASPZAP to perform the DUMPT function for CSECT IEWFETCH of load module IEANUC01.

Example 3: Inspecting and Modifying Two CSECTs in the Same Load Module

This example shows how to inspect and modify two control sections in the same module.

```
//CHANGIT            JOB       MSGLEVEL=(1,1)
//STEP               EXEC      PGM=IMASPZAP
//SYSPRINT           DD        SYSOUT=A
//SYSLIB             DD        DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN              DD        *
    NAME      IEFX5000  IEFQMSSS
    VERIFY    0284 4780,C096
    REP       0284 4770,C096
    IDRDATA   PTF01483
    SETSSI    01212448
    DUMPT     IEFX5000  IEFQMSSS
    NAME      IEFX5000  IEFQMRAW
    VERIFY    0154 4780,C042
    REP       0514 4770,C042
    IDRDATA   PTF01483
    SETSSI    01212448
    DUMPT     IEFX5000  IEFQMRAW
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the data set to be accessed by IMASPZAP while performing the operations specified by the control statements. In this case, it defines the system library SYS1.LINKLIB containing the load module IEFX5000 that is to be changed by IMASPZAP.

NAME Control Statement #1

instructs IMASPZAP that the operations requested via the control statements immediately following it are to be performed on CSECT IEFQMSSS in load module IEFX5000 that resides in the data set defined by the SYSLIB DD statement.

VERIFY Control Statement #1

requests that IMASPZAP check the hexadecimal data at offset X'0284' from the beginning of CSECT IEFQMSSS to make sure it is the same as the data specified in this control statement. If the two data fields match, IMASPZAP continues processing the control statements that follow sequentially. If the data is not identical, IMASPZAP dumps a formatted hexadecimal image of CSECT IEFQMSSS to the SYSPRINT data set. If a "VERIFY REJECT" occurred, IMASPZAP would not perform the REP or SETSSI functions for CSECT IEFQMSSS, but it would implement the DUMPT function specified for this CSECT and continue to process the control statements that follow in the same job step.

REP Control Statement #1

causes IMASPZAP to replace the data at hexadecimal displacement 0284 from the beginning of CSECT IEFQMSSS with the hexadecimal data given in this control statement.

IDRDATA Control Statement #1

causes IMASPZAP to update the IDR in module IEFX5000 for CSECT IEFQMSSS with the data PTF01483, if the first REP operation is successful.

SETSSI Control Statement #1

instructs IMASPZAP that it is to replace the system status information in the directory entry for module IEFX5000 with the SSI data given. The new SSI will contain:

1. A change level of 01,

2. A flag byte of 21,

3. A serial number of 2448.

DUMPT Control Statement #1

causes IMASPZAP to perform the DUMPT operation on CSECT IEFQMSSS, and nullifies any previous "VERIFY REJECTS" that may have been encountered.

NAME Control Statement #2

indicates that the operations defined by the control statements that immediately follow this statement are to be performed on CSECT IEFQMRAW in the load module IEFX5000.

VERIFY Control Statement #2

requests that IMASPZAP perform the VERIFY function at offset X'0154' from the start of CSECT IEFQMRAW. If the VERIFY operation is successful, IMASPZAP will continue processing the subsequent control statements sequentially. If the VERIFY is rejected, however, IMASPZAP will not perform the following REP or SETSSI operations, but it will dump a hexadecimal image of CSECT IEFQMRAW to the SYSPRINT data set and perform the DUMPT operation as requested.

REP Control Statement #2

causes IMASPZAP to replace the data at hexadecimal offset X'0154' from the start of CSECT IEFQMRAW with the hexadecimal data that is specified in this control statement.

IDRDATA Control Statement #2

causes IMASPZAP to update the IDR in module IEFX5000 for CSECT IEFQMRAW with the data PTF01483, if the second REP operation is successful.

SETSSI Control Statement #2

causes IMASPZAP to perform the same function as the previous SETSSI, but it is performed only if the second VERIFY is not rejected.

DUMPT Control Statement #2

causes IMASPZAP to perform the DUMPT function on control section IEFQMRAW.

Example 4: Inspecting and Modifying a Data Record

In this example, the data set to be modified is a volume table of contents.

```
//ZAPIT       JOB       MSGLEVEL=(1,1)
//STEP        EXEC      PGM=IMASPZAP
//SYSPRINT    DD        SYSOUT=A
//SYSLIB      DD        DSNAME=FORMAT4.DSCB,DISP=OLD,
//        UNIT=2311,VOLUME=SER=111111,DCB=(KEYLEN=44)
//SYSIN       DD        *
    CCHHR           005000001
    VERIFY          2C    0504
    REP             2C    0A08
    REP             2E    0001,03000102
    ABSDUMPT        ALL
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the data set to be accessed by IMASPZAP in performing the operations specified by the control statements. In this example, it defines the VTOC (a Format 4 DSCB) on a 2311 volume with a serial number of 111111. DCB=(KEYLEN=44) is specified so that the dump produced by the ABSDUMPT control statement will show the dsname which is a 44 byte key. Note that this is not necessary for the VERIFY and REP control statements.

CCHHR Control Statement

indicates that IMASPZAP is to access the direct access record address "0005000001" in the data set defined by the SYSLIB DD statement while performing the operations specified by the following control statements.

VERIFY Control Statement

requests that IMASPZAP check the data at hexadecimal displacement X'2C' from the start of the data record defined in the CCHHR statement to make sure it is the same as the hexadecimal data specified in this control statement. If the data is the same, IMASPZAP continues processing the following control statements sequentially. If the data is not identical, IMASPZAP dumps a formatted hexadecimal image of the data record defined by the CCHHR statement to the SYSPRINT data set. If a "VERIFY REJECT" occurred, IMASPZAP would not perform the REP functions requested, but it would give the dump specified by the ABSDUMPT statement.

REP Control Statements

cause the eight bytes of data starting at displacement 2C from the beginning of the record to be replaced with the hexadecimal data in the REP control statements. The 2C displacement value allows for a 44-byte key at the beginning of the record.

ABSDUMPT Control Statement

causes IMASPZAP to dump the entire data set to the SYSPRINT data set. Since DCB=(KEYLEN=44) is specified on the SYSLIB DD statement, the 44 byte dsname will also be dumped.

Note: If the VTOC is to be modified, message IMA117D will be issued to the operator, requesting permission for the modification.

Example 5: Entering SPZAP Control Statements Through the Console

This example shows how to enter IMASPZAP control statement through the console.

```
//CONSOLIN          JOB       MSGLEVEL=(1,1)
//STEP              EXEC      PGM=IMASPZAP
//SYSPRINT          DD        SYSOUT=A
//SYSLIB            DD        DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN             DD        *
         CONSOLE
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the data set that contains the module to be updated.

SYSIN DD Statement

defines the input stream.

CONSOLE Control Statement

indicates that IMASPZAP control statements are to be entered through the console.

Example 6: Using the BASE Control Statement

This example shows how to inspect and modify a CSECT whose starting address does not coincide with assembly listing location zero.

```
//MODIFY      JOB      MSGLEVEL=(1,1)
//STEP        EXEC     PGM=IMASPZAP
//SYSPRINT    DD       SYSOUT=A
//SYSLIB      DD       DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN       DD       *
    NAME           IEFMCVOL  IEFCVOL2
    BASE           0398
    IDRDATA        MOD04
    VERIFY         039A 5820C010
    REP            039A 47000000
    DUMP           IEFMCVOL  IEFCVOL2
/*
```

JOB Statement

initiates the job.

EXEC Statement

invokes IMASPZAP.

SYSPRINT DD Statement

defines the message data set.

SYSLIB DD Statement

defines the data set to be accessed by IMASPZAP when performing the operations requested via the control statements. In this case, it defines the system library, SYS1.LINKLIB, that contains the module IEFMCVOL in which the CSECT to be changed, IEFCVOL2, resides.)

SYSIN DD Statement

defines the input stream that contains the IMASPZAP control statements.

NAME Control Statement

instructs IMASPZAP that the operations defined by the control statements that immediately follow it are to be performed on CSECT IEFCVOL2 in the load module IEFMCVOL.

BASE Control Statement

provides IMASPZAP with a base value that is to be used to readjust the offsets on the VERIFY and REP statements that follow it.

IDRDATA Control Statement

causes IMASPZAP to update the IDR in module IEFMCVOL for CSECT IEFCVOL2 with the data MOD04, the the REP operation is successful.

VERIFY Control Statment

requests that IMASPZAP inspect the data at offset X'039A'. The base value X'0398' given in the previous BASE statement is subtracted from this offset to determine the proper displacement of the data within CSECT IEFCVOL2. Therefore, IMASPZAP checks the data at the location that is actually displaced X'0002' bytes from the beginning of CSECT IEFCVOL2 to ensure that it is the same as the hexadecimal data specified in this control statement.

If the data is the same, IMASPZAP continues processing the following statements in the order in which they are encountered. If the data is not identical, IMASPZAP dumps a hexadecimal image of CSECT IEFCVOL2 to the SYSPRINT data set.

If a "VERIFY REJECT" occurs, IMASPZAP will not perform the REP, SETSSI, and IDRDATA functions, but it will perform the DUMP function requested for CSECT IEFCVOL2.

REP Control Statement

causes IMASPZAP to replace the data at displacement X'0002' (offset 039A minus base value 0398) into CSECT IEFCVOL2 with the hexadecimal data specified in this control statement.

DUMP Control Statement

requests that IMASPZAP dump a hexadecimal image of CSECT IEFCVOL2 to the SYSPRINT data set. Since the DUMP statement follows the REP statement, the image will reflect the changes made by IMASPZAP (assuming no verification has been rejected).

Appendix: Writing EDIT User Programs

Tells how to write and use EDIT user programs. → 309

Contents

INTRODUCTION 313

USER PROGRAM INTERFACES 314
Gaining Control 314
Using the Parameter List 314
  Input Record 315
  GTF Option Word 316
Returning to EDIT 316
  Exit Routine Return Codes 317
  Format Appendage Return Codes 317
Handling Errors 317
  Errors in Finding or Loading a User Program 318
  Invalid Return Codes and Program Checks 318

AVOIDING UNRECOVERABLE ERRORS .319

SAMPLE USER EXIT ROUTINE 321

SAMPLE FORMAT APPENDAGE 324

DEBUGGING A USER PROGRAM 329

JCL AND CONTROL STATEMENT EXAMPLES 331
Example 1: Link Editing a User Exit Routine into a Library 331
Example 2: Testing a User Exit Routine 332
Example 3: Testing a User Format Appendage 333

Figures

Figure APNDX-1. EDIT Parameter List and Related Fields 315
Figure APNDX-2. Contents of GTF Option Word, Showing GTF Options in Effect 316
Figure APNDX-3. PRDMP/EDIT Actions in Response to Errors in Finding or Loading User Programs 318
Figure APNDX-4. Sample Exit Routine (Part 1 of 3) 321
Figure APNDX-4. Sample Exit Routine (Part 2 of 3) 322
Figure APNDX-4. Sample Exit Routine (Part 3 of 3) 323
Figure APNDX-5. Sample Format Appendage (Part 1 of 2) 324
Figure APNDX-5. Sample Format Appendage (Part 2 of 2) 325
Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 1 of 3) 326
Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 2 of 3) 327
Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 3 of 3) 328

Introduction

You may want to code special programs to supplement GTF and IMDPRDMP/EDIT operation. EDIT allows for two types of user programs: exit routines and format appendages. Neither type may occupy more than 10K bytes of main storage.

- An exit routine allows you to inspect each input trace record before EDIT begins processing it; on the basis of the inspection you must decide whether EDIT should process the record normally or take special action.
- A format appendage allows you to format all user trace records of a specified type. A format appendage must be named IMDUSRxx, where xx is the hexadecimal form of the format identifier (FID) specified in the GTRACE macro when the record was created.

This appendix is designed to help you write efficient, helpful user programs.

User Program Interfaces

A user program interfaces with the EDIT function of IMDPRDMP in the following ways:

Gaining Control

Until EDIT calls them, user programs reside in SYS1.LINKLIB or in a data set defined by the JOBLIB or STEPLIB DD statement. Once a user program is loaded into main storage, it remains there until EDIT processing is complete, or until it is deleted due to a need for space.

An exit routine is named in the EXIT= parameter of the EDIT control statement. It gets control every time EDIT reads an input trace record, and always completes its examination of the record before EDIT processes it.

A format appendage is invoked only when EDIT encounters a record that contains an FID field corresponding to the name of the format appendage. It remains in main storage until deleted, but only gets control when EDIT encounters a record with the corresponding FID.

Using the Parameter List

When EDIT passes control to a user program, register 1 contains the address of a parameter list. The contents of that parameter list, and its related fields are shown in figure APNDX-1. The exit routine or format appendage uses the parameter list to find the record it is to process, determine how to process it, and decide where to put the processed record.

Input record

As shown in Figure APNDX-1, the first four bytes of the parameter list give the address of the input record. Four-byte fields at offset 12 and 16, respectively, point to the event identifier (EID) field and the data area in the input record.

For a complete description of the input record format, see Figure GTF-8 in Chapter 3: GTF (Generalized Trace Facility).

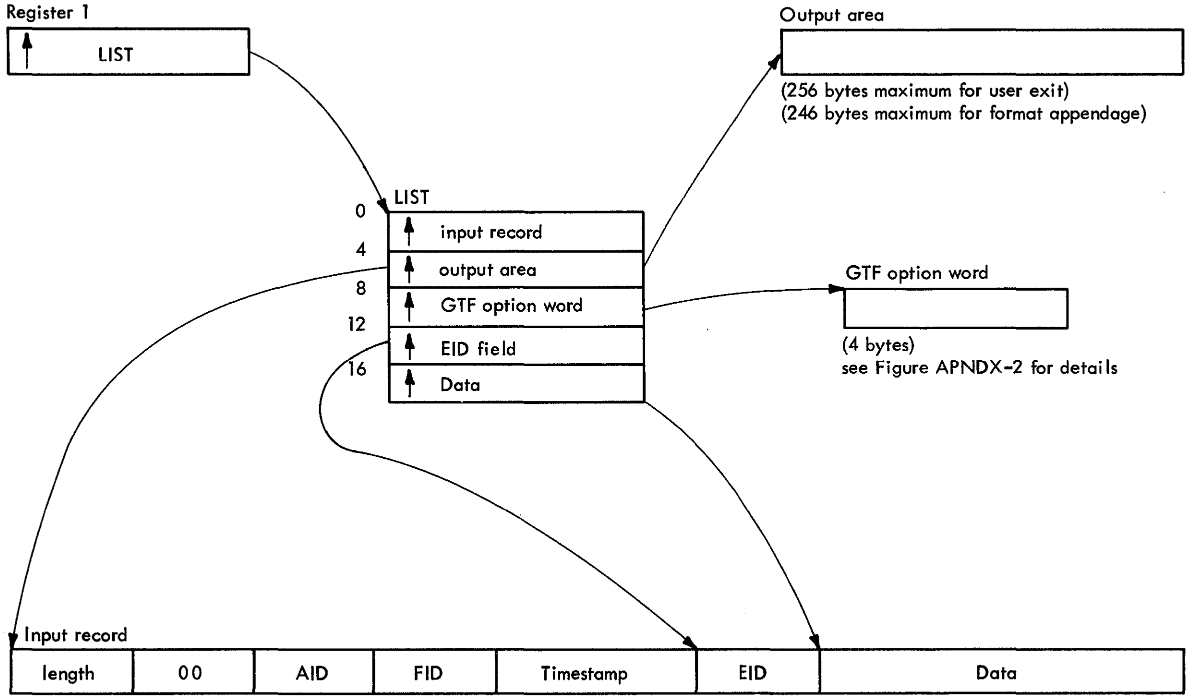


Figure APNDX-1. EDIT Parameter List and Related Fields

GTF Option Word

A four-byte field at offset 8 in the parameter list gives the address of the GTF option word, a four-byte table that summarizes the GTF options in effect when the input trace records were produced. Figure APNDX-2 lists the contents of the GTF option word.

BYTES	BITS	OPTIONS IN EFFECT DURING TRACE
Byte 1	1...	SYSM-- minimal tracing for system events
	.1..	SYSP-- maximum tracing, prompting requested.
	..1.	SYS-- maximum tracing for system events
	...1	USR -- all GTRACE-generated interrupts traced
	.... 1...	TRC -- all GTF interrupts traced
	.... .1..	DSP -- all task-switches traced
	.... ..1.	SSM -- all SSM interrupts traced (MP only)
	1	PCI -- program-controlled interrupts traced
Byte 2	1...	SVC -- all SVC interrupts traced
	.1..	SVCP -- SVC interrupts selected by prompting
	..1.	SIO -- all SIO events traced
	...1	SIOP -- SIO events selected by prompting
	.... 1...	PI -- all program interrupts traced
	.... .1..	PIP -- program interrupts selected by prompting
	.... ..1.	IO -- all I/O interrupts traced
	1	IOP -- I/O interrupts selected by prompting
Byte 3	1...	EXT -- external interrupts traced
	.xxx xxx.	reserved bits
	1	IO=SIO -- identical devices selected for IO & SIO
Byte 4	1...	tracing system - MFT
	.1..	tracing system - MFT with ATTACH
	..1.	tracing system - MVT
	...1	tracing system - Model 65 Multiprocessing
	.... 1...	real Monitor Call instruction
	.... 0...	simulated monitor call instruction
	.... .1..	no timer option selected at SYSGEN
	.... ..1.	Tracing system has time-of-day clock
	1	user timestamp requested

Figure APNDX-2. Contents of GTF Option Word, showing GTF Options in Effect During Trace

For more information about any of the GTF options, refer to Chapter 3, GTF (Generalized Trace Facility).

Returning to EDIT

A user program must return to EDIT with one of the return codes listed below. If EDIT recieves an invalid return code from a user program, it takes action as specified by the ER= subparameter of the PARM= parameter of the EXEC statement that invokes IMDPRDMP. This parameter, its values and their meanings are described in Chapter 8: PRDMP in the section "Job Control Language Statements".

Exit Routine Return Codes

An exit routine must return to EDIT with one of the following return codes:

Code Meaning

0 EDIT should print the contents of the output area, clear the area, and return immediately to the exit routine. This allows the exit routine to print more than one line of output. (Note that the output buffer may be in a different location when the format appendage receives control again.)

4 EDIT should print the contents of the output area and obtain the next logical record.

8 EDIT should format and print the trace record according to the selectivity specified in the EDIT control statement.

12 EDIT should obtain the next logical input trace record without printing the contents of the output buffer.

16 EDIT should print the contents of the output buffer and no longer invoke the exit routine, which is no longer needed.

20 EDIT should format and print the trace record according to the selectivity specified in the EDIT control statement, and should no longer invoke the exit routine, which is no longer needed.

24 EDIT should terminate processing and return control to IMDPRDMP so that the next IMDPRDMP control statement may be processed.

28 EDIT should format and print this record as though no selectivity had been specified in the EDIT control statement.

Format Appendage Return Codes

A format appendage must return to EDIT with one of the following return codes:

Code Meaning

0 EDIT should print the contents of the output buffer and return immediately to the format appendage. (Note that the output buffer may be in a different location when the format appendage receives control again.)

4 EDIT should print the contents of the output buffer and obtain the next logical input trace record.

8 EDIT should obtain the next logical input trace record without printing the contents of the output buffer.

Handling Errors

EDIT is prepared to handle two types of errors: invalid return codes and program checks. Other types of errors and their consequences are discussed later in this appendix, in the section "Avoiding Unrecoverable Errors".

Errors in Finding or Loading a User Program

If EDIT cannot find or load a user program, it takes action as shown in Figure APNDX-3.

Error Input Type	Exit Routine Not Found	 Not Loaded	Format Appendage Not Found	 Not Loaded
Dump	A	A	B	B
Trace Data Set	A	A	B	A

Action A: EDIT terminates processing and returns control to IMDPRDMP, which obtains the next IMDPRDMP control statement.

Action B: EDIT dumps the associated record in hexadecimal and obtains the next input trace record. Any subsequent records that have the same FID will be dumped in hexadecimal.

Figure APNDX-3. IMDPRDMP/EDIT Actions in Response to Errors in Finding or Loading User Programs.

Invalid Return Codes and Program Checks

EDIT's action in response to invalid return codes and program checks depends on the value for ER= that you specify in the PARM= parameter of the EXEC statement that invokes IMDPRDMP. For an explanation of the valid values for ER=, refer to the section "Job Control Language Statements" in Chapter 10: IMDPRDMP.

Avoiding Unrecoverable Errors

As shown in the previous sections, EDIT can recover from two kinds of errors in a user program: invalid return codes and program checks. EDIT cannot protect you, however, against errors that you may generate, for example by performing I/O operations or issuing GETMAIN macro instructions. In fact, you should avoid issuing any SVCs in your user program. Ordinarily this is not difficult, since EDIT provides you with the ability to examine records, manipulate data, and request formatted output to be printed. If you must issue an SVC, EDIT will permit you to do so; you should be prepared, however, for possibly unpredictable results if an error occurs during an operation that you have requested by issuing an SVC.

Another error condition that EDIT cannot handle, but which you can avoid, arises when you assign IMDPRDMP too small a region. You must specify a region large enough to accomodate all of IMDPRDMP's work areas and buffers plus all format appendages that can be called plus any exit routine. If you do not do so, IMDPRDMP may delete one or more user programs already in main storage to make room for a new one.

Deletion is critical if the deleted program issues an OPEN because the reinitialization that is necessary when the program is reloaded can cause two DCBs to be open at the same time. Deletion is also critical if the deleted program is an exit routine that sets a switch before relinquishing control and tests the same switch when it gets control again. Resulting errors may not cause abnormal termination, but they can prevent successful operation of the exit routine.

If none of your user programs will be damaged by deletion, you need not allow extra space for them in IMDPRDMP's region. IMDPRDMP's minimum region size includes 10K for use by system format appendages and user programs.

If your user program must issue a GETMAIN macro, be sure to specify a region large enough to include the amount of main storage requested in the user program. Also be sure to reserve that amount of storage for your own use by means of the FREEnnn subparameter of the PARM= parameter in the EXEC statement. If you do not reserve it, IMDPRDMP will make available to your program only a limited amount of storage and your GETMAIN may fail. For more information about the FREEnnn parameter, refer to the section "Job Control Language Statements" in Chapter 10: IMDPRDMP.

On completion of your user program, be sure to issue a FREEMAIN macro for all storage that you reserved for your own use. If you do not do so, and your program is deleted, the storage you reserved will remain allocated to you and thus unavailable to subsequent users.

A few examples may further clarify the areas in which EDIT does not provide error recovery:

- A user program, known as module A, issues the LINK SVC for module B. A program check occurs in module B. EDIT will attempt error recovery, since the error is a program check, but it knows nothing about module B. Therefore when it produces diagnostic information it will give the entry point of module A as the entry point of the failing module, and attribute the registers at the time of the program check to module A.

- A user program issues the OPEN SVC (SVC X'13') unsuccessfully and is posted with a system completion code of 213. EDIT cannot recover, so EDIT, the user program and IMDPRDMP will all be terminated.

- A user program opens a DCB. Before it can close the DCB, the program is deleted to make room for another user program. When the deleted program is reloaded, it creates a new DCB and opens it. Thus there are two open DCBs with the same name in storage at the same time. The operating system will not tolerate this situation, so the user program is abnormally terminated.

- A user program issues the SPIE SVC, thereby nullifying EDIT's SPIE routine. As a result any program checks in the user program that EDIT would normally handle will go through the user's own SPIE routine, perhaps with unpredictable results.

Sample User Exit Routine

Figure APNDX-4 shows a sample exit routine. This routine, named ABENDXIT, was written to aid diagnosis of an abnormal termination condition in a particular job. It scans each input trace record, suppressing printing until it finds a record with the specified jobname. When it finds such a record, ABENDXIT signals IMDPRDMP to print that record. All subsequent records will be printed until ABENDXIT encounters an SVC 13 record for the specified jobname; then ABENDXIT instructs IMDPRDMP to print that record and terminate.

Note that this program decides how to treat each new record on the basis of the way it treated previous records. To do this it must maintain certain switches intact between records, and as a result this program is not serially reusable. To guarantee the integrity of the switches in the program, therefore, it is necessary to specify a region large enough to hold both IMDPRDMP and the user exit routine contiguously. This is the only way to make sure that the exit routine will not be deleted if EDIT needs more room to execute.

```
**********************************************************************
*  ABENDXIT IS AN EDIT USER EXIT ROUTINE DESIGNED TO CONTROL PRINTING
*  OF ALL GTF RECORDS ASSOCIATED WITH A PROGRAM THAT HAS
*  PROGRAM CHECKED AND ABENDED
**********************************************************************
ABENDXIT CSECT
*   EQUATE STATEMENTS
FRSTREG       EQU  0
PARMREG       EQU  1
EIDREG        EQU  2
DATAREG       EQU  3
WORKREG       EQU  4
CHAINREG      EQU  9
BASE          EQU  12
SAVEPTR       EQU  13
RETPTR        EQU  14
CODEREG       EQU  15
         STM  RETPTR,BASE,12(SAVEPTR)  STORE REGISTERS
         BALR BASE,0                   ESTABLISH ADDRESSABILITY
         USING *,BASE                  USING REGISTER 12
         ST   SAVEPTR,SAVE+4           BACKWARD CHAINING
         LA   CHAINREG,SAVE            MY SAVE AREA POINTER
         ST   CHAINREG,8(SAVEPTR)      FORWARD CHAINING
         LR   SAVEPTR,CHAINREG         REG 13 ADDRESSES SAVE AREA
```

Figure APNDX-4. Sample Exit Routine. (Part 1 of 3)

```
         IMDMEDIT                      SYMBOLIC EID MACRO
+*/*****************************************************************/
+*/*  THE IMDMEDIT MACRO MAPS THE EID VALUES ASSOCIATED WITH IBM     */
+*/*  SYSTEM AND SUBSYSTEM EVENTS.  THE STORAGE FOR ANY OR ALL OF    */
+*/*  THE MAPPED VALUES MUST BE CONTAINED IN THE MODULE REFERENCING */
+*/*  THE DESIRED EIDS.  IMDMEDIT IS DESIGNED TO BE USED BY IBM-     */
+*/*  SUPPLIED FORMAT APPENDAGES, AND USER-SUPPLIED USER EXIT        */
+*/*  MODULES.                                                       */
+*/*****************************************************************/
+IMDMPCI EQU  X'2FDF' PCI I/O INTERRUPT
+IMDMSVC EQU  X'3FFF' SVC INTERRUPT
+IMDMDSP EQU  X'4FE7' TASK SWITCH
+IMDMIO1 EQU  X'5FEE' I/O INTERRUPT
+IMDMIO2 EQU  X'5FEF' I/O INTERRUPT
+IMDMSIO EQU  X'5FF0' SIO OPERATION
+IMDMSSM EQU  X'DFFC' SSM INTERRUPT
+IMDMPI  EQU  X'DFFD' PROGRAM INTERRUPT
+IMDMEXT EQU  X'DFFE' EXTERNAL INTERRUPT
+IMDMDMA1 EQU X'EFFF' OPEN/CLOSE/EOV
         TM   TERMSW,X'01'         Q/HAS TERMINATION BEEN REQSTD
         BC   1,FINISH             YES,TELL EDIT TO TERMINATE
         L    EIDREG,12(PARMREG)   GET POINTER TO EID
         L    DATAREG,16(PARMREG)  GET POINTER TO DATA(JOBNAME)
         TM   PRINTSW,X'01'        Q/HAS JOBN ALREADY BEEN FOUND
         BC   1,PRINTALL           YES, SO PRINT THIS RECORD
         LA   WORKREG,0            GET ZERO CONSTANT
         C    WORKREG,ECB1         Q/HAS THIS ECB BEEN POSTED
         BC   7,MYJOBLAB           YES, CHECK IF JOBN FOUND
         WTOR 'SPECIFY 8-CHARACTER JOBNAME OF ABENDING PROGRAM',
              MYJOBN,8,ECB1
         WAIT ECB=ECB1
         LA   WORKREG,MYJOBN       ADDRESS OF JOBNAME SELECTED
         OC   0(8,WORKREG),BLANKS CONVERT LOWER-CASE CHARS TO
*                                  UPPER CASE
MYJOBLAB CLC  0(8,DATAREG),MYJOBN Q/IS THIS MY JOBNAME
         BC   7,NOPRINT            NO -- JUST RETURN
*   ONCE JOBNAME FOUND| SET SWITCH AND PRINT ALL RECORDS UNTIL
*   ENCOUNTER AN SVC 13 (ABEND) CONTAINING THIS JOBNAME
         OI   PRINTSW,X'01'        TURN ON JOBNAME FOUND SWITCH
PRINTALL CLC  0(2,EIDREG),SVCEID   Q/ IS THIS AN SVC RECORD
         BC   7,PRINTREC           NO, SO PRINT AND CONTINUE
         CLI  15(DATAREG),X'0D'    Q/IS THIS AN SVC 13 (ABEND)
         BC   7,PRINTREC           NO, SO PRINT AND CONTINUE
         CLC  0(8,DATAREG),MYJOBN Q/IS THIS MY JOBNAME
         BC   7,PRINTREC           NO, SO PRINT AND CONTINUE
EXIT     OI   TERMSW,X'01'         INDICATE THAT THIS IS LAST
*                                  RECORD TO BE PRINTED
PRINTREC LA   CODEREG,8            FORMAT AND PRINT THIS RECORD
         L    SAVEPTR,4(SAVEPTR)   RESTORE SAVE AREA POINTER
         L    RETPTR,12(SAVEPTR)   RESTORE REGISTER 14
         LM   FRSTREG,BASE,20(SAVEPTR) RESTORE OTHER REGS EXCEPT 15
         BCR  15,RETPTR            RETURN TO EDIT
```

Figure APNDX-4. Sample Exit Routine (Part 2 of 3)

```
FINISH   LA    CODEREG,24           TERMINATE EDIT PROCESSING
*                                   SINCE SVC 13 WAS LAST RECORD
         B     RETURN               RESTORE REGISTERS AND RETURN
NOPRINT  LA    CODEREG,12           IGNORE RECORD
         B     RETURN               RESTORE REGISTERS AND RETURN
SAVE     DC    18F'0'               SAVE AREA
SVCEID   DC    AL2(IMDMSVC)         ESTABLISH REAL AREA FOR
*                                   EID FROM IMDMEDIT MAP MACRO
TERMSW   DC    X'00'                INDICATION TO REQUEST TERM
PRINTSW  DC    X'00'                JOBN FOUND, SO PRINT REC IND
ECB1     DC    F'0'                 FOR POST
MYJOBN   DC    C'        '          PLACE FOR OPR TO PUT JOBNAME
BLANKS   DC    C'        '          TO CONVERT LOWER TO UPPER CASE
         END
/*
```

Figure APNDX-4. Sample Exit Routine. (Part 3 of 3)

Some instructions in the sample exit routine require special attention. These are shaded in Figure APNDX-4, and they are discussed below.

IMDMEDIT

This mapping macro expands, as shown, into a list of equate statements that supply symbolic names for the event identifiers (EIDs). You should use the symbolic name in your program; this is your protection against program failure, if for any reason, the EID values are later changed.

TM TERMSW,X'01'

This instruction tests a switch to determine a course of action. Because of instructions like these, which any user exit is likely to use, you should always make sure your region is large enough so that the user exit need not be deleted at any time during EDIT execution.

L EIDREG,12(PARMREG)

L DATAREG,16(PARMREG)

These two instructions access the EDIT parameter list. (See Figure APNDX-1.)

WTOR 'SPECIFY 8-CHARACTER JOBNAME OF ABENDING PROGRAM', MYJOBN,8,ECB1

This instruction requests information that cannot be obtained from the EDIT parameter list. You can use a WTOR to request any information that the operator is likely to have, such as the EDIT options in effect. Note, however, that when you issue an SVC in a user program you risk abnormal termination if an error occurs during the SVC operation. For more information about this point, refer to the section "Avoiding Unrecoverable Errors" eariler in this chapter.

SVCEID DC AL2(IMDMSVC)

This establishes a main storage location for the value equated to IMDMSVC in the expansion of the IMDMEDIT mapping macro.

Sample Format Appendage

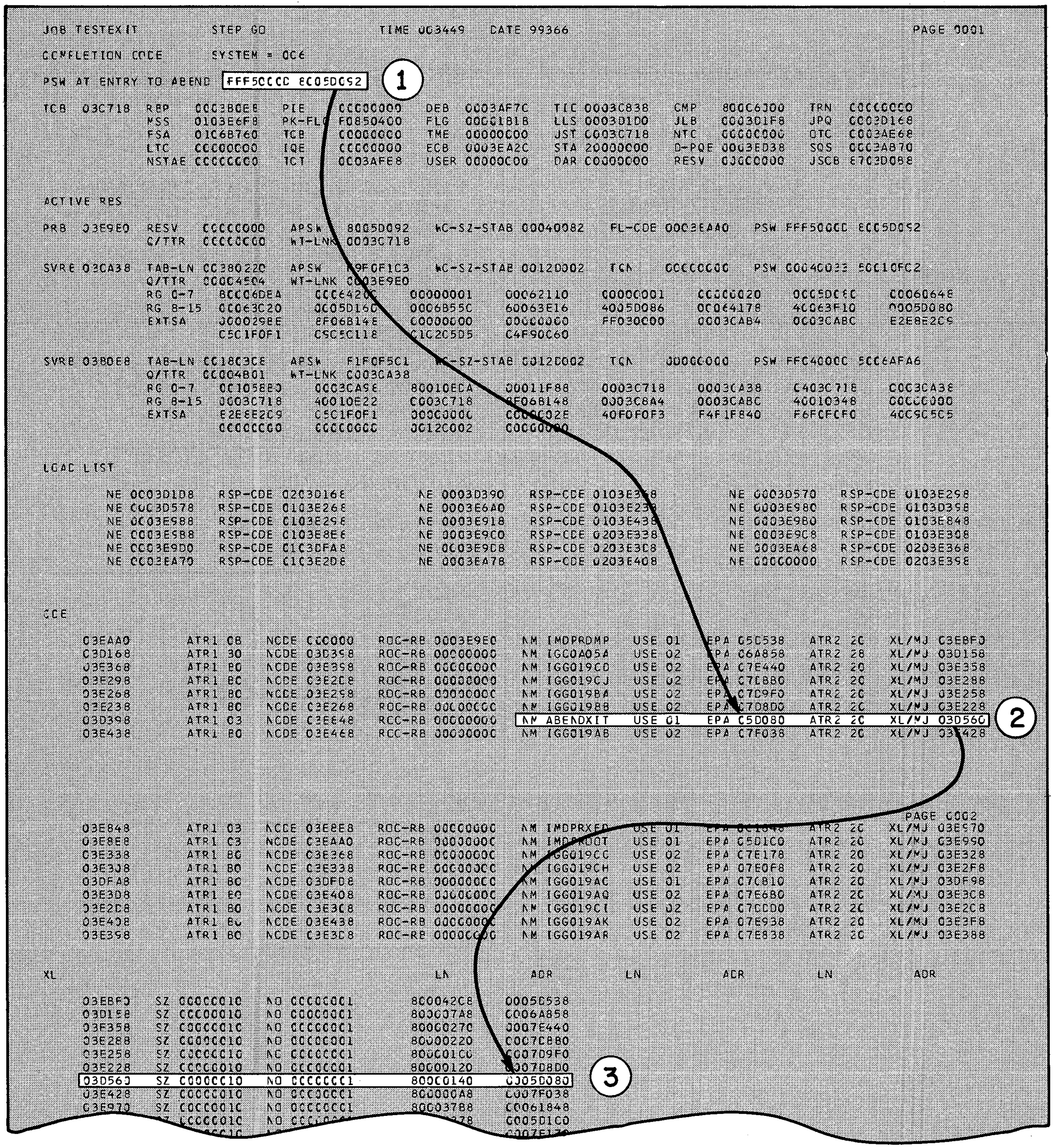
Figure APNDX-5 shows how to use the EDIT parameter list and how to handle multiple EIDs. It consists of excerpts from a sample format appendage named IMDUSR01, which formats three different types of user records. For each record IMDUSR01 produces two lines of output. The first line varies according to the record type. The second line is the same for all records.

```
**********************************************************************
*  IMDUSR01 IS AN EDIT USER FORMAT APPENDAGE MODULE THAT PROCESSES
*  THREE DIFFERENT TYPES OF INPUT RECORDS, THUS, THREE DIFFERENT EIDS.
*  LINE ONE OF THE FORMATTED OUTPUT VARIES ACCORDING TO THE EID.  LINE
*  TWO OF THE FORMATTED OUTPUT IS THE SAME FOR ALL EIDS, AND IS
*  PRODUCED IN COMMON CODE.
**********************************************************************
IMDUSR01 CSECT
*   EQUATE STATEMENTS
FRSTREG  EQU  0
PARMREG  EQU  1
EIDREG   EQU  2
DATAREG  EQU  3
CHAINREG EQU  9
BASE     EQU  12
SAVEPTR  EQU  13
RETPTR   EQU  14
CODEREG  EQU  15
         STM  RETPTR,BASE,12(SAVEPTR)  STORE REGISTERS
         BALR BASE,0                   ESTABLISH ADDRESSABILITY
         USING *,BASE                  USING REGISTER 12
         ST   SAVEPTR,SAVE+4           BACKWARD CHAINING
         LA   CHAINREG,SAVE            MY SAVE AREA POINTER
         ST   CHAINREG,8(SAVEPTR)      FORWARD CHAINING
         LR   SAVEPTR,CHAINREG         REG 13 ADDREESES SAVE AREA
         L    EIDREG,12(PARMREG)       GET POINTER TO EID
         L    DATAREG,16(PARMREG)      GET POINTER TO FIRST LINE DATA
         TM   SWITCH,X'01'             Q/ HAS FIRST LINE BEEN OUTPUTTED
         BC   1,LINETWO                YES, BRANCH TO FORMAT LINE TWO
*                  WHICH IS COMMON TO ALL THREE EID RTNS
         CLC  0(2,EIDREG),EID1         NO--Q/IS THIS A RECORD WITH EID1
         BC   8,RTN1                   YES--FORMAT LINE ONE
         CLC  0(2,EIDREG),EID2         Q/IS THIS A RECORD WITH EID2
         BC   8,RTN2                   YES--FORMAT LINE ONE
         CLC  0(2,EIDREG),EID3         Q/IS THIS A RECORD WITH EID3
         BC   8,RTN3                   YES--FORMAT LINE ONE
         LA   CODEREG,8                NO--IF NONE OF THESE EIDS, IGNORE
         B    RETURN                   REC, RESTORE REGS, AND RETURN
     .
     .
     .
RTN1
     .
     .
     .
         B    ZEROCODE                 SET ZERO RETURN CODE
```

Figure APNDX-5. Sample Format Appendage (Part 1 of 2)

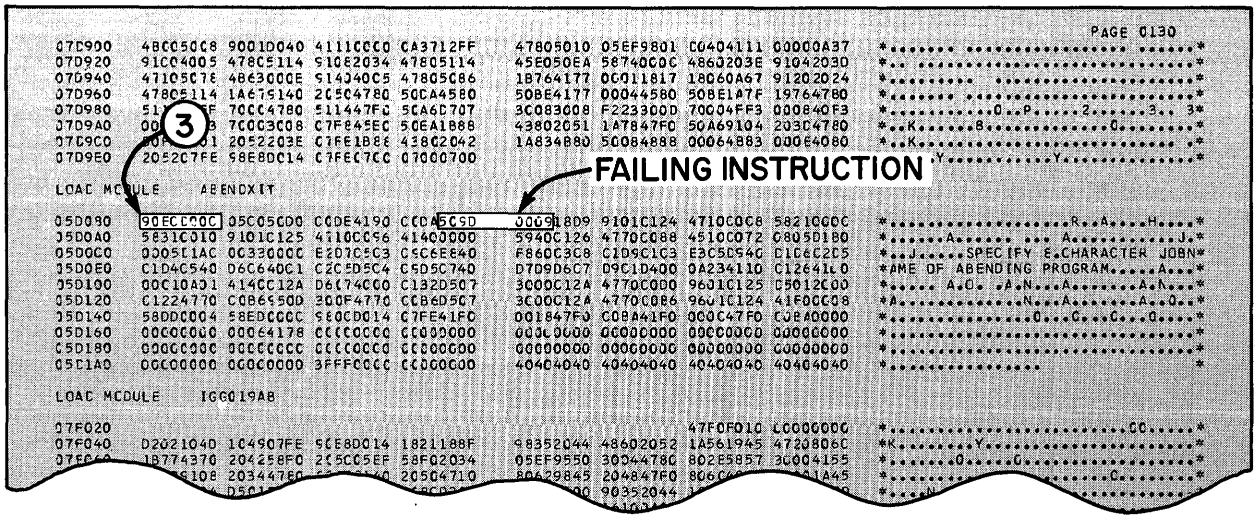
```
RTN2
    .
    .
    .
         B    ZEROCODE                 SET ZERO RETURN CODE
RTN3
    .
    .
    .
ZEROCODE OI   SWITCH,X'01'             FIRST LINE COMPLETE INDICATOR
         SR   CODEREG,CODEREG          OUTPUT THIS LINE AND RETURN
*                  IMMEDIATELY TO THIS FORMAT APPENDAGE
         B    RETURN                   RESTORE REGISTERS AND RETURN
LINETWO
    .
    .
    .
         NI   SWITCH,X'FE'             TURN OFF LINE 2 INDICATOR
         LA   CODEREG,4                OUTPUT THIS LINE--COMPLETE
RETURN   L    SAVEPTR,4(SAVEPTR)       RESTORE SAVE AREA POINTER
         L    RETPTR,12(SAVEPTR)       RESTORE REGISTER 14
         LM   FRSTREG,BASE,20(SAVEPTR) RESTORE OTHER REGS EXCEPT 15
         BCR  15,RETPTR                RETURN TO EDIT

SAVE     DC   18F'0'                   REGISTER SAVE AREA
SWITCH   DC   X'00'                    READY FOR LINE TWO SWITCH
EID1     DC   X'E001'                  EID1
EID2     DC   X'E002'                  EID2
EID3     DC   X'E003'                  EID3
    .
    .
    .
         END
/*
```

Figure APNDX-5. Sample Format Appendage (Part 2 of 2)

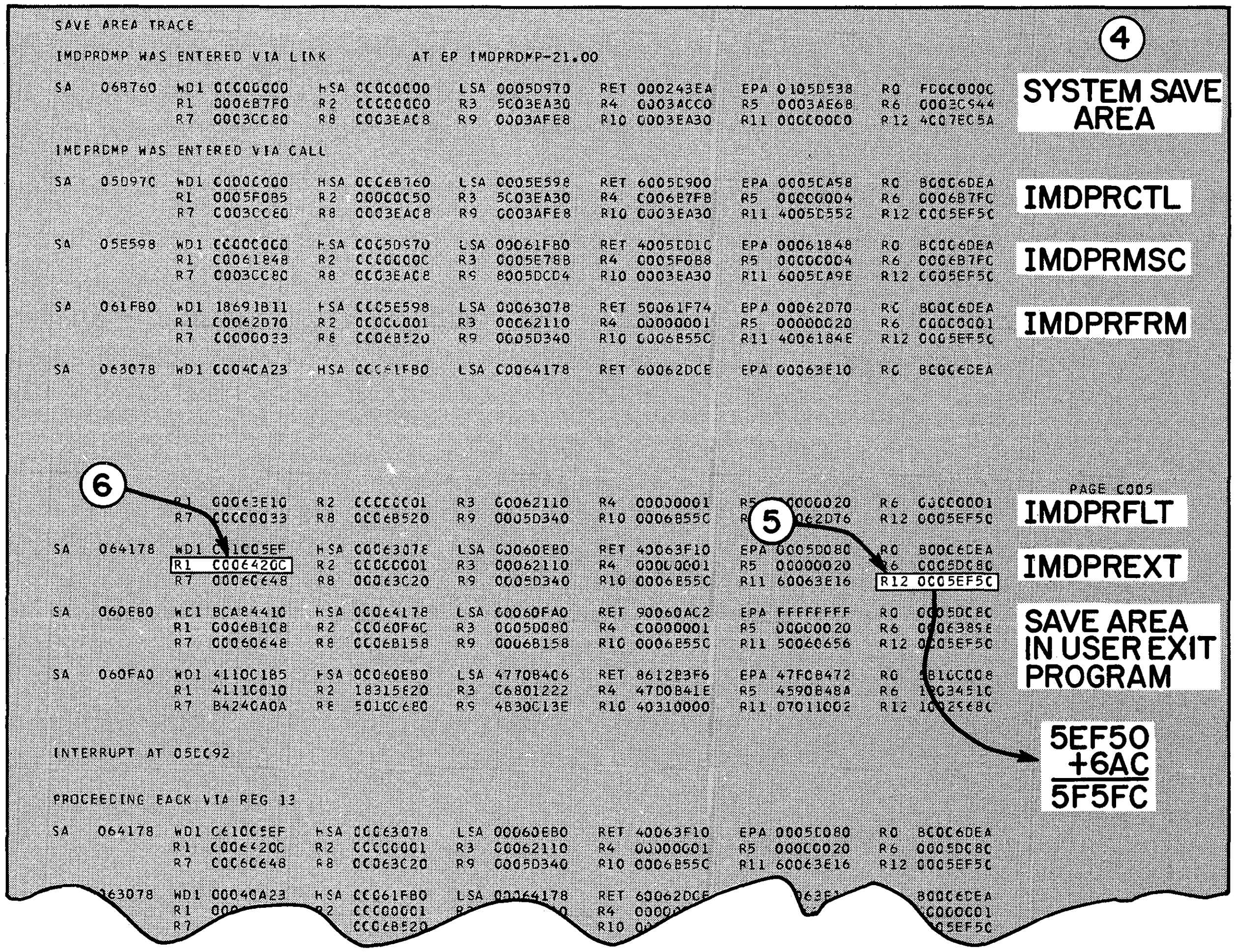


```
JOB TESTEXIT          STEP GO                 TIME 003449   DATE 99366                                          PAGE 0001

COMPLETION CODE       SYSTEM = 0C6

PSW AT ENTRY TO ABEND FFF5CCCD 8C05DC92

TCB  03C718  RBP    0003B0E8   PIE     CCC00000   DEB  0003AF7C   TIC 0003C838   CMP    800C6J00   TRN   CCCCCCCO
             MSS    0103E6F8   PK-FLG  F0850400   FLG  00CC1B18   LLS 0003D1D0   JLB    0003D1F8   JPQ   CC03D168
             FSA    01C6B760   TCB     C000000O   TME  00000C00   JST C003C718   NTC    CCC0C00C   OTC   C003AE68
             LTC    CCC00C00   IQE     CCCC0000   ECB  0003EA2C   STA 20000C00   D-PQE  0003ED38   SQS   0CC3AB70
             NSTAE CCCCCCCO    TCT     0C03AFE8   USER 00000C00   DAR C0J00000   RESV   0J0C0000   JSCB  8703D0B8


ACTIVE RBS

PRB  03E9E0  RESV   CCCCC000   APSW    8005D092   WC-SZ-STAB 00040082   FL-CDE 0003EAA0   PSW FFF500CD 8C05DC92
             Q/TTR  CCCCCCC0   WT-LNK  0003C718

SVRB 03CA38  TAB-LN CC38022C   APSW    F9F0F1C3   WC-SZ-STAB 0012D002   TCN    CCCCCCCC   PSW C0040C33 50C10FC2
             Q/TTR  C00C4504   WT-LNK  C003E9E0
             RG 0-7    8CCC6DEA    CCC6420C    00000001    00062110    00000001    CC0C0020    0CC5DC8C    C0060648
             RG 8-15   0C063C20    0C05D16C    0006B55C    60063E16    4005D086    0C064178    40C63F10    0005D080
             EXTSA     J00029BE    8F06B148    C0000000    00C00000    FF030C00    0003CAB4    0C03CABC    E2E8E2C9
                       C5C1F0F1    C9C5C118    C1C2C5D5    C4F90C60

SVRB 0380E8  TAB-LN CC18C3C8   APSW    F1F0F5C1   WC-SZ-STAB C012C002   TCN    0D00C000   PSW FFC4000C 5CC6AFA6
             Q/TTR  CC004B01   WT-LNK  C003CA38
             RG 0-7    0C1058E0    0003CA98    8C010ECA    00011F88    0003C718    0003CA38    C403C718    C0C3CA38
             RG 8-15   0003C718    40010E22    C003C718    8F06B148    0003C8A4    0003CABC    40010348    C0CCC000
             EXTSA     E2E8E2C9    C5C1F0F1    000C000C    C000C02E    40F0F0F3    F4F1F840    F6FCFCF0    40C9C5C5
                       CCCCCCCO    CCCCC0CC    0C12C002    C0CG0000


LOAD LIST

          NE 0C03D1D8   RSP-CDE 0203D168          NE 0003D390   RSP-CDE 0103E368          NE 0003D570   RSP-CDE 0103E298
          NE C0C3D578   RSP-CDE 0103E268          NE 0003E6A0   RSP-CDE 0103E238          NE 0003E98C   RSP-CDE 0103D398
          NE 0C03E988   RSP-CDE C103E298          NE 0003E918   RSP-CDE 0103E438          NE 0003E9B0   RSP-CDE 0103E848
          NE C003E9B8   RSP-CDE C103E8E8          NE 0003E9C0   RSP-CDE 0203E338          NE 0003E9C8   RSP-CDE 0103E308
          NE CC03E9D0   RSP-CDE C1C3DFA8          NE 0C03E9D8   RSP-CDE 0203E3C8          NE 0003EA68   RSP-CDE 0203E368
          NE CC03EA70   RSP-CDE C1C3E2D8          NE 0003EA78   RSP-CDE 0203E408          NE 000C0000   RSP-CDE 0203E398


CDE

    C3EAA0        ATR1 CB   NCDE CC0000   ROC-RB 0003E9E0   NM IMDPRDMP   USE 01   EPA C5C538   ATR2 2C   XL/MJ C3EBF0
    C3D168        ATR1 30   NCDE 03D398   ROC-RB 00000000   NM IGC0A05A   USE 02   EPA C6A858   ATR2 28   XL/MJ 03D158
    03E368        ATR1 B0   NCDE 03E358   ROC-RB C0C0C000   NM IGG019CD   USE 02   EPA C7E440   ATR2 20   XL/MJ 03E358
    03E298        ATR1 BC   NCDE C3E2C8   ROC-RB 00000000   NM IGG019CJ   USE 02   EPA C7CBB0   ATR2 2C   XL/MJ 03E288
    03E268        ATR1 BC   NCDE 03E298   ROC-RB C000000C   NM IGG019BA   USE 02   EPA C7D9F0   ATR2 2C   XL/MJ 03E258
    03E238        ATR1 B0   NCDE C3E268   ROC-RB 00C00CCC   NM IGG019BB   USE 02   EPA C7D8D0   ATR2 2C   XL/MJ 03E228
    03D398        ATR1 C3   NCDE C3E848   RCC-RB 00000000   NM ABENDXIT   USE 01   EPA C5D080   ATR2 2C   XL/MJ 03D560
    03E438        ATR1 B0   NCDE 03E468   RCC-RB 0000000C   NM IGG019AB   USE 02   EPA C7F038   ATR2 2C   XL/MJ 03E428

                                                                                                           PAGE C002
    03E848        ATR1 03   NCDE 03E8E8   ROC-RB 00CC000C   NM IMDPRXFD   USE 01   EPA 0C1048   ATR2 2C   XL/MJ 03E970
    03E8E8        ATR1 C3   NCDE C3EAA0   ROC-RB 0000000C   NM IMDPRDOT   USE 01   EPA C5D1C0   ATR2 20   XL/MJ C3E990
    03E338        ATR1 BC   NCDE C3E368   ROC-RB 0C00000C   NM IGG019CC   USE 02   EPA C7E178   ATR2 20   XL/MJ 03E328
    03E308        ATR1 B0   NCDE C3E338   ROC-RB C0C000CC   NM IGG019CH   USE 02   EPA C7E0F8   ATR2 20   XL/MJ 03E2F8
    03DFA8        ATR1 BC   NCDE C3DFD8   ROC-RB 000000CC   NM IGG019AC   USE 01   EPA C7C810   ATR2 20   XL/MJ 03DF98
    03E3D8        ATR1 E0   NCDE C3E408   ROC-RB 0J00C00C   NM IGG019AQ   USE 02   EPA C7E6B0   ATR2 2C   XL/MJ 03E3C8
    03E2D8        ATR1 B0   NCDE C3E3C8   ROC-RB 0000000C   NM IGG019CI   USE 02   EPA C7CDD0   ATR2 2C   XL/MJ 03E2C8
    03E408        ATR1 BC   NCDE 03E438   ROC-RB 0000000C   NM IGG019AK   USE 02   EPA C7E938   ATR2 20   XL/MJ 03E3F8
    03E398        ATR1 B0   NCDE C3E3C8   ROC-RB 0000C000   NM IGG019AR   USE 02   EPA C7E838   ATR2 2C   XL/MJ 03E388


XL                                                     LN          ADR              LN              ADR              LN              ADR

    03EBF0   SZ 0C0CC010   NO CCCCCCC1          8C0042C8   00050538
    03D158   SZ CCC00010   NO C000C0C1          800007A8   C006A858
    03F358   SZ C0CCC01C   NO CCCCC0C1          80000270   J007E440
    03E288   SZ C00CC010   NO CCCC0C01          80000220   C007CBB0
    03E258   SZ CCCCCC10   NC CCCCCCC1          800C01C0   C0070 9F0
    03E228   SZ CCCCC010   NO CCCCCCC1          80C00120   0007D8D0
    03D560   SZ C000CC10   NO CCCCCCC1          800C0140   0005D080
    03E428   SZ C0CCC010   NO 0CC0CCC1          800000A8   C007F038
    C3E970   SZ C0CC001C   NO 0CCCCCC1          80C03788   C0061848
             SZ CCCCC01C   NO CCC                     78   C005D1C0
                                                           C007E17
```

Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 1 of 3)

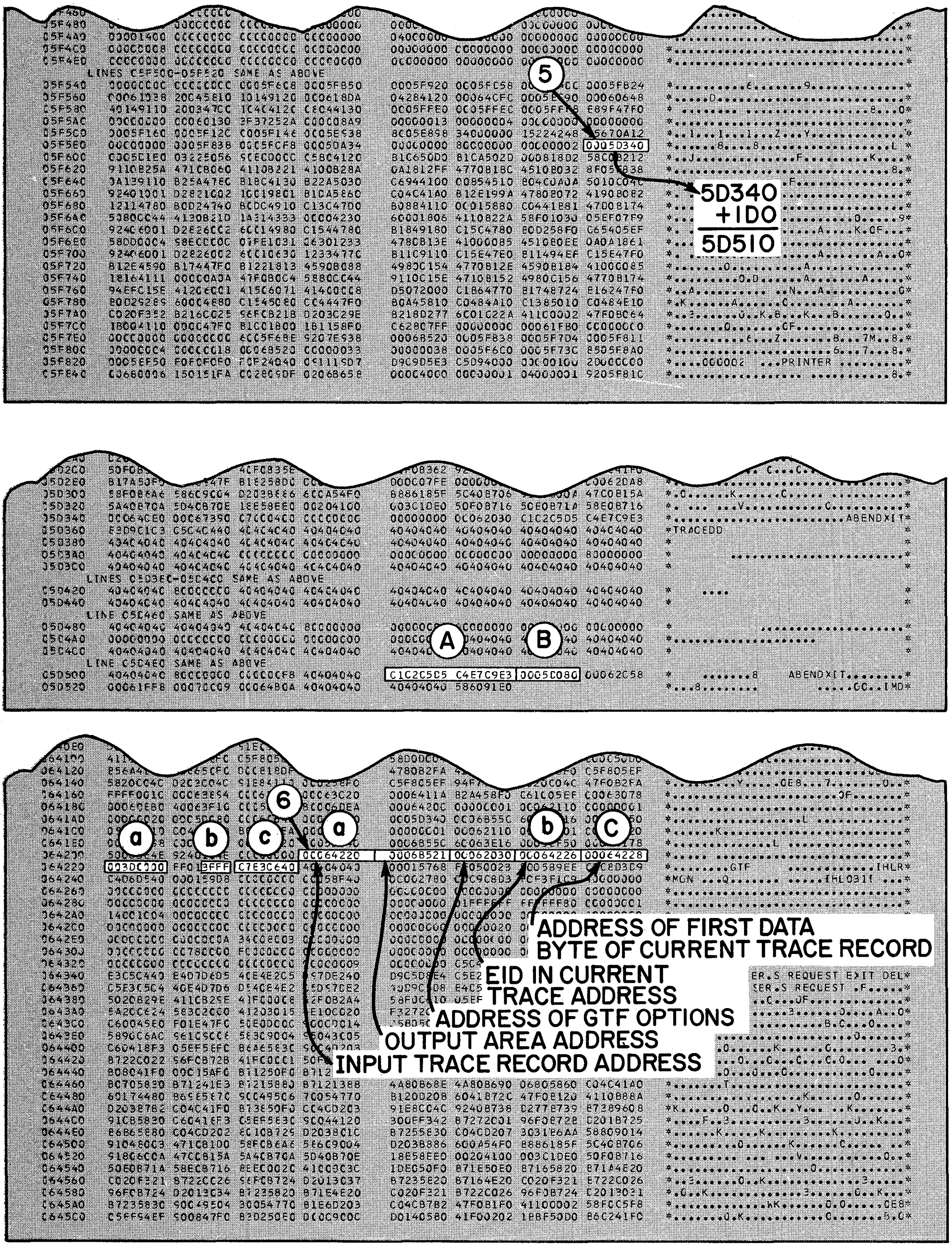

3
FAILING INSTRUCTION

```
                                                                                                  PAGE 0130
07D900   4BC050C8 9001D040 4111C0C0 CA3712FF    47805010 05EF9801 C0404111 00000A37   *....... .................. ......*
07D920   91C04005 478C5114 91082034 47805114    45E050EA 5874000C 4860203E 9104203D   *.. ..............................*
07D940   47105C78 4863000E 914040C5 47805C86    1B764177 00011817 18060A67 91202024   *.........  ......................*
07D960   47805114 1A675140 20504780 50CA4580    50BE4177 00044580 50BE1A7F 19764780   *....... .........................*
07D980   51     F 70C04780 511447FC 5CA6C707    3C083008 F223300D 70004FF3 000840F3   *...........0..P.....2......3.. 3*
07D9A0   00     8 7C003C08 C7F845EC 50EA1888    43802051 1A7847F0 50A69104 203C4780   *..K......8...............0.......*
07D9C0   0F    01 2052203E C7FE1B8E 43802042    1A834880 50084888 00064883 000E4080   *..K............................ .*
07D9E0   2052C7FE 98E8DC14 C7FEC7CC 07000700                                          *  .Y...........Y.............*

LOAD MODULE     ABENDXIT

05D080   90ECD00C 05C05CD0 C0DE4190 C0DA5C9D      00091809 9101C124 4710C0C8 5821000C   *........................R..A....H....*
05D0A0   5831C010 9101C125 4710C096 41400000    5940C126 4770C088 4510C072 0805D180   *......A...... ... A...........J.*
05D0C0   0005E1AC 0033000C E2D7C5C3 C9C6E840    F860C3C8 C1D9C1C3 E3C5D940 C1C6C2C5   *..J.....SPECIFY E.CHARACTER JOBN*
05D0E0   C1D4C540 D6C640C1 C2C5D5C4 C9D5C740    D7D9D6C7 D9C1D400 0A234110 C12641C0   *AME OF ABENDING PROGRAM.....A...*
05D100   00C10A01 414CC12A D6C74C00 C132D507    3000C12A 4770C0D0 9601C125 C5012C00   *..... A.0. .A.N...A.......A.N...*
05D120   C1224770 C0B6950D 300F4770 C0B6D507    3C00C12A 4770C0B6 9601C124 41F000C8   *A.............N...A.........A..0..*
05D140   58DC0004 58EDC00C 980CD014 C7FE41F0    001847F0 C0BA41F0 000C47F0 C0BA0000   *...................0...C...C...0....*
05D160   00000000 00064178 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
05D180   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
05D1A0   00000000 00000000 3FFF0000 00000000    40404040 40404040 40404040 40404040   *..............                  *

LOAD MODULE     IGG019A8

07F020                                                            47F0F010 C0000000   *.........................00......*
07F040   D202104D 104907FE 98E8D014 1821188F    98352044 48602052 1A561945 4720806C   *K........Y.......................*
07F060   18774370 204258F0 205C05EF 58F02034    05EF9550 30044780 802E5857 30004155   *.......0.....0...................*
          6108 20344780        20504710    80629845 204847F0 806C       01A45   *...........................C.......*
```


4
SYSTEM SAVE AREA
IMDPRCTL
IMDPRMSC
IMDPRFRM
IMDPRFLT
IMDPREXT
SAVE AREA IN USER EXIT PROGRAM
6
5
5EF50
+6AC
5F5FC

```
SAVE AREA TRACE

IMDPRDMP WAS ENTERED VIA LINK          AT EP IMDPRDMP-21.00

SA   068760  WD1 00000000   HSA 00000000   LSA 0005D970   RET 000243EA   EPA 0105D538   R0  FD000000C
             R1  0006B7F0   R2  00000000   R3  5C03EA30   R4  0003ACC0   R5  0003AE68   R6  0003C944
             R7  0003CC80   R8  0003EAC8   R9  0003AFE8   R10 0003EA30   R11 00C00000   R12 4C07EC5A

IMDPRDMP WAS ENTERED VIA CALL

SA   05D970  WD1 00000000   HSA 0006B760   LSA 0005E598   RET 6005C900   EPA 0005CA58   R0  B0006DEA
             R1  0005F0B5   R2  0000C050   R3  5C03EA30   R4  C006B7F8   R5  00000004   R6  0006B7FC
             R7  C003CC80   R8  0003EAC8   R9  C003AFE8   R10 0003EA30   R11 4005C552   R12 C005EF5C

SA   05E598  WD1 00000000   HSA 0005D970   LSA 00061FB0   RET 4005CD1C   EPA 00061848   R0  B0006DEA
             R1  C0061848   R2  0000000C   R3  0005E78B   R4  0005F0B8   R5  00000004   R6  0006B7FC
             R7  0003CC80   R8  C003EAC8   R9  8005DCD4   R10 0003EA30   R11 6005CA9E   R12 C005EF5C

SA   061FB0  WD1 18691B11   HSA 0005E598   LSA 00063078   RET 50061F74   EPA 00062D70   R0  B0006DEA
             R1  C0062D70   R2  00000001   R3  00062110   R4  00000001   R5  00000020   R6  00000001
             R7  00000033   R8  C006B520   R9  0005D340   R10 0006B55C   R11 4006184E   R12 0005EF5C

SA   063078  WD1 C004CA23   HSA 00061FB0   LSA C0064178   RET 60062DCE   EPA 00063E10   R0  B0006DEA

                                                                                            PAGE 0005
             R1  00063E10   R2  00000001   R3  00062110   R4  00000001   R5   0000020   R6  00000001
             R7  00000033   R8  0006B520   R9  0005D340   R10 0006B55C   R      062D76   R12 0005EF5C

SA   064178  WD1 C 1C05EF   HSA 00063078   LSA 00060EB0   RET 40063F10   EPA 0005D080   R0  B0006DEA
             R1  C006420C   R2  00000001   R3  00062110   R4  00000001   R5  00000020    6  C005DC8C
             R7  00060648   R8  00063C20   R9  0005D340   R10 0006B55C   R11 60063E16   R12 0C05EF5C

SA   060EB0  WD1 BCA8441C   HSA 00064178   LSA C0060FA0   RET 90060AC2   EPA FFFFFFFF   R0  0 05DC8C
             R1  G006B1C8   R2  CC060F6C   R3  0005D080   R4  C0000001   R5  00000020   R6  0 06389E
             R7  00060648   R8  C006B158   R9  0006B158   R10 0006B55C   R11 50060656   R12 0 05EF5C

SA   060FA0  WD1 4110C1B5   HSA 00060EB0   LSA 47708406   RET 8612B3F6   EPA 47F0B472   R0   B10C008
             R1  4111C010   R2  18315E20   R3  C6801222   R4  4700B41E   R5  4590B48A   R6  1 C3451C
             R7  B424CA0A   R8  501CC680   R9  4B30C13E   R10 40310000   R11 07011002   R12 1 0256BC

INTERRUPT AT 05CC92

PROCEEDING BACK VIA REG 13

SA   064178  WD1 C610C5EF   HSA 00063078   LSA 00060EB0   RET 40063F10   EPA 0005D080   R0  B0006DEA
             R1  C006420C   R2  00000001   R3  00062110   R4  00000001   R5  00000020   R6  0005DC8C
             R7  C0060648   R8  00063C20   R9  0005D340   R10 0006B55C   R11 60063E16   R12 0005EF5C

  063078     WD1 00040A23   HSA 00061FB0   LSA 00064178   RET 60062DCE      63E1          B000C6DEA
             R1  000            2  00000001                  R4  0000                     C000001
             R7                    C006B520                  R10 00                        05EF5C
```

Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 2 of 3)



```
 5F460                       CCCCCC                      00                                     CC0CC0
 5F480    0         00CCCC0C CCCC                    0000000    000C                     00C00000 000                *
05F4A0    00001400 CCCCCCCC CCCCCCCC CCCCC000    040C0000              CC00 CC000000 00000000    *.......................*
05F4C0    000C0008 CCCCCCCC CCCCCCCC CCCCC000    00000000 C0000000 00003000 C0000000    *................................*
C5F4E0    CCCCC000 0CCCCCCC CCCCCCCC CCCCC000    0C000000 00000000 00    CC CJ0C000C    *................................*
     LINES C5F500-05F520 SAME AS ABOVE
05F540    0000CCCC CCCCCCCC C005F6C8 0005FB50    0005F920 0C05FC58 00    CC 0005FB24    *..........6.......9.............*
05F560    C0061038 20C45810 10149120 0C0618DA    04284120 00064CFC 0005E 90 00060648    *.....D..........................*
C5F580    40149110 200347CC 1C4C412C C8044130    0C05FFE0 0C05FFEC 0C05FF 8 E89F47F0    * ...... ...............8...0*
05F5A0    00CC0000 CC060130 3F37252A C00C08A9    00000013 00000004 00C000    00000000    *................................*
05F5C0    0005F160 0005F12C C005F146 0C05E538    8C05E898 34000000 15224248  5670A12    *..1...1...1...Z...Y.............*
05F5E0    00C00000 0005F838 0CC5FCF8 0005DA34    00C00000 8C000000 00C00002 0005D340    *.....8....8...................L *
05F600    C005C1E0 03225056 9CECDOCC C58C412C    B1C650D0 B1CA502D 000818D2 58C0B212    *..J............F.........K....*
05F620    9110B25A 471C8060 41108221 4100828A    0A1812FF 4770818C 4510B032 8F05 838    *.............................8.*
C5F640    0A139110 825A478C B18C4130 B22A5030    C6944100 00854510 B04C0A0A 5010C04C    *  .............F...............*
05F660    92401001 D2821C02 1CC198C1 B1CA586C    C04C41A0 812E199A 4780B072 4190BC82    *. .............................*
05F680    12114780 B0D24740 BCDC4910 C13C47D0    B0884110 0C015880 C0441881 4700B174    *...............................*
05F6A0    5080CC44 4130B210 1A314333 0C004230    60001806 4110822A 58F01030 05EF07F9    *...................0.....9*
05F6C0    924C6001 D2826CC2 6CC14980 C1544780    B1849180 C15C4780 B0D258F0 C65405EF    *.  ..............A....K.0F...*
05F6E0    58DD00C4 58ECCC0C 07FE1031 06301233    4780B13E 41000085 451080EE 0A0A1861    *...............................*
05F700    92406001 D2826002 6CC10630 1233477C    B11C9110 C15E47E0 B11494EF C15E47F0    *. .............A.......A..0*
05F720    812E4590 817447FC B1221813 4590B088    4980C154 4770812E 45908184 4100C085    *.......0.........A.............*
05F740    18164111 000C0A0A 47F0B0C4 5880CC44    9110C15E 47108152 4980C156 4770B174    *.........0.D........A.......A.....*
05F760    94EFC15E 412C6CC1 415C6071 4140CCC8    D5072000 C1864770 B1748724 E16247F0    *..A........... ..N...A..........0*
05F780    B0D29289 600C4880 C1545080 CC4447F0    B0A45810 C0484A10 C1385010 C0484E10    *.K.......A......C.........A.......*
05F7A0    C020F352 B216C025 96FCB218 D203C29E    B218D277 6C01C22A 411C0002 47F0BC64    *..3.......0..K.B...K...B.......0..*
05F7C0    1B004110 000C47FC B1C01800 1B1158F0    C628C7FF 0000000C 00061FB0 CC0000C0    *........0.......0F..............*
05F7E0    00CC0000 C000CCCC 6CC5F68E 9207E938    00068520 0005F838 0005F7D4 0005F811    *...........6...Z.........8...7M..8.*
05F800    0000CCC4 CCCCCC18 0C068520 CC000033    00000038 0005F6C0 0005F73C 8505F8A0    *..........................6...7...8.*
05F820    0005EF50 F0F0F0F0 F0F24040 C9111907    D9C9D5E3 C5D94000 00C00100 20000000    *....000002  ...PRINTER ........*
C5F840    00680006 150151FA CC28C9DF 0206B658    000C4000 00000001 04000001 9205F81C    *................  ...........8.*
```

```
  2A0     D2
 5D2C0    50F0B                 4CF0835E           F0B362 92                          41F0     *.                  C...C.
 5D2E0    B17A50F        47F B1E258D0 C0           000C07FE 00000                   C0062DA8   *..........................*
 5D300    58F0B6A6 586C9C04 D203B8E6 6CCA54F0    B886185F 5C40B706            A 47C0815A    *.0.......K.....C..... ........*
 5D320    5A40B70A 5D40B70E 18E58EE0 00204100    003C1DE0 50F0B716 50E0B71A 58E0B716    *. ... ...V..........C.........*
 5D340    0C064CE0 00C67390 C7CC04C0 CCCC0C0C    00000000 0C062030 C1C2C5D5 C4E7C9E3    *......................ABENDXIT*
 5D360    E3D9C1C3 C5C4C440 4C4C4C40 40404C40    40404040 40404040 40404040 404C4040    *TRACEDD                       *
C5D380    404C4040 404C4040 4C4C404C 404C4C40    40404040 4040404C 40404040 40404040    *                              *
 5D3A0    40404040 404C4C4C CCCCCCCC C0000000    00C00000 0C000C00 00000000 80000000    *          ....................*
 5D3C0    40404040 404C404C 4C4C4040 4C4C4040    40404040 40404040 40404040 40404040    *                              *
     LINES C5D3EC-05D4CO SAME AS ABOVE
C5D420    4040404C 8C0CCCC0 40404040 404C404C    40404C40 4C404040 40404040 40404040    *      ...                     *
 5D440    40404C40 404C4040 4C404040 40404040    40404C40 40404040 40404040 40404040    *                              *
     LINE C5C460 SAME AS ABOVE
 5D480    404C404C 40404040 4C404C4C 8C000000    00000C   C000000 0      00 00C00000    *            ..................*
C5C4A0    00000000 0CCCCCCC CCC000C0 00C00C00    000CC      0404040 4       40 40404040    *................               *
 5C4C0    40404040 40404040 4C4C404C 4C404040    40404         0404040 4       4C 40404040    *                              *
     LINE C5C4E0 SAME AS ABOVE
C5D500    4040404C 8CCCC000 CCCC0CF8 4040404C    C1C2C5D5 C4E7C9E3 0005C080 00062C58    *       ......8     ABENDXIT.......*
 5D520    00061FF8 0007CC09 0C064B0A 40404040    40404040 586091E0                      *...8.......            .....CC..IMD*
```

```
  40E0    5                     91EC                                                                  *
 64100    411                 FC C5F805          58D0DC0                       C00C50D0    *C...........
 64120    856A4        C65CFC DCC818DF            4780B2FA 4                      F0 C5F805EF    *.................
 64140    5820CC4C D2C3C04C 91E84113 0C0258F0    C5F8C5EF 94F7           20C04C 47F0B2FA    *.. ....Y.....0E8...7.......0..*
 64160    FFFF001C 0GC63E94 CC6         0C063C20    0006411A B2A458F0 C61C05EF C0063078    *.......................0F.......*
 64180    00060EB0 40063F1C CC5          BC006DEA    0006420C 0000C001 0C062110 CG000001    *.... ..........................*
 641A0    000C0020 00C50680 CC                0C         0   0C05D340 CC06855C 6        16 0         50    *........................L......*
 641C0    0         10 C04               B             EA 000           0000CC01 00062110 0        01 C0        20    *......K........................*
C641E0    0           8 C0                0             20    0C05         C    0006B55C 6C063E16 00    F50 0         178    *.................L..............*
 64200    50            4E 9240     E C           C00 0C064220       J006B521 0C062030 0C064226 00064228    *.... ..........................*
 64220    003DC000 FF013FFF C7E3C640 40404043     00015768 F 050023   00589EE C C8D3C9    *........GTF      ..........IHLR*
 64240    C4D6D540 00015908 CCCCCCCC C 58F40    0C002780 C C9C8D3 CF3F1C9 0000000    *MON ...Q....... ....IHL031  ...*
C64260    09CCC000 6CCCCCCC CCCCCCCC C 000000    0C0C0C00 0 C00000 0000000 00000000    *...............................*
 64280    00CC0C00 0C0C0CC0 CCCCCCCC C C00000    00000000 01FFF FF FF FFF80 CC000001    *...............................*
 642A0    14001C04 00C0C0CC CCCCCCCC C C00000    0C003000 000C 000 00
 642C0    03C00000 000C0CCC CCCC0CC0 C 0CC000    00000000 00  0020 00
 642E0    0C0CCCC0 CUCC0CCA 34C08C03 C C00000    000C000  0C 00C00 00
 64300    03CCCCCC CC78CCC0 FC3C0CCC C CCC0000    000C000  0C 0000 00
 64320    0CCCCC00 CCCCCCCC CCCCCCCC  C0C0009    0C0C00  0 C5C
 64340    E3C5C440 E4D7D6D5 4CE4E2C5 597DE240    D9C5D E4 C5E2                                ER.S REQUEST EXIT DEL*
C64360    C5E3C5C4 40E4D7D6 D54CE4E2 5D97DE2     40D9C D8 E4C5                               SER.S REQUEST .F.....*
 64380    5020829E 411CB29E 41FC00C8 2F0B2A4     58F0C 10 05EF                               ..C...JF.............*
 643A0    5A20C624 583C2C00 41203015 E10C020     F3272C                                       .....3......J....A..0*
 643C0    C60045E0 F01E47FC 50E0D0C0 9C0CD014    15805C                                       ...........B.C....0.....*
 643E0    5890C6AC 961C9CCE 5E3C9004 95043C05                                                  .....................0.....*
 64400    C60418F3 05EF5EFC 8CA65E3C 9C04D203                                                  .0......K.......C.........3.*
 64420    B722C022 96FCB72B 41FC0CC1 50F                                                        ...0...C....C....C.0...  ...0*
 64440    B08041F0 00C15AFC B11250F0 B712                                                       .0...0...C...........C.....*
C64460    BC705830 B71241E3 B7215880 B7121388    4A80B68E 4A80B690 06805860 C04C41A0    *.......T........................*
C64480    60174480 B69E5E7C 9CC49506 7C054770    B120D20B 6041B72C 47F0E120 4110B88A    *................K......0.........*
C644A0    D203B782 C04C41F0 B73E50F0 CC4CD203    91E8C04C 9240B738 D277B739 B7389608    *K.......0....0..K...Y...  ..K.......*
 644C0    91CB5830 C60416F3 C5EF583C 9C044120    300EF342 87272C01 96F08728 D201B725    *....F..3..........3.......J...K....*
 644E0    86B65880 C04CD202 8C10B729 D2038C1C    B7255830 C04CD207 3031B6AA 58809014    *......K......K..........K.........*
C64500    91048003 471C8100 58FCB6A6 586C9004    D203B886 600A54F0 B886185F 5C40B706    *...........0.......K........C.....  ..*
 64520    918C6C0A 47CCB15A 5A4CB70A 5D40B70E    18E58EE0 00204100 003C1DE0 50F0B716    *........ ... ...V.............0..*
C64540    50E0B71A 58ECB716 8EEC002C 41C03C3C    1DEC50F0 B71E50E0 B7165820 B71A4E20    *..................0................*
C64560    C020F321 B722CC26 96FCB724 D2013C37    B7235820 B7164E20 C020F321 B722C026    *..3.........0..K...............3......*
C64580    96F0B724 D2013C34 B7235820 B71E4E20    C020F321 B722C026 96F0B724 D2013C31    *.0..K..................3.........G..K....*
C645A0    B7235830 90C49504 30054770 B1E6D203    C04CB782 47F081F0 41100002 58FCC5F8    *................WK..........0.0.......0E8*
C645C0    C5EF94EF 900847FC B2D250E0 D0C0900C    D0140580 41F00202 1BBF50D0 B6C241FC    *........0.K..........0........B.0*
```

5
5D340
+1D0
5D510
A
B
6
a
b
c
a
b
c
ADDRESS OF FIRST DATA BYTE OF CURRENT TRACE RECORD
EID IN CURRENT TRACE ADDRESS
ADDRESS OF GTF OPTIONS
OUTPUT AREA ADDRESS
INPUT TRACE RECORD ADDRESS

Figure APNDX-6. Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT (Part 3 of 3)

Debugging a User Program

Figure APNDX-6 shows a sample ABEND dump of the user exit routine ABENDXIT, shown in Figure APNDX-5. Certain important fields are highlighted in the figure and marked with numbers; the numbers refer to the explanations below:

1. PSW for the abnormally terminating program. The address in the second half of the PSW is an address in the abnormally terminated program. To find the entry point and name of the program, compare this address to the entry point addresses in the contents directory entry list. The abnormally terminating program is the one whose entry point address is closest to and greater than the address in the PSW.

   NOTE: If the address in the PSW does not immediately indicate the entry point address of the failing program, you can locate the beginning address of the abnormally terminating program by tracing IMDPRDMP's save area chain. See point 4, below.

2. Part of a contents directory entry (CDE). This shows the name of the abnormally terminating program, ABENDXIT, its entry point, X'05D080', and the pointer to the appropriate entry in the extent list.

3. An extent list entry. This shows the beginning address (not necessarily the entry point) of the abnormally terminating program. Subtract this address from the address in the PSW to find the address of the instruction following the instruction that failed.

   For example, in this case:

   address in PSW - beginning address = offset (hex)

   5D092 - 5D080 = 12

   The failing instruction in ABENDXIT can be found at offset X'12' in the program. (See part 2 of Figure APNDX-6, number 3.)

4. The first save area in the save area trace table (system save area) is chained to the following IMDPRDMP module save areas:

   IMDPRCTL - IMDPRDMP control routine
   IMDPRMSC - IMDPRDMP scan routine
   IMDPRFRM - EDIT control routine
   IMDPRFLT - EDIT trace record selection routine
   IMDPREXT (or IMDPRAPP) - EDIT user program selection routine.

5. The user program's registers are stored in IMDPREXT's or IMDPRAPP's save area. Add the contents of register 12 to X'6AC' to get the address of a fullword that points to an EDIT communication table. At offset X'1D0' into this table are the following:

   A. The 8-byte EBCDIC name of the current user program (the failing program).

   B. The entry point address of the current user program (the failing program).

These fields are shown in part 3 of Figure APNDX-6.

6. Register 1 in IMDPREXT's or IMDPRAPP's save area points to the parameter list that EDIT passes to the user program. (See Figure APNDX-1.)

JCL and Control Statement Examples

The following examples show how to test a user program.

Example 1: Link Editing a User Exit Routine into a Library

This example shows how to make a user exit routine available to IMDPRDMP by link-editing it into a system library.

```
//LKUSRPGM     JOB       MSGLEVEL=(1,1)
//             EXEC      PGM=IEWL,PARM='XREF,LET,LIST,NCAL',
//         REGION=96K
//SYSPRINT     DD        SYSOUT=A
//SYSLMOD      DD        DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSLIN       DD        *
    object deck
         NAME       EXITNAME
/*
```

In this example:

EXEC Statement

invokes the linkage editor and requests maximum diagnostic listings.

SYSPRINT DD Statement

defines the message data set.

SYSLMOD DD Statement

defines the output data set, in this case the linkage library, SYS1.LINKLIB. The output data set can also be a permanent library to be invoked later by a JOBLIB or STEPLIB DD statement; in that case the SYSLMOD DD statement should be coded as follows:

```
//SYSLMOD DD   DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//             DISP=(NEW,KEEP),SPACE=(1024,(20,2,1))
```

SYSLIN DD Statement

defines the input data set, in this case, the object deck for the user program.

NAME Control Statement

specifies the member name, and thus the program name, to be assigned to the user program. In this case, the member name is EXITNAME; to invoke this program in a later execution of IMDPRDMP, you would have to speciy EXIT=EXITNAME on the EDIT control statement.

Example 2: Testing a User Exit Routine

This example shows how to link edit a user exit routine into a library for testing.

```
//TSEXTRTN    JOB       MSGLEVEL=(1,1)
//STEP1            EXEC      PGM=IEWL,PARM='XREF,LET,LIST,NCAL',
//       REGION=96K
//SYSPRINT    DD        SYSOUT=A
//SYSLMOD     DD        DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//       DISP=(NEW,KEEP),SPACE=(1024,(20,2,1))
//SYSLIB      DD        *
    object deck
         NAME      MYEXIT
/*
//STEP2       EXEC      PGM=IMDPRDMP,PARM='ER=1'
//STEPLIB     DD        DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//       DISP=OLD
//SYSPRINT    DD        SYSOUT=A
//PRINTER     DD        SYSOUT=A
//TRACEDD     DD        DSNAME=TRACE2,UNIT=2400,VOL=SER=TRC2TP,
//       LABEL=(,NL),DISP=OLD
//SYSIN       DD        *
         EDIT      DDNAME=TRACEDD,SYS,EXIT=MYEXIT
/*
```

This example consists of two steps. In the first step:

EXEC Statement

invokes the linkage editor and requests diagnostic information.

SYSPRINT DD Statement

defines the message data set.

SYSLMOD DD Statement

defines the output data set, in this case a permanent job or step library named MYLIB.

SYSLIN DD Statement

defines the input data set, in this case an object deck containing the user program.

NAME Control Statement

specifies a member name (program name) to be assigned to the user program. Specify this program name on the EDIT control statement (EXIT=MYEXIT) when you need the exit routine for a particular IMDPRDMP execution.

In the second step:

EXEC Statement

invokes IMDPRDMP and specifies that, if an error occurs in the exit routine, EDIT should print the record associated with the error and delete the exit routine. (See the discussion of the EXEC statement in the section "Job Control Language Statements" earlier in this chapter.)

STEPLIB DD Statement

defines the data set that contains the exit routine, which, in this case, is MYLIB, a data set defined in STEP1 by the SYSLMOD DD statement.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the data set to which IMDPRDMP output will be directed.

TRACEDD DD Statement

defines the data set containing trace records to be processed by the exit routine.

SYSIN DD Statement

defines the data set that contains the IMDPRDMP control statement. The data set follows immediately.

EDIT Control Statement

invokes the EDIT function of IMDPRDMP, specifies that the trace data exists as an external trace data set, and supplies the name of the exit routine. Note that this name is the same as the membername specified in the NAME control statement in STEP1.

Example 3: Testing a User Format Appendage

This example shows how to add a user format appendage to a temporary data set for testing.

```
//TSTFMT      JOB       MSGLEVEL=(1,1)
//STEP1       EXEC      PGM=IEWL,PARM='XREF,LET,LIST,NCAL',
//       REGION=96K
//SYSPRINT    DD        SYSOUT=A
//SYSLMOD     DD        DSNAME=&TEMPLIB,UNIT=SSYSDA,
//       SPACE=(1024,(20,2,1)),DISP=(NEW,PASS)
//SYSLIN      DD        *
    object deck
         NAME      IMDUSR01
/*
//STEP2       EXEC      PGM=IMDPRDMP,PARM='ER=3'
//STEPLIB     DD        DSNAME=&TEMPLIB,DISP=OLD
//SYSPRINT    DD        SYSOUT=A
//PRINTER     DD        SYSOUT=A
//TRACEDD     DD        DSNAME=TRACE,UNIT=2400,VOL=SER=TRCTPE,
//       LABEL=(,NL),DISP=OLD
//SYSIN       DD        *
         EDIT      DDNAME=TRACEDD,USR=ALL
/*
```

This example consists of two steps. In the first step:

EXEC Statement

invokes the linkage editor.

SYSPRINT DD Statement

defines the message data set.

SYSLMOD DD Statement

defines a temporary data set that contains the format appendage.

SYSLIN DD Statement

defines the input data set, in this case the object deck containing the format appendage.

NAME Control Statement

specifies a member name (program name) for the format appendage. Note that the name shown in this example conforms to the convention for naming format appendages; that is, it is formed from the prefix IMDUSR concatenated with the format identifier (FID) to be specified in the GTRACE macro when user records are created.

<u>In the second step</u>:

EXEC Statement

invokes IMDPRDMP and specifies that ABEND processing should not be suppressed if a program check occurs in the format appendage. (See the discussion of the EXEC statement in the section "Job Control Language Statements" earlier in this chapter.)

STEPLIB DD Statement

defines the data set where the format appendage resides.

SYSPRINT DD Statement

defines the message data set.

PRINTER DD Statement

defines the data set to which the format appendage will direct its output.

TRACEDD DD Statement

defines the trace data set containing the records that the format appendage will process. In this case, the trace data set is on tape.

SYSIN DD Statement

defines the data set containing IMDPRDMP control statements. The data set follows immediately.

EDIT Control Statement

invokes the EDIT function of IMDPRDMP, specifies that the trace data exists as an external trace data set, and specifies that EDIT is to process all user-created records.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Operating System: Systems Reference Library Master Index, Order No. C28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

@USE parameter
   of PTFLE EXEC statement 227,228
* control statement 295
   used in SPZAP
      function 295
      parameter 295

abbreviations for PRDMP control statements 183
ABSDUMP control statement 293-294
   used in SPZAP
      format 293
      function 293
      parameters 293-294
ABSDUMPT control statement 293-294
   used in SPZAP
      example 305
      format 293
      function 293
      parameters 293-294
AID (application identifier)
   in GTF output record 70,76,77
ALL parameter
   (see PRINT control statement)
allocated main storage
   (see PRINT control statement)
application identifier
   (see AID)

BASE control statement 294
   used in SPZAP
      example 307
      format 294
      function 294
      parameters 294

CCHHR control statement 289-290
   used in SPZAP
      example 305
      format 289
      function 289
      parameters 290
changing space allocation for SYS1.LOGREC 23
choosing a service aid 13-14
combining PRDMP control statements 192
comment control statement
   (see * control statement)
communications vector table
   address of, how to specify 184
   (see also CVT control statement)
comprehensive trace
   how to request 60
CONSOLE= parameter
   (see IMDSADMP macro instruction)
console communications
   in GTF 63
   in JQDMP 91-93
   in OSJQD 161
   in PRDMP 183
   in SADMP 265-266
CONSOLE control statement 295
   used in SPZAP
      examples 306
      format 295
      function 295
      parameters 295
consoles supported by SADMP 257-258,268
control block formatting by PRDMP 177,186
   (see also FORMAT control statement)
control section, description of 131
   (see also CSECT Identification Record)
control statements
   for DIP00 23
   for EREP0 32
   for LIST 109-111
   for OSJQD 162
   for PRDMP 183-192
   for PTFLE 230-232
   for SPZAP 289-295
CPU= parameter
   (see IMDSADMP macro instruction)
cross-reference listing
   output of LIST
      contents of 115-116
      how to obtain 109
CSECT identification record
   how to create (PTFLE) 231
   how to print (LIST) 111
   how to update (SPZAP) 292
CURRENT parameter
   (see PRINT control statement)
current task's main storage, printing of
   (see PRINT control statement)
CVT address, how to specify 184
CVT (communications vector table)
   (see CVT= control statement)
CVT control statement 184
   used in PRDMP
      function 184
      format 184

damage assessment routine
   (see F03)
DAR (damage assessment routine)
   (see F03)

DATA= parameter
   (see GTRACE macro instruction)
data management records, printing of 188
DD statements
   in DIP00 23
      SERERDS 23
   in EREP0 32
      ACCIN 32
      ACCDEV 32
      EREPT 32
      JOBLIB 32
      SERLOG 32
   in LIST 119-124
      input 119-124
      output 119-124
      SYSPRINT 119-124
   in MDMAP 140
      input 140
      SNAPDUMP 140
      SYSABEND 140
      SYSPRINT 140
      SYSUDUMP 140
   in OSJQD 159
      OSJQDIN 159
      OSJQDOUT 159
      SYSIN 159
      SYSPRINT 159
   in PRDMP 180-182
      anyname 180
      PRINTER 181
      SYSIN 181
      SYSPRINT 181
      SYSUT1 182
      SYSUT2 182
      SYSWAPmn 181
      TAPE 180
   in PTFLE 228,229
      OUTF 228,229
      PCHF 228,229
      PRINT 228,229
      PTF.MODF 237
      SYSLMOD 228
      SYSPRINT 228
      SYSUT1 228
      SYSUT2 228
   in SPZAP 288
      SYSABEND 288
      SYSIN 288
      SYSLIB 288
      SYSPRINT 288
DDN= parameter
   (see LISTIDR control statement;
   LISTLOAD control statement;
   LISTOBJ control statement)
DDNAME= Parameter
   (see NEWDUMP control statement;
   EDIT control statement)
DEBUG= parameter
   (see GTF START command parameters)
device identification command (JQDMP) 91
differences between JQDMP and OSJQD 157
DIP00 service aid 17
   control statements 22
   JCL statements 22-24
   how to run 23-24
   input 22
DMA1 190
DSP trace option in GTF 62
DSP parameter
   (see EDIT control statement)
DUMP control statement 293
   used in SPZAP
      example 300
      format 293
      function 293
      parameters 293
dump title, how to specify
   in LIST
      in LISTIDR control statement 111
      in LISTLOAD control statement 109
      in LISTOBJ control statement 110
   in PRDMP
      in TITLE control statement 185
dumping main storage 245
dumping SYS1.LOGREC 29
DUMPT control statement 293
   used in SPZAP
      example 301
      format 293
      function 293
      parameters 293

EDIT control statement 187-191
      used in PRDMP
      format 188
      function 187
      parameters 188-190
         DSP 190
         EXIT= 188
         EXT 190
         DDNAME= 188
         IO 189
         IO= 189
         IO=SIO 189
         IO=SIO= 189
         JOBNAME= 189
         PI 190
         PI= 190
         SIO 189
         SIO= 189
         SIO=IO 189
         SIO=IO= 189
         START= 188
         STOP= 188
         SVC 189
         SVC= 189
         SYS 189
         TCB= 189
         USR= 190
EDIT function
   control statement format 188
   defaults 191-192
   error recovery 180
   examples 215-217
   JCL 179-182
   output 207
   output space requirements 194-195
   parameters 188-190
   storage requirements 179
EDIT parameter defaults 191

EDIT parameter priorities 191
editing GTF trace data 187-191
  from buffers in a dump 215
  from external trace data set 216
EID
  as field in GTF output 70-71
  extracted by IMDMEDIT macro 322,323
END control statement
  used in PRDMP 185
environment records
  as input to EREP0 31
ER= parameter
  of PRDMP EXEC statement 180
EREP0 service aid 25
  capabilities 29,30
  control statements 32-35
  examples 36-40
  how to execute 32,36-40
  input 31
  JCL 32
  output 41-49
ESD definition 132
event identifier
  (see EID)
examples
  DIP00 23,24
  EREP0 36-40
  GTF 68
  LIST 119-123
  MDMAP 146-151
  OSJQD 169
  PRDMP 208-216
  PTFLE 236-237
  SADMP 260,261,263
  SPZAP 300-308
EXEC statement parameters
  used in GTF cataloged procedure 59
  used in MDMAP 140-142
  used in PRDMP 179-180
  used in PTFLE 238
EXIT= parameter
  (see EDIT control statement)
exit routines
  function 313
  sample exit routine 321-323
EXT parameter
  (see EDIT control statement)
EXT trace option in GTF 61

FID
  as field in GTF output 70,77,78
  as parameter in GTRACE macro 67,68
  used in naming format appendage 67
FID= parameter
  (see GTRACE macro instruction)
format appendages
  function 313
  sample format appendage 324
FORMAT control statement 186
  used in PRDMP
    example 212
    format 186
    function 186
format control statements in PRDMP 185-190
  EDIT 187-190
  FORMAT 186
  LPAMAP 185
  PRINT 186
  QCBTRACE 185
  TSO 187
format identifier
  (see FID)
FREEnnn parameter
  of PRDMP EXEC statement 179
function control statements in PRDMP 183-185
  CVT= 184
  END 185
  GO 185
  NEWDUMP 184
  NEWTAPE 184
  ONGO 185
  TITLE 185
functions of service aids, summary of 13
F03 parameter
  (see PRINT control statement)

GO control statement 185
  used in PRDMP
    format 185
    function 185
    use with ONGO control statement 185
GO option
  (see GO control statement)
GTF service aid 51
  calculating storage requirements 65
  error recovery handling 69
  output 70-77
    control record format 76-77
    trace record format 70
  recording user data 67-68
    coding the GTRACE macro 67
    printing user data 67
  starting GTF 57-64
    cataloged procedure 59
    prompting 62
    specifying trace options 60-61
    START command 57-58
    storing trace options in SYS1.PARMLIB 63-64
GTF START command parameters 57
  devaddr 57
  keyword=option 59
  parmvalue 57-58
    MODE= 57
    TIME= 58
    DEBUG= 58
  procname.identifier 57
  REG= 59
  volser 57
GTF trace options 60-62
  DSP 62
  EXT 61
  IO,IOP 61
  PI,PIP 61
  PCI 62
  SIO,SIOP 61
  SSM 62
  SVC,SVCP 61

SYS,SYSM,SYSP 60
TRC 62
USR 62
GTRACE macro instruction in GTF 67
effect on EDIT user programs 67
function 67
how to code 67
parameters 67-68
DATA= 67
LNG= 67
ID= 68
FID= 68

high-speed dump
as output of SADMP
how to print 245
how to specify 248,249,258-265

ID= parameter
(see GTRACE macro instruction)
IDR
(see CSECT identification record)
IDRDATA control statement
used in SPZAP 292
IFCDIP00
(see DIP00 service aid)
IFCEREP0
(see EREP0 service aid)
IHLGTF
(see GTF)
IMaPTFLE
(see PTFLE service aid)
IMASPZAP
(see SPZAP service aid)
IMBLIST
(see LIST service aid)
IMBMDMAP
(see MDMAP service aid)
IMCJQDMP
(see JQDMP service aid)
IMCOSJQD
(see OSJQD service aid)
IMDPRDMP
(see PRDMP service aid)
IMDSADMP
(see SADMP service aid)
IMDSADMP macro instruction
format 254
function 253
parameters 254-258
CONSOLE= 257-258
CPU= 257
IPL= 254
OUTPUT= 255
PROTECT= 256
START= 256
TYPE= 255
input address parameter in JQDMP 92
IO parameter (PRDMP)
(see EDIT control statement)
IO trace option in GTF 61
IOP trace option in GTF 61
(see also prompting, how to request)
IPL= parameter
(see IMDSADMP macro instruction)

JCL
(see job control language statements)
job control language statements
DIP00 22-24
EREP0 32
LIST 119-123
MDMAP 140
OSJQD 159
PRDMP 179-182
PTFLE 238,239
SADMP
for initializing dump program 261
for retrieving macro instruction 260
SPZAP 288
job queue data set
JQDMP 88-90
OSJQD 162
JOBNAME= parameter
in JQDMP 93
in OSJQD 163
in PRDMP
in PRINT control statement 186
in EDIT control statement 189
JQDMP service aid 79
device identification command 91-93
error handling 95
job queue format 88
operational considerations 102
output 97-101
retrieval 15

LINECNT= parameter
of PRDMP EXEC statement 179
link pack area formatting
(see LPAMAP control statement)
link pack area maps
MDMAP 137,146
PRDMP 185
LINKPACK parameter
of MDMAP EXEC statement 141
LIST service aid 103
control statements 109-111
LISTIDR 111
LISTLOAD 109
LISTOBJ 110
examples 119-123
executing LIST 109-111
listing a load module 109-110
listing an object module 110
listing CSECT identification records 111
features 108
JCL 119-123
output 112-118
LISTIDR control statement 111
used in LIST
example 121
format 111
function 111
parameters 111
listing local fixes 111
listing PTFs 111

- LISTLOAD control statement 109
  - used in LIST
    - example 119
    - format 109
    - function 109
    - parameters 109
- LISTOBJ control statement 110
  - used in LIST
    - example 120
    - format 110
    - function 110
    - parameters 110
- LNG= parameter
  - (see GTRACE macro instruction)
- load module attribute definitions 135-136
  - linkage-editor assigned 136
  - programmer assigned 135
- load module, definition of 131
- load module listing
  - output of LIST
    - contents of 115-116
    - how to obtain 109
- load module map
  - output of MDMAP
    - contents of 143-145
    - how to obtain 140
- low speed dumps
  - output of SADMP
    - printing 250
    - specifying 255
- LPAMAP control statement
  - used in PRDMP 185
- LPA maps
  - (see link pack area maps)

- macro expansion messages (SADMP) 272-273
- main storage, printing of by PRDMP
  - allocated storage 186
  - current task 186
  - specific addressses 186
  - jobnames, by 186
  - DAR terminated task 186
  - F03 186
- main storage requirements
  - EDIT user programs 313
  - GTF 59
  - OSJQD 160
  - PRDMP 179
  - PTFLE 227
- major control blocks, formatting of by PRDMP 186
- maps
  - link pack area 137,185
  - load modules 131
  - main storage
    - MVT 186
    - TSO UMSM 187
  - nucleus 138
- MDMAP service aid 125
  - definitions
    - of input 131-138
    - of output 143-145
  - examples 146-151
  - EXEC statement parameters 140
    - BASIC 141
    - DEBUG 141
    - hhhhhh 141
    - LINKPACK 141
  - input types
    - load modules 131
    - MFT resident reenterable load module area 137
    - MVT link pack area 137
    - nucleus 138
  - JCL 140
  - operational considerations 152
  - output 143-145
- messages
  - (see output comments; output error indicators)
- minimal trace
  - how to request 61
- MODE= parameter
  - (see GTF START command parameters)
- MODF DD statement
  - used in PTFLE 229
- modifying data (SPZAP) 283
- module definition
  - (see load module definition, object module definition)
- multiple dump processing (PRDMP) 210

- N parameter
  - of PRDMP EXEC statement 179
- NAME control statement 289
  - used in SPZAP
    - example 300
    - format 289
    - function 289
    - parameters 289
- NEWDUMP control statement 184
  - used in PRDMP
    - example 210
    - format 184
    - function 184
    - parameters
      - DDNAME= 184
      - FILESEQ= 184
      - DUMPSEQ= 184
- NEWTAPE control statement 184
  - used in PRDMP
    - format 184
    - function 184
- NUCLEUS parameter
  - (see PRINT control statement)
- nucleus maps 146
  - output of MDMAP
    - contents of 146
    - how to obtain 146

object module, definition of (MDMAP) 131
ONGO control statement 185
  used in PRDMP
    example 210
    format 185
    function 185
    parameters 185
      CVT= 185
      EDIT 185
      FORMAT 185
      LPAMAP 185
      PRINT 185
      QCBTRACE 185
      TSO 185
  relationship to GO control
   statement 185
OSJQD service aid 153
  control statements 161,162-163
    ALL 163
    END 161
    JOBNAME= 163
    QCR= 162
  JCL 159
  output 164,168
OUTF DD statement
  used in PTFLE 228,229
output address parameter in JQDMP 91
OUTPUT= parameter
  (see IMDSADMP macro instruction)
output comments
  OSJQD 167
object module listing
  output of LIST
    contents of 117
    how to obtain 110
output
  of DIP00 23,24
  of EREP0 41-49
  of GTF 70-78
  of JQDMP 97-101
  of LIST 112-118
  of MDMAP 143-145,147,149,151
  of OSJQD 164-168
  of PRDMP 197-207
  of PTFLE 233-235
  of SADMP 248-252
  of SPZAP 296-299
output space requirements (PRDMP) 193-19

P control statement
  (see PRINT control statement (PRDMP))
parameters
  of control statements
    (see DIP00 service aid;
    EREP0 service aid; LIST
    service aid; OSJQD service
    aid; PRDMP service aid;
    PTFLE service aid; and
    SPZAP service aid)
  of EXEC statement
    in GTF cataloged procedure 59
    in MDMAP 140-142
    in PRDMP 179-180
    in PTFLE 227
  in GTF START command 57-59
  in IMDSADMP macro instruction 254-258
PARM= parameter in EXEC statement
  in EREP0 33-35
    ACC= 35
    CUA= 34
    DATE= 34
    DEV= 34
    HIST= 35
    MES= 34
    MOD= 34
    M67= 35
    PRINT= 35
    RDESUM= 35
    TERMIN= 35
    TYPE= 33
    VOLID= 34
    ZERO= 34
  in GTF cataloged procedure 57-58
    DEBUG= 57
    MODE= 58
    TIME= 58
  in MDMAP 140-142
    base address 141
    BASIC 141
    DEBUG 141-142
    LINKPACK 141
  in PRDMP 179-180
    ER=x 180
    FREEnnn 179
    LINECNT 179
    n 179
    S 179
    T 179
  in PTFLE 228
    &USE 228
PCHF DD statement
  used in PTFLE 228,229
PCI trace option in GTF 62
PI parameter
  (see EDIT control statement)
PI trace option in GTF 61
PIP trace option in GTF 61
  (see also prompting, how to request)
PRDMP cataloged procedure 196
PRDMP service aid 171
  cataloged procedure 196
  control statements 183
    CVT= 184
    END 185
    FORMAT 186
    GO 185
    LPAMAP 185
    NEWDUMP 184
    NEWTAPE 184
    ONGO 185
    PRINT 186
    QCBTRACE 185
    TITLE 185
    TSO 187
  EDIT function 187-191
    control statement 187-191
    defaults 191
    error recovery 180
    examples 215-216
    JCL 179-182
    output 207
    storage requirements 179

  - examples 208-216
  - functional summary 175
  - JCL 179-182
  - output 197-207
  - storage requirements 179
- PRINT control statement 186
  - used in PRDMP
    - format 186
    - function 186
    - parameters 186
      - ALL 186
      - CURRENT 186
      - F03 186
      - JOBNAME= 186
      - NUCLEUS 186
      - STORAGE= 186
- PRINT DD statement
  - used in PTFLE 228,229
- printing and accumulating records (EREP0) 38,39
- printing machine check records (EREP0) 36-37
- prompting, how to request (GTF) 62
- PROTECT= parameter
  - (see IMDSADMP macro instruction)
- PTF 223
- PTF.MOD DD statement
  - used in PTFLE 237
- PTFLE control statement 230
  - example 236,237
  - format 230
  - function 230
- PTFLE service aid 219
  - application function
    - control statements 230-232
    - examples 237
    - execution 227-229
    - introduction 225
    - JCL 228
    - operational considerations 239
    - output 233
  - generate function
    - control statements 230-232
    - examples 236
    - execution 228
    - introduction 223-224
    - JCL 228-229
    - operational considerations 239
    - output 233

- QCB formatting
  - (see QCBTRACE control statement)
- QCBTRACE control statement
  - used in PRDMP 185
- QCR= parameter
  - in JQDMP 92
  - in OSJQD 162
- queue control block formatting
  - (see QCBTRACE control statement)

- RDE records (EREP0) 35
- RDESUM parameter (EREP0) 35
- reinitializing SYS1.LOGREC data set
  - (see DIP00 service aid)
- REP control statement 291
  - used in SPZAP
    - example 300
    - format 291
    - function 291
    - parameters 291
- retrieving stand-alone service aids
  - JQDMP 15
  - SADMP 15

- S parameter
  - of PRDMP EXEC statement 179
- SADMP macro instruction
  - (see IMDSADMP macro instruction)
- SADMP output, printing of
  - low-speed
    - by IEBGENER 263
    - by PRDMP 177
  - high-speed
    - by PRDMP 175
- SADMP service aid
  - assembling the macro instruction 258-259
  - coding the macro instruction 254-258
  - error conditions 270-272
  - executing the dump program 265-266
  - IMDSADMP macro instruction 254
  - initializing the residence volume 262-263
  - operational considerations 267-269
  - output 248-252
- select options
  - (see select parameters)
- select parameters
  - in JQDMP 92-93
  - in OSJQD 162-163
- SERERDS DD statement
  - used in DIP00 123
    - (see also OSJQD control statements)
- SETSSI control statement 292
  - used in SPZAP
    - example 300
    - format 292
    - function 292
    - parameters 292
- service aids 13
- SIO parameter
  - (see EDIT control statement)
- SIO trace option in GTF 61
- SIOP trace options in GTF 61
  - (see also prompting, how to request)
- SNAPDUMP DD statement
  - used by GTF 60
  - used by MDMAP 140,142,150
- specialized tracing action 56
- specifying the GO option
  - (see GO control statement)
- SPZAP service aid 275
  - control statements 289-295
  - data modification and inspection 218-283
  - dumping data 284
  - examples 300-308
  - executing SPZAP 288
  - JCL 288
  - operational considerations 287
  - output 296-299
  - updating system status information 285-286

- SSM trace option in GTF 62
- START= parameter
  - in PRDMP
    - (see EDIT control statement)
  - in SADMP 256
- STOP= parameter
  - (see EDIT control statement)
- storage requirements
  - (see main storage requirements)
- STORAGE= parameter
  - (see PRINT control statement)
- SVC parameter
  - (see EDIT control statement)
- SVC trace option in GTF 61
- SVCP trace option in GTF 61
  - (see also prompting, how to request)
- SWAP data sets, how to print 213
- SYS parameter
  - (see EDIT control statement)
- SYS trace option in GTF 60
- SYSABEND DD statement
  - used in MDMAP 140,142
  - used in SPZAP 288
- SYSIN DD statement
  - used in PRDMP 181
  - used in SPZAP 288
- SYSLIB DD statement
  - used in SPZAP 288
- SYSM trace option in GTF
  - function 61
  - How to request 60
- SYSOUT space, allocation of by PRDMP 193-195
- SYSP trace option in GTF 61
  - (see also prompting, how to request)
- SYSPRINT DD statement
  - used in PRDMP 181
  - in SPZAP 288
- system events (GTF) 56
- SYSTEM= parameter
  - (see TSO control statement)
- SYSUDUMP DD statement
  - used by MDMAP 140,142
- SYSUT1 DD statement
  - used in PRDMP 182
  - used in PTFLE 228
- SYSUT2 DD statement
  - used in PRDMP 182
  - used in PTFLE 228
- SYSWAPmn DD statement
  - used in PRDMP 181
- SYS1.DUMP data set
  - as input to PRDMP
    - printing the dump data set 175
    - clearing the dump data set 209
- SYS1.LOGREC data set
  - changing space allocation 23
  - dumping 29
  - initializing 23
  - processing selected records 29
    - accumulating 29,38-40
    - editing and writing 29,36-37
    - summarizing 30
- T parameter
  - of PRDMP EXEC statement 179
- TAPE DD statement
  - used in PRDMP 180
- TIME= parameter
  - (see GTF START command parameters)
- timestamp
  - how to request 58
  - field in GTF output 70,77,78
- TITLE control statement 185
  - used in PRDMP
    - format 185
    - function 185
- title, how to specify
  - (see dump title, how to specify)
- trace options
  - (see GTF trace options)
- tracing with prompting 62
- tracing without prompting 60-62
- TRC trace option in GTF 62
- TSO control statement 187
  - used in PRDMP
    - format 187
    - function 187
    - parameters
      - SYSTEM= 187
      - USER= 187
- TSO dumps, how to print 187
- TYPE= parameter
  - (see IMDSADMP macro instruction)
- TYPE=HI option 255
- TYPE=LO option 255

- user programs 309
  - error handling 317-318
  - exit routines 313,321-323
  - format appendages 313,324-325
  - interfaces with EDIT 314-318
  - parameter list 314
  - return codes 317
- USER= parameter
  - (see TSO control statement)
- USR= parameter
  - (see EDIT control statement)
- USR trace option in GTF 62
  - (see also GTRACE macro)

- VER control statement
  - (see VERIFY control statement)
- VERIFY control statement 290-291
  - used in SPZAP
    - example 303
    - format 290
    - function 290
    - parameters 290-291

- work data set, use of in PRDMP 182

GC28-6719-2

System/360 Service Aids (S360-31) Printed in U.S.A. GC28-6719-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System:
Service Aids

Order No. GC28-6719-2

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - □ Frequently for reference in my work.
  - □ As an introduction to the subject.
  - □ As a textbook in a course.
  - □ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: 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.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System:
Service Aids

Order No. GC28-6719-2

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - □ Frequently for reference in my work.
  - □ As an introduction to the subject.
  - □ As a textbook in a course.
  - □ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: 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.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]