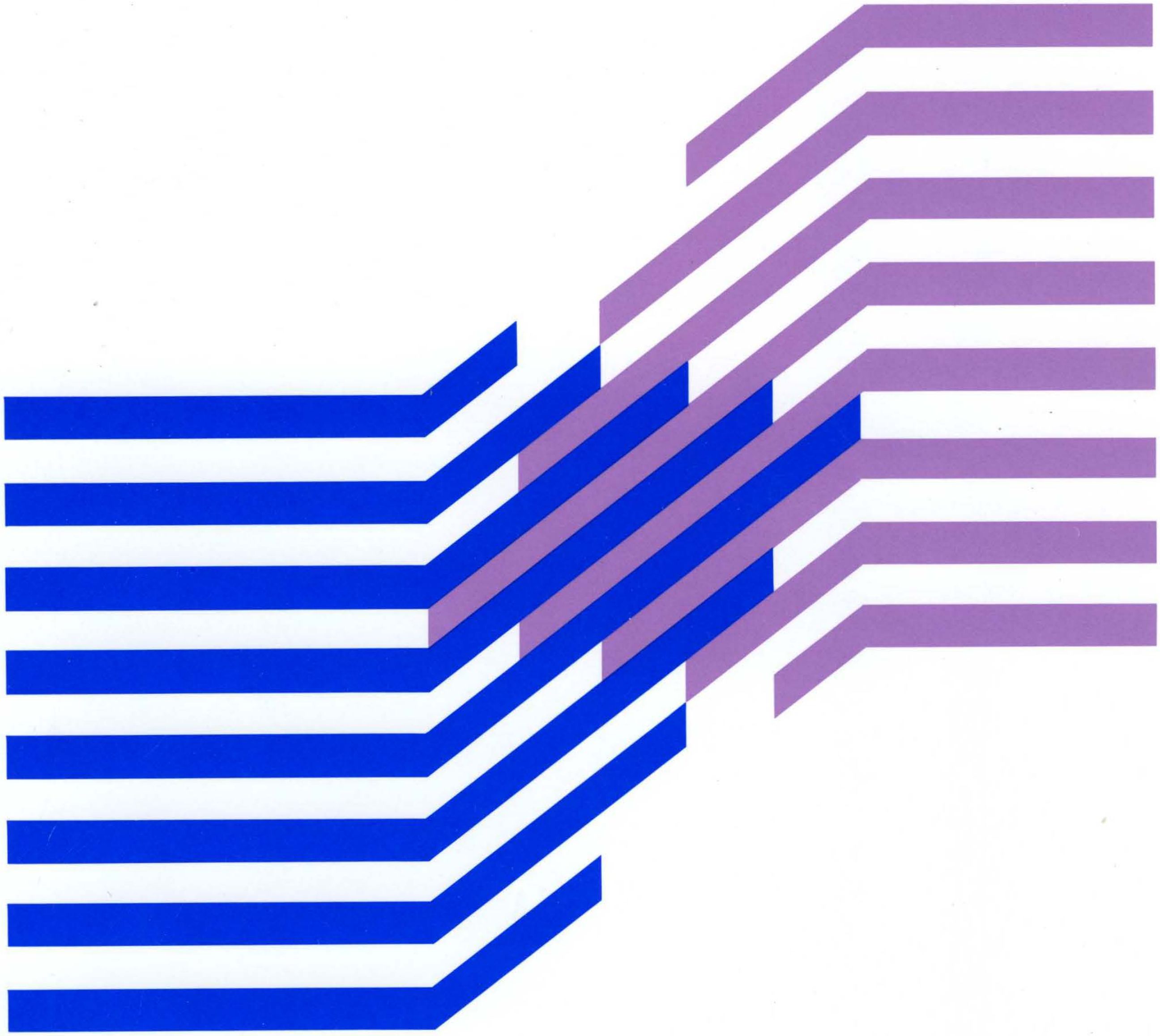
IBM

MVS/ESA Service Aids

GC28-1844-2

MVS/System Product:
JES2 Version 3
JES3 Version 3



GC28-1844-2

MVS/ESA Service Aids

MVS/System Product:
JES2 Version 3
JES3 Version 3

Third Edition (December 1989)

This is a major revision of, and obsoletes, GC28-1844-1. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 3 of MVS/System Product 5685-001 or 5685-002 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. References to IBM products in this document do not imply that functionally equivalent products may be used. The security certification of the trusted computing base that includes the products discussed herein covers certain IBM products. Please contact the manufacturer of any product you may consider to be functionally equivalent for information on that product's security classification. This statement does not expressly or implicitly waive any intellectual property right IBM may hold in any product mentioned herein.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 950, Poughkeepsie, NY 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PROGRAMMING INTERFACES

The majority of this book consists of guidance information to help the system programmer diagnose and fix failures in system or application programs. Such information should never be used as programming interface information. However, this book also contains general-use programming interface information.

General-Use Programming Interfaces: General-use programming interfaces are provided to allow a customer installation to write programs that use the services of MVS/System Product Version 3. General-use programming interfaces do not have significant dependencies on detailed product design or implementation.

General-use programming interface information is identified by brackets before and after the information, as follows:

GENERAL-USE PROGRAMMING INTERFACE

Description of the interface.

End of GENERAL-USE PROGRAMMING INTERFACE

Contents

Figures

About This Book

Service aids are programs designed to help system programmers and IBM program support representatives diagnose and fix failures in system or application programs. The service aids described in this book are designed to assist in identifying the symptoms of the problem, gathering relevant data from system data areas to isolate the problem to the component level, and analyzing the component to determine the cause of the problem. This publication explains how, why, and when to use IBM service aids programs.

The service aids are:

- GTF (Generalized Trace Facility) - Traces selected system events such as SVC and I/O interruptions.
- AMBLIST - Formats and prints object modules, load modules, and CSECT identification records; maps reenterable load module area.
- AMDSADMP - Operates as a stand-alone program to produce a dump of central and virtual storage and processor-related data.
- AMASPZAP - Verifies or replaces instructions or data in a load module.
- IPCS (interactive problem control system) - Provides installations with the expanded capabilities for diagnosing software failures and facilities for managing problem information and status.

  Note: IPCS is not discussed in this publication; for information on IPCS, see the *MVS/ESA Interactive Problem Control System User's Guide*.

Trademarks

The following are trademarks of International Business Machines Corporation.

Enterprise Systems Architecture/370™
MVS/ESA™
MVS/DFP™
MVS/SP™

Who Should Use This Book

This book is intended for anyone who must determine and diagnose system problems and debug a failed system. Usually, this person is a system programmer. The book assumes that the reader can:

- Code JCL statements to execute programs or cataloged procedures
- Code in assembler language and read assembler and linkage editor output
- Understand basic data processing terminology.

How This Book Is Organized

Each service aid is explained in a separate chapter and the chapters are arranged in alphabetical order. The chapter headings show the names of the programs *without* the three-character component identifier (such as AMD). This means that you should expect to see AMDSADMP referred to as simply SADMP, except in JCL examples and other situations where the full name is necessary.

Note that throughout the text each service aid is referred to by its abbreviated name, except where the full name of the program is necessary for technical accuracy. Although you may be confused by the abbreviations at first, you will soon find that the shorter names are easier to remember because they remind you of the functions that the service aids perform.

Think of the abbreviated names as acronyms, like this:

GTF - generalized trace facility
LIST - module listing program
SADMP - stand-alone dump program
SPZAP - superzap (data checker and modifier).

How To Use This Book

Service aids have three general functions:

Information Gathering

- To dump central storage, use the stand-alone program SADMP. To dump virtual storage, all central storage, and processor-related data, use the high-speed version of SADMP. You can use IPCS to view the output of SADMP at the terminal or format the output for printing.
- To trace system events such as SVC and I/O interruptions, use GTF. GTF output can be formatted and printed using the GTFTRACE subcommand of IPCS.

Formatting and Printing: Mapping

- To summarize and print records in the SYS1.LOGREC data set, use EREP which is described in the publication *SYS1.LOGREC Error Recording.*
- To format and print load modules, object modules and CSECT identification records, or to map the reenterable load module area or the link pack area, use LIST.
- To format, print, and view SADMP output, other system dumps, and GTF trace output, use IPCS.
- To process dumps interactively, see *IPCS User's Guide.*

Generating and Applying Fixes

- To apply new releases, PTFs, or user modifications, use SMP. For information on SMP, see the references listed under "Related Information" on page xiv.
- To verify and/or replace instructions in a load module, or data on a direct access device, use SPZAP.
- To initialize the SYS1.LOGREC data set, use IFCDIP00, which is described in the publication *SYS1.LOGREC Error Recording.*

Notation for Defining Control Statement Parameters

The following discussion describes the notations this publication uses in the format descriptions. For further coding conventions, see *MVS/ESA JCL Reference*.

1. On the control statement, code uppercase letters, words, and the following characters exactly as they appear in the format description.

ampersand	&
asterisk	*
comma	,
equal sign	=
parenthesis	()
period	.

2. Lowercase letters, words, and symbols appearing in the format description represent variables for which you substitute specific information when coding the parameter.

   For example, DDN=ddname is the format description for the DDN parameter of the LISTLOAD control statement. When you code the DDN parameter, you substitute an alphameric character for the word 'ddname'.

3. Braces {} are a special notation and you never code them on a control statement. Braces group related items.

   For example, {IDENT|**ALL**} is part of the format description of the OUTPUT parameter of the LISTIDR control statement of LIST. When you use LISTIDR, code either IDENT or ALL. If you omit the OUTPUT parameter, LIST will assume a default of OUTPUT=ALL.

4. Brackets [] are a special notation and you never code them on a control statement. Brackets indicate that the enclosed item or items are optional and you can code one or none of the items.

   For example, [,MLPA] is part of the format description for the LISTLPA control statement. When you code the LISTLPA control statement, you can include MLPA or omit it.

   An example of more than one item enclosed in brackets is [S|I|SI], which is part of the format description for the CCW trace option of GTF. When coding the CCW trace option, you can include 'S' or 'I' or 'SI' or omit them all.

5. An ellipsis...(three consecutive periods) is a special notation and you never code it on a control statement. An ellipsis indicates that you can code the preceding item more than once in succession.

   For example, ASID=(asid1 ... asidn) is a possible response to GTF prompting. The ellipsis indicates that you can repeat asid.

6. Underlining is a special notation and you never code it on a control statement. When either brackets or braces enclose a group of items, and you do not code any of the grouped items, then the underlined item in that group is the default.

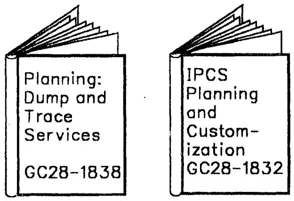
   For example, [,SYSUT={unit|**SYSDA**}] is part of the AMDSADMP macro instruction. The brackets indicate that SYSUT is an optional parameter. If you code SYSUT, the braces indicate that you can code either unit or SYSDA. If you omit both unit and SYSDA, then SYSUT=SYSDA is the default.

Related Information

Where necessary, this book references information in other books, using shortened versions of the book title. The following table shows the shortened titles, complete titles, and order numbers of the books you might need while you are using this book.

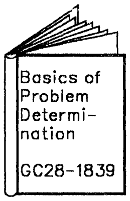
Short Title Used in This Book	Title	Order Number
Application Development Macro Reference	MVS/ESA Application Development Macro Reference	GC28-1822
Component Diagnosis: Service Aids	MVS/ESA Component Diagnosis and Logic: Service Aids	LY28-1846
DFP: Diagnosis Reference	MVS/ESA Data Facility Product Version 3: Diagnosis Reference MVS/ESA Data Facility Product Version 3 Release 2: Diagnosis Reference	LY27-9551 LY27-9571
Initialization and Tuning	MVS/ESA System Programming Library: Initialization and Tuning	GC28-1828
IPCS Command Reference	MVS/ESA Interactive Problem Control System (IPCS) Command Reference	GC28-1834
IPCS Planning and Customization	MVS/ESA Interactive Problem Control System Planning	GC28-1832
IPCS User's Guide	MVS/ESA Interactive Problem Control System (IPCS) User's Guide	GC28-1833
JCL Reference	MVS/ESA JCL Reference	GC28-1829
Managing Non-VSAM Data Sets	MVS/DFP Version 3 Release 2: Managing Non-VSAM Data Sets	SC26-4557
SPL: Application Development Guide	MVS/ESA System Programming Library: Application Development Guide	GC28-1852
SPL: Application Development Macro Reference	MVS/ESA System Programming Library: Application Development Macro Reference	GC28-1857
System Codes	MVS/ESA Message Library: System Codes	GC28-1815
System Commands	MVS/ESA Operations: System Commands	GC28-1826
Using Dumps and Traces	MVS/ESA Diagnosis: Using Dumps and Traces	LY28-1843
Utilities	MVS Data Administration: Utilities MVS/Data Facility Product Version 3 Release 2: Utilities	GC26-4018 SC26-4559
VTAM Operations	Advanced Communications Function/Virtual Telecommunications Access Method Operations	SC27-0612

PLANNING:

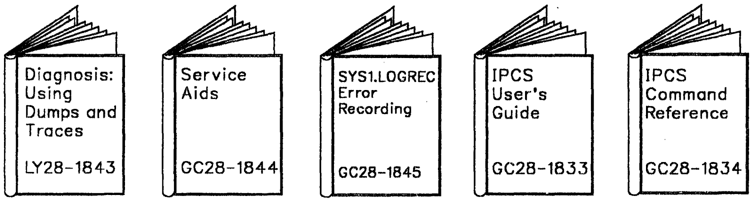

Planning: Dump and Trace Services
GC28-1838
IPCS Planning and Custom-ization
GC28-1832

DIAGNOSIS:

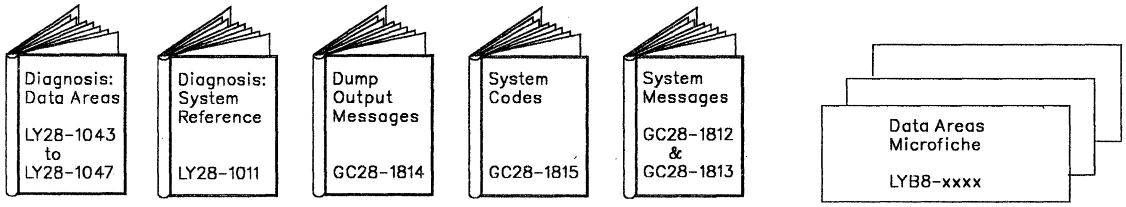
For Diagnostic Procedures



For Data Collection

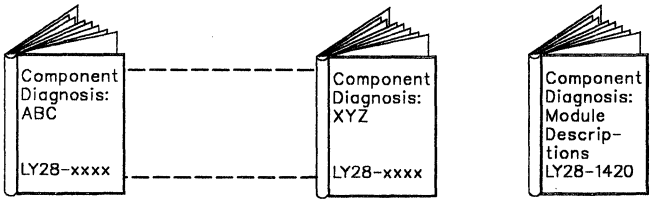


For Data Interpretation



For Component-Specific Information

(To identify the component, see Basics of Problem Determination)



Summary of Changes

Summary of Changes
for GC28-1844-2
MVS/System Product Version 3 Release 1.3

This edition contains the following new or changed information for MVS/SP 3.1.3.

New Information

- A new requirement for the JCL that builds the stand-alone dump program. System macros that SADMP needs are now in SYS1.MODGEN as well as in SYS1.MACLIB, so the JCL must include a SYSLIB DD statement that concatenates these two data sets.
- A new type of partitioned data set (PDS) introduced by MVS/DFP Version 3 Release 2. Partitioned data sets extended (PDSEs) can consist of source and object modules that you might want to format and print. Certain discussions in the LIST chapter now include PDSEs, where appropriate.

Changed Information

- Minor editorial and maintenance changes.
- A terminology change:

Storage

This book uses the term *central storage* for the storage that has been called *real storage.* In the 3090 processor, storage consists of:

Central storage + expanded storage = processor storage

Virtual storage consists of pages contained in processor storage and auxiliary storage.

Summary of Changes
for GC28-1844-1
as updated December 1988

This edition contains the following new or changed information:

- Changes to GTF (APARs OY13937, OY14565, and UY14566). GTF now:
  - Allows users to address the data for the ASM CCWs
  - Gives users more control over the buffers that GTF uses
  - Increases the tape block size for GTF data sets going to tape.
- Minor editorial and maintenance changes.

Do not use the new GTF functions until the PTFs for the cited APARs are installed.

Summary of Changes
for GC28-1844-0
as updated September 16, 1988
by Technical Newsletter GN28-1259

This newsletter contains an update for AMASPZAP.

Summary of Changes
for GC28-1844-0
MVS/System Product Version 3 Release 1.0

This book contains information previously presented in *MVS/Extended Architecture Service Aids*, GC28-1159. The following summarizes the changes to that information.

Changed Information: All the chapters, headings, figures, and so forth, have been altered to accommodate the elimination of Print Dump.

Deleted Information: Chapter 3, Print Dump, has been deleted. All references to Print Dump have been removed or changed to reference appropriate replacement information.

Chapter 1. GTF

Introduction

The generalized trace facility is a service aid program that is available for determining and diagnosing system problems. GTF records system and user-defined program events. Through GTF you can trace:

- Any combination of system events, such as all I/O interruptions and all SVC interruptions
- Specific incidences of one type of system event, such as all I/O interruptions on one particular device
- User-defined events which are generated by the GTRACE macro.

GTF produces output trace records of system events, subsystem events, and user events directed to buffers in virtual storage. The user may also direct output to a data set (IEFRDER). IPCS may be used to format, display, and print the GTF output. See *IPCS User's Guide* for further information about using IPCS to process GTF output.

The following apply to GTF 31-bit addressing support:

- GTF receives control from all branch callers in 31-bit addressing mode, regardless of where the caller resides in storage.
- GTF 31-bit support allows the tracing of user and system data above 16 megabytes.
- Users can issue the GTRACE macro in either 24- or 31-bit addressing mode. However, a user must execute in 31-bit mode to trace data above 16 megabytes.

Notes:

- GTF traces events on all processors regardless of the specification for GTF on the AFFINITY macro during system generation.
- Installations can run with both system trace and GTF active. Starting GTF does not alter the status of system trace.

Using GTF

GTF is an integral part of the system defined at system generation and runs as a system task.

Features of GTF

GTF provides many features to allow you to trace a variety of system and user events. You can trace channel programs and associated data for start subchannel and resume subchannel operations and I/O interruptions by means of the CCW trace option. PCI causes GTF to record intermediate status interruptions in the same format that GTF uses to create other I/O trace records. GTF can also record system recovery routine operations including STAE/ESTAE operations through the RR option. For a complete summary of GTF trace options see the topic "GTF Trace Options" on page 1-9.

Note: For special considerations in the use of GTF to trace events in indexed VTOC processing, see *DFP: Diagnosis Reference*.

GTF Trace Output

For the DSP, EXT, PI, RNIO, RR, SRM, and SVC options, GTF produces system trace records with two kinds of format: comprehensive and minimal. For all other GTF options, GTF produces trace records in only one format. To see what the records contain for each record type, see *Using Dumps and Traces*.

GTF writes trace record output in a trace table in virtual storage (internal mode) and can also write to the IEFRDER data set on an external storage device (external or deferred mode). The external storage device can be either a tape or a direct access device. When the trace records fill up the internal trace table or the data set, GTF overlays previously stored or written output beginning at the oldest buffer or physical block.

Retrieving GTF Trace Output

IPCS makes it possible to format and print internal and external GTF trace records or to view them at the terminal. In addition, you may format and print the trace records generated by the GTRACE macro. For information on using the GTRACE macro, see *SPL: Application Development Macro Reference*.

If you request that trace data be included in an ABEND, SNAP, SVC, or stand-alone dump, and if GTF is active, you can use IPCS to format the records created by GTRACE. Formatting occurs independently of the trace mode or options for GTF. You control the number of buffers that GTF formats when you specify the ABDUMP, SDUMP or SADMP parameter in the START GTF command. Also, for ABEND and SNAP dumps, only those records directly associated with the failing address space are formatted. GTF does not format the channel program trace data associated with the failing address space in ABEND and SNAP dumps.

How to Start GTF

You invoke GTF as a system task in an address space by entering a START command from the operator's console; you cannot start GTF as a job. Using the START command, you select the GTF cataloged procedure or your own cataloged procedure. Optional parameters in the cataloged procedure and START command allow you to specify internal or external tracing, timestamps on records, what action should occur if GTF encounters an error during processing, and the number of buffers which are to appear on ABDUMP/SNAP or SVC dumps. If you specify one or more of the START command parameters, the EXEC parameters from the catalogued procedure are ignored. To select the trace options, you either specify each option directly through the console or retrieve (via the cataloged procedure) a set of previously stored options which exist as a member of SYS1.PARMLIB.

How to Specify the START Command

Figure 1-1 shows the general format of the START command as it is used to invoke GTF. Since all messages go to the master console, (also called the integrated operators console), the START command should be entered only from a console eligible to be a master console.

```
{START|S}{GTF|procname}[.identifier][,devname][,volserial]]

   [,(parm[,parm]...)][,MEMBER={GTFPARM|userparm}]

    [,keyword=option[,option]...]
```

devname and **volserial** are positional parameters all other parameters are keyword parameters. When you omit a positional parameter and code any keyword parameters, you must indicate the absence of the positional parameter by coding a comma in place of the positional parameter.

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

The descriptions below explain the parameters of the START command as they are used by GTF:

GTF
indicates the name of the IBM-supplied cataloged procedure that invokes GTF.

procname
identifies the name of the user-written cataloged procedure that you write to invoke GTF.

identifier
specifies the user-specified name identifying this GTF session.

devname
specifies the device number or the device type of an output device to contain the trace data set. If you do not specify a device number or device type on the START command, GTF uses the device number provided on the IEFRDER DD statement in the cataloged procedure. **devname** is a positional parameter. When you omit devname and code any keyword parameters, you must code a comma to indicate the absence of devname.

volserial

indicates the serial number of a magnetic tape or direct access volume which is to contain the trace data set. **volserial** is a positional parameter. When you omit volserial and code any keyword parameters, you must code a comma to indicate the absence of the volserial.

(parm)

overrides the value specified in the PARM= parameter of the EXEC statement in the cataloged procedure and may contain any combination of the following parameters:

```
([parm][,parm] ...)

   where parm is one of the following:

                     MODE={INT|EXT|DEFER}
                     SADMP={nnnnnnK|nnnnnnM|40K}
                     SDUMP={nnnnnnK|nnnnnnM|40K}
                     ABDUMP={nnnnnnK|nnnnnnM|0K}
                     BLOK={nnnnn|10}
                     TIME={YES|NO}
                     DEBUG={YES|NO}
```

MODE={INT|EXT|DEFER}

defines where GTF maintains the trace data. MODE=INT causes GTF to maintain the trace data in the GTF address space. MODE=EXT causes GTF to maintain the trace data in an external data set that is defined by the IEFRDER DD statement in the cataloged procedure. MODE=DEFER causes GTF to maintain the trace data in the GTF address space until the STOP GTF command is issued. Then, during its termination processing, GTF transfers the data from the GTF address space to the GTF output data set.

When you code the START command without any parameters, GTF obtains the MODE= parameter from the EXEC parameter in the cataloged procedure. The default is MODE=EXT.

When tracing to an external device, you can use IPCS to format data in the trace data set.

{SADMP|SA}={nnnnnnK|nnnnnnM|40K}

allows you to specify the amount of storage you need to save GTF trace data for stand-alone dumps. You must specify the amount of storage in terms of either K (kilobytes) or M (megabytes). The minimum amount is 40K, and the maximum is 2048M-400K; the amount you specify is rounded up to 4K boundaries for DASD data sets, or 32K boundaries for tape data sets.

Instead of the BUF= parameter, use the SADMP= parameter on the START GTF command. The system ignores BUF= and uses the defaults for the SADMP=, SDUMP=, or ABDUMP= parameters.

When you code the START command without any parameters, GTF obtains the SADMP= parameter from the EXEC parameter in the cataloged procedure. The default is 40K if the GTF data set is on DASD, or 64K if the GTF data set is on tape.

If the system takes a stand-alone dump, you can use IPCS to format this storage.

{SDUMP|SD} = {nnnnnnK|nnnnnnM|40K}
allows you to specify the amount of storage you need to save GTF trace data for SVC dumps. You must specify the amount of storage in terms of either K (kilobytes) or M (megabytes). The minimum amount is zero, and the maximum cannot exceed the maximum amount of storage defined by the SADMP= parameter. The amount you specify is rounded up to 4K boundaries for DASD data sets, or 32K boundaries for tape data sets.

Instead of the BUF= parameter, use the SDUMP= parameter in conjunction with SADMP= and ABDUMP= on the START GTF command. The system ignores BUF= and uses the defaults for the SADMP=, SDUMP=, or ABDUMP= parameters.

When you code the START command without any parameters, GTF obtains the SDUMP= parameter from the EXEC parameter in the cataloged procedure. The default is 40K if the GTF data set is on DASD, or 64K if the GTF data set is on tape.

If the system takes an SDUMP, you can use IPCS to format this storage.

{ABDUMP|AB} = {nnnnnnK|nnnnnnM|40K}
allows you to specify the amount of GTF data to be formatted in an ABEND or SNAP dump. You must specify the amount of data in terms of either K (kilobytes) or M (megabytes). The minimum amount is zero, and the maximum cannot exceed the maximum amount of storage defined by the SADMP= parameter. The amount you specify is rounded up to 4K boundaries for DASD data sets, or 64K boundaries for tape data sets.

Instead of the BUF= parameter, use the ABDUMP= parameter in conjunction with SADMP= and SDUMP= on the START GTF command. The system ignores BUF= and uses the defaults for the SADMP=, SDUMP=, or ABDUMP= parameters.

When you code the START command without any parameters, GTF obtains the ABDUMP= parameter from the EXEC parameter in the cataloged procedure. The default is zero, which means that no GTF data will appear in SNAP or ABEND dumps.

If the system takes an ABEND or SNAP dump, you can use IPCS to format this storage.

BLOK = {nnnnn|10}
allows you to specify the number (1 to 99999) of pages of common storage to contain the GTF trace records. The pages of storage will reside in ESQA.

When you code the START command without any parameters, GTF obtains the BLOK= parameter form the EXEC statement in the cataloged procedure. The default is 10 pages of storage.

TIME = {YES|NO}

YES
requests that every trace record be time-stamped in addition to the block time stamp associated with every block of data. The time stamp is the 8-byte TOD clock value at the local time the record is put into the trace buffers. (TOD clock values are described in *Principles of Operation.*)

When TIME=YES is specified and trace records are formatted and printed by IPCS, a timestamp record follows each trace record. These timestamp records can be used to calculate the elapsed time between trace entries. The timestamp record is described in *Using Dumps and Traces*.

When you code the START command without any parameters, GTF obtains the TIME= parameter from the EXEC parameter in the cataloged procedure. The default is TIME=NO.

NO

requests no time stamping of individual trace records. That is, no time stamp recording or place-holder is kept for the trace record.

DEBUG={YES|NO}

YES

requests that all error recovery be bypassed, making all errors terminate GTF.

When DEBUG=YES is in effect and an error occurs in the tracing process, GTF issues an error message and immediately terminates, whether or not the error is recoverable.

When you code the START command without any parameters, GTF obtains the DEBUG= parameter from the EXEC parameter in the cataloged procedure. The default is DEBUG=NO.

NO

requests that GTF attempt to recover from an error, and continue.

When DEBUG=NO is in effect and an error occurs in the tracing process, GTF issues an error message but does not terminate.

MEMBER={GTFPARM|userparm}

specifies the member of SYS1.PARMLIB that contains the GTF trace options. If not specified in the START command, the IBM-supplied GTF procedure specifies the SYS1.PARMLIB member GTFPARM. See Figure 1-3 on page 1-7.

keyword=option

specifies parameters to override or add to JCL parameters, especially DD 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 specify an existing data set, code DISP=OLD. (Note: If you specify DISP=MOD, GTF will change the data set disposition to OLD.)
- To specify a REGION parameter, code REG=value K. Note that the minimum value is 800 and that "K" must be included. See Figure 1-7 on page 1-26 for further GTF storage information.

IBM-Supplied Cataloged Procedure

An IBM-supplied cataloged procedure for GTF is supplied in SYS1.PROCLIB with a member name of GTF. The format of the cataloged procedure is shown in Figure 1-2.

```
//GTF        PROC   MEMBER=GTFPARM
//IEFPROC    EXEC   PGM=AHLGTF,REGION=2880K,TIME=1440,
//                  PARM=('MODE=EXT,DEBUG=NO,TIME=NO')
//IEFRDER    DD     DSNAME=SYS1.TRACE,UNIT=SYSDA,
//                  SPACE=(4096,20),DISP=(NEW,KEEP)
//SYSLIB     DD     DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

Figure 1-2. IBM-Supplied Cataloged Procedure

The following description explains the statements in the cataloged procedure:

PROC Statement
defines the cataloged procedure GTF.

EXEC Statement
calls for the execution of AHLGTF.

IEFRDER DD Statement
defines the trace output data set, according to the following defaults: the trace output data set has the name SYS1.TRACE; it is directed to a direct access device with sufficient allocation to allow the data set to contain twenty 4096-byte physical blocks. When the primary allocation is filled, recording continues at the beginning of the data set.

Note that the data set and attributes on the IEFRDER may be changed using the START command.

If the TRACE data set is directed to tape on the START command, normal end-of-volume processing occurs.

SYSLIB DD Statement (Optional)
defines a member in the SYS1.PARMLIB data set that contains GTF options. If such a member exists, GTF uses the options in the member. If the member does not exist, GTF issues an error message and stops.

If you start GTF with a procedure which does not contain a SYSLIB DD statement, GTF issues message AHL100A. This message requests that you supply trace options through the console.

IBM-Supplied SYS1.PARMLIB Member

The GTF cataloged procedure automatically invokes the GTFPARM member of SYS1.PARMLIB. Figure 1-3 shows the format of the GTFPARM member in SYS1.PARMLIB. The options in GTFPARM cause GTF to record specific events. See the topic "GTF Trace Options" on page 1-9 for an explanation of the options.

```
TRACE=SYSM,USR,TRC,DSP,PCI,SRM
```

Figure 1-3. GTFPARM Member in SYS1.PARMLIB

How to Specify GTF Trace Options

You select trace options by either directly specifying each option through the system console or retrieving a set of options previously stored as a member of SYS1.PARMLIB. When you start GTF using the IBM-supplied cataloged procedure, GTF retrieves trace options from the GTF-defined member in SYS1.PARMLIB. If you set up a GTF cataloged procedure, you may define the SYS1.PARMLIB member and GTF retrieves trace options from it. If you do not define options, you must specify them directly through the console.

How GTF Identifies Options in SYS1.PARMLIB

GTF identifies the options set up in SYS1.PARMLIB by issuing the console messages AHL121I and AHL103I. You have the opportunity either to accept these options or to reject them and respecify your own. This sequence appears as:

```
AHL121I SYS1.PARMLIB INPUT INDICATED
AHL103I TRACE OPTIONS SELECTED -- options from SYS1.PARMLIB
AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
```

Some GTF options cause GTF to prompt you for keywords if you specify these options through the system console. If the SYS1.PARMLIB member contains any of these options, GTF will not prompt you for keywords; the keywords must also appear in the member.

If you choose to reject the options in the SYS1.PARMLIB member, you will completely override all options specified in that member. Respecifying trace options in response to AHL125A is not a method of modifying the options in SYS1.PARMLIB.

If you start GTF with a user procedure that does not contain a SYSLIB DD statement, you must reply to supply options to the following message:

```
AHL100A SPECIFY TRACE OPTIONS
```

How You Indicate Trace Options

To respecify new options or specify options for the first time, you respond to the message AHL125A or AHL100A with TRACE= keyword, to indicate events to be traced during GTF execution. The format of this response is:

```
TRACE=trace option[,trace option]...
```

Note that the trace options you specify determine the GTF storage requirements. See Figure 1-7 on page 1-26.

GTF Trace Options

You can specify the following trace option values:

ASIDP
requests that GTF tracing be limited to a subset of address spaces. ASIDP requests GTF prompting for one to five address space identifiers in which you want GTF tracing to occur. ASIDP only works when you also specify a GTF option that generates tracing, such as SVC or IO.

CCW
requests tracing of channel programs and associated data for I/O events. CCW is valid only if the other trace options you specify include SSCH, SSCHP, IO, or IOP.

CCWP
requests tracing of channel programs and associated data for I/O events, and requests GTF prompting for the following information: tracing CCWs for SSCH operations or I/O interruptions or both; maximum number of CCWs for each event; maximum number of bytes of data for each CCW; optional IOSB and EWA tracing; and size of the PCI table. For information on responding to GTF prompts, see the topic "Prompting" on page 1-13. CCWP is valid only if the other trace options you specify include SSCH, SSCHP, IO, or IOP.

CSCH
requests recording for all clear subchannel operations.

DSP
requests recording for all dispatchable units of work (that is, SRB, LSR, TCB and SVC prologue dispatch events). When you specify both SYSM and DSP, GTF records minimal trace data for DSP. Otherwise, GTF records comprehensive trace data for DSP.

EXT
requests comprehensive recording for all external interruptions.

HSCH
requests recording for all halt subchannel operations.

IO requests recording of all non-program-controlled I/O interruptions. Unless you also specify the PCI trace option, GTF does not record program-controlled interruptions.

IOP
requests GTF prompting for specific device numbers for which you want GTF to record I/O interruptions. Unless you also specify the PCI trace option, GTF does not record program-controlled interruptions. For information on responding to GTF prompts, see "Prompting" on page 1-13.

JOBNAMEP
requests that GTF tracing be limited to a subset of jobs. JOBNAMEP requests GTF prompting for one to five jobnames for which you want GTF tracing to occur. JOBNAMEP only works when you also specify a GTF option that generates tracing, such as SVC or IO.

MSCH
requests recording for all modify subchannel operations.

PCI

requests recording of intermediate status interruptions in the same format as other I/O trace records that GTF creates. Specifically, PCI causes GTF to record program-controlled I/O interruptions, initial status request interruptions, and resume and suspend channel program interruptions. When you select specific devices as a result of prompting for I/O events (IOP), GTF records intermediate status interruptions only for those devices. PCI only works when you specify PCI and the other trace options that you specify include IO, IOP, SYS, SYSM, or SYSP.

PI requests comprehensive recording for all program interruptions (0-255).

PIP

requests GTF prompting for those interruption codes for which you want GTF to record program interruptions. For information on responding to GTF prompts, see "Prompting" on page 1-13.

RNIO

requests recording of all VTAM network activity. When you specify both SYSM and RNIO, GTF records minimal trace data for RNIO. Otherwise, GTF records comprehensive trace data for RNIO.

Note: For successful processing, VTAM trace must be active.

RR

requests comprehensive recording of data associated with all invocations of functional recovery routines (such as STAE and ESTAE routines). GTF creates a trace record describing the activity of the recovery routine when control passes from the recovery routine back to the recovery termination manager (RTM).

$\left\{\begin{array}{l}\textbf{SIO}\\ \textbf{SIOP}\end{array}\right\}$

If you request the SIO or SIOP option, GTF processes your request as a request for SSCH or SSCHP. GTF issues message AHL138I to indicate this substitution. Subsequent messages refer to the original SIO or SIOP option.

SLIP

requests that a trace entry be made each time that a match occurs for a SLIP trap with a tracing action specified or each time a SLIP trap with the SLIP DEBUG option is checked. The amount of data and the type of SLIP trace record to be built is specified on the SLIP command. The SLIP option is not included in the specification of SYS or SYSM; it must be specified additionally. Specification of the SYS or SYSM option does not affect the data collected on the SLIP trace record.

SRM

requests recording of trace data each time the system resource manager is invoked. When you specify both SYSM and SRM, GTF records minimal trace data for SRM. Otherwise, GTF records comprehensive trace data for SRM. Further information regarding the use of this option is in *Initialization and Tuning.*

SSCH

requests recording for start subchannel and resume subchannel operations.

SSCHP

requests GTF prompting for the specific device numbers for which you want GTF to record start subchannel and resume subchannel events. For information on responding to GTF prompts, see the topic "Prompting" on page 1-13.

SVC

requests comprehensive recording for all SVC interruptions.

SVCP

requests GTF prompting for those SVC numbers for which you want data recorded. For information on responding to GTF prompts, see the topic "Prompting" on page 1-13.

SYS

requests recording of comprehensive trace data for all external interruptions (EXT), program interruptions (PI), recovery routines (RR), and supervisor call interruptions (SVC). SYS causes recording of all I/O interruptions (IO), start subchannel and resume channel operations (SSCH), clear subchannel operations (CSCH), halt subchannel operations (HSCH), and modify subchannel operations (MSCH). When you specify DSP, RNIO, or SRM in addition to SYS, GTF produces comprehensive trace data for those events.

Note: Specification of SYS, SYSM, or SYSP causes GTF to ignore the following trace options if you specify them in any form: CSCH, HSCH, MSCH, SSCH, IO, SVC, PI, EXT, RR.

SYSM

requests recording of minimal trace data for all external interruptions (EXT), program interruptions (PI), recovery routines (RR), and supervisor call interruptions (SVC). SYSM causes recording of all I/O interruptions (IO), start subchannel and resume channel operations (SSCH), clear subchannel operations (CSCH), halt subchannel operations (HSCH), and modify subchannel operations (MSCH). When you specify DSP, RNIO, or SRM in addition to SYSM, GTF produces minimal trace data for those events.

Note: Specification of SYS, SYSM, or SYSP causes GTF to ignore the following trace options if you specify them in any form: CSCH, HSCH, MSCH, SSCH, IO, SVC, PI, EXT, RR.

SYSP

requests recording for the same events as the SYS option, but causes GTF to prompt you for selection of specific SVC, IO, SSCH, and PI events that you want recorded. When you specify DSP, RNIO, or SRM in addition to SYSP, GTF produces comprehensive trace data for those events. For information on responding to prompts, refer to the topic "Prompting" on page 1-13.

Note: Specification of SYS, SYSM, or SYSP causes GTF to ignore the following trace options if you specify them in any form: CSCH, HSCH, MSCH, SSCH, IO, SVC, PI, EXT, RR.

TRC

requests recording of those trace events which are associated with GTF itself. Unless you request TRC, the GTF associated events are filtered out and not recorded. TRC only works when you also specify a GTF option that generates tracing, such as SVC or IO.

USR

requests recording of all data that the GTRACE macro passes to GTF. You must specify USR or USRP if you want to code the GTRACE macro. When you code the GTRACE macro but do not specify USR or USRP, GTF ignores the GTRACE macro.

USRP

causes GTF prompting for specific event identifiers (EIDs). See "Prompting" on page 1-13. GTF builds an internal table of the EIDs that you specify. The TEST parameter of the GTRACE macro tests whether or not tracing is active for the EIDs that you specify for USRP. USRP does *not* limit GTF tracing to those user EIDs that you specify. The purpose of USRP is to cause GTF to build an internal table of EIDs that GTF uses when you specify TEST=YES on the GTRACE macro.

The GTRACE data consists of user event trace records and/or IBM subsystem event records. The subsystems are VTAM, JES2, OPEN/CLOSE/EOV, SAM/PAM/DAM, and VSAM.

Combining Certain GTF Options

Figure 1-4 shows those TRACE= options that GTF will *not* use in combination. If you specify two or more options from the same row, GTF uses the option that has the lower column number and ignores the other options. For example, if you specify both SYSP and PI (see row D), GTF uses SYSP (column 2) and ignores PI (column 5).

	1	2	3	4	5
A	SYSM	SYSP	SYS	SSCHP	SSCH
B	SYSM	SYSP	SYS	IOP	IO
C	SYSM	SYSP	SYS	SVCP	SVC
D	SYSM	SYSP	SYS	PIP	PI
E	SYSM	SYSP	SYS	EXT	
F	SYSM	SYSP	SYS	RR	
G	SYSM	SYSP	SYS	CSCH	
H	SYSM	SYSP	SYS	HSCH	
I	SYSM	SYSP	SYS	MSCH	
J	CCWP	CCW			
K	USRP	USR			

Figure 1-4. Combining Certain GTF Options

Prompting

When you specify ASIDP, CCWP, IOP, JOBNAMEP, PIP, SSCHP, SVCP, SYSP, or USRP as trace options, GTF prompts you to supply specific values by the following message:

```
AHL101A SPECIFY TRACE EVENT KEYWORDS -- keyword=,...,keyword=
```

The keywords in the message correspond to those trace options that cause prompting (ASID=, CCW=, IO=, JOBNAME=, PI=, SSCH=, SVC=, SYS=, or USR=). GTF accepts only these keywords in your reply. If you specify SYSP, the valid keywords are: IO=, IO=SSCH=, SSCH=, PI=, and SVC=. Specify only those keywords for which you want specific event recording. Keywords not specified default to cause recording of all events within those classes.

END is also a keyword and signifies that the event definition is complete. If END is not encountered in a reply, GTF prompts the operator to continue specification. Event keywords are as follows:

ASID=(asid1[,asidn]...[,asid5])
specifies one to five address space identifiers in which you want GTF tracing to occur. The values 'asid1' through 'asid5' are hexadecimal numbers from X'0001' to the maximum number of entries in the address space vector table (ASVT). When you specify ASIDP, GTF traces events for the address spaces you specify. If you specify ASIDP, but do not specify ASID= before replying END, then no ASID filtering takes place and GTF traces all address space identifiers.

If you use more than one line to specify ASIDs, GTF stacks your replies until you specify the maximum of five ASIDs. If a line of your reply contains an error in the specification of ASIDs, GTF prompts you to respecify the invalid value, and leaves intact the valid stacked values from other lines of your reply.

Note: If you specify both ASIDP and JOBNAMEP, GTF might trace address spaces that ASIDP did not identify. This occurs if the jobs that JOBNAMEP identified are running in address spaces that ASIDP did not identify.

CCW=[(S|I|SI][,CCWN=nnnnn][,DATA=nnnnn][,IOSB][,PCITAB=n])
specifies different options for tracing channel programs. If you specify CCW= more than once, GTF uses your last specification of CCW=.

If you specify CCWP, but do not specify CCW= before replying END, then the following defaults are in effect:

TRACE OPTIONS SELECTED	CCW SUBPARAMETER DEFAULTS
SSCH or SSCHP	S
IO or IOP	I
SSCH or SSCHP, and IO or IOP	SI
PCI	PCITAB=1
ANY	CCWN=50
ANY	DATA=20

```
Examples:
TRACE=IO,CCWP            CCW defaults to:  CCW=(I,CCWN=50,DATA=20)
TRACE=IOP,SSCH,PCI,CCWP  CCW defaults to:  CCW=(SI,CCWN=50,DATA=20,PCITAB=1)
```

Figure 1-5. CCW Defaults for Selected TRACE Options

If you specify an option more than once in one line, GTF uses your last specification of that option. An exception is that GTF uses your first specification of S, I, or SI. If a line contains an error, GTF prompts you to respecify the invalid value.

S|I|SI
specifies the type of I/O event for which you want channel programs traced. If you specify more than one option, GTF uses the first option that you specified. If you do not specify any option, SI is the default.

S specifies GTF tracing of channel programs for start subchannel and resume subchannel operations. CCW=S only works if you specify SSCH or SSCHP as trace options.

I specifies GTF tracing of channel programs for I/O interruptions, including program-controlled interruptions if you specify PCI as a trace option. CCW=I only works if you specify IO or IOP as trace options.

SI specifies GTF tracing of channel programs for start subchannel and resume subchannel operations and I/O interruptions. CCW=SI only works if you specify either SSCH or SSCHP and either IO or IOP as trace options.

CCWN=nnnnn
specifies the maximum number of CCWs that you want traced for each event. The value 'nnnnn' is a decimal number. It is defined as any integer from 1 to 32767. The default is 50.

DATA=nnnnn
specifies the maximum number of bytes of data that you want traced for each CCW. The value 'nnnnn' is a decimal number. It is defined as any integer from zero to 32767. The default is 20.

GTF treats each CCW that belongs to a chain of 'data-chained' CCWs as one CCW. Therefore, GTF traces 'nnnnn' bytes of data for each CCW on the data chain. GTF also traces 'nnnnn' bytes of data for each word in an IDAW (indirect data addressing word) list.

For start subchannel or resume subchannel operations, GTF does not trace data for read, read backwards, or sense commands in the channel programs. If the skip bit is on, (that is, no data is being transferred) regardless of the type of I/O operation, GTF does not trace data for read, read backwards, or sense commands. When the data count in the CCW is equal to or less than 'nnnnn', GTF traces all data in the data buffer. When the data count in the CCW is greater than 'nnnnn', GTF traces data only from the beginning and end of the data buffer. The first half of the traced data is measured from the start of the data buffer. The second half of the traced data is measured backward from the end of the data buffer. Examination of the traced data shows whether the channel completely filled the buffer on a read operation.

Note: GTF uses a different CCW tracing method for a data transfer that is in progress when an I/O interruption occurs. Instead of using the data count in the CCW, GTF tracing depends on the transmitted data count. The transmitted data count is the difference between the data count in the CCW and the residual count in the CSW. If the residual count in the CSW is greater than the data count in the CCW, then GTF traces all of the data in the CCW. When the transmitted data count is less than or equal to 'nnnnn', GTF traces all of the transmitted data. When the transmitted data count is greater than 'nnnnn', GTF traces data only from the beginning and end of the transmitted data. The first half of the traced data is measured from the start of the transmitted data. The second half of the traced data is measured backward from the end of the transmitted data.

IOSB

specifies tracing of the IOS block (IOSB) and, if available, the ERP work area (EWA), for all CCW events. If you do not specify IOSB, then GTF performs IOSB and EWA tracing only when GTF encounters an exceptional condition when tracing a channel program.

PCITAB = n

specifies a decimal number of 100-entry increments that you want GTF to allocate in an internal PCI table. The value of 'n' is an integer from 1 to 9. The default is 1 (100 entries).

The PCI table keeps track of the channel programs that use PCI. One entry in the PCI table contains information about a program-controlled interruption in one channel program. An entry in the PCI table includes a CCW address and an IOSB address.

GTF initializes an entry in the table when the first program-controlled interruption occurs for an IOSB that represents a channel program requesting PCI. For each subsequent program-controlled interruption that occurs when tracing channel programs, the address of the first CCW traced is taken from the PCI table. When GTF completes tracing for each event, GTF updates the entry in the PCI table by changing the CCW address to equal the CSW address minus eight bytes. GTF deletes the entry when the channel program terminates. If the table is not large enough, GTF writes a message to the trace data set indicating that the GTF trace data might be incorrect.

IO = (devnum1 [,devnumn]...[,devnum50])

specifies one to 50 device numbers (hexadecimal notation) for which you want I/O interruptions traced. All other I/O interruptions are filtered out. If you specify IOP or SYSP and do not specify IO = in response to the prompting messages, no I/O interruption filtering takes place and GTF traces all non-program-controlled interruptions.

IO = SSCH = (devnum1[,devnumn]...[,devnum50])

only valid after you request either SYSP, or both IOP and SSCHP; specifies one to 50 device numbers for which you want GTF to trace both IO and SSCH events. GTF filters out all other IO and SSCH events, except those requested specifically by IO= or SSCH=.

JOBNAME = (jobname1[,jobnamen]...[,jobname5])

specifies one to five jobnames for which you want GTF tracing to occur. The values 'job1' through 'job5' must be valid jobnames. When you specify JOBNAMEP, GTF traces events for the jobs you specify. If you specify JOBNAMEP, but do not specify JOBNAME= before replying END, then no JOBNAME filtering takes place and GTF traces all jobnames.

If you use more than one line to specify jobnames, GTF stacks your replies until you specify the maximum of five jobnames. If any line of your reply contains an error in the specification of jobnames, GTF prompts you to respecify the invalid value, and leaves intact the valid stacked values from other lines of your reply.

Note: If you specify both JOBNAMEP and ASIDP, GTF might trace jobs that JOBNAMEP did not identify. This occurs if the address spaces that ASIDP identified contain jobs that JOBNAMEP did not identify.

PI = (code0[,coden]...[,code255])

specifies one to 256 program interruption codes (decimal notation) that you want traced. All other program interruptions are filtered out. If you specify PIP or SYSP, and do not specify PI= in response to this prompting message, no program interruption filtering takes place and GTF traces all program interruptions.

SSCH = (devnum1[,devnumn]...[,devnum50])

specifies one to 50 device numbers (hexadecimal notation) for which you want SSCH operations traced. All other SSCH operations are filtered out. If you specify SSCHP or SYSP, and do not specify SSCH= in response to the prompting message, no SSCH filtering takes place and GTF traces all SSCH operations.

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

specifies one to 50 SVC numbers (decimal notation) that you want traced. All other SVC numbers are filtered out. If you specify SVCP or SYSP, and do not specify SVC= in response to the prompting message, no SVC filtering takes place and GTF traces all SVC numbers.

USR = (event1[,eventn]...[,event50])

specifies one to 50 user event identifiers (EIDs) that you want GTF to test when you specify TEST=YES on the GTRACE macro. When you specify TEST=YES, GTF tests whether or not you specified the EID in the list of EIDs that you selected for USRP. The values 'event1' through 'event50' are three-digit hexadecimal numbers from X'000' to X'FFF'. If you specify USRP and do not specify USR= in response to the prompting message, all executions of GTRACE using TEST=YES return an indication that tracing is not active.

USRP does *not* limit GTF tracing to those user EIDs that you specify. The purpose of USRP is to cause GTF to build an internal table of EIDs that GTF uses when you specify TEST=YES on the GTRACE macro.

Notes:

- GTF imposes a limit on the number of specific values you can supply through prompting. When you exceed this limit, GTF issues a message and you must respecify all values.
- You may specify one to 50 device numbers for IO= or SSCH=; you may specify one to 50 device numbers for IO=SSCH=. However, the sum of device numbers that you specify using IO= and IO=SSCH= may not exceed 50; likewise the sum of device numbers that you specify using SSCH= and IO=SSCH= may not exceed 50.
- The device number is *not* the same as the subchannel number. You must specify device numbers for IO=, IO=SSCH=, and SSCH=.
- Within a given reply, each keyword that you specify must be complete. If you need to specify more events for the same category, respecify the keyword in a subsequent reply with the additional events as follows:

  ```
  Reply #1 IO=(191,192,193),SVC=(1,2,3,4,5)
  Reply #2 SVC=(6,7,8,9,10)
  ```

- If you use more than one reply to specify values for the same keyword, the maximum number of values you can specify for that keyword does not change. For example:

  ```
  Reply #1 IO=(191,192,193),ASID=(1,C)
  Reply #2 ASID=(3,A,B)
  ```

  Although you use two replies to specify ASID=, the maximum number of ASIDs you can specify is still 5.

- To ensure recording IO events for a device with multiple addresses, specify all addresses in the reply.
- If END is not encountered within a reply, the following message prompts for further specification by the user/operator:

```
    AHL102A CONTINUE TRACE DEFINITION OR REPLY END
```

  When trace option specification is complete, the operator is notified which trace parameters are accepted. (Message AHL103I).

- For sample prompting sequences, refer to "Example 6: Prompting Keywords Stored in SYS1.PARMLIB", "Example 7: Specifying Which System Events GTF Traces, Using Trace Options SYSP and USRP", and "Example 9: Specifying Which System Events GTF Traces, Using Trace Options SSCHP, IOP, PCI, CCWP, SVC, and JOBNAMEP".
- Prompting increases GTF storage requirements. Refer to Figure 1-7 on page 1-26 GTF Storage Requirements for further information.

GTF Examples

Using the GTF Cataloged Procedure

Example 1: Initialization

You initialize GTF by starting a cataloged procedure that indicates the parmlib member GTFPARM. (See example 4). The trace options are specified in the parmlib member record. In this example, the options are TRACE=SYSM, DSP, PCI, SRM, TRC, USR. This example shows the messages and reply, (r), generated by the initial START command.

```
START GTF.EXAMPLE1

AHL121I SYS1.PARMLIB INPUT INDICATED

AHL103I TRACE OPTIONS SELECTED--SYSM,USR,TRC,DSP,PCI,SRM

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 00,U

AHL031I GTF INITIALIZATION COMPLETE
```

Example 2: Internal Tracking

This example shows GTF started with MODE=INT. The trace data is maintained in virtual memory and is not recorded on an external device. In this example, the operator overrides the trace options given in the supplied SYS1.PARMLIB member.

```
START GTF.EXAMPLE2,,,(MODE=INT),DSN=NULLFILE

AHL121I SYS1.PARMLIB INPUT INDICATED

AHL103I TRACE OPTIONS SELECTED - SYSM,USR,TRC,DSP,PCI,SRM

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 00,TRACE=IO,SSCH,SVC,DSP

AHL103I TRACE OPTIONS SELECTED -- DSP,SVC,IO,SSCH

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 01,U

AHL031I GTF INITIALIZATION COMPLETE
```

Example 3: Tracing Output to an Existing Data Set on Tape

This example shows how the START command is used to direct GTF trace output to an existing data set residing on tape rather than to an existing data set residing on a DASD. The device type and volume serial number are supplied. The disposition and name of the trace data set are changed from DISP=(NEW,KEEP) and DSNAME=SYS1.TRACE to DISP=(OLD,KEEP) and DSNAME=TPOUTPUT. The specified tape has a volume serial of TRCTAP and resides on a 3400 tape drive. Note that the GTFPARM member of SYS1.PARMLIB is used to specify the trace options.

```
START GTF,3400,TRCTAP,(MODE=EXT),DISP=OLD,DSNAME=TPOUTPUT

AHL103I TRACE OPTIONS SELECTED--SYSM,DSP,PCI,SRM,TRC,USR

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 00,U

AHL031I GTF INITIALIZATION COMPLETE
```

Example 4: Storing Trace Options in SYS1.PARMLIB

To save time when starting GTF, first store one or more combinations of trace options as members in SYS1.PARMLIB, and include a SYSLIB DD statement in the cataloged procedure. If you do this, GTF will retrieve the trace options from SYS1.PARMLIB, instead of prompting you to supply them through the console. GTF will display the trace options for you, and then issue AHL125A, to which you reply 'U' to accept the SYS1.PARMLIB options.

This example shows the job control statements and utility control statements needed to add trace options to SYS1.PARMLIB using IEBUPDTE:

```
//GTFPARM    JOB     MSGLEVEL=(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=(D34,D0C),SSCH=ED8,PI=15
./       ADD     NAME=GTFB,LIST=ALL,SOURCE=0
TRACE=IO,SSCH,TRC
./       ADD     NAME=GTFC,LIST=ALL,SOURCE=0
TRACE=SYS,PCI
/*
```

For full descriptions of the statements, refer to *Utilities* and *JCL Reference*. For further information regarding SYS1.PARMLIB, refer to *Initialization and Tuning*.

A sample SYSLIB DD statement to be included in a GTF cataloged procedure might look like this:

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

The new member name can also be specified on the START command while using the IBM-supplied GTF procedure, as in the following example:

```
S GTF,,,(MODE=EXT,TIME=YES),MEMBER=GTFB
```

Example 5: Starting GTF With a User Cataloged Procedure That Does Not Have a SYSLIB DD Statement

When GTF is started with a user procedure containing no SYSLIB DD statement, the operator receives the following message:

```
AHL100A SPECIFY TRACE OPTIONS
```

The operator must then reply with the TRACE= keyword to specify the events to be recorded during GTF execution.

In the following example, a user cataloged procedure (USRPROC) is invoked to start GTF in external mode to a direct access data set, ABCTRC, on device 250. The trace options selected by the operator result in trace data being gathered for all SVC and IO interruptions, for all SSCH operations, for all matching SLIP traps with a tracing action specified or SLIP traps in DEBUG mode, and for all dispatcher events. Also, all issuers of the GTRACE macro will have their user data recorded in the trace buffers. The trace data is written into the data set ABCTRC. (Note that when the end of the primary extent is reached, writing continues at the beginning).

```
START USRPROC,250,333005,(MODE=EXT),DSN=ABCTRC

00 AHL100A SPECIFY TRACE OPTIONS

R 00,TRACE=SVC,SSCH,IO,DSP,SLIP,USR

AHL103I TRACE OPTIONS SELECTED--USR,DSP,SVC,IO,SLIP,SSCH

01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 01,U

AHL031I GTF INITIALIZATION COMPLETE
```

Example 6: Prompting Keywords Stored in SYS1.PARMLIB

Some GTF options cause GTF to prompt you for keywords if you specify these options through the system console. If the SYS1.PARMLIB member contains any of these options, GTF will not prompt you to enter keywords through the console; the prompting keywords must appear in the member's records. A SYSLIB DD statement in a cataloged procedure causes the prompting keywords to be read from the specified SYS1.PARMLIB member. The second and subsequent logical records in the member should contain only those keywords for which prompting is allowed.

Prompting input from PARMLIB is complete when either the END keyword is encountered, or when end-of-file is reached on the member. Each keyword must be complete for each prompting record. If the need arises to indicate more events for the same keyword, respecify the keyword in a subsequent prompting record with the additional events as follows:

```
Record #1 TRACE=IOP,SVCP,SSCH

Record #2 IO=(D34,D0C),SVC=(1,2,3)

Record #3 SVC=(4,5,6,7,8,9,10),END
```

At this point, do not attempt to respecify the keyword through the system console, or you will override all of the options and keywords in the SYS1.PARMLIB member.

When GTF finishes reading the options and prompting keywords in the SYS1.PARMLIB member, it displays the options through message AHL103I:

```
AHL103I TRACE OPTIONS SELECTED--SYSP,USR
AHL103I IO=(D34,D0C),SSCH=(ED8),SVC=(1,2,3,4,10)
```

This message may be multilined depending on the number of options selected by the operator. If the set of devices specified for IO= and SSCH= are identical, message AHL103I will show them as if specified by use of IO=SSCH.

After GTF displays all of the options specified, it gives you the opportunity to accept the SYS1.PARMLIB options, or completely change the options by respecifying them through the console:

```
AHL125A RESPECIFY TRACE OPTIONS OR REPLY U.
```

Example 7: Specifying Which System Events GTF Traces, Using Trace Options SYSP and USRP

In this example, the operator started GTF in external mode to the data set defined in the cataloged procedure. The operator selected two trace options in reply 00. SYSP requests that GTF trace specific system event types; USRP requests that GTF trace specific user entries that the GTRACE macro generates. Message AHL101A instructed the operator to specify values for the SVC, IO, SSCH, PI, and USR keywords. In reply 01, the operator selected five SVCs, two devices for non-program-controlled I/O interruptions, one device for SSCH operations, and three user event identifiers. GTF does not record any other SVC, IO, and SSCH events. Because the operator did not specify any program interruption codes for PI=, GTF would trace all program interruptions.

```
START MYPROC.EXAMPLE7,,,(MODE=EXT)

00 AHL100A SPECIFY TRACE OPTIONS
R 00,TRACE=SYSP,USRP
01 AHL101A SPECIFY TRACE EVENT KEYWORDS--IO=,SSCH=,SVC=,PI=,USR=
01 AHL101A SPECIFY TRACE EVENT KEYWORDS--IO=SSCH=
R 01,SVC=(1,2,3,4,10),IO=(191,192),USR=(10,07A,AB)
02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END
R 02,SSCH=282,END
AHL103I TRACE OPTIONS SELECTED--SYSP,PI,IO=(191,192),SSCH=(282)
AHL103I SVC=(1,2,3,4,10),USR=(010,07A,0AB)
03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 03,U
```

Example 8: Starting GTF to Trace VTAM Remote Network Activity

GTF can be used to trace VTAM activity only if VTAM is started with the GTF option. See VTAM Operations for details. In the following example, GTF options are not stored in SYS1.PARMLIB; the operator enters the trace options directly at the console. Three GTF options are required to record all VTAM traces:

- RNIO must be specified so that the VTAM I/O trace can function for an NCP or a remote device attached to the NCP.
- IO or IOP must be specified so that the VTAM I/O trace can function for a local device.
- USR must be specified so that the VTAM buffer and the NCP line traces can function.

GTF must be started with the GTF START command before a trace can be activated from VTAM.

```
START MYPROC.EXAMPLE8,,,(MODE=EXT,TIME=YES)

00 AHL100A SPECIFY TRACE OPTIONS

R 00,TRACE=RNIO,IO,USR

AHL103I TRACE OPTIONS SELECTED--IO,USR,RNIO

01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 01,U

AHL031I GTF INITIALIZATION COMPLETE
```

Example 9: Specifying Which System Events GTF Traces, Using Trace Options SSCHP, IOP, PCI, CCWP, SVC, and JOBNAMEP

In this example, the operator started GTF in external mode to the data set defined in the cataloged procedure. The operator selected six trace options in reply 00. Message AHL101A instructed the operator to specify values for the IO, SSCH, CCW, and JOBNAME keywords. In reply 01 the operator selected one device for tracing both IO and SSCH events, limited GTF tracing to one job, and specified five options for CCW tracing. As a result of the operator's specifications, GTF would trace CCWs for both start subchannel operations and I/O interruptions at device 580 for the job BACKWARD, and all SVCs in BACKWARD's address space. GTF would allocate 200 entries in the PCI table, and trace up to 100 CCWs, up to 40 bytes of data for each CCW, and the IOSB.

```
START USRPROC,,,(MOD=EXT)

00 AHL100A SPECIFY TRACE OPTIONS

R 00, TRACE=SSCHP,IOP,PCI,CCWP,SVC,JOBNAMEP

01 AHL101A SPECIFY TRACE EVENT KEYWORDS
        --IO=,SSCH=,CCW=,JOBNAME=,IO=SSCH=

R 01,JOBNAME=(BACKWARD),IO=SSCH=580

02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END

R 02,CCW=(CCWN=100,DATA=40,PCITAB=2,IOSB,SI),END

AHL103I TRACE OPTIONS SELECTED--PCI,SVC,IO=SSCH=(580)

AHL103I CCW=(SI,IOSB,CCWN=100,DATA=40,PCITAB=2)

AHL103I JOBNAME=(BACKWARD)

03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 03,U
```

How to STOP GTF

To stop GTF processing, you specify the STOP command and either the GTF identifier that you specified in the START command, or the device number of the GTF trace data set. See the description under "How to Specify the START Command" on page 1-3. If you are not sure of the identifier or device number, use the operator display command:

```
DISPLAY ACTIVE,LIST
```

This command causes the system to display the number of:

- Active batch jobs
- Active time sharing users
- Mount commands in execution
- Started tasks, including GTF

The LIST parameter causes the system to include jobnames and V=R region boundaries in the A display.

How to Specify the STOP Command

Figure 1-6 shows the general format of the STOP command that you use to stop GTF processing. The STOP command is similar to the START command. When you enter either the START or STOP command, you must enter the command from a master console.

```
{STOP|P}{{identifier|GTF}|device number}
```

Figure 1-6. General Format of the STOP Command

The identifier on the STOP command is the same identifier that you specified on the START command when you started GTF. The device number on the STOP command is the same device number that you specified on the START command when you started GTF. If you started GTF in internal mode and did not specify an identifier, the identifier is 'GTF'.

You may enter the STOP command at any time during GTF processing. The STOP command stops anything that satisfies the parameters on the STOP command. For example, if you start both an external writer and GTF with the identifier 162, and later specify 'STOP 162', both the external writer and GTF stop.

Note: If GTF does not respond to the STOP command with message AHL006I, then the STOP command is not in effect and the GTF session continues. The GTF session remains active in the system until the next IPL. The CANCEL and FORCE commands have no effect because GTF is a non-cancellable system task. However, you can stop GTF by using the FORCE ARM command. Refer to *Operations: System Commands* of the FORCE ARM command.

Sample STOP Commands

Example 1: Using the Identifier

This example starts a GTF session with the identifier EXAMPLE and with trace data maintained in the GTF address space. The DSN keyword is entered to prevent allocation of an external trace data set as specified in the cataloged procedure.

```
START GTF.EXAMPLE,,,(MODE=INT),DSN=NULLFILE
```

This command would stop the GTF session started in the previous example:

```
STOP EXAMPLE
```

Example 2: Using the Device Number

This example starts a GTF session with trace data recorded on the non-labeled tape on device 282. Each trace record will be timestamped. Twenty kilobytes of GTF data will be formatted if an SVC dump is taken.

```
START GTF,282,,(TIME=YES,SDUMP=20K),LABEL=(,NL)
```

This command would stop the GTF session started in the previous example:

```
STOP 282
```

Example 3: When You Must Display Active Jobs

This example starts a GTF session with trace data recorded on an external device. Since it is not apparent which is the GTF recording device, you have to display active jobs with the D A,LIST command before you can stop GTF. The GTF session started in this example could run in an address space of a maximum of 1000K.

```
START GTF,,,(MODE=EXT),REGION=1000K
```

GTF Storage Requirements

Extended Pageable Link Pack Area

Fix = Opt + Prmpt + 8K

Fix: Fixed storage in pageable EPLPA while GTF is active.

Opt: Sum of storage required for each GTF option specified. See the table below to calculate OPT.

Prmpt: Optional additional 1.5K if any prompting options specified.

8K: 8K required for services.

Option	Size Required
SYSM	4K
SYS with DSP and/or SRM and/or RNIO	7K
SYS, SYSP	18K
PI, DSP, PIP	2.5K
EXT	2K
IO, IOP, SIO, SIOP, SSCH, SSCHP	6K
SVC, SVCP	10K
SRM, RR, RNIO	3K
SLIP	8K
USR, USRP	1.5K
PCI, TRC	No Requirement
CCW, CCWP	9.3K

Notes:

1. When you specify more than one event from a line, the size requirement is the same as if you specified only one option i.e., DSP and PI require 2.5K.
2. For the maximum storage requirement round up the storage requirement for each option you specified, to the nearest 4K boundary.
3. For the minimum storage requirement, round up the 'FIX' value to the nearest 4K boundary.

Example —

1) Options = IOP, SSCHP, SVC
   Fix = 10.5 + 1.5 + 8 = 20K minimum or
   = 12 + 1.5 + 8 = 21.5 = 24K maximum
2) Options = SYSM, SRM, USR, TRC
   Fix = 8.5 + 0 + 8K = 16.5 = 20K minimum or
   = 12 + 0 + 8K = 20K maximum

System Queue Area

SQA = 16500 + REG + SAVE + CBLOC

SQA: System Queue Area storage requirement.

REG: 232 bytes per processor are required for register save areas, regardless of whether or not GTF is active.

SAVE: 1352 bytes per processor are required for save/work areas when GTF is active.

CBLOC: 1700-2200 bytes are needed for control blocks when GTF is active.

Notes:

1. When you specify PCI and either CCW or CCWP, GTF requires the following additional SQA storage:
   16 + 1200 * (value of PCITAB in bytes)
2. When you specify either CCW or CCWP, GTF uses 4096 additional bytes of the SQA for each processor.
3. When you specify USRP, GTF uses 4096 additional bytes of the SQA for each processor.

Extended System Queue Area

ESQA = N

N: 4096 times the number of blocks specified on the BLOK = keyword parameter of the GTF START command.

The default is 40960 bytes.

Region Storage

SUBPOOL: GTF uses 4-16K in subpools 5 and 6 for control blocks; this area is fixed while GTF is active.

REGION: GTF requires a minimum of an 800K virtual region to execute. Also, if GTF must hold large amounts of trace data in its address space, it can use a maximum of 750 pages in the page data set. To acquire this space you specify the REGION= parameter on an EXEC card or START command with one of the following values:

1) G + 708K (only if BUF= specifies a value of 57 or defaults.)
2) G + 1400K
3) G + 2080K
4) G + 2770K

G: The amount of address space required for the maximum size combination GTF and BSAM.

Note:

Coding a large REGION size does not mean that GTF will use the maximum available address space. The space is used as long as it is necessary to hold trace data, and then when the trace data is moved into trace data set the space is freed: GTF drops to its normal requirement; G + 40K or G/4K + 10 pages.

Figure 1-7. GTF Storage Requirements

User Trace Data Created With GTRACE

If you want your own trace data to be recorded in the GTF trace buffers, you can use the GTRACE macro instruction to define the data. In one invocation of GTRACE, an application program can record up to 256 bytes of data in a GTF trace buffer. The number of bytes of data in the data field of the GTF trace record is equal to the number of bytes of data that you specify plus 12 bytes. The additional 12 bytes are the GTRACE header, which consists of a 4-byte ASCB address followed by an 8-byte jobname.

GTRACE is effective only when GTF is active and is accepting user data -- that is when GTF was started with at least TRACE=USR specified.

For information on coding the GTRACE macro instruction, see *SPL: Application Development Macro Reference*.

EID Assignment for User Events

Events traced by the GTRACE macro will use an event identifier (EID) from one of the three ranges listed below:

0000-1023	user events
1024-1535	reserved for program products
1536-4095	reserved for IBM components and subsystems

EIDs in the first range are available for general use by all GTF users. EIDs in the second and third ranges are reserved.

How to Print User Data

Like other trace data, information recorded by the GTRACE macro can be printed using IPCS. Also, IPCS allows the writing of user exits to format specific types of data records. For information on writing IPCS user exits, see *IPCS Planning and Customization*.

GTF Error Recovery Handling

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

If you specify DEBUG=YES, GTF does not attempt error recovery. It issues an error message and then terminates, so that the contents of the GTF buffers immediately prior to the error are preserved.

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

- For minor errors in the routine that builds the trace record (the build routine), GTF flags the field in the trace record that led to the error and continues processing. GTF does not issue a message to the operator's console, nor does GTF disable the function that caused the error. Instead, GTF proceeds as if no error had occurred.
- For severe errors in the build routine, GTF flags the entire record that was being built, issues a message to the console, suppresses the error and continues processing without the function that caused the error.
- 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, gathering all events of the type that encountered the error.

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

Note: The termination of GTF does not cause termination of a user's task.

GTF Output

GTF creates two kinds of records: trace records and control records. For information about the format of trace records prior to GTF processing, refer to *Using Dumps and Traces*.

Chapter 2. LIST

Introduction

LIST is a service aid that operates as a problem program. It produces several kinds of output that you need to perform certain diagnostic functions; these functions are described below:

Verifying an object module. LIST produces a formatted listing that contains the external symbol dictionary (ESD), the relocation dictionary (RLD), the text of the program containing instructions and data, and the END record.

Mapping CSECTs in a load module. LIST produces a listing of the load module along with its module map and cross-reference listing, which you can examine to determine the organization of CSECTs within the load module, the overlay structure, and the cross-references for each CSECT.

Verifying the contents of the nucleus. LIST can produce a map and cross-reference listing of a nucleus. The map no longer represents the IPL version of the nucleus and message AMB129I is issued. Use IPCS to format a NUCMAP. For information on using IPCS see *IPCS User's Guide*.

Tracing modifications to the executable code in a CSECT. LIST produces a formatted listing of all information in a 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 CSECT was translated. (Translation data is available only for CSECTs 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 that may have been performed by SPZAP.

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

Mapping the link pack area. LIST produces a map of all modules in the fixed link pack area, the modified link pack area, and the pageable link pack area.

Note: Any load module to be formatted and printed by AMBLIST must have the same format as those created by the linkage editor.

JCL Statements

The minimum partition or region for executing of AMBLIST is 64K for all functions except LISTLPA, which requires 100K.

LIST requires the following JCL statements:

JOB Statement
initiates the job.

EXEC Statement
calls for the execution of AMBLIST.

SYSPRINT DD Statement
defines the message data set.

anyname DD Statement
defines an input data set. This statement cannot define a concatenated data set.

SYSIN DD Statement
defines the data set (in the input stream) that contains LIST control statements.

Control Statements

You control LIST 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 or a non-blank character in column 72 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.

The control statements and their parameters are:

```
LISTLOAD [OUTPUT={MODLIST|XREF|BOTH}][,TITLE=('title',position)]
 [,DDN=ddname][,MEMBER={member|(member1,membern...)}]
  [,RELOC=hhhhhh]
```

OUTPUT = {MODLIST|XREF|BOTH}
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. If you specify a position greater than 80, the indentation from the margin defaults to 1.

Note: Do not punctuate your title with commas; since LIST recognizes a comma as a delimiter, 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, LIST will assume SYSLIB as the default ddname.

MEMBER = {member|(member1,membern...)}
identifies the input load module by member name 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, LIST will print all modules in the data set.

RELOC = hhhhhh

specifies a relocation or base address in hexadecimal of up to eight characters. When the relocation address is added to each relative map and cross-reference address, it gives the absolute main storage address for each item on the output listing. If you omit the RELOC= parameter, no relocation is performed.

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

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. If you specify a position greater than 80, the indentation from the margin defaults to 1.

Note: Do not punctuate your title with commas; since LIST recognizes a comma as a delimiter, 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, LIST will assume SYSLIB as the default ddname.

MEMBER = {member|(member1[,membern]...)}

identifies the input object module by member name 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 (PDS or PDSE). If you do not include the MEMBER= parameter, LIST will assume that the input data set is organized sequentially and that it contains a single, continuous object module.

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

  [,MODLIB]
```

OUTPUT = {IDENT|ALL}

specifies whether LIST should print all CSECT identification records or only those containing AMASPZAP data and user data. If you specify OUTPUT=ALL, all IDRs associated with the module will be printed. If you specify OUTPUT=IDENT, LIST will print only those IDRs that contain SPZAP data or user-supplied data. If you omit this parameter, LIST will assume a default of OUTPUT=ALL. Do not specify OUTPUT if you specify the MODLIB parameter.

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 is 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. If a position greater than 80 is specified, the indentation from the margin defaults to 1.

Note: Do not punctuate your title with commas; since LIST recognizes a comma as a delimiter, anything that follows an embedded comma in a title will be ignored. If the MODLIB parameter is specified, do not indicate a TITLE because it will be ignored.

DDN=ddname

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

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

identifies the input load module by member name 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, LIST will print all modules in the data set. Do not specify MEMBER if you specify the MODLIB parameter.

MODLIB

prevents LIST from printing the module summary. LIST prints the IDRs that contain SPZAP data or user-supplied data. No page ejects occur between modules. When you specify MODLIB, the TITLE= parameter is ignored, and the OUTPUT= or MEMBER= parameters are not valid parameters.

```
LISTLPA [FLPA][,MLPA][,PLPA]
```

LISTLPA

lists the modules in the fixed link pack area, the modified link pack area, and the pageable link pack area. The map includes modules residing in the extended sections of each link pack area. If you do not specify any parameters on the LISTLPA control statement, then LIST maps modules from all three link pack areas.

Note: LIST reflects only the system currently operating.

FLPA

requests mapping of the modules in the fixed link pack area.

MLPA

requests mapping of the modules in the modified link pack area.

PLPA

requests mapping of the modules in the pageable link pack area.

Output

LIST 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, LISTLPA, or MODLIB with LISTIDR; 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 2-1 shows a typical module summary. Note that the linkage editor attributes are not represented by a bit map.

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

MEMBER NAME   IGC0002I        MAIN ENTRY POINT    000000         AMODE OF MAIN ENTRY POINT  31

             ** ALIASES **   ALIAS ENTRY POINT   AMODE OF ALIAS ENTRY POINT

              IGG029ER         0042F0              31
              IGG0290A         0011F0              31
              IGG030CM         000948              31
              IGG0300F         0047B0              31

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

                 ****   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             10  EP-ZERO          11  RLD
               12  EDIT             13  NO-SYMS          14  F-LEVEL          15  NOT-REFR

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

                          MODULE SSI:  NONE
                          APFCODE      00000000
                          RMODE        ANY

             *****LOAD MODULE PROCESSED BY VS LINKAGE EDITOR
```

Figure 2-1. Sample Module Summary of LISTLOAD

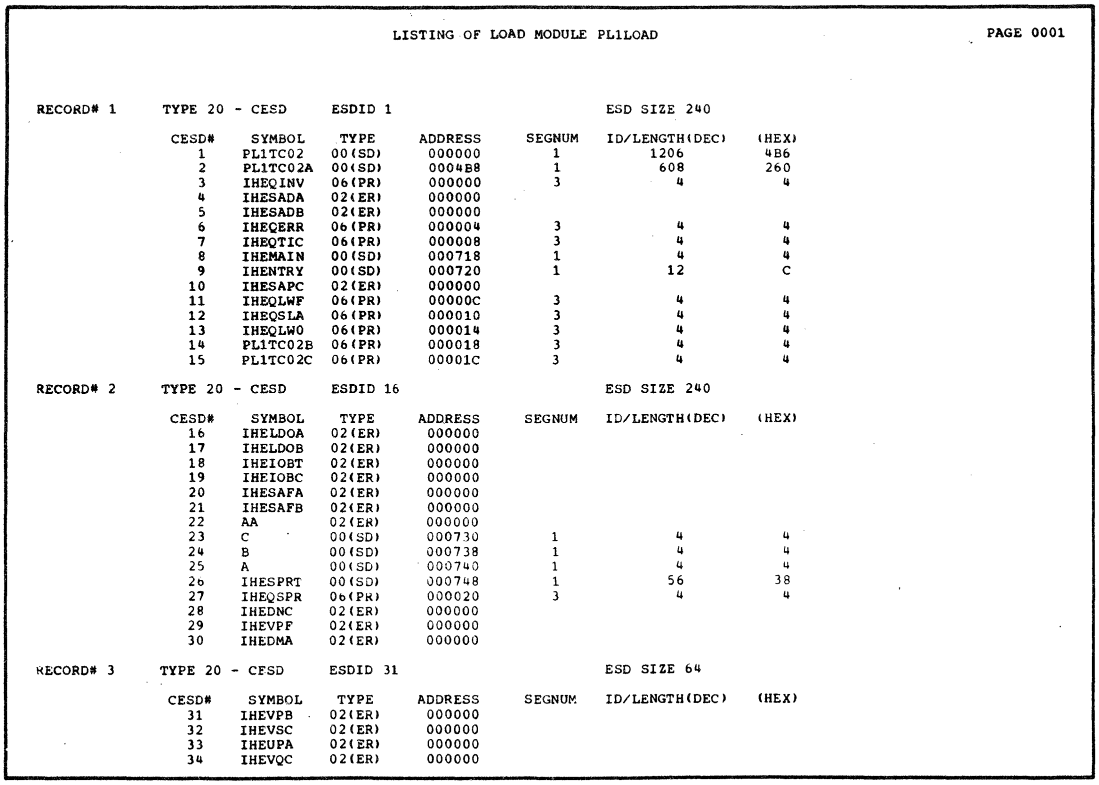
The third page of the listing (or, for LISTOBJ, LISTLPA, or MODLIB with LISTIDR the second page) is the beginning of the formatted output itself.

For LISTLOAD, the formatted output consists of the load module, or the module map and cross-reference listing, or both. Figure 2-2 on page 2-8 shows an example of LISTLOAD module map output. Figure 2-3 on page 2-10 shows an example of the cross-reference listing for the same module.

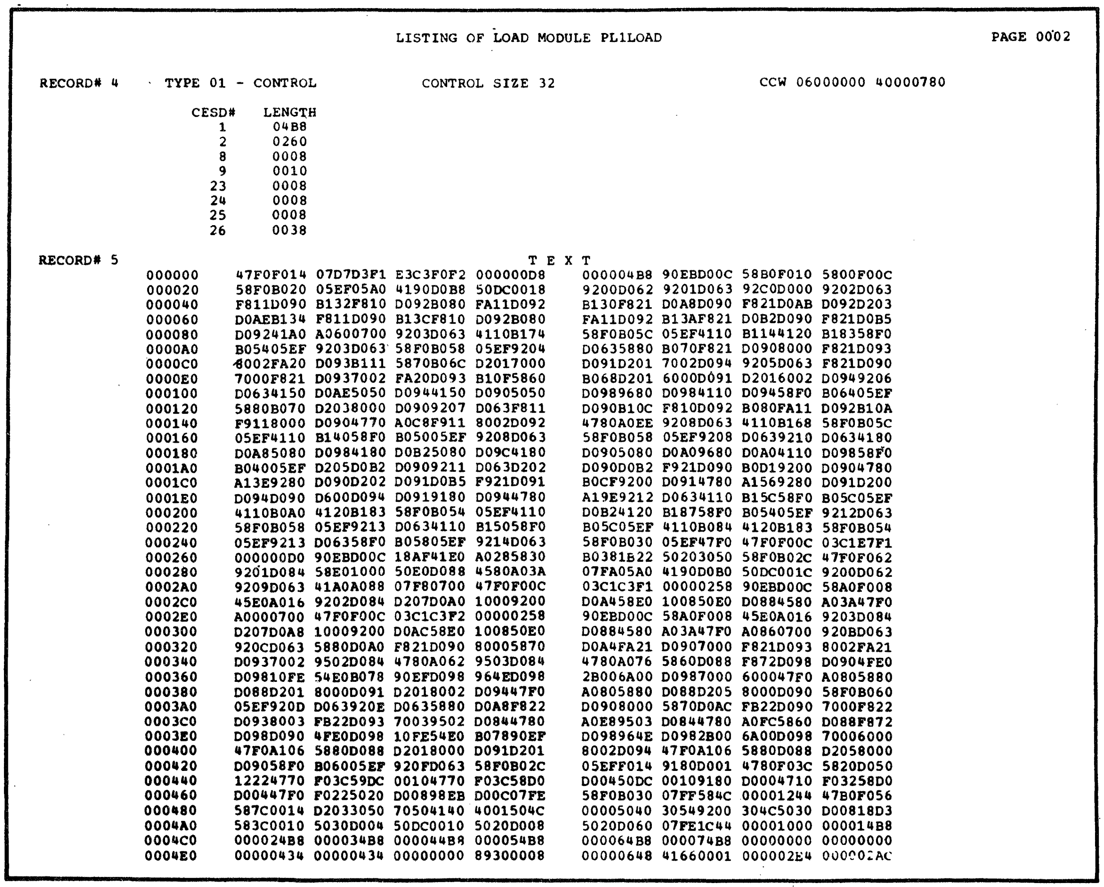
For LISTOBJ, the body of the listing consists of the object module listing, the module's external symbol dictionary, and its relocation dictionary. Figure 2-4 on page 2-12 shows an example of LISTOBJ output.

For LISTIDR, the third page of the listing begins a complete list of all CSECT identification records for the module. Figure 2-5 on page 2-13 shows an example of LISTIDR output.

For LISTLPA, the second page of the listing is a map of the link pack area, with modules ordered alphabetically by name. Figure 2-6 on page 2-14 shows an example of LISTLPA output.

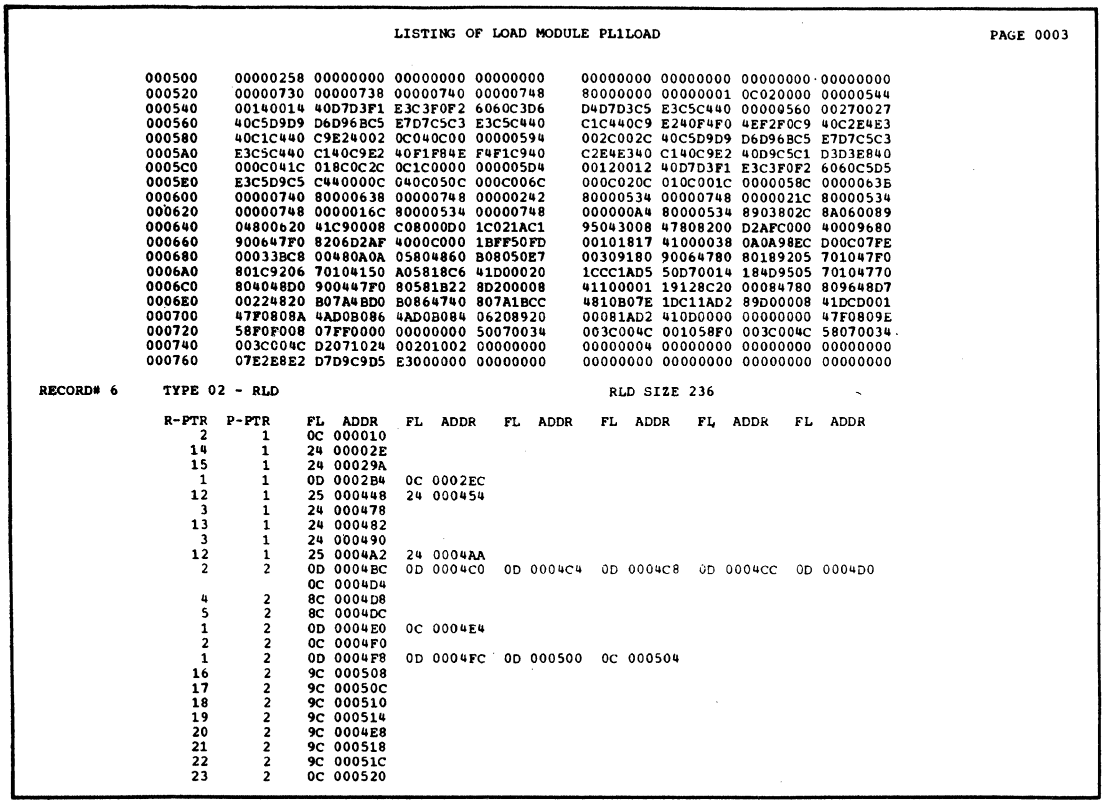


```
                                   LISTING OF LOAD MODULE PL1LOAD                                    PAGE 0001


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     IHEQLWO    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
```

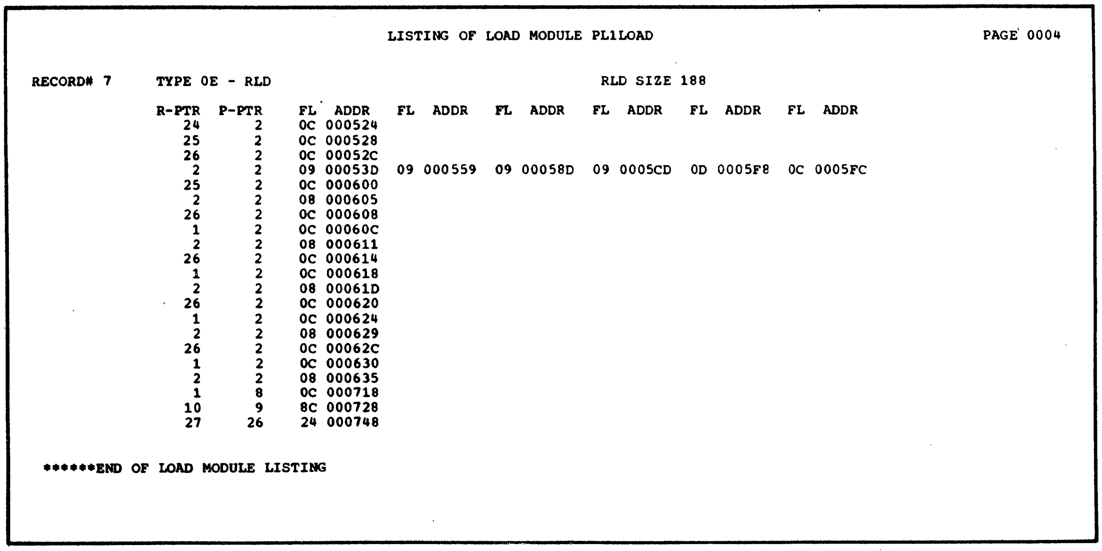


```
                                   LISTING OF LOAD MODULE PL1LOAD                                    PAGE 0002

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

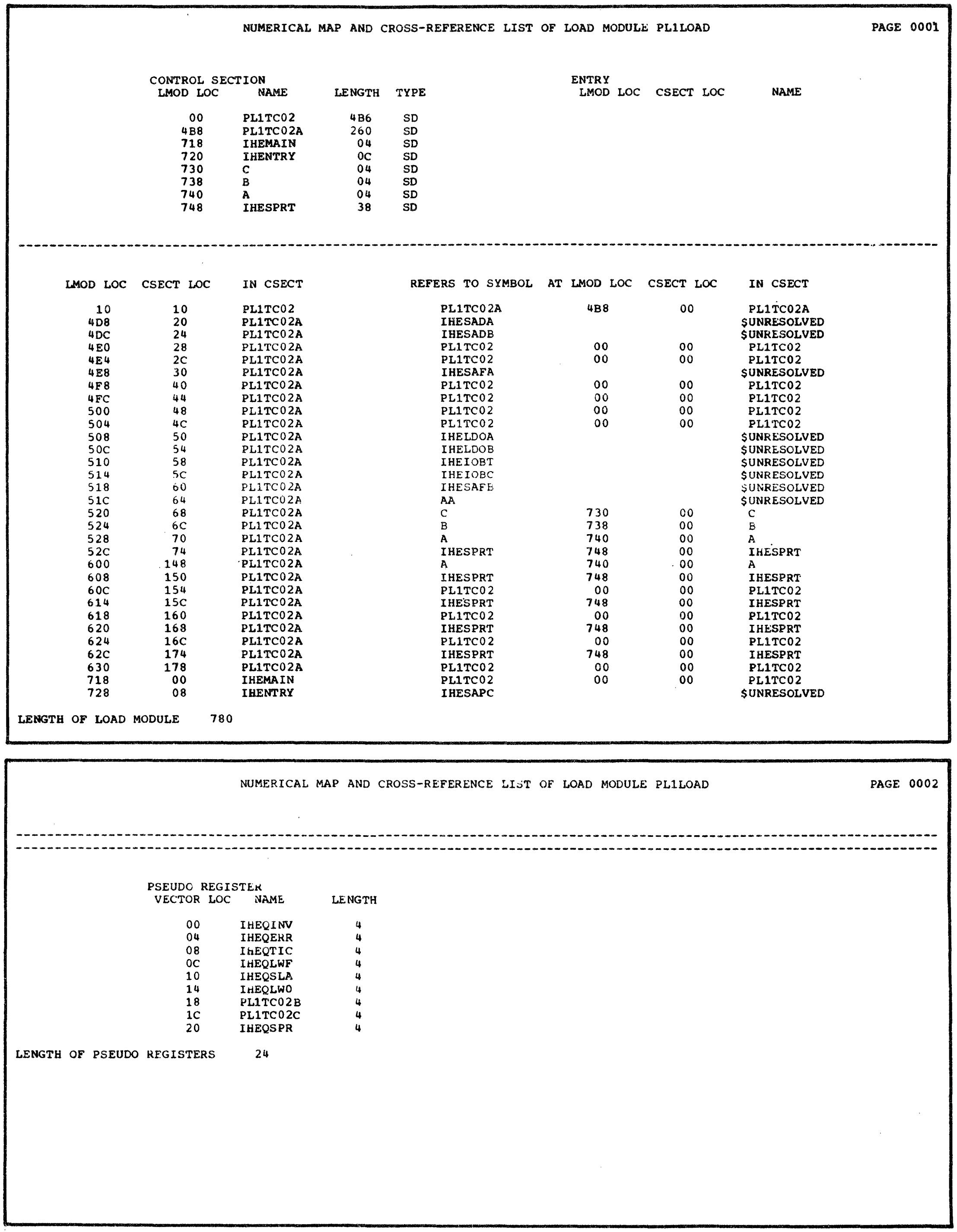


```
                                     LISTING OF LOAD MODULE PL1LOAD                                PAGE 0003

            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
```

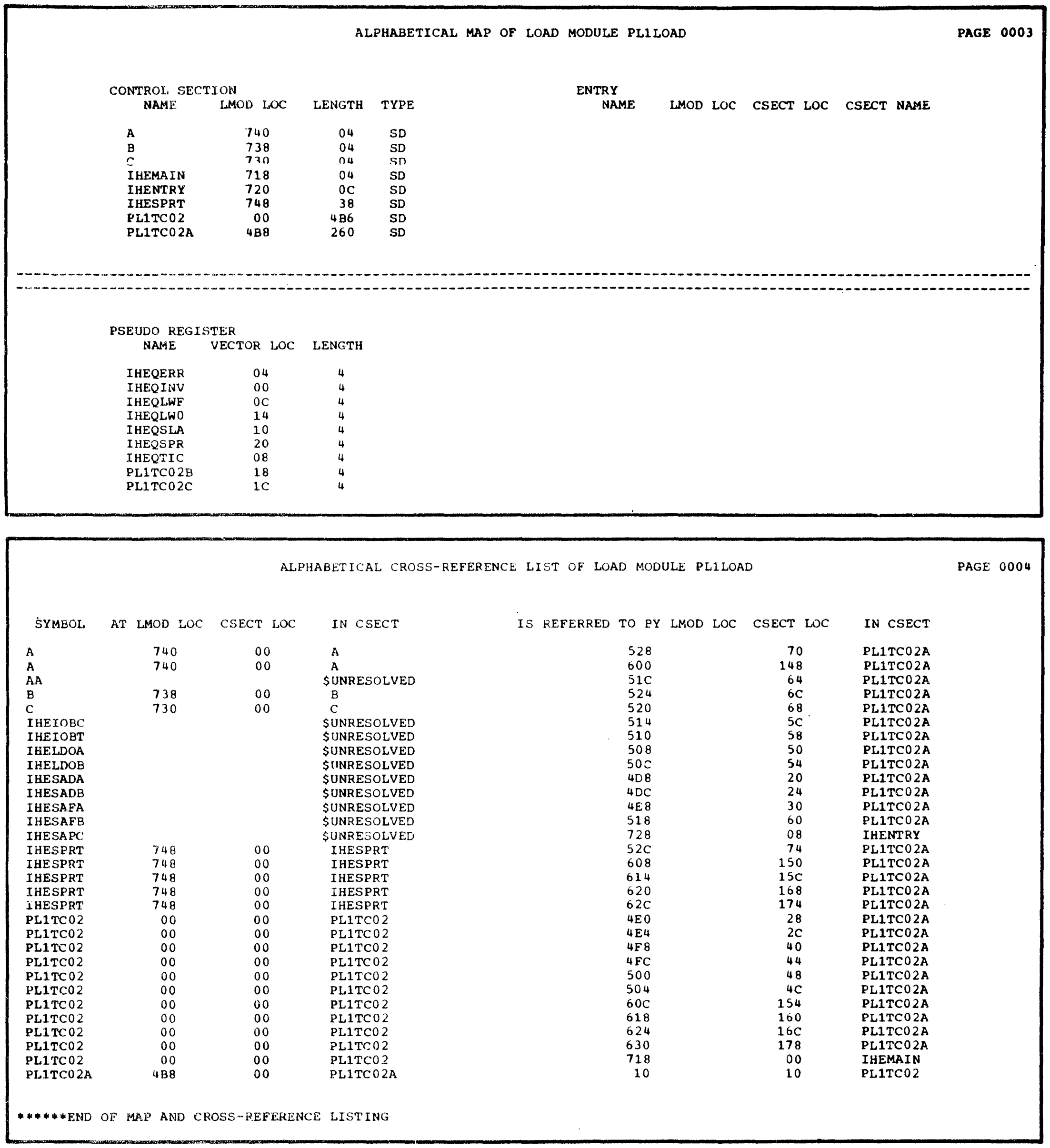
Figure 2-2 (Part 2 of 2). Sample LISTLOAD Output Load-Module Map



```
                        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
```

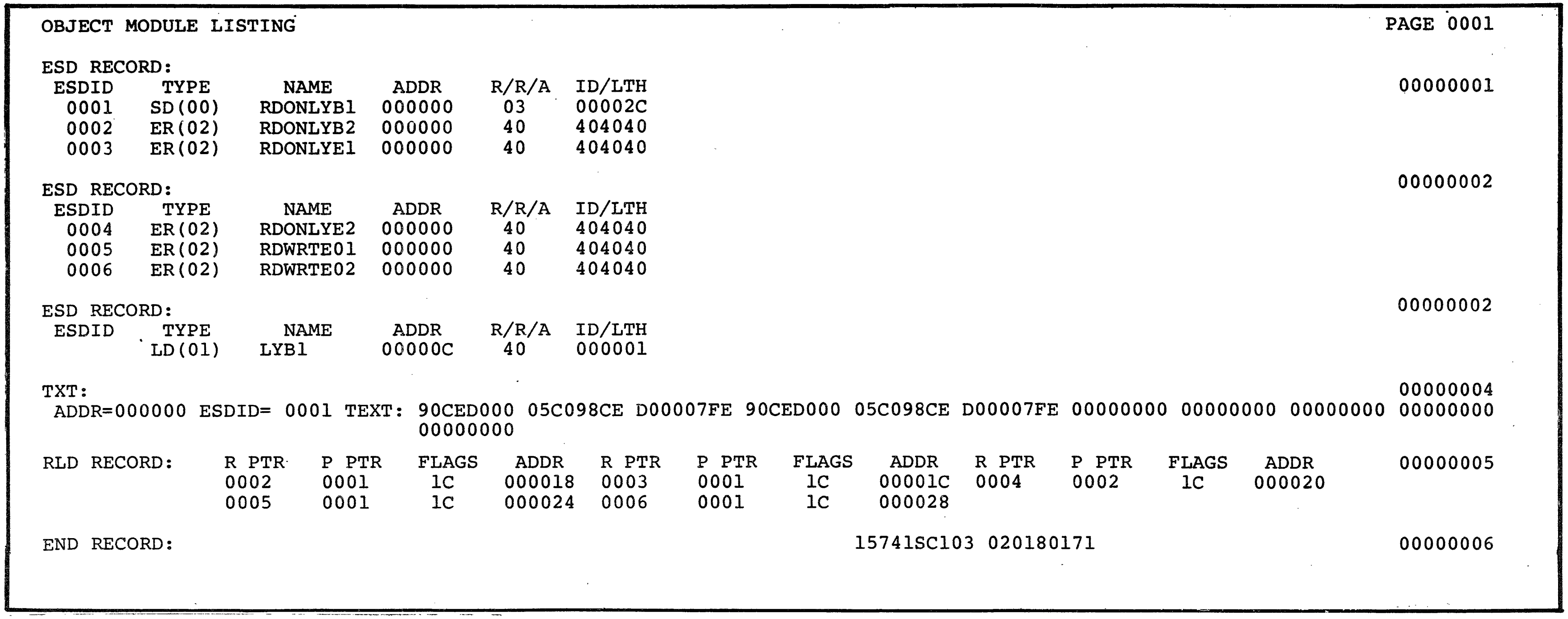
Figure 2-3 (Part 1 of 2). Sample LISTLOAD Output - Cross-Reference Listing



```
                              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
            IHEQLW0        14         4
            IHEQSLA        10         4
            IHEQSPR        20         4
            IHEQTIC        08         4
            PL1TC02B       18         4
            PL1TC02C       1C         4
```

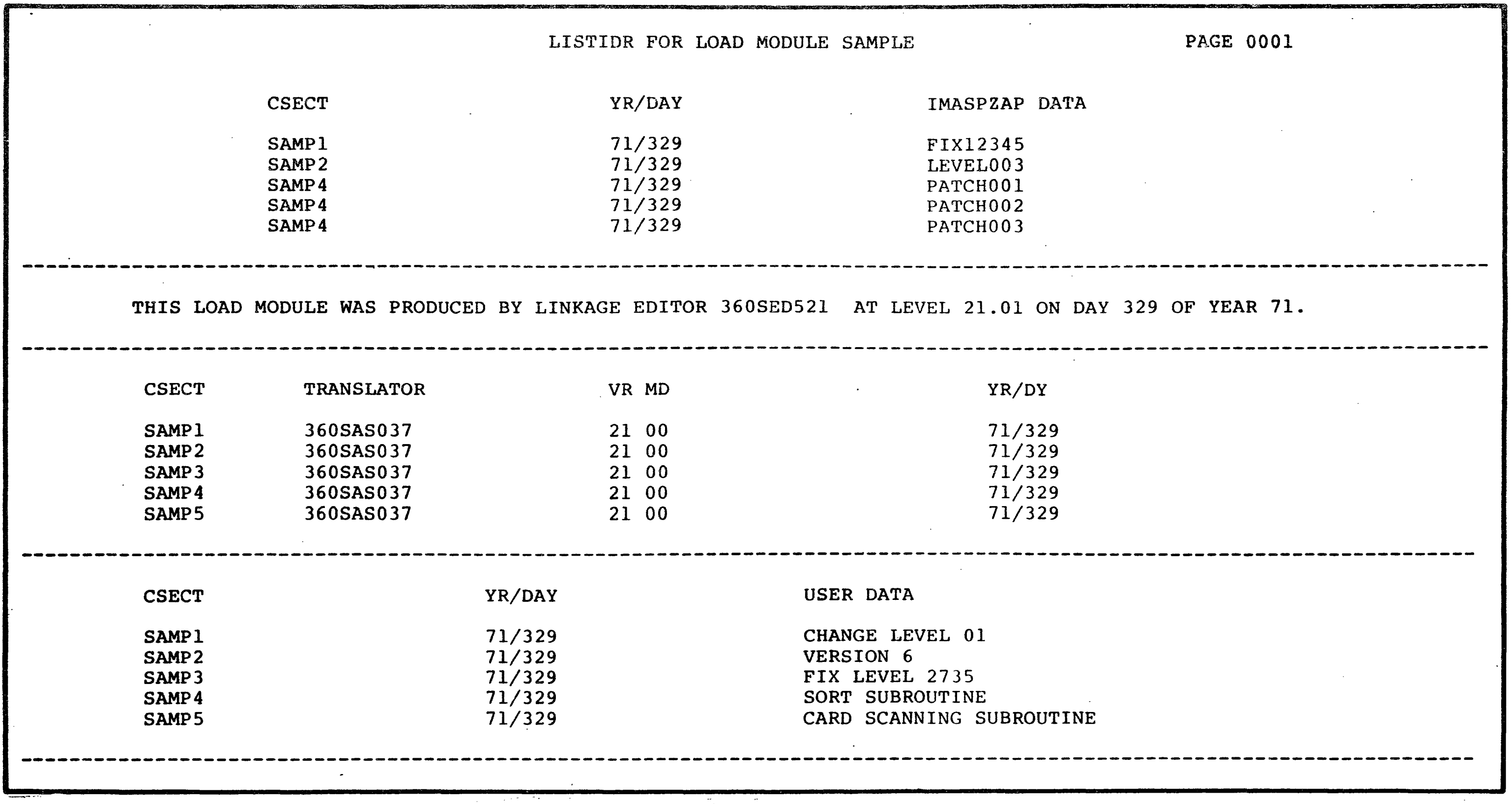
```
                         ALPHABETICAL CROSS-REFERENCE LIST OF LOAD MODULE PL1LOAD                          PAGE 0004


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



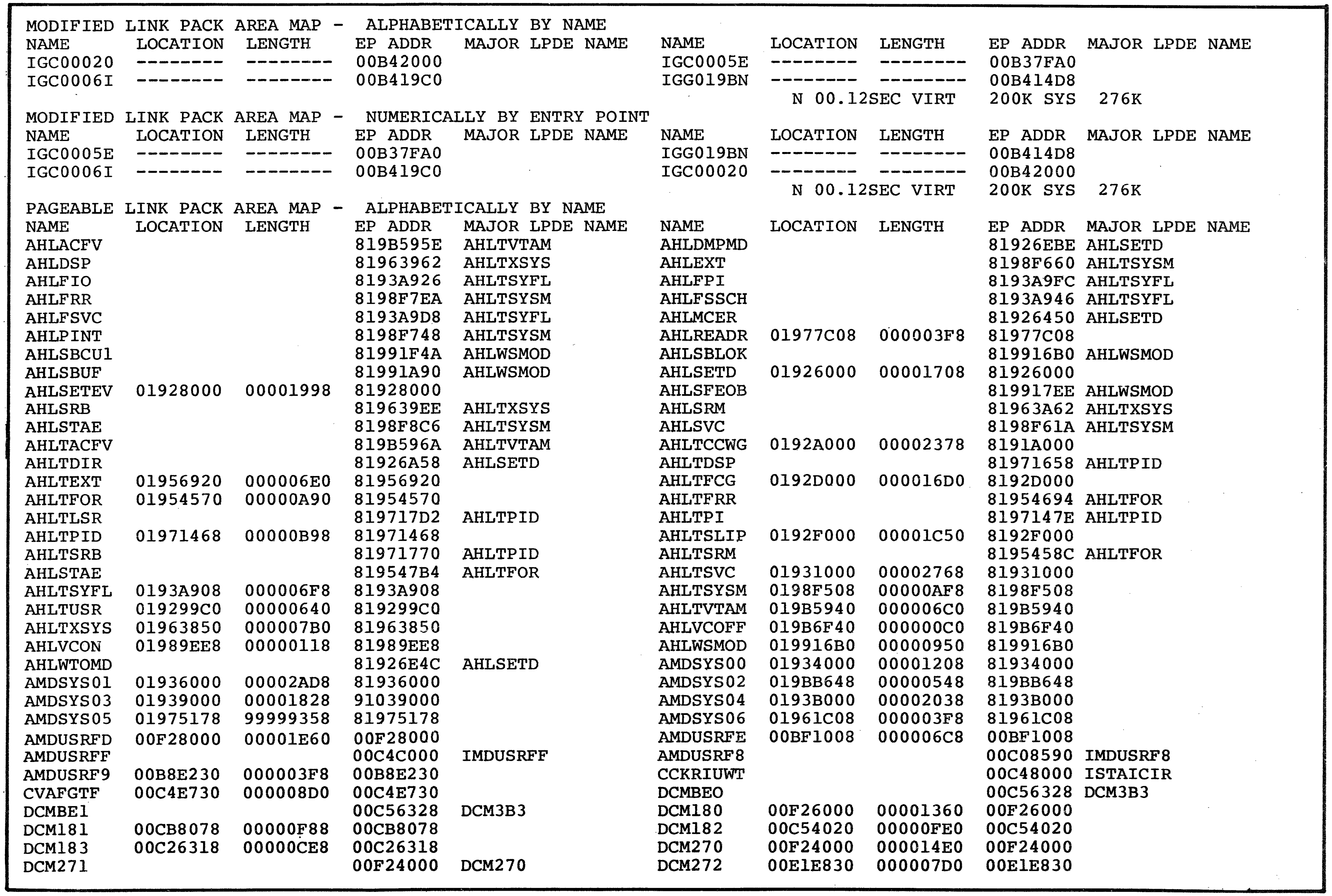
```
OBJECT MODULE LISTING                                                                                    PAGE 0001

ESD RECORD:
 ESDID     TYPE      NAME      ADDR     R/R/A  ID/LTH                                                    00000001
  0001    SD(00)    RDONLYB1  000000     03    00002C
  0002    ER(02)    RDONLYB2  000000     40    404040
  0003    ER(02)    RDONLYE1  000000     40    404040

ESD RECORD:                                                                                              00000002
 ESDID     TYPE      NAME      ADDR     R/R/A  ID/LTH
  0004    ER(02)    RDONLYE2  000000     40    404040
  0005    ER(02)    RDWRTE01  000000     40    404040
  0006    ER(02)    RDWRTE02  000000     40    404040

ESD RECORD:                                                                                              00000002
 ESDID     TYPE      NAME      ADDR     R/R/A  ID/LTH
          LD(01)    LYB1      00000C     40    000001

TXT:                                                                                                     00000004
 ADDR=000000 ESDID= 0001 TEXT: 90CED000 05C098CE D00007FE 90CED000 05C098CE D00007FE 00000000 00000000 00000000 00000000
                               00000000

RLD RECORD:     R PTR   P PTR   FLAGS   ADDR    R PTR   P PTR   FLAGS   ADDR    R PTR   P PTR   FLAGS   ADDR     00000005
                0002    0001     1C    000018   0003    0001     1C    00001C   0004    0002     1C    000020
                0005    0001     1C    000024   0006    0001     1C    000028

END RECORD:                                                      15741SC103 020180171                    00000006
```

Figure 2-4. 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 2-5. Sample LISTIDR Output



```
MODIFIED LINK PACK AREA MAP -  ALPHABETICALLY BY NAME
NAME      LOCATION  LENGTH    EP ADDR   MAJOR LPDE NAME   NAME      LOCATION  LENGTH    EP ADDR  MAJOR LPDE NAME
IGC00020  --------  --------  00B42000                    IGC0005E  --------  --------  00B37FA0
IGC0006I  --------  --------  00B419C0                    IGG019BN  --------  --------  00B414D8
                                                                      N 00.12SEC VIRT   200K SYS  276K
MODIFIED LINK PACK AREA MAP -  NUMERICALLY BY ENTRY POINT
NAME      LOCATION  LENGTH    EP ADDR   MAJOR LPDE NAME   NAME      LOCATION  LENGTH    EP ADDR  MAJOR LPDE NAME
IGC0005E  --------  --------  00B37FA0                    IGG019BN  --------  --------  00B414D8
IGC0006I  --------  --------  00B419C0                    IGC00020  --------  --------  00B42000
                                                                      N 00.12SEC VIRT   200K SYS  276K
PAGEABLE LINK PACK AREA MAP -  ALPHABETICALLY BY NAME
NAME      LOCATION  LENGTH    EP ADDR   MAJOR LPDE NAME   NAME      LOCATION  LENGTH    EP ADDR  MAJOR LPDE NAME
AHLACFV                       819B595E  AHLTVTAM          AHLDMPMD                      81926EBE AHLSETD
AHLDSP                        81963962  AHLTXSYS          AHLEXT                        8198F660 AHLTSYSM
AHLFIO                        8193A926  AHLTSYFL          AHLFPI                        8193A9FC AHLTSYFL
AHLFRR                        8198F7EA  AHLTSYSM          AHLFSSCH                      8193A946 AHLTSYFL
AHLFSVC                       8193A9D8  AHLTSYFL          AHLMCER                       81926450 AHLSETD
AHLPINT                       8198F748  AHLTSYSM          AHLREADR  01977C08  000003F8  81977C08
AHLSBCU1                      81991F4A  AHLWSMOD          AHLSBLOK                      819916B0 AHLWSMOD
AHLSBUF                       81991A90  AHLWSMOD          AHLSETD   01926000  00001708  81926000
AHLSETEV  01928000  00001998  81928000                    AHLSFEOB                      819917EE AHLWSMOD
AHLSRB                        819639EE  AHLTXSYS          AHLSRM                        81963A62 AHLTXSYS
AHLSTAE                       8198F8C6  AHLTSYSM          AHLSVC                        8198F61A AHLTSYSM
AHLTACFV                      819B596A  AHLTVTAM          AHLTCCWG  0192A000  00002378  8191A000
AHLTDIR                       81926A58  AHLSETD           AHLTDSP                       81971658 AHLTPID
AHLTEXT   01956920  000006E0  81956920                    AHLTFCG   0192D000  000016D0  8192D000
AHLTFOR   01954570  00000A90  81954570                    AHLTFRR                       81954694 AHLTFOR
AHLTLSR                       819717D2  AHLTPID           AHLTPI                        8197147E AHLTPID
AHLTPID   01971468  00000B98  81971468                    AHLTSLIP  0192F000  00001C50  8192F000
AHLTSRB                       81971770  AHLTPID           AHLTSRM                       8195458C AHLTFOR
AHLSTAE                       819547B4  AHLTFOR           AHLTSVC   01931000  00002768  81931000
AHLTSYFL  0193A908  000006F8  8193A908                    AHLTSYSM  0198F508  00000AF8  8198F508
AHLTUSR   019299C0  00000640  819299C0                    AHLTVTAM  019B5940  000006C0  819B5940
AHLTXSYS  01963850  000007B0  81963850                    AHLVCOFF  019B6F40  000000C0  819B6F40
AHLVCON   01989EE8  00000118  81989EE8                    AHLWSMOD  019916B0  00000950  819916B0
AHLWTOMD                      81926E4C  AHLSETD           AMDSYS00  01934000  00001208  81934000
AMDSYS01  01936000  00002AD8  81936000                    AMDSYS02  019BB648  00000548  819BB648
AMDSYS03  01939000  00001828  91039000                    AMDSYS04  0193B000  00002038  8193B000
AMDSYS05  01975178  99999358  81975178                    AMDSYS06  01961C08  000003F8  81961C08
AMDUSRFD  00F28000  00001E60  00F28000                    AMDUSRFE  00BF1008  000006C8  00BF1008
AMDUSRFF                      00C4C000  IMDUSRFF          AMDUSRF8                      00C08590 IMDUSRF8
AMDUSRF9  00B8E230  000003F8  00B8E230                    CCKRIUWT                      00C48000 ISTAICIR
CVAFGTF   00C4E730  000008D0  00C4E730                    DCMBEO                        00C56328 DCM3B3
DCMBE1                        00C56328  DCM3B3            DCM180    00F26000  00001360  00F26000
DCM181    00CB8078  00000F88  00CB8078                    DCM182    00C54020  00000FE0  00C54020
DCM183    00C26318  00000CE8  00C26318                    DCM270    00F24000  000014E0  00F24000
DCM271                        00F24000  DCM270            DCM272    00E1E830  000007D0  00E1E830
```

Figure 2-6. Sample LISTLPA Output

Examples

The following examples show sample uses of LIST.

Example 1: Listing Several Object Modules

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

```
//OBJLIST       JOB    MSGLEVEL=(1,1)
//LISTSTEP      EXEC   PGM=AMBLIST,REGION=64K
//SYSPRINT      DD     SYSOUT=A
//OBJLIB        DD     DSN=OBJMODS,DISP=SHR
//OBJSDS        DD     DSN=OBJMOD,DISP=SHR
//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)
/*
```

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 LIST control statements.

LISTOBJ Control Statement #1
instructs LIST to format the data set defined by the OBJSDS DD statement, treating them as a single member. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTOBJ Control Statement #2
instructs LIST to format three members of the partitioned data set (PDS or PDSE) 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.

Example 2: Listing Several Load Modules

In this example, LIST is used to produce formatted listings of several load modules.

```
//LOADLIST   JOB         MSGLEVEL=(1,1)
//LISTSTEP   EXEC        PGM=AMBLIST,REGION=64K
//SYSPRINT   DD          SYSOUT=A
//SYSLIB     DD          DSNAME=SYS1.LINKLIB,DISP=SHR
//LOADLIB    DD          DSNAME=LOADMOD,DISP=SHR
//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)
/*
```

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 LIST control statements.

LISTLOAD Control Statement #1
instructs LIST 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 LIST 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 LIST 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: Listing IDR Information for Several Load Modules

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

```
//IDRLIST     JOB     MSGLEVEL=(1,1)
//LISTSTEP    EXEC    PGM=AMBLIST,REGION=64K
//SYSPRINT    DD      SYSOUT=A
//SYSLIB      DD      DSN=SYS1.LINKLIB,DISP=SHR
//LOADLIB     DD      DSN=LOADMODS,DISP=SHR
//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)
    LISTIDR   DDN=LOADLIB,MODLIB
/*
```

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 LIST control statements.

LISTIDR Control Statement #1
instructs LIST 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 LIST to list CSECT identification records that contain SPZAP 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 Statement #3
instructs LIST 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.

LISTIDR Control Statement #4
instructs LIST to list CSECT identification records that contain SPZAP or user-supplied data for the LOADLIB data set. The module summary print out is suppressed.

Example 4: Verifying an Object Deck

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

```
//LSTOBJDK     JOB     MSGLEVEL=(1,1)
//             EXEC    PGM=AMBLIST,REGION=64K
//SYSPRINT     DD      SYSOUT=A
//OBJDECK      DD      *
    object deck
/*
//SYSIN        DD      *
    LISTOBJ         DDN=OBJDECK,
        TITLE=('OBJECT DECK LISTING FOR MYJOB',25)
/*
```

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 LIST control statements, which follows immediately.

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

Example 5: Verifying Several Load Modules

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

```
//LSTLDOBJ      JOB     MSGLEVEL=(1,1)
//              EXEC    PGM=AMBLIST,REGION=64K
//SYSPRINT      DD      SYSOUT=A
//OBJMOD        DD      DSN=MYMOD,DISP=SHR
//LOADMOD1      DD      DSN=YOURMOD,DISP=SHR
//LOADMOD2      DD      DSN=HISMOD,DISP=SHR
//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 HSMOD',25)
    LISTIDR         DDN=LOADMOD2,OUTPUT=ALL,
        TITLE=('IDRS FOR HISMOD',25)
/*
```

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 LIST control statements, which follows immediately.

LISTOBJ Control Statement
instructs LIST 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 LIST 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 LIST 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 LIST 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.

LISTIDR Control Statement #2

instructs LIST to list all CSECT identification 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.

Example 6: Listing a System Nucleus and Mapping the Link Pack Area

This example shows how to use the LISTLOAD and LISTLPA control statements to list a system nucleus and map the fixed link pack area, the modified link pack area, and the pageable link pack area. Note that in this example the data set containing the nucleus is named SYS1.NUCLEUS, and the nucleus occupies the member named IEANUC01. The map no longer represents the IPL version of the nucleus and message AMB129I will be issued. Use IPCS to format the NUCMAP. For information on using IPCS see the IPCS User's Guide.

```
//LISTNUC      JOB     MSGLEVEL=(1,1)
//STEP         EXEC    PGM=AMBLIST,REGION=100K
//SYSPRINT     DD      SYSOUT=A
//SYSLIB       DD      DSN=SYS1.NUCLEUS,DISP=SHR,UNIT=3330,
//    VOL=SER=nnnnn
//SYSIN        DD      *
    LISTLOAD           DDN=SYSLIB,MEMBER=IEANUC01,
       TITLE=('LISTING FOR NUCLEUS IEANUC01',25)
    LISTLPA
/*
```

SYSLIB DD Statement

defines the input data set, which in this case contains the nucleus.

SYSIN DD Statement

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

LISTLOAD Control Statement

instructs LIST to format the control and text records including the external symbol dictionary and relocation dictionary records of the load module IEANUC01 in the data set defined by the SYSLIB DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTLPA Control Statement

instructs LIST to map the fixed link pack area, the modified link pack area, and the pageable link pack area.

Chapter 3. SADMP program

Introduction

AMDSADMP (SADMP) is a stand-alone program designed to dump storage from a system that has failed. It is available to help you diagnose the failed system.

SADMP produces an unformatted dump of central (also called real) storage and parts of paged-out virtual storage, on a tape. This is also called a 'high-speed' stand-alone dump. The interactive problem control system (IPCS) can be used to format the dump and view it on the console or print it on a printer. For information on using IPCS, see the *IPCS User's Guide*.

SADMP can also produce a formatted dump of portions of central storage, on a tape or printer. This is called a 'low-speed' stand-alone dump.

You can use the IEBPTPCH utility program to print the formatted dump tape.

You can generate different versions of SADMP by coding several AMDSADMP macros and varying the values of keywords on the macros. AMDSADMP is supplied in the system library SYS1.MACLIB. The SADMP program is generated under the operating system, but its execution is a stand-alone operation.

Notes:

1. Stand-alone dump uses only on-line devices. When you dump to or from devices that have both real and virtual addresses, specify only the real addresses to SADMP. SADMP must reside on an on-line storage device.
2. You cannot direct SADMP output to its residence volume.
3. The SADMP residence volume, output device, and console do not have to be attached to the same processor.
4. To ensure that SADMP is available and successfully processes, the SYS1.PAGEDUMP data set should not be deleted or moved to another volume or pack.

The rest of this chapter describes the SADMP program and SADMP output, and shows how to generate and execute SADMP.

SADMP Program

There are two types of SADMP programs:

1. Unformatted (high-speed dump)
   - data set resides on a direct access device with output directed to a tape volume
   - data set resides on a tape device, with output directed to a tape volume
2. Formatted (low-speed dump)
   - data set resides on a direct access device with output directed to a tape volume
   - data set resides on a direct access device with output directed to a printer

SADMP unformatted output is intended for processing by IPCS.

The unformatted high-speed version of SADMP dumps all central storage and some areas of virtual storage not backed by central storage. The output includes:

- The prefixed save areas (PSAs)
- The nucleus and extended nucleus
- The system queue area (SQA) and the extended SQA
- The common service area (CSA) and the extended CSA
- Subpools 203-205, 213-215, 229, 230, 236, 237, 247, and 248 for the eligible address spaces based on the MINASID option specified
- The local system queue area (LSQA) and the extended LSQA for eligible address spaces based on the MINASID option specified
- A dump title
- The processor STORE STATUS information for each processor
- Main storage from address 0 to the top of main storage in absolute address sequence (some blocks may be missing because of off-line storage elements)
- Instruction trace data created by the instruction address trace
- Vector data for each processor that has a Vector Facility installed.
- Virtual storage areas selected by the DUMP= keyword, or selected by the operator at execution time.
- A message log, normally consisting of all console messages issued by the virtual dump program (whether or not messages were suppressed).
- Eligible address spaces that are physically swapped-in or all address spaces as specified by the MINASID keyword or by the operator at execution.

If SADMP detects an internal error, the output may also include one or more SADMP self-dumps.

To format and print the SADMP message log, invoke the SADMPMSG VERBEXIT under IPCS.

SADMP dumps the instruction trace data (created by the instruction address trace) only if SADMP loads the virtual storage dump program, AMDSAPGE. SADMP loads AMDSAPGE only when the central (also called real) storage dump program

AMDSARDM does not detect any unexpected program errors and when SADMP determines that the virtual environment is valid. SADMP does not dump the instruction trace data (created by instruction address trace) when any of the following conditions exist:

- The prefix register is invalid.
- Any of the following control blocks fail the SADMP validity check: the CVT, the RSM control blocks, or the SADMP RLT.
- The LPA has not been initialized.
- SADMP cannot read AMDSAPGE into storage due to an I/O error.

You can request that SADMP dump additional storage by specifying dump tailoring options. See "Dumping Additional Storage" on page 3-19.

The formatted low-speed version of SADMP produces a dump title specified at dump execution time, followed by processor related data for each available processor, followed by a dump of user selected areas of central storage.

If your program contains Vector Facility data and you are unable to dump that data, one of the following conditions exist:

1. The Vector Facility was not on-line at the time the dump was requested.
2. A machine check occurred on the processor on which your program was running while SADMP was issuing vector instructions.
3. While dumping vector data, an error occurred in SADMP. Some data might be missing on the SADMP output tape.

SADMP does not issue a message if it cannot dump data.

Creating the SADMP Program

Before you can run SADMP, you must create a SADMP program in ready-to-load form on an appropriate device. The procedure for creating a ready-to-load SADMP program is as follows:

1. **Device selection** - Select a tape or direct access device as the SADMP IPL volume ("residence volume"). If the residence volume is a direct access device, make sure that it does not already contain a SYS1.PAGEDUMP dataset; otherwise, SADMP initialization will fail. The SADMP volume mount attribute must be PRIVATE.
2. **Dump specification** - Specify the type of SADMP program that you want by coding the AMDSADMP macro.
3. **Residence volume initialization** - Put the SADMP program onto the residence volume in ready-to-load form, using either a two-stage generation or a one-step generation. In two-stage generation, first assemble the AMDSADMP macro. This will produce the input required to run the second stage of the JCL. Then run stage two to initialize the SADMP residence volume. In one-step generation, execute the AMDSAOSG program as a single job step, using the AMDSADMP macro as input data (SYSIN control statement).

With both two-stage or one-step generation, SADMP residence volume initialization consists of three phases:

- Phase One
  - The AMDSADM2 macro is assembled to produce the SADMP central storage dump program AMDSARDM, which dumps central storage, and the SADMP common communication table AMDSACCT, an internal control block.
- Phase Two
  - The SADMP build module AMDSABLD puts the output from phase one onto the residence volume in ready-to-load form. AMDSABLD locates the SADMP IPL program AMDSAIPL and the SADMP virtual storage dump program AMDSAPGE, and puts them onto the residence volume.
- Phase Three
  - If the residence volume is a direct access device, the device utility ICKDSF is invoked to put SADMP's IPL text onto the device's IPL track (cylinder 0, track 0).

Considerations in Creating Stand-Alone Dumps

The JCL for the AMDSADM2 assembly in stage two of the two-stage generation must provide a SYSLIB DDNAME that refers to a macro library containing the system macros BLSRDRPX, BLSRDATS, IEZBITS, IHAIRB, IHAMSF, IHAORB, IHAPSA, IHASCCB, and IHASCHIB. The same is true of any SYSLIB DDNAME in the JCL for the one-step generation program AMDSAOSG.

If you are using MVS/SP 3.1.3, some of these system macros are in SYS1.MODGEN, so make sure that the SYSLIB DDNAME concatenates SYS1.MODGEN to SYS1.MACLIB. Your installation should catalog the SYS1.MODGEN data set before creating the SADMP program; otherwise, the JCL that AMDSADMP produces will fail to build the stand-alone dump program.

You should consider some form of password or other security protection for SYS1.PAGEDUMP; this data set contains copies of several pages of central storage whose contents are unpredictable. You should also consider protecting the SADMP macros and modules from unauthorized modification.

To ensure that SADMP is available and successfully processes, the SYS1.PAGEDUMP data set should not be deleted or moved to another volume or pack.

You should consider which MINASID keyword option default is appropriate for your installation:

- MINASID(PHYSIN) will reduce the overall execution time of SADMP.
- MINASID(ALL) will cause SADMP output to contain a more complete image of your system at the time the dump is taken but will also increase the time required for execution.

Note: ALL address spaces are likely to be needed for hangs, enabled waits, and performance problems. PHYSIN should suffice for coded waits, loops, and spin loops.

Dump Specification: Coding the AMDSADMP Macro

This section describes the syntax of the AMDSADMP macro instruction used to produce both high-speed and low-speed versions of the dump program. For examples using the AMDSADMP macro, see the "SADMP Examples" on page 3-35.

GENERAL-USE PROGRAMMING INTERFACE

Syntax of the AMDSADMP Macro for an Unformatted Dump Program

Figure 3-1 shows the AMDSADMP macro parameters.

```
[symbol] AMDSADMP [TYPE={HI|UNFORMATTED}]

[,IPL={Tunit|Dunit|DSYSDA}]

 [,VOLSER={volser|SADUMP}][,ULABEL={PURGE|NOPURGE}]

  [,CONSOLE=({cnum|(cnum,ctype) [,(cnum,ctype)]...|01F,3278})]

   [,SYSUT={unit|SYSDA}][,OUTPUT={Tunit|T282}]

    [,DUMP='dto'][,PROMPT][,LOADPT={loadpt|X'1000'}]

     [,MSG={ACTION|ALL}][,MINASID={ALL|PHYSIN}]
```

Figure 3-1. Format of AMDSADMP Macro Instruction Used to Generate a High-Speed Dump Program

symbol
an arbitrary name you can assign to the AMDSADMP macro instruction. SADMP uses this symbol to create a job name for use in the initialization step.

AMDSADMP
the name of the macro instruction.

TYPE={HI|UNFORMATTED}
indicates the high-speed version of the dump program. When you omit this parameter, SADMP assumes TYPE=HI as the default. TYPE=HI and TYPE=UNFORMATTED have the same meaning.

IPL={Tunit|Dunit|DSYSDA}
indicates the unit address or the device type of the SADMP residence volume. The first character indicates the volume type; T for tape, D for DASD. SADMP uses the unit character string as the UNIT=value to allocate the residence volume for initialization. IPL=DSYSDA is the default. When you specify IPL=T, SADMP assumes T3400. When you specify IPL=D, SADMP assumes DSYSDA.

VOLSER={volser|SADUMP}
indicates the VOL=SER= value to allocate the residence volume for initialization. When you specify a tape volume, it must be NL (no labels). VOLSER=SADUMP is the default.

ULABEL={PURGE|NOPURGE}
indicates whether SADMP deletes (PURGE) or retains (NOPURGE) existing user labels on a DASD residence volume. When you specify NOPURGE, the SADMP program is written on cylinder 0 track 0 of the residence volume, immediately

following all user labels. If the user labels occupy so much space that the SADMP program does not fit on track 0, the initialization program issues an error message and terminates.

ULABEL=NOPURGE is the default.

CONSOLE=({cnum|(cnum,ctype)[,(cnum,ctype)]...|01F,3278})

indicates the device numbers and device types of the system consoles that SADMP is to use while taking the dump. When you specify CONSOLE=cnum, SADMP assumes (cnum,3278). You can specify from 2 to 21 consoles by coding:

```
CONSOLE=((cnum,ctype),(cnum,ctype),[,(cnum,ctype)]...)
```

The 3277, 3278, 3279, and 3290 device types are valid, and are interchangeable.

CONSOLE=(01F,3278) is the default.

SYSUT={unit|SYSDA}

specifies the UNIT=value of the device that SADMP uses for work files during the initialization stage. You may specify the device as a group name (for example, SYSDA), a device type (for example, 3330), or a unit address (for example, 131). SYSUT=SYSDA is the default.

OUTPUT={Tunit|T282}

specifies the unit address of the output device that SADMP uses as a default value if you use the EXTERNAL INTERRUPT key to bypass console communication, or if you give a null response to message AMD001A during SADMP IPL. You must always direct high-speed dump output to a tape device. This parameter does not allow the same flexibility that the IPL parameter allows (for example, T3400 is not a valid OUTPUT parameter). With a response to message AMD001A, you can override the address specified on OUTPUT= at execution time. OUTPUT=T282 is the default.

DUMP='dto'

indicates additional virtual storage that you want dumped. This storage is described as address ranges and subpools in address spaces. See the topic "Dumping Additional Storage" on page 3-19. When you do not specify DUMP, SADMP does not dump any additional storage unless you specify PROMPT.

PROMPT

causes SADMP, at execution time, to prompt you for additional virtual storage that you want dumped. You may respond with the same information that can be specified for the DUMP keyword. See the topic "Dumping Additional Storage" on page 3-19. When you do not specify PROMPT, SADMP does not prompt you to specify additional storage that you want dumped.

LOADPT={loadpt|X'1000'}

indicates the real address where SADMP will load the high-speed version of the dump program. The load point address must be a hexadecimal number larger than X'FFF' and smaller, by at least X'15000', than the highest real address in the configuration. The load point is rounded down to a page boundary. The SADMP central storage dump program requires four page frames of contiguous central storage. With the default LOADPT of X'1000', the dump program will use X'1000' - X'4FFF'.

An alternate SADMP version can be created so that the dump program can be IPLed even if there is bad or offline storage at locations X'1000' - X'15000'. The LOADPT can be set equal to one megabyte (X'100000') less than the address at the top of on-line central storage.

MSG = {ACTION|ALL}

indicates the type of SADMP messages that are to appear on the console. When you specify ACTION, SADMP writes only messages that require operator action. When you specify ALL, SADMP suppresses no messages. ALL is the default.

This keyword has no effect on the SADMP message log; even if you specify MSG = ACTION, the SADMP virtual dump program writes messages to the message log on the output tape.

MINASID = {ALL|PHYSIN}

indicates the status of the address spaces that are to be included in the minimal dump. Specify PHYSIN to dump the minimum virtual storage (LSQA and selected system subpools) for the physically swapped-in address spaces only. Specify ALL to dump the minimum virtual storage (LSQA and selected system subpools) for all of the address spaces. ALL is the default.

At execution, if PHYSIN was specified, SADMP writes message AMD082I to the operator's console to warn the operator that some virtual storage may be excluded from the dump.

Syntax of the AMDSADMP Macro for a Formatted Dump Program

Figure 3-2 shows the syntax of the AMDSADMP macro instruction for producing a low-speed dump program.

```
[symbol] AMDSADMP TYPE={LO|FORMATTED}}[,IPL={Dunit|DSYSDA}]

 [,VOLSER={volser|SADUMP}][,ULABEL={PURGE|NOPURGE}]

  [,CONSOLE=({cnum|(cnum,ct) [,(cnum,ct)] ...|01F,3278})

   [,SYSUT={unit|SYSDA}][,OUTPUT={Tunit|Punit|P00E}]

    [,ADDR={VIRTUAL|REAL}][,LOADPT={loadpt|'1000'}]

     [,MSG={ACTION|ALL}]
```

Figure 3-2. Format of AMDSADMP Macro Instruction Used to Generate a Low-Speed Dump Program

symbol

an arbitrary name you can assign to the AMDSADMP macro instruction. SADMP uses this symbol to create a job name for use in the initialization step.

AMDSADMP

the name of the macro instruction.

TYPE = {LO|FORMATTED}

specifies the low-speed version of the dump program. When you do not specify TYPE =, SADMP uses a high-speed dump as the default. LO and FORMATTED have the same meaning.

IPL = {Dunit|DSYSDA}

specifies the unit address or the device type of the SADMP residence volume. The first character must be D (for DASD). SADMP uses the unit character string as the UNIT = value to allocate the residence volume for initialization. IPL = DSYSDA is the default.

OUTPUT = {Tunit|Punit|P00E}

specifies the device to which SADMP writes output. The first character specifies the output device type: P for printer, T for tape. The unit character string specifies the unit address that SADMP uses if you use the EXTERNAL INTERRUPT key to bypass console communication during SADMP IPL. This parameter does not allow the same flexibility that the IPL parameter allows (for example, T3400 is not a valid OUTPUT parameter). At execution time, you can override the address that you specified on OUTPUT= with a response to message AMD001A. Note that a null response causes SADMP to use the OUTPUT= value. OUTPUT=P00E (that is, a printer) is the default.

VOLSER = {volser|SADUMP}

specifies the VOL=SER= value to allocate the residence volume for initialization. VOLSER=SADUMP is the default.

CONSOLE = ({cnum|(cnum,ctype)[,(cnum,ctype)]...|01F,3278})

indicates the device numbers and device types of the system consoles that SADMP is to use while taking the dump. When you specify CONSOLE=cnum, SADMP assumes (cnum,3278). You can specify from 2 to 21 consoles by coding:

```
CONSOLE=((cnum,ctype),(cnum,ctype),[,(cnum,ctype)]...)
```

The 3277, 3278, 3279, and 3290 device types are valid, and are interchangeable.

CONSOLE=(01F,3278) is the default.

SYSUT = {unit|SYSDA}

specifies the UNIT= value of the device that SADMP uses for work files during the initialization stage. You may specify the device as a group name (for example, SYSDA), a device type (for example, 3330), or a unit address (for example, 131). SYSUT=SYSDA is the default.

ULABEL = {PURGE|NOPURGE}

specifies whether SADMP deletes (PURGE) or retains (NOPURGE) existing user labels on a DASD residence volume. When you specify NOPURGE, the SADMP program is written on cylinder 0 track 0 of the residence volume, immediately following all user labels. If the user labels occupy so much space that the SADMP program does not fit on track 0, the initialization program issues an error message and terminates. ULABEL=NOPURGE is the default.

ADDR = {REAL|VIRTUAL}

specifies the default action that SADMP takes if the console is unavailable or if you specify end of block to a prompting message for the type of dump desired. REAL specifies that central storage (from 0 to 2048 megabytes) is dumped in ascending order by real addresses. VIRTUAL specifies that central storage (from 0 to 2048 megabytes) is dumped in ascending order by virtual addresses using the segment table of the address space in control when the IPLed processor was stopped or if no STORE STATUS was done, the master address space. The default is REAL.

Note that the ADDR keyword is valid for low-speed SADMP only.

LOADPT = {loadpt|X'1000'}
specifies the real address where SADMP loads the low-speed version of the dump program. The address 'loadpt' must be a hexadecimal number larger than X'FFF' and smaller, by at least X'4000', than the highest real address in the configuration. The load point is rounded down to a page boundary. The SADMP central storage dump program requires four page frames of contiguous central storage. With the default LOADPT of X'1000', the dump program will use X'1000' - X'4FFF'.

An alternate SADMP version can be created so that the dump program can be IPLed even if there is bad or offline storage at locations X'1000' - X'4FFF'. The LOADPT can be set equal to one megabyte (X'100000') less than the address at the top of on-line central storage.

MSG = {ACTION|ALL}
specifies the type of SADMP messages that appear on the console. When you specify ACTION, SADMP writes only messages that require operator action. When you specify ALL, SADMP writes all messages. ALL is the default.

End of GENERAL-USE PROGRAMMING INTERFACE

Two-Stage Generation:

Assembling the Macro Instruction

Figure 3-3 and Figure 3-4 are examples of the JCL statements needed to assemble the AMDSADMP macro. Figure 3-3 is a sample for installations using a release prior to MVS/SP 3.1.3; Figure 3-4 on page 3-11 is for installations using MVS/SP 3.1.3. In both examples, the stage two JCL is placed in the SYSPUNCH data set.

```
//ASSEMSAD    JOB         MSGLEVEL=(1,1)
//ASM         EXEC        PGM=IEV90,PARM='DECK'
//SYSLIB      DD          DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1      DD          UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT    DD          SYSOUT=A
//SYSPUNCH    DD          SYSOUT=B
//SYSIN       DD          *
              AMDSADMP    TYPE=HI
              END
/*
```

Figure 3-3. Sample JCL to Assemble AMDSADMP Macro (For releases of MVS/SP Version 3 prior to MVS/SP 3.1.3)

```
//ASSEMSAD     JOB         MSGLEVEL=(1,1)
//ASM          EXEC        PGM=IEV90,PARM='DECK'
//SYSLIB       DD          DSN=SYS1.MACLIB,DISP=SHR
//             DD          DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1       DD          UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT     DD          SYSOUT=A
//SYSPUNCH     DD          SYSOUT=B
//SYSIN        DD          *
               AMDSADMP    TYPE=HI
               END
/*
```

Figure 3-4. Sample JCL to Assemble AMDSADMP Macro (MVS/SP 3.1.3)

The SYSLIB data set must contain the AMDSADMP macro.

To direct the punched output to tape, use the following SYSPUNCH DD statement:

```
//SYSPUNCH DD UNIT=tape,LABEL=(,NL),DISP=(NEW,KEEP),
// VOL=SER=SCRTCH
```

To direct the punched output to a new direct access data set, use the following SYSPUNCH DD statement:

```
//SYSPUNCH DD UNIT=dasd,SPACE=(80,(30,10)),DSN=dsname,
// DISP=(NEW,KEEP),VOL=SER=volser
```

Assembling Multiple Versions of AMDSADMP

You can assemble multiple versions of AMDSADMP at the same time, provided that each version specifies a different residence volume. Differentiate between versions by coding a unique symbol at the beginning of each macro instruction. AMDSADMP uses the symbol you indicate to create unique stage-two job names. The output from a multiple assembly is a single listing and a single object deck, which may be broken into separate jobs if desired.Figure 3-5 and Figure 3-6 show sample JCL for coding multiple versions of AMDSADMP. Figure 3-5 is a sample for installations using releases prior to MVS/SP 3.1.3; Figure 3-6 is for installations using MVS/SP 3.1.3.

```
//MULTISAD JOB  MSGLEVEL=(1,1)
//ASM      EXEC PGM=IEV90,PARM='DECK'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSIN    DD   *
HITAPE     AMDSADMP IPL=T3400,VOLSER=SADMP1
HIDASD1    AMDSADMP VOLSER=SADMP2,MINASID=PHYSIN
HIDASD2    AMDSADMP VOLSER=SADMP3,LOADPT=X'10000000'
LOTAPE     AMDSADMP TYPE=LO,OUTPUT=T282,VOLSER=SADMP4
LOPRINTER  AMDSADMP TYPE=LO,VOLSER=SADMP5
           END
/*
```

Figure 3-5. Sample JCL for Assembling Multiple Versions of AMDSADMP Macro (For releases of MVS/SP Version 3 prior to MVS/SP 3.1.3)

```
//MULTISAD JOB  MSGLEVEL=(1,1)
//ASM      EXEC PGM=IEV90,PARM='DECK'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSIN    DD   *
HITAPE     AMDSADMP IPL=T3400,VOLSER=SADMP1
HIDASD1    AMDSADMP VOLSER=SADMP2,MINASID=PHYSIN
HIDASD2    AMDSADMP VOLSER=SADMP3,LOADPT=X'10000000'
LOTAPE     AMDSADMP TYPE=LO,OUTPUT=T282,VOLSER=SADMP4
LOPRINTER  AMDSADMP TYPE=LO,VOLSER=SADMP5
           END
/*
```

Figure 3-6. Sample JCL for Assembling Multiple Versions of AMDSADMP Macro (MVS/SP 3.1.3)

Message Output from AMDSADMP

The output listing might contain error messages, which describe errors that you may have made in specifying the AMDSADMP macro instruction. To respond to one of these messages, check your specification of the macro instruction and run the assembly step again.

Note: Words shown here that begin with & are replaced in the error messages with the keyword values in error.

```
CONSOLE PARM NOT DETECTED. DEFAULT (01F, 3278) WILL BE USED.
```

Explanation: Either the console parameter was not specified or it was not specified correctly on the continuation statement. The parameter was probably not continued correctly on the next defined statement. Continue the interrupted parameter or field beginning in any column from 4 through 16.

(See *JCL Reference.* Read the topic covering 'Continued Statements'.)

Severity Code: 4.

```
IPLUNIT WAS NOT SPECIFIED OR IPL= TYPE (D OR T) WAS SPECIFIED
INCORRECTLY. UNIT WILL BE DEFAULTED TO SYSDA.
```

Explanation: The IPL parameter should be specified as IPL=duuu, where 'd' is for D for direct access or T for tape, and 'uuu' is a valid unit type or address for the SADMP IPL volume as described by the UNIT=uuu JCL parameter.

Severity Code: 0.

```
IPL=&IPL IS INVALID.  FIRST CHARACTER MUST BE D OR T,
AND HAS BEEN REPLACED WITH A D.
```

Explanation: The IPL operand is invalid. It is not prefixed with a 'D' or a 'T'.

Severity Code: 4.

```
IPL=&IPL IS TOO LONG. THE UNIT NAME WILL BE TRUNCATED.
```

Explanation: The unit name can be at most 8 characters long.

Severity Code: 4.

```
CONSOLE ADDRESS &CONAD IS INVALID. IT MUST BE A DEVICE
NUMBER.  01F IS SUBSTITUTED.
```

Explanation: The console address operand is not three hexadecimal digits.

Severity Code: 4.

```
CONSOLE TYPE &CONTP IS INVALID. IT MUST BE A 4 DIGIT
NUMBER. 3278 HAS BEEN USED.
```

Explanation: An invalid console type was specified. Only 3277, 3278, and 3279 are acceptable. The length of the console type was not equal to 4.

Severity Code: 4.

```
TYPE=&TYPE IS INVALID. IT MUST BE EITHER UNFORMATTED OR
FORMATTED. TYPE=UNFORMATTED HAS BEEN USED.
```

Explanation: Type operand must be HI, LO, UNFORMATTED, or FORMATTED.

Severity Code: 4.

```
TYPE=&TYPE CAN ONLY BE RESIDENT ON A DASD. A DASD
RESIDENCE VOLUME HAS BEEN USED.
```

Explanation: TYPE = LO and TYPE = FORMATTED can only be resident on a direct access device.

Severity Code: 8.

```
OUTPUT=&OUTPUT IS INVALID. IT MUST BE A T OR P FOLLOWED BY
A DEVICE NUMBER. OUTPUT=P00E HAS BEEN USED.
```

Explanation: For TYPE = LO the output address was not prefixed with a 'T' or 'P' or the address was not a 3-character address.

Severity Code: 4.

```
OUTPUT=&OUTPUT IS INVALID. IT MUST BE A T FOLLOWED BY
A DEVICE NUMBER. OUTPUT=T282 HAS BEEN USED.
```

Explanation: For TYPE=HI the output address was not prefixed by a 'T' or the address was not a 3-character address.

Severity Code: 4.

```
ULABEL=NOPURGE IS NOT POSSIBLE FOR A TAPE RESIDENCE VOLUME.
```

Explanation: The ULABEL cannot be NOPURGE when the IPL device is tape. SADMP ignores your ULABEL specification.

Severity Code: 8.

```
ADDR=&ADDR IS INVALID. ADDR=REAL HAS BEEN USED.
```

Explanation: The ADDR operand is not REAL or VIRTUAL.

Severity Code: 4.

```
MSG=&MSG IS INVALID. IT MUST BE ALL OR ACTION.
MSG=ALL HAS BEEN USED.
```

Explanation: The MSG operand is not ALL or ACTION.

Severity Code: 4.

```
LOADPT=&VALUE IS INVALID. X'1000' HAS BEEN USED.
```

Explanation: The LOADPT operand must be a hexadecimal number from X'1000' to X'7FFFFFFF'.

Severity Code: 4.

Initializing the Residence Volume

You must make sure that the SADMP residence device does not contain a SYS1.PAGEDUMP data set if you are generating a direct access resident dump program. When SADMP finds a SYS1.PAGEDUMP data set on the direct access device to be initialized as the residence volume, initialization terminates.

Execution of the stage-two JCL initializes the SADMP residence volume. The execution of this stage creates a SYS1.PAGEDUMP data set (on the residence volume) that contains the SADMP programs.

Physical output from the assembly part of the initialization step is a listing for the SADMP central storage dump program, AMDSARDM. The remainder of the output consists of messages from the SADMP build module AMDSABLD and, when the residence volume is direct access, the device utility ICKDSF.

One-Step Generation

The SADMP utility program, AMDSAOSG, initializes a SADMP residence volume in one job step by dynamically allocating data sets and invoking the appropriate programs. To run the one-step generation program, indicate one AMDSADMP macro as a control statement for ddname GENPARMS, just as you would do on the SYSIN statement in the first stage of a two-stage generation. Figure 3-7 is a sample job to generate SADMP in one job step:

```
//SADMPGEN JOB  MSGLEVEL=(1,1)
//OSG      EXEC PGM=AMDSAOSG
//GENPARMS DD   *
         AMDSADMP TYPE=HI,IPL=DSYSDA,VOLSER=SPOOL2,                    X
               CONSOLE=(1A0,3277)
         END
/*
```

Figure 3-7. Sample JCL for One-Step Generation of SADMP

The output from AMDSAOSG is the same as the output from the residence volume initialization stage of two-stage generation, followed by a message from AMDSAOSG. AMDSAOSG returns the following codes:

Code	Message
0	RESIDENCE VOLUME INITIALIZED
4	RESIDENCE VOLUME NOT INITIALIZED DUE TO ERROR, OR A WARNING WAS ISSUED DURING AMDSADMP ASSEMBLY
8	RESIDENCE VOLUME NOT INITIALIZED; GENPRINT COULD NOT BE OPENED

AMDSAOSG allocates several ddnames for its own use and for use by the programs it calls. You can override these allocations by specifying DD statements in the AMDSAOSG job step. For example, when you want to:

1. use a local modification of the AMDSABLD program in the cataloged load library SADMP.LOAD; **and**
2. use a local modification of the AMDSADM2 macro in the cataloged macro library SADMP.MACLIB; **and**
3. preserve the output listing in the cataloged data set SADMP.LIST

you would code the JCL shown in Figure 3-8.

```
//SADMPGEN JOB  MSGLEVEL=(1,1)
//OSG      EXEC PGM=AMDSAOSG
//STEPLIB  DD   DSN=SADMP.LOAD,DISP=SHR
//SYSLIB   DD   DSN=SADMP.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MACLIB,DISP=SHR
//GENPRINT DD   DSN=SADMP.LIST,DISP=OLD
//GENPARMS DD   *
         AMDSADMP TYPE=HI,IPL=DSYSDA,VOLSER=SPOOL2,                    X
               CONSOLE=(1A0,3277)
         END
/*
```

Figure 3-8. One-Step Generation With Overriding ddnames

Figure 3-9 (which has related notes that follow it) shows the ddnames AMDSAOSG uses, and the defaults for the ddnames.

Figure 3-9. ddnames and Defaults Used by AMDSAOSG

ddname	Default Value	Use
GENPARMS	Must be preallocated.	Input for AMDSAOSG, passed to assembler.
GENPRINT	SYSOUT=A	Output listing from AMDSAOSG.
IPLDEV	DSN=SYS1.PAGEDUMP,UNIT=iplunit, VOL=(PRIVATE,SER=iplser),	SADMP program, output from AMDSABLD. ICKDSF uses VOL keywords to describe the residence volume.
	DISP=OLD,DCB=(BLKSIZE=12288,RECFM=U, DSORG=PS), LABEL=(,NL)	Tape IPL volume.
	DISP=(NEW,KEEP),DCB=(LRECL=4096,BLKSIZE=4096, RECFM=F,DSORG=PS),SPACE=(4096,(58),CONTIG), LABEL=EXPDT=99366	DASD IPL volume.
IPLTEXT	DSN=SYS1.LINKLIB(AMDSAIPL),DISP=SHR	Input for AMDSABLD.
PGETEXT	DSN=SYS1.LINKLIB(AMDSAPGE),DISP=SHR	Input for AMDSABLD.
STEPLIB	None	AMDSAOSG, H assembler IEV90, AMDSABLD and ICKDSF programs. This must be an APF-authorized library.
SYSIN	Must not be preallocated.	Input for assembler and ICKSDF.

SYSLIB	DSN=SYS1.MACLIB,DISP=SHR
	DSN=SYS1.MODGEN,DISP=SHR
		AMDSADMP and AMDSADM2 macros.

SYSPRINT	Must not be preallocated	Temporary listings from called programs.
SYSPUNCH	DSN=&OBJ,UNIT=SYSDA,SPACE=(80,(250,50))	Object module passed from assembler to AMDSABLD.
SYSTERM	None	Assembly messages.
SYSUT1	UNIT=SYSDA,SPACE=(1700,(50,50))	Work file for assembler.
TRKOTEXT	DSN=&TRKOTEXT,UNIT=iplunit, VOL=SER=iplser,SPACE=(4096,(2,1))	Cylinder 0, Track 0 IPL text from AMDSABLD to ICKDSF.

Notes:

1. To ensure that SADMP is available and successfully processes, the SYS1.PAGEDUMP data set should not be deleted or moved to another volume or pack.
2. You **must** specify the GENPARMS ddname on the job step.
3. You may **not** specify the SYSPRINT and SYSIN DD statements in the job step.
4. In GENPARMS, you specify values for UNIT= and VOLSER= on the AMDSADMP macro statement.

Using One-Step Generation

You must make sure that the SADMP residence device does not contain a SYS1.PAGEDUMP data set if you are generating a direct access resident dump program. When SADMP finds a data set on the device to be initialized as the residence device, initialization terminates.

Execution of AMDSAOSG initializes the SADMP residence volume. The execution of the SADMP utility program creates a SYS1.PAGEDUMP data set (on the residence volume); this data set contains the SADMP program.

Physical output from the assembly part of the initialization step is a listing for the SADMP central storage dump program AMDSARDM. The remainder of the output consists of messages from SADMP build modules AMDSAOSG and AMDSABLD, and when the residence volume is direct access, the device utility ICKDSF.

Dumping Additional Storage

You can request that SADMP dump additional storage by specifying dump tailoring options, either when the SADMP residence volume is initialized, or at execution time. For instance, you may want to request more storage to retrieve information from a user address space.

Requesting Additional Storage During SADMP Generation

Indicate the dump tailoring options described in “Dump Tailoring Options” on page 3-20 within parentheses and single quotes as the value of the DUMP keyword on the AMDSADMP macro.

Examples:

```
DUMP=('SP(5,37,18) IN ASID('JES3')')
DUMP=('RANGE(0:1000000) IN ASID(1)')
DUMP=('DATASPACES OF ASID('RASP')')
```

Note: Do not double the quotes within the DUMP options. The DUMP options may not exceed 255 characters in length.

Requesting Additional Storage During SADMP Execution

By coding the PROMPT keyword on the AMDSADMP macro, you can have SADMP prompt the operator to dump additional storage during execution. When you code PROMPT=YES, and the virtual storage dump program gets control, it issues the following message:

```
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
```

The operator responds with one of the following:

1. DUMP followed by dump options. In this case, the ‘=’ after DUMP is optional.
2. SET followed by the MINASID options.
3. LIST. On the console, SADMP displays the current virtual storage areas to be dumped.
4. END. SADMP stops prompting the operator for options and begins processing.

When SADMP detects an error in the dumping command, it repeats the incorrect line at the console, underscores the invalid part with ‘*’s, and prompts the operator for replacement text. When the dump command input is longer than 255 characters, SADMP marks the whole line in error.

A system restart during the virtual storage dump program causes SADMP to reprompt the operator for dump options. SADMP does not use any of the dump options that the operator specified before the system restart.

Figure 3-10 shows a sample exchange between SADMP and the operator. The operator's replies are in lowercase.

```
  AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
  AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
     dump sp(0::9) inasid('jes2')
  AMD060I ERROR IN INPUT TEXT INDICATED BY '*':
  DUMP SP(0::9) INASID('JES2')
            *
 AMD065A ENTER TEXT TO BE SUBSTITUTED FOR THE TEXT IN ERROR.
  >
  AMD060I ERROR IN INPUT TEXT INDICATED BY '*':
  DUMP SP(0:9) INASID('JES2')
               ******
  AMD065A ENTER TEXT TO BE SUBSTITUTED FOR THE TEXT IN ERROR.
  > in asid
  AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
  AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
  > list
  AMD067I CURRENT DUMP OPTIONS:
    CSA ALSO LSQA, SP(203:205,213:215,229:230,236:237,247:248) IN ASID(PHYSIN)
    ALSO SP(0:9) IN ASID('JES2')
  AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
  AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
  > end
  AMD010I PROCESSING ASID=0001 ASCB=00FDAF00 JOBNAME=*MASTER*
```

Figure 3-10. Sample Exchange Between SADMP and the Operator

Dump Tailoring Options

You request additional storage that you want dumped by specifying address ranges, subpools, or LSQA in a list of address spaces when you reply to the message:

`AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.` (as in Figure 3-10)

```
RANGE(qualifier)  IN ASID(qualifier)
SP(qualifier)
LSQA
{DATASPACES|DSP} OF ASID(qualifier)
{PAGETABLES OF DATASPACES}
```

The above example specifies one or more ranges of storage addresses, subpools, or LSQA that you want dumped from particular address spaces that you specify as ASIDs, jobnames, or TCB system keys. The example also requests dumping of user data spaces owned by specific address spaces and RSM data spaces. You can extend this syntax by combining parts into lists. See "SADMP DUMP Command Syntax" on page 3-23.

RANGE Option

When you specify RANGE, SADMP dumps all pages within the address range that you specify. You can specify RANGE as any of the following:

RANGE(xxx:yyy,xxx:yyy...)
specifies one or more ranges of storage that you want dumped. xxx and yyy are hexadecimal addresses from 0 to X'7FFFFFFF'

RANGE(ALL)
specifies dumping of all storage from 0 to X'7FFFFFFF'

SP Option

When you specify SP, SADMP dumps only the storage allocated to the subpools that you specify. You can specify SP as any of the following:

SP(ddd)
causes SADMP to dump subpool ddd. ddd is a decimal integer from 0 to 255.

SP(ddd:eee)
causes SADMP to dump all subpools from ddd to eee, inclusive.

SP(ddd:eee,ddd:eee,...)
causes SADMP to dump the combination of subpools that you specify.

SP(ALL)
causes SADMP to dump all subpools, from 0 to 255 inclusive.

LSQA
causes SADMP to dump the LSQA.

ASID Option

You can specify ASID as any of the following:

ASID(xxx:yyy)
causes SADMP to dump storage for the range of address spaces whose address space identifiers begin at xxx and end at yyy, inclusive. xxx and yyy are hexadecimal numbers from X'1' to X'FFFF'.

ASID('jjj')
causes SADMP to dump storage for the address space that jobname jjj identifies. Note that you must enclose the jobname in single quotes.

ASID(SYSKEY)
causes SADMP to dump storage for all address spaces whose active TCB has an associated storage key of 0 to 7.

ASID(combination)
You may combine any of the above specifications. An example of a valid combination is ASID(2,'IMSJOB',SYSKEY).

ASID(PHYSIN)
causes SADMP to dump storage for physically swapped-in address spaces.

ASID(ALL)
causes SADMP to dump storage for all address spaces. Note that you cannot specify ASID(ALL) in combination with any of the other ASID specifications.

DATASPACES Option

DATASPACES OF ASID(qualifier)

When you specify the `DATASPACES OF ASID(qualifier)` keyword, SADMP dumps all data spaces owned by the specified address space. For each requested data space, SADMP:

- Dumps pages backed by central storage during the central storage dump.
- Copies into central storage and dumps every page that is not a first reference page and not backed by central storage

PAGETABLES OF DATASPACES

When you specify the `PAGETABLES OF DATASPACES` keyword, SADMP dumps paged-out virtual storage that contains the page tables for all data spaces.

When SADMP dumps the storage that you specify, SADMP dumps all listed subpools and address ranges in all specified address spaces for each specification of dump options. However, SADMP does not merge your specifications across the dump options that you specify. For example,

```
DUMP SP(0,1) IN ASID(A,B)
```

causes SADMP to dump subpools 0 and 1 in address space A, and subpools 0 and 1 in address space B.

```
DUMP SP(0) IN ASID(A) ALSO SP(1) IN ASID(B)
```

causes SADMP to dump subpool 0 in address space A and subpool 1 in address space B.

Specifying The Minimal SADMP

You can request SADMP to dump certain system related storage ranges in all address spaces that are physically swapped-in at the time the dump is taken.

Requesting The Minimal Dump Option

Use the `MINASID` keyword on the AMDSADMP macro to specify the minimal dump option. If `MINASID` is not specified on the AMDSADMP macro invocation, the default of `MINASID=ALL` is assumed. You can also use the new command `SET MINASID` in the SADMP dialogue if you requested prompting on the AMDSADMP macro invocation.

If you specify `MINASID=ALL` on the AMDSADMP macro invocation, or `MINASID` is omitted, or if you enter `SET MINASID(ALL)` in the SADMP dialogue, the minimal dump will include:

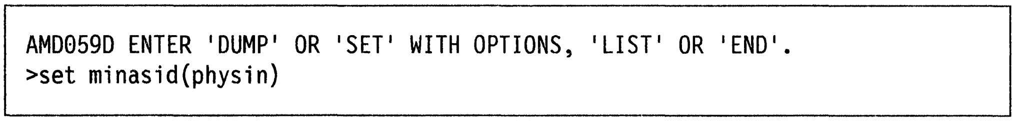
```
DUMP CSA ALSO LSQA, SP(203:205,213:215,229:230,236:237,247:248) IN ASID(ALL)
```

If you specify `MINASID=PHYSIN` on the AMDSADMP macro invocation, or if you enter `SET MINASID(PHYSIN)` in the SADMP dialogue, the minimal dump will include:

```
DUMP CSA ALSO LSQA, SP(203:205,213:215,229:230,236:237,247:248) IN ASID(PHYSIN)
```

```
AMDSADMP TYPE=HI,IPL=SYSDA,VOLSER=VSSA02,MINASID=PHYSIN
```

Figure 3-11. Requesting The Minimal Dump Option During SADMP Generation



```
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
>set minasid(physin)
```

Figure 3-12. Requesting The Minimal Dump Option During SADMP Execution

SADMP DUMP Command Syntax

dump-command	::= DUMP [=] dump-spec-list
	::= SET MINASID(ALL | PHYSIN)
	::= LIST
	::= END
dump-spec-list	::= range-spec-list IN domain-spec-list [ALSO...]
	::={DATASPACES | DSP} OF domain-spec-list [, ...]
	::= PAGETABLES OF DATASPACES
	::= (dump-spec-list)
range-spec-list	::= {SP(subpool-list)|RANGE(address-range-list)|LSQA}[,...]
	::= (range-spec-list)
subpool-list	::= subpool-number TO subpool-number [,...]
	::= ALL
address-range-list	::= address TO address [,...]
	::= ALL
domain-spec-list	::= ASID(address-space-list)[,...]
	::= (domain-spec-list)
address-space-list	::= {asid TO asid|jobname|SYSKEY|PHYSIN}[,...]
	::= ALL

where

- **address** is a hexadecimal number from 0 to X'7FFFFFFF'
- **subpool-number** is a decimal number from 0 to 255
- **asid** is a hexadecimal number from 0 to X'FFFF'
- **jobname** is a valid jobname enclosed in single quotes
- **range-spec-list** is a list of subpools, a list of storage ranges, or both
- **domain-spec-list** is a list of address spaces
- 'TO' and ':' are synonyms
- 'DATASPACES' and 'DSP' are synonyms

You may truncate keywords, such as DATASPACES, on the right, provided the truncated form is unambiguous. You may enter letters either in lower or upper case. You may insert blanks between numbers, keywords, and separators; you may not insert blanks within numbers or keywords.

The following are examples of valid specifications:

```
DUMP SP(0:7,15),RANGE(0:10000000) IN ASID(SYSKEY),ASID(8)
DU  (SP(0 TO 7 OR 15),SP(255)) IN AS('TCAM')
DUMP RANGE(ALL) IN ASID(1) ALSO SP(0) IN ASID(SYSKEY,8)
DU DAT OF AS(ALL)
DUMP ( SP(0:127) IN ASID('GENER') ALSO SP(0) IN ASID('IMS') )
DUMP LSQA IN AS('MYJOB',14)
DU SP(128),LS IN ASID(C,PHYSIN)
```

Executing SADMP

This section describes four procedures for executing SADMP:

- Initialization and execution
- Restart
- Reinitialization
- Dumping

Procedure A: IPLing and Executing SADMP: Use procedure A to IPL the SADMP program and dump storage. If you want to rerun SADMP, for instance when SADMP fails, use procedure B, procedure C, or procedure D.

Procedure B: Restarting SADMP: When you want to rerun SADMP, use procedure B (restart) before you try procedure C (re-IPL that dumps MVS) or procedure D (new IPL that dumps SADMP). Procedures C and D can result in the loss of some central storage from the output, whereas procedure B usually does not.

However, a system restart cannot always work, either because it occurs at a point when SADMP internal resources are unserialized, or because SADMP has been too heavily damaged to function. If the restart does not work, try procedure C (re-IPL).

If SADMP terminates abnormally while dumping central storage (before message AMD005I appears), try to restart SADMP by performing procedure B. If the restart succeeds, SADMP reruns the entire dump. It will first enter wait state X'140000' to allow you to specify a new console and output tape. You can do this to recover from a terminating I/O error on the output tape.

Messages AMD005I and AMD010I indicate that SADMP is beginning to dump virtual storage. If you cause a system restart at this time, SADMP redumps only the paged-out virtual storage. Do *not* issue a system restart if the previous output tape has been rewound and has not been replaced with a new one.

The maximum number of times that you can restart the virtual storage dump program is two.

Procedure C: ReIPLing SADMP: When you re-IPL SADMP, the previous execution of SADMP has already overlaid some parts of central storage and modified the page frame table.

If the virtual storage dump program was in control, a re-IPL may not dump paged-out virtual storage. The number of times that you can IPL SADMP to dump paged-out virtual storage is equal to the number of processors present.

Procedure D: Dumping SADMP: Use a new IPL of SADMP to debug SADMP if SADMP fails. The self-dumps and system restart are two features of SADMP error recovery. When errors occur during virtual storage dump execution, SADMP can take a maximum of two self-dumps. You can use these to diagnose SADMP processing errors. The SADMP system restart capability also assists you in testing and debugging SADMP.

Self-Dumps: During virtual storage dump execution, when SADMP error recovery detects errors in SADMP, SADMP may take a self-dump before proceeding. At most, SADMP takes two self-dumps; on the third request for a self-dump, module AMDSAAUD terminates SADMP processing. SADMP places both the self-dump and the operating system dump onto the output tape.

You can use the LIST subcommand of IPCS to print SADMP self-dumps. The format of the subcommand is:

`LIST` *address* `COMPDATA(AMDSA00x)` where x = 1 or 2

For complete information on using the LIST subcommand of IPCS, see *IPCS Command Reference*.

System Restart: SADMP has a built-in system restart capability that assists you in testing and debugging SADMP. By causing a system restart, you can reinitialize and restart a failing SADMP program. For a virtual storage dump program, you can restart SADMP at most twice.

Procedure A: IPLing and Executing SADMP

1. Use the global STOP function to STOP all processors. Do **not** clear storage.
2. Select a processor that was on-line when you stopped the system.
3. If the processor provides a function to IPL a stand-alone dump without performing a manual STORE STATUS, use this function to IPL SADMP. If you do not use such a function, perform a STORE STATUS before IPLing stand-alone dump.

   Note

   Do **not** use the LOAD CLEAR option. If the LOAD CLEAR option is used, main storage is erased and the dump data set will contain invalid information.

4. Mount the volume that contains the stand-alone dump program on a device attached to the selected processor; ready the device.

   Note: If this is a tape volume, make sure that the file-protect ring is in place. If it is a disk volume, make sure it is write-enabled.

5. IPL SADMP.

   SADMP does not communicate with the operator console. Instead, SADMP loads an enabled wait PSW with wait reason code X'140000'.

   Note: SADMP is waiting for a console I/O interrupt or an external interrupt.

6. Identify the console and output device. Select one operator console whose device address is in the console list that you specified at SADMP generation time. Cause that console to generate an interruption. Depending on the type of console, pressing one or more of the following keys will generate the required interrupt: ATTENTION, CLEAR, ENTER, or SYSTEM REQUEST. (On some consoles, you might have to press RESET first.) This interruption informs SADMP of the console's address, and SADMP responds with message AMD001A.

   a. Ready an output device. For tapes, make sure that the tape is initialized with a tape mark and the file protect ring is inserted. Reply with the three-digit address of the device. If the device is the default that you specified at SADMP generation time, then press ENTER instead of providing the three-digit device address.

      Note: If you reply with the address of an attached device that is not the required device type, or if the device causes certain types of I/O errors, SADMP might load a disabled wait PSW with wait reason code X'150900'. When this occurs, use procedure B to restart SADMP.

   b. SADMP prompts you, with message AMD045A, to specify whether or not you want to write over the existing label on a labeled tape.

   c. SADMP prompts you, with message AMD011A, for a dump title.

   d. For formatted dumps, SADMP issues message AMD008A. Respond with R for real or V for virtual, followed by the address range that you want dumped. Specify addresses that are eight hexadecimal digits in length. When the output device is a tape, SADMP dumps storage for the address range that you specify. When the output device is a printer, SADMP reprompts the operator with message AMD008A.

7. When no console is available, you can run SADMP without a console by entering a null response to AMD001A:

   a. Determine the device address that you specified as the default output address (in the OUTPUT parameter on the AMDSADMP macro) during SADMP residence volume initialization. To execute SADMP, you must define this address to the processor. Ready this device. For tapes, ensure that the file protect ring is in.

   b. Enter an external interruption on the processor that SADMP was IPLed from. SADMP proceeds using the default output device. No messages appear on any consoles; SADMP uses PSW wait reason codes to communicate to the operator.

8. When TYPE=LO and the output device is a tape, SADMP loads a wait PSW with wait reason code X'410000' after completing the dump. This PSW indicates normal termination of SADMP.

9. When SADMP finishes dumping central storage, it issues message AMD005I. SADMP may terminate at this step.

   a. When message AMD068I appears, followed by wait reason code X'150200', SADMP is unable to dump paged-out virtual storage, so SADMP unloads the tape and stops processing. This is normal termination of SADMP when:

      - The system being dumped was at an early stage of initialization, **or**
      - The system being dumped was not MVS/ESA, **or**
      - No STORE STATUS was performed.

   b. Wait reason codes equal to X'25xxxx' are normal when MVS/ESA was not fully initialized. SADMP does not unload the output tape, but it has written an end-of-file.

10. If you specified PROMPT on the AMDSADMP macro, SADMP prompts you for additional storage that you want dumped by issuing message AMD059D.

11. SADMP dumps instruction trace data, paged-out virtual storage, and the SADMP message log.

12. When SADMP completes processing, SADMP unloads the tape and enters a wait reason code X'410000'.

Procedure B: Restarting SADMP

1. Enter a system restart on the processor that you IPLed SADMP from.
2. If the restart is successful, SADMP backs up to a certain point, and continues as in procedure A. During the central storage dump, a system restart causes SADMP to reenter wait reason code X'140000'. During the virtual storage dump, a system restart causes SADMP to repeat the virtual storage dump.
3. Continue procedure A. If you restarted SADMP during the central storage dump program, continue procedure A at step 5. If you restarted SADMP during the virtual storage dump program, continue procedure A at step 9.

Procedure C: ReIPLing SADMP

1. Repeat procedure A, but do **not** issue a STORE STATUS. When you are IPLing using a stand-alone dump hardware function, the STORE STATUS is omitted from all IPLs of SADMP after the first IPL. If the previous IPL of SADMP did not load a wait state and reason code of X'250000' or higher and the reIPL succeeds, SADMP usually completes processing as in procedure A. Some storage locations might not reflect the original contents of central storage because, during a previous IPL, SADMP overlaid the contents. These locations include the absolute PSA and possibly other PSAs.

Procedure D: Dumping SADMP

When you use SADMP to dump itself, you specify either an unformatted (high-speed) SADMP, or a formatted (formatted) SADMP to a printer.

1. Record all messages and wait state and reason codes from the failed SADMP.
2. Select a high-speed or a formatted dump.

   For a high-speed dump, do the following:

   a. Perform a STORE STATUS.

   b. Do procedure A.

   For a formatted dump, do the following:

   a. (Optional) Instead of examining the printed dump in steps e and f that follow, you can inspect central storage to determine which addresses to dump.

   b. Perform a STORE STATUS.

   c. Do procedure A.

   d. Respond 'R,00000000:00000FFF' to message AMD008A. This dumps real page 0.

   e. Examine the printed output from step d. Respond 'R,aaaaaaaa:bbbbbbbb' to message AMD008A, where aaaaaaaa is the contents of the fullword at location X'208' and bbbbbbbb is the contents of the fullword at location X'20C'. This dumps the failed central storage dump program, if the central storage dump program was in storage at the time of the failure.

   f. Examine the printed output from step d. cccccccc is the contents of the fullword at location X'244'. If cccccccc equals X'00000000', then do not continue trying to dump SADMP. If cccccccc is non-zero, dddddddd equals cccccccc plus X'80000'. Respond 'V,cccccccc:dddddddd' to message AMD008A. This dumps the virtual storage dump program that failed.
3. Save the original SADMP output as well as the dump of SADMP.

Messages and Operator Communications During Execution

After IPL, SADMP enters an enabled wait state and reason code of X'140000' to wait for operator communication. Select one of the consoles whose address you specified on the CONSOLE= keyword, and press ENTER at that console. On some consoles, you may have to press RESET before you press ENTER. If no console is available, enter an external interruption. The dump program then bypasses operator communication and attempts to dump storage to the unit that you specified in the OUTPUT= keyword.

If the dump output is directed to tape, you receive this message:

```
AMD001A ENTER ADDRESS OF OUTPUT TAPE.
```

After readying the tape that you want to use, enter the tape's three-character hexadecimal address. When you press ENTER and do not enter an address, SADMP uses the address that you specified on the OUTPUT= keyword.

SADMP looks at the tape label, if one is present, for indication of protection. If the:

- 'security' code, in the case of an 'IBM Standard Data Set Label 1', or the
- 'accessibility' code, in the case of an 'ANSI Standard Data Set Label 1',

indicates that the tape is 'protected' by either standard, then SADMP does not read the tape.

If the tape is not protected, SADMP issues this message:

```
AMD045D TAPE LABEL=vvvvvv. REPLY 'USE' or 'UNLOAD'.
```

where vvvvvv is the volume serial number. Reply USE to write over the label and use the tape. When you reply UNLOAD, SADMP unloads the tape and issues the following message:

```
AMD051A MOUNT ANOTHER TAPE.
```

Continue until a suitable tape is mounted.

If the dump output is directed to a printer, you receive this message:

```
AMD001A ENTER ADDRESS OF PRINTER.
```

You can use the printer that you specified in the macro instruction, or you can specify a different printer.

After you specify the output device, SADMP issues message AMD011A to prompt you for a dump title.

```
AMD011A TITLE=
```

You can specify a title of up to 100 characters. The dump title appears at the top of each page of output that SADMP formats, or at the top of each page of unformatted output after IPCS formats the output. You should select a title that explains why the dump was taken.

For the formatted (formatted) dump program, SADMP prompts for a real or virtual address range to dump after processing the dump title. SADMP issues the following message:

```
AMD008A ENTER ADDRESS RANGE. "R,NNNNNNNN:MMMMMMMM"
```

You must specify 'R' (real addresses) or 'V' (virtual addresses) followed by an eight-digit hexadecimal starting address, ':', and an eight-digit hexadecimal ending address. For example, 'R,000070000:00007FFFF' and 'V,00C00000:01800000' are valid specifications. Before dumping, SADMP rounds the starting address down to the closest 4K boundary, and rounds the ending address up to the closest 4K boundary.

When the formatted (formatted) dump output is directed to tape, SADMP dumps only one storage range. When the formatted dump is directed to a printer, SADMP reprompts the operator for address ranges.

When SADMP completes the dump of central storage, SADMP issues this message:

```
AMD005I DUMPING OF REAL STORAGE COMPLETED.
```

After completion of the central storage dump program, the high-speed dump program starts to dump console trace data and paged-out virtual storage.

If you specified PROMPT on the AMDSADMP macro, SADMP prompts the operator for additional virtual storage to dump by issuing the following message:

```
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
                                        (as in Figure 3-10 on page 3-20)
```

See "Requesting Additional Storage During SADMP Generation" on page 3-19 for the format and meaning of your reply to this message.

When the console screen is full with messages, SADMP issues this message:

```
AMD029D REPLY W TO WAIT AFTER NEXT FULL SCREEN, ELSE
REPLY N; REPLY=
```

When you reply W, SADMP erases the screen and writes message AMD029D again the next time the screen is full. When you reply N, SADMP erases the screen whenever the screen is full, and does not issue message AMD029D again.

To terminate the virtual storage dump program before the dump would ordinarily end, cause an external interruption on the processor that SADMP is executing on.

The following is a sample exchange between SADMP and the operator during execution of an unformatted SADMP program. The operator's replies are in lowercase.

```
AMD001A ENTER ADDRESS OF OUTPUT TAPE. 571
AMD011A TITLE= sample dump
AMD045D TAPE LABEL= T75638. REPLY 'USE' OR 'UNLOAD'. use
AMD005I DUMPING OF REAL STORAGE COMPLETED.
AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> set minasid(all)
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> dump sp(all) in asid('jes2')
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
   dump dataspaces of asid('dumpsrv')
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> list
AMD067I CURRENT DUMP OPTIONS:
  CSA ALSO LSQA, SP(203:205,213:215,229:230,236:237,247:248) IN ASID(ALL)
  ALSO SP(ALL) IN ASID('JES2')
  ALSO DATASPACES OF ASID('DUMPSRV')
AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
> end
AMD010I PROCESSING ASID=0001 ASCB=00FDAF00 JOBNAME=*MASTER*
AMD010I PROCESSING ASID=0002 ASCB=00F24400 JOBNAME=PCAUTH
AMD010I PROCESSING ASID=0003 ASCB=00F24180 JOBNAME=RASP
AMD010I PROCESSING ASID=0004 ASCB=00F0AE00 JOBNAME=TRACE
AMD010I PROCESSING ASID=0005 ASCB=00F0AC00 JOBNAME=GRS
AMD057I COMPLETED SPECIFIC DUMPING FOR GRS
AMD010I PROCESSING ASID=0006 ASCB=00F27E00 JOBNAME=DUMPSRV
AMD076I PROCESSING DATA SPACE SDUMPCSA, OWNED BY ASID 0006.
AMD010I PROCESSING ASID=0008 ASCB=00F40E00 JOBNAME=CONSOLE
AMD081I ASID 0009 NOT DUMPED, PHYSICALLY SWAPPED OUT (JOBNAME=INIT    ).
AMD010I PROCESSING ASID=000A ASCB=00F0AA00 JOBNAME=SMF
AMD081I ASID 000B NOT DUMPED, PHYSICALLY SWAPPED OUT (JOBNAME=INIT    ).
AMD081I ASID 000C NOT DUMPED, PHYSICALLY SWAPPED OUT (JOBNAME=INIT    ).
AMD029D REPLY W TO WAIT AFTER NEXT FULL SCREEN, ELSE REPLY N; REPLY= w
```

SADMP also communicates to the operator using wait state and reason codes. See *System Codes*.

SADMP Messages on the 3480 Display

When stand-alone dump output is sent to a 3480 magnetic tape subsystem, SADMP uses the subsystem's eight-character message display to inform and prompt the operator. The leftmost position on the message display indicates a requested operator action. The eighth position (rightmost) gives additional information.

The SADMP messages that can appear on the 3480 display are:

Dvolser **(alternating)**
MSADMP#U
informs the operator that a labeled tape has been rejected and a new tape must be mounted.

MSADMP#U **(blinking)**
requests that the operator mount a new tape.

RSADMP#U **(blinking)**
indicates that the SADMP program has finished writing to the tape.

RSADMP# **(alternating)**
MSADMP#U
informs the operator that an end-of-reel condition has occurred and a new tape must be mounted.

SADMP# **(blinking)**
indicates that the tape is in use by SADMP.

SADMP# **(alternating)**
NTRDY
informs the operator that some type of intervention is required.

The symbols used in the messages are:

#
is the actual number of reels SADMP has mounted. It is a decimal digit starting at 1 and increasing by 1 after each end-of-reel condition. When the # value exceeds 9, it is reset to 0.

D
means to demount the tape and retain it for further system use, for example as a scratch tape. SADMP does not write on the tape.

M
means to mount a new tape.

R
means to demount the tape and retain it for future SADMP use.

U
means the new tape should not be file-protected.

volser
is the volume serial number on the existing tape label.

SADMP Output

The format of SADMP output depends on the version of the stand-alone program that generated it.

Unformatted Output

Unformatted SADMP output must be displayed at a terminal or printed using IPCS. For full information, refer to the *IPCS User's Guide.*

Use the SADMPMSG verb name of the IPCS VERBEXIT subcommand to format and print the SADMP message log, or, using IPCS, view the message log on the screen. See the *IPCS User's Guide* for details.

Formatted Output

If you direct formatted dump output to a printer, you can immediately use the output as a diagnostic aid. Figure 3-13 shows an example of SADMP formatted output directed to a printer. The IEBPTPCH and IEBGENER utilities can be used to print SADMP output. Figure 3-14 on page 3-34 shows how to use IEBPTPCH to print SADMP output. For information on using IEBGENER, see the publication *Utilities*.

You can also use IEBPTPCH to display the contents of a SADMP output on tape. You can code your JCL so that IEBPTPCH displays all or selected SADMP output records, and displays each record totally or partially. Figure 3-15 on page 3-34 shows how to use IEBPTPCH to print portions of a SADMP output tape.

Copying SADMP Output

You can copy SADMP output from tape to a DASD data set. Figure 3-16 on page 3-34 shows how to use IEBGENER to copy tape output to DASD. Two advantages from copying SADMP tape output to DASD are:

- When SADMP terminates prematurely and does not give the SADMP output (SYSUT1) an end-of-file, the SYSUT2 data set does contain an end-of file. (SYSUT2 is the data set to which SADMP output is copied.) This occurs even when SYSUT2 is another tape. IEBGENER might end with an I/O error on SYSUT1; this is normal termination if SYSUT1 does not contain an end-of-file.
- When SYSUT2 is a direct access data set and you use it as input for IPCS, this saves IPCS processing time.

```
------->>> SAMPLE LOW SPEED (FORMATTED) SADMP OUTPUT
   CURRENT PSW   000C000080D1F1D0        PREFIX   00D21000          CPU ID   0001
GR 0-7      FFFFFFFC 00000004 7FFEF000 7FFEF000  00010003 00FF5768 00AC5FF8 FD000000   *........".0.".0...........¬8....*
AR 0-7      00000000 00000000 00000000 00010003  00000000 00000000 00000000 00000000   *................................*
GR 8-F      00000000 01F01FB8 01F01E20 01F02E1F  81F009F8 01F01E20 0000000D 00000000   *.....0.8.0...0..A0.8.0..........*
AR 8-F      00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
CR 0-7      5EB1EE40 0051207F 002E3D40 8000000C  0000000C 00C90300 FE000000 0051207F   *;1. ..."... .........I........."*
CR 8-F      00000000 00000000 00000000 00000000  0005223B 0051207F DF880C6B 7FFE40B0   *.......................".H.,". 0*
FR 0-2      00000000 00000000 00000000 00000000                                        *................                *
FR 4-6      00000000 00000000 00000000 00000000                                        *................                *
            STORAGE KEY 0E
00000000 V 040C0000 81116CA8 00000000 00000000  00FD8460 00000000 070C0000 80FE3076   *....A.%Y..........D-............*
00000020 V 071C0000 80015C38 070D6000 81F00E5E  00000000 00000000 070C0000 811F350C   *......*...-.A0.;............A...*
00000040 V 00000000 00000000 00000000 00FD8460  00000000 00000000 040C0000 8110ACD8   *..............D-............A..Q*
00000060 V 040C0000 80FE2E00 000C0000 80D1F1D0  00080000 80D203A8 040C0000 8101CF00   *.............J1......K.Y....A...*
00000080 V 00000000 00001004 00020072 0006002B  7FFE4003 00000000 01F00E5E 00000000   *................". ......0.;....*
000000A0 V 03000000 0101CE08 00000000 00000000  00000000 00000000 00010000 00F69270   *.............................6K.*
000000C0 V 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
 PRINTING OF DUPLICATE LINES WAS SUPPRESSED.
00000200 V D7E2C140 00010041 00F0C008 00CDE008  00F2B000 00CFC000 00AEAA58 00AEAA58   *PSA .....0.......20.............*
00000220 V 00F34200 00F34200 00000000 00000000  00000000 00000000 00000000 00000072   *.3...3..........................*
00000240 V 00000000 00000000 00000000 01129E90  040C0000 00FE4F00 040C0000 80039D5C   *......................|........**
00000260 V AD070950 AD000950 AD040950 AD070950  AD070950 00000000 00000704 00000000   *................................*
00000280 V 00FD8C78 00000000 00FD8C80 00000000  00000000 00000000 00000000 00000000   *................................*
000002A0 V 00000000 00000000 00FD8C88 00000000  00FD8CC0 00000000 00000000 00000000   *...........H....................*
000002C0 V 00000000 00FD8CD0 00000000 00000000  00FD8CC8 00FD8CD8 00FD8CE0 00000000   *..................H...Q.........*
000002E0 V 00000000 00000000 00000000 00000000  00000000 00000041 00000000 00FEFBB0   *...............................0*
00000300 V 00FC5D98 00000000 5EB10200 00040000  00000000 00000000 00000001 00FFE07F   *..)Q....;1....................."*
00000320 V 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
 PRINTING OF DUPLICATE LINES WAS SUPPRESSED.
00000360 V 00000000 00000000 00000000 00000000  00000000 00000000 00000B0E 0B0E0000   *................................*
00000380 V 00000C00 00000C00 00F2E370 00000C00  01AB1108 00000000 00F2EC68 00000C00   *.........2T..............2......*
000003A0 V 00F2F560 00000C00 00F2FE58 00F2F560  00F30750 00000000 01AB1A00 00000000   *.25-.....2...25-.3..............*
000003C0 V 00000000 00000000 00000000 00000000  01AB0810 00000000 01AAFF18 00000000   *................................*
000003E0 V 00000000 00000000 00000000 00000000  8007D000 581003F0 0A0D0000 AD00027B   *.......................0.......#*
00000400 V 040CE000 80FE3F36 00000000 00000000  01AACFD8 01A94C90 040C0000 80FF23DE   *..................Q.Z<..........*
00000420 V 070C0000 80FEDA62 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
00000440 V 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
00000460 V 00000000 00000000 070D6000 81F00E5E  00000000 035E1A00 00000000 00000000   *..........-.A0.;.....;..........*
00000480 V 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*
000004A0 V 00000007 00000000 00000000 00000000  00000000 00FF01A0 00000000 00000000   *................................*
```

Figure 3-13. Sample of a Formatted, or Low-Speed, Dump

```
//PRINTLO       JOB      MSGLEVEL=(1,1)
//LIST          EXEC     PGM=IEBPTPCH
//SYSPRINT      DD       SYSOUT=A
//SYSUT1        DD       UNIT=tape,VOL=SER=DUMPTP,LABEL=(,NL),
//      DISP=SHR,DCB=(BLKSIZE=121,RECFM=FBS)
//SYSUT2        DD       SYSOUT=A
//SYSIN         DD       *
        PRINT PREFORM=A
/*
```

Figure 3-14. Sample JCL Used to Invoke IEBPTPCH to Print Formatted SADMP Output

```
//PRINTAPE JOB  MSGLEVEL=(1,1)
//LIST1    EXEC PGM=IEBPTPCH
//*
//* PRINT THE FIRST 10 RECORDS
//*
//SYSUT1   DD   DSN=SADMP.OUTPUT,VOL=SER=DUMPTP,LABEL=(,NL),
//   DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=29120),UNIT=tape,DISP=SHR
//SYSUT2   DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
  PRINT STOPAFT=10,TOTCONV=XE
/*
//LIST2    EXEC PGM=IEBPTPCH
//*
//*PRINT THE HEADERS FOR ALL BUT THE FIRST 10 RECORDS
//*
//SYSUT1   DD   DSN=SADMP.OUTPUT,VOL=SER=TAPENO,LABEL=(,NL),
//   DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=29120),UNIT=tape,DISP=SHR
//SYSUT2   DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
  PRINT MAXFLDS=9,STRTAFT=10
  RECORD FIELD=(8,1,XE,1)
/*
```

Figure 3-15. Sample JCL Used to Invoke IEBPTPCH to Display Portions of a SADMP Output Tape

```
//SADCOPY  JOB  MSGLEVEL=(1,1)
//COPY     EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT1   DD   DSN=SADUMP.TAPE,UNIT=tape,
//         VOL=SER=SADOUT,LABEL=(,NL),DISP=SHR,
//         DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=29120,DEN=4)
//* DEN=3 IS FOR 1600 BPI;DEN=4 IS FOR 6250 BPI
//SYSUT2   DD   DSN=SADUMP.COPY,UNIT=dasd,
//         VOL=SER=SADCPY,DISP=(NEW,CATLG),
//         DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=29120),
//         SPACE=(4104,(8000,4000),RLSE)
```

Figure 3-16. Sample JCL Used to Invoke IEBGENER to Copy SADMP Output Tape to DASD

SADMP Examples

The following examples show how to code the AMDSADMP macro instruction to create various kinds of stand-alone dump programs.

Example 1: Accepting All Defaults

This example shows the AMDSADMP macro instruction coded with no parameters to generate an unformatted, direct access resident dump program.

```
DUMP1 AMDSADMP
```

This is equivalent to coding the following parameters:

```
TYPE=UNFORMATTED
IPL=DSYSDA
VOLSER=SADUMP
ULABEL=NOPURGE
CONSOLE=(01F,3278)
SYSUT=SYSDA
OUTPUT=T282
LOADPT=X'1000'
MSG=ALL
MINASID=ALL
```

Example 2: Generating an Unformatted, Tape Resident Dump Program

In this example, the IPL= parameter specifies tape as the residence volume, and the VOLSER= parameter identifies that tape. All other parameters are allowed to default.

```
AMDSADMP IPL=T3400-2,VOLSER=SATAPE
```

The implied defaults are:

```
TYPE=UNFORMATTED
CONSOLE=(01F,3278)
SYSUT=SYSDA
OUTPUT=T282
LOADPT=X'1000'
MSG=ALL
MINASID=ALL
```

Example 3: Generating a Formatted Dump Program with Defaults

In this example the TYPE= parameter specifies a formatted dump. All other parameters are allowed to default.

```
AMDSADMP TYPE=LO
```

The implied defaults are:

```
IPL=DSYSDA
OUTPUT=P00E
VOLSER=SADUMP
CONSOLE=(01F,3278)
ULABEL=NOPURGE
SYSUT=SYSDA
ADDR=REAL
LOADPT=X'1000'
```

Example 4: Generating a Formatted Dump Program with Output Directed to Tape

In this example, the TYPE= and OUTPUT= parameters specify a formatted dump directed to tape. All other parameters are allowed to default.

```
DUMP2 AMDSADMP TYPE=LO,OUTPUT=T571
```

The implied defaults are:

```
IPL=DSYSDA
VOLSER=SADUMP
CONSOLE=(01F,3278)
ULABEL=NOPURGE
SYSUT=SYSDA
ADDR=REAL
LOADPT=X'1000'
```

Chapter 4. SPZAP

Introduction

SPZAP is a service aid program that operates as a problem program. Its purpose is to allow you to dynamically update and maintain programs and data sets. With SPZAP, an authorized user can:

- Inspect and modify instructions and data in any load module that is a member of a partitioned data set
- Inspect and modify data in a specific record 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.

SPZAP cannot inspect, modify, or dump data in partitioned data sets extended (PDSEs); PDSEs have a data structure that is different from that of partitioned data sets (PDSs). For more information about PDSEs and their data structure, refer to *Managing Non-VSAM Data Sets*.

Capabilities of SPZAP

The functions of SPZAP provide many capabilities. Three of these are described below:

- Using the inspect and modify functions of SPZAP, you can fix programming errors that require only the replacement of instructions in a load module without recompiling the program.
- Using the modify function of SPZAP, you can set traps in a program by inserting invalid instructions. The invalid instructions will force abnormal termination; the dump of storage provided as a result of the abnormal termination is a valuable diagnostic tool, because it shows the contents of storage at a predictable point during execution.
- Using SPZAP to replace data directly on a direct access device, you can reconstruct VTOCs or data records that may have been destroyed as the result of an I/O error or a programming error.

Monitoring the Use of SPZAP

SPZAP allows its user 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 such damage by SPZAP, an installation controls the use of SPZAP. SPZAP is subject to an installation's security protection scheme, except possibly for the VTOC. For the VTOC, the console operator must respond to message AMA117D before a job can update a VTOC.

One means of protecting against unauthorized use of SPZAP is to store SPZAP in a security protected private library. Only authorized users of that library would be able to execute that private version of SPZAP. Note however, that the private version of SPZAP must be linkedited into the authorized library with an authorization code equal to one (AC=1) if the private version will ever be used to update a VTOC, or to zap a VSAM data set or catalog.

Inspecting and Modifying Data

SPZAP 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 SPZAP modification function is controlled by the REP (replace) control statement. The REP control control statement allows you to replace instructions or data at a specific location in a load module or physical record.

The inspection function is controlled by the VERIFY statement. VERIFY allows you 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 are not performed.

To avoid possible errors in replacing data, you should always precede any REP operation with a VERIFY operation.

Inspecting and Modifying a Load Module

To inspect or modify data in a load module, you need a NAME control statement to supply SPZAP with the member name of the load module. 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), you must also supply SPZAP with the name of the CSECT that is to be inspected or modified. If no CSECT name is given in the NAME statement, SPZAP assumes that the control section to be processed is the first one encountered in searching the load module.

Whenever SPZAP replaces a CSECT in a load module in response to your REP and NAME control statements, it also puts descriptive maintenance data in a CSECT identification record (IDR) associated with the load 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 "SPZAP Control Statements" on page 4-11 for an explanation of the IDRDATA statement.

Accessing a Load Module

Once the CSECT has been found, SPZAP 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, SPZAP 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 you to use the assembly listing locations, SPZAP 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 SPZAP 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 begins. SPZAP then subtracts this value from the offset specified on any VERIFY or REP statement that follows the BASE statement, and uses the difference as the displacement of the data.

For a complete description of the control statements mentioned in this discussion, see the topic "SPZAP Control Statements" on page 4-11.

Figure 4-1 on page 4-4 is a sample assembly listing showing more than one control section. To refer to the second CSECT (IEFCVOL2), you could include in the input to SPZAP a BASE statement with a location of 0398. Then, to refer to the subsequent LOAD instruction (LR2,CTJCTAD), you could use an offset of 039A in the VERIFY or REP statements that follow in the SPZAP 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 1001 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          CSECT                 56600017
000398 0590                        387          BALR  R9.0            56700017
00039A                             388          USING *.R9            56800017
00039A 5820 C010             00010 389          L     R2,LCTJCTAD     56900017

                                    .
                                    .
                                    .
```

Figure 4-1. Sample Assembly Listing Showing Multiple Control Sections

Inspecting and Modifying a Data Record

To inspect or modify a specific data record, you must use a CCHHR control statement to specify its direct access address. This CCHHR address must be within the limits of the direct access data set defined in the SYSLIB DD control statement.

If you request a REP operation for a record identified by a CCHHR control statement, SPZAP issues message AMA112I to provide a record of your request.

Accessing a Data Record

When you use the CCHHR control statement, SPZAP reads directly the physical record you want 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 hexadecimal offsets must define the displacement of data relative to the beginning of the record and include the length of any key field.

Dumping Data

SPZAP's dumping options provide a visual representation of the load module or data record that has been changed, thus allowing you to double check the modifications you have made.

You use the DUMP and ABSDUMP control statements to specify the SPZAP dumping options. The operation codes in the DUMP and ABSDUMP statements indicate the kind of dump you want: a formatted hexadecimal dump or a translated dump. The parameters identify the portion of the data to be dumped. See "SPZAP Control Statements" on page 4-11 for additional information on the DUMP and ABSDUMP control statements.

Updating System Status Information

The system status index (SSI) is a 4-byte field created by the linkage editor in the directory entry of a load module. It is useful for keeping track of any modifications that are performed on a load module. SPZAP updates the system status index automatically whenever it replaces data in the associated module.

SPZAP also supplies the SETSSI control statement, that you can use to overlay the existing data in the SSI with your own data. For a complete description of the SETSSI control statement, see "SPZAP Control Statements" on page 4-11.

Not all load modules have system status information. In those that do, the SSI system status index is located in the last four bytes of the user data field in the directory entry. Figure 4-2 shows the position of the SSI in load module directory entries.

Figure 4-2. SSI Bytes in a Load Module Directory Entry

Member Name	TTR	C	User Data Field	SSI
1 8	9 11	12	13 to 70 maximum	variable

Figure 4-3 on page 4-6 shows the composition of the SSI field and the flag bits used to indicate the types of changes made to the corresponding load module program.

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 increased by one for each release that includes a new version of that program. If you make a change to the SSI for any of the IBM-released programs, take care not to destroy this maintenance level indicator unless you purposely mean to do so. To keep the change level byte at its original value, find out what information is contained in the SSI before using the SETSSI function. The LISTLOAD control statement of the LIST service aid can give you the information you need.

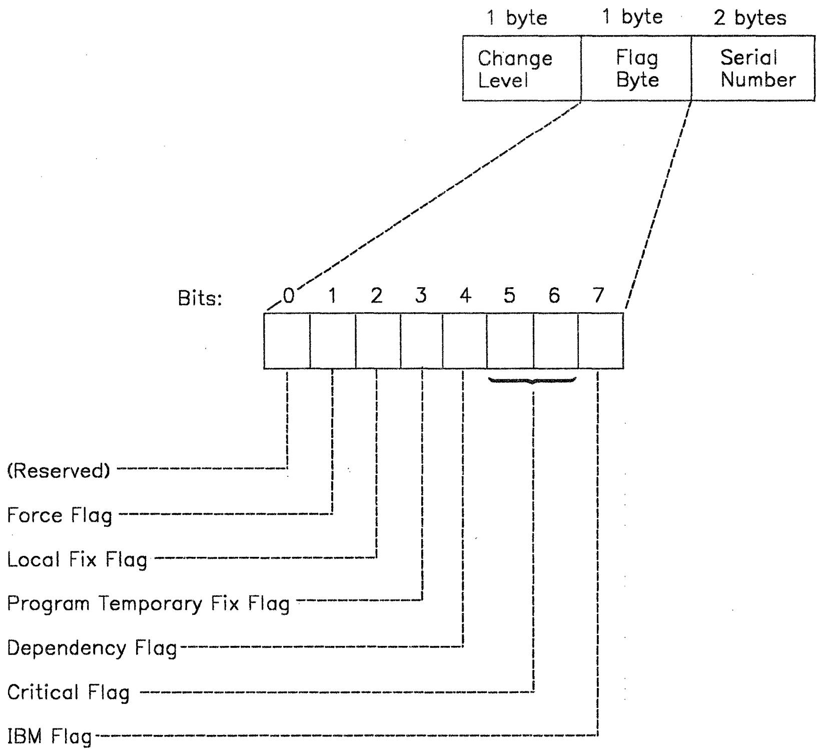


Figure 4-3. Flag Bytes in the System Status Index Field

The second byte of the SSI is termed the *flag byte*. Bits within the flag byte contain information reflecting the member's maintenance status. You need only be concerned with two of the eight bits when you are using SPZAP:

- Bit 2, the local fix flag, indicates that the user has modified a particular member. (It is not used to reflect modifications made by IBM-supplied program temporary fix or a PTF.) SPZAP sets this local fix flag bit to one after successfully modifying a load module.
- Bit 3, the program temporary fix flag, is set to one when an IBM-authorized PTF is applied to a system library to correct an error in an IBM module.

All other bits in the flag byte should be retained in the SSI as they appeared before the SETSSI operation took place, 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. SPZAP will not change these bytes unless you request a change by using the SETSSI control statement.

Operational Considerations

Consider the following points when you run SPZAP:

- SPZAP cannot inspect, modify, or dump data in partitioned data sets extended (PDSEs); PDSEs have a data structure that is different from that of partitioned data sets (PDSs). For more information about PDSEs and their data structure, refer to *Managing Non-VSAM Data Sets*.

- SPZAP uses the system OPEN macro. Therefore, SPZAP cannot modify or inspect security protected data sets when SPZAP cannot successfully complete the authorization checks that occur during the OPEN.
- Unexpired data sets such as system libraries cannot be modified unless the operator replies r xx,'U' to the expiration message that occurs during OPEN.
- If SPZAP is used to modify an operating system module that is made resident in virtual storage only at IPL time, an additional IPL is required to invoke the new version of the altered module. (Note that this includes all modules in SYS1.LPALIB.)
- The SYSLIB DD statement cannot define a concatenated or a multi-volume data set.
- SPZAP supports only direct access devices for the SYSLIB device.
- SPZAP is a non-reusable module.
- When modifying a system data set, such as SYS1.LINKLIB, specify DISP=OLD on the SYSLIB DD statement.

JCL Statements

SPZAP is executed using the following job control statements. The minimum region size needed is 17K 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 AMASPZAP using either PGM=AMASPZAP or PGM=IMASPZAP. The only valid parameter that you may specify is PARM=IGNIDRFULL, which enables SPZAP to override the standard restrictions placed upon CSECT updates (via NAME and REP) when IDR space for the module is found to be full.

Note: Use PARM=IGNIDRFULL with caution. It should be avoided for IBM-maintained modules.

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 SPZAP.

SYSLIB DD Statement
defines the direct access data set that will be accessed by SPZAP when performing the operations specified on the control statements. The DSNAME parameter and DISP=OLD or DISP=SHR are required. The VOLUME and UNIT parameters are necessary only if the data set is not cataloged. When this data set is the VTOC, you must specify DSNAME=FORMAT4.DSCB. This statement cannot define a concatenated or multi-volume data set. It is required for the execution of SPZAP.

Notes:

1. When you access a record in the VTOC (that is, a DSCB) for modification, SPZAP issues message AMA117D to the console. No message is issued, however, when an ABSDUMPT operation is performed on the VTOC.
2. When you access a VSAM object (for example, rebuilding a catalogue), you are required to reference the appropriate catalogue. If you fail to include a STEPCAT or JOBCAT card referring to the appropriate user catalogue, your job might fail. If it does, your job is assigned an abend 913-C; the data set is dumped; and, the system displays message IEC150I.

SYSABEND DD Statement
defines a sequential output data set to be used in case SPZAP terminates abnormally. The data set can be written to a printer, a magnetic tape volume, or a direct access volume. This statement is optional.

SYSIN DD Statement
defines the input stream data set that contains SPZAP control statements.

Return Codes

When SPZAP terminates, it issues one of the following return codes:

Code	Meaning
0	Successful completion.
4	Warning of a condition that may result in future errors if remedial action is not taken.
8	A SPZAP input statement contains an error or was overridden by operator intervention.
12	A requested JCL statement is absent or specifies a data set that was not successfully opened. SPZAP terminates immediately.
16	A permanent I/O error has occurred, perhaps caused by a JCL error, such as invalid blocksize. SPZAP terminates immediately.
20	A record is larger than the blocksize. SPZAP terminates immediately.

Dynamic Invocation of SPZAP

SPZAP can be invoked by an application program at execution time through the use of the CALL, LINK, XCTL, or ATTACH macro instruction. The program must supply a list of alternate ddnames of data sets to be used by SPZAP if the standard ddnames are not used.

The general form of these macros when used to invoke SPZAP is shown below.

```
(anyname)   CALL AMASPZAP,(oplistad,ddnamadr),VL
(anyname)   XCTL EP=AMASPZAP
(anyname)   LINK EP=AMASPZAP,PARAM=(oplistad,ddnamadr),VL=1
(anyname)   ATTACH EP=AMASPZAP,PARAM=(oplistad,ddnamadr),VL=1
```

anyname
indicates an optional statement label on the macro statement.

EP is the entry point - in each case for the SPZAP program.

PARAM
specifies, as a sublist, address parameters to be passed from the program to SPZAP.

oplistad
specifies the address, if present, of either a halfword of zeros (indicating no options) or a non-zero halfword followed by a character string whose length is given in halfwords. For possible parameter values, see "JCL Statements" in this chapter.

ddnamadr
specifies the address of a variable-length list containing alternate ddnames for data sets to be used during SPZAP processing. If all the standard ddnames (SYSPRINT, SYSLIB, and SYSIN) are used, then this parameter can be omitted.

The ddname list must begin on a halfword boundary. The first two bytes contain a count of the number of bytes in the rest of the list. The format of the list is fixed, with each entry having eight bytes. Any name of less than eight bytes must be left justified and padded with blanks. If a name is left out in the list, the entry must contain binary zeros; the standard name is then assumed. Names can be omitted from the end of the ddname list by shortening the list.

The sequence of 8-byte entries in the list is as follows:

Entry	**Standard name**
0-7	not applicable
8-15	not applicable
16-23	not applicable
24-31	SYSLIB
32-39	SYSIN
4-47	SYSPRINT

VL
VL=1

indicates that the sign bit is to be set to 1 in the last word of the address parameter list.

Figure 4-4 is an example of two functionally equivalent dynamic invocations of SPZAP.

```
EXSPZAP  CSECT
         USING *,15            ASSUME REG15 IS BASE
         MODID                 MODULE ID AND DATE IN PROLOG
         SAVE  (14,12)         SAVE REGISTERS
         BALR  12,0            ESTABLISH BASE REGISTER
         USING *,12
         ST    13,SAVEAREA+4   CHAIN NEW SAVEAREA TO PREVIOUS
         LR    2,13            TEMPORARILY SAVE ADDRESS OF OLD SAVEAREA
         LA    13,SAVEAREA     INIT REG13 WITH ADDRESS OF NEW SAVEAREA
         ST    13,8(0,2)       CHAIN PREVIOUS SAVEAREA TO NEW
*************************************************************************
*                                                                       *
*   THIS EXAMPLE SHOWS TWO FUNCTIONALLY EQUIVALENT DYNAMIC              *
*   INVOCATIONS OF SUPERZAP.                                            *
*           NO OPTIONS ARE PASSED.                                      *
*           THE DDNAME FOR THE SYSLIB FILE IS CHANGED TO TESTLIBR.      *
*           THE DDNAME FOR THE SYSIN FILE IS NOT CHANGED.               *
*           THE DDNAME FOR THE SYSPRINT FILE IS CHANGED TO PRINTOUT.    *
*                                                                       *
*************************************************************************
LINKZAP1 LINK  EP=AMASPZAP,PARAM=(OPTLIST,DDLIST),VL=1
LINKZAP2 LINK  EP=AMASPZAP,PARAM=(0,DDLIST),VL=1
         L     13,SAVEAREA+4   LOAD ADDRESS OF PREVIOUS SAVEAREA
         RETURN (14,12),T,RC=0 RETURN TO CALLER
OPTLIST  DC    H'0'            NO OPTIONS ARE PASSED TO AMASPZAP
DDLIST   DS    0H              ALIGN DDNAMES TO HALFWORD BOUNDARY
         DC    H'48'           LENGTH OF THE CHARACTER STRING
*                               CONTAINING DDNAME OVERRIDES
         DC    24XL1'00'       FIRST 24 CHARACTERS ARE IGNORED
         DC    CL8'TESTLIBR'   CHANGE SYSLIB FILE TO DDNAME OF TESTLIBR
         DC    8XL1'00'        USE SYSIN FILE FOR INPUT OF CONTROL
*                               STATEMENTS
         DC    CL8'PRINTOUT'   CHANGE SYSPRINT FILE TO DDNAME OF
*                               PRINTOUT
SAVEAREA DC    18F'0'          REGISTER SAVEAREA
         END
```

Figure 4-4. Sample Assembler Code for Dynamic Invocation of SPZAP

SPZAP Control Statements

SPZAP control statements (entered either through the user's input stream or through the system console) define the processing functions to be performed during a particular execution of SPZAP.

Coding Rules for SPZAP Control Statements

- They can 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 embedded blanks within data fields are not permitted.
- Data and offset values must be specified as a multiple of two hexadecimal digits.
- The size of an SPZAP control statement is 80 bytes.
- Following the last required parameter and its blank delimiter, the rest of the space on most control statements 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.

Following are detailed descriptions of the SPZAP control statements, in the order in which you usually code them.

NAME member [csect]

identifies a CSECT in a load module that is to be the object of subsequent VERIFY, REP, or SETSSI operations. The variables are:

member

the member name of the load module that contains the control section in which the data to be inspected 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. If this variable is omitted, it is assumed that the first CSECT contained in the load module is the one to be used. If there is only one CSECT in the load module, this variable is not necessary.

Note: You can define more than one NAME statement in your input to SPZAP. 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

identifies 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 variable is:

record address

the actual direct access address of the record containing data to be replaced or verified. It must be specified as a 10-digit hexadecimal number in the form ccccchhhhrr, where 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. A zero record number is invalid and defaults to 1.

Note: You can define more than one CCHHR statement in your input to SPZAP. However, the VERIFY, REP and SETSSI statements associated with each CCHHR statement must immediately follow the specific CCHHR statement to which they apply.

{**VERIFY|VER**} **offset expected-content**

causes the data at a specified location within a CSECT or physical record to be compared with the data supplied in the statement.

offset

is the hexadecimal displacement of 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; that is, 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 variable, 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, expected content might look like this:

```
5840C032 (without commas),
    or like this:
5840,C032 (with commas)
```

If all the data does not fit into one VERIFY statement (80-byte logical record), then another VERIFY statement must be defined.

Note: If the two fields being compared are not in agreement, that is, if the VERIFY operation is rejected, no succeeding REP or SETSSI operations are performed until the next NAME or CCHHR control statement is encountered. SPZAP provides a formatted dump of each CSECT or record for which a VERIFY operation failed.

REP offset data

modifies data at a specified location in a CSECT or physical record that was 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 variable field. (Always use the VERIFY function to make sure you know what you are going to change with the REP function.) SPZAP issues message AMA122I to record the contents of the specified location as they were before the change was made.

offset

provides the hexadecimal displacement of 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 the offset value is outside the limits of 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; that is, offset zero is the first byte of the key, not of the data.

data

defines the bytes of data to be inserted at the location. As with the offset variable, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the variable 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 variable may look like this:

```
4160B820 (without commas)
    or like this:
4160,B820 (with commas).
```

If all the data to be modified does not fit into one REP statement (an 80-byte logical record), you can code another REP statement.

Notes:

- Remember that SPZAP automatically updates the system status index (SSI) when it successfully modifies the associated load module. For more detailed information about SSI, see "Updating System Status Information" in this chapter.
- If you are performing multiple VERIFY and REP operations on a CSECT, make sure that all the VERIFY statements precede all the REP statements. This procedure ensures that all REP operations are ignored if one VERIFY reject occurs.
- When you access a record in the VTOC (that is, the DSCB) for modification, SPZAP issues the message AMA117D to the console. No message is issued, however, when an ABSDUMPT operation is performed on the VTOC.

IDRDATA xxxxxxxx

causes SPZAP to place up to eight bytes of user data into the SPZAP 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 was processed by the linkage editor to include CSECT identification records. The variable is:

xxxxxxxx

eight (or fewer) bytes of user data (with no embedded blanks) that are to be placed in the user data field of the SPZAP IDR of the named load module. If more than eight characters are in the variable field, only the first eight characters will be used.

Note: 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 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 variable is:

xxyynnnn

four 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 indicating the following:

```
xx   - change level
yy   - flag byte
nnnn - modification serial number
```

If SPZAP detects an error in any previous VERIFY or REP operation, the SETSSI function is not performed.

Note: Because all bits in the SSI entry are set (reset) by the SETSSI statement, be very careful when using it to avoid altering the vital maintenance-status information. SPZAP issues message AMA122I to record the SSI as it was before the SETSSI operation was performed. (See "Updating System Status Information" on page 4-5.)

{**DUMP|DUMPT**} **member** {**csect|ALL**}

dumps a specific control section or all control sections in a load module. The output format of this dump is hexadecimal. See the topic "SPZAP Output" on page 4-18 for further information. The DUMPT statement differs from the DUMP statement in that it also produces an EBCDIC and an instruction mnemonic translation of the hexadecimal data. The variables 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 variable is omitted entirely, SPZAP assumes that you mean to dump only the first control section contained in the load module.

Note: DUMP or DUMPT applied to a CSECT consisting only of space allocations (DS statements) will produce no output between the statement printback and the dump-completed message.

{**ABSDUMP|ABSDUMPT**}{**startaddr stopaddr|membername|ALL**}

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 "SPZAP Output" on page 4-18). The variables 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. If a record number of 0 is specified, SPZAP will change it to 1 since the READ routine skips over such records. 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 variable 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, SPZAP dumps until it reaches end of file.
- For indexed sequential and direct access data sets, SPZAP dumps all extents.
- For partitioned data sets, SPZAP dumps all extents, including all linkage editor control records, if any exist.

BASE xxxxxx

used by SPZAP 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 SPZAP input stream. When SPZAP 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 variable is:

xxxxxx

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.

Note: The BASE statement should be included in the SPZAP input stream immediately following the NAME statement that identifies the control section that is to be involved in the SPZAP operations. The specified base value remains in effect until all VERIFY, REP, and SETSSI operations for the CSECT have been processed.

CONSOLE

indicates that SPZAP 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:

```
AMA116A  ENTER AMASPZAP CONTROL STATEMENT OR END
```

The operator may then enter in any valid SPZAP control statement conforming to the specifications described in 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. SPZAP 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)**

used to annotate the SPZAP input stream and output listing. Any number of comment statements can be included in the input stream. When such a statement is encountered, SPZAP writes the entire statement to the data set specified for SYSPRINT.

Note: 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.

CHECKSUM [hhhhhhhh]

used to print or verify a fullword checksum (parity-check). All of the valid hexadecimal operands since the preceding CHECKSUM statement or SPZAP initialization are logically concatenated into a single string divided into fullwords, the sum of which is the checksum. For example, the string 12345678FACE produces the checksum 0D025678. Each CHECKSUM statement resets the accumulated checksum value to zeros.

The CHECKSUM statement is effective in detecting clerical errors that may occur when transcribing an SPZAP type of fix. CHECKSUM does not prevent errors; it only causes a message to be issued. By the time the CHECKSUM statement is processed, all prior replaces have been done.

hhhhhhhh

are 8 hexadecimal characters that are compared with the checksum. If the two values are equal, a message is written indicating that the checksum was correct and has been reset.

If the operand field is blank, a message is written giving the actual value of the checksum, and indicating that the checksum has been reset.

When the CHECKSUM control statement is provided with an incorrect operand, the REP and SETSSI statements processed already are not affected.

If the operand is invalid or is not equal to the checksum, a message is written indicating invalid operand or checksum error. All subsequent REP and SETSSI statements are ignored until the next NAME or CCHHR statement is encountered. The results of previously processed statements are not affected.

SPZAP Output

SPZAP provides two different dump formats for the purpose of checking the data that has been verified 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.

Formatted Hexadecimal Dump

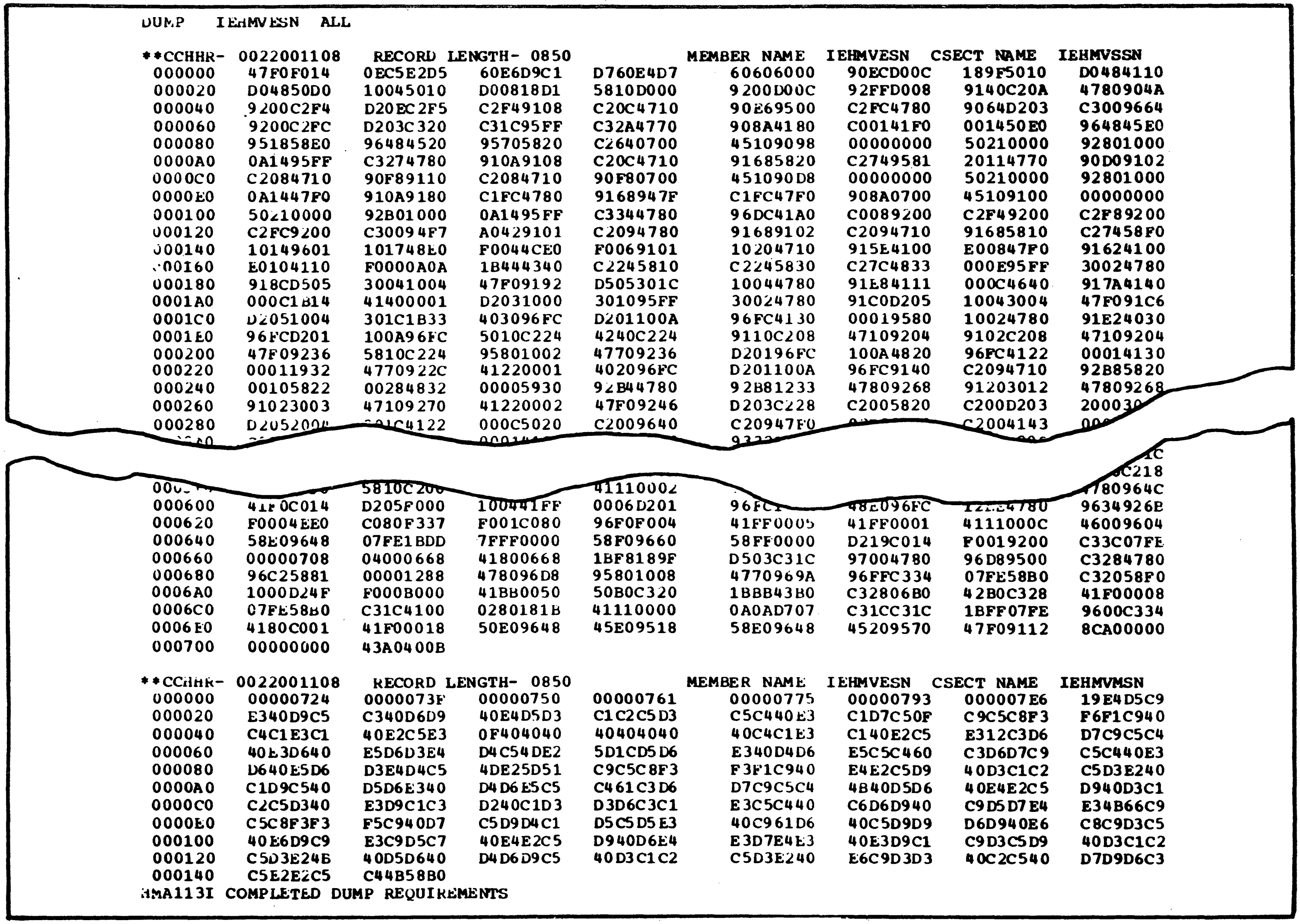
When DUMP or ABSDUMP is the control statement used, the resulting printout is a hexadecimal representation of the requested data. Figure 4-5 on page 4-19 gives a sample of the formatted hexadecimal dump. A heading line is printed at the beginning of each block. This heading consists of the hexadecimal 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 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:

```
AMA113I  COMPLETED DUMP REQUIREMENTS
```

is printed under the last line of the dump printout.

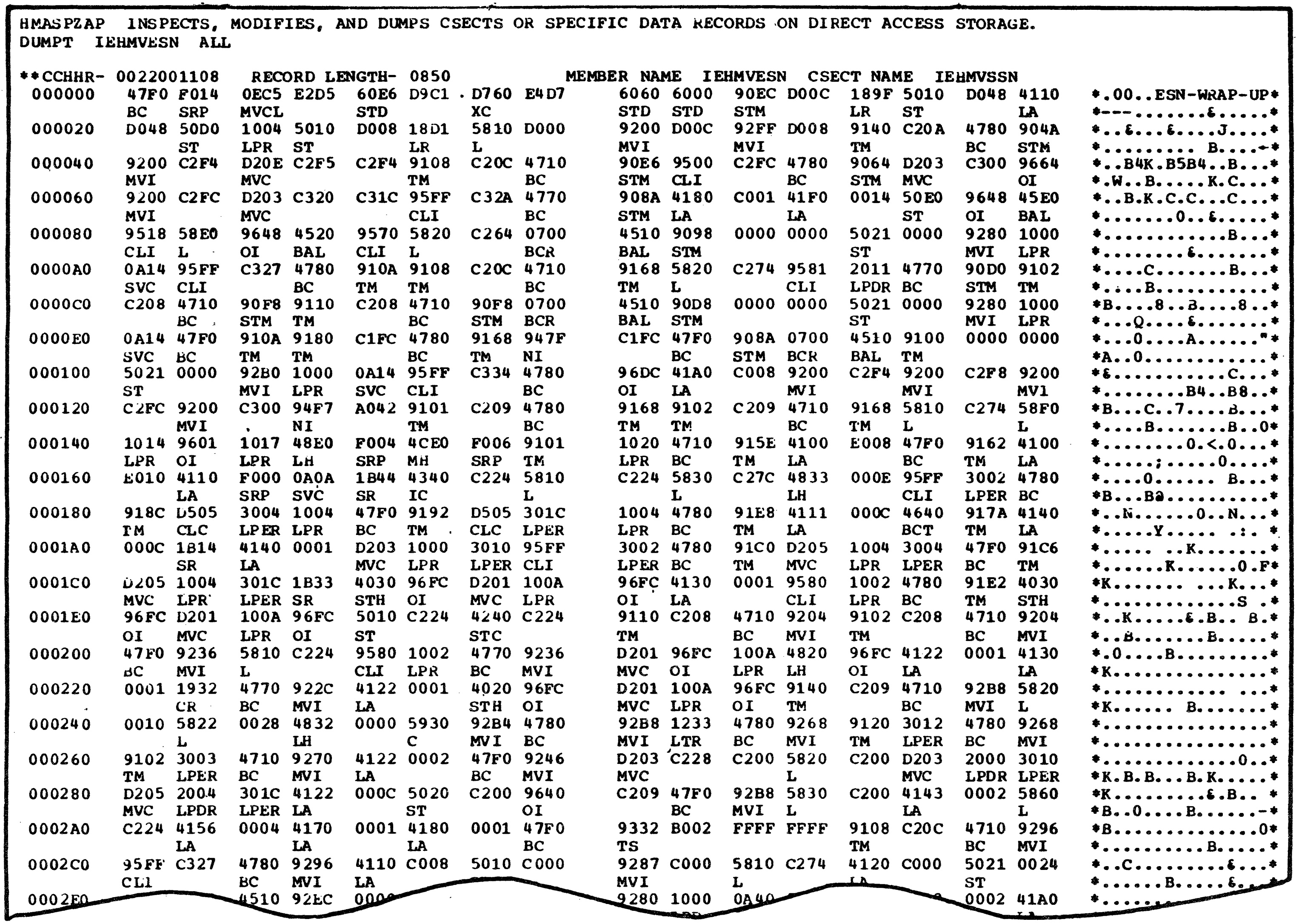
Translated Dump

The control statements DUMPT and ABSDUMPT also provide an operation code translation and an EBCDIC representation of the data contained in the dump. Figure 4-6 on page 4-20 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 not 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.



```
DUMP    IEHMVESN  ALL

**CCHHR-  0022001108   RECORD LENGTH- 0850           MEMBER NAME  IEHMVESN  CSECT NAME  IEHMVSSN
 000000    47F0F014    0EC5E2D5    60E6D9C1    D760E4D7    60606000    90ECD00C    189F5010    D0484110
 000020    D04850D0    10045010    D00818D1    5810D000    9200D00C    92FFD008    9140C20A    4780904A
 000040    9200C2F4    D20EC2F5    C2F49108    C20C4710    90E69500    C2FC4780    9064D203    C3009664
 000060    9200C2FC    D203C320    C31C95FF    C32A4770    908A4180    C00141F0    001450E0    964845E0
 000080    951858E0    96484520    95705820    C2640700    45109098    00000000    50210000    92801000
 0000A0    0A1495FF    C3274780    910A9108    C20C4710    91685820    C2749581    20114770    90D09102
 0000C0    C2084710    90F89110    C2084710    90F80700    451090D8    00000000    50210000    92801000
 0000E0    0A1447F0    910A9180    C1FC4780    9168947F    C1FC47F0    908A0700    45109100    00000000
 000100    50210000    92B01000    0A1495FF    C3344780    96DC41A0    C0089200    C2F49200    C2F89200
 000120    C2FC9200    C30094F7    A0429101    C2094780    91689102    C2094710    91685810    C27458F0
 000140    10149601    101748E0    F0044CE0    F0069101    10204710    915E4100    E00847F0    91624100
 000160    E0104110    F0000A0A    1B444340    C2245810    C2245830    C27C4833    000E95FF    30024780
 000180    918CD505    30041004    47F09192    D505301C    10044780    91E84111    000C4640    917A4140
 0001A0    000C1B14    41400001    D2031000    301095FF    30024780    91C0D205    10043004    47F091C6
 0001C0    D2051004    301C1B33    403096FC    D201100A    96FC4130    00019580    10024780    91E24030
 0001E0    96FCD201    100A96FC    5010C224    4240C224    9110C208    47109204    9102C208    47109204
 000200    47F09236    5810C224    95801002    47709236    D20196FC    100A4820    96FC4122    00014130
 000220    00011932    4770922C    41220001    402096FC    D201100A    96FC9140    C2094710    92B85820
 000240    00105822    00284832    00005930    92B44780    92B81233    47809268    91203012    47809268
 000260    91023003    47109270    41220002    47F09246    D203C228    C2005820    C200D203    20003
 000280    D205200     C4122       000C5020    C2009640    C20947F0                C2004143    00

                                                                                             C218
 000                   5810C20                 41110002                                      780964C
 000600    41F0C014    D205F000    100441FF    0006D201    96FC        48E096FC    4780        9634926B
 000620    F0004EE0    C080F337    F001C080    96F0F004    41FF0005    41FF0001    4111000C    46009604
 000640    58E09648    07FE1BDD    7FFF0000    58F09660    58FF0000    D219C014    F0019200    C33C07FE
 000660    00000708    04000668    41800668    1BF8189F    D503C31C    97004780    96D89500    C3284780
 000680    96C25881    00001288    478096D8    95801008    4770969A    96FFC334    07FE58B0    C32058F0
 0006A0    1000D24F    F000B000    41BB0050    50B0C320    1BBB43B0    C32806B0    42B0C328    41F00008
 0006C0    07FE58B0    C31C4100    0280181B    41110000    0A0AD707    C31CC31C    1BFF07FE    9600C334
 0006E0    4180C001    41F00018    50E09648    45E09518    58E09648    45209570    47F09112    8CA00000
 000700    00000000    43A0400B

**CCHHR-  0022001108   RECORD LENGTH- 0850           MEMBER NAME  IEHMVESN  CSECT NAME  IEHMVMSN
 000000    00000724    0000073F    00000750    00000761    00000775    00000793    000007E6    19E4D5C9
 000020    E340D9C5    C340D6D9    40E4D5D3    C1C2C5D3    C5C440E3    C1D7C50F    C9C5C8F3    F6F1C940
 000040    C4C1E3C1    40E2C5E3    0F404040    40404040    40C4C1E3    C140E2C5    E312C3D6    D7C9C5C4
 000060    40E3D640    E5D6D3E4    D4C54DE2    5D1CD5D6    E340D4D6    E5C5C460    C3D6D7C9    C5C440E3
 000080    D640E5D6    D3E4D4C5    4DE25D51    C9C5C8F3    F3F1C940    E4E2C5D9    40D3C1C2    C5D3E240
 0000A0    C1D9C540    D5D6E340    D4D6E5C5    C461C3D6    D7C9C5C4    4B40D5D6    40E4E2C5    D940D3C1
 0000C0    C2C5D340    E3D9C1C3    D240C1D3    D3D6C3C1    E3C5C440    C6D6D940    C9D5D7E4    E34B66C9
 0000E0    C5C8F3F3    F5C940D7    C5D9D4C1    D5C5D5E3    40C961D6    40C5D9D9    D6D940E6    C8C9D3C5
 000100    40E6D9C9    E3C9D5C7    40E4E2C5    D940D6E4    E3D7E4E3    40E3D9C1    C9D3C5D9    40D3C1C2
 000120    C5D3E24B    40D5D640    D4D6D9C5    40D3C1C2    C5D3E240    E6C9D3D3    40C2C540    D7D9D6C3
 000140    C5E2E2C5    C44B58B0
HMA113I COMPLETED DUMP REQUIREMENTS
```

Figure 4-5. Sample Formatted Hexadecimal Dump



```
HMASPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
DUMPT  IEHMVESN  ALL

**CCHHR- 0022001108   RECORD LENGTH- 0850           MEMBER NAME  IEHMVESN  CSECT NAME  IEHMVSSN
 000000   47F0 F014  0EC5 E2D5  60E6 D9C1  D760 E4D7      6060 6000  90EC D00C  189F 5010  D048 4110   *.00..ESN-WRAP-UP*
          BC   SRP   MVCL       STD        XC             STD  STD   STM        LR   ST         LA     *---.......&.....*
 000020   D048 50D0  1004 5010  D008 18D1  5810 D000      9200 D00C  92FF D008  9140 C20A  4780 904A   *..&...&....J....*
               ST    LPR  ST         LR    L              MVI        MVI        TM         BC   STM    *......... B....-*
 000040   9200 C2F4  D20E C2F5  C2F4 9108  C20C 4710      90E6 9500  C2FC 4780  9064 D203  C300 9664   *..B4K.B5B4..B...*
          MVI        MVC             TM         BC        STM  CLI        BC    STM  MVC        OI     *.W..B.....K.C...*
 000060   9200 C2FC  D203 C320  C31C 95FF  C32A 4770      908A 4180  C001 41F0  0014 50E0  9648 45E0   *..B.K.C.C...C...*
          MVI        MVC             CLI        BC        STM  LA         LA         ST    OI   BAL    *.......0..&.....*
 000080   9518 58E0  9648 4520  9570 5820  C264 0700      4510 9098  0000 0000  5021 0000  9280 1000   *............B...*
          CLI  L     OI   BAL   CLI  L          BCR       BAL  STM              ST         MVI  LPR    *.........&......*
 0000A0   0A14 95FF  C327 4780  910A 9108  C20C 4710      9168 5820  C274 9581  2011 4770  90D0 9102   *....C.......B...*
          SVC  CLI        BC    TM   TM         BC        TM   L          CLI   LPDR BC    STM  TM     *....B...........*
 0000C0   C208 4710  90F8 9110  C208 4710  90F8 0700      4510 90D8  0000 0000  5021 0000  9280 1000   *B....8..B....8..*
               BC    STM  TM         BC    STM  BCR       BAL  STM              ST         MVI  LPR    *...Q....&.......*
 0000E0   0A14 47F0  910A 9180  C1FC 4780  9168 947F      C1FC 47F0  908A 0700  4510 9100  0000 0000   *...0....A......"*
          SVC  BC    TM   TM         BC    TM   NI             BC    STM  BCR   BAL  TM                *A..0............*
 000100   5021 0000  92B0 1000  0A14 95FF  C334 4780      96DC 41A0  C008 9200  C2F4 9200  C2F8 9200   *&...........C...*
          ST         MVI  LPR   SVC  CLI        BC        OI   LA         MVI        MVI        MV1    *........B4..B8..*
 000120   C2FC 9200  C300 94F7  A042 9101  C209 4780      9168 9102  C209 4710  9168 5810  C274 58F0   *B...C..7....B...*
               MVI        NI         TM         BC        TM   TM         BC    TM   L          L      *....B.......B..0*
 000140   1014 9601  1017 48E0  F004 4CE0  F006 9101      1020 4710  915E 4100  E008 47F0  9162 4100   *........0.<.0...*
          LPR  OI    LPR  LH    SRP  MH    SRP  TM        LPR  BC    TM   LA         BC    TM   LA     *.....;.....0....*
 000160   E010 4110  F000 0A0A  1B44 4340  C224 5810      C224 5830  C27C 4833  000E 95FF  3002 4780   *....0...... B...*
               LA    SRP  SVC   SR   IC         L              L          LH         CLI   LPER BC     *B...Ba..........*
 000180   918C D505  3004 1004  47F0 9192  D505 301C      1004 4780  91E8 4111  000C 4640  917A 4140   *..N......0..N...*
          TM   CLC   LPER LPR   BC   TM    CLC  LPER      LPR  BC    TM   LA         BCT   TM   LA     *.....Y..... .:. *
 0001A0   000C 1B14  4140 0001  D203 1000  3010 95FF      3002 4780  91C0 D205  1004 3004  47F0 91C6   *..... ..K.......*
               SR    LA         MVC  LPR   LPER CLI       LPER BC    TM   MVC   LPR  LPER  BC   TM     *......K......0.F*
 0001C0   D205 1004  301C 1B33  4030 96FC  D201 100A      96FC 4130  0001 9580  1002 4780  91E2 4030   *K....... ...K...*
          MVC  LPR   LPER SR    STH  OI    MVC  LPR       OI   LA         CLI   LPR  BC    TM   STH    *.............S .*
 0001E0   96FC D201  100A 96FC  5010 C224  4240 C224      9110 C208  4710 9204  9102 C208  4710 9204   *..K.....&.B.. B.*
          OI   MVC   LPR  OI    ST         STC            TM         BC   MVI   TM         BC   MVI    *..B.......B.....*
 000200   47F0 9236  5810 C224  9580 1002  4770 9236      D201 96FC  100A 4820  96FC 4122  0001 4130   *.0....B.........*
          BC   MVI   L          CLI  LPR   BC   MVI       MVC  OI    LPR  LH    OI   LA         LA     *K...............*
 000220   0001 1932  4770 922C  4122 0001  4020 96FC      D201 100A  96FC 9140  C209 4710  92B8 5820   *............ ...*
               CR    BC   MVI   LA         STH  OI        MVC  LPR   OI   TM         BC    MVI  L      *K...... B.......*
 000240   0010 5822  0028 4832  0000 5930  92B4 4780      92B8 1233  4780 9268  9120 3012  4780 9268   *................*
               L          LH         C     MVI  BC        MVI  LTR   BC   MVI   TM   LPER  BC   MVI    *................*
 000260   9102 3003  4710 9270  4122 0002  47F0 9246      D203 C228  C200 5820  C200 D203  2000 3010   *.............0..*
          TM   LPER  BC   MVI   LA         BC   MVI       MVC                   L          MVC   LPDR LPER   *K.B.B...B.K.....*
 000280   D205 2004  301C 4122  000C 5020  C200 9640      C209 47F0  92B8 5830  C200 4143  0002 5860   *K.........&.B.. *
          MVC  LPDR  LPER LA         ST          OI             BC    MVI  L          LA         L      *B..0....B......-*
 0002A0   C224 4156  0004 4170  0001 4180  0001 47F0      9332 B002  FFFF FFFF  9108 C20C  4710 9296   *B..............0*
               LA         LA         LA         BC        TS                    TM         BC   MVI    *..........B.....*
 0002C0   95FF C327  4780 9296  4110 C008  5010 C000      9287 C000  5810 C274  4120 C000  5021 0024   *..C.........&...*
          CLI        BC   MVI   LA                        MVI        L          LA         ST          *......B.....&...*
 0002E0              4510 92EC  000                       9280 1000  0A40                  0002 41A0   *......
```

Figure 4-6. Sample Translated Dump

SPZAP Examples

Example 1: Inspecting and Modifying a Load Module Containing a Single CSECT

This example shows how to inspect and modify a load module containing a single CSECT.

```
//ZAPCSECT    JOB         MSGLEVEL=(1,1)
//STEP        EXEC        PGM=AMASPZAP
//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:

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 SPZAP that the operations defined by the control statements that follow are to be performed on the module IEEVLNKT.

VERIFY Control Statement
requests that SPZAP check the hexadecimal data at offset X'0018' in 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, SPZAP continues processing the subsequent statements sequentially. If the data is not identical, SPZAP will not perform the REP and SETSSI operations requested for the module. It will, however, perform the requested DUMP operation before discontinuing the processing. It will also dump a hexadecimal image of the module IEEVLNKT to the SYSPRINT data set.

REP Control Statement
causes SPZAP to replace the data at offset X'0018' in module IEEVLNKT with the data given in this control statement, provided the VERIFY statement was successful.

SETSSI Control Statement
instructs SPZAP to replace the system status information in the directory entry for module IEEVLNKT with the SSI data given in the statement, if the VERIFY statement was successful. The new SSI is to contain:

- A change level of 01.
- A flag byte of 21.
- A serial number of 1234.

IDRDATA Control Statement
causes SPZAP 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 SPZAP if the VERIFY operation was successful.

Example 2: Inspecting and Modifying a CSECT in a Load Module Containing Several CSECTs

This example shows how to apply an IBM-supplied PTF in the form of an SPZAP fix, rather than a module replacement PTF.

```
//PTF40228   JOB      MSGLEVEL=(1,1)
//STEP       EXEC     PGM=AMASPZAP
//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
/*
```

SYSLIB DD Statement
defines the library (SYS1.NUCLEUS) that contains input module IEANUC01.

SYSIN DD Statement
defines the input stream that contains the SPZAP control statements.

NAME Control Statement
instructs SPZAP 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 SPZAP to update the IDR in module IEANUC01 for CSECT IEWFETCH with the date LOCFIX01, if either of the REP operations is successful.

VERIFY Control Statements

request that SPZAP 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, SPZAP continues processing subsequent control statements sequentially. However, if the data at the locations does not compare identically to the data given in the VERIFY control statements, SPZAP dumps a hexadecimal image of CSECT IEWFETCH to the SYSPRINT data set; the subsequent REP and SETSSI statements are ignored. The DUMPT function specified will be performed before SPZAP terminates processing.

REP Control Statements

cause SPZAP to replace the data at 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

causes SPZAP to 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:

- A change level of 02.
- A flag byte of 11.
- A serial number of 4228.

DUMPT Control Statement

causes SPZAP to produce a translated dump 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=AMASPZAP
//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       0154 4770,C042
 IDRDATA   PTF01483
 SETSSI    01212448
 DUMPT     IEFX5000  IEFQMRAW
/*
```

SYSLIB DD Statement

defines the data set to be accessed by SPZAP 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 SPZAP.

NAME Control Statement #1

instructs SPZAP 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 SPZAP check the hexadecimal data at offset X'0284' in CSECT IEFQMSSS to make sure it is the same as the data specified in this control statement. If the data is identical, SPZAP continues processing the control statements. If the data is not identical, SPZAP does not perform the REP or SETSSI for CSECT IEFQMSSS, but it does perform the DUMPT operation. It also provides a hexadecimal dump of CSECT IEFQMSSS.

REP Control Statement #1

causes SPZAP to replace the data at offset X'0284' in CSECT IEFQMSSS with the hexadecimal data given in this control statement.

IDRDATA Control Statement #1

causes SPZAP 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 SPZAP to replace the system status information in the directory entry for module IEFX5000 with the SSI data given. The new SSI will contain:

- A change level of 01.
- A flag byte of 21.
- A serial number of 2448.

DUMPT Control Statement #1

causes SPZAP to provide a translated dump of CSECT IEFQMSSS.

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 SPZAP perform the VERIFY function at offset X'0154' from the start of CSECT IEFQMRAW. If the VERIFY operation is successful, SPZAP continues processing the subsequent control statements sequentially. If the VERIFY is rejected, however, SPZAP does not perform the following REP or SETSSI operations, but it does dump a hexadecimal image of CSECT IEFQMRAW to the SYSPRINT data set and performs the DUMPT operation as requested.

REP Control Statement #2

causes SPZAP 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 SPZAP 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 SPZAP to perform the same function as the previous SETSSI, but only if the second VERIFY is not rejected.

DUMPT Control Statement #2
causes SPZAP 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=AMASPZAP
//SYSPRINT      DD          SYSOUT=A
//SYSLIB        DD          DSNAME=FORMAT4.DSCB,DISP=OLD,
//      UNIT=3330,VOLUME=SER=111111,DCB=(KEYLEN=44)
//SYSIN         DD          *
 CCHHR             0005000001
 VERIFY            2C   0504
 REP               2C   0A08
 REP               2E   0001,03000102
 ABSDUMPT          ALL
/*
```

SYSPRINT DD Statement
defines the message data set.

SYSLIB DD Statement
defines the data set to be accessed by SPZAP in performing the operations specified by the control statements. In this example, it defines the VTOC (a Format 4 DSCB) on a 3330 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 SPZAP 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 SPZAP 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, SPZAP continues processing the following control statements sequentially. If the data is not identical, SPZAP does not perform the REP function but does perform the ABSDUMPT operation; it also dumps a formatted hexadecimal image of the data record defined by the CCHHR statement to the SYSPRINT data set.

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 SPZAP 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 is also dumped.

Note: If the VTOC is to be modified, message AMA117D is to 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 SPZAP control statements through the console.

```
//CONSOLIN   JOB      MSGLEVEL=(1,1)
//STEP       EXEC     PGM=AMASPZAP
//SYSPRINT   DD       SYSOUT=A
//SYSLIB     DD       DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN      DD       *
         CONSOLE
/*
```

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 SPZAP control statements are to be entered through the console.

Example 6: Using the BASE Control Statement for Inspecting and Modifying a Load Module

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=AMASPZAP
//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
/*
```

SYSLIB DD Statement

defines the data set to be accessed by SPZAP 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 SPZAP control statements.

NAME Control Statement

instructs SPZAP 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 SPZAP 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 SPZAP to update the IDR in module IEFMCVOL for CSECT IEFCVOL2 with the data MOD04, if the REP operation is successful.

VERIFY Control Statement

requests that SPZAP 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, SPZAP 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, SPZAP continues processing the following statements in the order in which they are encountered. If the data is not identical, SPZAP does not perform the REP, SETSSI, or IDRDATA functions, but it does perform the DUMPs operation; it also dumps a hexadecimal image of CSECT IEFCVOL2 to the SYSPRINT data set.

REP Control Statement

causes SPZAP 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 SPZAP 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 SPZAP (assuming no verification has been rejected).

Chapter 5. Abbreviation Dictionary

Abbreviation	Meaning
AID	record identifier
ASCB	address space control block
ASID	address space identifier
ASXB	address space control block extension
ASVT	address space vector table
ASMVT	auxiliary storage manager vector table
BCB	buffer control block
BSAM	basic sequential access method
CCT	common control table
CCW	channel command word
CDE	contents directory entry
CESD	composite external symbol dictionary
CHPID	channel path identifier
COM	common communication area
CS	control section name
CSCB	command scheduling control block
CSCH	clear subchannel
CSD	common system data area
CSECT	control section
CVT	communication vector table
DA	data area or direct access
DCB	data control block
DEB	data extent block
DLIB	distribution library
DQE	description queue element
DS	data set
DSCB	data set control block
EBCDIC	extended binary-coded-decimal-interchange code
ECB	event control block
EID	event identifier
EOF	end of file
EOV	end of volume
EP	entry point name
EPA	entry point address
ERB	error recovery block
EREP	environmental record error and printing program
ESD	external symbol dictionary
FID	format identifier
FSS	functional subsystem
FXTAB	fix table
GSMQ	global service manage queue
GSPL	global service priority list queue
GTF	generalized trace facility service aid program
GTFBCB	GTF buffer control block
GTFBLOK	GTF blocking area
GTFBUFR	GTF buffer
GTFPCT	GTF primary control table
HSCH	halt subchannel
ICR	independent component release
IDR	CSECT identification record

INITDATA	initialization data
I/O	input/output
IOS	input/output supervisor
IPL	initial program load
IQE	interruption queue element
JCL	job control language
JFCB	job file control block
JOBNAME	jobname
LCCA	logical configuration communication area
LCCAVT	logical configuration communication area vector table
LGVT	logical group vector table
LIST	AMBLIST service aid program
LLE	load list element
LPA	link pack area
LPID	logical page identifier
LPRB	loaded program request block
LR	label reference
LRECL	logical record length
LSMQ	logical service manage queue
LSPL	logical service priority list
LSQA	local system queue area
LT	logical track
LTH	logical track header
MC	monitor call
MCAWSA	monitor call application work/save area
MCCD	monitor call class directory
MCCE	monitor call control element
MCCLE	monitor call class element
MCED	monitor call event directory
MCEE	monitor call event element
MCHEAD	monitor call base table
MCQE	monitor call queue element
MCRWSA	monitor call router work/save area
MN	module name
MSCH	modify subchannel
PCB	print control block
PCI	program controlled interruption
PDS	partitioned data set
PDSE	partitioned data set extended
PER	program event recording
PICA	program interruption control area
PSW	program status word
PTF	program temporary fix
QCB	queue control block
QCR	queue control record
QEL	queue element
RANGETAB	range table
RB	request block
RCSW	real channel status word
RE	record entry
RECFM	record format
RLD	relocatable load dictionary
RNIO	remote network input/output
RQE	reply queue element
SADMP	AMDSADMP service aid program
SCSW	subchannel status word

SD	section definition
SDATA	service data area
SLE	save list element
SLH	subchannel logout handler
SLIP	serviceability level indication processing
SMP	System Modification Program
SPZAP	AMASPZAP service aid program
SQA	system queue area
SR	subroutine
SSCH	start subchannel
SSI	system index status
STA	starting address
SVC	supervisor call
SYSGEN	system generation
SYSIN	system input
SYSOUT	system output
TCAM	telecommunications access method
TCB	task control block
TIOT	task input/output table
TOD	time of day
TQE	timer queue element
TTR	relative trace and record address
UCB	unit control block
VCCT	virtual common communications table
VOLID	volume identification
VPA	virtual page address
VS	virtual storage

Index

MVS/ESA Service Aids

READER'S COMMENT FORM

GC28-1844-2

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

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation? ______________________________

How do you use this publication? ______________________________

Number of latest Newsletter associated with this publication: ____________________

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

MVS/ESA Service Aids
GC28-1844-2

S370-37

c u t a l o n g t h i s l i n e

Reader's Comment Form

Fold and Tape

Please Do Not Staple

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 950
Poughkeepsie, New York 12602-9935

Fold and Tape

Please Do Not Staple

Fold and Tape

Printed in U.S.A.

IBM ®

Program Number
5665-001
5665-002

File Number
S370-37

GC28-1844-2

Printed in U.S.A.